

Milena (Olena)
User documentation 2.0 Id

Generated by Doxygen 1.7.1

Fri Sep 16 2011 16:33:16

Contents

1	Documentation of milena	1
1.1	Introduction	1
1.2	Overview of Milena.	1
1.3	Copyright and License.	2
2	Quick Reference Guide	3
2.1	Installation	4
2.1.1	Requirements	4
2.1.1.1	To compile the user examples	4
2.1.1.2	To compile the documentation (Optional)	4
2.1.1.3	To develop in Olena	4
2.1.2	Getting Olena	4
2.1.3	Building Olena	4
2.2	Foreword	4
2.2.1	Generality	4
2.2.2	Directory hierarchy	4
2.2.3	Writing and compiling a program with Olena	4
2.3	Site	4
2.4	Site set	4
2.4.1	Basic interface	4
2.4.2	Optional interface	4
2.5	Image	5
2.5.1	Definition	5
2.5.2	Possible image types	5
2.5.3	Possible value types	5
2.5.4	Domain	5
2.5.5	Border and extension	6
2.5.5.1	Image border	6

2.5.5.2	Generality on image extension	6
2.5.5.3	Different extensions	6
2.5.6	Interface	8
2.5.7	Load and save images	8
2.5.8	Create an image	8
2.5.9	Access and modify values	9
2.5.10	Image size	9
2.6	Structural elements: Window and neighborhood	10
2.6.1	Define an element	10
2.6.1.1	Window	10
2.6.1.2	Neighborhood	10
2.6.1.3	Custom structural elements	10
2.6.1.4	Conversion between Neighborhoods and Windows	10
2.7	Sites, psites and dpoints	10
2.7.1	Need for site	10
2.7.2	Need for psite	10
2.7.3	From psite to site	11
2.7.4	Dpoint	11
2.8	Iterators	11
2.9	Memory management	12
2.10	Basic routines	12
2.10.1	Fill	12
2.10.2	Paste	12
2.10.3	Blobs	13
2.10.4	Logical not	13
2.10.5	Compute	13
2.10.5.1	Accumulators	13
2.10.5.2	Example with labeling::compute()	13
2.10.6	Working with parts of an image	14
2.10.6.1	Restrict an image with a site set	14
2.10.6.2	Restrict an image with a predicate	15
2.10.6.3	Restrict an image with a C function	15
2.11	Input / Output	16
2.11.1	ImageMagick	16
2.11.2	GDCM	16
2.12	Graphs and images	16

2.12.1	Description	16
2.12.2	Example	16
2.13	Useful global variables	19
2.14	Useful macros	19
2.14.1	Variable declaration macros	19
2.14.2	Iterator type macros	19
2.14.2.1	Default iterator types	19
2.14.2.2	Forward iterator types	19
2.14.2.3	Backward iterators	19
2.14.2.4	Graph iterators	19
2.15	Common Compilation Errors	19
3	Tutorial	21
3.1	Welcome	21
3.1.1	How to learn Milena	21
3.1.2	Obtaining the library	21
3.1.3	Downloading the library	21
3.1.3.1	Downloading from SVN	21
3.1.3.2	Downloading packaged releases	21
3.1.4	Join the mailing lists	22
3.1.5	Directory structure	22
3.1.6	Documentation	22
3.1.7	Community and Support	22
3.1.8	Project status	22
3.1.9	A brief history of Milena	22
3.1.10	Contacts	22
3.2	Installation	22
3.2.1	Bootstrap (SVN Sources)	22
3.2.2	Configure	22
3.2.3	Install	22
3.2.4	Optional compilation	22
3.2.4.1	Examples	22
3.2.4.2	Tools	22
3.2.4.3	Tests	22
3.2.5	Installation content	22
3.3	Getting started with Milena	23
3.3.1	Getting familiar with genericity	23

3.3.2	First generic algorithm	23
3.3.3	Compilation	25
3.3.3.1	Include path	25
3.3.3.2	Library linking	25
3.3.3.3	Disable Debug	25
3.3.3.4	Compiler optimization flags	25
3.3.4	Debug hints	25
3.3.4.1	Using assertions and GDB	25
3.3.4.2	Traces	25
3.3.4.3	Debug routines	25
3.4	Data representation	26
3.4.1	Sites	26
3.4.2	Site sets	26
3.4.2.1	Creating a site set	26
3.4.2.2	Getting access to sites	27
3.4.3	Images	27
3.4.3.1	Creating an image	27
3.4.3.2	Reading an image from a file	27
3.4.3.3	Accessing data	27
3.5	Load and save images	27
3.6	Create your first image	27
3.7	Read and write images	28
3.8	Regions of interest	28
3.8.1	Image domain restricted by a site set	29
3.8.2	Image domain restricted by a function	29
3.8.3	Image domain restricted by a mask	29
3.8.4	Image domain restricted by a predicate	29
4	Module Index	31
4.1	Modules	31
5	Namespace Index	33
5.1	Namespace List	33
6	Class Index	37
6.1	Class Hierarchy	37
7	Class Index	81

7.1	Class List	81
8	Module Documentation	91
8.1	On site sets	91
8.1.1	Detailed Description	91
8.2	On images	91
8.2.1	Detailed Description	92
8.3	On values	92
8.3.1	Detailed Description	94
8.4	Multiple accumulators	94
8.4.1	Detailed Description	94
8.5	Graphes	94
8.5.1	Detailed Description	95
8.6	Images	95
8.6.1	Detailed Description	95
8.7	Basic types	95
8.7.1	Detailed Description	96
8.8	Image morphers	96
8.9	Values morphers	96
8.9.1	Detailed Description	97
8.10	Domain morphers	97
8.10.1	Detailed Description	97
8.11	Identity morphers	98
8.11.1	Detailed Description	98
8.12	Types	98
8.12.1	Detailed Description	99
8.13	Accumulators	99
8.13.1	Detailed Description	99
8.14	Routines	99
8.15	Canvas	99
8.16	Functions	100
8.16.1	Detailed Description	101
8.17	Neighborhoods	101
8.17.1	Detailed Description	101
8.18	1D neighborhoods	101
8.18.1	Detailed Description	102
8.18.2	Typedef Documentation	102

8.18.2.1	neighb1d	102
8.18.3	Function Documentation	102
8.18.3.1	c2	102
8.19	2D neighborhoods	102
8.19.1	Detailed Description	103
8.19.2	Typedef Documentation	103
8.19.2.1	neighb2d	103
8.19.3	Function Documentation	103
8.19.3.1	c2_col	103
8.19.3.2	c2_row	103
8.19.3.3	c4	103
8.19.3.4	c8	104
8.20	3D neighborhoods	104
8.20.1	Detailed Description	105
8.20.2	Typedef Documentation	105
8.20.2.1	neighb3d	105
8.20.3	Function Documentation	105
8.20.3.1	c18	105
8.20.3.2	c26	105
8.20.3.3	c2_3d_sli	106
8.20.3.4	c4_3d	106
8.20.3.5	c6	107
8.20.3.6	c8_3d	107
8.21	Site sets	108
8.21.1	Detailed Description	108
8.22	Basic types	108
8.22.1	Detailed Description	108
8.23	Graph based	109
8.23.1	Detailed Description	109
8.24	Complex based	109
8.24.1	Detailed Description	109
8.25	Sparse types	109
8.25.1	Detailed Description	110
8.26	Queue based	110
8.26.1	Detailed Description	110
8.27	Utilities	110

8.27.1 Detailed Description	111
8.28 Windows	111
8.28.1 Detailed Description	112
8.29 1D windows	112
8.29.1 Detailed Description	112
8.29.2 Typedef Documentation	112
8.29.2.1 segment1d	112
8.29.2.2 window1d	113
8.30 2D windows	113
8.30.1 Detailed Description	114
8.30.2 Typedef Documentation	114
8.30.2.1 disk2d	114
8.30.2.2 hline2d	114
8.30.2.3 vline2d	114
8.30.2.4 window2d	115
8.30.3 Function Documentation	115
8.30.3.1 win_c4p	115
8.30.3.2 win_c8p	115
8.31 3D windows	115
8.31.1 Detailed Description	116
8.31.2 Typedef Documentation	116
8.31.2.1 sline3d	116
8.31.2.2 sphere3d	117
8.31.2.3 window3d	117
8.31.3 Function Documentation	117
8.31.3.1 win_c4p_3d	117
8.31.3.2 win_c8p_3d	117
8.32 N-D windows	118
8.32.1 Detailed Description	118
8.33 Multiple windows	118
8.33.1 Detailed Description	118
8.34 v2w2v functions	118
8.35 v2w_w2v functions	119
8.36 vv2b functions	119
9 Namespace Documentation	121
9.1 mln Namespace Reference	121

9.1.1	Detailed Description	143
9.1.2	Typedef Documentation	145
9.1.2.1	bin_1complex_image2d	145
9.1.2.2	bin_2complex_image3df	145
9.1.2.3	box1d	145
9.1.2.4	box2d	146
9.1.2.5	box2d_h	146
9.1.2.6	box3d	146
9.1.2.7	discrete_plane_1complex_geometry	146
9.1.2.8	discrete_plane_2complex_geometry	146
9.1.2.9	dpoint1d	146
9.1.2.10	dpoint2d	146
9.1.2.11	dpoint2d_h	147
9.1.2.12	dpoint3d	147
9.1.2.13	float_2complex_image3df	147
9.1.2.14	int_u8_1complex_image2d	147
9.1.2.15	int_u8_2complex_image2d	147
9.1.2.16	int_u8_2complex_image3df	147
9.1.2.17	p_run2d	147
9.1.2.18	p_runs2d	148
9.1.2.19	point1d	148
9.1.2.20	point1df	148
9.1.2.21	point2d	148
9.1.2.22	point2d_h	148
9.1.2.23	point2df	148
9.1.2.24	point3d	148
9.1.2.25	point3df	148
9.1.2.26	rgb8_2complex_image3df	149
9.1.2.27	space_2complex_geometry	149
9.1.2.28	unsigned_2complex_image3df	149
9.1.2.29	vec2d_d	149
9.1.2.30	vec2d_f	149
9.1.2.31	vec3d_d	149
9.1.2.32	vec3d_f	149
9.1.2.33	w_window1d_float	149
9.1.2.34	w_window1d_int	150

9.1.2.35	w_window2d_float	150
9.1.2.36	w_window2d_int	150
9.1.2.37	w_window3d_float	150
9.1.2.38	w_window3d_int	150
9.1.3	Function Documentation	150
9.1.3.1	a_point_of	150
9.1.3.2	apply_p2p	150
9.1.3.3	apply_p2p	151
9.1.3.4	compose	151
9.1.3.5	duplicate	151
9.1.3.6	extend	151
9.1.3.7	extend	152
9.1.3.8	extend	152
9.1.3.9	implies	152
9.1.3.10	initialize	152
9.1.3.11	larger_than	152
9.1.3.12	make_debug_graph_image	153
9.1.3.13	mln_exact	153
9.1.3.14	mln_gen_complex_neighborhood	153
9.1.3.15	mln_gen_complex_neighborhood	153
9.1.3.16	mln_gen_complex_neighborhood	153
9.1.3.17	mln_gen_complex_neighborhood	153
9.1.3.18	mln_gen_complex_neighborhood	154
9.1.3.19	mln_gen_complex_neighborhood	154
9.1.3.20	mln_gen_complex_window	154
9.1.3.21	mln_gen_complex_window	154
9.1.3.22	mln_gen_complex_window	154
9.1.3.23	mln_gen_complex_window	154
9.1.3.24	mln_gen_complex_window	154
9.1.3.25	mln_gen_complex_window	154
9.1.3.26	mln_gen_complex_window_p	154
9.1.3.27	mln_gen_complex_window_p	155
9.1.3.28	mln_gen_complex_window_p	155
9.1.3.29	mln_gen_complex_window_p	155
9.1.3.30	mln_gen_complex_window_p	155
9.1.3.31	mln_gen_complex_window_p	155

9.1.3.32	<code>mln_regular</code>	155
9.1.3.33	<code>mln_trait_op_geq</code>	155
9.1.3.34	<code>mln_trait_op_greater</code>	156
9.1.3.35	<code>mln_trait_op_leq</code>	156
9.1.3.36	<code>mln_trait_op_neq</code>	156
9.1.3.37	<code>operator!=</code>	157
9.1.3.38	<code>operator!=</code>	157
9.1.3.39	<code>operator*</code>	157
9.1.3.40	<code>operator++</code>	157
9.1.3.41	<code>operator-</code>	157
9.1.3.42	<code>operator-</code>	158
9.1.3.43	<code>operator--</code>	158
9.1.3.44	<code>operator<</code>	158
9.1.3.45	<code>operator<</code>	158
9.1.3.46	<code>operator<</code>	159
9.1.3.47	<code>operator<<</code>	159
9.1.3.48	<code>operator<<</code>	159
9.1.3.49	<code>operator<<</code>	159
9.1.3.50	<code>operator<<</code>	159
9.1.3.51	<code>operator<=</code>	159
9.1.3.52	<code>operator<=</code>	160
9.1.3.53	<code>operator<=</code>	160
9.1.3.54	<code>operator<=</code>	160
9.1.3.55	<code>operator<=</code>	160
9.1.3.56	<code>operator==</code>	160
9.1.3.57	<code>operator==</code>	161
9.1.3.58	<code>operator==</code>	161
9.1.3.59	<code>operator==</code>	161
9.1.3.60	<code>operator==</code>	161
9.1.3.61	<code>operator==</code>	162
9.1.3.62	<code>operator==</code>	162
9.1.3.63	<code>operator </code>	162
9.1.3.64	<code>operator </code>	162
9.1.3.65	<code>operator </code>	162
9.1.3.66	<code>operator </code>	163
9.1.3.67	<code>operator </code>	163

9.1.3.68	operator	163
9.1.3.69	primary	163
9.1.3.70	ptransform	163
9.1.4	Variable Documentation	164
9.1.4.1	before	164
9.1.4.2	sagittal_dec	164
9.1.4.3	up	164
9.2	mln::accu Namespace Reference	164
9.2.1	Detailed Description	166
9.2.2	Function Documentation	166
9.2.2.1	compute	166
9.2.2.2	line	166
9.2.2.3	mln_meta_accu_result	167
9.2.2.4	take	167
9.3	mln::accu::image Namespace Reference	168
9.3.1	Detailed Description	168
9.4	mln::accu::impl Namespace Reference	168
9.4.1	Detailed Description	168
9.5	mln::accu::logic Namespace Reference	168
9.5.1	Detailed Description	168
9.6	mln::accu::math Namespace Reference	168
9.6.1	Detailed Description	169
9.7	mln::accu::meta::logic Namespace Reference	169
9.7.1	Detailed Description	169
9.8	mln::accu::meta::math Namespace Reference	169
9.8.1	Detailed Description	170
9.9	mln::accu::meta::shape Namespace Reference	170
9.9.1	Detailed Description	170
9.10	mln::accu::meta::stat Namespace Reference	170
9.10.1	Detailed Description	171
9.11	mln::accu::shape Namespace Reference	171
9.11.1	Detailed Description	172
9.12	mln::accu::stat Namespace Reference	172
9.12.1	Detailed Description	173
9.12.2	Function Documentation	173
9.12.2.1	operator==	173

9.13	<code>mln::algebra</code> Namespace Reference	173
9.13.1	Detailed Description	174
9.13.2	Function Documentation	174
9.13.2.1	<code>ldlt_decomp</code>	174
9.13.2.2	<code>ldlt_solve</code>	174
9.13.2.3	<code>operator*</code>	175
9.13.2.4	<code>vprod</code>	175
9.14	<code>mln::arith</code> Namespace Reference	175
9.14.1	Detailed Description	177
9.14.2	Function Documentation	178
9.14.2.1	<code>diff_abs</code>	178
9.14.2.2	<code>div</code>	178
9.14.2.3	<code>div_cst</code>	178
9.14.2.4	<code>div_inplace</code>	179
9.14.2.5	<code>min</code>	179
9.14.2.6	<code>min_inplace</code>	180
9.14.2.7	<code>minus</code>	180
9.14.2.8	<code>minus</code>	180
9.14.2.9	<code>minus</code>	181
9.14.2.10	<code>minus_cst</code>	181
9.14.2.11	<code>minus_cst</code>	182
9.14.2.12	<code>minus_cst_inplace</code>	182
9.14.2.13	<code>minus_inplace</code>	182
9.14.2.14	<code>plus</code>	183
9.14.2.15	<code>plus</code>	183
9.14.2.16	<code>plus</code>	184
9.14.2.17	<code>plus_cst</code>	184
9.14.2.18	<code>plus_cst</code>	184
9.14.2.19	<code>plus_cst</code>	185
9.14.2.20	<code>plus_cst_inplace</code>	185
9.14.2.21	<code>plus_inplace</code>	186
9.14.2.22	<code>revert</code>	186
9.14.2.23	<code>revert_inplace</code>	186
9.14.2.24	<code>times</code>	187
9.14.2.25	<code>times_cst</code>	187
9.14.2.26	<code>times_inplace</code>	188

9.15	mln::arith::impl Namespace Reference	188
9.15.1	Detailed Description	188
9.16	mln::arith::impl::generic Namespace Reference	188
9.16.1	Detailed Description	188
9.17	mln::binarization Namespace Reference	189
9.17.1	Detailed Description	189
9.17.2	Function Documentation	189
9.17.2.1	binarization	189
9.17.2.2	threshold	189
9.18	mln::border Namespace Reference	190
9.18.1	Detailed Description	190
9.18.2	Function Documentation	190
9.18.2.1	adjust	190
9.18.2.2	duplicate	191
9.18.2.3	equalize	191
9.18.2.4	fill	192
9.18.2.5	find	192
9.18.2.6	get	192
9.18.2.7	mirror	193
9.18.2.8	resize	193
9.19	mln::border::impl Namespace Reference	193
9.19.1	Detailed Description	193
9.20	mln::border::impl::generic Namespace Reference	194
9.20.1	Detailed Description	194
9.21	mln::canvas Namespace Reference	194
9.21.1	Detailed Description	195
9.21.2	Function Documentation	195
9.21.2.1	distance_front	195
9.21.2.2	distance_geodesic	195
9.22	mln::canvas::browsing Namespace Reference	195
9.22.1	Detailed Description	196
9.23	mln::canvas::impl Namespace Reference	196
9.23.1	Detailed Description	196
9.24	mln::canvas::labeling Namespace Reference	196
9.24.1	Detailed Description	197
9.24.2	Function Documentation	197

9.24.2.1	blobs	197
9.25	mln::canvas::labeling::impl Namespace Reference	197
9.25.1	Detailed Description	197
9.26	mln::canvas::morpho Namespace Reference	197
9.26.1	Detailed Description	197
9.27	mln::convert Namespace Reference	198
9.27.1	Detailed Description	200
9.27.2	Function Documentation	200
9.27.2.1	from_to	200
9.27.2.2	from_to	200
9.27.2.3	from_to	200
9.27.2.4	from_to	200
9.27.2.5	mln_image_from_grid	201
9.27.2.6	mln_image_from_grid	201
9.27.2.7	mln_image_from_grid	201
9.27.2.8	mln_image_from_grid	201
9.27.2.9	mln_window	201
9.27.2.10	to	201
9.27.2.11	to_dpoint	201
9.27.2.12	to_fun	201
9.27.2.13	to_image	202
9.27.2.14	to_p_array	202
9.27.2.15	to_p_array	202
9.27.2.16	to_p_array	202
9.27.2.17	to_p_set	202
9.27.2.18	to_p_set	202
9.27.2.19	to_p_set	202
9.27.2.20	to_p_set	203
9.27.2.21	to_p_set	203
9.27.2.22	to_qimage	203
9.27.2.23	to_upper_window	203
9.27.2.24	to_upper_window	203
9.27.2.25	to_window	203
9.27.2.26	to_window	204
9.27.2.27	to_window	204
9.27.3	Variable Documentation	204

9.27.3.1	<code>to_fun</code>	204
9.28	<code>mln::data</code> Namespace Reference	204
9.28.1	Detailed Description	206
9.28.2	Function Documentation	206
9.28.2.1	<code>abs</code>	206
9.28.2.2	<code>abs_inplace</code>	207
9.28.2.3	<code>apply</code>	207
9.28.2.4	<code>compute</code>	207
9.28.2.5	<code>compute</code>	208
9.28.2.6	<code>convert</code>	208
9.28.2.7	<code>fast_median</code>	208
9.28.2.8	<code>fill</code>	209
9.28.2.9	<code>fill_with_image</code>	209
9.28.2.10	<code>fill_with_value</code>	209
9.28.2.11	<code>median</code>	210
9.28.2.12	<code>mln_meta_accu_result</code>	210
9.28.2.13	<code>paste</code>	210
9.28.2.14	<code>paste_without_localization</code>	211
9.28.2.15	<code>replace</code>	211
9.28.2.16	<code>saturate</code>	211
9.28.2.17	<code>saturate</code>	212
9.28.2.18	<code>saturate_inplace</code>	212
9.28.2.19	<code>sort_offsets_increasing</code>	212
9.28.2.20	<code>sort_psites_decreasing</code>	212
9.28.2.21	<code>sort_psites_increasing</code>	213
9.28.2.22	<code>stretch</code>	213
9.28.2.23	<code>to_enc</code>	213
9.28.2.24	<code>transform</code>	214
9.28.2.25	<code>transform</code>	214
9.28.2.26	<code>transform_inplace</code>	214
9.28.2.27	<code>transform_inplace</code>	215
9.28.2.28	<code>update</code>	215
9.28.2.29	<code>wrap</code>	215
9.29	<code>mln::data::approx</code> Namespace Reference	216
9.29.1	Detailed Description	216
9.29.2	Function Documentation	216

9.29.2.1	median	216
9.29.2.2	median	216
9.29.2.3	median	217
9.30	mln::data::approx::impl Namespace Reference	217
9.30.1	Detailed Description	217
9.31	mln::data::impl Namespace Reference	217
9.31.1	Detailed Description	218
9.31.2	Function Documentation	218
9.31.2.1	paste_without_localization_fast	218
9.31.2.2	paste_without_localization_fastest	219
9.31.2.3	paste_without_localization_lines	219
9.31.2.4	stretch	220
9.31.2.5	transform_inplace_lowq	220
9.31.2.6	update_fastest	220
9.32	mln::data::impl::generic Namespace Reference	220
9.32.1	Detailed Description	221
9.32.2	Function Documentation	221
9.32.2.1	fill_with_image	221
9.32.2.2	fill_with_value	222
9.32.2.3	paste	222
9.32.2.4	transform	222
9.32.2.5	transform	222
9.32.2.6	transform_inplace	223
9.32.2.7	transform_inplace	223
9.32.2.8	update	223
9.33	mln::data::naive Namespace Reference	223
9.33.1	Detailed Description	224
9.33.2	Function Documentation	224
9.33.2.1	median	224
9.34	mln::data::naive::impl Namespace Reference	224
9.34.1	Detailed Description	224
9.35	mln::debug Namespace Reference	225
9.35.1	Detailed Description	226
9.35.2	Function Documentation	226
9.35.2.1	draw_graph	226
9.35.2.2	draw_graph	227

9.35.2.3	draw_graph	227
9.35.2.4	filename	227
9.35.2.5	format	227
9.35.2.6	format	228
9.35.2.7	format	228
9.35.2.8	format	228
9.35.2.9	iota	228
9.35.2.10	mosaic	228
9.35.2.11	println	229
9.35.2.12	println	229
9.35.2.13	println_with_border	229
9.35.2.14	put_word	229
9.35.2.15	slices_2d	229
9.35.2.16	slices_2d	229
9.35.2.17	superpose	230
9.35.2.18	superpose	230
9.35.2.19	z_order	230
9.36	mln::debug::impl Namespace Reference	230
9.36.1	Detailed Description	231
9.37	mln::def Namespace Reference	231
9.37.1	Detailed Description	231
9.37.2	Typedef Documentation	231
9.37.2.1	coord	231
9.37.2.2	coordf	231
9.37.3	Enumeration Type Documentation	231
9.37.3.1	"@21	231
9.38	mln::display Namespace Reference	232
9.38.1	Detailed Description	232
9.39	mln::display::impl Namespace Reference	232
9.39.1	Detailed Description	232
9.40	mln::display::impl::generic Namespace Reference	232
9.40.1	Detailed Description	232
9.41	mln::doc Namespace Reference	232
9.41.1	Detailed Description	234
9.42	mln::draw Namespace Reference	234
9.42.1	Detailed Description	234

9.42.2	Function Documentation	234
9.42.2.1	box	234
9.42.2.2	box_plain	235
9.42.2.3	dashed_line	235
9.42.2.4	line	236
9.42.2.5	plot	236
9.42.2.6	polygon	237
9.42.2.7	site_set	237
9.43	mln::estim Namespace Reference	237
9.43.1	Detailed Description	238
9.43.2	Function Documentation	238
9.43.2.1	mean	238
9.43.2.2	mean	238
9.43.2.3	min_max	239
9.43.2.4	sum	239
9.43.2.5	sum	239
9.44	mln::extension Namespace Reference	239
9.44.1	Detailed Description	240
9.44.2	Function Documentation	240
9.44.2.1	adjust	240
9.44.2.2	adjust	241
9.44.2.3	adjust	241
9.44.2.4	adjust	241
9.44.2.5	adjust_duplicate	241
9.44.2.6	adjust_fill	241
9.44.2.7	duplicate	241
9.44.2.8	fill	242
9.45	mln::fun Namespace Reference	242
9.45.1	Detailed Description	243
9.46	mln::fun::access Namespace Reference	243
9.46.1	Detailed Description	243
9.47	mln::fun::i2v Namespace Reference	244
9.47.1	Detailed Description	244
9.47.2	Function Documentation	244
9.47.2.1	operator<<	244
9.48	mln::fun::n2v Namespace Reference	244

9.48.1 Detailed Description	244
9.49 mln::fun::p2b Namespace Reference	244
9.49.1 Detailed Description	245
9.50 mln::fun::p2p Namespace Reference	245
9.50.1 Detailed Description	245
9.51 mln::fun::p2v Namespace Reference	245
9.51.1 Detailed Description	245
9.52 mln::fun::stat Namespace Reference	245
9.52.1 Detailed Description	245
9.53 mln::fun::v2b Namespace Reference	245
9.53.1 Detailed Description	246
9.54 mln::fun::v2i Namespace Reference	246
9.54.1 Detailed Description	246
9.55 mln::fun::v2v Namespace Reference	246
9.55.1 Detailed Description	247
9.55.2 Variable Documentation	247
9.55.2.1 f_hsi_to_rgb_3x8	247
9.55.2.2 f_hsl_to_rgb_3x8	247
9.55.2.3 f_rgb_to_hsi_f	247
9.55.2.4 f_rgb_to_hsl_f	247
9.56 mln::fun::v2w2v Namespace Reference	248
9.56.1 Detailed Description	248
9.57 mln::fun::v2w_w2v Namespace Reference	248
9.57.1 Detailed Description	248
9.58 mln::fun::vv2b Namespace Reference	248
9.58.1 Detailed Description	249
9.59 mln::fun::vv2v Namespace Reference	249
9.59.1 Detailed Description	250
9.60 mln::fun::x2p Namespace Reference	250
9.60.1 Detailed Description	250
9.61 mln::fun::x2v Namespace Reference	250
9.61.1 Detailed Description	250
9.62 mln::fun::x2x Namespace Reference	250
9.62.1 Detailed Description	251
9.63 mln::geom Namespace Reference	251
9.63.1 Detailed Description	255

9.63.2	Function Documentation	255
9.63.2.1	bbox	255
9.63.2.2	bbox	255
9.63.2.3	bbox	255
9.63.2.4	bbox	256
9.63.2.5	chamfer	256
9.63.2.6	delta	256
9.63.2.7	delta	256
9.63.2.8	delta	256
9.63.2.9	horizontal_symmetry	256
9.63.2.10	max_col	257
9.63.2.11	max_col	257
9.63.2.12	max_ind	257
9.63.2.13	max_row	257
9.63.2.14	max_row	257
9.63.2.15	max_sli	257
9.63.2.16	mesh_corner_point_area	258
9.63.2.17	mesh_curvature	258
9.63.2.18	mesh_normal	258
9.63.2.19	min_col	259
9.63.2.20	min_col	259
9.63.2.21	min_ind	259
9.63.2.22	min_row	259
9.63.2.23	min_row	259
9.63.2.24	min_sli	259
9.63.2.25	ncols	260
9.63.2.26	ncols	260
9.63.2.27	ninds	260
9.63.2.28	nrows	260
9.63.2.29	nrows	260
9.63.2.30	nsites	261
9.63.2.31	nslis	261
9.63.2.32	pmin_pmax	261
9.63.2.33	pmin_pmax	261
9.63.2.34	pmin_pmax	261
9.63.2.35	pmin_pmax	261

9.63.2.36 rotate	262
9.63.2.37 rotate	262
9.63.2.38 rotate	262
9.63.2.39 rotate	262
9.63.2.40 rotate	263
9.63.2.41 seeds2tiling	263
9.63.2.42 translate	263
9.63.2.43 translate	264
9.63.2.44 translate	264
9.63.2.45 vertical_symmetry	264
9.64 mln::geom::impl Namespace Reference	264
9.64.1 Detailed Description	264
9.64.2 Function Documentation	265
9.64.2.1 seeds2tiling	265
9.65 mln::graph Namespace Reference	265
9.65.1 Detailed Description	265
9.65.2 Function Documentation	266
9.65.2.1 compute	266
9.65.2.2 labeling	266
9.65.2.3 to_neighb	266
9.65.2.4 to_win	267
9.66 mln::grid Namespace Reference	267
9.66.1 Detailed Description	267
9.67 mln::histo Namespace Reference	267
9.67.1 Detailed Description	268
9.67.2 Function Documentation	268
9.67.2.1 compute	268
9.67.2.2 equalize	268
9.68 mln::histo::impl Namespace Reference	269
9.68.1 Detailed Description	269
9.69 mln::histo::impl::generic Namespace Reference	269
9.69.1 Detailed Description	269
9.70 mln::impl Namespace Reference	269
9.70.1 Detailed Description	269
9.71 mln::io Namespace Reference	269
9.71.1 Detailed Description	271

9.72	mln::io::cloud Namespace Reference	271
9.72.1	Detailed Description	271
9.72.2	Function Documentation	271
9.72.2.1	load	271
9.72.2.2	save	271
9.73	mln::io::dicom Namespace Reference	272
9.73.1	Detailed Description	272
9.73.2	Function Documentation	272
9.73.2.1	get_header	272
9.73.2.2	load	272
9.74	mln::io::dump Namespace Reference	273
9.74.1	Detailed Description	273
9.74.2	Function Documentation	273
9.74.2.1	get_header	273
9.74.2.2	load	274
9.74.2.3	save	274
9.75	mln::io::fits Namespace Reference	274
9.75.1	Detailed Description	274
9.75.2	Function Documentation	274
9.75.2.1	load	274
9.75.2.2	load	275
9.76	mln::io::fld Namespace Reference	275
9.76.1	Detailed Description	275
9.76.2	Function Documentation	276
9.76.2.1	load	276
9.76.2.2	read_header	276
9.76.2.3	write_header	276
9.77	mln::io::magick Namespace Reference	276
9.77.1	Detailed Description	277
9.77.2	Function Documentation	277
9.77.2.1	load	277
9.77.2.2	save	277
9.77.2.3	save	278
9.78	mln::io::off Namespace Reference	278
9.78.1	Detailed Description	278
9.78.2	Function Documentation	278

9.78.2.1	load	278
9.78.2.2	save	279
9.78.2.3	save_bin_alt	279
9.79	mln::io::pbm Namespace Reference	279
9.79.1	Detailed Description	280
9.79.2	Function Documentation	280
9.79.2.1	load	280
9.79.2.2	load	280
9.79.2.3	save	281
9.80	mln::io::pbm::impl Namespace Reference	281
9.80.1	Detailed Description	281
9.81	mln::io::pbms Namespace Reference	281
9.81.1	Detailed Description	281
9.81.2	Function Documentation	281
9.81.2.1	load	281
9.82	mln::io::pbms::impl Namespace Reference	282
9.82.1	Detailed Description	282
9.83	mln::io::pfm Namespace Reference	282
9.83.1	Detailed Description	282
9.83.2	Function Documentation	283
9.83.2.1	load	283
9.83.2.2	load	283
9.83.2.3	save	283
9.84	mln::io::pfm::impl Namespace Reference	283
9.84.1	Detailed Description	283
9.85	mln::io::pgm Namespace Reference	284
9.85.1	Detailed Description	284
9.85.2	Function Documentation	284
9.85.2.1	load	284
9.85.2.2	load	284
9.85.2.3	save	285
9.86	mln::io::pgms Namespace Reference	285
9.86.1	Detailed Description	285
9.86.2	Function Documentation	285
9.86.2.1	load	285
9.87	mln::io::plot Namespace Reference	285

9.87.1	Detailed Description	286
9.87.2	Function Documentation	286
9.87.2.1	load	286
9.87.2.2	save	286
9.87.2.3	save	287
9.87.2.4	save	287
9.88	mln::io::pnm Namespace Reference	287
9.88.1	Detailed Description	288
9.88.2	Function Documentation	288
9.88.2.1	load	288
9.88.2.2	load	288
9.88.2.3	load_ascii_builtin	288
9.88.2.4	load_ascii_value	288
9.88.2.5	load_raw_2d	289
9.88.2.6	max_component	289
9.88.2.7	save	289
9.89	mln::io::pnm::impl Namespace Reference	289
9.89.1	Detailed Description	289
9.90	mln::io::pnms Namespace Reference	289
9.90.1	Detailed Description	290
9.90.2	Function Documentation	290
9.90.2.1	load	290
9.90.2.2	load	290
9.91	mln::io::ppm Namespace Reference	290
9.91.1	Detailed Description	291
9.91.2	Function Documentation	291
9.91.2.1	load	291
9.91.2.2	load	291
9.91.2.3	save	291
9.92	mln::io::ppms Namespace Reference	292
9.92.1	Detailed Description	292
9.92.2	Function Documentation	292
9.92.2.1	load	292
9.93	mln::io::raw Namespace Reference	292
9.93.1	Detailed Description	293
9.93.2	Function Documentation	293

9.93.2.1	get_header	293
9.93.2.2	load	293
9.93.2.3	save	293
9.94	mln::io::tiff Namespace Reference	294
9.94.1	Detailed Description	294
9.94.2	Function Documentation	294
9.94.2.1	load	294
9.95	mln::io::txt Namespace Reference	294
9.95.1	Detailed Description	294
9.95.2	Function Documentation	295
9.95.2.1	save	295
9.96	mln::labeling Namespace Reference	295
9.96.1	Detailed Description	298
9.96.2	Function Documentation	298
9.96.2.1	background	298
9.96.2.2	blobs	299
9.96.2.3	blobs_and_compute	299
9.96.2.4	colorize	300
9.96.2.5	colorize	300
9.96.2.6	colorize	300
9.96.2.7	compute	300
9.96.2.8	compute	301
9.96.2.9	compute	301
9.96.2.10	compute	302
9.96.2.11	compute	302
9.96.2.12	compute_image	302
9.96.2.13	compute_image	303
9.96.2.14	compute_image	303
9.96.2.15	fill_holes	304
9.96.2.16	flat_zones	304
9.96.2.17	foreground	305
9.96.2.18	pack	305
9.96.2.19	pack	306
9.96.2.20	pack_inplace	306
9.96.2.21	pack_inplace	306
9.96.2.22	regional_maxima	306

9.96.2.23	regional_minima	307
9.96.2.24	relabel	307
9.96.2.25	relabel	307
9.96.2.26	relabel_inplace	308
9.96.2.27	relabel_inplace	308
9.96.2.28	superpose	308
9.96.2.29	value	309
9.96.2.30	value_and_compute	309
9.96.2.31	wrap	310
9.96.2.32	wrap	310
9.97	mln::labeling::impl Namespace Reference	310
9.97.1	Detailed Description	311
9.97.2	Function Documentation	311
9.97.2.1	compute_fastest	311
9.97.2.2	compute_fastest	312
9.98	mln::labeling::impl::generic Namespace Reference	312
9.98.1	Detailed Description	312
9.98.2	Function Documentation	313
9.98.2.1	compute	313
9.98.2.2	compute	313
9.98.2.3	compute	313
9.98.2.4	compute	314
9.99	mln::linear Namespace Reference	314
9.99.1	Detailed Description	315
9.99.2	Function Documentation	315
9.99.2.1	gaussian	315
9.99.2.2	gaussian	316
9.99.2.3	gaussian_1st_derivative	316
9.99.2.4	gaussian_1st_derivative	316
9.99.2.5	gaussian_2nd_derivative	316
9.99.2.6	gaussian_2nd_derivative	317
9.99.2.7	mln_ch_convolve	317
9.99.2.8	mln_ch_convolve	317
9.99.2.9	mln_ch_convolve	317
9.99.2.10	mln_ch_convolve	318
9.99.2.11	mln_ch_convolve_grad	318

9.100mln::linear::impl Namespace Reference	318
9.100.1 Detailed Description	318
9.101mln::linear::local Namespace Reference	318
9.101.1 Detailed Description	319
9.101.2 Function Documentation	319
9.101.2.1 convolve	319
9.101.2.2 convolve	319
9.102mln::linear::local::impl Namespace Reference	320
9.102.1 Detailed Description	320
9.103mln::literal Namespace Reference	320
9.103.1 Detailed Description	323
9.103.2 Variable Documentation	323
9.103.2.1 black	323
9.103.2.2 blue	323
9.103.2.3 brown	323
9.103.2.4 cyan	323
9.103.2.5 dark_gray	323
9.103.2.6 green	324
9.103.2.7 identity	324
9.103.2.8 light_gray	324
9.103.2.9 lime	324
9.103.2.10magenta	324
9.103.2.11lmax	324
9.103.2.12medium_gray	324
9.103.2.13min	324
9.103.2.14olive	325
9.103.2.15one	325
9.103.2.16orange	325
9.103.2.17origin	325
9.103.2.18pink	325
9.103.2.19purple	325
9.103.2.20red	325
9.103.2.21teal	325
9.103.2.22violet	326
9.103.2.23white	326
9.103.2.24yellow	326

9.103.2.25zero	326
9.104mln::logical Namespace Reference	326
9.104.1 Detailed Description	327
9.104.2 Function Documentation	327
9.104.2.1 and_inplace	327
9.104.2.2 and_not	327
9.104.2.3 and_not_inplace	328
9.104.2.4 not_inplace	328
9.104.2.5 or_inplace	329
9.104.2.6 xor_inplace	329
9.105mln::logical::impl Namespace Reference	329
9.105.1 Detailed Description	330
9.106mln::logical::impl::generic Namespace Reference	330
9.106.1 Detailed Description	330
9.107mln::make Namespace Reference	330
9.107.1 Detailed Description	335
9.107.2 Function Documentation	335
9.107.2.1 attachment	335
9.107.2.2 box1d	335
9.107.2.3 box1d	336
9.107.2.4 box2d	336
9.107.2.5 box2d	337
9.107.2.6 box2d_h	337
9.107.2.7 box2d_h	338
9.107.2.8 box3d	338
9.107.2.9 box3d	338
9.107.2.10cell	339
9.107.2.11couple	339
9.107.2.12detachment	339
9.107.2.13dpoint2d_h	340
9.107.2.14dummy_p_edges	340
9.107.2.15dummy_p_edges	340
9.107.2.16dummy_p_vertices	341
9.107.2.17dummy_p_vertices	341
9.107.2.18edge_image	341
9.107.2.19edge_image	342

9.107.2.20	edge_image	342
9.107.2.21	edge_image	342
9.107.2.22	edge_image	343
9.107.2.23	edge_image	343
9.107.2.24	h_mat	343
9.107.2.25	image	343
9.107.2.26	image	344
9.107.2.27	image	344
9.107.2.28	image2d	344
9.107.2.29	image3d	345
9.107.2.30	image3d	345
9.107.2.31	influence_zone_adjacency_graph	345
9.107.2.32	mat	345
9.107.2.33	ord_pair	346
9.107.2.34	p_edges_with_mass_centers	346
9.107.2.35	p_vertices_with_mass_centers	346
9.107.2.36	pix	347
9.107.2.37	pixel	347
9.107.2.38	pixel	347
9.107.2.39	point2d_h	347
9.107.2.40	rag_and_labeled_wsl	347
9.107.2.41	region_adjacency_graph	348
9.107.2.42	relabelfun	348
9.107.2.43	relabelfun	349
9.107.2.44	vec	349
9.107.2.45	vec	350
9.107.2.46	vec	350
9.107.2.47	vec	350
9.107.2.48	vertex_image	351
9.107.2.49	vertex_image	351
9.107.2.50	voronoi	351
9.107.2.51	w_window	352
9.107.2.52	w_window1d	352
9.107.2.53	w_window1d_int	352
9.107.2.54	w_window2d	353
9.107.2.55	w_window2d_int	353

9.107.2.56w_window3d	353
9.107.2.57w_window3d_int	354
9.107.2.58w_window_directional	354
9.108mln::math Namespace Reference	355
9.108.1 Detailed Description	355
9.108.2 Function Documentation	355
9.108.2.1 abs	355
9.108.2.2 abs	355
9.108.2.3 abs	355
9.109mln::metal Namespace Reference	356
9.109.1 Detailed Description	356
9.110mln::metal::impl Namespace Reference	356
9.110.1 Detailed Description	357
9.111mln::metal::math Namespace Reference	357
9.111.1 Detailed Description	357
9.112mln::metal::math::impl Namespace Reference	357
9.112.1 Detailed Description	357
9.113mln::morpho Namespace Reference	357
9.113.1 Detailed Description	360
9.113.2 Function Documentation	360
9.113.2.1 complementation	360
9.113.2.2 complementation_inplace	360
9.113.2.3 contrast	361
9.113.2.4 dilation	361
9.113.2.5 erosion	361
9.113.2.6 general	361
9.113.2.7 gradient	361
9.113.2.8 gradient_external	362
9.113.2.9 gradient_internal	362
9.113.2.10hit_or_miss	362
9.113.2.11hit_or_miss_background_closing	362
9.113.2.12hit_or_miss_background_opening	363
9.113.2.13hit_or_miss_closing	363
9.113.2.14hit_or_miss_opening	363
9.113.2.15laplacian	363
9.113.2.16line_gradient	363

9.113.2.17	meyer_wst	364
9.113.2.18	meyer_wst	364
9.113.2.19	min	364
9.113.2.20	min_inplace	365
9.113.2.21	minus	365
9.113.2.22	plus	365
9.113.2.23	rank_filter	365
9.113.2.24	thick_miss	365
9.113.2.25	thickening	366
9.113.2.26	thin_fit	366
9.113.2.27	thinning	366
9.113.2.28	top_hat_black	366
9.113.2.29	top_hat_self_complementary	366
9.113.2.30	top_hat_white	367
9.114	mln::morpho::approx Namespace Reference	367
9.114.1	Detailed Description	367
9.115	mln::morpho::attribute Namespace Reference	367
9.115.1	Detailed Description	368
9.116	mln::morpho::closing::approx Namespace Reference	368
9.116.1	Detailed Description	368
9.116.2	Function Documentation	368
9.116.2.1	structural	368
9.117	mln::morpho::elementary Namespace Reference	368
9.117.1	Detailed Description	369
9.117.2	Function Documentation	369
9.117.2.1	closing	369
9.117.2.2	mln_trait_op_minus_twice	369
9.117.2.3	opening	370
9.117.2.4	top_hat_black	370
9.117.2.5	top_hat_self_complementary	370
9.117.2.6	top_hat_white	370
9.118	mln::morpho::impl Namespace Reference	370
9.118.1	Detailed Description	371
9.119	mln::morpho::impl::generic Namespace Reference	371
9.119.1	Detailed Description	371
9.120	mln::morpho::opening::approx Namespace Reference	371

9.120.1 Detailed Description	371
9.120.2 Function Documentation	371
9.120.2.1 structural	371
9.121mln::morpho::reconstruction Namespace Reference	372
9.121.1 Detailed Description	372
9.122mln::morpho::reconstruction::by_dilation Namespace Reference	372
9.122.1 Detailed Description	372
9.123mln::morpho::reconstruction::by_erosion Namespace Reference	372
9.123.1 Detailed Description	372
9.124mln::morpho::tree Namespace Reference	372
9.124.1 Detailed Description	374
9.124.2 Function Documentation	374
9.124.2.1 compute_attribute_image	374
9.124.2.2 compute_attribute_image_from	375
9.124.2.3 compute_parent	375
9.124.2.4 dual_input_max_tree	376
9.124.2.5 max_tree	376
9.124.2.6 min_tree	377
9.124.2.7 propagate_if	377
9.124.2.8 propagate_if	378
9.124.2.9 propagate_if_value	378
9.124.2.10propagate_if_value	378
9.124.2.11propagate_node_to_ancestors	378
9.124.2.12propagate_node_to_ancestors	379
9.124.2.13propagate_node_to_descendants	379
9.124.2.14propagate_node_to_descendants	379
9.124.2.15propagate_representative	380
9.125mln::morpho::tree::filter Namespace Reference	380
9.125.1 Detailed Description	380
9.125.2 Function Documentation	381
9.125.2.1 direct	381
9.125.2.2 filter	381
9.125.2.3 max	381
9.125.2.4 min	382
9.125.2.5 subtractive	382
9.126mln::morpho::watershed Namespace Reference	382

9.126.1 Detailed Description	383
9.126.2 Function Documentation	383
9.126.2.1 flooding	383
9.126.2.2 flooding	384
9.126.2.3 superpose	384
9.126.2.4 superpose	384
9.126.2.5 topological	384
9.127mln::morpho::watershed::watershed Namespace Reference	385
9.127.1 Detailed Description	385
9.128mln::morpho::watershed::watershed::generic Namespace Reference	385
9.128.1 Detailed Description	385
9.129mln::norm Namespace Reference	385
9.129.1 Detailed Description	386
9.129.2 Function Documentation	386
9.129.2.1 l1	386
9.129.2.2 l1_distance	386
9.129.2.3 l2	387
9.129.2.4 l2_distance	387
9.129.2.5 linfty	387
9.129.2.6 linfty_distance	387
9.129.2.7 sqr_l2	387
9.130mln::norm::impl Namespace Reference	387
9.130.1 Detailed Description	387
9.131mln::opt Namespace Reference	388
9.131.1 Detailed Description	388
9.131.2 Function Documentation	388
9.131.2.1 at	388
9.131.2.2 at	389
9.131.2.3 at	389
9.131.2.4 at	389
9.131.2.5 at	389
9.131.2.6 at	389
9.132mln::opt::impl Namespace Reference	389
9.132.1 Detailed Description	389
9.133mln::pw Namespace Reference	389
9.133.1 Detailed Description	390

9.134mln::registration Namespace Reference	390
9.134.1 Detailed Description	391
9.134.2 Function Documentation	391
9.134.2.1 get_rot	391
9.134.2.2 icp	391
9.134.2.3 icp	392
9.134.2.4 registration1	392
9.134.2.5 registration2	392
9.134.2.6 registration3	392
9.135mln::select Namespace Reference	393
9.135.1 Detailed Description	393
9.136mln::set Namespace Reference	393
9.136.1 Detailed Description	394
9.136.2 Function Documentation	394
9.136.2.1 card	394
9.136.2.2 compute	394
9.136.2.3 compute_with_weights	394
9.136.2.4 compute_with_weights	395
9.136.2.5 get	395
9.136.2.6 has	395
9.136.2.7 mln_meta_accu_result	395
9.136.2.8 mln_meta_accu_result	395
9.137mln::subsampling Namespace Reference	396
9.137.1 Detailed Description	396
9.137.2 Function Documentation	396
9.137.2.1 antialiased	396
9.137.2.2 antialiased	397
9.137.2.3 gaussian_subsampling	397
9.137.2.4 subsampling	397
9.138mln::tag Namespace Reference	397
9.138.1 Detailed Description	397
9.139mln::test Namespace Reference	397
9.139.1 Detailed Description	398
9.139.2 Function Documentation	398
9.139.2.1 positive	398
9.139.2.2 predicate	398

9.139.2.3 predicate	399
9.139.2.4 predicate	399
9.140 mln::test::impl Namespace Reference	399
9.140.1 Detailed Description	399
9.141 mln::topo Namespace Reference	399
9.141.1 Detailed Description	404
9.141.2 Function Documentation	404
9.141.2.1 detach	404
9.141.2.2 edge	404
9.141.2.3 is_facet	405
9.141.2.4 make_algebraic_face	405
9.141.2.5 make_algebraic_n_face	405
9.141.2.6 operator!=	405
9.141.2.7 operator!=	405
9.141.2.8 operator!=	405
9.141.2.9 operator!=	406
9.141.2.10 operator+	406
9.141.2.11 operator-	406
9.141.2.12 operator-	406
9.141.2.13 operator-	406
9.141.2.14 operator<	407
9.141.2.15 operator<	407
9.141.2.16 operator<	407
9.141.2.17 operator<	407
9.141.2.18 operator<<	408
9.141.2.19 operator<<	408
9.141.2.20 operator<<	408
9.141.2.21 operator<<	408
9.141.2.22 operator<<	408
9.141.2.23 operator==	408
9.141.2.24 operator==	409
9.141.2.25 operator==	409
9.141.2.26 operator==	409
9.141.2.27 operator==	409
9.142 mln::trace Namespace Reference	410
9.142.1 Detailed Description	410

9.143	mln::trait Namespace Reference	410
9.143.1	Detailed Description	410
9.144	mln::transform Namespace Reference	410
9.144.1	Detailed Description	411
9.144.2	Function Documentation	411
9.144.2.1	distance_and_closest_point_geodesic	411
9.144.2.2	distance_and_closest_point_geodesic	412
9.144.2.3	distance_and_influence_zone_geodesic	412
9.144.2.4	distance_front	413
9.144.2.5	distance_geodesic	413
9.144.2.6	hough	413
9.144.2.7	influence_zone_front	414
9.144.2.8	influence_zone_front	414
9.144.2.9	influence_zone_geodesic	414
9.144.2.10	influence_zone_geodesic_saturated	414
9.144.2.11	influence_zone_geodesic_saturated	415
9.145	mln::util Namespace Reference	415
9.145.1	Detailed Description	418
9.145.2	Typedef Documentation	418
9.145.2.1	vertex_id_t	418
9.145.3	Function Documentation	419
9.145.3.1	display_branch	419
9.145.3.2	display_tree	419
9.145.3.3	lemmings	419
9.145.3.4	make_greater_point	419
9.145.3.5	make_greater_psite	420
9.145.3.6	operator<	420
9.145.3.7	operator<<	420
9.145.3.8	operator<<	420
9.145.3.9	operator==	420
9.145.3.10	operator==	420
9.145.3.11	lord_strict	420
9.145.3.12	lord_weak	421
9.145.3.13	tree_fast_to_image	421
9.145.3.14	tree_to_fast	421
9.145.3.15	tree_to_image	421

9.146	mln::util::impl Namespace Reference	422
9.146.1	Detailed Description	422
9.147	mln::value Namespace Reference	422
9.147.1	Detailed Description	426
9.147.2	Typedef Documentation	426
9.147.2.1	float01_16	426
9.147.2.2	float01_8	427
9.147.2.3	gl16	427
9.147.2.4	gl8	427
9.147.2.5	glf	427
9.147.2.6	int_s16	427
9.147.2.7	int_s32	427
9.147.2.8	int_s8	427
9.147.2.9	int_u12	427
9.147.2.10	int_u16	427
9.147.2.11	int_u32	428
9.147.2.12	int_u8	428
9.147.2.13	label_16	428
9.147.2.14	label_32	428
9.147.2.15	label_8	428
9.147.2.16	rgb16	428
9.147.2.17	rgb8	428
9.147.3	Function Documentation	428
9.147.3.1	cast	428
9.147.3.2	equiv	429
9.147.3.3	operator*	429
9.147.3.4	operator*	429
9.147.3.5	operator+	429
9.147.3.6	operator+	429
9.147.3.7	operator-	429
9.147.3.8	operator-	429
9.147.3.9	operator/	430
9.147.3.10	operator/	430
9.147.3.11	operator<<	430
9.147.3.12	operator<<	430
9.147.3.13	operator<<	430

9.147.3.14	operator<<	431
9.147.3.15	operator<<	431
9.147.3.16	operator<<	431
9.147.3.17	operator<<	431
9.147.3.18	operator<<	432
9.147.3.19	operator<<	432
9.147.3.20	operator<<	432
9.147.3.21	operator<<	433
9.147.3.22	operator==	433
9.147.3.23	operator==	433
9.147.3.24	other	433
9.147.3.25	stack	433
9.148	mln::value::impl Namespace Reference	433
9.148.1	Detailed Description	433
9.149	mln::win Namespace Reference	433
9.149.1	Detailed Description	435
9.149.2	Function Documentation	435
9.149.2.1	diff	435
9.149.2.2	mln_regular	435
9.149.2.3	mln_regular	436
9.149.2.4	sym	436
9.149.2.5	sym	436
10	Class Documentation	437
10.1	mln::accu::center< P, V > Struct Template Reference	437
10.1.1	Detailed Description	437
10.1.2	Member Function Documentation	438
10.1.2.1	init	438
10.1.2.2	is_valid	438
10.1.2.3	nsites	438
10.1.2.4	take_as_init	438
10.1.2.5	take_n_times	438
10.1.2.6	to_result	438
10.2	mln::accu::convolve< T1, T2, R > Struct Template Reference	439
10.2.1	Detailed Description	439
10.2.2	Member Function Documentation	439
10.2.2.1	init	439

10.2.2.2	is_valid	440
10.2.2.3	take_as_init	440
10.2.2.4	take_n_times	440
10.2.2.5	to_result	440
10.3	mln::accu::count_adjacent_vertices< F, S > Struct Template Reference	440
10.3.1	Detailed Description	441
10.3.2	Member Function Documentation	441
10.3.2.1	init	441
10.3.2.2	is_valid	441
10.3.2.3	set_value	441
10.3.2.4	take_as_init	441
10.3.2.5	take_n_times	442
10.3.2.6	to_result	442
10.4	mln::accu::count_labels< L > Struct Template Reference	442
10.4.1	Detailed Description	442
10.4.2	Member Function Documentation	443
10.4.2.1	init	443
10.4.2.2	is_valid	443
10.4.2.3	set_value	443
10.4.2.4	take_as_init	443
10.4.2.5	take_n_times	443
10.4.2.6	to_result	443
10.5	mln::accu::count_value< V > Struct Template Reference	443
10.5.1	Detailed Description	444
10.5.2	Member Function Documentation	444
10.5.2.1	init	444
10.5.2.2	is_valid	444
10.5.2.3	set_value	444
10.5.2.4	take_as_init	445
10.5.2.5	take_n_times	445
10.5.2.6	to_result	445
10.6	mln::accu::histo< V > Struct Template Reference	445
10.6.1	Detailed Description	446
10.6.2	Member Function Documentation	446
10.6.2.1	is_valid	446
10.6.2.2	take	446

10.6.2.3	take_as_init	446
10.6.2.4	take_n_times	446
10.6.2.5	vect	446
10.7	mln::accu::label_used< L > Struct Template Reference	446
10.7.1	Detailed Description	447
10.7.2	Member Function Documentation	447
10.7.2.1	init	447
10.7.2.2	is_valid	447
10.7.2.3	take	447
10.7.2.4	take_as_init	448
10.7.2.5	take_n_times	448
10.7.2.6	to_result	448
10.8	mln::accu::logic::land Struct Reference	448
10.8.1	Detailed Description	449
10.8.2	Member Function Documentation	449
10.8.2.1	init	449
10.8.2.2	is_valid	449
10.8.2.3	take_as_init	449
10.8.2.4	take_n_times	449
10.8.2.5	to_result	449
10.9	mln::accu::logic::land_basic Struct Reference	449
10.9.1	Detailed Description	450
10.9.2	Member Function Documentation	450
10.9.2.1	can_stop	450
10.9.2.2	init	450
10.9.2.3	is_valid	450
10.9.2.4	take_as_init	451
10.9.2.5	take_n_times	451
10.9.2.6	to_result	451
10.10	mln::accu::logic::lor Struct Reference	451
10.10.1	Detailed Description	451
10.10.2	Member Function Documentation	452
10.10.2.1	init	452
10.10.2.2	is_valid	452
10.10.2.3	take_as_init	452
10.10.2.4	take_n_times	452

10.10.2.5 <code>to_result</code>	452
10.11 <code>mln::accu::logic::lor_basic</code> Struct Reference	452
10.11.1 Detailed Description	453
10.11.2 Member Function Documentation	453
10.11.2.1 <code>can_stop</code>	453
10.11.2.2 <code>init</code>	453
10.11.2.3 <code>is_valid</code>	453
10.11.2.4 <code>take_as_init</code>	453
10.11.2.5 <code>take_n_times</code>	454
10.11.2.6 <code>to_result</code>	454
10.12 <code>mln::accu::maj_h< T ></code> Struct Template Reference	454
10.12.1 Detailed Description	454
10.12.2 Member Function Documentation	455
10.12.2.1 <code>init</code>	455
10.12.2.2 <code>is_valid</code>	455
10.12.2.3 <code>take_as_init</code>	455
10.12.2.4 <code>take_n_times</code>	455
10.12.2.5 <code>to_result</code>	455
10.13 <code>mln::accu::math::count< T ></code> Struct Template Reference	455
10.13.1 Detailed Description	456
10.13.2 Member Function Documentation	456
10.13.2.1 <code>init</code>	456
10.13.2.2 <code>is_valid</code>	456
10.13.2.3 <code>set_value</code>	456
10.13.2.4 <code>take_as_init</code>	456
10.13.2.5 <code>take_n_times</code>	457
10.13.2.6 <code>to_result</code>	457
10.14 <code>mln::accu::math::inf< T ></code> Struct Template Reference	457
10.14.1 Detailed Description	457
10.14.2 Member Function Documentation	458
10.14.2.1 <code>init</code>	458
10.14.2.2 <code>is_valid</code>	458
10.14.2.3 <code>take_as_init</code>	458
10.14.2.4 <code>take_n_times</code>	458
10.14.2.5 <code>to_result</code>	458
10.15 <code>mln::accu::math::sum< T, S ></code> Struct Template Reference	458

10.15.1 Detailed Description	459
10.15.2 Member Function Documentation	459
10.15.2.1 init	459
10.15.2.2 is_valid	459
10.15.2.3 take_as_init	459
10.15.2.4 take_n_times	459
10.15.2.5 to_result	460
10.16 <code>mln::accu::math::sup< T ></code> Struct Template Reference	460
10.16.1 Detailed Description	460
10.16.2 Member Function Documentation	460
10.16.2.1 init	460
10.16.2.2 is_valid	461
10.16.2.3 take_as_init	461
10.16.2.4 take_n_times	461
10.16.2.5 to_result	461
10.17 <code>mln::accu::max_site< I ></code> Struct Template Reference	461
10.17.1 Detailed Description	462
10.17.2 Member Function Documentation	462
10.17.2.1 init	462
10.17.2.2 is_valid	462
10.17.2.3 take_as_init	462
10.17.2.4 take_n_times	462
10.17.2.5 to_result	462
10.18 <code>mln::accu::meta::center</code> Struct Reference	463
10.18.1 Detailed Description	463
10.19 <code>mln::accu::meta::count_adjacent_vertices</code> Struct Reference	463
10.19.1 Detailed Description	464
10.20 <code>mln::accu::meta::count_labels</code> Struct Reference	464
10.20.1 Detailed Description	465
10.21 <code>mln::accu::meta::count_value</code> Struct Reference	465
10.21.1 Detailed Description	466
10.22 <code>mln::accu::meta::histo</code> Struct Reference	466
10.22.1 Detailed Description	467
10.23 <code>mln::accu::meta::label_used</code> Struct Reference	467
10.23.1 Detailed Description	468
10.24 <code>mln::accu::meta::logic::land</code> Struct Reference	468

10.24.1 Detailed Description	469
10.25mln::accu::meta::logic::lor Struct Reference	469
10.25.1 Detailed Description	470
10.26mln::accu::meta::logic::lor_basic Struct Reference	470
10.26.1 Detailed Description	471
10.27mln::accu::meta::maj_h Struct Reference	471
10.27.1 Detailed Description	472
10.28mln::accu::meta::math::count Struct Reference	472
10.28.1 Detailed Description	473
10.29mln::accu::meta::math::inf Struct Reference	473
10.29.1 Detailed Description	474
10.30mln::accu::meta::math::sum Struct Reference	474
10.30.1 Detailed Description	475
10.31mln::accu::meta::math::sup Struct Reference	475
10.31.1 Detailed Description	476
10.32mln::accu::meta::max_site Struct Reference	476
10.32.1 Detailed Description	477
10.33mln::accu::meta::nil Struct Reference	477
10.33.1 Detailed Description	478
10.34mln::accu::meta::p< mA > Struct Template Reference	478
10.34.1 Detailed Description	479
10.35mln::accu::meta::pair< A1, A2 > Struct Template Reference	479
10.35.1 Detailed Description	480
10.36mln::accu::meta::rms Struct Reference	480
10.36.1 Detailed Description	481
10.37mln::accu::meta::shape::bbox Struct Reference	481
10.37.1 Detailed Description	482
10.38mln::accu::meta::shape::height Struct Reference	482
10.38.1 Detailed Description	483
10.39mln::accu::meta::shape::volume Struct Reference	483
10.39.1 Detailed Description	484
10.40mln::accu::meta::stat::max Struct Reference	484
10.40.1 Detailed Description	485
10.41mln::accu::meta::stat::max_h Struct Reference	485
10.41.1 Detailed Description	486
10.42mln::accu::meta::stat::mean Struct Reference	486

10.42.1 Detailed Description	487
10.43 <code>mln::accu::meta::stat::median_alt< T ></code> Struct Template Reference	487
10.43.1 Detailed Description	488
10.44 <code>mln::accu::meta::stat::median_h</code> Struct Reference	488
10.44.1 Detailed Description	489
10.45 <code>mln::accu::meta::stat::min</code> Struct Reference	489
10.45.1 Detailed Description	490
10.46 <code>mln::accu::meta::stat::min_h</code> Struct Reference	490
10.46.1 Detailed Description	491
10.47 <code>mln::accu::meta::stat::rank</code> Struct Reference	491
10.47.1 Detailed Description	492
10.48 <code>mln::accu::meta::stat::rank_high_quant</code> Struct Reference	492
10.48.1 Detailed Description	493
10.49 <code>mln::accu::meta::tuple< n, ></code> Struct Template Reference	493
10.49.1 Detailed Description	494
10.50 <code>mln::accu::meta::val< mA ></code> Struct Template Reference	494
10.50.1 Detailed Description	495
10.51 <code>mln::accu::nil< T ></code> Struct Template Reference	495
10.51.1 Detailed Description	496
10.51.2 Member Function Documentation	496
10.51.2.1 <code>init</code>	496
10.51.2.2 <code>is_valid</code>	496
10.51.2.3 <code>take_as_init</code>	496
10.51.2.4 <code>take_n_times</code>	496
10.51.2.5 <code>to_result</code>	497
10.52 <code>mln::accu::p< A ></code> Struct Template Reference	497
10.52.1 Detailed Description	497
10.52.2 Member Function Documentation	497
10.52.2.1 <code>init</code>	497
10.52.2.2 <code>is_valid</code>	498
10.52.2.3 <code>take_as_init</code>	498
10.52.2.4 <code>take_n_times</code>	498
10.52.2.5 <code>to_result</code>	498
10.53 <code>mln::accu::pair< A1, A2, T ></code> Struct Template Reference	498
10.53.1 Detailed Description	500
10.53.2 Member Function Documentation	500

10.53.2.1	first	500
10.53.2.2	first_accu	500
10.53.2.3	init	500
10.53.2.4	is_valid	500
10.53.2.5	second	500
10.53.2.6	second_accu	501
10.53.2.7	take_as_init	501
10.53.2.8	take_n_times	501
10.53.2.9	to_result	501
10.54	mln::accu::rms< T, V > Struct Template Reference	501
10.54.1	Detailed Description	502
10.54.2	Member Function Documentation	502
10.54.2.1	init	502
10.54.2.2	is_valid	502
10.54.2.3	take_as_init	502
10.54.2.4	take_n_times	502
10.54.2.5	to_result	502
10.55	mln::accu::shape::bbox< P > Struct Template Reference	503
10.55.1	Detailed Description	503
10.55.2	Member Function Documentation	503
10.55.2.1	init	503
10.55.2.2	is_valid	503
10.55.2.3	take_as_init	504
10.55.2.4	take_n_times	504
10.55.2.5	to_result	504
10.56	mln::accu::shape::height< I > Struct Template Reference	504
10.56.1	Detailed Description	505
10.56.2	Member Typedef Documentation	505
10.56.2.1	argument	505
10.56.2.2	value	505
10.56.3	Member Function Documentation	505
10.56.3.1	init	505
10.56.3.2	is_valid	505
10.56.3.3	set_value	506
10.56.3.4	take_as_init	506
10.56.3.5	take_n_times	506

10.56.3.6	to_result	506
10.57	mln::accu::shape::volume< I > Struct Template Reference	506
10.57.1	Detailed Description	507
10.57.2	Member Typedef Documentation	507
10.57.2.1	argument	507
10.57.2.2	value	507
10.57.3	Member Function Documentation	507
10.57.3.1	init	507
10.57.3.2	is_valid	508
10.57.3.3	set_value	508
10.57.3.4	take_as_init	508
10.57.3.5	take_n_times	508
10.57.3.6	to_result	508
10.58	mln::accu::site_set::rectangularity< P > Class Template Reference	508
10.58.1	Detailed Description	509
10.58.2	Constructor & Destructor Documentation	509
10.58.2.1	rectangularity	509
10.58.3	Member Function Documentation	509
10.58.3.1	area	509
10.58.3.2	bbox	509
10.58.3.3	take_as_init	510
10.58.3.4	take_n_times	510
10.58.3.5	to_result	510
10.59	mln::accu::stat::deviation< T, S, M > Struct Template Reference	510
10.59.1	Detailed Description	511
10.59.2	Member Function Documentation	511
10.59.2.1	init	511
10.59.2.2	is_valid	511
10.59.2.3	take_as_init	511
10.59.2.4	take_n_times	511
10.59.2.5	to_result	511
10.60	mln::accu::stat::histo3d_rgb< V > Struct Template Reference	512
10.60.1	Detailed Description	512
10.60.2	Constructor & Destructor Documentation	513
10.60.2.1	histo3d_rgb	513
10.60.3	Member Function Documentation	513

10.60.3.1	init	513
10.60.3.2	is_valid	513
10.60.3.3	take	513
10.60.3.4	take	513
10.60.3.5	take_as_init	514
10.60.3.6	take_n_times	514
10.60.3.7	to_result	514
10.61	mln::accu::stat::max< T > Struct Template Reference	514
10.61.1	Detailed Description	515
10.61.2	Member Function Documentation	515
10.61.2.1	init	515
10.61.2.2	is_valid	515
10.61.2.3	set_value	515
10.61.2.4	take_as_init	515
10.61.2.5	take_n_times	515
10.61.2.6	to_result	516
10.62	mln::accu::stat::max_h< V > Struct Template Reference	516
10.62.1	Detailed Description	516
10.62.2	Member Function Documentation	516
10.62.2.1	init	516
10.62.2.2	is_valid	517
10.62.2.3	take_as_init	517
10.62.2.4	take_n_times	517
10.62.2.5	to_result	517
10.63	mln::accu::stat::mean< T, S, M > Struct Template Reference	517
10.63.1	Detailed Description	518
10.63.2	Member Function Documentation	518
10.63.2.1	count	518
10.63.2.2	init	518
10.63.2.3	is_valid	518
10.63.2.4	sum	518
10.63.2.5	take_as_init	519
10.63.2.6	take_n_times	519
10.63.2.7	to_result	519
10.64	mln::accu::stat::median_alt< S > Struct Template Reference	519
10.64.1	Detailed Description	520

10.64.2 Member Function Documentation	520
10.64.2.1 is_valid	520
10.64.2.2 take	520
10.64.2.3 take_as_init	520
10.64.2.4 take_n_times	520
10.64.2.5 to_result	521
10.65mln::accu::stat::median_h< V > Struct Template Reference	521
10.65.1 Detailed Description	522
10.65.2 Member Function Documentation	522
10.65.2.1 init	522
10.65.2.2 is_valid	522
10.65.2.3 take_as_init	522
10.65.2.4 take_n_times	522
10.65.2.5 to_result	522
10.66mln::accu::stat::meta::deviation Struct Reference	522
10.66.1 Detailed Description	523
10.67mln::accu::stat::min< T > Struct Template Reference	523
10.67.1 Detailed Description	524
10.67.2 Member Function Documentation	524
10.67.2.1 init	524
10.67.2.2 is_valid	524
10.67.2.3 set_value	524
10.67.2.4 take_as_init	524
10.67.2.5 take_n_times	525
10.67.2.6 to_result	525
10.68mln::accu::stat::min_h< V > Struct Template Reference	525
10.68.1 Detailed Description	525
10.68.2 Member Function Documentation	526
10.68.2.1 init	526
10.68.2.2 is_valid	526
10.68.2.3 take_as_init	526
10.68.2.4 take_n_times	526
10.68.2.5 to_result	526
10.69mln::accu::stat::min_max< V > Struct Template Reference	526
10.69.1 Detailed Description	528
10.69.2 Member Function Documentation	528

10.69.2.1	first	528
10.69.2.2	first_accu	528
10.69.2.3	init	528
10.69.2.4	is_valid	528
10.69.2.5	second	528
10.69.2.6	second_accu	528
10.69.2.7	take_as_init	529
10.69.2.8	take_n_times	529
10.69.2.9	to_result	529
10.70	mln::accu::stat::rank< T > Struct Template Reference	529
10.70.1	Detailed Description	530
10.70.2	Member Function Documentation	530
10.70.2.1	init	530
10.70.2.2	is_valid	530
10.70.2.3	k	530
10.70.2.4	take_as_init	530
10.70.2.5	take_n_times	530
10.70.2.6	to_result	531
10.71	mln::accu::stat::rank< bool > Struct Template Reference	531
10.71.1	Detailed Description	531
10.71.2	Member Function Documentation	531
10.71.2.1	init	531
10.71.2.2	is_valid	532
10.71.2.3	take_as_init	532
10.71.2.4	take_n_times	532
10.71.2.5	to_result	532
10.72	mln::accu::stat::rank_high_quant< T > Struct Template Reference	532
10.72.1	Detailed Description	533
10.72.2	Member Function Documentation	533
10.72.2.1	init	533
10.72.2.2	is_valid	533
10.72.2.3	take_as_init	533
10.72.2.4	take_n_times	533
10.72.2.5	to_result	533
10.73	mln::accu::stat::var< T > Struct Template Reference	534
10.73.1	Detailed Description	534

10.73.2 Member Typedef Documentation	535
10.73.2.1 mean_t	535
10.73.3 Member Function Documentation	535
10.73.3.1 init	535
10.73.3.2 is_valid	535
10.73.3.3 mean	535
10.73.3.4 n_items	535
10.73.3.5 take_as_init	535
10.73.3.6 take_n_times	535
10.73.3.7 to_result	536
10.73.3.8 variance	536
10.74mln::accu::stat::variance< T, S, R > Struct Template Reference	536
10.74.1 Detailed Description	537
10.74.2 Member Function Documentation	537
10.74.2.1 init	537
10.74.2.2 is_valid	537
10.74.2.3 mean	537
10.74.2.4 n_items	537
10.74.2.5 standard_deviation	538
10.74.2.6 sum	538
10.74.2.7 take_as_init	538
10.74.2.8 take_n_times	538
10.74.2.9 to_result	538
10.74.2.10var	538
10.75mln::accu::tuple< A, n, > Struct Template Reference	538
10.75.1 Detailed Description	539
10.75.2 Member Function Documentation	539
10.75.2.1 init	539
10.75.2.2 is_valid	539
10.75.2.3 take_as_init	540
10.75.2.4 take_n_times	540
10.75.2.5 to_result	540
10.76mln::accu::val< A > Struct Template Reference	540
10.76.1 Detailed Description	541
10.76.2 Member Function Documentation	541
10.76.2.1 init	541

10.76.2.2	is_valid	541
10.76.2.3	take_as_init	541
10.76.2.4	take_n_times	541
10.76.2.5	to_result	541
10.77	mln::Accumulator< E > Struct Template Reference	541
10.77.1	Detailed Description	543
10.77.2	Member Function Documentation	543
10.77.2.1	take_as_init	543
10.77.2.2	take_n_times	543
10.78	mln::algebra::h_mat< d, T > Struct Template Reference	543
10.78.1	Detailed Description	544
10.78.2	Member Enumeration Documentation	544
10.78.2.1	"@7	544
10.78.3	Constructor & Destructor Documentation	544
10.78.3.1	h_mat	544
10.78.3.2	h_mat	544
10.78.4	Member Function Documentation	545
10.78.4.1	_1	545
10.78.4.2	t	545
10.79	mln::algebra::h_vec< d, C > Class Template Reference	545
10.79.1	Detailed Description	546
10.79.2	Member Enumeration Documentation	546
10.79.2.1	"@8	546
10.79.3	Constructor & Destructor Documentation	546
10.79.3.1	h_vec	546
10.79.3.2	h_vec	546
10.79.4	Member Function Documentation	546
10.79.4.1	operator mat< n, 1, U >	546
10.79.4.2	t	547
10.79.4.3	to_vec	547
10.79.5	Member Data Documentation	547
10.79.5.1	origin	547
10.79.5.2	zero	547
10.80	mln::bkd_pixterId< I > Class Template Reference	547
10.80.1	Detailed Description	548
10.80.2	Member Typedef Documentation	548

10.80.2.1 image	548
10.80.3 Constructor & Destructor Documentation	548
10.80.3.1 bkd_pixter1d	548
10.80.4 Member Function Documentation	548
10.80.4.1 next	548
10.81 mln::bkd_pixter2d< I > Class Template Reference	548
10.81.1 Detailed Description	549
10.81.2 Member Typedef Documentation	549
10.81.2.1 image	549
10.81.3 Constructor & Destructor Documentation	549
10.81.3.1 bkd_pixter2d	549
10.81.4 Member Function Documentation	549
10.81.4.1 next	549
10.82 mln::bkd_pixter3d< I > Class Template Reference	550
10.82.1 Detailed Description	550
10.82.2 Member Typedef Documentation	550
10.82.2.1 image	550
10.82.3 Constructor & Destructor Documentation	551
10.82.3.1 bkd_pixter3d	551
10.82.4 Member Function Documentation	551
10.82.4.1 next	551
10.83 mln::box< P > Class Template Reference	551
10.83.1 Detailed Description	554
10.83.2 Member Typedef Documentation	554
10.83.2.1 bkd_piter	554
10.83.2.2 element	554
10.83.2.3 fwd_piter	555
10.83.2.4 piter	555
10.83.2.5 psite	555
10.83.2.6 site	555
10.83.3 Member Enumeration Documentation	555
10.83.3.1 "@31	555
10.83.4 Constructor & Destructor Documentation	555
10.83.4.1 box	555
10.83.4.2 box	555
10.83.4.3 box	555

10.83.5 Member Function Documentation	556
10.83.5.1 bbox	556
10.83.5.2 crop_wrt	556
10.83.5.3 enlarge	556
10.83.5.4 enlarge	556
10.83.5.5 has	556
10.83.5.6 is_empty	557
10.83.5.7 is_valid	557
10.83.5.8 len	557
10.83.5.9 memory_size	557
10.83.5.10 merge	557
10.83.5.11 nsites	557
10.83.5.12 pcenter	558
10.83.5.13 pmax	558
10.83.5.14 pmax	558
10.83.5.15 pmin	558
10.83.5.16 pmin	558
10.83.5.17 to_larger	558
10.83.6 Friends And Related Function Documentation	559
10.83.6.1 operator<<	559
10.84 mIn::Box< E > Struct Template Reference	559
10.84.1 Detailed Description	561
10.84.2 Member Function Documentation	562
10.84.2.1 bbox	562
10.84.2.2 is_empty	562
10.84.2.3 len	562
10.84.2.4 nsites	562
10.84.3 Friends And Related Function Documentation	563
10.84.3.1 diff	563
10.84.3.2 inter	563
10.84.3.3 operator<	563
10.84.3.4 operator<	563
10.84.3.5 operator<<	563
10.84.3.6 operator<=	564
10.84.3.7 operator<=	564
10.84.3.8 operator==	564

10.84.3.9 sym_diff	564
10.84.3.10 uni	564
10.84.3.11 unique	565
10.85mln::box_runend_piter< P > Class Template Reference	565
10.85.1 Detailed Description	565
10.85.2 Constructor & Destructor Documentation	565
10.85.2.1 box_runend_piter	565
10.85.3 Member Function Documentation	566
10.85.3.1 next	566
10.85.3.2 run_length	566
10.86mln::box_runstart_piter< P > Class Template Reference	566
10.86.1 Detailed Description	566
10.86.2 Constructor & Destructor Documentation	567
10.86.2.1 box_runstart_piter	567
10.86.3 Member Function Documentation	567
10.86.3.1 next	567
10.86.3.2 run_length	567
10.87mln::Browsing< E > Struct Template Reference	567
10.87.1 Detailed Description	568
10.88mln::canvas::browsing::backdiagonal2d_t Struct Reference	568
10.88.1 Detailed Description	569
10.89mln::canvas::browsing::breadth_first_search_t Struct Reference	570
10.89.1 Detailed Description	570
10.90mln::canvas::browsing::depth_first_search_t Struct Reference	570
10.90.1 Detailed Description	570
10.91mln::canvas::browsing::diagonal2d_t Struct Reference	570
10.91.1 Detailed Description	571
10.92mln::canvas::browsing::dir_struct_elt_incr_update_t Struct Reference	572
10.92.1 Detailed Description	572
10.93mln::canvas::browsing::directional_t Struct Reference	573
10.93.1 Detailed Description	574
10.94mln::canvas::browsing::fwd_t Struct Reference	575
10.94.1 Detailed Description	576
10.95mln::canvas::browsing::hyper_directional_t Struct Reference	576
10.95.1 Detailed Description	577
10.96mln::canvas::browsing::snake_fwd_t Struct Reference	578

10.96.1 Detailed Description	578
10.97mln::canvas::browsing::snake_generic_t Struct Reference	579
10.97.1 Detailed Description	580
10.98mln::canvas::browsing::snake_vert_t Struct Reference	581
10.98.1 Detailed Description	581
10.99mln::canvas::chamfer< F > Struct Template Reference	582
10.99.1 Detailed Description	582
10.100mln::category< R(*) (A) > Struct Template Reference	582
10.100.1 Detailed Description	582
10.101mln::complex_image< D, G, V > Class Template Reference	583
10.101.1 Detailed Description	584
10.101.2 Member Typedef Documentation	584
10.101.2.1 geom	584
10.101.2.2 value	584
10.101.2.3 value	584
10.101.2.4 skeleton	584
10.101.2.5 value	584
10.101.3 Constructor & Destructor Documentation	585
10.101.3.1 complex_image	585
10.101.4 Member Function Documentation	585
10.101.4.1 domain	585
10.101.4.2 operator()	585
10.101.4.3 operator()	585
10.101.4.4 values	585
10.101.5 Member Data Documentation	585
10.101.5.1 dim	585
10.102mln::complex_neighborhood_bkd_piter< I, G, N > Class Template Reference	586
10.102.1 Detailed Description	586
10.102.2 Member Typedef Documentation	586
10.102.2.1 liter_type	586
10.102.2.2 psite	587
10.102.3 Constructor & Destructor Documentation	587
10.102.3.1 complex_neighborhood_bkd_piter	587
10.102.4 Member Function Documentation	587
10.102.4.1 liter	587
10.102.4.2 next	587

10.103	mln::complex_neighborhood_fwd_piter< I, G, N > Class Template Reference	587
10.103.1	Detailed Description	588
10.103.2	Member Typedef Documentation	588
10.103.2.1	liter_type	588
10.103.2.2	psite	588
10.103.3	Constructor & Destructor Documentation	588
10.103.3.1	lcomplex_neighborhood_fwd_piter	588
10.103.4	Member Function Documentation	589
10.103.4.1	liter	589
10.103.4.2	next	589
10.104	mln::complex_psite< D, G > Class Template Reference	589
10.104.1	Detailed Description	590
10.104.2	Constructor & Destructor Documentation	590
10.104.2.1	lcomplex_psite	590
10.104.2.2	complex_psite	590
10.104.3	Member Function Documentation	590
10.104.3.1	lchange_target	590
10.104.3.2	face	591
10.104.3.3	face_id	591
10.104.3.4	invalidate	591
10.104.3.5	is_valid	591
10.104.3.6	mln	591
10.104.3.7	site_set	592
10.105	mln::complex_window_bkd_piter< I, G, W > Class Template Reference	592
10.105.1	Detailed Description	592
10.105.2	Member Typedef Documentation	593
10.105.2.1	liter_type	593
10.105.2.2	psite	593
10.105.3	Constructor & Destructor Documentation	593
10.105.3.1	lcomplex_window_bkd_piter	593
10.105.4	Member Function Documentation	593
10.105.4.1	liter	593
10.105.4.2	next	593
10.106	mln::complex_window_fwd_piter< I, G, W > Class Template Reference	594
10.106.1	Detailed Description	594
10.106.2	Member Typedef Documentation	594

10.106.2.liter_type	594
10.106.2.psite	595
10.106.3.Constructor & Destructor Documentation	595
10.106.3.1complex_window_fwd_piter	595
10.106.4.Member Function Documentation	595
10.106.4.liter	595
10.106.4.2next	595
10.107.mln::decorated_image< I, D > Struct Template Reference	595
10.107.Detailed Description	596
10.107.2.Member Typedef Documentation	596
10.107.2.1lvalue	596
10.107.2.2psite	597
10.107.2.3rvalue	597
10.107.2.4skeleton	597
10.107.3.Constructor & Destructor Documentation	597
10.107.3.1decorated_image	597
10.107.3.2~decorated_image	597
10.107.4.Member Function Documentation	597
10.107.4.1decoration	597
10.107.4.2decoration	597
10.107.4.3operator decorated_image< const I, D >	598
10.107.4.4operator()	598
10.107.4.5operator()	598
10.108.mln::Delta_Point_Site< E > Struct Template Reference	598
10.108.Detailed Description	599
10.109.mln::Delta_Point_Site< void > Struct Template Reference	599
10.109.Detailed Description	599
10.110.mln::doc::Accumulator< E > Struct Template Reference	599
10.110.Detailed Description	600
10.110.2.Member Typedef Documentation	600
10.110.2.largument	600
10.110.3.Member Function Documentation	600
10.110.3.linit	600
10.110.3.2ake	600
10.110.3.3ake	600
10.111.mln::doc::Box< E > Struct Template Reference	600

10.111. Detailed Description	602
10.111. Member Typedef Documentation	602
10.111.2.1 bkd_piter	602
10.111.2.2 fwd_piter	602
10.111.2.3 psite	602
10.111.2.4 site	602
10.111. Member Function Documentation	602
10.111.3.1 bbox	602
10.111.3.2 has	603
10.111.3.3 nsites	603
10.111.3.4 pmax	603
10.111.3.5 pmin	603
10.111. In::doc::Dpoint< E > Struct Template Reference	603
10.112. Detailed Description	604
10.112. Member Typedef Documentation	604
10.112.2.1 coord	604
10.112.2.2 dpoint	605
10.112.2.3 point	605
10.112. Member Enumeration Documentation	605
10.112.3.1 "@19	605
10.112. Member Function Documentation	605
10.112.4.1 operator[]	605
10.111. In::doc::Fastest_Image< E > Struct Template Reference	606
10.113. Detailed Description	608
10.113. Member Typedef Documentation	608
10.113.2.1 bkd_piter	608
10.113.2.2 coord	608
10.113.2.3 dpoint	608
10.113.2.4 fwd_piter	609
10.113.2.5 value	609
10.113.2.6 point	609
10.113.2.7 pset	609
10.113.2.8 psite	609
10.113.2.9 rvalue	609
10.113.2.10 skeleton	610
10.113.2.11 value	610

10.113.2.12set	610
10.113.3Member Function Documentation	610
10.113.3.1bbox	610
10.113.3.2border	610
10.113.3.3buffer	611
10.113.3.4delta_index	611
10.113.3.5domain	611
10.113.3.6has	611
10.113.3.7has	611
10.113.3.8s_valid	612
10.113.3.9elements	612
10.113.3.10sites	612
10.113.3.11operator()	612
10.113.3.12operator()	612
10.113.3.13operator[]	613
10.113.3.14operator[]	613
10.113.3.15point_at_index	613
10.113.3.16values	613
10.114Inn::doc::Generalized_Pixel< E > Struct Template Reference	614
10.114.1Detailed Description	614
10.114.2Member Typedef Documentation	615
10.114.2.1image	615
10.114.2.2value	615
10.114.2.3value	615
10.114.3Member Function Documentation	615
10.114.3.1ima	615
10.114.3.2val	615
10.115Inn::doc::Image< E > Struct Template Reference	615
10.115.1Detailed Description	617
10.115.2Member Typedef Documentation	618
10.115.2.1bkd_piter	618
10.115.2.2coord	618
10.115.2.3dpoint	618
10.115.2.4fwd_piter	618
10.115.2.5value	618
10.115.2.6point	618

10.115.2.7pset	619
10.115.2.8psite	619
10.115.2.9rvalue	619
10.115.2.10skeleton	619
10.115.2.11value	619
10.115.2.12set	619
10.115.3Member Function Documentation	620
10.115.3.1bbox	620
10.115.3.2domain	620
10.115.3.3has	620
10.115.3.4has	620
10.115.3.5is_valid	620
10.115.3.6sites	621
10.115.3.7operator()	621
10.115.3.8operator()	621
10.115.3.9values	621
10.116In::doc::Iterator< E > Struct Template Reference	621
10.116.Detailed Description	622
10.116.2Member Function Documentation	622
10.116.2.1invalidate	622
10.116.2.2is_valid	622
10.116.2.3start	623
10.117In::doc::Neighborhood< E > Struct Template Reference	623
10.117.Detailed Description	624
10.117.2Member Typedef Documentation	624
10.117.2.1bkd_niter	624
10.117.2.2dpoint	624
10.117.2.3fwd_niter	624
10.117.2.4niter	624
10.117.2.5point	624
10.118In::doc::Object< E > Struct Template Reference	624
10.118.Detailed Description	625
10.119In::doc::Pixel_Iterator< E > Struct Template Reference	625
10.119.Detailed Description	627
10.119.2Member Typedef Documentation	627
10.119.2.1image	627

10.119.2.2value	627
10.119.2.3value	627
10.119.2.4value	627
10.119.3Member Function Documentation	627
10.119.3.lima	627
10.119.3.2invalidate	628
10.119.3.3is_valid	628
10.119.3.4start	628
10.119.3.5val	628
10.120Inn::doc::Point_Site< E > Struct Template Reference	628
10.120.Detailed Description	629
10.120.2Member Typedef Documentation	629
10.120.2.1coord	629
10.120.2.2dpoint	629
10.120.2.3mesh	629
10.120.2.4point	630
10.120.3Member Enumeration Documentation	630
10.120.3.1"@20	630
10.120.4Member Function Documentation	630
10.120.4.1operator[]	630
10.120.4.2to_point	630
10.121Inn::doc::Site_Iterator< E > Struct Template Reference	631
10.121.Detailed Description	632
10.121.2Member Typedef Documentation	632
10.121.2.1psite	632
10.121.3Member Function Documentation	632
10.121.3.1invalidate	632
10.121.3.2is_valid	632
10.121.3.3operator psite	632
10.121.3.4start	632
10.122Inn::doc::Site_Set< E > Struct Template Reference	633
10.122.Detailed Description	634
10.122.2Member Typedef Documentation	634
10.122.2.1bkd_piter	634
10.122.2.2fwd_piter	634
10.122.2.3psite	634

10.122.2.4site	634
10.122.3Member Function Documentation	634
10.122.3.1has	634
10.122.3.2In::doc::Value_Iterator< E > Struct Template Reference	635
10.123.1Detailed Description	636
10.123.2Member Typedef Documentation	636
10.123.2.1value	636
10.123.3Member Function Documentation	636
10.123.3.1invalidate	636
10.123.3.2is_valid	636
10.123.3.3operator value	636
10.123.3.4start	636
10.122.3.4In::doc::Value_Set< E > Struct Template Reference	636
10.124.1Detailed Description	637
10.124.2Member Typedef Documentation	638
10.124.2.1bkd_viter	638
10.124.2.2fwd_viter	638
10.124.2.3value	638
10.124.3Member Function Documentation	638
10.124.3.1has	638
10.124.3.2index_of	638
10.124.3.3nvalues	638
10.124.3.4operator[]	639
10.122.3.5In::doc::Weighted_Window< E > Struct Template Reference	639
10.125.1Detailed Description	640
10.125.2Member Typedef Documentation	640
10.125.2.1bkd_qiter	640
10.125.2.2dpoint	640
10.125.2.3fwd_qiter	640
10.125.2.4point	641
10.125.2.5weight	641
10.125.2.6window	641
10.125.3Member Function Documentation	641
10.125.3.1delta	641
10.125.3.2is_centered	641
10.125.3.3is_empty	641

10.125.3.4sym	641
10.125.3.5win	641
10.126. <code>mln::doc::Window< E ></code> Struct Template Reference	641
10.126. Detailed Description	642
10.126. Member Typedef Documentation	642
10.126.2. <code>l_bkd_qiter</code>	642
10.126.2. <code>2fwd_qiter</code>	643
10.126.2. <code>3qiter</code>	643
10.127. <code>mln::Dpoint< E ></code> Struct Template Reference	643
10.127. Detailed Description	643
10.127. Member Function Documentation	644
10.127.2. <code>lto_dpoint</code>	644
10.128. <code>mln::dpoint< G, C ></code> Struct Template Reference	644
10.128. Detailed Description	646
10.128. Member Typedef Documentation	646
10.128.2. <code>lcoord</code>	646
10.128.2. <code>2grid</code>	646
10.128.2. <code>3psite</code>	646
10.128.2. <code>4site</code>	646
10.128.2. <code>5vec</code>	646
10.128. Member Enumeration Documentation	647
10.128.3. <code>l"@22</code>	647
10.128. Constructor & Destructor Documentation	647
10.128.4. <code>ldpoint</code>	647
10.128.4. <code>2dpoint</code>	647
10.128.4. <code>3dpoint</code>	647
10.128.4. <code>4dpoint</code>	647
10.128.4. <code>5dpoint</code>	647
10.128. Member Function Documentation	648
10.128.5. <code>loperator mln::algebra::vec< dpoint< G, C >::dim, Q ></code>	648
10.128.5. <code>2operator[]</code>	648
10.128.5. <code>3operator[]</code>	648
10.128.5. <code>4set_all</code>	648
10.128.5. <code>5to_vec</code>	649
10.129. <code>mln::dpoints_bkd_pixter< I ></code> Class Template Reference	649
10.129. Detailed Description	650

10.129. Constructor & Destructor Documentation	650
10.129.2. 1dpoints_bkd_pixter	650
10.129.2. 2dpoints_bkd_pixter	650
10.129. Member Function Documentation	650
10.129.3. 1center_val	650
10.129.3. 2invalidate	650
10.129.3. 3is_valid	651
10.129.3. 4next	651
10.129.3. 5start	651
10.129.3. 6update	651
10.130. In::dpoints_fwd_pixter< I > Class Template Reference	651
10.130. Detailed Description	652
10.130. Constructor & Destructor Documentation	652
10.130.2. 1dpoints_fwd_pixter	652
10.130.2. 2dpoints_fwd_pixter	653
10.130. Member Function Documentation	653
10.130.3. 1center_val	653
10.130.3. 2invalidate	653
10.130.3. 3is_valid	653
10.130.3. 4next	653
10.130.3. 5start	654
10.130.3. 6update	654
10.131. In::dpsites_bkd_piter< V > Class Template Reference	654
10.131. Detailed Description	654
10.131. Constructor & Destructor Documentation	655
10.131.2. 1dpsites_bkd_piter	655
10.131.2. 2dpsites_bkd_piter	655
10.131. Member Function Documentation	655
10.131.3. 1next	655
10.132. In::dpsites_fwd_piter< V > Class Template Reference	655
10.132. Detailed Description	656
10.132. Constructor & Destructor Documentation	656
10.132.2. 1dpsites_fwd_piter	656
10.132.2. 2dpsites_fwd_piter	656
10.132. Member Function Documentation	656
10.132.3. 1next	656

10.132	<code>mln::Edge< E ></code> Struct Template Reference	657
10.133	Detailed Description	657
10.134	<code>mln::edge_image< P, V, G ></code> Class Template Reference	657
10.134	Detailed Description	658
10.134	Member Typedef Documentation	658
10.134.2	<code>ledge_nbh_t</code>	658
10.134.2	<code>ledge_win_t</code>	658
10.134.2	<code>graph_t</code>	658
10.134.2	<code>nbh_t</code>	658
10.134.2	<code>site_function_t</code>	659
10.134.2	<code>skeleton</code>	659
10.134.2	<code>win_t</code>	659
10.134	Constructor & Destructor Documentation	659
10.134.3	<code>ledge_image</code>	659
10.134	Member Function Documentation	659
10.134.4	<code>operator()</code>	659
10.135	<code>mln::extended< I ></code> Struct Template Reference	659
10.135	Detailed Description	660
10.135	Member Typedef Documentation	660
10.135.2	<code>lskeleton</code>	660
10.135.2	<code>value</code>	660
10.135	Constructor & Destructor Documentation	660
10.135.3	<code>lextended</code>	660
10.135.3	<code>extended</code>	661
10.135	Member Function Documentation	661
10.135.4	<code>ldomain</code>	661
10.136	<code>mln::extension_fun< I, F ></code> Class Template Reference	661
10.136	Detailed Description	662
10.136	Member Typedef Documentation	662
10.136.2	<code>lvalue</code>	662
10.136.2	<code>skeleton</code>	662
10.136.2	<code>value</code>	662
10.136	Constructor & Destructor Documentation	662
10.136.3	<code>lextension_fun</code>	662
10.136.3	<code>extension_fun</code>	663
10.136	Member Function Documentation	663

10.136.4.1extension	663
10.136.4.2has	663
10.136.4.3operator()	663
10.136.4.4operator()	663
10.137.1In::extension_ima< I, J > Class Template Reference	663
10.137.1.Detailed Description	664
10.137.2Member Typedef Documentation	664
10.137.2.1rvalue	664
10.137.2.2skeleton	665
10.137.2.3value	665
10.137.3Constructor & Destructor Documentation	665
10.137.3.1extension_ima	665
10.137.3.2extension_ima	665
10.137.4Member Function Documentation	665
10.137.4.1extension	665
10.137.4.2has	665
10.137.4.3operator()	665
10.137.4.4operator()	666
10.138.1In::extension_val< I > Class Template Reference	666
10.138.1.Detailed Description	667
10.138.2Member Typedef Documentation	667
10.138.2.1rvalue	667
10.138.2.2skeleton	667
10.138.2.3value	667
10.138.3Constructor & Destructor Documentation	667
10.138.3.1extension_val	667
10.138.3.2extension_val	667
10.138.4Member Function Documentation	667
10.138.4.1change_extension	667
10.138.4.2extension	668
10.138.4.3has	668
10.138.4.4operator()	668
10.138.4.5operator()	668
10.139.1In::faces_psite< N, D, P > Class Template Reference	668
10.139.1.Detailed Description	669
10.139.1.Constructor & Destructor Documentation	669

10.139.2.1faces_psite	669
10.139.2.2faces_psite	669
10.139.3Member Function Documentation	670
10.139.3.1change_target	670
10.139.3.2face	670
10.139.3.3face_id	670
10.139.3.4invalidate	670
10.139.3.5is_valid	670
10.139.3.6n	670
10.139.3.7site_set	671
10.140mln::flat_image< T, S > Struct Template Reference	671
10.140.1Detailed Description	672
10.140.2Member Typedef Documentation	672
10.140.2.1lvalue	672
10.140.2.2rvalue	672
10.140.2.3skeleton	672
10.140.2.4value	672
10.140.3Constructor & Destructor Documentation	672
10.140.3.1flat_image	672
10.140.3.2flat_image	673
10.140.4Member Function Documentation	673
10.140.4.1domain	673
10.140.4.2has	673
10.140.4.3operator()	673
10.140.4.4operator()	673
10.141mln::fun::from_accu< A > Struct Template Reference	673
10.141.1Detailed Description	673
10.142mln::fun::n2v::white_gaussian< V > Struct Template Reference	674
10.142.1Detailed Description	674
10.143mln::fun::p2b::antilogy Struct Reference	674
10.143.1Detailed Description	675
10.144mln::fun::p2b::tautology Struct Reference	675
10.144.1Detailed Description	676
10.145mln::fun::v2b::lnot< V > Struct Template Reference	677
10.145.1Detailed Description	677
10.146mln::fun::v2b::threshold< V > Struct Template Reference	678

10.146. Detailed Description	678
10.147. <code>std::function::operator_value< F, V ></code> Class Template Reference	679
10.147. Detailed Description	679
10.148. <code>std::function::component< T, i ></code> Struct Template Reference	680
10.148. Detailed Description	680
10.149. <code>std::function::l1_norm< V, R ></code> Struct Template Reference	680
10.149. Detailed Description	681
10.150. <code>std::function::l2_norm< V, R ></code> Struct Template Reference	681
10.150. Detailed Description	682
10.151. <code>std::function::linear< V, T, R ></code> Struct Template Reference	682
10.151. Detailed Description	683
10.152. <code>std::function::linfty_norm< V, R ></code> Struct Template Reference	683
10.152. Detailed Description	684
10.153. <code>std::function::rgb8_to_rgbn< n ></code> Struct Template Reference	684
10.153. Detailed Description	685
10.153. Member Function Documentation	686
10.153.2. <code>operator()</code>	686
10.154. <code>std::function::cos< V ></code> Struct Template Reference	686
10.154. Detailed Description	687
10.155. <code>std::function::l1_norm< V, R ></code> Struct Template Reference	687
10.155. Detailed Description	688
10.156. <code>std::function::l2_norm< V, R ></code> Struct Template Reference	689
10.156. Detailed Description	689
10.157. <code>std::function::linfty_norm< V, R ></code> Struct Template Reference	690
10.157. Detailed Description	690
10.158. <code>std::function::eq< L, R ></code> Struct Template Reference	691
10.158. Detailed Description	691
10.159. <code>std::function::ge< L, R ></code> Struct Template Reference	691
10.159. Detailed Description	692
10.160. <code>std::function::gt< L, R ></code> Struct Template Reference	692
10.160. Detailed Description	693
10.161. <code>std::function::implies< L, R ></code> Struct Template Reference	693
10.161. Detailed Description	694
10.162. <code>std::function::le< L, R ></code> Struct Template Reference	694
10.162. Detailed Description	695
10.163. <code>std::function::lt< L, R ></code> Struct Template Reference	695

10.163. Detailed Description	696
10.164. <code>std::diff_abs< V ></code> Struct Template Reference	696
10.164. Detailed Description	697
10.165. <code>std::land< L, R ></code> Struct Template Reference	697
10.165. Detailed Description	698
10.166. <code>std::land_not< L, R ></code> Struct Template Reference	698
10.166. Detailed Description	699
10.167. <code>std::lor< L, R ></code> Struct Template Reference	699
10.167. Detailed Description	700
10.168. <code>std::lxor< L, R ></code> Struct Template Reference	700
10.168. Detailed Description	701
10.169. <code>std::max< V ></code> Struct Template Reference	701
10.169. Detailed Description	702
10.170. <code>std::min< L, R ></code> Struct Template Reference	702
10.170. Detailed Description	703
10.171. <code>std::vec< V ></code> Struct Template Reference	703
10.171. Detailed Description	704
10.172. <code>std::closest_point< P ></code> Struct Template Reference	704
10.172. Detailed Description	705
10.173. <code>std::bilinear< I ></code> Struct Template Reference	705
10.173. Detailed Description	705
10.173. Member Function Documentation	705
10.173.2. <code>operator()</code>	705
10.173.2. <code>operator()</code>	706
10.174. <code>std::trilinear< I ></code> Struct Template Reference	706
10.174. Detailed Description	706
10.175. <code>std::composed< T2, T1 ></code> Struct Template Reference	706
10.175. Detailed Description	706
10.175. Constructor & Destructor Documentation	707
10.175.2. <code>lcomposed</code>	707
10.175.2. <code>composed</code>	707
10.176. <code>std::linear< I ></code> Struct Template Reference	707
10.176. Detailed Description	708
10.176. Constructor & Destructor Documentation	708
10.176.2. <code>llinear</code>	708
10.176. Member Function Documentation	708

10.176.3. loperator()	708
10.176.4. Member Data Documentation	708
10.176.4. lima	708
10.177.1.1.1. <code>ln::fun::x2x::rotation< n, C ></code> Struct Template Reference	708
10.177.2. Detailed Description	710
10.177.3. Member Typedef Documentation	710
10.177.2.1. ldata_t	710
10.177.2.2. linvert	710
10.177.4. Constructor & Destructor Documentation	710
10.177.3.1. lrotation	710
10.177.3.2. rotation	711
10.177.3.3. rotation	711
10.177.3.4. rotation	711
10.177.5. Member Function Documentation	711
10.177.4.1. linv	711
10.177.4.2. loperator()	711
10.177.4.3. lset_alpha	711
10.177.4.4. lset_axis	711
10.178.1.1.1. <code>ln::fun::x2x::translation< n, C ></code> Struct Template Reference	712
10.178.2. Detailed Description	713
10.178.3. Member Typedef Documentation	713
10.178.2.1. ldata_t	713
10.178.2.2. linvert	713
10.178.4. Constructor & Destructor Documentation	713
10.178.3.1. ltranslation	713
10.178.3.2. translation	714
10.178.5. Member Function Documentation	714
10.178.4.1. linv	714
10.178.4.2. loperator()	714
10.178.4.3. lset_t	714
10.178.4.4. l	714
10.179.1.1.1. <code>ln::fun_image< F, I ></code> Struct Template Reference	714
10.179.2. Detailed Description	715
10.179.3. Member Typedef Documentation	715
10.179.2.1. llvalue	715
10.179.2.2. rvalve	715

10.179.2.3skeleton	716
10.179.2.4value	716
10.179.3.Constructor & Destructor Documentation	716
10.179.3.1fun_image	716
10.179.3.2fun_image	716
10.179.3.3fun_image	716
10.179.4.Member Function Documentation	716
10.179.4.1operator()	716
10.179.4.2operator()	716
10.180.mln::Function< E > Struct Template Reference	717
10.180.Detailed Description	717
10.180.Constructor & Destructor Documentation	717
10.180.2.lFunction	717
10.181.mln::Function< void > Struct Template Reference	717
10.181.Detailed Description	717
10.182.mln::Function_n2v< E > Struct Template Reference	718
10.182.Detailed Description	718
10.183.mln::Function_v2b< E > Struct Template Reference	718
10.183.Detailed Description	719
10.184.mln::Function_v2v< E > Struct Template Reference	719
10.184.Detailed Description	720
10.185.mln::Function_vv2b< E > Struct Template Reference	720
10.185.Detailed Description	720
10.186.mln::Function_vv2v< E > Struct Template Reference	720
10.186.Detailed Description	721
10.187.mln::fwd_pixter1d< I > Class Template Reference	721
10.187.Detailed Description	722
10.187.Member Typedef Documentation	722
10.187.2.limage	722
10.187.Constructor & Destructor Documentation	722
10.187.3.lfwd_pixter1d	722
10.187.Member Function Documentation	722
10.187.4.lnext	722
10.188.mln::fwd_pixter2d< I > Class Template Reference	722
10.188.Detailed Description	723
10.188.Member Typedef Documentation	723

10.188.2. limage	723
10.188. Constructor & Destructor Documentation	723
10.188.3. lfwd_pixter2d	723
10.188. Member Function Documentation	724
10.188.4. lnext	724
10.189. mln::fwd_pixter3d< I > Class Template Reference	724
10.189. Detailed Description	724
10.189. Member Typedef Documentation	724
10.189.2. limage	724
10.189. Constructor & Destructor Documentation	725
10.189.3. lfwd_pixter3d	725
10.189. Member Function Documentation	725
10.189.4. lnext	725
10.190. mln::Gdpoint< E > Struct Template Reference	725
10.190. Detailed Description	726
10.191. mln::Gdpoint< void > Struct Template Reference	726
10.191. Detailed Description	726
10.192. mln::Generalized_Pixel< E > Struct Template Reference	727
10.192. Detailed Description	727
10.193. mln::geom::complex_geometry< D, P > Class Template Reference	727
10.193. Detailed Description	728
10.193. Constructor & Destructor Documentation	728
10.193.2. lcomplex_geometry	728
10.193. Member Function Documentation	728
10.193.3. ladd_location	728
10.193.3.2. operator()	729
10.194. mln::Gpoint< E > Struct Template Reference	729
10.194. Detailed Description	730
10.194. Friends And Related Function Documentation	730
10.194.2. operator+	730
10.194.2.2. operator+=	731
10.194.2.3. operator-	731
10.194.2.4. operator-=	732
10.194.2.5. operator/	732
10.194.2.6. operator<<	732
10.194.2.7. operator==	733

10.195	<code>ln::Graph< E ></code> Struct Template Reference	733
10.195	Detailed Description	733
10.196	<code>ln::graph::attribute::card_t</code> Struct Reference	734
10.196	Detailed Description	734
10.196	Member Typedef Documentation	734
10.196.2	<code>lresult</code>	734
10.197	<code>ln::graph::attribute::representative_t</code> Struct Reference	734
10.197	Detailed Description	735
10.197	Member Typedef Documentation	735
10.197.2	<code>lresult</code>	735
10.198	<code>ln::graph_elt_mixed_neighborhood< G, S, S2 ></code> Struct Template Reference	735
10.198	Detailed Description	736
10.198	Member Typedef Documentation	736
10.198.2	<code>lbkd_niter</code>	736
10.198.2	<code>2fwd_niter</code>	736
10.198.2	<code>3niter</code>	736
10.199	<code>ln::graph_elt_mixed_window< G, S, S2 ></code> Class Template Reference	736
10.199	Detailed Description	738
10.199	Member Typedef Documentation	738
10.199.2	<code>lbkd_qiter</code>	738
10.199.2	<code>2center_t</code>	738
10.199.2	<code>3fwd_qiter</code>	739
10.199.2	<code>4graph_element</code>	739
10.199.2	<code>5psite</code>	739
10.199.2	<code>6qiter</code>	739
10.199.2	<code>7site</code>	739
10.199.2	<code>8target</code>	739
10.199	Member Function Documentation	739
10.199.3	<code>ldelta</code>	739
10.199.3	<code>2is_centered</code>	740
10.199.3	<code>3is_empty</code>	740
10.199.3	<code>4is_symmetric</code>	740
10.199.3	<code>5is_valid</code>	740
10.199.3	<code>6sym</code>	740
10.200	<code>ln::graph_elt_neighborhood< G, S ></code> Struct Template Reference	740
10.200	Detailed Description	741

10.200.2Member Typedef Documentation	742
10.200.2.1bkd_niter	742
10.200.2.2fwd_niter	742
10.200.2.3niter	742
10.201.1In::graph_elt_neighborhood_if< G, S, I > Struct Template Reference	742
10.201.1.1Detailed Description	743
10.201.1.2Member Typedef Documentation	743
10.201.1.2.1bkd_niter	743
10.201.1.2.2fwd_niter	743
10.201.1.2.3niter	744
10.201.1.3Constructor & Destructor Documentation	744
10.201.1.3.1graph_elt_neighborhood_if	744
10.201.1.3.2graph_elt_neighborhood_if	744
10.201.1.4Member Function Documentation	744
10.201.1.4.1mask	744
10.202.1In::graph_elt_window< G, S > Class Template Reference	744
10.202.1.1Detailed Description	746
10.202.1.2Member Typedef Documentation	746
10.202.1.2.1bkd_qiter	746
10.202.1.2.2center_t	746
10.202.1.2.3fwd_qiter	747
10.202.1.2.4graph_element	747
10.202.1.2.5psite	747
10.202.1.2.6qiter	747
10.202.1.2.7site	747
10.202.1.2.8target	747
10.202.1.3Member Function Documentation	747
10.202.1.3.1delta	747
10.202.1.3.2is_centered	748
10.202.1.3.3is_empty	748
10.202.1.3.4is_symmetric	748
10.202.1.3.5is_valid	748
10.202.1.3.6sym	748
10.203.1In::graph_elt_window_if< G, S, I > Class Template Reference	748
10.203.1.1Detailed Description	750
10.203.1.2Member Typedef Documentation	750

10.203.2.1	<code>bkd_qiter</code>	750
10.203.2.2	<code> fwd_qiter</code>	751
10.203.2.3	<code>mask_t</code>	751
10.203.2.4	<code>psite</code>	751
10.203.2.5	<code>qiter</code>	751
10.203.2.6	<code>site</code>	751
10.203.2.7	<code>target</code>	751
10.203.3	Constructor & Destructor Documentation	752
10.203.3.1	<code>graph_elt_window_if</code>	752
10.203.3.2	<code>graph_elt_window_if</code>	752
10.203.4	Member Function Documentation	752
10.203.4.1	<code>change_mask</code>	752
10.203.4.2	<code>delta</code>	752
10.203.4.3	<code>is_centered</code>	752
10.203.4.4	<code>is_empty</code>	752
10.203.4.5	<code>is_symmetric</code>	753
10.203.4.6	<code>is_valid</code>	753
10.203.4.7	<code>mask</code>	753
10.203.4.8	<code>sym</code>	753
10.204.1	<code>ln::graph_window_base< P, E ></code> Class Template Reference	753
10.204.1	Detailed Description	754
10.204.2	Member Typedef Documentation	754
10.204.2.1	<code>site</code>	754
10.204.3	Member Function Documentation	754
10.204.3.1	<code>delta</code>	754
10.204.3.2	<code>is_centered</code>	754
10.204.3.3	<code>is_empty</code>	755
10.204.3.4	<code>is_symmetric</code>	755
10.204.3.5	<code>is_valid</code>	755
10.204.3.6	<code>sym</code>	755
10.205.1	<code>ln::graph_window_if_piter< S, W, I ></code> Class Template Reference	755
10.205.1	Detailed Description	756
10.205.2	Member Typedef Documentation	756
10.205.2.1	<code>IP</code>	756
10.205.3	Constructor & Destructor Documentation	756
10.205.3.1	<code>graph_window_if_piter</code>	756

10.205.4Member Function Documentation	756
10.205.4.1element	756
10.205.4.2id	756
10.205.4.3next	757
10.206In::graph_window_piter< S, W, I > Class Template Reference	757
10.206.1Detailed Description	758
10.206.2Member Typedef Documentation	758
10.206.2.1center_t	758
10.206.2.2graph_element	758
10.206.2.3P	759
10.206.3Constructor & Destructor Documentation	759
10.206.3.1graph_window_piter	759
10.206.3.2graph_window_piter	759
10.206.3.3graph_window_piter	759
10.206.4Member Function Documentation	759
10.206.4.1change_target_site_set	759
10.206.4.2element	760
10.206.4.3id	760
10.206.4.4next	760
10.206.4.5target_site_set	760
10.207In::hexa< I > Struct Template Reference	760
10.207.1Detailed Description	762
10.207.2Member Typedef Documentation	762
10.207.2.1bkd_piter	762
10.207.2.2fwd_piter	762
10.207.2.3value	762
10.207.2.4psite	763
10.207.2.5value	763
10.207.2.6skeleton	763
10.207.2.7value	763
10.207.3Constructor & Destructor Documentation	763
10.207.3.1hexa	763
10.207.3.2hexa	763
10.207.4Member Function Documentation	763
10.207.4.1domain	763
10.207.4.2has	763

10.207.4.3operator()	764
10.207.4.4operator()	764
10.208. <code>nl::histo::array< T ></code> Struct Template Reference	764
10.208. Detailed Description	764
10.209. <code>nl::Image< E ></code> Struct Template Reference	764
10.209. Detailed Description	766
10.210. <code>nl::image1d< T ></code> Struct Template Reference	766
10.210. Detailed Description	767
10.210.2 Member Typedef Documentation	768
10.210.2.1 lvalue	768
10.210.2.2 rvalue	768
10.210.2.3 skeleton	768
10.210.2.4 value	768
10.210.3 Constructor & Destructor Documentation	768
10.210.3.1 image1d	768
10.210.3.2 image1d	768
10.210.3.3 image1d	768
10.210.4 Member Function Documentation	769
10.210.4.1 bbox	769
10.210.4.2 border	769
10.210.4.3 buffer	769
10.210.4.4 buffer	769
10.210.4.5 delta_index	769
10.210.4.6 domain	769
10.210.4.7 element	769
10.210.4.8 element	770
10.210.4.9 has	770
10.210.4.10 elements	770
10.210.4.11 finds	770
10.210.4.12 operator()	770
10.210.4.13 operator()	770
10.210.4.14 point_at_index	770
10.210.4.15 bbox	771
10.211. <code>nl::image2d< T ></code> Class Template Reference	771
10.211. Detailed Description	772
10.211.2 Member Typedef Documentation	773

10.211.2.1lvalue	773
10.211.2.2rvalue	773
10.211.2.3skeleton	773
10.211.2.4value	773
10.211.3Constructor & Destructor Documentation	773
10.211.3.1image2d	773
10.211.3.2image2d	773
10.211.3.3image2d	773
10.211.4Member Function Documentation	774
10.211.4.1bbox	774
10.211.4.2border	774
10.211.4.3buffer	774
10.211.4.4buffer	774
10.211.4.5delta_index	774
10.211.4.6domain	774
10.211.4.7element	774
10.211.4.8element	775
10.211.4.9has	775
10.211.4.10cols	775
10.211.4.11elements	775
10.211.4.12rows	775
10.211.4.13operator()	775
10.211.4.14operator()	775
10.211.4.15point_at_index	776
10.212In::image2d_h< V > Struct Template Reference	776
10.212.1Detailed Description	777
10.212.2Member Typedef Documentation	777
10.212.2.1bkd_piter	777
10.212.2.2fwd_piter	777
10.212.2.3value	778
10.212.2.4psite	778
10.212.2.5value	778
10.212.2.6skeleton	778
10.212.2.7value	778
10.212.3Constructor & Destructor Documentation	778
10.212.3.1image2d_h	778

10.212.4	Member Function Documentation	778
10.212.4.1	domain	778
10.212.4.2	has	779
10.212.4.3	operator()	779
10.212.4.4	operator()	779
10.213	nl::image3d< T > Struct Template Reference	779
10.213.1	Detailed Description	781
10.213.2	Member Typedef Documentation	781
10.213.2.1	lvalue	781
10.213.2.2	rvalue	781
10.213.2.3	skeleton	781
10.213.2.4	value	781
10.213.3	Constructor & Destructor Documentation	782
10.213.3.1	image3d	782
10.213.3.2	image3d	782
10.213.3.3	image3d	782
10.213.4	Member Function Documentation	782
10.213.4.1	lbbox	782
10.213.4.2	border	782
10.213.4.3	buffer	782
10.213.4.4	buffer	782
10.213.4.5	delta_index	783
10.213.4.6	domain	783
10.213.4.7	element	783
10.213.4.8	element	783
10.213.4.9	has	783
10.213.4.10	cols	783
10.213.4.11	elements	783
10.213.4.12	rows	784
10.213.4.13	slis	784
10.213.4.14	operator()	784
10.213.4.15	operator()	784
10.213.4.16	point_at_index	784
10.213.4.17	rbbox	784
10.214	nl::image_if< I, F > Struct Template Reference	784
10.214.1	Detailed Description	785

10.214.2	Member Typedef Documentation	785
10.214.2.1	skeleton	785
10.214.3	Constructor & Destructor Documentation	785
10.214.3.1	image_if	785
10.214.3.2	image_if	786
10.214.4	Member Function Documentation	786
10.214.4.1	domain	786
10.214.4.2	operator image_if< const I, F >	786
10.215	in::interpolated< I, F > Struct Template Reference	786
10.215.1	Detailed Description	787
10.215.2	Member Typedef Documentation	787
10.215.2.1	lvalue	787
10.215.2.2	psite	787
10.215.2.3	rvalue	787
10.215.2.4	skeleton	787
10.215.2.5	value	788
10.215.3	Constructor & Destructor Documentation	788
10.215.3.1	interpolated	788
10.215.4	Member Function Documentation	788
10.215.4.1	lhas	788
10.215.4.2	is_valid	788
10.216	in::io::dicom::dicom_header Struct Reference	788
10.216.1	Detailed Description	788
10.217	in::io::dump::dump_header Struct Reference	788
10.217.1	Detailed Description	789
10.218	in::io::fld::fld_header Struct Reference	789
10.218.1	Detailed Description	789
10.219	in::io::raw::raw_header Struct Reference	789
10.219.1	Detailed Description	789
10.220	in::Iterator< E > Struct Template Reference	789
10.220.1	Detailed Description	790
10.220.2	Member Function Documentation	791
10.220.2.1	next	791
10.221	in::labeled_image< I > Class Template Reference	791
10.221.1	Detailed Description	793
10.221.2	Member Typedef Documentation	793

10.221.2.lbbox_t	793
10.221.2.2skeleton	793
10.221.3.Constructor & Destructor Documentation	793
10.221.3.labeled_image	793
10.221.3.2labeled_image	793
10.221.3.3labeled_image	793
10.221.4.Member Function Documentation	794
10.221.4.lbbox	794
10.221.4.2bbboxes	794
10.221.4.3nlabels	794
10.221.4.4relabel	794
10.221.4.5relabel	794
10.221.4.6subdomain	794
10.221.4.7update_data	794
10.222.mn::labeled_image_base< I, E > Class Template Reference	795
10.222.Detailed Description	796
10.222.2.Member Typedef Documentation	796
10.222.2.lbbox_t	796
10.222.3.Constructor & Destructor Documentation	796
10.222.3.labeled_image_base	796
10.222.4.Member Function Documentation	797
10.222.4.lbbox	797
10.222.4.2bbboxes	797
10.222.4.3nlabels	797
10.222.4.4relabel	797
10.222.4.5relabel	797
10.222.4.6subdomain	798
10.222.4.7update_data	798
10.223.mn::lazy_image< I, F, B > Struct Template Reference	798
10.223.Detailed Description	799
10.223.2.Member Typedef Documentation	799
10.223.2.lvalue	799
10.223.2.2rvalue	799
10.223.2.3skeleton	799
10.223.3.Constructor & Destructor Documentation	800
10.223.3.lazy_image	800

10.223.3.2lazy_image	800
10.223.4Member Function Documentation	800
10.223.4.1domain	800
10.223.4.2has	800
10.223.4.3operator()	800
10.223.4.4operator()	800
10.223.4.5operator()	800
10.223.4.6operator()	801
10.224mln::Literal< E > Struct Template Reference	801
10.224. Detailed Description	803
10.225mln::literal::black_t Struct Reference	803
10.225. Detailed Description	803
10.226mln::literal::blue_t Struct Reference	804
10.226. Detailed Description	804
10.227mln::literal::brown_t Struct Reference	804
10.227. Detailed Description	805
10.228mln::literal::cyan_t Struct Reference	805
10.228. Detailed Description	806
10.229mln::literal::green_t Struct Reference	806
10.229. Detailed Description	807
10.230mln::literal::identity_t Struct Reference	807
10.230. Detailed Description	808
10.231mln::literal::light_gray_t Struct Reference	808
10.231. Detailed Description	809
10.232mln::literal::lime_t Struct Reference	809
10.232. Detailed Description	810
10.233mln::literal::magenta_t Struct Reference	810
10.233. Detailed Description	811
10.234mln::literal::max_t Struct Reference	811
10.234. Detailed Description	812
10.235mln::literal::min_t Struct Reference	812
10.235. Detailed Description	813
10.236mln::literal::olive_t Struct Reference	813
10.236. Detailed Description	814
10.237mln::literal::one_t Struct Reference	814
10.237. Detailed Description	815

10.238	<code>ln::literal::orange_t</code> Struct Reference	815
	10.238. Detailed Description	816
10.239	<code>ln::literal::origin_t</code> Struct Reference	816
	10.239. Detailed Description	817
10.240	<code>ln::literal::pink_t</code> Struct Reference	817
	10.240. Detailed Description	818
10.241	<code>ln::literal::purple_t</code> Struct Reference	818
	10.241. Detailed Description	819
10.242	<code>ln::literal::red_t</code> Struct Reference	819
	10.242. Detailed Description	820
10.243	<code>ln::literal::teal_t</code> Struct Reference	820
	10.243. Detailed Description	821
10.244	<code>ln::literal::violet_t</code> Struct Reference	821
	10.244. Detailed Description	822
10.245	<code>ln::literal::white_t</code> Struct Reference	822
	10.245. Detailed Description	823
10.246	<code>ln::literal::yellow_t</code> Struct Reference	823
	10.246. Detailed Description	824
10.247	<code>ln::literal::zero_t</code> Struct Reference	824
	10.247. Detailed Description	825
10.248	<code>ln::Mesh< E ></code> Struct Template Reference	825
	10.248. Detailed Description	826
10.249	<code>ln::Meta_Accumulator< E ></code> Struct Template Reference	826
	10.249. Detailed Description	827
10.250	<code>ln::Meta_Function< E ></code> Struct Template Reference	828
	10.250. Detailed Description	828
10.251	<code>ln::Meta_Function_v2v< E ></code> Struct Template Reference	829
	10.251. Detailed Description	829
10.252	<code>ln::Meta_Function_vv2v< E ></code> Struct Template Reference	829
	10.252. Detailed Description	830
10.253	<code>ln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 ></code> Struct Template Reference	830
	10.253. Detailed Description	830
10.254	<code>ln::metal::converts_to< T, U ></code> Struct Template Reference	830
	10.254. Detailed Description	830
10.255	<code>ln::metal::equal< T1, T2 ></code> Struct Template Reference	830
	10.255. Detailed Description	831

10.256	<code>ln::metal::goes_to< T, U ></code> Struct Template Reference	831
10.256	Detailed Description	831
10.257	<code>ln::metal::is< T, U ></code> Struct Template Reference	831
10.257	Detailed Description	831
10.258	<code>ln::metal::is_a< T, M ></code> Struct Template Reference	832
10.258	Detailed Description	832
10.259	<code>ln::metal::is_not< T, U ></code> Struct Template Reference	832
10.259	Detailed Description	832
10.260	<code>ln::metal::is_not_a< T, M ></code> Struct Template Reference	832
10.260	Detailed Description	832
10.261	<code>ln::mixed_neighb< W ></code> Class Template Reference	832
10.261	Detailed Description	833
10.261	Member Typedef Documentation	833
10.261.2	<code>l_bkd_niter</code>	833
10.261.2	<code>2fwd_niter</code>	833
10.261.2	<code>3niter</code>	833
10.261	Constructor & Destructor Documentation	834
10.261.3	<code>lmixed_neighb</code>	834
10.261.3	<code>2mixed_neighb</code>	834
10.262	<code>ln::morpho::attribute::card< I ></code> Class Template Reference	834
10.262	Detailed Description	834
10.262	Member Function Documentation	835
10.262.2	<code>linit</code>	835
10.262.2	<code>2is_valid</code>	835
10.262.2	<code>3take_as_init</code>	835
10.262.2	<code>4take_n_times</code>	835
10.262.2	<code>5to_result</code>	835
10.263	<code>ln::morpho::attribute::count_adjacent_vertices< I ></code> Struct Template Reference	835
10.263	Detailed Description	836
10.263	Member Function Documentation	836
10.263.2	<code>linit</code>	836
10.263.2	<code>2is_valid</code>	836
10.263.2	<code>3take_as_init</code>	836
10.263.2	<code>4take_n_times</code>	836
10.263.2	<code>5to_result</code>	837
10.264	<code>ln::morpho::attribute::height< I ></code> Struct Template Reference	837

10.264. Detailed Description	837
10.264. Member Function Documentation	838
10.264.2. lbase_level	838
10.264.2. 2init	838
10.264.2. 3is_valid	838
10.264.2. 4take_as_init	838
10.264.2. 5take_n_times	838
10.264.2. 6to_result	838
10.265. In::morpho::attribute::sharpness< I > Struct Template Reference	838
10.265. Detailed Description	839
10.265. Member Function Documentation	839
10.265.2. larea	839
10.265.2. 2height	840
10.265.2. 3init	840
10.265.2. 4is_valid	840
10.265.2. 5take_as_init	840
10.265.2. 6take_n_times	840
10.265.2. 7to_result	840
10.265.2. 8volume	840
10.266. In::morpho::attribute::sum< I, S > Class Template Reference	841
10.266. Detailed Description	841
10.266. Member Function Documentation	841
10.266.2. linit	841
10.266.2. 2is_valid	842
10.266.2. 3set_value	842
10.266.2. 4take_as_init	842
10.266.2. 5take_n_times	842
10.266.2. 6to_result	842
10.266.2. 7untake	842
10.267. In::morpho::attribute::volume< I > Struct Template Reference	842
10.267. Detailed Description	843
10.267. Member Function Documentation	843
10.267.2. larea	843
10.267.2. 2init	843
10.267.2. 3is_valid	843
10.267.2. 4take_as_init	844

10.267.2.5take_n_times	844
10.267.2.6to_result	844
10.268.1mln::neighb< W > Class Template Reference	844
10.268.1.Detailed Description	845
10.268.2.Member Typedef Documentation	845
10.268.2.1bkd_niter	845
10.268.2.2fwd_niter	845
10.268.2.3niter	845
10.268.3.Constructor & Destructor Documentation	845
10.268.3.1neighb	845
10.268.3.2neighb	845
10.269.1mln::Neighborhood< E > Struct Template Reference	846
10.269.1.Detailed Description	846
10.270.1mln::Neighborhood< void > Struct Template Reference	846
10.270.1.Detailed Description	846
10.271.1mln::Object< E > Struct Template Reference	846
10.271.1.Detailed Description	847
10.272.1mln::p2p_image< I, F > Struct Template Reference	847
10.272.1.Detailed Description	848
10.272.2.Member Typedef Documentation	848
10.272.2.1skeleton	848
10.272.3.Constructor & Destructor Documentation	848
10.272.3.1p2p_image	848
10.272.3.2p2p_image	848
10.272.4.Member Function Documentation	848
10.272.4.1domain	848
10.272.4.2fun	848
10.272.4.3operator()	849
10.272.4.4operator()	849
10.273.1mln::p_array< P > Class Template Reference	849
10.273.1.Detailed Description	851
10.273.2.Member Typedef Documentation	851
10.273.2.1bkd_piter	851
10.273.2.2element	851
10.273.2.3fwd_piter	851
10.273.2.4element	851

10.273.2.5	piter	851
10.273.2.6	psite	851
10.273.3	Constructor & Destructor Documentation	852
10.273.3.1	p_array	852
10.273.3.2	p_array	852
10.273.4	Member Function Documentation	852
10.273.4.1	append	852
10.273.4.2	append	852
10.273.4.3	change	852
10.273.4.4	clear	852
10.273.4.5	has	852
10.273.4.6	has	853
10.273.4.7	insert	853
10.273.4.8	s_valid	853
10.273.4.9	memory_size	853
10.273.4.10	sites	853
10.273.4.11	operator[]	853
10.273.4.12	operator[]	854
10.273.4.13	operator[]	854
10.273.4.14	reserve	854
10.273.4.15	size	854
10.273.4.16	id_vector	854
10.274	ln::p_centered< W > Class Template Reference	854
10.274.1	Detailed Description	855
10.274.2	Member Typedef Documentation	856
10.274.2.1	bkd_piter	856
10.274.2.2	element	856
10.274.2.3	fwd_piter	856
10.274.2.4	piter	856
10.274.2.5	psite	856
10.274.2.6	site	856
10.274.3	Constructor & Destructor Documentation	856
10.274.3.1	p_centered	856
10.274.3.2	p_centered	856
10.274.4	Member Function Documentation	857
10.274.4.1	lcenter	857

10.274.4.2has	857
10.274.4.3is_valid	857
10.274.4.4memory_size	857
10.274.4.5window	857
10.275. <code>std::p_complex< D, G ></code> Class Template Reference	857
10.275.1. Detailed Description	859
10.275.2. Member Typedef Documentation	859
10.275.2.1bkd_piter	859
10.275.2.2element	859
10.275.2.3fwd_piter	859
10.275.2.4piter	859
10.275.2.5psite	859
10.275.3. Constructor & Destructor Documentation	860
10.275.3.1p_complex	860
10.275.4. Member Function Documentation	860
10.275.4.1cplx	860
10.275.4.2cplx	860
10.275.4.3geom	860
10.275.4.4has	860
10.275.4.5is_valid	861
10.275.4.6faces	861
10.275.4.7nfaces_of_dim	861
10.275.4.8nsites	861
10.276. <code>std::p_edges< G, F ></code> Class Template Reference	861
10.276.1. Detailed Description	863
10.276.2. Member Typedef Documentation	863
10.276.2.1bkd_piter	863
10.276.2.2edge	863
10.276.2.3element	863
10.276.2.4fun_t	864
10.276.2.5fwd_piter	864
10.276.2.6graph_element	864
10.276.2.7graph_t	864
10.276.2.8piter	864
10.276.2.9psite	864
10.276.3. Constructor & Destructor Documentation	865

10.276.3.1p_edges	865
10.276.3.2p_edges	865
10.276.3.3p_edges	865
10.276.3.4p_edges	865
10.276.4Member Function Documentation	866
10.276.4.1function	866
10.276.4.2graph	866
10.276.4.3has	866
10.276.4.4has	866
10.276.4.5invalidate	866
10.276.4.6is_valid	866
10.276.4.7memory_size	867
10.276.4.8nedges	867
10.276.4.9nsites	867
10.277In::p_faces< N, D, P > Struct Template Reference	867
10.277.1Detailed Description	868
10.277.2Member Typedef Documentation	868
10.277.2.1bkd_piter	868
10.277.2.2element	868
10.277.2.3fwd_piter	869
10.277.2.4piter	869
10.277.2.5psite	869
10.277.3Constructor & Destructor Documentation	869
10.277.3.1p_faces	869
10.277.3.2p_faces	869
10.277.4Member Function Documentation	869
10.277.4.1cplx	869
10.277.4.2cplx	870
10.277.4.3is_valid	870
10.277.4.4nfaces	870
10.277.4.5nsites	870
10.278In::p_graph_piter< S, I > Class Template Reference	870
10.278.1Detailed Description	871
10.278.2Constructor & Destructor Documentation	871
10.278.2.1p_graph_piter	871
10.278.3Member Function Documentation	871

10.278.3.1graph	871
10.278.3.2d	871
10.278.3.3mln_q_subject	872
10.278.3.4next	872
10.279.1mln::p_if< S, F > Class Template Reference	872
10.279.1.Detailed Description	873
10.279.2.Member Typedef Documentation	873
10.279.2.1bkd_piter	873
10.279.2.2element	873
10.279.2.3fwd_piter	873
10.279.2.4piter	874
10.279.2.5psite	874
10.279.3.Constructor & Destructor Documentation	874
10.279.3.1p_if	874
10.279.3.2p_if	874
10.279.4.Member Function Documentation	874
10.279.4.1has	874
10.279.4.2is_valid	874
10.279.4.3memory_size	874
10.279.4.4overset	875
10.279.4.5pred	875
10.279.4.6predicate	875
10.280.1mln::p_image< I > Class Template Reference	875
10.280.1.Detailed Description	876
10.280.2.Member Typedef Documentation	877
10.280.2.1bkd_piter	877
10.280.2.2element	877
10.280.2.3fwd_piter	877
10.280.2.4_element	877
10.280.2.5piter	877
10.280.2.6psite	877
10.280.2.7_element	877
10.280.2.8S	877
10.280.3.Constructor & Destructor Documentation	878
10.280.3.1p_image	878
10.280.3.2p_image	878

10.280.4	Member Function Documentation	878
10.280.4.1	clear	878
10.280.4.2	has	878
10.280.4.3	insert	878
10.280.4.4	is_valid	878
10.280.4.5	memory_size	879
10.280.4.6	nsites	879
10.280.4.7	operator typename internal::p_image_site_set< I >::ret	879
10.280.4.8	remove	879
10.280.4.9	toggle	879
10.281	mln::p_indexed_bkd_piter< S > Class Template Reference	879
10.281.1	Detailed Description	880
10.281.2	Constructor & Destructor Documentation	880
10.281.2.1	lp_indexed_bkd_piter	880
10.281.2.2	2p_indexed_bkd_piter	880
10.281.3	Member Function Documentation	880
10.281.3.1	index	880
10.281.3.2	next	880
10.282	mln::p_indexed_fwd_piter< S > Class Template Reference	881
10.282.1	Detailed Description	881
10.282.2	Constructor & Destructor Documentation	881
10.282.2.1	lp_indexed_fwd_piter	881
10.282.2.2	2p_indexed_fwd_piter	881
10.282.3	Member Function Documentation	882
10.282.3.1	index	882
10.282.3.2	next	882
10.283	mln::p_indexed_psite< S > Class Template Reference	882
10.283.1	Detailed Description	882
10.284	mln::p_key< K, P > Class Template Reference	882
10.284.1	Detailed Description	884
10.284.2	Member Typedef Documentation	884
10.284.2.1	bkd_piter	884
10.284.2.2	element	884
10.284.2.3	fwd_piter	885
10.284.2.4	element	885
10.284.2.5	piter	885

10.284.2.6	psite	885
10.284.2.7	r_element	885
10.284.3	Constructor & Destructor Documentation	885
10.284.3.1	p_key	885
10.284.4	Member Function Documentation	885
10.284.4.1	change_key	885
10.284.4.2	change_keys	886
10.284.4.3	clear	886
10.284.4.4	exists_key	886
10.284.4.5	has	886
10.284.4.6	has	886
10.284.4.7	insert	886
10.284.4.8	insert	886
10.284.4.9	s_valid	887
10.284.4.10	key	887
10.284.4.11	keys	887
10.284.4.12	memory_size	887
10.284.4.13	sites	887
10.284.4.14	operator()	887
10.284.4.15	move	887
10.284.4.16	move_key	888
10.285	mln::p_line2d Class Reference	888
10.285.1	Detailed Description	889
10.285.2	Member Typedef Documentation	889
10.285.2.1	bkd_piter	889
10.285.2.2	element	889
10.285.2.3	fwd_piter	889
10.285.2.4	piter	890
10.285.2.5	psite	890
10.285.2.6	q_box	890
10.285.3	Constructor & Destructor Documentation	890
10.285.3.1	p_line2d	890
10.285.3.2	p_line2d	890
10.285.4	Member Function Documentation	890
10.285.4.1	bbox	890
10.285.4.2	begin	890

10.285.4.3end	891
10.285.4.4has	891
10.285.4.5has	891
10.285.4.6is_valid	891
10.285.4.7memory_size	891
10.285.4.8sites	891
10.285.4.9operator[]	891
10.285.4.10id_vector	892
10.286 mln::p_mutable_array_of< S > Class Template Reference	892
10.286.1.Detailed Description	893
10.286.2.Member Typedef Documentation	893
10.286.2.1bkd_piter	893
10.286.2.2element	893
10.286.2.3fwd_piter	893
10.286.2.4_element	894
10.286.2.5piter	894
10.286.2.6psite	894
10.286.3.Constructor & Destructor Documentation	894
10.286.3.1p_mutable_array_of	894
10.286.4.Member Function Documentation	894
10.286.4.1clear	894
10.286.4.2has	894
10.286.4.3insert	894
10.286.4.4is_valid	895
10.286.4.5memory_size	895
10.286.4.6elements	895
10.286.4.7operator[]	895
10.286.4.8operator[]	895
10.286.4.9reserve	895
10.287 mln::p_n_faces_bkd_piter< D, G > Class Template Reference	895
10.287.1.Detailed Description	896
10.287.2.Constructor & Destructor Documentation	896
10.287.2.1p_n_faces_bkd_piter	896
10.287.3.Member Function Documentation	896
10.287.3.1in	896
10.288 mln::p_n_faces_fwd_piter< D, G > Class Template Reference	896

10.288. Detailed Description	897
10.288. Constructor & Destructor Documentation	897
10.288.2. lp_n_faces_fwd_piter	897
10.288. Member Function Documentation	897
10.288.3. ln	897
10.288.3.2next	897
10.288. ln::p_priority< P, Q > Class Template Reference	898
10.289. Detailed Description	899
10.289. Member Typedef Documentation	899
10.289.2. lbkd_piter	899
10.289.2.2element	900
10.289.2.3fwd_piter	900
10.289.2.4element	900
10.289.2.5piter	900
10.289.2.6psite	900
10.289. Constructor & Destructor Documentation	900
10.289.3. lp_priority	900
10.289. Member Function Documentation	900
10.289.4. lclear	900
10.289.4.2exists_priority	901
10.289.4.3front	901
10.289.4.4has	901
10.289.4.5highest_priority	901
10.289.4.6insert	901
10.289.4.7insert	902
10.289.4.8s_valid	902
10.289.4.9lowest_priority	902
10.289.4.10memory_size	902
10.289.4.11sites	902
10.289.4.12operator()	902
10.289.4.13pop	903
10.289.4.14pop_front	903
10.289.4.15priorities	903
10.289.4.16push	903
10.290. ln::p_queue< P > Class Template Reference	903
10.290. Detailed Description	905

10.290.2	Member Typedef Documentation	905
10.290.2.1	l b k d _p i t e r	905
10.290.2.2	e l e m e n t	905
10.290.2.3	f w d_ p i t e r	905
10.290.2.4	e l e m e n t	905
10.290.2.5	p i t e r	906
10.290.2.6	p s i t e	906
10.290.3	Constructor & Destructor Documentation	906
10.290.3.1	l p _q u e u e	906
10.290.4	Member Function Documentation	906
10.290.4.1	l c l e a r	906
10.290.4.2	f r o n t	906
10.290.4.3	h a s	906
10.290.4.4	h a s	906
10.290.4.5	i n s e r t	907
10.290.4.6	i s _v a l i d	907
10.290.4.7	m e m o r y _s i z e	907
10.290.4.8	s i t e s	907
10.290.4.9	o p e r a t o r []	907
10.290.4.10	p o p	907
10.290.4.11	p o p_ f r o n t	907
10.290.4.12	p u s h	908
10.290.4.13	s t d_ d e q u e	908
10.291	mln::p_queue_fast< P > Class Template Reference	908
10.291.1	Detailed Description	910
10.291.2	Member Typedef Documentation	910
10.291.2.1	l b k d _p i t e r	910
10.291.2.2	e l e m e n t	910
10.291.2.3	f w d_ p i t e r	910
10.291.2.4	e l e m e n t	910
10.291.2.5	p i t e r	910
10.291.2.6	p s i t e	911
10.291.3	Constructor & Destructor Documentation	911
10.291.3.1	l p _q u e u e_ f a s t	911
10.291.4	Member Function Documentation	911
10.291.4.1	l c l e a r	911

10.291.4.2	compute_has	911
10.291.4.3	empty	911
10.291.4.4	front	911
10.291.4.5	has	911
10.291.4.6	has	912
10.291.4.7	insert	912
10.291.4.8	is_valid	912
10.291.4.9	memory_size	912
10.291.4.10	sites	912
10.291.4.11	operator[]	912
10.291.4.12	pop	912
10.291.4.13	pop_front	913
10.291.4.14	push	913
10.291.4.15	push	913
10.291.4.16	reserve	913
10.291.4.17	std_vector	913
10.292	mln::p_run< P > Class Template Reference	913
10.292.1	Detailed Description	915
10.292.2	Member Typedef Documentation	915
10.292.2.1	bkd_piter	915
10.292.2.2	element	915
10.292.2.3	fwd_piter	915
10.292.2.4	piter	915
10.292.2.5	psite	916
10.292.2.6	q_box	916
10.292.3	Constructor & Destructor Documentation	916
10.292.3.1	lp_run	916
10.292.3.2	p_run	916
10.292.3.3	pp_run	916
10.292.4	Member Function Documentation	916
10.292.4.1	lbbox	916
10.292.4.2	end	916
10.292.4.3	has	917
10.292.4.4	has	917
10.292.4.5	has_index	917
10.292.4.6	nit	917

10.292.4.7	is_valid	917
10.292.4.8	length	917
10.292.4.9	memory_size	918
10.292.4.10	sites	918
10.292.4.11	operator[]	918
10.292.4.12	start	918
10.293	inline::p_set< P > Class Template Reference	918
10.293.1	Detailed Description	920
10.293.2	Member Typedef Documentation	920
10.293.2.1	bkd_piter	920
10.293.2.2	element	920
10.293.2.3	fwd_piter	920
10.293.2.4	element	920
10.293.2.5	piter	920
10.293.2.6	psite	920
10.293.2.7	element	920
10.293.3	Constructor & Destructor Documentation	921
10.293.3.1	p_set	921
10.293.4	Member Function Documentation	921
10.293.4.1	clear	921
10.293.4.2	has	921
10.293.4.3	has	921
10.293.4.4	has	921
10.293.4.5	insert	921
10.293.4.6	is_valid	921
10.293.4.7	memory_size	922
10.293.4.8	sites	922
10.293.4.9	operator[]	922
10.293.4.10	move	922
10.293.4.11	std_vector	922
10.293.4.12	til_set	922
10.294	inline::p_set_of< S > Class Template Reference	922
10.294.1	Detailed Description	924
10.294.2	Member Typedef Documentation	924
10.294.2.1	bkd_piter	924
10.294.2.2	element	924

10.294.2.3	10.294.2.3	10.294.2.3	924
10.294.2.4	10.294.2.4	10.294.2.4	924
10.294.2.5	10.294.2.5	10.294.2.5	924
10.294.2.6	10.294.2.6	10.294.2.6	924
10.294.3	10.294.3	10.294.3	925
10.294.3.1	10.294.3.1	10.294.3.1	925
10.294.4	10.294.4	10.294.4	925
10.294.4.1	10.294.4.1	10.294.4.1	925
10.294.4.2	10.294.4.2	10.294.4.2	925
10.294.4.3	10.294.4.3	10.294.4.3	925
10.294.4.4	10.294.4.4	10.294.4.4	925
10.294.4.5	10.294.4.5	10.294.4.5	925
10.294.4.6	10.294.4.6	10.294.4.6	925
10.294.4.7	10.294.4.7	10.294.4.7	926
10.295	10.295	10.295	926
10.295.1	10.295.1	10.295.1	926
10.295.2	10.295.2	10.295.2	927
10.295.3	10.295.3	10.295.3	928
10.295.4	10.295.4	10.295.4	928
10.295.5	10.295.5	10.295.5	928
10.296	10.296	10.296	929
10.296.1	10.296.1	10.296.1	929
10.296.2	10.296.2	10.296.2	929

10.296.3	Member Function Documentation	930
10.296.3.1	change_target	930
10.296.3.2	next	930
10.297	nl::p_vaccess< V, S > Class Template Reference	930
10.297.1	Detailed Description	932
10.297.2	Member Typedef Documentation	932
10.297.2.1	bkd_piter	932
10.297.2.2	element	932
10.297.2.3	fwd_piter	932
10.297.2.4	element	932
10.297.2.5	piter	932
10.297.2.6	set	932
10.297.2.7	psite	933
10.297.2.8	value	933
10.297.2.9	vset	933
10.297.3	Constructor & Destructor Documentation	933
10.297.3.1	p_vaccess	933
10.297.4	Member Function Documentation	933
10.297.4.1	has	933
10.297.4.2	has	933
10.297.4.3	insert	933
10.297.4.4	insert	934
10.297.4.5	is_valid	934
10.297.4.6	memory_size	934
10.297.4.7	operator()	934
10.297.4.8	values	934
10.298	nl::p_vertices< G, F > Class Template Reference	934
10.298.1	Detailed Description	936
10.298.2	Member Typedef Documentation	936
10.298.2.1	bkd_piter	936
10.298.2.2	element	936
10.298.2.3	fun_t	937
10.298.2.4	fwd_piter	937
10.298.2.5	graph_element	937
10.298.2.6	graph_t	937
10.298.2.7	piter	937

10.298.2.8psite	937
10.298.2.9vertex	937
10.298.3.Constructor & Destructor Documentation	938
10.298.3.1p_vertices	938
10.298.3.2p_vertices	938
10.298.3.3p_vertices	938
10.298.3.4p_vertices	938
10.298.3.5p_vertices	939
10.298.4.Member Function Documentation	939
10.298.4.1function	939
10.298.4.2graph	939
10.298.4.3has	939
10.298.4.4has	939
10.298.4.5invalidate	940
10.298.4.6is_valid	940
10.298.4.7memory_size	940
10.298.4.8nsites	940
10.298.4.9nvertices	940
10.298.4.10operator()	940
10.299.mn::pixel< I > Struct Template Reference	941
10.299.1.Detailed Description	941
10.299.2.Constructor & Destructor Documentation	942
10.299.2.1pixel	942
10.299.2.2pixel	942
10.299.3.Member Function Documentation	942
10.299.3.1change_to	942
10.299.3.2is_valid	942
10.300.mn::Pixel_Iterator< E > Struct Template Reference	942
10.300.1.Detailed Description	943
10.300.2.Member Function Documentation	943
10.300.2.1next	943
10.301.mn::plain< I > Class Template Reference	943
10.301.1.Detailed Description	944
10.301.2.Member Typedef Documentation	944
10.301.2.1skeleton	944
10.301.3.Constructor & Destructor Documentation	944

10.301.3.1plain	944
10.301.3.2plain	944
10.301.3.3plain	945
10.301.4Member Function Documentation	945
10.301.4.1operator I	945
10.301.4.2operator=	945
10.301.4.3operator=	945
10.302mln::Point< P > Struct Template Reference	945
10.302.1Detailed Description	947
10.302.2Member Typedef Documentation	947
10.302.2.1point	947
10.302.3Member Function Documentation	947
10.302.3.1to_point	947
10.302.4Friends And Related Function Documentation	947
10.302.4.1operator+=	947
10.302.4.2operator-=	948
10.302.4.3operator/	948
10.303mln::point< G, C > Struct Template Reference	948
10.303.1Detailed Description	951
10.303.2Member Typedef Documentation	951
10.303.2.1coord	951
10.303.2.2delta	951
10.303.2.3dpsite	951
10.303.2.4grid	951
10.303.2.5h_vec	952
10.303.2.6vec	952
10.303.3Member Enumeration Documentation	952
10.303.3.1"@30	952
10.303.4Constructor & Destructor Documentation	952
10.303.4.1point	952
10.303.4.2point	952
10.303.4.3point	952
10.303.4.4point	952
10.303.4.5point	953
10.303.5Member Function Documentation	953
10.303.5.1last_coord	953

10.303.5.2last_coord	953
10.303.5.3minus_infty	953
10.303.5.4operator+=	953
10.303.5.5operator-=	953
10.303.5.6operator[]	953
10.303.5.7operator[]	954
10.303.5.8plus_infty	954
10.303.5.9set_all	954
10.303.5.10h_vec	954
10.303.5.11b_vec	954
10.303.6Member Data Documentation	955
10.303.6.1origin	955
10.304Inn::Point_Site< E > Struct Template Reference	955
10.304.1Detailed Description	956
10.304.2Friends And Related Function Documentation	956
10.304.2.1operator+	956
10.304.2.2operator-	957
10.304.2.3operator-	957
10.304.2.4operator<<	958
10.304.2.5operator==	958
10.305Inn::Point_Site< void > Struct Template Reference	958
10.305.1Detailed Description	959
10.306Inn::Proxy< E > Struct Template Reference	959
10.306.1Detailed Description	959
10.307Inn::Proxy< void > Struct Template Reference	959
10.307.1Detailed Description	959
10.308Inn::Pseudo_Site< E > Struct Template Reference	959
10.308.1Detailed Description	960
10.309Inn::Pseudo_Site< void > Struct Template Reference	960
10.309.1Detailed Description	960
10.310Inn::pw::image< F, S > Class Template Reference	960
10.310.1Detailed Description	961
10.310.2Member Typedef Documentation	961
10.310.2.1skeleton	961
10.310.3Constructor & Destructor Documentation	961
10.310.3.1image	961

10.310.3.2image	961
10.311ln::registration::closest_point_basic< P > Class Template Reference	961
10.311. Detailed Description	962
10.312ln::registration::closest_point_with_map< P > Class Template Reference	962
10.312. Detailed Description	962
10.313ln::Regular_Grid< E > Struct Template Reference	962
10.313. Detailed Description	962
10.314ln::safe_image< I > Class Template Reference	963
10.314. Detailed Description	963
10.314.2Member Typedef Documentation	963
10.314.2.1skeleton	963
10.314.3Member Function Documentation	963
10.314.3.1operator safe_image< const I >	963
10.315ln::select::p_of< P > Struct Template Reference	963
10.315. Detailed Description	964
10.316ln::Site< E > Struct Template Reference	964
10.316. Detailed Description	964
10.317ln::Site< void > Struct Template Reference	965
10.317. Detailed Description	965
10.318ln::Site_Iterator< E > Struct Template Reference	965
10.318. Detailed Description	967
10.318.2Member Function Documentation	967
10.318.2.1next	967
10.319ln::Site_Proxy< E > Struct Template Reference	967
10.319. Detailed Description	967
10.320ln::Site_Proxy< void > Struct Template Reference	968
10.320. Detailed Description	968
10.321ln::Site_Set< E > Struct Template Reference	968
10.321. Detailed Description	969
10.321.2Friends And Related Function Documentation	970
10.321.2.1diff	970
10.321.2.2inter	970
10.321.2.3operator<	970
10.321.2.4operator<<	970
10.321.2.5operator<=	970
10.321.2.6operator==	971

10.321.2.7sym_diff	971
10.321.2.8uni	971
10.321.2.9unique	971
10.322nl::Site_Set< void > Struct Template Reference	971
10.322.Detailed Description	971
10.323nl::slice_image< I > Struct Template Reference	971
10.323.Detailed Description	972
10.323.Member Typedef Documentation	972
10.323.2.Iskeleton	972
10.323.Constructor & Destructor Documentation	973
10.323.3.Islice_image	973
10.323.3.2slice_image	973
10.323.Member Function Documentation	973
10.323.4.Idomain	973
10.323.4.2operator slice_image< const I >	973
10.323.4.3operator()	973
10.323.4.4operator()	973
10.323.4.5li	973
10.324nl::sub_image< I, S > Class Template Reference	974
10.324.Detailed Description	974
10.324.Member Typedef Documentation	974
10.324.2.Iskeleton	974
10.324.Constructor & Destructor Documentation	975
10.324.3.Isub_image	975
10.324.3.2sub_image	975
10.324.Member Function Documentation	975
10.324.4.Idomain	975
10.324.4.2operator sub_image< const I, S >	975
10.325nl::sub_image_if< I, S > Struct Template Reference	975
10.325.Detailed Description	976
10.325.Member Typedef Documentation	976
10.325.2.Iskeleton	976
10.325.Constructor & Destructor Documentation	976
10.325.3.Isub_image_if	976
10.325.3.2sub_image_if	976
10.325.Member Function Documentation	976

10.325.4. <code>ldomain</code>	976
10.326 <code>mln::thru_image< I, F ></code> Class Template Reference	977
10.326. Detailed Description	977
10.326. Member Function Documentation	977
10.326.2. <code>operator thru_image< const I, F ></code>	977
10.327 <code>mln::thrubin_image< I1, I2, F ></code> Class Template Reference	977
10.327. Detailed Description	978
10.327. Member Typedef Documentation	978
10.327.2. <code>lpsite</code>	978
10.327.2. <code>2value</code>	978
10.327.2. <code>3skeleton</code>	978
10.327.2. <code>4value</code>	978
10.328 <code>mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D ></code> Class Template Reference	978
10.328. Detailed Description	979
10.328. Constructor & Destructor Documentation	979
10.328.2. <code>ladj_higher_dim_connected_n_face_bkd_iter</code>	979
10.328. Member Function Documentation	979
10.328.3. <code>lnext</code>	979
10.329 <code>mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D ></code> Class Template Reference	980
10.329. Detailed Description	980
10.329. Constructor & Destructor Documentation	980
10.329.2. <code>ladj_higher_dim_connected_n_face_fwd_iter</code>	980
10.329. Member Function Documentation	980
10.329.3. <code>lnext</code>	980
10.330 <code>mln::topo::adj_higher_face_bkd_iter< D ></code> Class Template Reference	981
10.330. Detailed Description	981
10.330. Constructor & Destructor Documentation	981
10.330.2. <code>ladj_higher_face_bkd_iter</code>	981
10.330. Member Function Documentation	982
10.330.3. <code>lnext</code>	982
10.331 <code>mln::topo::adj_higher_face_fwd_iter< D ></code> Class Template Reference	982
10.331. Detailed Description	982
10.331. Constructor & Destructor Documentation	982
10.331.2. <code>ladj_higher_face_fwd_iter</code>	982
10.331. Member Function Documentation	983
10.331.3. <code>lnext</code>	983

10.332	mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D > Class Template Reference	983
10.332.	Detailed Description	983
10.332.	Constructor & Destructor Documentation	984
10.332.2.	ladj_lower_dim_connected_n_face_bkd_iter	984
10.332.	Member Function Documentation	984
10.332.3.	lnext	984
10.333	mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D > Class Template Reference	984
10.333.	Detailed Description	985
10.333.	Constructor & Destructor Documentation	985
10.333.2.	ladj_lower_dim_connected_n_face_fwd_iter	985
10.333.	Member Function Documentation	985
10.333.3.	lnext	985
10.334	mln::topo::adj_lower_face_bkd_iter< D > Class Template Reference	985
10.334.	Detailed Description	986
10.334.	Constructor & Destructor Documentation	986
10.334.2.	ladj_lower_face_bkd_iter	986
10.334.	Member Function Documentation	986
10.334.3.	lnext	986
10.335	mln::topo::adj_lower_face_fwd_iter< D > Class Template Reference	986
10.335.	Detailed Description	987
10.335.	Constructor & Destructor Documentation	987
10.335.2.	ladj_lower_face_fwd_iter	987
10.335.	Member Function Documentation	987
10.335.3.	lnext	987
10.336	mln::topo::adj_lower_higher_face_bkd_iter< D > Class Template Reference	987
10.336.	Detailed Description	988
10.336.	Constructor & Destructor Documentation	988
10.336.2.	ladj_lower_higher_face_bkd_iter	988
10.336.	Member Function Documentation	988
10.336.3.	lnext	988
10.337	mln::topo::adj_lower_higher_face_fwd_iter< D > Class Template Reference	989
10.337.	Detailed Description	989
10.337.	Constructor & Destructor Documentation	989
10.337.2.	ladj_lower_higher_face_fwd_iter	989
10.337.	Member Function Documentation	989
10.337.3.	lnext	989

10.338	<code>nl::topo::adj_m_face_bkd_iter< D ></code> Class Template Reference	990
10.338.	Detailed Description	990
10.338.	Constructor & Destructor Documentation	990
10.338.2.	<code>ladj_m_face_bkd_iter</code>	990
10.338.2.	<code>2adj_m_face_bkd_iter</code>	991
10.338.	Member Function Documentation	991
10.338.3.	<code>lnext</code>	991
10.339	<code>nl::topo::adj_m_face_fwd_iter< D ></code> Class Template Reference	991
10.339.	Detailed Description	991
10.339.	Constructor & Destructor Documentation	992
10.339.2.	<code>ladj_m_face_fwd_iter</code>	992
10.339.2.	<code>2adj_m_face_fwd_iter</code>	992
10.339.	Member Function Documentation	992
10.339.3.	<code>lnext</code>	992
10.340	<code>nl::topo::algebraic_face< D ></code> Class Template Reference	992
10.340.	Detailed Description	994
10.340.	Constructor & Destructor Documentation	994
10.340.2.	<code>lalgebraic_face</code>	994
10.340.2.	<code>2algebraic_face</code>	995
10.340.2.	<code>3algebraic_face</code>	995
10.340.2.	<code>4algebraic_face</code>	995
10.340.	Member Function Documentation	995
10.340.3.	<code>lcplx</code>	995
10.340.3.	<code>2data</code>	995
10.340.3.	<code>3dec_face_id</code>	995
10.340.3.	<code>4dec_n</code>	996
10.340.3.	<code>5face_id</code>	996
10.340.3.	<code>6higher_dim_adj_faces</code>	996
10.340.3.	<code>7nc_face_id</code>	996
10.340.3.	<code>8nc_n</code>	996
10.340.3.	<code>9invalidate</code>	996
10.340.3.	<code>10_valid</code>	996
10.340.3.	<code>11bwer_dim_adj_faces</code>	997
10.340.3.	<code>12</code>	997
10.340.3.	<code>13set_cplx</code>	997
10.340.3.	<code>14set_face_id</code>	997

10.340.3.15	set_n	997
10.340.3.16	set_sign	997
10.340.3.17	sign	997
10.341	ln::topo::algebraic_n_face< N, D > Class Template Reference	998
10.341.1	Detailed Description	999
10.341.2	Constructor & Destructor Documentation	999
10.341.2.1	algebraic_n_face	999
10.341.2.2	algebraic_n_face	1000
10.341.2.3	algebraic_n_face	1000
10.341.3	Member Function Documentation	1000
10.341.3.1	cplx	1000
10.341.3.2	data	1000
10.341.3.3	dec_face_id	1000
10.341.3.4	face_id	1000
10.341.3.5	higher_dim_adj_faces	1001
10.341.3.6	inc_face_id	1001
10.341.3.7	invalidate	1001
10.341.3.8	is_valid	1001
10.341.3.9	lower_dim_adj_faces	1001
10.341.3.10		1001
10.341.3.11	set_cplx	1002
10.341.3.12	set_face_id	1002
10.341.3.13	set_sign	1002
10.341.3.14	sign	1002
10.342	ln::topo::center_only_iter< D > Class Template Reference	1002
10.342.1	Detailed Description	1003
10.342.2	Constructor & Destructor Documentation	1003
10.342.2.1	center_only_iter	1003
10.342.3	Member Function Documentation	1003
10.342.3.1	next	1003
10.343	ln::topo::centered_bkd_iter_adapter< D, I > Class Template Reference	1003
10.343.1	Detailed Description	1004
10.343.2	Constructor & Destructor Documentation	1004
10.343.2.1	centered_bkd_iter_adapter	1004
10.343.3	Member Function Documentation	1004
10.343.3.1	next	1004

10.344	<code>ln::topo::centered_fwd_iter_adapter< D, I ></code> Class Template Reference	1005
10.344.1	Detailed Description	1005
10.344.2	Constructor & Destructor Documentation	1005
10.344.2.1	<code>centered_fwd_iter_adapter</code>	1005
10.344.3	Member Function Documentation	1005
10.344.3.1	<code>next</code>	1005
10.345	<code>ln::topo::complex< D ></code> Class Template Reference	1006
10.345.1	Detailed Description	1007
10.345.2	Member Typedef Documentation	1007
10.345.2.1	<code>bkd_citer</code>	1007
10.345.2.2	<code>fwd_citer</code>	1007
10.345.3	Constructor & Destructor Documentation	1007
10.345.3.1	<code>complex</code>	1007
10.345.4	Member Function Documentation	1007
10.345.4.1	<code>add_face</code>	1007
10.345.4.2	<code>add_face</code>	1008
10.345.4.3	<code>addr</code>	1008
10.345.4.4	<code>nfaces</code>	1008
10.345.4.5	<code>nfaces_of_dim</code>	1008
10.345.4.6	<code>nfaces_of_static_dim</code>	1008
10.345.4.7	<code>print</code>	1009
10.345.4.8	<code>print_faces</code>	1009
10.346	<code>ln::topo::face< D ></code> Class Template Reference	1009
10.346.1	Detailed Description	1010
10.346.2	Constructor & Destructor Documentation	1011
10.346.2.1	<code>iface</code>	1011
10.346.2.2	<code>face</code>	1011
10.346.2.3	<code>face</code>	1011
10.346.3	Member Function Documentation	1011
10.346.3.1	<code>lcplx</code>	1011
10.346.3.2	<code>data</code>	1011
10.346.3.3	<code>dec_face_id</code>	1011
10.346.3.4	<code>dec_n</code>	1011
10.346.3.5	<code>face_id</code>	1012
10.346.3.6	<code>higher_dim_adj_faces</code>	1012
10.346.3.7	<code>nc_face_id</code>	1012

10.346.3.8	<code>inc_n</code>	1012
10.346.3.9	<code>invalidate</code>	1012
10.346.3.10	<code>is_valid</code>	1012
10.346.3.11	<code>lower_dim_adj_faces</code>	1012
10.346.3.12	<code>l2</code>	1013
10.346.3.13	<code>set_cplx</code>	1013
10.346.3.14	<code>set_face_id</code>	1013
10.346.3.15	<code>set_n</code>	1013
10.347	<code>mln::topo::face_bkd_iter< D ></code> Class Template Reference	1013
10.347.1	Detailed Description	1014
10.347.2	Constructor & Destructor Documentation	1014
10.347.2.1	<code>face_bkd_iter</code>	1014
10.347.3	Member Function Documentation	1014
10.347.3.1	<code>next</code>	1014
10.347.3.2	<code>start</code>	1014
10.348	<code>mln::topo::face_fwd_iter< D ></code> Class Template Reference	1014
10.348.1	Detailed Description	1015
10.348.2	Constructor & Destructor Documentation	1015
10.348.2.1	<code>face_fwd_iter</code>	1015
10.348.3	Member Function Documentation	1015
10.348.3.1	<code>next</code>	1015
10.348.3.2	<code>start</code>	1016
10.349	<code>mln::topo::is_n_face< N ></code> Struct Template Reference	1016
10.349.1	Detailed Description	1017
10.350	<code>mln::topo::is_simple_2d_t< N ></code> Struct Template Reference	1017
10.350.1	Detailed Description	1017
10.351	<code>mln::topo::is_simple_cell< I ></code> Class Template Reference	1017
10.351.1	Detailed Description	1019
10.351.2	Member Typedef Documentation	1019
10.351.2.1	<code>lpsite</code>	1019
10.351.2.2	<code>result</code>	1019
10.351.3	Member Function Documentation	1019
10.351.3.1	<code>mln_geom</code>	1019
10.351.3.2	<code>operator()</code>	1019
10.351.3.3	<code>set_image</code>	1020
10.351.4	Member Data Documentation	1020

10.351.4.ID	1020
10.352.In::topo::n_face< N, D > Class Template Reference	1020
10.352.Detailed Description	1021
10.352.Constructor & Destructor Documentation	1021
10.352.2.In_face	1021
10.352.2.2n_face	1022
10.352.3.Member Function Documentation	1022
10.352.3.1cplx	1022
10.352.3.2data	1022
10.352.3.3dec_face_id	1022
10.352.3.4face_id	1022
10.352.3.5higher_dim_adj_faces	1022
10.352.3.6nc_face_id	1023
10.352.3.7invalidate	1023
10.352.3.8s_valid	1023
10.352.3.9lower_dim_adj_faces	1023
10.352.3.10	1023
10.352.3.11set_cplx	1023
10.352.3.12set_face_id	1023
10.353.In::topo::n_face_bkd_iter< D > Class Template Reference	1024
10.353.Detailed Description	1024
10.353.Constructor & Destructor Documentation	1024
10.353.2.In_face_bkd_iter	1024
10.353.3.Member Function Documentation	1025
10.353.3.1n	1025
10.353.3.2next	1025
10.353.3.3start	1025
10.354.In::topo::n_face_fwd_iter< D > Class Template Reference	1025
10.354.Detailed Description	1026
10.354.Constructor & Destructor Documentation	1026
10.354.2.In_face_fwd_iter	1026
10.354.3.Member Function Documentation	1026
10.354.3.1n	1026
10.354.3.2next	1026
10.354.3.3start	1027
10.355.In::topo::n_faces_set< N, D > Class Template Reference	1027

10.355. Detailed Description	1027
10.355. Member Typedef Documentation	1027
10.355.2. lfaces_type	1027
10.355. Member Function Documentation	1028
10.355.3. ladd	1028
10.355.3.2. lfaces	1028
10.355.3.3. lreserve	1028
10.356. mln::topo::skeleton::is_simple_point< N > Struct Template Reference	1028
10.356. Detailed Description	1028
10.357. mln::topo::static_n_face_bkd_iter< N, D > Class Template Reference	1029
10.357. Detailed Description	1029
10.357. Constructor & Destructor Documentation	1029
10.357.2. lstatic_n_face_bkd_iter	1029
10.357. Member Function Documentation	1030
10.357.3. lnext	1030
10.357.3.2. lstart	1030
10.358. mln::topo::static_n_face_fwd_iter< N, D > Class Template Reference	1030
10.358. Detailed Description	1030
10.358. Constructor & Destructor Documentation	1031
10.358.2. lstatic_n_face_fwd_iter	1031
10.358. Member Function Documentation	1031
10.358.3. lnext	1031
10.358.3.2. lstart	1031
10.359. mln::tr_image< S, I, T > Struct Template Reference	1031
10.359. Detailed Description	1032
10.359. Member Typedef Documentation	1033
10.359.2. llvalue	1033
10.359.2.2. lpsite	1033
10.359.2.3. lrvalue	1033
10.359.2.4. lsite	1033
10.359.2.5. lskeleton	1033
10.359.2.6. lvalue	1033
10.359. Constructor & Destructor Documentation	1033
10.359.3. ltr_image	1033
10.359. Member Function Documentation	1034
10.359.4. ldomain	1034

10.359.4.2has	1034
10.359.4.3is_valid	1034
10.359.4.4operator()	1034
10.359.4.5set_tr	1034
10.359.4.6r	1034
10.360. In::unproject_image< I, D, F > Struct Template Reference	1034
10.360. Detailed Description	1035
10.360. Constructor & Destructor Documentation	1035
10.360.2. unproject_image	1035
10.360.2.2 unproject_image	1035
10.360.3. Member Function Documentation	1035
10.360.3.1 domain	1035
10.360.3.2 operator()	1036
10.360.3.3 operator()	1036
10.361. In::util::adjacency_matrix< V > Class Template Reference	1036
10.361. Detailed Description	1036
10.361. Constructor & Destructor Documentation	1036
10.361.2. adjacency_matrix	1036
10.361.2.2 adjacency_matrix	1037
10.362. In::util::array< T > Class Template Reference	1037
10.362. Detailed Description	1039
10.362. Member Typedef Documentation	1039
10.362.2. l_bkd_eiter	1039
10.362.2.2 eiter	1039
10.362.2.3 element	1040
10.362.2.4 fwd_eiter	1040
10.362.2.5 result	1040
10.362. Constructor & Destructor Documentation	1040
10.362.3. larray	1040
10.362.3.2 array	1040
10.362.3.3 array	1040
10.362.4. Member Function Documentation	1040
10.362.4.1 lappend	1040
10.362.4.2 append	1041
10.362.4.3 clear	1041
10.362.4.4 fill	1041

10.362.4.5	is_empty	1041
10.362.4.6	last	1041
10.362.4.7	last	1042
10.362.4.8	memory_size	1042
10.362.4.9	elements	1042
10.362.4.10	operator()	1042
10.362.4.11	operator()	1042
10.362.4.12	operator[]	1043
10.362.4.13	operator[]	1043
10.362.4.14	reserve	1043
10.362.4.15	size	1043
10.362.4.16	size	1043
10.362.4.17	size	1043
10.362.4.18	sd_vector	1044
10.363	util::branch< T > Class Template Reference	1044
10.363.1	Detailed Description	1044
10.363.2	Constructor & Destructor Documentation	1044
10.363.2.1	branch	1044
10.363.3	Member Function Documentation	1045
10.363.3.1	apex	1045
10.363.3.2	util_tree	1045
10.364	util::branch_iter< T > Class Template Reference	1045
10.364.1	Detailed Description	1046
10.364.2	Member Function Documentation	1046
10.364.2.1	deepness	1046
10.364.2.2	invalidate	1046
10.364.2.3	is_valid	1046
10.364.2.4	next	1046
10.364.2.5	operator util::tree_node< T > &	1047
10.364.2.6	start	1047
10.365	util::branch_iter_ind< T > Class Template Reference	1047
10.365.1	Detailed Description	1047
10.365.2	Member Function Documentation	1048
10.365.2.1	deepness	1048
10.365.2.2	invalidate	1048
10.365.2.3	is_valid	1048

10.365.2.4next	1048
10.365.2.5operator util::tree_node< T > &	1048
10.365.2.6start	1048
10.366In::util::couple< T, U > Class Template Reference	1048
10.366.Detailed Description	1049
10.366.Member Function Documentation	1050
10.366.2.lchange_both	1050
10.366.2.2change_first	1050
10.366.2.3change_second	1050
10.366.2.4first	1050
10.366.2.5second	1050
10.367In::util::eat Struct Reference	1050
10.367.Detailed Description	1051
10.368In::util::edge< G > Class Template Reference	1051
10.368.Detailed Description	1052
10.368.Member Typedef Documentation	1053
10.368.2.lcategory	1053
10.368.2.2graph_t	1053
10.368.2.3id_t	1053
10.368.2.4id_value_t	1053
10.368.Constructor & Destructor Documentation	1053
10.368.3.ledge	1053
10.368.Member Function Documentation	1053
10.368.4.lchange_graph	1053
10.368.4.2graph	1053
10.368.4.3id	1054
10.368.4.4invalidate	1054
10.368.4.5is_valid	1054
10.368.4.6th_nbh_edge	1054
10.368.4.7nmax_nbh_edges	1054
10.368.4.8operator edge_id_t	1054
10.368.4.9update_id	1054
10.368.4.10l	1055
10.368.4.112	1055
10.368.4.12_other	1055
10.369In::util::fibonacci_heap< P, T > Class Template Reference	1055

10.369. Detailed Description	1057
10.369. Constructor & Destructor Documentation	1057
10.369.2. lfibonacci_heap	1057
10.369.2.2 fibonacci_heap	1057
10.369.3. Member Function Documentation	1057
10.369.3.1 clear	1057
10.369.3.2 front	1057
10.369.3.3 is_empty	1058
10.369.3.4 is_valid	1058
10.369.3.5 elements	1058
10.369.3.6 operator=	1058
10.369.3.7 pop_front	1058
10.369.3.8 push	1058
10.369.3.9 push	1059
10.370. In::util::graph Class Reference	1059
10.370. Detailed Description	1061
10.370. Member Typedef Documentation	1061
10.370.2.1 edge_fwd_iter	1061
10.370.2.2 edge_nbh_edge_fwd_iter	1061
10.370.2.3 edges_set_t	1061
10.370.2.4 edges_t	1062
10.370.2.5 vertex_fwd_iter	1062
10.370.2.6 vertex_nbh_edge_fwd_iter	1062
10.370.2.7 vertex_nbh_vertex_fwd_iter	1062
10.370.2.8 vertices_t	1062
10.370. Constructor & Destructor Documentation	1062
10.370.3.1 graph	1062
10.370.3.2 graph	1062
10.370.4. Member Function Documentation	1063
10.370.4.1 add_edge	1063
10.370.4.2 add_vertex	1063
10.370.4.3 add_vertices	1063
10.370.4.4 e_ith_nbh_edge	1063
10.370.4.5 e_nmax	1064
10.370.4.6 e_nmax_nbh_edges	1064
10.370.4.7 edge	1064

10.370.4.8	edge	1064
10.370.4.9	edges	1064
10.370.4.10	has_e	1064
10.370.4.11	has_v	1065
10.370.4.12	is_subgraph_of	1065
10.370.4.13		1065
10.370.4.14		1065
10.370.4.15	ith_nbh_edge	1065
10.370.4.16	ith_nbh_vertex	1065
10.370.4.17	nmax	1066
10.370.4.18	nmax_nbh_edges	1066
10.370.4.19	nmax_nbh_vertices	1066
10.370.4.20	vertex	1066
10.371	util::greater_point< I > Class Template Reference	1066
10.371.1	Detailed Description	1066
10.371.2	Member Function Documentation	1067
10.371.2.1	operator()	1067
10.372	util::greater_site< I > Class Template Reference	1067
10.372.1	Detailed Description	1067
10.372.2	Member Function Documentation	1067
10.372.2.1	operator()	1067
10.373	util::head< T, R > Class Template Reference	1067
10.373.1	Detailed Description	1068
10.374	util::ignore Struct Reference	1068
10.374.1	Detailed Description	1068
10.375	util::ilcell< T > Struct Template Reference	1068
10.375.1	Detailed Description	1069
10.376	util::line_graph< G > Class Template Reference	1069
10.376.1	Detailed Description	1071
10.376.2	Member Typedef Documentation	1071
10.376.2.1	edge_fwd_iter	1071
10.376.2.2	edge_nbh_edge_fwd_iter	1071
10.376.2.3	edges_t	1071
10.376.2.4	vertex_fwd_iter	1071
10.376.2.5	vertex_nbh_edge_fwd_iter	1072
10.376.2.6	vertex_nbh_vertex_fwd_iter	1072

10.376.2.7vertices_t	1072
10.376.3Member Function Documentation	1072
10.376.3.1e_ith_nbh_edge	1072
10.376.3.2e_nmax	1072
10.376.3.3e_nmax_nbh_edges	1072
10.376.3.4edge	1073
10.376.3.5graph	1073
10.376.3.6has	1073
10.376.3.7has	1073
10.376.3.8has_e	1073
10.376.3.9has_v	1073
10.376.3.10_subgraph_of	1074
10.376.3.11	1074
10.376.3.12	1074
10.376.3.13_ith_nbh_edge	1074
10.376.3.14_ith_nbh_vertex	1074
10.376.3.15_nmax	1075
10.376.3.16_nmax_nbh_edges	1075
10.376.3.17_nmax_nbh_vertices	1075
10.376.3.18vertex	1075
10.377In::util::nil Struct Reference	1075
10.377.Detailed Description	1076
10.378In::util::node< T, R > Class Template Reference	1076
10.378.Detailed Description	1076
10.379In::util::object_id< Tag, V > Class Template Reference	1076
10.379.Detailed Description	1077
10.379.Member Typedef Documentation	1078
10.379.2.ivalue_t	1078
10.379.Constructor & Destructor Documentation	1078
10.379.3.lobject_id	1078
10.380In::util::ord< T > Struct Template Reference	1078
10.380.Detailed Description	1078
10.381In::util::ord_pair< T > Struct Template Reference	1078
10.381.Detailed Description	1079
10.381.Member Function Documentation	1080
10.381.2.lchange_both	1080

10.381.2.2	change_first	1080
10.381.2.3	change_second	1080
10.381.2.4	first	1080
10.381.2.5	second	1080
10.382	util::pix< I > Struct Template Reference	1081
10.382.1	Detailed Description	1081
10.382.2	Member Typedef Documentation	1081
10.382.2.1	site	1081
10.382.2.2	value	1081
10.382.3	Constructor & Destructor Documentation	1082
10.382.3.1	pix	1082
10.382.4	Member Function Documentation	1082
10.382.4.1	ima	1082
10.382.4.2	p	1082
10.382.4.3	v	1082
10.383	util::set< T > Class Template Reference	1082
10.383.1	Detailed Description	1084
10.383.2	Member Typedef Documentation	1085
10.383.2.1	bkd_eiter	1085
10.383.2.2	eiter	1085
10.383.2.3	element	1085
10.383.2.4	fwd_eiter	1085
10.383.3	Constructor & Destructor Documentation	1085
10.383.3.1	set	1085
10.383.4	Member Function Documentation	1085
10.383.4.1	clear	1085
10.383.4.2	first_element	1085
10.383.4.3	has	1086
10.383.4.4	insert	1086
10.383.4.5	insert	1086
10.383.4.6	is_empty	1087
10.383.4.7	last_element	1087
10.383.4.8	memory_size	1087
10.383.4.9	elements	1087
10.383.4.10	operator[]	1087
10.383.4.11	remove	1088

10.383.4.1	std::vector	1088
10.384	std::site_pair< P > Class Template Reference	1088
10.384.1	Detailed Description	1089
10.384.2	Member Function Documentation	1089
10.384.2.1	first	1089
10.384.2.2	pair	1090
10.384.2.3	second	1090
10.385	std::soft_heap< T, R > Class Template Reference	1090
10.385.1	Detailed Description	1091
10.385.2	Member Typedef Documentation	1091
10.385.2.1	element	1091
10.385.3	Constructor & Destructor Documentation	1091
10.385.3.1	soft_heap	1091
10.385.3.2	~soft_heap	1092
10.385.4	Member Function Documentation	1092
10.385.4.1	clear	1092
10.385.4.2	is_empty	1092
10.385.4.3	is_valid	1092
10.385.4.4	elements	1092
10.385.4.5	pop_front	1092
10.385.4.6	push	1093
10.385.4.7	push	1093
10.386	std::timer Class Reference	1093
10.386.1	Detailed Description	1094
10.387	std::tracked_ptr< T > Struct Template Reference	1094
10.387.1	Detailed Description	1094
10.387.2	Constructor & Destructor Documentation	1095
10.387.2.1	tracked_ptr	1095
10.387.2.2	tracked_ptr	1095
10.387.2.3	~tracked_ptr	1095
10.387.3	Member Function Documentation	1095
10.387.3.1	operator bool	1095
10.387.3.2	operator!	1095
10.387.3.3	operator->	1095
10.387.3.4	operator->	1095
10.387.3.5	operator=	1096

10.387.3.operator=	1096
10.388. In::util::tree< T > Class Template Reference	1096
10.388. Detailed Description	1097
10.388. Constructor & Destructor Documentation	1097
10.388.2. ltree	1097
10.388.2.2tree	1097
10.388. Member Function Documentation	1097
10.388.3. ladd_tree_down	1097
10.388.3.2add_tree_up	1097
10.388.3.3check_consistency	1098
10.388.3.4main_branch	1098
10.388.3.5root	1098
10.389. In::util::tree_node< T > Class Template Reference	1098
10.389. Detailed Description	1099
10.389. Constructor & Destructor Documentation	1100
10.389.2. ltree_node	1100
10.389.2.2tree_node	1100
10.389. Member Function Documentation	1100
10.389.3. ladd_child	1100
10.389.3.2add_child	1100
10.389.3.3check_consistency	1101
10.389.3.4children	1101
10.389.3.5children	1101
10.389.3.6delete_tree_node	1101
10.389.3.7elt	1101
10.389.3.8elt	1102
10.389.3.9parent	1102
10.389.3.10print	1102
10.389.3.11search	1102
10.389.3.12search_rec	1103
10.389.3.13set_parent	1103
10.390. In::util::vertex< G > Class Template Reference	1103
10.390. Detailed Description	1105
10.390. Member Typedef Documentation	1105
10.390.2. ICategory	1105
10.390.2.2graph_t	1105

10.390.2.3id_t	1105
10.390.2.4id_value_t	1105
10.390.3.Constructor & Destructor Documentation	1105
10.390.3.1vertex	1105
10.390.4.Member Function Documentation	1106
10.390.4.1change_graph	1106
10.390.4.2edge_with	1106
10.390.4.3graph	1106
10.390.4.4id	1106
10.390.4.5invalidate	1106
10.390.4.6is_valid	1106
10.390.4.7th_nbh_edge	1107
10.390.4.8th_nbh_vertex	1107
10.390.4.9max_nbh_edges	1107
10.390.4.10max_nbh_vertices	1107
10.390.4.11operator vertex_id_t	1107
10.390.4.12other	1107
10.390.4.13update_id	1107
10.391.mn::util::yes Struct Reference	1108
10.391.1.Detailed Description	1108
10.392.mn::Value< E > Struct Template Reference	1108
10.392.1.Detailed Description	1109
10.393.mn::value::float01 Class Reference	1110
10.393.1.Detailed Description	1111
10.393.2.Member Typedef Documentation	1111
10.393.2.1lenc	1111
10.393.2.2equiv	1111
10.393.3.Constructor & Destructor Documentation	1111
10.393.3.1float01	1111
10.393.3.2float01	1111
10.393.3.3float01	1111
10.393.4.Member Function Documentation	1111
10.393.4.1nbits	1111
10.393.4.2operator float	1111
10.393.4.3set_nbits	1112
10.393.4.4to_nbits	1112

10.393.4.5value	1112
10.393.4.6value_ind	1112
10.394. ln::value::float01_f Struct Reference	1112
10.394. Detailed Description	1113
10.394. Constructor & Destructor Documentation	1113
10.394.2. float01_f	1113
10.394.2.2 float01_f	1113
10.394.3. Member Function Documentation	1113
10.394.3.1 operator float	1113
10.394.3.2 operator=	1113
10.394.3.3 value	1113
10.395. ln::value::graylevel< n > Struct Template Reference	1113
10.395. Detailed Description	1115
10.395. Constructor & Destructor Documentation	1115
10.395.2.1 graylevel	1115
10.395.2.2 graylevel	1115
10.395.2.3 graylevel	1115
10.395.2.4 graylevel	1115
10.395.2.5 graylevel	1116
10.395.3. Member Function Documentation	1116
10.395.3.1 operator=	1116
10.395.3.2 operator=	1116
10.395.3.3 operator=	1116
10.395.3.4 operator=	1116
10.395.3.5 to_float	1116
10.395.3.6 value	1116
10.396. ln::value::graylevel_f Struct Reference	1117
10.396. Detailed Description	1118
10.396. Constructor & Destructor Documentation	1118
10.396.2.1 graylevel_f	1118
10.396.2.2 graylevel_f	1118
10.396.2.3 graylevel_f	1118
10.396.2.4 graylevel_f	1118
10.396.2.5 graylevel_f	1118
10.396.3. Member Function Documentation	1118
10.396.3.1 operator graylevel< n >	1118

10.396.3.2operator=	1118
10.396.3.3operator=	1119
10.396.3.4operator=	1119
10.396.3.5operator=	1119
10.396.3.6value	1119
10.397.1In::value::int_s< n > Struct Template Reference	1119
10.397.1.Detailed Description	1121
10.397.1.Constructor & Destructor Documentation	1121
10.397.2.1int_s	1121
10.397.2.2int_s	1121
10.397.2.3int_s	1121
10.397.1.Member Function Documentation	1121
10.397.3.1operator int	1121
10.397.3.2operator=	1121
10.397.1.Member Data Documentation	1121
10.397.4.1one	1121
10.397.4.2zero	1122
10.398.1In::value::int_u< n > Struct Template Reference	1122
10.398.1.Detailed Description	1123
10.398.1.Constructor & Destructor Documentation	1123
10.398.2.1int_u	1123
10.398.2.2int_u	1123
10.398.2.3int_u	1123
10.398.1.Member Function Documentation	1123
10.398.3.1next	1123
10.398.3.2operator unsigned	1123
10.398.3.3operator-	1124
10.398.3.4operator=	1124
10.399.1In::value::int_u_sat< n > Struct Template Reference	1124
10.399.1.Detailed Description	1125
10.399.1.Constructor & Destructor Documentation	1125
10.399.2.1int_u_sat	1125
10.399.2.2int_u_sat	1125
10.399.1.Member Function Documentation	1125
10.399.3.1operator int	1125
10.399.3.2operator+=	1126

10.399.3.3operator-=	1126
10.399.3.4operator=	1126
10.399.4Member Data Documentation	1126
10.399.4.1one	1126
10.399.4.2zero	1126
10.400In::value::Integer< E > Struct Template Reference	1126
10.400.Detailed Description	1127
10.401In::value::Integer< void > Struct Template Reference	1127
10.401.Detailed Description	1127
10.402In::value::label< n > Struct Template Reference	1127
10.402.Detailed Description	1128
10.402.2Member Typedef Documentation	1128
10.402.2.1enc	1128
10.402.3Constructor & Destructor Documentation	1129
10.402.3.1label	1129
10.402.3.2label	1129
10.402.3.3label	1129
10.402.4Member Function Documentation	1129
10.402.4.1next	1129
10.402.4.2operator unsigned	1129
10.402.4.3operator++	1129
10.402.4.4operator--	1129
10.402.4.5operator=	1130
10.402.4.6operator=	1130
10.402.4.7prev	1130
10.403In::value::lut_vec< S, T > Struct Template Reference	1130
10.403.Detailed Description	1132
10.403.2Member Typedef Documentation	1132
10.403.2.1bkd_viter	1132
10.403.2.2fwd_viter	1132
10.403.2.3value	1132
10.403.3Constructor & Destructor Documentation	1133
10.403.3.1lut_vec	1133
10.403.3.2lut_vec	1133
10.403.3.3lut_vec	1133
10.403.4Member Function Documentation	1133

10.403.4.1has	1133
10.403.4.2index_of	1133
10.403.4.3nvalues	1133
10.403.4.4operator[]	1134
10.404.1In::value::proxy< I > Class Template Reference	1134
10.404.1.Detailed Description	1135
10.404.2Member Typedef Documentation	1135
10.404.2.1enc	1135
10.404.2.2equiv	1135
10.404.3Constructor & Destructor Documentation	1135
10.404.3.1proxy	1135
10.404.3.2proxy	1136
10.404.3.3~proxy	1136
10.404.4Member Function Documentation	1136
10.404.4.1operator=	1136
10.404.4.2operator=	1136
10.404.4.3to_value	1136
10.405.1In::value::qt::rgb32 Struct Reference	1136
10.405.1.Detailed Description	1137
10.405.2Constructor & Destructor Documentation	1137
10.405.2.1rgb32	1137
10.405.2.2rgb32	1137
10.405.2.3rgb32	1137
10.405.2.4rgb32	1138
10.405.3Member Function Documentation	1138
10.405.3.1operator=	1138
10.405.3.2red	1138
10.405.4Member Data Documentation	1138
10.405.4.1zero	1138
10.406.1In::value::rgb< n > Struct Template Reference	1138
10.406.1.Detailed Description	1139
10.406.2Constructor & Destructor Documentation	1139
10.406.2.1rgb	1139
10.406.2.2rgb	1139
10.406.2.3rgb	1139
10.406.2.4rgb	1139

10.406.3	Member Function Documentation	1140
10.406.3.1	operator=	1140
10.406.3.2	red	1140
10.406.4	Member Data Documentation	1140
10.406.4.1	zero	1140
10.407	ln::value::set< T > Struct Template Reference	1140
10.407.1	Detailed Description	1140
10.407.2	Member Function Documentation	1141
10.407.2.1	the	1141
10.408	ln::value::sign Class Reference	1141
10.408.1	Detailed Description	1142
10.408.2	Member Typedef Documentation	1142
10.408.2.1	lenc	1142
10.408.2.2	equiv	1142
10.408.3	Constructor & Destructor Documentation	1142
10.408.3.1	lsign	1142
10.408.3.2	sign	1142
10.408.3.3	sign	1142
10.408.4	Member Function Documentation	1143
10.408.4.1	operator int	1143
10.408.4.2	operator=	1143
10.408.5	Member Data Documentation	1143
10.408.5.1	lone	1143
10.408.5.2	zero	1143
10.409	ln::value::stack_image< n, I > Struct Template Reference	1143
10.409.1	Detailed Description	1144
10.409.2	Member Typedef Documentation	1144
10.409.2.1	ldomain_t	1144
10.409.2.2	value	1144
10.409.2.3	psite	1145
10.409.2.4	rvalue	1145
10.409.2.5	skeleton	1145
10.409.2.6	value	1145
10.409.3	Constructor & Destructor Documentation	1145
10.409.3.1	lstack_image	1145
10.409.4	Member Function Documentation	1145

10.409.4.1lis_valid	1145
10.409.4.2operator()	1145
10.409.4.3operator()	1146
10.410ln::value::super_value< sign > Struct Template Reference	1146
10.410. Detailed Description	1146
10.411ln::value::value_array< T, V > Struct Template Reference	1146
10.411. Detailed Description	1146
10.411. Constructor & Destructor Documentation	1147
10.411.2. lvalue_array	1147
10.411. Member Function Documentation	1147
10.411.3. loperator()	1147
10.411.3.2operator[]	1147
10.411.3.3vset	1147
10.412ln::Value_Iterator< E > Struct Template Reference	1147
10.412. Detailed Description	1148
10.412. Member Function Documentation	1149
10.412.2. lnext	1149
10.412. Friends And Related Function Documentation	1149
10.412.3. loperator<<	1149
10.413ln::Value_Set< E > Struct Template Reference	1149
10.413. Detailed Description	1150
10.414ln::Vertex< E > Struct Template Reference	1150
10.414. Detailed Description	1150
10.415ln::vertex_image< P, V, G > Class Template Reference	1151
10.415. Detailed Description	1152
10.415. Member Typedef Documentation	1152
10.415.2. lgraph_t	1152
10.415.2.2nbh_t	1152
10.415.2.3site_function_t	1152
10.415.2.4skeleton	1152
10.415.2.5vertex_nbh_t	1152
10.415.2.6vertex_win_t	1152
10.415.2.7win_t	1153
10.415. Constructor & Destructor Documentation	1153
10.415.3. lvertex_image	1153
10.415. Member Function Documentation	1153

10.415.4. <code>operator()</code>	1153
10.416. <code>mln::violent_cast_image< T, I ></code> Struct Template Reference	1153
10.416. Detailed Description	1154
10.416. Member Typedef Documentation	1154
10.416.2. <code>llvalue</code>	1154
10.416.2. <code>rvalue</code>	1154
10.416.2. <code>skeleton</code>	1154
10.416.2. <code>value</code>	1154
10.416. Constructor & Destructor Documentation	1155
10.416.3. <code>lvioleat_image</code>	1155
10.416. Member Function Documentation	1155
10.416.4. <code>operator()</code>	1155
10.416.4. <code>operator()</code>	1155
10.417. <code>mln::w_window< D, W ></code> Struct Template Reference	1155
10.417. Detailed Description	1156
10.417. Member Typedef Documentation	1157
10.417.2. <code>lback_qiter</code>	1157
10.417.2. <code>dpsite</code>	1157
10.417.2. <code>fwq_qiter</code>	1157
10.417.2. <code>weight</code>	1157
10.417. Constructor & Destructor Documentation	1157
10.417.3. <code>lw_window</code>	1157
10.417. Member Function Documentation	1157
10.417.4. <code>lclear</code>	1157
10.417.4. <code>linsert</code>	1157
10.417.4. <code>lissymmetric</code>	1158
10.417.4. <code>lstd_vector</code>	1158
10.417.4. <code>lsym</code>	1158
10.417.4. <code>lw</code>	1158
10.417.4. <code>lweights</code>	1158
10.417.4. <code>lwin</code>	1158
10.417. Friends And Related Function Documentation	1159
10.417.5. <code>loperator<<</code>	1159
10.417.5. <code>loperator==</code>	1159
10.418. <code>mln::Weighted_Window< E ></code> Struct Template Reference	1159
10.418. Detailed Description	1160

10.418. Friends And Related Function Documentation	1160
10.418.2. loperator-	1160
10.419. mln::win::backdiag2d Struct Reference	1160
10.419. Detailed Description	1160
10.419. Constructor & Destructor Documentation	1161
10.419.2. lbackdiag2d	1161
10.419.3. Member Function Documentation	1161
10.419.3. llength	1161
10.420. mln::win::ball< G, C > Struct Template Reference	1161
10.420. Detailed Description	1161
10.420. Constructor & Destructor Documentation	1162
10.420.2. lball	1162
10.420.3. Member Function Documentation	1162
10.420.3. ldiameter	1162
10.421. mln::win::cube3d Struct Reference	1162
10.421. Detailed Description	1162
10.421. Constructor & Destructor Documentation	1163
10.421.2. lcube3d	1163
10.421.3. Member Function Documentation	1163
10.421.3. llength	1163
10.422. mln::win::cuboid3d Struct Reference	1163
10.422. Detailed Description	1164
10.422. Constructor & Destructor Documentation	1164
10.422.2. lcuboid3d	1164
10.422.3. Member Function Documentation	1165
10.422.3. ldepth	1165
10.422.3. lheight	1165
10.422.3. lvolume	1165
10.422.3. lwidth	1165
10.423. mln::win::diag2d Struct Reference	1165
10.423. Detailed Description	1165
10.423. Constructor & Destructor Documentation	1166
10.423.2. ldiag2d	1166
10.423.3. Member Function Documentation	1166
10.423.3. llength	1166
10.424. mln::win::line< M, i, C > Struct Template Reference	1166

10.424. Detailed Description	1167
10.424. Member Enumeration Documentation	1167
10.424.2. l"@87	1167
10.424. Constructor & Destructor Documentation	1167
10.424.3. lline	1167
10.424. Member Function Documentation	1168
10.424.4. llength	1168
10.424.4. 2size	1168
10.425. mln::win::multiple< W, F > Class Template Reference	1168
10.425. Detailed Description	1168
10.426. mln::win::multiple_size< n, W, F > Class Template Reference	1168
10.426. Detailed Description	1168
10.427. mln::win::octagon2d Struct Reference	1169
10.427. Detailed Description	1169
10.427. Constructor & Destructor Documentation	1169
10.427.2. loctagon2d	1169
10.427. Member Function Documentation	1170
10.427.3. larea	1170
10.427.3. 2length	1170
10.428. mln::win::rectangle2d Struct Reference	1170
10.428. Detailed Description	1170
10.428. Constructor & Destructor Documentation	1171
10.428.2. lrectangle2d	1171
10.428. Member Function Documentation	1171
10.428.3. larea	1171
10.428.3. 2height	1171
10.428.3. 3std_vector	1171
10.428.3. 4width	1171
10.429. mln::Window< E > Struct Template Reference	1171
10.429. Detailed Description	1172
10.430. mln::window< D > Class Template Reference	1172
10.430. Detailed Description	1174
10.430. Member Typedef Documentation	1174
10.430.2. lbkd_qiter	1174
10.430.2. 2fwd_qiter	1174
10.430.2. 3qiter	1174

10.430.2.4regular	1174
10.430.3Constructor & Destructor Documentation	1174
10.430.3.1window	1174
10.430.4Member Function Documentation	1175
10.430.4.1clear	1175
10.430.4.2delta	1175
10.430.4.3dp	1175
10.430.4.4has	1175
10.430.4.5insert	1175
10.430.4.6insert	1175
10.430.4.7insert	1176
10.430.4.8is_centered	1176
10.430.4.9is_empty	1176
10.430.4.10is_symmetric	1176
10.430.4.11print	1176
10.430.4.12size	1176
10.430.4.13std_vector	1177
10.430.4.14sym	1177
10.430.5Friends And Related Function Documentation	1177
10.430.5.1operator==	1177
10.431mln::world::inter_pixel::is_separator Struct Reference	1177
10.431.1Detailed Description	1178
10.432trait::graph< I > Struct Template Reference	1178
10.432.1Detailed Description	1179
10.433trait::graph< mln::complex_image< 1, G, V > > Struct Template Reference	1179
10.433.1Detailed Description	1179
10.434trait::graph< mln::image2d< T > > Struct Template Reference	1179
10.434.1Detailed Description	1179

Chapter 1

Documentation of milena

1.1 Introduction

This is the documentation of Milena.

1.2 Overview of Milena.

- [mln](#)
- [mln::accu](#)
- [mln::algebra](#)
- [mln::arith](#)
- [mln::binarization](#)
- [mln::border](#)
- [mln::canvas](#)
- [mln::convert](#)
- [mln::data](#)
- [mln::debug](#)
- [mln::display](#)
- [mln::draw](#)
- [mln::estim](#)
- [mln::extension](#)
- [mln::fun](#)
- [mln::geom](#)
- [mln::graph](#)
- [mln::histo](#)

- [mln::io](#)
- [mln::labeling](#)
- [mln::data](#)
- [mln::linear](#)
- [mln::literal](#)
- [mln::logical](#)
- [mln::make](#)
- [mln::math](#)
- [mln::metal](#)
- [mln::morpho](#)
- [mln::norm](#)
- [mln::opt](#)
- [mln::pw](#)
- [mln::registration](#)
- [mln::set](#)
- [mln::tag](#)
- [mln::test](#)
- [mln::topo](#)
- [mln::trace](#)
- [mln::trait](#)
- [mln::transform](#)
- [mln::util](#)
- [mln::value](#)
- [mln::win](#)

1.3 Copyright and License.

Copyright (C) 2007, 2008, 2009, 2010, 2011 EPITA Research and Development (LRDE)

This documentation is part of Olena.

Olena is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, version 2 of the License.

Olena is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with Olena. If not, see <http://www.gnu.org/licenses/>.

Chapter 2

Quick Reference Guide

- [Installation](#)
- [Foreword](#)
- [Site](#)
- [Site set](#)
- [Image](#)
- [Structural elements: Window and neighborhood](#)
- [Sites, psites and dpoints](#)
- [Iterators](#)
- [Memory management](#)
- [Basic routines](#)
- [Input / Output](#)
- [Graphs and images](#)
- [Useful global variables](#)
- [Useful macros](#)
- [Common Compilation Errors](#)

2.1 Installation

2.1.1 Requirements

2.1.1.1 To compile the user examples

2.1.1.2 To compile the documentation (Optional)

2.1.1.3 To develop in Olena

2.1.2 Getting Olena

2.1.3 Building Olena

2.2 Foreword

2.2.1 Generality

2.2.2 Directory hierarchy

2.2.3 Writing and compiling a program with Olena

2.3 Site

2.4 Site set

Iterators

2.4.1 Basic interface

2.4.2 Optional interface

```
box2d b(2,3);

// The bbox can be retrived in constant time.
std::cout << b.bbox() << std::endl;

// nsites can be retrieved in constant time.
std::cout << "nsites = " << b.nsites() << std::endl;

[(0,0)..(1,2)]
nsites = 6

p_array<point2d> arr;
arr.insert(point2d(1,0));
arr.insert(point2d(1,1));

// The bbox is computed thanks to bbox() algorithm.
box2d box = geom::bbox(arr);
std::cout << box << std::endl;

// p_array provides nsites(),
```

```
// it can be retrieved in constant time.
std::cout << "nsites = " << arr.nsites() << std::endl;

[(1,0)..(1,1)]
nsites = 2
```

2.5 Image

2.5.1 Definition

2.5.2 Possible image types

2.5.3 Possible value types

2.5.4 Domain

```
// Define a box2d from (-2,-3) to (3,5).
box2d b = make::box2d(-2,-3, 3,5);
// Initialize an image with b as domain.
image2d<int> ima(b);

std::cout << "b = " << b << std::endl;
std::cout << "domain = " << ima.domain() << std::endl;

b = [(-2,-3)..(3,5)]
domain = [(-2,-3)..(3,5)]

// Create an image on a 2D box
// with 10 columns and 10 rows.
image2d<bool> ima(make::box2d(10, 10));

mIn_site_(image2d<bool>) p1(20, 20);
mIn_site_(image2d<bool>) p2(3, 3);

std::cout << "has(p1)? "
  << (ima.has(p1) ? "true" : "false")
  << std::endl;

std::cout << "has(p2)? "
  << (ima.has(p2) ? "true" : "false")
  << std::endl;

has(p1)? false
has(p2)? true

point2d p(9,9);

// At (9, 9), both values change.
ima1(p) = 'M';
ima2(p) = 'W';

bool b = (ima1(p) == ima2(p));
std::cout << (b ? "True" : "False") << std::endl;

False
```

2.5.5 Border and extension

2.5.5.1 Image border

```

bool vals[3][3] = { { 0, 1, 1 },
                   { 1, 0, 0 },
                   { 1, 1, 0 } };

image2d<bool> ima_def = make::image(vals);
border::fill(ima_def, false);
debug::println_with_border(ima_def);

std::cout << "======" << std::endl << std::endl;

border::thickness = 0;
image2d<bool> ima_bt0 = make::image(vals);
debug::println_with_border(ima_bt0);

```

```

- - - - -
- - - - -
- - - - -
- - - | | - -
- - | - - - -
- - | | - - -
- - - - -
- - - - -
- - - - -

```

```

=====

```

```

- | |
| - -
| | -

```

2.5.5.2 Generality on image extension

imamorphed

2.5.5.3 Different extensions

```

image2d<rgb8> lena;
io::ppm::load(lena, MLN_IMG_DIR "/small.ppm");
box2d bbox_enlarged = lena.domain();
bbox_enlarged.enlarge(border::thickness);
mln_VAR(ima_roi, lena | fun::p2b::big_chess<box2d>(lena.domain(), 10));

```

2.5.5.3.1 Extension with a value

```

mln_VAR(ext_with_val, extended_to(extend(ima_roi, literal::blue), bbox_enlarged
));

```

2.5.5.3.2 Extension with a function

```

namespace mln
{
    struct my_ext : public Function_v2v<my_ext>
    {

```



```

typedef value::rgb8 result;

value::rgb8 operator()(const point2d& p) const
{
    if ((p.row() + p.col()) % 20)
        return literal::black;
    return literal::white;
}

};

} // end of namespace mln

mln_VAR(ext_with_fun, extended_to(extend(ima_roi, my_ext()), bbox_enlarged));

```

2.5.5.3.3 Extension with an image

```

mln_VAR(ext_with_ima, extend(ima_roi, lena));

// Default border size is set to 0.

// Image defined on a box2d from
// (0, 0) to (2, 2)
image2d<int> ima1(2, 3);

std::cout << "ima1.has(0, 0) : "
           << ima1.has(point2d(0, 0)) << std::endl;

std::cout << "ima1.has(-3, 0) : "
           << ima1.has(point2d(-3, 0)) << std::endl;

std::cout << "ima1.has(2, 5) : "
           << ima1.has(point2d(2, 5)) << std::endl;

std::cout << "======" << std::endl;

// Set default border size to 0.
border::thickness = 0;

// Image defined on a box2d from
// (0, 0) to (2, 2)
image2d<int> ima2(2, 3);

std::cout << "ima2.has(0, 0) : "
           << ima2.has(point2d(0, 0)) << std::endl;

std::cout << "ima2.has(-3, 0) : "
           << ima2.has(point2d(-3, 0)) << std::endl;

std::cout << "ima2.has(2, 5) : "
           << ima2.has(point2d(2, 5)) << std::endl;

ima1.has(0, 0) : 1
ima1.has(-3, 0) : 1
ima1.has(2, 5) : 1
====
ima2.has(0, 0) : 1
ima2.has(-3, 0) : 0
ima2.has(2, 5) : 0

border::thickness = 30;

// Declare the image to be rotated.

```

```

image2d<value::rgb8> ima1(220, 220);
data::fill(ima1, literal::cyan);
border::fill(ima1, literal::yellow);
// Set an infinite extension.
mln_VAR(ima1, extend(ima1, pw::cst(literal::yellow)));

// Declare the output image.
image2d<value::rgb8> ima2(220, 220);
data::fill(ima2, literal::cyan);
border::fill(ima2, literal::yellow);

box2d extended_domain= ima1.domain();
extended_domain.enlarge(border::thickness);

// Draw the domain bounding box
draw::box(ima1, geom::bbox(ima1), literal::red);
// Save the image, including its border.
doc::ppmsave(ima1 | extended_domain, "ima2d-rot");

// Define and apply a point-wise rotation
fun::x2x::rotation<2,float> rot1(0.5, literal::zero);
image2d<value::rgb8>::fwd_piter p(ima1.domain());
for_all(p)
{
    algebra::vec<2,float> pv = p.to_site().to_vec();
    algebra::vec<2,float> v = rot1.inv()(pv);
    ima2(p) = ima1(v);
}

draw::box(ima2, ima2.bbox(), literal::red);
doc::ppmsave(extended_to(ima2, extended_domain), "ima2d-rot");

my_routine(ima | ima.domain());

```

2.5.6 Interface

2.5.7 Load and save images

```

image2d<bool> ima;
io::pbm::load(ima, MLN_DOC_DIR "/img/small.pbm");

io::pbm::save(ima, MLN_DOC_DIR "/figures/ima_save.pbm");

```

2.5.8 Create an image

```

// Build an empty image;
image2d<value::int_u8> imgl;

// Build an image with 2 rows
// and 3 columns sites
image2d<value::int_u8> imglb(box2d(2, 3));
image2d<value::int_u8> imglc(2, 3);

bool vals[6][5] = {
    {0, 1, 1, 0, 0},
    {0, 1, 1, 0, 0},
    {0, 0, 0, 0, 0},
    {1, 1, 0, 1, 0},
    {1, 0, 1, 1, 1},
    {1, 0, 0, 0, 0}
};
image2d<bool> ima = make::image(vals);

```

```

image2d<value::int_u8> img2a(2, 3);
image2d<value::int_u8> img2b;

initialize(img2b, img2a);
data::fill(img2b, img2a);

```

Fill

2.5.9 Access and modify values

```

box2d b(2,3);
image2d<value::int_u8> ima(b);

// On image2d, Site <=> point2d
point2d p(1, 2);

// Associate '9' as value for the site/point2d (1,2).
// The value is returned by reference and can be changed.
opt::at(ima, 1,2) = 9;
std::cout << "opt::at(ima, 1,2) = " << opt::at(ima, 1,2)
          << std::endl;
std::cout << "ima(p) = " << ima(p) << std::endl;

std::cout << "---" << std::endl;

// Associate '2' as value for the site/point2d (1,2).
// The value is returned by reference
// and can be changed as well.
ima(p) = 2;
std::cout << "opt::at(ima, 1,2) = " << opt::at(ima, 1,2)
          << std::endl;
std::cout << "ima(p) = " << ima(p) << std::endl;

opt::at(ima, 1,2) = 9
ima(p) = 9
---
opt::at(ima, 1,2) = 2
ima(p) = 2

```

Iterators

2.5.10 Image size

```

image2d<int> ima(make::box2d(0,0, 10,12));

std::cout << "nrows = " << ima.nrows()
          << " - "
          << "ncols = " << ima.ncols()
          << std::endl;

nrows = 11 - ncols = 13

```

2.6 Structural elements: Window and neighborhood

2.6.1 Define an element

2.6.1.1 Window

2.6.1.2 Neighborhood

```
label_8 nlabels;
image2d<label_8> lbl = labeling::blobs(ima, c4(), nlabels);
```

2.6.1.3 Custom structural elements

```
window2d win;
win.insert(-1, -1);
win.insert(-1, 0);
win.insert(-1, 1);
```

- o -
- o X
- o -

```
bool b[9] = { 1, 0, 0,
             1, 0, 0,
             1, 0, 0 };
```

```
bool b2[3][3] = { { 1, 0, 0 },
                  { 1, 0, 0 },
                  { 1, 0, 0 } };
```

```
window2d win = convert::to<window2d>(b);
window2d win2 = convert::to<window2d>(b2);
```

2.6.1.4 Conversion between Neighborhoods and Windows

2.7 Sites, psites and dpoints

2.7.1 Need for site

```
c 0 1 2 3
r
+---+---+
0 | |x| | |
+---+---+
1 | | | | |
+---+---+

```

2.7.2 Need for psite

```
unsigned my_values(const mln::point2d& p)
{
    if (p.row() == 0)
        return 8;
    return 9;
}
```

```

p_array<point2d> arr;
arr.append(point2d(3, 6));
arr.append(point2d(3, 7));
arr.append(point2d(3, 8));
arr.append(point2d(4, 8));
arr.append(point2d(4, 9));

mln_VAR(ima, my_values | arr);

c  6 7 8 9
r
  +---+
3  | |x| |
  +---+
4   | | |
   +---+

arr[] = 0 1 2 3 4
        +---+
        | |x| | | |
        +---+

```

2.7.3 From psite to site

2.7.4 Dpoint

```

dpoint2d dp(-1,0);
point2d p(1,1);

std::cout << p + dp << std::endl;

(0,1)

```

2.8 Iterators

```

box2d b(3, 2);
mln_piter_(box2d) p(b);

for_all(p)
  std::cout << p; //prints every site coordinates.

(0,0) (0,1) (1,0) (1,1) (2,0) (2,1)

template <typename I>
void fill(I& ima, mln_value(I) v)
{
  mln_piter(I) p(ima.domain());
  for_all(p)
    ima(p) = v;
}

template <typename I, typename J>
void paste(const I& data, J& dest)
{
  mln_piter(I) p(data.domain());
  for_all(p)
    dest(p) = data(p);
}

```

Useful macros

2.9 Memory management

```

image2d<int> ima1(box2d(2, 3));
image2d<int> ima2;
point2d p(1,2);

ima2 = ima1; // ima1.id() == ima2.id()
// and both point to the same memory area.

ima2(p) = 2; // ima1 is modified as well.

// prints "2 - 2"
std::cout << ima2(p) << " - " << ima1(p) << std::endl;
// prints "true"
std::cout << (ima2.id_() == ima1.id_()) << std::endl;

image2d<int> ima1(5, 5);
image2d<int> ima3 = duplicate(ima1); // Makes a deep copy.

point2d p(2, 2);
ima3(p) = 3;

std::cout << ima3(p) << " - " << ima1(p) << std::endl;
std::cout << (ima3.id_() == ima1.id_()) << std::endl;

3 - 0
0

```

2.10 Basic routines

2.10.1 Fill

```

image2d<char> imga(5, 5);

data::fill(imga, 'a');

data::fill((imga | box2d(1,2)).rw(), 'a');

```

2.10.2 Paste

```

image2d<unsigned char> imgb(make::box2d(5,5, 7,8));
// Initialize imga with the same domain as imgb.
image2d<unsigned char> imga(imgb.domain());

// Initialize the image values.
data::fill(imgb, 'b');

// Paste the content of imgb in imga.
data::paste(imgb, imga);

debug::println(imga);

98 98 98 98
98 98 98 98
98 98 98 98

image2d<int> ima1(5, 5);

```

```

image2d<int> ima2(10, 10);

std::cout << "ima1.domain() = " << ima1.domain()
<< std::endl;
std::cout << "ima2.domain() = " << ima2.domain()
<< std::endl;

image2d<int> ima1(5, 5);
image2d<int> ima2(10, 10);

std::cout << "ima1.domain() = " << ima1.domain()
<< std::endl;
std::cout << "ima2.domain() = " << ima2.domain()
<< std::endl;

```

2.10.3 Blobs

```

bool vals[6][5] = {
    {0, 1, 1, 0, 0},
    {0, 1, 1, 0, 0},
    {0, 0, 0, 0, 0},
    {1, 1, 0, 1, 0},
    {1, 0, 1, 1, 1},
    {1, 0, 0, 0, 0}
};
image2d<bool> ima = make::image(vals);

label_8 nlabels;
image2d<label_8> lbl = labeling::blobs(ima, c4(), nlabels);

```

2.10.4 Logical not

```

bool vals[5][5] = {
    {1, 0, 1, 0, 0},
    {0, 1, 0, 1, 0},
    {1, 0, 1, 0, 0},
    {0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0}
};
image2d<bool> ima = make::image(vals);

image2d<bool> ima_neg = logical::not_(ima);

logical::not_inplace(ima);

```

2.10.5 Compute

2.10.5.1 Accumulators

```

data::compute(accu::meta::stat::max, ima);

data::compute(accu::meta::stat::max(), ima);

```

2.10.5.2 Example with labeling::compute()

```

bool vals[6][5] = {
    {0, 1, 1, 0, 0},

```

```

        {0, 1, 1, 0, 0},
        {0, 0, 0, 0, 0},
        {1, 1, 0, 1, 0},
        {1, 0, 1, 1, 1},
        {1, 0, 0, 0, 0}
    };
    image2d<bool> ima = make::image(vals);

    label_8 nlabels;
    image2d<label_8> lbl = labeling::blobs(ima, c4(), nlabels);

    util::array<box2d> boxes =
        labeling::compute(accu::meta::shape::bbox(),
                        lbl,
                        nlabels);

    for (unsigned i = 1; i <= nlabels; ++i)
        std::cout << boxes[i] << std::endl;

    [(0,1)..(1,2)]
    [(3,0)..(5,1)]
    [(3,2)..(4,4)]

    unsigned nsites = geom::nsites(ima);

```

2.10.6 Working with parts of an image

```

//function_p2b
bool my_function_p2b(mln::point2d p);

//function_p2v
//V is the value type used in the image.
template <typename V>
V my_function_p2v(mln::point2d p);

bool vals[6][5] = {
    {0, 1, 1, 0, 0},
    {0, 1, 1, 0, 0},
    {0, 0, 0, 0, 0},
    {1, 1, 0, 1, 0},
    {1, 0, 1, 1, 1},
    {1, 0, 0, 0, 0}
};
image2d<bool> ima = make::image(vals);

```

2.10.6.1 Restrict an image with a site set

```

p_array<point2d> arr;

// We add two points in the array.
arr.append(point2d(0, 1));
arr.append(point2d(4, 0));

// We restrict the image to the sites
// contained in arr and fill these ones
// with 0.
// We must call "rw()" here.
data::fill((ima | arr).rw(), 0);

debug::println((ima | arr));

```



```

mln_VAR(ima2, ima | arr);
// We do not need to call "rw()" here.
data::fill(ima2, 0);

```

```

-
-
-
-

```

2.10.6.2 Restrict an image with a predicate

```

label_8 nlabels;
image2d<label_8> lbl = labeling::blobs(ima, c4(), nlabels);

mln_VAR(lbl_2, lbl | (pw::value(lbl) == pw::cst(2u)));

image2d<rgb8> ima2;
initialize(ima2, ima);
data::fill(ima2, literal::black);

data::fill((ima2 | lbl_2.domain()).rw(), literal::red);

label_8 nlabels;
image2d<label_8> lab = labeling::blobs(ima, c4(), nlabels);

image2d<rgb8> ima2;
initialize(ima2, ima);
data::fill(ima2, literal::black);

data::fill((ima2 | (pw::value(lab) == pw::cst(2u))).rw(), literal::red);

```

2.10.6.3 Restrict an image with a C function

```

bool row_oddity(mln::point2d p)
{
    return p.row() % 2;
}

image2d<rgb8> ima2;
initialize(ima2, ima);
data::fill(ima2, literal::black);

data::fill((ima2 | row_oddity).rw(), literal::red);

ima | sub_D

0 1 0
1 1 1

mln_VAR(imab1, ima | (pw::value(ima) == pw::cst(1u)));

```

```

1
1 1 1

box2d b1(1,0, 1, 2);
mln_VAR(imac, imabl | b1);

// Print:
// 1 1 1
debug::println(imac);

box2d b2(0,0, 1, 1);
// Will fail at runtime.
// ima.domain().has((0,0)) is false.
mln_VAR(imad, imabl | b2);
debug::println(imad);

ima / sub_D

```

2.11 Input / Output

2.11.1 ImageMagick

2.11.2 GDCM

2.12 Graphs and images

2.12.1 Description

2.12.2 Example

```

      0 1 2 3 4
    .-----
0 | 0       2
1 |   \   / |
2 |     1   |
3 |       \ |
4 |         3-4

util::graph g;

for (unsigned i = 0; i < 5; ++i)
    g.add_vertex(); // Add vertex 'i';

g.add_edge(0, 1); // Associated to edge 0.
g.add_edge(1, 2); // Associated to edge 1.
g.add_edge(1, 3); // Associated to edge 2.
g.add_edge(3, 4); // Associated to edge 3.
g.add_edge(4, 2); // Associated to edge 4.

typedef fun::i2v::array<point2d> F;
F f(5); // We need to map 5 vertices.
f(0) = point2d(0, 0);
f(1) = point2d(2, 2);
f(2) = point2d(0, 4);
f(3) = point2d(4, 3);
f(4) = point2d(4, 4);

```

```

typedef p_vertices<util::graph, F> pv_t;
pv_t pv(g, f);

template <typename S>
struct viota_t : public mln::Function_v2v< viota_t<S> >
{
    typedef unsigned result;

    viota_t(unsigned size)
    {
        v_.resize(size);
        for(unsigned i = 0; i < size; ++i)
            v_[i] = 10 + i;
    }

    unsigned
    operator()(const mln_psite(S)& p) const
    {
        return v_[p.v().id()];
    }

protected:
    std::vector<result> v_;
};

// Constructs an image
viota_t<pv_t> viota(pv.nsites());
mln_VAR(graph_vertices_ima, viota | pv);

//Prints each vertex and its associated data.
mln_piter_(graph_vertices_ima_t) p(graph_vertices_ima.domain());
for_all(p)
    std::cout << "graph_vertices_ima(" << p << ") = "
                << graph_vertices_ima(p) << std::endl;

graph_vertices_ima((0,0)) = 10
graph_vertices_ima((2,2)) = 11
graph_vertices_ima((0,4)) = 12
graph_vertices_ima((4,3)) = 13
graph_vertices_ima((4,4)) = 14

// Function which maps sites to data.
viota_t viota(g.v_nmax());

// Iterator on vertices.
mln_vertex_iter_(util::graph) v(g);

// Prints each vertex and its associated value.
for_all(v)
    std::cout << v << " : " << viota(v) << std::endl;

0 : 10
1 : 11
2 : 12
3 : 13
4 : 14

// Iterator on vertices.
mln_vertex_iter_(util::graph) v(g);

// Iterator on v's edges.
mln_vertex_nbh_edge_iter_(util::graph) e(v);

```

```

// Prints the graph
// List all edges for each vertex.
for_all(v)
{
    std::cout << v << " : ";
    for_all(e)
        std::cout << e << " ";
    std::cout << std::endl;
}

0 : (0,1)
1 : (0,1) (1,2) (1,3)
2 : (1,2) (2,4)
3 : (1,3) (3,4)
4 : (3,4) (2,4)

// Iterator on edges.
mIn_edge_iter_(util::graph) e(g);

// Iterator on edges adjacent to e.
mIn_edge_nbh_edge_iter_(util::graph) ne(e);

// Prints the graph
// List all adjacent edges for each edge.
for_all(e)
{
    std::cout << e << " : ";
    for_all(ne)
        std::cout << ne << " ";
    std::cout << std::endl;
}

(0,1) : (1,2) (1,3)
(1,2) : (0,1) (1,3) (2,4)
(1,3) : (0,1) (1,2) (3,4)
(3,4) : (1,3) (2,4)
(2,4) : (1,2) (3,4)

// Iterator on vertices.
mIn_vertex_iter_(util::graph) v(g);

// Iterator on vertices adjacent to v.
mIn_vertex_nbh_vertex_iter_(util::graph) nv(v);

// Prints the graph
// List all adjacent edges for each edge.
for_all(v)
{
    std::cout << v << " : ";
    for_all(nv)
        std::cout << nv << " ";
    std::cout << std::endl;
}

0 : 1
1 : 0 2 3
2 : 1 4
3 : 1 4
4 : 3 2

```

2.13 Useful global variables

2.14 Useful macros

2.14.1 Variable declaration macros

2.14.2 Iterator type macros

2.14.2.1 Default iterator types

2.14.2.2 Forward iterator types

2.14.2.3 Backward iterators

2.14.2.4 Graph iterators

2.15 Common Compilation Errors

Chapter 3

Tutorial

- [Welcome](#)
- [Installation](#)
- [Getting started with Milena](#)
- [Data representation](#)
- [Load and save images](#)
- [Create your first image](#)
- [Read and write images](#)
- [Regions of interest](#)

3.1 Welcome

3.1.1 How to learn Milena

3.1.2 Obtaining the library

3.1.3 Downloading the library

3.1.3.1 Downloading from SVN

[Installation](#) [Directory structure](#) [Join the mailing lists](#)

3.1.3.2 Downloading packaged releases

[Installation](#) [Directory structure](#) [Join the mailing lists](#) [Documentation](#)

3.1.4 Join the mailing lists

3.1.5 Directory structure

3.1.6 Documentation

3.1.7 Community and Support

[Join the mailing lists](#) [Contacts](#)

3.1.8 Project status

[Join the mailing lists](#)

3.1.9 A brief history of Milena

3.1.10 Contacts

[Installation](#)

3.2 Installation

3.2.1 Bootstrap (SVN Sources)

[Configure](#)

3.2.2 Configure

3.2.3 Install

[Optional compilation](#) [Installation content](#)

3.2.4 Optional compilation

3.2.4.1 Examples

3.2.4.2 Tools

3.2.4.3 Tests

3.2.5 Installation content

[Welcome](#) [Getting started with Milena](#)

3.3 Getting started with Milena

3.3.1 Getting familiar with genericity

```
// Java or C -like code

void fill(image *ima, unsigned char v)
{
    for (int i = 0; i < ima->nrows; ++i)
        for (int j = 0; j < ima->ncols; ++j)
            ima->data[i][j] = v;
}

template <typename I>
void fill(I& ima, mln_value(I) v)
{
    mln_piter(I) p(ima.domain());
    for_all(p)
        ima(p) = v;
}

fill(ima, literal::green);

box2d b(20,20);
fill((ima | b).rw(), literal::green);
```

3.3.2 First generic algorithm

```
#include <mln/core/image/image2d.hh>
#include <mln/core/image/dmorph/image_if.hh>
#include <mln/core/alias/neighb2d.hh>

#include <mln/data/fill.hh>

#include <mln/labeling/blobs.hh>
#include <mln/labeling/compute.hh>
#include <mln/labeling/blobs.hh>

#include <mln/data/compare.hh>

#include <mln/util/array.hh>

#include <mln/value/label_8.hh>

#include <mln/accu/math/count.hh>

#include <mln/pw/all.hh>

#include <tests/data.hh>
#include <doc/tools/sample_utils.hh>

namespace mln
{
    template <typename I, typename N>
    mln_concrete(I)
    my_algorithm(const Image<I>& ima_,
                const Neighborhood<N>& nbh_)
    {
        trace::entering("my_algorithm");

        const I& ima = exact(ima_);
        const N& nbh = exact(nbh_);
```

```

mln_precondition(ima.is_valid());
mln_precondition(nbh.is_valid());

typedef value::label_8 V;
V nlabels;
mln_ch_value(I,V) lbl = labeling::blobs(ima, nbh, nlabels);
util::array<unsigned>
    count = labeling::compute(accu::meta::math::count(),
                              lbl,
                              nlabels);

mln_concrete(I) output;
initialize(output, ima);
data::fill(output, literal::one);

for (unsigned i = 1; i <= nlabels; ++i)
    if (count[i] < 10u)
        data::fill((output | (pw::value(lbl) == pw::cst(i))).rw(),
                    literal::zero);

trace::exiting("my_algorithm");
return output;
}

} // end of namespace mln

template <typename I, typename N>
mln_concrete(I)
my_algorithm(const Image<I>& ima_,
             const Neighborhood<N>& nbh_)

    trace::entering("my_algorithm");

```

Debug hints

```

const I& ima = exact(ima_);
const N& nbh = exact(nbh_);
mln_precondition(ima.is_valid());
mln_precondition(nbh.is_valid());

typedef value::label_8 V;
V nlabels;
mln_ch_value(I,V) lbl = labeling::blobs(ima, nbh, nlabels);
util::array<unsigned>
    count = labeling::compute(accu::meta::math::count(),
                              lbl,
                              nlabels);

mln_concrete(I) output;
initialize(output, ima);
data::fill(output, literal::one);

for (unsigned i = 1; i <= nlabels; ++i)
    if (count[i] < 10u)
        data::fill((output | (pw::value(lbl) == pw::cst(i))).rw(),
                    literal::zero);

trace::exiting("my_algorithm");
return output;

```

3.3.3 Compilation

3.3.3.1 Include path

3.3.3.2 Library linking

Input / Output

3.3.3.3 Disable Debug

3.3.3.4 Compiler optimization flags

3.3.3.4.1 GCC

3.3.3.4.2 Other compilers

3.3.4 Debug hints

3.3.4.1 Using assertions and GDB

3.3.4.2 Traces

```
// ...
trace::quiet = false;

labeling::blobs(ima, c4(), nlabels);

trace::quiet = true;

geom::bbox(ima);
// ...
```

3.3.4.3 Debug routines

```
image2d<int_u8> ima(5,5);
data::fill(ima, 2);
debug::println(ima);

2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2

image2d<int_u8> ima(5,5);
data::fill(ima, 2);
border::fill(ima, 7);
debug::println_with_border(ima);

7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7
7 7 7 2 2 2 2 7 7 7
7 7 7 2 2 2 2 7 7 7
7 7 7 2 2 2 2 7 7 7
```

```

7 7 7 2 2 2 2 7 7 7
7 7 7 2 2 2 2 7 7 7
7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7

```

```

int_u8 vals[25] = { 100, 100, 200, 200, 230,
                  100, 100, 200, 230, 230,
                  140, 140, 140,  0,  0,
                  65, 186, 65, 127, 127,
                  65,  65, 65, 127, 127 };

image2d<int_u8> ima = make::image2d(vals);
image2d<rgb8> ima_color = labeling::colorize(rgb8(), ima, 230);

```

Installation Data representation

3.4 Data representation

3.4.1 Sites

```

point2d p(3,3);
std::cout << p << std::endl;

```

```
(3,3)
```

3.4.2 Site sets

Site set

3.4.2.1 Creating a site set

```

box2d b(4,4);

(0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (1,3), (2,0), (2,1), (2,2), (2,3),
(3,0), (3,1), (3,2), (3,3),

mln_piter_(p_array<point2d>) p(arr);
for_all(p)
  std::cout << p << ", ";
std::cout << std::endl;

(-2,-2), (-2,-1), (-2,0), (-2,1), (-2,2), (-1,-2), (-1,-1), (-1,0), (-1,1), (-1,2),
(0,-2), (0,-1), (0,0), (0,1), (0,2), (1,-2), (1,-1), (1,0), (1,1), (1,2), (2,-2),
(2,-1), (2,0), (2,1), (2,2),

mln_piter_(box2d) p(b);
for_all(p)
  std::cout << p << ", ";
std::cout << std::endl;

(2,2), (1,2),

```

3.4.2.2 Getting access to sites

3.4.3 Images

3.4.3.1 Creating an image

3.4.3.2 Reading an image from a file

3.4.3.3 Accessing data

[Getting started with Milena Load and save images](#)

3.5 Load and save images

```

image2d<bool> ima;
io::pbm::load(ima, MLN_DOC_DIR "/img/small.pbm");

io::pbm::save(ima, MLN_DOC_DIR "/figures/ima_save.pbm");

```

[Load and save images Data representation Create your first image](#)

3.6 Create your first image

See also

[tuto2_first_image.cc](#)

```

bool vals[13][21] = {
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0},
    {0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0},
    {0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0},
    {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0},
    {0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
};

```

```

image2d<bool> ima = make::image(vals);

```

[Create an image](#)

```

debug::println(ima);

```

```

- - - - -
- | - | - | | | - | - - - | - - - - | - -
- | - | - | - - - | - - - | - - - | - | -
- | | | - | | | - | - - - | - - - | - | -
- | - | - | - - - | - - - | - - - | - | -

```

```

- | - | - | | | - | | | - | | | - - | - -
- - - - - - - - - - - - - - - - - - - -
- | - | - - | - - | | - - | - - - | | - -
- | - | - | - | - | - | - | - - - | - | -
- | - | - | - | - | | - - | - - - | - | -
- | | | - | - | - | - | - | - - - | - | -
- | - | - - | - - | - | - | | - | | - -
- - - - - - - - - - - - - - - - - - - -

```

```
doc::pbmsave(ima, "tuto2_first_image");
```

[Possible value types](#) [Load and save images](#) [Read and write images](#)

3.7 Read and write images

See also

[tuto3_rw_image.cc](#)

```

image2d<value::rgb8> ima(40, 40);

data::fill(ima, literal::red);

for (def::coord row = 20; row < 30; ++row)
  for (def::coord col = 20; col < 30; ++col)
    ima(point2d(row, col)) = literal::blue;

for (def::coord row = 20; row < 30; ++row)
  for (def::coord col = 20; col < 30; ++col)
    opt::at(ima, row, col) = literal::blue;

image2d<value::rgb8> lena;
io::ppm::load(lena, MLN_IMG_DIR "/small.ppm");

data::fill(ima, lena);

data::paste(ima, lena);

```

[Access and modify values](#) [Fill](#) [Paste](#) [Create your first image](#) [Regions of interest](#)

3.8 Regions of interest

See also

[tuto4_genericity_and_algorithms.cc](#)

```

image2d<value::rgb8> lena;
io::ppm::load(lena, MLN_IMG_DIR "/small.ppm");

namespace data
{
  template <typename I, typename D>
  void fill(Image<I>& ima, const D& data);
}

```

3.8.1 Image domain restricted by a site set

```
box2d roi = make::box2d(20, 20, 39, 39);

data::fill((lena | roi).rw(), literal::green);
```

3.8.2 Image domain restricted by a function

```
p_array<point2d> arr;
for (def::coord row = geom::min_row(lena); row < geom::max_row(lena); ++row)
  for (def::coord col = geom::min_col(lena); col < geom::max_col(lena); ++col)
    if (((row + col) % 2) == 0)
      arr.append(point2d(row, col));

for (def::coord row = geom::min_row(lena); row < geom::max_row(lena); ++row)
  for (def::coord col = geom::min_col(lena); col < geom::max_col(lena); ++col)
    if (((row + col) % 2) == 0)
      opt::at(lena, row, col) = literal::green;

data::fill((lena | fun::p2b::chess()).rw(), literal::green);
```

3.8.3 Image domain restricted by a mask

```
image2d<bool> mask;
initialize(mask, lena);
data::fill(mask, false);
data::fill((mask | make::box2d(10, 10, 14, 14)).rw(), true);
data::fill((mask | make::box2d(25, 15, 29, 18)).rw(), true);
data::fill((mask | make::box2d(50, 50, 54, 54)).rw(), true);

data::fill((lena | pw::value(mask)).rw(), literal::green);
```

3.8.4 Image domain restricted by a predicate

```
image2d<bool> lena_bw = binarization::binarization(lena, keep_specific_colors()
);
value::label_8 nlabels;
image2d<value::label_8> label = labeling::blobs(lena_bw, c8(), nlabels);

data::fill((lena | (pw::value(label) == pw::cst(0u))).rw(), literal::blue);
```

[Read and write images](#)

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

Types	98
Graphes	94
Images	95
Basic types	95
Image morphers	96
Values morphers	96
Domain morphers	97
Identity morphers	98
Neighborhoods	101
1D neighborhoods	101
2D neighborhoods	102
3D neighborhoods	104
Site sets	108
Basic types	108
Graph based	109
Complex based	109
Sparse types	109
Queue based	110
Utilities	110
Windows	111
1D windows	112
2D windows	113
3D windows	115
N-D windows	118
Multiple windows	118
Accumulators	99
On site sets	91
On images	91
On values	92
Multiple accumulators	94
Routines	99
Canvas	99

Functions	100
v2w2v functions	118
v2w_w2v functions	119
vv2b functions	119

Chapter 5

Namespace Index

5.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

mln (Mln/convert/to_image.hh)	121
mln::accu (Namespace of accumulators)	164
mln::accu::image (Namespace of accumulator image routines)	168
mln::accu::impl (Implementation namespace of accumulator namespace)	168
mln::accu::logic (Namespace of logical accumulators)	168
mln::accu::math (Namespace of mathematic accumulators)	168
mln::accu::meta::logic (Namespace of logical meta-accumulators)	169
mln::accu::meta::math (Namespace of mathematic meta-accumulators)	169
mln::accu::meta::shape (Namespace of shape meta-accumulators)	170
mln::accu::meta::stat (Namespace of statistical meta-accumulators)	170
mln::accu::shape (Namespace of shape accumulators)	171
mln::accu::stat (Namespace of statistical accumulators)	172
mln::algebra (Namespace of algebraic structure)	173
mln::arith (Namespace of arithmetic)	175
mln::arith::impl (Implementation namespace of arith namespace)	188
mln::arith::impl::generic (Generic implementation namespace of arith namespace)	188
mln::binarization (Namespace of "point-wise" expression tools)	189
mln::border (Namespace of routines related to image virtual (outer) border)	190
mln::border::impl (Implementation namespace of border namespace)	193
mln::border::impl::generic (Generic implementation namespace of border namespace)	194
mln::canvas (Namespace of canvas)	194
mln::canvas::browsing (Namespace of browsing canvas)	195
mln::canvas::impl (Implementation namespace of canvas namespace)	196
mln::canvas::labeling (Namespace of labeling canvas)	196
mln::canvas::labeling::impl (Implementation namespace of labeling canvas namespace)	197
mln::canvas::morpho (Namespace of morphological canvas)	197
mln::convert (Namespace of conversion routines)	198
mln::data (Namespace of image processing routines related to pixel data)	204
mln::data::approx (Namespace of image processing routines related to pixel levels with approximation)	216
mln::data::approx::impl (Implementation namespace of data::approx namespace)	217
mln::data::impl (Implementation namespace of data namespace)	217
mln::data::impl::generic (Generic implementation namespace of data namespace)	220

<code>mln::data::naive</code> (Namespace of image processing routines related to pixel levels with naive approach)	223
<code>mln::data::naive::impl</code> (Implementation namespace of <code>data::naive</code> namespace)	224
<code>mln::debug</code> (Namespace of routines that help to debug)	225
<code>mln::debug::impl</code> (Implementation namespace of debug namespace)	230
<code>mln::def</code> (Namespace for core definitions)	231
<code>mln::display</code> (Namespace of routines that help to display images)	232
<code>mln::display::impl</code> (Implementation namespace of display namespace)	232
<code>mln::display::impl::generic</code> (Generic implementation namespace of display namespace)	232
<code>mln::doc</code> (The namespace <code>mln::doc</code> is only for documentation purpose)	232
<code>mln::draw</code> (Namespace of drawing routines)	234
<code>mln::estim</code> (Namespace of estimation materials)	237
<code>mln::extension</code> (Namespace of extension tools)	239
<code>mln::fun</code> (Namespace of functions)	242
<code>mln::fun::access</code> (Namespace for access functions)	243
<code>mln::fun::i2v</code> (Namespace of integer-to-value functions)	244
<code>mln::fun::n2v</code> (Namespace of functions from nil to value)	244
<code>mln::fun::p2b</code> (Namespace of functions from point to boolean)	244
<code>mln::fun::p2p</code> (Namespace of functions from grid point to grid point)	245
<code>mln::fun::p2v</code> (Namespace of functions from point to value)	245
<code>mln::fun::stat</code> (Namespace of statistical functions)	245
<code>mln::fun::v2b</code> (Namespace of functions from value to logic value)	245
<code>mln::fun::v2i</code> (Namespace of value-to-integer functions)	246
<code>mln::fun::v2v</code> (Namespace of functions from value to value)	246
<code>mln::fun::v2w2v</code> (Namespace of bijective functions)	248
<code>mln::fun::v2w_w2v</code> (Namespace of functions from value to value)	248
<code>mln::fun::vv2b</code> (Namespace of functions from value to value)	248
<code>mln::fun::vv2v</code> (Namespace of functions from a couple of values to a value)	249
<code>mln::fun::x2p</code> (Namespace of functions from point to value)	250
<code>mln::fun::x2v</code> (Namespace of functions from vector to value)	250
<code>mln::fun::x2x</code> (Namespace of functions from vector to vector)	250
<code>mln::geom</code> (Namespace of all things related to geometry)	251
<code>mln::geom::impl</code> (Implementation namespace of geom namespace)	264
<code>mln::graph</code> (Namespace of graph related routines)	265
<code>mln::grid</code> (Namespace of grids definitions)	267
<code>mln::histo</code> (Namespace of histograms)	267
<code>mln::histo::impl</code> (Implementation namespace of histo namespace)	269
<code>mln::histo::impl::generic</code> (Generic implementation namespace of histo namespace)	269
<code>mln::impl</code> (Implementation namespace of mln namespace)	269
<code>mln::io</code> (Namespace of input/output handling)	269
<code>mln::io::cloud</code> (Namespace of cloud input/output handling)	271
<code>mln::io::dicom</code> (Namespace of DICOM input/output handling)	272
<code>mln::io::dump</code> (Namespace of dump input/output handling)	273
<code>mln::io::fits</code> (Namespace of fits input/output handling)	274
<code>mln::io::fld</code> (Namespace of pgm input/output handling)	275
<code>mln::io::magick</code> (Namespace of magick input/output handling)	276
<code>mln::io::off</code> (Namespace of off input/output handling)	278
<code>mln::io::pbm</code> (Namespace of pbm input/output handling)	279
<code>mln::io::pbm::impl</code> (Namespace of pbm implementation details)	281
<code>mln::io::pbms</code> (Namespace of pbms input/output handling)	281
<code>mln::io::pbms::impl</code> (Namespace of pbms implementation details)	282
<code>mln::io::pfm</code> (Namespace of pfm input/output handling)	282
<code>mln::io::pfm::impl</code> (Implementation namespace of pfm namespace)	283
<code>mln::io::pgm</code> (Namespace of pgm input/output handling)	284

mln::io::pgms (Namespace of pgms input/output handling)	285
mln::io::plot (Namespace of plot input/output handling)	285
mln::io::pnm (Namespace of pnm input/output handling)	287
mln::io::pnm::impl (Namespace of pnm's implementation details)	289
mln::io::pnms (Namespace of pnms input/output handling)	289
mln::io::ppm (Namespace of ppm input/output handling)	290
mln::io::ppms (Namespace of ppms input/output handling)	292
mln::io::raw (Namespace of raw input/output handling)	292
mln::io::tiff (Namespace of tiff input/output handling)	294
mln::io::txt (Namespace of txt input/output handling)	294
mln::labeling (Namespace of labeling routines)	295
mln::labeling::impl (Implementation namespace of labeling namespace)	310
mln::labeling::impl::generic (Generic implementation namespace of labeling namespace)	312
mln::linear (Namespace of linear image processing routines)	314
mln::linear::impl (Namespace of linear image processing routines implementation details)	318
mln::linear::local (Specializations of local linear routines)	318
mln::linear::local::impl (Namespace of local linear routines implementation details)	320
mln::literal (Namespace of literals)	320
mln::logical (Namespace of logic)	326
mln::logical::impl (Implementation namespace of logical namespace)	329
mln::logical::impl::generic (Generic implementation namespace of logical namespace)	330
mln::make (Namespace of routines that help to make Milena's objects)	330
mln::math (Namespace of mathematical routines)	355
mln::metal (Namespace of meta-programming tools)	356
mln::metal::impl (Implementation namespace of metal namespace)	356
mln::metal::math (Namespace of static mathematical functions)	357
mln::metal::math::impl (Implementation namespace of metal::math namespace)	357
mln::morpho (Namespace of mathematical morphology routines)	357
mln::morpho::approx (Namespace of approximate mathematical morphology routines)	367
mln::morpho::attribute (Namespace of attributes used in mathematical morphology)	367
mln::morpho::closing::approx (Namespace of approximate mathematical morphology closing routines)	368
mln::morpho::elementary (Namespace of image processing routines of elementary mathematical morphology)	368
mln::morpho::impl (Namespace of mathematical morphology routines implementations)	370
mln::morpho::impl::generic (Namespace of mathematical morphology routines generic implementations)	371
mln::morpho::opening::approx (Namespace of approximate mathematical morphology opening routines)	371
mln::morpho::reconstruction (Namespace of morphological reconstruction routines)	372
mln::morpho::reconstruction::by_dilation (Namespace of morphological reconstruction by dilation routines)	372
mln::morpho::reconstruction::by_erosion (Namespace of morphological reconstruction by erosion routines)	372
mln::morpho::tree (Namespace of morphological tree-related routines)	372
mln::morpho::tree::filter (Namespace for attribute filtering)	380
mln::morpho::watershed (Namespace of morphological watershed routines)	382
mln::morpho::watershed::watershed (Namespace of morphological watershed routines implementations)	385
mln::morpho::watershed::watershed::generic (Namespace of morphological watershed routines generic implementations)	385
mln::norm (Namespace of norms)	385
mln::norm::impl (Implementation namespace of norm namespace)	387
mln::opt (Namespace of optional routines)	388

mln::opt::impl (Implementation namespace of opt namespace)	389
mln::pw (Namespace of "point-wise" expression tools)	389
mln::registration (Namespace of "point-wise" expression tools)	390
mln::select (Select namespace (FIXME doc))	393
mln::set (Namespace of image processing routines related to pixel sets)	393
mln::subsampling (Namespace of "point-wise" expression tools)	396
mln::tag (Namespace of image processing routines related to tags)	397
mln::test (Namespace of image processing routines related to pixel tests)	397
mln::test::impl (Implementation namespace of test namespace)	399
mln::topo (Namespace of "point-wise" expression tools)	399
mln::trace (Namespace of routines related to the trace mechanism)	410
mln::trait (Namespace where traits are defined)	410
mln::transform (Namespace of transforms)	410
mln::util (Namespace of tools using for more complex algorithm)	415
mln::util::impl (Implementation namespace of util namespace)	422
mln::value (Namespace of materials related to pixel value types)	422
mln::value::impl (Implementation namespace of value namespace)	433
mln::win (Namespace of image processing routines related to win)	433

Chapter 6

Class Index

6.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

mln::Generalized_Pixel< bkd_pixter1d< I > >	727
mln::Pixel_Iterator< bkd_pixter1d< I > >	942
pixel_iterator_base_< I, bkd_pixter1d< I > >	
backward_pixel_iterator_base_< I, bkd_pixter1d< I > >	
mln::bkd_pixter1d< I >	547
mln::Generalized_Pixel< bkd_pixter2d< I > >	727
mln::Pixel_Iterator< bkd_pixter2d< I > >	942
pixel_iterator_base_< I, bkd_pixter2d< I > >	
backward_pixel_iterator_base_< I, bkd_pixter2d< I > >	
mln::bkd_pixter2d< I >	548
mln::Generalized_Pixel< bkd_pixter3d< I > >	727
mln::Pixel_Iterator< bkd_pixter3d< I > >	942
pixel_iterator_base_< I, bkd_pixter3d< I > >	
backward_pixel_iterator_base_< I, bkd_pixter3d< I > >	
mln::bkd_pixter3d< I >	550
mln::Generalized_Pixel< dpoints_bkd_pixter< I > >	727
mln::Pixel_Iterator< dpoints_bkd_pixter< I > >	942
mln::dpoints_bkd_pixter< I >	649
mln::Generalized_Pixel< dpoints_fwd_pixter< I > >	727
mln::Pixel_Iterator< dpoints_fwd_pixter< I > >	942
mln::dpoints_fwd_pixter< I >	651
mln::Generalized_Pixel< fwd_pixter1d< I > >	727
mln::Pixel_Iterator< fwd_pixter1d< I > >	942
pixel_iterator_base_< I, fwd_pixter1d< I > >	
forward_pixel_iterator_base_< I, fwd_pixter1d< I > >	
mln::fwd_pixter1d< I >	721
mln::Generalized_Pixel< fwd_pixter2d< I > >	727
mln::Pixel_Iterator< fwd_pixter2d< I > >	942
pixel_iterator_base_< I, fwd_pixter2d< I > >	
forward_pixel_iterator_base_< I, fwd_pixter2d< I > >	
mln::fwd_pixter2d< I >	722

mln::Generalized_Pixel< fwd_pixter3d< I > >	727
mln::Pixel_Iterator< fwd_pixter3d< I > >	942
pixel_iterator_base_< I, fwd_pixter3d< I > >	
forward_pixel_iterator_base_< I, fwd_pixter3d< I > >	
mln::fwd_pixter3d< I >	724
mln::Generalized_Pixel< pixel< I > >	727
mln::pixel< I >	941
mln::internal::image_base< algebra::vec< n, I::value >, I::domain_t, stack_image< n, I > >	
image_morpher< I, algebra::vec< n, I::value >, I::domain_t, stack_image< n, I > >	
image_value_morpher< I, algebra::vec< n, I::value >, stack_image< n, I > >	
mln::value::stack_image< n, I >	1143
mln::internal::image_base< const I::value, I::domain_t, E >	
image_morpher< const I, const I::value, I::domain_t, E >	
image_identity< const I, I::domain_t, E >	
mln::labeled_image_base< I, E >	795
mln::internal::image_base< const I::value, I::domain_t, labeled_image< I > >	
image_morpher< const I, const I::value, I::domain_t, labeled_image< I > >	
image_identity< const I, I::domain_t, labeled_image< I > >	
mln::labeled_image_base< I, labeled_image< I > >	795
mln::labeled_image< I >	791
mln::internal::image_base< F::result, I1::domain_t, thrubin_image< I1, I2, F > >	
image_morpher< I1, F::result, I1::domain_t, thrubin_image< I1, I2, F > >	
image_value_morpher< I1, F::result, thrubin_image< I1, I2, F > >	
mln::thrubin_image< I1, I2, F >	977
mln::internal::image_base< F::result, I::domain_t, fun_image< F, I > >	
image_morpher< I, F::result, I::domain_t, fun_image< F, I > >	
image_value_morpher< I, F::result, fun_image< F, I > >	
mln::fun_image< F, I >	714
mln::internal::image_base< F::result, S, E >	
image_primary< F::result, S, E >	
mln::internal::image_base< F::result, S, image< F, S > >	
image_primary< F::result, S, image< F, S > >	
image_base< F, S, image< F, S > >	
mln::pw::image< F, S >	960
mln::internal::image_base< fun::i2v::array< V >::result, p_edges< G, internal::efsite_selector< P, G >::site_function_t >, edge_image< P, V, G > >	
image_primary< fun::i2v::array< V >::result, p_edges< G, internal::efsite_selector< P, G >::site_function_t >, edge_image< P, V, G > >	
image_base< fun::i2v::array< V >, p_edges< G, internal::efsite_selector< P, G >::site_function_t >, edge_image< P, V, G > >	
mln::edge_image< P, V, G >	657
mln::internal::image_base< fun::i2v::array< V >::result, p_vertices< G, internal::vfsite_selector< P, G >::site_function_t >, vertex_image< P, V, G > >	
image_primary< fun::i2v::array< V >::result, p_vertices< G, internal::vfsite_selector< P, G >::site_function_t >, vertex_image< P, V, G > >	
image_base< fun::i2v::array< V >, p_vertices< G, internal::vfsite_selector< P, G >::site_function_t >, vertex_image< P, V, G > >	
mln::vertex_image< P, V, G >	1151
mln::internal::image_base< I::value, box2d, slice_image< I > >	
image_morpher< I, I::value, box2d, slice_image< I > >	
image_domain_morpher< I, box2d, slice_image< I > >	
mln::slice_image< I >	971
mln::internal::image_base< I::value, box2d_h, hexa< I > >	

image_morpher< I, I::value, box2d_h, hexa< I > >	
image_domain_morpher< I, box2d_h, hexa< I > >	
mln::hexa< I >	760
mln::internal::image_base< I::value, box< I::site >, extended< I > >	
image_morpher< I, I::value, box< I::site >, extended< I > >	
image_domain_morpher< I, box< I::site >, extended< I > >	
mln::extended< I >	659
mln::internal::image_base< I::value, D, unproject_image< I, D, F > >	
image_morpher< I, I::value, D, unproject_image< I, D, F > >	
image_domain_morpher< I, D, unproject_image< I, D, F > >	
mln::unproject_image< I, D, F >	1034
mln::internal::image_base< I::value, I::domain_t, decorated_image< I, D > >	
image_morpher< I, I::value, I::domain_t, decorated_image< I, D > >	
image_identity< I, I::domain_t, decorated_image< I, D > >	
mln::decorated_image< I, D >	595
mln::internal::image_base< I::value, I::domain_t, extension_fun< I, F > >	
image_morpher< I, I::value, I::domain_t, extension_fun< I, F > >	
image_identity< I, I::domain_t, extension_fun< I, F > >	
mln::extension_fun< I, F >	661
mln::internal::image_base< I::value, I::domain_t, extension_ima< I, J > >	
image_morpher< I, I::value, I::domain_t, extension_ima< I, J > >	
image_identity< I, I::domain_t, extension_ima< I, J > >	
mln::extension_ima< I, J >	663
mln::internal::image_base< I::value, I::domain_t, extension_val< I > >	
image_morpher< I, I::value, I::domain_t, extension_val< I > >	
image_identity< I, I::domain_t, extension_val< I > >	
mln::extension_val< I >	666
mln::internal::image_base< I::value, I::domain_t, interpolated< I, F > >	
image_morpher< I, I::value, I::domain_t, interpolated< I, F > >	
image_identity< I, I::domain_t, interpolated< I, F > >	
mln::interpolated< I, F >	786
mln::internal::image_base< I::value, I::domain_t, p2p_image< I, F > >	
image_morpher< I, I::value, I::domain_t, p2p_image< I, F > >	
image_domain_morpher< I, I::domain_t, p2p_image< I, F > >	
mln::p2p_image< I, F >	847
mln::internal::image_base< I::value, I::domain_t, plain< I > >	
image_morpher< I, I::value, I::domain_t, plain< I > >	
image_identity< I, I::domain_t, plain< I > >	
mln::plain< I >	943
mln::internal::image_base< I::value, I::domain_t, safe_image< I > >	
image_morpher< I, I::value, I::domain_t, safe_image< I > >	
image_identity< I, I::domain_t, safe_image< I > >	
mln::safe_image< I >	963
mln::internal::image_base< I::value, p_if< I::domain_t, F >, image_if< I, F > >	
image_morpher< I, I::value, p_if< I::domain_t, F >, image_if< I, F > >	
image_domain_morpher< I, p_if< I::domain_t, F >, image_if< I, F > >	
mln::image_if< I, F >	784
mln::internal::image_base< I::value, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > >	
image_morpher< I, I::value, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > >	
image_domain_morpher< I, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > >	
mln::sub_image_if< I, S >	975
mln::internal::image_base< I::value, S, E >	
image_morpher< I, I::value, S, E >	
mln::internal::image_base< I::value, S, sub_image< I, S > >	

```

    image_morpher< I, I::value, S, sub_image< I, S > >
    image_domain_morpher< I, S, sub_image< I, S > >
    mln::sub_image< I, S > . . . . . 974
mln::internal::image_base< I::value, S, tr_image< S, I, T > >
    image_morpher< I, I::value, S, tr_image< S, I, T > >
    image_identity< I, S, tr_image< S, I, T > >
    mln::tr_image< S, I, T > . . . . . 1031
mln::internal::image_base< image2d< V >::value, box2d_h, hexa< image2d< V > > >
    image_morpher< image2d< V >, image2d< V >::value, box2d_h, hexa< image2d< V > > >
    image_domain_morpher< image2d< V >, box2d_h, hexa< image2d< V > > >
    mln::hexa< image2d< V > > . . . . . 760
    mln::image2d_h< V > . . . . . 776
mln::internal::image_base< mln::trait::ch_value< I, F::result >::ret::value, I::domain_t, lazy_image<
I, F, B > >
    image_morpher< mln::trait::ch_value< I, F::result >::ret, mln::trait::ch_value< I, F::result
>::ret::value, I::domain_t, lazy_image< I, F, B > >
    image_identity< mln::trait::ch_value< I, F::result >::ret, I::domain_t, lazy_image< I, F, B >
>
    mln::lazy_image< I, F, B > . . . . . 798
mln::internal::image_base< T, box1d, image1d< T > >
    image_primary< T, box1d, image1d< T > >
    mln::image1d< T > . . . . . 766
mln::internal::image_base< T, box3d, image3d< T > >
    image_primary< T, box3d, image3d< T > >
    mln::image3d< T > . . . . . 779
mln::internal::image_base< T, I::domain_t, E >
    image_morpher< I, T, I::domain_t, E >
mln::internal::image_base< T, I::domain_t, violent_cast_image< T, I > >
    image_morpher< I, T, I::domain_t, violent_cast_image< T, I > >
    image_value_morpher< I, T, violent_cast_image< T, I > >
    mln::violent_cast_image< T, I > . . . . . 1153
mln::internal::image_base< T, mln::box2d, image2d< T > >
    image_primary< T, mln::box2d, image2d< T > >
    mln::image2d< T > . . . . . 771
mln::internal::image_base< T, S, flat_image< T, S > >
    image_primary< T, S, flat_image< T, S > >
    mln::flat_image< T, S > . . . . . 671
mln::internal::image_base< V, p_complex< D, G >, complex_image< D, G, V > >
    image_primary< V, p_complex< D, G >, complex_image< D, G, V > >
    mln::complex_image< D, G, V > . . . . . 583
mln::internal::check::image_fastest< complex_image< D, G, V >, mln::metal::equal< mln_trait_
image_speed< complex_image< D, G, V >, trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< decorated_image< I, D >, mln::metal::equal< mln_trait_
image_speed< decorated_image< I, D >, trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< E, mln::metal::equal< mln_trait_image_speed(E),
trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< edge_image< P, V, G >, mln::metal::equal< mln_trait_image_
speed< edge_image< P, V, G >, trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< extended< I >, mln::metal::equal< mln_trait_image_
speed< extended< I >, trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< extension_fun< I, F >, mln::metal::equal< mln_trait_image_
speed< extension_fun< I, F >, trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest< extension_ima< I, J >, mln::metal::equal< mln_trait_image_
speed< extension_ima< I, J >, trait::image::speed::fastest >::eval >

```

```

mln::internal::check::image_fastest_< extension_val< I >, mln::metal::equal< mln_trait_image_
speed(extension_val< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< flat_image< T, S >, mln::metal::equal< mln_trait_image_
speed(flat_image< T, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< fun_image< F, I >, mln::metal::equal< mln_trait_image_
speed(fun_image< F, I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< hexa< I >, mln::metal::equal< mln_trait_image_speed(hexa<
I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< hexa< image2d< V > >, mln::metal::equal< mln_trait_
image_speed(hexa< image2d< V > >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image1d< T >, mln::metal::equal< mln_trait_image_
speed(image1d< T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image2d< T >, mln::metal::equal< mln_trait_image_
speed(image2d< T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image3d< T >, mln::metal::equal< mln_trait_image_
speed(image3d< T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image< F, S >, mln::metal::equal< mln_trait_image_
speed(image< F, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image_if< I, F >, mln::metal::equal< mln_trait_image_
speed(image_if< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< interpolated< I, F >, mln::metal::equal< mln_trait_image_
speed(interpolated< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< labeled_image< I >, mln::metal::equal< mln_trait_image_
speed(labeled_image< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< lazy_image< I, F, B >, mln::metal::equal< mln_trait_image_
speed(lazy_image< I, F, B >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< p2p_image< I, F >, mln::metal::equal< mln_trait_image_
speed(p2p_image< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< plain< I >, mln::metal::equal< mln_trait_image_speed(plain<
I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< safe_image< I >, mln::metal::equal< mln_trait_image_
speed(safe_image< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< slice_image< I >, mln::metal::equal< mln_trait_image_
speed(slice_image< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< stack_image< n, I >, mln::metal::equal< mln_trait_image_
speed(stack_image< n, I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< sub_image< I, S >, mln::metal::equal< mln_trait_image_
speed(sub_image< I, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< sub_image_if< I, S >, mln::metal::equal< mln_trait_image_
speed(sub_image_if< I, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< thrubin_image< I1, I2, F >, mln::metal::equal< mln_trait_
image_speed(thrubin_image< I1, I2, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< tr_image< S, I, T >, mln::metal::equal< mln_trait_image_
speed(tr_image< S, I, T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< unproject_image< I, D, F >, mln::metal::equal< mln_trait_
image_speed(unproject_image< I, D, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< vertex_image< P, V, G >, mln::metal::equal< mln_trait_
image_speed(vertex_image< P, V, G >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< violent_cast_image< T, I >, mln::metal::equal< mln_trait_
image_speed(violent_cast_image< T, I >), trait::image::speed::fastest >::eval >
mln::internal::impl_selector< W::center_t, W::psite, graph_window_piter< S, W, I > >
mln::graph_window_piter< S, W, I > . . . . . 757
mln::value::Integer< graylevel< n > > . . . . . 1126

```

mln::value::graylevel< n >	1113
mln::value::Integer< int_s< n > >	1126
mln::value::int_s< n >	1119
mln::value::Integer< int_u< n > >	1126
mln::value::int_u< n >	1122
mln::value::Integer< int_u_sat< n > >	1126
mln::value::int_u_sat< n >	1124
mln::value::Integer< object_id< Tag, V > >	1126
mln::util::object_id< Tag, V >	1076
mln::internal::is_masked_impl_selector< S, W::mask_t::domain_t, graph_window_if_piter< S, W, I > >	
mln::graph_window_if_piter< S, W, I >	755
mln::algebra::h_mat< d, T >	543
mln::algebra::h_vec< d, C >	545
mln::canvas::chamfer< F >	582
mln::category< R(*) (A) >	582
mln::Delta_Point_Site< void >	599
mln::doc::Accumulator< E >	599
mln::doc::Generalized_Pixel< E >	614
mln::doc::Pixel_Iterator< E >	625
mln::doc::Object< E >	624
mln::doc::Dpoint< E >	603
mln::doc::Image< E >	615
mln::doc::Fastest_Image< E >	606
mln::doc::Iterator< E >	621
mln::doc::Pixel_Iterator< E >	625
mln::doc::Site_Iterator< E >	631
mln::doc::Value_Iterator< E >	635
mln::doc::Neighborhood< E >	623
mln::doc::Site_Set< E >	633
mln::doc::Box< E >	600
mln::doc::Value_Set< E >	636
mln::doc::Weighted_Window< E >	639
mln::doc::Window< E >	641
mln::doc::Point_Site< E >	628
mln::Edge< E >	657
mln::fun::from_accu< A >	673
mln::fun::internal::ch_function_value_impl< F, V >	
mln::fun::v2v::ch_function_value< F, V >	679
mln::fun::x2p::closest_point< P >	704
mln::fun::x2x::composed< T2, T1 >	706
mln::Function< void >	717
mln::Gdpoint< void >	726
mln::Generalized_Pixel< E >	727
mln::Pixel_Iterator< E >	942
mln::internal::pixel_iterator_base_	
mln::geom::complex_geometry< D, P >	727
mln::graph::attribute::card_t	734
mln::graph::attribute::representative_t	734
mln::histo::array< T >	764
mln::internal::image_base< T, S, E >	

mln::internal::neighborhood_base< W, E >	
mln::internal::neighb_base	
mln::neighb< graph_elt_mixed_window< G, S, S2 > >	844
mln::graph_elt_mixed_neighborhood< G, S, S2 >	735
mln::neighb< graph_elt_window< G, S > >	844
mln::graph_elt_neighborhood< G, S >	740
mln::neighb< graph_elt_window_if< G, S, I > >	844
mln::graph_elt_neighborhood_if< G, S, I >	742
mln::internal::pixel_impl_< I, E >	
mln::internal::pixel_iterator_base_	
mln::io::dicom::dicom_header	788
mln::io::dump::dump_header	788
mln::io::fld::fld_header	789
mln::io::raw::raw_header	789
mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 >	830
mln::metal::bool_< false >	
mln::metal::equal< T1::coord, T2::coord >	830
mln::metal::equal< T1::point, T2::point >	830
mln::metal::equal< T1, T2 >	830
mln::metal::converts_to< T, U >	830
mln::metal::goes_to< T, U >	831
mln::metal::is< T, U >	831
mln::metal::is_a< T, M >	832
mln::metal::is_not< T, U >	832
mln::metal::is_not_a< T, M >	832
mln::Neighborhood< void >	846
mln::Object< E >	846
mln::Browsing< E >	567
mln::Delta_Point_Site< E >	598
mln::Dpoint< E >	643
mln::Function< E >	717
mln::Function_n2v< E >	718
mln::Function_v2v< E >	719
mln::fun::x2v::bilinear< I >	705
mln::fun::x2v::trilinear< I >	706
mln::fun::x2x::linear< I >	707
mln::Function_v2b< E >	718
mln::Function_vv2b< E >	720
mln::Function_vv2v< E >	720
mln::Gdpoint< E >	725
mln::Graph< E >	733
mln::Image< E >	764
mln::Iterator< E >	789
mln::Pixel_Iterator< E >	942
mln::topo::internal::complex_iterator_base	
mln::topo::internal::complex_relative_iterator_base	
mln::Value_Iterator< E >	1147
mln::Literal< E >	801
mln::Mesh< E >	825
mln::Regular_Grid< E >	962
mln::Meta_Accumulator< E >	826
mln::Meta_Function< E >	828
mln::Meta_Function_v2v< E >	829

mln::Meta_Function_vv2v< E >	829
mln::Neighborhood< E >	846
mln::Point_Site< E >	955
mln::Proxy< E >	959
mln::Accumulator< E >	541
mln::accu::internal::base	
couple< accu::shape::bbox< P >, accu::math::count< P >, float, rectangularity< P >>	
mln::accu::site_set::rectangularity< P >	508
mln::accu::pair< min< V >, max< V >>	498
mln::accu::stat::min_max< V >	526
mln::Site_Proxy< E >	967
mln::Pseudo_Site< E >	959
mln::Site_Iterator< E >	965
mln::internal::site_iterator_base	
mln::internal::site_set_iterator_base	
mln::p_transformed_piter< Pi, S, F >	929
mln::Site< E >	964
mln::Gpoint< E >	729
mln::Site_Set< E >	968
mln::Box< E >	559
mln::Value< E >	1108
mln::Value_Set< E >	1149
mln::Weighted_Window< E >	1159
mln::Window< E >	1171
mln::graph_window_base< P, E >	753
mln::internal::window_base	
mln::Point_Site< void >	958
mln::Proxy< void >	959
mln::Pseudo_Site< void >	960
mln::registration::closest_point_basic< P >	961
mln::registration::closest_point_with_map< P >	962
mln::select::p_of< P >	963
mln::Site< void >	965
mln::Site_Proxy< void >	968
mln::Site_Set< void >	971
mln::thru_image< I, F >	977
mln::topo::complex< D >	1006
mln::topo::face< D >	1009
mln::topo::algebraic_face< D >	992
mln::topo::is_simple_2d_t< N >	1017
mln::topo::n_face< N, D >	1020
mln::topo::algebraic_n_face< N, D >	998
mln::topo::n_faces_set< N, D >	1027
mln::topo::skeleton::is_simple_point< N >	1028
mln::util::adjacency_matrix< V >	1036
mln::util::array< T >	1037
mln::util::branch< T >	1044
mln::util::branch_iter< T >	1045
mln::util::branch_iter_ind< T >	1047
mln::util::greater_point< I >	1066
mln::util::greater_psite< I >	1067

mln::util::head< T, R >	1067
mln::util::ilcell< T >	1068
mln::util::internal::edge_impl_< G >	
mln::util::edge< G >	1051
mln::util::internal::vertex_impl_< G >	
mln::util::vertex< G >	1103
mln::util::node< T, R >	1076
mln::util::ord< T >	1078
mln::util::pix< I >	1081
mln::util::tracked_ptr< T >	1094
mln::util::tree< T >	1096
mln::util::tree_node< T >	1098
mln::value::float01	1110
mln::value::Integer< E >	1126
mln::value::Integer< void >	1127
mln::value::internal::value_like_< V, C, N, E >	
mln::value::float01_f	1112
mln::value::graylevel< n >	1113
mln::value::graylevel_f	1117
mln::value::int_s< n >	1119
mln::value::int_u< n >	1122
mln::value::int_u_sat< n >	1124
mln::value::label< n >	1127
mln::value::qt::rgb32	1136
mln::value::rgb< n >	1138
mln::value::set< T >	1140
mln::value::sign	1141
mln::value::super_value< sign >	1146
mln::value::value_array< T, V >	1146
mln::Vertex< E >	1150
mln::internal::neighborhood_base< W, mixed_neighb< W > >	
neighb_base< W, mixed_neighb< W > >	
mln::mixed_neighb< W >	832
mln::internal::neighborhood_base< W, neighb< W > >	
neighb_base< W, neighb< W > >	
mln::neighb< W >	844
mln::Object< abs >	846
mln::Meta_Function< abs >	828
mln::Meta_Function_v2v< abs >	829
mln::Object< abs< V > >	846
mln::Function< abs< V > >	717
mln::Function_v2v< abs< V > >	719
mln::Object< accu_result >	846
mln::Meta_Function< accu_result >	828
mln::Meta_Function_v2v< accu_result >	829
mln::Object< adj_higher_dim_connected_n_face_bkd_iter< D > >	846
mln::Iterator< adj_higher_dim_connected_n_face_bkd_iter< D > >	789
complex_iterator_base< algebraic_face< D >, adj_higher_dim_connected_n_face_bkd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_bkd_iter< D > >	
backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_bkd_iter< D > >	

mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D >	978
mln::Object< adj_higher_dim_connected_n_face_fwd_iter< D > >	846
mln::Iterator< adj_higher_dim_connected_n_face_fwd_iter< D > >	789
complex_iterator_base< algebraic_face< D >, adj_higher_dim_connected_n_face_fwd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_fwd_iter< D > >	
forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_fwd_iter< D > >	
mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D >	980
mln::Object< adj_higher_face_bkd_iter< D > >	846
mln::Iterator< adj_higher_face_bkd_iter< D > >	789
complex_iterator_base< algebraic_face< D >, adj_higher_face_bkd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_bkd_iter< D > >	
backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_bkd_iter< D > >	
mln::topo::adj_higher_face_bkd_iter< D >	981
mln::Object< adj_higher_face_fwd_iter< D > >	846
mln::Iterator< adj_higher_face_fwd_iter< D > >	789
complex_iterator_base< algebraic_face< D >, adj_higher_face_fwd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_fwd_iter< D > >	
forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_fwd_iter< D > >	
mln::topo::adj_higher_face_fwd_iter< D >	982
mln::Object< adj_lower_dim_connected_n_face_bkd_iter< D > >	846
mln::Iterator< adj_lower_dim_connected_n_face_bkd_iter< D > >	789
complex_iterator_base< algebraic_face< D >, adj_lower_dim_connected_n_face_bkd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_bkd_iter< D > >	
backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_bkd_iter< D > >	
mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D >	983
mln::Object< adj_lower_dim_connected_n_face_fwd_iter< D > >	846
mln::Iterator< adj_lower_dim_connected_n_face_fwd_iter< D > >	789
complex_iterator_base< algebraic_face< D >, adj_lower_dim_connected_n_face_fwd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_fwd_iter< D > >	
forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_fwd_iter< D > >	
mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D >	984
mln::Object< adj_lower_face_bkd_iter< D > >	846
mln::Iterator< adj_lower_face_bkd_iter< D > >	789
complex_iterator_base< algebraic_face< D >, adj_lower_face_bkd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_face_bkd_iter< D > >	
backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_face_bkd_iter< D > >	
mln::topo::adj_lower_face_bkd_iter< D >	985

mln::Object< adj_lower_face_fwd_iter< D > >	846
mln::Iterator< adj_lower_face_fwd_iter< D > >	789
complex_iterator_base< algebraic_face< D >, adj_lower_face_fwd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_face_- fwd_iter< D > >	
forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_- lower_face_fwd_iter< D > >	
mln::topo::adj_lower_face_fwd_iter< D >	986
mln::Object< adj_lower_higher_face_bkd_iter< D > >	846
mln::Iterator< adj_lower_higher_face_bkd_iter< D > >	789
complex_relative_iterator_sequence< adj_higher_face_bkd_iter< D >, adj_lower_face_bkd_- iter< D >, adj_lower_higher_face_bkd_iter< D > >	
mln::topo::adj_lower_higher_face_bkd_iter< D >	987
mln::Object< adj_lower_higher_face_fwd_iter< D > >	846
mln::Iterator< adj_lower_higher_face_fwd_iter< D > >	789
complex_relative_iterator_sequence< adj_lower_face_fwd_iter< D >, adj_higher_face_fwd_- iter< D >, adj_lower_higher_face_fwd_iter< D > >	
mln::topo::adj_lower_higher_face_fwd_iter< D >	989
mln::Object< adj_m_face_bkd_iter< D > >	846
mln::Iterator< adj_m_face_bkd_iter< D > >	789
complex_iterator_base< algebraic_face< D >, adj_m_face_bkd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_- bkd_iter< D > >	
backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_bkd_iter< D > >	
mln::topo::adj_m_face_bkd_iter< D >	990
mln::Object< adj_m_face_fwd_iter< D > >	846
mln::Iterator< adj_m_face_fwd_iter< D > >	789
complex_iterator_base< algebraic_face< D >, adj_m_face_fwd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_- fwd_iter< D > >	
forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_- m_face_fwd_iter< D > >	
mln::topo::adj_m_face_fwd_iter< D >	991
mln::Object< all_to< T > >	846
mln::Function< all_to< T > >	717
mln::Function_v2v< all_to< T > >	719
mln::Object< antilogy >	846
mln::Function< antilogy >	717
mln::Function_v2v< antilogy >	719
mln::Function_v2b< antilogy >	718
mln::fun::p2b::antilogy	674
mln::Object< array1d< T, Size > >	846
mln::Object< array2d< T, r, c > >	846
mln::Object< array3d< T, s, r, c > >	846
mln::Object< array_bkd_iter< T > >	846
mln::Proxy< array_bkd_iter< T > >	959
mln::Object< array_fwd_iter< T > >	846
mln::Proxy< array_fwd_iter< T > >	959
mln::Object< asc_propagation >	846

mln::Object< backdiag2d >	846
mln::Window< backdiag2d >	1171
window_base< dpoint2d, backdiag2d >	
classical_window_base< dpoint2d, backdiag2d >	
mln::win::backdiag2d	1160
mln::Object< backdiagonal2d_t >	846
mln::Browsing< backdiagonal2d_t >	567
mln::canvas::browsing::backdiagonal2d_t	568
mln::Object< ball< G, C > >	846
mln::Window< ball< G, C > >	1171
window_base< dpoint< G, C >, ball< G, C > >	
classical_window_base< dpoint< G, C >, ball< G, C > >	
mln::win::ball< G, C >	1161
mln::Object< bbox >	846
mln::Meta_Accumulator< bbox >	826
mln::accu::meta::shape::bbox	481
mln::Object< bbox< P > >	846
mln::Proxy< bbox< P > >	959
mln::Accumulator< bbox< P > >	541
base< const box< P > &, bbox< P > >	
mln::accu::shape::bbox< P >	503
mln::Object< big_chess< B > >	846
mln::Function< big_chess< B > >	717
mln::Function_v2v< big_chess< B > >	719
mln::Function_v2b< big_chess< B > >	718
mln::Object< bin_off_loader >	846
mln::Object< bin_off_saver >	846
mln::Object< binary< Fun, T1, T2 > >	846
mln::Function< binary< Fun, T1, T2 > >	717
mln::Function_v2v< binary< Fun, T1, T2 > >	719
mln::Object< bkd_pixter1d< I > >	846
mln::Iterator< bkd_pixter1d< I > >	789
mln::Pixel_Iterator< bkd_pixter1d< I > >	942
mln::Object< bkd_pixter2d< I > >	846
mln::Iterator< bkd_pixter2d< I > >	789
mln::Pixel_Iterator< bkd_pixter2d< I > >	942
mln::Object< bkd_pixter3d< I > >	846
mln::Iterator< bkd_pixter3d< I > >	789
mln::Pixel_Iterator< bkd_pixter3d< I > >	942
mln::Object< black_t >	846
mln::Literal< black_t >	801
mln::literal::black_t	803
mln::Object< blue >	846
mln::Meta_Function< blue >	828
mln::Meta_Function_v2v< blue >	829
mln::Object< blue_t >	846
mln::Literal< blue_t >	801
mln::literal::blue_t	804
mln::Object< box< P > >	846

mln::Site_Set< box< P > >	968
mln::Box< box< P > >	559
mln::box< P >	551
mln::Object< box_runend_piter< P > >	846
mln::Proxy< box_runend_piter< P > >	959
mln::Site_Proxy< box_runend_piter< P > >	967
mln::Site_Iterator< box_runend_piter< P > >	965
site_iterator_base< box< P >, box_runend_piter< P > >	
site_set_iterator_base< box< P >, box_runend_piter< P > >	
mln::box_runend_piter< P >	565
mln::Object< box_runstart_piter< P > >	846
mln::Proxy< box_runstart_piter< P > >	959
mln::Site_Proxy< box_runstart_piter< P > >	967
mln::Site_Iterator< box_runstart_piter< P > >	965
site_iterator_base< box< P >, box_runstart_piter< P > >	
site_set_iterator_base< box< P >, box_runstart_piter< P > >	
mln::box_runstart_piter< P >	566
mln::Object< breadth_first_search_t >	846
mln::Browsing< breadth_first_search_t >	567
graph_first_search_t< breadth_first_search_t, std::queue >	
mln::canvas::browsing::breadth_first_search_t	570
mln::Object< brown_t >	846
mln::Literal< brown_t >	801
mln::literal::brown_t	804
mln::Object< card< I > >	846
mln::Proxy< card< I > >	959
mln::Accumulator< card< I > >	541
base< unsigned, card< I > >	
mln::morpho::attribute::card< I >	834
mln::Object< cast< V > >	846
mln::Function< cast< V > >	717
mln::Function_v2v< cast< V > >	719
mln::Object< center >	846
mln::Meta_Accumulator< center >	826
mln::accu::meta::center	463
mln::Object< center< P, V > >	846
mln::Proxy< center< P, V > >	959
mln::Accumulator< center< P, V > >	541
base< V, center< P, V > >	
mln::accu::center< P, V >	437
mln::Object< center_only_iter< D > >	846
mln::Iterator< center_only_iter< D > >	789
complex_iterator_base< algebraic_face< D >, center_only_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, center_only_	
iter< D > >	
forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >,	
center_only_iter< D > >	
mln::topo::center_only_iter< D >	1002
mln::Object< centered_bkd_iter_adapter< D, I > >	846
mln::Iterator< centered_bkd_iter_adapter< D, I > >	789

complex_relative_iterator_sequence< I, center_only_iter< D >, centered_bkd_iter_adapter< D, I >>	
mln::topo::centered_bkd_iter_adapter< D, I >>	1003
mln::Object< centered_fwd_iter_adapter< D, I >>	846
mln::Iterator< centered_fwd_iter_adapter< D, I >>	789
complex_relative_iterator_sequence< center_only_iter< D >, I, centered_fwd_iter_adapter< D, I >>	
mln::topo::centered_fwd_iter_adapter< D, I >>	1005
mln::Object< ch_function_value< F, V >>	846
mln::Function< ch_function_value< F, V >>	717
mln::Function_v2v< ch_function_value< F, V >>	719
mln::fun::v2v::ch_function_value< F, V >>	679
mln::Object< ch_piter_image< I, Fwd >>	846
mln::Image< ch_piter_image< I, Fwd >>	764
mln::Object< chess >	846
mln::Function< chess >	717
mln::Function_v2v< chess >	719
mln::Function_v2b< chess >	718
mln::Object< col >	846
mln::Meta_Function< col >	828
mln::Meta_Function_v2v< col >	829
mln::Object< colorize >	846
mln::Function< colorize >	717
mln::Function_v2v< colorize >	719
mln::Object< comp >	846
mln::Meta_Function< comp >	828
mln::Meta_Function_v2v< comp >	829
mln::Object< comp_count >	846
mln::Meta_Function< comp_count >	828
mln::Meta_Function_v2v< comp_count >	829
mln::Object< complex_image< D, G, V >>	846
mln::Image< complex_image< D, G, V >>	764
mln::Object< complex_neighborhood_bkd_piter< I, G, N >>	846
mln::Proxy< complex_neighborhood_bkd_piter< I, G, N >>	959
mln::Site_Proxy< complex_neighborhood_bkd_piter< I, G, N >>	967
mln::Site_Iterator< complex_neighborhood_bkd_piter< I, G, N >>	965
site_iterator_base< N, complex_neighborhood_bkd_piter< I, G, N >>	
site_relative_iterator_base< N, complex_neighborhood_bkd_piter< I, G, N >>	
mln::complex_neighborhood_bkd_piter< I, G, N >>	586
mln::Object< complex_neighborhood_fwd_piter< I, G, N >>	846
mln::Proxy< complex_neighborhood_fwd_piter< I, G, N >>	959
mln::Site_Proxy< complex_neighborhood_fwd_piter< I, G, N >>	967
mln::Site_Iterator< complex_neighborhood_fwd_piter< I, G, N >>	965
site_iterator_base< N, complex_neighborhood_fwd_piter< I, G, N >>	
site_relative_iterator_base< N, complex_neighborhood_fwd_piter< I, G, N >>	
mln::complex_neighborhood_fwd_piter< I, G, N >>	587
mln::Object< complex_psite< D, G >>	846
mln::Proxy< complex_psite< D, G >>	959
mln::Site_Proxy< complex_psite< D, G >>	967

mln::Pseudo_Site< complex_psite< D, G > >	959
pseudo_site_base< const G::site &, complex_psite< D, G > >	
mln::complex_psite< D, G >	589
mln::Object< complex_window_bkd_piter< I, G, W > >	846
mln::Proxy< complex_window_bkd_piter< I, G, W > >	959
mln::Site_Proxy< complex_window_bkd_piter< I, G, W > >	967
mln::Site_Iterator< complex_window_bkd_piter< I, G, W > >	965
site_iterator_base< W, complex_window_bkd_piter< I, G, W > >	
site_relative_iterator_base< W, complex_window_bkd_piter< I, G, W > >	
mln::complex_window_bkd_piter< I, G, W >	592
mln::Object< complex_window_fwd_piter< I, G, W > >	846
mln::Proxy< complex_window_fwd_piter< I, G, W > >	959
mln::Site_Proxy< complex_window_fwd_piter< I, G, W > >	967
mln::Site_Iterator< complex_window_fwd_piter< I, G, W > >	965
site_iterator_base< W, complex_window_fwd_piter< I, G, W > >	
site_relative_iterator_base< W, complex_window_fwd_piter< I, G, W > >	
mln::complex_window_fwd_piter< I, G, W >	594
mln::Object< component< T, i > >	846
mln::Function< component< T, i > >	717
mln::Function_v2v< component< T, i > >	719
mln::fun::v2v::component< T, i >	680
mln::Object< compose >	846
mln::Meta_Function< compose >	828
mln::Meta_Function_vv2v< compose >	829
mln::Object< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v, G > > . .	846
mln::Meta_Function< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v,	
G > >	828
mln::Meta_Function_v2v< composition< mln::Meta_Function_v2v, F, mln::Meta_-	
Function_v2v, G > >	829
mln::Object< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_vv2v, G > > . .	846
mln::Meta_Function< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_-	
vv2v, G > >	828
mln::Meta_Function_vv2v< composition< mln::Meta_Function_v2v, F, mln::Meta_-	
Function_vv2v, G > >	829
mln::Object< concrete >	846
mln::Object< convert< V > >	846
mln::Function< convert< V > >	717
mln::Function_v2v< convert< V > >	719
mln::Object< convolve< T1, T2, R > >	846
mln::Proxy< convolve< T1, T2, R > >	959
mln::Accumulator< convolve< T1, T2, R > >	541
base< R, convolve< T1, T2, R > >	
mln::accu::convolve< T1, T2, R >	439
mln::Object< cos >	846
mln::Meta_Function< cos >	828
mln::Meta_Function_v2v< cos >	829
mln::Object< cos< V > >	846
mln::Function< cos< V > >	717
mln::Function_v2v< cos< V > >	719

mln::fun::v2w2v::cos< V >	686
mln::Object< count >	846
mln::Meta_Accumulator< count >	826
mln::accu::meta::math::count	472
mln::Object< count< T > >	846
mln::Proxy< count< T > >	959
mln::Accumulator< count< T > >	541
base< unsigned, count< T > >	
mln::accu::math::count< T >	455
mln::Object< count_adjacent_vertices >	846
mln::Meta_Accumulator< count_adjacent_vertices >	826
mln::accu::meta::count_adjacent_vertices	463
mln::Object< count_adjacent_vertices< F, S > >	846
mln::Proxy< count_adjacent_vertices< F, S > >	959
mln::Accumulator< count_adjacent_vertices< F, S > >	541
base< unsigned, count_adjacent_vertices< F, S > >	
mln::accu::count_adjacent_vertices< F, S >	440
mln::Object< count_adjacent_vertices< I > >	846
mln::Proxy< count_adjacent_vertices< I > >	959
mln::Accumulator< count_adjacent_vertices< I > >	541
base< unsigned, count_adjacent_vertices< I > >	
mln::morpho::attribute::count_adjacent_vertices< I >	835
mln::Object< count_labels >	846
mln::Meta_Accumulator< count_labels >	826
mln::accu::meta::count_labels	464
mln::Object< count_labels< L > >	846
mln::Proxy< count_labels< L > >	959
mln::Accumulator< count_labels< L > >	541
base< unsigned, count_labels< L > >	
mln::accu::count_labels< L >	442
mln::Object< count_value >	846
mln::Meta_Accumulator< count_value >	826
mln::accu::meta::count_value	465
mln::Object< count_value< V > >	846
mln::Proxy< count_value< V > >	959
mln::Accumulator< count_value< V > >	541
base< unsigned, count_value< V > >	
mln::accu::count_value< V >	443
mln::Object< couple< T, U > >	846
mln::util::couple< T, U >	1048
mln::Object< cube >	846
mln::Mesh< cube >	825
mln::Regular_Grid< cube >	962
mln::Object< cube3d >	846
mln::Window< cube3d >	1171
window_base< dpoint3d, cube3d >	
classical_window_base< dpoint3d, cube3d >	
mln::win::cube3d	1162
mln::Object< cuboid3d >	846

mln::Window< cuboid3d >	1171
window_base< dpoint3d, cuboid3d >	
classical_window_base< dpoint3d, cuboid3d >	
mln::win::cuboid3d	1163
mln::Object< cyan_t >	846
mln::Literal< cyan_t >	801
mln::literal::cyan_t	805
mln::Object< d_t >	846
mln::Function< d_t >	717
mln::Function_vv2v< d_t >	720
mln::Object< dark_gray_t >	846
mln::Literal< dark_gray_t >	801
mln::Object< dashed_line_f< I, dim > >	846
mln::Function< dashed_line_f< I, dim > >	717
mln::Function_v2v< dashed_line_f< I, dim > >	719
mln::Function_v2b< dashed_line_f< I, dim > >	718
mln::Object< decorated_image< I, D > >	846
mln::Image< decorated_image< I, D > >	764
mln::Object< depth1st_piter< T > >	846
mln::Proxy< depth1st_piter< T > >	959
mln::Site_Proxy< depth1st_piter< T > >	967
mln::Site_Iterator< depth1st_piter< T > >	965
mln::Object< depth_first_search_t >	846
mln::Browsing< depth_first_search_t >	567
graph_first_search_t< depth_first_search_t, std::stack >	
mln::canvas::browsing::depth_first_search_t	570
mln::Object< desc_propagation >	846
mln::Object< deviation >	846
mln::Meta_Accumulator< deviation >	826
mln::accu::stat::meta::deviation	522
mln::Object< deviation< T, S, M > >	846
mln::Proxy< deviation< T, S, M > >	959
mln::Accumulator< deviation< T, S, M > >	541
base< M, deviation< T, S, M > >	
mln::accu::stat::deviation< T, S, M >	510
mln::Object< diag2d >	846
mln::Window< diag2d >	1171
window_base< dpoint2d, diag2d >	
classical_window_base< dpoint2d, diag2d >	
mln::win::diag2d	1165
mln::Object< diagonal2d_t >	846
mln::Browsing< diagonal2d_t >	567
mln::canvas::browsing::diagonal2d_t	570
mln::Object< diff_abs< V > >	846
mln::Function< diff_abs< V > >	717
mln::Function_vv2v< diff_abs< V > >	720
mln::fun::vv2v::diff_abs< V >	696
mln::Object< dir_struct_elt_incr_update_t >	846
mln::Browsing< dir_struct_elt_incr_update_t >	567

mln::canvas::browsing::dir_struct_elt_incr_update_t	572
mln::Object< directional_t >	846
mln::Browsing< directional_t >	567
mln::canvas::browsing::directional_t	573
mln::Object< dist >	846
mln::Function< dist >	717
mln::Function_vv2v< dist >	720
mln::Object< dist_t >	846
mln::Function< dist_t >	717
mln::Function_vv2v< dist_t >	720
mln::Object< dn_leaf_piter< T > >	846
mln::Proxy< dn_leaf_piter< T > >	959
mln::Site_Proxy< dn_leaf_piter< T > >	967
mln::Site_Iterator< dn_leaf_piter< T > >	965
mln::Object< dn_node_piter< T > >	846
mln::Proxy< dn_node_piter< T > >	959
mln::Site_Proxy< dn_node_piter< T > >	967
mln::Site_Iterator< dn_node_piter< T > >	965
mln::Object< dn_site_piter< T > >	846
mln::Proxy< dn_site_piter< T > >	959
mln::Site_Proxy< dn_site_piter< T > >	967
mln::Site_Iterator< dn_site_piter< T > >	965
mln::Object< dpoint< G, C > >	846
mln::Gdpoint< dpoint< G, C > >	725
mln::dpoint< G, C >	644
mln::Object< dpoints_bkd_pixter< I > >	846
mln::Iterator< dpoints_bkd_pixter< I > >	789
mln::Pixel_Iterator< dpoints_bkd_pixter< I > >	942
mln::Object< dpoints_fwd_pixter< I > >	846
mln::Iterator< dpoints_fwd_pixter< I > >	789
mln::Pixel_Iterator< dpoints_fwd_pixter< I > >	942
mln::Object< dpsites_bkd_piter< V > >	846
mln::Proxy< dpsites_bkd_piter< V > >	959
mln::Site_Proxy< dpsites_bkd_piter< V > >	967
mln::Site_Iterator< dpsites_bkd_piter< V > >	965
site_iterator_base< V, dpsites_bkd_piter< V > >	
site_relative_iterator_base< V, dpsites_bkd_piter< V > >	
mln::dpsites_bkd_piter< V >	654
mln::Object< dpsites_fwd_piter< V > >	846
mln::Proxy< dpsites_fwd_piter< V > >	959
mln::Site_Proxy< dpsites_fwd_piter< V > >	967
mln::Site_Iterator< dpsites_fwd_piter< V > >	965
site_iterator_base< V, dpsites_fwd_piter< V > >	
site_relative_iterator_base< V, dpsites_fwd_piter< V > >	
mln::dpsites_fwd_piter< V >	655
mln::Object< eat >	846
mln::util::eat	1050
mln::Object< edge_bkd_iterator< G > >	846

mln::Proxy< edge_bkd_iterator< G > >	959
mln::Object< edge_fwd_iterator< G > >	846
mln::Proxy< edge_fwd_iterator< G > >	959
mln::Object< edge_image< P, V, G > >	846
mln::Image< edge_image< P, V, G > >	764
mln::Object< edge_nbh_edge_bkd_iterator< G > >	846
mln::Proxy< edge_nbh_edge_bkd_iterator< G > >	959
mln::Object< edge_nbh_edge_fwd_iterator< G > >	846
mln::Proxy< edge_nbh_edge_fwd_iterator< G > >	959
mln::Object< edge_to_color< I, V > >	846
mln::Function< edge_to_color< I, V > >	717
mln::Function_v2v< edge_to_color< I, V > >	719
mln::Object< enc< V > >	846
mln::Function< enc< V > >	717
mln::Function_v2v< enc< V > >	719
mln::Object< eq< L, R > >	846
mln::Function< eq< L, R > >	717
mln::Function_vv2b< eq< L, R > >	720
mln::fun::vv2b::eq< L, R >	691
mln::Object< extended< I > >	846
mln::Image< extended< I > >	764
mln::Object< extension_fun< I, F > >	846
mln::Image< extension_fun< I, F > >	764
mln::Object< extension_ima< I, J > >	846
mln::Image< extension_ima< I, J > >	764
mln::Object< extension_val< I > >	846
mln::Image< extension_val< I > >	764
mln::Object< f_16_to_8 >	846
mln::Function< f_16_to_8 >	717
mln::Function_v2v< f_16_to_8 >	719
mln::Object< f_box1d_t >	846
mln::Function< f_box1d_t >	717
mln::Function_v2v< f_box1d_t >	719
mln::Function_v2b< f_box1d_t >	718
mln::Object< f_box2d_t >	846
mln::Function< f_box2d_t >	717
mln::Function_v2v< f_box2d_t >	719
mln::Function_v2b< f_box2d_t >	718
mln::Object< f_box3d_t >	846
mln::Function< f_box3d_t >	717
mln::Function_v2v< f_box3d_t >	719
mln::Function_v2b< f_box3d_t >	718
mln::Object< f_hsi_to_rgb_< T_rgb > >	846
mln::Function< f_hsi_to_rgb_< T_rgb > >	717
mln::Function_v2v< f_hsi_to_rgb_< T_rgb > >	719
mln::Object< f_hsl_to_rgb_< T_rgb > >	846
mln::Function< f_hsl_to_rgb_< T_rgb > >	717

mln::Function_v2v< f_hsl_to_rgb_< T_rgb > >	719
mln::Object< f_rgb_to_hsi_< T_hsi > >	846
mln::Function< f_rgb_to_hsi_< T_hsi > >	717
mln::Function_v2v< f_rgb_to_hsi_< T_hsi > >	719
mln::Object< f_rgb_to_hsl_< T_hsl > >	846
mln::Function< f_rgb_to_hsl_< T_hsl > >	717
mln::Function_v2v< f_rgb_to_hsl_< T_hsl > >	719
mln::Object< face_bkd_iter< D > >	846
mln::Iterator< face_bkd_iter< D > >	789
complex_iterator_base< topo::face< D >, face_bkd_iter< D > >	
complex_set_iterator_base< topo::face< D >, face_bkd_iter< D > >	
mln::topo::face_bkd_iter< D >	1013
mln::Object< face_fwd_iter< D > >	846
mln::Iterator< face_fwd_iter< D > >	789
complex_iterator_base< topo::face< D >, face_fwd_iter< D > >	
complex_set_iterator_base< topo::face< D >, face_fwd_iter< D > >	
mln::topo::face_fwd_iter< D >	1014
mln::Object< faces_psite< N, D, P > >	846
mln::Proxy< faces_psite< N, D, P > >	959
mln::Site_Proxy< faces_psite< N, D, P > >	967
mln::Pseudo_Site< faces_psite< N, D, P > >	959
pseudo_site_base_< const P &, faces_psite< N, D, P > >	
mln::faces_psite< N, D, P >	668
mln::Object< fibonacci_heap< P, T > >	846
mln::util::fibonacci_heap< P, T >	1055
mln::Object< flat_image< T, S > >	846
mln::Image< flat_image< T, S > >	764
mln::Object< float01 >	846
mln::Value< float01 >	1108
mln::Object< float01_f >	846
mln::Value< float01_f >	1108
mln::Object< float_off_loader >	846
mln::Object< float_off_saver >	846
mln::Object< fold< P, dir_0, dir_1, dir_2 > >	846
mln::Function< fold< P, dir_0, dir_1, dir_2 > >	717
mln::Function_v2v< fold< P, dir_0, dir_1, dir_2 > >	719
mln::Object< from_accu< A > >	846
mln::Meta_Function< from_accu< A > >	828
mln::Meta_Function_v2v< from_accu< A > >	829
mln::Object< fun_image< F, I > >	846
mln::Image< fun_image< F, I > >	764
mln::Object< function< meta::blue< value::rgb< n > > > >	846
mln::Function< function< meta::blue< value::rgb< n > > > >	717
mln::Function_v2v< function< meta::blue< value::rgb< n > > > >	719
mln::Object< function< meta::first< util::couple< T, U > > > >	846
mln::Function< function< meta::first< util::couple< T, U > > > >	717
mln::Function_v2v< function< meta::first< util::couple< T, U > > > >	719
mln::Object< function< meta::green< value::rgb< n > > > >	846

mln::Function< function< meta::green< value::rgb< n > > > >	717
mln::Function_v2v< function< meta::green< value::rgb< n > > > >	719
mln::Object< function< meta::red< value::rgb< n > > > >	846
mln::Function< function< meta::red< value::rgb< n > > > >	717
mln::Function_v2v< function< meta::red< value::rgb< n > > > >	719
mln::Object< function< meta::second< util::couple< T, U > > > >	846
mln::Function< function< meta::second< util::couple< T, U > > > >	717
mln::Function_v2v< function< meta::second< util::couple< T, U > > > >	719
mln::Object< function< meta::to_enc< T > > >	846
mln::Function< function< meta::to_enc< T > > >	717
mln::Function_v2v< function< meta::to_enc< T > > >	719
mln::Object< fwd_pixter1d< I > >	846
mln::Iterator< fwd_pixter1d< I > >	789
mln::Pixel_Iterator< fwd_pixter1d< I > >	942
mln::Object< fwd_pixter2d< I > >	846
mln::Iterator< fwd_pixter2d< I > >	789
mln::Pixel_Iterator< fwd_pixter2d< I > >	942
mln::Object< fwd_pixter3d< I > >	846
mln::Iterator< fwd_pixter3d< I > >	789
mln::Pixel_Iterator< fwd_pixter3d< I > >	942
mln::Object< fwd_t >	846
mln::Browsing< fwd_t >	567
mln::canvas::browsing::fwd_t	575
mln::Object< ge< L, R > >	846
mln::Function< ge< L, R > >	717
mln::Function_vv2b< ge< L, R > >	720
mln::fun::vv2b::ge< L, R >	691
mln::Object< graph >	846
mln::Graph< graph >	733
graph_base< graph >	
mln::util::graph	1059
mln::Object< graph_elt_mixed_window< G, S, S2 > >	846
mln::Window< graph_elt_mixed_window< G, S, S2 > >	1171
mln::graph_window_base< S2::fun_t::result, graph_elt_mixed_window< G, S, S2 > >	753
mln::graph_elt_mixed_window< G, S, S2 >	736
mln::Object< graph_elt_window< G, S > >	846
mln::Window< graph_elt_window< G, S > >	1171
mln::graph_window_base< S::fun_t::result, graph_elt_window< G, S > >	753
mln::graph_elt_window< G, S >	744
mln::Object< graph_elt_window_if< G, S, I > >	846
mln::Window< graph_elt_window_if< G, S, I > >	1171
mln::graph_window_base< S::fun_t::result, graph_elt_window_if< G, S, I > >	753
mln::graph_elt_window_if< G, S, I >	748
mln::Object< graph_window_if_piter< S, W, I > >	846
mln::Proxy< graph_window_if_piter< S, W, I > >	959
mln::Site_Proxy< graph_window_if_piter< S, W, I > >	967
mln::Site_Iterator< graph_window_if_piter< S, W, I > >	965
site_iterator_base< W, graph_window_if_piter< S, W, I > >	

site_relative_iterator_base< W, graph_window_if_piter< S, W, I > >	
mln::graph_window_if_piter< S, W, I >	755
mln::Object< graph_window_piter< S, W, I > >	846
mln::Proxy< graph_window_piter< S, W, I > >	959
mln::Site_Proxy< graph_window_piter< S, W, I > >	967
mln::Site_Iterator< graph_window_piter< S, W, I > >	965
site_iterator_base< W, graph_window_piter< S, W, I > >	
site_relative_iterator_base< W, graph_window_piter< S, W, I >, W::center_t >	
mln::graph_window_piter< S, W, I >	757
mln::Object< gray_f >	846
mln::Value< gray_f >	1108
mln::Object< graylevel< n > >	846
mln::Value< graylevel< n > >	1108
mln::Object< graylevel_f >	846
mln::Object< green >	846
mln::Meta_Function< green >	828
mln::Meta_Function_v2v< green >	829
mln::Object< green_t >	846
mln::Literal< green_t >	801
mln::literal::green_t	806
mln::Object< gt< L, R > >	846
mln::Function< gt< L, R > >	717
mln::Function_vv2b< gt< L, R > >	720
mln::fun::vv2b::gt< L, R >	692
mln::Object< has< I > >	846
mln::Function< has< I > >	717
mln::Function_v2v< has< I > >	719
mln::Function_v2b< has< I > >	718
mln::Object< height >	846
mln::Meta_Accumulator< height >	826
mln::accu::meta::shape::height	482
mln::Object< height< I > >	846
mln::Proxy< height< I > >	959
mln::Accumulator< height< I > >	541
base< unsigned, height< I > >	
mln::accu::shape::height< I >	504
mln::morpho::attribute::height< I >	837
mln::Object< hexa >	846
mln::Mesh< hexa >	825
mln::Regular_Grid< hexa >	962
mln::Object< hexa< I > >	846
mln::Image< hexa< I > >	764
mln::Object< hexa< image2d< V > > >	846
mln::Image< hexa< image2d< V > > >	764
mln::Object< histo >	846
mln::Meta_Accumulator< histo >	826
mln::accu::meta::histo	466
mln::Object< histo3d_rgb >	846

mln::Meta_Accumulator< histo3d_rgb >	826
mln::Object< histo3d_rgb< V > >	846
mln::Proxy< histo3d_rgb< V > >	959
mln::Accumulator< histo3d_rgb< V > >	541
base< image3d< unsigned >, histo3d_rgb< V > >	
mln::accu::stat::histo3d_rgb< V >	512
mln::Object< histo< V > >	846
mln::Proxy< histo< V > >	959
mln::Accumulator< histo< V > >	541
base< const std::vector< unsigned > &, histo< V > >	
mln::accu::histo< V >	445
mln::Object< hyper_directional_t >	846
mln::Browsing< hyper_directional_t >	567
mln::canvas::browsing::hyper_directional_t	576
mln::Object< id2element< G, Elt > >	846
mln::Function< id2element< G, Elt > >	717
mln::Function_v2v< id2element< G, Elt > >	719
mln::Object< identity_t >	846
mln::Literal< identity_t >	801
mln::literal::identity_t	807
mln::Object< ignore >	846
mln::util::ignore	1068
mln::Object< image1d< T > >	846
mln::Image< image1d< T > >	764
mln::Object< image2d< T > >	846
mln::Image< image2d< T > >	764
mln::Object< image3d< T > >	846
mln::Image< image3d< T > >	764
mln::Object< image< F, S > >	846
mln::Image< image< F, S > >	764
mln::Object< image_if< I, F > >	846
mln::Image< image_if< I, F > >	764
mln::Object< implies< L, R > >	846
mln::Function< implies< L, R > >	717
mln::Function_vv2b< implies< L, R > >	720
mln::fun::vv2b::implies< L, R >	693
mln::Object< index_of_value< bool > >	846
mln::Function< index_of_value< bool > >	717
mln::Function_v2v< index_of_value< bool > >	719
mln::Object< index_of_value< T > >	846
mln::Function< index_of_value< T > >	717
mln::Function_v2v< index_of_value< T > >	719
mln::Object< inf >	846
mln::Meta_Accumulator< inf >	826
mln::accu::meta::math::inf	473
mln::Meta_Function< inf >	828
mln::Meta_Function_vv2v< inf >	829
mln::Object< inf< T > >	846

mln::Proxy< inf< T > >	959
mln::Accumulator< inf< T > >	541
base< const T &, inf< T > >	
mln::accu::math::inf< T >	457
mln::Object< int_s< n > >	846
mln::Value< int_s< n > >	1108
mln::Object< int_u8_off_saver >	846
mln::Object< int_u< n > >	846
mln::Value< int_u< n > >	1108
mln::Object< int_u_sat< n > >	846
mln::Value< int_u_sat< n > >	1108
mln::Object< interpolated< I, F > >	846
mln::Image< interpolated< I, F > >	764
mln::Object< iota >	846
mln::Function< iota >	717
mln::Function_v2v< iota >	719
mln::Object< is_dot >	846
mln::Function< is_dot >	717
mln::Function_v2v< is_dot >	719
mln::Function_v2b< is_dot >	718
mln::Object< is_edge >	846
mln::Function< is_edge >	717
mln::Function_v2v< is_edge >	719
mln::Function_v2b< is_edge >	718
mln::Object< is_n_face< N > >	846
mln::Function< is_n_face< N > >	717
mln::Function_v2v< is_n_face< N > >	719
mln::Function_v2b< is_n_face< N > >	718
mln::topo::is_n_face< N >	1016
mln::Object< is_pixel >	846
mln::Function< is_pixel >	717
mln::Function_v2v< is_pixel >	719
mln::Function_v2b< is_pixel >	718
mln::Object< is_row_odd >	846
mln::Function< is_row_odd >	717
mln::Function_v2v< is_row_odd >	719
mln::Function_v2b< is_row_odd >	718
mln::Object< is_separator >	846
mln::Function< is_separator >	717
mln::Function_v2v< is_separator >	719
mln::Function_v2b< is_separator >	718
mln::world::inter_pixel::is_separator	1177
mln::Object< is_simple_cell< I > >	846
mln::Function< is_simple_cell< I > >	717
mln::Function_v2v< is_simple_cell< I > >	719
mln::Function_v2b< is_simple_cell< I > >	718
mln::topo::is_simple_cell< I >	1017
mln::Object< ithcomp >	846

mln::Meta_Function< ithcomp >	828
mln::Meta_Function_vv2v< ithcomp >	829
mln::Object< keep_specific_colors >	846
mln::Function< keep_specific_colors >	717
mln::Function_v2v< keep_specific_colors >	719
mln::Function_v2b< keep_specific_colors >	718
mln::Object< l1 >	846
mln::Meta_Function< l1 >	828
mln::Meta_Function_v2v< l1 >	829
mln::Object< l1_norm< V > >	846
mln::Function< l1_norm< V > >	717
mln::Function_v2v< l1_norm< V > >	719
mln::Object< l1_norm< V, R > >	846
mln::Function< l1_norm< V, R > >	717
mln::Function_v2v< l1_norm< V, R > >	719
mln::fun::v2v::l1_norm< V, R >	680
mln::fun::v2w_w2v::l1_norm< V, R >	687
mln::Object< l2 >	846
mln::Meta_Function< l2 >	828
mln::Meta_Function_v2v< l2 >	829
mln::Object< l2_norm< V, R > >	846
mln::Function< l2_norm< V, R > >	717
mln::Function_v2v< l2_norm< V, R > >	719
mln::fun::v2v::l2_norm< V, R >	681
mln::fun::v2w_w2v::l2_norm< V, R >	689
mln::Object< label< n > >	846
mln::Value< label< n > >	1108
mln::Object< label_used >	846
mln::Meta_Accumulator< label_used >	826
mln::accu::meta::label_used	467
mln::Object< label_used< L > >	846
mln::Proxy< label_used< L > >	959
mln::Accumulator< label_used< L > >	541
base< const fun::i2v::array< bool > &, label_used< L > >	
mln::accu::label_used< L >	446
mln::Object< labeled_image< I > >	846
mln::Image< labeled_image< I > >	764
mln::Object< land >	846
mln::Meta_Accumulator< land >	826
mln::accu::meta::logic::land	468
mln::Proxy< land >	959
mln::Accumulator< land >	541
base< bool, land >	
mln::accu::logic::land	448
mln::Object< land< L, R > >	846
mln::Function< land< L, R > >	717
mln::Function_vv2v< land< L, R > >	720
mln::fun::vv2v::land< L, R >	697

mln::Object< land_basic >	846
mln::Meta_Accumulator< land_basic >	826
mln::Proxy< land_basic >	959
mln::Accumulator< land_basic >	541
base< bool, land_basic >	
mln::accu::logic::land_basic	449
mln::Object< land_not< L, R > >	846
mln::Function< land_not< L, R > >	717
mln::Function_vv2v< land_not< L, R > >	720
mln::fun::vv2v::land_not< L, R >	698
mln::Object< lazy_image< I, F, B > >	846
mln::Image< lazy_image< I, F, B > >	764
mln::Object< le< L, R > >	846
mln::Function< le< L, R > >	717
mln::Function_vv2b< le< L, R > >	720
mln::fun::vv2b::le< L, R >	694
mln::Object< light_gray_t >	846
mln::Literal< light_gray_t >	801
mln::literal::light_gray_t	808
mln::Object< lime_t >	846
mln::Literal< lime_t >	801
mln::literal::lime_t	809
mln::Object< line< M, i, C > >	846
mln::Window< line< M, i, C > >	1171
window_base< dpoint< M, C >, line< M, i, C > >	
classical_window_base< dpoint< M, C >, line< M, i, C > >	
mln::win::line< M, i, C >	1166
mln::Object< line_graph< G > >	846
mln::Graph< line_graph< G > >	733
graph_base< line_graph< G > >	
mln::util::line_graph< G >	1069
mln::Object< linear< V, T, R > >	846
mln::Function< linear< V, T, R > >	717
mln::Function_v2v< linear< V, T, R > >	719
mln::fun::v2v::linear< V, T, R >	682
mln::Object< linear_sat< V, T, R > >	846
mln::Function< linear_sat< V, T, R > >	717
mln::Function_v2v< linear_sat< V, T, R > >	719
mln::Object< linfty >	846
mln::Meta_Function< linfty >	828
mln::Meta_Function_v2v< linfty >	829
mln::Object< linfty_norm< V, R > >	846
mln::Function< linfty_norm< V, R > >	717
mln::Function_v2v< linfty_norm< V, R > >	719
mln::fun::v2v::linfty_norm< V, R >	683
mln::fun::v2w_w2v::linfty_norm< V, R >	690
mln::Object< lnot< V > >	846
mln::Function< lnot< V > >	717
mln::Function_v2v< lnot< V > >	719

mln::Function_v2b< lnot< V > >	718
mln::fun::v2b::lnot< V >	677
mln::Object< lor >	846
mln::Meta_Accumulator< lor >	826
mln::accu::meta::logic::lor	469
mln::Proxy< lor >	959
mln::Accumulator< lor >	541
base< bool, lor >	
mln::accu::logic::lor	451
mln::Object< lor< L, R > >	846
mln::Function< lor< L, R > >	717
mln::Function_vv2v< lor< L, R > >	720
mln::fun::vv2v::lor< L, R >	699
mln::Object< lor_basic >	846
mln::Meta_Accumulator< lor_basic >	826
mln::accu::meta::logic::lor_basic	470
mln::Proxy< lor_basic >	959
mln::Accumulator< lor_basic >	541
base< bool, lor_basic >	
mln::accu::logic::lor_basic	452
mln::Object< lt< L, R > >	846
mln::Function< lt< L, R > >	717
mln::Function_vv2b< lt< L, R > >	720
mln::fun::vv2b::lt< L, R >	695
mln::Object< lut_vec< S, T > >	846
mln::Value_Set< lut_vec< S, T > >	1149
mln::value::lut_vec< S, T >	1130
mln::Object< lxor< L, R > >	846
mln::Function< lxor< L, R > >	717
mln::Function_vv2v< lxor< L, R > >	720
mln::fun::vv2v::lxor< L, R >	700
mln::Object< magenta_t >	846
mln::Literal< magenta_t >	801
mln::literal::magenta_t	810
mln::Object< mahalanobis< V > >	846
mln::Function< mahalanobis< V > >	717
mln::Function_v2v< mahalanobis< V > >	719
mln::Object< maj_h >	846
mln::Meta_Accumulator< maj_h >	826
mln::accu::meta::maj_h	471
mln::Object< maj_h< T > >	846
mln::Proxy< maj_h< T > >	959
mln::Accumulator< maj_h< T > >	541
base< const T &, maj_h< T > >	
mln::accu::maj_h< T >	454
mln::Object< mat< n, m, T > >	846
mln::Object< max >	846
mln::Meta_Accumulator< max >	826
mln::accu::meta::stat::max	484

mln::Object< max< T > >	846
mln::Proxy< max< T > >	959
mln::Accumulator< max< T > >	541
base< const T &, max< T > >	
mln::accu::stat::max< T >	514
mln::Object< max< V > >	846
mln::Function< max< V > >	717
mln::Function_vv2v< max< V > >	720
mln::fun::vv2v::max< V >	701
mln::Object< max_h >	846
mln::Meta_Accumulator< max_h >	826
mln::accu::meta::stat::max_h	485
mln::Object< max_h< V > >	846
mln::Proxy< max_h< V > >	959
mln::Accumulator< max_h< V > >	541
base< const V &, max_h< V > >	
mln::accu::stat::max_h< V >	516
mln::Object< max_site >	846
mln::Meta_Accumulator< max_site >	826
mln::accu::meta::max_site	476
mln::Object< max_site< I > >	846
mln::Proxy< max_site< I > >	959
mln::Accumulator< max_site< I > >	541
base< I::psite, max_site< I > >	
mln::accu::max_site< I >	461
mln::Object< max_t >	846
mln::Literal< max_t >	801
mln::literal::max_t	811
mln::Object< mean >	846
mln::Meta_Accumulator< mean >	826
mln::accu::meta::stat::mean	486
mln::Meta_Function< mean >	828
mln::Meta_Function_v2v< mean >	829
mln::Object< mean< T, S, M > >	846
mln::Proxy< mean< T, S, M > >	959
mln::Accumulator< mean< T, S, M > >	541
base< M, mean< T, S, M > >	
mln::accu::stat::mean< T, S, M >	517
mln::Object< median_alt< S > >	846
mln::Proxy< median_alt< S > >	959
mln::Accumulator< median_alt< S > >	541
base< const S::value &, median_alt< S > >	
mln::accu::stat::median_alt< S >	519
mln::Object< median_alt< T > >	846
mln::Meta_Accumulator< median_alt< T > >	826
mln::accu::meta::stat::median_alt< T >	487
mln::Object< median_alt< value::set< T > > >	846
mln::Proxy< median_alt< value::set< T > > >	959
mln::Accumulator< median_alt< value::set< T > > >	541

base< const value::set< T >::value &, median_alt< value::set< T > > >	
mln::accu::stat::median_alt< value::set< T > >	519
mln::Object< median_h >	846
mln::Meta_Accumulator< median_h >	826
mln::accu::meta::stat::median_h	488
mln::Object< median_h< V > >	846
mln::Proxy< median_h< V > >	959
mln::Accumulator< median_h< V > >	541
base< const V &, median_h< V > >	
mln::accu::stat::median_h< V >	521
mln::Object< medium_gray_t >	846
mln::Literal< medium_gray_t >	801
mln::Object< min >	846
mln::Meta_Accumulator< min >	826
mln::accu::meta::stat::min	489
mln::Object< min< L, R > >	846
mln::Function< min< L, R > >	717
mln::Function_vv2v< min< L, R > >	720
mln::fun::vv2v::min< L, R >	702
mln::Object< min< T > >	846
mln::Proxy< min< T > >	959
mln::Accumulator< min< T > >	541
base< const T &, min< T > >	
mln::accu::stat::min< T >	523
mln::Object< min_h >	846
mln::Meta_Accumulator< min_h >	826
mln::accu::meta::stat::min_h	490
mln::Object< min_h< V > >	846
mln::Proxy< min_h< V > >	959
mln::Accumulator< min_h< V > >	541
base< const V &, min_h< V > >	
mln::accu::stat::min_h< V >	525
mln::Object< min_t >	846
mln::Literal< min_t >	801
mln::literal::min_t	812
mln::Object< mirror< B > >	846
mln::Function< mirror< B > >	717
mln::Function_v2v< mirror< B > >	719
mln::Object< mixed_neighb< W > >	846
mln::Neighborhood< mixed_neighb< W > >	846
mln::Object< mln::util::set< T > >	846
mln::util::set< T >	1082
mln::Object< multi_site< P > >	846
mln::Object< multiple< W, F > >	846
mln::Window< multiple< W, F > >	1171
window_base< W::dpsite, multiple< W, F > >	
mln::win::multiple< W, F >	1168
mln::Object< multiple_qiter< W, F > >	846

mln::Proxy< multiple_qiter< W, F > >	959
mln::Site_Proxy< multiple_qiter< W, F > >	967
mln::Site_Iterator< multiple_qiter< W, F > >	965
mln::Object< multiple_size< n, W, F > >	846
mln::Window< multiple_size< n, W, F > >	1171
window_base< W::dpsite, multiple_size< n, W, F > >	
mln::win::multiple_size< n, W, F >	1168
mln::Object< multiple_size_qiter< n, W, F > >	846
mln::Proxy< multiple_size_qiter< n, W, F > >	959
mln::Site_Proxy< multiple_size_qiter< n, W, F > >	967
mln::Site_Iterator< multiple_size_qiter< n, W, F > >	965
mln::Object< my_box2d >	846
mln::Function< my_box2d >	717
mln::Function_v2v< my_box2d >	719
mln::Function_v2b< my_box2d >	718
mln::Object< my_ext >	846
mln::Function< my_ext >	717
mln::Function_v2v< my_ext >	719
mln::Object< my_fun< G > >	846
mln::Function< my_fun< G > >	717
mln::Object< my_image2d< T > >	846
mln::Image< my_image2d< T > >	764
mln::Object< my_values_t >	846
mln::Function< my_values_t >	717
mln::Function_v2v< my_values_t >	719
mln::Object< myfun >	846
mln::Function< myfun >	717
mln::Function_vv2v< myfun >	720
mln::Object< mysqrt >	846
mln::Function< mysqrt >	717
mln::Function_v2v< mysqrt >	719
mln::Object< n_face_bkd_iter< D > >	846
mln::Iterator< n_face_bkd_iter< D > >	789
complex_iterator_base< topo::face< D >, n_face_bkd_iter< D > >	
complex_set_iterator_base< topo::face< D >, n_face_bkd_iter< D > >	
mln::topo::n_face_bkd_iter< D >	1024
mln::Object< n_face_fwd_iter< D > >	846
mln::Iterator< n_face_fwd_iter< D > >	789
complex_iterator_base< topo::face< D >, n_face_fwd_iter< D > >	
complex_set_iterator_base< topo::face< D >, n_face_fwd_iter< D > >	
mln::topo::n_face_fwd_iter< D >	1025
mln::Object< neighb< graph_elt_mixed_window< G, S, S2 > > >	846
mln::Neighborhood< neighb< graph_elt_mixed_window< G, S, S2 > > >	846
mln::Object< neighb< graph_elt_window< G, S > > >	846
mln::Neighborhood< neighb< graph_elt_window< G, S > > >	846
mln::Object< neighb< graph_elt_window_if< G, S, I > > >	846
mln::Neighborhood< neighb< graph_elt_window_if< G, S, I > > >	846
mln::Object< neighb< W > >	846

mln::Neighborhood< neighb< W > >	846
mln::Object< neighb_bkd_niter< W > >	846
mln::Proxy< neighb_bkd_niter< W > >	959
mln::Site_Proxy< neighb_bkd_niter< W > >	967
mln::Site_Iterator< neighb_bkd_niter< W > >	965
mln::Object< neighb_fwd_niter< W > >	846
mln::Proxy< neighb_fwd_niter< W > >	959
mln::Site_Proxy< neighb_fwd_niter< W > >	967
mln::Site_Iterator< neighb_fwd_niter< W > >	965
mln::Object< nil >	846
mln::Meta_Accumulator< nil >	826
mln::accu::meta::nil	477
mln::util::nil	1075
mln::Object< nil< T > >	846
mln::Proxy< nil< T > >	959
mln::Accumulator< nil< T > >	541
base< util::ignore, nil< T > >	
mln::accu::nil< T >	495
mln::Object< not_to_remove >	846
mln::Function< not_to_remove >	717
mln::Function_v2v< not_to_remove >	719
mln::Function_v2b< not_to_remove >	718
mln::Object< object_id< Tag, V > >	846
mln::Value< object_id< Tag, V > >	1108
mln::Object< octagon2d >	846
mln::Window< octagon2d >	1171
window_base< dpoint2d, octagon2d >	
classical_window_base< dpoint2d, octagon2d >	
mln::win::octagon2d	1169
mln::Object< olive_t >	846
mln::Literal< olive_t >	801
mln::literal::olive_t	813
mln::Object< one_t >	846
mln::Literal< one_t >	801
mln::literal::one_t	814
mln::Object< orange_t >	846
mln::Literal< orange_t >	801
mln::literal::orange_t	815
mln::Object< ord_pair< T > >	846
mln::util::ord_pair< T >	1078
mln::Object< origin_t >	846
mln::Literal< origin_t >	801
mln::literal::origin_t	816
mln::Object< P >	846
mln::Point_Site< P >	955
mln::Point< P >	945
mln::Object< p2p_image< I, F > >	846
mln::Image< p2p_image< I, F > >	764

mln::Object< p< A > >	846
mln::Proxy< p< A > >	959
mln::Accumulator< p< A > >	541
base< const A::result &, p< A > >	
mln::accu::p< A >	497
mln::Object< p< mA > >	846
mln::Meta_Accumulator< p< mA > >	826
mln::accu::meta::p< mA >	478
mln::Object< p_array< P > >	846
mln::Site_Set< p_array< P > >	968
site_set_base_< P, p_array< P > >	
mln::p_array< P >	849
mln::Object< p_centered< W > >	846
mln::Site_Set< p_centered< W > >	968
site_set_base_< W::psite, p_centered< W > >	
mln::p_centered< W >	854
mln::Object< p_centered_piter< W > >	846
mln::Proxy< p_centered_piter< W > >	959
mln::Site_Proxy< p_centered_piter< W > >	967
mln::Site_Iterator< p_centered_piter< W > >	965
mln::Object< p_complex< D, G > >	846
mln::Site_Set< p_complex< D, G > >	968
site_set_base_< complex_psite< D, G >, p_complex< D, G > >	
mln::p_complex< D, G >	857
mln::Object< p_double_piter< S, I1, I2 > >	846
mln::Proxy< p_double_piter< S, I1, I2 > >	959
mln::Site_Proxy< p_double_piter< S, I1, I2 > >	967
mln::Site_Iterator< p_double_piter< S, I1, I2 > >	965
mln::Object< p_double_psite< S, Sp > >	846
mln::Proxy< p_double_psite< S, Sp > >	959
mln::Site_Proxy< p_double_psite< S, Sp > >	967
mln::Pseudo_Site< p_double_psite< S, Sp > >	959
mln::Object< p_edges< G, F > >	846
mln::Site_Set< p_edges< G, F > >	968
site_set_base_< F::result, p_edges< G, F > >	
mln::p_edges< G, F >	861
mln::Object< p_edges_psite< G, F > >	846
mln::Proxy< p_edges_psite< G, F > >	959
mln::Site_Proxy< p_edges_psite< G, F > >	967
mln::Pseudo_Site< p_edges_psite< G, F > >	959
mln::Object< p_faces< N, D, P > >	846
mln::Site_Set< p_faces< N, D, P > >	968
site_set_base_< faces_psite< N, D, P >, p_faces< N, D, P > >	
mln::p_faces< N, D, P >	867
mln::Object< p_graph_piter< S, I > >	846
mln::Proxy< p_graph_piter< S, I > >	959
mln::Site_Proxy< p_graph_piter< S, I > >	967
mln::Site_Iterator< p_graph_piter< S, I > >	965
site_iterator_base< S, p_graph_piter< S, I > >	

site_set_iterator_base< S, p_graph_piter< S, I > >	
mln::p_graph_piter< S, I >	870
mln::Object< p_if< S, F > >	846
mln::Site_Set< p_if< S, F > >	968
site_set_base_< S::psite, p_if< S, F > >	
mln::p_if< S, F >	872
mln::Object< p_image< I > >	846
mln::Site_Set< p_image< I > >	968
site_set_base_< I::psite, p_image< I > >	
mln::p_image< I >	875
mln::Object< p_indexed_bkd_piter< S > >	846
mln::Proxy< p_indexed_bkd_piter< S > >	959
mln::Site_Proxy< p_indexed_bkd_piter< S > >	967
mln::Site_Iterator< p_indexed_bkd_piter< S > >	965
site_iterator_base< S, p_indexed_bkd_piter< S > >	
site_set_iterator_base< S, p_indexed_bkd_piter< S > >	
mln::p_indexed_bkd_piter< S >	879
mln::Object< p_indexed_fwd_piter< S > >	846
mln::Proxy< p_indexed_fwd_piter< S > >	959
mln::Site_Proxy< p_indexed_fwd_piter< S > >	967
mln::Site_Iterator< p_indexed_fwd_piter< S > >	965
site_iterator_base< S, p_indexed_fwd_piter< S > >	
site_set_iterator_base< S, p_indexed_fwd_piter< S > >	
mln::p_indexed_fwd_piter< S >	881
mln::Object< p_indexed_psite< S > >	846
mln::Proxy< p_indexed_psite< S > >	959
mln::Site_Proxy< p_indexed_psite< S > >	967
mln::Pseudo_Site< p_indexed_psite< S > >	959
pseudo_site_base_< const S::element &, p_indexed_psite< S > >	
mln::p_indexed_psite< S >	882
mln::Object< p_key< K, P > >	846
mln::Site_Set< p_key< K, P > >	968
site_set_base_< P, p_key< K, P > >	
mln::p_key< K, P >	882
mln::Object< p_line2d >	846
mln::Site_Set< p_line2d >	968
site_set_base_< point2d, p_line2d >	
mln::p_line2d	888
mln::Object< p_mutable_array_of< S > >	846
mln::Site_Set< p_mutable_array_of< S > >	968
site_set_base_< S::site, p_mutable_array_of< S > >	
mln::p_mutable_array_of< S >	892
mln::Object< p_n_faces_bkd_piter< D, G > >	846
mln::Proxy< p_n_faces_bkd_piter< D, G > >	959
mln::Site_Proxy< p_n_faces_bkd_piter< D, G > >	967
mln::Site_Iterator< p_n_faces_bkd_piter< D, G > >	965
site_iterator_base< p_complex< D, G >, p_n_faces_bkd_piter< D, G > >	
site_set_iterator_base< p_complex< D, G >, p_n_faces_bkd_piter< D, G > >	
p_complex_piter_base_< topo::n_face_bkd_iter< D >, p_complex< D, G >, G::site, p_n_faces_bkd_piter< D, G > >	

mln::p_n_faces_bkd_piter< D, G >	895
mln::Object< p_n_faces_fwd_piter< D, G > >	846
mln::Proxy< p_n_faces_fwd_piter< D, G > >	959
mln::Site_Proxy< p_n_faces_fwd_piter< D, G > >	967
mln::Site_Iterator< p_n_faces_fwd_piter< D, G > >	965
site_iterator_base< p_complex< D, G >, p_n_faces_fwd_piter< D, G > >	
site_set_iterator_base< p_complex< D, G >, p_n_faces_fwd_piter< D, G > >	
p_complex_piter_base< topo::n_face_fwd_iter< D >, p_complex< D, G >, G::site, p_n_faces_fwd_piter< D, G > >	
mln::p_n_faces_fwd_piter< D, G >	896
mln::Object< p_priority< P, Q > >	846
mln::Site_Set< p_priority< P, Q > >	968
site_set_base< Q::site, p_priority< P, Q > >	
mln::p_priority< P, Q >	898
mln::Object< p_queue< P > >	846
mln::Site_Set< p_queue< P > >	968
site_set_base< P, p_queue< P > >	
mln::p_queue< P >	903
mln::Object< p_queue_fast< P > >	846
mln::Site_Set< p_queue_fast< P > >	968
site_set_base< P, p_queue_fast< P > >	
mln::p_queue_fast< P >	908
mln::Object< p_run< P > >	846
mln::Site_Set< p_run< P > >	968
site_set_base< P, p_run< P > >	
mln::p_run< P >	913
mln::Object< p_run_psite< P > >	846
mln::Proxy< p_run_psite< P > >	959
mln::Site_Proxy< p_run_psite< P > >	967
mln::Pseudo_Site< p_run_psite< P > >	959
mln::Object< p_set< P > >	846
mln::Site_Set< p_set< P > >	968
site_set_base< P, p_set< P > >	
mln::p_set< P >	918
mln::Object< p_set_of< S > >	846
mln::Site_Set< p_set_of< S > >	968
site_set_base< S::site, p_set_of< S > >	
mln::p_set_of< S >	922
mln::Object< p_transformed< S, F > >	846
mln::Site_Set< p_transformed< S, F > >	968
site_set_base< S::psite, p_transformed< S, F > >	
mln::p_transformed< S, F >	926
mln::Object< p_transformed_piter< Pi, S, F > >	846
mln::Proxy< p_transformed_piter< Pi, S, F > >	959
mln::Site_Proxy< p_transformed_piter< Pi, S, F > >	967
mln::Site_Iterator< p_transformed_piter< Pi, S, F > >	965
mln::Object< p_vaccess< V, S > >	846
mln::Site_Set< p_vaccess< V, S > >	968
site_set_base< S::site, p_vaccess< V, S > >	

mln::p_vaccess< V, S >	930
mln::Object< p_vertices< G, F > >	846
mln::Site_Set< p_vertices< G, F > >	968
site_set_base_< F::result, p_vertices< G, F > >	
mln::p_vertices< G, F >	934
mln::Object< p_vertices_psite< G, F > >	846
mln::Proxy< p_vertices_psite< G, F > >	959
mln::Site_Proxy< p_vertices_psite< G, F > >	967
mln::Pseudo_Site< p_vertices_psite< G, F > >	959
mln::Object< pair< A1, A2 > >	846
mln::Meta_Accumulator< pair< A1, A2 > >	826
mln::accu::meta::pair< A1, A2 >	479
mln::Object< pair< A1, A2, T > >	846
mln::Proxy< pair< A1, A2, T > >	959
mln::Accumulator< pair< A1, A2, T > >	541
base< std::pair< A1::result, A2::result >, pair< A1, A2, T > >	
mln::accu::pair< A1, A2, T >	498
mln::Object< pair< min< V >, max< V >, mln_argument(min< V >) > >	846
mln::Proxy< pair< min< V >, max< V >, mln_argument(min< V >) > >	959
mln::Accumulator< pair< min< V >, max< V >, mln_argument(min< V >) > >	541
mln::Object< pink_t >	846
mln::Literal< pink_t >	801
mln::literal::pink_t	817
mln::Object< pixel< I > >	846
mln::pixel< I >	941
mln::Object< plain< I > >	846
mln::Image< plain< I > >	764
mln::Object< point< G, C > >	846
mln::Site< point< G, C > >	964
mln::Gpoint< point< G, C > >	729
mln::point< G, C >	948
mln::Object< point_from_value< T > >	846
mln::Function< point_from_value< T > >	717
mln::Function_v2v< point_from_value< T > >	719
mln::Object< projection< P, dir > >	846
mln::Function< projection< P, dir > >	717
mln::Function_v2v< projection< P, dir > >	719
mln::Object< proxy< I > >	846
mln::Proxy< proxy< I > >	959
mln::value::proxy< I >	1134
mln::Object< purple_t >	846
mln::Literal< purple_t >	801
mln::literal::purple_t	818
mln::Object< qrde >	846
mln::Function< qrde >	717
mln::Function_v2v< qrde >	719
mln::Object< qt_rgb_to_int_u< n > >	846
mln::Function< qt_rgb_to_int_u< n > >	717

mln::Function_v2v< qt_rgb_to_int_u< n > >	719
mln::Object< quat >	846
mln::Value< quat >	1108
mln::Object< rank >	846
mln::Meta_Accumulator< rank >	826
mln::accu::meta::stat::rank	491
mln::Object< rank< bool > >	846
mln::Proxy< rank< bool > >	959
mln::Accumulator< rank< bool > >	541
base< bool, rank< bool > >	
mln::accu::stat::rank< bool >	531
mln::Object< rank< T > >	846
mln::Proxy< rank< T > >	959
mln::Accumulator< rank< T > >	541
base< const T &, rank< T > >	
mln::accu::stat::rank< T >	529
mln::Object< rank_high_quant >	846
mln::Meta_Accumulator< rank_high_quant >	826
mln::accu::meta::stat::rank_high_quant	492
mln::Object< rank_high_quant< T > >	846
mln::Proxy< rank_high_quant< T > >	959
mln::Accumulator< rank_high_quant< T > >	541
base< const T &, rank_high_quant< T > >	
mln::accu::stat::rank_high_quant< T >	532
mln::Object< rectangle2d >	846
mln::Window< rectangle2d >	1171
window_base< dpoint2d, rectangle2d >	
classical_window_base< dpoint2d, rectangle2d >	
mln::win::rectangle2d	1170
mln::Object< rectangularity< P > >	846
mln::Proxy< rectangularity< P > >	959
mln::Accumulator< rectangularity< P > >	541
mln::Object< red >	846
mln::Meta_Function< red >	828
mln::Meta_Function_v2v< red >	829
mln::Object< red_t >	846
mln::Literal< red_t >	801
mln::literal::red_t	819
mln::Object< ref_data >	846
mln::Function< ref_data >	717
mln::Function_v2v< ref_data >	719
mln::Object< rgb32 >	846
mln::Value< rgb32 >	1108
mln::Object< rgb8_off_loader >	846
mln::Object< rgb8_off_saver >	846
mln::Object< rgb8_to_rgbn< n > >	846
mln::Function< rgb8_to_rgbn< n > >	717
mln::Function_v2v< rgb8_to_rgbn< n > >	719
mln::fun::v2v::rgb8_to_rgbn< n >	684

mln::Object< rgb< n > >	846
mln::Value< rgb< n > >	1108
mln::Object< rgb_to_dist< T, n > >	846
mln::Function< rgb_to_dist< T, n > >	717
mln::Function_v2v< rgb_to_dist< T, n > >	719
mln::Object< rgb_to_int_u< n > >	846
mln::Function< rgb_to_int_u< n > >	717
mln::Function_v2v< rgb_to_int_u< n > >	719
mln::Object< rgb_to_luma< T_luma > >	846
mln::Function< rgb_to_luma< T_luma > >	717
mln::Function_v2v< rgb_to_luma< T_luma > >	719
mln::Object< rgbn_to_lbl8< n > >	846
mln::Function< rgbn_to_lbl8< n > >	717
mln::Function_v2v< rgbn_to_lbl8< n > >	719
mln::Object< rms >	846
mln::Meta_Accumulator< rms >	826
mln::accu::meta::rms	480
mln::Object< rms< T, V > >	846
mln::Proxy< rms< T, V > >	959
mln::Accumulator< rms< T, V > >	541
base< V, rms< T, V > >	
mln::accu::rms< T, V >	501
mln::Object< rotation< n, C > >	846
mln::Function< rotation< n, C > >	717
mln::Function_v2v< rotation< n, C > >	719
mln::fun::x2x::rotation< n, C >	708
mln::Object< round< R > >	846
mln::Function< round< R > >	717
mln::Function_v2v< round< R > >	719
mln::Object< row >	846
mln::Meta_Function< row >	828
mln::Meta_Function_v2v< row >	829
mln::Object< safe_image< I > >	846
mln::Image< safe_image< I > >	764
mln::Object< saturate< V > >	846
mln::Function< saturate< V > >	717
mln::Function_v2v< saturate< V > >	719
mln::Object< saturate_rgb8 >	846
mln::Function< saturate_rgb8 >	717
mln::Function_v2v< saturate_rgb8 >	719
mln::Object< scomp< ith > >	846
mln::Meta_Function< scomp< ith > >	828
mln::Meta_Function_v2v< scomp< ith > >	829
mln::Object< set_bkd_iter< T > >	846
mln::Proxy< set_bkd_iter< T > >	959
mln::Object< set_fwd_iter< T > >	846
mln::Proxy< set_fwd_iter< T > >	959
mln::Object< sharpness< I > >	846

mln::Proxy< sharpness< I > >	959
mln::Accumulator< sharpness< I > >	541
base< double, sharpness< I > >	
mln::morpho::attribute::sharpness< I >	838
mln::Object< shell< F, I > >	846
mln::Proxy< shell< F, I > >	959
mln::Object< sign >	846
mln::Value< sign >	1108
mln::Object< site_pair< P > >	846
mln::util::site_pair< P >	1088
mln::Object< sli >	846
mln::Meta_Function< sli >	828
mln::Meta_Function_v2v< sli >	829
mln::Object< slice_image< I > >	846
mln::Image< slice_image< I > >	764
mln::Object< snake_fwd_t >	846
mln::Browsing< snake_fwd_t >	567
mln::canvas::browsing::snake_fwd_t	578
mln::Object< snake_generic_t >	846
mln::Browsing< snake_generic_t >	567
mln::canvas::browsing::snake_generic_t	579
mln::Object< snake_vert_t >	846
mln::Browsing< snake_vert_t >	567
mln::canvas::browsing::snake_vert_t	581
mln::Object< soft_heap< T, R > >	846
mln::util::soft_heap< T, R >	1090
mln::Object< sqrt >	846
mln::Function< sqrt >	717
mln::Function_v2v< sqrt >	719
mln::Object< square >	846
mln::Mesh< square >	825
mln::Regular_Grid< square >	962
mln::Object< stack_image< n, I > >	846
mln::Image< stack_image< n, I > >	764
mln::Object< static_n_face_bkd_iter< N, D > >	846
mln::Iterator< static_n_face_bkd_iter< N, D > >	789
complex_iterator_base< topo::face< D >, static_n_face_bkd_iter< N, D > >	
complex_set_iterator_base< topo::face< D >, static_n_face_bkd_iter< N, D > >	
mln::topo::static_n_face_bkd_iter< N, D >	1029
mln::Object< static_n_face_fwd_iter< N, D > >	846
mln::Iterator< static_n_face_fwd_iter< N, D > >	789
complex_iterator_base< topo::face< D >, static_n_face_fwd_iter< N, D > >	
complex_set_iterator_base< topo::face< D >, static_n_face_fwd_iter< N, D > >	
mln::topo::static_n_face_fwd_iter< N, D >	1030
mln::Object< sub_image< I, S > >	846
mln::Image< sub_image< I, S > >	764
mln::Object< sub_image_if< I, S > >	846
mln::Image< sub_image_if< I, S > >	764

mln::Object< sum >	846
mln::Meta_Accumulator< sum >	826
mln::accu::meta::math::sum	474
mln::Object< sum< I, S > >	846
mln::Proxy< sum< I, S > >	959
mln::Accumulator< sum< I, S > >	541
base< S, sum< I, S > >	
mln::morpho::attribute::sum< I, S >	841
mln::Object< sum< T, S > >	846
mln::Proxy< sum< T, S > >	959
mln::Accumulator< sum< T, S > >	541
base< const S &, sum< T, S > >	
mln::accu::math::sum< T, S >	458
mln::Object< sup >	846
mln::Meta_Accumulator< sup >	826
mln::accu::meta::math::sup	475
mln::Meta_Function< sup >	828
mln::Meta_Function_vv2v< sup >	829
mln::Object< sup< T > >	846
mln::Proxy< sup< T > >	959
mln::Accumulator< sup< T > >	541
base< const T &, sup< T > >	
mln::accu::math::sup< T >	460
mln::Object< tautology >	846
mln::Function< tautology >	717
mln::Function_v2v< tautology >	719
mln::Function_v2b< tautology >	718
mln::fun::p2b::tautology	675
mln::Object< teal_t >	846
mln::Literal< teal_t >	801
mln::literal::teal_t	820
mln::Object< test >	846
mln::Function< test >	717
mln::Function_v2v< test >	719
mln::Object< threshold< V > >	846
mln::Function< threshold< V > >	717
mln::Function_v2v< threshold< V > >	719
mln::Function_v2b< threshold< V > >	718
mln::fun::v2b::threshold< V >	678
mln::Object< thru_image< I, F > >	846
mln::Image< thru_image< I, F > >	764
mln::Object< thrubin_image< I1, I2, F > >	846
mln::Image< thrubin_image< I1, I2, F > >	764
mln::Object< tick >	846
mln::Mesh< tick >	825
mln::Regular_Grid< tick >	962
mln::Object< timer >	846
mln::Proxy< timer >	959

mln::util::timer	1093
mln::Object< to16bits >	846
mln::Function< to16bits >	717
mln::Function_v2v< to16bits >	719
mln::Object< to19bits >	846
mln::Function< to19bits >	717
mln::Function_v2v< to19bits >	719
mln::Object< to23bits >	846
mln::Function< to23bits >	717
mln::Function_v2v< to23bits >	719
mln::Object< to27bits >	846
mln::Function< to27bits >	717
mln::Function_v2v< to27bits >	719
mln::Object< to8bits >	846
mln::Function< to8bits >	717
mln::Function_v2v< to8bits >	719
mln::Object< tofloat01 >	846
mln::Function< tofloat01 >	717
mln::Function_v2v< tofloat01 >	719
mln::Object< tr_image< S, I, T > >	846
mln::Image< tr_image< S, I, T > >	764
mln::Object< transformed_image< I, F > >	846
mln::Image< transformed_image< I, F > >	764
mln::Object< translation< n, C > >	846
mln::Function< translation< n, C > >	717
mln::Function_v2v< translation< n, C > >	719
mln::fun::x2x::translation< n, C >	712
mln::Object< translation_t< P > >	846
mln::Function< translation_t< P > >	717
mln::Function_v2v< translation_t< P > >	719
mln::Object< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >	846
mln::Proxy< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >	959
mln::Accumulator< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >	541
base< boost::tuple< BOOST_PP_REPEAT(10, RESULT_ACCU, Le Ricard ya que ca de vrai!) >, tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >	
mln::accu::tuple< A, n, >	538
mln::Object< tuple< n, BOOST_PP_ENUM_PARAMS(10, T)> >	846
mln::Meta_Accumulator< tuple< n, BOOST_PP_ENUM_PARAMS(10, T)> >	826
mln::accu::meta::tuple< n, >	493
mln::Object< unary< Fun, T > >	846
mln::Function< unary< Fun, T > >	717
mln::Function_v2v< unary< Fun, T > >	719
mln::Object< unproject_image< I, D, F > >	846
mln::Image< unproject_image< I, D, F > >	764
mln::Object< up_leaf_piter< T > >	846
mln::Proxy< up_leaf_piter< T > >	959
mln::Site_Proxy< up_leaf_piter< T > >	967
mln::Site_Iterator< up_leaf_piter< T > >	965

mln::Object< up_node_piter< T > >	846
mln::Proxy< up_node_piter< T > >	959
mln::Site_Proxy< up_node_piter< T > >	967
mln::Site_Iterator< up_node_piter< T > >	965
mln::Object< up_site_piter< T > >	846
mln::Proxy< up_site_piter< T > >	959
mln::Site_Proxy< up_site_piter< T > >	967
mln::Site_Iterator< up_site_piter< T > >	965
mln::Object< val< A > >	846
mln::Proxy< val< A > >	959
mln::Accumulator< val< A > >	541
base< const A::result &, val< A > >	
mln::accu::val< A >	540
mln::Object< val< mA > >	846
mln::Meta_Accumulator< val< mA > >	826
mln::accu::meta::val< mA >	494
mln::Object< value_at_index< bool > >	846
mln::Function< value_at_index< bool > >	717
mln::Function_v2v< value_at_index< bool > >	719
mln::Object< value_at_index< T > >	846
mln::Function< value_at_index< T > >	717
mln::Function_v2v< value_at_index< T > >	719
mln::Object< var< T > >	846
mln::Proxy< var< T > >	959
mln::Accumulator< var< T > >	541
base< algebra::mat< T::dim, T::dim, float >, var< T > >	
mln::accu::stat::var< T >	534
mln::Object< variance< T, S, R > >	846
mln::Proxy< variance< T, S, R > >	959
mln::Accumulator< variance< T, S, R > >	541
base< R, variance< T, S, R > >	
mln::accu::stat::variance< T, S, R >	536
mln::Object< vec< 1, T > >	846
mln::Object< vec< 2, T > >	846
mln::Object< vec< 3, T > >	846
mln::Object< vec< 4, T > >	846
mln::Object< vec< n, C > >	846
mln::Object< vec< n, T > >	846
mln::Object< vec< V > >	846
mln::Function< vec< V > >	717
mln::Function_vv2v< vec< V > >	720
mln::fun::vv2v::vec< V >	703
mln::Object< vertex< G > >	846
mln::Site< vertex< G > >	964
mln::util::vertex< G >	1103
mln::Object< vertex_bkd_iterator< G > >	846
mln::Proxy< vertex_bkd_iterator< G > >	959
mln::Object< vertex_fwd_iterator< G > >	846
mln::Proxy< vertex_fwd_iterator< G > >	959

mln::Object< vertex_image< P, V, G > >	846
mln::Image< vertex_image< P, V, G > >	764
mln::Object< vertex_nbh_edge_bkd_iterator< G > >	846
mln::Proxy< vertex_nbh_edge_bkd_iterator< G > >	959
mln::Object< vertex_nbh_edge_fwd_iterator< G > >	846
mln::Proxy< vertex_nbh_edge_fwd_iterator< G > >	959
mln::Object< vertex_nbh_vertex_bkd_iterator< G > >	846
mln::Proxy< vertex_nbh_vertex_bkd_iterator< G > >	959
mln::Object< vertex_nbh_vertex_fwd_iterator< G > >	846
mln::Proxy< vertex_nbh_vertex_fwd_iterator< G > >	959
mln::Object< violent_cast_image< T, I > >	846
mln::Image< violent_cast_image< T, I > >	764
mln::Object< violet_t >	846
mln::Literal< violet_t >	801
mln::literal::violet_t	821
mln::Object< viota_t >	846
mln::Function< viota_t >	717
mln::Function_v2v< viota_t >	719
mln::Object< viota_t< S > >	846
mln::Function< viota_t< S > >	717
mln::Function_v2v< viota_t< S > >	719
mln::Object< volume >	846
mln::Meta_Accumulator< volume >	826
mln::accu::meta::shape::volume	483
mln::Object< volume< I > >	846
mln::Proxy< volume< I > >	959
mln::Accumulator< volume< I > >	541
base< unsigned, volume< I > >	
mln::accu::shape::volume< I >	506
mln::morpho::attribute::volume< I >	842
mln::Object< W >	846
mln::Object< w_window< D, W > >	846
mln::Weighted_Window< w_window< D, W > >	1159
weighted_window_base< mln::window< D >, w_window< D, W > >	
mln::w_window< D, W >	1155
mln::Object< white_gaussian< V > >	846
mln::Function< white_gaussian< V > >	717
mln::Function_n2v< white_gaussian< V > >	718
mln::fun::n2v::white_gaussian< V >	674
mln::Object< white_t >	846
mln::Literal< white_t >	801
mln::literal::white_t	822
mln::Object< window< D > >	846
mln::Window< window< D > >	1171
window_base< D, window< D > >	
mln::window< D >	1172
mln::Object< wrap >	846
mln::Function< wrap >	717

mln::Function_v2v< wrap >	719
mln::Object< wrap< L > >	846
mln::Function< wrap< L > >	717
mln::Function_v2v< wrap< L > >	719
mln::Object< yellow_t >	846
mln::Literal< yellow_t >	801
mln::literal::yellow_t	823
mln::Object< yes >	846
mln::util::yes	1108
mln::Object< zero_t >	846
mln::Literal< zero_t >	801
mln::literal::zero_t	824
mln::internal::pixel_impl_< I, bkd_pixter1d< I > >	
pixel_iterator_base_< I, bkd_pixter1d< I > >	
mln::internal::pixel_impl_< I, bkd_pixter2d< I > >	
pixel_iterator_base_< I, bkd_pixter2d< I > >	
mln::internal::pixel_impl_< I, bkd_pixter3d< I > >	
pixel_iterator_base_< I, bkd_pixter3d< I > >	
mln::internal::pixel_impl_< I, dpoints_bkd_pixter< I > >	
mln::dpoints_bkd_pixter< I >	649
mln::internal::pixel_impl_< I, dpoints_fwd_pixter< I > >	
mln::dpoints_fwd_pixter< I >	651
mln::internal::pixel_impl_< I, fwd_pixter1d< I > >	
pixel_iterator_base_< I, fwd_pixter1d< I > >	
mln::internal::pixel_impl_< I, fwd_pixter2d< I > >	
pixel_iterator_base_< I, fwd_pixter2d< I > >	
mln::internal::pixel_impl_< I, fwd_pixter3d< I > >	
pixel_iterator_base_< I, fwd_pixter3d< I > >	
mln::internal::pixel_impl_< I, pixel< I > >	
mln::pixel< I >	941
trait::graph< I >	1178
trait::graph< mln::complex_image< I, G, V > >	1179
trait::graph< mln::image2d< T > >	1179

Chapter 7

Class Index

7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

mln::accu::center< P, V > (Mass center accumulator)	437
mln::accu::convolve< T1, T2, R > (Generic convolution accumulator class)	439
mln::accu::count_adjacent_vertices< F, S > (Accumulator class counting the number of vertices adjacent to a set of mln::p_edges_psite (i.e., a set of edges))	440
mln::accu::count_labels< L > (Count the number of different labels in an image)	442
mln::accu::count_value< V > (Define an accumulator that counts the occurrence of a given value)	443
mln::accu::histo< V > (Generic histogram class over a value set with type V)	445
mln::accu::label_used< L > (References all the labels used)	446
mln::accu::logic::land ("Logical-and" accumulator)	448
mln::accu::logic::land_basic ("Logical-and" accumulator)	449
mln::accu::logic::lor ("Logical-or" accumulator)	451
mln::accu::logic::lor_basic ("Logical-or" accumulator class)	452
mln::accu::maj_h< T > (Compute the majority value)	454
mln::accu::math::count< T > (Generic counter accumulator)	455
mln::accu::math::inf< T > (Generic inf accumulator class)	457
mln::accu::math::sum< T, S > (Generic sum accumulator class)	458
mln::accu::math::sup< T > (Generic sup accumulator class)	460
mln::accu::max_site< I > (Define an accumulator that computes the first site with the maximum value in an image)	461
mln::accu::meta::center (Meta accumulator for center)	463
mln::accu::meta::count_adjacent_vertices (Meta accumulator for count_adjacent_vertices)	463
mln::accu::meta::count_labels (Meta accumulator for count_labels)	464
mln::accu::meta::count_value (FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for count_value)	465
mln::accu::meta::histo (Meta accumulator for histo)	466
mln::accu::meta::label_used (Meta accumulator for label_used)	467
mln::accu::meta::logic::land (Meta accumulator for land)	468
mln::accu::meta::logic::lor (Meta accumulator for lor)	469
mln::accu::meta::logic::lor_basic (Meta accumulator for lor_basic)	470
mln::accu::meta::maj_h (Meta accumulator for maj_h)	471
mln::accu::meta::math::count (Meta accumulator for count)	472
mln::accu::meta::math::inf (Meta accumulator for inf)	473
mln::accu::meta::math::sum (Meta accumulator for sum)	474

mln::accu::meta::math::sup (Meta accumulator for sup)	475
mln::accu::meta::max_site (Meta accumulator for max_site)	476
mln::accu::meta::nil (Meta accumulator for nil)	477
mln::accu::meta::p< mA > (Meta accumulator for p)	478
mln::accu::meta::pair< A1, A2 > (Meta accumulator for pair)	479
mln::accu::meta::rms (Meta accumulator for rms)	480
mln::accu::meta::shape::bbox (Meta accumulator for bbox)	481
mln::accu::meta::shape::height (Meta accumulator for height)	482
mln::accu::meta::shape::volume (Meta accumulator for volume)	483
mln::accu::meta::stat::max (Meta accumulator for max)	484
mln::accu::meta::stat::max_h (Meta accumulator for max)	485
mln::accu::meta::stat::mean (Meta accumulator for mean)	486
mln::accu::meta::stat::median_alt< T > (Meta accumulator for median_alt)	487
mln::accu::meta::stat::median_h (Meta accumulator for median_h)	488
mln::accu::meta::stat::min (Meta accumulator for min)	489
mln::accu::meta::stat::min_h (Meta accumulator for min)	490
mln::accu::meta::stat::rank (Meta accumulator for rank)	491
mln::accu::meta::stat::rank_high_quant (Meta accumulator for rank_high_quant)	492
mln::accu::meta::tuple< n, > (Meta accumulator for tuple)	493
mln::accu::meta::val< mA > (Meta accumulator for val)	494
mln::accu::nil< T > (Define an accumulator that does nothing)	495
mln::accu::p< A > (Generic p of accumulators)	497
mln::accu::pair< A1, A2, T > (Generic pair of accumulators)	498
mln::accu::rms< T, V > (Generic root mean square accumulator class)	501
mln::accu::shape::bbox< P > (Generic bounding box accumulator class)	503
mln::accu::shape::height< I > (Height accumulator)	504
mln::accu::shape::volume< I > (Volume accumulator class)	506
mln::accu::site_set::rectangularity< P > (Compute the rectangularity of a site set)	508
mln::accu::stat::deviation< T, S, M > (Generic standard deviation accumulator class)	510
mln::accu::stat::histo3d_rgb< V > (Define a histogram as accumulator which returns an image3d)	512
mln::accu::stat::max< T > (Generic max accumulator class)	514
mln::accu::stat::max_h< V > (Generic max function based on histogram over a value set with type V)	516
mln::accu::stat::mean< T, S, M > (Generic mean accumulator class)	517
mln::accu::stat::median_alt< S > (Generic median_alt function based on histogram over a value set with type S)	519
mln::accu::stat::median_h< V > (Generic median function based on histogram over a value set with type V)	521
mln::accu::stat::meta::deviation (Meta accumulator for deviation)	522
mln::accu::stat::min< T > (Generic min accumulator class)	523
mln::accu::stat::min_h< V > (Generic min function based on histogram over a value set with type V)	525
mln::accu::stat::min_max< V > (Generic min and max accumulator class)	526
mln::accu::stat::rank< T > (Generic rank accumulator class)	529
mln::accu::stat::rank< bool > (Rank accumulator class for Boolean)	531
mln::accu::stat::rank_high_quant< T > (Generic rank accumulator class)	532
mln::accu::stat::var< T > (Var accumulator class)	534
mln::accu::stat::variance< T, S, R > (Variance accumulator class)	536
mln::accu::tuple< A, n, > (Generic tuple of accumulators)	538
mln::accu::val< A > (Generic val of accumulators)	540
mln::Accumulator< E > (Base class for implementation of accumulators)	541
mln::algebra::h_mat< d, T > (N-Dimensional matrix with homogeneous coordinates)	543
mln::algebra::h_vec< d, C > (N-Dimensional vector with homogeneous coordinates)	545
mln::bkd_pixterId< I > (Backward pixel iterator on a 1-D image with border)	547

mln::bkd_pixter2d< I > (Backward pixel iterator on a 2-D image with border)	548
mln::bkd_pixter3d< I > (Backward pixel iterator on a 3-D image with border)	550
mln::box< P > (Generic box class: site set containing points of a regular grid)	551
mln::Box< E > (Base class for implementation classes of boxes)	559
mln::box_runend_piter< P > (A generic backward iterator on points by lines)	565
mln::box_runstart_piter< P > (A generic forward iterator on points by lines)	566
mln::Browsing< E > (Base class for implementation classes that are browsings)	567
mln::canvas::browsing::backdiagonal2d_t (Browsing in a certain direction)	568
mln::canvas::browsing::breadth_first_search_t (Breadth-first search algorithm for graph, on vertices)	570
mln::canvas::browsing::depth_first_search_t (Breadth-first search algorithm for graph, on vertices)	570
mln::canvas::browsing::diagonal2d_t (Browsing in a certain direction)	570
mln::canvas::browsing::dir_struct_elt_incr_update_t (Browsing in a certain direction with a segment)	572
mln::canvas::browsing::directional_t (Browsing in a certain direction)	573
mln::canvas::browsing::fwd_t (Canvas for forward browsing)	575
mln::canvas::browsing::hyper_directional_t (Browsing in a certain direction)	576
mln::canvas::browsing::snake_fwd_t (Browsing in a snake-way, forward)	578
mln::canvas::browsing::snake_generic_t (Multidimensional Browsing in a given-way)	579
mln::canvas::browsing::snake_vert_t (Browsing in a snake-way, forward)	581
mln::canvas::chamfer< F > (Compute chamfer distance)	582
mln::category< R(*)(&A) > (Category declaration for a unary C function)	582
mln::complex_image< D, G, V > (Image based on a complex)	583
mln::complex_neighborhood_bkd_piter< I, G, N > (Backward iterator on complex neighborhood)	586
mln::complex_neighborhood_fwd_piter< I, G, N > (Forward iterator on complex neighborhood)	587
mln::complex_psite< D, G > (Point site associated to a mln::p_complex)	589
mln::complex_window_bkd_piter< I, G, W > (Backward iterator on complex window)	592
mln::complex_window_fwd_piter< I, G, W > (Forward iterator on complex window)	594
mln::decorated_image< I, D > (Image that can have additional features)	595
mln::Delta_Point_Site< E > (FIXME: Doc!)	598
mln::Delta_Point_Site< void > (Delta point site category flag type)	599
mln::doc::Accumulator< E > (Documentation class for mln::Accumulator)	599
mln::doc::Box< E > (Documentation class for mln::Box)	600
mln::doc::Dpoint< E > (Documentation class for mln::Dpoint)	603
mln::doc::Fastest_Image< E > (Documentation class for the concept of images that have the speed property set to "fastest")	606
mln::doc::Generalized_Pixel< E > (Documentation class for mln::Generalized_Pixel)	614
mln::doc::Image< E > (Documentation class for mln::Image)	615
mln::doc::Iterator< E > (Documentation class for mln::Iterator)	621
mln::doc::Neighborhood< E > (Documentation class for mln::Neighborhood)	623
mln::doc::Object< E > (Documentation class for mln::Object)	624
mln::doc::Pixel_Iterator< E > (Documentation class for mln::Iterator)	625
mln::doc::Point_Site< E > (Documentation class for mln::Point_Site)	628
mln::doc::Site_Iterator< E > (Documentation class for mln::Site_Iterator)	631
mln::doc::Site_Set< E > (Documentation class for mln::Site_Set)	633
mln::doc::Value_Iterator< E > (Documentation class for mln::Value_Iterator)	635
mln::doc::Value_Set< E > (Documentation class for mln::Value_Set)	636
mln::doc::Weighted_Window< E > (Documentation class for mln::Weighted_Window)	639
mln::doc::Window< E > (Documentation class for mln::Window)	641
mln::Dpoint< E > (Base class for implementation of delta-point classes)	643
mln::dpoint< G, C > (Generic delta-point class)	644

<code>mln::dpoints_bkd_pixter< I ></code> (A generic backward iterator on the pixels of a dpoint-based window or neighborhood)	649
<code>mln::dpoints_fwd_pixter< I ></code> (A generic forward iterator on the pixels of a dpoint-based window or neighborhood)	651
<code>mln::dpsites_bkd_piter< V ></code> (A generic backward iterator on points of windows and of neighborhoods)	654
<code>mln::dpsites_fwd_piter< V ></code> (A generic forward iterator on points of windows and of neighborhoods)	655
<code>mln::Edge< E ></code> (Edge category flag type)	657
<code>mln::edge_image< P, V, G ></code> (Image based on graph edges)	657
<code>mln::extended< I ></code> (Makes an image become restricted by a point set)	659
<code>mln::extension_fun< I, F ></code> (Extends the domain of an image with a function)	661
<code>mln::extension_ima< I, J ></code> (Extends the domain of an image with an image)	663
<code>mln::extension_val< I ></code> (Extends the domain of an image with a value)	666
<code>mln::faces_psite< N, D, P ></code> (Point site associated to a <code>mln::p_faces</code>)	668
<code>mln::flat_image< T, S ></code> (Image with a single value)	671
<code>mln::fun::from_accu< A ></code> (Wrap an accumulator into a function)	673
<code>mln::fun::n2v::white_gaussian< V ></code> (Generate a White Gaussian Noise)	674
<code>mln::fun::p2b::antilogy</code> (A <code>p2b</code> function always returning <code>false</code>)	674
<code>mln::fun::p2b::tautology</code> (A <code>p2b</code> function always returning <code>true</code>)	675
<code>mln::fun::v2b::lnot< V ></code> (Functor computing logical-not on a value)	677
<code>mln::fun::v2b::threshold< V ></code> (Threshold function)	678
<code>mln::fun::v2v::ch_function_value< F, V ></code> (Wrap a function <code>v2v</code> and convert its result to another type)	679
<code>mln::fun::v2v::component< T, i ></code> (Functor that accesses the <code>i</code> -th component of a value)	680
<code>mln::fun::v2v::l1_norm< V, R ></code> (L1-norm)	680
<code>mln::fun::v2v::l2_norm< V, R ></code> (L2-norm)	681
<code>mln::fun::v2v::linear< V, T, R ></code> (Linear function. $f(v) = a * v + b$. <code>V</code> is the type of input values; <code>T</code> is the type used to compute the result; <code>R</code> is the result type)	682
<code>mln::fun::v2v::linfty_norm< V, R ></code> (L-infty norm)	683
<code>mln::fun::v2v::rgb8_to_rgbn< n ></code> (Convert a <code>rgb8</code> value to a <code>rgn</code> , $n < 8$)	684
<code>mln::fun::v2w2v::cos< V ></code> (Cosinus bijective functor)	686
<code>mln::fun::v2w_w2v::l1_norm< V, R ></code> (L1-norm)	687
<code>mln::fun::v2w_w2v::l2_norm< V, R ></code> (L2-norm)	689
<code>mln::fun::v2w_w2v::linfty_norm< V, R ></code> (L-infty norm)	690
<code>mln::fun::vv2b::eq< L, R ></code> (Functor computing equal between two values)	691
<code>mln::fun::vv2b::ge< L, R ></code> (Functor computing "greater or equal than" between two values)	691
<code>mln::fun::vv2b::gt< L, R ></code> (Functor computing "greater than" between two values)	692
<code>mln::fun::vv2b::implies< L, R ></code> (Functor computing logical-implies between two values)	693
<code>mln::fun::vv2b::le< L, R ></code> (Functor computing "lower or equal than" between two values)	694
<code>mln::fun::vv2b::lt< L, R ></code> (Functor computing "lower than" between two values)	695
<code>mln::fun::vv2v::diff_abs< V ></code> (A functor computing the <code>diff_abs</code> imum of two values)	696
<code>mln::fun::vv2v::land< L, R ></code> (Functor computing logical-and between two values)	697
<code>mln::fun::vv2v::land_not< L, R ></code> (Functor computing logical-and-not between two values)	698
<code>mln::fun::vv2v::lor< L, R ></code> (Functor computing logical-or between two values)	699
<code>mln::fun::vv2v::lxor< L, R ></code> (Functor computing logical-xor between two values)	700
<code>mln::fun::vv2v::max< V ></code> (A functor computing the maximum of two values)	701
<code>mln::fun::vv2v::min< L, R ></code> (A functor computing the minimum of two values)	702
<code>mln::fun::vv2v::vec< V ></code> (A functor computing the vecimum of two values)	703
<code>mln::fun::x2p::closest_point< P ></code> (FIXME: doxygen + concept checking)	704
<code>mln::fun::x2v::bilinear< I ></code> (Represent a bilinear interolation of values from an underlying image)	705
<code>mln::fun::x2v::trilinear< I ></code> (Represent a trilinear interolation of values from an underlying image)	706

mln::fun::x2x::composed< T2, T1 > (Represent a composition of two transformations)	706
mln::fun::x2x::linear< I > (Represent a linear interolation of values from an underlying image)	707
mln::fun::x2x::rotation< n, C > (Represent a rotation function)	708
mln::fun::x2x::translation< n, C > (Translation function-object)	712
mln::fun_image< F, I > (Image read through a function)	714
mln::Function< E > (Base class for implementation of function-objects)	717
mln::Function< void > (Function category flag type)	717
mln::Function_n2v< E > (Base class for implementation of function-objects from Nil to value)	718
mln::Function_v2b< E > (Base class for implementation of function-objects from a value to a Boolean)	718
mln::Function_v2v< E > (Base class for implementation of function-objects from value to value)	719
mln::Function_vv2b< E > (Base class for implementation of function-objects from a couple of values to a Boolean)	720
mln::Function_vv2v< E > (Base class for implementation of function-objects from a couple of values to a value)	720
mln::fwd_pixter1d< I > (Forward pixel iterator on a 1-D image with border)	721
mln::fwd_pixter2d< I > (Forward pixel iterator on a 2-D image with border)	722
mln::fwd_pixter3d< I > (Forward pixel iterator on a 3-D image with border)	724
mln::Gdpoint< E > (FIXME: Doc!)	725
mln::Gdpoint< void > (Delta point site category flag type)	726
mln::Generalized_Pixel< E > (Base class for implementation classes that are pixels or that have the behavior of pixels)	727
mln::geom::complex_geometry< D, P > (A functor returning the sites of the faces of a complex where the locations of each 0-face is stored)	727
mln::Gpoint< E > (Base class for implementation of point classes)	729
mln::Graph< E > (Base class for implementation of graph classes)	733
mln::graph::attribute::card_t (Compute the cardinality of every component in a graph)	734
mln::graph::attribute::representative_t (Compute the representative vertex of every component in a graph)	734
mln::graph_elt_mixed_neighborhood< G, S, S2 > (Elementary neighborhood on graph class)	735
mln::graph_elt_mixed_window< G, S, S2 > (Elementary window on graph class)	736
mln::graph_elt_neighborhood< G, S > (Elementary neighborhood on graph class)	740
mln::graph_elt_neighborhood_if< G, S, I > (Elementary neighborhood_if on graph class)	742
mln::graph_elt_window< G, S > (Elementary window on graph class)	744
mln::graph_elt_window_if< G, S, I > (Custom window on graph class)	748
mln::graph_window_base< P, E >	753
mln::graph_window_if_piter< S, W, I > (Forward iterator on line graph window)	755
mln::graph_window_piter< S, W, I > (Forward iterator on line graph window)	757
mln::hexa< I > (Hexagonal image class)	760
mln::histo::array< T > (Generic histogram class over a value set with type T)	764
mln::Image< E > (Base class for implementation of image classes)	764
mln::image1d< T > (Basic 1D image class)	766
mln::image2d< T > (Basic 2D image class)	771
mln::image2d_h< V > (2d image based on an hexagonal mesh)	776
mln::image3d< T > (Basic 3D image class)	779
mln::image_if< I, F > (Image which domain is restricted by a function 'site -> Boolean')	784
mln::interpolated< I, F > (Makes the underlying image being accessed with floating coordinates)	786
mln::io::dicom::dicom_header (Store dicom file header)	788
mln::io::dump::dump_header (Store dump file header)	788
mln::io::fld::fld_header (Define the header structure of an AVS field data file)	789
mln::io::raw::raw_header (Store raw file header)	789
mln::Iterator< E > (Base class for implementation classes that are iterators)	789
mln::labeled_image< I > (Morpher providing an improved interface for labeled image)	791

<code>mln::labeled_image_base< I, E ></code> (Base class Morpher providing an improved interface for labeled image)	795
<code>mln::lazy_image< I, F, B ></code> (Image values are computed on the fly)	798
<code>mln::Literal< E ></code> (Base class for implementation classes of literals)	801
<code>mln::literal::black_t</code> (Type of literal black)	803
<code>mln::literal::blue_t</code> (Type of literal blue)	804
<code>mln::literal::brown_t</code> (Type of literal brown)	804
<code>mln::literal::cyan_t</code> (Type of literal cyan)	805
<code>mln::literal::green_t</code> (Type of literal green)	806
<code>mln::literal::identity_t</code> (Type of literal identity)	807
<code>mln::literal::light_gray_t</code> (Type of literal grays)	808
<code>mln::literal::lime_t</code> (Type of literal lime)	809
<code>mln::literal::magenta_t</code> (Type of literal magenta)	810
<code>mln::literal::max_t</code> (Type of literal max)	811
<code>mln::literal::min_t</code> (Type of literal min)	812
<code>mln::literal::olive_t</code> (Type of literal olive)	813
<code>mln::literal::one_t</code> (Type of literal one)	814
<code>mln::literal::orange_t</code> (Type of literal orange)	815
<code>mln::literal::origin_t</code> (Type of literal origin)	816
<code>mln::literal::pink_t</code> (Type of literal pink)	817
<code>mln::literal::purple_t</code> (Type of literal purple)	818
<code>mln::literal::red_t</code> (Type of literal red)	819
<code>mln::literal::teal_t</code> (Type of literal teal)	820
<code>mln::literal::violet_t</code> (Type of literal violet)	821
<code>mln::literal::white_t</code> (Type of literal white)	822
<code>mln::literal::yellow_t</code> (Type of literal yellow)	823
<code>mln::literal::zero_t</code> (Type of literal zero)	824
<code>mln::Mesh< E ></code> (Base class for implementation classes of meshes)	825
<code>mln::Meta_Accumulator< E ></code> (Base class for implementation of meta accumulators)	826
<code>mln::Meta_Function< E ></code> (Base class for implementation of meta functions)	828
<code>mln::Meta_Function_v2v< E ></code> (Base class for implementation of function-objects from value to value)	829
<code>mln::Meta_Function_vv2v< E ></code> (Base class for implementation of function-objects from value to value)	829
<code>mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 ></code> (Ands type)	830
<code>mln::metal::converts_to< T, U ></code> ("converts-to" check)	830
<code>mln::metal::equal< T1, T2 ></code> (Definition of a static 'equal' test)	830
<code>mln::metal::goes_to< T, U ></code> ("goes-to" check)	831
<code>mln::metal::is< T, U ></code> ("is" check)	831
<code>mln::metal::is_a< T, M ></code> ("is_a" check)	832
<code>mln::metal::is_not< T, U ></code> ("is_not" check)	832
<code>mln::metal::is_not_a< T, M ></code> ("is_not_a" static Boolean expression)	832
<code>mln::mixed_neighb< W ></code> (Adapter class from window to neighborhood)	832
<code>mln::morpho::attribute::card< I ></code> (Cardinality accumulator class)	834
<code>mln::morpho::attribute::count_adjacent_vertices< I ></code> (Count_Adjacent_Vertices accumulator class)	835
<code>mln::morpho::attribute::height< I ></code> (Height accumulator class)	837
<code>mln::morpho::attribute::sharpness< I ></code> (Sharpness accumulator class)	838
<code>mln::morpho::attribute::sum< I, S ></code> (Suminality accumulator class)	841
<code>mln::morpho::attribute::volume< I ></code> (Volume accumulator class)	842
<code>mln::neighb< W ></code> (Adapter class from window to neighborhood)	844
<code>mln::Neighborhood< E ></code> (Base class for implementation classes that are neighborhoods)	846
<code>mln::Neighborhood< void ></code> (Neighborhood category flag type)	846
<code>mln::Object< E ></code> (Base class for almost every class defined in Milena)	846

<code>mln::p2p_image< I, F ></code> (FIXME: Doc!)	847
<code>mln::p_array< P ></code> (Multi-set of sites)	849
<code>mln::p_centered< W ></code> (Site set corresponding to a window centered on a site)	854
<code>mln::p_complex< D, G ></code> (A complex psite set based on the N-faces of a complex of dimension D (a D-complex))	857
<code>mln::p_edges< G, F ></code> (Site set mapping graph edges and image sites)	861
<code>mln::p_faces< N, D, P ></code> (A complex psite set based on a the N-faces of a complex of dimension D (a D-complex))	867
<code>mln::p_graph_piter< S, I ></code> (Generic iterator on point sites of a <code>mln::S</code>)	870
<code>mln::p_if< S, F ></code> (Site set restricted w.r.t)	872
<code>mln::p_image< I ></code> (Site set based on an image of Booleans)	875
<code>mln::p_indexed_bkd_piter< S ></code> (Backward iterator on sites of an indexed site set)	879
<code>mln::p_indexed_fwd_piter< S ></code> (Forward iterator on sites of an indexed site set)	881
<code>mln::p_indexed_psite< S ></code> (Psite class for indexed site sets such as <code>p_array</code>)	882
<code>mln::p_key< K, P ></code> (Priority queue class)	882
<code>mln::p_line2d</code> (2D discrete line of points)	888
<code>mln::p_mutable_array_of< S ></code> (<code>P_mutable_array_of</code> is a mutable array of site sets)	892
<code>mln::p_n_faces_bkd_piter< D, G ></code> (Backward iterator on the n-faces sites of an <code>mln::p_- complex<D, G></code>)	895
<code>mln::p_n_faces_fwd_piter< D, G ></code> (Forward iterator on the n-faces sites of an <code>mln::p_- complex<D, G></code>)	896
<code>mln::p_priority< P, Q ></code> (Priority queue)	898
<code>mln::p_queue< P ></code> (Queue of sites (based on <code>std::deque</code>))	903
<code>mln::p_queue_fast< P ></code> (Queue of sites class (based on <code>p_array</code>))	908
<code>mln::p_run< P ></code> (Point set class in run)	913
<code>mln::p_set< P ></code> (Mathematical set of sites (based on <code>util::set</code>))	918
<code>mln::p_set_of< S ></code> (<code>P_set_of</code> is a set of site sets)	922
<code>mln::p_transformed< S, F ></code> (Site set transformed through a function)	926
<code>mln::p_transformed_piter< Pi, S, F ></code> (Iterator on <code>p_transformed<S,F></code>)	929
<code>mln::p_vaccess< V, S ></code> (Site set in which sites are grouped by their associated value)	930
<code>mln::p_vertices< G, F ></code> (Site set based mapping graph vertices to sites)	934
<code>mln::pixel< I ></code> (Generic pixel class)	941
<code>mln::Pixel_Iterator< E ></code> (Base class for the implementation of pixel iterator classes)	942
<code>mln::plain< I ></code> (Prevents an image from sharing its data)	943
<code>mln::Point< P ></code> (Base class for implementation of point classes)	945
<code>mln::point< G, C ></code> (Generic point class)	948
<code>mln::Point_Site< E ></code> (Base class for implementation classes of the notion of "point site")	955
<code>mln::Point_Site< void ></code> (Point site category flag type)	958
<code>mln::Proxy< E ></code> (Base class for implementation classes of the notion of "proxy")	959
<code>mln::Proxy< void ></code> (Proxy category flag type)	959
<code>mln::Pseudo_Site< E ></code> (Base class for implementation classes of the notion of "pseudo site")	959
<code>mln::Pseudo_Site< void ></code> (Pseudo_Site category flag type)	960
<code>mln::pw::image< F, S ></code> (A generic point-wise image implementation)	960
<code>mln::registration::closest_point_basic< P ></code> (Closest point functor based on map distance)	961
<code>mln::registration::closest_point_with_map< P ></code> (Closest point functor based on map distance)	962
<code>mln::Regular_Grid< E ></code> (Base class for implementation classes of regular grids)	962
<code>mln::safe_image< I ></code> (Makes an image accessible at undefined location)	963
<code>mln::select::p_of< P ></code> (Structure <code>p_of</code>)	963
<code>mln::Site< E ></code> (Base class for classes that are explicitly sites)	964
<code>mln::Site< void ></code> (Site category flag type)	965
<code>mln::Site_Iterator< E ></code> (Base class for implementation of classes of iterator on points)	965
<code>mln::Site_Proxy< E ></code> (Base class for implementation classes of the notion of "site proxy")	967
<code>mln::Site_Proxy< void ></code> (Site_Proxy category flag type)	968
<code>mln::Site_Set< E ></code> (Base class for implementation classes of site sets)	968

<code>mln::Site_Set< void ></code> (Site_Set category flag type)	971
<code>mln::slice_image< I ></code> (2D image extracted from a slice of a 3D image)	971
<code>mln::sub_image< I, S ></code> (Image having its domain restricted by a site set)	974
<code>mln::sub_image_if< I, S ></code> (Image having its domain restricted by a site set and a function)	975
<code>mln::thru_image< I, F ></code> (Morph image values through a function)	977
<code>mln::thru_bin_image< I1, I2, F ></code> (Morphes values from two images through a binary function)	977
<code>mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D ></code> (Backward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an <code>mln::complex<D></code>)	978
<code>mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D ></code> (Forward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an <code>mln::complex<D></code>)	980
<code>mln::topo::adj_higher_face_bkd_iter< D ></code> (Backward iterator on all the adjacent (n+1)-faces of the n-face of an <code>mln::complex<D></code>)	981
<code>mln::topo::adj_higher_face_fwd_iter< D ></code> (Forward iterator on all the adjacent (n+1)-faces of the n-face of an <code>mln::complex<D></code>)	982
<code>mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D ></code> (Backward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an <code>mln::complex<D></code>)	983
<code>mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D ></code> (Forward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an <code>mln::complex<D></code>)	984
<code>mln::topo::adj_lower_face_bkd_iter< D ></code> (Backward iterator on all the adjacent (n-1)-faces of the n-face of an <code>mln::complex<D></code>)	985
<code>mln::topo::adj_lower_face_fwd_iter< D ></code> (Forward iterator on all the adjacent (n-1)-faces of the n-face of an <code>mln::complex<D></code>)	986
<code>mln::topo::adj_lower_higher_face_bkd_iter< D ></code> (Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an <code>mln::complex<D></code>)	987
<code>mln::topo::adj_lower_higher_face_fwd_iter< D ></code> (Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an <code>mln::complex<D></code>)	989
<code>mln::topo::adj_m_face_bkd_iter< D ></code> (Backward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex)	990
<code>mln::topo::adj_m_face_fwd_iter< D ></code> (Forward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex)	991
<code>mln::topo::algebraic_face< D ></code> (Algebraic face handle in a complex; the face dimension is dynamic)	992
<code>mln::topo::algebraic_n_face< N, D ></code> (Algebraic N-face handle in a complex)	998
<code>mln::topo::center_only_iter< D ></code> (Iterator on all the adjacent (n-1)-faces of the n-face of an <code>mln::complex<D></code>)	1002
<code>mln::topo::centered_bkd_iter_adapter< D, I ></code> (Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces)	1003
<code>mln::topo::centered_fwd_iter_adapter< D, I ></code> (Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces)	1005
<code>mln::topo::complex< D ></code> (General complex of dimension D)	1006
<code>mln::topo::face< D ></code> (Face handle in a complex; the face dimension is dynamic)	1009
<code>mln::topo::face_bkd_iter< D ></code> (Backward iterator on all the faces of an <code>mln::complex<D></code>)	1013
<code>mln::topo::face_fwd_iter< D ></code> (Forward iterator on all the faces of an <code>mln::complex<D></code>)	1014
<code>mln::topo::is_n_face< N ></code> (A functor testing wheter a <code>mln::complex_psite</code> is an N-face)	1016
<code>mln::topo::is_simple_2d_t< N ></code> (Test if a point is simple or not)	1017
<code>mln::topo::is_simple_cell< I ></code> (A predicate for the simplicity of a point based on the collapse property of the attachment)	1017
<code>mln::topo::n_face< N, D ></code> (N-face handle in a complex)	1020
<code>mln::topo::n_face_bkd_iter< D ></code> (Backward iterator on all the faces of an <code>mln::complex<D></code>)	1024
<code>mln::topo::n_face_fwd_iter< D ></code> (Forward iterator on all the faces of an <code>mln::complex<D></code>)	1025

<code>mln::topo::n_faces_set< N, D ></code> (Set of face handles of dimension <code>N</code>)	1027
<code>mln::topo::skeleton::is_simple_point< N ></code>	1028
<code>mln::topo::static_n_face_bkd_iter< N, D ></code> (Backward iterator on all the <code>N</code> -faces of a <code>mln::complex<D></code>)	1029
<code>mln::topo::static_n_face_fwd_iter< N, D ></code> (Forward iterator on all the <code>N</code> -faces of a <code>mln::complex<D></code>)	1030
<code>mln::tr_image< S, I, T ></code> (Transform an image by a given transformation)	1031
<code>mln::unproject_image< I, D, F ></code> (Un-projects an image)	1034
<code>mln::util::adjacency_matrix< V ></code> (A class of adjacency matrix)	1036
<code>mln::util::array< T ></code> (A dynamic array class)	1037
<code>mln::util::branch< T ></code> (Class of generic branch)	1044
<code>mln::util::branch_iter< T ></code> (Basic 2D image class)	1045
<code>mln::util::branch_iter_ind< T ></code> (Basic 2D image class)	1047
<code>mln::util::couple< T, U ></code> (Definition of a couple)	1048
<code>mln::util::eat</code> (Eat structure)	1050
<code>mln::util::edge< G ></code> (Edge of a graph <code>G</code>)	1051
<code>mln::util::fibonacci_heap< P, T ></code> (Fibonacci heap)	1055
<code>mln::util::graph</code> (Undirected graph)	1059
<code>mln::util::greater_point< I ></code> (A "greater than" functor comparing points w.r.t)	1066
<code>mln::util::greater_psite< I ></code> (A "greater than" functor comparing psites w.r.t)	1067
<code>mln::util::head< T, R ></code> (Top structure of the soft heap)	1067
<code>mln::util::ignore</code> (Ignore structure)	1068
<code>mln::util::ilcell< T ></code> (Element of an item list. Store the data (key) used in <code>soft_heap</code>)	1068
<code>mln::util::line_graph< G ></code> (Undirected line graph of a graph of type <code>G</code>)	1069
<code>mln::util::nil</code> (Nil structure)	1075
<code>mln::util::node< T, R ></code> (Meta-data of an element in the heap)	1076
<code>mln::util::object_id< Tag, V ></code> (Base class of an object id)	1076
<code>mln::util::ord< T ></code> (Function-object that defines an ordering between objects with type <code>T</code> : <i>lhs</i> <code>R rhs</code>)	1078
<code>mln::util::ord_pair< T ></code> (Ordered pair structure s.a)	1078
<code>mln::util::pix< I ></code> (Structure <code>pix</code>)	1081
<code>mln::util::set< T ></code> (An "efficient" mathematical set class)	1082
<code>mln::util::site_pair< P ></code> (A pair of sites)	1088
<code>mln::util::soft_heap< T, R ></code> (Soft heap)	1090
<code>mln::util::timer</code> (Timer structure)	1093
<code>mln::util::tracked_ptr< T ></code> (Smart pointer for shared data with tracking)	1094
<code>mln::util::tree< T ></code> (Class of generic tree)	1096
<code>mln::util::tree_node< T ></code> (Class of generic <code>tree_node</code> for tree)	1098
<code>mln::util::vertex< G ></code> (Vertex of a graph <code>G</code>)	1103
<code>mln::util::yes</code> (Object that always says "yes")	1108
<code>mln::Value< E ></code> (Base class for implementation classes of values)	1108
<code>mln::value::float01</code> (Class for floating values restricted to the interval <code>[0..1]</code> and discretized with <code>n</code> bits)	1110
<code>mln::value::float01_f</code> (Class for floating values restricted to the interval <code>[0..1]</code>)	1112
<code>mln::value::graylevel< n ></code> (General gray-level class on <code>n</code> bits)	1113
<code>mln::value::graylevel_f</code> (General gray-level class on <code>n</code> bits)	1117
<code>mln::value::int_s< n ></code> (Signed integer value class)	1119
<code>mln::value::int_u< n ></code> (Unsigned integer value class)	1122
<code>mln::value::int_u_sat< n ></code> (Unsigned integer value class with saturation behavior)	1124
<code>mln::value::Integer< E ></code> (Concept of integer)	1126
<code>mln::value::Integer< void ></code> (Category flag type)	1127
<code>mln::value::label< n ></code> (Label value class)	1127
<code>mln::value::lut_vec< S, T ></code> (Class that defines <code>FIXME</code>)	1130
<code>mln::value::proxy< I ></code> (Generic proxy class for an image pixel value)	1134

<code>mln::value::qt::rgb32</code> (Color class for red-green-blue where every component is n-bit encoded) .	1136
<code>mln::value::rgb< n ></code> (Color class for red-green-blue where every component is n-bit encoded)	1138
<code>mln::value::set< T ></code> (Class that defines the set of values of type <code>T</code>)	1140
<code>mln::value::sign</code> (Value type composed by the set (-1, 0, 1) sign value type is a subset of the int value type)	1141
<code>mln::value::stack_image< n, I ></code> (Stack image class)	1143
<code>mln::value::super_value< sign ></code> (Specializations:)	1146
<code>mln::value::value_array< T, V ></code> (Generic array class over indexed by a value set with type <code>T</code>) .	1146
<code>mln::Value_Iterator< E ></code> (Base class for implementation of classes of iterator on values) . . .	1147
<code>mln::Value_Set< E ></code> (Base class for implementation classes of sets of values)	1149
<code>mln::Vertex< E ></code> (<code>Vertex</code> category flag type)	1150
<code>mln::vertex_image< P, V, G ></code> (<code>Image</code> based on graph vertices)	1151
<code>mln::violent_cast_image< T, I ></code> (Violently cast image values to a given type)	1153
<code>mln::w_window< D, W ></code> (Generic <code>w_window</code> class)	1155
<code>mln::Weighted_Window< E ></code> (Base class for implementation classes that are weighted_ windows)	1159
<code>mln::win::backdiag2d</code> (Diagonal line window defined on the 2D square grid)	1160
<code>mln::win::ball< G, C ></code> (Generic ball window defined on a given grid)	1161
<code>mln::win::cube3d</code> (Cube window defined on the 3D grid)	1162
<code>mln::win::cuboid3d</code> (Cuboid defined on the 3-D square grid)	1163
<code>mln::win::diag2d</code> (Diagonal line window defined on the 2D square grid)	1165
<code>mln::win::line< M, i, C ></code> (Generic line window defined on a given grid in the given dimension)	1166
<code>mln::win::multiple< W, F ></code> (Multiple window)	1168
<code>mln::win::multiple_size< n, W, F ></code> (Definition of a multiple-size window)	1168
<code>mln::win::octagon2d</code> (Octagon window defined on the 2D square grid)	1169
<code>mln::win::rectangle2d</code> (Rectangular window defined on the 2D square grid)	1170
<code>mln::Window< E ></code> (Base class for implementation classes that are windows)	1171
<code>mln::window< D ></code> (Generic window class)	1172
<code>mln::world::inter_pixel::is_separator</code> (Functor returning whether a site is a separator in an inter-pixel image)	1177
<code>trait::graph< I ></code> (Graph traits)	1178
<code>trait::graph< mln::complex_image< 1, G, V > ></code> (Graph traits for 1-complexes images)	1179
<code>trait::graph< mln::image2d< T > ></code> (Graph traits for <code>mln::image2d</code>)	1179

Chapter 8

Module Documentation

8.1 On site sets

Accumulators working on site sets.

Classes

- struct `mln::accu::center< P, V >`
Mass center accumulator.
- struct `mln::accu::math::count< T >`
Generic counter accumulator.
- struct `mln::accu::shape::bbox< P >`
Generic bounding box accumulator class.
- class `mln::accu::site_set::rectangularity< P >`
Compute the rectangularity of a site set.

8.1.1 Detailed Description

Accumulators working on site sets.

8.2 On images

Accumulators working on images.

Classes

- struct `mln::accu::count_adjacent_vertices< F, S >`
Accumulator class counting the number of vertices adjacent to a set of `mln::p_edges_psite` (i.e., a set of edges).

- struct `mln::accu::max_site< I >`
Define an accumulator that computes the first site with the maximum value in an image.
- struct `mln::accu::shape::height< I >`
Height accumulator.
- struct `mln::accu::shape::volume< I >`
Volume accumulator class.

8.2.1 Detailed Description

Accumulators working on images.

8.3 On values

Accumulators working on image values.

Classes

- struct `mln::accu::convolve< T1, T2, R >`
Generic convolution accumulator class.
- struct `mln::accu::count_labels< L >`
Count the number of different labels in an image.
- struct `mln::accu::count_value< V >`
Define an accumulator that counts the occurrence of a given value.
- struct `mln::accu::histo< V >`
Generic histogram class over a value set with type V.
- struct `mln::accu::label_used< L >`
References all the labels used.
- struct `mln::accu::logic::land`
"Logical-and" accumulator.
- struct `mln::accu::logic::land_basic`
"Logical-and" accumulator.
- struct `mln::accu::logic::lor`
"Logical-or" accumulator.
- struct `mln::accu::logic::lor_basic`
"Logical-or" accumulator class.

- struct `mln::accu::maj_h< T >`
Compute the majority value.
- struct `mln::accu::math::inf< T >`
Generic inf accumulator class.
- struct `mln::accu::math::sum< T, S >`
Generic sum accumulator class.
- struct `mln::accu::math::sup< T >`
Generic sup accumulator class.
- struct `mln::accu::rms< T, V >`
Generic root mean square accumulator class.
- struct `mln::accu::stat::deviation< T, S, M >`
Generic standard deviation accumulator class.
- struct `mln::accu::stat::histo3d_rgb< V >`
Define a histogram as accumulator which returns an `image3d`.
- struct `mln::accu::stat::max< T >`
Generic max accumulator class.
- struct `mln::accu::stat::max_h< V >`
Generic max function based on histogram over a value set with type `V`.
- struct `mln::accu::stat::mean< T, S, M >`
Generic mean accumulator class.
- struct `mln::accu::stat::median_alt< S >`
Generic `median_alt` function based on histogram over a value set with type `S`.
- struct `mln::accu::stat::median_h< V >`
Generic median function based on histogram over a value set with type `V`.
- struct `mln::accu::stat::min< T >`
Generic min accumulator class.
- struct `mln::accu::stat::min_h< V >`
Generic min function based on histogram over a value set with type `V`.
- struct `mln::accu::stat::min_max< V >`
Generic min and max accumulator class.
- struct `mln::accu::stat::rank< T >`
Generic rank accumulator class.
- struct `mln::accu::stat::rank< bool >`
rank accumulator class for Boolean.

- struct `mln::accu::stat::rank_high_quant< T >`
Generic rank accumulator class.
- struct `mln::accu::stat::var< T >`
Var accumulator class.
- struct `mln::accu::stat::variance< T, S, R >`
Variance accumulator class.

8.3.1 Detailed Description

Accumulators working on image values.

8.4 Multiple accumulators

Set of special accumulators for computing several accumulators at the same time.

Classes

- struct `mln::accu::pair< A1, A2, T >`
Generic pair of accumulators.
- struct `mln::accu::tuple< A, n, >`
Generic tuple of accumulators.

8.4.1 Detailed Description

Set of special accumulators for computing several accumulators at the same time.

8.5 Graphes

All graphes implementations.

Classes

- class `mln::util::graph`
Undirected graph.
- class `mln::util::line_graph< G >`
Undirected line graph of a graph of type G.

8.5.1 Detailed Description

All graphes implementations.

8.6 Images

All the generic image types provided in Olena.

Modules

- [Basic types](#)
Concrete images.
- [Image morphers](#)
Morpher on both image values and domain.
- [Values morphers](#)
Morpher on image values.
- [Domain morphers](#)
Morpher on image domain.
- [Identity morphers](#)
Morpher adding new fonctionnalities.

8.6.1 Detailed Description

All the generic image types provided in Olena.

8.7 Basic types

Concrete images.

Classes

- class [mln::complex_image< D, G, V >](#)
Image based on a complex.
- class [mln::edge_image< P, V, G >](#)
Image based on graph edges.
- struct [mln::flat_image< T, S >](#)
Image with a single value.
- struct [mln::imageId< T >](#)

Basic 1D image class.

- class `mln::image2d< T >`
Basic 2D image class.
- struct `mln::image2d_h< V >`
2d image based on an hexagonal mesh.
- struct `mln::image3d< T >`
Basic 3D image class.
- class `mln::pw::image< F, S >`
A generic point-wise image implementation.
- class `mln::vertex_image< P, V, G >`
Image based on graph vertices.

8.7.1 Detailed Description

Concrete images.

8.8 Image morphers

Morpher on both image values and domain.

Morpher on both image values and domain.

8.9 Values morphers

Morpher on image values.

Classes

- struct `mln::fun_image< F, I >`
Image read through a function.
- class `mln::thru_image< I, F >`
Morph image values through a function.
- class `mln::thrubin_image< I1, I2, F >`
Morphes values from two images through a binary function.
- struct `mln::violent_cast_image< T, I >`
Violently cast image values to a given type.

8.9.1 Detailed Description

Morpher on image values.

8.10 Domain morphers

Morpher on image domain.

Classes

- struct `mln::extended< I >`
Makes an image become restricted by a point set.
- class `mln::extension_fun< I, F >`
Extends the domain of an image with a function.
- class `mln::extension_ima< I, J >`
Extends the domain of an image with an image.
- class `mln::extension_val< I >`
Extends the domain of an image with a value.
- struct `mln::hexa< I >`
hexagonal image class.
- struct `mln::image_if< I, F >`
Image which domain is restricted by a function 'site -> Boolean'.
- struct `mln::p2p_image< I, F >`
FIXME: Doc!
- struct `mln::slice_image< I >`
2D image extracted from a slice of a 3D image.
- class `mln::sub_image< I, S >`
Image having its domain restricted by a site set.
- struct `mln::sub_image_if< I, S >`
Image having its domain restricted by a site set and a function.
- struct `mln::unproject_image< I, D, F >`
Un-projects an image.

8.10.1 Detailed Description

Morpher on image domain.

8.11 Identity morphers

Morpher adding new fonctionnalités.

Classes

- struct `mln::decorated_image< I, D >`
Image that can have additional features.
- class `mln::labeled_image< I >`
Morpher providing an improved interface for labeled image.
- struct `mln::lazy_image< I, F, B >`
Image values are computed on the fly.
- class `mln::plain< I >`
Prevents an image from sharing its data.
- class `mln::safe_image< I >`
Makes an image accessible at undefined location.
- struct `mln::tr_image< S, I, T >`
Transform an image by a given transformation.

8.11.1 Detailed Description

Morpher adding new fonctionnalités.

8.12 Types

Milena Object types.

Modules

- [Graphes](#)
All graphes implementations.
- [Images](#)
All the generic image types provided in Olena.
- [Neighborhoods](#)
All the predefined generic neighborhoods.
- [Site sets](#)
All Site set types.

- [Utilities](#)

Miscellaneous useful containers/structures.

- [Windows](#)

All the predefined generic windows.

8.12.1 Detailed Description

Milena Object types.

8.13 Accumulators

All accumulator types.

Modules

- [On site sets](#)

Accumulators working on site sets.

- [On images](#)

Accumulators working on images.

- [On values](#)

Accumulators working on image values.

- [Multiple accumulators](#)

Set of special accumulators for computing several accumulators at the same time.

8.13.1 Detailed Description

All accumulator types.

8.14 Routines

All algorithms/routines provided in Milena.

All algorithms/routines provided in Milena.

8.15 Canvas

All canvas.

All canvas.

8.16 Functions

All predefined functions.

Classes

- struct [mln::Function< E >](#)
Base class for implementation of function-objects.
- struct [mln::Function_n2v< E >](#)
Base class for implementation of function-objects from Nil to value.
- struct [mln::Function_v2b< E >](#)
Base class for implementation of function-objects from a value to a Boolean.
- struct [mln::Function_v2v< E >](#)
Base class for implementation of function-objects from value to value.
- struct [mln::Function_vv2b< E >](#)
Base class for implementation of function-objects from a couple of values to a Boolean.
- struct [mln::Function_vv2v< E >](#)
Base class for implementation of function-objects from a couple of values to a value.

Namespaces

- namespace [mln::fun::i2v](#)
Namespace of integer-to-value functions.
- namespace [mln::fun::n2v](#)
Namespace of functions from nil to value.
- namespace [mln::fun::stat](#)
Namespace of statistical functions.
- namespace [mln::fun::v2i](#)
Namespace of value-to-integer functions.
- namespace [mln::fun::v2v](#)
Namespace of functions from value to value.

Modules

- [v2w2v functions](#)
All bijective functions.

- [v2w_w2v functions](#)
All bijective function.
- [vv2b functions](#)
All functions mapping two values to a logical value.

8.16.1 Detailed Description

All predefined functions.

8.17 Neighborhoods

All the predefined generic neighborhoods.

Modules

- [1D neighborhoods](#)
Predefined 1D neighborhoods.
- [2D neighborhoods](#)
Predefined 2D neighborhoods.
- [3D neighborhoods](#)
Predefined 3D neighborhoods.

8.17.1 Detailed Description

All the predefined generic neighborhoods.

8.18 1D neighborhoods

Predefined 1D neighborhoods.

Typedefs

- `typedef neighb< window1d > mln::neighb1d`
Type alias for a neighborhood defined on the 1D square grid with integer coordinates.

Functions

- `const neighb1d & mln::c2 ()`
2-connectivity neighborhood on the 1D grid.

8.18.1 Detailed Description

Predefined 1D neighborhoods.

8.18.2 Typedef Documentation

8.18.2.1 `typedef neighb<window1d> mln::neighb1d`

Type alias for a neighborhood defined on the 1D square grid with integer coordinates.

Definition at line 47 of file `neighb1d.hh`.

8.18.3 Function Documentation

8.18.3.1 `const neighb1d & mln::c2 () [inline]`

2-connectivity neighborhood on the 1D grid.

○ × ○

Returns

A `neighb1d`.

Definition at line 67 of file `neighb1d.hh`.

8.19 2D neighborhoods

Predefined 2D neighborhoods.

Typedefs

- `typedef neighb< window2d > mln::neighb2d`
Type alias for a neighborhood defined on the 2D square grid with integer coordinates.

Functions

- `const neighb2d & mln::c2_col ()`
Vertical 2-connectivity neighborhood on the 2D grid.
- `const neighb2d & mln::c2_row ()`
Horizontal 2-connectivity neighborhood on the 2D grid.
- `const neighb2d & mln::c4 ()`
4-connectivity neighborhood on the 2D grid.
- `const neighb2d & mln::c8 ()`
8-connectivity neighborhood on the 2D grid.

8.19.1 Detailed Description

Predefined 2D neighborhoods.

8.19.2 Typedef Documentation

8.19.2.1 typedef neighb<window2d> mln::neighb2d

Type alias for a neighborhood defined on the 2D square grid with integer coordinates.

Definition at line 51 of file core/alias/neighb2d.hh.

8.19.3 Function Documentation

8.19.3.1 const neighb2d & mln::c2_col () [inline]

Vertical 2-connectivity neighborhood on the 2D grid.

```
- o -
- x -
- o -
```

Returns

A neighb2d.

Definition at line 190 of file core/alias/neighb2d.hh.

8.19.3.2 const neighb2d & mln::c2_row () [inline]

Horizontal 2-connectivity neighborhood on the 2D grid.

```
- - -
o x o
- - -
```

Returns

A neighb2d.

Definition at line 176 of file core/alias/neighb2d.hh.

8.19.3.3 const neighb2d & mln::c4 () [inline]

4-connectivity neighborhood on the 2D grid.

```
- o -
o x o
- o -
```

Returns

A `neighb2d`.

Definition at line 148 of file `core/alias/neighb2d.hh`.

8.19.3.4 const neighb2d & mln::c8 () [inline]

8-connectivity neighborhood on the 2D grid.

```

○ ○ ○
○ × ○
○ ○ ○

```

Returns

A `neighb2d`.

Definition at line 162 of file `core/alias/neighb2d.hh`.

8.20 3D neighborhoods

Predefined 3D neighborhoods.

Typedefs

- `typedef neighb< window3d > mln::neighb3d`
Type alias for a neighborhood defined on the 3D square grid with integer coordinates.

Functions

- `const neighb3d & mln::c18 ()`
18-connectivity neighborhood on the 3D grid.
- `const neighb3d & mln::c26 ()`
26-connectivity neighborhood on the 3D grid.
- `const neighb3d & mln::c2_3d_sli ()`
depth 2-connectivity neighborhood on the 3D grid.
- `const neighb3d & mln::c4_3d ()`
4-connectivity neighborhood on the 3D grid.
- `const neighb3d & mln::c6 ()`
6-connectivity neighborhood on the 3D grid.
- `const neighb3d & mln::c8_3d ()`
8-connectivity neighborhood on the 3D grid.

8.20.1 Detailed Description

Predefined 3D neighborhoods.

8.20.2 Typedef Documentation

8.20.2.1 `typedef neighb<window3d> mln::neighb3d`

Type alias for a neighborhood defined on the 3D square grid with integer coordinates.

Definition at line 50 of file `neighb3d.hh`.

8.20.3 Function Documentation

8.20.3.1 `const neighb3d & mln::c18 () [inline]`

18-connectivity neighborhood on the 3D grid.

```

  . o .
  o o o
  . o .

  o o o
  o x o
  o o o

  . o .
  o o o
  . o .

```

Returns

A `neighb3d`.

Definition at line 288 of file `neighb3d.hh`.

References `mln::c6()`, `mln::window< D >::insert()`, and `mln::win::sym()`.

Referenced by `mln::c26()`.

8.20.3.2 `const neighb3d & mln::c26 () [inline]`

26-connectivity neighborhood on the 3D grid.

```

  o o o
  o o o
  o o o

  o o o
  o x o
  o o o

  o o o
  o o o

```

o o o

Returns

A `neighb3d`.

Definition at line 309 of file `neighb3d.hh`.

References `mln::c18()`, `mln::window< D >::insert()`, and `mln::win::sym()`.

8.20.3.3 `const neighb3d & mln::c2_3d_sli () [inline]`

depth 2-connectivity neighborhood on the 3D grid.

```

. . .
. o .
. . .

. . .
. x .
. . .

. . .
. o .
. . .

```

Returns

A `neighb3d`.

Definition at line 226 of file `neighb3d.hh`.

References `mln::window< D >::insert()`.

8.20.3.4 `const neighb3d & mln::c4_3d () [inline]`

4-connectivity neighborhood on the 3D grid.

```

. . .
. . .
. . .

. o .
o x o
. o .

. . .
. . .
. . .

```

Returns

A `neighb3d`.

Definition at line 241 of file neighb3d.hh.

References `mln::window< D >::insert()`, and `mln::win::sym()`.

8.20.3.5 `const neighb3d & mln::c6 () [inline]`

6-connectivity neighborhood on the 3D grid.

```

. . .
. o .
. . .

. o .
o x o
. o .

. . .
. o .
. . .

```

Returns

A `neighb3d`.

Definition at line 271 of file neighb3d.hh.

References `mln::window< D >::insert()`, and `mln::win::sym()`.

Referenced by `mln::c18()`.

8.20.3.6 `const neighb3d & mln::c8_3d () [inline]`

8-connectivity neighborhood on the 3D grid.

```

. . .
. . .
. . .

o o o
o x o
o o o

. . .
. . .
. . .

```

Returns

A `neighb3d`.

Definition at line 257 of file neighb3d.hh.

8.21 Site sets

All Site set types.

Modules

- [Basic types](#)
Basic site sets.
- [Graph based](#)
Site sets based on a graph.
- [Complex based](#)
Site sets based on a complexes.
- [Sparse types](#)
Sparse site sets.
- [Queue based](#)
Site sets based on a queue.

8.21.1 Detailed Description

All Site set types.

8.22 Basic types

Basic site sets.

Classes

- class [mln::box< P >](#)
Generic box class: site set containing points of a regular grid.
- class [mln::p_line2d](#)
2D discrete line of points.
- class [mln::p_mutable_array_of< S >](#)
p_mutable_array_of is a mutable array of site sets.
- class [mln::p_run< P >](#)
Point set class in run.

8.22.1 Detailed Description

Basic site sets.

8.23 Graph based

Site sets based on a graph.

Classes

- class `mln::p_edges< G, F >`
Site set mapping graph edges and image sites.
- struct `mln::p_faces< N, D, P >`
A complex psite set based on the N-faces of a complex of dimension D (a D-complex).
- class `mln::p_vertices< G, F >`
Site set based mapping graph vertices to sites.

8.23.1 Detailed Description

Site sets based on a graph.

8.24 Complex based

Site sets based on a complexes.

Classes

- class `mln::p_complex< D, G >`
A complex psite set based on the N-faces of a complex of dimension D (a D-complex).

8.24.1 Detailed Description

Site sets based on a complexes.

8.25 Sparse types

Sparse site sets.

Classes

- class `mln::p_array< P >`
Multi-set of sites.
- class `mln::p_centered< W >`
Site set corresponding to a window centered on a site.

- class `mln::p_if< S, F >`
Site set restricted w.r.t.
- class `mln::p_image< I >`
Site set based on an image of Booleans.
- class `mln::p_set< P >`
Mathematical set of sites (based on `util::set`).
- class `mln::p_transformed< S, F >`
Site set transformed through a function.
- class `mln::p_vaccess< V, S >`
Site set in which sites are grouped by their associated value.

8.25.1 Detailed Description

Sparse site sets.

8.26 Queue based

Site sets based on a queue.

Classes

- class `mln::p_key< K, P >`
Priority queue class.
- class `mln::p_priority< P, Q >`
Priority queue.
- class `mln::p_queue< P >`
Queue of sites (based on `std::deque`).
- class `mln::p_queue_fast< P >`
Queue of sites class (based on `p_array`).

8.26.1 Detailed Description

Site sets based on a queue.

8.27 Utilities

Miscellaneous useful containers/structures.

Classes

- class `mln::util::adjacency_matrix< V >`
A class of adjacency matrix.
- class `mln::util::array< T >`
A dynamic array class.
- class `mln::util::couple< T, U >`
Definition of a couple.
- struct `mln::util::eat`
Eat structure.
- class `mln::util::fibonacci_heap< P, T >`
Fibonacci heap.
- struct `mln::util::ignore`
Ignore structure.
- struct `mln::util::nil`
Nil structure.
- struct `mln::util::ord_pair< T >`
Ordered pair structure s.a.
- class `mln::util::set< T >`
An "efficient" mathematical set class.
- class `mln::util::site_pair< P >`
A pair of sites.
- class `mln::util::soft_heap< T, R >`
Soft heap.
- struct `mln::util::tracked_ptr< T >`
Smart pointer for shared data with tracking.
- struct `mln::util::yes`
Object that always says "yes".

8.27.1 Detailed Description

Miscellaneous useful containers/structures.

8.28 Windows

All the predefined generic windows.

Modules

- [1D windows](#)
Predefined 1D windows.
- [2D windows](#)
Predefined 2D windows.
- [3D windows](#)
Predefined 3D windows.
- [N-D windows](#)
Predefined N-D windows.
- [Multiple windows](#)
Generic multiple windows.

8.28.1 Detailed Description

All the predefined generic windows.

8.29 1D windows

Predefined 1D windows.

Typedefs

- typedef line< grid::tick, 0, def::coord > [mln::win::segment1d](#)
Segment window defined on the 1D grid.
- typedef window< [mln::dpoint1d](#) > [mln::window1d](#)
Type alias for a window with arbitrary shape, defined on the 1D square grid with integer coordinates.

8.29.1 Detailed Description

Predefined 1D windows.

8.29.2 Typedef Documentation

8.29.2.1 typedef line<grid::tick, 0, def::coord> mln::win::segment1d

Segment window defined on the 1D grid.

An `segment1d` is centered and symmetric; so its height (length) is odd.

For instance:

○ × ○

is defined with length = 3.

Definition at line 56 of file segment1d.hh.

8.29.2.2 typedef window<mln::dpoint1d> mln::window1d

Type alias for a window with arbitrary shape, defined on the 1D square grid with integer coordinates.

Definition at line 47 of file window1d.hh.

8.30 2D windows

Predefined 2D windows.

Classes

- struct [mln::win::backdiag2d](#)
Diagonal line window defined on the 2D square grid.
- struct [mln::win::diag2d](#)
Diagonal line window defined on the 2D square grid.
- struct [mln::win::octagon2d](#)
Octagon window defined on the 2D square grid.
- struct [mln::win::rectangle2d](#)
Rectangular window defined on the 2D square grid.

Typedefs

- typedef ball< grid::square, def::coord > [mln::win::disk2d](#)
2D disk window; precisely, ball-shaped window defined on the 2D square grid.
- typedef line< grid::square, 1, def::coord > [mln::win::hline2d](#)
Horizontal line window defined on the 2D square grid.
- typedef line< grid::square, 0, def::coord > [mln::win::vline2d](#)
Vertical line window defined on the 2D square grid.
- typedef window< [mln::dpoint2d](#) > [mln::window2d](#)
Type alias for a window with arbitrary shape, defined on the 2D square grid with integer coordinates.

Functions

- `const window2d & mln::win_c4p ()`
4-connectivity window on the 2D grid, including the center.
- `const window2d & mln::win_c8p ()`
8-connectivity window on the 2D grid, including the center.

8.30.1 Detailed Description

Predefined 2D windows.

8.30.2 Typedef Documentation

8.30.2.1 `typedef ball<grid::square, def::coord> mln::win::disk2d`

2D disk window; precisely, ball-shaped window defined on the 2D square grid.

Definition at line 49 of file `disk2d.hh`.

8.30.2.2 `typedef line<grid::square, 1, def::coord> mln::win::hline2d`

Horizontal line window defined on the 2D square grid.

An `hline2d` is centered and symmetric; so its height is 1 and its width (length) is odd.

For instance:

```
o o x o o
```

is defined with `length = 5`.

Definition at line 57 of file `hline2d.hh`.

8.30.2.3 `typedef line<grid::square, 0, def::coord> mln::win::vline2d`

Vertical line window defined on the 2D square grid.

An `vline2d` is centered and symmetric; so its width is 1 and its height (length) is odd.

For instance:

```
o
x
o
```

is defined with `length = 3`.

Definition at line 58 of file `vline2d.hh`.

8.30.2.4 typedef window<mln::dpoint2d> mln::window2d

Type alias for a window with arbitrary shape, defined on the 2D square grid with integer coordinates.

Definition at line 49 of file window2d.hh.

8.30.3 Function Documentation**8.30.3.1 const window2d & mln::win_c4p () [inline]**

4-connectivity window on the 2D grid, including the center.

```

- o -
o x o
- o -

```

Returns

A window2d.

Definition at line 105 of file window2d.hh.

References mln::window< D >::insert(), and mln::window< D >::size().

8.30.3.2 const window2d & mln::win_c8p () [inline]

8-connectivity window on the 2D grid, including the center.

```

o o o
o x o
o o o

```

Returns

A window2d.

Definition at line 121 of file window2d.hh.

References mln::window< D >::insert(), and mln::window< D >::size().

8.31 3D windows

Predefined 3D windows.

Classes

- struct [mln::win::cube3d](#)
Cube window defined on the 3D grid.
- struct [mln::win::cuboid3d](#)
Cuboid defined on the 3-D square grid.

Typedefs

- typedef line< grid::cube, 0, def::coord > [mln::win::sline3d](#)
Depth line window defined on the 3D cubic grid.
- typedef ball< grid::cube, def::coord > [mln::win::sphere3d](#)
3D sphere window; precisely, ball-shaped window defined on the 3D cubic grid.
- typedef window< [mln::dpoint3d](#) > [mln::window3d](#)
Type alias for a window with arbitrary shape, defined on the 3D square grid with integer coordinates.

Functions

- const window3d & [mln::win_c4p_3d](#) ()
4-connectivity window on the 3D grid, including the center.
- const window3d & [mln::win_c8p_3d](#) ()
8-connectivity window on the 3D grid, including the center.

8.31.1 Detailed Description

Predefined 3D windows.

8.31.2 Typedef Documentation

8.31.2.1 typedef line<grid::cube, 0, def::coord> mln::win::sline3d

Depth line window defined on the 3D cubic grid.

An sline3d is centered and symmetric; so its height and its width are 1 and its depth is odd.

For instance:

```

. . .
. o .
. . .

. . .
. x .
. . .

. . .
. o .
. . .

```

is defined with length = 3.

Definition at line 68 of file sline3d.hh.

8.31.2.2 typedef ball<grid::cube, def::coord> mln::win::sphere3d

3D sphere window; precisely, ball-shaped window defined on the 3D cubic grid.

Definition at line 47 of file sphere3d.hh.

8.31.2.3 typedef window<mln::dpoint3d> mln::window3d

Type alias for a window with arbitrary shape, defined on the 3D square grid with integer coordinates.

Definition at line 48 of file window3d.hh.

8.31.3 Function Documentation**8.31.3.1 const window3d & mln::win_c4p_3d () [inline]**

4-connectivity window on the 3D grid, including the center.

```

- - -
- - -
- - -
  - o -
  o x o
  - o -
  - - -
  - - -
  - - -

```

Returns

A window3d.

Definition at line 119 of file window3d.hh.

References mln::window< D >::insert(), and mln::window< D >::size().

8.31.3.2 const window3d & mln::win_c8p_3d () [inline]

8-connectivity window on the 3D grid, including the center.

```

- - -
- - -
- - -
  o o o
  o x o
  o o o
  - - -
  - - -
  - - -

```

Returns

A window3d.

Definition at line 135 of file window3d.hh.

References `mln::window< D >::insert()`, and `mln::window< D >::size()`.

8.32 N-D windows

Predefined N-D windows.

Classes

- struct `mln::win::ball< G, C >`
Generic ball window defined on a given grid.
- struct `mln::win::line< M, i, C >`
Generic line window defined on a given grid in the given dimension.

8.32.1 Detailed Description

Predefined N-D windows.

8.33 Multiple windows

Generic multiple windows.

Classes

- class `mln::win::multiple< W, F >`
Multiple window.
- class `mln::win::multiple_size< n, W, F >`
Definition of a multiple-size window.

8.33.1 Detailed Description

Generic multiple windows.

8.34 v2w2v functions

All bijective functions.

All bijective functions.

8.35 v2w_w2v functions

All bijective function.

All bijective function.

8.36 vv2b functions

All functions mapping two values to a logical value.

All functions mapping two values to a logical value.

Chapter 9

Namespace Documentation

9.1 mln Namespace Reference

[mln/convert/to_image.hh](#)

Namespaces

- namespace [accu](#)
Namespace of accumulators.
- namespace [algebra](#)
Namespace of algebraic structure.
- namespace [arith](#)
Namespace of arithmetic.
- namespace [binarization](#)
Namespace of "point-wise" expression tools.
- namespace [border](#)
Namespace of routines related to image virtual (outer) border.
- namespace [canvas](#)
Namespace of canvas.
- namespace [convert](#)
Namespace of conversion routines.
- namespace [data](#)
Namespace of image processing routines related to pixel data.
- namespace [debug](#)
Namespace of routines that help to debug.
- namespace [def](#)

Namespace for core definitions.

- namespace [display](#)
Namespace of routines that help to display images.
- namespace [doc](#)
The namespace `mln::doc` is only for documentation purpose.
- namespace [draw](#)
Namespace of drawing routines.
- namespace [estim](#)
Namespace of estimation materials.
- namespace [extension](#)
Namespace of extension tools.
- namespace [fun](#)
Namespace of functions.
- namespace [geom](#)
Namespace of all things related to geometry.
- namespace [graph](#)
Namespace of graph related routines.
- namespace [grid](#)
Namespace of grids definitions.
- namespace [histo](#)
Namespace of histograms.
- namespace [impl](#)
Implementation namespace of mln namespace.
- namespace [io](#)
Namespace of input/output handling.
- namespace [labeling](#)
Namespace of labeling routines.
- namespace [linear](#)
Namespace of linear image processing routines.
- namespace [literal](#)
Namespace of literals.
- namespace [logical](#)
Namespace of logic.

- namespace [make](#)
Namespace of routines that help to make Milena's objects.
- namespace [math](#)
Namespace of mathematical routines.
- namespace [metal](#)
Namespace of meta-programming tools.
- namespace [morpho](#)
Namespace of mathematical morphology routines.
- namespace [norm](#)
Namespace of norms.
- namespace [opt](#)
Namespace of optional routines.
- namespace [pw](#)
Namespace of "point-wise" expression tools.
- namespace [registration](#)
Namespace of "point-wise" expression tools.
- namespace [select](#)
Select namespace (FIXME doc).
- namespace [set](#)
Namespace of image processing routines related to pixel sets.
- namespace [subsampling](#)
Namespace of "point-wise" expression tools.
- namespace [tag](#)
Namespace of image processing routines related to tags.
- namespace [test](#)
Namespace of image processing routines related to pixel tests.
- namespace [topo](#)
Namespace of "point-wise" expression tools.
- namespace [trace](#)
Namespace of routines related to the trace mechanism.
- namespace [trait](#)
Namespace where traits are defined.
- namespace [transform](#)
Namespace of transforms.

- namespace [util](#)
Namespace of tools using for more complex algorithm.
- namespace [value](#)
Namespace of materials related to pixel value types.
- namespace [win](#)
Namespace of image processing routines related to win.

Classes

- struct [Accumulator](#)
Base class for implementation of accumulators.
- class [bkd_pixter1d](#)
Backward pixel iterator on a 1-D image with border.
- class [bkd_pixter2d](#)
Backward pixel iterator on a 2-D image with border.
- class [bkd_pixter3d](#)
Backward pixel iterator on a 3-D image with border.
- class [box](#)
Generic box class: site set containing points of a regular grid.
- struct [Box](#)
Base class for implementation classes of boxes.
- class [box_runend_piter](#)
A generic backward iterator on points by lines.
- class [box_runstart_piter](#)
A generic forward iterator on points by lines.
- struct [Browsing](#)
Base class for implementation classes that are browsings.
- struct [category< R\(*\) \(A\) >](#)
Category declaration for a unary C function.
- class [complex_image](#)
Image based on a complex.
- class [complex_neighborhood_bkd_piter](#)
Backward iterator on complex neighborhood.

- class [complex_neighborhood_fwd_piter](#)
Forward iterator on complex neighborhood.
- class [complex_psite](#)
Point site associated to a `mln::p_complex`.
- class [complex_window_bkd_piter](#)
Backward iterator on complex window.
- class [complex_window_fwd_piter](#)
Forward iterator on complex window.
- struct [decorated_image](#)
Image that can have additional features.
- struct [Delta_Point_Site](#)
FIXME: Doc!
- struct [Delta_Point_Site< void >](#)
Delta point site category flag type.
- struct [dpoint](#)
Generic delta-point class.
- struct [Dpoint](#)
Base class for implementation of delta-point classes.
- class [dpoints_bkd_pixter](#)
A generic backward iterator on the pixels of a dpoint-based window or neighborhood.
- class [dpoints_fwd_pixter](#)
A generic forward iterator on the pixels of a dpoint-based window or neighborhood.
- class [dpsites_bkd_piter](#)
A generic backward iterator on points of windows and of neighborhoods.
- class [dpsites_fwd_piter](#)
A generic forward iterator on points of windows and of neighborhoods.
- struct [Edge](#)
edge category flag type.
- class [edge_image](#)
Image based on graph edges.
- struct [extended](#)
Makes an image become restricted by a point set.
- class [extension_fun](#)
Extends the domain of an image with a function.

- class [extension_ima](#)
Extends the domain of an image with an image.
- class [extension_val](#)
Extends the domain of an image with a value.
- class [faces_psite](#)
Point site associated to a `mln::p_faces`.
- struct [flat_image](#)
Image with a single value.
- struct [fun_image](#)
Image read through a function.
- struct [Function](#)
Base class for implementation of function-objects.
- struct [Function< void >](#)
Function category flag type.
- struct [Function_n2v](#)
Base class for implementation of function-objects from Nil to value.
- struct [Function_v2b](#)
Base class for implementation of function-objects from a value to a Boolean.
- struct [Function_v2v](#)
Base class for implementation of function-objects from value to value.
- struct [Function_vv2b](#)
Base class for implementation of function-objects from a couple of values to a Boolean.
- struct [Function_vv2v](#)
Base class for implementation of function-objects from a couple of values to a value.
- class [fwd_pixter1d](#)
Forward pixel iterator on a 1-D image with border.
- class [fwd_pixter2d](#)
Forward pixel iterator on a 2-D image with border.
- class [fwd_pixter3d](#)
Forward pixel iterator on a 3-D image with border.
- struct [Gdpoint](#)
FIXME: Doc!
- struct [Gdpoint< void >](#)

Delta point site category flag type.

- struct [Generalized_Pixel](#)
Base class for implementation classes that are pixels or that have the behavior of pixels.
- struct [Gpoint](#)
Base class for implementation of point classes.
- struct [Graph](#)
Base class for implementation of graph classes.
- struct [graph_elt_mixed_neighborhood](#)
Elementary neighborhood on graph class.
- class [graph_elt_mixed_window](#)
Elementary window on graph class.
- struct [graph_elt_neighborhood](#)
Elementary neighborhood on graph class.
- struct [graph_elt_neighborhood_if](#)
Elementary neighborhood_if on graph class.
- class [graph_elt_window](#)
Elementary window on graph class.
- class [graph_elt_window_if](#)
Custom window on graph class.
- class [graph_window_base](#)
- class [graph_window_if_piter](#)
Forward iterator on line graph window.
- class [graph_window_piter](#)
Forward iterator on line graph window.
- struct [hexa](#)
hexagonal image class.
- struct [Image](#)
Base class for implementation of image classes.
- struct [image1d](#)
Basic 1D image class.
- class [image2d](#)
Basic 2D image class.
- struct [image2d_h](#)
2d image based on an hexagonal mesh.

- struct [image3d](#)
Basic 3D image class.
- struct [image_if](#)
Image which domain is restricted by a function 'site -> Boolean'.
- struct [interpolated](#)
Makes the underlying image being accessed with floating coordinates.
- struct [Iterator](#)
Base class for implementation classes that are iterators.
- class [labeled_image](#)
Morpher providing an improved interface for labeled image.
- class [labeled_image_base](#)
Base class Morpher providing an improved interface for labeled image.
- struct [lazy_image](#)
Image values are computed on the fly.
- struct [Literal](#)
Base class for implementation classes of literals.
- struct [Mesh](#)
Base class for implementation classes of meshes.
- struct [Meta_Accumulator](#)
Base class for implementation of meta accumulators.
- struct [Meta_Function](#)
Base class for implementation of meta functions.
- struct [Meta_Function_v2v](#)
Base class for implementation of function-objects from value to value.
- struct [Meta_Function_vv2v](#)
Base class for implementation of function-objects from value to value.
- class [mixed_neighb](#)
Adapter class from window to neighborhood.
- class [neighb](#)
Adapter class from window to neighborhood.
- struct [Neighborhood](#)
Base class for implementation classes that are neighborhoods.
- struct [Neighborhood< void >](#)

Neighborhood category flag type.

- struct [Object](#)
Base class for almost every class defined in Milena.
- struct [p2p_image](#)
FIXME: Doc!
- class [p_array](#)
Multi-set of sites.
- class [p_centered](#)
Site set corresponding to a window centered on a site.
- class [p_complex](#)
A complex psite set based on the N-faces of a complex of dimension D (a D-complex).
- class [p_edges](#)
Site set mapping graph edges and image sites.
- struct [p_faces](#)
A complex psite set based on a the N-faces of a complex of dimension D (a D-complex).
- class [p_graph_piter](#)
Generic iterator on point sites of a `mln::S`.
- class [p_if](#)
Site set restricted w.r.t.
- class [p_image](#)
Site set based on an image of Booleans.
- class [p_indexed_bkd_piter](#)
Backward iterator on sites of an indexed site set.
- class [p_indexed_fwd_piter](#)
Forward iterator on sites of an indexed site set.
- class [p_indexed_psite](#)
Psite class for indexed site sets such as [p_array](#).
- class [p_key](#)
Priority queue class.
- class [p_line2d](#)
2D discrete line of points.
- class [p_mutable_array_of](#)
[p_mutable_array_of](#) is a mutable array of site sets.

- class [p_n_faces_bkd_piter](#)
Backward iterator on the n-faces sites of an `mln::p_complex<D, G>`.
- class [p_n_faces_fwd_piter](#)
Forward iterator on the n-faces sites of an `mln::p_complex<D, G>`.
- class [p_priority](#)
Priority queue.
- class [p_queue](#)
Queue of sites (based on `std::deque`).
- class [p_queue_fast](#)
Queue of sites class (based on `p_array`).
- class [p_run](#)
Point set class in run.
- class [p_set](#)
Mathematical set of sites (based on `util::set`).
- class [p_set_of](#)
`p_set_of` is a set of site sets.
- class [p_transformed](#)
Site set transformed through a function.
- struct [p_transformed_piter](#)
Iterator on `p_transformed<S,F>`.
- class [p_vaccess](#)
Site set in which sites are grouped by their associated value.
- class [p_vertices](#)
Site set based mapping graph vertices to sites.
- struct [pixel](#)
Generic pixel class.
- struct [Pixel_Iterator](#)
Base class for the implementation of pixel iterator classes.
- class [plain](#)
Prevents an image from sharing its data.
- struct [point](#)
Generic point class.
- struct [Point](#)
Base class for implementation of point classes.

- struct [Point_Site](#)
Base class for implementation classes of the notion of "point site".
- struct [Point_Site< void >](#)
Point site category flag type.
- struct [Proxy](#)
Base class for implementation classes of the notion of "proxy".
- struct [Proxy< void >](#)
Proxy category flag type.
- struct [Pseudo_Site](#)
Base class for implementation classes of the notion of "pseudo site".
- struct [Pseudo_Site< void >](#)
Pseudo_Site category flag type.
- struct [Regular_Grid](#)
Base class for implementation classes of regular grids.
- class [safe_image](#)
Makes an image accessible at undefined location.
- struct [Site](#)
Base class for classes that are explicitly sites.
- struct [Site< void >](#)
Site category flag type.
- struct [Site_Iterator](#)
Base class for implementation of classes of iterator on points.
- struct [Site_Proxy](#)
Base class for implementation classes of the notion of "site proxy".
- struct [Site_Proxy< void >](#)
Site_Proxy category flag type.
- struct [Site_Set](#)
Base class for implementation classes of site sets.
- struct [Site_Set< void >](#)
Site_Set category flag type.
- struct [slice_image](#)
2D image extracted from a slice of a 3D image.
- class [sub_image](#)

Image having its domain restricted by a site set.

- struct [sub_image_if](#)
Image having its domain restricted by a site set and a function.
- class [thru_image](#)
Morph image values through a function.
- class [thrubin_image](#)
Morphes values from two images through a binary function.
- struct [tr_image](#)
Transform an image by a given transformation.
- struct [unproject_image](#)
Un-projects an image.
- struct [Value](#)
Base class for implementation classes of values.
- struct [Value_Iterator](#)
Base class for implementation of classes of iterator on values.
- struct [Value_Set](#)
Base class for implementation classes of sets of values.
- struct [Vertex](#)
Vertex category flag type.
- class [vertex_image](#)
Image based on graph vertices.
- struct [violent_cast_image](#)
Violently cast image values to a given type.
- struct [w_window](#)
Generic [w_window](#) class.
- struct [Weighted_Window](#)
Base class for implementation classes that are [weighted_windows](#).
- class [window](#)
Generic window class.
- struct [Window](#)
Base class for implementation classes that are windows.

Typedefs

- typedef `mln::complex_image< 1, mln::discrete_plane_1complex_geometry, bool > bin_1complex_image2d`
Type alias for a binary image based on a 1-complex, where 0-faces are located at discrete (integer) 2-dimensional points.
- typedef `mln::complex_image< 2, mln::space_2complex_geometry, bool > bin_2complex_image3df`
Type alias for a binary image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.
- typedef `box< mln::point1d > box1d`
Type alias for a box defined on the 1D square grid with integer coordinates.
- typedef `box< mln::point2d > box2d`
Type alias for a box defined on the 2D square grid with integer coordinates.
- typedef `box< point2d_h > box2d_h`
FIXME.
- typedef `box< point3d > box3d`
Type alias for a box defined on the 3D square grid with integer coordinates.
- typedef `mln::geom::complex_geometry< 1, point2d > discrete_plane_1complex_geometry`
Type alias for the geometry of a 1-complex (e.g., a graph) located in a discrete 2-dimensional plane (with integer coordinates).
- typedef `mln::geom::complex_geometry< 2, point2d > discrete_plane_2complex_geometry`
Type alias for the geometry of a 2-complex located in a discrete 2-dimensional plane (with integer coordinates).
- typedef `dpoint< mln::grid::tick, def::coord > dpoint1d`
Type alias for a delta-point defined on the 1D square grid with integer coordinates.
- typedef `dpoint< mln::grid::square, mln::def::coord > dpoint2d`
Type alias for a delta-point defined on the 2D square grid with integer coordinates.
- typedef `dpoint< mln::grid::hexa, def::coord > dpoint2d_h`
Type alias for a delta-point defined on the 2D square grid with integer coordinates.
- typedef `dpoint< mln::grid::cube, def::coord > dpoint3d`
Type alias for a delta-point defined on the 3D square grid with integer coordinates.
- typedef `mln::complex_image< 2, mln::space_2complex_geometry, float > float_2complex_image3df`
Type alias for a floating-point image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.
- typedef `mln::complex_image< 1, mln::discrete_plane_1complex_geometry, mln::value::int_u8 > int_u8_1complex_image2d`

Type alias for an 8-bit gray-level image based on a 1-complex, where 0-faces are located at discrete (integer) 2-dimensional points.

- typedef `mln::complex_image< 2, mln::discrete_plane_2complex_geometry, mln::value::int_u8 > int_u8_2complex_image2d`

Type alias for an 8-bit gray-level image based on a 2-complex, where 0-faces are located at discrete (integer) 2-dimensional points.

- typedef `mln::complex_image< 2, mln::space_2complex_geometry, mln::value::int_u8 > int_u8_2complex_image3df`

Type alias for an 8-bit gray-level image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

- typedef `neighb< window1d > neighb1d`

Type alias for a neighborhood defined on the 1D square grid with integer coordinates.

- typedef `neighb< window2d > neighb2d`

Type alias for a neighborhood defined on the 2D square grid with integer coordinates.

- typedef `neighb< window3d > neighb3d`

Type alias for a neighborhood defined on the 3D square grid with integer coordinates.

- typedef `p_run< point2d > p_run2d`

Type alias for a run of 2d points.

- typedef `p_set_of< p_run2d > p_runs2d`

Type alias for a set of runs of 2d points.

- typedef `point< grid::tick, def::coordf > point1df`

Type alias for a point defined on the 1D ruler with floating-point coordinates.

- typedef `point< mln::grid::square, mln::def::coordf > point2df`

Type alias for a point defined on the 2D square grid with floating-point coordinates.

- typedef `point< grid::cube, def::coordf > point3df`

Type alias for a point defined on the 3D square grid with floating-point coordinates.

- typedef `mln::complex_image< 2, mln::space_2complex_geometry, mln::value::rgb8 > rgb8_2complex_image3df`

Type alias for a (3x8-bit) RGB image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

- typedef `mln::geom::complex_geometry< 2, point3df > space_2complex_geometry`

Type alias for the geometry of a 2-complex located in a 3-dimensional space (with floating-point coordinates).

- typedef `mln::complex_image< 2, mln::space_2complex_geometry, unsigned > unsigned_2complex_image3df`

Type alias for a gray-level image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

- typedef algebra::vec< 2u, double > [vec2d_d](#)
2D vector with double coordinates.
- typedef algebra::vec< 2u, float > [vec2d_f](#)
2D vector with float coordinates.
- typedef algebra::vec< 3u, double > [vec3d_d](#)
3D vector with double coordinates.
- typedef algebra::vec< 3u, float > [vec3d_f](#)
3D vector with float coordinates.
- typedef [w_window](#)< [dpoint1d](#), float > [w_window1d_float](#)
Type alias for a [w_window](#) with arbitrary shape, defined on the 1D grid (with integer coordinates) and whose weights are floating values.
- typedef [w_window](#)< [dpoint1d](#), int > [w_window1d_int](#)
Type alias for a [w_window](#) with arbitrary shape, defined on the 1D grid (with integer coordinates) and whose weights are integers.
- typedef [w_window](#)< [dpoint2d](#), float > [w_window2d_float](#)
Type alias for a [w_window](#) with arbitrary shape, defined on the 2D square grid (with integer coordinates) and whose weights are floating values.
- typedef [w_window](#)< [dpoint2d](#), int > [w_window2d_int](#)
Type alias for a [w_window](#) with arbitrary shape, defined on the 2D square grid (with integer coordinates) and whose weights are integers.
- typedef [w_window](#)< [dpoint3d](#), float > [w_window3d_float](#)
Type alias for a [w_window](#) with arbitrary shape, defined on the 3D grid (with integer coordinates) and whose weights are floating values.
- typedef [w_window](#)< [dpoint3d](#), int > [w_window3d_int](#)
Type alias for a [w_window](#) with arbitrary shape, defined on the 3D grid (with integer coordinates) and whose weights are integers.
- typedef [window](#)< [mln::dpoint1d](#) > [window1d](#)
Type alias for a window with arbitrary shape, defined on the 1D square grid with integer coordinates.
- typedef [window](#)< [mln::dpoint2d](#) > [window2d](#)
Type alias for a window with arbitrary shape, defined on the 2D square grid with integer coordinates.
- typedef [window](#)< [mln::dpoint3d](#) > [window3d](#)
Type alias for a window with arbitrary shape, defined on the 3D square grid with integer coordinates.
- typedef [point](#)< [grid::tick](#), [def::coord](#) > [point1d](#)
Type alias for a point defined on the 1D ruler with integer coordinates.
- typedef [point](#)< [mln::grid::square](#), [mln::def::coord](#) > [point2d](#)
Type alias for a point defined on the 2D square grid with integer coordinates.

- typedef [point](#)< [grid::hexa](#), [def::coord](#) > [point2d_h](#)
Type alias for a point defined on the 2D hexagonal grid with integer coordinates.
- typedef [point](#)< [grid::cube](#), [def::coord](#) > [point3d](#)
Type alias for a point defined on the 3D square grid with integer coordinates.

Functions

- template<typename I >
I::psite [a_point_of](#) (const [Image](#)< I > &ima)
Give a point of an image.
- template<typename I, typename F >
[p2p_image](#)< I, F > [apply_p2p](#) ([Image](#)< I > &ima, const [Function_v2v](#)< F > &f)
FIXME: Doc!
- template<typename I, typename F >
[p2p_image](#)< const I, F > [apply_p2p](#) (const [Image](#)< I > &ima, const [Function_v2v](#)< F > &f)
FIXME: Doc!
- const [neighb3d](#) & [c18](#) ()
18-connectivity neighborhood on the 3D grid.
- const [neighb1d](#) & [c2](#) ()
2-connectivity neighborhood on the 1D grid.
- const [neighb3d](#) & [c26](#) ()
26-connectivity neighborhood on the 3D grid.
- const [neighb3d](#) & [c2_3d_sli](#) ()
depth 2-connectivity neighborhood on the 3D grid.
- const [neighb2d](#) & [c2_col](#) ()
Vertical 2-connectivity neighborhood on the 2D grid.
- const [neighb2d](#) & [c2_row](#) ()
Horizontal 2-connectivity neighborhood on the 2D grid.
- const [neighb2d](#) & [c4](#) ()
4-connectivity neighborhood on the 2D grid.
- const [neighb3d](#) & [c4_3d](#) ()
4-connectivity neighborhood on the 3D grid.
- const [neighb3d](#) & [c6](#) ()
6-connectivity neighborhood on the 3D grid.
- const [neighb2d](#) & [c8](#) ()

8-connectivity neighborhood on the 2D grid.

- const [neighb3d](#) & [c8_3d](#) ()
8-connectivity neighborhood on the 3D grid.
- template<typename T2 , typename T1 >
[fun::x2x::composed](#)< T2, T1 > [compose](#) (T2 f, T1 g)
Do a composition of two transformations.
- template<typename I >
[mln::trait::concrete](#)< I >::ret [duplicate](#) (const [Image](#)< I > &model)
Duplicate the image model with the values of the image data.
- template<typename I , typename F >
[extension_fun](#)< const I, F > [extend](#) (const [Image](#)< I > &ima, const [Function_v2v](#)< F > &fun)
Routines for domain extension with a function.
- template<typename I , typename J >
[extension_ima](#)< const I, const J > [extend](#) (const [Image](#)< I > &ima, const [Image](#)< J > &ext)
Routines for domain extension with an image.
- template<typename I >
[extension_val](#)< const I > [extend](#) (const [Image](#)< I > &ima, const typename I::value &val)
Routines for domain extension with a value.
- bool [implies](#) (bool lexpr, bool rexpr)
Implication.
- template<typename I , typename J >
void [initialize](#) ([Image](#)< I > &target, const [Image](#)< J > &model)
- template<typename P >
[box](#)< P > [larger_than](#) (const [box](#)< P > a, const [box](#)< P > b)
Return the minimum box including box a and box b.
- template<typename I , typename V , typename E >
[image2d](#)< typename I::value > [make_debug_graph_image](#) (const I &input, const V &ima_v, const E &ima_e, const [value::rgb8](#) &bg)
Draw a graph.
- [mln_gen_complex_neighborhood](#) (complex_higher_dim_connected_n_face_neighborhood, complex_higher_dim_connected_n_face_window)
Neighborhood centered on an n-face of complex returning the n-faces sharing an (n+1)-face with the center n-face.
- [mln_gen_complex_neighborhood](#) (complex_m_face_neighborhood, complex_m_face_window)
Neighborhood centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face.
- [mln_gen_complex_neighborhood](#) (complex_lower_neighborhood, complex_lower_window)
Neighborhood centered on an n-face of complex returning its adjacent (n-1)-faces.

- [mln_gen_complex_neighborhood](#) (complex_higher_neighborhood, complex_higher_window)
Neighborhood centered on an n -face of complex returning its adjacent $(n+1)$ -faces.
- [mln_gen_complex_neighborhood](#) (complex_lower_higher_neighborhood, complex_lower_higher_window)
Neighborhood centered on an n -face of complex returning its adjacent $(n-1)$ -faces and $(n+1)$ -faces.
- [mln_gen_complex_neighborhood](#) (complex_lower_dim_connected_n_face_neighborhood, complex_lower_dim_connected_n_face_window)
Neighborhood centered on an n -face of complex returning the n -faces sharing an $(n-1)$ -face with the center n -face.
- [mln_gen_complex_window](#) (complex_lower_window, topo::adj_lower_face_fwd_iter, topo::adj_lower_face_bkd_iter)
Window centered on an n -face of complex returning its adjacent $(n-1)$ -faces.
- [mln_gen_complex_window](#) (complex_higher_window, topo::adj_higher_face_fwd_iter, topo::adj_higher_face_bkd_iter)
Window centered on an n -face of complex returning its adjacent $(n+1)$ -faces.
- [mln_gen_complex_window](#) (complex_lower_higher_window, topo::adj_lower_higher_face_fwd_iter, topo::adj_lower_higher_face_bkd_iter)
Window centered on an n -face of complex returning its adjacent $(n-1)$ -faces and $(n+1)$ -faces.
- [mln_gen_complex_window](#) (complex_lower_dim_connected_n_face_window, topo::adj_lower_dim_connected_n_face_fwd_iter, topo::adj_lower_dim_connected_n_face_bkd_iter)
Window centered on an n -face of complex returning the n -faces sharing an $(n-1)$ -face with the center n -face.
- [mln_gen_complex_window](#) (complex_higher_dim_connected_n_face_window, topo::adj_higher_dim_connected_n_face_fwd_iter, topo::adj_higher_dim_connected_n_face_bkd_iter)
Window centered on an n -face of complex returning the n -faces sharing an $(n+1)$ -face with the center n -face.
- [mln_gen_complex_window](#) (complex_m_face_window, topo::adj_m_face_fwd_iter, topo::adj_m_face_bkd_iter)
Window centered on an n -face of complex returning the m -faces transitively adjacent to this center n -face.
- [mln_gen_complex_window_p](#) (complex_lower_window_p, topo::adj_lower_face_fwd_iter, topo::adj_lower_face_bkd_iter)
Window centered on an n -face of complex returning its adjacent $(n-1)$ -faces as well as the center n -face.
- [mln_gen_complex_window_p](#) (complex_higher_window_p, topo::adj_higher_face_fwd_iter, topo::adj_higher_face_bkd_iter)
Window centered on an n -face of complex returning its adjacent $(n+1)$ -faces as well as the center n -face.
- [mln_gen_complex_window_p](#) (complex_lower_higher_window_p, topo::adj_lower_higher_face_fwd_iter, topo::adj_lower_higher_face_bkd_iter)
Window centered on an n -face of complex returning its adjacent $(n-1)$ -faces and $(n+1)$ -faces as well as the center n -face.
- [mln_gen_complex_window_p](#) (complex_lower_dim_connected_n_face_window_p, topo::adj_lower_dim_connected_n_face_fwd_iter, topo::adj_lower_dim_connected_n_face_bkd_iter)

Window centered on an n -face of complex returning the n -faces sharing an $(n-1)$ -face with the center n -face, as well as this center n -face.

- `mln_gen_complex_window_p` (`complex_higher_dim_connected_n_face_window_p`, `topo::adj_higher_dim_connected_n_face_fwd_iter`, `topo::adj_higher_dim_connected_n_face_bkd_iter`)

Window centered on an n -face of complex returning the n -faces sharing an $(n+1)$ -face with the center n -face, as well as this center n -face.

- `mln_gen_complex_window_p` (`complex_m_face_window_p`, `topo::adj_m_face_fwd_iter`, `topo::adj_m_face_bkd_iter`)

Window centered on an n -face of complex returning the m -faces transitively adjacent to this center n -face, as well as this center n -face.

- `template<typename W1 , typename W2 >`
`mln_reguar` (W1) `operator-(const Window< W1 > &win1`
`win2)`
Set difference between a couple of windows `win1` and `win2`.
- `template<typename O1 , typename O2 >`
`mln_trait_op_geq` (O1, O2) `operator>`
General definition of the "greater than or equal to" operator.
- `template<typename O1 , typename O2 >`
`mln_trait_op_greater` (O1, O2) `operator>(const Object< O1 > &lhs`
`rhs)`
General definition of the "greater than" operator.
- `template<typename O1 , typename O2 >`
`mln_trait_op_leq` (O1, O2) `operator<`
Default definition of the "less than or equal to" operator.
- `template<typename O1 , typename O2 >`
`mln_trait_op_neq` (O1, O2) `operator!`
General definition of the "not equal to" operator.
- `template<typename P , typename S >`
`P operator*` (const `Gpoint< P > &p`, const `value::scalar_< S > &s`)
Multiply a point `p` by a scalar `s`.
- `template<typename S >`
`S & operator++` (`value::Scalar< S > &rhs`)
Pre-incrementation for any scalar type.
- `template<typename P , typename D >`
`P operator-` (const `Gpoint< P > &p`, const `Gdpoint< D > &dp`)
Subtract a delta-point `dp` to a grid point `p`.
- `template<typename N1 , typename N2 >`
`neighb< typename N1::window::regular > operator-` (const `Neighborhood< N1 > &nbh1`, const `Neighborhood< N2 > &nbh2`)
Set difference between a couple of neighborhoods `nbh1` and `nbh2`.
- `template<typename S >`
`S & operator--` (`value::Scalar< S > &rhs`)

Pre-decrementation for any scalar type.

- `template<typename L , typename R >`
`bool operator< (const Image< L > &lhs, const Image< R > &rhs)`
Point-wise test if the pixel values of lhs are point-wise less than the pixel values of rhs.

- `template<typename I , typename G , typename W >`
`std::ostream & operator<< (std::ostream &ostr, const complex_window_fwd_piter< I, G, W > &p)`
Print an `mln::complex_window_fwd_piter`.

- `template<typename I , typename G , typename W >`
`std::ostream & operator<< (std::ostream &ostr, const complex_window_bkd_piter< I, G, W > &p)`
Print an `mln::complex_window_bkd_piter`.

- `template<typename I , typename G , typename N >`
`std::ostream & operator<< (std::ostream &ostr, const complex_neighborhood_fwd_piter< I, G, N > &p)`
Print an `mln::complex_neighborhood_fwd_piter`.

- `template<typename I , typename G , typename N >`
`std::ostream & operator<< (std::ostream &ostr, const complex_neighborhood_bkd_piter< I, G, N > &p)`
Print an `mln::complex_neighborhood_bkd_piter`.

- `template<unsigned N, unsigned D, typename P >`
`bool operator<= (const p_faces< N, D, P > &lhs, const p_faces< N, D, P > &rhs)`
Inclusion of a `mln::p_faces` in another one.

- `template<typename G , typename F >`
`bool operator<= (const p_edges< G, F > &lhs, const p_edges< G, F > &rhs)`
Inclusion of a `mln::p_edges` in another one.

- `template<typename L , typename R >`
`bool operator<= (const Image< L > &lhs, const Image< R > &rhs)`
Point-wise test if the pixel values of lhs are point-wise less than or equal to the pixel values of rhs.

- `template<unsigned D, typename G >`
`bool operator<= (const p_complex< D, G > &lhs, const p_complex< D, G > &rhs)`
Inclusion of a `mln::p_complex` in another one.

- `template<typename G , typename F >`
`bool operator<= (const p_vertices< G, F > &lhs, const p_vertices< G, F > &rhs)`
Inclusion of a `mln::p_vertices` in another one.

- `template<typename G , typename F >`
`bool operator== (const p_edges< G, F > &lhs, const p_edges< G, F > &rhs)`
Comparison between two `mln::p_edges`'s.

- `template<typename L , typename R >`
`bool operator== (const Image< L > &lhs, const Image< R > &rhs)`
Point-wise test if the pixel values of lhs are equal to the pixel values of rhs.
- `template<unsigned D, typename G >`
`bool operator== (const p_complex< D, G > &lhs, const p_complex< D, G > &rhs)`
Comparison between two mln::p_complex's.
- `template<unsigned N, unsigned D, typename P >`
`bool operator== (const p_faces< N, D, P > &lhs, const p_faces< N, D, P > &rhs)`
Comparison between two mln::p_faces's.
- `template<typename G , typename F >`
`bool operator== (const p_vertices< G, F > &lhs, const p_vertices< G, F > &rhs)`
Comparison between two mln::p_vertices's.
- `template<typename I , typename F >`
`image_if< I, F > operator| (Image< I > &ima, const Function_v2b< F > &f)`
ima | f creates an image_if with the image ima and the function f.
- `template<typename V , typename G , typename P >`
`edge_image< P, V, G > operator| (const fun::i2v::array< V > &edge_values, const p_edges< G, fun::i2v::array< P > > &pe)`
Construct a edge image from a fun::i2v::array and a p_edges.
- `template<typename S , typename F >`
`p_if< S, F > operator| (const Site_Set< S > &s, const Function_v2b< F > &f)`
Restrict a site set s to points that verify f.
- `template<typename F , typename S >`
`pw::image< F, S > operator| (const Function_v2v< F > &f, const Site_Set< S > &ps)`
Construct an image from a function and a site set.
- `template<typename I , typename F >`
`image_if< const I, F > operator| (const Image< I > &ima, const Function_v2b< F > &f)`
ima | f creates an image_if with the image ima and the function f.
- `template<typename V , typename G , typename P >`
`vertex_image< P, V, G > operator| (const fun::i2v::array< V > &vertex_values, const p_vertices< G, fun::i2v::array< P > > &pv)`
Construct a vertex image from a fun::i2v::array and a p_vertices.
- `template<typename I >`
`const internal::primary_type< I >::ret & primary (const Image< I > &input)`
FIXME: Doc!
- `template<typename S , typename F >`
`p_transformed< S, F > ptransform (const Site_Set< S > &s, const Function_v2v< F > &f)`
Transform a site set s through the function f.
- `const window2d & win_c4p ()`

4-connectivity window on the 2D grid, including the center.

- const `window3d` & `win_c4p_3d` ()
4-connectivity window on the 3D grid, including the center.
- const `window2d` & `win_c8p` ()
8-connectivity window on the 2D grid, including the center.
- const `window3d` & `win_c8p_3d` ()
8-connectivity window on the 3D grid, including the center.
- template<unsigned N, unsigned D, typename P >
bool `operator==` (const `faces_psite`< N, D, P > &lhs, const `faces_psite`< N, D, P > &rhs)
Comparison of two instances of `mln::faces_psite`.
- template<unsigned N, unsigned D, typename P >
bool `operator!=` (const `faces_psite`< N, D, P > &lhs, const `faces_psite`< N, D, P > &rhs)
Is lhs equal to rhs?
- template<unsigned N, unsigned D, typename P >
bool `operator<` (const `faces_psite`< N, D, P > &lhs, const `faces_psite`< N, D, P > &rhs)
Is lhs "less" than rhs?
- template<typename T >
`mln_exact` (T)*`exact`(T *ptr)
Exact cast routine for `mln` objects.
- template<unsigned D, typename G >
bool `operator==` (const `complex_psite`< D, G > &lhs, const `complex_psite`< D, G > &rhs)
Comparison of two instances of `mln::complex_psite`.
- template<unsigned D, typename G >
bool `operator!=` (const `complex_psite`< D, G > &lhs, const `complex_psite`< D, G > &rhs)
Is lhs not equal to rhs?
- template<unsigned D, typename G >
bool `operator<` (const `complex_psite`< D, G > &lhs, const `complex_psite`< D, G > &rhs)
Is lhs "less" than rhs?

Variables

- const `dpoint1d before` = `dpoint1d`(-1)
Definition of a shortcut for delta point in 1d.
- const `dpoint2d up` = `dpoint2d`(-1, 0)
Definition of a shortcut for delta point in 2d.

- const [dpoint3d sagittal_dec](#) = [dpoint3d](#)(0, 0, -1)
Definition of a shortcut for delta point in 3d.

9.1.1 Detailed Description

[mln/convert/to_image.hh](#) This implementation is not an usual heap, it allows to set an error rate so that some nodes may be "corrupted".

Generic class for hierarchical queues.

Merge with [mln/topo/skeleton/is_simple_point.hh](#) The fastest version give an approximation of the result.

The generic dual input tree algorithm for high quantized image.

The dual input tree algorithm specialized for low quantized image.

[mln/linear/convolve_directional.hh](#)

Read AVS header from a file.

Define a function which aborts a process in io module.

Forward declaration.

[mln/core/def/all.hh](#)

The namespace mln corresponds to the Milena (mini-Olena) project.

This accumulator uses an [mln::util::pix](#) (pixel) to update the reference level, area and volume information of the component.

The class [mln/accu/volume](#) is not a general-purpose accumulator; it is used to implement volume-based connected filters.

See also

[mln::morpho::closing::volume](#)
[mln::morpho::opening::volume](#)

The functor should provide the following methods:

- `template <typename G>=""> void init(const Graph<G>& g)` Will be called at the beginning.
- `bool to_be_treated(unsigned id)` Return whether this vertex has already been marked or if it may be a a component representative.
- `void new_component_from_vertex(unsigned id)` will be called for the first vertex encountered for each component.
- `void process_vertex(unsigned id)` Will be called for each vertex queued.
- `bool to_be_queued(unsigned id)` Return whether this vertex has already been marked or if it can be added to the current component.
- `void added_to_queue(unsigned id)` Will be called for every vertex encountered in each component, except the first one.
- `void next_component()` Will be called after all vertices from a component have been treated.

- void final() Will be called at the end;

Conversions to [mln::Image](#).

FIXME: Re-write this description.

The contents of mln mimics the contents of the olena project but in a simplified way. Some classes have the same name in both projects and roughly have the same behavior.

Warning

The Milena project is independent from the Olena project; the user has to choose between both the project she wants to work with.

File that includes all core definitions.

The set of operators defined in this file is:

```

l += r : l = l + r, -> l&
l -= r : l = l - r, -> l&
l *= r : l = l * r, -> l&
l /= r : l = l / r, -> l&
l %= r : l = l % r, -> l&

+ r    : -> r
- r    : -> (0 - r)

l ++   : t = l, ++l, -> t
l --   : t = l, --l, -> t

++ r   : r += 1, -> r&
-- r   : r -= 1, -> r&

l != r : -> ! (l == r)

l > r  : -> (r < l)
l >= r : -> (r <= l)
l <= r : -> ! (r < l)    warning: re-define when partial ordering

```

As a consequence, the set of operators to be defined along with a client class is:

```

l + r
l - r
l * r
l / r

l == r

l < r
l <= r in case of partial ordering

```

Convolution by a line-shaped (directional) kernel.

This implementation is based on P. Salembier algorithm using hierarchical queues. This implies a low-quantized input image so that the number of queues is limited.

TODO: Think about how to extend f domain in a more generic way. The actual implementation doubles the size of the first dimension. It implies a boxed domain.

TODO: Use the less functor. The actual implementation is for max-tree.

TODO: During the canonization pass, we build the tree site set from the sorted site set of f, so that we compute twice f histogram (can be avoided).

This implementation is based on tarjan's union method, so that image quantization does not impact on the computation time.

TODO: Think about how to extend f domain in a more generic way. The actual implementation doubles the size of the first dimension. It implies a boxed domain.

TODO: Use the less functor. The actual implementation is for max-tree.

Do we want it to be exact?

Hierarchical queues are often used with connected operators (P. Salembier's max tree algorithm relies on these queues). To be efficient, the hierarchy is a static array and each are preallocated using an histogram.

FIXME: consider hqueues as a site set ?

A "corrupted node" means that its correct order is not totally preserved for performance reasons. Of course, it will have an impact on the returned values. As a result, be ware of not using this data structure if the element order is relevant for to you.

A corruption threshold can be passed to the constructor. This threshold means that if nodes have a rank higher than this threshold they can be "corrupted" and therefore their rank can be reduced. Tuning this threshold may have an impact on the structure entropy thus on the returned values order. It may also have an impact on the performance.

More implementation details are available in: "The soft heap: an approximate priority queue with optimal error rate", Bernard Chazelle, JACM, 2000.

URL: <http://www.cs.princeton.edu/~chazelle/pubs/sheap.pdf>

9.1.2 Typedef Documentation

9.1.2.1 `typedef mln::complex_image<1, mln::discrete_plane_1complex_geometry, bool> mln::bin_1complex_image2d`

Type alias for a binary image based on a 1-complex, where 0-faces are located at discrete (integer) 2-dimensional points.

Definition at line 53 of file mln/core/alias/complex_image.hh.

9.1.2.2 `typedef mln::complex_image<2, mln::space_2complex_geometry, bool> mln::bin_2complex_image3df`

Type alias for a binary image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

Definition at line 85 of file mln/core/alias/complex_image.hh.

9.1.2.3 `typedef box<mln::point1d> mln::box1d`

Type alias for a box defined on the 1D square grid with integer coordinates.

See also

`mln::win::rectangle1d`.

Definition at line 47 of file core/alias/box1d.hh.

9.1.2.4 `typedef box<mln::point2d> mln::box2d`

Type alias for a box defined on the 2D square grid with integer coordinates.

See also

[mln::win::rectangle2d](#).

Definition at line 45 of file core/alias/box2d.hh.

9.1.2.5 `typedef box<point2d_h> mln::box2d_h`

FIXME.

Definition at line 47 of file core/alias/box2d_h.hh.

9.1.2.6 `typedef box<point3d> mln::box3d`

Type alias for a box defined on the 3D square grid with integer coordinates.

See also

[mln::win::rectangle3d](#).

Definition at line 45 of file core/alias/box3d.hh.

9.1.2.7 `typedef mln::geom::complex_geometry<1, point2d> mln::discrete_plane_1complex_geometry`

Type alias for the geometry of a 1-complex (e.g., a graph) located in a discrete 2-dimensional plane (with integer coordinates).

Definition at line 45 of file core/alias/complex_geometry.hh.

9.1.2.8 `typedef mln::geom::complex_geometry<2, point2d> mln::discrete_plane_2complex_geometry`

Type alias for the geometry of a 2-complex located in a discrete 2-dimensional plane (with integer coordinates).

Definition at line 50 of file core/alias/complex_geometry.hh.

9.1.2.9 `typedef dpoint<mln::grid::tick, def::coord> mln::dpoint1d`

Type alias for a delta-point defined on the 1D square grid with integer coordinates.

Definition at line 44 of file dpoint1d.hh.

9.1.2.10 `typedef dpoint<mln::grid::square, mln::def::coord> mln::dpoint2d`

Type alias for a delta-point defined on the 2D square grid with integer coordinates.

Definition at line 44 of file dpoint2d.hh.

9.1.2.11 typedef dpoint<mln::grid::hexa, def::coord> mln::dpoint2d_h

Type alias for a delta-point defined on the 2D square grid with integer coordinates.

Definition at line 45 of file core/alias/dpoint2d_h.hh.

9.1.2.12 typedef dpoint<mln::grid::cube, def::coord> mln::dpoint3d

Type alias for a delta-point defined on the 3D square grid with integer coordinates.

Definition at line 43 of file dpoint3d.hh.

**9.1.2.13 typedef mln::complex_image<2, mln::space_2complex_geometry, float>
mln::float_2complex_image3df**

Type alias for a floating-point image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

Definition at line 106 of file mln/core/alias/complex_image.hh.

**9.1.2.14 typedef mln::complex_image<1, mln::discrete_plane_1complex_geometry,
mln::value::int_u8> mln::int_u8_1complex_image2d**

Type alias for an 8-bit gray-level image based on a 1-complex, where 0-faces are located at discrete (integer) 2-dimensional points.

Definition at line 61 of file mln/core/alias/complex_image.hh.

**9.1.2.15 typedef mln::complex_image<2, mln::discrete_plane_2complex_geometry,
mln::value::int_u8> mln::int_u8_2complex_image2d**

Type alias for an 8-bit gray-level image based on a 2-complex, where 0-faces are located at discrete (integer) 2-dimensional points.

Definition at line 74 of file mln/core/alias/complex_image.hh.

**9.1.2.16 typedef mln::complex_image<2, mln::space_2complex_geometry, mln::value::int_u8>
mln::int_u8_2complex_image3df**

Type alias for an 8-bit gray-level image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

Definition at line 92 of file mln/core/alias/complex_image.hh.

9.1.2.17 typedef p_run<point2d> mln::p_run2d

Type alias for a run of 2d points.

Definition at line 42 of file p_run2d.hh.

9.1.2.18 typedef p_set_of<p_run2d> mln::p_runs2d

Type alias for a set of runs of 2d points.

Definition at line 42 of file p_runs2d.hh.

9.1.2.19 typedef point< grid::tick, def::coord > mln::point1d

Type alias for a point defined on the 1D ruler with integer coordinates.

Definition at line 45 of file point1d.hh.

9.1.2.20 typedef point<grid::tick, def::coordf> mln::point1df

Type alias for a point defined on the 1D ruler with floating-point coordinates.

Definition at line 49 of file point1d.hh.

9.1.2.21 typedef point< grid::square, def::coord > mln::point2d

Type alias for a point defined on the 2D square grid with integer coordinates.

Definition at line 45 of file point2d.hh.

9.1.2.22 typedef point< grid::hexa, def::coord > mln::point2d_h

Type alias for a point defined on the 2D hexagonal grid with integer coordinates.

Definition at line 43 of file core/alias/point2d_h.hh.

9.1.2.23 typedef point<mln::grid::square, mln::def::coordf> mln::point2df

Type alias for a point defined on the 2D square grid with floating-point coordinates.

Definition at line 49 of file point2d.hh.

9.1.2.24 typedef point< grid::cube, def::coord > mln::point3d

Type alias for a point defined on the 3D square grid with integer coordinates.

Definition at line 44 of file point3d.hh.

9.1.2.25 typedef point<grid::cube, def::coordf> mln::point3df

Type alias for a point defined on the 3D square grid with floating-point coordinates.

Definition at line 48 of file point3d.hh.

9.1.2.26 `typedef mln::complex_image<2, mln::space_2complex_geometry, mln::value::rgb8>
mln::rgb8_2complex_image3df`

Type alias for a (3x8-bit) RGB image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

Definition at line 113 of file mln/core/alias/complex_image.hh.

9.1.2.27 `typedef mln::geom::complex_geometry<2, point3df> mln::space_2complex_geometry`

Type alias for the geometry of a 2-complex located in a 3-dimensional space (with floating-point coordinates).

Definition at line 54 of file core/alias/complex_geometry.hh.

9.1.2.28 `typedef mln::complex_image<2, mln::space_2complex_geometry, unsigned>
mln::unsigned_2complex_image3df`

Type alias for a gray-level image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.

Definition at line 99 of file mln/core/alias/complex_image.hh.

9.1.2.29 `typedef algebra::vec<2u,double> mln::vec2d_d`

2D vector with double coordinates.

Definition at line 43 of file vec2d.hh.

9.1.2.30 `typedef algebra::vec<2u,float> mln::vec2d_f`

2D vector with float coordinates.

Definition at line 40 of file vec2d.hh.

9.1.2.31 `typedef algebra::vec<3u,double> mln::vec3d_d`

3D vector with double coordinates.

Definition at line 43 of file vec3d.hh.

9.1.2.32 `typedef algebra::vec<3u,float> mln::vec3d_f`

3D vector with float coordinates.

Definition at line 40 of file vec3d.hh.

9.1.2.33 `typedef w_window<dpoint1d, float> mln::w_window1d_float`

Type alias for a [w_window](#) with arbitrary shape, defined on the 1D grid (with integer coordinates) and whose weights are floating values.

Definition at line 45 of file w_window1d_float.hh.

9.1.2.34 typedef w_window<dpoint1d, int> mln::w_window1d_int

Type alias for a [w_window](#) with arbitrary shape, defined on the 1D grid (with integer coordinates) and whose weights are integers.

Definition at line 45 of file core/alias/w_window1d_int.hh.

9.1.2.35 typedef w_window<dpoint2d, float> mln::w_window2d_float

Type alias for a [w_window](#) with arbitrary shape, defined on the 2D square grid (with integer coordinates) and whose weights are floating values.

Definition at line 45 of file w_window2d_float.hh.

9.1.2.36 typedef w_window<dpoint2d, int> mln::w_window2d_int

Type alias for a [w_window](#) with arbitrary shape, defined on the 2D square grid (with integer coordinates) and whose weights are integers.

Definition at line 45 of file core/alias/w_window2d_int.hh.

9.1.2.37 typedef w_window<dpoint3d, float> mln::w_window3d_float

Type alias for a [w_window](#) with arbitrary shape, defined on the 3D grid (with integer coordinates) and whose weights are floating values.

Definition at line 45 of file w_window3d_float.hh.

9.1.2.38 typedef w_window<dpoint3d, int> mln::w_window3d_int

Type alias for a [w_window](#) with arbitrary shape, defined on the 3D grid (with integer coordinates) and whose weights are integers.

Definition at line 45 of file core/alias/w_window3d_int.hh.

9.1.3 Function Documentation**9.1.3.1 template<typename I> I::psite mln::a_point_of(const Image< I > & ima) [inline]**

Give a point of an image.

Definition at line 51 of file a_point_of.hh.

9.1.3.2 template<typename I, typename F> p2p_image< I, F > mln::apply_p2p(Image< I > & ima, const Function_v2v< F > & f) [inline]

FIXME: Doc!

Definition at line 242 of file p2p_image.hh.

Referenced by mln::debug::mosaic(), and mln::debug::slices_2d().

9.1.3.3 `template<typename I, typename F> p2p_image< const I, F> mln::apply_p2p (const Image< I> & ima, const Function_v2v< F> & f) [inline]`

FIXME: Doc!

Definition at line 256 of file p2p_image.hh.

9.1.3.4 `template<typename T2, typename T1> fun::x2x::composed< T2, T1> mln::compose (T2 f, T1 g) [inline]`

Do a composition of two transformations.

Parameters

[in] *f* The second transformation.

[in] *g* The first transformation.

Returns

The composed transformation fog.

Definition at line 306 of file composed.hh.

References `compose()`.

Referenced by `compose()`, and `mln::geom::rotate()`.

9.1.3.5 `template<typename I> mln::trait::concrete< I>::ret mln::duplicate (const Image< I> & model) [inline]`

Duplicate the image `model` with the values of the image `data`.

Parameters

[in] *model* The image to be duplicated.

Returns

The duplicate.

Precondition

`model.is_valid`

Definition at line 56 of file core/routine/duplicate.hh.

References `mln::data::fill()`, and `initialize()`.

Referenced by `mln::registration::icp()`, `mln::plain< I>::operator I()`, `mln::geom::rotate()`, `mln::geom::impl::seeds2tiling()`, and `mln::labeling::superpose()`.

9.1.3.6 `template<typename I, typename F> extension_fun< const I, F> mln::extend (const Image< I> & ima, const Function_v2v< F> & fun) [inline]`

Routines for domain extension with a function.

Definition at line 140 of file extend.hh.

Referenced by `mln::geom::translate()`.

9.1.3.7 `template<typename I, typename J > extension_ima< const I, const J > mln::extend (const Image< I > & ima, const Image< J > & ext)`

Routines for domain extension with an image.

Definition at line 162 of file extend.hh.

9.1.3.8 `template<typename I > extension_val< const I > mln::extend (const Image< I > & ima, const typename I::value & val) [inline]`

Routines for domain extension with a value.

Definition at line 175 of file extend.hh.

9.1.3.9 `bool mln::implies (bool lexpr, bool rexpr) [inline]`

Implication.

Definition at line 68 of file contract.hh.

Referenced by `mln::p_line2d::is_valid()`.

9.1.3.10 `template<typename I, typename J > void mln::initialize (Image< I > & target, const Image< J > & model) [inline]`

Initialize the image `target` with data extracted from image `model`.

Parameters

[in, out] *target* The image to be initialized.

[in] *model* The image to provide data for the initialization.

Precondition

(not `target.is_valid()` and `model.is_valid()`)

Definition at line 56 of file initialize.hh.

Referenced by `duplicate()`, `mln::histo::equalize()`, `mln::morpho::tree::filter::filter()`, `mln::linear::gaussian()`, `mln::linear::gaussian_1st_derivative()`, `mln::linear::gaussian_2nd_derivative()`, `mln::graph::labeling()`, `mln::io::magick::load()`, `mln::io::dicom::load()`, `make_debug_graph_image()`, `mln::morpho::tree::filter::max()`, `mln::morpho::meyer_wst()`, `mln::morpho::tree::filter::min()`, `mln::arith::min()`, `mln::arith::minus()`, `mln::arith::plus()`, `mln::arith::revert()`, `mln::geom::rotate()`, `mln::data::impl::stretch()`, `mln::morpho::watershed::topological()`, and `mln::data::impl::generic::transform()`.

9.1.3.11 `template<typename P > box< P > mln::larger_than (const box< P > a, const box< P > b) [inline]`

Return the minimum box including box `a` and box `b`.

Definition at line 362 of file core/site_set/box.hh.

References `mln::box< P >::pmax()`, and `mln::box< P >::pmin()`.

9.1.3.12 `template<typename I , typename V , typename E > image2d<typename I ::value>
mln::make_debug_graph_image (const I & input, const V & ima_v, const E & ima_e,
const value::rgb8 & bg) [inline]`

Draw a graph.

Definition at line 123 of file demos/graph/region_adjacency_graph.cc.

References `mln::box< P >::crop_wrt()`, `mln::image2d< T >::domain()`, `mln::debug::draw_graph()`, `mln::data::fill()`, `mln::literal::green`, `initialize()`, and `mln::convert::to()`.

9.1.3.13 `template<typename T > mln::mln_exact (T) [inline]`

Exact cast routine for mln objects.

This set of routines can be used to downcast an object towards its exact type. The only argument, respectively `ptr` or `ref`, should be an `mln::Object`.

The parameter `E` is the exact type of the object.

Returns

The return follows the nature of the argument (either a pointer or a reference, const or not).

Definition at line 92 of file routine/exact.hh.

Referenced by `mln::geom::rotate()`, `mln::Accumulator< E >::take_as_init()`, `mln::Accumulator< E >::take_n_times()`, `mln::convert::to()`, and `mln::geom::translate()`.

9.1.3.14 `mln::mln_gen_complex_neighborhood (complex_higher_neighborhood ,
complex_higher_window)`

[Neighborhood](#) centered on an n-face of complex returning its adjacent (n+1)-faces.

9.1.3.15 `mln::mln_gen_complex_neighborhood (complex_lower_higher_neighborhood ,
complex_lower_higher_window)`

[Neighborhood](#) centered on an n-face of complex returning its adjacent (n-1)-faces and (n+1)-faces.

9.1.3.16 `mln::mln_gen_complex_neighborhood (complex_lower_dim_connected_
n_face_neighborhood , complex_lower_dim_connected_n_face_window
)`

[Neighborhood](#) centered on an n-face of complex returning the n-faces sharing an (n-1)-face with the center n-face.

9.1.3.17 `mln::mln_gen_complex_neighborhood (complex_higher_dim_connected_
n_face_neighborhood , complex_higher_dim_connected_n_face_window
)`

[Neighborhood](#) centered on an n-face of complex returning the n-faces sharing an (n+1)-face with the center n-face.

9.1.3.18 `mln::mln_gen_complex_neighborhood (complex_m_face_neighborhood , complex_m_face_window)`

[Neighborhood](#) centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face.

9.1.3.19 `mln::mln_gen_complex_neighborhood (complex_lower_neighborhood , complex_lower_window)`

[Neighborhood](#) centered on an n-face of complex returning its adjacent (n-1)-faces.

9.1.3.20 `mln::mln_gen_complex_window (complex_lower_window , topo::adj_lower_face_fwd_iter , topo::adj_lower_face_bkd_iter)`

[Window](#) centered on an n-face of complex returning its adjacent (n-1)-faces.

9.1.3.21 `mln::mln_gen_complex_window (complex_higher_window , topo::adj_higher_face_fwd_iter , topo::adj_higher_face_bkd_iter)`

[Window](#) centered on an n-face of complex returning its adjacent (n+1)-faces.

9.1.3.22 `mln::mln_gen_complex_window (complex_lower_higher_window , topo::adj_lower_higher_face_fwd_iter , topo::adj_lower_higher_face_bkd_iter)`

[Window](#) centered on an n-face of complex returning its adjacent (n-1)-faces and (n+1)-faces.

9.1.3.23 `mln::mln_gen_complex_window (complex_lower_dim_connected_n_face_window , topo::adj_lower_dim_connected_n_face_fwd_iter , topo::adj_lower_dim_connected_n_face_bkd_iter)`

[Window](#) centered on an n-face of complex returning the n-faces sharing an (n-1)-face with the center n-face.

9.1.3.24 `mln::mln_gen_complex_window (complex_higher_dim_connected_n_face_window , topo::adj_higher_dim_connected_n_face_fwd_iter , topo::adj_higher_dim_connected_n_face_bkd_iter)`

[Window](#) centered on an n-face of complex returning the n-faces sharing an (n+1)-face with the center n-face.

9.1.3.25 `mln::mln_gen_complex_window (complex_m_face_window , topo::adj_m_face_fwd_iter , topo::adj_m_face_bkd_iter)`

[Window](#) centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face.

9.1.3.26 `mln::mln_gen_complex_window_p (complex_lower_window_p , topo::adj_lower_face_fwd_iter , topo::adj_lower_face_bkd_iter)`

[Window](#) centered on an n-face of complex returning its adjacent (n-1)-faces as well as the center n-face.

9.1.3.27 `mln::mln_gen_complex_window_p (complex_higher_window_p ,
topo::adj_higher_face_fwd_iter , topo::adj_higher_face_bkd_iter)`

Window centered on an n-face of complex returning its adjacent (n+1)-faces as well as the center n-face.

9.1.3.28 `mln::mln_gen_complex_window_p (complex_lower_higher_window_p ,
topo::adj_lower_higher_face_fwd_iter , topo::adj_lower_higher_face_bkd_iter)`

Window centered on an n-face of complex returning its adjacent (n-1)-faces and (n+1)-faces as well as the center n-face.

9.1.3.29 `mln::mln_gen_complex_window_p (complex_higher_dim_connected_
n_face_window_p , topo::adj_higher_dim_connected_n_face_fwd_iter ,
topo::adj_higher_dim_connected_n_face_bkd_iter)`

Window centered on an n-face of complex returning the n-faces sharing an (n+1)-face with the center n-face, as well as this center n-face.

9.1.3.30 `mln::mln_gen_complex_window_p (complex_lower_dim_connected_
n_face_window_p , topo::adj_lower_dim_connected_n_face_fwd_iter ,
topo::adj_lower_dim_connected_n_face_bkd_iter)`

Window centered on an n-face of complex returning the n-faces sharing an (n-1)-face with the center n-face, as well as this center n-face.

9.1.3.31 `mln::mln_gen_complex_window_p (complex_m_face_window_p ,
topo::adj_m_face_fwd_iter , topo::adj_m_face_bkd_iter)`

Window centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face, as well as this center n-face.

9.1.3.32 `template<typename W1 , typename W2 > mln::mln_regular (W1) const [inline]`

Set difference between a couple of windows `win1` and `win2`.

Inter a window `win` with a delta-point `dp`.

It just calls `mln::win::diff`.

9.1.3.33 `template<typename O1 , typename O2 > mln::mln_trait_op_geq (O1 , O2)`

General definition of the "greater than or equal to" operator.

The "greater than or equal to" operator is here defined for every Milena objects. It relies on the definition of the "less than or equal to" operator. It returns "`rhs <= lhs`".

Warning

There shall not be any other definition of this operator in Milena when applying on a couple of `mln::Object`.

9.1.3.34 `template<typename O1 , typename O2 > mln::mln_trait_op_greater (O1 , O2) const`

General definition of the "greater than" operator.

The "greater than" operator is here defined for every milena objects. It relies on the definition of the "less than" operator. It returns "rhs < lhs".

Warning

There shall not be any other definition of this operator in Milena when applying on a couple of [mln::Object](#).

9.1.3.35 `template<typename O1 , typename O2 > mln::mln_trait_op_leq (O1 , O2)`

Default definition of the "less than or equal to" operator.

A default version of the "less than or equal to" operator is defined for every Milena objects. It relies on the definition of the "less than" operator. It returns "not (rhs < lhs)".

Warning

In the case of partial ordering between objects, this operator has to be re-defined.

9.1.3.36 `template<typename O1 , typename O2 > mln::mln_trait_op_neq (O1 , O2)`
`[inline]`**Initial value:**

```
(const Object<O1>& lhs, const Object<O2>& rhs)
{
    return ! (exact(lhs) == exact(rhs));
}

template <typename O1, typename O2>
inline
mln_trait_op_greater(O1, O2)
operator>(const Object<O1>& lhs, const Object<O2>& rhs)
{
    return exact(rhs) < exact(lhs);
}

template <typename O1
```

General definition of the "not equal to" operator.

The "not equal to" operator is here defined for every milena objects. It relies on the definition of the "equal to" operator. It returns "not (lhs == rhs)".

Warning

There shall not be any other definition of this operator in Milena when applying on a couple of [mln::Object](#).

9.1.3.37 `template<unsigned D, typename G > bool mln::operator!=(const complex_psite< D, G > & lhs, const complex_psite< D, G > & rhs)`

Is *lhs* not equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::p_complex](#).

Definition at line 353 of file `complex_psite.hh`.

References `mln::complex_psite< D, G >::face()`, and `mln::complex_psite< D, G >::site_set()`.

9.1.3.38 `template<unsigned N, unsigned D, typename P > bool mln::operator!=(const faces_psite< N, D, P > & lhs, const faces_psite< N, D, P > & rhs)`

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same `mln::complex`.

Definition at line 336 of file `faces_psite.hh`.

References `mln::faces_psite< N, D, P >::face()`, and `mln::faces_psite< N, D, P >::site_set()`.

9.1.3.39 `template<typename P, typename S > P mln::operator*(const Gpoint< P > & p, const value::scalar_< S > & s) [inline]`

Multiply a point *p* by a scalar *s*.

Definition at line 405 of file `gpoint.hh`.

9.1.3.40 `template<typename S > S & mln::operator++ (value::Scalar< S > & rhs) [inline]`

Pre-incrementation for any scalar type.

Definition at line 77 of file `concept/scalar.hh`.

References `mln::literal::one`.

9.1.3.41 `template<typename N1, typename N2 > N2 neighb< typename N1::window::regular > mln::operator-(const Neighborhood< N1 > & nbh1, const Neighborhood< N2 > & nbh2)`

Set difference between a couple of neighborhoods *nbh1* and *nbh2*.

It just calls [mln::win::diff](#).

Definition at line 155 of file `win/diff.hh`.

References `mln::win::diff()`.

9.1.3.42 `template<typename P, typename D> P mln::operator- (const Gpoint< P > & p, const Gdpoint< D > & dp) [inline]`

Subtract a delta-point `dp` to a grid point `p`.

Parameters

[in] `p` A grid point.

[in] `dp` A delta-point.

The type of `dp` has to compatible with the type of `p`.

Returns

A point (temporary object).

See also

[mln::Gdpoint](#)

[mln::Gdpoint](#)

Definition at line 395 of file `gpoint.hh`.

9.1.3.43 `template<typename S> S & mln::operator-- (value::Scalar< S > & rhs) [inline]`

Pre-decrementation for any scalar type.

Definition at line 85 of file `concept/scalar.hh`.

References `mln::literal::one`.

9.1.3.44 `template<unsigned D, typename G> bool mln::operator< (const complex_psite< D, G > & lhs, const complex_psite< D, G > & rhs)`

Is `lhs` “less” than `rhs`?

This comparison is required by algorithms sorting psites.

Precondition

Arguments `lhs` and `rhs` must belong to the same [mln::p_complex](#).

Definition at line 362 of file `complex_psite.hh`.

9.1.3.45 `template<unsigned N, unsigned D, typename P> bool mln::operator< (const faces_psite< N, D, P > & lhs, const faces_psite< N, D, P > & rhs)`

Is `lhs` “less” than `rhs`?

This comparison is required by algorithms sorting psites.

Precondition

Arguments `lhs` and `rhs` must belong to the same `mln::complex`.

Definition at line 345 of file `faces_psite.hh`.

9.1.3.46 `template<typename L , typename R > bool mln::operator< (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise test if the pixel values of `lhs` are point-wise less than the pixel values of `rhs`.

Parameters

[in] *lhs* A first image.

[in] *rhs* A second image.

Precondition

`lhs.domain == rhs.domain`

Definition at line 107 of file `compare.hh`.

References `mln::test::predicate()`.

9.1.3.47 `template<typename I , typename G , typename W > std::ostream & mln::operator<< (std::ostream & ostr, const complex_window_bkd_piter< I, G, W > & p) [inline]`

Print an [mln::complex_window_bkd_piter](#).

Definition at line 401 of file `complex_window_piter.hh`.

9.1.3.48 `template<typename I , typename G , typename W > std::ostream & mln::operator<< (std::ostream & ostr, const complex_window_fwd_piter< I, G, W > & p) [inline]`

Print an [mln::complex_window_fwd_piter](#).

Definition at line 292 of file `complex_window_piter.hh`.

9.1.3.49 `template<typename I , typename G , typename N > std::ostream & mln::operator<< (std::ostream & ostr, const complex_neighborhood_fwd_piter< I, G, N > & p) [inline]`

Print an [mln::complex_neighborhood_fwd_piter](#).

Definition at line 292 of file `complex_neighborhood_piter.hh`.

9.1.3.50 `template<typename I , typename G , typename N > std::ostream & mln::operator<< (std::ostream & ostr, const complex_neighborhood_bkd_piter< I, G, N > & p) [inline]`

Print an [mln::complex_neighborhood_bkd_piter](#).

Definition at line 399 of file `complex_neighborhood_piter.hh`.

9.1.3.51 `template<unsigned D, typename G > bool mln::operator<= (const p_complex< D, G > & lhs, const p_complex< D, G > & rhs)`

Inclusion of a [mln::p_complex](#) in another one.

This inclusion relation is very strict for the moment, since our infrastructure for complexes is simple: a [mln::p_complex](#) is included in another one if their are equal.

Definition at line 330 of file p_complex.hh.

9.1.3.52 `template<typename G , typename F > bool mln::operator<= (const p_edges< G, F > & lhs, const p_edges< G, F > & rhs)`

Inclusion of a [mln::p_edges](#) in another one.

Definition at line 338 of file p_edges.hh.

9.1.3.53 `template<typename L , typename R > bool mln::operator<= (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise test if the pixel values of `lhs` are point-wise less than or equal to the pixel values of `rhs`.

Parameters

[in] *lhs* A first image.

[in] *rhs* A second image.

Precondition

`lhs.domain == rhs.domain`

Definition at line 126 of file compare.hh.

References `mln::test::predicate()`.

9.1.3.54 `template<typename G , typename F > bool mln::operator<= (const p_vertices< G, F > & lhs, const p_vertices< G, F > & rhs)`

Inclusion of a [mln::p_vertices](#) in another one.

This inclusion relation is very strict for the moment, since our infrastructure for graphs is simple: a [mln::p_vertices](#) is included in another one if their are equal.

Definition at line 399 of file p_vertices.hh.

9.1.3.55 `template<unsigned N, unsigned D, typename P > bool mln::operator<= (const p_faces< N, D, P > & lhs, const p_faces< N, D, P > & rhs)`

Inclusion of a [mln::p_faces](#) in another one.

This inclusion relation is very strict for the moment, since our infrastructure for complexes is simple: a [mln::p_faces](#) is included in another one if their are equal.

Definition at line 287 of file p_faces.hh.

9.1.3.56 `template<unsigned N, unsigned D, typename P > bool mln::operator== (const p_faces< N, D, P > & lhs, const p_faces< N, D, P > & rhs)`

Comparison between two [mln::p_faces](#)'s.

Two `mln::p_faces`'s are considered equal if they share the same complex.

Definition at line 278 of file `p_faces.hh`.

References `mln::p_faces< N, D, P >::cplx()`.

9.1.3.57 `template<unsigned N, unsigned D, typename P > bool mln::operator==(const faces_psite< N, D, P > & lhs, const faces_psite< N, D, P > & rhs)`

Comparison of two instances of `mln::faces_psite`.

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same `mln::complex`.

Definition at line 327 of file `faces_psite.hh`.

References `mln::faces_psite< N, D, P >::face()`, and `mln::faces_psite< N, D, P >::site_set()`.

9.1.3.58 `template<typename L, typename R > bool mln::operator==(const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise test if the pixel values of *lhs* are equal to the pixel values of *rhs*.

Parameters

[in] *lhs* A first image.

[in] *rhs* A second image.

Precondition

`lhs.domain == rhs.domain`

Definition at line 86 of file `compare.hh`.

References `mln::test::predicate()`.

9.1.3.59 `template<unsigned D, typename G > bool mln::operator==(const p_complex< D, G > & lhs, const p_complex< D, G > & rhs)`

Comparison between two `mln::p_complex`'s.

Two `mln::p_complex`'s are considered equal if they share the same complex.

Definition at line 321 of file `p_complex.hh`.

References `mln::p_complex< D, G >::cplx()`.

9.1.3.60 `template<unsigned D, typename G > bool mln::operator==(const complex_psite< D, G > & lhs, const complex_psite< D, G > & rhs)`

Comparison of two instances of `mln::complex_psite`.

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::p_complex](#).

Definition at line 344 of file `complex_psite.hh`.

References `mln::complex_psite< D, G >::face()`, and `mln::complex_psite< D, G >::site_set()`.

9.1.3.61 `template<typename G , typename F > bool mln::operator==(const p_vertices< G, F > & lhs, const p_vertices< G, F > & rhs)`

Comparison between two [mln::p_vertices](#)'s.

Two [mln::p_vertices](#)'s are considered equal if they share the same graph.

Definition at line 392 of file `p_vertices.hh`.

References `mln::p_vertices< G, F >::graph()`.

9.1.3.62 `template<typename G , typename F > bool mln::operator==(const p_edges< G, F > & lhs, const p_edges< G, F > & rhs)`

Comparison between two [mln::p_edges](#)'s.

Two [mln::p_edges](#)'s are considered equal if they share the same graph.

Definition at line 331 of file `p_edges.hh`.

References `mln::p_edges< G, F >::graph()`.

9.1.3.63 `template<typename F , typename S > pw::image<F,S> mln::operator|(const Function_v2v< F > & f, const Site_Set< S > & ps)`

Construct an image from a function and a site set.

`image = function | site_set`.

Definition at line 138 of file `pw/image.hh`.

9.1.3.64 `template<typename V , typename G , typename P > edge_image<P,V,G> mln::operator|(const fun::i2v::array< V > & edge_values, const p_edges< G, fun::i2v::array< P > > & pe)`

Construct an edge image from a `fun::i2v::array` and a [p_edges](#).

`image = fun::i2v::array | p_edges`.

Definition at line 217 of file `core/image/edge_image.hh`.

9.1.3.65 `template<typename S , typename F > p_if<S, F> mln::operator|(const Site_Set< S > & s, const Function_v2b< F > & f)`

Restrict a site set *s* to points that verify *f*.

Parameters

[in] *s* A site set.

[in] *f* A function from point to Boolean.

Returns

A subset of points.

Definition at line 147 of file p_if.hh.

9.1.3.66 `template<typename V , typename G , typename P > vertex_image<P,V,G>
mln::operator| (const fun::i2v::array< V > & vertex_values, const p_vertices< G,
fun::i2v::array< P > & pv)`

Construct a vertex image from a `fun::i2v::array` and a `p_vertices`.

`image = fun::i2v::array | p_vertices`.

Definition at line 215 of file core/image/vertex_image.hh.

9.1.3.67 `template<typename I , typename F > image_if<I,F> mln::operator| (Image< I > &
ima, const Function_v2b< F > & f)`

`ima | f` creates an `image_if` with the image `ima` and the function `f`.

Definition at line 233 of file image_if.hh.

9.1.3.68 `template<typename I , typename F > image_if<const I,F> mln::operator| (const
Image< I > & ima, const Function_v2b< F > & f)`

`ima | f` creates an `image_if` with the image `ima` and the function `f`.

Definition at line 242 of file image_if.hh.

9.1.3.69 `template<typename I > const internal::primary_type< I >::ret & mln::primary (const
Image< I > & input) [inline]`

FIXME: Doc!

Definition at line 129 of file primary.hh.

Referenced by `mln::border::resize()`.

9.1.3.70 `template<typename S , typename F > p_transformed< S, F > mln::ptransform (const
Site_Set< S > & s, const Function_v2v< F > & f) [inline]`

Transform a site set `s` through the function `f`.

Parameters

[in] *s* A site set.

[in] *f* A function from site to site.

Returns

The transformed site set.

Definition at line 145 of file p_transformed.hh.

9.1.4 Variable Documentation

9.1.4.1 `const dpoint1d mln::before = dpoint1d(-1)`

Definition of a shortcut for delta point in 1d.

Definition at line 73 of file `dpoint1d.hh`.

9.1.4.2 `const dpoint3d mln::sagittal_dec = dpoint3d(0, 0, -1)`

Definition of a shortcut for delta point in 3d.

Definition at line 72 of file `dpoint3d.hh`.

9.1.4.3 `const dpoint2d mln::up = dpoint2d(-1, 0)`

Definition of a shortcut for delta point in 2d.

Definition at line 76 of file `dpoint2d.hh`.

9.2 `mln::accu` Namespace Reference

Namespace of accumulators.

Namespaces

- namespace [image](#)
Namespace of accumulator image routines.
- namespace [impl](#)
Implementation namespace of accumulator namespace.
- namespace [logic](#)
Namespace of logical accumulators.
- namespace [math](#)
Namespace of mathematic accumulators.
- namespace [shape](#)
Namespace of shape accumulators.
- namespace [stat](#)
Namespace of statistical accumulators.

Classes

- struct [center](#)
Mass center accumulator.

- struct [convolve](#)
Generic convolution accumulator class.
- struct [count_adjacent_vertices](#)
Accumulator class counting the number of vertices adjacent to a set of mln::p_edges_psite (i.e., a set of edges).
- struct [count_labels](#)
Count the number of different labels in an image.
- struct [count_value](#)
Define an accumulator that counts the occurrence of a given value.
- struct [histo](#)
Generic histogram class over a value set with type V.
- struct [label_used](#)
References all the labels used.
- struct [maj_h](#)
Compute the majority value.
- struct [max_site](#)
Define an accumulator that computes the first site with the maximum value in an image.
- struct [nil](#)
Define an accumulator that does nothing.
- struct [p](#)
Generic p of accumulators.
- struct [pair](#)
Generic pair of accumulators.
- struct [rms](#)
Generic root mean square accumulator class.
- struct [tuple](#)
Generic tuple of accumulators.
- struct [val](#)
Generic val of accumulators.

Functions

- `template<typename A , typename I >`
`A::result compute (const Accumulator< A > &a, const Image< I > &input)`
Make an accumulator compute the pixels of the image `input`.

- `template<typename Meta_Accu , unsigned Dir, typename I , typename O >`
`void line (const Image< I > &input, const typename I::site &p_start, unsigned len, unsigned half_length, Image< O > &output)`
- `template<typename A , typename I >`
`mln_meta_accu_result (A, util::pix< I >) compute(const Meta_Accumulator< A > &a`
Make an accumulator compute the pixels of the image input.
- `template<typename A , typename I >`
`void take (const Image< I > &input, Accumulator< A > &a)`
Make an accumulator take the pixels of the image input.

9.2.1 Detailed Description

Namespace of accumulators.

9.2.2 Function Documentation

9.2.2.1 `template<typename A , typename I > A::result mln::accu::compute (const Accumulator< A > & a, const Image< I > & input) [inline]`

Make an accumulator compute the pixels of the image `input`.

Parameters

- [in] *input* The input image.
- [in] *a* An accumulator.

This routine runs:

`a.take(make::pix(input, p));` on all pixels on the images.

Warning

This routine does not perform `a.init()`.

Definition at line 130 of file `accu/compute.hh`.

9.2.2.2 `template<typename Meta_Accu , unsigned Dir, typename I , typename O > void mln::accu::line (const Image< I > & input, const typename I::site & p_start, unsigned len, unsigned half_length, Image< O > & output)`

Line an accumulator onto the pixel values of the image `input`.

Parameters

- [in] *input* The input image.
- [in] *p_start* The starting site of the line.
- [in] *len* The line length.

[in] *half_length* The half length of the line.

[in, out] *output* The resulting image.

This routine runs:

```
tmp = a
tmp.init()
accu::take(input, tmp)
return tmp.to_result()
```

Definition at line 381 of file accu/line.hh.

9.2.2.3 `template<typename A , typename I > mln::accu::mln_meta_accu_result (A , util::pix< I >) const [inline]`

Make an accumulator compute the pixels of the image `input`.

Parameters

[in] *input* The input image.

[in] *a* A meta accumulator.

This routine runs:

`a.take(make::pix(input, p));` on all pixels on the images.

Warning

This routine does not perform `a.init()`.

9.2.2.4 `template<typename A , typename I > void mln::accu::take (const Image< I > & input, Accumulator< A > & a) [inline]`

Make an accumulator take the pixels of the image `input`.

Parameters

[in] *input* The input image.

[in, out] *a* The accumulator.

This routine runs:

for all `p` of `input`, `a.take(pix(input, p))`

Warning

This routine does not perform `a.init()`.

Definition at line 87 of file take.hh.

9.3 mln::accu::image Namespace Reference

Namespace of accumulator image routines.

9.3.1 Detailed Description

Namespace of accumulator image routines.

9.4 mln::accu::impl Namespace Reference

Implementation namespace of accumulator namespace.

9.4.1 Detailed Description

Implementation namespace of accumulator namespace.

9.5 mln::accu::logic Namespace Reference

Namespace of logical accumulators.

Classes

- struct [land](#)
"Logical-and" accumulator.
- struct [land_basic](#)
"Logical-and" accumulator.
- struct [lor](#)
"Logical-or" accumulator.
- struct [lor_basic](#)
"Logical-or" accumulator class.

9.5.1 Detailed Description

Namespace of logical accumulators.

9.6 mln::accu::math Namespace Reference

Namespace of mathematic accumulators.

Classes

- struct [count](#)
Generic counter accumulator.
- struct [inf](#)
Generic inf accumulator class.
- struct [sum](#)
Generic sum accumulator class.
- struct [sup](#)
Generic sup accumulator class.

9.6.1 Detailed Description

Namespace of mathematic accumulators.

9.7 mln::accu::meta::logic Namespace Reference

Namespace of logical meta-accumulators.

Classes

- struct [land](#)
Meta accumulator for land.
- struct [lor](#)
Meta accumulator for lor.
- struct [lor_basic](#)
Meta accumulator for [lor_basic](#).

9.7.1 Detailed Description

Namespace of logical meta-accumulators.

9.8 mln::accu::meta::math Namespace Reference

Namespace of mathematic meta-accumulators.

Classes

- struct [count](#)
Meta accumulator for count.
- struct [inf](#)
Meta accumulator for inf.
- struct [sum](#)
Meta accumulator for sum.
- struct [sup](#)
Meta accumulator for sup.

9.8.1 Detailed Description

Namespace of mathematic meta-accumulators.

9.9 mln::accu::meta::shape Namespace Reference

Namespace of shape meta-accumulators.

Classes

- struct [bbox](#)
Meta accumulator for bbox.
- struct [height](#)
Meta accumulator for height.
- struct [volume](#)
Meta accumulator for volume.

9.9.1 Detailed Description

Namespace of shape meta-accumulators.

9.10 mln::accu::meta::stat Namespace Reference

Namespace of statistical meta-accumulators.

Classes

- struct [max](#)
Meta accumulator for max.
- struct [max_h](#)
Meta accumulator for max.
- struct [mean](#)
Meta accumulator for mean.
- struct [median_alt](#)
Meta accumulator for [median_alt](#).
- struct [median_h](#)
Meta accumulator for [median_h](#).
- struct [min](#)
Meta accumulator for min.
- struct [min_h](#)
Meta accumulator for min.
- struct [rank](#)
Meta accumulator for rank.
- struct [rank_high_quant](#)
Meta accumulator for [rank_high_quant](#).

9.10.1 Detailed Description

Namespace of statistical meta-accumulators.

9.11 mln::accu::shape Namespace Reference

Namespace of shape accumulators.

Classes

- struct [bbox](#)
Generic bounding box accumulator class.
- struct [height](#)
Height accumulator.
- struct [volume](#)
Volume accumulator class.

9.11.1 Detailed Description

Namespace of shape accumulators.

9.12 mln::accu::stat Namespace Reference

Namespace of statistical accumulators.

Classes

- struct [deviation](#)
Generic standard deviation accumulator class.
- struct [histo3d_rgb](#)
Define a histogram as accumulator which returns an [image3d](#).
- struct [max](#)
Generic max accumulator class.
- struct [max_h](#)
Generic max function based on histogram over a value set with type V .
- struct [mean](#)
Generic mean accumulator class.
- struct [median_alt](#)
Generic [median_alt](#) function based on histogram over a value set with type S .
- struct [median_h](#)
Generic median function based on histogram over a value set with type V .
- struct [min](#)
Generic min accumulator class.
- struct [min_h](#)
Generic min function based on histogram over a value set with type V .
- struct [min_max](#)
Generic min and max accumulator class.
- struct [rank](#)
Generic rank accumulator class.
- struct [rank< bool >](#)
rank accumulator class for Boolean.
- struct [rank_high_quant](#)
Generic rank accumulator class.

- struct [var](#)
Var accumulator class.
- struct [variance](#)
Variance accumulator class.

Functions

- template<typename V >
bool [operator==](#) (const [histo3d_rgb](#)< V > &histo1, const [histo3d_rgb](#)< V > &histo2)
Check whether an histogram is equal to another one.

9.12.1 Detailed Description

Namespace of statistical accumulators.

9.12.2 Function Documentation

9.12.2.1 template<typename V > bool mln::accu::stat::operator== (const [histo3d_rgb](#)< V > & *histo1*, const [histo3d_rgb](#)< V > & *histo2*)

Check whether an histogram is equal to another one.

Parameters

- [in] *histo1* the first histogram to compare with.
- [in] *histo2* the second histogram.

The operator compares all the bins from the two histograms. Nobody uses this method unless unitary tests.

Definition at line 315 of file [histo3d_rgb.hh](#).

References [mln::image3d< T >::domain\(\)](#).

9.13 mln::algebra Namespace Reference

Namespace of algebraic structure.

Classes

- struct [h_mat](#)
N-Dimensional matrix with homogeneous coordinates.
- class [h_vec](#)
N-Dimensional vector with homogeneous coordinates.

Functions

- `template<unsigned N, typename T >`
`bool ldlt_decomp (mat< N, N, T > &A, vec< N, T > &rdiag)`
Perform LDL^T decomposition of a symmetric positive definite matrix.
- `template<unsigned N, typename T >`
`void ldlt_solve (const mat< N, N, T > &A, const vec< N, T > &rdiag, const vec< N, T > &B, vec< N, T > &x)`
Solve $Ax = B$ after [mln::algebra::ldlt_decomp](#).
- `template<unsigned n, typename T, typename U >`
`mln::trait::value_< typename mln::trait::op::times< T, U >::ret >::sum operator* (const vec< n, T > &lhs, const vec< n, U > &rhs)`
Scalar product (dot product).
- `template<typename T, typename U >`
`vec< 3, typename mln::trait::op::times< T, U >::ret > vprod (const vec< 3, T > &lhs, const vec< 3, U > &rhs)`
Vectorial product (cross product).

9.13.1 Detailed Description

Namespace of algebraic structure.

9.13.2 Function Documentation

9.13.2.1 `template<unsigned N, typename T > bool mln::algebra::ldlt_decomp (mat< N, N, T > & A, vec< N, T > & rdiag) [inline]`

Perform LDL^T decomposition of a symmetric positive definite matrix.

Like Cholesky, but no square roots. Overwrites lower triangle of matrix.

From Trimesh's `ldltde` routine.

Definition at line 79 of file `misc.hh`.

Referenced by `mln::geom::mesh_curvature()`.

9.13.2.2 `template<unsigned N, typename T > void mln::algebra::ldlt_solve (const mat< N, N, T > & A, const vec< N, T > & rdiag, const vec< N, T > & B, vec< N, T > & x) [inline]`

Solve $Ax = B$ after [mln::algebra::ldlt_decomp](#).

Definition at line 112 of file `misc.hh`.

Referenced by `mln::geom::mesh_curvature()`.

9.13.2.3 `template<unsigned n, typename T , typename U > mln::trait::value_< typename mln::trait::op::times< T, U >::ret >::sum mln::algebra::operator* (const vec< n, T > & lhs, const vec< n, U > & rhs) [inline]`

Scalar product (dot product).

Definition at line 632 of file algebra/vec.hh.

References mln::literal::zero.

9.13.2.4 `template<typename T , typename U > vec< 3, typename mln::trait::op::times< T, U >::ret > mln::algebra::vprod (const vec< 3, T > & lhs, const vec< 3, U > & rhs) [inline]`

Vectorial product (cross product).

Definition at line 713 of file algebra/vec.hh.

References vprod().

Referenced by mln::geom::mesh_corner_point_area(), mln::geom::mesh_curvature(), mln::geom::mesh_normal(), and vprod().

9.14 mln::arith Namespace Reference

Namespace of arithmetic.

Namespaces

- namespace [impl](#)
Implementation namespace of arith namespace.

Functions

- `template<typename I > mln::trait::concrete< I >::ret diff_abs (const Image< I > &lhs, const Image< I > &rhs)`
Point-wise absolute difference of images lhs and rhs.
- `template<typename L , typename R , typename O > void div (const Image< L > &lhs, const Image< R > &rhs, Image< O > &output)`
Point-wise division of images lhs and rhs.
- `template<typename I , typename V , typename O > void div_cst (const Image< I > &input, const V &val, Image< O > &output)`
Point-wise division of the value val to image input.
- `template<typename L , typename R > void div_inplace (Image< L > &lhs, const Image< R > &rhs)`
Point-wise division of image rhs in image lhs.

- `template<typename L, typename R >`
`mln::trait::concrete< L >::ret min (const Image< L > &lhs, const Image< R > &rhs)`
Point-wise min of images lhs and rhs.
- `template<typename L, typename R >`
`void min_inplace (Image< L > &lhs, const Image< R > &rhs)`
Point-wise min of image lhs in image rhs.
- `template<typename L, typename R >`
`mln::trait::op::minus< L, R >::ret minus (const Image< L > &lhs, const Image< R > &rhs)`
Point-wise addition of images lhs and rhs.
- `template<typename L, typename R, typename F >`
`mln::trait::ch_value< L, typename F::result >::ret minus (const Image< L > &lhs, const Image< R > &rhs, const Function_v2v< F > &f)`
Point-wise addition of images lhs and rhs.
- `template<typename V, typename L, typename R >`
`mln::trait::ch_value< L, V >::ret minus (const Image< L > &lhs, const Image< R > &rhs)`
Point-wise addition of images lhs and rhs.
- `template<typename I, typename V >`
`mln::trait::op::minus< I, V >::ret minus_cst (const Image< I > &input, const V &val)`
Point-wise addition of the value val to image input.
- `template<typename I, typename V, typename F >`
`mln::trait::ch_value< I, typename F::result >::ret minus_cst (const Image< I > &input, const V &val, const Function_v2v< F > &f)`
Point-wise addition of the value val to image input.
- `template<typename I, typename V >`
`I & minus_cst_inplace (Image< I > &input, const V &val)`
Point-wise addition of the value val to image input.
- `template<typename L, typename R >`
`void minus_inplace (Image< L > &lhs, const Image< R > &rhs)`
Point-wise addition of image rhs in image lhs.
- `template<typename L, typename R, typename F >`
`mln::trait::ch_value< L, typename F::result >::ret plus (const Image< L > &lhs, const Image< R > &rhs, const Function_v2v< F > &f)`
Point-wise addition of images lhs and rhs.
- `template<typename V, typename L, typename R >`
`mln::trait::ch_value< L, V >::ret plus (const Image< L > &lhs, const Image< R > &rhs)`
Point-wise addition of images lhs and rhs.
- `template<typename L, typename R >`
`mln::trait::op::plus< L, R >::ret plus (const Image< L > &lhs, const Image< R > &rhs)`
Point-wise addition of images lhs and rhs.

- `template<typename I, typename V >`
`mln::trait::op::plus< I, V >::ret plus_cst (const Image< I > &input, const V &val)`
Point-wise addition of the value val to image input.

- `template<typename I, typename V, typename F >`
`mln::trait::ch_value< I, typename F::result >::ret plus_cst (const Image< I > &input, const V &val, const Function_v2v< F > &f)`
Point-wise addition of the value val to image input.

- `template<typename W, typename I, typename V >`
`mln::trait::ch_value< I, W >::ret plus_cst (const Image< I > &input, const V &val)`
Point-wise addition of the value val to image input.

- `template<typename I, typename V >`
`I & plus_cst_inplace (Image< I > &input, const V &val)`
Point-wise addition of the value val to image input.

- `template<typename L, typename R >`
`void plus_inplace (Image< L > &lhs, const Image< R > &rhs)`
Point-wise addition of image rhs in image lhs.

- `template<typename I >`
`mln::trait::concrete< I >::ret revert (const Image< I > &input)`
Point-wise reversion of image input.

- `template<typename I >`
`void revert_inplace (Image< I > &input)`
Point-wise in-place reversion of image input.

- `template<typename L, typename R, typename O >`
`void times (const Image< L > &lhs, const Image< R > &rhs, Image< O > &output)`
Point-wise addition of images lhs and rhs.

- `template<typename I, typename V, typename O >`
`void times_cst (const Image< I > &input, const V &val, Image< O > &output)`
Point-wise addition of the value val to image input.

- `template<typename L, typename R >`
`void times_inplace (Image< L > &lhs, const Image< R > &rhs)`
Point-wise addition of image rhs in image lhs.

9.14.1 Detailed Description

Namespace of arithmetic.

9.14.2 Function Documentation

9.14.2.1 `template<typename I> mln::trait::concrete< I >::ret mln::arith::diff_abs (const Image< I > & lhs, const Image< I > & rhs) [inline]`

Point-wise absolute difference of images `lhs` and `rhs`.

Parameters

- [in] *lhs* First operand image.
- [in] *rhs* Second operand image.

Returns

The result image.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 63 of file `arith/diff_abs.hh`.

References `mln::data::transform()`.

9.14.2.2 `template<typename L , typename R , typename O > void mln::arith::div (const Image< L > & lhs, const Image< R > & rhs, Image< O > & output) [inline]`

Point-wise division of images `lhs` and `rhs`.

Parameters

- [in] *lhs* First operand image.
- [in] *rhs* Second operand image.
- [out] *output* The result image.

Precondition

```
output.domain == lhs.domain == rhs.domain
```

Definition at line 227 of file `arith/div.hh`.

9.14.2.3 `template<typename I , typename V , typename O > void mln::arith::div_cst (const Image< I > & input, const V & val, Image< O > & output) [inline]`

Point-wise division of the value `val` to image `input`.

Parameters

- [in] *input* The image.
- [in] *val* The value.
- [out] *output* The result image.

Precondition

```
output.domain == input.domain
```

Definition at line 242 of file arith/div.hh.

References `div_cst()`.

Referenced by `div_cst()`.

9.14.2.4 `template<typename L , typename R > void mln::arith::div_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise division of image `rhs` in image `lhs`.

Parameters

[in] *lhs* First operand image (subject to division).

[in, out] *rhs* Second operand image (to div `lhs`).

This addition performs:

for all `p` of `rhs.domain`

`lhs(p) /= rhs(p)`

Precondition

```
rhs.domain <= lhs.domain
```

Definition at line 255 of file arith/div.hh.

References `div_inplace()`.

Referenced by `div_inplace()`.

9.14.2.5 `template<typename L , typename R > mln::trait::concrete< L >::ret mln::arith::min (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise min of images `lhs` and `rhs`.

Parameters

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

Returns

The result image.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 112 of file arith/min.hh.

References `mln::initialize()`.

9.14.2.6 `template<typename L , typename R > void mln::arith::min_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise min of image `lhs` in image `rhs`.

Parameters

[in, out] *lhs* First operand image.

[in] *rhs* Second operand image.

Precondition

```
rhs.domain == lhs.domain
```

Definition at line 128 of file `arith/min.hh`.

9.14.2.7 `template<typename L , typename R > mln::trait::op::minus< L, R >::ret mln::arith::minus (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

Returns

The result image.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 331 of file `arith/minus.hh`.

References `mln::initialize()`.

9.14.2.8 `template<typename L , typename R , typename F > mln::trait::ch_value< L, typename F::result >::ret mln::arith::minus (const Image< L > & lhs, const Image< R > & rhs, const Function_v2v< F > & f) [inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

[in] *f* [Function](#).

Returns

The result image.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 350 of file arith/minus.hh.

References mln::initialize().

9.14.2.9 `template<typename V , typename L , typename R > mln::trait::ch_value< L, V >::ret mln::arith::minus (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

Returns

The result image.

The free parameter `V` sets the destination value type.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 369 of file arith/minus.hh.

9.14.2.10 `template<typename I , typename V > mln::trait::op::minus< I, V >::ret mln::arith::minus_cst (const Image< I > & input, const V & val) [inline]`

Point-wise addition of the value `val` to image `input`.

Parameters

[in] *input* The image.

[in] *val* The value.

Returns

The result image.

Precondition

```
input.is_valid
```

Definition at line 387 of file arith/minus.hh.

9.14.2.11 `template<typename I, typename V, typename F> mln::trait::ch_value< I, typename F::result >::ret mln::arith::minus_cst (const Image< I > & input, const V & val, const Function_v2v< F > & f) [inline]`

Point-wise addition of the value *val* to image *input*.

Parameters

[in] *input* The image.

[in] *val* The value.

[in] *f* [Function](#).

Returns

The result image.

Precondition

`input.is_valid`

Definition at line 405 of file arith/minus.hh.

9.14.2.12 `template<typename I, typename V> I & mln::arith::minus_cst_inplace (Image< I > & input, const V & val) [inline]`

Point-wise addition of the value *val* to image *input*.

Parameters

[in, out] *input* The image.

[in] *val* The value.

Precondition

`input.is_valid`

Definition at line 440 of file arith/minus.hh.

References `minus_cst_inplace()`, and `minus_inplace()`.

Referenced by `minus_cst_inplace()`.

9.14.2.13 `template<typename L, typename R> void mln::arith::minus_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of image *rhs* in image *lhs*.

Parameters

[in, out] *lhs* First operand image (subject to addition).

[in] *rhs* Second operand image (to be added to *lhs*).

This addition performs:

for all *p* of *rhs*.domain

`lhs(p) -= rhs(p)`

Precondition

```
rhs.domain == lhs.domain
```

Definition at line 424 of file arith/minus.hh.

References `minus_inplace()`.

Referenced by `minus_cst_inplace()`, and `minus_inplace()`.

9.14.2.14 `template<typename L , typename R , typename F > mln::trait::ch_value< L, typename F::result >::ret mln::arith::plus (const Image< L > & lhs, const Image< R > & rhs, const Function_v2v< F > & f) [inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

[in] *f* Function.

Returns

The result image.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 367 of file arith/plus.hh.

References `mln::initialize()`.

9.14.2.15 `template<typename V , typename L , typename R > mln::trait::ch_value< L, V >::ret mln::arith::plus (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

Returns

The result image.

The free parameter `V` sets the destination value type.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 386 of file arith/plus.hh.

9.14.2.16 `template<typename L , typename R > mln::trait::op::plus< L, R >::ret
mln::arith::plus (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

- [in] *lhs* First operand image.
- [in] *rhs* Second operand image.

Returns

The result image.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 348 of file `arith/plus.hh`.

References `mln::initialize()`.

9.14.2.17 `template<typename I , typename V > mln::trait::op::plus< I, V >::ret
mln::arith::plus_cst (const Image< I > & input, const V & val) [inline]`

Point-wise addition of the value `val` to image `input`.

Parameters

- [in] *input* The image.
- [in] *val* The value.

Returns

The result image.

Precondition

```
input.is_valid
```

Definition at line 404 of file `arith/plus.hh`.

Referenced by `plus_cst()`.

9.14.2.18 `template<typename I , typename V , typename F > mln::trait::ch_value< I, typename
F::result >::ret mln::arith::plus_cst (const Image< I > & input, const V & val, const
Function_v2v< F > & f) [inline]`

Point-wise addition of the value `val` to image `input`.

Parameters

- [in] *input* The image.
- [in] *val* The value.

[in] *f* Function.

Returns

The result image.

Precondition

`input.is_valid`

Definition at line 422 of file arith/plus.hh.

9.14.2.19 `template<typename W , typename I , typename V > mln::trait::ch_value< I, W >::ret mln::arith::plus_cst (const Image< I > & input, const V & val) [inline]`

Point-wise addition of the value `val` to image `input`.

Parameters

[in] *input* The image.

[in] *val* The value.

Returns

The result image.

Precondition

`input.is_valid`

Definition at line 441 of file arith/plus.hh.

References `plus_cst()`.

9.14.2.20 `template<typename I , typename V > I & mln::arith::plus_cst_inplace (Image< I > & input, const V & val) [inline]`

Point-wise addition of the value `val` to image `input`.

Parameters

[in, out] *input* The image.

[in] *val* The value.

Precondition

`input.is_valid`

Definition at line 475 of file arith/plus.hh.

References `plus_cst_inplace()`, and `plus_inplace()`.

Referenced by `plus_cst_inplace()`.

9.14.2.21 `template<typename L , typename R > void mln::arith::plus_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of image `rhs` in image `lhs`.

Parameters

[in, out] *lhs* First operand image (subject to addition).

[in] *rhs* Second operand image (to be added to `lhs`).

This addition performs:

for all `p` of `rhs.domain`

`lhs(p) += rhs(p)`

Precondition

```
rhs.domain == lhs.domain
```

Definition at line 459 of file `arith/plus.hh`.

References `plus_inplace()`.

Referenced by `plus_cst_inplace()`, and `plus_inplace()`.

9.14.2.22 `template<typename I > mln::trait::concrete< I >::ret mln::arith::revert (const Image< I > & input) [inline]`

Point-wise reversion of image `input`.

Parameters

[in] *input* the input image.

Returns

The result image.

Precondition

```
input.is_valid
```

It performs:

for all `p` of `input.domain`

`output(p) = min + (max - input(p))`

Definition at line 158 of file `revert.hh`.

References `mln::initialize()`.

9.14.2.23 `template<typename I > void mln::arith::revert_inplace (Image< I > & input) [inline]`

Point-wise in-place reversion of image `input`.

Parameters

[in, out] *input* The target image.

Precondition

```
input.is_valid
```

It performs:

for all p of input.domain

input(p) = min + (max - input(p))

Definition at line 174 of file revert.hh.

9.14.2.24 `template<typename L , typename R , typename O > void mln::arith::times (const Image< L > & lhs, const Image< R > & rhs, Image< O > & output) [inline]`

Point-wise addition of images lhs and rhs.

Parameters

[in] *lhs* First operand image.

[in] *rhs* Second operand image.

[out] *output* The result image.

Precondition

```
output.domain == lhs.domain == rhs.domain
```

Definition at line 226 of file arith/times.hh.

9.14.2.25 `template<typename I , typename V , typename O > void mln::arith::times_cst (const Image< I > & input, const V & val, Image< O > & output) [inline]`

Point-wise addition of the value val to image input.

Parameters

[in] *input* The image.

[in] *val* The value.

[out] *output* The result image.

Precondition

```
output.domain == input.domain
```

Definition at line 241 of file arith/times.hh.

References times_cst().

Referenced by times_cst().

9.14.2.26 `template<typename L , typename R > void mln::arith::times_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of image `rhs` in image `lhs`.

Parameters

- [in] *lhs* First operand image (subject to addition).
- [in, out] *rhs* Second operand image (to be added to `lhs`).

This addition performs:

for all `p` of `rhs.domain`

`lhs(p) *= rhs(p)`

Precondition

`rhs.domain <= lhs.domain`

Definition at line 254 of file `arith/times.hh`.

References `times_inplace()`.

Referenced by `times_inplace()`.

9.15 `mln::arith::impl` Namespace Reference

Implementation namespace of `arith` namespace.

Namespaces

- namespace [generic](#)
Generic implementation namespace of `arith` namespace.

9.15.1 Detailed Description

Implementation namespace of `arith` namespace.

9.16 `mln::arith::impl::generic` Namespace Reference

Generic implementation namespace of `arith` namespace.

9.16.1 Detailed Description

Generic implementation namespace of `arith` namespace.

9.17 mln::binarization Namespace Reference

Namespace of "point-wise" expression tools.

Functions

- `template<typename I, typename F > mln::trait::ch_value< I, bool >::ret binarization (const Image< I > &input, const Function_v2b< F > &fun)`

Thresholds the values of `input` so that they can be stored in the output binary image.

- `template<typename I > mln::trait::ch_value< I, bool >::ret threshold (const Image< I > &input, const typename I::value threshold)`

Thresholds the values of `input` so that they can be stored in the output binary image.

9.17.1 Detailed Description

Namespace of "point-wise" expression tools.

9.17.2 Function Documentation

9.17.2.1 `template<typename I, typename F > mln::trait::ch_value< I, bool >::ret mln::binarization::binarization (const Image< I > & input, const Function_v2b< F > & fun) [inline]`

Thresholds the values of `input` so that they can be stored in the output binary image.

Parameters

[in] *input* The input image.

[in] *fun* The thresholding function, from value(I) to bool.

`for_all(p), output(p) = fun(p)`

Definition at line 86 of file `binarization.hh`.

Referenced by `threshold()`.

9.17.2.2 `template<typename I > mln::trait::ch_value< I, bool >::ret mln::binarization::threshold (const Image< I > & input, const typename I::value threshold) [inline]`

Thresholds the values of `input` so that they can be stored in the output binary image.

Parameters

[in] *input* The input image.

[in] *threshold* The threshold.

If `input(p)` is greater or equal than the threshold, the value in the output image in the same point will be `TRUE`, else `FALSE`.

Definition at line 61 of file `binarization/threshold.hh`.

References `binarization()`.

9.18 `mln::border` Namespace Reference

Namespace of routines related to image virtual (outer) border.

Namespaces

- namespace `impl`

Implementation namespace of border namespace.

Functions

- `template<typename I > void adjust (const Image< I > &ima, unsigned min_thickness)`
- `template<typename I > void duplicate (const Image< I > &ima)`
- `template<typename I, typename J > void equalize (const Image< I > &ima1, const Image< J > &ima2, unsigned min_thickness)`
- `template<typename I > void fill (const Image< I > &ima, const typename I::value &v)`
- `template<typename I > unsigned find (const Image< I > &ima)`
- `template<typename I > unsigned get (const Image< I > &ima)`
- `template<typename I > void mirror (const Image< I > &ima)`
- `template<typename I > void resize (const Image< I > &ima, unsigned thickness)`

Facade.

9.18.1 Detailed Description

Namespace of routines related to image virtual (outer) border.

9.18.2 Function Documentation

9.18.2.1 `template<typename I > void mln::border::adjust (const Image< I > & ima, unsigned min_thickness) [inline]`

Adjust the virtual (outer) border of image `ima` so that its size is at least `min_thickness`.

Parameters

- [in, out] *ima* The image whose border is to be adjusted.
[in] *min_thickness* The expected border minimum thickness.

Precondition

ima has to be initialized.

Warning

If the image border is already larger than *min_thickness*, this routine is a no-op.

Definition at line 62 of file border/adjust.hh.

References `get()`, and `resize()`.

9.18.2.2 `template<typename I> void mln::border::duplicate (const Image< I > & ima)`

Assign the virtual (outer) border of image *ima* with the duplicate of the inner border of this image.

Parameters

- [in, out] *ima* The image whose border is to be duplicated.

Precondition

ima has to be initialized.

Definition at line 252 of file border/duplicate.hh.

References `get()`.

9.18.2.3 `template<typename I, typename J> void mln::border::equalize (const Image< I > & ima1, const Image< J > & ima2, unsigned min_thickness) [inline]`

Equalize the virtual (outer) border of images *ima1* and *ima2* so that their size is equal and is at least *min_thickness*.

Parameters

- [in, out] *ima1* The first image whose border is to be equalized.
[in, out] *ima2* The second image whose border is to be equalized.
[in] *min_thickness* The expected border minimum thickness of both images.

Precondition

ima1 has to be initialized.
ima2 has to be initialized.

Warning

If both image borders already have the same thickness and if this thickness is larger than *min_thickness*, this routine is a no-op.

Definition at line 112 of file border/equalize.hh.

References `get()`.

9.18.2.4 `template<typename I> void mln::border::fill (const Image< I > & ima, const typename I::value & v) [inline]`

Fill the virtual (outer) border of image *ima* with the single value *v*.

Parameters

[in, out] *ima* The image whose border is to be filled.

[in] *v* The value to assign to all border pixels.

Precondition

ima has to be initialized.

Definition at line 201 of file border/fill.hh.

9.18.2.5 `template<typename I> unsigned mln::border::find (const Image< I > & ima) [inline]`

Find the virtual (outer) border thickness of image *ima*.

Parameters

[in] *ima* The image.

Returns

The border thickness (0 if there is no border).

Precondition

ima has to be initialized.

Definition at line 95 of file find.hh.

9.18.2.6 `template<typename I> unsigned mln::border::get (const Image< I > & ima) [inline]`

Get the virtual (outer) border thickness of image *ima*.

Parameters

[in] *ima* The image.

Returns

The border thickness (0 if there is no border).

Precondition

ima has to be initialized.

Definition at line 90 of file border/get.hh.

Referenced by `adjust()`, `duplicate()`, and `equalize()`.

9.18.2.7 `template<typename I> void mln::border::mirror (const Image< I > & ima) [inline]`

Mirror the virtual (outer) border of image `ima` with the (inner) level contents of this image.

Parameters

[in, out] *ima* The image whose border is to be mirrored.

Precondition

`ima` has to be initialized.

Definition at line 211 of file `border/mirror.hh`.

9.18.2.8 `template<typename I> void mln::border::resize (const Image< I > & ima, unsigned thickness) [inline]`

Facade.

Resize the virtual (outer) border of image `ima` to exactly `thickness`.

Parameters

[in, out] *ima* The image whose border is to be resized.

[in] *thickness* The expected border thickness.

Precondition

`ima` has to be initialized.

Warning

If the image border already has the expected thickness, this routine is a no-op.

Definition at line 126 of file `resize.hh`.

References `mln::primary()`, and `resize()`.

Referenced by `adjust()`, and `resize()`.

9.19 mln::border::impl Namespace Reference

Implementation namespace of border namespace.

Namespaces

- namespace [generic](#)
Generic implementation namespace of border namespace.

9.19.1 Detailed Description

Implementation namespace of border namespace.

9.20 mln::border::impl::generic Namespace Reference

Generic implementation namespace of border namespace.

9.20.1 Detailed Description

Generic implementation namespace of border namespace.

9.21 mln::canvas Namespace Reference

Namespace of canvas.

Namespaces

- namespace [browsing](#)
Namespace of browsing canvas.
- namespace [impl](#)
Implementation namespace of canvas namespace.
- namespace [labeling](#)
Namespace of labeling canvas.
- namespace [morpho](#)
Namespace of morphological canvas.

Classes

- struct [chamfer](#)
Compute chamfer distance.

Functions

- `template<typename I, typename N, typename W, typename D, typename F >
mln::trait::ch_value< I, D >::ret distance_front (const Image< I > &input, const Neighborhood< N > &nbh, const Weighted_Window< W > &w_win, D max, F &functor)
Canvas of discrete distance computation by thick front propagation.`
- `template<typename I, typename N, typename D, typename F >
mln::trait::ch_value< I, D >::ret distance_geodesic (const Image< I > &input, const Neighborhood< N > &nbh, D max, F &functor)
Discrete geodesic distance canvas.`

9.21.1 Detailed Description

Namespace of canvas.

9.21.2 Function Documentation

9.21.2.1 `template<typename I , typename N , typename W , typename D , typename F > mln::trait::ch_value< I, D >::ret mln::canvas::distance_front (const Image< I > & input, const Neighborhood< N > & nbh, const Weighted_Window< W > & w_win, D max, F & functor) [inline]`

Canvas of discrete distance computation by thick front propagation.

Definition at line 397 of file canvas/distance_front.hh.

Referenced by mln::transform::influence_zone_front().

9.21.2.2 `template<typename I , typename N , typename D , typename F > mln::trait::ch_value< I, D >::ret mln::canvas::distance_geodesic (const Image< I > & input, const Neighborhood< N > & nbh, D max, F & functor) [inline]`

Discrete geodesic distance canvas.

Definition at line 321 of file canvas/distance_geodesic.hh.

Referenced by mln::transform::influence_zone_geodesic_saturated().

9.22 mln::canvas::browsing Namespace Reference

Namespace of browsing canvas.

Classes

- struct [backdiagonal2d_t](#)
Browsing in a certain direction.
- struct [breadth_first_search_t](#)
Breadth-first search algorithm for graph, on vertices.
- struct [depth_first_search_t](#)
Breadth-first search algorithm for graph, on vertices.
- struct [diagonal2d_t](#)
Browsing in a certain direction.
- struct [dir_struct_elt_incr_update_t](#)
Browsing in a certain direction with a segment.
- struct [directional_t](#)
Browsing in a certain direction.

- struct [fwd_t](#)
Canvas for forward browsing.
- struct [hyper_directional_t](#)
Browsing in a certain direction.
- struct [snake_fwd_t](#)
Browsing in a snake-way, forward.
- struct [snake_generic_t](#)
Multidimensional Browsing in a given-way.
- struct [snake_vert_t](#)
Browsing in a snake-way, forward.

9.22.1 Detailed Description

Namespace of browsing canvas.

9.23 `mln::canvas::impl` Namespace Reference

Implementation namespace of canvas namespace.

9.23.1 Detailed Description

Implementation namespace of canvas namespace.

9.24 `mln::canvas::labeling` Namespace Reference

Namespace of labeling canvas.

Namespaces

- namespace [impl](#)
Implementation namespace of labeling canvas namespace.

Functions

- `template<typename I, typename N, typename L, typename F >
mln::trait::ch_value< I, L >::ret blobs (const Image< I > &input_, const Neighborhood< N >
&nbh_, L &nlabels, F &functor)`
Canvas for connected component labeling of the binary objects of a binary image using a queue-based algorithm.

9.24.1 Detailed Description

Namespace of labeling canvas.

9.24.2 Function Documentation

9.24.2.1 `template<typename I , typename N , typename L , typename F > mln::trait::ch_value< I, L >::ret mln::canvas::labeling::blobs (const Image< I > & input_, const Neighborhood< N > & nbh_, L & nlabels, F & functor) [inline]`

Canvas for connected component labeling of the binary objects of a binary image using a queue-based algorithm.

Parameters

[in] *input* The input image.

[in] *nbh* The connexity of the objects.

[out] *nlabels* The Number of labels. Its value is set in the algorithms.

[in, out] *functor* A functor computing data while labeling.

Returns

The label image.

Precondition

The input image has to be binary (checked at compile-time).

A fast queue is used so that the algorithm is not recursive and can handle large binary objects (blobs).

Definition at line 167 of file canvas/labeling/blobs.hh.

9.25 mln::canvas::labeling::impl Namespace Reference

Implementation namespace of labeling canvas namespace.

9.25.1 Detailed Description

Implementation namespace of labeling canvas namespace.

9.26 mln::canvas::morpho Namespace Reference

Namespace of morphological canvas.

9.26.1 Detailed Description

Namespace of morphological canvas.

9.27 mln::convert Namespace Reference

Namespace of conversion routines.

Functions

- `template<typename V >`
`void from_to (const float &from, Value< V > &to)`
Conversion of a float `from` towards a value `to`.
- `template<typename V >`
`void from_to (const int &from, Value< V > &to)`
Conversion of a int `from` towards a value `to`.
- `template<typename V >`
`void from_to (const double &from, Value< V > &to)`
Conversion of a double `from` towards a value `to`.
- `template<typename V >`
`void from_to (const unsigned &from, Value< V > &to)`
Conversion of an unsigned `from` towards a value `to`.
- `template<typename S >`
`mln_image_from_grid (typename S::site::grid, bool) to_image(const Site_Set< S > &pset)`
Convert a point set `pset` into a binary image.
- `template<typename W >`
`mln_image_from_grid (typename W::site::grid, bool) to_image(const Window< W > &win)`
Convert a window `win` into a binary image.
- `template<typename W >`
`mln_image_from_grid (typename W::site::grid, mln_weight(W)) to_image(const Weighted_Window< W > &w_win)`
Convert a weighted window `w_win` into an image.
- `template<typename N >`
`mln_image_from_grid (typename N::site::grid, bool) to_image(const Neighborhood< N > &nbh)`
Convert a neighborhood `nbh` into a binary image.
- `template<typename N >`
`mln_window (N) to_window(const Neighborhood< N > &nbh)`
Convert a neighborhood `nbh` into a window.
- `template<typename T, typename O >`
`T to (const O &from)`
Conversion of the object `from` towards an object with type `T`.
- `template<typename P >`
`P::dpoint to_dpoint (const Point_Site< P > &p)`
Convert a point site `p` into a delta-point.

- `template<typename I >`
`pw::value_< I > to_fun (const Image< I > &ima)`
Convert an image into a function.
- `template<typename T >`
`imageId< unsigned > to_image (const histo::array< T > &h)`
Convert an histo h into an imageId<unsigned>.
- `template<typename I >`
`p_array< typename I::psite > to_p_array (const Image< I > &img)`
Convert an image img into a p_array.
- `template<typename S >`
`p_array< typename S::psite > to_p_array (const Site_Set< S > &pset)`
Convert a point set pset into a p_array (point set vector).
- `template<typename W >`
`p_array< typename W::psite > to_p_array (const Window< W > &win, const typename W::psite &p)`
Convert a window win centered at point p into a p_array (point set vector).
- `template<typename N >`
`p_set< typename N::psite > to_p_set (const Neighborhood< N > &nbh)`
Convert a neighborhood nbh into a site set.
- `template<typename I >`
`p_set< typename I::psite > to_p_set (const Image< I > &ima)`
Convert a binary image ima into a site set.
- `template<typename P, typename C >`
`p_set< P > to_p_set (const std::set< P, C > &s)`
Convert an std::set s of sites into a site set.
- `template<typename S >`
`p_set< typename S::psite > to_p_set (const Site_Set< S > &ps)`
Convert any site set ps into a 'mlnp_set' site set.
- `template<typename W >`
`p_set< typename W::psite > to_p_set (const Window< W > &win)`
Convert a Window win into a site set.
- `template<typename I >`
`QImage to_qimage (const Image< I > &ima)`
Convert a Milena image to a QImage.
- `template<typename N >`
`window< typename N::dpoint > to_upper_window (const Neighborhood< N > &nbh)`
Convert a neighborhood nbh into an upper window.

- `template<typename W >`
`window< typename W::dpsite > to_upper_window (const Window< W > &win)`
Convert a window `nbh` into an upper window.
- `template<typename D , typename C >`
`window< D > to_window (const std::set< D, C > &s)`
Convert an `std::set` `s` of delta-sites into a window.
- `template<typename I >`
`window< typename I::site::dpsite > to_window (const Image< I > &ima)`
Convert a binary image `ima` into a window.
- `template<typename S >`
`window< typename S::site::dpsite > to_window (const Site_Set< S > &pset)`
Convert a site set `pset` into a window.

Variables

- `fun::C< R(*) (A)> to_fun (R(*f)(A))`
Convert a C unary function into an `mln::fun::C`.

9.27.1 Detailed Description

Namespace of conversion routines.

9.27.2 Function Documentation

9.27.2.1 `template<typename V > void mln::convert::from_to (const float & from, Value< V > & to)`

Conversion of a float `from` towards a value `to`.

9.27.2.2 `template<typename V > void mln::convert::from_to (const int & from, Value< V > & to)`

Conversion of a int `from` towards a value `to`.

9.27.2.3 `template<typename V > void mln::convert::from_to (const double & from, Value< V > & to)`

Conversion of a double `from` towards a value `to`.

9.27.2.4 `template<typename V > void mln::convert::from_to (const unsigned & from, Value< V > & to)`

Conversion of an unsigned `from` towards a value `to`.

9.27.2.5 `template<typename S > mln::convert::mln_image_from_grid (typename S::site::grid , bool) const [inline]`

Convert a point set `pset` into a binary image.

Width of the converted image will be `pset.bbox + 2 * border`.

9.27.2.6 `template<typename W > mln::convert::mln_image_from_grid (typename W::site::grid , bool) const`

Convert a window `win` into a binary image.

9.27.2.7 `template<typename W > mln::convert::mln_image_from_grid (typename W::site::grid , mln_weight(W)) const`

Convert a weighted window `w_win` into an image.

9.27.2.8 `template<typename N > mln::convert::mln_image_from_grid (typename N::site::grid , bool) const`

Convert a neighborhood `nbh` into a binary image.

9.27.2.9 `template<typename N > mln::convert::mln_window (N) const [inline]`

Convert a neighborhood `nbh` into a window.

Definition at line 74 of file `to_window.hh`.

9.27.2.10 `template<typename T , typename O > T mln::convert::to (const O & from) [inline]`

Conversion of the object `from` towards an object with type `T`.

Definition at line 63 of file `to.hh`.

References `mln::mln_exact()`.

Referenced by `mln::make_debug_graph_image()`.

9.27.2.11 `template<typename P > P::dpoint mln::convert::to_dpoint (const Point_Site< P > & p) [inline]`

Convert a point site `p` into a delta-point.

Definition at line 52 of file `to_dpoint.hh`.

9.27.2.12 `template<typename I > pw::value_<I> mln::convert::to_fun (const Image< I > & ima)`

Convert an image into a function.

Definition at line 64 of file `to_fun.hh`.

9.27.2.13 `template<typename T > image1d<unsigned> mln::convert::to_image (const histo::array< T > & h)`

Convert an `histo h` into an `image1d<unsigned>`.

9.27.2.14 `template<typename I > p_array< typename I::psite > mln::convert::to_p_array (const Image< I > & img) [inline]`

Convert an image `img` into a `p_array`.

Definition at line 98 of file `to_p_array.hh`.

References `mln::p_array< P >::append()`.

9.27.2.15 `template<typename S > p_array< typename S::psite > mln::convert::to_p_array (const Site_Set< S > & pset) [inline]`

Convert a point set `pset` into a `p_array` (point set vector).

Definition at line 68 of file `to_p_array.hh`.

References `mln::p_array< P >::append()`.

9.27.2.16 `template<typename W > p_array< typename W::psite > mln::convert::to_p_array (const Window< W > & win, const typename W::psite & p) [inline]`

Convert a window `win` centered at point `p` into a `p_array` (point set vector).

Definition at line 82 of file `to_p_array.hh`.

References `mln::p_array< P >::append()`, and `mln::p_array< P >::reserve()`.

9.27.2.17 `template<typename N > p_set< typename N::psite > mln::convert::to_p_set (const Neighborhood< N > & nbh) [inline]`

Convert a neighborhood `nbh` into a site set.

Definition at line 83 of file `to_p_set.hh`.

References `mln::p_set< P >::insert()`.

9.27.2.18 `template<typename I > p_set< typename I::psite > mln::convert::to_p_set (const Image< I > & ima) [inline]`

Convert a binary image `ima` into a site set.

Definition at line 97 of file `to_p_set.hh`.

References `mln::p_set< P >::insert()`.

9.27.2.19 `template<typename P, typename C > p_set< P > mln::convert::to_p_set (const std::set< P, C > & s) [inline]`

Convert an `std::set s` of sites into a site set.

C is the comparison functor.

Definition at line 130 of file to_p_set.hh.

References mln::p_set< P >::insert().

9.27.2.20 `template<typename S > p_set< typename S::psite > mln::convert::to_p_set (const Site_Set< S > & ps) [inline]`

Convert any site set `ps` into a 'mlnp_set' site set.

Definition at line 144 of file to_p_set.hh.

References mln::p_set< P >::insert().

9.27.2.21 `template<typename W > p_set< typename W::psite > mln::convert::to_p_set (const Window< W > & win) [inline]`

Convert a `Window` `win` into a site set.

Definition at line 117 of file to_p_set.hh.

References mln::p_set< P >::insert().

9.27.2.22 `template<typename I > QImage mln::convert::to_qimage (const Image< I > & ima) [inline]`

Convert a Milena image to a QImage.

Definition at line 279 of file to_qimage.hh.

9.27.2.23 `template<typename N > window< typename N::dpoint > mln::convert::to_upper_window (const Neighborhood< N > & nbh) [inline]`

Convert a neighborhood `nbh` into an upper window.

Definition at line 82 of file to_upper_window.hh.

References mln::window< D >::insert().

9.27.2.24 `template<typename W > window< typename W::dpsite > mln::convert::to_upper_window (const Window< W > & win) [inline]`

Convert a window `nbh` into an upper window.

Definition at line 67 of file to_upper_window.hh.

References mln::window< D >::insert().

9.27.2.25 `template<typename D, typename C > window< D > mln::convert::to_window (const std::set< D, C > & s) [inline]`

Convert an `std::set` `s` of delta-sites into a window.

Definition at line 128 of file to_window.hh.

References `mln::window< D >::insert()`.

9.27.2.26 `template<typename I > window< typename I::site::dpsite > mln::convert::to_window (const Image< I > & ima) [inline]`

Convert a binary image `ima` into a window.

Definition at line 94 of file `to_window.hh`.

References `mln::window< D >::insert()`.

Referenced by `to_window()`.

9.27.2.27 `template<typename S > window< typename S::site::dpsite > mln::convert::to_window (const Site_Set< S > & pset) [inline]`

Convert a site set `pset` into a window.

Definition at line 117 of file `to_window.hh`.

References `to_window()`.

9.27.3 Variable Documentation

9.27.3.1 `pw::value_< I > mln::convert::to_fun [inline]`

Convert a C unary function into an `mln::fun::C`.

Definition at line 45 of file `to_fun.hh`.

9.28 mln::data Namespace Reference

Namespace of image processing routines related to pixel data.

Namespaces

- namespace [approx](#)
Namespace of image processing routines related to pixel levels with approximation.
- namespace [impl](#)
Implementation namespace of data namespace.
- namespace [naive](#)
Namespace of image processing routines related to pixel levels with naive approach.

Functions

- `template<typename I, typename O > void abs (const Image< I > &input, Image< O > &output)`

- template<typename I >
void [abs_inplace](#) (Image< I > &input)
- template<typename I, typename F >
void [apply](#) (Image< I > &input, const [Function_v2v](#)< F > &f)
- template<typename A, typename I >
A::result [compute](#) (const [Accumulator](#)< A > &a, const Image< I > &input)
Compute an accumulator onto the pixel values of the image input.
- template<typename A, typename I >
A::result [compute](#) ([Accumulator](#)< A > &a, const Image< I > &input)
Compute an accumulator onto the pixel values of the image input.
- template<typename V, typename I >
mln::trait::ch_value< I, V >::ret [convert](#) (const V &v, const Image< I > &input)
Convert the image input by changing the value type.
- template<typename I, typename W, typename O >
void [fast_median](#) (const Image< I > &input, const [Window](#)< W > &win, Image< O > &output)
- template<typename I, typename D >
void [fill](#) (Image< I > &ima, const D &data)
- template<typename I, typename J >
void [fill_with_image](#) (Image< I > &ima, const Image< J > &data)
Fill the image ima with the values of the image data.
- template<typename I, typename W >
mln::trait::concrete< I >::ret [median](#) (const Image< I > &input, const [Window](#)< W > &win)
- template<typename A, typename I >
[mln_meta_accu_result](#) (A, typename I::value) compute(const [Meta_Accumulator](#)< A > &a
Compute an accumulator onto the pixel values of the image input.
- template<typename I, typename J >
void [paste](#) (const Image< I > &input, Image< J > &output)
Paste the contents of image input into the image output.
- template<typename I, typename J >
void [paste_without_localization](#) (const Image< I > &input, Image< J > &output)
Paste the contents of image input into the image output without taking into account the localization of sites.
- template<typename I >
void [replace](#) (Image< I > &input, const typename I::value &old_value, const typename I::value &new_value)
- template<typename I, typename V >
mln::trait::ch_value< I, V >::ret [saturate](#) (const Image< I > &input, const V &min, const V &max)
- template<typename V, typename I >
mln::trait::ch_value< I, V >::ret [saturate](#) (V v, const Image< I > &input)
- template<typename I >
void [saturate_inplace](#) (Image< I > &input, const typename I::value &min, const typename I::value &max)
- template<typename I >
[util::array](#)< unsigned > [sort_offsets_increasing](#) (const Image< I > &input)

Sort pixel offsets of the image input wrt increasing pixel values.

- `template<typename I >`
`p_array< typename I::psite > sort_psites_decreasing (const Image< I > &input)`
- `template<typename I >`
`p_array< typename I::psite > sort_psites_increasing (const Image< I > &input)`
- `template<typename V , typename I >`
`mln::trait::ch_value< I, V >::ret stretch (const V &v, const Image< I > &input)`
- `template<typename I , typename O >`
`void to_enc (const Image< I > &input, Image< O > &output)`
- `template<typename I1 , typename I2 , typename F >`
`mln::trait::ch_value< I1, typename F::result >::ret transform (const Image< I1 > &input1, const Image< I2 > &input2, const Function_vv2v< F > &f)`
- `template<typename I , typename F >`
`mln::trait::ch_value< I, typename F::result >::ret transform (const Image< I > &input, const Function_v2v< F > &f)`
- `template<typename I1 , typename I2 , typename F >`
`void transform_inplace (Image< I1 > &ima, const Image< I2 > &aux, const Function_vv2v< F > &f)`
- `template<typename I , typename F >`
`void transform_inplace (Image< I > &ima, const Function_v2v< F > &f)`
- `template<typename A , typename I >`
`A::result update (Accumulator< A > &a, const Image< I > &input)`
- `template<typename V , typename I >`
`mln::trait::ch_value< I, V >::ret wrap (const V &v, const Image< I > &input)`
Routine to wrap values such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

- `template<typename I , typename V >`
`void fill_with_value (Image< I > &ima, const V &val)`
Fill the whole image ima with the single value v.

9.28.1 Detailed Description

Namespace of image processing routines related to pixel data.

9.28.2 Function Documentation

9.28.2.1 `template<typename I , typename O > void mln::data::abs (const Image< I > & input, Image< O > & output) [inline]`

Apply the absolute value (abs) function to image pixel values.

Parameters

- [in] *input* The input image.
- [out] *output* The output image.

Definition at line 68 of file data/abs.hh.

References transform().

9.28.2.2 `template<typename I > void mln::data::abs_inplace (Image< I > & input)` `[inline]`

Apply the absolute value (abs) function to image pixel values.

Parameters

[in, out] *input* The input image.

Definition at line 80 of file data/abs.hh.

References `apply()`.

9.28.2.3 `template<typename I, typename F > void mln::data::apply (Image< I > & input,` `const Function_v2v< F > & f) [inline]`

Apply a function-object to the image `input`.

Parameters

[in, out] *input* The input image.

[in] *f* The function-object.

This routine runs:

for all `p` of `input`, `input (p) = f(input (p))`

This routine is equivalent to `data::transform(input, f, input)` but it is faster since a single iterator is required.

Definition at line 95 of file `apply.hh`.

Referenced by `abs_inplace()`, and `saturate_inplace()`.

9.28.2.4 `template<typename A, typename I > A::result mln::data::compute (const` `Accumulator< A > & a, const Image< I > & input) [inline]`

Compute an accumulator onto the pixel values of the image `input`.

Be ware that the given accumulator won't be modified and won't store any result.

Parameters

[in] *a* An accumulator.

[in] *input* The input image.

Returns

The accumulator result.

It fully relies on [data::update](#).

Definition at line 93 of file `data/compute.hh`.

Referenced by `mln::labeled_image< I >::labeled_image()`, `mln::estim::mean()`, `mln::estim::min_max()`, and `mln::estim::sum()`.

9.28.2.5 `template<typename A , typename I > A::result mln::data::compute (Accumulator< A > & a, const Image< I > & input) [inline]`

Compute an accumulator onto the pixel values of the image `input`.

Parameters

[in, out] *a* An accumulator.

[in] *input* The input image.

Returns

The accumulator result.

It fully relies on [data::update](#).

Definition at line 104 of file `data/compute.hh`.

9.28.2.6 `template<typename V , typename I > mln::trait::ch_value< I, V >::ret mln::data::convert (const V & v, const Image< I > & input) [inline]`

Convert the image `input` by changing the value type.

Parameters

[in] *v* A value of the destination type.

[in] *input* The input image.

Definition at line 154 of file `mln/data/convert.hh`.

Referenced by `mln::morpho::watershed::superpose()`, and `mln::debug::superpose()`.

9.28.2.7 `template<typename I , typename W , typename O > void mln::data::fast_median (const Image< I > & input, const Window< W > & win, Image< O > & output) [inline]`

Compute in `output` the median filter of image `input` by the window `win`.

Parameters

[in] *input* The image to be filtered.

[in] *win* The window.

[in, out] *output* The output image.

Precondition

`input` and `output` have to be initialized.

Definition at line 167 of file `fast_median.hh`.

9.28.2.8 `template<typename I, typename D > void mln::data::fill (Image< I > & ima, const D & data) [inline]`

Fill the whole image `ima` with the data provided by `aux`.

Parameters

- `[in, out]` *ima* The image to be filled.
- `[in]` *data* The auxiliary data to fill the image `ima`.

Precondition

`ima` has to be initialized.

Definition at line 138 of file `data/fill.hh`.

Referenced by `mln::draw::box_plain()`, `mln::draw::dashed_line()`, `mln::topo::detach()`, `mln::util::display_branch()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::duplicate()`, `mln::make::edge_image()`, `mln::morpho::tree::filter::filter()`, `mln::transform::hough()`, `mln::registration::icp()`, `mln::accu::stat::histo3d_rgb< V >::init()`, `mln::graph::labeling()`, `mln::morpho::laplacian()`, `mln::make_debug_graph_image()`, `mln::morpho::tree::filter::max()`, `mln::geom::mesh_corner_point_area()`, `mln::geom::mesh_normal()`, `mln::morpho::meyer_wst()`, `mln::morpho::tree::filter::min()`, `mln::debug::mosaic()`, `mln::debug::slices_2d()`, `mln::morpho::watershed::superpose()`, `mln::debug::superpose()`, `mln::morpho::watershed::topological()`, and `mln::geom::translate()`.

9.28.2.9 `template<typename I, typename J > void mln::data::fill_with_image (Image< I > & ima, const Image< J > & data) [inline]`

Fill the image `ima` with the values of the image `data`.

Parameters

- `[in, out]` *ima* The image to be filled.
- `[in]` *data* The image.

Warning

The definition domain of `ima` has to be included in the one of `data`.

Precondition

`ima.domain <= data.domain`.

Definition at line 124 of file `fill_with_image.hh`.

9.28.2.10 `template<typename I, typename V > void mln::data::fill_with_value (Image< I > & ima, const V & val) [inline]`

Fill the whole image `ima` with the single value `v`.

Parameters

- `[in, out]` *ima* The image to be filled.
- `[in]` *val* The value to assign to all sites.

Precondition

`ima` has to be initialized.

Definition at line 130 of file `fill_with_value.hh`.

Referenced by `mln::p_image< I >::clear()`.

9.28.2.11 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::data::median (const Image< I > & input, const Window< W > & win)`

Compute in `output` the median filter of image `input` by the window `win`.

Parameters

[in] *input* The image to be filtered.

[in] *win* The window.

Precondition

`input` have to be initialized.

Definition at line 270 of file `median.hh`.

9.28.2.12 `template<typename A , typename I > mln::data::mln_meta_accu_result (A , typename I::value) const [inline]`

Compute an accumulator onto the pixel values of the image `input`.

Parameters

[in] *a* A meta-accumulator.

[in] *input* The input image.

Returns

The accumulator result.

9.28.2.13 `template<typename I , typename J > void mln::data::paste (const Image< I > & input, Image< J > & output) [inline]`

Paste the contents of image `input` into the image `output`.

Parameters

[in] *input* The input image providing pixels values.

[in, out] *output* The image in which values are assigned.

This routine runs:

for all `p` of `input`, `output (p) = input (p)`.

Warning

The definition domain of `input` has to be included in the one of `output`; so using [mln::safe_image](#) does not make pasting outside the output domain work.

Precondition

```
input.domain <= output.domain
```

Definition at line 137 of file `paste.hh`.

Referenced by `mln::make::image3d()`, `mln::draw::line()`, `mln::debug::mosaic()`, `mln::geom::rotate()`, `mln::debug::slices_2d()`, and `mln::labeling::superpose()`.

9.28.2.14 `template<typename I, typename J > void mln::data::paste_without_localization (const Image< I > & input, Image< J > & output) [inline]`

Paste the contents of image `input` into the image `output` without taking into account the localization of sites.

Parameters

[in] *input* The input image providing pixels values.

[in, out] *output* The image in which values are assigned.

Definition at line 349 of file `paste_without_localization.hh`.

9.28.2.15 `template<typename I > void mln::data::replace (Image< I > & input, const typename I::value & old_value, const typename I::value & new_value)`

Replace `old_value` by `new_value` in the image `input`

Parameters

[in] *input* The input image.

[in] *old_value* The value to be replaced...

[in] *new_value* ...by this one.

Definition at line 88 of file `replace.hh`.

9.28.2.16 `template<typename V, typename I > mln::trait::ch_value< I, V >::ret mln::data::saturate (V v, const Image< I > & input) [inline]`

Apply the saturate function to image pixel values.

Parameters

[in] *v* A value of the output type.

[in] *input* The input image.

The saturation is based on the min and max values of the output value type. This assumes that the range of values in the input image is larger than the one of the output image.

Definition at line 87 of file `data/saturate.hh`.

References `transform()`.

9.28.2.17 `template<typename I, typename V> mln::trait::ch_value< I, V >::ret
mln::data::saturate (const Image< I > & input, const V & min, const V & max)
[inline]`

Apply the saturate function to image pixel values.

Parameters

- [in] *input* The input image.
- [in] *min* The minimum output value.
- [in] *max* The maximum output value.

Definition at line 103 of file data/saturate.hh.

References transform().

9.28.2.18 `template<typename I> void mln::data::saturate_inplace (Image< I > & input, const
typename I::value & min, const typename I::value & max) [inline]`

Apply the saturate function to image pixel values.

Parameters

- [in, out] *input* The input image.
- [in] *min* The minimum output value.
- [in] *max* The maximum output value

Definition at line 119 of file data/saturate.hh.

References apply().

9.28.2.19 `template<typename I> util::array< unsigned > mln::data::sort_offsets_increasing (
const Image< I > & input) [inline]`

Sort pixel offsets of the image *input* wrt increasing pixel values.

Definition at line 298 of file sort_offsets.hh.

9.28.2.20 `template<typename I> p_array< typename I::psite > mln::data::sort_psites_
decreasing (const Image< I > & input) [inline]`

Sort psites the image *input* through a function *f* to set the *output* image in decreasing way.

Parameters

- [in] *input* The input image.

Precondition

`input.is_valid`

Definition at line 229 of file sort_psites.hh.

Referenced by mln::morpho::tree::min_tree().

9.28.2.21 `template<typename I > p_array< typename I::psite > mln::data::sort_sites_increasing (const Image< I > & input) [inline]`

Sort psites the image `input` through a function `f` to set the `output` image in increasing way.

Parameters

[in] *input* The input image.

Precondition

`input.is_valid`

Definition at line 219 of file `sort_psites.hh`.

Referenced by `mln::morpho::tree::max_tree()`.

9.28.2.22 `template<typename V , typename I > mln::trait::ch_value< I, V >::ret mln::data::stretch (const V & v, const Image< I > & input) [inline]`

Stretch the values of `input` so that they can be stored in `output`.

Parameters

[in] *v* A value to set the output value type.

[in] *input* The input image.

Returns

A stretch image with values of the same type as *v*.

Precondition

`input.is_valid`

Definition at line 128 of file `stretch.hh`.

References `mln::data::impl::stretch()`.

9.28.2.23 `template<typename I , typename O > void mln::data::to_enc (const Image< I > & input, Image< O > & output) [inline]`

Set the `output` image with the encoding values of the image `input` pixels.

Parameters

[in] *input* The input image.

[out] *output* The result image.

Precondition

`output.domain >= input.domain`

Definition at line 60 of file `data/to_enc.hh`.

References `transform()`.

9.28.2.24 `template<typename I , typename F > mln::trait::ch_value< I , typename F::result >::ret mln::data::transform (const Image< I > & input, const Function_v2v< F > & f) [inline]`

Transform the image *input* through a function *f*.

Parameters

[in] *input* The input image.

[in] *f* The function.

This routine runs:

for all *p* of *input*, $output(p) = f(input(p))$.

Definition at line 202 of file `data/transform.hh`.

Referenced by `abs()`, `mln::logical::and_not()`, `mln::labeling::colorize()`, `mln::arith::diff_abs()`, `mln::labeling::fill_holes()`, `mln::linear::mln_ch_convolve()`, `mln::linear::mln_ch_convolve_grad()`, `mln::labeling::pack()`, `mln::labeling::pack_inplace()`, `mln::labeling::relabel()`, `saturate()`, `mln::data::impl::stretch()`, `to_enc()`, `mln::labeling::wrap()`, and `wrap()`.

9.28.2.25 `template<typename I1 , typename I2 , typename F > mln::trait::ch_value< I1 , typename F::result >::ret mln::data::transform (const Image< I1 > & input1, const Image< I2 > & input2, const Function_vv2v< F > & f) [inline]`

Transform two images *input1* *input2* through a function *f*.

Parameters

[in] *input1* The 1st input image.

[in] *input2* The 2nd input image.

[in] *f* The function.

This routine runs:

for all *p* of *input*, $output(p) = f(input1(p), input2(p))$.

Definition at line 219 of file `data/transform.hh`.

9.28.2.26 `template<typename I1 , typename I2 , typename F > void mln::data::transform_inplace (Image< I1 > & ima, const Image< I2 > & aux, const Function_vv2v< F > & f)`

Transform inplace the image *ima* with the image *aux* through a function *f*.

Parameters

[in] *ima* The image to be transformed.

[in] *aux* The auxiliary image.

[in] *f* The function.

This routine runs:

for all *p* of *ima*, $ima(p) = f(ima(p), aux(p))$.

Definition at line 502 of file `transform_inplace.hh`.

9.28.2.27 `template<typename I , typename F > void mln::data::transform_inplace (Image< I > & ima, const Function_v2v< F > & f)`

Transform inplace the image *ima* through a function *f*.

Parameters

[in, out] *ima* The image to be transformed.

[in] *f* The function.

This routine runs:

for all *p* of *ima*, $ima(p) = f(ima(p))$.

Definition at line 490 of file transform_inplace.hh.

Referenced by `mln::logical::and_inplace()`, `mln::logical::and_not_inplace()`, `mln::logical::not_inplace()`, `mln::logical::or_inplace()`, `mln::labeling::relabel_inplace()`, and `mln::logical::xor_inplace()`.

9.28.2.28 `template<typename A , typename I > A::result mln::data::update (Accumulator< A > & a, const Image< I > & input) [inline]`

Update an accumulator with the pixel values of the image *input*.

Parameters

[in] *a* The accumulator.

[in] *input* The input image.

Returns

The accumulator result.

Definition at line 191 of file update.hh.

9.28.2.29 `template<typename V , typename I > mln::trait::ch_value< I, V >::ret mln::data::wrap (const V & v, const Image< I > & input)`

Routine to wrap values such as $0 \rightarrow 0$ and $[1, lmax]$ maps to $[1, Lmax]$ (using modulus).

Parameters

[in] *v* The target value type.

[in] *input* Input image.

Returns

An image with wrapped values.

Definition at line 65 of file data/wrap.hh.

References `transform()`.

9.29 mln::data::approx Namespace Reference

Namespace of image processing routines related to pixel levels with approximation.

Namespaces

- namespace [impl](#)
Implementation namespace of [data::approx](#) namespace.

Functions

- `template<typename I > mln::trait::concrete< I >::ret median (const Image< I > &input, const win::rectangle2d &win)`
- `template<typename I > mln::trait::concrete< I >::ret median (const Image< I > &input, const win::octagon2d &win)`
- `template<typename I > mln::trait::concrete< I >::ret median (const Image< I > &input, const win::disk2d &win)`

9.29.1 Detailed Description

Namespace of image processing routines related to pixel levels with approximation.

9.29.2 Function Documentation

9.29.2.1 `template<typename I > mln::trait::concrete< I >::ret mln::data::approx::median (const Image< I > & input, const win::rectangle2d & win) [inline]`

Compute in `output` an approximate of the median filter of image `input` by the 2D rectangle `win`.

Parameters

- [in] *input* The image to be filtered.
[in] *win* The rectangle.

The approximation is based on a vertical median ran after an horizontal median.

Precondition

`input` and `output` have to be initialized.

Definition at line 112 of file `approx/median.hh`.

Referenced by `median()`.

9.29.2.2 `template<typename I > mln::trait::concrete< I >::ret mln::data::approx::median (const Image< I > & input, const win::octagon2d & win) [inline]`

Compute in `output` an approximate of the median filter of image `input` by the 2D octagon `win`.

Parameters

- [in] *input* The image to be filtered.
- [in] *win* The octagon.

The approximation is based on a vertical median and an horizontal median an two diagonal median.

Precondition

`input` and `output` have to be initialized.

Definition at line 159 of file `approx/median.hh`.

References `median()`.

9.29.2.3 `template<typename I > mln::trait::concrete< I >::ret mln::data::approx::median (const Image< I > & input, const win::disk2d & win) [inline]`

Compute in `output` an approximate of the median filter of image `input` by the 2D disk `win`.

Parameters

- [in] *input* The image to be filtered.
- [in] *win* The disk.

The approximation is based on a vertical median and an horizontal median an two diagonal median.

Precondition

`input` and `output` have to be initialized.

Definition at line 132 of file `approx/median.hh`.

References `median()`.

9.30 `mln::data::approx::impl` Namespace Reference

Implementation namespace of [data::approx](#) namespace.

9.30.1 Detailed Description

Implementation namespace of [data::approx](#) namespace.

9.31 `mln::data::impl` Namespace Reference

Implementation namespace of `data` namespace.

Namespaces

- namespace [generic](#)
Generic implementation namespace of data namespace.

Functions

- `template<typename I , typename J >`
`void paste_without_localization_fast (const Image< I > &input_, Image< J > &output_)`
Paste data to an image without using localization. Performs a point-wise copy.
- `template<typename I , typename J >`
`void paste_without_localization_fastest (const Image< I > &input_, Image< J > &output_)`
Paste data to an image without using localization. Performs a one-block memory copy.
- `template<typename I , typename J >`
`void paste_without_localization_lines (const Image< I > &input_, Image< J > &output_)`
Paste data to an image without using localization. Performs a line-per-line memory copy.
- `template<typename V , typename I >`
`mln::trait::ch_value< I, V >::ret stretch (const V &v, const Image< I > &input)`
Generic implementation of `data::stretch`.
- `template<typename I , typename F >`
`void transform_inplace_lowq (Image< I > &input_, const Function_v2v< F > &f_)`
Specialized implementation.
- `template<typename A , typename I >`
`A::result update_fastest (Accumulator< A > &a_, const Image< I > &input_)`
Fastest implementation of `data::update`.

9.31.1 Detailed Description

Implementation namespace of data namespace.

9.31.2 Function Documentation

9.31.2.1 `template<typename I , typename J > void mln::data::impl::paste_without_localization_fast (const Image< I > & input_, Image< J > & output_) [inline]`

Paste data to an image without using localization. Performs a point-wise copy.

`input` and `output` must have both the following properties:

- `mln::trait::image::value_alignment::with_grid`
- `mln::trait::image::value_storage::one_block`
- `mln::trait::image::value_access::direct`
- `mln::trait::image::ext_domain::some`

They must also fulfill the following conditions:

- Same domain size.

Definition at line 220 of file `paste_without_localization.hh`.

9.31.2.2 `template<typename I , typename J > void mln::data::impl::paste_without_localization_fastest (const Image< I > & input_ , Image< J > & output_) [inline]`

Paste data to an image without using localization. Performs a one-block memory copy.

`input` and `output` must have both the following properties:

- `mln::trait::image::value_alignment::with_grid`
- `mln::trait::image::value_storage::one_block`
- `mln::trait::image::value_access::direct`
- `mln::trait::image::ext_domain::some`

They must also fulfill the following conditions:

- Same border size.
- Same domain size.
- Same value type.

Definition at line 142 of file `paste_without_localization.hh`.

9.31.2.3 `template<typename I , typename J > void mln::data::impl::paste_without_localization_lines (const Image< I > & input_ , Image< J > & output_) [inline]`

Paste data to an image without using localization. Performs a line-per-line memory copy.

`input` and `output` must have both the following properties:

- `mln::trait::image::value_alignment::with_grid`
- `mln::trait::image::value_storage::one_block`
- `mln::trait::image::value_access::direct`
- `mln::trait::image::ext_domain::some`

They must also fulfill the following conditions:

- Same domain size.
- Same value type.

Definition at line 179 of file `paste_without_localization.hh`.

9.31.2.4 `template<typename V , typename I > mln::trait::ch_value< I , V >::ret
mln::data::impl::stretch (const V & v, const Image< I > & input) [inline]`

Generic implementation of [data::stretch](#).

Parameters

[in] *v* A value to set the output value type.

[in] *input* The input image.

Returns

A stretch image with values of the same type as *v*.

Definition at line 83 of file stretch.hh.

References `mln::initialize()`, `mln::estim::min_max()`, and `mln::data::transform()`.

Referenced by `mln::data::stretch()`.

9.31.2.5 `template<typename I , typename F > void mln::data::impl::transform_inplace_lowq (
Image< I > & input_, const Function_v2v< F > & f_)`

Specialized implementation.

Definition at line 203 of file transform_inplace.hh.

9.31.2.6 `template<typename A , typename I > A ::result mln::data::impl::update_fastest (
Accumulator< A > & a_, const Image< I > & input_) [inline]`

Fastest implementation of [data::update](#).

Parameters

[in] *a_* The accumulator.

[in] *input_* The input image.

Returns

The accumulator result.

Definition at line 129 of file update.hh.

9.32 mln::data::impl::generic Namespace Reference

Generic implementation namespace of data namespace.

Functions

- `template<typename I , typename J >
void fill_with_image (Image< I > &ima_, const Image< J > &data_)`
Generic implementation.

- `template<typename I, typename V >`
`void fill_with_value (Image< I > &ima_, const V &val)`
Fill the whole image `ima_` with the single value `v`.
- `template<typename I, typename J >`
`void paste (const Image< I > &input_, Image< J > &output_)`
Generic implementation of `data::paste`.
- `template<typename I, typename F >`
`mln::trait::ch_value< I, typename F::result >::ret transform (const Image< I > &input_, const Function_v2v< F > &f_)`
Generic implementation of `data::transform`.
- `template<typename I1, typename I2, typename F >`
`mln::trait::ch_value< I1, typename F::result >::ret transform (const Image< I1 > &input1_, const Image< I2 > &input2_, const Function_vv2v< F > &f_)`
Generic implementation of `data::transform`.
- `template<typename I1, typename I2, typename F >`
`void transform_inplace (Image< I1 > &ima_, const Image< I2 > &aux_, const Function_vv2v< F > &f_)`
Generic implementation of `transform_inplace`.
- `template<typename I, typename F >`
`void transform_inplace (Image< I > &ima_, const Function_v2v< F > &f_)`
Generic implementation of `transform_inplace`.
- `template<typename A, typename I >`
`A::result update (Accumulator< A > &a_, const Image< I > &input_)`
Generic implementation of `data::update`.

9.32.1 Detailed Description

Generic implementation namespace of data namespace.

9.32.2 Function Documentation

9.32.2.1 `template<typename I, typename J > void mln::data::impl::generic::fill_with_image (Image< I > & ima_, const Image< J > & data_)`

Generic implementation.

Parameters

- [in, out] `ima_` The image to be filled.
- [in] `data_` The image.

Definition at line 100 of file `fill_with_image.hh`.

9.32.2.2 `template<typename I , typename V > void mln::data::impl::generic::fill_with_value (Image< I > & ima_ , const V & val)`

Fill the whole image `ima` with the single value `v`.

Parameters

[in, out] `ima_` The image to be filled.

[in] `val` The value to assign to all sites.

Precondition

`ima` has to be initialized.

Definition at line 103 of file `fill_with_value.hh`.

9.32.2.3 `template<typename I , typename J > void mln::data::impl::generic::paste (const Image< I > & input_ , Image< J > & output_) [inline]`

Generic implementation of [data::paste](#).

Parameters

[in] `input_` The input image providing pixels values.

[in, out] `output_` The image in which values are assigned.

Definition at line 111 of file `paste.hh`.

9.32.2.4 `template<typename I , typename F > mln::trait::ch_value< I , typename F ::result >::ret mln::data::impl::generic::transform (const Image< I > & input_ , const Function_v2v< F > & f_)`

Generic implementation of [data::transform](#).

Parameters

[in] `input_` The input image.

[in] `f_` The function.

Definition at line 137 of file `data/transform.hh`.

References `mln::initialize()`.

9.32.2.5 `template<typename I1 , typename I2 , typename F > mln::trait::ch_value< I1 , typename F ::result >::ret mln::data::impl::generic::transform (const Image< I1 > & input1_ , const Image< I2 > & input2_ , const Function_vv2v< F > & f_)`

Generic implementation of [data::transform](#).

Parameters

[in] `input1_` The 1st input image.

[in] *input2_* The 2nd input image.

[in] *f_* The function.

Definition at line 166 of file data/transform.hh.

References mln::initialize().

9.32.2.6 `template<typename I1 , typename I2 , typename F > void
mln::data::impl::generic::transform_inplace (Image< I1 > & ima_ , const Image< I2 >
& aux_ , const Function_vv2v< F > & f_)`

Generic implementation of transform_inplace.

Parameters

[in] *ima_* The image to be transformed.

[in] *aux_* The auxiliary image.

[in] *f_* The function.

Definition at line 176 of file transform_inplace.hh.

9.32.2.7 `template<typename I , typename F > void mln::data::impl::generic::transform_inplace (
Image< I > & ima_ , const Function_v2v< F > & f_)`

Generic implementation of transform_inplace.

Parameters

[in, out] *ima_* The image to be transformed.

[in] *f_* The function.

Definition at line 149 of file transform_inplace.hh.

9.32.2.8 `template<typename A , typename I > A ::result mln::data::impl::generic::update (
Accumulator< A > & a_ , const Image< I > & input_) [inline]`

Generic implementation of [data::update](#).

Parameters

[in] *a_* The accumulator.

[in] *input_* The input image.

Returns

The accumulator result.

Definition at line 100 of file update.hh.

9.33 mln::data::naive Namespace Reference

Namespace of image processing routines related to pixel levels with naive approach.

Namespaces

- namespace [impl](#)
Implementation namespace of [data::naive](#) namespace.

Functions

- `template<typename I , typename W , typename O >`
`void median (const Image< I > &input, const Window< W > &win, Image< O > &output)`
Compute in output the median filter of image input by the window win.

9.33.1 Detailed Description

Namespace of image processing routines related to pixel levels with naive approach.

9.33.2 Function Documentation

- 9.33.2.1** `template<typename I , typename W , typename O > void mln::data::naive::median (`
`const Image< I > & input, const Window< W > & win, Image< O > & output)`
`[inline]`

Compute in output the median filter of image input by the window win.

Parameters

- [in] *input* The image to be filtered.
- [in] *win* The window.
- [in, out] *output* The output image.

This is a NAIVE version for test / comparison purpose so do NOT use it.

Precondition

input and output have to be initialized.

See also

[mln::data::median](#)

Definition at line 99 of file naive/median.hh.

9.34 mln::data::naive::impl Namespace Reference

Implementation namespace of [data::naive](#) namespace.

9.34.1 Detailed Description

Implementation namespace of [data::naive](#) namespace.

9.35 mln::debug Namespace Reference

Namespace of routines that help to debug.

Namespaces

- namespace `impl`
Implementation namespace of debug namespace.

Functions

- `template<typename I, typename G, typename F >`
`void draw_graph (Image< I > &ima, const p_vertices< G, F > &pv, typename I::value vcolor, typename I::value ecolor)`
Draw an image `ima` from a `mln::p_vertices` `pv`, with value `vcolor` for vertices, value `ecolor` for edges and 0 for the background.
- `template<typename I, typename G, typename F, typename V, typename E >`
`void draw_graph (Image< I > &ima, const p_vertices< G, F > &pv, const Function< V > &vcolor_f_, const Function< E > &ecolor_f_)`
Draw an image `ima` from a `mln::p_vertices` `pv`.
- `template<typename I, typename G, typename F, typename V, typename E >`
`void draw_graph (Image< I > &ima, const p_vertices< util::line_graph< G >, F > &pv, const Function< V > &vcolor_f_, const Function< E > &ecolor_f_)`
Draw an image `ima` from a `mln::p_vertices` `pv`.
- `std::string filename (const std::string &filename, int id)`
Constructs and returns a formatted output file name.
- signed short `format` (signed char v)
Format a signed char to print it properly, i.e., like an integer value.
- unsigned short `format` (unsigned char v)
Format an unsigned char to print it properly, i.e., like an integer value.
- `template<typename T >`
`const T & format (const T &v)`
Default version for formatting a value is a no-op.
- char `format` (bool v)
Format a Boolean to print it nicely: "|" for true and "-" for false.
- `template<typename I >`
`void iota (Image< I > &input, unsigned base_index)`
- `template<typename I >`
`mln::trait::concrete< I >::ret mosaic (const util::array< I > &input, unsigned n_horizontal, const typename I::value &bg)`
Create a single image from an array of image.

- `template<typename I >`
`void println (const std::string &msg, const Image< I > &input)`
Print the message msg and the image input on the standard output.
- `template<typename I >`
`void println (const Image< I > &input)`
Print the image input on the standard output.
- `template<typename I >`
`void println_with_border (const Image< I > &input)`
Print the image input on the standard output.
- `void put_word (image2d< char > &inout, const point2d &word_start, const std::string &word)`
Put the word starting at location word_start in the image inout.
- `template<typename I >`
`image2d< typename I::value > slices_2d (const Image< I > &input, unsigned n_horizontal, unsigned n_vertical, const typename I::value &bg)`
Create a 2D image of the slices of the 3D image input.
- `template<typename I >`
`image2d< typename I::value > slices_2d (const Image< I > &input, float ratio_hv, const typename I::value &bg)`
Create a 2D image of the slices of the 3D image input.
- `template<typename I , typename J >`
`mln::trait::ch_value< I, value::rgb8 >::ret superpose (const Image< I > &input, const Image< J > &object)`
- `template<typename I , typename J >`
`mln::trait::ch_value< I, value::rgb8 >::ret superpose (const Image< I > &input_, const Image< J > &object_, const value::rgb8 &object_color)`
Superpose two images.
- `template<typename I >`
`void z_order (Image< I > &input)`

9.35.1 Detailed Description

Namespace of routines that help to debug.

9.35.2 Function Documentation

9.35.2.1 `template<typename I , typename G , typename F > void mln::debug::draw_graph (Image< I > & ima, const p_vertices< G, F > & pv, typename I::value vcolor, typename I::value ecolor) [inline]`

Draw an image ima from a `mln::p_vertices` pv, with value vcolor for vertices, value ecolor for edges and 0 for the background.

Definition at line 142 of file draw_graph.hh.

References mln::p_vertices< G, F >::graph(), and mln::draw::line().

Referenced by mln::make_debug_graph_image().

9.35.2.2 `template<typename I , typename G , typename F , typename V , typename E > void
mln::debug::draw_graph (Image< I > & ima, const p_vertices< G, F > & pv, const
Function< V > & vcolor_f_, const Function< E > & ecolor_f_) [inline]`

Draw an image *ima* from a mln::p_vertices *pv*.

Colors for vertices are defined through *vcolor_f_*. Colors for edges are defined through *ecolor_f_*.

Definition at line 173 of file draw_graph.hh.

References mln::draw::box_plain(), mln::box< P >::crop_wrt(), mln::p_vertices< G, F >::graph(), and mln::draw::line().

9.35.2.3 `template<typename I , typename G , typename F , typename V , typename E > void
mln::debug::draw_graph (Image< I > & ima, const p_vertices< util::line_graph< G
>, F > & pv, const Function< V > & vcolor_f_, const Function< E > & ecolor_f_)
[inline]`

Draw an image *ima* from a mln::p_vertices *pv*.

Colors for vertices are defined through *vcolor_f_*. Colors for edges are defined through *ecolor_f_*.

Definition at line 211 of file draw_graph.hh.

References mln::p_line2d::begin(), mln::p_line2d::end(), mln::p_vertices< G, F >::graph(), and mln::draw::line().

9.35.2.4 `std::string mln::debug::filename (const std::string & filename, int id = -1)
[inline]`

Constructs and returns a formatted output file name.

The file name is formatted as follow:

‘filename_prefix’_‘id’_‘filename’

Where:

- ‘filename_prefix’ can be set through the global variable debug::internal::filename_prefix.

‘postfix_id’ is autoincremented by default. Its value can be forced.

- ‘filename’ is the given filename

Definition at line 86 of file filename.hh.

9.35.2.5 `signed short mln::debug::format (signed char v) [inline]`

Format a signed char to print it properly, i.e., like an integer value.

Definition at line 78 of file format.hh.

9.35.2.6 unsigned short mln::debug::format (unsigned char v) [inline]

Format an unsigned char to print it properly, i.e., like an integer value.

Definition at line 85 of file format.hh.

9.35.2.7 template<typename T > const T & mln::debug::format (const T & v) [inline]

Default version for formatting a value is a no-op.

Definition at line 64 of file format.hh.

Referenced by mln::value::operator<<(), and mln::Gpoint< E >::operator<<().

9.35.2.8 char mln::debug::format (bool v) [inline]

Format a Boolean to print it nicely: "|" for true and "-" for false.

Definition at line 71 of file format.hh.

9.35.2.9 template<typename I > void mln::debug::iota (Image< I > & input, unsigned base_index) [inline]

Fill the image `input` with successive values.

Parameters

[in, out] *input* The image in which values are assigned.

Definition at line 88 of file debug/iota.hh.

References `iota()`.

Referenced by `iota()`.

9.35.2.10 template<typename I > mln::trait::concrete< I >::ret mln::debug::mosaic (const util::array< I > & input, unsigned n_horizontal, const typename I::value & bg) [inline]

Create a single image from an array of image.

The size of the output image is defined by:

$$\text{width} = n_horizontal * \max(\text{input}[i].\text{ncols}()) \quad \text{height} = (\text{input.size}() / n_horizontal) * \max(\text{input}[i].\text{nrows}())$$
Returns

a single image where all the input images are displayed as a mosaic.

Definition at line 77 of file mosaic.hh.

References `mln::apply_p2p()`, `mln::data::fill()`, and `mln::data::paste()`.

9.35.2.11 `template<typename I> void mln::debug::println (const std::string & msg, const Image< I > & input)`

Print the message `msg` and the image `input` on the standard output.

Definition at line 98 of file `println.hh`.

References `println()`.

9.35.2.12 `template<typename I> void mln::debug::println (const Image< I > & input) [inline]`

Print the image `input` on the standard output.

Definition at line 87 of file `println.hh`.

References `mln::geom::bbox()`.

Referenced by `println()`.

9.35.2.13 `template<typename I> void mln::debug::println_with_border (const Image< I > & input) [inline]`

Print the image `input` on the standard output.

Definition at line 79 of file `println_with_border.hh`.

References `mln::geom::bbox()`.

9.35.2.14 `void mln::debug::put_word (image2d< char > & inout, const point2d & word_start, const std::string & word) [inline]`

Put the `word` starting at location `word_start` in the image `inout`.

Definition at line 55 of file `put_word.hh`.

References `mln::image2d< T >::has()`, and `mln::point< G, C >::last_coord()`.

9.35.2.15 `template<typename I> image2d< typename I::value > mln::debug::slices_2d (const Image< I > & input, unsigned n_horizontal, unsigned n_vertical, const typename I::value & bg) [inline]`

Create a 2D image of the slices of the 3D image `input`.

Definition at line 79 of file `slices_2d.hh`.

References `mln::apply_p2p()`, `mln::data::fill()`, and `mln::data::paste()`.

Referenced by `slices_2d()`.

9.35.2.16 `template<typename I> image2d< typename I::value > mln::debug::slices_2d (const Image< I > & input, float ratio_hv, const typename I::value & bg)`

Create a 2D image of the slices of the 3D image `input`.

Definition at line 162 of file `slices_2d.hh`.

References `slices_2d()`.

9.35.2.17 `template<typename I , typename J > mln::trait::ch_value< I, value::rgb8 >::ret
mln::debug::superpose (const Image< I > & input, const Image< J > & object)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 106 of file debug/superpose.hh.

References `mln::literal::red`, and `superpose()`.

9.35.2.18 `template<typename I , typename J > mln::trait::ch_value< I, value::rgb8 >::ret
mln::debug::superpose (const Image< I > & input_, const Image< J > & object_
const value::rgb8 & object_color)`

Superpose two images.

Parameters

- [in] *input_* An image. Its value type must be convertible toward `value::rgb8` thanks to a conversion operator or `convert::from_to`.
- [in] *object_* A scalar or labeled image. Objects used for superposition. have their pixel values different from 0.
- [in] *object_color* The color used to draw the objects in `object_`.

Precondition

`input_` and `object_` must have the same domain.

Returns

A color image.

Definition at line 79 of file debug/superpose.hh.

References `mln::data::convert()`, `mln::data::fill()`, and `mln::literal::zero`.

Referenced by `superpose()`.

9.35.2.19 `template<typename I > void mln::debug::z_order (Image< I > & input) [inline]`

Fill the image `input` with Z-order (curve) values.

Parameters

- [in, out] *input* The image in which values are assigned.

Reference: [http://en.wikipedia.org/wiki/Z-order_\(curve\)](http://en.wikipedia.org/wiki/Z-order_(curve))

Definition at line 142 of file `z_order.hh`.

9.36 mln::debug::impl Namespace Reference

Implementation namespace of `debug` namespace.

9.36.1 Detailed Description

Implementation namespace of debug namespace.

9.37 mln::def Namespace Reference

Namespace for core definitions.

Typedefs

- typedef short [coord](#)
Definition of the default coordinate type: 'short'.
- typedef float [coordf](#)
Definition of the floating coordinate type.

Enumerations

- enum
Definition of the number of bits of the low quantization threshold.

9.37.1 Detailed Description

Namespace for core definitions.

9.37.2 Typedef Documentation

9.37.2.1 typedef short mln::def::coord

Definition of the default coordinate type: 'short'.

Definition at line 43 of file coord.hh.

9.37.2.2 typedef float mln::def::coordf

Definition of the floating coordinate type.

Definition at line 41 of file coordf.hh.

9.37.3 Enumeration Type Documentation

9.37.3.1 anonymous enum

Definition of the number of bits of the low quantization threshold.

Definition at line 43 of file low_quant_nbits.hh.

9.38 mln::display Namespace Reference

Namespace of routines that help to display images.

Namespaces

- namespace [impl](#)

Implementation namespace of display namespace.

9.38.1 Detailed Description

Namespace of routines that help to display images.

9.39 mln::display::impl Namespace Reference

Implementation namespace of display namespace.

Namespaces

- namespace [generic](#)

Generic implementation namespace of display namespace.

9.39.1 Detailed Description

Implementation namespace of display namespace.

9.40 mln::display::impl::generic Namespace Reference

Generic implementation namespace of display namespace.

9.40.1 Detailed Description

Generic implementation namespace of display namespace.

9.41 mln::doc Namespace Reference

The namespace [mln::doc](#) is only for documentation purpose.

Classes

- struct [Accumulator](#)
Documentation class for `mln::Accumulator`.
- struct [Box](#)
Documentation class for `mln::Box`.
- struct [Dpoint](#)
Documentation class for `mln::Dpoint`.
- struct [Fastest_Image](#)
Documentation class for the concept of images that have the speed property set to "fastest".
- struct [Generalized_Pixel](#)
Documentation class for `mln::Generalized_Pixel`.
- struct [Image](#)
Documentation class for `mln::Image`.
- struct [Iterator](#)
Documentation class for `mln::Iterator`.
- struct [Neighborhood](#)
Documentation class for `mln::Neighborhood`.
- struct [Object](#)
Documentation class for `mln::Object`.
- struct [Pixel_Iterator](#)
Documentation class for `mln::Iterator`.
- struct [Point_Site](#)
Documentation class for `mln::Point_Site`.
- struct [Site_Iterator](#)
Documentation class for `mln::Site_Iterator`.
- struct [Site_Set](#)
Documentation class for `mln::Site_Set`.
- struct [Value_Iterator](#)
Documentation class for `mln::Value_Iterator`.
- struct [Value_Set](#)
Documentation class for `mln::Value_Set`.
- struct [Weighted_Window](#)
Documentation class for `mln::Weighted_Window`.

- struct [Window](#)

Documentation class for `mln::Window`.

9.41.1 Detailed Description

The namespace `mln::doc` is only for documentation purpose. Since concepts are not yet part of the C++ Standard, they are not explicitly expressed in code. Their documentation is handled by their respective ghost class, located in this namespace.

Warning

The ghost classes located in `mln::doc` should not be used by the client.

9.42 `mln::draw` Namespace Reference

Namespace of drawing routines.

Functions

- `template<typename I, typename B >`
void `box` (`Image< I > &ima`, const `Box< B > &b`, const `typename I::value &v`)
- `template<typename I, typename B >`
void `box_plain` (`Image< I > &ima`, const `Box< B > &b`, const `typename I::value &v`)
- `template<typename I >`
void `dashed_line` (`Image< I > &ima`, const `typename I::psite &beg`, const `typename I::psite &end`, const `typename I::value &v`)
- `template<typename I >`
void `line` (`Image< I > &ima`, const `typename I::psite &beg`, const `typename I::psite &end`, const `typename I::value &v`)
- `template<typename I >`
void `plot` (`Image< I > &ima`, const `typename I::point &p`, const `typename I::value &v`)
- `template<typename I >`
void `polygon` (`Image< I > &ima`, const `p_array< typename I::site > &par`, const `typename I::value &v`, unsigned `output_ratio`)
- `template<typename I, typename S >`
void `site_set` (`Image< I > &ima`, const `Site_Set< S > &s`, const `typename I::value &v`, unsigned `output_ratio=1`)

9.42.1 Detailed Description

Namespace of drawing routines.

9.42.2 Function Documentation

- 9.42.2.1** `template<typename I, typename B > void mln::draw::box (Image< I > & ima, const Box< B > & b, const typename I::value & v) [inline]`

Draw a box at value `v` in image `ima`

Parameters

- [in, out] *ima* The image to be drawn.
- [in] *b* the box to draw.
- [in] *v* The value to assign to all drawn pixels.

Precondition

- ima* has to be initialized.
- ima* has beg.
- ima* has end.

Definition at line 72 of file draw/box.hh.

References line().

9.42.2.2 `template<typename I , typename B > void mln::draw::box_plain (Image< I > & ima, const Box< B > & b, const typename I::value & v) [inline]`

Draw a plain box at value *v* in image *ima*

Parameters

- [in, out] *ima* The image to be drawn.
- [in] *b* the box to draw.
- [in] *v* The value to assign to all drawn pixels.

Precondition

- ima* has to be initialized.
- ima* has beg.
- ima* has end.

Definition at line 71 of file box_plain.hh.

References mln::data::fill().

Referenced by mln::debug::draw_graph().

9.42.2.3 `template<typename I > void mln::draw::dashed_line (Image< I > & ima, const typename I::psite & beg, const typename I::psite & end, const typename I::value & v) [inline]`

Draw a dashed line at level *v* in image *ima* between the points *beg* and *end*.

Parameters

- [in, out] *ima* The image to be drawn.
- [in] *beg* The start point to drawn dashed_line.
- [in] *end* The end point to drawn dashed_line.
- [in] *v* The value to assign to all drawn pixels.

Precondition

`ima` has to be initialized.
`ima` has `beg`.
`ima` has `end`.

Definition at line 91 of file `dashed_line.hh`.

References `mln::data::fill()`.

9.42.2.4 `template<typename I> void mln::draw::line (Image< I > & ima, const typename I::psite & beg, const typename I::psite & end, const typename I::value & v)`
`[inline]`

Draw a line at level `v` in image `ima` between the points `beg` and `end`.

Parameters

`[in, out]` *ima* The image to be drawn.
`[in]` *beg* The start point to drawn line.
`[in]` *end* The end point to drawn line.
`[in]` *v* The value to assign to all drawn pixels.

Precondition

`ima` has to be initialized.
`ima` has `beg`.
`ima` has `end`.

Definition at line 72 of file `draw/line.hh`.

References `mln::data::paste()`.

Referenced by `box()`, `mln::debug::draw_graph()`, and `polygon()`.

9.42.2.5 `template<typename I> void mln::draw::plot (Image< I > & ima, const typename I::point & p, const typename I::value & v)`

Plot a point at level `v` in image `ima`

Parameters

`[in, out]` *ima* The image to be drawn.
`[in]` *p* The point to be plotted.
`[in]` *v* The value to assign to all drawn pixels.

Precondition

`ima` has to be initialized.
`ima` has `p`.

9.42.2.6 `template<typename I> void mln::draw::polygon (Image< I > & ima, const p_array< typename I::site > & par, const typename I::value & v, unsigned output_ratio)`

Draw a polygon at level v in image ima .

Parameters

- [in, out] *ima* The image to be drawn.
- [in] *par* The polygon site set.
- [in] v The value to assign to all drawn pixels.

Precondition

ima has to be initialized.

Definition at line 70 of file polygon.hh.

References `line()`.

9.42.2.7 `template<typename I, typename S> void mln::draw::site_set (Image< I > & ima, const Site_Set< S > & s, const typename I::value & v, unsigned output_ratio = 1)`

Draw a sites with value v in image ima

Parameters

- [in, out] *ima* The image to be drawn.
- [in] *b* the site set to draw.
- [in] v The value to assign to all drawn pixels.
- [in] *output_ratio* size ratio between output image and the image from which the bboxes were calculated.

Precondition

s is included in ima domain.

Definition at line 65 of file mln/draw/site_set.hh.

9.43 mln::estim Namespace Reference

Namespace of estimation materials.

Functions

- `template<typename I>`
`mln::value::props< typename I::value >::sum mean (const Image< I > &input)`
Compute the mean value of the pixels of image input.
- `template<typename S, typename I, typename M>`
`void mean (const Image< I > &input, M &result)`

Compute the mean value of the pixels of image `input`.

- `template<typename I >`
`void min_max (const Image< I > &input, typename I::value &min, typename I::value &max)`
Compute the min and max values of the pixels of image `input`.
- `template<typename I >`
`mln::value::props< typename I::value >::sum sum (const Image< I > &input)`
Compute the sum value of the pixels of image `input`.
- `template<typename I, typename S >`
`void sum (const Image< I > &input, S &result)`
Compute the sum value of the pixels of image `input`.

9.43.1 Detailed Description

Namespace of estimation materials.

9.43.2 Function Documentation

9.43.2.1 `template<typename I > mln::value::props< typename I::value >::sum mln::estim::mean (const Image< I > & input) [inline]`

Compute the mean value of the pixels of image `input`.

Parameters

[in] *input* The image.

Returns

The mean value.

Definition at line 68 of file `estim/mean.hh`.

References `mln::data::compute()`.

9.43.2.2 `template<typename S, typename I, typename M > void mln::estim::mean (const Image< I > & input, M & result) [inline]`

Compute the mean value of the pixels of image `input`.

Parameters

[in] *input* The image.

[out] *result* The mean value.

The free parameter `S` is the type used to compute the summation.

Definition at line 76 of file `estim/mean.hh`.

References `mln::data::compute()`.

9.43.2.3 `template<typename I> void mln::estim::min_max (const Image< I > & input, typename I::value & min, typename I::value & max) [inline]`

Compute the min and max values of the pixels of image `input`.

Parameters

- [in] *input* The image.
- [out] *min* The minimum pixel value of `input`.
- [out] *max* The maximum pixel value of `input`.

Definition at line 61 of file `estim/min_max.hh`.

References `mln::data::compute()`.

Referenced by `mln::data::impl::stretch()`, and `mln::make::voronoi()`.

9.43.2.4 `template<typename I> mln::value::props< typename I::value >::sum mln::estim::sum (const Image< I > & input) [inline]`

Compute the sum value of the pixels of image `input`.

Parameters

- [in] *input* The image.

Returns

The sum value.

Definition at line 67 of file `estim/sum.hh`.

References `mln::data::compute()`.

9.43.2.5 `template<typename I, typename S> void mln::estim::sum (const Image< I > & input, S & result) [inline]`

Compute the sum value of the pixels of image `input`.

Parameters

- [in] *input* The image.
- [out] *result* The sum value.

Definition at line 75 of file `estim/sum.hh`.

References `mln::data::compute()`.

9.44 mln::extension Namespace Reference

Namespace of extension tools.

Functions

- `template<typename I, typename W >`
`void adjust (const Image< I > &ima, const Window< W > &win)`
Adjust the domain extension of image `ima` with the size of the window `win`.
- `template<typename I, typename W >`
`void adjust (const Image< I > &ima, const Weighted_Window< W > &wwin)`
Adjust the domain extension of image `ima` with the size of the weighted window `wwin`.
- `template<typename I >`
`void adjust (const Image< I > &ima, unsigned delta)`
Adjust the domain extension of image `ima` with the size `delta`.
- `template<typename I, typename N >`
`void adjust (const Image< I > &ima, const Neighborhood< N > &nbh)`
Adjust the domain extension of image `ima` with the size of the neighborhood `nbh`.
- `template<typename I, typename W >`
`void adjust_duplicate (const Image< I > &ima, const Window< W > &win)`
Adjust then duplicate.
- `template<typename I, typename W >`
`void adjust_fill (const Image< I > &ima, const Window< W > &win, const typename I::value &val)`
Adjust then fill.
- `template<typename I >`
`void duplicate (const Image< I > &ima)`
Assign the contents of the domain extension by duplicating the values of the inner boundary of image `ima`.
- `template<typename I >`
`void fill (const Image< I > &ima, const typename I::value &val)`

9.44.1 Detailed Description

Namespace of extension tools.

9.44.2 Function Documentation

9.44.2.1 `template<typename I, typename W > void mln::extension::adjust (const Image< I > & ima, const Window< W > & win)`

Adjust the domain extension of image `ima` with the size of the window `win`.

Definition at line 89 of file `extension/adjust.hh`.

References `mln::geom::delta()`.

Referenced by `adjust()`, `adjust_duplicate()`, and `adjust_fill()`.

9.44.2.2 `template<typename I, typename W> void mln::extension::adjust (const Image< I > & ima, const Weighted_Window< W > & wwin)`

Adjust the domain extension of image `ima` with the size of the weighted window `wwin`.

Definition at line 97 of file `extension/adjust.hh`.

References `adjust()`, and `mln::geom::delta()`.

9.44.2.3 `template<typename I> void mln::extension::adjust (const Image< I > & ima, unsigned delta)`

Adjust the domain extension of image `ima` with the size `delta`.

Definition at line 113 of file `extension/adjust.hh`.

References `adjust()`.

9.44.2.4 `template<typename I, typename N> void mln::extension::adjust (const Image< I > & ima, const Neighborhood< N > & nbh)`

Adjust the domain extension of image `ima` with the size of the neighborhood `nbh`.

Definition at line 105 of file `extension/adjust.hh`.

References `adjust()`, and `mln::geom::delta()`.

9.44.2.5 `template<typename I, typename W> void mln::extension::adjust_duplicate (const Image< I > & ima, const Window< W > & win)`

Adjust then duplicate.

Definition at line 70 of file `adjust_duplicate.hh`.

References `adjust()`, and `duplicate()`.

9.44.2.6 `template<typename I, typename W> void mln::extension::adjust_fill (const Image< I > & ima, const Window< W > & win, const typename I::value & val)`

Adjust then fill.

Definition at line 72 of file `adjust_fill.hh`.

References `adjust()`, and `fill()`.

9.44.2.7 `template<typename I> void mln::extension::duplicate (const Image< I > & ima)`

Assign the contents of the domain extension by duplicating the values of the inner boundary of image `ima`.

Definition at line 58 of file `extension/duplicate.hh`.

Referenced by `adjust_duplicate()`.

9.44.2.8 `template<typename I> void mln::extension::fill (const Image< I > & ima, const typename I::value & val)`

Fill the domain extension of image `ima` with the single value `v`.

Parameters

- `[in, out]` ***ima*** The image whose domain extension is to be filled.
- `[in]` ***val*** The value to assign.

Precondition

`ima` has to be initialized.

Definition at line 173 of file `extension/fill.hh`.

Referenced by `adjust_fill()`.

9.45 `mln::fun` Namespace Reference

Namespace of functions.

Namespaces

- namespace [access](#)
Namespace for access functions.
- namespace [i2v](#)
Namespace of integer-to-value functions.
- namespace [n2v](#)
Namespace of functions from nil to value.
- namespace [p2b](#)
Namespace of functions from point to boolean.
- namespace [p2p](#)
Namespace of functions from grid point to grid point.
- namespace [p2v](#)
Namespace of functions from point to value.
- namespace [stat](#)
Namespace of statistical functions.
- namespace [v2b](#)
Namespace of functions from value to logic value.
- namespace [v2i](#)
Namespace of value-to-integer functions.

- namespace [v2v](#)
Namespace of functions from value to value.
- namespace [v2w2v](#)
Namespace of bijective functions.
- namespace [v2w_w2v](#)
Namespace of functions from value to value.
- namespace [vv2b](#)
Namespace of functions from value to value.
- namespace [vv2v](#)
Namespace of functions from a couple of values to a value.
- namespace [x2p](#)
Namespace of functions from point to value.
- namespace [x2v](#)
Namespace of functions from vector to value.
- namespace [x2x](#)
Namespace of functions from vector to vector.

Classes

- struct [from_accu](#)
Wrap an accumulator into a function.

9.45.1 Detailed Description

Namespace of functions. Forward declarations.

`fun::i2v::array`

Forward declaration.

9.46 mln::fun::access Namespace Reference

Namespace for access functions.

9.46.1 Detailed Description

Namespace for access functions.

9.47 mln::fun::i2v Namespace Reference

Namespace of integer-to-value functions.

Functions

- `template<typename T > std::ostream & operator<< (std::ostream &ostr, const array< T > &a)`
Operator<<.

9.47.1 Detailed Description

Namespace of integer-to-value functions.

9.47.2 Function Documentation

- 9.47.2.1** `template<typename T > std::ostream & mln::fun::i2v::operator<< (std::ostream & ostr, const array< T > & a)`

Operator<<.

Definition at line 353 of file fun/i2v/array.hh.

9.48 mln::fun::n2v Namespace Reference

Namespace of functions from nil to value.

Classes

- struct `white_gaussian`
Generate a White Gaussian Noise.

9.48.1 Detailed Description

Namespace of functions from nil to value.

9.49 mln::fun::p2b Namespace Reference

Namespace of functions from point to boolean.

Classes

- struct [antilogy](#)
A *p2b* function always returning *false*.
- struct [tautology](#)
A *p2b* function always returning *true*.

9.49.1 Detailed Description

Namespace of functions from point to boolean.

9.50 mln::fun::p2p Namespace Reference

Namespace of functions from grid point to grid point.

9.50.1 Detailed Description

Namespace of functions from grid point to grid point.

9.51 mln::fun::p2v Namespace Reference

Namespace of functions from point to value.

9.51.1 Detailed Description

Namespace of functions from point to value.

9.52 mln::fun::stat Namespace Reference

Namespace of statistical functions.

9.52.1 Detailed Description

Namespace of statistical functions.

9.53 mln::fun::v2b Namespace Reference

Namespace of functions from value to logic value.

Classes

- struct [lnot](#)
Functor computing logical-not on a value.
- struct [threshold](#)
Threshold function.

9.53.1 Detailed Description

Namespace of functions from value to logic value.

9.54 mln::fun::v2i Namespace Reference

Namespace of value-to-integer functions.

9.54.1 Detailed Description

Namespace of value-to-integer functions.

9.55 mln::fun::v2v Namespace Reference

Namespace of functions from value to value.

Classes

- class [ch_function_value](#)
Wrap a function $v2v$ and convert its result to another type.
- struct [component](#)
Functor that accesses the i -th component of a value.
- struct [l1_norm](#)
L1-norm.
- struct [l2_norm](#)
L2-norm.
- struct [linear](#)
*Linear function. $f(v) = a * v + b$. V is the type of input values; T is the type used to compute the result; R is the result type.*
- struct [linfty_norm](#)
L-infty norm.

- struct [rgb8_to_rgbn](#)
Convert a rgb8 value to a rgn, $n < 8$.

Variables

- [f_hsi_to_rgb_3x8_t](#) [f_hsi_to_rgb_3x8](#)
Global variable.
- [f_hsl_to_rgb_3x8_t](#) [f_hsl_to_rgb_3x8](#)
Global variables.
- [f_rgb_to_hsi_f_t](#) [f_rgb_to_hsi_f](#)
Global variables.
- [f_rgb_to_hsl_f_t](#) [f_rgb_to_hsl_f](#)
Global variables.

9.55.1 Detailed Description

Namespace of functions from value to value.

9.55.2 Variable Documentation

9.55.2.1 [f_hsi_to_rgb_3x8_t](#) mln::fun::v2v::f_hsi_to_rgb_3x8

Global variable.

Definition at line 74 of file hsi_to_rgb.hh.

9.55.2.2 [f_hsl_to_rgb_3x8_t](#) mln::fun::v2v::f_hsl_to_rgb_3x8

Global variables.

Definition at line 92 of file hsl_to_rgb.hh.

9.55.2.3 [f_rgb_to_hsi_f_t](#) mln::fun::v2v::f_rgb_to_hsi_f

Global variables.

Definition at line 66 of file rgb_to_hsi.hh.

9.55.2.4 [f_rgb_to_hsl_f_t](#) mln::fun::v2v::f_rgb_to_hsl_f

Global variables.

Definition at line 75 of file rgb_to_hsl.hh.

9.56 `mln::fun::v2w2v` Namespace Reference

Namespace of bijective functions.

Classes

- struct `cos`
Cosinus bijective functor.

9.56.1 Detailed Description

Namespace of bijective functions.

9.57 `mln::fun::v2w_w2v` Namespace Reference

Namespace of functions from value to value.

Classes

- struct `l1_norm`
L1-norm.
- struct `l2_norm`
L2-norm.
- struct `linfty_norm`
L-infty norm.

9.57.1 Detailed Description

Namespace of functions from value to value.

9.58 `mln::fun::vv2b` Namespace Reference

Namespace of functions from value to value.

Classes

- struct `eq`
Functor computing equal between two values.
- struct `ge`
Functor computing "greater or equal than" between two values.

- struct [gt](#)
Functor computing "greater than" between two values.
- struct [implies](#)
Functor computing logical-implies between two values.
- struct [le](#)
Functor computing "lower or equal than" between two values.
- struct [lt](#)
Functor computing "lower than" between two values.

9.58.1 Detailed Description

Namespace of functions from value to value.

9.59 mln::fun::vv2v Namespace Reference

Namespace of functions from a couple of values to a value.

Classes

- struct [diff_abs](#)
A functor computing the diff_absimum of two values.
- struct [land](#)
Functor computing logical-and between two values.
- struct [land_not](#)
Functor computing logical and-not between two values.
- struct [lor](#)
Functor computing logical-or between two values.
- struct [lxor](#)
Functor computing logical-xor between two values.
- struct [max](#)
A functor computing the maximum of two values.
- struct [min](#)
A functor computing the minimum of two values.
- struct [vec](#)
A functor computing the vecimum of two values.

9.59.1 Detailed Description

Namespace of functions from a couple of values to a value.

9.60 mln::fun::x2p Namespace Reference

Namespace of functions from point to value.

Classes

- struct [closest_point](#)
FIXME: doxygen + concept checking.

9.60.1 Detailed Description

Namespace of functions from point to value.

9.61 mln::fun::x2v Namespace Reference

Namespace of functions from vector to value.

Classes

- struct [bilinear](#)
Represent a bilinear interolation of values from an underlying image.
- struct [trilinear](#)
Represent a trilinear interolation of values from an underlying image.

9.61.1 Detailed Description

Namespace of functions from vector to value.

9.62 mln::fun::x2x Namespace Reference

Namespace of functions from vector to vector.

Classes

- struct [composed](#)
Represent a composition of two transformations.

- struct [linear](#)
Represent a linear interpolation of values from an underlying image.
- struct [rotation](#)
Represent a rotation function.
- struct [translation](#)
Translation function-object.

9.62.1 Detailed Description

Namespace of functions from vector to vector.

9.63 mln::geom Namespace Reference

Namespace of all things related to geometry.

Namespaces

- namespace [impl](#)
Implementation namespace of geom namespace.

Classes

- class [complex_geometry](#)
A functor returning the sites of the faces of a complex where the locations of each 0-face is stored.

Functions

- `template<typename S >`
`box< typename S::site > bbox (const Site_Set< S > &pset)`
Compute the precise bounding box of a point set pset.
- `template<typename I >`
`box< typename I::site > bbox (const Image< I > &ima)`
Compute the precise bounding box of a point set pset.
- `template<typename W >`
`box< typename W::psite > bbox (const Window< W > &win)`
Compute the precise bounding box of a window win.
- `template<typename W >`
`box< typename W::psite > bbox (const Weighted_Window< W > &win)`
Compute the precise bounding box of a weighted window win.

- `template<typename I, typename W >`
`mln::trait::ch_value< I, unsigned >::ret chamfer (const Image< I > &input_, const W &w_win_, unsigned max=mln_max(unsigned))`
Apply chamfer algorithm to a binary image.
- `template<typename W >`
`unsigned delta (const Window< W > &win)`
Compute the delta of a window win.
- `template<typename W >`
`unsigned delta (const Weighted_Window< W > &wwin)`
Compute the delta of a weighted window wwin.
- `template<typename N >`
`unsigned delta (const Neighborhood< N > &nbh)`
Compute the delta of a neighborhood nbh.
- `template<typename I >`
`mln::trait::concrete< I >::ret horizontal_symmetry (const Image< I > &input)`
Performs a horizontal symmetry.
- `template<typename I >`
`I::site::coord max_col (const Image< I > &ima)`
Give the maximum column of an image.
- `template<typename B >`
`B::site::coord max_col (const Box< B > &b)`
Give the maximum col of an box 2d or 3d.
- `template<typename I >`
`I::site::coord max_ind (const Image< I > &ima)`
Give the maximum ind of an image.
- `template<typename I >`
`I::site::coord max_row (const Image< I > &ima)`
Give the maximum row of an image.
- `template<typename B >`
`B::site::coord max_row (const Box< B > &b)`
Give the maximum row of an box 2d or 3d.
- `template<typename I >`
`I::site::coord max_sli (const Image< I > &ima)`
Give the maximum sli of an image.
- `std::pair< complex_image< 2, mln::space_2complex_geometry, algebra::vec< 3, float > >, complex_image< 2, mln::space_2complex_geometry, float > > mesh_corner_point_area (const p_complex< 2, space_2complex_geometry > &mesh)`
Compute the area "belonging" to normals at vertices.

- `std::pair< complex_image< 2, mln::space_2complex_geometry, float >, complex_image< 2, mln::space_2complex_geometry, float > > mesh_curvature (const p_complex< 2, space_2complex_geometry > &mesh)`
Compute the principal curvatures of a surface at vertices.
- `complex_image< 2, mln::space_2complex_geometry, algebra::vec< 3, float > > mesh_normal (const p_complex< 2, space_2complex_geometry > &mesh)`
Compute normals at vertices.
- `template<typename I >
I::site::coord min_col (const Image< I > &ima)`
Give the minimum column of an image.
- `template<typename B >
B::site::coord min_col (const Box< B > &b)`
Give the minimum column of a box 2d or 3d.
- `template<typename I >
I::site::coord min_ind (const Image< I > &ima)`
Give the minimum ind of an image.
- `template<typename I >
I::site::coord min_row (const Image< I > &ima)`
Give the minimum row of an image.
- `template<typename B >
B::site::coord min_row (const Box< B > &b)`
Give the minimum row of a box 2d or 3d.
- `template<typename I >
I::site::coord min_sli (const Image< I > &ima)`
Give the minimum sli of an image.
- `template<typename I >
unsigned ncols (const Image< I > &ima)`
Give the number of columns of an image.
- `template<typename B >
unsigned ncols (const Box< B > &b)`
Give the number of cols of a box 2d or 3d.
- `template<typename I >
unsigned ninds (const Image< I > &ima)`
Give the number of inds of an image.
- `template<typename I >
unsigned nrows (const Image< I > &ima)`
Give the number of rows of an image.
- `template<typename B >
unsigned nrows (const Box< B > &b)`

Give the number of rows of a box 2d or 3d.

- `template<typename I >`
`unsigned nsites (const Image< I > &input)`
Compute the number of sites of the image `input`.
- `template<typename I >`
`unsigned nslis (const Image< I > &ima)`
Give the number of slices of an image.
- `template<typename S >`
`void pmin_pmax (const Site_Set< S > &s, typename S::site &pmin, typename S::site &pmax)`
Compute the minimum and maximum points, `pmin` and `max`, of point set `s`.
- `template<typename S >`
`std::pair< typename S::site, typename S::site > pmin_pmax (const Site_Set< S > &s)`
Compute the minimum and maximum points of point set `s`.
- `template<typename I >`
`std::pair< typename I::site, typename I::site > pmin_pmax (const Site_Iterator< I > &p)`
Compute the minimum and maximum points when browsing with iterator `p`.
- `template<typename I >`
`void pmin_pmax (const Site_Iterator< I > &p, typename I::site &pmin, typename I::site &pmax)`
Compute the minimum and maximum points, `pmin` and `max`, when browsing with iterator `p`.
- `template<typename I , typename Ext >`
`mln::trait::concrete< I >::ret rotate (const Image< I > &input, double angle, const Ext &extension)`
- `template<typename B >`
`B rotate (const Box< B > &box, double angle)`
This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. The rotation center `ref` is set to `box.pcenter()`.
- `template<typename I , typename Ext , typename S >`
`mln::trait::concrete< I >::ret rotate (const Image< I > &input, double angle, const Ext &extension, const Site_Set< S > &output_domain)`
Perform a rotation from the center of an image.
- `template<typename I >`
`mln::trait::concrete< I >::ret rotate (const Image< I > &input, double angle)`
This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use `literal::zero` as default value for the extension.
- `template<typename B >`
`B rotate (const Box< B > &box_, double angle, const typename B::site &ref)`
Rotate a box.
- `template<typename I , typename N >`
`mln::trait::concrete< I >::ret seeds2tiling (const Image< I > &ima_, const Neighborhood< N > &nbh)`
Take a labeled image `ima_` with seeds and extend them until creating tiles.

- `template<typename I, typename V, typename Ext, typename S >`
`mln::trait::concrete< I >::ret translate (const Image< I > &input, const algebra::vec< I::site::dim, V > &ref, const Ext &extension, const Site_Set< S > &output_domain)`
Perform a translation from the center of an image.
- `template<typename I, typename V >`
`mln::trait::concrete< I >::ret translate (const Image< I > &input, const algebra::vec< I::site::dim, V > &ref)`
This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use [literal::zero](#) as default value for the extension.
- `template<typename I, typename V, typename Ext >`
`mln::trait::concrete< I >::ret translate (const Image< I > &input, const algebra::vec< I::site::dim, V > &ref, const Ext &extension)`
- `template<typename I >`
`mln::trait::concrete< I >::ret vertical_symmetry (const Image< I > &input)`
Performs a vertical symmetry.

9.63.1 Detailed Description

Namespace of all things related to geometry. Namespace of essential things related to geometry.

9.63.2 Function Documentation

9.63.2.1 `template<typename S > box< typename S::site > mln::geom::bbox (const Site_Set< S > & pset) [inline]`

Compute the precise bounding box of a point set `pset`.

Definition at line 122 of file `geom/bbox.hh`.

Referenced by `bbox()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::registration::icp()`, `max_col()`, `max_row()`, `max_sli()`, `min_col()`, `min_row()`, `min_sli()`, `mln::debug::println()`, `mln::debug::println_with_border()`, and `rotate()`.

9.63.2.2 `template<typename I > box< typename I::site > mln::geom::bbox (const Image< I > & ima)`

Compute the precise bounding box of a point set `pset`.

Definition at line 133 of file `geom/bbox.hh`.

References `bbox()`.

9.63.2.3 `template<typename W > box< typename W::psite > mln::geom::bbox (const Window< W > & win)`

Compute the precise bounding box of a window `win`.

Definition at line 143 of file `geom/bbox.hh`.

References `mln::literal::origin`.

9.63.2.4 `template<typename W> box< typename W::psite > mln::geom::bbox (const Weighted_Window< W> & win)`

Compute the precise bounding box of a weighted window `win`.

Definition at line 156 of file `geom/bbox.hh`.

References `bbox()`.

9.63.2.5 `template<typename I, typename W> mln::trait::ch_value< I, unsigned >::ret mln::geom::chamfer (const Image< I> & input_, const W & w_win_, unsigned max = mln_max(unsigned))`

Apply chamfer algorithm to a binary image.

Definition at line 113 of file `geom/chamfer.hh`.

9.63.2.6 `template<typename W> unsigned mln::geom::delta (const Window< W> & win)`

Compute the delta of a window `win`.

Definition at line 96 of file `delta.hh`.

Referenced by `mln::extension::adjust()`, and `delta()`.

9.63.2.7 `template<typename W> unsigned mln::geom::delta (const Weighted_Window< W> & wwin)`

Compute the delta of a weighted window `wwin`.

Definition at line 105 of file `delta.hh`.

References `delta()`.

9.63.2.8 `template<typename N> unsigned mln::geom::delta (const Neighborhood< N> & nbh)`

Compute the delta of a neighborhood `nbh`.

Definition at line 112 of file `delta.hh`.

References `delta()`.

9.63.2.9 `template<typename I> mln::trait::concrete< I>::ret mln::geom::horizontal_symmetry (const Image< I> & input)`

Performs a horizontal symmetry.

Definition at line 172 of file `horizontal_symmetry.hh`.

9.63.2.10 `template<typename I > I::site::coord mln::geom::max_col (const Image< I > & ima) [inline]`

Give the maximum column of an image.

Definition at line 56 of file max_col.hh.

References `bbox()`.

Referenced by `mln::io::magick::load()`, and `ncols()`.

9.63.2.11 `template<typename B > B::site::coord mln::geom::max_col (const Box< B > & b) [inline]`

Give the maximum col of an box 2d or 3d.

Definition at line 67 of file max_col.hh.

9.63.2.12 `template<typename I > I::site::coord mln::geom::max_ind (const Image< I > & ima) [inline]`

Give the maximum ind of an image.

Definition at line 51 of file max_ind.hh.

Referenced by `ninds()`.

9.63.2.13 `template<typename I > I::site::coord mln::geom::max_row (const Image< I > & ima) [inline]`

Give the maximum row of an image.

Definition at line 57 of file max_row.hh.

References `bbox()`.

Referenced by `mln::io::magick::load()`, and `nrows()`.

9.63.2.14 `template<typename B > B::site::coord mln::geom::max_row (const Box< B > & b) [inline]`

Give the maximum row of an box 2d or 3d.

Definition at line 68 of file max_row.hh.

9.63.2.15 `template<typename I > I::site::coord mln::geom::max_sli (const Image< I > & ima) [inline]`

Give the maximum sli of an image.

Definition at line 53 of file max_sli.hh.

References `bbox()`.

Referenced by `nslis()`.

9.63.2.16 `std::pair< complex_image< 2, mln::space_2complex_geometry, algebra::vec<3, float> >, complex_image< 2, mln::space_2complex_geometry, float > >`
`mln::geom::mesh_corner_point_area (const p_complex< 2, space_2complex_geometry > & mesh) [inline]`

Compute the area “belonging” to normals at vertices.

Inspired from the method `Trimesh::need_pointareas` of the Trimesh library.

See also

<http://www.cs.princeton.edu/gfx/proj/trimesh2/>

From the documentation of Trimesh:

“Compute the area "belonging" to each vertex or each corner of a triangle (defined as Voronoi area restricted to the 1-ring of a vertex, or to the triangle).”

Definition at line 249 of file `misc.hh`.

References `mln::data::fill()`, `mln::norm::sqr_l2()`, `mln::algebra::vprod()`, and `mln::literal::zero`.

Referenced by `mesh_curvature()`.

9.63.2.17 `std::pair< complex_image< 2, mln::space_2complex_geometry, float >, complex_image< 2, mln::space_2complex_geometry, float > >`
`mln::geom::mesh_curvature (const p_complex< 2, space_2complex_geometry > & mesh) [inline]`

Compute the principal curvatures of a surface at vertices.

These principal curvatures are names `kappa_1` and `kappa_2` in

Sylvie Philipp-Foliguet, Michel Jordan Laurent Najman and Jean Cousty. Artwork 3D Model Database Indexing and Classification.

Parameters

[in] *mesh* The surface (triangle mesh) on which the curvature is to be computed.

Definition at line 486 of file `misc.hh`.

References `mln::algebra::ldlt_decomp()`, `mln::algebra::ldlt_solve()`, `mesh_corner_point_area()`, `mesh_normal()`, `mln::algebra::vprod()`, and `mln::literal::zero`.

9.63.2.18 `complex_image< 2, mln::space_2complex_geometry, algebra::vec<3, float> >`
`mln::geom::mesh_normal (const p_complex< 2, space_2complex_geometry > & mesh) [inline]`

Compute normals at vertices.

Inspired from the method `Trimesh::need_normals` of the Trimesh library.

See also

<http://www.cs.princeton.edu/gfx/proj/trimesh2/>

For simplicity purpose, and contrary to Trimesh, this routine only compute normals from a mesh, not from a cloud of points.

Definition at line 161 of file misc.hh.

References mln::data::fill(), mln::norm::sqr_l2(), mln::algebra::vprod(), and mln::literal::zero.

Referenced by mesh_curvature().

9.63.2.19 `template<typename I > I::site::coord mln::geom::min_col (const Image< I > & ima)
[inline]`

Give the minimum column of an image.

Definition at line 57 of file min_col.hh.

References bbox().

Referenced by mln::transform::hough(), mln::io::magick::load(), and ncols().

9.63.2.20 `template<typename B > B::site::coord mln::geom::min_col (const Box< B > & b)
[inline]`

Give the minimum column of an box 2d or 3d.

Definition at line 68 of file min_col.hh.

9.63.2.21 `template<typename I > I::site::coord mln::geom::min_ind (const Image< I > & ima
) [inline]`

Give the minimum ind of an image.

Definition at line 51 of file min_ind.hh.

Referenced by ninds().

9.63.2.22 `template<typename I > I::site::coord mln::geom::min_row (const Image< I > & ima
) [inline]`

Give the minimum row of an image.

Definition at line 60 of file min_row.hh.

References bbox().

Referenced by mln::transform::hough(), mln::io::magick::load(), and nrows().

9.63.2.23 `template<typename B > B::site::coord mln::geom::min_row (const Box< B > & b)
[inline]`

Give the minimum row of an box 2d or 3d.

Definition at line 71 of file min_row.hh.

9.63.2.24 `template<typename I > I::site::coord mln::geom::min_sli (const Image< I > & ima)
[inline]`

Give the minimum sli of an image.

Definition at line 53 of file min_sli.hh.

References bbox().

Referenced by nslis().

9.63.2.25 `template<typename I > unsigned mln::geom::ncols (const Image< I > & ima)` `[inline]`

Give the number of columns of an image.

Definition at line 57 of file ncols.hh.

References max_col(), and min_col().

Referenced by mln::labeling::impl::compute_fastest(), mln::subsampling::gaussian_subsampling(), mln::transform::hough(), ncols(), mln::io::magick::save(), and mln::subsampling::subsampling().

9.63.2.26 `template<typename B > unsigned mln::geom::ncols (const Box< B > & b)`

Give the number of cols of a box 2d or 3d.

Definition at line 67 of file ncols.hh.

References max_col(), min_col(), and ncols().

9.63.2.27 `template<typename I > unsigned mln::geom::ninds (const Image< I > & ima)` `[inline]`

Give the number of inds of an image.

Definition at line 52 of file ninds.hh.

References max_ind(), and min_ind().

9.63.2.28 `template<typename I > unsigned mln::geom::nrows (const Image< I > & ima)` `[inline]`

Give the number of rows of an image.

Definition at line 57 of file nrows.hh.

References max_row(), and min_row().

Referenced by mln::subsampling::gaussian_subsampling(), mln::transform::hough(), nrows(), mln::io::magick::save(), and mln::subsampling::subsampling().

9.63.2.29 `template<typename B > unsigned mln::geom::nrows (const Box< B > & b)`

Give the number of rows of a box 2d or 3d.

Definition at line 66 of file nrows.hh.

References max_row(), min_row(), and nrows().

9.63.2.30 `template<typename I > unsigned mln::geom::nsites (const Image< I > & input) [inline]`

Compute the number of sites of the image `input`.

Definition at line 52 of file `nsites.hh`.

Referenced by `pmin_pmax()`.

9.63.2.31 `template<typename I > unsigned mln::geom::nslis (const Image< I > & ima) [inline]`

Give the number of slices of an image.

Definition at line 53 of file `nslis.hh`.

References `max_sli()`, and `min_sli()`.

9.63.2.32 `template<typename S > void mln::geom::pmin_pmax (const Site_Set< S > & s, typename S::site & pmin, typename S::site & pmax) [inline]`

Compute the minimum and maximum points, `pmin` and `max`, of point set `s`.

Definition at line 148 of file `pmin_pmax.hh`.

References `nsites()`.

9.63.2.33 `template<typename S > std::pair< typename S::site, typename S::site > mln::geom::pmin_pmax (const Site_Set< S > & s) [inline]`

Compute the minimum and maximum points of point set `s`.

Definition at line 157 of file `pmin_pmax.hh`.

References `nsites()`.

Referenced by `pmin_pmax()`.

9.63.2.34 `template<typename I > std::pair< typename I::site, typename I::site > mln::geom::pmin_pmax (const Site_Iterator< I > & p) [inline]`

Compute the minimum and maximum points when browsing with iterator `p`.

Definition at line 106 of file `pmin_pmax.hh`.

References `pmin_pmax()`.

9.63.2.35 `template<typename I > void mln::geom::pmin_pmax (const Site_Iterator< I > & p, typename I::site & pmin, typename I::site & pmax) [inline]`

Compute the minimum and maximum points, `pmin` and `max`, when browsing with iterator `p`.

Definition at line 84 of file `pmin_pmax.hh`.

9.63.2.36 `template<typename I , typename Ext > mln::trait::concrete< I >::ret
mln::geom::rotate (const Image< I > & input, double angle, const Ext & extension)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 185 of file rotate.hh.

References rotate().

9.63.2.37 `template<typename B > B mln::geom::rotate (const Box< B > & box, double angle)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. The rotation center `ref` is set to `box.pcenter()`.

Definition at line 252 of file rotate.hh.

References rotate().

9.63.2.38 `template<typename I , typename Ext , typename S > mln::trait::concrete< I >::ret
mln::geom::rotate (const Image< I > & input, double angle, const Ext & extension,
const Site_Set< S > & output_domain)`

Perform a rotation from the center of an image.

Parameters

[in] *input* An image.

[in] *angle* An angle in degrees.

[in] *extension* [Function](#), image or value which will be used as extension. This extension allows to map values to sites which where not part of the domain before the rotation.

[in] *output_domain* The domain of the output image. An invalid domain, causes the routine to use a domain large enough to display the whole original image.

Returns

An image with the same domain as `input`.

Definition at line 123 of file rotate.hh.

References `bbox()`, `mln::compose()`, `mln::duplicate()`, `mln::initialize()`, `mln::mln_exact()`, `mln::literal::origin`, and `mln::data::paste()`.

Referenced by rotate().

9.63.2.39 `template<typename I > mln::trait::concrete< I >::ret mln::geom::rotate (const
Image< I > & input, double angle)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use `literal::zero` as default value for the extension.

Definition at line 197 of file rotate.hh.

References rotate(), and `mln::literal::zero`.

9.63.2.40 `template<typename B > B mln::geom::rotate (const Box< B > & box_, double angle, const typename B::site & ref)`

Rotate a box.

FIXME: the return type may be too generic and may lead to invalid covariance.

Definition at line 205 of file rotate.hh.

References `mln::compose()`, `mln::literal::origin`, and `mln::accu::shape::bbox< P >::to_result()`.

9.63.2.41 `template<typename I , typename N > mln::trait::concrete< I >::ret mln::geom::seeds2tiling (const Image< I > & ima_, const Neighborhood< N > & nbh) [inline]`

Take a labeled image `ima_` with seeds and extend them until creating tiles.

Parameters

[in, out] *ima_* The labeled image with seed.

[in] *nbh* The neighborhood to use on this algorithm.

Returns

A tiled image.

Precondition

`ima_` has to be initialized.

Definition at line 136 of file seeds2tiling.hh.

References `mln::geom::impl::seeds2tiling()`.

9.63.2.42 `template<typename I , typename V , typename Ext , typename S > mln::trait::concrete< I >::ret mln::geom::translate (const Image< I > & input, const algebra::vec< I::site::dim, V > & ref, const Ext & extension, const Site_Set< S > & output_domain)`

Perform a translation from the center of an image.

Parameters

[in] *input* An image.

[in] *ref* The translation vector.

[in] *extension* [Function](#), image or value which will be used as extension. This extension allows to map values to sites which where not part of the domain before the translation.

[in] *output_domain* The domain of the output image. An invalid domain, causes the routine to use the translated `input_domain`.

Returns

An image with the same domain as `input`.

Definition at line 99 of file translate.hh.

References `mln::extend()`, `mln::data::fill()`, and `mln::mln_exact()`.

Referenced by `translate()`.

9.63.2.43 `template<typename I , typename V > mln::trait::concrete< I >::ret mln::geom::translate (const Image< I > & input, const algebra::vec< I::site::dim, V > & ref)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use [literal::zero](#) as default value for the extension.

Definition at line 146 of file `translate.hh`.

References `translate()`, and `mln::literal::zero`.

9.63.2.44 `template<typename I , typename V , typename Ext > mln::trait::concrete< I >::ret mln::geom::translate (const Image< I > & input, const algebra::vec< I::site::dim, V > & ref, const Ext & extension)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 132 of file `translate.hh`.

References `translate()`.

9.63.2.45 `template<typename I > mln::trait::concrete< I >::ret mln::geom::vertical_symmetry (const Image< I > & input)`

Performs a vertical symmetry.

Definition at line 174 of file `vertical_symmetry.hh`.

9.64 mln::geom::impl Namespace Reference

Implementation namespace of `geom` namespace.

Functions

- `template<typename I , typename N > mln::trait::concrete< I >::ret seeds2tiling (const Image< I > &ima_, const Neighborhood< N > &nbh_)`

Generic implementation of `geom::seed2tiling`.

9.64.1 Detailed Description

Implementation namespace of `geom` namespace.

9.64.2 Function Documentation

9.64.2.1 `template<typename I , typename N > mln::trait::concrete< I >::ret
mln::geom::impl::seeds2tiling (const Image< I > & ima_ , const Neighborhood< N > &
nbh_) [inline]`

Generic implementation of geom::seed2tiling.

Parameters

[in, out] *ima_* The labeled image with seed.

[in] *nbh_* The neighborhood to use on this algorithm.

Definition at line 77 of file seeds2tiling.hh.

References mln::duplicate(), mln::p_queue< P >::front(), mln::p_queue< P >::pop(), and mln::p_queue< P >::push().

Referenced by mln::geom::seeds2tiling().

9.65 mln::graph Namespace Reference

Namespace of graph related routines.

Functions

- `template<typename G , typename F >
F::result compute (const Graph< G > &g_ , F &functor)`
Base routine to compute attributes on a graph.
- `template<typename I , typename N , typename L >
mln::trait::ch_value< I , L >::ret labeling (const Image< I > &graph_image_ , const Neighborhood<
N > &nbh_ , L &nlabels)`
Label graph components.
- `template<typename I , typename M >
graph_elt_neighborhood_if< mln_graph(I) , typename I::domain_t , M > to_neighb (const Image< I
> &graph_image_ , const Image< M > &graph_mask_image_)`
Make a custom graph neighborhood from a mask image.
- `template<typename I , typename M >
graph_elt_window_if< mln_graph(I) , typename I::domain_t , M > to_win (const Image< I >
&graph_image_ , const Image< M > &graph_mask_image_)`
Make a custom graph window from a mask image.

9.65.1 Detailed Description

Namespace of graph related routines.

9.65.2 Function Documentation

9.65.2.1 `template<typename G , typename F > F::result mln::graph::compute (const Graph< G > & g_, F & functor)`

Base routine to compute attributes on a graph.

Parameters

- [in] `g_` A graph.
- [in] `functor` A functor implementing the right interface.

Returns

The computed data.

See also

`canvas::browsing::depth_first_search`

Definition at line 63 of file `graph/compute.hh`.

9.65.2.2 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret mln::graph::labeling (const Image< I > & graph_image_, const Neighborhood< N > & nbh_, L & nlabels)`

Label graph components.

[Vertex](#) with id 0, usually used to represent the background component, will be labeled with an id different from 0. Therefore, the labeling starts from 1.

Parameters

- [in] `graph_image_` A graph image (

See also

[vertex_image](#), [edge_image](#)).

Parameters

- [in] `nbh_` A graph neighborhood.
- [in, out] `nlabels` The number of labels found.

Returns

a [Graph](#) image of labels.

Definition at line 72 of file `labeling.hh`.

References `mln::labeling::blobs()`, `mln::data::fill()`, and `mln::initialize()`.

9.65.2.3 `template<typename I , typename M > graph_elt_neighborhood_if< mln_graph(I), typename I::domain_t, M > mln::graph::to_neighb (const Image< I > & graph_image_, const Image< M > & graph_mask_image_)`

Make a custom graph neighborhood from a mask image.

Parameters

[in] *graph_image_* A graph image (

See also

[vertex_image](#) and [edge_image](#)).

Parameters

[in] *graph_mask_image_* A graph image of bool used as a mask.

Returns

A masked neighborhood on graph.

Definition at line 57 of file to_neighb.hh.

9.65.2.4 `template<typename I, typename M > graph_elt_window_if< mln_graph(I), typename I::domain_t, M > mln::graph::to_win (const Image< I > & graph_image_, const Image< M > & graph_mask_image_)`

Make a custom graph window from a mask image.

Parameters

[in] *graph_image_* A graph image (

See also

[vertex_image](#) and [edge_image](#)).

Parameters

[in] *graph_mask_image_* A graph image of bool used as a mask.

Returns

A masked window on graph.

Definition at line 57 of file to_win.hh.

9.66 mln::grid Namespace Reference

Namespace of grids definitions.

9.66.1 Detailed Description

Namespace of grids definitions. Compute the image::space trait from a point type.

9.67 mln::histo Namespace Reference

Namespace of histograms.

Namespaces

- namespace [impl](#)
Implementation namespace of histo namespace.

Classes

- struct [array](#)
Generic histogram class over a value set with type T.

Functions

- `template<typename I > histo::array< typename I::value > compute (const Image< I > &input)`
Compute the histogram of image input.
- `template<typename I > mln::trait::concrete< I >::ret equalize (const Image< I > &input)`
Equalizes the histogram of image input.

9.67.1 Detailed Description

Namespace of histograms.

9.67.2 Function Documentation

9.67.2.1 `template<typename I > histo::array< typename I::value > mln::histo::compute (const Image< I > & input) [inline]`

Compute the histogram of image `input`.

Definition at line 79 of file `histo/compute.hh`.

Referenced by `equalize()`.

9.67.2.2 `template<typename I > mln::trait::concrete< I >::ret mln::histo::equalize (const Image< I > & input)`

Equalizes the histogram of image `input`.

Author

J. Fabrizio, R. Levillain

Definition at line 55 of file `histo/equalize.hh`.

References `compute()`, and `mln::initialize()`.

9.68 mln::histo::impl Namespace Reference

Implementation namespace of histo namespace.

Namespaces

- namespace [generic](#)
Generic implementation namespace of histo namespace.

9.68.1 Detailed Description

Implementation namespace of histo namespace.

9.69 mln::histo::impl::generic Namespace Reference

Generic implementation namespace of histo namespace.

9.69.1 Detailed Description

Generic implementation namespace of histo namespace.

9.70 mln::impl Namespace Reference

Implementation namespace of mln namespace.

9.70.1 Detailed Description

Implementation namespace of mln namespace.

9.71 mln::io Namespace Reference

Namespace of input/output handling.

Namespaces

- namespace [cloud](#)
Namespace of cloud input/output handling.
- namespace [dicom](#)
Namespace of DICOM input/output handling.
- namespace [dump](#)

Namespace of dump input/output handling.

- namespace [fits](#)
Namespace of fits input/output handling.
- namespace [fld](#)
Namespace of pgm input/output handling.
- namespace [magick](#)
Namespace of magick input/output handling.
- namespace [off](#)
Namespace of off input/output handling.
- namespace [pbm](#)
Namespace of pbm input/output handling.
- namespace [pbms](#)
Namespace of pbms input/output handling.
- namespace [pfm](#)
Namespace of pfm input/output handling.
- namespace [pgm](#)
Namespace of pgm input/output handling.
- namespace [pgms](#)
Namespace of pgms input/output handling.
- namespace [plot](#)
Namespace of plot input/output handling.
- namespace [pnm](#)
Namespace of pnm input/output handling.
- namespace [pnms](#)
Namespace of pnms input/output handling.
- namespace [ppm](#)
Namespace of ppm input/output handling.
- namespace [ppms](#)
Namespace of ppms input/output handling.
- namespace [raw](#)
Namespace of raw input/output handling.
- namespace [tiff](#)
Namespace of tiff input/output handling.

- namespace `txt`

Namespace of txt input/output handling.

9.71.1 Detailed Description

Namespace of input/output handling.

9.72 mln::io::cloud Namespace Reference

Namespace of cloud input/output handling.

Functions

- `template<typename P > void load (p_array< P > &arr, const std::string &filename)`
Load a cloud of points.
- `template<typename P > void save (const p_array< P > &arr, const std::string &filename)`
Load a cloud of points.

9.72.1 Detailed Description

Namespace of cloud input/output handling.

9.72.2 Function Documentation

9.72.2.1 `template<typename P > void mln::io::cloud::load (p_array< P > & arr, const std::string & filename)`

Load a cloud of points.

Parameters

[in, out] *arr* the site set where to load the data.

[in] *filename* file to load.

Definition at line 88 of file cloud/load.hh.

9.72.2.2 `template<typename P > void mln::io::cloud::save (const p_array< P > & arr, const std::string & filename)`

Load a cloud of points.

Parameters

- [in] *arr* the cloud of points to save.
- [in] *filename* the destination.

Definition at line 83 of file cloud/save.hh.

9.73 mln::io::dicom Namespace Reference

Namespace of DICOM input/output handling.

Classes

- struct [dicom_header](#)
Store dicom file header.

Functions

- [dicom_header get_header](#) (const std::string &filename)
Retrieve header in a dicom file.
- template<typename I >
void [load](#) ([Image](#)< I > &ima, const std::string &filename)

9.73.1 Detailed Description

Namespace of DICOM input/output handling.

9.73.2 Function Documentation**9.73.2.1 dicom_header mln::io::dicom::get_header (const std::string & filename)**

Retrieve header in a dicom file.

Definition at line 76 of file dicom/get_header.hh.

References [mln::util::array< T >::append\(\)](#).

9.73.2.2 template<typename I > void mln::io::dicom::load (Image< I > & ima, const std::string & filename) [inline]

Load a DICOM file in a Milena image.

Parameters

- [out] *ima* A reference to the image which will receive data.
- [in] *filename* The source.

Common compilation flags to link to gdcem if this file is used:

```
-lgdcmCommon -lgdcmDICT -lgdcmDSED -lgdcmIOD -lgdcmMSFF -lgdcmexpat -lgdcmjpeg12 -  
-lgdcmjpeg16 -lgdcmjpeg8 -lgdcmopenjpeg -lgdcmuuid -lgdcmzlib
```

Definition at line 96 of file dicom/load.hh.

References mln::initialize().

9.74 mln::io::dump Namespace Reference

Namespace of dump input/output handling.

Classes

- struct [dump_header](#)
Store dump file header.

Functions

- [dump_header get_header](#) (const std::string &filename)
Retrieve header in a dump file.
- [template<typename I >
void load](#) ([Image](#)< I > &ima_, const std::string &filename)
Load a Milena image by dumped into a file.
- [template<typename I >
void save](#) (const [Image](#)< I > &ima_, const std::string &filename)
Save a Milena image by dumping its data to a file.

9.74.1 Detailed Description

Namespace of dump input/output handling.

9.74.2 Function Documentation

9.74.2.1 [dump_header mln::io::dump::get_header](#) ([const std::string & filename](#))

Retrieve header in a dump file.

Definition at line 68 of file dump/get_header.hh.

References mln::util::array< T >::resize().

9.74.2.2 `template<typename I > void mln::io::dump::load (Image< I > & ima_, const std::string & filename)`

Load a Milena image by dumped into a file.

Parameters

[in, out] *ima_* The image to load.

[in] *filename* the destination.

Definition at line 171 of file dump/load.hh.

9.74.2.3 `template<typename I > void mln::io::dump::save (const Image< I > & ima_, const std::string & filename)`

Save a Milena image by dumping its data to a file.

Parameters

[in] *ima_* The image to save.

[in] *filename* the destination.

Definition at line 131 of file dump/save.hh.

9.75 `mln::io::fits` Namespace Reference

Namespace of fits input/output handling.

Functions

- void [load](#) ([image2d](#)< float > &ima, const std::string &filename)
Load a fits image in a Milena image.
- [image2d](#)< float > [load](#) (const std::string &filename)
Load a fits image in a [image2d](#)<float>.

9.75.1 Detailed Description

Namespace of fits input/output handling.

9.75.2 Function Documentation

9.75.2.1 `void mln::io::fits::load (image2d< float > & ima, const std::string & filename)` `[inline]`

Load a fits image in a Milena image.

Parameters

- [out] *ima* A reference to the image2d<float> which will receive data.
- [in] *filename* The source.

Definition at line 132 of file fits/load.hh.

9.75.2.2 image2d<float> mln::io::fits::load (const std::string & filename) [inline]

Load a fits image in a image2d<float>.

Parameters

- [in] *filename* The image source.

Returns

An image2d<float> which contains loaded data.

Definition at line 85 of file fits/load.hh.

9.76 mln::io::fld Namespace Reference

Namespace of pgm input/output handling.

Classes

- struct [fld_header](#)
Define the header structure of an AVS field data file.

Functions

- template<typename I >
void [load](#) (Image< I > &ima_, const char *filename)
Load an image from an AVS field file.
- [fld_header read_header](#) (std::istream &ins)
Read the header form an AVS field file.
- void [write_header](#) (std::ostream &file, const [fld_header](#) &h)
Write the AVS header in a file.

9.76.1 Detailed Description

Namespace of pgm input/output handling.

9.76.2 Function Documentation

9.76.2.1 `template<typename I> void mln::io::fld::load (Image< I > & ima_, const char * filename) [inline]`

Load an image from an AVS field file.

Parameters

- `[in, out]` *ima_* The image to load.
- `[in]` *filename* The path to the AVS file.

Definition at line 198 of file fld/load.hh.

References `mln::box< P >::pmax()`, `mln::box< P >::pmin()`, and `read_header()`.

9.76.2.2 `fld_header mln::io::fld::read_header (std::istream & ins) [inline]`

Read the header form an AVS field file.

Parameters

- ins* The file to read.

Returns

- The header.

Definition at line 73 of file fld/load_header.hh.

Referenced by `load()`.

9.76.2.3 `void mln::io::fld::write_header (std::ostream & file, const fld_header & h) [inline]`

Write the AVS header in a file.

Parameters

- file* The file to write.
- h* The AVS header.

Definition at line 58 of file write_header.hh.

9.77 `mln::io::magick` Namespace Reference

Namespace of magick input/output handling.

Functions

- `template<typename I> void load (Image< I > &ima, const std::string &filename)`

Load data from a file into a Milena image using Magick++.

- `template<typename I, typename J >`
`void save (const Image< I > &ima, const Image< J > &opacity_mask, const std::string &filename)`

Save a Milena image into a file using Magick++.

- `template<typename I >`
`void save (const Image< I > &ima, const std::string &filename)`

Save a Milena image into a file using Magick++.

9.77.1 Detailed Description

Namespace of magick input/output handling.

9.77.2 Function Documentation

9.77.2.1 `template<typename I > void mln::io::magick::load (Image< I > & ima, const std::string & filename) [inline]`

Load data from a file into a Milena image using Magick++.

Parameters

[out] *ima* The image data are loaded into.

[in] *filename* The name of the input file.

Definition at line 139 of file magick/load.hh.

References `mln::initialize()`, `mln::geom::max_col()`, `mln::geom::max_row()`, `mln::geom::min_col()`, and `mln::geom::min_row()`.

9.77.2.2 `template<typename I, typename J > void mln::io::magick::save (const Image< I > & ima, const Image< J > & opacity_mask, const std::string & filename)`

Save a Milena image into a file using Magick++.

Parameters

[in] *ima* The image to save.

[in] *opacity_mask* Mask used to set pixel opacity_mask in output image. Output format must support this feature to be taken into account.

[in] *filename* The name of the output file.

Definition at line 228 of file magick/save.hh.

References `mln::geom::ncols()`, and `mln::geom::nrows()`.

9.77.2.3 `template<typename I> void mln::io::magick::save (const Image< I > & ima, const std::string & filename) [inline]`

Save a Milena image into a file using Magick++.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

- [in] *ima* The image to save.
- [in] *filename* The name of the output file.

Definition at line 288 of file magick/save.hh.

9.78 `mln::io::off` Namespace Reference

Namespace of off input/output handling.

Functions

- void `load (bin_2complex_image3df &ima, const std::string &filename)`
Load a (binary) OFF image into a complex image.
- void `save (const bin_2complex_image3df &ima, const std::string &filename)`
Save a (binary) OFF image into a complex image.
- `template<typename I>`
void `save_bin_alt (const I &ima, const std::string &filename)`
FIXME: Similar to `mln::io::off::save(const bin_2complex_image3df&, const std::string&)`, but does not save faces whose value is 'false'.

9.78.1 Detailed Description

Namespace of off input/output handling.

9.78.2 Function Documentation

9.78.2.1 `void mln::io::off::load (bin_2complex_image3df & ima, const std::string & filename)`

Load a (binary) OFF image into a complex image.

Load a 3x8-bit RGB (color) OFF image into a complex image.

Load a floating-point OFF image into a complex image.

Parameters

- [out] *ima* A reference to the image to construct.
- [in] *filename* The name of the file to load.

The image is said binary since data only represent the existence of faces.

Parameters

[out] *ima* A reference to the image to construct.

[in] *filename* The name of the file to load.

Read floating-point data is attached to 2-faces only; 1-faces and 0-faces are set to 0.0f.

Definition at line 195 of file off/load.hh.

9.78.2.2 void mln::io::off::save (const bin_2complex_image3df & *ima*, const std::string & *filename*)

Save a (binary) OFF image into a complex image.

Save a 3x8-bit RGB (color) OFF image into a complex image.

Save a floating-point value grey-level OFF image into a complex image.

Save an 8-bit grey-level OFF image into a complex image.

Parameters

[in] *ima* The image to save.

[in] *filename* The name of the file where to save the image.

The image is said binary since data represent only the existence of faces.

Parameters

[in] *ima* The image to save.

[in] *filename* The name of the file where to save the image.

Only data is attached to 2-faces is saved; the OFF file cannot store data attached to faces of other dimensions.

Definition at line 189 of file off/save.hh.

9.78.2.3 template<typename I> void mln::io::off::save_bin_alt (const I & *ima*, const std::string & *filename*)

FIXME: Similar to [mln::io::off::save\(const bin_2complex_image3df&, const std::string&\)](#), but does not save faces whose value is 'false'.

Definition at line 59 of file save_bin_alt.hh.

9.79 mln::io::pbm Namespace Reference

Namespace of pbm input/output handling.

Namespaces

- namespace [impl](#)
Namespace of pbm implementation details.

Functions

- void [load](#) ([image2d](#)< bool > &ima, const std::string &filename)
Load a pbm image in a Milena image.
- [image2d](#)< bool > [load](#) (const std::string &filename)
Load a pbm image in a image2d<float>.
- template<typename I >
void [save](#) (const [Image](#)< I > &ima, const std::string &filename)

9.79.1 Detailed Description

Namespace of pbm input/output handling.

9.79.2 Function Documentation

9.79.2.1 void mln::io::pbm::load ([image2d](#)< bool > & *ima*, const std::string & *filename*) [\[inline\]](#)

Load a pbm image in a Milena image.

Parameters

- [out] *ima* A reference to the [image2d](#)<bool> which will receive data.
[in] *filename* The source.

Definition at line 156 of file pbm/load.hh.

9.79.2.2 [image2d](#)< bool > mln::io::pbm::load (const std::string & *filename*) [\[inline\]](#)

Load a pbm image in a [image2d](#)<float>.

Parameters

- [in] *filename* The image source.

Returns

An [image2d](#)<float> which contains loaded data.

Definition at line 128 of file pbm/load.hh.

9.79.2.3 `template<typename I> void mln::io::pbm::save (const Image< I > & ima, const std::string & filename) [inline]`

Save a Milena image as a pbm image.

Parameters

[in] *ima* The image to save.

[in, out] *filename* the destination.

Definition at line 118 of file pbm/save.hh.

9.80 mln::io::pbm::impl Namespace Reference

Namespace of pbm implementation details.

9.80.1 Detailed Description

Namespace of pbm implementation details.

9.81 mln::io::pbms Namespace Reference

Namespace of pbms input/output handling.

Namespaces

- namespace [impl](#)
Namespace of pbms implementation details.

Functions

- void [load](#) ([image3d](#)< bool > &ima, const [util::array](#)< std::string > &filenames)
Load pbms images as slices of a 3D Milena image.

9.81.1 Detailed Description

Namespace of pbms input/output handling.

9.81.2 Function Documentation

9.81.2.1 `void mln::io::pbms::load (image3d< bool > & ima, const util::array< std::string > & filenames) [inline]`

Load pbms images as slices of a 3D Milena image.

Parameters

[out] *ima* A reference to the 3D image which will receive data.

[in] *filenames* The list of 2D images to load..

Definition at line 65 of file pbms/load.hh.

9.82 mln::io::pbms::impl Namespace Reference

Namespace of pbms implementation details.

9.82.1 Detailed Description

Namespace of pbms implementation details.

9.83 mln::io::pfm Namespace Reference

Namespace of pfm input/output handling.

Namespaces

- namespace [impl](#)

Implementation namespace of pfm namespace.

Functions

- void [load](#) ([image2d](#)< float > &ima, const std::string &filename)

Load a pfm image in a Milena image.

- [image2d](#)< float > [load](#) (const std::string &filename)

Load a pfm image in a image2d<float>.

- template<typename I >

void [save](#) (const [Image](#)< I > &ima, const std::string &filename)

Save a Milena image as a pfm image.

9.83.1 Detailed Description

Namespace of pfm input/output handling.

9.83.2 Function Documentation

9.83.2.1 `void mln::io::pfm::load (image2d<float> & ima, const std::string & filename)`
`[inline]`

Load a pfm image in a Milena image.

Parameters

[out] *ima* A reference to the image2d<float> which will receive data.

[in] *filename* The source.

Definition at line 162 of file pfm/load.hh.

9.83.2.2 `image2d<float> mln::io::pfm::load (const std::string & filename)` `[inline]`

Load a pfm image in a image2d<float>.

Parameters

[in] *filename* The image source.

Returns

An image2d<float> which contains loaded data.

Definition at line 138 of file pfm/load.hh.

9.83.2.3 `template<typename I> void mln::io::pfm::save (const Image<I> & ima, const std::string & filename)` `[inline]`

Save a Milena image as a pfm image.

Parameters

[in] *ima* The image to save.

[in, out] *filename* the destination.

Definition at line 101 of file pfm/save.hh.

9.84 mln::io::pfm::impl Namespace Reference

Implementation namespace of pfm namespace.

9.84.1 Detailed Description

Implementation namespace of pfm namespace.

9.85 mln::io::pgm Namespace Reference

Namespace of pgm input/output handling.

Functions

- `template<typename I >`
`void load (Image< I > &ima, const std::string &filename)`
Load a pgm image in a Milena image.
- `template<typename V >`
`image2d< V > load (const std::string &filename)`
Load a pgm image in a Milena image.
- `template<typename I >`
`void save (const Image< I > &ima, const std::string &filename)`

9.85.1 Detailed Description

Namespace of pgm input/output handling.

9.85.2 Function Documentation

9.85.2.1 `template<typename I > void mln::io::pgm::load (Image< I > & ima, const std::string & filename) [inline]`

Load a pgm image in a Milena image.

Parameters

- [out] *ima* A reference to the image which will receive data.
 [in] *filename* The source.

Definition at line 87 of file `pgm/load.hh`.

9.85.2.2 `template<typename V > image2d< V > mln::io::pgm::load (const std::string & filename) [inline]`

Load a pgm image in a Milena image.

To use this routine, you should specialize the template with the value type of the image loaded. (ex : `load<value::int_u8>("...")`)

Parameters

- [in] *filename* The image source.

Returns

An `image2d` which contains loaded data.

Definition at line 77 of file `pgm/load.hh`.

9.85.2.3 `template<typename I > void mln::io::pgms::save (const Image< I > & ima, const std::string & filename) [inline]`

Save a Milena image as a pgm image.

Parameters

- [in] *ima* The image to save.
- [in, out] *filename* the destination.

Definition at line 77 of file pgm/save.hh.

9.86 mln::io::pgms Namespace Reference

Namespace of pgms input/output handling.

Functions

- `template<typename V > void load (image3d< V > &ima, const util::array< std::string > &filenames)`
Load pgm images as slices of a 3D Milena image.

9.86.1 Detailed Description

Namespace of pgms input/output handling.

9.86.2 Function Documentation

9.86.2.1 `template<typename V > void mln::io::pgms::load (image3d< V > & ima, const util::array< std::string > & filenames) [inline]`

Load pgm images as slices of a 3D Milena image.

Parameters

- [out] *ima* A reference to the 3D image which will receive data.
- [in] *filenames* The list of 2D images to load..

Definition at line 69 of file pgms/load.hh.

9.87 mln::io::plot Namespace Reference

Namespace of plot input/output handling.

Functions

- `template<typename I >`
`void load (util::array< I > &arr, const std::string &filename)`
- `template<typename T >`
`void save (const histo::array< T > &arr, const std::string &filename)`
- `template<typename T >`
`void save (const util::array< T > &arr, const std::string &filename, int start_value=0)`
Save a Milena array in a plot file.
- `template<typename I >`
`void save (const image1d< I > &ima, const std::string &filename)`
Save a Milena 1D image in a plot file.

9.87.1 Detailed Description

Namespace of plot input/output handling.

9.87.2 Function Documentation

9.87.2.1 `template<typename I > void mln::io::plot::load (util::array< I > & arr, const std::string & filename) [inline]`

Load a Milena 1D image from a plot file.

Parameters

- [in] *ima* A reference to the image to load.
- [out] *filename* The output file.
- [in] *start_value* The start index value of the plot (optional).

Load a Milena array from a plot file.

Parameters

- [in] *arr* A reference to the array to load.
- [out] *filename* The output file.

Definition at line 93 of file plot/load.hh.

References `mln::util::array< T >::append()`, and `mln::util::array< T >::clear()`.

9.87.2.2 `template<typename T > void mln::io::plot::save (const histo::array< T > & arr, const std::string & filename) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 103 of file plot/save.hh.

9.87.2.3 `template<typename T > void mln::io::plot::save (const util::array< T > & arr, const std::string & filename, int start_value = 0) [inline]`

Save a Milena array in a plot file.

Parameters

- [in] *arr* A reference to the array to save.
- [out] *filename* The output file.
- [in] *start_value* The start index value of the plot (optional).

Definition at line 89 of file plot/save.hh.

References `mln::util::array< T >::nelements()`.

9.87.2.4 `template<typename I > void mln::io::plot::save (const image1d< I > & ima, const std::string & filename)`

Save a Milena 1D image in a plot file.

Parameters

- [in] *ima* A reference to the image to save.
- [out] *filename* The output file.

9.88 mln::io::pnm Namespace Reference

Namespace of pnm input/output handling.

Namespaces

- namespace [impl](#)
Namespace of pnm's implementation details.

Functions

- `template<typename V > image2d< V > load (char type_, const std::string &filename)`
main function : load pnm format
- `template<typename I > void load (char type_, Image< I > &ima_, const std::string &filename)`
An other way to load pnm files : the destination is an argument to check if the type match the file to load.
- `template<typename I > void load_ascii_builtin (std::ifstream &file, I &ima)`
load_ascii for builtin value types.

- `template<typename I >`
`void load_ascii_value (std::ifstream &file, I &ima)`
load_ascii for Milena value types.
- `template<typename I >`
`void load_raw_2d (std::ifstream &file, I &ima)`
load_raw_2d.
- `template<typename V >`
`unsigned int max_component (const V &)`
Give the maximum value which can be stored as a component value type V.
- `template<typename I >`
`void save (char type, const Image< I > &ima_, const std::string &filename)`

9.88.1 Detailed Description

Namespace of pnm input/output handling.

9.88.2 Function Documentation

9.88.2.1 `template<typename V > image2d<V> mln::io::pnm::load (char type_, const std::string & filename) [inline]`

main function : load pnm format

Definition at line 210 of file pnm/load.hh.

References `load_raw_2d()`, and `max_component()`.

9.88.2.2 `template<typename I > void mln::io::pnm::load (char type_, Image< I > & ima_, const std::string & filename) [inline]`

An other way to load pnm files : the destination is an argument to check if the type match the file to load.

Definition at line 257 of file pnm/load.hh.

References `mln::make::box2d()`, `load_raw_2d()`, and `max_component()`.

9.88.2.3 `template<typename I > void mln::io::pnm::load_ascii_builtin (std::ifstream & file, I & ima) [inline]`

`load_ascii` for builtin value types.

Definition at line 178 of file pnm/load.hh.

9.88.2.4 `template<typename I > void mln::io::pnm::load_ascii_value (std::ifstream & file, I & ima) [inline]`

`load_ascii` for Milena value types.

Definition at line 164 of file pnm/load.hh.

9.88.2.5 `template<typename I > void mln::io::pnm::load_raw_2d (std::ifstream & file, I & ima) [inline]`

load_raw_2d.

for all pnm 8/16 bits formats

Definition at line 198 of file pnm/load.hh.

Referenced by load().

9.88.2.6 `template<typename V > unsigned int mln::io::pnm::max_component (const V &) [inline]`

Give the maximum value which can be stored as a component value type V.

Definition at line 56 of file max_component.hh.

Referenced by load().

9.88.2.7 `template<typename I > void mln::io::pnm::save (char type, const Image<I > & ima_, const std::string & filename) [inline]`

Save a Milena image as a pnm image.

Parameters

[in] *type* The type of the image to save (can be PPM, PGM, PBM).

[in] *ima_* The image to save.

[in, out] *filename* the destination.

Definition at line 185 of file pnm/save.hh.

9.89 mln::io::pnm::impl Namespace Reference

Namespace of pnm's implementation details.

9.89.1 Detailed Description

Namespace of pnm's implementation details.

9.90 mln::io::pnms Namespace Reference

Namespace of pnms input/output handling.

Functions

- `template<typename V > void load (char type, image3d< V > &ima, const util::array< std::string > &filenames)`

Load pnm images as slices of a 3D Milena image.

- void `load` (char type, `image3d`< bool > &ima, const `util::array`< std::string > &filenames)

9.90.1 Detailed Description

Namespace of pnms input/output handling.

9.90.2 Function Documentation

9.90.2.1 `template<typename V > void mln::io::pnms::load (char type, image3d< V > & ima, const util::array< std::string > & filenames) [inline]`

Load pnm images as slices of a 3D Milena image.

Parameters

- [in] *type* The type of the pnm files.
- [out] *ima* A reference to the 3D image which will receive data.
- [in] *filenames* The list of 2D images to load..

Definition at line 79 of file pnms/load.hh.

References `mln::make::image3d()`, `mln::util::array< T >::is_empty()`, and `mln::util::array< T >::nelements()`.

Referenced by `load()`.

9.90.2.2 `void mln::io::pnms::load (char type, image3d< bool > & ima, const util::array< std::string > & filenames) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 102 of file pnms/load.hh.

References `mln::make::image3d()`, `mln::util::array< T >::is_empty()`, `load()`, and `mln::util::array< T >::nelements()`.

9.91 mln::io::ppm Namespace Reference

Namespace of ppm input/output handling.

Functions

- `template<typename I > void load (Image< I > &ima, const std::string &filename)`

Load a ppm image in a Milena image.

- `template<typename V >`
`image2d< V > load` (const std::string &filename)
Load a ppm image in a Milena image.
- `template<typename I >`
`void save` (const `Image< I >` &ima, const std::string &filename)

9.91.1 Detailed Description

Namespace of ppm input/output handling.

9.91.2 Function Documentation

9.91.2.1 `template<typename I > void mln::io::ppm::load (Image< I > & ima, const std::string & filename) [inline]`

Load a ppm image in a Milena image.

Parameters

- [out] *ima* A reference to the image which will receive data.
- [in] *filename* The source.

Definition at line 89 of file ppm/load.hh.

9.91.2.2 `template<typename V > image2d< V > mln::io::ppm::load (const std::string & filename) [inline]`

Load a ppm image in a Milena image.

To use this routine, you should specialize the template with the value type of the image loaded. (ex : `load<value::int_u8>("...")`)

Parameters

- [in] *filename* The image source.

Returns

An `image2d` which contains loaded data.

Definition at line 79 of file ppm/load.hh.

9.91.2.3 `template<typename I > void mln::io::ppm::save (const Image< I > & ima, const std::string & filename) [inline]`

Save a Milena image as a ppm image.

Parameters

- [in] *ima* The image to save.

[in, out] *filename* the destination.

Definition at line 65 of file ppm/save.hh.

Referenced by mln::registration::icp().

9.92 mln::io::ppms Namespace Reference

Namespace of ppms input/output handling.

Functions

- `template<typename V > void load (image3d< V > &ima, const util::array< std::string > &filenames)`
Load ppm images as slices of a 3D Milena image.

9.92.1 Detailed Description

Namespace of ppms input/output handling.

9.92.2 Function Documentation

9.92.2.1 `template<typename V > void mln::io::ppms::load (image3d< V > & ima, const util::array< std::string > & filenames) [inline]`

Load ppm images as slices of a 3D Milena image.

Parameters

- [out] *ima* A reference to the 3D image which will receive data.
 [in] *filenames* The list of 2D images to load..

Definition at line 67 of file ppm/load.hh.

9.93 mln::io::raw Namespace Reference

Namespace of raw input/output handling.

Classes

- struct `raw_header`
Store raw file header.

Functions

- [raw_header get_header](#) (const std::string &filename)
Retrieve header in a raw file.
- `template<typename I >`
`void load (Image< I > &ima_, const std::string &filename)`
Load an image saved as a raw data file.
- `template<typename I >`
`void save (const Image< I > &ima_, const std::string &filename)`
Save a Milena image as a raw data file.

9.93.1 Detailed Description

Namespace of raw input/output handling.

9.93.2 Function Documentation

9.93.2.1 raw_header mln::io::raw::get_header (const std::string & filename)

Retrieve header in a raw file.

Definition at line 68 of file raw/get_header.hh.

References `mln::util::array< T >::resize()`.

9.93.2.2 `template<typename I > void mln::io::raw::load (Image< I > & ima_, const std::string & filename)`

Load an image saved as a raw data file.

Parameters

[in, out] *ima_* The image to load.

[in] *filename* the destination.

This routine try to read two input files: 'filename' and 'filename.info'. 'filename' is the raw data. 'filename.info' store various information about the image.

Definition at line 184 of file raw/load.hh.

9.93.2.3 `template<typename I > void mln::io::raw::save (const Image< I > & ima_, const std::string & filename)`

Save a Milena image as a raw data file.

Parameters

[in] *ima_* The image to save.

[in] *filename* the destination.

This routine produce two output files: 'filename' and 'filename.info'. 'filename' is the raw data. 'filename.info' store various information about the image.

Definition at line 135 of file raw/save.hh.

9.94 mln::io::tiff Namespace Reference

Namespace of tiff input/output handling.

Functions

- `template<typename I >`
`void load (Image< I > &ima_, const std::string &filename)`
Load a TIFF image to a Milena image.

9.94.1 Detailed Description

Namespace of tiff input/output handling.

9.94.2 Function Documentation

- 9.94.2.1** `template<typename I > void mln::io::tiff::load (Image< I > & ima_, const std::string & filename) [inline]`

Load a TIFF image to a Milena image.

Definition at line 323 of file tiff/load.hh.

9.95 mln::io::txt Namespace Reference

Namespace of txt input/output handling.

Functions

- `void save (const image2d< char > &ima, const std::string &filename)`
Save an image as txt file.

9.95.1 Detailed Description

Namespace of txt input/output handling.

9.95.2 Function Documentation

9.95.2.1 `void mln::io::txt::save (const image2d< char > & ima, const std::string & filename)`
`[inline]`

Save an image as txt file.

Parameters

- [in] *ima* The image to save. Must be an image of char.
- [in] *filename* the destination.

Definition at line 63 of file txt/save.hh.

References `mln::image2d< T >::domain()`.

9.96 mln::labeling Namespace Reference

Namespace of labeling routines.

Namespaces

- namespace [impl](#)
Implementation namespace of labeling namespace.

Functions

- `template<typename I, typename N, typename L >`
`mln::trait::ch_value< I, L >::ret background (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`
- `template<typename I, typename N, typename L >`
`mln::trait::ch_value< I, L >::ret blobs (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`
Connected component labeling of the binary objects of a binary image.
- `template<typename I, typename N, typename L, typename A >`
`util::couple< mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result >, util::array< A > > > blobs_and_compute (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels, const Accumulator< A > &accu)`
- `template<typename V, typename L >`
`mln::trait::ch_value< L, V >::ret colorize (const V &value, const Image< L > &labeled_image, const typename L::value &nlabels)`
Create a new color image from a labeled image and fill each component with a random color.
- `template<typename V, typename L >`
`mln::trait::ch_value< L, V >::ret colorize (const V &value, const Image< L > &labeled_image)`
- `template<typename L >`
`mln::trait::ch_value< L, mln::value::rgb8 >::ret colorize (const Image< L > &input, const typename L::value &nlabels)`

- `template<typename A , typename I , typename L >`
`util::array< mln_meta_accu_result(A, typename I::value)> compute (const Meta_Accumulator< A`
`> &a, const Image< I > &input, const Image< L > &label, const typename L::value &nlabels)`
Compute an accumulator onto the pixel values of the image `input`.
- `template<typename A , typename L >`
`util::array< typename A::result > compute (const Accumulator< A > &a, const Image< L > &la-`
`bel, const typename L::value &nlabels)`
Compute an accumulator onto the pixel sites of each component domain of `label`.
- `template<typename A , typename L >`
`util::array< mln_meta_accu_result(A, typename L::psite)> compute (const Meta_Accumulator< A`
`> &a, const Image< L > &label, const typename L::value &nlabels)`
Compute an accumulator onto the pixel sites of each component domain of `label`.
- `template<typename A , typename I , typename L >`
`util::array< typename A::result > compute (util::array< A > &a, const Image< I > &input, const`
`Image< L > &label, const typename L::value &nlabels)`
Compute an accumulator onto the pixel values of the image `input`.
- `template<typename A , typename I , typename L >`
`util::array< typename A::result > compute (const Accumulator< A > &a, const Image< I > &in-`
`put, const Image< L > &label, const typename L::value &nlabels)`
Compute an accumulator onto the pixel values of the image `input`.
- `template<typename A , typename I , typename L >`
`mln::trait::ch_value< L, typename A::result >::ret compute_image (const util::array< typename`
`A::result > &a, const Image< I > &input, const Image< L > &labels, const typename L::value`
`&nlabels)`
Compute an accumulator onto the pixel values of the image `input`.
- `template<typename A , typename I , typename L >`
`mln::trait::ch_value< L, typename A::result >::ret compute_image (const Accumulator< A >`
`&accu, const Image< I > &input, const Image< L > &labels, const typename L::value &nlabels)`
Compute an accumulator onto the pixel values of the image `input`.
- `template<typename A , typename I , typename L >`
`mln::trait::ch_value< L, typename mln::internal::meta_accu_ret_result_helper< A, typename`
`I::value >::result >::ret compute_image (const Meta_Accumulator< A > &accu, const Image<`
`I > &input, const Image< L > &labels, const typename L::value &nlabels)`
Compute an accumulator onto the pixel values of the image `input`.
- `template<typename I , typename N , typename L >`
`mln::trait::concrete< I >::ret fill_holes (const Image< I > &input, const Neighborhood< N >`
`&nbh, L &nlabels)`
Filling holes of a single object in a binary image.
- `template<typename I , typename N , typename L >`
`mln::trait::ch_value< I, L >::ret flat_zones (const Image< I > &input, const Neighborhood< N >`
`&nbh, L &nlabels)`
Connected component labeling of the flat zones of an image.

- `template<typename I, typename N, typename L >`
`mln::trait::ch_value< I, L >::ret foreground (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`
- `template<typename I >`
`mln::trait::concrete< I >::ret pack (const Image< I > &label, typename I::value &new_nlabels, fun::i2v::array< typename I::value > &repack_fun)`

Relabel a labeled image in order to have a contiguous labeling.

- `template<typename I >`
`mln::trait::concrete< I >::ret pack (const Image< I > &label, typename I::value &new_nlabels)`
- `template<typename I >`
`void pack_inplace (Image< I > &label, typename I::value &new_nlabels)`
- `template<typename I >`
`void pack_inplace (Image< I > &label, typename I::value &new_nlabels, fun::i2v::array< typename I::value > &repack_fun)`

Relabel inplace a labeled image in order to have a contiguous labeling.

- `template<typename I, typename N, typename L >`
`mln::trait::ch_value< I, L >::ret regional_maxima (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`
- `template<typename I, typename N, typename L >`
`mln::trait::ch_value< I, L >::ret regional_minima (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`
- `template<typename I, typename F >`
`mln::trait::concrete< I >::ret relabel (const Image< I > &label, const typename I::value &nlabels, typename I::value &new_nlabels, const Function_v2b< F > &fv2b)`

Remove components and relabel a labeled image.

- `template<typename I, typename F >`
`mln::trait::concrete< I >::ret relabel (const Image< I > &label, const typename I::value &nlabels, const Function_v2v< F > &fv2v)`

Remove components and relabel a labeled image.

- `template<typename I, typename F >`
`void relabel_inplace (Image< I > &label, const typename I::value &nlabels, const Function_v2v< F > &fv2v)`

Remove components and relabel a labeled image inplace.

- `template<typename I, typename F >`
`void relabel_inplace (Image< I > &label, const typename I::value &nlabels, const Function_v2b< F > &fv2b)`

Remove components and relabel a labeled image inplace.

- `template<typename I, typename J >`
`mln::trait::concrete< I >::ret superpose (const Image< I > &lhs, const typename I::value &lhs_nlabels, const Image< J > &rhs, const typename J::value &rhs_nlabels, typename I::value &new_nlabels)`

Superpose two labeled image.

- `template<typename I, typename N, typename L >`
`mln::trait::ch_value< I, L >::ret value` (const `Image< I >` &input, const `typename I::value` &val, const `Neighborhood< N >` &nbh, L &nlabels)

Connected component labeling of the image sites at a given value.

- `template<typename I, typename N, typename L, typename A >`
`util::couple< mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result >, util::array< A > > >` `value_and_compute` (const `Image< I >` &input, const `typename I::value` &val, const `Neighborhood< N >` &nbh, L &nlabels, const `Accumulator< A >` &accu)

Connected component labeling of the image sites at a given value.

- `template<typename V, typename I >`
`mln::trait::ch_value< I, V >::ret wrap` (const `V` &value_type, const `Image< I >` &input)

Wrap labels such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

- `template<typename I >`
`mln::trait::ch_value< I, mln::value::label_8 >::ret wrap` (const `Image< I >` &input)

Wrap labels such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

9.96.1 Detailed Description

Namespace of labeling routines.

9.96.2 Function Documentation

9.96.2.1 `template<typename I, typename N, typename L > mln::trait::ch_value< I, L >::ret mln::labeling::background (const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels) [inline]`

Connected component labeling of the background part in a binary image.

Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the background.
- [out] *nlabels* The number of labels.

Returns

The label image.

Precondition

The input image has to be binary (checked at compile-time).

This routine actually calls `mln::labeling::value` with the value set to `false`.

See also

[mln::labeling::value](#)

Definition at line 69 of file background.hh.

References value().

Referenced by fill_holes().

9.96.2.2 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret
mln::labeling::blobs (const Image< I > & input, const Neighborhood< N > & nbh, L
& nlabels) [inline]`

Connected component labeling of the binary objects of a binary image.

Parameters

[in] *input* The input image.

[in] *nbh* The connexity of the objects.

[out] *nlabels* The Number of labels. Its value is set in the algorithms.

Returns

The label image.

Precondition

The input image has to be binary (checked at compile-time).

A fast queue is used so that the algorithm is not recursive and can handle large binary objects (blobs).

Definition at line 102 of file labeling/blobs.hh.

Referenced by blobs_and_compute(), and mln::graph::labeling().

9.96.2.3 `template<typename I , typename N , typename L , typename A > util::couple<
mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result > ,
util::array< A > > > mln::labeling::blobs_and_compute (const Image< I > & input,
const Neighborhood< N > & nbh, L & nlabels, const Accumulator< A > & accu)`

Label an image and compute given accumulators.

Parameters

[in] *input* A binary image.

[in] *nbh* A neighborhood used for labeling.

[in, out] *nlabels* The number of labels found.

[in] *accu* An accumulator to be computed while labeling.

Returns

The labeled image, computed attributes for each regions and an array of the accumulators used to compute the attributes.

Definition at line 160 of file blobs_and_compute.hh.

References blobs(), and mln::make::couple().

9.96.2.4 `template<typename V , typename L > mln::trait::ch_value< L, V >::ret
mln::labeling::colorize (const V & value, const Image< L > & labeled_image, const
typename L::value & nlabels) [inline]`

Create a new color image from a labeled image and fill each component with a random color.

`litera::black` is used for component 0, e.g. the background. Min and max values for RGB values can be set through the global variables `mln::labeling::colorize_::min_value` and `mln::labeling::colorize_::max_value`.

Parameters

- [in] *value* value type used in the returned image.
- [in] *labeled_image* A labeled image (

See also

[labeling::blobs](#)).

Parameters

- [in] *nlabels* Number of labels.

Definition at line 190 of file `colorize.hh`.

References `mln::literal::black`, and `mln::data::transform()`.

Referenced by `colorize()`.

9.96.2.5 `template<typename V , typename L > mln::trait::ch_value< L, V >::ret
mln::labeling::colorize (const V & value, const Image< L > & labeled_image)
[inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 228 of file `colorize.hh`.

References `colorize()`, and `compute()`.

9.96.2.6 `template<typename L > mln::trait::ch_value< L, mln::value::rgb8 >::ret
mln::labeling::colorize (const Image< L > & input, const typename L::value & nlabels
) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 247 of file `colorize.hh`.

References `colorize()`.

9.96.2.7 `template<typename A , typename I , typename L > util::array<
mln_meta_accu_result(A, typename I::value)> mln::labeling::compute (const
Meta_Accumulator< A > & a, const Image< I > & input, const Image< L > & label,
const typename L::value & nlabels) [inline]`

Compute an accumulator onto the pixel values of the image `input`.

for each component of the image `label`.

Parameters

- [in] *a* A meta-accumulator.
- [in] *input* The input image.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in `label`.

Returns

A `util::array` of accumulator result (one result per label).

Definition at line 734 of file `labeling/compute.hh`.

References `compute()`.

9.96.2.8 `template<typename A , typename L > util::array< typename A::result >
mln::labeling::compute (const Accumulator< A > & a, const Image< L > & label,
const typename L::value & nlabels) [inline]`

Compute an accumulator onto the pixel sites of each component domain of `label`.

Parameters

- [in] *a* An accumulator.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in `label`.

Returns

A `util::array` of accumulator result (one result per label).

Definition at line 771 of file `labeling/compute.hh`.

9.96.2.9 `template<typename A , typename L > util::array< mln_meta_accu_result(A, typename
L::psite)> mln::labeling::compute (const Meta_Accumulator< A > & a, const Image<
L > & label, const typename L::value & nlabels) [inline]`

Compute an accumulator onto the pixel sites of each component domain of `label`.

Parameters

- [in] *a* A meta-accumulator.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in `label`.

Returns

A `util::array` of accumulator result (one result per label).

Definition at line 792 of file `labeling/compute.hh`.

References `compute()`.

9.96.2.10 `template<typename A , typename I , typename L > util::array< typename A::result > mln::labeling::compute (util::array< A > & a, const Image< I > & input, const Image< L > & label, const typename L::value & nlabels) [inline]`

Compute an accumulator onto the pixel values of the image `input`.
for each component of the image `label`.

Parameters

- [in] *a* An array of accumulator.
- [in] *input* The input image.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in `label`.

Returns

A `util::array` of accumulator result (one result per label).

Definition at line 696 of file `labeling/compute.hh`.

9.96.2.11 `template<typename A , typename I , typename L > util::array< typename A::result > mln::labeling::compute (const Accumulator< A > & a, const Image< I > & input, const Image< L > & label, const typename L::value & nlabels) [inline]`

Compute an accumulator onto the pixel values of the image `input`.
for each component of the image `label`.

Parameters

- [in] *a* An accumulator.
- [in] *input* The input image.
- [in] *label* The labeled image.
- [in] *nlabels* The number of labels in `label`.

Returns

A `util::array` of accumulator result (one result per label).

Definition at line 715 of file `labeling/compute.hh`.

Referenced by `colorize()`, `compute()`, `compute_image()`, `fill_holes()`, `mln::make::p_edges_with_mass_centers()`, `mln::make::p_vertices_with_mass_centers()`, `pack()`, and `pack_inplace()`.

9.96.2.12 `template<typename A , typename I , typename L > mln::trait::ch_value< L , typename A ::result >::ret mln::labeling::compute_image (const util::array< typename A::result > & a, const Image< I > & input, const Image< L > & labels, const typename L::value & nlabels)`

Compute an accumulator onto the pixel values of the image `input`.
for each component of the image `label`.

Parameters

- [in] *a* The [mln::p_array](#) of accumulator result.
- [in] *input* The input image (values).
- [in] *labels* The label image.
- [in] *nlabels* The count of labels.

Returns

The image where labels are replaced by the result of the accumulator.

Referenced by `compute_image()`.

9.96.2.13 `template<typename A , typename I , typename L > mln::trait::ch_value< L, typename A::result >::ret mln::labeling::compute_image (const Accumulator< A > & accu, const Image< I > & input, const Image< L > & labels, const typename L::value & nlabels) [inline]`

Compute an accumulator onto the pixel values of the image `input`.
for each component of the image `label`.

Parameters

- [in] *accu* The accumulator.
- [in] *input* The input image (values).
- [in] *labels* The label image.
- [in] *nlabels* The count of labels.

Returns

The image where labels are replaced by the result of the accumulator.

Definition at line 161 of file `compute_image.hh`.

References `compute()`, and `compute_image()`.

9.96.2.14 `template<typename A , typename I , typename L > mln::trait::ch_value< L, typename mln::internal::meta_accu_ret_result_helper< A, typename I::value >::result >::ret mln::labeling::compute_image (const Meta_Accumulator< A > & accu, const Image< I > & input, const Image< L > & labels, const typename L::value & nlabels) [inline]`

Compute an accumulator onto the pixel values of the image `input`.
for each component of the image `label`.

Parameters

- [in] *accu* The meta-accumulator.
- [in] *input* The input image (values).
- [in] *labels* The label image.
- [in] *nlabels* The count of labels.

Returns

The image where labels are replaced by the result of the accumulator.

Definition at line 181 of file `compute_image.hh`.

References `compute()`, and `compute_image()`.

9.96.2.15 `template<typename I , typename N , typename L > mln::trait::concrete< I >::ret
mln::labeling::fill_holes (const Image< I > & input, const Neighborhood< N > &
nbh, L & nlabels) [inline]`

Filling holes of a single object in a binary image.

Parameters

[in] *input* The input image.

[in] *nbh* The connexity of the background.

[out] *nlabels* The number of labels.

Returns

The binary image with a simple object without holes.

Precondition

The input image has to be binary (checked at compile-time).

This routine actually calls [mln::labeling::background](#)

See also

[mln::labeling::background](#)

Definition at line 73 of file `fill_holes.hh`.

References `background()`, `compute()`, `mln::util::array< T >::nelements()`, and `mln::data::transform()`.

9.96.2.16 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret
mln::labeling::flat_zones (const Image< I > & input, const Neighborhood< N > &
nbh, L & nlabels)`

Connected component labeling of the flat zones of an image.

Parameters

[in] *input* The input image.

[in] *nbh* The connexity of the flat zones.

[out] *nlabels* The number of labels.

Returns

The label image.

Definition at line 124 of file `flat_zones.hh`.

9.96.2.17 `template<typename I, typename N, typename L > mln::trait::ch_value< I, L >::ret mln::labeling::foreground (const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels) [inline]`

Connected component labeling of the object part in a binary image.

Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the foreground.
- [out] *nlabels* The number of labels.

Returns

The label image.

Precondition

The input image has to be binary (checked at compile-time).

This routine actually calls [mln::labeling::value](#) with the value set to `true`.

See also

[mln::labeling::value](#)

Definition at line 69 of file foreground.hh.

References `value()`.

9.96.2.18 `template<typename I > mln::trait::concrete< I >::ret mln::labeling::pack (const Image< I > & label, typename I::value & new_nlabels, fun::i2v::array< typename I::value > & repack_fun)`

Relabel a labeled image in order to have a contiguous labeling.

Parameters

- [in] *label* The labeled image.
- [out] *new_nlabels* The number of labels after relabeling.
- [out] *repack_fun* The function used to repack the labels.

Returns

The relabeled image.

Definition at line 124 of file pack.hh.

References `compute()`, `mln::make::relabelfun()`, and `mln::data::transform()`.

Referenced by `pack()`.

9.96.2.19 `template<typename I > mln::trait::concrete< I >::ret mln::labeling::pack (const Image< I > & label, typename I::value & new_nlabels)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 115 of file pack.hh.

References pack().

9.96.2.20 `template<typename I > void mln::labeling::pack_inplace (Image< I > & label, typename I::value & new_nlabels)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 148 of file pack.hh.

References pack_inplace().

9.96.2.21 `template<typename I > void mln::labeling::pack_inplace (Image< I > & label, typename I::value & new_nlabels, fun::i2v::array< typename I::value > & repack_fun)`

Relabel inplace a labeled image in order to have a contiguous labeling.

Parameters

- [in] *label* The labeled image.
- [out] *new_nlabels* The number of labels after relabeling.
- [out] *repack_fun* The function used to repack the labels.

Definition at line 157 of file pack.hh.

References compute(), mln::make::relabelfun(), and mln::data::transform().

Referenced by pack_inplace().

9.96.2.22 `template<typename I, typename N, typename L > mln::trait::ch_value< I, L >::ret mln::labeling::regional_maxima (const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels)`

Connected component labeling of the regional maxima of an image.

Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the regional maxima.
- [out] *nlabels* The number of labeled regions.

Returns

The label image.

Definition at line 130 of file regional_maxima.hh.

9.96.2.23 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret mln::labeling::regional_minima (const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels)`

Connected component labeling of the regional minima of an image.

Parameters

- [in] *input* The input image.
- [in] *nbh* The connexity of the regional minima.
- [out] *nlabels* The number of labeled regions.

Returns

The label image.

Definition at line 140 of file regional_minima.hh.

Referenced by mln::morpho::meyer_wst().

9.96.2.24 `template<typename I , typename F > mln::trait::concrete< I >::ret mln::labeling::relabel (const Image< I > & label, const typename I::value & nlabels, typename I::value & new_nlabels, const Function_v2b< F > & fv2b) [inline]`

Remove components and relabel a labeled image.

Parameters

- [in] *label* the labeled image.
- [in] *nlabels* the number of labels in *label*.
- [out] *new_nlabels* the number of labels after relabeling.
- [in] *fv2b* function returning whether a label must be replaced by the background.

Returns

the relabeled image.

Definition at line 200 of file relabel.hh.

References mln::make::relabelfun().

Referenced by superpose().

9.96.2.25 `template<typename I , typename F > mln::trait::concrete< I >::ret mln::labeling::relabel (const Image< I > & label, const typename I::value & nlabels, const Function_v2v< F > & fv2v) [inline]`

Remove components and relabel a labeled image.

Parameters

- [in] *label* the labeled image.
- [in] *nlabels* the number of labels in *label*.

[in] *fv2v* function returning the new component id for each pixel value.

Returns

the relabeled image.

Definition at line 179 of file relabel.hh.

References `mln::data::transform()`.

9.96.2.26 `template<typename I , typename F > void mln::labeling::relabel_inplace (Image< I > & label, const typename I::value & nlabels, const Function_v2v< F > & fv2v)`
[inline]

Remove components and relabel a labeled image inplace.

Parameters

[in, out] *label* the labeled image.

[in] *nlabels* the number of labels in `label`.

[in] *fv2v* function returning the new component id for each pixel value.

Definition at line 221 of file relabel.hh.

References `mln::data::transform_inplace()`.

9.96.2.27 `template<typename I , typename F > void mln::labeling::relabel_inplace (Image< I > & label, const typename I::value & nlabels, const Function_v2b< F > & fv2b)`
[inline]

Remove components and relabel a labeled image inplace.

Parameters

[in, out] *label* the labeled image.

[in] *nlabels* the number of labels in `label`.

[in] *fv2b* function returning whether a label must be replaced by the background.

Definition at line 240 of file relabel.hh.

References `mln::make::relabelfun()`.

Referenced by `mln::labeled_image_base< I, E >::relabel()`.

9.96.2.28 `template<typename I , typename J > mln::trait::concrete< I >::ret mln::labeling::superpose (const Image< I > & lhs, const typename I::value & lhs_nlabels, const Image< J > & rhs, const typename J::value & rhs_nlabels, typename I::value & new_nlabels)`

Superpose two labeled image.

Labels in `lhs` are preserved in the output. Labels of `rhs` are renumbered from the last label value of `lhs`. It avoids duplicate label values in several components.

Parameters

- [in] *lhs* A labeled image.
- [in] *lhs_nlabels* The number of labels in *lhs*.
- [in] *rhs* A labeled image.
- [in] *rhs_nlabels* The number of labels in *rhs*.
- [out] *new_nlabels* The number of labels in the output image.

Returns

An image with all the components of *rhs* and *lhs*.

Precondition

- rhs* and *lhs* must have the same domain.
- The value type of *rhs* must be convertible towards *lhs*'s.

Definition at line 83 of file labeling/superpose.hh.

References `mln::duplicate()`, `mln::value::equiv()`, `mln::data::paste()`, `relabel()`, and `mln::literal::zero`.

9.96.2.29 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret mln::labeling::value (const Image< I > & input, const typename I::value & val, const Neighborhood< N > & nbh, L & nlabels)`

Connected component labeling of the image sites at a given value.

Parameters

- [in] *input* The input image.
- [in] *val* The value to consider.
- [in] *nbh* The connectivity of components.
- [out] *nlabels* The number of labels.

Returns

The label image.

Definition at line 149 of file labeling/value.hh.

Referenced by `background()`, and `foreground()`.

9.96.2.30 `template<typename I , typename N , typename L , typename A > util::couple< mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result >, util::array< A > > > mln::labeling::value_and_compute (const Image< I > & input, const typename I::value & val, const Neighborhood< N > & nbh, L & nlabels, const Accumulator< A > & accu)`

Connected component labeling of the image sites at a given value.

Parameters

- [in] *input* The input image.

- [in] *val* The value to consider.
- [in] *nbh* The connectivity of components.
- [out] *nlabels* The number of labels.

Returns

The label image.

Definition at line 212 of file `value_and_compute.hh`.

References `mln::make::couple()`.

9.96.2.31 `template<typename V , typename I > mln::trait::ch_value< I, V >::ret
mln::labeling::wrap (const V & value_type, const Image< I > & input) [inline]`

Wrap labels such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

Parameters

- [in] *value_type* The type used to wrap the label type.
- [in] *input* The label image.

Returns

A new image with values wrapped with type V.

Definition at line 75 of file `labeling/wrap.hh`.

References `mln::data::transform()`.

Referenced by `wrap()`.

9.96.2.32 `template<typename I > mln::trait::ch_value< I, mln::value::label_8 >::ret
mln::labeling::wrap (const Image< I > & input) [inline]`

Wrap labels such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

Use `label_8` as label type.

Parameters

- [in] *input* The label image.

Returns

A new image with values wrapped with type `label_8`.

Definition at line 93 of file `labeling/wrap.hh`.

References `wrap()`.

9.97 mln::labeling::impl Namespace Reference

Implementation namespace of labeling namespace.

Namespaces

- namespace [generic](#)

Generic implementation namespace of labeling namespace.

Functions

- `template<typename A , typename I , typename L > util::array< typename A::result > compute_fastest (const Accumulator< A > &a_, const Image< I > &input_, const Image< L > &label_, const typename L::value &nlabels)`

Fastest implementation of [labeling::compute](#).

- `template<typename A , typename I , typename L > util::array< typename A::result > compute_fastest (util::array< A > &accus, const Image< I > &input_, const Image< L > &label_, const typename L::value &nlabels)`

Fastest implementation of [labeling::compute](#).

9.97.1 Detailed Description

Implementation namespace of labeling namespace.

9.97.2 Function Documentation

9.97.2.1 `template<typename A , typename I , typename L > util::array<typename A ::result> mln::labeling::impl::compute_fastest (const Accumulator< A > & a_, const Image< I > & input_, const Image< L > & label_, const typename L::value & nlabels)`
[inline]

Fastest implementation of [labeling::compute](#).

Parameters

- [in] *a_* An accumulator.
- [in] *input_* The input image.
- [in] *label_* The labeled image.
- [in] *nlabels* The number of labels in *label*.

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 373 of file `labeling/compute.hh`.

References `mln::geom::ncols()`.

9.97.2.2 `template<typename A , typename I , typename L > util::array<typename A ::result>
mln::labeling::impl::compute_fastest (util::array< A > & accus, const Image< I > &
input_, const Image< L > & label_, const typename L::value & nlabels) [inline]`

Fastest implementation of `labeling::compute`.

Parameters

- [in] *accus* An array of accumulators.
- [in] *input_* The input image.
- [in] *label_* The labeled image.
- [in] *nlabels* The number of labels in `label_`.

Returns

A `util::array` of accumulator result (one result per label).

Definition at line 427 of file `labeling/compute.hh`.

References `mln::geom::ncols()`, `mln::util::array< T >::resize()`, and `mln::util::array< T >::size()`.

9.98 mln::labeling::impl::generic Namespace Reference

Generic implementation namespace of labeling namespace.

Functions

- `template<typename A , typename L >
util::array< typename A::result > compute (const Accumulator< A > &a_, const Image< L >
&label_, const typename L::value &nlabels)
Generic implementation of labeling::compute.`
- `template<typename A , typename L >
util::array< typename A::result > compute (util::array< A > &accus, const Image< L > &label_,
const typename L::value &nlabels)
Generic implementation of labeling::compute.`
- `template<typename A , typename I , typename L >
util::array< typename A::result > compute (util::array< A > &accus, const Image< I > &input_,
const Image< L > &label_, const typename L::value &nlabels)
Generic implementation of labeling::compute.`
- `template<typename A , typename I , typename L >
util::array< typename A::result > compute (const Accumulator< A > &a_, const Image< I >
&input_, const Image< L > &label_, const typename L::value &nlabels)
Generic implementation of labeling::compute.`

9.98.1 Detailed Description

Generic implementation namespace of labeling namespace.

9.98.2 Function Documentation

9.98.2.1 `template<typename A , typename L > util::array<typename A ::result>
mln::labeling::impl::generic::compute (const Accumulator< A > & a_ , const Image<
L > & label_ , const typename L::value & nlabels) [inline]`

Generic implementation of [labeling::compute](#).

Parameters

- [in] *a_* An accumulator.
- [in] *label_* The labeled image.
- [in] *nlabels* The number of labels in *label*.

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 204 of file `labeling/compute.hh`.

9.98.2.2 `template<typename A , typename L > util::array<typename A ::result>
mln::labeling::impl::generic::compute (util::array< A > & accus , const Image< L >
& label_ , const typename L::value & nlabels) [inline]`

Generic implementation of [labeling::compute](#).

Parameters

- [in] *accus_* An array of accumulators. If the size is set to *nlabels* + 1, the accumulators are considered as initialized. Otherwise, the size is adjusted.
- [in] *label_* The labeled image.
- [in] *nlabels* The number of labels in *label*.

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 241 of file `labeling/compute.hh`.

References `mln::util::array< T >::resize()`, and `mln::util::array< T >::size()`.

9.98.2.3 `template<typename A , typename I , typename L > util::array<typename A ::result>
mln::labeling::impl::generic::compute (util::array< A > & accus , const Image< I > &
input_ , const Image< L > & label_ , const typename L::value & nlabels) [inline]`

Generic implementation of [labeling::compute](#).

Parameters

- [in] *accus* An array of accumulators.
- [in] *input_* The input image.
- [in] *label_* The labeled image.

[in] *nlabels* The number of labels in `label`.

Returns

A `util::array` of accumulator result (one result per label).

Definition at line 321 of file `labeling/compute.hh`.

References `mln::util::array< T >::resize()`, and `mln::util::array< T >::size()`.

9.98.2.4 `template<typename A , typename I , typename L > util::array<typename A ::result>
mln::labeling::impl::generic::compute (const Accumulator< A > & a_, const Image< I
> & input_, const Image< L > & label_, const typename L::value & nlabels)
[inline]`

Generic implementation of `labeling::compute`.

Parameters

[in] *a_* An accumulator.

[in] *input_* The input image.

[in] *label_* The labeled image.

[in] *nlabels* The number of labels in `label`.

Returns

A `util::array` of accumulator result (one result per label).

Definition at line 283 of file `labeling/compute.hh`.

9.99 mln::linear Namespace Reference

Namespace of linear image processing routines.

Namespaces

- namespace `impl`
Namespace of linear image processing routines implementation details.
- namespace `local`
Specializations of local linear routines.

Functions

- `template<typename I >
mln::trait::concrete< I >::ret gaussian (const Image< I > &input, float sigma)`
Gaussian filter of an image `input`.

- `template<typename I >`
`mln::trait::concrete< I >::ret gaussian (const Image< I > &input, float sigma, int dir)`
- `template<typename I >`
`mln::trait::concrete< I >::ret gaussian_1st_derivative (const Image< I > &input, float sigma)`
- `template<typename I >`
`mln::trait::concrete< I >::ret gaussian_1st_derivative (const Image< I > &input, float sigma, int dir)`
- `template<typename I >`
`mln::trait::concrete< I >::ret gaussian_2nd_derivative (const Image< I > &input, float sigma)`
- `template<typename I >`
`mln::trait::concrete< I >::ret gaussian_2nd_derivative (const Image< I > &input, float sigma, int dir)`
- `template<typename I , typename W , unsigned Sh, unsigned Sv>`
`mln_ch_convolve (I, W) convolve_2x1d(const Image< I > &input`
- `template<typename I , typename W >`
`mln_ch_convolve (I, W) convolve(const Image< I > &input`
- `template<typename I , typename W , unsigned S>`
`mln_ch_convolve (I, W) convolve_directional(const Image< I > &input`
- `template<typename I >`
`mln_ch_convolve_grad (I, int) sobel_2d(const Image< I > &input)`
Compute the vertical component of the 2D Sobel gradient.

- `template<typename I >`
`mln_ch_convolve (I, int) sobel_2d_h(const Image< I > &input)`
Sobel_2d gradient components.

9.99.1 Detailed Description

Namespace of linear image processing routines.

9.99.2 Function Documentation

9.99.2.1 `template<typename I > mln::trait::concrete< I >::ret mln::linear::gaussian (const Image< I > & input, float sigma) [inline]`

Gaussian filter of an image `input`.

Precondition

`output.domain = input.domain`

Apply an approximated gaussian filter of `sigma` on `input`. This filter is applied in all the input image direction.

Precondition

`input.is_valid`

Definition at line 750 of file `gaussian.hh`.

References `mln::initialize()`.

Referenced by `mln::subsampling::gaussian_subsampling()`.

9.99.2.2 `template<typename I> mln::trait::concrete< I >::ret mln::linear::gaussian (const Image< I> & input, float sigma, int dir) [inline]`

Apply an approximated gaussian filter of `sigma` on `input`. on a specific direction `dir` if `dir = 0`, the filter is applied on the first image dimension. if `dir = 1`, the filter is applied on the second image dimension. And so on...

Precondition

`input.is_valid`
`dir < dimension(input)`

Definition at line 653 of file `gaussian.hh`.

References `mln::initialize()`.

9.99.2.3 `template<typename I> mln::trait::concrete< I >::ret mln::linear::gaussian_1st_derivative (const Image< I> & input, float sigma) [inline]`

Apply an approximated first derivative gaussian filter of `sigma` on `input` This filter is applied in all the input image direction.

Precondition

`input.is_valid`

Definition at line 779 of file `gaussian.hh`.

References `mln::initialize()`.

9.99.2.4 `template<typename I> mln::trait::concrete< I >::ret mln::linear::gaussian_1st_derivative (const Image< I> & input, float sigma, int dir) [inline]`

Apply an approximated first derivative gaussian filter of `sigma` on `input`. on a specific direction `dir` if `dir = 0`, the filter is applied on the first image dimension. if `dir = 1`, the filter is applied on the second image dimension. And so on...

Precondition

`input.is_valid`
`dir < dimension(input)`

Definition at line 685 of file `gaussian.hh`.

References `mln::initialize()`.

9.99.2.5 `template<typename I> mln::trait::concrete< I >::ret mln::linear::gaussian_2nd_derivative (const Image< I> & input, float sigma) [inline]`

Apply an approximated second derivative gaussian filter of `sigma` on `input` This filter is applied in all the input image direction.

Precondition

input.is_valid

Definition at line 807 of file gaussian.hh.

References mln::initialize().

9.99.2.6 `template<typename I > mln::trait::concrete< I >::ret mln::linear::gaussian_2nd_derivative (const Image< I > & input, float sigma, int dir) [inline]`

Apply an approximated second derivative gaussian filter of `sigma` on `input`. on a specific direction `dir` if `dir = 0`, the filter is applied on the first image dimension. if `dir = 1`, the filter is applied on the second image dimension. And so on...

Precondition

input.is_valid
dir < dimension(input)

Definition at line 718 of file gaussian.hh.

References mln::initialize().

9.99.2.7 `template<typename I > mln::linear::mln_ch_convolve (I, int) const [inline]`

Sobel_2d gradient components.

Compute the L1 norm of the 2D Sobel gradient.

Compute the vertical component of the 2D Sobel gradient.

Compute the horizontal component of the 2D Sobel gradient.

Definition at line 142 of file sobel_2d.hh.

References mln_ch_convolve(), mln_ch_convolve_grad(), and mln::data::transform().

9.99.2.8 `template<typename I , typename W , unsigned Sh, unsigned Sv> mln::linear::mln_ch_convolve (I, W) const`

Convolution of an image `input` by two weighted line-shapes windows.

Warning

The weighted window is used as-is, considering that its symmetrization is handled by the client.

Precondition

input.is_valid

9.99.2.9 `template<typename I , typename W > mln::linear::mln_ch_convolve (I, W) const`

Convolution of an image `input` by the weighted window `w_win`.

Warning

Computation of `output(p)` is performed with the value type of `output`.
The weighted window is used as-is, considering that its symmetrization is handled by the client.

Precondition

`input.is_valid`

Referenced by `mln_ch_convolve()`, and `mln_ch_convolve_grad()`.

9.99.2.10 `template<typename I, typename W, unsigned S> mln::linear::mln_ch_convolve (I, W) const [inline]`

Convolution of an image `input` by a line-shaped (directional) weighted window defined by the array of `weights`.

Warning

Computation of `output(p)` is performed with the value type of `output`.
The weighted window is used as-is, considering that its symmetrization is handled by the client.

Precondition

`input.is_valid`

9.99.2.11 `template<typename I > mln::linear::mln_ch_convolve_grad (I, int) const`

Compute the vertical component of the 2D Sobel gradient.

Definition at line 124 of file `sobel_2d.hh`.

References `mln_ch_convolve()`, and `mln::data::transform()`.

Referenced by `mln_ch_convolve()`.

9.100 `mln::linear::impl` Namespace Reference

Namespace of linear image processing routines implementation details.

9.100.1 Detailed Description

Namespace of linear image processing routines implementation details.

9.101 `mln::linear::local` Namespace Reference

Specializations of local linear routines.

Namespaces

- namespace [impl](#)

Namespace of local linear routines implementation details.

Functions

- `template<typename I , typename P , typename W , typename R >`
void [convolve](#) (const [Image](#)< I > &input, const [Site](#)< P > &p, const [Weighted_Window](#)< W > &w_win, R &result)
- `template<typename P , typename W , typename R >`
void [convolve](#) (const [Generalized_Pixel](#)< P > &p, const [Weighted_Window](#)< W > &w_win, R &result)

9.101.1 Detailed Description

Specializations of local linear routines.

9.101.2 Function Documentation

9.101.2.1 `template<typename I , typename P , typename W , typename R > void`
`mln::linear::local::convolve (const Image< I > & input, const Site< P > & p, const`
`Weighted_Window< W > & w_win, R & result) [inline]`

Local convolution of image `input` at point `p` by the weighted window `w_win`.

Warning

Computation of the `result` is performed with the type `R`.

The weighted window is used as-is, considering that its symmetrization is handled by the client.

Definition at line 149 of file `linear/local/convolve.hh`.

Referenced by `convolve()`.

9.101.2.2 `template<typename P , typename W , typename R > void mln::linear::local::convolve (`
`const Generalized_Pixel< P > & p, const Weighted_Window< W > & w_win, R &`
`result) [inline]`

Local convolution around (generalized) pixel by the weighted window `w_win`.

Warning

Computation of the `result` is performed with the type `R`.

The weighted window is used as-is, considering that its symmetrization is handled by the client.

Definition at line 161 of file `linear/local/convolve.hh`.

References `convolve()`.

9.102 mln::linear::local::impl Namespace Reference

Namespace of local linear routines implementation details.

9.102.1 Detailed Description

Namespace of local linear routines implementation details.

9.103 mln::literal Namespace Reference

Namespace of literals.

Classes

- struct [black_t](#)
Type of literal black.
- struct [blue_t](#)
Type of literal blue.
- struct [brown_t](#)
Type of literal brown.
- struct [cyan_t](#)
Type of literal cyan.
- struct [green_t](#)
Type of literal green.
- struct [identity_t](#)
Type of literal identity.
- struct [light_gray_t](#)
Type of literal grays.
- struct [lime_t](#)
Type of literal lime.
- struct [magenta_t](#)
Type of literal magenta.
- struct [max_t](#)
Type of literal max.
- struct [min_t](#)
Type of literal min.
- struct [olive_t](#)

Type of literal olive.

- struct [one_t](#)
Type of literal one.
- struct [orange_t](#)
Type of literal orange.
- struct [origin_t](#)
Type of literal origin.
- struct [pink_t](#)
Type of literal pink.
- struct [purple_t](#)
Type of literal purple.
- struct [red_t](#)
Type of literal red.
- struct [teal_t](#)
Type of literal teal.
- struct [violet_t](#)
Type of literal violet.
- struct [white_t](#)
Type of literal white.
- struct [yellow_t](#)
Type of literal yellow.
- struct [zero_t](#)
Type of literal zero.

Variables

- const [black_t](#) & [black](#) = [black_t](#)()
Literal black.
- const [blue_t](#) & [blue](#) = [blue_t](#)()
Literal blue.
- const [brown_t](#) & [brown](#) = [brown_t](#)()
Literal brown.
- const [cyan_t](#) & [cyan](#) = [cyan_t](#)()
Literal cyan.

- `const dark_gray_t & dark_gray = dark_gray_t()`
Literal dark gray.
- `const green_t & green = green_t()`
Literal green.
- `const identity_t & identity = identity_t()`
Literal identity.
- `const light_gray_t & light_gray = light_gray_t()`
Literal light gray.
- `const lime_t & lime = lime_t()`
Literal lime.
- `const magenta_t & magenta = magenta_t()`
Literal magenta.
- `const max_t & max = max_t()`
Literal max.
- `const medium_gray_t & medium_gray = medium_gray_t()`
Literal medium_gray.
- `const min_t & min = min_t()`
Literal min.
- `const olive_t & olive = olive_t()`
Literal olive.
- `const one_t & one = one_t()`
Literal one.
- `const orange_t & orange = orange_t()`
Literal orange.
- `const origin_t & origin = origin_t()`
Literal origin.
- `const pink_t & pink = pink_t()`
Literal pink.
- `const purple_t & purple = purple_t()`
Literal purple.
- `const red_t & red = red_t()`
Literal red.
- `const teal_t & teal = teal_t()`
Literal teal.

- const [violet_t](#) & [violet](#) = [violet_t\(\)](#)
Literal violet.
- const [white_t](#) & [white](#) = [white_t\(\)](#)
Literal white.
- const [yellow_t](#) & [yellow](#) = [yellow_t\(\)](#)
Literal yellow.
- const [zero_t](#) & [zero](#) = [zero_t\(\)](#)
Literal zero.

9.103.1 Detailed Description

Namespace of literals.

9.103.2 Variable Documentation

9.103.2.1 const [black_t](#) & [mln::literal::black](#) = [black_t\(\)](#)

[Literal](#) black.

Definition at line 60 of file [black.hh](#).

Referenced by [mln::labeling::colorize\(\)](#), and [mln::registration::icp\(\)](#).

9.103.2.2 const [blue_t](#) & [mln::literal::blue](#) = [blue_t\(\)](#)

[Literal](#) blue.

Definition at line 162 of file [colors.hh](#).

9.103.2.3 const [brown_t](#) & [mln::literal::brown](#) = [brown_t\(\)](#)

[Literal](#) brown.

Definition at line 164 of file [colors.hh](#).

9.103.2.4 const [cyan_t](#) & [mln::literal::cyan](#) = [cyan_t\(\)](#)

[Literal](#) cyan.

Definition at line 178 of file [colors.hh](#).

9.103.2.5 const [dark_gray_t](#) & [mln::literal::dark_gray](#) = [dark_gray_t\(\)](#)

[Literal](#) dark gray.

Definition at line 70 of file [grays.hh](#).

9.103.2.6 `const green_t & mln::literal::green = green_t()`

[Literal](#) green.

Definition at line 160 of file colors.hh.

Referenced by `mln::registration::icp()`, and `mln::make_debug_graph_image()`.

9.103.2.7 `const identity_t & mln::literal::identity = identity_t()`

[Literal](#) identity.

Definition at line 54 of file identity.hh.

9.103.2.8 `const light_gray_t & mln::literal::light_gray = light_gray_t()`

[Literal](#) light gray.

Definition at line 66 of file grays.hh.

9.103.2.9 `const lime_t & mln::literal::lime = lime_t()`

[Literal](#) lime.

Definition at line 166 of file colors.hh.

9.103.2.10 `const magenta_t & mln::literal::magenta = magenta_t()`

[Literal](#) magenta.

Definition at line 180 of file colors.hh.

9.103.2.11 `const max_t & mln::literal::max = max_t()`

[Literal](#) max.

Definition at line 66 of file literal/max.hh.

9.103.2.12 `const medium_gray_t & mln::literal::medium_gray = medium_gray_t()`

[Literal](#) medium_gray.

Definition at line 68 of file grays.hh.

9.103.2.13 `const min_t & mln::literal::min = min_t()`

[Literal](#) min.

Definition at line 66 of file literal/min.hh.

9.103.2.14 const olive_t & mln::literal::olive = olive_t()

[Literal](#) olive.

Definition at line 184 of file colors.hh.

9.103.2.15 const one_t & mln::literal::one = one_t()

[Literal](#) one.

Definition at line 69 of file one.hh.

Referenced by mln::algebra::h_vec< d, C >::h_vec(), mln::operator++(), and mln::operator--().

9.103.2.16 const orange_t & mln::literal::orange = orange_t()

[Literal](#) orange.

Definition at line 168 of file colors.hh.

9.103.2.17 const origin_t & mln::literal::origin = origin_t()

[Literal](#) origin.

Definition at line 55 of file origin.hh.

Referenced by mln::win::ball< G, C >::ball(), mln::geom::bbox(), mln::box< P >::box(), mln::geom::rotate(), and mln::make::w_window().

9.103.2.18 const pink_t & mln::literal::pink = pink_t()

[Literal](#) pink.

Definition at line 170 of file colors.hh.

9.103.2.19 const purple_t & mln::literal::purple = purple_t()

[Literal](#) purple.

Definition at line 172 of file colors.hh.

9.103.2.20 const red_t & mln::literal::red = red_t()

[Literal](#) red.

Definition at line 158 of file colors.hh.

Referenced by mln::morpho::watershed::superpose(), and mln::debug::superpose().

9.103.2.21 const teal_t & mln::literal::teal = teal_t()

[Literal](#) teal.

Definition at line 174 of file colors.hh.

9.103.2.22 const violet_t & mln::literal::violet = violet_t()

[Literal](#) violet.

Definition at line 176 of file colors.hh.

9.103.2.23 const white_t & mln::literal::white = white_t()

[Literal](#) white.

Definition at line 60 of file white.hh.

Referenced by mln::registration::icp().

9.103.2.24 const yellow_t & mln::literal::yellow = yellow_t()

[Literal](#) yellow.

Definition at line 182 of file colors.hh.

9.103.2.25 const zero_t & mln::literal::zero = zero_t()

[Literal](#) zero.

Definition at line 69 of file zero.hh.

Referenced by mln::transform::influence_zone_geodesic_saturated(), mln::accu::shape::volume< I >::init(), mln::accu::stat::variance< T, S, R >::init(), mln::morpho::attribute::sum< I, S >::init(), mln::accu::math::sum< T, S >::init(), mln::accu::rms< T, V >::init(), mln::accu::stat::histo3d_rgb< V >::init(), mln::accu::convolve< T1, T2, R >::init(), mln::accu::center< P, V >::init(), mln::window< D >::is_centered(), mln::accu::stat::variance< T, S, R >::mean(), mln::accu::stat::var< T >::mean(), mln::geom::mesh_corner_point_area(), mln::geom::mesh_curvature(), mln::geom::mesh_normal(), mln::morpho::meyer_wst(), mln::algebra::operator*(), mln::test::positive(), mln::make::relabelfun(), mln::geom::rotate(), mln::accu::shape::volume< I >::set_value(), mln::morpho::watershed::superpose(), mln::labeling::superpose(), mln::debug::superpose(), mln::accu::stat::var< T >::to_result(), mln::geom::translate(), and mln::make::w_window_directional().

9.104 mln::logical Namespace Reference

Namespace of logic.

Namespaces

- namespace [impl](#)

Implementation namespace of logical namespace.

Functions

- `template<typename L, typename R >`
void [and_inplace](#) ([Image](#)< L > &lhs, const [Image](#)< R > &rhs)

- `template<typename L , typename R >`
`mln::trait::ch_value< L, typename mln::fun::vv2v::land_not< typename L::value, typename R::value >::result >::ret and_not (const Image< L > &lhs, const Image< R > &rhs)`
- `template<typename L , typename R >`
`void and_not_inplace (Image< L > &lhs, const Image< R > &rhs)`
- `template<typename I >`
`void not_inplace (Image< I > &input)`
- `template<typename L , typename R >`
`void or_inplace (Image< L > &lhs, const Image< R > &rhs)`
- `template<typename L , typename R >`
`void xor_inplace (Image< L > &lhs, const Image< R > &rhs)`

9.104.1 Detailed Description

Namespace of logic.

9.104.2 Function Documentation

9.104.2.1 `template<typename L , typename R > void mln::logical::and_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise in-place "logical and" of image `rhs` in image `lhs`.

Parameters

- `[in, out]` *lhs* First operand image.
- `[in]` *rhs* Second operand image.

It performs:

for all `p` of `rhs.domain`

`lhs(p) = lhs(p) and rhs(p)`

Precondition

`rhs.domain >= lhs.domain`

Definition at line 91 of file `logical/and.hh`.

References `mln::data::transform_inplace()`.

9.104.2.2 `template<typename L , typename R > mln::trait::ch_value< L, typename mln::fun::vv2v::land_not< typename L::value, typename R::value >::result >::ret mln::logical::and_not (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise "logical and-not" between images `lhs` and `rhs`.

Parameters

- `[in]` *lhs* First operand image.
- `[in]` *rhs* Second operand image.

Returns

The result image.

Precondition

```
lhs.domain == rhs.domain
```

Definition at line 76 of file and_not.hh.

References mln::data::transform().

9.104.2.3 `template<typename L , typename R > void mln::logical::and_not_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise in-place "logical and-not" of image `rhs` in image `lhs`.

Parameters

[in, out] *lhs* First operand image.

[in] *rhs* Second operand image.

It performs:

for all `p` of `rhs.domain`

`lhs(p) = lhs(p) and not rhs(p)`

Precondition

```
rhs.domain >= lhs.domain
```

Definition at line 91 of file and_not.hh.

References mln::data::transform_inplace().

9.104.2.4 `template<typename I > void mln::logical::not_inplace (Image< I > & input) [inline]`

Point-wise in-place "logical not" of image `input`.

Parameters

[in, out] *input* The target image.

It performs:

for all `p` of `input.domain`

`input(p) = not input(p)`

Precondition

```
input.is_valid
```

Definition at line 88 of file logical/not.hh.

References mln::data::transform_inplace().

9.104.2.5 `template<typename L , typename R > void mln::logical::or_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise in-place "logical or" of image `rhs` in image `lhs`.

Parameters

[in, out] *lhs* First operand image.

[in] *rhs* Second operand image.

It performs:

for all `p` of `rhs.domain`

`lhs(p) = lhs(p) or rhs(p)`

Precondition

`rhs.domain >= lhs.domain`

Definition at line 91 of file `logical/or.hh`.

References `mln::data::transform_inplace()`.

9.104.2.6 `template<typename L , typename R > void mln::logical::xor_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise in-place "logical xor" of image `rhs` in image `lhs`.

Parameters

[in, out] *lhs* First operand image.

[in] *rhs* Second operand image.

It performs:

for all `p` of `rhs.domain`

`lhs(p) = lhs(p) xor rhs(p)`

Precondition

`rhs.domain >= lhs.domain`

Definition at line 91 of file `logical/xor.hh`.

References `mln::data::transform_inplace()`.

9.105 mln::logical::impl Namespace Reference

Implementation namespace of logical namespace.

Namespaces

- namespace [generic](#)

Generic implementation namespace of logical namespace.

9.105.1 Detailed Description

Implementation namespace of logical namespace.

9.106 mln::logical::impl::generic Namespace Reference

Generic implementation namespace of logical namespace.

9.106.1 Detailed Description

Generic implementation namespace of logical namespace.

9.107 mln::make Namespace Reference

Namespace of routines that help to make Milena's objects.

Functions

- `template<unsigned D, typename G, typename V > p_set< complex_site< D, G > > attachment (const complex_site< D, G > &f, const complex_image< D, G, V > &ima)`
Compute the attachment of the cell corresponding to the facet f to the image ima .
- `mln::box1d box1d (def::coord min_ind, def::coord max_ind)`
Create an `mln::box1d`.
- `mln::box1d box1d (unsigned ninds)`
Create an `mln::box1d`.
- `mln::box2d box2d (unsigned nrows, unsigned ncols)`
Create an `mln::box2d`.
- `mln::box2d box2d (def::coord min_row, def::coord min_col, def::coord max_row, def::coord max_col)`
Create an `mln::box2d`.
- `mln::box2d_h box2d_h (def::coord min_row, def::coord min_col, def::coord max_row, def::coord max_col)`
Create an `mln::box2d_h`.
- `mln::box2d_h box2d_h (unsigned nrows, unsigned ncols)`
Create an `mln::box2d_h`.
- `mln::box3d box3d (unsigned nslis, unsigned nrows, unsigned ncols)`
Create an `mln::box3d`.

- `mln::box3d box3d (def::coord min_sli, def::coord min_row, def::coord min_col, def::coord max_sli, def::coord max_row, def::coord max_col)`
Create an `mln::box3d`.
- `template<unsigned D, typename G >`
`p_set< complex_psit< D, G > > cell (const complex_psit< D, G > &f)`
Compute the set of faces of the cell corresponding to the facet `f`.
- `template<typename T, typename U >`
`util::couple< T, U > couple (const T &val1, const T &val2)`
Construct an `mln::util::couple` on-the-fly.
- `template<unsigned D, typename G, typename V >`
`p_set< complex_psit< D, G > > detachment (const complex_psit< D, G > &f, const complex_image< D, G, V > &ima)`
Compute the detachment of the cell corresponding to the facet `f` to the image `ima`.
- `mln::dpoint2d_h dpoint2d_h (def::coord row, def::coord col)`
Create an `mln::dpoint2d_h`.
- `template<typename G, typename P >`
`p_edges< G, pw::cst_< P > > dummy_p_edges (const Graph< G > &g_, const P &dummy_site)`
Create a `p_edges` which associate a graph element to a constant site.
- `template<typename G >`
`p_edges< G > dummy_p_edges (const Graph< G > &g)`
Create a `p_edges` which associate a graph element to a constant site.
- `template<typename G, typename P >`
`p_vertices< G, pw::cst_< P > > dummy_p_vertices (const Graph< G > &g_, const P &dummy_site)`
Create a `p_vertices` which associate a graph element to a constant site.
- `template<typename G >`
`p_vertices< G > dummy_p_vertices (const Graph< G > &g)`
Create a `p_vertices` which associate a graph element to a constant site.
- `template<typename V, typename G >`
`mln::edge_image< void, V, G > edge_image (const Graph< G > &g, const fun::i2v::array< V > &fv)`
Construct an edge image.
- `template<typename FV, typename G >`
`mln::edge_image< void, typename FV::result, G > edge_image (const Graph< G > &g, const Function_v2v< FV > &fv)`
Construct an edge image.
- `template<typename FP, typename FV, typename G >`
`mln::edge_image< typename FP::result, typename FV::result, G > edge_image (const Graph< G > &g_, const Function_v2v< FP > &fp, const Function_v2v< FV > &fv)`
Construct an edge image.

- `template<typename P, typename V, typename G, typename FP, typename FV >`
`mln::edge_image< typename FP::result, typename FV::result, G > edge_image (const mln::vertex_image< P, V, G > &v_ima_, const p_edges< G, FP > pe, const Function_vv2v< FV > &fv_)`
Construct an edge image.
- `template<typename P, typename V, typename G, typename FV >`
`mln::edge_image< void, typename FV::result, G > edge_image (const mln::vertex_image< P, V, G > &v_ima_, const Function_vv2v< FV > &fv_)`
Construct an edge image.
- `template<typename P, typename V, typename G, typename F >`
`mln::edge_image< void, bool, G > edge_image (const mln::vertex_image< P, V, G > &v_ima_, const Function_v2b< F > &fv_)`
Construct an edge image.
- `template<typename T, unsigned N>`
`algebra::h_mat< mlc_sqrt_int(N), T > h_mat (const T(&tab)[N])`
Create an mln::algebra::mat<n,n,T>.
- `template<typename V, unsigned L>`
`mln::image1d< V > image (V(&values)[L])`
Create an image1d from an 1D array of values.
- `template<typename V, unsigned R, unsigned C>`
`mln::image2d< V > image (V(&values)[R][C])`
Create an image2d from an 2D array of values.
- `template<typename V, unsigned S, unsigned R, unsigned C>`
`mln::image3d< V > image (V(&values)[S][R][C])`
Create an image3d from an 3D array of values.
- `template<typename V, unsigned S>`
`mln::image2d< V > image2d (V(&values)[S])`
Create an image2d from an 2D array of values.
- `template<typename I >`
`mln::image3d< typename I::value > image3d (const Image< I > &ima)`
Create an image3d from a 2D image.
- `template<typename I >`
`mln::image3d< typename I::value > image3d (const util::array< I > &ima)`
Create an image3d from an array of 2D images.
- `template<typename I, typename N >`
`util::graph_influence_zone_adjacency_graph (const Image< I > &iz_, const Neighborhood< N > &nbh, const typename I::value &nlabels)`
Create a graph from an influence zone image.
- `template<unsigned n, unsigned m, typename T >`
`algebra::mat< n, m, T > mat (const T(&tab)[n * m])`

Create an `mln::algebra::mat<n,m,T>`.

- `template<typename T >`
`util::ord_pair< T > ord_pair (const T &val1, const T &val2)`
Construct an `mln::util::ord_pair` on-the-fly.
- `template<typename W , typename G >`
`p_edges< G, fun::i2v::array< util::site_pair< typename W::site > > > p_edges_with_mass_centers`
`(const Image< W > &wst_, const Graph< G > &g_)`
Construct a `p_edges` from a watershed image and a region adjacency graph (RAG).
- `template<typename W , typename G >`
`p_vertices< G, fun::i2v::array< typename W::site > > p_vertices_with_mass_centers (const Im-`
`age< W > &wst_, const Graph< G > &g_)`
Construct a `p_vertices` from a watershed image and a region adjacency graph (RAG).
- `template<typename I >`
`mln::util::pix< I > pix (const Image< I > &ima, const typename I::psite &p)`
Create an `mln::util::pix` from an image `ima` and a psite `p`.
- `template<typename I >`
`mln::pixel< I > pixel (Image< I > &ima, const typename I::psite &p)`
Create a `mln::pixel` from a mutable image `ima` and a point `p`.
- `template<typename I >`
`mln::pixel< const I > pixel (const Image< I > &ima, const typename I::psite &p)`
Create a `mln::pixel` from a constant image `ima` and a point `p`.
- `mln::point2d_h point2d_h (def::coord row, def::coord col)`
Create an `mln::point2d_h`.
- `template<typename I , typename N >`
`util::couple< util::graph, typename mln::trait::concrete< I >::ret > rag_and_labeled_wsl (const Im-`
`age< I > &wshd_, const Neighborhood< N > &nbh_, const typename I::value &nbasins)`
Create a region adjacency graph and a label image of the watershed line from a watershed image.
- `template<typename I , typename N >`
`util::graph_region_adjacency_graph (const Image< I > &wshd_, const Neighborhood< N > &nbh,`
`const typename I::value &nbasins)`
Create a region adjacency graph from a watershed image.
- `template<typename V , typename F >`
`fun::i2v::array< V > relabelfun (const Function_v2b< F > &fv2b, const V &nlabels, V &new_-`
`nlabels)`
Create a `i2v` function from a `v2b` function.
- `template<typename V , typename F >`
`fun::i2v::array< V > relabelfun (const Function_v2v< F > &fv2v, const V &nlabels, V &new_-`
`nlabels)`
Create a `i2v` function from a `v2v` function.

- `template<typename T >`
`algebra::vec< 1, T > vec (const T &v_0)`
Create an `mln::algebra::vec<n,T>`.
- `template<typename T >`
`algebra::vec< 4, T > vec (const T &v_0, const T &v_1, const T &v_2, const T &v_3)`
Create an `mln::algebra::vec<4,T>`.
- `template<typename T >`
`algebra::vec< 3, T > vec (const T &v_0, const T &v_1, const T &v_2)`
Create an `mln::algebra::vec<3,T>`.
- `template<typename T >`
`algebra::vec< 2, T > vec (const T &v_0, const T &v_1)`
Create an `mln::algebra::vec<2,T>`.
- `template<typename G , typename FV >`
`mln::vertex_image< void, typename FV::result, G > vertex_image (const Graph< G > &g, const Function_v2v< FV > &fv)`
Construct a vertex image.
- `template<typename FP , typename FV , typename G >`
`mln::vertex_image< typename FP::result, typename FV::result, G > vertex_image (const Graph< G > &g_, const Function_v2v< FP > &fp, const Function_v2v< FV > &fv)`
Construct a vertex image.
- `template<typename I , typename N >`
`p_vertices< util::graph, fun::i2v::array< typename I::site > > voronoi (Image< I > &ima_, Image< I > &orig_, const Neighborhood< N > &nbh)`
Apply the Voronoi algorithm on `ima_` with the original image `orig_` for node computing with neighborhood `nbh`.
- `template<typename W , typename F >`
`mln::w_window< typename W::dpsite, typename F::result > w_window (const Window< W > &win, const Function_v2v< F > &wei)`
Create a `mln::w_window` from a window and a weight function.
- `template<typename W , unsigned M>`
`mln::w_window< mln::dpoint1d, W > w_window1d (W(&weights)[M])`
Create a 1D `mln::w_window` from an array of weights.
- `template<unsigned M>`
`mln::w_window1d_int w_window1d_int (int(&weights)[M])`
Create a `mln::w_window1d_int`.
- `template<typename W , unsigned S>`
`mln::w_window< mln::dpoint2d, W > w_window2d (W(&weights)[S])`
Create a 2D `mln::w_window` from an array of weights.
- `template<unsigned M>`
`mln::w_window2d_int w_window2d_int (int(&weights)[M])`

Create a `mln::w_window2d_int`.

- `template<typename W , unsigned M>`
`mln::w_window< mln::dpoint3d, W > w_window3d (W(&weights)[M])`
 Create a 3D `mln::w_window` from an array of weights.
- `template<unsigned M>`
`mln::w_window3d_int w_window3d_int (int(&weights)[M])`
 Create a `mln::w_window3d_int`.
- `template<typename D , typename W , unsigned L>`
`mln::w_window< D, W > w_window_directional (const Gdpoint< D > &dp, W(&weights)[L])`
 Create a directional centered weighted window.

9.107.1 Detailed Description

Namespace of routines that help to make Milena's objects.

9.107.2 Function Documentation

9.107.2.1 `template<unsigned D, typename G , typename V > p_set< complex_site< D, G > >`
`mln::make::attachment (const complex_site< D, G > & f, const complex_image< D,`
`G, V > & ima) [inline]`

Compute the attachment of the cell corresponding to the facet *f* to the image *ima*.

Precondition

f is a facet (it does not belong to any face of higher dimension).
ima is an image of Boolean values.

Returns

a set of faces containing the attachment.

We do not use the formal definition of the attachment here (see `coupric.08.pami`). We use the following (equivalent) definition: an N-face F in CELL is in the attachment of CELL to IMA if it is adjacent to at least an (N-1)-face or an (N+1)-face that does not belong to CELL.

Definition at line 68 of file `attachment.hh`.

References `cell()`, and `mln::topo::is_facet()`.

Referenced by `mln::topo::is_simple_cell< I >::operator()`.

9.107.2.2 `mln::box1d mln::make::box1d (def::coord min_ind, def::coord max_ind)`
`[inline]`

Create an `mln::box1d`.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

[in] *min_ind* Minimum index.

[in] *max_ind* Maximum index.

Precondition

`max_ind >= min_ind.`

Returns

A 1D box.

Definition at line 79 of file `make/box1d.hh`.

References `box1d()`.

9.107.2.3 `mln::box1d mln::make::box1d (unsigned ninds) [inline]`

Create an [mln::box1d](#).

Parameters

[in] *ninds* Number of indices.

Precondition

`ninds != 0 and ncols != 0.`

Returns

A 1D box.

Definition at line 70 of file `make/box1d.hh`.

Referenced by `box1d()`, and `mln::image1d< T >::image1d()`.

9.107.2.4 `mln::box2d mln::make::box2d (unsigned nrows, unsigned ncols) [inline]`

Create an [mln::box2d](#).

Parameters

[in] *nrows* Number of rows.

[in] *ncols* Number of columns.

Precondition

`nrows != 0 and ncols != 0.`

Returns

A 2D box.

Definition at line 78 of file `make/box2d.hh`.

Referenced by `mln::image2d< T >::image2d()`, and `mln::io::pnm::load()`.

9.107.2.5 mln::box2d mln::make::box2d (def::coord *min_row*, def::coord *min_col*, def::coord *max_row*, def::coord *max_col*) [inline]

Create an [mln::box2d](#).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

- [in] *min_row* Index of the top most row.
- [in] *min_col* Index of the left most column.
- [in] *max_row* Index of the botton most row.
- [in] *max_col* Index of the right most column.

Precondition

```
max_row >= min_row and max_col >= min_col.
```

Returns

A 2D box.

Definition at line 88 of file make/box2d.hh.

9.107.2.6 mln::box2d_h mln::make::box2d_h (def::coord *min_row*, def::coord *min_col*, def::coord *max_row*, def::coord *max_col*) [inline]

Create an [mln::box2d_h](#).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

- [in] *min_row* Index of the top most row.
- [in] *min_col* Index of the left most column.
- [in] *max_row* Index of the botton most row.
- [in] *max_col* Index of the right most column.

Precondition

```
max_row >= min_row and max_col >= min_col.
```

Returns

A 2D_H box.

Definition at line 85 of file make/box2d_h.hh.

References [point2d_h\(\)](#).

9.107.2.7 `mln::box2d_h mln::make::box2d_h (unsigned nrows, unsigned ncols) [inline]`

Create an [mln::box2d_h](#).

Parameters

- [in] *nrows* Number of rows.
- [in] *ncols* Number of columns.

Precondition

`nrows != 0 and ncols != 0.`

Returns

A 2D_H box.

Definition at line 75 of file `make/box2d_h.hh`.

References `point2d_h()`.

9.107.2.8 `mln::box3d mln::make::box3d (unsigned nslis, unsigned nrows, unsigned ncols) [inline]`

Create an [mln::box3d](#).

Parameters

- [in] *nslis* Number of slices.
- [in] *nrows* Number of rows.
- [in] *ncols* Number of columns.

Precondition

`ninds != 0 and ncols != 0 and nslis != 0.`

Returns

A 3D box.

Definition at line 80 of file `make/box3d.hh`.

Referenced by `image3d()`, and `mln::image3d< T >::image3d()`.

9.107.2.9 `mln::box3d mln::make::box3d (def::coord min_sli, def::coord min_row, def::coord min_col, def::coord max_sli, def::coord max_row, def::coord max_col) [inline]`

Create an [mln::box3d](#).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

- [in] *min_sli* Index of the lowest slice.
- [in] *min_row* Index of the top most row.

- [in] *min_col* Index of the left most column.
- [in] *max_sli* Index of the highest slice.
- [in] *max_row* Index of the bottom most row.
- [in] *max_col* Index of the right most column.

Precondition

```
max_sli >= min_sli.
max_row >= min_row.
max_col >= min_col.
```

Returns

A 3D box.

Definition at line 92 of file make/box3d.hh.

9.107.2.10 `template<unsigned D, typename G > p_set< complex_psite< D, G > > mln::make::cell (const complex_psite< D, G > & f) [inline]`

Compute the set of faces of the cell corresponding to the facet *f*.

Precondition

f is a facet (it does not belong to any face of higher dimension).

Returns

An `mln::p_set` of sites (faces) containing the attachment.

Definition at line 63 of file cell.hh.

References `mln::topo::is_facet()`, and `mln::complex_psite< D, G >::n()`.

Referenced by `attachment()`, and `detachment()`.

9.107.2.11 `template<typename T, typename U > util::couple<T,U> mln::make::couple (const T & val1, const T & val2)`

Construct an `mln::util::couple` on-the-fly.

Referenced by `mln::labeling::blobs_and_compute()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::transform::distance_and_influence_zone_geodesic()`, and `mln::labeling::value_and_compute()`.

9.107.2.12 `template<unsigned D, typename G, typename V > p_set< complex_psite< D, G > > mln::make::detachment (const complex_psite< D, G > & f, const complex_image< D, G, V > & ima) [inline]`

Compute the detachment of the cell corresponding to the facet *f* to the image *ima*.

Precondition

f is a facet (it does not belong to any face of higher dimension).
ima is an image of Boolean values.

Returns

a set of faces containing the detachment.

We do not use the formal definition of the detachment here (see `coupric.08.pami`). We use the following (equivalent) definition: an N-face F in CELL is not in the detachment of CELL from IMA if it is adjacent to at least an (N-1)-face or an (N+1)-face that does not belong to CELL.

Definition at line 68 of file `detachment.hh`.

References `cell()`, and `mln::topo::is_facet()`.

Referenced by `mln::topo::detach()`.

9.107.2.13 `mln::dpoint2d_h mln::make::dpoint2d_h (def::coord row, def::coord col)` **[inline]**

Create an `mln::dpoint2d_h`.

Parameters

[in] *row* Row coordinate.

[in] *col* Column coordinate.

Returns

A 2D dpoint.

Definition at line 55 of file `make/dpoint2d_h.hh`.

9.107.2.14 `template<typename G , typename P > p_edges< G, pw::cst_< P > >` `mln::make::dummy_p_edges (const Graph< G > & g_, const P & dummy_site)`

Create a `p_edges` which associate a graph element to a constant site.

Parameters

[in] *g_* A graph.

[in] *dummy_site* The dummy site mapped to graph edges.

Returns

A `p_edges`.

Definition at line 77 of file `dummy_p_edges.hh`.

9.107.2.15 `template<typename G > p_edges< G > mln::make::dummy_p_edges (const Graph<` `G > & g)`

Create a `p_edges` which associate a graph element to a constant site.

0 (int) is used as dummy site.

Parameters

[in] *g* A graph.

Returns

A [p_edges](#).

Definition at line 93 of file dummy_p_edges.hh.

9.107.2.16 `template<typename G , typename P > p_vertices< G, pw::cst_< P > >
mln::make::dummy_p_vertices (const Graph< G > & g_, const P & dummy_site)`

Create a [p_vertices](#) which associate a graph element to a constant site.

Parameters

[in] *g_* A graph.

[in] *dummy_site* The dummy site mapped to graph vertices.

Returns

A [p_vertices](#).

Definition at line 77 of file dummy_p_vertices.hh.

9.107.2.17 `template<typename G > p_vertices< G > mln::make::dummy_p_vertices (const
Graph< G > & g)`

Create a [p_vertices](#) which associate a graph element to a constant site.

0 (int) is used as dummy site.

Parameters

[in] *g* A graph.

Returns

A [p_vertices](#).

Definition at line 93 of file dummy_p_vertices.hh.

9.107.2.18 `template<typename V , typename G > mln::edge_image< void, V, G >
mln::make::edge_image (const Graph< G > & g, const fun::i2v::array< V > & fv)
[inline]`

Construct an edge image.

Parameters

[in] *g* A graph.

[in] *fv* A function mapping edge ids to values.

Returns

an edge image.

Definition at line 141 of file make/edge_image.hh.

9.107.2.19 `template<typename FV , typename G > mln::edge_image< void, typename FV::result, G > mln::make::edge_image (const Graph< G > & g, const Function_v2v< FV > & fv)`

Construct an edge image.

Parameters

- [in] *g* A graph.
- [in] *fv* A function mapping edge ids to values.

Returns

an edge image.

Definition at line 155 of file make/edge_image.hh.

9.107.2.20 `template<typename FP , typename FV , typename G > mln::edge_image< typename FP::result, typename FV::result, G > mln::make::edge_image (const Graph< G > & g_, const Function_v2v< FP > & fp, const Function_v2v< FV > & fv) [inline]`

Construct an edge image.

Parameters

- [in] *g_* A graph.
- [in] *fp* A function mapping edge ids to sites.
- [in] *fv* A function mapping edge ids to values.

Returns

an edge image.

Definition at line 179 of file make/edge_image.hh.

9.107.2.21 `template<typename P , typename V , typename G , typename FP , typename FV > mln::edge_image< typename FP::result, typename FV::result, G > mln::make::edge_image (const mln::vertex_image< P, V, G > & v_ima_, const p_edges< G, FP > pe, const Function_vv2v< FV > & fv_) [inline]`

Construct an edge image.

Parameters

- [in] *v_ima_* A vertex image.
- [in] *pe* A [p_edges](#) mapping graph elements to sites.
- [in] *fv_* A function mapping two vertex ids to a value. The result is associated to the corresponding edge.

Returns

an edge image.

Definition at line 199 of file make/edge_image.hh.

9.107.2.22 `template<typename P , typename V , typename G , typename FV > mln::edge_image< void, typename FV::result, G > mln::make::edge_image (const mln::vertex_image< P, V, G > & v_ima_ , const Function_vv2v< FV > & fv_) [inline]`

Construct an edge image.

Parameters

[in] *v_ima_* A vertex image.

[in] *fv_* A function mapping two vertices' values to a value. The result is associated to the corresponding edge.

Returns

an edge image without localization information mapped to graph elements.

Definition at line 225 of file make/edge_image.hh.

9.107.2.23 `template<typename P , typename V , typename G , typename F > mln::edge_image< void, bool, G > mln::make::edge_image (const mln::vertex_image< P, V, G > & v_ima_ , const Function_v2b< F > & fv_) [inline]`

Construct an edge image.

Parameters

[in] *v_ima_* A vertex image.

[in] *fv_* A predicate on a vertex's value. The (Boolean) result is associated to the edges adjacent to the vertex.

Returns

an edge image without localization information mapped to graph elements.

Definition at line 250 of file make/edge_image.hh.

References mln::data::fill().

9.107.2.24 `template<typename T , unsigned N> algebra::h_mat< mlc_sqrt_int(N), T > mln::make::h_mat (const T(&) tab[N]) [inline]`

Create an mln::algebra::mat<n,n,T>.

Definition at line 60 of file make/h_mat.hh.

Referenced by mln::fun::x2x::rotation< n, C >::rotation().

9.107.2.25 `template<typename V , unsigned L> mln::image1d< V > mln::make::image (V(&) values[L])`

Create an [image1d](#) from an 1D array of values.

Parameters

[in] *values* 1D array.

Returns

A 1D image.

Definition at line 85 of file make/image.hh.

9.107.2.26 `template<typename V , unsigned R, unsigned C> mln::image2d< V >
mln::make::image (V(&) values[R][C])`

Create an [image2d](#) from an 2D array of values.

Parameters

[in] *values* 2D array.

Returns

A 2D image.

Definition at line 97 of file make/image.hh.

References mln::opt::at().

9.107.2.27 `template<typename V , unsigned S, unsigned R, unsigned C> mln::image3d< V >
mln::make::image (V(&) values[S][R][C])`

Create an [image3d](#) from an 3D array of values.

Parameters

[in] *values* 3D array.

Returns

A 3D image.

Definition at line 112 of file make/image.hh.

References mln::opt::at().

9.107.2.28 `template<typename V , unsigned S> mln::image2d< V > mln::make::image2d (V(&)
values[S])`

Create an [image2d](#) from an 2D array of values.

Parameters

[in] *values* 2D array.

Returns

A 2D image.

Definition at line 61 of file make/image2d.hh.

9.107.2.29 `template<typename I> mln::image3d< typename I::value > mln::make::image3d (const Image< I > & ima) [inline]`

Create an [image3d](#) from a 2D image.

Definition at line 92 of file make/image3d.hh.

References [box3d\(\)](#), and [mln::data::paste\(\)](#).

9.107.2.30 `template<typename I> mln::image3d< typename I::value > mln::make::image3d (const util::array< I > & ima) [inline]`

Create an [image3d](#) from an array of 2D images.

Definition at line 70 of file make/image3d.hh.

References [box3d\(\)](#), [mln::util::array< T >::is_empty\(\)](#), [mln::util::array< T >::nelements\(\)](#), [mln::data::paste\(\)](#), [mln::box< P >::pmax\(\)](#), and [mln::box< P >::pmin\(\)](#).

Referenced by [mln::io::pnms::load\(\)](#).

9.107.2.31 `template<typename I, typename N> util::graph mln::make::influence_zone_adjacency_graph (const Image< I > & iz, const Neighborhood< N > & nbh, const typename I::value & nlabels) [inline]`

Create a graph from an influence zone image.

Parameters

[in] *iz* influence zone image.

[in] *nbh* A neighborhood.

[in] *nlabels* number of influence zone in *iz*.

Returns

[util::graph Graph](#) based on the adjacency of the influence zones.

Definition at line 175 of file influence_zone_adjacency_graph.hh.

9.107.2.32 `template<unsigned n, unsigned m, typename T> algebra::mat< n, m, T > mln::make::mat (const T(&) tab[n *m]) [inline]`

Create an [mln::algebra::mat<n,m,T>](#).

Parameters

[in] *tab* Array of values.

Precondition

The array dimension has to be $n * m$.

Definition at line 61 of file make/mat.hh.

9.107.2.33 `template<typename T > util::ord_pair< T > mln::make::ord_pair (const T & val1, const T & val2) [inline]`

Construct an [mln::util::ord_pair](#) on-the-fly.

Definition at line 277 of file ord_pair.hh.

9.107.2.34 `template<typename W , typename G > p_edges< G, fun::i2v::array< util::site_pair< typename W::site > > > > mln::make::p_edges_with_mass_centers (const Image< W > & wst_, const Graph< G > & g_) [inline]`

Construct a [p_edges](#) from a watershed image and a region adjacency graph (RAG).

Map each graph edge to a pair of mass centers of two adjacent regions.

Parameters

wst_ A watershed image.

g_ A region adjacency graph.

Returns

A [p_edges](#).

See also

[edge_image](#), [p_edges](#), [make::region_adjacency_graph](#)

Definition at line 81 of file p_edges_with_mass_centers.hh.

References [mln::labeling::compute\(\)](#).

9.107.2.35 `template<typename W , typename G > p_vertices< G, fun::i2v::array< typename W::site > > > mln::make::p_vertices_with_mass_centers (const Image< W > & wst_, const Graph< G > & g_) [inline]`

Construct a [p_vertices](#) from a watershed image and a region adjacency graph (RAG).

Map each graph vertex to the mass center of its corresponding region.

Parameters

wst_ A watershed image.

g_ A region adjacency graph.

Returns

A [p_vertices](#).

See also

[edge_image](#), [vertex_image](#), [p_vertices](#), [p_edges](#), [make::region_adjacency_graph](#)

Definition at line 77 of file p_vertices_with_mass_centers.hh.

References [mln::labeling::compute\(\)](#).

9.107.2.36 `template<typename I> mln::util::pix< I> mln::make::pix (const Image< I> & ima, const typename I::psite & p) [inline]`

Create an [mln::util::pix](#) from an image `ima` and a psite `p`.

Parameters

[in] `ima` The input image.

[in] `p` The point site.

Returns

An [mln::util::pix](#).

Definition at line 58 of file `make/pix.hh`.

9.107.2.37 `template<typename I> mln::pixel< I> mln::make::pixel (Image< I> & ima, const typename I::psite & p) [inline]`

Create a [mln::pixel](#) from a mutable image `ima` and a point `p`.

Definition at line 63 of file `make/pixel.hh`.

9.107.2.38 `template<typename I> mln::pixel< const I> mln::make::pixel (const Image< I> & ima, const typename I::psite & p) [inline]`

Create a [mln::pixel](#) from a constant image `ima` and a point `p`.

Definition at line 55 of file `make/pixel.hh`.

9.107.2.39 `mln::point2d_h mln::make::point2d_h (def::coord row, def::coord col) [inline]`

Create an [mln::point2d_h](#).

Parameters

[in] `row` Row coordinate.

[in] `col` Column coordinate.

Returns

A 2D point.

Definition at line 55 of file `make/point2d_h.hh`.

Referenced by `box2d_h()`.

9.107.2.40 `template<typename I, typename N> util::couple< util::graph, typename mln::trait::concrete< I>::ret> mln::make::rag_and_labeled_wsl (const Image< I> & wshd_, const Neighborhood< N> & nbh_, const typename I::value & nbasins) [inline]`

Create a region adjacency graph and a label image of the watershed line from a watershed image.

Parameters

- [in] *wshd_* Watershed image.
- [in] *nbh_* [Neighborhood](#)
- [in] *nbasins* Number of influence zone in *wshd*.

Returns

A couple. First element is the graph, second element is an image with a labeled watershed line.

```

|-----|           |-----|
| 1 1 1 0 2 2 0 3 |           | . . . 1 . . 2 . |
| 1 1 0 2 2 2 0 3 |           | . . 1 . . . 2 . |
| 1 0 4 0 2 0 3 3 |           | . 1 . 3 . 4 . . |
| 0 4 4 4 0 5 0 3 |           | 1 . . . 5 . 6 . |
|-----|           |-----|

```

Watershed image Labeled watershed line
(watershed line labeled with 0)

```

|
|
|
v

```

```

1 -- 2 - 3
 \ / /
 4 -- 5

```

Region Adjacency graph (RAG)

Definition at line 229 of file `rag_and_labeled_wsl.hh`.

9.107.2.41 `template<typename I, typename N > util::graph mln::make::region_adjacency_graph (const Image< I > & wshd_, const Neighborhood< N > & nbh, const typename I::value & nbasins) [inline]`

Create a region adjacency graph from a watershed image.

Parameters

- [in] *wshd_* watershed image.
- [in] *nbh* A neighborhood.
- [in] *nbasins* number of influence zone in *wshd*.

Returns

[util::graph Graph](#) based on the adjacency of the influence zones.

Definition at line 179 of file `region_adjacency_graph.hh`.

9.107.2.42 `template<typename V, typename F > fun::i2v::array< V > mln::make::relabelfun (const Function_v2b< F > & fv2b, const V & nlabels, V & new_nlabels) [inline]`

Create a i2v function from a v2b function.

This function can be used to relabel a labeled image.

Parameters

- [in] *fv2b* A v2b function.
- [in] *nlabels* The number of labels.
- [in] *new_nlabels* The number of labels after relabeling.

Returns

a i2v function.

See also

[mln::labeling::relabel](#)

Definition at line 83 of file relabelfun.hh.

References mln::literal::zero.

Referenced by mln::labeling::pack(), mln::labeling::pack_inplace(), mln::labeling::relabel(), mln::labeled_image_base< I, E >::relabel(), and mln::labeling::relabel_inplace().

9.107.2.43 `template<typename V , typename F > fun::i2v::array< V > mln::make::relabelfun (const Function_v2v< F > & fv2v, const V & nlabels, V & new_nlabels) [inline]`

Create a i2v function from a v2v function.

This function can be used to relabel a labeled image.

Parameters

- [in] *fv2v* A v2v function. This function maps an id to an already existing one.
- [in] *nlabels* The number of labels.
- [in] *new_nlabels* The number of labels after relabeling.

Returns

a i2v function.

See also

[mln::labeling::relabel](#)

Definition at line 106 of file relabelfun.hh.

References mln::literal::zero.

9.107.2.44 `template<typename T > algebra::vec< 1, T > mln::make::vec (const T & v_0) [inline]`

Create an mln::algebra::vec<n,T>.

Parameters

- [in] *v_0* First coordinate.

Returns

A 1D vector.

Definition at line 91 of file make/vec.hh.

9.107.2.45 `template<typename T > algebra::vec< 4, T > mln::make::vec (const T & v_0, const T & v_1, const T & v_2, const T & v_3) [inline]`

Create an `mln::algebra::vec<4,T>`.

Parameters

- [in] `v_0` First coordinate.
- [in] `v_1` Second coordinate.
- [in] `v_2` Third coordinate.
- [in] `v_3` Fourth coordinate.

Returns

A 4D vector.

Definition at line 121 of file `make/vec.hh`.

9.107.2.46 `template<typename T > algebra::vec< 3, T > mln::make::vec (const T & v_0, const T & v_1, const T & v_2) [inline]`

Create an `mln::algebra::vec<3,T>`.

Parameters

- [in] `v_0` First coordinate.
- [in] `v_1` Second coordinate.
- [in] `v_2` Third coordinate.

Returns

A 3D vector.

Definition at line 110 of file `make/vec.hh`.

9.107.2.47 `template<typename T > algebra::vec< 2, T > mln::make::vec (const T & v_0, const T & v_1) [inline]`

Create an `mln::algebra::vec<2,T>`.

Parameters

- [in] `v_0` First coordinate.
- [in] `v_1` Second coordinate.

Returns

A 2D vector.

Definition at line 100 of file `make/vec.hh`.

9.107.2.48 `template<typename G , typename FV > mln::vertex_image< void, typename FV::result, G > mln::make::vertex_image (const Graph< G > & g, const Function_v2v< FV > & fv)`

Construct a vertex image.

Parameters

- [in] *g* A graph.
- [in] *fv* A function mapping vertex ids to values.

Returns

A vertex image.

Definition at line 77 of file make/vertex_image.hh.

9.107.2.49 `template<typename FP , typename FV , typename G > mln::vertex_image< typename FP::result, typename FV::result, G > mln::make::vertex_image (const Graph< G > & g_, const Function_v2v< FP > & fp, const Function_v2v< FV > & fv)`

Construct a vertex image.

Parameters

- [in] *g_* A graph.
- [in] *fp* A function mapping vertex ids to sites.
- [in] *fv* A function mapping vertex ids to values.

Returns

A vertex image.

Definition at line 92 of file make/vertex_image.hh.

9.107.2.50 `template<typename I , typename N > p_vertices< util::graph, fun::i2v::array< typename I::site > > mln::make::voronoi (Image< I > & ima_, Image< I > & orig_, const Neighborhood< N > & nbh) [inline]`

Apply the Voronoi algorithm on *ima_* with the original image *orig_* for node computing with neighborhood *nbh*.

Parameters

- [in] *ima_* The labeling image.
- [in] *orig_* The original image.
- [in] *nbh* The neighborhood for computing algorithm.

Returns

The computed graph.

Definition at line 68 of file voronoi.hh.

References `mln::util::graph::add_edge()`, `mln::util::graph::add_vertex()`, and `mln::estim::min_max()`.

9.107.2.51 `template<typename W , typename F > mln::w_window< typename W::dpsite, typename F::result > mln::make::w_window (const Window< W > & win, const Function_v2v< F > & wei) [inline]`

Create a [mln::w_window](#) from a window and a weight function.

Parameters

[in] *win* A simple window.

[in] *wei* A weight function.

Returns

A weighted window.

Definition at line 63 of file `make/w_window.hh`.

References `mln::w_window< D, W >::insert()`, and `mln::literal::origin`.

9.107.2.52 `template<typename W , unsigned M> mln::w_window< mln::dpoint1d, W > mln::make::w_window1d (W(&) weights[M]) [inline]`

Create a 1D [mln::w_window](#) from an array of weights.

Parameters

[in] *weights* Array.

Precondition

The array size, *M*, has to be a square of an odd integer.

Returns

A 1D weighted window.

Definition at line 63 of file `w_window1d.hh`.

References `mln::w_window< D, W >::insert()`.

Referenced by `w_window1d_int()`.

9.107.2.53 `template<unsigned M> mln::w_window1d_int mln::make::w_window1d_int (int(&) weights[M]) [inline]`

Create a [mln::w_window1d_int](#).

Parameters

[in] *weights* Array of integers.

Precondition

The array size, *M*, has to be a square of an odd integer.

Returns

A 1D int-weighted window.

Definition at line 61 of file make/w_window1d_int.hh.

References [w_window1d\(\)](#).

9.107.2.54 `template<typename W , unsigned S> mln::w_window< mln::dpoint2d, W >
mln::make::w_window2d (W(&) weights[S]) [inline]`

Create a 2D [mln::w_window](#) from an array of weights.

Parameters

[in] *weights* Array.

Precondition

The array size, *S*, has to be a square of an odd integer.

Returns

A 2D weighted window.

Definition at line 62 of file w_window2d.hh.

Referenced by [w_window2d_int\(\)](#).

9.107.2.55 `template<unsigned M> mln::w_window2d_int mln::make::w_window2d_int (int(&
weights[M]) [inline]`

Create a [mln::w_window2d_int](#).

Parameters

[in] *weights* Array of integers.

Precondition

The array size, *M*, has to be a square of an odd integer.

Returns

A 2D int-weighted window.

Definition at line 61 of file make/w_window2d_int.hh.

References [w_window2d\(\)](#).

9.107.2.56 `template<typename W , unsigned M> mln::w_window< mln::dpoint3d, W >
mln::make::w_window3d (W(&) weights[M]) [inline]`

Create a 3D [mln::w_window](#) from an array of weights.

Parameters

[in] *weights* Array.

Precondition

The array size, M, has to be a cube of an odd integer.

Returns

A 3D weighted window.

Definition at line 64 of file w_window3d.hh.

References mln::w_window< D, W >::insert().

Referenced by w_window3d_int().

9.107.2.57 `template<unsigned M> mln::w_window3d_int mln::make::w_window3d_int (int(&)
weights[M]) [inline]`

Create a [mln::w_window3d_int](#).

Parameters

[in] *weights* Array of integers.

Precondition

The array size, M, has to be a cube of an odd integer.

Returns

A 3D int-weighted window.

Definition at line 61 of file make/w_window3d_int.hh.

References w_window3d().

9.107.2.58 `template<typename D , typename W , unsigned L> mln::w_window< D, W >
mln::make::w_window_directional (const Gdpoint< D > & dp, W(&) weights[L])
[inline]`

Create a directional centered weighted window.

Parameters

[in] *dp* A delta-point to set the orientation.

[in] *weights* An array of weights.

Returns

A weighted window.

The window length L has to be odd.

Definition at line 61 of file w_window_directional.hh.

References mln::w_window< D, W >::insert(), and mln::literal::zero.

9.108 mln::math Namespace Reference

Namespace of mathematical routines.

Functions

- `template<typename T >`
`T abs (const T &v)`
Generic version.
- `template<unsigned n>`
`value::int_u< n > abs (const value::int_u< n > &v)`
Specialization for `mln::value::int_u`.
- `int abs (int v)`
Specializations for existing overloads of `std::abs`.

9.108.1 Detailed Description

Namespace of mathematical routines.

9.108.2 Function Documentation

9.108.2.1 `template<typename T > T mln::math::abs (const T & v) [inline]`

Generic version.

Definition at line 74 of file `math/abs.hh`.

Referenced by `abs()`, and `mln::morpho::line_gradient()`.

9.108.2.2 `int mln::math::abs (int v) [inline]`

Specializations for existing overloads of `std::abs`.

Reference: ISO/IEC 14882:2003 C++ standard, section 26.5 (C Library, [`lib.c.math`]).

Definition at line 79 of file `math/abs.hh`.

References `abs()`.

9.108.2.3 `template<unsigned n> value::int_u< n > mln::math::abs (const value::int_u< n > & v) [inline]`

Specialization for `mln::value::int_u`.

Definition at line 88 of file `math/abs.hh`.

9.109 mln::metal Namespace Reference

Namespace of meta-programming tools.

Namespaces

- namespace [impl](#)
Implementation namespace of metal namespace.
- namespace [math](#)
Namespace of static mathematical functions.

Classes

- struct [ands](#)
Ands type.
- struct [converts_to](#)
"converts-to" check.
- struct [equal](#)
Definition of a static 'equal' test.
- struct [goes_to](#)
"goes-to" check.
- struct [is](#)
"is" check.
- struct [is_a](#)
"is_a" check.
- struct [is_not](#)
"is_not" check.
- struct [is_not_a](#)
"is_not_a" static Boolean expression.

9.109.1 Detailed Description

Namespace of meta-programming tools.

9.110 mln::metal::impl Namespace Reference

Implementation namespace of metal namespace.

9.110.1 Detailed Description

Implementation namespace of metal namespace.

9.111 mln::metal::math Namespace Reference

Namespace of static mathematical functions.

Namespaces

- namespace [impl](#)
Implementation namespace of [metal::math](#) namespace.

9.111.1 Detailed Description

Namespace of static mathematical functions.

9.112 mln::metal::math::impl Namespace Reference

Implementation namespace of [metal::math](#) namespace.

9.112.1 Detailed Description

Implementation namespace of [metal::math](#) namespace.

9.113 mln::morpho Namespace Reference

Namespace of mathematical morphology routines.

Namespaces

- namespace [approx](#)
Namespace of approximate mathematical morphology routines.
- namespace [attribute](#)
Namespace of attributes used in mathematical morphology.
- namespace [elementary](#)
Namespace of image processing routines of elementary mathematical morphology.
- namespace [impl](#)
Namespace of mathematical morphology routines implementations.

- namespace [reconstruction](#)
Namespace of morphological reconstruction routines.
- namespace [tree](#)
Namespace of morphological tree-related routines.
- namespace [watershed](#)
Namespace of morphological watershed routines.

Functions

- `template<typename I >`
`mln::trait::concrete< I >::ret complementation (const Image< I > &input)`
- `template<typename I >`
`void complementation_inplace (Image< I > &input)`
- `template<typename I , typename W >`
`mln::trait::concrete< I >::ret contrast (const Image< I > &input, const Window< W > &win)`
- `template<typename I , typename W >`
`mln::trait::concrete< I >::ret dilation (const Image< I > &input, const Window< W > &win)`
Morphological dilation.
- `template<typename I , typename W >`
`mln::trait::concrete< I >::ret erosion (const Image< I > &input, const Window< W > &win)`
Morphological erosion.
- `template<typename Op , typename I , typename W >`
`mln::trait::concrete< I >::ret general (const Op &op, const Image< I > &input, const Window< W > &win)`
Morphological general routine.
- `template<typename I , typename W >`
`mln::trait::concrete< I >::ret gradient (const Image< I > &input, const Window< W > &win)`
Morphological gradient.
- `template<typename I , typename W >`
`mln::trait::concrete< I >::ret gradient_external (const Image< I > &input, const Window< W > &win)`
Morphological external gradient.
- `template<typename I , typename W >`
`mln::trait::concrete< I >::ret gradient_internal (const Image< I > &input, const Window< W > &win)`
Morphological internal gradient.
- `template<typename I , typename Wh , typename Wm >`
`mln::trait::concrete< I >::ret hit_or_miss (const Image< I > &input, const Window< Wh > &win_
hit, const Window< Wm > &win_miss)`
Morphological hit-or-miss.

- `template<typename I, typename Wh, typename Wm >`
`mln::trait::concrete< I >::ret hit_or_miss_background_closing (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`
Morphological hit-or-miss closing of the background.
- `template<typename I, typename Wh, typename Wm >`
`mln::trait::concrete< I >::ret hit_or_miss_background_opening (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`
Morphological hit-or-miss opening of the background.
- `template<typename I, typename Wh, typename Wm >`
`mln::trait::concrete< I >::ret hit_or_miss_closing (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`
Morphological hit-or-miss closing.
- `template<typename I, typename Wh, typename Wm >`
`mln::trait::concrete< I >::ret hit_or_miss_opening (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`
Morphological hit-or-miss opening.
- `template<typename I, typename W, typename O >`
`void laplacian (const Image< I > &input, const Window< W > &win, Image< O > &output)`
- `template<typename V >`
`edge_image< util::site_pair< point2d >, V, util::graph > line_gradient (const mln::image2d< V > &ima)`
Create a line graph image representing the gradient norm of a [mln::image2d](#).
- `template<typename L, typename I, typename N >`
`mln::trait::ch_value< I, L >::ret meyer_wst (const Image< I > &input, const Neighborhood< N > &nbh, L &nbasins)`
Meyer's Watershed Transform (WST) algorithm.
- `template<typename L, typename I, typename N >`
`mln::trait::ch_value< I, L >::ret meyer_wst (const Image< I > &input, const Neighborhood< N > &nbh)`
Meyer's Watershed Transform (WST) algorithm, with no count of basins.
- `template<typename I, typename J >`
`mln::trait::concrete< I >::ret min (const Image< I > &lhs, const Image< J > &rhs)`
- `template<typename I, typename J >`
`void min_inplace (Image< I > &lhs, const Image< J > &rhs)`
- `template<typename I, typename J >`
`mln::trait::concrete< I >::ret minus (const Image< I > &lhs, const Image< J > &rhs)`
- `template<typename I, typename J >`
`mln::trait::concrete< I >::ret plus (const Image< I > &lhs, const Image< J > &rhs)`
- `template<typename I, typename W >`
`mln::trait::concrete< I >::ret rank_filter (const Image< I > &input, const Window< W > &win, unsigned k)`
Morphological rank_filter.

- `template<typename I, typename Wfg, typename Wbg >`
`mln::trait::concrete< I >::ret thick_miss (const Image< I > &input, const Window< Wfg > &win_
fg, const Window< Wbg > &win_bg)`
- `template<typename I, typename Wfg, typename Wbg >`
`mln::trait::concrete< I >::ret thickening (const Image< I > &input, const Window< Wfg > &win_
fg, const Window< Wbg > &win_bg)`
- `template<typename I, typename Wfg, typename Wbg >`
`mln::trait::concrete< I >::ret thin_fit (const Image< I > &input, const Window< Wfg > &win_fg,
const Window< Wbg > &win_bg)`
- `template<typename I, typename Wfg, typename Wbg >`
`mln::trait::concrete< I >::ret thinning (const Image< I > &input, const Window< Wfg > &win_fg,
const Window< Wbg > &win_bg)`
Morphological thinning.
- `template<typename I, typename W >`
`mln::trait::concrete< I >::ret top_hat_black (const Image< I > &input, const Window< W >
&win)`
Morphological black top-hat (for background / dark objects).
- `template<typename I, typename W >`
`mln::trait::concrete< I >::ret top_hat_self_complementary (const Image< I > &input, const Win-
dow< W > &win)`
Morphological self-complementary top-hat.
- `template<typename I, typename W >`
`mln::trait::concrete< I >::ret top_hat_white (const Image< I > &input, const Window< W >
&win)`
Morphological white top-hat (for object / light objects).

9.113.1 Detailed Description

Namespace of mathematical morphology routines.

9.113.2 Function Documentation

9.113.2.1 `template<typename I > mln::trait::concrete< I >::ret mln::morpho::complementation (const Image< I > & input) [inline]`

Morphological complementation: either a logical "not" (if morpho on sets) or an arithmetical complementation (if morpho on functions).

Definition at line 116 of file complementation.hh.

Referenced by `hit_or_miss_background_closing()`, `hit_or_miss_background_opening()`, `hit_or_miss_closing()`, and `thinning()`.

9.113.2.2 `template<typename I > void mln::morpho::complementation_inplace (Image< I > & input) [inline]`

Morphological complementation, inplace version: either a logical "not" (if morpho on sets) or an arithmetical complementation (if morpho on functions).

Definition at line 130 of file complementation.hh.

9.113.2.3 `template<typename I , typename W > mln::trait::concrete< I >::ret
mln::morpho::contrast (const Image< I > & input, const Window< W > & win)
[inline]`

Morphological contrast operator (based on top-hats).

This operator is $Id + wth_B - bth_B$.

Definition at line 57 of file contrast.hh.

References plus(), top_hat_black(), and top_hat_white().

9.113.2.4 `template<typename I , typename W > mln::trait::concrete< I >::ret
mln::morpho::dilation (const Image< I > & input, const Window< W > & win)
[inline]`

Morphological dilation.

Definition at line 162 of file dilation.hh.

References general().

Referenced by gradient(), gradient_external(), hit_or_miss_background_opening(), hit_or_miss_opening(), laplacian(), mln::morpho::opening::approx::structural(), and mln::morpho::closing::approx::structural().

9.113.2.5 `template<typename I , typename W > mln::trait::concrete< I >::ret
mln::morpho::erosion (const Image< I > & input, const Window< W > & win)
[inline]`

Morphological erosion.

Definition at line 163 of file erosion.hh.

References general().

Referenced by gradient(), gradient_internal(), laplacian(), mln::morpho::opening::approx::structural(), and mln::morpho::closing::approx::structural().

9.113.2.6 `template<typename Op , typename I , typename W > mln::trait::concrete< I >::ret
mln::morpho::general (const Op & op, const Image< I > & input, const Window<
W > & win) [inline]`

Morphological general routine.

Definition at line 168 of file general.hh.

Referenced by dilation(), and erosion().

9.113.2.7 `template<typename I , typename W > mln::trait::concrete< I >::ret
mln::morpho::gradient (const Image< I > & input, const Window< W > & win)
[inline]`

Morphological gradient.

This operator is $d_B - e_B$.

Definition at line 76 of file `gradient.hh`.

References `dilation()`, `erosion()`, `minus()`, and `mln::test::positive()`.

9.113.2.8 `template<typename I , typename W > mln::trait::concrete< I >::ret
mln::morpho::gradient_external (const Image< I > & input, const Window< W > &
win) [inline]`

Morphological external gradient.

This operator is $d_B - Id$.

Definition at line 110 of file `gradient.hh`.

References `dilation()`, `minus()`, and `mln::test::positive()`.

9.113.2.9 `template<typename I , typename W > mln::trait::concrete< I >::ret
mln::morpho::gradient_internal (const Image< I > & input, const Window< W > &
win) [inline]`

Morphological internal gradient.

This operator is $Id - e_B$.

Definition at line 93 of file `gradient.hh`.

References `erosion()`, `minus()`, and `mln::test::positive()`.

9.113.2.10 `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret
mln::morpho::hit_or_miss (const Image< I > & input, const Window< Wh > &
win_hit, const Window< Wm > & win_miss) [inline]`

Morphological hit-or-miss.

This operator is $HMT_(Bh, Bm) = e_{Bh} \wedge (e_{Bm} \circ C)$.

Definition at line 276 of file `hit_or_miss.hh`.

Referenced by `thickening()`, and `thinning()`.

9.113.2.11 `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret
mln::morpho::hit_or_miss_background_closing (const Image< I > & input, const
Window< Wh > & win_hit, const Window< Wm > & win_miss) [inline]`

Morphological hit-or-miss closing of the background.

This operator is $C \circ HMT_{peBG} \circ C$.

Definition at line 357 of file `hit_or_miss.hh`.

References `complementation()`, `hit_or_miss_background_opening()`, and `hit_or_miss_closing()`.

9.113.2.12 `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret
mln::morpho::hit_or_miss_background_opening (const Image< I > & input, const
Window< Wh > & win_hit, const Window< Wm > & win_miss) [inline]`

Morphological hit-or-miss opening of the background.

This operator is $HMT_{\text{TopeBG}} = HMT_{\text{Tope}}(B_m, B_h) \circ C = d_{-}(B_m) \circ HMT_{-}(B_h, B_m)$.

Definition at line 314 of file hit_or_miss.hh.

References `complementation()`, `dilation()`, `hit_or_miss_opening()`, and `mln::win::sym()`.

Referenced by `hit_or_miss_background_closing()`, and `thick_miss()`.

9.113.2.13 `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret
mln::morpho::hit_or_miss_closing (const Image< I > & input, const Window< Wh
> & win_hit, const Window< Wm > & win_miss) [inline]`

Morphological hit-or-miss closing.

This operator is $C \circ HMT_{\text{Tope}} \circ C$.

Definition at line 337 of file hit_or_miss.hh.

References `complementation()`, and `hit_or_miss_opening()`.

Referenced by `hit_or_miss_background_closing()`.

9.113.2.14 `template<typename I , typename Wh , typename Wm > mln::trait::concrete< I >::ret
mln::morpho::hit_or_miss_opening (const Image< I > & input, const Window< Wh
> & win_hit, const Window< Wm > & win_miss) [inline]`

Morphological hit-or-miss opening.

This operator is $HMT_{\text{Tope}}(B_h, B_m) = d_{-}(B_h) \circ HMT_{-}(B_h, B_m)$.

Definition at line 294 of file hit_or_miss.hh.

References `dilation()`, and `mln::win::sym()`.

Referenced by `hit_or_miss_background_opening()`, `hit_or_miss_closing()`, and `thin_fit()`.

9.113.2.15 `template<typename I , typename W , typename O > void mln::morpho::laplacian (
const Image< I > & input, const Window< W > & win, Image< O > & output)
[inline]`

Morphological laplacian.

This operator is $(d_B - Id) - (Id - e_B)$.

Definition at line 63 of file laplacian.hh.

References `dilation()`, `erosion()`, `mln::data::fill()`, and `minus()`.

9.113.2.16 `template<typename V > edge_image< util::site_pair< point2d >, V, util::graph >
mln::morpho::line_gradient (const mln::image2d< V > & ima)`

Create a line graph image representing the gradient norm of a [mln::image2d](#).

Definition at line 70 of file `line_gradient.hh`.

References `mln::math::abs()`, `mln::image2d< T >::domain()`, `mln::box< P >::has()`, `mln::window< D >::insert()`, and `mln::Box< E >::nsites()`.

9.113.2.17 `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret mln::morpho::meyer_wst (const Image< I > & input, const Neighborhood< N > & nbh, L & nbasins)`

Meyer's Watershed Transform (WST) algorithm.

Parameters

[in] *input* The input image.
 [in] *nbh* The connexity of markers.
 [out] *nbasins* The number of basins.

- `L` is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- `I` is the exact type of the input image.
- `N` is the exact type of the neighborhood used to express *input*'s connexity.

Definition at line 108 of file `meyer_wst.hh`.

References `mln::data::fill()`, `mln::p_priority< P, Q >::front()`, `mln::initialize()`, `mln::p_priority< P, Q >::pop()`, `mln::p_priority< P, Q >::push()`, `mln::labeling::regional_minima()`, and `mln::literal::zero`.

9.113.2.18 `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret mln::morpho::meyer_wst (const Image< I > & input, const Neighborhood< N > & nbh)`

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

Parameters

[in] *input* The input image.
 [in] *nbh* The connexity of markers.

- `L` is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- `I` is the exact type of the input image.
- `N` is the exact type of the neighborhood used to express *input*'s connexity.

Note that the first parameter, `L`, is not automatically valued from the type of the actual argument during implicit instantiation: you have to explicitly pass this parameter at call sites.

Definition at line 202 of file `meyer_wst.hh`.

9.113.2.19 `template<typename I , typename J > mln::trait::concrete< I >::ret mln::morpho::min (const Image< I > & lhs, const Image< J > & rhs) [inline]`

Morphological min: either a logical "and" (if morpho on sets) or an arithmetical min (if morpho on functions).

Definition at line 112 of file `morpho/min.hh`.

9.113.2.20 `template<typename I , typename J > void mln::morpho::min_inplace (Image< I > & lhs, const Image< J > & rhs) [inline]`

Morphological min, inplace version: either a logical "and" (if morpho on sets) or an arithmetical min (if morpho on functions).

Definition at line 125 of file morpho/min.hh.

9.113.2.21 `template<typename I , typename J > mln::trait::concrete< I >::ret mln::morpho::minus (const Image< I > & lhs, const Image< J > & rhs) [inline]`

Morphological minus: either a logical "and not" (if morpho on sets) or an arithmetical minus (if morpho on functions).

Definition at line 87 of file morpho/minus.hh.

Referenced by `gradient()`, `gradient_external()`, `gradient_internal()`, `laplacian()`, `thin_fit()`, `thinning()`, `top_hat_black()`, `mln::morpho::elementary::top_hat_black()`, `top_hat_self_complementary()`, `mln::morpho::elementary::top_hat_self_complementary()`, `top_hat_white()`, and `mln::morpho::elementary::top_hat_white()`.

9.113.2.22 `template<typename I , typename J > mln::trait::concrete< I >::ret mln::morpho::plus (const Image< I > & lhs, const Image< J > & rhs) [inline]`

Morphological plus: either a "logical or" (if morpho on sets) or an "arithmetical plus" (if morpho on functions).

Definition at line 86 of file morpho/plus.hh.

Referenced by `contrast()`, `thick_miss()`, and `thickening()`.

9.113.2.23 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::rank_filter (const Image< I > & input, const Window< W > & win, unsigned k) [inline]`

Morphological `rank_filter`.

Definition at line 213 of file rank_filter.hh.

9.113.2.24 `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I >::ret mln::morpho::thick_miss (const Image< I > & input, const Window< Wfg > & win_fg, const Window< Wbg > & win_bg) [inline]`

Morphological thick-miss.

This operator is `THICK_B = Id + HMTopeBG_B`, where `B = (Bfg, Bbg)`.

Definition at line 59 of file thick_miss.hh.

References `hit_or_miss_background_opening()`, and `plus()`.

9.113.2.25 `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I >::ret mln::morpho::thickening (const Image< I > & input, const Window< Wfg > & win_fg, const Window< Wbg > & win_bg) [inline]`

Morphological thickening.

This operator is THICK_B = Id + HMT_B, where B = (Bfg, Bbg).

Definition at line 88 of file thickening.hh.

References hit_or_miss(), and plus().

Referenced by thinning().

9.113.2.26 `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I >::ret mln::morpho::thin_fit (const Image< I > & input, const Window< Wfg > & win_fg, const Window< Wbg > & win_bg) [inline]`

Morphological thin-fit.

This operator is THIN_B = Id - HMTope_B where B = (Bfg, Bbg).

Definition at line 87 of file thin_fit.hh.

References hit_or_miss_opening(), and minus().

9.113.2.27 `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I >::ret mln::morpho::thinning (const Image< I > & input, const Window< Wfg > & win_fg, const Window< Wbg > & win_bg) [inline]`

Morphological thinning.

This operator is THIN_B = Id - HMT_B, where B = (Bfg, Bbg).

Definition at line 90 of file thinning.hh.

References complementation(), hit_or_miss(), minus(), and thickening().

9.113.2.28 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::top_hat_black (const Image< I > & input, const Window< W > & win) [inline]`

Morphological black top-hat (for background / dark objects).

This operator is clo_B - Id.

Definition at line 102 of file top_hat.hh.

References minus(), and mln::test::positive().

Referenced by contrast().

9.113.2.29 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::top_hat_self_complementary (const Image< I > & input, const Window< W > & win) [inline]`

Morphological self-complementary top-hat.

This operator is

= top_hat_white + top_hat_black
 = (input - opening) + (closing - input)
 = closing - opening.

Definition at line 121 of file top_hat.hh.

References minus(), and mln::test::positive().

9.113.2.30 `template<typename I , typename W > mln::trait::concrete< I >::ret
 mln::morpho::top_hat_white (const Image< I > & input, const Window< W > &
 win) [inline]`

Morphological white top-hat (for object / light objects).

This operator is Id - ope_B.

Definition at line 83 of file top_hat.hh.

References minus(), and mln::test::positive().

Referenced by contrast().

9.114 mln::morpho::approx Namespace Reference

Namespace of approximate mathematical morphology routines.

9.114.1 Detailed Description

Namespace of approximate mathematical morphology routines.

9.115 mln::morpho::attribute Namespace Reference

Namespace of attributes used in mathematical morphology.

Classes

- class [card](#)
Cardinality accumulator class.
- struct [count_adjacent_vertices](#)
Count_Adjacent_Vertices accumulator class.
- struct [height](#)
Height accumulator class.
- struct [sharpness](#)
Sharpness accumulator class.
- class [sum](#)

Suminality accumulator class.

- struct [volume](#)

Volume accumulator class.

9.115.1 Detailed Description

Namespace of attributes used in mathematical morphology.

9.116 mln::morpho::closing::approx Namespace Reference

Namespace of approximate mathematical morphology closing routines.

Functions

- `template<typename I, typename W > mln::trait::concrete< I >::ret structural (const Image< I > &input, const Window< W > &win)`
Approximate of morphological structural closing.

9.116.1 Detailed Description

Namespace of approximate mathematical morphology closing routines.

9.116.2 Function Documentation

- 9.116.2.1** `template<typename I, typename W > mln::trait::concrete< I >::ret mln::morpho::closing::approx::structural (const Image< I > & input, const Window< W > & win) [inline]`

Approximate of morphological structural closing.

This operator is $e_{\{-B\}} \circ d_B$.

Definition at line 65 of file closing/approx/structural.hh.

References `mln::morpho::dilation()`, `mln::morpho::erosion()`, and `mln::win::sym()`.

9.117 mln::morpho::elementary Namespace Reference

Namespace of image processing routines of elementary mathematical morphology.

Functions

- `template<typename I, typename N > mln::trait::concrete< I >::ret closing (const Image< I > &input, const Neighborhood< N > &nbh)`

Morphological elementary closing.

- `template<typename I , typename N > mln_trait_op_minus_twice (typename mln_trait::concrete< I >::ret) laplacian(const Image< I > &input`

Morphological elementary laplacian.

- `template<typename I , typename N > mln_trait::concrete< I >::ret opening (const Image< I > &input, const Neighborhood< N > &nbh)`

Morphological elementary opening.

- `template<typename I , typename N > mln_trait::concrete< I >::ret top_hat_black (const Image< I > &input, const Neighborhood< N > &nbh)`

Morphological elementary black top-hat (for background / dark objects).

- `template<typename I , typename N > mln_trait::concrete< I >::ret top_hat_self_complementary (const Image< I > &input, const Neighborhood< N > &nbh)`

Morphological elementary self-complementary top-hat.

- `template<typename I , typename N > mln_trait::concrete< I >::ret top_hat_white (const Image< I > &input, const Neighborhood< N > &nbh)`

Morphological elementary white top-hat (for object / light objects).

9.117.1 Detailed Description

Namespace of image processing routines of elementary mathematical morphology.

9.117.2 Function Documentation

9.117.2.1 `template<typename I , typename N > mln_trait::concrete< I >::ret mln::morpho::elementary::closing (const Image< I > & input, const Neighborhood< N > & nbh) [inline]`

Morphological elementary closing.

This operator is e o d.

Definition at line 58 of file closing.hh.

Referenced by `top_hat_black()`, and `top_hat_self_complementary()`.

9.117.2.2 `template<typename I , typename N > mln::morpho::elementary::mln_trait_op_minus_twice (typename mln_trait::concrete< I >::ret) const [inline]`

Morphological elementary laplacian.

This operator is $(d - id) - (id - e)$.

9.117.2.3 `template<typename I , typename N > mln::trait::concrete< I >::ret
mln::morpho::elementary::opening (const Image< I > & input, const Neighborhood<
N > & nbh) [inline]`

Morphological elementary opening.

This operator is `d o e`.

Definition at line 58 of file `opening.hh`.

Referenced by `top_hat_self_complementary()`, and `top_hat_white()`.

9.117.2.4 `template<typename I , typename N > mln::trait::concrete< I >::ret
mln::morpho::elementary::top_hat_black (const Image< I > & input, const
Neighborhood< N > & nbh) [inline]`

Morphological elementary black top-hat (for background / dark objects).

This operator is `clo - Id`.

Definition at line 105 of file `elementary/top_hat.hh`.

References `closing()`, `mln::morpho::minus()`, and `mln::test::positive()`.

9.117.2.5 `template<typename I , typename N > mln::trait::concrete< I >::ret
mln::morpho::elementary::top_hat_self_complementary (const Image< I > & input,
const Neighborhood< N > & nbh) [inline]`

Morphological elementary self-complementary top-hat.

This operator is

`= top_hat_white + top_hat_black`

`= (Id - opening) + (closing - Id)`

`= closing - opening`.

Definition at line 125 of file `elementary/top_hat.hh`.

References `closing()`, `mln::morpho::minus()`, `opening()`, and `mln::test::positive()`.

9.117.2.6 `template<typename I , typename N > mln::trait::concrete< I >::ret
mln::morpho::elementary::top_hat_white (const Image< I > & input, const
Neighborhood< N > & nbh) [inline]`

Morphological elementary white top-hat (for object / light objects).

This operator is `Id - ope`.

Definition at line 85 of file `elementary/top_hat.hh`.

References `mln::morpho::minus()`, `opening()`, and `mln::test::positive()`.

9.118 mln::morpho::impl Namespace Reference

Namespace of mathematical morphology routines implementations.

Namespaces

- namespace [generic](#)

Namespace of mathematical morphology routines generic implementations.

9.118.1 Detailed Description

Namespace of mathematical morphology routines implementations.

9.119 mln::morpho::impl::generic Namespace Reference

Namespace of mathematical morphology routines generic implementations.

9.119.1 Detailed Description

Namespace of mathematical morphology routines generic implementations.

9.120 mln::morpho::opening::approx Namespace Reference

Namespace of approximate mathematical morphology opening routines.

Functions

- `template<typename I , typename W > mln::trait::concrete< I >::ret structural (const Image< I > &input, const Window< W > &win)`

Approximate of morphological structural opening.

9.120.1 Detailed Description

Namespace of approximate mathematical morphology opening routines.

9.120.2 Function Documentation

- 9.120.2.1** `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::opening::approx::structural (const Image< I > & input, const Window< W > & win) [inline]`

Approximate of morphological structural opening.

This operator is $d_{\{-B\}} \circ e_B$.

Definition at line 64 of file opening/approx/structural.hh.

References `mln::morpho::dilation()`, `mln::morpho::erosion()`, and `mln::win::sym()`.

9.121 mln::morpho::reconstruction Namespace Reference

Namespace of morphological reconstruction routines.

Namespaces

- namespace [by_dilation](#)
Namespace of morphological reconstruction by dilation routines.
- namespace [by_erosion](#)
Namespace of morphological reconstruction by erosion routines.

9.121.1 Detailed Description

Namespace of morphological reconstruction routines.

9.122 mln::morpho::reconstruction::by_dilation Namespace Reference

Namespace of morphological reconstruction by dilation routines.

9.122.1 Detailed Description

Namespace of morphological reconstruction by dilation routines.

9.123 mln::morpho::reconstruction::by_erosion Namespace Reference

Namespace of morphological reconstruction by erosion routines.

9.123.1 Detailed Description

Namespace of morphological reconstruction by erosion routines.

9.124 mln::morpho::tree Namespace Reference

Namespace of morphological tree-related routines.

Namespaces

- namespace [filter](#)
Namespace for attribute filtering.

Functions

- `template<typename A , typename T >`
`mln::trait::ch_value< typename T::function, typename A::result >::ret compute_attribute_image`
`(const Accumulator< A > &a, const T &t, mln::trait::ch_value< typename T::function, A >::ret`
`*accu_image=0)`
Compute an attribute image using tree with a parent relationship between sites.
- `template<typename A , typename T , typename V >`
`mln::trait::ch_value< typename T::function, typename A::result >::ret compute_attribute_image_-`
`from (const Accumulator< A > &a, const T &t, const Image< V > &values, mln::trait::ch_value<`
`typename T::function, A >::ret *accu_image=0)`
The same as `compute_attribute_image` but uses the values stored by `values` image instead.
- `template<typename I , typename N , typename S >`
`mln::trait::ch_value< I, typename I::psite >::ret compute_parent (const Image< I > &f, const`
`Neighborhood< N > &nbh, const Site_Set< S > &s)`
Compute a tree with a parent relationship between sites.
- `template<typename I , typename N >`
`data< I, p_array< typename I::psite > > dual_input_max_tree (const Image< I > &f, const Image<`
`I > &m, const Neighborhood< N > &nbh)`
Compute the dual input max tree using mask-based connectivity.
- `template<typename I , typename N >`
`data< I, p_array< typename I::psite > > max_tree (const Image< I > &f, const Neighborhood< N`
`> &nbh)`
Compute a canonized max-tree.
- `template<typename I , typename N >`
`data< I, p_array< typename I::psite > > min_tree (const Image< I > &f, const Neighborhood< N`
`> &nbh)`
Compute a canonized min-tree.
- `template<typename T , typename A , typename P , typename W >`
`void propagate_if (const T &tree, Image< A > &a_, const way_of_propagation< W > &prop_,`
`const Function_v2b< P > &pred_, const typename A::value &v)`
- `template<typename T , typename A , typename P >`
`void propagate_if (const T &tree, Image< A > &a_, const desc_propagation &prop_, const`
`Function_v2b< P > &pred_)`
- `template<typename T , typename A , typename W >`
`void propagate_if_value (const T &tree, Image< A > &a_, const way_of_propagation< W >`
`&prop, const typename A::value &v)`
- `template<typename T , typename A , typename W >`
`void propagate_if_value (const T &tree, Image< A > &a_, const way_of_propagation< W >`
`&prop_, const typename A::value &v, const typename A::value &v_prop)`
- `template<typename T , typename A >`
`void propagate_node_to_ancestors (typename A::psite n, const T &t, Image< A > &a_, const type-`
`name A::value &v)`
- `template<typename T , typename A >`
`void propagate_node_to_ancestors (typename A::psite n, const T &t, Image< A > &a_)`

- `template<typename T, typename A >`
`void propagate_node_to_descendants (typename A::psite n, const T &t, Image< A > &a_, const`
`typename A::value &v, unsigned *nb_leaves=0)`
- `template<typename T, typename A >`
`void propagate_node_to_descendants (typename A::psite &n, const T &t, Image< A > &a_, un-`
`signed *nb_leaves=0)`
- `template<typename T, typename F >`
`void propagate_representative (const T &t, Image< F > &f_)`

Propagate the representative node's value to non-representative points of the component.

9.124.1 Detailed Description

Namespace of morphological tree-related routines.

9.124.2 Function Documentation

9.124.2.1 `template<typename A, typename T > mln::trait::ch_value< typename T::function,`
`typename A::result >::ret mln::morpho::tree::compute_attribute_image (const`
`Accumulator< A > & a, const T & t, mln::trait::ch_value< typename T::function, A`
`>::ret * accu_image = 0) [inline]`

Compute an attribute image using tree with a parent relationship between sites.

In the attribute image, the resulting value at a node is the 'sum' of its sub-components value + the attribute value at this node.

Warning: `s` translates the ordering related to the "natural" childhood relationship. The parenthood is thus inverted w.r.t. to `s`.

It is very convenient since all processing upon the parent tree are performed following `s` (in the default "forward" way).

FIXME: Put it more clearly...

The parent result image verifies:

- `p` is root iff `parent(p) == p`
- `p` is a node iff either `p` is root or `f(parent(p)) != f(p)`.

Parameters

[in] `a` Attribute.

[in] `t` Component tree.

[out] `accu_image` Optional argument used to store image of attribute accumulator.

Returns

The attribute image.

Definition at line 216 of file `compute_attribute_image.hh`.

Referenced by `compute_attribute_image_from()`.

9.124.2.2 `template<typename A , typename T , typename V > mln::trait::ch_value< typename T::function, typename A::result >::ret mln::morpho::tree::compute_attribute_image_ from (const Accumulator< A > & a, const T & t, const Image< V > & values, mln::trait::ch_value< typename T::function, A >::ret * accu_image = 0) [inline]`

The same as `compute_attribute_image` but uses the values stored by `values` image instead.

Parameters

- [in] *a* Attribute.
- [in] *t* Component tree.
- [in] *values* Value image.
- [out] *accu_image* Optional argument used to store image.

Returns

Definition at line 233 of file `compute_attribute_image.hh`.

References `compute_attribute_image()`.

9.124.2.3 `template<typename I , typename N , typename S > mln::trait::ch_value< I, typename I::psite >::ret mln::morpho::tree::compute_parent (const Image< I > & f, const Neighborhood< N > & nbh, const Site_Set< S > & s) [inline]`

Compute a tree with a parent relationship between sites.

Warning: *s* translates the ordering related to the "natural" childhood relationship. The parenthood is thus inverted w.r.t. to *s*.

It is very convenient since most processing routines upon the parent tree are performed following *s* (in the default "forward" way). Indeed that is the way to propagate information from parents to children.

The parent result image verifies:

- *p* is root iff `parent(p) == p`
- *p* is a node iff either *p* is root or `f(parent(p)) != f(p)`.

The choice "*s* means childhood" is consistent with labeling in binary images. In that particular case, while browsing the image in forward scan (video), we expect to find first a tree root (a first point, representative of a component) and then the other component points. Please note that it leads to increasing values of labels in the "natural" video scan.

Since mathematical morphology on functions is related to morphology on sets, we clearly want to keep the equivalence between "component labeling" and "component filtering" using trees.

FIXME: Put it more clearly... Insert pictures!

A binary image:

- | | - -
- | | - |
- - - - -

• - || -

where '|' means true and '-' means false.

Its labeling:

0 1 1 0 0

0 1 1 0 2

0 0 0 0 0

0 0 3 3 0

The corresponding forest:

x o . x x

x . . x o

x x x x x

x x o . x

where 'x' means "no data", 'o' is a tree root (representative point for a component), and '.' is a tree regular (non-root) point (in a component by not its representative point).

The forest, with the parent relationship looks like:

o < .

^ r

.. o

o < .

Definition at line 286 of file compute_parent.hh.

9.124.2.4 `template<typename I , typename N > morpho::tree::data< I, p_array< typename I::psite >> mln::morpho::tree::dual_input_max_tree (const Image< I > & f, const Image< I > & m, const Neighborhood< N > & nbh) [inline]`

Compute the dual input max tree using mask-based connectivity.

Parameters

[in] *f* The original image.

[in] *m* The connectivity mask.

[in] *nbh* The neighborhood of the mask.

Returns

The computed tree.

Definition at line 109 of file dual_input_tree.hh.

9.124.2.5 `template<typename I , typename N > data< I, p_array< typename I::psite >> mln::morpho::tree::max_tree (const Image< I > & f, const Neighborhood< N > & nbh) [inline]`

Compute a canonized max-tree.

Parameters

[in] *f* The input image.

[in] *nbh* The neighborhood.

Returns

The corresponding max-tree structure.

Definition at line 100 of file component_tree.hh.

References mln::data::sort_psites_increasing().

9.124.2.6 `template<typename I, typename N > data< I, p_array< typename I::psite > >
mln::morpho::tree::min_tree (const Image< I > & f, const Neighborhood< N > &
nbh) [inline]`

Compute a canonized min-tree.

Parameters

[in] *f* The input image.

[in] *nbh* The neighborhood.

Returns

The corresponding min-tree structure.

Definition at line 77 of file component_tree.hh.

References mln::data::sort_psites_decreasing().

9.124.2.7 `template<typename T, typename A, typename P, typename W > void
mln::morpho::tree::propagate_if (const T & tree, Image< A > & a_, const
way_of_propagation< W > & prop_, const Function_v2b< P > & pred_, const
typename A::value & v) [inline]`

Propagate nodes checking the predicate *pred* in the way defined by *way_of_propagation*.

Parameters

tree Component tree used for propagation.

a_ Attributed image where values are propagated.

prop_ Propagate node in ascendant or descendant way.

pred_ Predicate that node must check to be propagated.

v [Value](#) to be propagated. (By default *v* is the value at the node being propagated).

Definition at line 293 of file propagate_if.hh.

Referenced by propagate_if(), propagate_if_value(), and mln::morpho::tree::filter::subtractive().

9.124.2.8 `template<typename T , typename A , typename P > void
mln::morpho::tree::propagate_if (const T & tree, Image< A > & a_, const
desc_propagation & prop_, const Function_v2b< P > & pred_) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 310 of file `propagate_if.hh`.

References `propagate_if()`.

9.124.2.9 `template<typename T , typename A , typename W > void
mln::morpho::tree::propagate_if_value (const T & tree, Image< A > & a_, const
way_of_propagation< W > & prop, const typename A::value & v) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 278 of file `propagate_if.hh`.

References `propagate_if()`.

9.124.2.10 `template<typename T , typename A , typename W > void
mln::morpho::tree::propagate_if_value (const T & tree, Image< A > & a_, const
way_of_propagation< W > & prop_, const typename A::value & v, const typename
A::value & v_prop) [inline]`

Propagate nodes having the value `v` in the way defined by `way_of_propagation`.

Parameters

tree Component tree used for propagation.

a_ Attributed image where values are propagated.

prop_ Propagate node in ascendant or descendant way.

v Value that node must have to be propagated.

v_prop Value to propagate (By default it is the value at the node being propagated).

Definition at line 262 of file `propagate_if.hh`.

References `propagate_if()`.

9.124.2.11 `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_ -
ancestors (typename A::psite n, const T & t, Image< A > & a_, const typename
A::value & v)`

Propagate a value `v` from a node `n` to its ancestors.

Parameters

[in] *n* Node to propagate.

[in] *t* Component tree used for propagation.

[in] *a_* Attribute image where values are propagated.

[in] *v* Value to propagate.

Definition at line 170 of file propagate_node.hh.

Referenced by propagate_node_to_ancestors().

9.124.2.12 `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_ancestors (typename A::psite n, const T & t, Image< A > & a_) [inline]`

Propagate the node's value to its ancestors.

Parameters

[in] *n* Node to propagate.

[in] *t* Component tree used for propagation.

[in, out] *a_* Attribute image where values are propagated.

Definition at line 197 of file propagate_node.hh.

References propagate_node_to_ancestors().

9.124.2.13 `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_descendants (typename A::psite n, const T & t, Image< A > & a_, const typename A::value & v, unsigned * nb_leaves = 0) [inline]`

Propagate a value *v* from a node *n* to its descendants.

Parameters

[in] *n* Node to propagate.

[in] *t* Component tree used for propagation.

[in] *a_* Attribute image where values are propagated.

[in] *v* Value to propagate.

[out] *nb_leaves* Optional. Store the number of leaves in the component.

Definition at line 120 of file propagate_node.hh.

9.124.2.14 `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_descendants (typename A::psite & n, const T & t, Image< A > & a_, unsigned * nb_leaves = 0) [inline]`

Propagate the node's value to its descendants.

Parameters

[in] *n* Node to propagate.

[in] *t* Component tree used for propagation.

[in] *a_* Attribute image where values are propagated.

[out] *nb_leaves* Optional. Store the number of leaves in the component.

9.124.2.15 `template<typename T , typename F > void mln::morpho::tree::propagate_` `representative (const T & t, Image< F > & f_) [inline]`

Propagate the representative node's value to non-representative points of the component.

Parameters

t Component tree.

f_ Value image.

Definition at line 65 of file `propagate_representative.hh`.

9.125 `mln::morpho::tree::filter` Namespace Reference

Namespace for attribute filtering.

Functions

- `template<typename T , typename F , typename P >`
`void direct (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_)`
Direct non-pruning strategy.
- `template<typename T , typename F , typename P >`
`void filter (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_, const typename F::value &v)`
*Filter the image *f_* with a given value.*
- `template<typename T , typename F , typename P >`
`void max (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_)`
Max pruning strategy.
- `template<typename T , typename F , typename P >`
`void min (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_)`
Min pruning strategy.
- `template<typename T , typename F , typename P >`
`void subtractive (const T &tree, Image< F > &f_, const Function_v2b< P > &pred_)`
Subtractive pruning strategy.

9.125.1 Detailed Description

Namespace for attribute filtering.

9.125.2 Function Documentation

9.125.2.1 `template<typename T , typename F , typename P > void
mln::morpho::tree::filter::direct (const T & tree, Image< F > & f_, const
Function_v2b< P > & pred_) [inline]`

Direct non-pruning strategy.

A node is removed if it does not verify the predicate. The sub-components remain intact.

Parameters

- [in] *tree* Component tree.
- [out] *f_* [Image](#) to filter.
- [in] *pred_* Filtering criterion.

Definition at line 73 of file direct.hh.

9.125.2.2 `template<typename T , typename F , typename P > void mln::morpho::tree::filter::filter
(const T & tree, Image< F > & f_, const Function_v2b< P > & pred_, const
typename F::value & v) [inline]`

Filter the image *f_* with a given value.

The sub-components of nodes that does not match the predicate *pred_* are filled with the given value *v*.

Parameters

- tree* Component tree.
- f_* [Image](#) function.
- pred_* Predicate.
- v* [Value](#) to propagate.

Definition at line 73 of file morpho/tree/filter/filter.hh.

References `mln::data::fill()`, and `mln::initialize()`.

9.125.2.3 `template<typename T , typename F , typename P > void mln::morpho::tree::filter::max
(const T & tree, Image< F > & f_, const Function_v2b< P > & pred_) [inline]`

Max pruning strategy.

A node is removed iif all of its children are removed or if it does not verify the predicate *pred_*.

Parameters

- [in] *tree* Component tree.
- [out] *f_* [Image](#) to filter.
- [in] *pred_* Filtering criterion.

Definition at line 74 of file morpho/tree/filter/max.hh.

References `mln::data::fill()`, and `mln::initialize()`.

9.125.2.4 `template<typename T , typename F , typename P > void mln::morpho::tree::filter::min (const T & tree, Image< F > & f_, const Function_v2b< P > & pred_) [inline]`

Min pruning strategy.

A node is removed iif its parent is removed or if it does not verify the predicate `pred_`.

Parameters

- [in] *tree* Component tree.
- [out] *f_* [Image](#) to filter.
- [in] *pred_* Filtering criterion.

Definition at line 75 of file `morpho/tree/filter/min.hh`.

References `mln::data::fill()`, and `mln::initialize()`.

9.125.2.5 `template<typename T , typename F , typename P > void mln::morpho::tree::filter::subtractive (const T & tree, Image< F > & f_, const Function_v2b< P > & pred_) [inline]`

Subtractive pruning strategy.

The node is removed if it does not verify the predicate. The sub-components values are set to the value of the removed component.

Parameters

- [in] *tree* Component tree.
- [out] *f_* [Image](#) to filter.
- [in] *pred_* Filtering criterion.

Definition at line 77 of file `subtractive.hh`.

References `mln::morpho::tree::propagate_if()`.

9.126 mln::morpho::watershed Namespace Reference

Namespace of morphological watershed routines.

Namespaces

- namespace [watershed](#)
Namespace of morphological watershed routines implementations.

Functions

- `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret flooding (const Image< I > &input, const Neighborhood< N > &nbh, L &n_basins)`

Meyer's Watershed Transform (WST) algorithm.

- `template<typename L , typename I , typename N >`
`mln::trait::ch_value< I, L >::ret flooding (const Image< I > &input, const Neighborhood< N >`
`&nbh)`

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

- `template<typename I , typename J >`
`mln::trait::ch_value< I, value::rgb8 >::ret superpose (const Image< I > &input, const Image< J >`
`&ws_ima)`

Convert an image to a rgb8 image and draw the watershed lines.

- `template<typename I , typename J >`
`mln::trait::ch_value< I, value::rgb8 >::ret superpose (const Image< I > &input_, const Image< J`
`> &ws_ima_, const value::rgb8 &wsl_color)`

Convert an image to a rgb8 image and draw the watershed lines.

- `template<class T >`
`T::image_t topological (T &tree)`

Compute a toological watershed transform from tree.

9.126.1 Detailed Description

Namespace of morphological watershed routines.

9.126.2 Function Documentation

- 9.126.2.1** `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret`
`mln::morpho::watershed::flooding (const Image< I > & input, const Neighborhood<`
`N > & nbh, L & n_basins) [inline]`

Meyer's Watershed Transform (WST) algorithm.

Parameters

[in] *input* The input image.

[in] *nbh* The connexity of markers.

[out] *n_basins* The number of basins.

- `L` is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- `I` is the exact type of the input image.
- `N` is the exact type of the neighborhood used to express *input's* connexity.

Definition at line 381 of file flooding.hh.

9.126.2.2 `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret mln::morpho::watershed::flooding (const Image< I > & input, const Neighborhood< N > & nbh)`

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

Parameters

[in] *input* The input image.

[in] *nbh* The connexity of markers.

- *L* is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- *I* is the exact type of the input image.
- *N* is the exact type of the neighborhood used to express *input*'s connexity.

Note that the first parameter, *L*, is not automatically valued from the type of the actual argument during implicit instantiation: you have to explicitly pass this parameter at call sites.

Definition at line 396 of file `flooding.hh`.

9.126.2.3 `template<typename I , typename J > mln::trait::ch_value< I, value::rgb8 >::ret mln::morpho::watershed::superpose (const Image< I > & input, const Image< J > & ws_ima) [inline]`

Convert an image to a rgb8 image and draw the watershed lines.

Definition at line 109 of file `morpho/watershed/superpose.hh`.

References `mln::literal::red`, and `superpose()`.

9.126.2.4 `template<typename I , typename J > mln::trait::ch_value< I, value::rgb8 >::ret mln::morpho::watershed::superpose (const Image< I > & input_, const Image< J > & ws_ima_, const value::rgb8 & wsl_color) [inline]`

Convert an image to a rgb8 image and draw the watershed lines.

Definition at line 85 of file `morpho/watershed/superpose.hh`.

References `mln::data::convert()`, `mln::data::fill()`, and `mln::literal::zero`.

Referenced by `superpose()`.

9.126.2.5 `template<class T > T::image_t mln::morpho::watershed::topological (T & tree)`

Compute a toological watershed transform from *tree*.

Definition at line 675 of file `topological.hh`.

References `mln::data::fill()`, `mln::p_priority< P, Q >::front()`, `mln::initialize()`, `mln::p_priority< P, Q >::pop()`, `mln::p_priority< P, Q >::push()`, and `topological()`.

Referenced by `topological()`.

9.127 mln::morpho::watershed::watershed Namespace Reference

Namespace of morphological watershed routines implementations.

Namespaces

- namespace [generic](#)

Namespace of morphological watershed routines generic implementations.

9.127.1 Detailed Description

Namespace of morphological watershed routines implementations.

9.128 mln::morpho::watershed::watershed::generic Namespace Reference

Namespace of morphological watershed routines generic implementations.

9.128.1 Detailed Description

Namespace of morphological watershed routines generic implementations.

9.129 mln::norm Namespace Reference

Namespace of norms.

Namespaces

- namespace [impl](#)

Implementation namespace of norm namespace.

Functions

- `template<unsigned n, typename C >`
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum l2 (const C(&vec)[n])`
L2-norm of a vector vec.
- `template<unsigned n, typename C >`
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum l1 (const C(&vec)[n])`
L1-norm of a vector vec.

- `template<unsigned n, typename C >`
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum l1_distance` (const C(&vec1)[n], const C(&vec2)[n])
L1-norm distance between vectors vec1 and vec2.
- `template<unsigned n, typename C >`
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum sqr_l2` (const C(&vec)[n])
Squared L2-norm of a vector vec.
- `template<unsigned n, typename C >`
`mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum l2_distance` (const C(&vec1)[n], const C(&vec2)[n])
L2-norm distance between vectors vec1 and vec2.
- `template<unsigned n, typename C >`
`C linfy` (const C(&vec)[n])
L-infinity-norm of a vector vec.
- `template<unsigned n, typename C >`
`C linfy_distance` (const C(&vec1)[n], const C(&vec2)[n])
L-infinity-norm distance between vectors vec1 and vec2.

9.129.1 Detailed Description

Namespace of norms.

9.129.2 Function Documentation

9.129.2.1 `template<unsigned n, typename C > mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum mln::norm::l1 (const C(&) vec[n])`
[inline]

L1-norm of a vector *vec*.

Definition at line 108 of file l1.hh.

9.129.2.2 `template<unsigned n, typename C > mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum mln::norm::l1_distance (const C(&) vec1[n], const C(&) vec2[n])` **[inline]**

L1-norm distance between vectors *vec1* and *vec2*.

Definition at line 124 of file l1.hh.

9.129.2.3 `template<unsigned n, typename C > mln::trait::value_< typename
mln::trait::op::times< C, C >::ret >::sum mln::norm::l2 (const C(&) vec[n])
[inline]`

L2-norm of a vector *vec*.

Definition at line 139 of file l2.hh.

9.129.2.4 `template<unsigned n, typename C > mln::trait::value_< typename
mln::trait::op::times< C, C >::ret >::sum mln::norm::l2_distance (const C(&)
vec1[n], const C(&) vec2[n]) [inline]`

L2-norm distance between vectors *vec1* and *vec2*.

Definition at line 173 of file l2.hh.

9.129.2.5 `template<unsigned n, typename C > C mln::norm::linfty (const C(&) vec[n])
[inline]`

L-infinity-norm of a vector *vec*.

Definition at line 113 of file linfty.hh.

9.129.2.6 `template<unsigned n, typename C > C mln::norm::linfty_distance (const C(&)
vec1[n], const C(&) vec2[n]) [inline]`

L-infinity-norm distance between vectors *vec1* and *vec2*.

Definition at line 127 of file linfty.hh.

9.129.2.7 `template<unsigned n, typename C > mln::trait::value_< typename
mln::trait::op::times< C, C >::ret >::sum mln::norm::sqr_l2 (const C(&) vec[n])
[inline]`

Squared L2-norm of a vector *vec*.

Definition at line 156 of file l2.hh.

Referenced by `mln::geom::mesh_corner_point_area()`, and `mln::geom::mesh_normal()`.

9.130 `mln::norm::impl` Namespace Reference

Implementation namespace of norm namespace.

9.130.1 Detailed Description

Implementation namespace of norm namespace.

9.131 mln::opt Namespace Reference

Namespace of optional routines.

Namespaces

- namespace [impl](#)
Implementation namespace of opt namespace.

Functions

- `template<typename I >`
`I::rvalue at (const Image< I > &ima, def::coord ind)`
One dimension Read-only access to the ima value located at (ind).
- `template<typename I >`
`I::lvalue at (Image< I > &ima, def::coord ind)`
Read-write access to the ima value located at (ind).
- `template<typename I >`
`I::lvalue at (Image< I > &ima, def::coord row, def::coord col)`
Read-write access to the ima value located at (row, col).
- `template<typename I >`
`I::rvalue at (const Image< I > &ima, def::coord sli, def::coord row, def::coord col)`
Three dimensions Read-only access to the ima value located at (sli, row, col).
- `template<typename I >`
`I::rvalue at (const Image< I > &ima, def::coord row, def::coord col)`
Two dimensions Read-only access to the ima value located at (row, col).
- `template<typename I >`
`I::lvalue at (Image< I > &ima, def::coord sli, def::coord row, def::coord col)`
Read-write access to the ima value located at (sli, row, col).

9.131.1 Detailed Description

Namespace of optional routines.

9.131.2 Function Documentation

9.131.2.1 `template<typename I > I::rvalue mln::opt::at (const Image< I > & ima, def::coord ind) [inline]`

One dimension Read-only access to the ima value located at (ind).

Definition at line 151 of file at.hh.

Referenced by `mln::transform::hough()`, and `mln::make::image()`.

9.131.2.2 `template<typename I > I::lvalue mln::opt::at (Image< I > & ima, def::coord ind)`

Read-write access to the `ima` value located at (`ind`).

Definition at line 160 of file `at.hh`.

9.131.2.3 `template<typename I > I::lvalue mln::opt::at (Image< I > & ima, def::coord row, def::coord col)`

Read-write access to the `ima` value located at (`row`, `col`).

Definition at line 245 of file `at.hh`.

9.131.2.4 `template<typename I > I::rvalue mln::opt::at (const Image< I > & ima, def::coord sli, def::coord row, def::coord col) [inline]`

Three dimensions Read-only access to the `ima` value located at (`sli`, `row`, `col`).

Definition at line 320 of file `at.hh`.

9.131.2.5 `template<typename I > I::rvalue mln::opt::at (const Image< I > & ima, def::coord row, def::coord col) [inline]`

Two dimensions Read-only access to the `ima` value located at (`row`, `col`).

Definition at line 236 of file `at.hh`.

9.131.2.6 `template<typename I > I::lvalue mln::opt::at (Image< I > & ima, def::coord sli, def::coord row, def::coord col)`

Read-write access to the `ima` value located at (`sli`, `row`, `col`).

Definition at line 330 of file `at.hh`.

9.132 `mln::opt::impl` Namespace Reference

Implementation namespace of `opt` namespace.

9.132.1 Detailed Description

Implementation namespace of `opt` namespace. Three dimensions.

Two dimensions.

One dimension.

9.133 `mln::pw` Namespace Reference

Namespace of "point-wise" expression tools.

Classes

- class [image](#)
A generic point-wise image implementation.

9.133.1 Detailed Description

Namespace of "point-wise" expression tools.

9.134 mln::registration Namespace Reference

Namespace of "point-wise" expression tools.

Classes

- class [closest_point_basic](#)
Closest point functor based on map distance.
- class [closest_point_with_map](#)
Closest point functor based on map distance.

Functions

- `template<typename P, typename F >`
`algebra::quat get_rot (const p_array< P > &P_, const vec3d_f &mu_P, const vec3d_f &mu_Yk,`
`const F &closest_point, const algebra::quat &qR, const vec3d_f &qT)`
FIXME: work only for 3d images.
- `template<typename P, typename F >`
`std::pair< algebra::quat, mln_vec(P)> icp (const p_array< P > &P_, const p_array< P > &X, const`
`F &closest_point, const algebra::quat &initial_rot, const mln_vec(P)&initial_translation)`
Base version of the ICP algorithm. It is called in other variants.
- `template<typename P, typename F >`
`composed< translation< P::dim, float >, rotation< P::dim, float > > icp (const p_array< P > &P_,`
`const p_array< P > &X, const F &closest_point)`
- `template<typename P >`
`composed< translation< P::dim, float >, rotation< P::dim, float > > registration1 (const box< P`
`> &domain, const p_array< P > &P_, const p_array< P > &X)`
Call ICP once and return the resulting transformation.
- `template<typename P >`
`composed< translation< P::dim, float >, rotation< P::dim, float > > registration2 (const box< P`
`> &domain, const p_array< P > &P_, const p_array< P > &X)`
Call ICP 10 times.

- `template<typename P >`
`composed< translation< P::dim, float >, rotation< P::dim, float > >` `registration3` (const `box< P`
`>` &domain, const `p_array< P >` &P_, const `p_array< P >` &X)

Call ICP 10 times.

9.134.1 Detailed Description

Namespace of "point-wise" expression tools.

9.134.2 Function Documentation

- 9.134.2.1** `template<typename P , typename F > algebra::quat mln::registration::get_rot (const`
`p_array< P > & P_, const vec3d_f & mu_P, const vec3d_f & mu_Yk, const F &`
`closest_point, const algebra::quat & qR, const vec3d_f & qT)`

FIXME: work only for 3d images.

Definition at line 527 of file icp.hh.

References `mln::p_array< P >::nsites()`.

- 9.134.2.2** `template<typename P , typename F > std::pair< algebra::quat, mln_vec(P)>`
`mln::registration::icp (const p_array< P > & P_, const p_array< P > & X, const F &`
`closest_point, const algebra::quat & initial_rot, const mln_vec(P)& initial_translation)`
[inline]

Base version of the ICP algorithm. It is called in other variants.

Register point in `c` using a function of closest points `closest_point`. This overload allows to specify initial transformations.

Parameters

- [in] `P_` The cloud of points.
- [in] `X` the reference surface.
- [in] `closest_point` The function of closest points.
- [in] `initial_rot` An initial rotation.
- [in] `initial_translation` An initial translation.

Returns

the rigid transformation which may be use later to create a registered image.

WARNING: the function `closest_point` *MUST* take float/double vector as arguments. Otherwise the resulting transformation may be wrong due to the truncation of the vector coordinate values.

Precondition

`P_` and `X` must not be empty.

Reference article: "A Method for Registration of 3-D Shapes", Paul J. Besl and Neil D. McKay, IEEE, 2, February 1992.

Definition at line 612 of file icp.hh.

References `mln::geom::bbox()`, `mln::literal::black`, `mln::set::compute()`, `mln::duplicate()`, `mln::box< P >::enlarge()`, `mln::data::fill()`, `mln::literal::green`, `mln::io::ppm::save()`, and `mln::literal::white`.

9.134.2.3 `template<typename P , typename F > composed< translation<P::dim,float>,rotation<P::dim,float> > mln::registration::icp (const p_array< P > & P_, const p_array< P > & X, const F & closest_point)`

Register point in `c` using a function of closest points `closest_point`.

Parameters

- [in] `P_` The cloud of points.
- [in] `X` the reference surface.
- [in] `closest_point` The function of closest points.

Returns

the rigid transformation which may be use later to create a registered image.

9.134.2.4 `template<typename P > composed< translation< P::dim, float >, rotation< P::dim, float > > mln::registration::registration1 (const box< P > & domain, const p_array< P > & P_, const p_array< P > & X) [inline]`

Call ICP once and return the resulting transformation.

Definition at line 325 of file registration.hh.

9.134.2.5 `template<typename P > composed< translation< P::dim, float >, rotation< P::dim, float > > mln::registration::registration2 (const box< P > & domain, const p_array< P > & P_, const p_array< P > & X) [inline]`

Call ICP 10 times.

Do the first call to ICP with all sites then work on a subset of which size is decreasing. For each call, a distance criterion is computed on a subset. Sites part of the subset which are too far or too close are removed. Removed sites are *NOT* reused later in the subset.

Definition at line 345 of file registration.hh.

9.134.2.6 `template<typename P > composed< translation< P::dim, float >, rotation< P::dim, float > > mln::registration::registration3 (const box< P > & domain, const p_array< P > & P_, const p_array< P > & X) [inline]`

Call ICP 10 times.

Do the first call to ICP with all sites then work on a subset. For each call, a distance criterion is computed on a subset. A new subset is computed from the whole set of points according to this distance. It will be used in the next call. Removed Sites *MAY* be reintegrated.

Definition at line 365 of file registration.hh.

9.135 mln::select Namespace Reference

Select namespace (FIXME doc).

Classes

- struct [p_of](#)
Structure [p_of](#).

9.135.1 Detailed Description

Select namespace (FIXME doc).

9.136 mln::set Namespace Reference

Namespace of image processing routines related to pixel sets.

Functions

- template<typename S >
unsigned [card](#) (const [Site_Set](#)< S > &s)
Compute the cardinality of the site set s.
- template<typename A , typename S >
A::result [compute](#) (const [Accumulator](#)< A > &a, const [Site_Set](#)< S > &s)
Compute an accumulator onto a site set.
- template<typename A , typename I >
A::result [compute_with_weights](#) (const [Accumulator](#)< A > &a, const [Image](#)< I > &w)
Compute an accumulator on a site set described by an image.
- template<typename A , typename I , typename L >
[util::array](#)< typename A::result > [compute_with_weights](#) (const [Accumulator](#)< A > &a, const [Image](#)< I > &w, const [Image](#)< L > &label, const typename L::value &nlabels)
Compute an accumulator on every labeled sub-site-sets.
- template<typename S >
S::site [get](#) (const [Site_Set](#)< S > &s, size_t index)
FIXME.
- template<typename S >
bool [has](#) (const [Site_Set](#)< S > &s, const typename S::site &e)
FIXME.
- template<typename A , typename S >
[mln_meta_accu_result](#) (A, typename S::site) [compute](#)(const [Meta_Accumulator](#)< A > &a)
Compute an accumulator onto a site set.

- `template<typename A , typename I >`
`mln_meta_accu_result` (A, typename I::site) `compute_with_weights(const Meta_Accumulator< A`
`> &a`
Compute an accumulator on a site set described by an image.

9.136.1 Detailed Description

Namespace of image processing routines related to pixel sets.

9.136.2 Function Documentation

9.136.2.1 `template<typename S > unsigned mln::set::card (const Site_Set< S > & s)` `[inline]`

Compute the cardinality of the site set *s*.

Definition at line 134 of file `set/card.hh`.

9.136.2.2 `template<typename A , typename S > A::result mln::set::compute (const` `Accumulator< A > & a, const Site_Set< S > & s) [inline]`

Compute an accumulator onto a site set.

Parameters

`[in]` *a* An accumulator.

`[in]` *s* A site set.

Returns

The accumulator result.

Definition at line 112 of file `set/compute.hh`.

Referenced by `mln::registration::icp()`.

9.136.2.3 `template<typename A , typename I > A::result mln::set::compute_with_weights (const` `Accumulator< A > & a, const Image< I > & w) [inline]`

Compute an accumulator on a site set described by an image.

Parameters

`[in]` *a* An accumulator.

`[in]` *w* An image of weights (a site -> a weight).

Returns

The accumulator result.

Definition at line 217 of file `compute_with_weights.hh`.

Referenced by `compute_with_weights()`.

9.136.2.4 `template<typename A , typename I , typename L > util::array< typename A::result > mln::set::compute_with_weights (const Accumulator< A > & a, const Image< I > & w, const Image< L > & label, const typename L::value & nlabels)`

Compute an accumulator on every labeled sub-site-sets.

Parameters

- [in] *a* An accumulator.
- [in] *w* An image of weights (a site -> a weight).
- [in] *label* A label image.
- [in] *nlabels* The number of labels in *label*.

Returns

An array of accumulator result. One per label.

Definition at line 234 of file `compute_with_weights.hh`.

References `compute_with_weights()`.

9.136.2.5 `template<typename S > S::site mln::set::get (const Site_Set< S > & s, size_t index)`

FIXME.

Definition at line 56 of file `set/get.hh`.

9.136.2.6 `template<typename S > bool mln::set::has (const Site_Set< S > & s, const typename S::site & e)`

FIXME.

Definition at line 56 of file `set/has.hh`.

9.136.2.7 `template<typename A , typename S > mln::set::mln_meta_accu_result (A , typename S::site) const`

Compute an accumulator onto a site set.

Parameters

- [in] *a* A meta-accumulator.
- [in] *s* A site set.

9.136.2.8 `template<typename A , typename I > mln::set::mln_meta_accu_result (A , typename I::site) const [inline]`

Compute an accumulator on a site set described by an image.

Parameters

- [in] *a* A meta-accumulator.

[in] *w* An image of weights (a site -> a weight).

Returns

The accumulator result.

9.137 mln::subsampling Namespace Reference

Namespace of "point-wise" expression tools.

Functions

- `template<typename I > mln::trait::concrete< I >::ret antialiased (const Image< I > &input, unsigned factor, const typename I::domain_t &output_domain, unsigned border_thickness)`
Antialiased subsampling.
- `template<typename I > mln::trait::concrete< I >::ret antialiased (const Image< I > &input, unsigned factor)`
- `template<typename I > mln::trait::concrete< I >::ret gaussian_subsampling (const Image< I > &input, float sigma, const typename I::dpsite &first_p, const typename I::site::coord &gap)`
Gaussian subsampling FIXME : doxy.
- `template<typename I > mln::trait::concrete< I >::ret subsampling (const Image< I > &input, const typename I::site::delta &first_p, const typename I::site::coord &gap)`
Subsampling FIXME : doxy.

9.137.1 Detailed Description

Namespace of "point-wise" expression tools.

9.137.2 Function Documentation

9.137.2.1 `template<typename I > mln::trait::concrete< I >::ret mln::subsampling::antialiased (const Image< I > & input, unsigned factor, const typename I::domain_t & output_domain, unsigned border_thickness) [inline]`

Antialiased subsampling.

Parameters

- [in] *input* A gray-level image.
- [in] *factor* Subsampling ratio. Must be divisible by 2 or 3.
- [in] *output_domain* Force output domain.
- [in] *border_thickness* Force output border thickness.

Definition at line 418 of file antialiased.hh.

Referenced by antialiased().

9.137.2.2 `template<typename I> mln::trait::concrete< I >::ret mln::subsampling::antialiased (const Image< I > & input, unsigned factor) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 443 of file antialiased.hh.

References antialiased().

9.137.2.3 `template<typename I> mln::trait::concrete< I >::ret mln::subsampling::gaussian_subsampling (const Image< I > & input, float sigma, const typename I::dpsite & first_p, const typename I::site::coord & gap) [inline]`

Gaussian subsampling FIXME : doxy.

Definition at line 62 of file gaussian_subsampling.hh.

References mln::linear::gaussian(), mln::geom::ncols(), and mln::geom::nrows().

9.137.2.4 `template<typename I> mln::trait::concrete< I >::ret mln::subsampling::subsampling (const Image< I > & input, const typename I::site::delta & first_p, const typename I::site::coord & gap) [inline]`

Subsampling FIXME : doxy.

Definition at line 91 of file subsampling.hh.

References mln::geom::ncols(), and mln::geom::nrows().

9.138 mln::tag Namespace Reference

Namespace of image processing routines related to tags.

9.138.1 Detailed Description

Namespace of image processing routines related to tags.

9.139 mln::test Namespace Reference

Namespace of image processing routines related to pixel tests.

Namespaces

- namespace [impl](#)

Implementation namespace of test namespace.

Functions

- `template<typename I >`
`bool positive (const Image< I > &input)`
Test if an image only contains positive values.
- `template<typename S , typename F >`
`bool predicate (const Site_Set< S > &pset, const Function_v2b< F > &f)`
Test if all points of `pset` verify the predicate `f`.
- `template<typename I , typename J , typename F >`
`bool predicate (const Image< I > &lhs, const Image< J > &rhs, const Function_vv2b< F > &f)`
Test if all pixel values of `lhs` and `rhs` verify the predicate `f`.
- `template<typename I , typename F >`
`bool predicate (const Image< I > &ima, const Function_v2b< F > &f)`
Test if all pixel values of `ima` verify the predicate `f`.

9.139.1 Detailed Description

Namespace of image processing routines related to pixel tests.

9.139.2 Function Documentation

9.139.2.1 `template<typename I > bool mln::test::positive (const Image< I > & input)` `[inline]`

Test if an image only contains positive values.

Definition at line 54 of file `positive.hh`.

References `predicate()`, and `mln::literal::zero`.

Referenced by `mln::morpho::gradient()`, `mln::morpho::gradient_external()`, `mln::morpho::gradient_internal()`, `mln::morpho::top_hat_black()`, `mln::morpho::elementary::top_hat_black()`, `mln::morpho::top_hat_self_complementary()`, `mln::morpho::elementary::top_hat_self_complementary()`, `mln::morpho::top_hat_white()`, and `mln::morpho::elementary::top_hat_white()`.

9.139.2.2 `template<typename S , typename F > bool mln::test::predicate (const Site_Set< S > & pset, const Function_v2b< F > & f)` `[inline]`

Test if all points of `pset` verify the predicate `f`.

Parameters

[in] *pset* The point set.

[in] *f* The predicate.

Definition at line 242 of file `predicate.hh`.

9.139.2.3 `template<typename I , typename J , typename F > bool mln::test::predicate (const Image< I > & lhs, const Image< J > & rhs, const Function_vv2b< F > & f) [inline]`

Test if all pixel values of `lhs` and `rhs` verify the predicate `f`.

Parameters

- [in] `lhs` The image.
- [in] `rhs` The image.
- [in] `f` The predicate.

Definition at line 222 of file `predicate.hh`.

9.139.2.4 `template<typename I , typename F > bool mln::test::predicate (const Image< I > & ima, const Function_v2b< F > & f) [inline]`

Test if all pixel values of `ima` verify the predicate `f`.

Parameters

- [in] `ima` The image.
- [in] `f` The predicate.

Definition at line 207 of file `predicate.hh`.

Referenced by `mln::operator<()`, `mln::operator<=()`, `mln::operator==()`, and `positive()`.

9.140 mln::test::impl Namespace Reference

Implementation namespace of test namespace.

9.140.1 Detailed Description

Implementation namespace of test namespace.

9.141 mln::topo Namespace Reference

Namespace of "point-wise" expression tools.

Classes

- class [adj_higher_dim_connected_n_face_bkd_iter](#)
Backward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.
- class [adj_higher_dim_connected_n_face_fwd_iter](#)

Forward iterator on all the n -faces sharing an adjacent $(n+1)$ -face with a (reference) n -face of an `mln::complex<D>`.

- class [adj_higher_face_bkd_iter](#)
Backward iterator on all the adjacent $(n+1)$ -faces of the n -face of an `mln::complex<D>`.
- class [adj_higher_face_fwd_iter](#)
Forward iterator on all the adjacent $(n+1)$ -faces of the n -face of an `mln::complex<D>`.
- class [adj_lower_dim_connected_n_face_bkd_iter](#)
Backward iterator on all the n -faces sharing an adjacent $(n-1)$ -face with a (reference) n -face of an `mln::complex<D>`.
- class [adj_lower_dim_connected_n_face_fwd_iter](#)
Forward iterator on all the n -faces sharing an adjacent $(n-1)$ -face with a (reference) n -face of an `mln::complex<D>`.
- class [adj_lower_face_bkd_iter](#)
Backward iterator on all the adjacent $(n-1)$ -faces of the n -face of an `mln::complex<D>`.
- class [adj_lower_face_fwd_iter](#)
Forward iterator on all the adjacent $(n-1)$ -faces of the n -face of an `mln::complex<D>`.
- class [adj_lower_higher_face_bkd_iter](#)
Forward iterator on all the adjacent $(n-1)$ -faces and $(n+1)$ -faces of the n -face of an `mln::complex<D>`.
- class [adj_lower_higher_face_fwd_iter](#)
Forward iterator on all the adjacent $(n-1)$ -faces and $(n+1)$ -faces of the n -face of an `mln::complex<D>`.
- class [adj_m_face_bkd_iter](#)
Backward iterator on all the m -faces transitively adjacent to a (reference) n -face in a complex.
- class [adj_m_face_fwd_iter](#)
Forward iterator on all the m -faces transitively adjacent to a (reference) n -face in a complex.
- class [algebraic_face](#)
Algebraic face handle in a complex; the face dimension is dynamic.
- class [algebraic_n_face](#)
Algebraic N -face handle in a complex.
- class [center_only_iter](#)
Iterator on all the adjacent $(n-1)$ -faces of the n -face of an `mln::complex<D>`.
- class [centered_bkd_iter_adapter](#)
Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.
- class [centered_fwd_iter_adapter](#)
Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

- class [complex](#)
General complex of dimension D.
- class [face](#)
Face handle in a complex; the face dimension is dynamic.
- class [face_bkd_iter](#)
Backward iterator on all the faces of an mln::complex<D>.
- class [face_fwd_iter](#)
Forward iterator on all the faces of an mln::complex<D>.
- struct [is_n_face](#)
A functor testing wheter a mln::complex_psite is an N-face.
- struct [is_simple_2d_t](#)
Test if a point is simple or not.
- class [is_simple_cell](#)
A predicate for the simplicity of a point based on the collapse property of the attachment.
- class [n_face](#)
N-face handle in a complex.
- class [n_face_bkd_iter](#)
Backward iterator on all the faces of an mln::complex<D>.
- class [n_face_fwd_iter](#)
Forward iterator on all the faces of an mln::complex<D>.
- class [n_faces_set](#)
Set of face handles of dimension N.
- class [static_n_face_bkd_iter](#)
Backward iterator on all the N-faces of a mln::complex<D>.
- class [static_n_face_fwd_iter](#)
Forward iterator on all the N-faces of a mln::complex<D>.

Functions

- `template<unsigned D, typename G >
void detach (const complex_psite< D, G > &f, complex_image< D, G, bool > &ima)
Detach the cell corresponding to f from ima.`
- `template<unsigned D, typename G >
bool is_facet (const complex_psite< D, G > &f)
Is f a facet, i.e., a face not "included in" (adjacent to) a face of higher dimension?`

- `template<unsigned D>`
`algebraic_face< D > make_algebraic_face (const face< D > &f, bool sign)`
Create an algebraic face handle of a `D-complex`.
- `template<unsigned N, unsigned D>`
`algebraic_n_face< N, D > make_algebraic_n_face (const n_face< N, D > &f, bool sign)`
Create an algebraic `N-face` handle of a `D-complex`.
- `template<unsigned N, unsigned D>`
`std::ostream & operator<< (std::ostream &ostr, const algebraic_n_face< N, D > &f)`
Print an `mln::topo::algebraic_n_face`.
- `template<unsigned D>`
`std::ostream & operator<< (std::ostream &ostr, const algebraic_face< D > &f)`
Print an `mln::topo::algebraic_face`.
- `template<unsigned N, unsigned D>`
`std::ostream & operator<< (std::ostream &ostr, const n_face< N, D > &f)`
Print an `mln::topo::n_face`.
- `template<unsigned D>`
`std::ostream & operator<< (std::ostream &ostr, const face< D > &f)`
Print an `mln::topo::face`.
- `template<unsigned D>`
`std::ostream & operator<< (std::ostream &ostr, const complex< D > &c)`
Pretty print a complex.
- `template<unsigned D>`
`bool operator== (const complex< D > &lhs, const complex< D > &rhs)`
Compare two complexes for equality.
- `template<unsigned D>`
`algebraic_face< D > operator- (const face< D > &f)`
Inversion operators.
- `template<unsigned D>`
`bool operator== (const algebraic_face< D > &lhs, const algebraic_face< D > &rhs)`
Comparison of two instances of `mln::topo::algebraic_face`.
- `template<unsigned D>`
`bool operator!= (const algebraic_face< D > &lhs, const algebraic_face< D > &rhs)`
Is lhs different from rhs?
- `template<unsigned D>`
`bool operator< (const algebraic_face< D > &lhs, const algebraic_face< D > &rhs)`
Is lhs "less" than rhs?

- `template<unsigned N, unsigned D>`
`algebraic_n_face< N, D > operator-` (const `n_face< N, D >` &f)
Inversion operators.

- `template<unsigned N, unsigned D>`
`bool operator==` (const `algebraic_n_face< N, D >` &lhs, const `algebraic_n_face< N, D >` &rhs)
Comparison of two instances of `mln::topo::algebraic_n_face`.

- `template<unsigned N, unsigned D>`
`bool operator!=` (const `algebraic_n_face< N, D >` &lhs, const `algebraic_n_face< N, D >` &rhs)
Is lhs different from rhs?

- `template<unsigned N, unsigned D>`
`bool operator<` (const `algebraic_n_face< N, D >` &lhs, const `algebraic_n_face< N, D >` &rhs)
Is lhs "less" than rhs?

- `template<unsigned D>`
`algebraic_n_face< 1, D > edge` (const `n_face< 0, D >` &f1, const `n_face< 0, D >` &f2)
Helpers.

- `template<unsigned D>`
`bool operator==` (const `face< D >` &lhs, const `face< D >` &rhs)
Comparison of two instances of `mln::topo::face`.

- `template<unsigned D>`
`bool operator!=` (const `face< D >` &lhs, const `face< D >` &rhs)
Is lhs different from rhs?

- `template<unsigned D>`
`bool operator<` (const `face< D >` &lhs, const `face< D >` &rhs)
Is lhs "less" than rhs?

- `template<unsigned N, unsigned D>`
`bool operator==` (const `n_face< N, D >` &lhs, const `n_face< N, D >` &rhs)
Comparison of two instances of `mln::topo::n_face`.

- `template<unsigned N, unsigned D>`
`bool operator!=` (const `n_face< N, D >` &lhs, const `n_face< N, D >` &rhs)
Is lhs different from rhs?

- `template<unsigned N, unsigned D>`
`bool operator<` (const `n_face< N, D >` &lhs, const `n_face< N, D >` &rhs)
Is lhs "less" than rhs?

- `template<unsigned N, unsigned D>`
`n_faces_set< N, D > operator+` (const `algebraic_n_face< N, D >` &f1, const `algebraic_n_face< N, D >` &f2)
Addition.

- `template<unsigned N, unsigned D>`
`n_faces_set< N, D > operator-` (const `algebraic_n_face< N, D >` &f1, const `algebraic_n_face<`
`N, D >` &f2)
Subtraction.

9.141.1 Detailed Description

Namespace of "point-wise" expression tools.

9.141.2 Function Documentation

9.141.2.1 `template<unsigned D, typename G > void mln::topo::detach (const complex_site< D, G > & f, complex_image< D, G, bool > & ima) [inline]`

Detach the cell corresponding to *f* from *ima*.

Precondition

f is a facet (it does not belong to any face of higher dimension).
ima is an image of Boolean values.

Definition at line 58 of file detach.hh.

References `mln::make::detachment()`, `mln::data::fill()`, and `is_facet()`.

9.141.2.2 `template<unsigned D> algebraic_n_face< 1, D > mln::topo::edge (const n_face< 0, D > & f1, const n_face< 0, D > & f2)`

Helpers.

Return the algebraic 1-face (edge) linking the 0-faces (vertices) *f1* and *f2*. If there is no 1-face between *f1* and *f2*, return an invalid 1-face.

Precondition

f1 and *f2* must belong to the same complex.

Note: this routine assumes the complex is not degenerated, i.e,

- it does not check that *f1* and *f2* are the only 0-faces adjacent to an hypothetical 1-face; it just checks that *f1* and *f2* share a common 1-face;
- if there are several adjacent 1-faces shared by *f1* and *f2* (if the complex is ill-formed), there is no guarantee on the returned 1-face (the current implementation return the first 1-face found, but client code should not rely on this implementation-defined behavior).

Definition at line 286 of file algebraic_n_face.hh.

References `mln::topo::n_face< N, D >::higher_dim_adj_faces()`.

9.141.2.3 `template<unsigned D, typename G > bool mln::topo::is_facet (const complex_psite< D, G > & f) [inline]`

Is *f* a facet, i.e., a face not “included in” (adjacent to) a face of higher dimension?

Definition at line 58 of file `is_facet.hh`.

Referenced by `mln::make::attachment()`, `mln::make::cell()`, `detach()`, and `mln::make::detachment()`.

9.141.2.4 `template<unsigned D> algebraic_face< D > mln::topo::make_algebraic_face (const face< D > & f, bool sign)`

Create an algebraic face handle of a `D-complex`.

Definition at line 211 of file `algebraic_face.hh`.

9.141.2.5 `template<unsigned N, unsigned D> algebraic_n_face< N, D > mln::topo::make_algebraic_n_face (const n_face< N, D > & f, bool sign)`

Create an algebraic `N-face` handle of a `D-complex`.

Definition at line 213 of file `algebraic_n_face.hh`.

9.141.2.6 `template<unsigned D> bool mln::topo::operator!= (const algebraic_face< D > & lhs, const algebraic_face< D > & rhs) [inline]`

Is *lhs* different from *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same `mln::topo::complex`.

Definition at line 250 of file `algebraic_face.hh`.

References `mln::topo::face< D >::cplx()`.

9.141.2.7 `template<unsigned D> bool mln::topo::operator!= (const face< D > & lhs, const face< D > & rhs) [inline]`

Is *lhs* different from *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same `mln::topo::complex`.

Definition at line 394 of file `face.hh`.

References `mln::topo::face< D >::cplx()`.

9.141.2.8 `template<unsigned N, unsigned D> bool mln::topo::operator!= (const n_face< N, D > & lhs, const n_face< N, D > & rhs) [inline]`

Is *lhs* different from *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 291 of file `n_face.hh`.

References `mln::topo::n_face< N, D >::cplx()`.

9.141.2.9 `template<unsigned N, unsigned D> bool mln::topo::operator!=(const algebraic_n_face< N, D > & lhs, const algebraic_n_face< N, D > & rhs) [inline]`

Is *lhs* different from *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 250 of file `algebraic_n_face.hh`.

References `mln::topo::n_face< N, D >::cplx()`.

9.141.2.10 `template<unsigned N, unsigned D> n_faces_set< N, D > mln::topo::operator+ (const algebraic_n_face< N, D > & f1, const algebraic_n_face< N, D > & f2) [inline]`

Addition.

Definition at line 206 of file `n_faces_set.hh`.

References `mln::topo::n_faces_set< N, D >::add()`.

9.141.2.11 `template<unsigned N, unsigned D> algebraic_n_face< N, D > mln::topo::operator- (const n_face< N, D > & f)`

Inversion operators.

Definition at line 221 of file `algebraic_n_face.hh`.

9.141.2.12 `template<unsigned D> algebraic_face< D > mln::topo::operator- (const face< D > & f)`

Inversion operators.

Definition at line 219 of file `algebraic_face.hh`.

9.141.2.13 `template<unsigned N, unsigned D> n_faces_set< N, D > mln::topo::operator- (const algebraic_n_face< N, D > & f1, const algebraic_n_face< N, D > & f2) [inline]`

Subtraction.

Definition at line 284 of file `n_faces_set.hh`.

References `mln::topo::n_faces_set< N, D >::add()`.

9.141.2.14 `template<unsigned N, unsigned D> bool mln::topo::operator< (const n_face< N, D > & lhs, const n_face< N, D > & rhs) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting face handles.

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 301 of file n_face.hh.

9.141.2.15 `template<unsigned D> bool mln::topo::operator< (const face< D > & lhs, const face< D > & rhs) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting face handles.

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).
Arguments *lhs* and *rhs* must have the same dimension.

Definition at line 404 of file face.hh.

9.141.2.16 `template<unsigned D> bool mln::topo::operator< (const algebraic_face< D > & lhs, const algebraic_face< D > & rhs) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting algebraic face handles.

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).
Arguments *lhs* and *rhs* must have the same dimension.

Definition at line 260 of file algebraic_face.hh.

9.141.2.17 `template<unsigned N, unsigned D> bool mln::topo::operator< (const algebraic_n_face< N, D > & lhs, const algebraic_n_face< N, D > & rhs) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting algebraic face handles.

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 261 of file algebraic_n_face.hh.

9.141.2.18 `template<unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const complex< D > & c) [inline]`

Pretty print a complex.

Definition at line 670 of file complex.hh.

References `mln::topo::complex< D >::print()`.

9.141.2.19 `template<unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const face< D > & f) [inline]`

Print an `mln::topo::face`.

Definition at line 416 of file face.hh.

9.141.2.20 `template<unsigned N, unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const n_face< N, D > & f) [inline]`

Print an `mln::topo::n_face`.

Definition at line 312 of file n_face.hh.

9.141.2.21 `template<unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const algebraic_face< D > & f) [inline]`

Print an `mln::topo::algebraic_face`.

Definition at line 273 of file algebraic_face.hh.

9.141.2.22 `template<unsigned N, unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const algebraic_n_face< N, D > & f) [inline]`

Print an `mln::topo::algebraic_n_face`.

Definition at line 273 of file algebraic_n_face.hh.

9.141.2.23 `template<unsigned D> bool mln::topo::operator==(const face< D > & lhs, const face< D > & rhs) [inline]`

Comparison of two instances of `mln::topo::face`.

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same `mln::topo::complex`.

Definition at line 384 of file face.hh.

References `mln::topo::face< D >::cplx()`, `mln::topo::face< D >::face_id()`, and `mln::topo::face< D >::n()`.

9.141.2.24 `template<unsigned N, unsigned D> bool mln::topo::operator==(const n_face< N, D > & lhs, const n_face< N, D > & rhs) [inline]`

Comparison of two instances of [mln::topo::n_face](#).

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 281 of file `n_face.hh`.

References `mln::topo::n_face< N, D >::cplx()`, and `mln::topo::n_face< N, D >::face_id()`.

9.141.2.25 `template<unsigned N, unsigned D> bool mln::topo::operator==(const algebraic_n_face< N, D > & lhs, const algebraic_n_face< N, D > & rhs) [inline]`

Comparison of two instances of [mln::topo::algebraic_n_face](#).

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 239 of file `algebraic_n_face.hh`.

References `mln::topo::n_face< N, D >::cplx()`, `mln::topo::n_face< N, D >::face_id()`, and `mln::topo::algebraic_n_face< N, D >::sign()`.

9.141.2.26 `template<unsigned D> bool mln::topo::operator==(const algebraic_face< D > & lhs, const algebraic_face< D > & rhs) [inline]`

Comparison of two instances of [mln::topo::algebraic_face](#).

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 237 of file `algebraic_face.hh`.

References `mln::topo::face< D >::cplx()`, `mln::topo::face< D >::face_id()`, `mln::topo::face< D >::n()`, and `mln::topo::algebraic_face< D >::sign()`.

9.141.2.27 `template<unsigned D> bool mln::topo::operator==(const complex< D > & lhs, const complex< D > & rhs) [inline]`

Compare two complexes for equality.

Definition at line 657 of file `complex.hh`.

9.142 mln::trace Namespace Reference

Namespace of routines related to the trace mechanism.

9.142.1 Detailed Description

Namespace of routines related to the trace mechanism.

9.143 mln::trait Namespace Reference

Namespace where traits are defined.

9.143.1 Detailed Description

Namespace where traits are defined. Namespace for image traits.

9.144 mln::transform Namespace Reference

Namespace of transforms.

Functions

- `template<typename I, typename N, typename D >`
`util::couple< mln::trait::ch_value< I, D >::ret, mln::trait::ch_value< I, typename I::psite >::ret >`
`distance_and_closest_point_geodesic` (const `Image< I >` &input, const `Neighborhood< N >` &nbh,
 D max)
Discrete geodesic distance transform.
- `template<typename P, typename N, typename D >`
`util::couple< mln_image_from_grid(mln_grid(P), D), mln_image_from_grid(mln_grid(P),`
`unsigned)> distance_and_closest_point_geodesic` (const `p_array< P >` &pset, const `box< P`
`>` &closest_point_domain, const `Neighborhood< N >` &nbh, D max)
Discrete geodesic distance transform.
- `template<typename I, typename N, typename D >`
`util::couple< mln::trait::ch_value< I, D >::ret, I > distance_and_influence_zone_geodesic` (const
`Image< I >` &input, const `Neighborhood< N >` &nbh, D max)
Discrete geodesic distance transform.
- `template<typename I, typename N, typename W, typename D >`
`mln::trait::ch_value< I, D >::ret distance_front` (const `Image< I >` &input, const `Neighborhood<`
`N >` &nbh, const `Weighted_Window< W >` &w_win, D max)
Discrete front distance transform.
- `template<typename I, typename N, typename D >`
`mln::trait::ch_value< I, D >::ret distance_geodesic` (const `Image< I >` &input, const `Neighbor-`
`hood< N >` &nbh, D max)

Discrete geodesic distance transform.

- `template<typename I >`
`image2d< float > hough (const Image< I > &input_)`
Compute the hough transform from a binary image.
- `template<typename I , typename N , typename W >`
`mln::trait::concrete< I >::ret influence_zone_front (const Image< I > &input, const Neighborhood< N > &nbh, const Weighted_Window< W > &w_win)`
Influence zone transform.
- `template<typename I , typename N , typename W , typename D >`
`mln::trait::concrete< I >::ret influence_zone_front (const Image< I > &input, const Neighborhood< N > &nbh, const Weighted_Window< W > &w_win, D max)`
Influence zone transform.
- `template<typename I , typename N >`
`mln::trait::concrete< I >::ret influence_zone_geodesic (const Image< I > &input, const Neighborhood< N > &nbh)`
Geodesic influence zone transform.
- `template<typename I , typename N , typename D >`
`mln::trait::concrete< I >::ret influence_zone_geodesic_saturated (const Image< I > &input, const Neighborhood< N > &nbh, const D &max, const typename I::value &background_value)`
Geodesic influence zone transform.
- `template<typename I , typename N , typename D >`
`mln::trait::concrete< I >::ret influence_zone_geodesic_saturated (const Image< I > &input, const Neighborhood< N > &nbh, const D &max)`

9.144.1 Detailed Description

Namespace of transforms.

9.144.2 Function Documentation

- 9.144.2.1** `template<typename I , typename N , typename D > util::couple<`
`mln::trait::ch_value< I, D >::ret, mln::trait::ch_value< I, typename I::psite >::ret >`
`mln::transform::distance_and_closest_point_geodesic (const Image< I > & input,`
`const Neighborhood< N > & nbh, D max) [inline]`

Discrete geodesic distance transform.

Parameters

- [in] *input* Image from which the geodesic distance is computed.
- [in] *nbh* Neighborhood
- [in] *max* Max distance of propagation.

Returns

a couple of images. The first one is the distance map and the second one is the closest point image. The closest point image contains sites.

Postcondition

The returned images have the same domain as `input`.

Definition at line 90 of file `distance_and_closest_point_geodesic.hh`.

References `mln::make::couple()`, and `distance_geodesic()`.

```
9.144.2.2 template<typename P , typename N , typename D > util::couple<
mln_image_from_grid(mln_grid(P), D), mln_image_from_grid(mln_grid(P),
unsigned)> mln::transform::distance_and_closest_point_geodesic ( const p_array< P
> & pset, const box< P > & closest_point_domain, const Neighborhood< N > & nbh,
D max ) [inline]
```

Discrete geodesic distance transform.

Parameters

[in] *pset* an array of sites.

[in] *closest_point_domain* domain of the returned image.

[in] *nbh* neighborhood

[in] *max* max distance of propagation.

Returns

A couple of images. The first one is the distance map and the second one is the closest point image. The closest point image contains site indexes.

Postcondition

The returned image domains are defined on `closest_point_domain`.

Definition at line 110 of file `distance_and_closest_point_geodesic.hh`.

References `mln::geom::bbox()`, `mln::make::couple()`, `distance_geodesic()`, `mln::data::fill()`, and `mln::box< P >::is_valid()`.

```
9.144.2.3 template<typename I , typename N , typename D > util::couple< mln::trait::ch_value<
I, D >::ret, I > mln::transform::distance_and_influence_zone_geodesic ( const Image<
I > & input, const Neighborhood< N > & nbh, D max ) [inline]
```

Discrete geodesic distance transform.

Parameters

[in] *input* [Image](#) from which the geodesic distance is computed.

[in] *nbh* [Neighborhood](#)

[in] *max* Max distance of propagation.

Returns

a couple of images. The first one is the distance map and the second one is the closest point image. The closest point image contains sites.

Postcondition

The returned images have the same domain as `input`.

Definition at line 69 of file `distance_and_influence_zone_geodesic.hh`.

References `mln::make::couple()`, and `distance_geodesic()`.

9.144.2.4 `template<typename I, typename N, typename W, typename D > mln::trait::ch_value< I, D >::ret mln::transform::distance_front (const Image< I > & input, const Neighborhood< N > & nbh, const Weighted_Window< W > & w_win, D max) [inline]`

Discrete front distance transform.

Definition at line 56 of file `transform/distance_front.hh`.

9.144.2.5 `template<typename I, typename N, typename D > mln::trait::ch_value< I, D >::ret mln::transform::distance_geodesic (const Image< I > & input, const Neighborhood< N > & nbh, D max) [inline]`

Discrete geodesic distance transform.

Definition at line 55 of file `transform/distance_geodesic.hh`.

Referenced by `distance_and_closest_point_geodesic()`, and `distance_and_influence_zone_geodesic()`.

9.144.2.6 `template<typename I > image2d< float > mln::transform::hough (const Image< I > & input_)`

Compute the hough transform from a binary image.

Objects used for computation must be set to 'true'.

Parameters

[in] `input_` A binary image.

Returns

A 2D image of float. Rows are used for the distance and columns are used for the angles. Angles go from 0 to 359. Distance goes from 0 to the maximum distance between the center and a corner. The site having the maximum value indicates through its column index the document inclination.

Definition at line 98 of file `hough.hh`.

References `mln::opt::at()`, `mln::data::fill()`, `mln::geom::min_col()`, `mln::geom::min_row()`, `mln::geom::ncols()`, and `mln::geom::nrows()`.

9.144.2.7 `template<typename I , typename N , typename W > mln::trait::concrete< I >::ret mln::transform::influence_zone_front (const Image< I > & input, const Neighborhood< N > & nbh, const Weighted_Window< W > & w_win)`

Influence zone transform.

Definition at line 78 of file `influence_zone_front.hh`.

References `influence_zone_front()`.

9.144.2.8 `template<typename I , typename N , typename W , typename D > mln::trait::concrete< I >::ret mln::transform::influence_zone_front (const Image< I > & input, const Neighborhood< N > & nbh, const Weighted_Window< W > & w_win, D max)`

Influence zone transform.

Definition at line 60 of file `influence_zone_front.hh`.

References `mln::canvas::distance_front()`.

Referenced by `influence_zone_front()`.

9.144.2.9 `template<typename I , typename N > mln::trait::concrete< I >::ret mln::transform::influence_zone_geodesic (const Image< I > & input, const Neighborhood< N > & nbh)`

Geodesic influence zone transform.

Parameters

[in] *input* An image.

[in] *nbh* A neighborhood.

Returns

An image of influence zone.

Definition at line 216 of file `influence_zone_geodesic.hh`.

9.144.2.10 `template<typename I , typename N , typename D > mln::trait::concrete< I >::ret mln::transform::influence_zone_geodesic_saturated (const Image< I > & input, const Neighborhood< N > & nbh, const D & max, const typename I::value & background_value)`

Geodesic influence zone transform.

Parameters

[in] *input* An image.

[in] *nbh* A neighborhood.

[in] *max* The maximum influence zone distance.

[in] *background_value* The value used as background (i.e. not propagated).

Returns

An image of influence zone.

Definition at line 73 of file influence_zone_geodesic_saturated.hh.

References mln::canvas::distance_geodesic().

Referenced by influence_zone_geodesic_saturated().

9.144.2.11 `template<typename I , typename N , typename D > mln::trait::concrete< I >::ret
mln::transform::influence_zone_geodesic_saturated (const Image< I > & input,
const Neighborhood< N > & nbh, const D & max)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 93 of file influence_zone_geodesic_saturated.hh.

References influence_zone_geodesic_saturated(), and mln::literal::zero.

9.145 mln::util Namespace Reference

Namespace of tools using for more complex algorithm.

Namespaces

- namespace [impl](#)
Implementation namespace of util namespace.

Classes

- class [adjacency_matrix](#)
A class of adjacency matrix.
- class [array](#)
A dynamic array class.
- class [branch](#)
Class of generic branch.
- class [branch_iter](#)
Basic 2D image class.
- class [branch_iter_ind](#)
Basic 2D image class.
- class [couple](#)
Definition of a couple.
- struct [eat](#)
Eat structure.

- class [edge](#)
Edge of a graph G .
- class [fibonacci_heap](#)
Fibonacci heap.
- class [graph](#)
Undirected graph.
- class [greater_point](#)
A “greater than” functor comparing points w.r.t.
- class [greater_psite](#)
A “greater than” functor comparing psites w.r.t.
- class [head](#)
Top structure of the soft heap.
- struct [ignore](#)
Ignore structure.
- struct [ilcell](#)
Element of an item list. Store the data (key) used in [soft_heap](#).
- class [line_graph](#)
Undirected line graph of a graph of type G .
- struct [nil](#)
Nil structure.
- class [node](#)
Meta-data of an element in the heap.
- class [object_id](#)
Base class of an object id.
- struct [ord](#)
Function-object that defines an ordering between objects with type T : lhs R rhs.
- struct [ord_pair](#)
Ordered pair structure s.a.
- struct [pix](#)
Structure pix.
- class [set](#)
An “efficient” mathematical set class.
- class [site_pair](#)
A pair of sites.

- class [soft_heap](#)
Soft heap.
- class [timer](#)
Timer structure.
- struct [tracked_ptr](#)
Smart pointer for shared data with tracking.
- class [tree](#)
Class of generic tree.
- class [tree_node](#)
Class of generic [tree_node](#) for tree.
- class [vertex](#)
Vertex of a graph G.
- struct [yes](#)
Object that always says "yes".

Typedefs

- typedef [object_id](#)< [vertex_tag](#), unsigned > [vertex_id_t](#)
Vertex id type.

Functions

- template<typename I, typename J >
void [display_branch](#) (const [Image](#)< J > &ima_, [tree_node](#)< I > *[tree_node](#))
Display an arborescence from [tree_node](#).
- template<typename I, typename J >
void [display_tree](#) (const [Image](#)< J > &ima_, [tree](#)< I > &[tree](#))
Display a tree.
- template<typename I >
I::psite [lemmings](#) (const [Image](#)< I > &ima, const typename I::psite &pt, const typename I::psite::delta &dpt, const typename I::value &val)
Launch a lemmings on an image.
- template<typename I >
[greater_point](#)< I > [make_greater_point](#) (const [Image](#)< I > &ima)
Helper to build a [mln::util::greater_point](#).
- template<typename I >
[greater_psite](#)< I > [make_greater_psite](#) (const [Image](#)< I > &ima)

Helper to build a `mln::util::greater_psite`.

- `template<typename G >`
`bool operator< (const vertex< G > &lhs, const vertex< G > &rhs)`
Less operator. Test whether lhs.id() < rhs.id().

- `template<typename G >`
`std::ostream & operator<< (std::ostream &ostr, const vertex< G > &v)`
Push the vertex v in the output stream ostr.

- `template<typename T >`
`std::ostream & operator<< (std::ostream &ostr, const array< T > &a)`
Operator<<.

- `template<typename G >`
`bool operator== (const vertex< G > &v1, const vertex< G > &v2)`
Equality operator.

- `template<typename T >`
`bool operator== (const array< T > &lhs, const array< T > &rhs)`
Operator==.

- `template<typename T >`
`bool ord_strict (const T &lhs, const T &rhs)`
Routine to test if lhs is strictly "less-than" rhs.

- `template<typename T >`
`bool ord_weak (const T &lhs, const T &rhs)`
Routine to test if lhs is "less-than or equal-to" rhs.

- `template<typename T, typename I >`
`void tree_fast_to_image (tree_fast< T > &tree, Image< I > &output_)`
- `template<typename T >`
`tree_fast< T > tree_to_fast (tree< T > &input)`
Facade.

- `template<typename T, typename I >`
`void tree_to_image (tree< T > &tree, Image< I > &output_)`
Convert a tree into an image.

9.145.1 Detailed Description

Namespace of tools using for more complex algorithm. Forward declaration.

9.145.2 Typedef Documentation

9.145.2.1 typedef `object_id<vertex_tag, unsigned> mln::util::vertex_id_t`

`Vertex` id type.

Definition at line 43 of file `graph_ids.hh`.

9.145.3 Function Documentation

9.145.3.1 `template<typename I, typename J> void mln::util::display_branch (const Image< J > & ima_, tree_node< I > * tree_node) [inline]`

Display an arborescence from `tree_node`.

Parameters

- [in] `ima_` The domain of output image.
- [in] `tree_node` The root `tree_node` to display.

Definition at line 210 of file `tree_to_image.hh`.

References `mln::data::fill()`.

9.145.3.2 `template<typename I, typename J> void mln::util::display_tree (const Image< J > & ima_, tree< I > & tree) [inline]`

Display a tree.

Parameters

- [in] `ima_` The domain of output image.
- [in] `tree` The tree to display.

Definition at line 192 of file `tree_to_image.hh`.

References `mln::util::tree< T >::root()`.

9.145.3.3 `template<typename I> I::psite mln::util::lemmings (const Image< I > & ima, const typename I::psite & pt, const typename I::psite::delta & dpt, const typename I::value & val)`

Launch a lemmings on an image.

A lemmings is the point `pt` that you put on an image `ima`. This point will move through the image using the delta-point `dpt` while consider his value on the given image.

Returns

The first point that is not in the domain `domain` or which value on the given image is different to the value `val`.

Precondition

The domain `domain` must be contained in the domain of `ima`.

Definition at line 104 of file `lemmings.hh`.

9.145.3.4 `template<typename I> greater_point< I > mln::util::make_greater_point (const Image< I > & ima)`

Helper to build a `mln::util::greater_point`.

Definition at line 82 of file `greater_point.hh`.

9.145.3.5 `template<typename I > greater_psite< I > mln::util::make_greater_psite (const Image< I > & ima)`

Helper to build a [mln::util::greater_psite](#).

Definition at line 82 of file `greater_psite.hh`.

9.145.3.6 `template<typename G > bool mln::util::operator< (const vertex< G > & lhs, const vertex< G > & rhs) [inline]`

Less operator. Test whether `lhs.id() < rhs.id()`.

Definition at line 390 of file `vertex.hh`.

9.145.3.7 `template<typename G > std::ostream & mln::util::operator<< (std::ostream & ostr, const vertex< G > & v) [inline]`

Push the vertex `v` in the output stream `ostr`.

Definition at line 372 of file `vertex.hh`.

9.145.3.8 `template<typename T > std::ostream & mln::util::operator<< (std::ostream & ostr, const array< T > & a)`

Operator<<.

Definition at line 796 of file `util/array.hh`.

References `mln::util::array< T >::nelements()`.

9.145.3.9 `template<typename G > bool mln::util::operator== (const vertex< G > & v1, const vertex< G > & v2) [inline]`

Equality operator.

Test whether two vertices have the same id.

Definition at line 380 of file `vertex.hh`.

References `mln::util::vertex< G >::graph()`, and `mln::util::vertex< G >::id()`.

9.145.3.10 `template<typename T > bool mln::util::operator== (const array< T > & lhs, const array< T > & rhs)`

Operator==.

Definition at line 815 of file `util/array.hh`.

References `mln::util::array< T >::std_vector()`.

9.145.3.11 `template<typename T > bool mln::util::ord_strict (const T & lhs, const T & rhs) [inline]`

Routine to test if `lhs` is strictly "less-than" `rhs`.

Definition at line 90 of file util/ord.hh.

Referenced by mln::util::ord_pair< T >::change_both(), mln::util::ord_pair< T >::change_first(), and mln::util::ord_pair< T >::change_second().

9.145.3.12 `template<typename T > bool mln::util::ord_weak (const T & lhs, const T & rhs) [inline]`

Routine to test if *lhs* is "less-than or equal-to" *rhs*.

Definition at line 101 of file util/ord.hh.

Referenced by mln::util::ord_pair< T >::change_both(), mln::util::ord_pair< T >::change_first(), mln::util::ord_pair< T >::change_second(), and mln::box< P >::is_valid().

9.145.3.13 `template<typename T , typename I > void mln::util::tree_fast_to_image (tree_fast< T > & tree, Image< I > & output_) [inline]`

Convert a *tree_fast* into an image.

Parameters

[in] *tree* The tree to convert.

[out] *output_* The image containing tree informations.

Definition at line 99 of file tree_fast_to_image.hh.

9.145.3.14 `template<typename T > tree_fast< T > mln::util::tree_to_fast (tree< T > & input) [inline]`

Facade.

Convert a tree into an *tree_fast*.

Parameters

[in] *input* The tree to convert.

Returns

The *tree_fast* containing tree informations.

Definition at line 90 of file tree_to_fast.hh.

References mln::util::tree< T >::root().

9.145.3.15 `template<typename T , typename I > void mln::util::tree_to_image (tree< T > & tree, Image< I > & output_) [inline]`

Convert a tree into an image.

Parameters

[in] *tree* The tree to convert.

[out] *output_* The image containing tree information.

Definition at line 178 of file tree_to_image.hh.

9.146 `mln::util::impl` Namespace Reference

Implementation namespace of util namespace.

9.146.1 Detailed Description

Implementation namespace of util namespace.

9.147 `mln::value` Namespace Reference

Namespace of materials related to pixel value types.

Namespaces

- namespace `impl`
Implementation namespace of value namespace.

Classes

- class `float01`
Class for floating values restricted to the interval [0..1] and discretized with n bits.
- struct `float01_f`
Class for floating values restricted to the interval [0..1].
- struct `graylevel`
General gray-level class on n bits.
- struct `graylevel_f`
General gray-level class on n bits.
- struct `int_s`
Signed integer value class.
- struct `int_u`
Unsigned integer value class.
- struct `int_u_sat`
Unsigned integer value class with saturation behavior.
- struct `Integer`
Concept of integer.
- struct `Integer< void >`
Category flag type.

- struct [label](#)
Label value class.
- struct [lut_vec](#)
Class that defines FIXME.
- class [proxy](#)
Generic proxy class for an image pixel value.
- struct [rgb](#)
Color class for red-green-blue where every component is n-bit encoded.
- struct [set](#)
Class that defines the set of values of type T.
- class [sign](#)
The sign class represents the value type composed by the set (-1, 0, 1) sign value type is a subset of the int value type.
- struct [stack_image](#)
Stack image class.
- struct [super_value](#)< [sign](#) >
Specializations:
- struct [value_array](#)
Generic array class over indexed by a value set with type T.

Typedefs

- typedef [float01_](#)< 16 > [float01_16](#)
Alias for 16 bit [float01](#).
- typedef [float01_](#)< 8 > [float01_8](#)
Alias for 8 bit [float01](#).
- typedef [graylevel](#)< 16 > [gl16](#)
Alias for 16 bit [graylevel](#).
- typedef [graylevel](#)< 8 > [gl8](#)
Alias for 8 bit [graylevel](#).
- typedef [graylevel_f](#) [glf](#)
Alias for [graylevels](#) encoded by [float](#).
- typedef [int_s](#)< 16 > [int_s16](#)
Alias for signed 16-bit integers.
- typedef [int_s](#)< 32 > [int_s32](#)

Alias for signed 32-bit integers.

- typedef `int_s< 8 > int_s8`
Alias for signed 8-bit integers.
- typedef `int_u< 12 > int_u12`
Alias for unsigned 12-bit integers.
- typedef `int_u< 16 > int_u16`
Alias for unsigned 16-bit integers.
- typedef `mln::value::int_u< 32 > int_u32`
Alias for unsigned 32-bit integers.
- typedef `mln::value::int_u< 8 > int_u8`
Alias for unsigned 8-bit integers.
- typedef `label< 16 > label_16`
Alias for 16-bit integers.
- typedef `label< 32 > label_32`
Alias for 32-bit integers.
- typedef `mln::value::label< 8 > label_8`
Alias for 8-bit labels.
- typedef `rgb< 16 > rgb16`
Color class for red-green-blue where every component is 16-bit encoded.
- typedef `rgb< 8 > rgb8`
Color class for red-green-blue where every component is 8-bit encoded.

Functions

- template<typename Dest , typename Src >
Dest `cast` (const Src &src)
Cast a value `src` from type `Src` to type `Dest`.
- template<typename V >
internal::equiv_< V >::ret `equiv` (const mln::Value< V > &v)
Access to the equivalent value.
- template<unsigned n>
`rgb< n >::interop operator+` (const `rgb< n >` &lhs, const `rgb< n >` &rhs)
Addition.
- template<typename H , typename S , typename L >
`hsl_< H, S, L > operator+` (const `hsl_< H, S, L >` &lhs, const `hsl_< H, S, L >` &rhs)
Addition.

- `template<unsigned n>`
`std::ostream & operator<<< (std::ostream &ostr, const label< n > &l)`
Print a label `l` into the output stream `ostr`.
- `template<unsigned n>`
`std::ostream & operator<<< (std::ostream &ostr, const rgb< n > &c)`
Print an rgb `c` into the output stream `ostr`.
- `std::ostream & operator<<< (std::ostream &ostr, const graylevel_f &g)`
Op<<<.
- `template<typename T >`
`std::ostream & operator<<< (std::ostream &ostr, const scalar_< T > &s)`
Print a scalar `s` in an output stream `ostr`.
- `template<typename H, typename S, typename L >`
`std::ostream & operator<<< (std::ostream &ostr, const hsl_< H, S, L > &c)`
Print an hsl `c` into the output stream `ostr`.
- `template<unsigned n>`
`std::ostream & operator<<< (std::ostream &ostr, const graylevel< n > &g)`
Op<<<.
- `template<unsigned n>`
`std::ostream & operator<<< (std::ostream &ostr, const float01_< n > &f)`
Op<<<.
- `std::ostream & operator<<< (std::ostream &ostr, const sign &i)`
Print an signed integer `i` into the output stream `ostr`.
- `template<unsigned n>`
`std::ostream & operator<<< (std::ostream &ostr, const int_u< n > &i)`
Print an unsigned integer `i` into the output stream `ostr`.
- `template<unsigned n>`
`std::ostream & operator<<< (std::ostream &ostr, const int_s< n > &i)`
Print an signed integer `i` into the output stream `ostr`.
- `template<unsigned n>`
`std::ostream & operator<<< (std::ostream &ostr, const int_u_sat< n > &i)`
Print a saturated unsigned integer `i` into the output stream `ostr`.
- `bool operator== (const sign &lhs, const sign &rhs)`
Comparison operator.
- `template<typename V >`
`V other (const V &val)`
Give an other value than `val`.

- `template<typename H, typename S, typename L >`
`hsl_< H, S, L > operator-` (const `hsl_< H, S, L >` &lhs, const `hsl_< H, S, L >` &rhs)
Subtraction.
- `template<typename H, typename S, typename L, typename S2 >`
`hsl_< H, S, L > operator*` (const `hsl_< H, S, L >` &lhs, const `mln::value::scalar_< S2 >` &s)
Product.
- `template<typename H, typename S, typename L, typename S2 >`
`hsl_< H, S, L > operator/` (const `hsl_< H, S, L >` &lhs, const `mln::value::scalar_< S2 >` &s)
Division.
- `template<typename H, typename S, typename L >`
`bool operator==` (const `hsl_< H, S, L >` &lhs, const `hsl_< H, S, L >` &rhs)
Comparison.
- `template<unsigned n>`
`rgb< n >::interop operator-` (const `rgb< n >` &lhs, const `rgb< n >` &rhs)
Subtraction.
- `template<unsigned n, typename S >`
`rgb< n >::interop operator*` (const `rgb< n >` &lhs, const `mln::value::scalar_< S >` &s)
Product.
- `template<unsigned n, typename S >`
`rgb< n >::interop operator/` (const `rgb< n >` &lhs, const `mln::value::scalar_< S >` &s)
Division.
- `template<typename I >`
`stack_image< 2, const I > stack` (const `Image< I >` &ima1, const `Image< I >` &ima2)
Shortcut to build a stack with two images.

9.147.1 Detailed Description

Namespace of materials related to pixel value types.

9.147.2 Typedef Documentation

9.147.2.1 typedef `float01_<16> mln::value::float01_16`

Alias for 16 bit `float01`.

Definition at line 45 of file `float01_16.hh`.

9.147.2.2 typedef float01_<8> mln::value::float01_8

Alias for 8 bit [float01](#).

Definition at line 45 of file float01_8.hh.

9.147.2.3 typedef graylevel<16> mln::value::gl16

Alias for 16 bit graylevel.

Definition at line 45 of file gl16.hh.

9.147.2.4 typedef graylevel<8> mln::value::gl8

Alias for 8 bit graylevel.

Definition at line 45 of file gl8.hh.

9.147.2.5 typedef graylevel_f mln::value::glf

Alias for graylevels encoded by float.

Definition at line 44 of file glf.hh.

9.147.2.6 typedef int_s<16> mln::value::int_s16

Alias for signed 16-bit integers.

Definition at line 45 of file int_s16.hh.

9.147.2.7 typedef int_s<32> mln::value::int_s32

Alias for signed 32-bit integers.

Definition at line 45 of file int_s32.hh.

9.147.2.8 typedef int_s<8> mln::value::int_s8

Alias for signed 8-bit integers.

Definition at line 45 of file int_s8.hh.

9.147.2.9 typedef int_u<12> mln::value::int_u12

Alias for unsigned 12-bit integers.

Definition at line 45 of file int_u12.hh.

9.147.2.10 typedef int_u<16> mln::value::int_u16

Alias for unsigned 16-bit integers.

Definition at line 45 of file int_u16.hh.

9.147.2.11 typedef mln::value::int_u<32> mln::value::int_u32

Alias for unsigned 32-bit integers.

Definition at line 45 of file int_u32.hh.

9.147.2.12 typedef mln::value::int_u<8> mln::value::int_u8

Alias for unsigned 8-bit integers.

Definition at line 44 of file int_u8.hh.

9.147.2.13 typedef label<16> mln::value::label_16

Alias for 16-bit integers.

Definition at line 44 of file label_16.hh.

9.147.2.14 typedef label<32> mln::value::label_32

Alias for 32-bit integers.

Definition at line 44 of file label_32.hh.

9.147.2.15 typedef mln::value::label<8> mln::value::label_8

Alias for 8-bit labels.

Definition at line 44 of file label_8.hh.

9.147.2.16 typedef rgb<16> mln::value::rgb16

Color class for red-green-blue where every component is 16-bit encoded.

Definition at line 45 of file rgb16.hh.

9.147.2.17 typedef rgb<8> mln::value::rgb8

Color class for red-green-blue where every component is 8-bit encoded.

Definition at line 45 of file rgb8.hh.

9.147.3 Function Documentation**9.147.3.1 template<typename Dest , typename Src > Dest mln::value::cast (const Src & src)
[inline]**

Cast a value `src` from type `Src` to type `Dest`.

Definition at line 85 of file value/cast.hh.

9.147.3.2 `template<typename V > internal::equiv_< V >::ret mln::value::equiv (const mln::Value< V > & v) [inline]`

Access to the equivalent value.

Definition at line 153 of file equiv.hh.

Referenced by mln::labeling::superpose().

9.147.3.3 `template<unsigned n, typename S > rgb< n >::interop mln::value::operator* (const rgb< n > & lhs, const mln::value::scalar_< S > & s) [inline]`

Product.

Definition at line 717 of file value/rgb.hh.

9.147.3.4 `template<typename H, typename S, typename L, typename S2 > hsl_< H, S, L > mln::value::operator* (const hsl_< H, S, L > & lhs, const mln::value::scalar_< S2 > & s)`

Product.

Definition at line 357 of file hsl.hh.

9.147.3.5 `template<unsigned n> rgb< n >::interop mln::value::operator+ (const rgb< n > & lhs, const rgb< n > & rhs) [inline]`

Addition.

{

Definition at line 663 of file value/rgb.hh.

9.147.3.6 `template<typename H, typename S, typename L > hsl_< H, S, L > mln::value::operator+ (const hsl_< H, S, L > & lhs, const hsl_< H, S, L > & rhs)`

Addition.

{

Definition at line 337 of file hsl.hh.

9.147.3.7 `template<unsigned n> rgb< n >::interop mln::value::operator- (const rgb< n > & lhs, const rgb< n > & rhs) [inline]`

Subtraction.

Definition at line 690 of file value/rgb.hh.

9.147.3.8 `template<typename H, typename S, typename L > hsl_< H, S, L > mln::value::operator- (const hsl_< H, S, L > & lhs, const hsl_< H, S, L > & rhs)`

Subtraction.

Definition at line 347 of file hsl.hh.

9.147.3.9 `template<unsigned n, typename S > rgb< n >::interop mln::value::operator/ (const rgb< n > & lhs, const mln::value::scalar_< S > & s) [inline]`

Division.

Definition at line 735 of file value/rgb.hh.

9.147.3.10 `template<typename H , typename S , typename L , typename S2 > hsl_< H, S, L > mln::value::operator/ (const hsl_< H, S, L > & lhs, const mln::value::scalar_< S2 > & s)`

Division.

Definition at line 367 of file hsl.hh.

9.147.3.11 `template<typename T > std::ostream & mln::value::operator<< (std::ostream & ostr, const scalar_< T > & s) [inline]`

Print a scalar *s* in an output stream *ostr*.

Definition at line 130 of file scalar.hh.

9.147.3.12 `std::ostream & mln::value::operator<< (std::ostream & ostr, const sign & i) [inline]`

Print an signed integer *i* into the output stream *ostr*.

Parameters

[in, out] *ostr* An output stream.

[in] *i* An sign value

Returns

The modified output stream *ostr*.

Definition at line 184 of file value/sign.hh.

References `mln::debug::format()`.

9.147.3.13 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr, const int_s< n > & i) [inline]`

Print an signed integer *i* into the output stream *ostr*.

Parameters

[in, out] *ostr* An output stream.

[in] *i* An signed integer.

Returns

The modified output stream *ostr*.

Definition at line 260 of file int_s.hh.

References mln::debug::format().

9.147.3.14 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr, const graylevel< n > & g) [inline]`

Op<<.

Definition at line 591 of file graylevel.hh.

9.147.3.15 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr, const int_u< n > & i) [inline]`

Print an unsigned integer *i* into the output stream *ostr*.

Parameters

[in, out] *ostr* An output stream.

[in] *i* An unsigned integer.

Returns

The modified output stream *ostr*.

Definition at line 357 of file int_u.hh.

References mln::debug::format().

9.147.3.16 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr, const int_u_sat< n > & i) [inline]`

Print a saturated unsigned integer *i* into the output stream *ostr*.

Parameters

[in, out] *ostr* An output stream.

[in] *i* A saturated unsigned integer.

Returns

The modified output stream *ostr*.

Definition at line 220 of file int_u_sat.hh.

References mln::debug::format().

9.147.3.17 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr, const rgb< n > & c) [inline]`

Print an rgb *c* into the output stream *ostr*.

Parameters

[in, out] *ostr* An output stream.

[in] *c* An rgb.

Returns

The modified output stream *ostr*.

Definition at line 743 of file value/rgb.hh.

References `mln::debug::format()`.

9.147.3.18 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr, const float01_<n> & f) [inline]`

Op<<.

Definition at line 253 of file float01_.hh.

9.147.3.19 `template<typename H , typename S , typename L > std::ostream & mln::value::operator<< (std::ostream & ostr, const hsl_<H, S, L> & c) [inline]`

Print an hsl *c* into the output stream *ostr*.

Parameters

[in, out] *ostr* An output stream.

[in] *c* An rgb.

Returns

The modified output stream *ostr*.

Definition at line 326 of file hsl.hh.

References `mln::debug::format()`.

9.147.3.20 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr, const label<n> & l) [inline]`

Print a label *l* into the output stream *ostr*.

Parameters

[in, out] *ostr* An output stream.

[in] *l* A label.

Returns

The modified output stream *ostr*.

Definition at line 353 of file label.hh.

References `mln::debug::format()`.

9.147.3.21 `std::ostream & mln::value::operator<< (std::ostream & ostr, const graylevel_f & g) [inline]`

Op<<.

Definition at line 458 of file `graylevel_f.hh`.

References `mln::value::graylevel_f::value()`.

9.147.3.22 `template<typename H , typename S , typename L > bool mln::value::operator==(const hsl_< H, S, L > & lhs, const hsl_< H, S, L > & rhs)`

Comparison.

Definition at line 376 of file `hsl.hh`.

9.147.3.23 `bool mln::value::operator==(const sign & lhs, const sign & rhs) [inline]`

Comparison operator.

Definition at line 190 of file `value/sign.hh`.

9.147.3.24 `template<typename V > V mln::value::other (const V & val) [inline]`

Give an other value than `val`.

Definition at line 115 of file `other.hh`.

9.147.3.25 `template<typename I > stack_image< 2, const I > mln::value::stack (const Image< I > & ima1, const Image< I > & ima2) [inline]`

Shortcut to build a stack with two images.

Definition at line 306 of file `stack.hh`.

9.148 `mln::value::impl` Namespace Reference

Implementation namespace of value namespace.

9.148.1 Detailed Description

Implementation namespace of value namespace.

9.149 `mln::win` Namespace Reference

Namespace of image processing routines related to win.

Classes

- struct [backdiag2d](#)
Diagonal line window defined on the 2D square grid.
- struct [ball](#)
Generic ball window defined on a given grid.
- struct [cube3d](#)
Cube window defined on the 3D grid.
- struct [cuboid3d](#)
Cuboid defined on the 3-D square grid.
- struct [diag2d](#)
Diagonal line window defined on the 2D square grid.
- struct [line](#)
Generic line window defined on a given grid in the given dimension.
- class [multiple](#)
Multiple window.
- class [multiple_size](#)
Definition of a multiple-size window.
- struct [octagon2d](#)
Octagon window defined on the 2D square grid.
- struct [rectangle2d](#)
Rectangular window defined on the 2D square grid.

Typedefs

- typedef [ball](#)< grid::square, def::coord > [disk2d](#)
2D disk window; precisely, ball-shaped window defined on the 2D square grid.
- typedef [line](#)< grid::square, 1, def::coord > [hline2d](#)
Horizontal line window defined on the 2D square grid.
- typedef [line](#)< grid::tick, 0, def::coord > [segment1d](#)
Segment window defined on the 1D grid.
- typedef [line](#)< grid::cube, 0, def::coord > [sline3d](#)
Depth line window defined on the 3D cubic grid.
- typedef [ball](#)< grid::cube, def::coord > [sphere3d](#)
3D sphere window; precisely, ball-shaped window defined on the 3D cubic grid.

- typedef [line](#)< grid::square, 0, def::coord > [vline2d](#)

Vertical line window defined on the 2D square grid.

Functions

- template<typename N1 , typename N2 >
[neighb](#)< typename N1::window::regular > [diff](#) (const [Neighborhood](#)< N1 > &nbh1, const [Neighborhood](#)< N2 > &nbh2)

Set difference between a couple of neighborhoods nbh1 and nbh2.

- template<typename W1 , typename W2 >
[mln_regular](#) (W1) [diff](#)(const [Window](#)< W1 > &win1

Set difference between a couple of windows win1 and win2.

- template<typename W >
[mln_regular](#) (W) [shift](#)(const [Window](#)< W > &win

Shift a window win with a delta-point dp.

- template<typename W >
W [sym](#) (const [Window](#)< W > &win)

Give the symmetrical window of win.

- template<typename W >
W [sym](#) (const [Weighted_Window](#)< W > &w_win)

Give the symmetrical weighted window of w_win.

9.149.1 Detailed Description

Namespace of image processing routines related to win.

9.149.2 Function Documentation

- 9.149.2.1** template<typename N1 , typename N2 > N2 [neighb](#)< typename N1::window::regular > [mln::win::diff](#) (const [Neighborhood](#)< N1 > & *nbh1*, const [Neighborhood](#)< N2 > & *nbh2*)

Set difference between a couple of neighborhoods nbh1 and nbh2.

Definition at line 132 of file win/diff.hh.

Referenced by mln::operator-().

- 9.149.2.2** template<typename W1 , typename W2 > [mln::win::mln_regular](#) (W1) const **[inline]**

Set difference between a couple of windows win1 and win2.

9.149.2.3 `template<typename W > mln::win::mln_regular (W) const [inline]`

Shift a window `win` with a delta-point `dp`.

9.149.2.4 `template<typename W > W mln::win::sym (const Window< W > & win) [inline]`

Give the symmetrical window of `win`.

Definition at line 59 of file `sym.hh`.

Referenced by `mln::c18()`, `mln::c26()`, `mln::c4_3d()`, `mln::c6()`, `mln::morpho::hit_or_miss_background_opening()`, `mln::morpho::hit_or_miss_opening()`, `mln::morpho::opening::approx::structural()`, and `mln::morpho::closing::approx::structural()`.

9.149.2.5 `template<typename W > W mln::win::sym (const Weighted_Window< W > & w_win) [inline]`

Give the symmetrical weighted window of `w_win`.

Definition at line 71 of file `sym.hh`.

Chapter 10

Class Documentation

10.1 `mln::accu::center< P, V >` Struct Template Reference

Mass center accumulator.

```
#include <center.hh>
```

Inherits `base< V, center< P, V > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `unsigned nsites () const`
Return the number of sites taken in consideration.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `V to_result () const`
Get the value of the accumulator.

- `void init ()`
Manipulators.

10.1.1 Detailed Description

```
template<typename P, typename V = typename P::vec> struct mln::accu::center< P, V >
```

Mass center accumulator.

Template Parameters

P the type of site.

V the type of vector to be used as result. The default vector type is the one provided by P.

Definition at line 55 of file center.hh.

10.1.2 Member Function Documentation

10.1.2.1 `template<typename P , typename V > void mln::accu::center< P, V >::init ()`
[inline]

Manipulators.

Definition at line 116 of file center.hh.

References mln::literal::zero.

10.1.2.2 `template<typename P , typename V > bool mln::accu::center< P, V >::is_valid ()`
const [inline]

Check whether this accu is able to return a result.

Definition at line 160 of file center.hh.

Referenced by mln::accu::center< P, V >::to_result().

10.1.2.3 `template<typename P , typename V > unsigned mln::accu::center< P, V >::nsites ()`
const [inline]

Return the number of sites taken in consideration.

Definition at line 168 of file center.hh.

10.1.2.4 `void mln::Accumulator< center< P, V > >::take_as_init (const T & t)`
[inherited]

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.1.2.5 `void mln::Accumulator< center< P, V > >::take_n_times (unsigned n, const T & t)`
[inherited]

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.1.2.6 `template<typename P , typename V > V mln::accu::center< P, V >::to_result () const`
[inline]

Get the value of the accumulator.

Definition at line 142 of file center.hh.

References mln::accu::center< P, V >::is_valid().

10.2 mln::accu::convolve< T1, T2, R > Struct Template Reference

Generic convolution accumulator class.

```
#include <convolve.hh>
```

Inherits base< R, convolve< T1, T2, R > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- R [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.2.1 Detailed Description

```
template<typename T1, typename T2, typename R = typename mln::trait::value_< typename
mln::trait::op::times< T1, T2 >::ret >::sum> struct mln::accu::convolve< T1, T2, R >
```

Generic convolution accumulator class. Parameters T1 and T2 are the type of values to be convolved. Parameter R is the result type.

Definition at line 54 of file accu/convolve.hh.

10.2.2 Member Function Documentation

```
10.2.2.1 template<typename T1, typename T2, typename R > void mln::accu::convolve< T1,
T2, R >::init ( ) [inline]
```

Manipulators.

Definition at line 96 of file accu/convolve.hh.

References mln::literal::zero.

10.2.2.2 `template<typename T1 , typename T2 , typename R > bool mln::accu::convolve< T1, T2, R >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 137 of file accu/convolve.hh.

10.2.2.3 `void mln::Accumulator< convolve< T1, T2, R > >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.2.2.4 `void mln::Accumulator< convolve< T1, T2, R > >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.2.2.5 `template<typename T1 , typename T2 , typename R > R mln::accu::convolve< T1, T2, R >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 129 of file accu/convolve.hh.

10.3 `mln::accu::count_adjacent_vertices< F, S > Struct Template Reference`

[Accumulator](#) class counting the number of vertices adjacent to a set of `mln::p_edges_psite` (i.e., a set of edges).

```
#include <count_adjacent_vertices.hh>
```

Inherits base< unsigned, count_adjacent_vertices< F, S > >.

Public Member Functions

- bool [is_valid](#) () const
Return whether this accu can return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- unsigned [to_result](#) () const

Get the value of the accumulator.

- void `init()`
Manipulators.
- void `set_value(unsigned c)`
Force the value of the counter to c.

10.3.1 Detailed Description

`template<typename F, typename S> struct mln::accu::count_adjacent_vertices< F, S >`

Accumulator class counting the number of vertices adjacent to a set of `mln::p_edges_psite` (i.e., a set of edges). The type to be count is `mln::util::pix< pw::image<F, S> >` where `F` and `S` are the parameters of this class.

This accumulator is used by `mln::closing_area_on_vertices` and `mln::opening_area_on_vertices`.

Definition at line 58 of file `accu/count_adjacent_vertices.hh`.

10.3.2 Member Function Documentation

10.3.2.1 `template<typename F, typename S > void mln::accu::count_adjacent_vertices< F, S >::init() [inline]`

Manipulators.

Definition at line 123 of file `accu/count_adjacent_vertices.hh`.

10.3.2.2 `template<typename F, typename S > bool mln::accu::count_adjacent_vertices< F, S >::is_valid() const [inline]`

Return whether this `accu` can return a result.

Definition at line 177 of file `accu/count_adjacent_vertices.hh`.

10.3.2.3 `template<typename F, typename S > void mln::accu::count_adjacent_vertices< F, S >::set_value(unsigned c) [inline]`

Force the value of the counter to `c`.

Definition at line 159 of file `accu/count_adjacent_vertices.hh`.

10.3.2.4 `void mln::Accumulator< count_adjacent_vertices< F, S > >::take_as_init(const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.3.2.5 `void mln::Accumulator< count_adjacent_vertices< F, S >>::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.3.2.6 `template<typename F , typename S > unsigned mln::accu::count_adjacent_vertices< F, S >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 151 of file `accu/count_adjacent_vertices.hh`.

10.4 mln::accu::count_labels< L > Struct Template Reference

Count the number of different labels in an image.

```
#include <count_labels.hh>
```

Inherits `base< unsigned, count_labels< L >>`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `unsigned to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.
- `void set_value (unsigned c)`
Force the value of the counter to c .

10.4.1 Detailed Description

`template<typename L> struct mln::accu::count_labels< L >`

Count the number of different labels in an image. The parameter L is the label type to be count.

Definition at line 52 of file `count_labels.hh`.

10.4.2 Member Function Documentation

10.4.2.1 `template<typename L > void mln::accu::count_labels< L >::init () [inline]`

Manipulators.

Definition at line 113 of file count_labels.hh.

10.4.2.2 `template<typename L > bool mln::accu::count_labels< L >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 163 of file count_labels.hh.

10.4.2.3 `template<typename L > void mln::accu::count_labels< L >::set_value (unsigned c) [inline]`

Force the value of the counter to *c*.

Definition at line 155 of file count_labels.hh.

10.4.2.4 `void mln::Accumulator< count_labels< L > >::take_as_init (const T & t) [inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.4.2.5 `void mln::Accumulator< count_labels< L > >::take_n_times (unsigned n, const T & t) [inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.4.2.6 `template<typename L > unsigned mln::accu::count_labels< L >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 146 of file count_labels.hh.

10.5 mln::accu::count_value< V > Struct Template Reference

Define an accumulator that counts the occurrence of a given value.

```
#include <count_value.hh>
```

Inherits `base< unsigned, count_value< V > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `unsigned to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.
- `void set_value (unsigned c)`
Force the value of the counter to c .

10.5.1 Detailed Description

`template<typename V> struct mln::accu::count_value< V >`

Define an accumulator that counts the occurrence of a given value.

Definition at line 72 of file `count_value.hh`.

10.5.2 Member Function Documentation

10.5.2.1 `template<typename V > void mln::accu::count_value< V >::init () [inline]`

Manipulators.

Definition at line 153 of file `count_value.hh`.

10.5.2.2 `template<typename V > bool mln::accu::count_value< V >::is_valid () const [inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 216 of file `count_value.hh`.

10.5.2.3 `template<typename V > void mln::accu::count_value< V >::set_value (unsigned c) [inline]`

Force the value of the counter to c .

Definition at line 208 of file `count_value.hh`.

10.5.2.4 void mln::Accumulator< count_value< V > >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.5.2.5 void mln::Accumulator< count_value< V > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.5.2.6 template<typename V > unsigned mln::accu::count_value< V >::to_result () const [inline]

Get the value of the accumulator.

Definition at line 199 of file count_value.hh.

10.6 mln::accu::histo< V > Struct Template Reference

Generic histogram class over a value set with type V .

```
#include <histo.hh>
```

Inherits base< const std::vector< unsigned > &, histo< V > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- void [take](#) (const argument &t)
Manipulators.
- const std::vector< unsigned > & [vect](#) () const
Get the value of the accumulator.

10.6.1 Detailed Description

template<typename V> struct mln::accu::histo< V >

Generic histogram class over a value set with type V.

Definition at line 56 of file accu/histo.hh.

10.6.2 Member Function Documentation

10.6.2.1 template<typename V > bool mln::accu::histo< V >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 236 of file accu/histo.hh.

10.6.2.2 template<typename V > void mln::accu::histo< V >::take (const argument & t) [inline]

Manipulators.

Definition at line 129 of file accu/histo.hh.

10.6.2.3 void mln::Accumulator< histo< V > >::take_as_init (const T & t) [inherited]

Take as initialization the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.6.2.4 void mln::Accumulator< histo< V > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.6.2.5 template<typename V > const std::vector< unsigned > & mln::accu::histo< V >::vect () const [inline]

Get the value of the accumulator.

Definition at line 201 of file accu/histo.hh.

10.7 mln::accu::label_used< L > Struct Template Reference

References all the labels used.

```
#include <label_used.hh>
```

Inherits base< const fun::i2v::array< bool > &, label_used< L > >.

Public Member Functions

- void `init` ()
Initialize accumulator attributes.
- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
Take as initialization the value t .
- void `take_n_times` (unsigned n, const T &t)
Take n times the value t .
- const fun::i2v::array< bool > & `to_result` () const
Get the value of the accumulator.
- void `take` (const argument &)
Manipulators.

10.7.1 Detailed Description

`template<typename L> struct mln::accu::label_used< L >`

References all the labels used. The parameter L is the label type.

Definition at line 53 of file label_used.hh.

10.7.2 Member Function Documentation

10.7.2.1 `template<typename L > void mln::accu::label_used< L >::init () [inline]`

Initialize accumulator attributes.

Definition at line 110 of file label_used.hh.

10.7.2.2 `template<typename L > bool mln::accu::label_used< L >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 150 of file label_used.hh.

10.7.2.3 `template<typename L > void mln::accu::label_used< L >::take (const argument & l) [inline]`

Manipulators.

Definition at line 118 of file label_used.hh.

10.7.2.4 `void mln::Accumulator< label_used< L > >::take_as_init (const T & t)` **[inherited]**

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.7.2.5 `void mln::Accumulator< label_used< L > >::take_n_times (unsigned n, const T & t)` **[inherited]**

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.7.2.6 `template<typename L > const fun::i2v::array< bool > & mln::accu::label_used< L >::to_result () const` **[inline]**

Get the value of the accumulator.

Definition at line 142 of file `label_used.hh`.

10.8 `mln::accu::logic::land` Struct Reference

"Logical-and" accumulator.

```
#include <land.hh>
```

Inherits base< bool, land >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- bool [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.8.1 Detailed Description

"Logical-and" accumulator.

Definition at line 96 of file accu/logic/land.hh.

10.8.2 Member Function Documentation

10.8.2.1 void mln::accu::logic::land::init () [inline]

Manipulators.

Definition at line 136 of file accu/logic/land.hh.

10.8.2.2 bool mln::accu::logic::land::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 185 of file accu/logic/land.hh.

10.8.2.3 void mln::Accumulator< land >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.8.2.4 void mln::Accumulator< land >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.8.2.5 bool mln::accu::logic::land::to_result () const [inline]

Get the value of the accumulator.

Definition at line 178 of file accu/logic/land.hh.

10.9 mln::accu::logic::land_basic Struct Reference

"Logical-and" accumulator.

```
#include <land_basic.hh>
```

Inherits base< bool, land_basic >.

Public Member Functions

- bool [can_stop](#) () const

Test if it is worth for this accumulator to take extra data.

- bool `is_valid()` const

Check whether this `accu` is able to return a result.

- void `take_as_init(const T &t)`

Take as initialization the value `t`.

- void `take_n_times(unsigned n, const T &t)`

Take `n` times the value `t`.

- bool `to_result()` const

Get the value of the accumulator.

- void `init()`

Manipulators.

10.9.1 Detailed Description

"Logical-and" accumulator. Conversely to `accu::logic::land`, this version does not have the 'untake' method but features the 'can_stop' method.

Definition at line 99 of file `land_basic.hh`.

10.9.2 Member Function Documentation

10.9.2.1 `bool mln::accu::logic::land_basic::can_stop() const [inline]`

Test if it is worth for this accumulator to take extra data.

If the result is already 'false' (because this accumulator has already taken a 'false' value), `can_stop` returns true.

Definition at line 181 of file `land_basic.hh`.

10.9.2.2 `void mln::accu::logic::land_basic::init() [inline]`

Manipulators.

Definition at line 140 of file `land_basic.hh`.

10.9.2.3 `bool mln::accu::logic::land_basic::is_valid() const [inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 174 of file `land_basic.hh`.

10.9.2.4 void mln::Accumulator< land_basic >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.9.2.5 void mln::Accumulator< land_basic >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.9.2.6 bool mln::accu::logic::land_basic::to_result () const [inline]

Get the value of the accumulator.

Definition at line 167 of file `land_basic.hh`.

10.10 mln::accu::logic::lor Struct Reference

"Logical-or" accumulator.

```
#include <lor.hh>
```

Inherits base< bool, lor >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- bool [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.10.1 Detailed Description

"Logical-or" accumulator.

Definition at line 95 of file `accu/logic/lor.hh`.

10.10.2 Member Function Documentation

10.10.2.1 void mln::accu::logic::lor::init () [inline]

Manipulators.

Definition at line 134 of file accu/logic/lor.hh.

10.10.2.2 bool mln::accu::logic::lor::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 183 of file accu/logic/lor.hh.

10.10.2.3 void mln::Accumulator< lor >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.10.2.4 void mln::Accumulator< lor >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.10.2.5 bool mln::accu::logic::lor::to_result () const [inline]

Get the value of the accumulator.

Definition at line 176 of file accu/logic/lor.hh.

10.11 mln::accu::logic::lor_basic Struct Reference

"Logical-or" accumulator class.

```
#include <lor_basic.hh>
```

Inherits base< bool, lor_basic >.

Public Member Functions

- bool [can_stop](#) () const
Test if it is worth for this accumulator to take extra data.
- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)

Take as initialization the value t .

- void [take_n_times](#) (unsigned n, const T &t)

Take n times the value t .

- bool [to_result](#) () const

Get the value of the accumulator.

- void [init](#) ()

Manipulators.

10.11.1 Detailed Description

"Logical-or" accumulator class. Conversely to [accu::logic::lor](#), this version does not have the 'untake' method but features the 'can_stop' method.

Definition at line 98 of file lor_basic.hh.

10.11.2 Member Function Documentation

10.11.2.1 bool mln::accu::logic::lor_basic::can_stop () const `[inline]`

Test if it is worth for this accumulator to take extra data.

If the result is already 'true' (because this accumulator has already taken a 'true' value), can_stop returns true.

Definition at line 180 of file lor_basic.hh.

10.11.2.2 void mln::accu::logic::lor_basic::init () `[inline]`

Manipulators.

Definition at line 139 of file lor_basic.hh.

10.11.2.3 bool mln::accu::logic::lor_basic::is_valid () const `[inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 173 of file lor_basic.hh.

10.11.2.4 void mln::Accumulator<lor_basic>::take_as_init (const T & t) `[inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.11.2.5 void mln::Accumulator< lor_basic >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.11.2.6 bool mln::accu::logic::lor_basic::to_result () const [inline]

Get the value of the accumulator.

Definition at line 166 of file lor_basic.hh.

10.12 mln::accu::maj_h< T > Struct Template Reference

Compute the majority value.

```
#include <maj_h.hh>
```

Inherits base< const T &, maj_h< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- const T & [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.12.1 Detailed Description

```
template<typename T> struct mln::accu::maj_h< T >
```

Compute the majority value. It is based on a histogram. The parameter T is the type of values.

Definition at line 57 of file maj_h.hh.

10.12.2 Member Function Documentation

10.12.2.1 `template<typename T> void mln::accu::maj_h< T >::init () [inline]`

Manipulators.

Definition at line 129 of file maj_h.hh.

10.12.2.2 `template<typename T> bool mln::accu::maj_h< T >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 197 of file maj_h.hh.

10.12.2.3 `void mln::Accumulator< maj_h< T > >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.12.2.4 `void mln::Accumulator< maj_h< T > >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.12.2.5 `template<typename T> const T & mln::accu::maj_h< T >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 187 of file maj_h.hh.

10.13 mln::accu::math::count< T > Struct Template Reference

Generic counter accumulator.

```
#include <count.hh>
```

Inherits `base< unsigned, count< T > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .

- void `take_n_times` (unsigned n, const T &t)
Take n times the value t.
- unsigned `to_result` () const
Get the value of the accumulator.
- void `init` ()
Manipulators.
- void `set_value` (unsigned c)
Force the value of the counter to c.

10.13.1 Detailed Description

template<typename T> struct mln::accu::math::count< T >

Generic counter accumulator. The parameter *T* is the type to be count.

Definition at line 100 of file count.hh.

10.13.2 Member Function Documentation

10.13.2.1 template<typename T > void mln::accu::math::count< T >::init () [inline]

Manipulators.

Definition at line 145 of file count.hh.

10.13.2.2 template<typename T > bool mln::accu::math::count< T >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 203 of file count.hh.

10.13.2.3 template<typename T > void mln::accu::math::count< T >::set_value (unsigned c) [inline]

Force the value of the counter to *c*.

Definition at line 195 of file count.hh.

10.13.2.4 void mln::Accumulator< count< T > >::take_as_init (const T & t) [inherited]

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.13.2.5 `void mln::Accumulator< count< T > >::take_n_times (unsigned n, const T & t)`
[inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.13.2.6 `template<typename T > unsigned mln::accu::math::count< T >::to_result () const`
[inline]

Get the value of the accumulator.

Definition at line 187 of file `count.hh`.

10.14 mln::accu::math::inf< T > Struct Template Reference

Generic inf accumulator class.

```
#include <inf.hh>
```

Inherits `base< const T &, inf< T > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `const T & to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.14.1 Detailed Description

```
template<typename T> struct mln::accu::math::inf< T >
```

Generic inf accumulator class. The parameter `T` is the type of values.

Definition at line 56 of file `accu/math/inf.hh`.

10.14.2 Member Function Documentation

10.14.2.1 `template<typename T> void mln::accu::math::inf< T >::init () [inline]`

Manipulators.

Definition at line 126 of file `accu/math/inf.hh`.

10.14.2.2 `template<typename T> bool mln::accu::math::inf< T >::is_valid () const [inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 164 of file `accu/math/inf.hh`.

10.14.2.3 `void mln::Accumulator< inf< T > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.14.2.4 `void mln::Accumulator< inf< T > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.14.2.5 `template<typename T> const T & mln::accu::math::inf< T >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 156 of file `accu/math/inf.hh`.

10.15 `mln::accu::math::sum< T, S >` Struct Template Reference

Generic sum accumulator class.

```
#include <sum.hh>
```

Inherits `base< const S &, sum< T, S > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value `t`.

- void `take_n_times` (unsigned `n`, const `T` &`t`)
Take `n` times the value `t`.
- const `S` & `to_result` () const
Get the value of the accumulator.
- void `init` ()
Manipulators.

10.15.1 Detailed Description

template<typename T, typename S = typename mln::value::props< T >::sum> struct mln::accu::math::sum< T, S >

Generic sum accumulator class. Parameter `T` is the type of values that we sum. Parameter `S` is the type to store the value sum; the default type of `S` is the summation type (property) of `T`.

Definition at line 112 of file `accu/math/sum.hh`.

10.15.2 Member Function Documentation

10.15.2.1 template<typename T, typename S > void mln::accu::math::sum< T, S >::init ()
[inline]

Manipulators.

Definition at line 161 of file `accu/math/sum.hh`.

References `mln::literal::zero`.

10.15.2.2 template<typename T, typename S > bool mln::accu::math::sum< T, S >::is_valid ()
const [inline]

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 222 of file `accu/math/sum.hh`.

10.15.2.3 void mln::Accumulator< sum< T, S > >::take_as_init (const T & t) [inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.15.2.4 void mln::Accumulator< sum< T, S > >::take_n_times (unsigned n, const T & t)
[inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.15.2.5 `template<typename T , typename S > const S & mln::accu::math::sum< T, S >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 206 of file `accu/math/sum.hh`.

10.16 `mln::accu::math::sup< T >` Struct Template Reference

Generic sup accumulator class.

```
#include <sup.hh>
```

Inherits `base< const T &, sup< T > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `const T & to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.16.1 Detailed Description

`template<typename T> struct mln::accu::math::sup< T >`

Generic sup accumulator class. The parameter `T` is the type of values.

Definition at line 56 of file `accu/math/sup.hh`.

10.16.2 Member Function Documentation

10.16.2.1 `template<typename T > void mln::accu::math::sup< T >::init () [inline]`

Manipulators.

Definition at line 128 of file `accu/math/sup.hh`.

10.16.2.2 `template<typename T> bool mln::accu::math::sup< T >::is_valid () const` `[inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 166 of file accu/math/sup.hh.

10.16.2.3 `void mln::Accumulator< sup< T > >::take_as_init (const T & t)` `[inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.16.2.4 `void mln::Accumulator< sup< T > >::take_n_times (unsigned n, const T & t)` `[inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.16.2.5 `template<typename T> const T & mln::accu::math::sup< T >::to_result () const` `[inline]`

Get the value of the accumulator.

Definition at line 158 of file accu/math/sup.hh.

10.17 mln::accu::max_site< I > Struct Template Reference

Define an accumulator that computes the first site with the maximum value in an image.

```
#include <max_site.hh>
```

Inherits `base< I::psite, max_site< I > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `I::psite to_result () const`
Get the value of the accumulator.

- void `init()`
Manipulators.

10.17.1 Detailed Description

template<typename I> struct mln::accu::max_site< I >

Define an accumulator that computes the first site with the maximum value in an image.

Definition at line 53 of file max_site.hh.

10.17.2 Member Function Documentation

10.17.2.1 template<typename I > void mln::accu::max_site< I >::init () [inline]

Manipulators.

Definition at line 114 of file max_site.hh.

10.17.2.2 template<typename I > bool mln::accu::max_site< I >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 174 of file max_site.hh.

10.17.2.3 void mln::Accumulator< max_site< I > >::take_as_init (const T & t) [inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.17.2.4 void mln::Accumulator< max_site< I > >::take_n_times (unsigned n, const T & t) [inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.17.2.5 template<typename I > I::psite mln::accu::max_site< I >::to_result () const [inline]

Get the value of the accumulator.

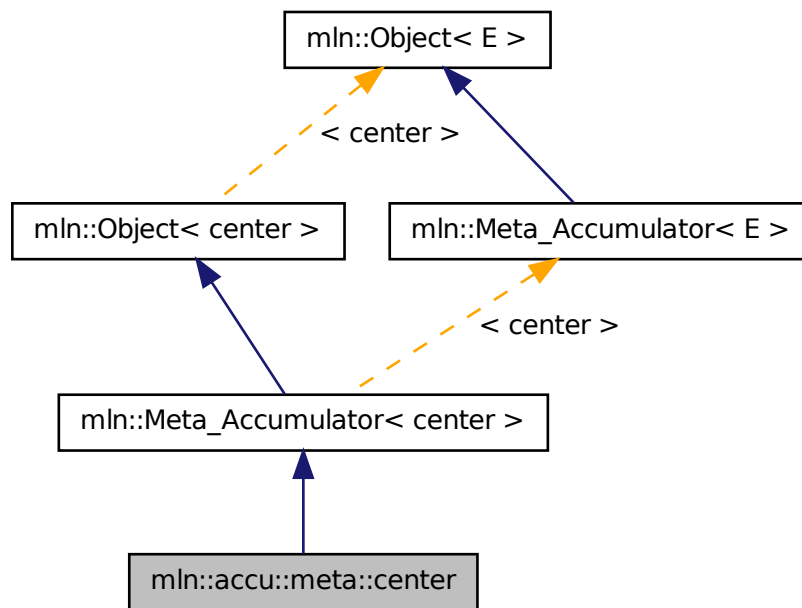
Definition at line 150 of file max_site.hh.

10.18 mln::accu::meta::center Struct Reference

Meta accumulator for center.

```
#include <center.hh>
```

Inheritance diagram for mln::accu::meta::center:



10.18.1 Detailed Description

Meta accumulator for center.

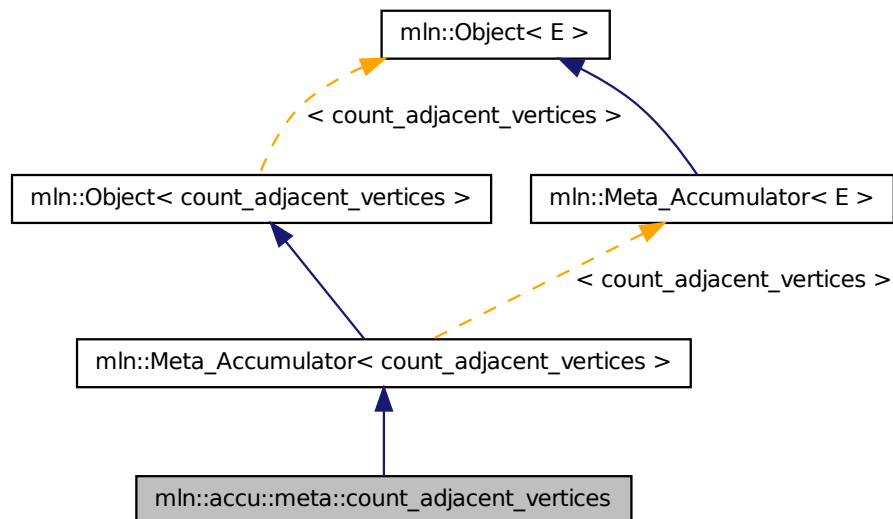
Definition at line 89 of file center.hh.

10.19 mln::accu::meta::count_adjacent_vertices Struct Reference

Meta accumulator for [count_adjacent_vertices](#).

```
#include <count_adjacent_vertices.hh>
```

Inheritance diagram for mln::accu::meta::count_adjacent_vertices:



10.19.1 Detailed Description

Meta accumulator for [count_adjacent_vertices](#).

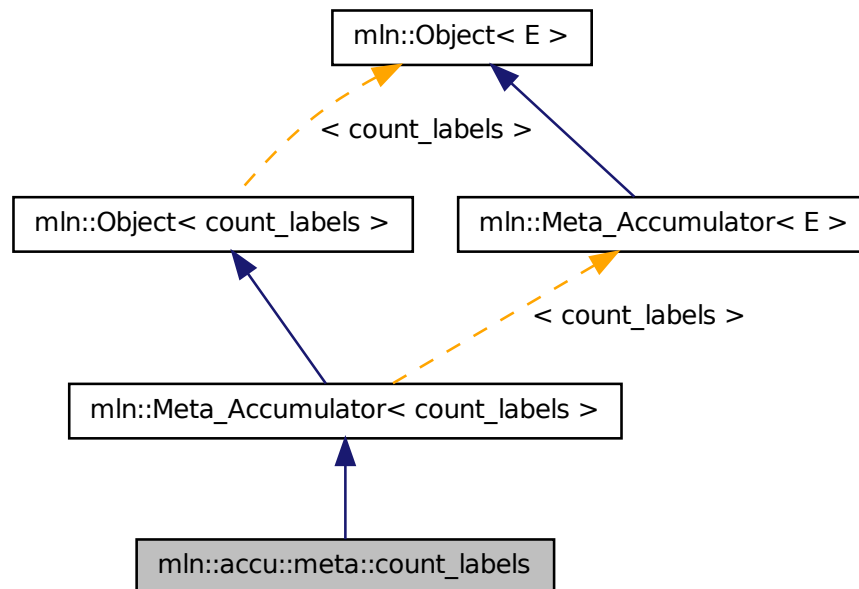
Definition at line 98 of file `accu/count_adjacent_vertices.hh`.

10.20 mln::accu::meta::count_labels Struct Reference

Meta accumulator for [count_labels](#).

```
#include <count_labels.hh>
```

Inheritance diagram for mln::accu::meta::count_labels:



10.20.1 Detailed Description

Meta accumulator for [count_labels](#).

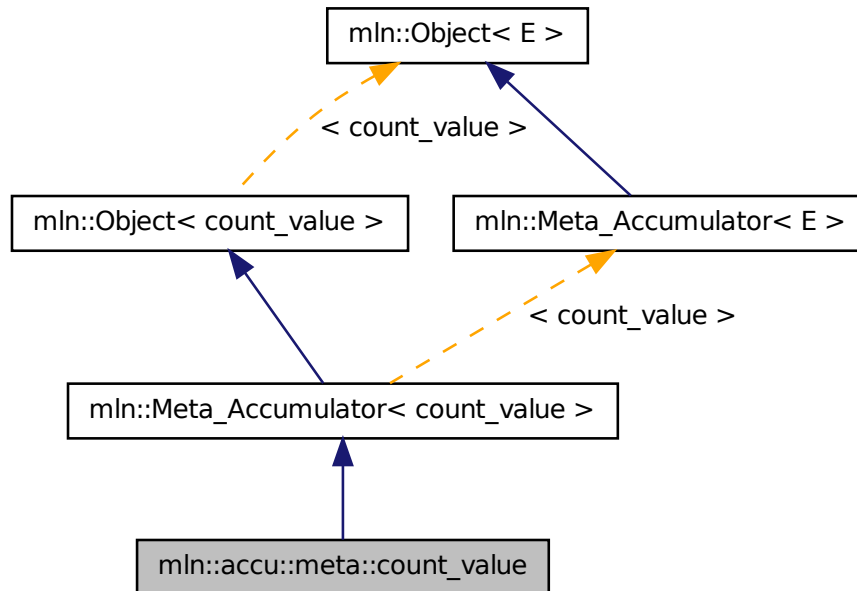
Definition at line 88 of file count_labels.hh.

10.21 mln::accu::meta::count_value Struct Reference

FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for [count_value](#).

```
#include <count_value.hh>
```

Inheritance diagram for `mln::accu::meta::count_value`:



10.21.1 Detailed Description

FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for `count_value`.

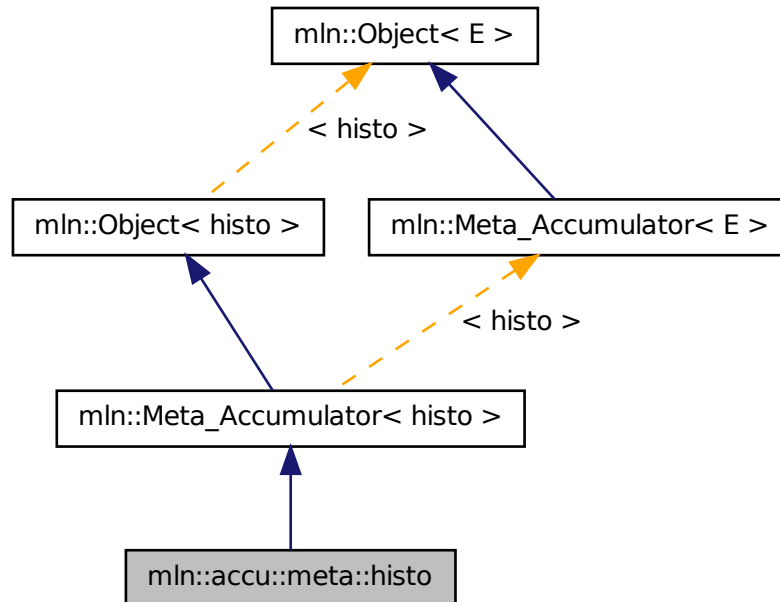
Definition at line 116 of file `count_value.hh`.

10.22 mln::accu::meta::histo Struct Reference

Meta accumulator for histo.

```
#include <histo.hh>
```

Inheritance diagram for mln::accu::meta::histo:



10.22.1 Detailed Description

Meta accumulator for histo.

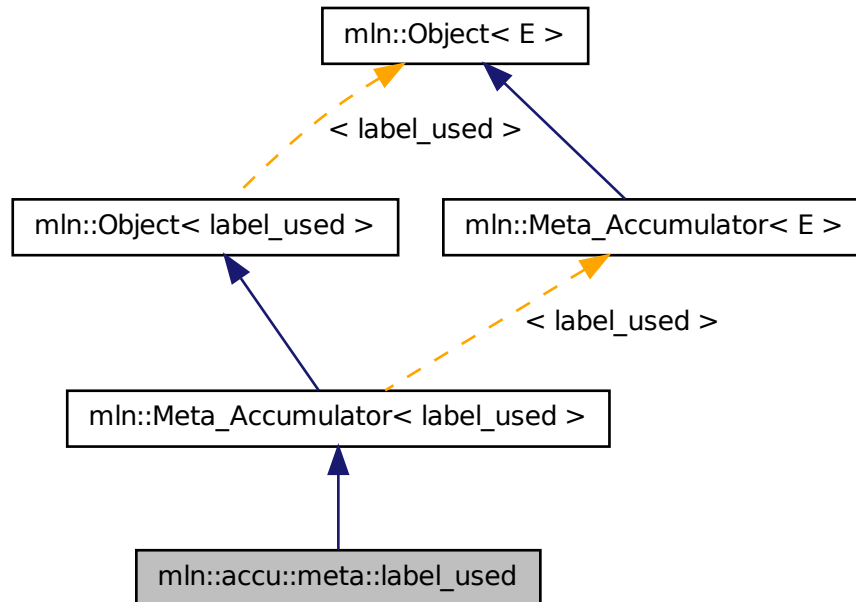
Definition at line 102 of file `accu/histo.hh`.

10.23 mln::accu::meta::label_used Struct Reference

Meta accumulator for [label_used](#).

```
#include <label_used.hh>
```

Inheritance diagram for `mln::accu::meta::label_used`:



10.23.1 Detailed Description

Meta accumulator for [label_used](#).

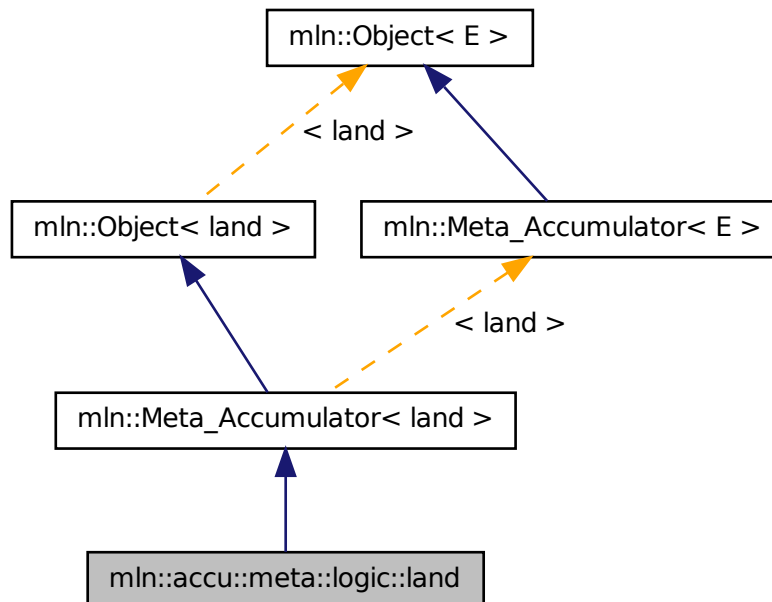
Definition at line 85 of file `label_used.hh`.

10.24 `mln::accu::meta::logic::land` Struct Reference

Meta accumulator for `land`.

```
#include <land.hh>
```

Inheritance diagram for mln::accu::meta::logic::land:



10.24.1 Detailed Description

Meta accumulator for land.

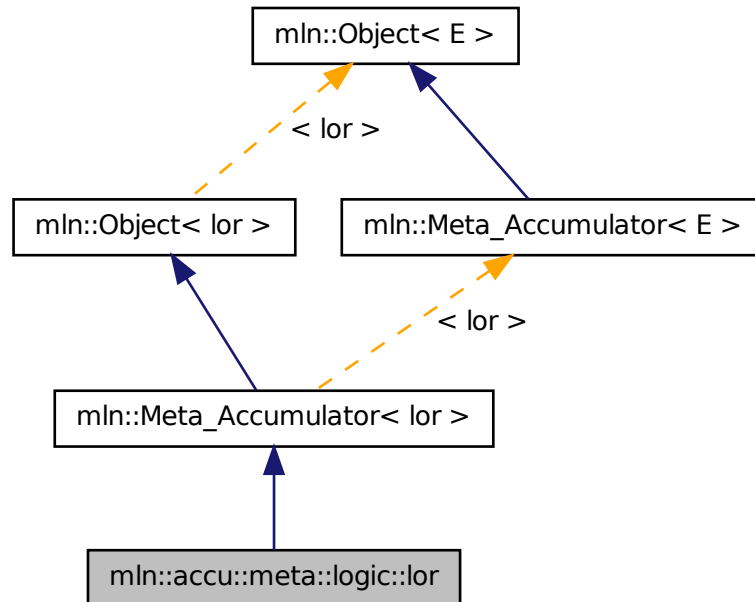
Definition at line 76 of file `accu/logic/land.hh`.

10.25 mln::accu::meta::logic::lor Struct Reference

Meta accumulator for lor.

```
#include <lor.hh>
```

Inheritance diagram for `mln::accu::meta::logic::lor`:



10.25.1 Detailed Description

Meta accumulator for `lor`.

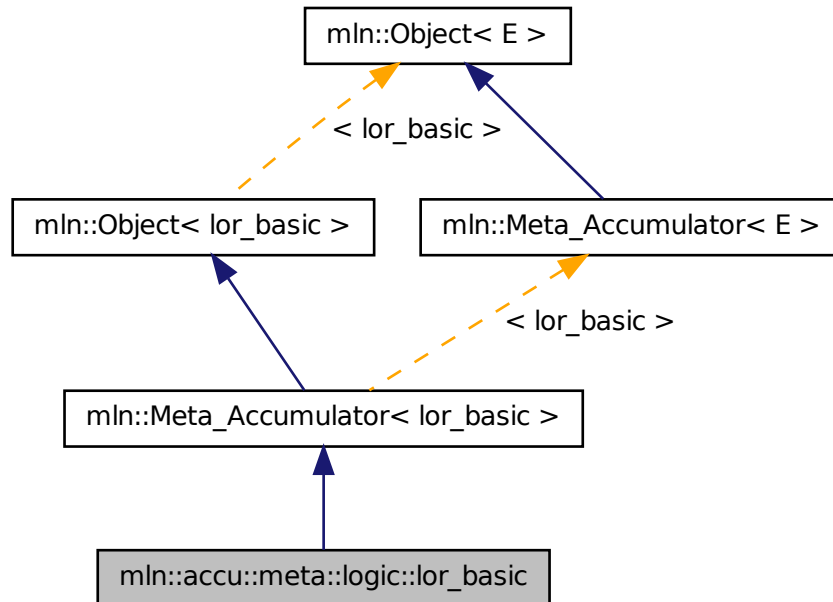
Definition at line 76 of file `accu/logic/lor.hh`.

10.26 `mln::accu::meta::logic::lor_basic` Struct Reference

Meta accumulator for `lor_basic`.

```
#include <lor_basic.hh>
```


Inheritance diagram for mln::accu::meta::logic::lor_basic:



10.26.1 Detailed Description

Meta accumulator for [lor_basic](#).

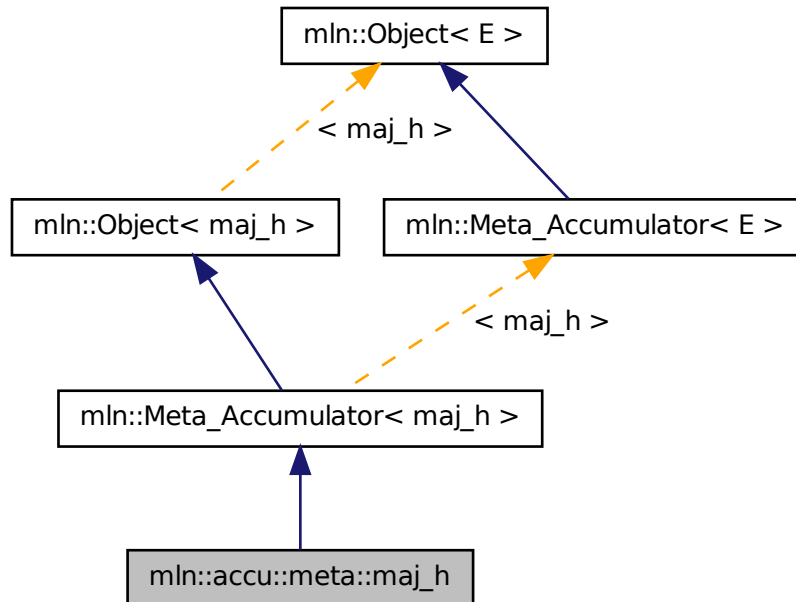
Definition at line 76 of file `lor_basic.hh`.

10.27 mln::accu::meta::maj_h Struct Reference

Meta accumulator for [maj_h](#).

```
#include <maj_h.hh>
```

Inheritance diagram for `mln::accu::meta::maj_h`:



10.27.1 Detailed Description

Meta accumulator for [maj_h](#).

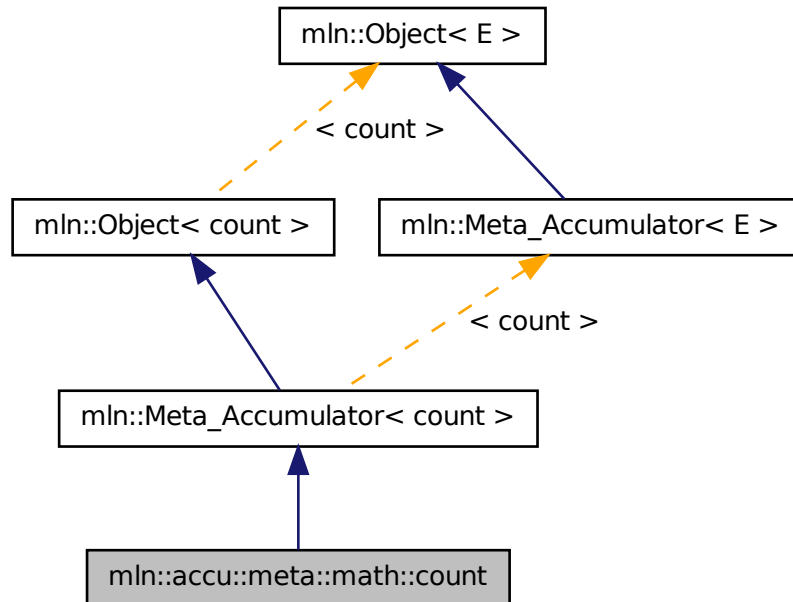
Definition at line 101 of file `maj_h.hh`.

10.28 mln::accu::meta::math::count Struct Reference

Meta accumulator for count.

```
#include <count.hh>
```

Inheritance diagram for mln::accu::meta::math::count:



10.28.1 Detailed Description

Meta accumulator for count.

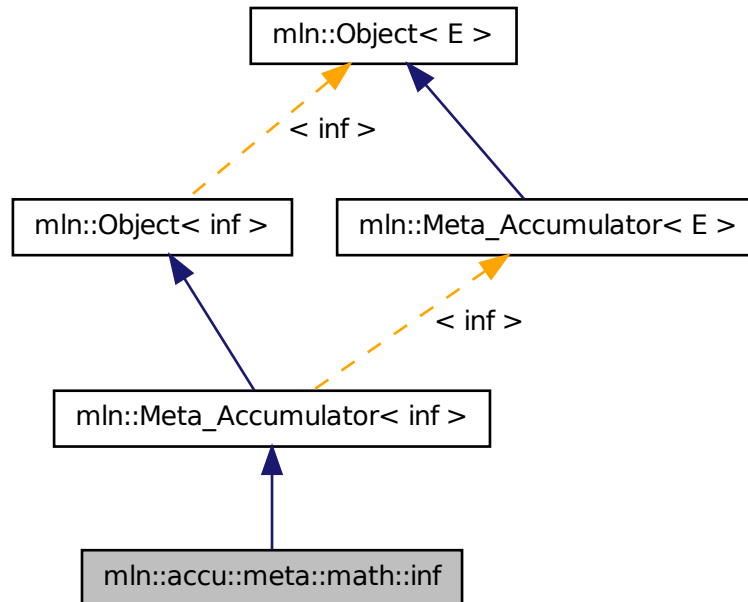
Definition at line 77 of file count.hh.

10.29 mln::accu::meta::math::inf Struct Reference

Meta accumulator for inf.

```
#include <inf.hh>
```

Inheritance diagram for `mln::accu::meta::math::inf`:



10.29.1 Detailed Description

Meta accumulator for `inf`.

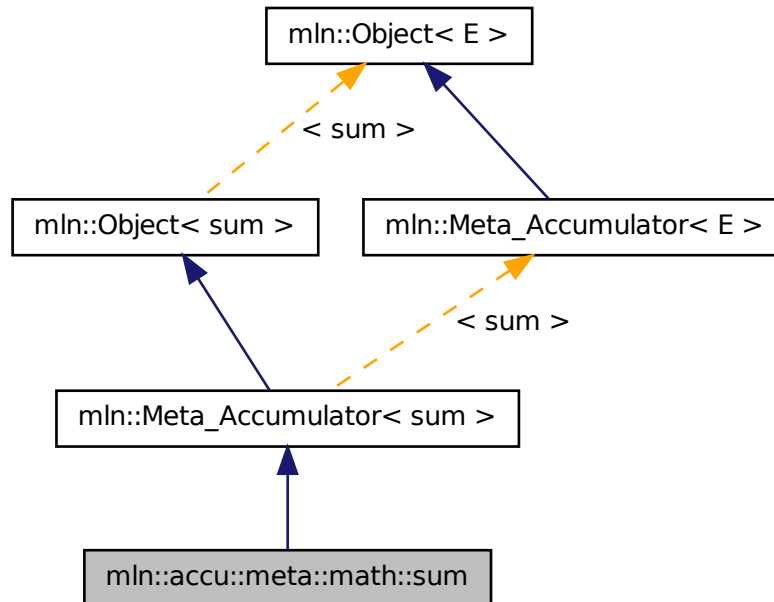
Definition at line 97 of file `accu/math/inf.hh`.

10.30 `mln::accu::meta::math::sum` Struct Reference

Meta accumulator for `sum`.

```
#include <sum.hh>
```

Inheritance diagram for mln::accu::meta::math::sum:



10.30.1 Detailed Description

Meta accumulator for sum.

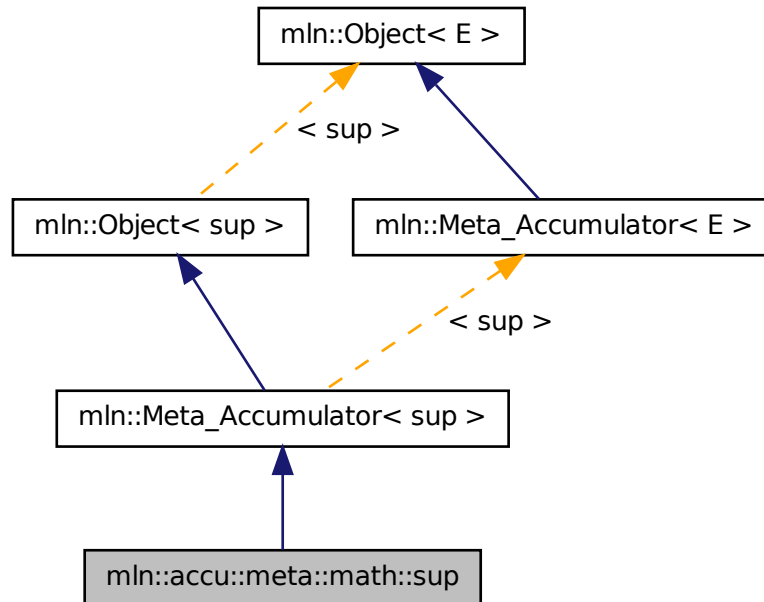
Definition at line 66 of file accu/math/sum.hh.

10.31 mln::accu::meta::math::sup Struct Reference

Meta accumulator for sup.

```
#include <sup.hh>
```

Inheritance diagram for `mln::accu::meta::math::sup`:



10.31.1 Detailed Description

Meta accumulator for `sup`.

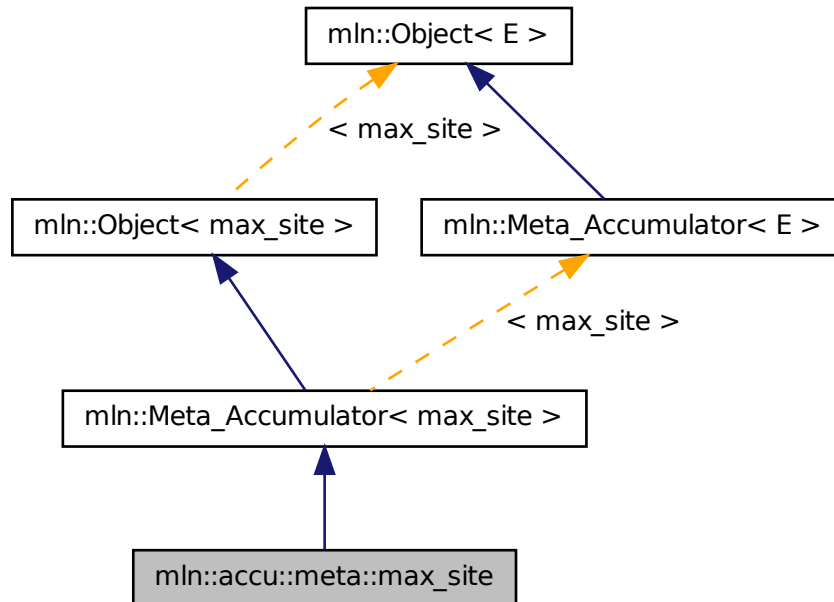
Definition at line 98 of file `accu/math/sup.hh`.

10.32 `mln::accu::meta::max_site` Struct Reference

Meta accumulator for `max_site`.

```
#include <max_site.hh>
```

Inheritance diagram for mln::accu::meta::max_site:



10.32.1 Detailed Description

Meta accumulator for [max_site](#).

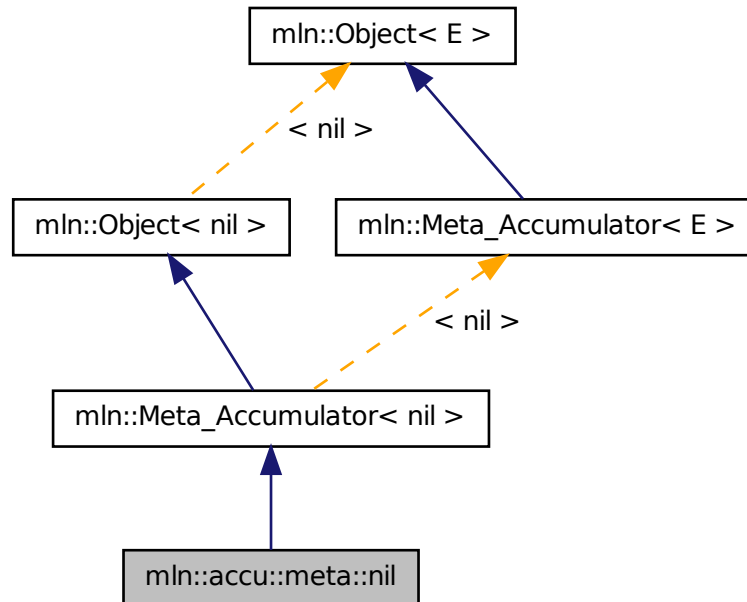
Definition at line 90 of file `max_site.hh`.

10.33 mln::accu::meta::nil Struct Reference

Meta accumulator for nil.

```
#include <nil.hh>
```

Inheritance diagram for mln::accu::meta::nil:



10.33.1 Detailed Description

Meta accumulator for nil.

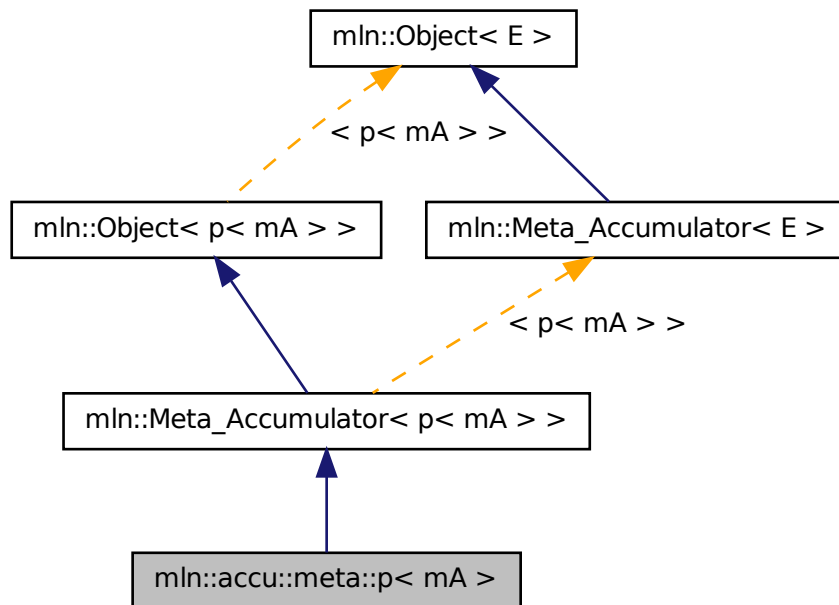
Definition at line 77 of file accu/nil.hh.

10.34 mln::accu::meta::p< mA > Struct Template Reference

Meta accumulator for p.

```
#include <p.hh>
```


Inheritance diagram for mln::accu::meta::p< mA >:



10.34.1 Detailed Description

```
template<typename mA> struct mln::accu::meta::p< mA >
```

Meta accumulator for p.

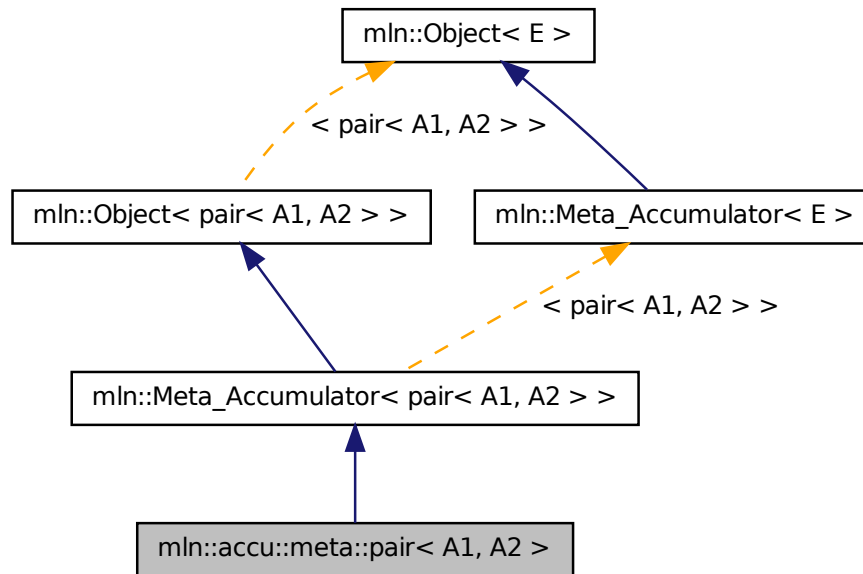
Definition at line 83 of file p.hh.

10.35 mln::accu::meta::pair< A1, A2 > Struct Template Reference

Meta accumulator for pair.

```
#include <pair.hh>
```

Inheritance diagram for `mln::accu::meta::pair< A1, A2 >`:



10.35.1 Detailed Description

```
template<typename A1, typename A2> struct mln::accu::meta::pair< A1, A2 >
```

Meta accumulator for pair.

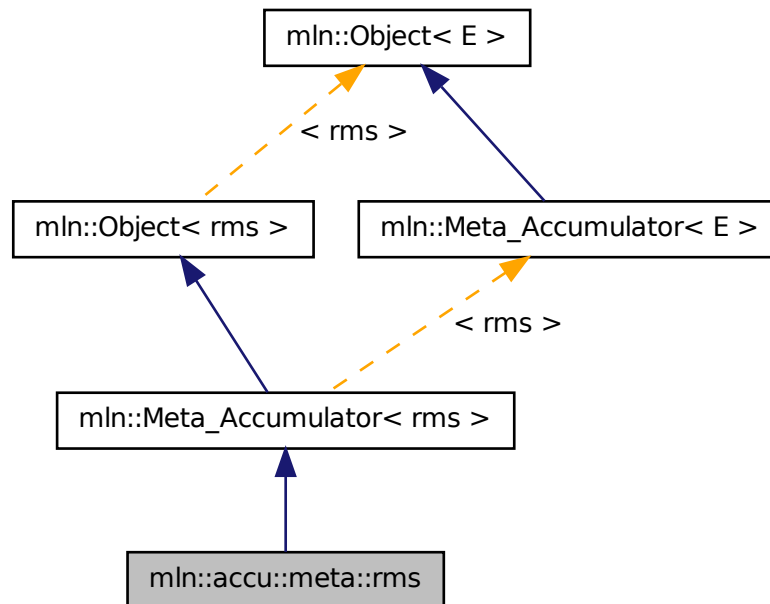
Definition at line 110 of file pair.hh.

10.36 mln::accu::meta::rms Struct Reference

Meta accumulator for rms.

```
#include <rms.hh>
```

Inheritance diagram for mln::accu::meta::rms:



10.36.1 Detailed Description

Meta accumulator for rms.

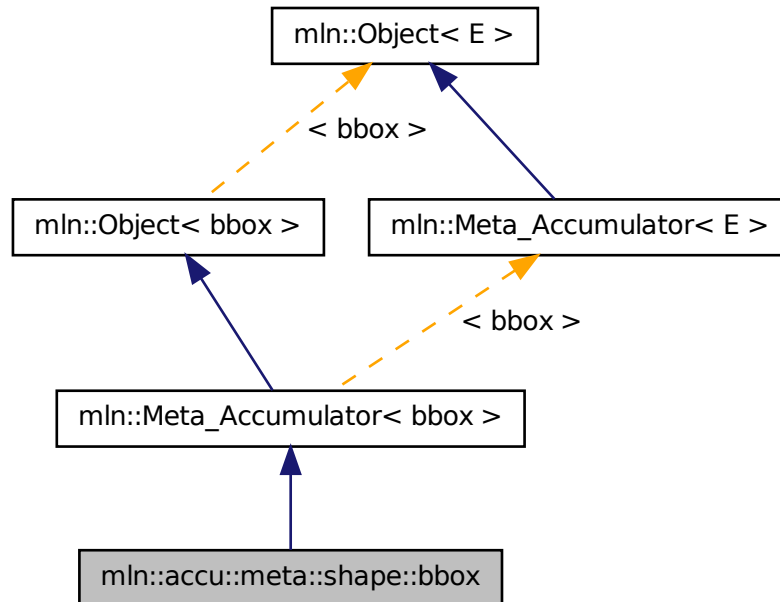
Definition at line 88 of file `accu/rms.hh`.

10.37 mln::accu::meta::shape::bbox Struct Reference

Meta accumulator for bbox.

```
#include <bbox.hh>
```

Inheritance diagram for mln::accu::meta::shape::bbox:



10.37.1 Detailed Description

Meta accumulator for bbox.

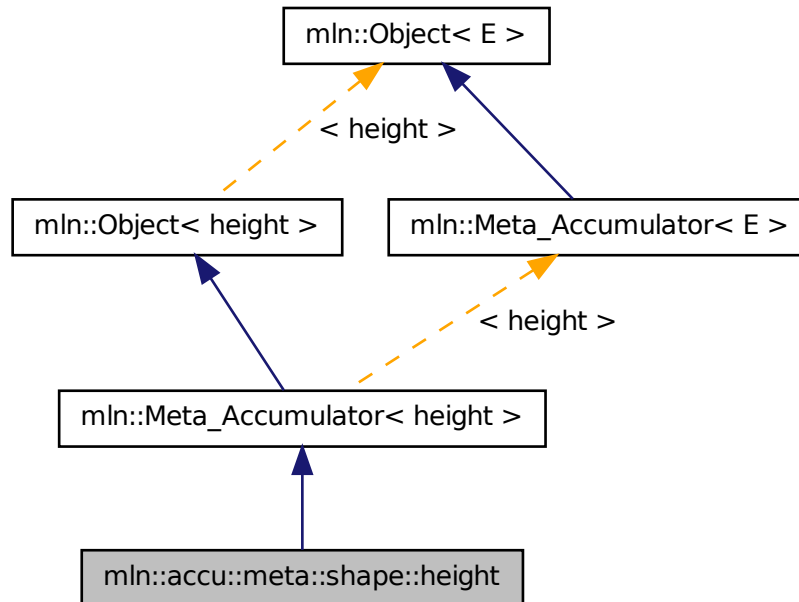
Definition at line 94 of file accu/shape/bbox.hh.

10.38 mln::accu::meta::shape::height Struct Reference

Meta accumulator for height.

```
#include <height.hh>
```

Inheritance diagram for mln::accu::meta::shape::height:



10.38.1 Detailed Description

Meta accumulator for height.

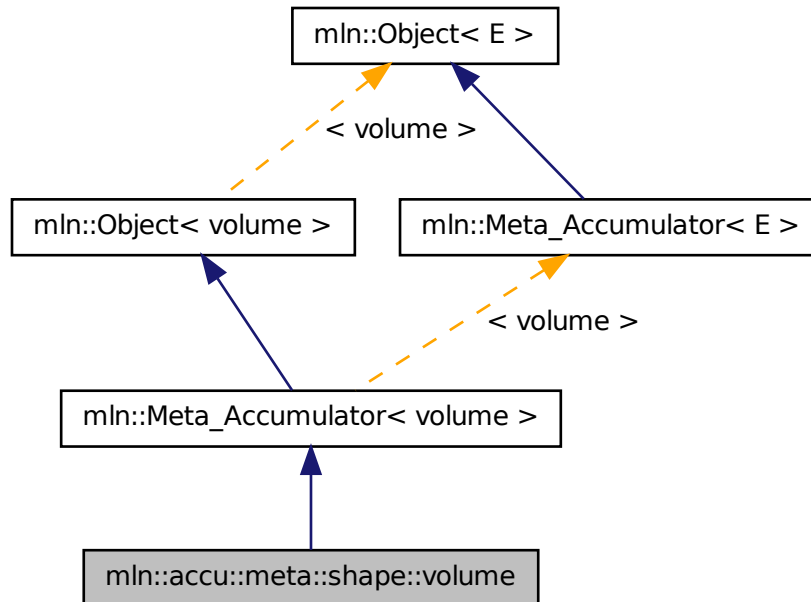
Definition at line 121 of file `accu/shape/height.hh`.

10.39 mln::accu::meta::shape::volume Struct Reference

Meta accumulator for volume.

```
#include <volume.hh>
```

Inheritance diagram for `mln::accu::meta::shape::volume`:



10.39.1 Detailed Description

Meta accumulator for volume.

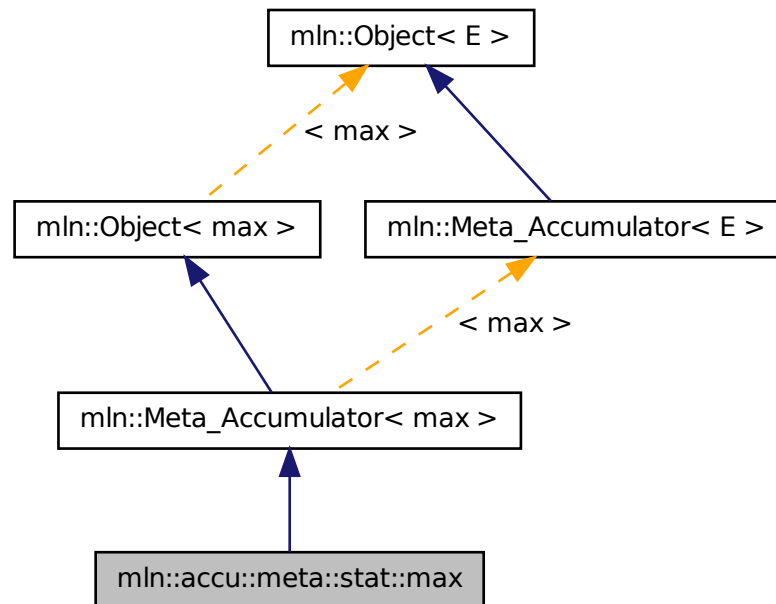
Definition at line 120 of file `accu/shape/volume.hh`.

10.40 `mln::accu::meta::stat::max` Struct Reference

Meta accumulator for max.

```
#include <max.hh>
```

Inheritance diagram for mln::accu::meta::stat::max:



10.40.1 Detailed Description

Meta accumulator for max.

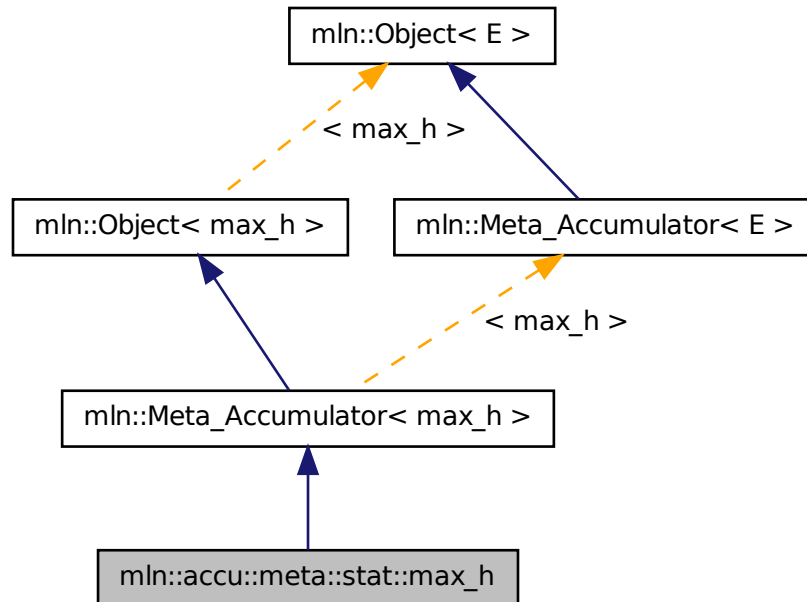
Definition at line 78 of file `accu/stat/max.hh`.

10.41 mln::accu::meta::stat::max_h Struct Reference

Meta accumulator for max.

```
#include <max_h.hh>
```

Inheritance diagram for mln::accu::meta::stat::max_h:



10.41.1 Detailed Description

Meta accumulator for max.

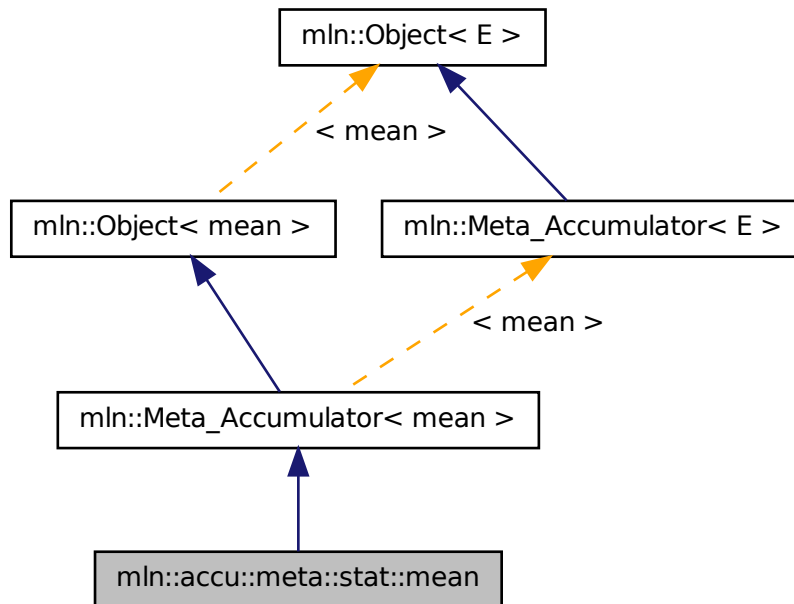
Definition at line 77 of file max_h.hh.

10.42 mln::accu::meta::stat::mean Struct Reference

Meta accumulator for mean.

```
#include <mean.hh>
```


Inheritance diagram for mln::accu::meta::stat::mean:



10.42.1 Detailed Description

Meta accumulator for mean.

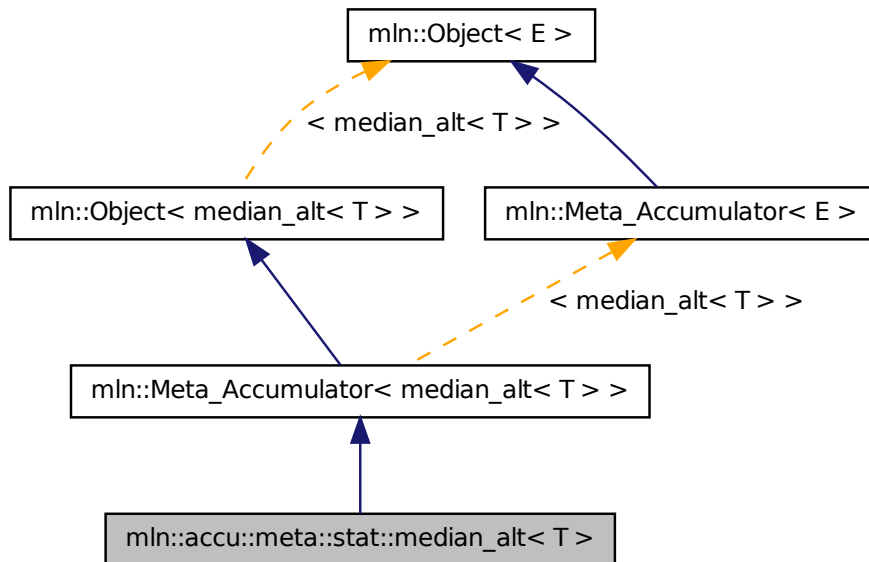
Definition at line 65 of file accu/stat/mean.hh.

10.43 mln::accu::meta::stat::median_alt< T > Struct Template Reference

Meta accumulator for [median_alt](#).

```
#include <median_alt.hh>
```

Inheritance diagram for `mln::accu::meta::stat::median_alt< T >`:



10.43.1 Detailed Description

```
template<typename T> struct mln::accu::meta::stat::median_alt< T >
```

Meta accumulator for [median_alt](#).

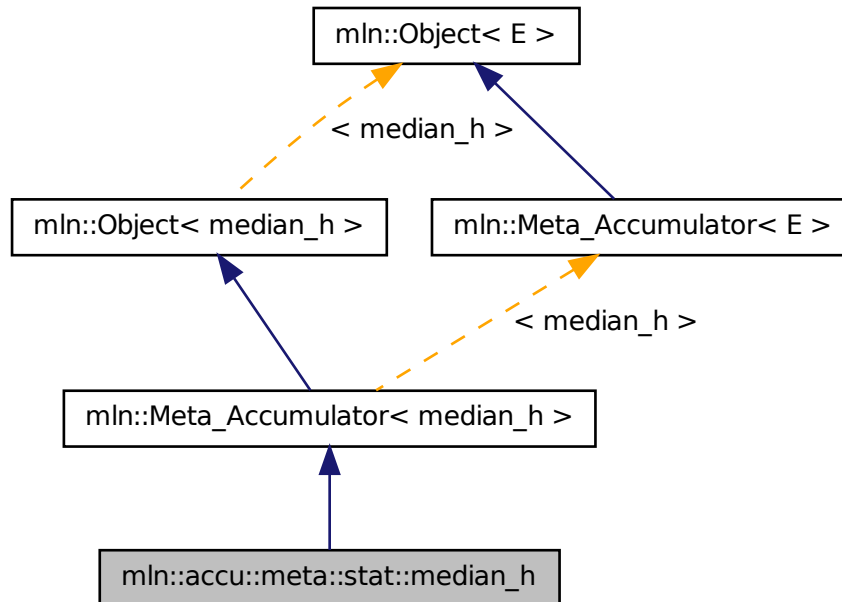
Definition at line 122 of file `median_alt.hh`.

10.44 mln::accu::meta::stat::median_h Struct Reference

Meta accumulator for [median_h](#).

```
#include <median_h.hh>
```

Inheritance diagram for mln::accu::meta::stat::median_h:



10.44.1 Detailed Description

Meta accumulator for [median_h](#).

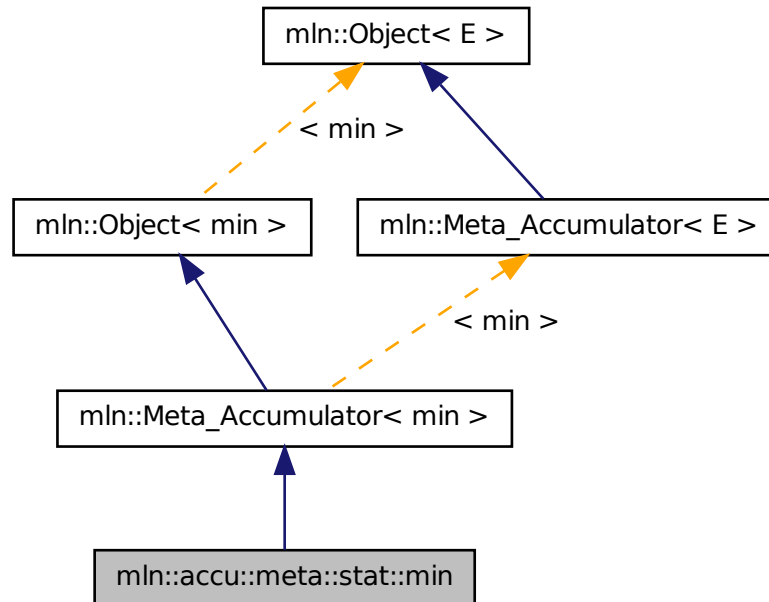
Definition at line 61 of file `median_h.hh`.

10.45 mln::accu::meta::stat::min Struct Reference

Meta accumulator for `min`.

```
#include <min.hh>
```

Inheritance diagram for mln::accu::meta::stat::min:



10.45.1 Detailed Description

Meta accumulator for min.

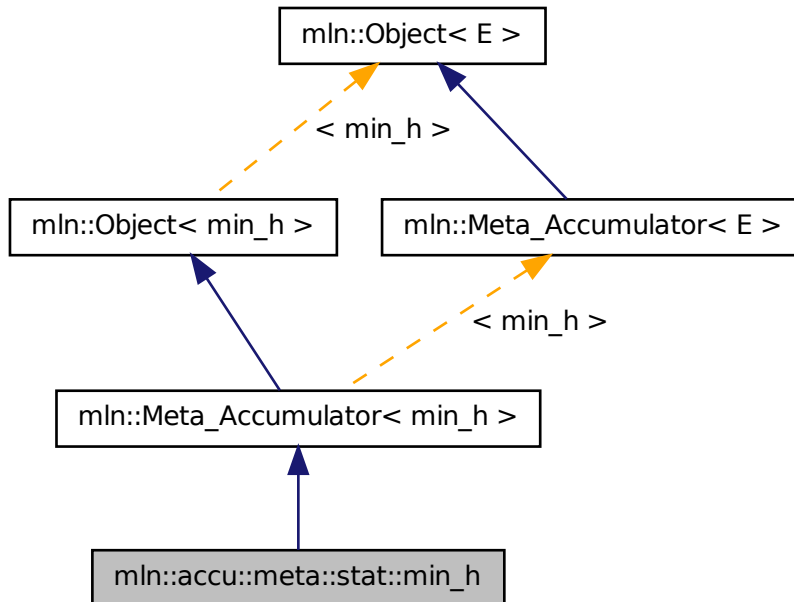
Definition at line 78 of file accu/stat/min.hh.

10.46 mln::accu::meta::stat::min_h Struct Reference

Meta accumulator for min.

```
#include <min_h.hh>
```

Inheritance diagram for mln::accu::meta::stat::min_h:



10.46.1 Detailed Description

Meta accumulator for min.

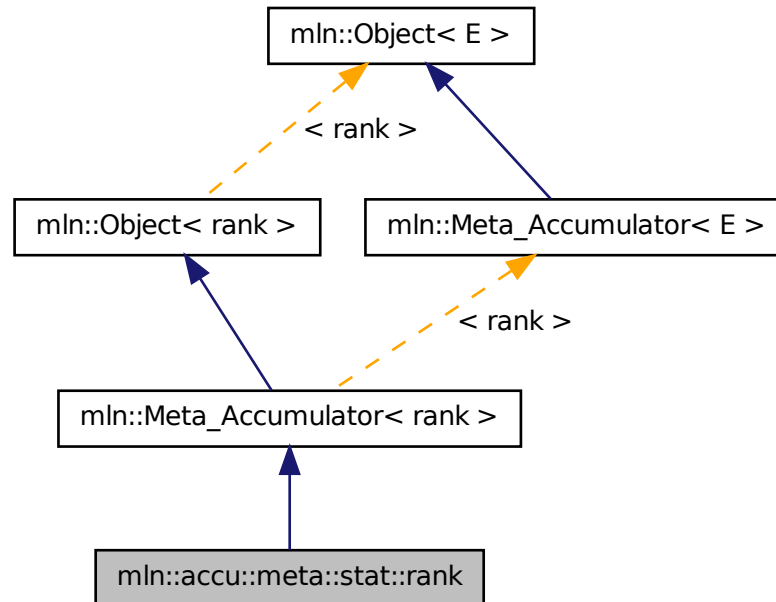
Definition at line 77 of file `min_h.hh`.

10.47 mln::accu::meta::stat::rank Struct Reference

Meta accumulator for rank.

```
#include <rank.hh>
```

Inheritance diagram for `mln::accu::meta::stat::rank`:



10.47.1 Detailed Description

Meta accumulator for rank.

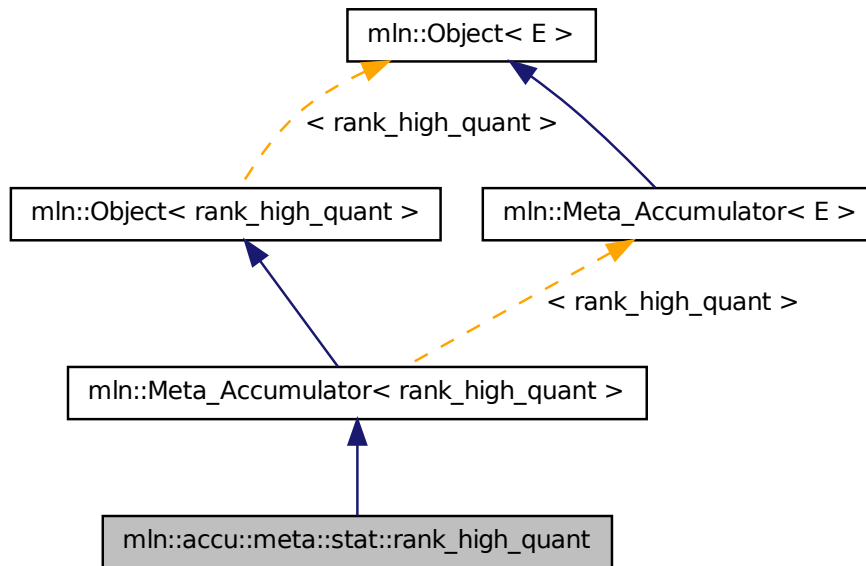
Definition at line 123 of file rank.hh.

10.48 mln::accu::meta::stat::rank_high_quant Struct Reference

Meta accumulator for [rank_high_quant](#).

```
#include <rank_high_quant.hh>
```

Inheritance diagram for mln::accu::meta::stat::rank_high_quant:



10.48.1 Detailed Description

Meta accumulator for [rank_high_quant](#).

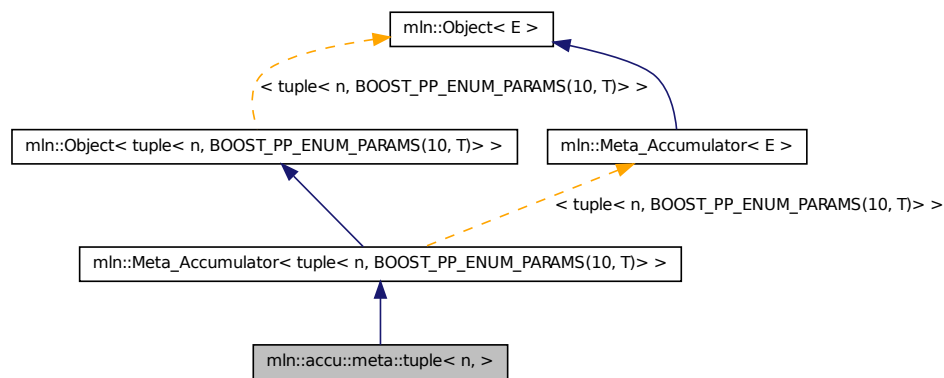
Definition at line 102 of file rank_high_quant.hh.

10.49 mln::accu::meta::tuple< n, > Struct Template Reference

Meta accumulator for tuple.

```
#include <tuple.hh>
```

Inheritance diagram for `mln::accu::meta::tuple< n, >`:



10.49.1 Detailed Description

```
template<unsigned n, BOOST_PP_ENUM_PARAMS_WITH_A_DEFAULT(10, typename T,
boost::tuples::null_type)> struct mln::accu::meta::tuple< n, >
```

Meta accumulator for tuple.

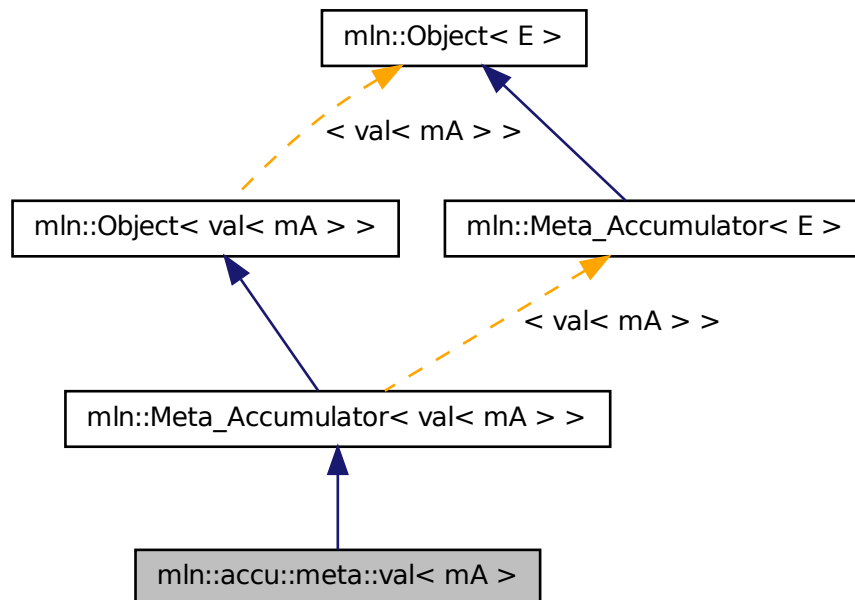
Definition at line 111 of file tuple.hh.

10.50 mln::accu::meta::val< mA > Struct Template Reference

Meta accumulator for val.

```
#include <v.hh>
```


Inheritance diagram for mln::accu::meta::val< mA > :



10.50.1 Detailed Description

```
template<typename mA> struct mln::accu::meta::val< mA >
```

Meta accumulator for val.

Definition at line 87 of file v.hh.

10.51 mln::accu::nil< T > Struct Template Reference

Define an accumulator that does nothing.

```
#include <nil.hh>
```

Inherits base< util::ignore, nil< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)

Take as initialization the value t .

- void `take_n_times` (unsigned n , const T & t)

Take n times the value t .

- `util::ignore_to_result` () const

Get the value of the accumulator.

- void `init` ()

Manipulators.

10.51.1 Detailed Description

template<typename T> struct mln::accu::nil< T >

Define an accumulator that does nothing.

Definition at line 49 of file accu/nil.hh.

10.51.2 Member Function Documentation

10.51.2.1 template<typename T > void mln::accu::nil< T >::init () [inline]

Manipulators.

Definition at line 100 of file accu/nil.hh.

10.51.2.2 template<typename T > bool mln::accu::nil< T >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 136 of file accu/nil.hh.

10.51.2.3 void mln::Accumulator< nil< T > >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.51.2.4 void mln::Accumulator< nil< T > >::take_n_times (unsigned n , const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.51.2.5 `template<typename T > util::ignore mln::accu::nil< T >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 128 of file accu/nil.hh.

10.52 mln::accu::p< A > Struct Template Reference

Generic p of accumulators.

```
#include <p.hh>
```

Inherits base< const A::result &, p< A > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- const A::result & [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.52.1 Detailed Description

`template<typename A> struct mln::accu::p< A >`

Generic p of accumulators. The parameter V is the type of values.

Definition at line 50 of file p.hh.

10.52.2 Member Function Documentation

10.52.2.1 `template<typename A > void mln::accu::p< A >::init () [inline]`

Manipulators.

Definition at line 115 of file p.hh.

10.52.2.2 `template<typename A > bool mln::accu::p< A >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 155 of file p.hh.

10.52.2.3 `void mln::Accumulator< p< A > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.52.2.4 `void mln::Accumulator< p< A > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.52.2.5 `template<typename A > const A::result & mln::accu::p< A >::to_result () const [inline]`

Get the value of the accumulator.

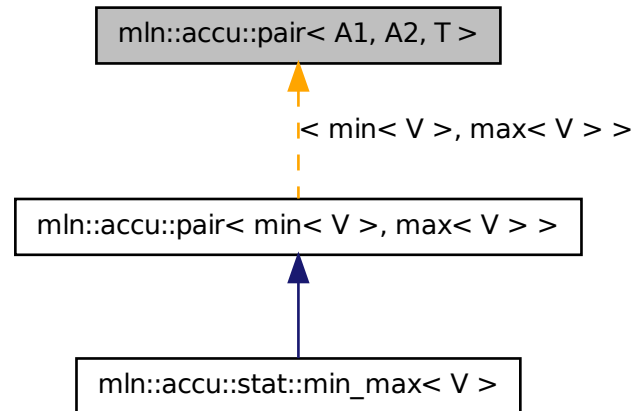
Definition at line 147 of file p.hh.

10.53 `mln::accu::pair< A1, A2, T >` Struct Template Reference

Generic pair of accumulators.

```
#include <pair.hh>
```

Inheritance diagram for mln::accu::pair< A1, A2, T >:



Public Member Functions

- A1::result [first](#) () const
Return the result of the first accumulator.
- A1 [first_accu](#) () const
Return the first accumulator.
- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- A2::result [second](#) () const
Return the result of the second accumulator.
- A2 [second_accu](#) () const
Return the second accumulator.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- void [init](#) ()
Manipulators.

- `std::pair< typename A1::result, typename A2::result > to_result () const`
Get the value of the accumulator.

10.53.1 Detailed Description

`template<typename A1, typename A2, typename T = mln_argument(A1)> struct mln::accu::pair< A1, A2, T >`

Generic pair of accumulators. The parameter T is the type of values.

Definition at line 58 of file pair.hh.

10.53.2 Member Function Documentation

10.53.2.1 `template<typename A1 , typename A2 , typename T > A1::result mln::accu::pair< A1, A2, T >::first () const` [`inline`]

Return the result of the first accumulator.

Definition at line 191 of file pair.hh.

10.53.2.2 `template<typename A1 , typename A2 , typename T > A1 mln::accu::pair< A1, A2, T >::first_accu () const` [`inline`]

Return the first accumulator.

Definition at line 209 of file pair.hh.

10.53.2.3 `template<typename A1 , typename A2 , typename T > void mln::accu::pair< A1, A2, T >::init ()` [`inline`]

Manipulators.

Definition at line 136 of file pair.hh.

10.53.2.4 `template<typename A1 , typename A2 , typename T > bool mln::accu::pair< A1, A2, T >::is_valid () const` [`inline`]

Check whether this accu is able to return a result.

Always true here.

Definition at line 226 of file pair.hh.

10.53.2.5 `template<typename A1 , typename A2 , typename T > A2::result mln::accu::pair< A1, A2, T >::second () const` [`inline`]

Return the result of the second accumulator.

Definition at line 199 of file pair.hh.

10.53.2.6 `template<typename A1 , typename A2 , typename T > A2 mln::accu::pair< A1, A2, T >::second_accu () const [inline]`

Return the second accumulator.

Definition at line 217 of file pair.hh.

10.53.2.7 `void mln::Accumulator< pair< A1, A2, T > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.53.2.8 `void mln::Accumulator< pair< A1, A2, T > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.53.2.9 `template<typename A1 , typename A2 , typename T > std::pair< typename A1::result, typename A2::result > mln::accu::pair< A1, A2, T >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 172 of file pair.hh.

10.54 mln::accu::rms< T, V > Struct Template Reference

Generic root mean square accumulator class.

```
#include <rms.hh>
```

Inherits base< V, rms< T, V > >.

Public Member Functions

- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
Take as initialization the value `t`.
- void `take_n_times` (unsigned n, const T &t)
Take `n` times the value `t`.
- V `to_result` () const
Get the value of the accumulator.

- void `init()`
Manipulators.

10.54.1 Detailed Description

`template<typename T, typename V> struct mln::accu::rms< T, V >`

Generic root mean square accumulator class. The parameter T is the type of the root mean square value.

Definition at line 52 of file `accu/rms.hh`.

10.54.2 Member Function Documentation

10.54.2.1 `template<typename T , typename V > void mln::accu::rms< T, V >::init ()`
`[inline]`

Manipulators.

Definition at line 112 of file `accu/rms.hh`.

References `mln::literal::zero`.

10.54.2.2 `template<typename T , typename V > bool mln::accu::rms< T, V >::is_valid () const`
`[inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 166 of file `accu/rms.hh`.

10.54.2.3 `void mln::Accumulator< rms< T, V > >::take_as_init (const T & t)` `[inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.54.2.4 `void mln::Accumulator< rms< T, V > >::take_n_times (unsigned n, const T & t)`
`[inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.54.2.5 `template<typename T , typename V > V mln::accu::rms< T, V >::to_result () const`
`[inline]`

Get the value of the accumulator.

Definition at line 148 of file `accu/rms.hh`.

10.55 mln::accu::shape::bbox< P > Struct Template Reference

Generic bounding box accumulator class.

```
#include <bbox.hh>
```

Inherits base< const box< P > &, bbox< P > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- const [box](#)< P > & [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.55.1 Detailed Description

```
template<typename P> struct mln::accu::shape::bbox< P >
```

Generic bounding box accumulator class. The parameter P is the type of points.

Definition at line 55 of file accu/shape/bbox.hh.

10.55.2 Member Function Documentation

10.55.2.1 `template<typename P > void mln::accu::shape::bbox< P >::init () [inline]`

Manipulators.

Definition at line 124 of file accu/shape/bbox.hh.

10.55.2.2 `template<typename P > bool mln::accu::shape::bbox< P >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 224 of file accu/shape/bbox.hh.

10.55.2.3 void mln::Accumulator< bbox< P > >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.55.2.4 void mln::Accumulator< bbox< P > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.55.2.5 template<typename P > const box< P > & mln::accu::shape::bbox< P >::to_result () const [inline]

Get the value of the accumulator.

Definition at line 215 of file `accu/shape/bbox.hh`.

Referenced by `mln::geom::rotate()`.

10.56 mln::accu::shape::height< I > Struct Template Reference

Height accumulator.

```
#include <height.hh>
```

Inherits base< unsigned, height< I > >.

Public Types

- typedef [util::pix< I >](#) [argument](#)
The accumulated data type.
- typedef [argument::value](#) [value](#)
The value type associated to the pixel type.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- unsigned [to_result](#) () const

Get the value of the accumulator.

- void [init](#) ()
Manipulators.
- void [set_value](#) (unsigned h)
Force the value of the counter to h.

10.56.1 Detailed Description

template<typename I> struct mln::accu::shape::height< I >

Height accumulator. The parameter I is the image type on which the accumulator of pixels is built.

Definition at line 68 of file accu/shape/height.hh.

10.56.2 Member Typedef Documentation

10.56.2.1 template<typename I> typedef util::pix<I> mln::accu::shape::height< I >::argument

The accumulated data type.

The height of component is represented by the height of its root pixel. See [mln::morpho::closing_height](#) and [mln::morpho::opening_height](#) for actual uses of this accumulator. **FIXME:** Replaced by [mln::morpho::attribute::height](#)

Definition at line 78 of file accu/shape/height.hh.

10.56.2.2 template<typename I> typedef argument::value mln::accu::shape::height< I >::value

The value type associated to the pixel type.

Definition at line 80 of file accu/shape/height.hh.

10.56.3 Member Function Documentation

10.56.3.1 template<typename I> void mln::accu::shape::height< I >::init () [inline]

Manipulators.

Definition at line 150 of file accu/shape/height.hh.

10.56.3.2 template<typename I> bool mln::accu::shape::height< I >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 199 of file accu/shape/height.hh.

10.56.3.3 `template<typename I> void mln::accu::shape::height< I >::set_value (unsigned h) [inline]`

Force the value of the counter to h .

Definition at line 188 of file `accu/shape/height.hh`.

10.56.3.4 `void mln::Accumulator< height< I > >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.56.3.5 `void mln::Accumulator< height< I > >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.56.3.6 `template<typename I> unsigned mln::accu::shape::height< I >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 180 of file `accu/shape/height.hh`.

10.57 `mln::accu::shape::volume< I >` Struct Template Reference

Volume accumulator class.

```
#include <volume.hh>
```

Inherits `base< unsigned, volume< I > >`.

Public Types

- typedef `util::pix< I >` `argument`
The accumulated data type.
- typedef `argument::value` `value`
The value type associated to the pixel type.

Public Member Functions

- bool `is_valid ()` const
Check whether this `accu` is able to return a result.
- void `take_as_init (const T &t)`
Take as initialization the value t .

- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- unsigned [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.
- void [set_value](#) (unsigned v)
Force the value of the counter to v.

10.57.1 Detailed Description

template<typename I> struct mln::accu::shape::volume< I >

Volume accumulator class. The parameter I is the image type on which the accumulator of pixels is built. Definition at line 66 of file accu/shape/volume.hh.

10.57.2 Member Typedef Documentation

10.57.2.1 template<typename I> typedef util::pix<I> mln::accu::shape::volume< I >::argument

The accumulated data type.

The volume of component is represented by the volume of its root pixel. See mln::morpho::closing_volume and mln::morpho::opening_volume for actual uses of this accumulator. FIXME: Replaced by [mln::morpho::attribute::volume](#)

Definition at line 76 of file accu/shape/volume.hh.

10.57.2.2 template<typename I> typedef argument::value mln::accu::shape::volume< I >::value

The value type associated to the pixel type.

Definition at line 78 of file accu/shape/volume.hh.

10.57.3 Member Function Documentation

10.57.3.1 template<typename I> void mln::accu::shape::volume< I >::init () [inline]

Manipulators.

Definition at line 148 of file accu/shape/volume.hh.

References mln::literal::zero.

10.57.3.2 `template<typename I> bool mln::accu::shape::volume< I >::is_valid () const`
[inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 215 of file accu/shape/volume.hh.

10.57.3.3 `template<typename I> void mln::accu::shape::volume< I >::set_value (unsigned v)`
[inline]

Force the value of the counter to *v*.

Definition at line 204 of file accu/shape/volume.hh.

References mln::literal::zero.

10.57.3.4 `void mln::Accumulator< volume< I > >::take_as_init (const T & t)` **[inherited]**

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.57.3.5 `void mln::Accumulator< volume< I > >::take_n_times (unsigned n, const T & t)`
[inherited]

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.57.3.6 `template<typename I> unsigned mln::accu::shape::volume< I >::to_result () const`
[inline]

Get the value of the accumulator.

Definition at line 196 of file accu/shape/volume.hh.

10.58 mln::accu::site_set::rectangularity< P > Class Template Reference

Compute the rectangularity of a site set.

```
#include <rectangularity.hh>
```

Inherits couple< accu::shape::bbox< P >, accu::math::count< P >, float, rectangularity< P > >.

Public Member Functions

- A2::result [area](#) () const

Return the site set area.

- A1::result [bbox](#) () const
Return the site set bounding box.
- [rectangularity](#) ()
Constructor.
- template<typename T >
void [take_as_init](#) (const T &t)
Take as initialization the value t .
- template<typename T >
void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- result [to_result](#) () const
Return the rectangularity value.

10.58.1 Detailed Description

template<typename P> class mln::accu::site_set::rectangularity< P >

Compute the rectangularity of a site set.

Definition at line 51 of file rectangularity.hh.

10.58.2 Constructor & Destructor Documentation

10.58.2.1 template<typename P > mln::accu::site_set::rectangularity< P >::rectangularity ()
[inline]

Constructor.

Definition at line 91 of file rectangularity.hh.

10.58.3 Member Function Documentation

10.58.3.1 template<typename P > rectangularity< P >::A2::result
mln::accu::site_set::rectangularity< P >::area () const [inline]

Return the site set area.

Definition at line 107 of file rectangularity.hh.

10.58.3.2 template<typename P > rectangularity< P >::A1::result
mln::accu::site_set::rectangularity< P >::bbox () const [inline]

Return the site set bounding box.

Definition at line 98 of file rectangularity.hh.

10.58.3.3 `template<typename E > template<typename T > void mln::Accumulator< E >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 186 of file `accumulator.hh`.

References `mln::mln_exact()`.

10.58.3.4 `template<typename E > template<typename T > void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 213 of file `accumulator.hh`.

References `mln::mln_exact()`.

10.58.3.5 `template<typename P > rectangularity< P >::result mln::accu::site_set::rectangularity< P >::to_result () const [inline]`

Return the rectangularity value.

Definition at line 116 of file `rectangularity.hh`.

10.59 `mln::accu::stat::deviation< T, S, M > Struct Template Reference`

Generic standard deviation accumulator class.

```
#include <deviation.hh>
```

Inherits `base< M, deviation< T, S, M > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `M to_result () const`
Get the value of the accumulator.

- void `init()`
Manipulators.

10.59.1 Detailed Description

`template<typename T, typename S = typename mln::value::props< T >::sum, typename M = S>`
`struct mln::accu::stat::deviation< T, S, M >`

Generic standard deviation accumulator class. Parameter `T` is the type of values that we sum. Parameter `S` is the type to store the standard deviation; the default type of `S` is the summation type (property) of `T`. Parameter `M` is the type of the mean value; the default type of `M` is `S`.

Definition at line 62 of file `deviation.hh`.

10.59.2 Member Function Documentation

10.59.2.1 `template<typename T, typename S, typename M > void mln::accu::stat::deviation< T, S, M >::init() [inline]`

Manipulators.

Definition at line 132 of file `deviation.hh`.

10.59.2.2 `template<typename T, typename S, typename M > bool mln::accu::stat::deviation< T, S, M >::is_valid() const [inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 177 of file `deviation.hh`.

10.59.2.3 `void mln::Accumulator< deviation< T, S, M > >::take_as_init(const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.59.2.4 `void mln::Accumulator< deviation< T, S, M > >::take_n_times(unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.59.2.5 `template<typename T, typename S, typename M > M mln::accu::stat::deviation< T, S, M >::to_result() const [inline]`

Get the value of the accumulator.

Definition at line 159 of file `deviation.hh`.

10.60 `mln::accu::stat::histo3d_rgb< V >` Struct Template Reference

Define a histogram as accumulator which returns an [image3d](#).

```
#include <histo3d_rgb.hh>
```

Inherits `base< image3d< unsigned >, histo3d_rgb< V > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accumulator is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `histo3d_rgb ()`
Constructors.
- `void init ()`
Manipulators.
- `void take (const argument &t)`
Update the histogram with the RGB pixel t .
- `void take (const histo3d_rgb< V > &other)`
Update the histogram with an other histogram.
- `result to_result () const`
Accessors.

10.60.1 Detailed Description

```
template<typename V> struct mln::accu::stat::histo3d_rgb< V >
```

Define a histogram as accumulator which returns an [image3d](#). Param `V` defines the type of the input image value. It is in this space that we count the values. For instance, this histogram works well for `image2d< rgb<2> >` or with `image2d< rgb<7> >`. The number of bins depends directly the values `V`. For 8 bits there is 256x3 bins. Note that less quantification works too.

Definition at line 167 of file `histo3d_rgb.hh`.

10.60.2 Constructor & Destructor Documentation

10.60.2.1 `template<typename V > mln::accu::stat::histo3d_rgb< V >::histo3d_rgb ()` [inline]

Constructors.

Infer the size of the resulting `image3d` domain. By evaluating the minimum and the maximum of `V`, we define the domain of the resulting `image3d`.

Definition at line 244 of file `histo3d_rgb.hh`.

10.60.3 Member Function Documentation

10.60.3.1 `template<typename V > void mln::accu::stat::histo3d_rgb< V >::init ()` [inline]

Manipulators.

Initialize the histogram with zero value. This method must be called just before starting the use of the histogram. If it's not, resulting values won't converge to the density.

Definition at line 268 of file `histo3d_rgb.hh`.

References `mln::data::fill()`, and `mln::literal::zero`.

10.60.3.2 `template<typename V > bool mln::accu::stat::histo3d_rgb< V >::is_valid () const` [inline]

Check whether this accumulator is able to return a result.

Depends if the resulting `image1d` is valid. We can assume it is quite always the case.

Definition at line 307 of file `histo3d_rgb.hh`.

10.60.3.3 `template<typename V > void mln::accu::stat::histo3d_rgb< V >::take (const argument & t)` [inline]

Update the histogram with the RGB pixel `t`.

Parameters

[in] `t` a graylevel pixel of type `V`.

The end user shouldn't call this method. In place of it, he can go through the data compute interface.

Definition at line 275 of file `histo3d_rgb.hh`.

10.60.3.4 `template<typename V> void mln::accu::stat::histo3d_rgb< V >::take (const histo3d_rgb< V > & other)` [inline]

Update the histogram with an other histogram.

Parameters

[in] `other` the other histogram.

The end user shouldn't call this method. This is part of data compute interface mechanism.

Definition at line 286 of file histo3d_rgb.hh.

10.60.3.5 `void mln::Accumulator< histo3d_rgb< V > >::take_as_init (const T & t)`
[inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.60.3.6 `void mln::Accumulator< histo3d_rgb< V > >::take_n_times (unsigned n, const T & t)`
[inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.60.3.7 `template<typename V > histo3d_rgb< V >::result mln::accu::stat::histo3d_rgb< V >::to_result () const` **[inline]**

Accessors.

Return the histogram as an RGB [image3d](#). This is the machinery to communicate with data compute interface. The end user should'nt use it.

Definition at line 293 of file histo3d_rgb.hh.

10.61 mln::accu::stat::max< T > Struct Template Reference

Generic max accumulator class.

```
#include <max.hh>
```

Inherits base< const T &, max< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [set_value](#) (const T &t)
Force the value of the min to t .
- void [take_as_init](#) (const T &t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t .
- const T & [to_result](#) () const
Get the value of the accumulator.

- void `init()`
Manipulators.

10.61.1 Detailed Description

template<typename T> struct mln::accu::stat::max< T >

Generic max accumulator class. The parameter T is the type of values.

Definition at line 101 of file `accu/stat/max.hh`.

10.61.2 Member Function Documentation

10.61.2.1 template<typename T > void mln::accu::stat::max< T >::init () [inline]

Manipulators.

Definition at line 146 of file `accu/stat/max.hh`.

10.61.2.2 template<typename T > bool mln::accu::stat::max< T >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 196 of file `accu/stat/max.hh`.

10.61.2.3 template<typename T > void mln::accu::stat::max< T >::set_value (const T & t) [inline]

Force the value of the min to *t*.

Definition at line 180 of file `accu/stat/max.hh`.

10.61.2.4 void mln::Accumulator< max< T > >::take_as_init (const T & t) [inherited]

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.61.2.5 void mln::Accumulator< max< T > >::take_n_times (unsigned n, const T & t) [inherited]

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.61.2.6 `template<typename T> const T & mln::accu::stat::max< T >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 188 of file `accu/stat/max.hh`.

10.62 `mln::accu::stat::max_h< V >` Struct Template Reference

Generic max function based on histogram over a value set with type `V`.

```
#include <max_h.hh>
```

Inherits `base< const V &, max_h< V > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value `t`.
- `void take_n_times (unsigned n, const T &t)`
Take `n` times the value `t`.
- `const argument & to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.62.1 Detailed Description

```
template<typename V> struct mln::accu::stat::max_h< V >
```

Generic max function based on histogram over a value set with type `V`.

Definition at line 100 of file `max_h.hh`.

10.62.2 Member Function Documentation

10.62.2.1 `template<typename V> void mln::accu::stat::max_h< V >::init () [inline]`

Manipulators.

Definition at line 280 of file `max_h.hh`.

10.62.2.2 `template<typename V > bool mln::accu::stat::max_h< V >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 323 of file max_h.hh.

10.62.2.3 `void mln::Accumulator< max_h< V > >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.62.2.4 `void mln::Accumulator< max_h< V > >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.62.2.5 `template<typename V > const max_h< V >::argument & mln::accu::stat::max_h< V >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 304 of file max_h.hh.

10.63 mln::accu::stat::mean< T, S, M > Struct Template Reference

Generic mean accumulator class.

```
#include <mean.hh>
```

Inherits base< M, mean< T, S, M > >.

Public Member Functions

- `accu::math::count< T >::result count () const`
Get the cardinality.
- `bool is_valid () const`
Check whether this accu is able to return a result.
- `accu::math::sum< T >::result sum () const`
Get the sum of values.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`

Take n times the value τ .

- `M to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.63.1 Detailed Description

```
template<typename T, typename S = typename mln::value::props< T >::sum, typename M = S>  
struct mln::accu::stat::mean< T, S, M >
```

Generic mean accumulator class. Parameter `T` is the type of values that we sum. Parameter `S` is the type to store the sum of values; the default type of `S` is the summation type (property) of `T`. Parameter `M` is the type of the mean value; the default type of `M` is `S`.

Definition at line 119 of file `accu/stat/mean.hh`.

10.63.2 Member Function Documentation

```
10.63.2.1 template<typename T, typename S, typename M > mln::math::count< T >::result  
mln::accu::stat::mean< T, S, M >::count ( ) const [inline]
```

Get the cardinality.

Definition at line 242 of file `accu/stat/mean.hh`.

```
10.63.2.2 template<typename T, typename S, typename M > void mln::accu::stat::mean< T, S,  
M >::init ( ) [inline]
```

Manipulators.

Definition at line 173 of file `accu/stat/mean.hh`.

```
10.63.2.3 template<typename T, typename S, typename M > bool mln::accu::stat::mean< T, S,  
M >::is_valid ( ) const [inline]
```

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 234 of file `accu/stat/mean.hh`.

```
10.63.2.4 template<typename T, typename S, typename M > mln::math::sum< T >::result  
mln::accu::stat::mean< T, S, M >::sum ( ) const [inline]
```

Get the sum of values.

Definition at line 251 of file `accu/stat/mean.hh`.

10.63.2.5 void mln::Accumulator< mean< T, S, M > >::take_as_init (const T & t)
[inherited]

Take as initialization the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.63.2.6 void mln::Accumulator< mean< T, S, M > >::take_n_times (unsigned n, const T & t)
[inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.63.2.7 template<typename T, typename S, typename M > M mln::accu::stat::mean< T, S, M >::to_result () const [inline]

Get the value of the accumulator.

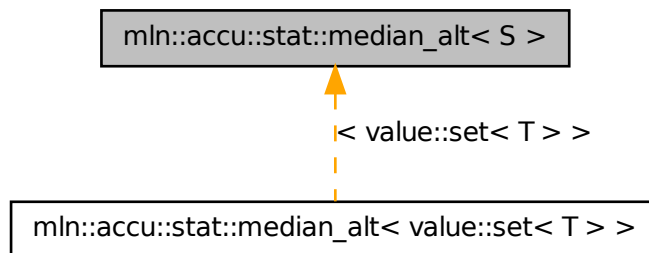
Definition at line 216 of file accu/stat/mean.hh.

10.64 mln::accu::stat::median_alt< S > Struct Template Reference

Generic [median_alt](#) function based on histogram over a value set with type S.

```
#include <median_alt.hh>
```

Inheritance diagram for mln::accu::stat::median_alt< S >:



Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)

Take as initialization the value t .

- void `take_n_times` (unsigned n , const T & t)

Take n times the value t .

- const argument & `to_result` () const

Get the value of the accumulator.

- void `take` (const argument & t)

Manipulators.

10.64.1 Detailed Description

`template<typename S> struct mln::accu::stat::median_alt< S >`

Generic `median_alt` function based on histogram over a value set with type S .

Definition at line 54 of file `median_alt.hh`.

10.64.2 Member Function Documentation

10.64.2.1 `template<typename S > bool mln::accu::stat::median_alt< S >::is_valid () const`
[inline]

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 282 of file `median_alt.hh`.

10.64.2.2 `template<typename S > void mln::accu::stat::median_alt< S >::take (const argument`
& t) [inline]

Manipulators.

Definition at line 165 of file `median_alt.hh`.

10.64.2.3 `void mln::Accumulator< median_alt< S > >::take_as_init (const T & t)`
[inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.64.2.4 `void mln::Accumulator< median_alt< S > >::take_n_times (unsigned n, const T & t`
) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.64.2.5 `template<typename S > const median_alt< S >::argument & mln::accu::stat::median_alt< S >::to_result () const [inline]`

Get the value of the accumulator.

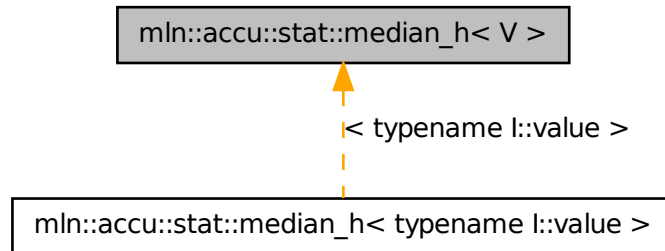
Definition at line 274 of file median_alt.hh.

10.65 `mln::accu::stat::median_h< V > Struct Template Reference`

Generic median function based on histogram over a value set with type `V`.

```
#include <median_h.hh>
```

Inheritance diagram for `mln::accu::stat::median_h< V >`:



Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value τ .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value τ .
- `const argument & to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.65.1 Detailed Description

```
template<typename V> struct mln::accu::stat::median_h< V >
```

Generic median function based on histogram over a value set with type *V*.

Definition at line 82 of file median_h.hh.

10.65.2 Member Function Documentation

10.65.2.1 `template<typename V > void mln::accu::stat::median_h< V >::init () [inline]`

Manipulators.

Definition at line 267 of file median_h.hh.

10.65.2.2 `template<typename V > bool mln::accu::stat::median_h< V >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 299 of file median_h.hh.

10.65.2.3 `void mln::Accumulator< median_h< V > >::take_as_init (const T & t) [inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.65.2.4 `void mln::Accumulator< median_h< V > >::take_n_times (unsigned n, const T & t) [inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.65.2.5 `template<typename V > const median_h< V >::argument & mln::accu::stat::median_h< V >::to_result () const [inline]`

Get the value of the accumulator.

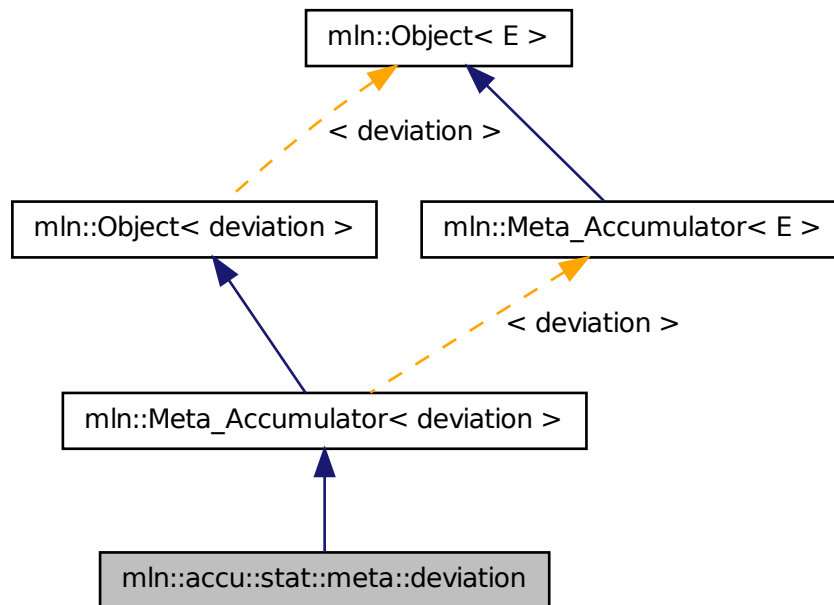
Definition at line 281 of file median_h.hh.

10.66 mln::accu::stat::meta::deviation Struct Reference

Meta accumulator for deviation.

```
#include <deviation.hh>
```

Inheritance diagram for mln::accu::stat::meta::deviation:



10.66.1 Detailed Description

Meta accumulator for deviation.

Definition at line 105 of file deviation.hh.

10.67 mln::accu::stat::min< T > Struct Template Reference

Generic min accumulator class.

```
#include <min.hh>
```

Inherits base< const T &, min< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [set_value](#) (const T &t)
Force the value of the min to t.

- void `take_as_init` (const T &t)
Take as initialization the value t .
- void `take_n_times` (unsigned n, const T &t)
Take n times the value t .
- const T & `to_result` () const
Get the value of the accumulator.
- void `init` ()
Manipulators.

10.67.1 Detailed Description

template<typename T> struct mln::accu::stat::min< T >

Generic min accumulator class. The parameter T is the type of values.

Definition at line 102 of file accu/stat/min.hh.

10.67.2 Member Function Documentation

10.67.2.1 template<typename T > void mln::accu::stat::min< T >::init () [inline]

Manipulators.

Definition at line 147 of file accu/stat/min.hh.

10.67.2.2 template<typename T > bool mln::accu::stat::min< T >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 195 of file accu/stat/min.hh.

10.67.2.3 template<typename T > void mln::accu::stat::min< T >::set_value (const T & t) [inline]

Force the value of the min to t .

Definition at line 179 of file accu/stat/min.hh.

10.67.2.4 void mln::Accumulator< min< T > >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.67.2.5 void mln::Accumulator< min< T > >::take_n_times (unsigned n, const T & t)
[inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.67.2.6 template<typename T > const T & mln::accu::stat::min< T >::to_result () const
[inline]

Get the value of the accumulator.

Definition at line 187 of file accu/stat/min.hh.

10.68 mln::accu::stat::min_h< V > Struct Template Reference

Generic min function based on histogram over a value set with type V.

```
#include <min_h.hh>
```

Inherits base< const V &, min_h< V > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- const argument & [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.68.1 Detailed Description

```
template<typename V> struct mln::accu::stat::min_h< V >
```

Generic min function based on histogram over a value set with type V.

Definition at line 100 of file min_h.hh.

10.68.2 Member Function Documentation

10.68.2.1 `template<typename V > void mln::accu::stat::min_h< V >::init () [inline]`

Manipulators.

Definition at line 256 of file min_h.hh.

10.68.2.2 `template<typename V > bool mln::accu::stat::min_h< V >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 298 of file min_h.hh.

10.68.2.3 `void mln::Accumulator< min_h< V > >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.68.2.4 `void mln::Accumulator< min_h< V > >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.68.2.5 `template<typename V > const min_h< V >::argument & mln::accu::stat::min_h< V >::to_result () const [inline]`

Get the value of the accumulator.

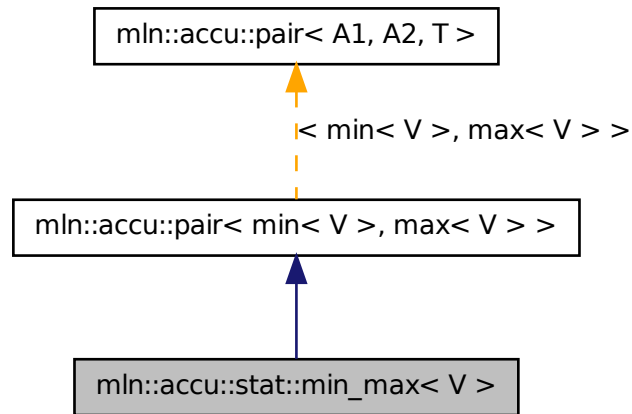
Definition at line 280 of file min_h.hh.

10.69 mln::accu::stat::min_max< V > Struct Template Reference

Generic min and max accumulator class.

```
#include <min_max.hh>
```


Inheritance diagram for mln::accu::stat::min_max< V >:



Public Member Functions

- `min< V >::result first () const`
Return the result of the first accumulator.
- `min< V > first_accu () const`
Return the first accumulator.
- `bool is_valid () const`
Check whether this accu is able to return a result.
- `max< V >::result second () const`
Return the result of the second accumulator.
- `max< V > second_accu () const`
Return the second accumulator.
- `template<typename T >`
`void take_as_init (const T &t)`
Take as initialization the value t.
- `template<typename T >`
`void take_n_times (unsigned n, const T &t)`
Take n times the value t.
- `void init ()`
Manipulators.

- `std::pair< typename min< V >::result, typename max< V >::result > to_result () const`
Get the value of the accumulator.

10.69.1 Detailed Description

`template<typename V> struct mln::accu::stat::min_max< V >`

Generic min and max accumulator class. The parameter `V` is the type of values.

Definition at line 61 of file `accu/stat/min_max.hh`.

10.69.2 Member Function Documentation

10.69.2.1 `min< V > ::result mln::accu::pair< min< V > , max< V > , mln_argument(min< V >) >::first () const` [*inherited*]

Return the result of the first accumulator.

10.69.2.2 `min< V > mln::accu::pair< min< V > , max< V > , mln_argument(min< V >) >::first_accu () const` [*inherited*]

Return the first accumulator.

10.69.2.3 `void mln::accu::pair< min< V > , max< V > , mln_argument(min< V >) >::init ()` [*inherited*]

Manipulators.

10.69.2.4 `bool mln::accu::pair< min< V > , max< V > , mln_argument(min< V >) >::is_valid () const` [*inherited*]

Check whether this `accu` is able to return a result.

Always true here.

10.69.2.5 `max< V > ::result mln::accu::pair< min< V > , max< V > , mln_argument(min< V >) >::second () const` [*inherited*]

Return the result of the second accumulator.

10.69.2.6 `max< V > mln::accu::pair< min< V > , max< V > , mln_argument(min< V >) >::second_accu () const` [*inherited*]

Return the second accumulator.

10.69.2.7 `template<typename E > template<typename T > void mln::Accumulator< E >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 186 of file `accumulator.hh`.

References `mln::mln_exact()`.

10.69.2.8 `template<typename E > template<typename T > void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 213 of file `accumulator.hh`.

References `mln::mln_exact()`.

10.69.2.9 `std::pair<typename min< V > ::result, typename max< V > ::result> mln::accu::pair< min< V > , max< V > , mln_argument(min< V >) >::to_result () const [inherited]`

Get the value of the accumulator.

10.70 mln::accu::stat::rank< T > Struct Template Reference

Generic rank accumulator class.

```
#include <rank.hh>
```

Inherits `base< const T &, rank< T > >`.

Public Member Functions

- bool `is_valid` () const
Check whether this accu is able to return a result.
- unsigned `k` () const
Give the rank.
- void `take_as_init` (const T &t)
Take as initialization the value t .
- void `take_n_times` (unsigned n, const T &t)
Take n times the value t .
- const T & `to_result` () const
Get the value of the accumulator.

- void `init()`
Manipulators.

10.70.1 Detailed Description

template<typename T> struct mln::accu::stat::rank< T >

Generic rank accumulator class. The parameter T is the type of values.

Definition at line 60 of file rank.hh.

10.70.2 Member Function Documentation

10.70.2.1 template<typename T > void mln::accu::stat::rank< T >::init () [inline]

Manipulators.

Definition at line 319 of file rank.hh.

10.70.2.2 template<typename T > bool mln::accu::stat::rank< T >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 343 of file rank.hh.

10.70.2.3 template<typename T > unsigned mln::accu::stat::rank< T >::k () const [inline]

Give the rank.

Definition at line 178 of file rank.hh.

10.70.2.4 void mln::Accumulator< rank< T > >::take_as_init (const T & t) [inherited]

Take as initialization the value t.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.70.2.5 void mln::Accumulator< rank< T > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.70.2.6 `template<typename T> const T & mln::accu::stat::rank< T >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 333 of file rank.hh.

10.71 `mln::accu::stat::rank< bool > Struct Template Reference`

rank accumulator class for Boolean.

```
#include <rank_bool.hh>
```

Inherits base< bool, rank< bool > >.

Public Member Functions

- bool `is_valid () const`
Check whether this accu is able to return a result.
- void `take_as_init (const T &t)`
Take as initialization the value t.
- void `take_n_times (unsigned n, const T &t)`
Take n times the value t.
- bool `to_result () const`
Get the value of the accumulator.
- void `init ()`
Manipulators.

10.71.1 Detailed Description

```
template<> struct mln::accu::stat::rank< bool >
```

rank accumulator class for Boolean.

Definition at line 58 of file rank_bool.hh.

10.71.2 Member Function Documentation

10.71.2.1 `void mln::accu::stat::rank< bool >::init () [inline]`

Manipulators.

Definition at line 105 of file rank_bool.hh.

10.71.2.2 `bool mln::accu::stat::rank< bool >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 157 of file rank_bool.hh.

10.71.2.3 `void mln::Accumulator< rank< bool > >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.71.2.4 `void mln::Accumulator< rank< bool > >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.71.2.5 `bool mln::accu::stat::rank< bool >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 150 of file rank_bool.hh.

10.72 `mln::accu::stat::rank_high_quant< T >` Struct Template Reference

Generic rank accumulator class.

```
#include <rank_high_quant.hh>
```

Inherits `base< const T &, rank_high_quant< T > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `const T & to_result () const`
Get the value of the accumulator.

- void `init()`
Manipulators.

10.72.1 Detailed Description

`template<typename T> struct mln::accu::stat::rank_high_quant< T >`

Generic rank accumulator class. The parameter `T` is the type of values.

Definition at line 57 of file `rank_high_quant.hh`.

10.72.2 Member Function Documentation

10.72.2.1 `template<typename T > void mln::accu::stat::rank_high_quant< T >::init ()`
`[inline]`

Manipulators.

Definition at line 148 of file `rank_high_quant.hh`.

10.72.2.2 `template<typename T > bool mln::accu::stat::rank_high_quant< T >::is_valid ()`
`const [inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 197 of file `rank_high_quant.hh`.

10.72.2.3 `void mln::Accumulator< rank_high_quant< T > >::take_as_init (const T & t)`
`[inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.72.2.4 `void mln::Accumulator< rank_high_quant< T > >::take_n_times (unsigned n, const T & t)`
`[inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.72.2.5 `template<typename T > const T & mln::accu::stat::rank_high_quant< T >::to_result ()`
`const [inline]`

Get the value of the accumulator.

Definition at line 183 of file `rank_high_quant.hh`.

10.73 mln::accu::stat::var< T > Struct Template Reference

Var accumulator class.

```
#include <var.hh>
```

Inherits base< algebra::mat< T::dim, T::dim, float >, var< T > >.

Public Types

- typedef algebra::vec< dim, float > [mean_t](#)
Type equipment.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu returns a valid result.
- [mean_t](#) [mean](#) () const
Get the mean vector.
- unsigned [n_items](#) () const
Get the number of items.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- result [to_result](#) () const
Get the accumulator result (the var value).
- result [variance](#) () const
Get the variance matrix.
- void [init](#) ()
Manipulators.

10.73.1 Detailed Description

```
template<typename T> struct mln::accu::stat::var< T >
```

Var accumulator class. Parameter T is the type of vectors

Definition at line 58 of file accu/stat/var.hh.

10.73.2 Member Typedef Documentation

10.73.2.1 `template<typename T> typedef algebra::vec<dim,float> mln::accu::stat::var< T >::mean_t`

Type equipment.

Definition at line 88 of file `accu/stat/var.hh`.

10.73.3 Member Function Documentation

10.73.3.1 `template<typename T> void mln::accu::stat::var< T >::init () [inline]`

Manipulators.

Definition at line 118 of file `accu/stat/var.hh`.

10.73.3.2 `template<typename T> bool mln::accu::stat::var< T >::is_valid () const [inline]`

Check whether this accu returns a valid result.

Definition at line 213 of file `accu/stat/var.hh`.

10.73.3.3 `template<typename T> var< T >::mean_t mln::accu::stat::var< T >::mean () const [inline]`

Get the mean vector.

Definition at line 200 of file `accu/stat/var.hh`.

References `mln::literal::zero`.

10.73.3.4 `template<typename T> unsigned mln::accu::stat::var< T >::n_items () const [inline]`

Get the number of items.

Definition at line 192 of file `accu/stat/var.hh`.

10.73.3.5 `void mln::Accumulator< var< T > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.73.3.6 `void mln::Accumulator< var< T > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.73.3.7 `template<typename T> var< T >::result mln::accu::stat::var< T >::to_result ()`
`const [inline]`

Get the accumulator result (the var value).

Definition at line 168 of file `accu/stat/var.hh`.

References `mln::literal::zero`.

10.73.3.8 `template<typename T> var< T >::result mln::accu::stat::var< T >::variance ()`
`const [inline]`

Get the variance matrix.

Definition at line 184 of file `accu/stat/var.hh`.

10.74 `mln::accu::stat::variance< T, S, R >` Struct Template Reference

Variance accumulator class.

`#include <variance.hh>`

Inherits `base< R, variance< T, S, R > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `R mean () const`
Get the mean value.
- `unsigned n_items () const`
Get the number of items.
- `R standard_deviation () const`
Get the standard deviation value.
- `S sum () const`
Get the sum value.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `R to_result () const`
Get the accumulator result (the variance value).

- R `var ()` const
Get the variance value.
- void `init ()`
Manipulators.

10.74.1 Detailed Description

template<typename T, typename S = typename mln::value::props< T >::sum, typename R = S>
struct mln::accu::stat::variance< T, S, R >

Variance accumulator class. Parameter `T` is the type of values that we sum. Parameter `S` is the type to store the value sum and the sum of value * value; the default type of `S` is the summation type (property) of `T`. Parameter `R` is the type of the mean and variance values; the default type of `R` is `S`.

Definition at line 61 of file variance.hh.

10.74.2 Member Function Documentation

10.74.2.1 **template<typename T, typename S, typename R > void mln::accu::stat::variance< T, S, R >::init () [inline]**

Manipulators.

Definition at line 125 of file variance.hh.

References `mln::literal::zero`.

10.74.2.2 **template<typename T, typename S, typename R > bool mln::accu::stat::variance< T, S, R >::is_valid () const [inline]**

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 229 of file variance.hh.

10.74.2.3 **template<typename T, typename S, typename R > R mln::accu::stat::variance< T, S, R >::mean () const [inline]**

Get the mean value.

Definition at line 187 of file variance.hh.

References `mln::literal::zero`.

10.74.2.4 **template<typename T, typename S, typename R > unsigned mln::accu::stat::variance< T, S, R >::n_items () const [inline]**

Get the number of items.

Definition at line 205 of file variance.hh.

10.74.2.5 `template<typename T, typename S, typename R > R mln::accu::stat::variance< T, S, R >::standard_deviation () const [inline]`

Get the standard deviation value.

Definition at line 221 of file variance.hh.

References `mln::accu::stat::variance< T, S, R >::to_result()`.

10.74.2.6 `template<typename T, typename S, typename R > S mln::accu::stat::variance< T, S, R >::sum () const [inline]`

Get the sum value.

Definition at line 197 of file variance.hh.

10.74.2.7 `void mln::Accumulator< variance< T, S, R > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.74.2.8 `void mln::Accumulator< variance< T, S, R > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.74.2.9 `template<typename T, typename S, typename R > R mln::accu::stat::variance< T, S, R >::to_result () const [inline]`

Get the accumulator result (the variance value).

Definition at line 176 of file variance.hh.

Referenced by `mln::accu::stat::variance< T, S, R >::standard_deviation()`, and `mln::accu::stat::variance< T, S, R >::var()`.

10.74.2.10 `template<typename T, typename S, typename R > R mln::accu::stat::variance< T, S, R >::var () const [inline]`

Get the variance value.

Definition at line 213 of file variance.hh.

References `mln::accu::stat::variance< T, S, R >::to_result()`.

10.75 mln::accu::tuple< A, n, > Struct Template Reference

Generic tuple of accumulators.

```
#include <tuple.hh>
```

Inherits base< boost::tuple< BOOST_PP_REPEAT(10, RESULT_ACCU, Le Ricard ya que ca de vrai!) >, tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- res [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.75.1 Detailed Description

template<typename A, unsigned n, BOOST_PP_ENUM_PARAMS_WITH_A_DEFAULT(10, typename T, boost::tuples::null_type)> struct mln::accu::tuple< A, n, >

Generic tuple of accumulators. The parameter T is the type of values.

Definition at line 74 of file tuple.hh.

10.75.2 Member Function Documentation

10.75.2.1 **template<typename A , unsigned n, BOOST_PP_ENUM_PARAMS(10, typename T) > void mln::accu::tuple< A, n, >::init () [inline]**

Manipulators.

Definition at line 197 of file tuple.hh.

10.75.2.2 **template<typename A , unsigned n, BOOST_PP_ENUM_PARAMS(10, typename T) > bool mln::accu::tuple< A, n, >::is_valid () const [inline]**

Check whether this accu is able to return a result.

Always true here.

Definition at line 239 of file tuple.hh.

10.75.2.3 `void mln::Accumulator< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.75.2.4 `void mln::Accumulator< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.75.2.5 `template<typename A , unsigned n, BOOST_PP_ENUM_PARAMS(10, typename T) > tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T) >::res mln::accu::tuple< A, n, >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 229 of file `tuple.hh`.

10.76 `mln::accu::val< A >` Struct Template Reference

Generic `val` of accumulators.

```
#include <v.hh>
```

Inherits `base< const A::result &, val< A > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t .
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `const A::result & to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.76.1 Detailed Description

template<typename A> struct mln::accu::val< A >

Generic val of accumulators.

Definition at line 50 of file v.hh.

10.76.2 Member Function Documentation

10.76.2.1 template<typename A > void mln::accu::val< A >::init () [inline]

Manipulators.

Definition at line 119 of file v.hh.

10.76.2.2 template<typename A > bool mln::accu::val< A >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 177 of file v.hh.

10.76.2.3 void mln::Accumulator< val< A > >::take_as_init (const T & t) [inherited]

Take as initialization the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.76.2.4 void mln::Accumulator< val< A > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.76.2.5 template<typename A > const A::result & mln::accu::val< A >::to_result () const [inline]

Get the value of the accumulator.

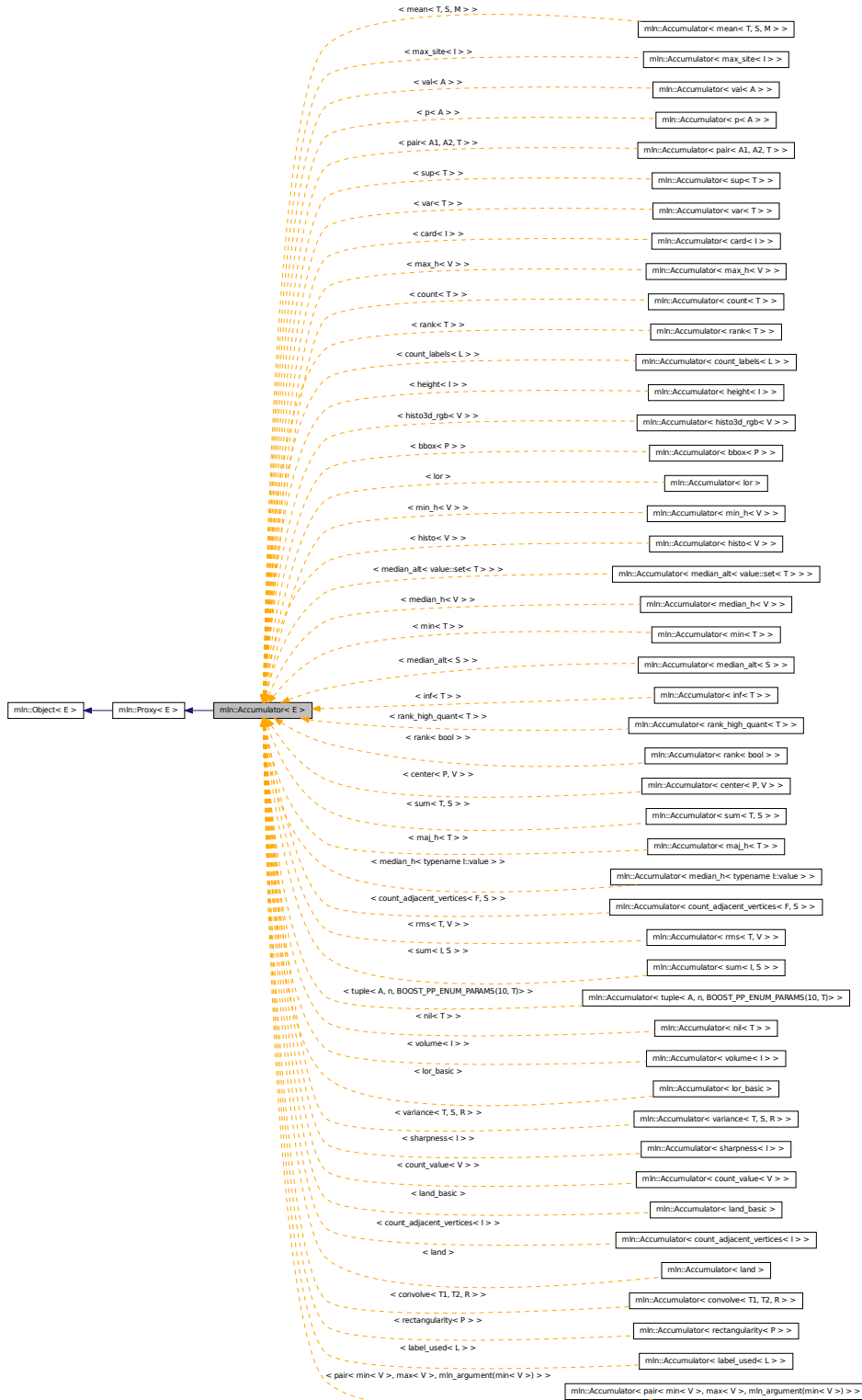
Definition at line 169 of file v.hh.

10.77 mln::Accumulator< E > Struct Template Reference

Base class for implementation of accumulators.

```
#include <accumulator.hh>
```

Inheritance diagram for mln::Accumulator< E >:



Public Member Functions

- `template<typename T >`
`void take_as_init (const T &t)`
Take as initialization the value t .
- `template<typename T >`
`void take_n_times (unsigned n, const T &t)`
Take n times the value t .

10.77.1 Detailed Description

`template<typename E> struct mln::Accumulator< E >`

Base class for implementation of accumulators. The parameter E is the exact type.

See also

[mln::doc::Accumulator](#) for a complete documentation of this class contents.

Definition at line 78 of file accumulator.hh.

10.77.2 Member Function Documentation

10.77.2.1 `template<typename E > template<typename T > void mln::Accumulator< E >::take_as_init (const T & t)`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 186 of file accumulator.hh.

References `mln::mln_exact()`.

10.77.2.2 `template<typename E > template<typename T > void mln::Accumulator< E >::take_n_times (unsigned n, const T & t)`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 213 of file accumulator.hh.

References `mln::mln_exact()`.

10.78 `mln::algebra::h_mat< d, T >` Struct Template Reference

N-Dimensional matrix with homogeneous coordinates.

```
#include <h_mat.hh>
```

Inherits `mln::algebra::mat< d+1, d+1, T >`.

Public Types

- enum

Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).

Public Member Functions

- `mat< n, m, T > _1 () const`
Return the inverse of the matrix.
- `h_mat ()`
Constructor without argument.
- `h_mat (const mat< d+1, d+1, T > &x)`
Constructor with the underlying matrix.
- `mat< m, n, T > t () const`
Return the transpose of the matrix.

10.78.1 Detailed Description

`template<unsigned d, typename T> struct mln::algebra::h_mat< d, T >`

N-Dimensional matrix with homogeneous coordinates.

Definition at line 49 of file algebra/h_mat.hh.

10.78.2 Member Enumeration Documentation

10.78.2.1 `template<unsigned d, typename T> anonymous enum`

Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).

Definition at line 52 of file algebra/h_mat.hh.

10.78.3 Constructor & Destructor Documentation

10.78.3.1 `template<unsigned d, typename T > mln::algebra::h_mat< d, T >::h_mat ()` `[inline]`

Constructor without argument.

Definition at line 67 of file algebra/h_mat.hh.

10.78.3.2 `template<unsigned d, typename T > mln::algebra::h_mat< d, T >::h_mat (const` `mat< d+1, d+1, T > & x) [inline]`

Constructor with the underlying matrix.

Definition at line 74 of file algebra/h_mat.hh.

10.78.4 Member Function Documentation

10.78.4.1 `template<unsigned n, unsigned m, typename T > mat< n, m, T > mln::algebra::mat< n, m, T >::_1 () const [inline, inherited]`

Return the inverse of the matrix.

Only compile on square matrix.

Definition at line 604 of file algebra/mat.hh.

10.78.4.2 `template<unsigned n, unsigned m, typename T > mat< m, n, T > mln::algebra::mat< n, m, T >::t () const [inline, inherited]`

Return the transpose of the matrix.

Definition at line 538 of file algebra/mat.hh.

10.79 mln::algebra::h_vec< d, C > Class Template Reference

N-Dimensional vector with homogeneous coordinates.

```
#include <h_vec.hh>
```

Inherits mln::algebra::vec< d+1, C >.

Public Types

- enum

Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).

Public Member Functions

- [h_vec](#) ()

Constructor without argument.

- [h_vec](#) (const vec< d+1, C > &other)

Constructor with the underlying vector.

- template<typename U >
[operator mat< n, 1, U > \(\) const](#)

Conversion to a matrix.

- mat< 1, n, T > [t](#) () const

Transposition.

- vec< d, C > [to_vec](#) () const

Back to the natural (non-homogeneous) space.

Static Public Attributes

- static const vec< n, T > **origin** = all_to(0)
Origin value.
- static const vec< n, T > **zero** = all_to(0)
Zero value.

10.79.1 Detailed Description

template<unsigned d, typename C> class mln::algebra::h_vec< d, C >

N-Dimensional vector with homogeneous coordinates.

Definition at line 94 of file h_vec.hh.

10.79.2 Member Enumeration Documentation

10.79.2.1 template<unsigned d, typename C> anonymous enum

Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).

Definition at line 98 of file h_vec.hh.

10.79.3 Constructor & Destructor Documentation

10.79.3.1 template<unsigned d, typename C > mln::algebra::h_vec< d, C >::h_vec ()
[inline]

Constructor without argument.

Definition at line 117 of file h_vec.hh.

References mln::literal::one.

10.79.3.2 template<unsigned d, typename C > mln::algebra::h_vec< d, C >::h_vec (const vec<
d+1, C > & other) [inline]

Constructor with the underlying vector.

Definition at line 128 of file h_vec.hh.

10.79.4 Member Function Documentation

10.79.4.1 template<unsigned n, typename T > template<typename U > mln::algebra::vec< n, T
>::operator mat< n, 1, U > () const [inline, inherited]

Conversion to a matrix.

Definition at line 368 of file algebra/mat.hh.

10.79.4.2 `template<unsigned n, typename T> mat< 1, n, T > mln::algebra::vec< n, T >::t () const [inline, inherited]`

Transposition.

Definition at line 858 of file algebra/mat.hh.

10.79.4.3 `template<unsigned d, typename C> vec< d, C > mln::algebra::h_vec< d, C >::to_vec () const [inline]`

Back to the natural (non-homogeneous) space.

Definition at line 145 of file h_vec.hh.

10.79.5 Member Data Documentation

10.79.5.1 `template<unsigned n, typename T> const vec< n, T > mln::algebra::vec< n, T >::origin = all_to(0) [static, inherited]`

Origin value.

Definition at line 247 of file algebra/vec.hh.

10.79.5.2 `template<unsigned n, typename T> const vec< n, T > mln::algebra::vec< n, T >::zero = all_to(0) [static, inherited]`

Zero value.

Definition at line 244 of file algebra/vec.hh.

10.80 mln::bkd_pixter1d< I > Class Template Reference

Backward pixel iterator on a 1-D image with border.

```
#include <pixter1d.hh>
```

Inherits `backward_pixel_iterator_base_< I, bkd_pixter1d< I > >`.

Public Types

- typedef `I image`
Image type.

Public Member Functions

- `bkd_pixter1d (I &image)`
Constructor.
- void `next ()`
Go to the next element.

10.80.1 Detailed Description

`template<typename I> class mln::bkd_pixter1d< I >`

Backward pixel iterator on a 1-D image with border.

Definition at line 69 of file `pixter1d.hh`.

10.80.2 Member Typedef Documentation

10.80.2.1 `template<typename I > typedef I mln::bkd_pixter1d< I >::image`

[Image](#) type.

Definition at line 76 of file `pixter1d.hh`.

10.80.3 Constructor & Destructor Documentation

10.80.3.1 `template<typename I > mln::bkd_pixter1d< I >::bkd_pixter1d (I & image)`
`[inline]`

Constructor.

Parameters

`[in]` *image* The image this pixel iterator is bound to.

Definition at line 117 of file `pixter1d.hh`.

10.80.4 Member Function Documentation

10.80.4.1 `void mln::Iterator< bkd_pixter1d< I > >::next ()` `[inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.81 mln::bkd_pixter2d< I > Class Template Reference

Backward pixel iterator on a 2-D image with border.

```
#include <pixter2d.hh>
```

Inherits `backward_pixel_iterator_base_< I, bkd_pixter2d< I > >`.

Public Types

- typedef I [image](#)
Image type.

Public Member Functions

- [bkd_pixter2d](#) (I &[image](#))
Constructor.
- void [next](#) ()
Go to the next element.

10.81.1 Detailed Description

`template<typename I> class mln::bkd_pixter2d< I >`

Backward pixel iterator on a 2-D image with border.

Definition at line 87 of file `pixter2d.hh`.

10.81.2 Member Typedef Documentation

10.81.2.1 `template<typename I > typedef I mln::bkd_pixter2d< I >::image`

[Image](#) type.

Definition at line 94 of file `pixter2d.hh`.

10.81.3 Constructor & Destructor Documentation

10.81.3.1 `template<typename I > mln::bkd_pixter2d< I >::bkd_pixter2d (I & image)`
`[inline]`

Constructor.

Parameters

`[in]` *image* The image this pixel iterator is bound to.

Definition at line 169 of file `pixter2d.hh`.

10.81.4 Member Function Documentation

10.81.4.1 `void mln::Iterator< bkd_pixter2d< I > >::next ()` `[inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-define this method. The actual "next" operation has to be defined through the *next_method*.

Precondition

The iterator is valid.

10.82 mln::bkd_pixter3d< I > Class Template Reference

Backward pixel iterator on a 3-D image with border.

```
#include <pixter3d.hh>
```

Inherits backward_pixel_iterator_base_< I, bkd_pixter3d< I > >.

Public Types

- typedef I [image](#)
Image type.

Public Member Functions

- [bkd_pixter3d](#) (I &[image](#))
Constructor.
- void [next](#) ()
Go to the next element.

10.82.1 Detailed Description

```
template<typename I> class mln::bkd_pixter3d< I >
```

Backward pixel iterator on a 3-D image with border.

Definition at line 100 of file pixter3d.hh.

10.82.2 Member Typedef Documentation

10.82.2.1 `template<typename I > typedef I mln::bkd_pixter3d< I >::image`

[Image](#) type.

Definition at line 107 of file pixter3d.hh.

10.82.3 Constructor & Destructor Documentation

10.82.3.1 `template<typename I > mln::bkd_pixter3d< I >::bkd_pixter3d (I & image)`
[inline]

Constructor.

Parameters

[in] *image* The image this pixel iterator is bound to.

Definition at line 207 of file `pixter3d.hh`.

10.82.4 Member Function Documentation

10.82.4.1 `void mln::Iterator< bkd_pixter3d< I > >::next ()` [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-define this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

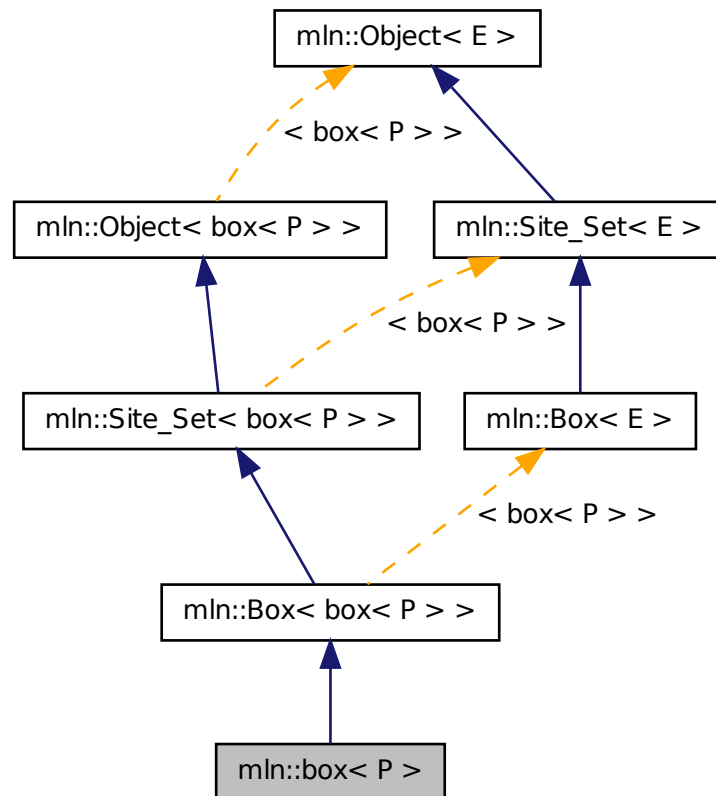
The iterator is valid.

10.83 mln::box< P > Class Template Reference

Generic box class: site set containing points of a regular grid.

```
#include <box.hh>
```

Inheritance diagram for `mln::box< P >`:



Public Types

- enum `Dimension`
Dimension.
- typedef `box_bkd_piter_< P >` `bkd_piter`
Backward [Site_Iterator](#) associated type.
- typedef `P` `element`
Element associated type.
- typedef `box_fwd_piter_< P >` `fwd_piter`
Forward [Site_Iterator](#) associated type.
- typedef `fwd_piter` `piter`
[Site_Iterator](#) associated type.

- typedef P [psite](#)
Psite associated type.
- typedef P [site](#)
Site associated type.

Public Member Functions

- const [box](#)< P > & [bbox](#) () const
Give the bounding box of this site set.
- [box](#) ()
Constructor without argument.
- [box](#) (const [site](#) &pmin, const [site](#) &pmax)
Constructor of a box going from pmin to pmax.
- void [crop_wrt](#) (const [box](#)< P > &b)
Crop this bbox in order to fit in the reference box b.
- void [enlarge](#) (unsigned b)
Enlarge the box with a border b.
- void [enlarge](#) (unsigned dim, unsigned b)
Enlarge the box with a border b for dimension dim.
- bool [has](#) (const P &p) const
Test if p belongs to the box.
- bool [is_empty](#) () const
Test if this box is empty.
- bool [is_valid](#) () const
Test that the box owns valid data, i.e., is initialized and with pmin being 'less-than' pmax.
- unsigned [len](#) (unsigned i) const
Give the length of the i-th side of the box.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- void [merge](#) (const [box](#)< P > &b)
Merge inplace with another box.
- unsigned [nsites](#) () const
Give the number of sites of this box.

- `P pcenter ()` const
Return the approximated central site of this box.
- `P & pmax ()`
Reference to the maximum point.
- `P pmax ()` const
Maximum point.
- `P pmin ()` const
Minimum point.
- `P & pmin ()`
Reference to the minimum point.
- `box< P > to_larger (unsigned b)` const
Give a larger box.
- `box (typename P::coord ninds)`

Related Functions

(Note that these are not member functions.)

- `template<typename P > std::ostream & operator<< (std::ostream &ostr, const box< P > &b)`
Print a generic box b into the output stream `ostr`.

10.83.1 Detailed Description

`template<typename P> class mln::box< P >`

Generic box class: site set containing points of a regular grid. Parameter `P` is the corresponding type of point.

Definition at line 81 of file `core/site_set/box.hh`.

10.83.2 Member Typedef Documentation

10.83.2.1 `template<typename P> typedef box_bkd_piter_<P> mln::box< P >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 105 of file `core/site_set/box.hh`.

10.83.2.2 `template<typename P> typedef P mln::box< P >::element`

Element associated type.

Definition at line 90 of file `core/site_set/box.hh`.

10.83.2.3 `template<typename P> typedef box_fwd_piter_<P> mln::box< P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 99 of file core/site_set/box.hh.

10.83.2.4 `template<typename P> typedef fwd_piter mln::box< P >::piter`

[Site_Iterator](#) associated type.

Definition at line 102 of file core/site_set/box.hh.

10.83.2.5 `template<typename P> typedef P mln::box< P >::psite`

Psite associated type.

Definition at line 93 of file core/site_set/box.hh.

10.83.2.6 `template<typename P> typedef P mln::box< P >::site`

[Site](#) associated type.

Definition at line 96 of file core/site_set/box.hh.

10.83.3 Member Enumeration Documentation**10.83.3.1** `template<typename P> anonymous enum`

Dimension.

Definition at line 87 of file core/site_set/box.hh.

10.83.4 Constructor & Destructor Documentation**10.83.4.1** `template<typename P> mln::box< P >::box () [inline]`

Constructor without argument.

Definition at line 275 of file core/site_set/box.hh.

10.83.4.2 `template<typename P> mln::box< P >::box (const site & pmin, const site & pmax) [inline]`

Constructor of a box going from pmin to pmax.

Definition at line 284 of file core/site_set/box.hh.

References mln::box< P >::is_valid().

10.83.4.3 `template<typename P> mln::box< P >::box (typename P::coord ninds) [inline, explicit]`

Constructors with different numbers of arguments (sizes) w.r.t. the dimension.

Definition at line 293 of file core/site_set/box.hh.

References `mln::literal::origin`.

10.83.5 Member Function Documentation

10.83.5.1 `const box< P > & mln::Box< box< P > >::bbox () const [inherited]`

Give the bounding box of this site set.

Return the bounding box of this site set, so that is itself. This method is declared by the [mln::Site_Set](#) concept.

Warning

This method is final for all box classes.

10.83.5.2 `template<typename P > void mln::box< P >::crop_wrt (const box< P > & b) [inline]`

Crop this bbox in order to fit in the reference box `b`.

Definition at line 205 of file core/site_set/box.hh.

References `mln::box< P >::pmax()`, and `mln::box< P >::pmin()`.

Referenced by `mln::debug::draw_graph()`, and `mln::make_debug_graph_image()`.

10.83.5.3 `template<typename P > void mln::box< P >::enlarge (unsigned b) [inline]`

Enlarge the box with a border `b`.

Definition at line 337 of file core/site_set/box.hh.

References `mln::box< P >::is_valid()`.

Referenced by `mln::registration::icp()`.

10.83.5.4 `template<typename P > void mln::box< P >::enlarge (unsigned dim, unsigned b) [inline]`

Enlarge the box with a border `b` for dimension `dim`.

Definition at line 351 of file core/site_set/box.hh.

References `mln::box< P >::is_valid()`.

10.83.5.5 `template<typename P > bool mln::box< P >::has (const P & p) const [inline]`

Test if `p` belongs to the box.

Parameters

[in] `p` A point site.

Definition at line 325 of file `core/site_set/box.hh`.

References `mln::box< P >::is_valid()`.

Referenced by `mln::morpho::line_gradient()`.

10.83.5.6 `bool mln::Box< box< P > >::is_empty () const [inherited]`

Test if this box is empty.

10.83.5.7 `template<typename P > bool mln::box< P >::is_valid () const [inline]`

Test that the box owns valid data, i.e., is initialized and with `pmin` being 'less-than' `pmax`.

Definition at line 195 of file `core/site_set/box.hh`.

References `mln::util::ord_weak()`.

Referenced by `mln::box< P >::box()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::box< P >::enlarge()`, `mln::box< P >::has()`, `mln::box< P >::merge()`, `mln::box< P >::pcenter()`, `mln::box< P >::pmax()`, `mln::box< P >::pmin()`, and `mln::box< P >::to_larger()`.

10.83.5.8 `unsigned mln::Box< box< P > >::len (unsigned i) const [inherited]`

Give the length of the `i`-th side of the box.

Precondition

`i < site::dim`

Warning

This method is final for all box classes.

10.83.5.9 `template<typename P > std::size_t mln::box< P >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 407 of file `core/site_set/box.hh`.

10.83.5.10 `template<typename P > void mln::box< P >::merge (const box< P > & b) [inline]`

Merge inplace with another box.

Definition at line 221 of file `core/site_set/box.hh`.

References `mln::box< P >::is_valid()`, `mln::box< P >::pmax()`, and `mln::box< P >::pmin()`.

10.83.5.11 `unsigned mln::Box< box< P > >::nsites () const [inherited]`

Give the number of sites of this box.

Return the number of sites of this box. This method is declared by the `mln::Site_Set` concept.

Warning

This method is final for all box classes.

10.83.5.12 `template<typename P> P mln::box<P>::pcenter () const [inline]`

Return the approximated central site of this box.

Definition at line 395 of file core/site_set/box.hh.

References `mln::box<P>::is_valid()`.

10.83.5.13 `template<typename P> P mln::box<P>::pmax () const [inline]`

Maximum point.

Definition at line 259 of file core/site_set/box.hh.

References `mln::box<P>::is_valid()`.

Referenced by `mln::box<P>::crop_wrt()`, `mln::make::image3d()`, `mln::larger_than()`, `mln::io::fld::load()`, and `mln::box<P>::merge()`.

10.83.5.14 `template<typename P> P & mln::box<P>::pmax () [inline]`

Reference to the maximum point.

Definition at line 268 of file core/site_set/box.hh.

10.83.5.15 `template<typename P> P & mln::box<P>::pmin () [inline]`

Reference to the minimum point.

Definition at line 251 of file core/site_set/box.hh.

10.83.5.16 `template<typename P> P mln::box<P>::pmin () const [inline]`

Minimum point.

Definition at line 242 of file core/site_set/box.hh.

References `mln::box<P>::is_valid()`.

Referenced by `mln::box<P>::crop_wrt()`, `mln::make::image3d()`, `mln::larger_than()`, `mln::io::fld::load()`, and `mln::box<P>::merge()`.

10.83.5.17 `template<typename P> box<P> mln::box<P>::to_larger (unsigned b) const [inline]`

Give a larger box.

Definition at line 378 of file core/site_set/box.hh.

References `mln::box<P>::is_valid()`.

10.83.6 Friends And Related Function Documentation

10.83.6.1 `template<typename P > std::ostream & operator<< (std::ostream & ostr, const box< P > & b)` [**related**]

Print a generic box *b* into the output stream *ostr*.

Parameters

[in, out] *ostr* An output stream.

[in] *b* A generic box.

Returns

The modified output stream *ostr*.

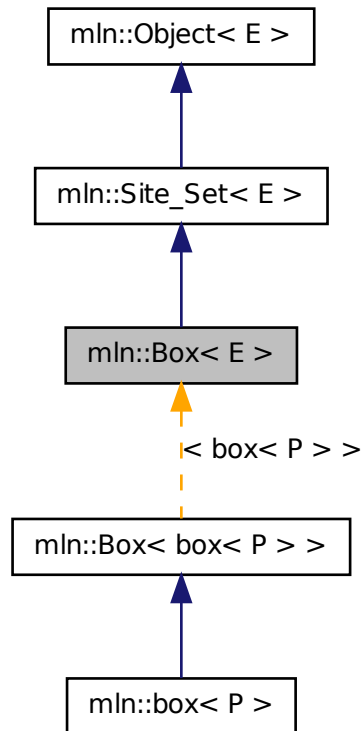
Definition at line 414 of file core/site_set/box.hh.

10.84 mln::Box< E > Struct Template Reference

Base class for implementation classes of boxes.

```
#include <box.hh>
```

Inheritance diagram for `mln::Box< E >`:



Public Member Functions

- `const E & bbox () const`
Give the bounding box of this site set.
- `bool is_empty () const`
Test if this box is empty.
- `unsigned len (unsigned i) const`
*Give the length of the *i*-th side of the box.*
- `unsigned nsites () const`
Give the number of sites of this box.

Related Functions

(Note that these are not member functions.)

- `template<typename SI , typename Sr >`
`p_set< typename SI::site > diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic difference of lhs and rhs.
- `template<typename SI , typename Sr >`
`p_set< typename SI::site > inter (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Intersection between a couple of point sets.
- `template<typename SI , typename Sr >`
`bool operator< (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Strict inclusion test between site sets lhs and rhs.
- `template<typename BI , typename Br >`
`bool operator< (const Box< BI > &lhs, const Box< Br > &rhs)`
Strict inclusion test between boxes lhs and rhs.
- `template<typename S >`
`std::ostream & operator<< (std::ostream &ostr, const Site_Set< S > &set)`
Print a site set set into the output stream ostr.
- `template<typename SI , typename Sr >`
`bool operator<= (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Inclusion test between site sets lhs and rhs.
- `template<typename BI , typename Br >`
`bool operator<= (const Box< BI > &lhs, const Box< Br > &rhs)`
Inclusion test between boxes lhs and rhs.
- `template<typename SI , typename Sr >`
`bool operator== (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Equality test between site sets lhs and rhs.
- `template<typename SI , typename Sr >`
`p_set< typename SI::site > sym_diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic symmetrical difference of lhs and rhs.
- `template<typename SI , typename Sr >`
`p_set< typename SI::site > uni (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Union of a couple of point sets.
- `template<typename S >`
`p_set< typename S::site > unique (const Site_Set< S > &s)`
Give the unique set of s.

10.84.1 Detailed Description

`template<typename E> struct mln::Box< E >`

Base class for implementation classes of boxes. Boxes are particular site sets useful to bound any set of sites defined on a regular grid.

See also

[mln::doc::Box](#) for a complete documentation of this class contents.

Definition at line 48 of file core/concept/box.hh.

10.84.2 Member Function Documentation

10.84.2.1 `template<typename E> const E & mln::Box< E >::bbox () const [inline]`

Give the bounding box of this site set.

Return the bounding box of this site set, so that is itself. This method is declared by the [mln::Site_Set](#) concept.

Warning

This method is final for all box classes.

Definition at line 124 of file core/concept/box.hh.

10.84.2.2 `template<typename E> bool mln::Box< E >::is_empty () const [inline]`

Test if this box is empty.

Definition at line 167 of file core/concept/box.hh.

10.84.2.3 `template<typename E> unsigned mln::Box< E >::len (unsigned i) const [inline]`

Give the length of the *i*-th side of the box.

Precondition

$i < \text{site::dim}$

Warning

This method is final for all box classes.

Definition at line 131 of file core/concept/box.hh.

10.84.2.4 `template<typename E> unsigned mln::Box< E >::nsites () const [inline]`

Give the number of sites of this box.

Return the number of sites of this box. This method is declared by the [mln::Site_Set](#) concept.

Warning

This method is final for all box classes.

Definition at line 153 of file core/concept/box.hh.

Referenced by `mln::morpho::line_gradient()`.

10.84.3 Friends And Related Function Documentation

10.84.3.1 `template<typename SI , typename Sr > p_set< typename SI::site > diff (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Set theoretic difference of `lhs` and `rhs`.

Definition at line 66 of file `set/diff.hh`.

10.84.3.2 `template<typename SI , typename Sr > p_set< typename SI::site > inter (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Intersection between a couple of point sets.

Definition at line 62 of file `set/inter.hh`.

10.84.3.3 `template<typename SI , typename Sr > bool operator< (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Strict inclusion test between site sets `lhs` and `rhs`.

Parameters

[in] *lhs* A site set (strictly included?).

[in] *rhs* Another site set (includer?).

Definition at line 479 of file `operators.hh`.

10.84.3.4 `template<typename BI , typename Br > bool operator< (const Box< BI > & lhs, const Box< Br > & rhs) [related]`

Strict inclusion test between boxes `lhs` and `rhs`.

Parameters

[in] *lhs* A box (strictly included?).

[in] *rhs* Another box (includer?).

Definition at line 193 of file `core/concept/box.hh`.

10.84.3.5 `template<typename S > std::ostream & operator<< (std::ostream & ostr, const Site_Set< S > & set) [related, inherited]`

Print a site set `set` into the output stream `ostr`.

Parameters

[in, out] *ostr* An output stream.

[in] *set* A site set.

Returns

The modified output stream `ostr`.

Definition at line 505 of file `operators.hh`.

10.84.3.6 `template<typename Sl, typename Sr > bool operator<= (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Inclusion test between site sets `lhs` and `rhs`.

Parameters

[in] *lhs* A site set (included?).

[in] *rhs* Another site set (included?).

Definition at line 491 of file operators.hh.

10.84.3.7 `template<typename Bl, typename Br > bool operator<= (const Box< Bl > & lhs, const Box< Br > & rhs) [related]`

Inclusion test between boxes `lhs` and `rhs`.

Parameters

[in] *lhs* A box (included?).

[in] *rhs* Another box (included?).

Definition at line 178 of file core/concept/box.hh.

10.84.3.8 `template<typename Sl, typename Sr > bool operator== (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Equality test between site sets `lhs` and `rhs`.

Parameters

[in] *lhs* A site set.

[in] *rhs* Another site set.

Definition at line 467 of file operators.hh.

10.84.3.9 `template<typename Sl, typename Sr > p_set< typename Sl::site > sym_diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Set theoretic symmetrical difference of `lhs` and `rhs`.

Definition at line 65 of file sym_diff.hh.

10.84.3.10 `template<typename Sl, typename Sr > p_set< typename Sl::site > uni (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Union of a couple of point sets.

Definition at line 61 of file uni.hh.

10.84.3.11 `template<typename S > p_set< typename S::site > unique (const Site_Set< S > & s) [related, inherited]`

Give the unique set of *s*.

Definition at line 61 of file unique.hh.

10.85 mln::box_runend_piter< P > Class Template Reference

A generic backward iterator on points by lines.

```
#include <box_runend_piter.hh>
```

Inherits `site_set_iterator_base< box< P >, box_runend_piter< P > >`.

Public Member Functions

- `box_runend_piter` (const `box< P > &b`)

Constructor.

- void `next` ()

Go to the next element.

- unsigned `run_length` () const

Give the length of the run.

10.85.1 Detailed Description

```
template<typename P> class mln::box_runend_piter< P >
```

A generic backward iterator on points by lines. The parameter *P* is the type of points.

Definition at line 49 of file box_runend_piter.hh.

10.85.2 Constructor & Destructor Documentation

10.85.2.1 `template<typename P > mln::box_runend_piter< P >::box_runend_piter (const box< P > & b) [inline]`

Constructor.

Parameters

[in] *b* A box.

Definition at line 105 of file box_runend_piter.hh.

10.85.3 Member Function Documentation

10.85.3.1 void mln::Site_Iterator< box_runend_piter< P > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.85.3.2 template<typename P > unsigned mln::box_runend_piter< P >::run_length () const [inline]

Give the length of the run.

Definition at line 167 of file box_runend_piter.hh.

10.86 mln::box_runstart_piter< P > Class Template Reference

A generic forward iterator on points by lines.

```
#include <box_runstart_piter.hh>
```

Inherits site_set_iterator_base< box< P >, box_runstart_piter< P > >.

Public Member Functions

- [box_runstart_piter](#) (const [box](#)< P > &b)

Constructor.

- void [next](#) ()

Go to the next element.

- unsigned [run_length](#) () const

Give the length of the run.

10.86.1 Detailed Description

```
template<typename P> class mln::box_runstart_piter< P >
```

A generic forward iterator on points by lines. The parameter P is the type of points.

Definition at line 49 of file box_runstart_piter.hh.

10.86.2 Constructor & Destructor Documentation

10.86.2.1 `template<typename P > mln::box_runstart_piter< P >::box_runstart_piter (const box< P > & b) [inline]`

Constructor.

Parameters

[in] *b* A box.

Definition at line 105 of file box_runstart_piter.hh.

10.86.3 Member Function Documentation

10.86.3.1 `void mln::Site_Iterator< box_runstart_piter< P > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.86.3.2 `template<typename P > unsigned mln::box_runstart_piter< P >::run_length () const [inline]`

Give the lenght of the run.

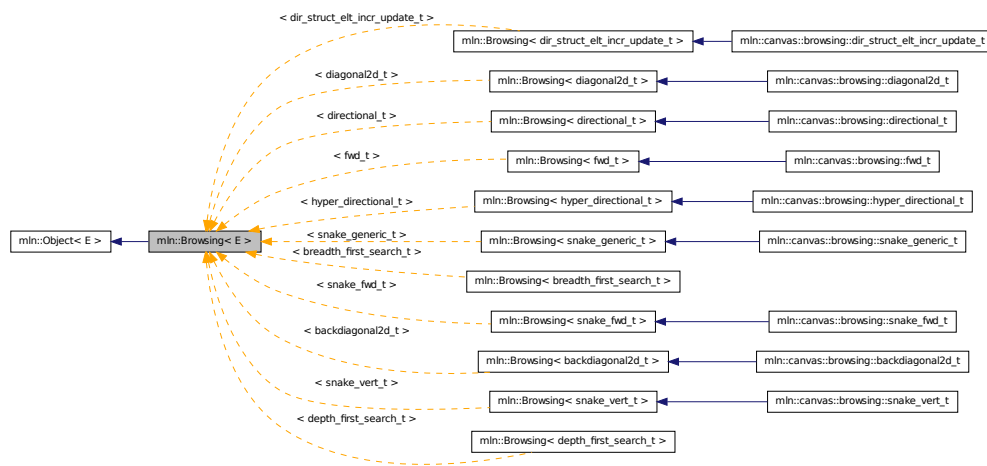
Definition at line 167 of file box_runstart_piter.hh.

10.87 mln::Browsing< E > Struct Template Reference

Base class for implementation classes that are browsings.

```
#include <browsing.hh>
```

Inheritance diagram for `mln::Browsing< E >`:



10.87.1 Detailed Description

`template<typename E> struct mln::Browsing< E >`

Base class for implementation classes that are browsings.

See also

`mln::doc::Browsing` for a complete documentation of this class contents.

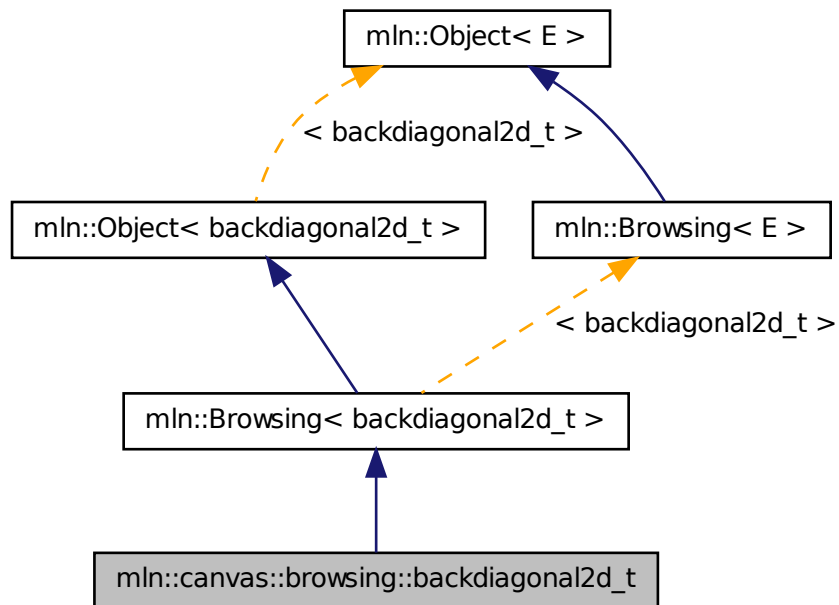
Definition at line 56 of file `browsing.hh`.

10.88 `mln::canvas::browsing::backdiagonal2d_t` Struct Reference

[Browsing](#) in a certain direction.

```
#include <backdiagonal2d.hh>
```

Inheritance diagram for mln::canvas::browsing::backdiagonal2d_t:



10.88.1 Detailed Description

Browsing in a certain direction. This canvas browse all the point of an image 'input' of type 'I' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'I', 'dim' and 'dir') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()`: Will be called at the end.

F shall features :

```

{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;

```

```

p;
--- as methods:
void init();
void next();
void final();
}

```

Example :

```
-----> | 4 7 9 | 2 5 8 | 1 3 6
```

Definition at line 83 of file backdiagonal2d.hh.

10.89 mln::canvas::browsing::breadth_first_search_t Struct Reference

Breadth-first search algorithm for graph, on vertices.

```
#include <breadth_first_search.hh>
```

Inherits graph_first_search_t< breadth_first_search_t, std::queue >.

10.89.1 Detailed Description

Breadth-first search algorithm for graph, on vertices.

Definition at line 79 of file breadth_first_search.hh.

10.90 mln::canvas::browsing::depth_first_search_t Struct Reference

Breadth-first search algorithm for graph, on vertices.

```
#include <depth_first_search.hh>
```

Inherits graph_first_search_t< depth_first_search_t, std::stack >.

10.90.1 Detailed Description

Breadth-first search algorithm for graph, on vertices.

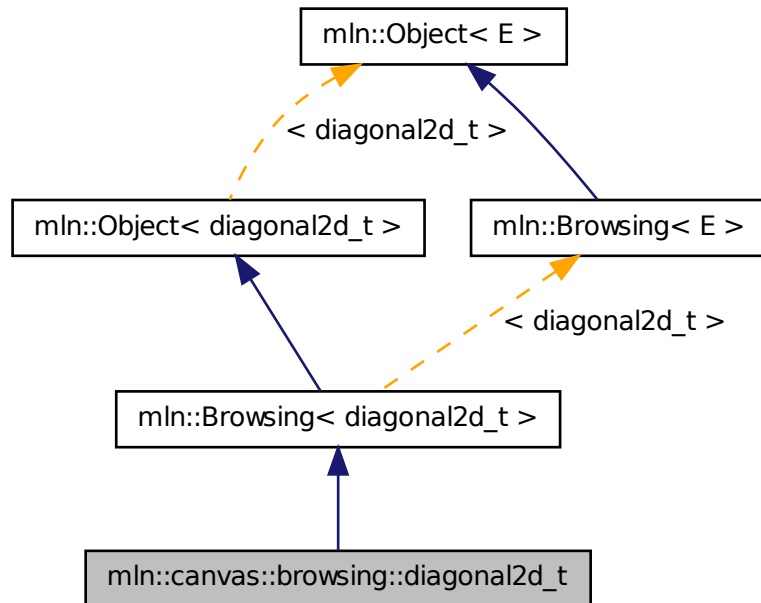
Definition at line 80 of file depth_first_search.hh.

10.91 mln::canvas::browsing::diagonal2d_t Struct Reference

[Browsing](#) in a certain direction.

```
#include <diagonal2d.hh>
```

Inheritance diagram for mln::canvas::browsing::diagonal2d_t:



10.91.1 Detailed Description

Browsing in a certain direction. This canvas browse all the point of an image 'input' of type 'I' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'I', 'dim' and 'dir') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```

{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;

```

```

p;
--- as methods:
void init();
void next();
void final();
}

```

Example :

```
| 1 3 6 | 2 5 8 | 4 7 9 L----->
```

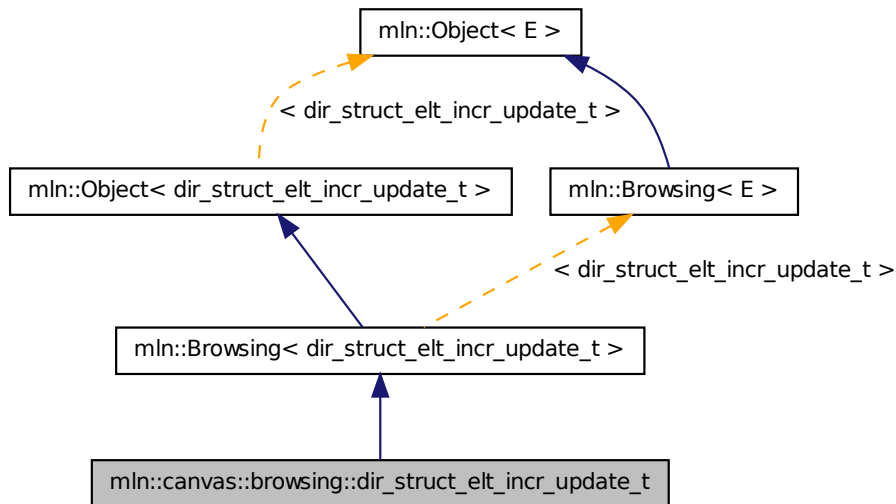
Definition at line 82 of file diagonal2d.hh.

10.92 mln::canvas::browsing::dir_struct_elt_incr_update_t Struct Reference

Browsing in a certain direction with a segment.

```
#include <dir_struct_elt_incr_update.hh>
```

Inheritance diagram for mln::canvas::browsing::dir_struct_elt_incr_update_t:



10.92.1 Detailed Description

Browsing in a certain direction with a segment. This canvas browse all the point of an image 'input' of type 'I', of dimension 'dim' in the direction 'dir' with considering weigh the 'length' nearest points.

The functor should provide (In addition to 'input', 'I', 'dim', 'dir' and 'length') six methods :

- `init()` : Will be called at the beginning.
- `init_line()` : Will be called at the beginning of each line.
- `add_point(q)` : Will be called for taking the new point 'q' into account.
- `remove_point(q)` : Will be called for untaking the new point 'q' into account.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```
{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;
p;
length;
--- as methods:
void init();
void init_line();
void add_point(q)
void remove_point(q)
void next();
void final();
}
```

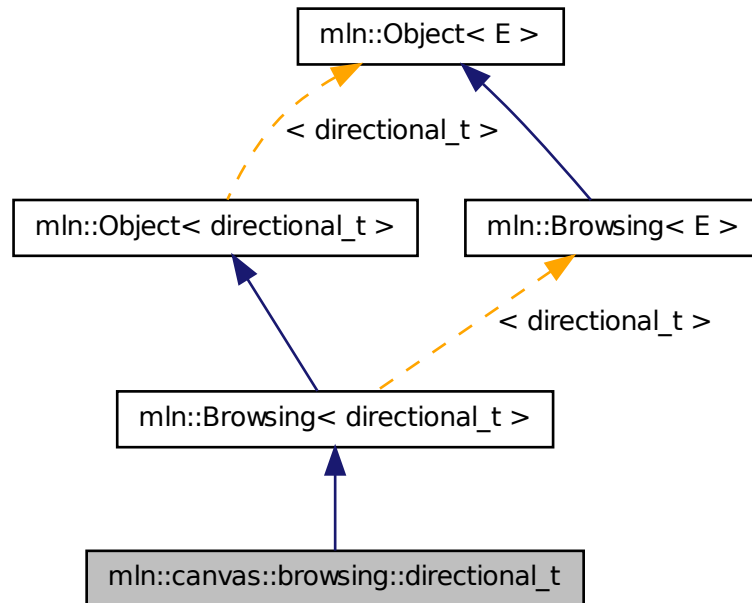
Definition at line 86 of file `dir_struct_elt_incr_update.hh`.

10.93 mln::canvas::browsing::directional_t Struct Reference

[Browsing](#) in a certain direction.

```
#include <directional.hh>
```

Inheritance diagram for `mIn::canvas::browsing::directional_t`:



10.93.1 Detailed Description

Browsing in a certain direction. This canvas browse all the point of an image 'input' of type 'I' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'I', 'dim' and 'dir') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```

{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;

```



```
p;  
--- as methods:  
void init();  
void next();  
void final();  
}
```

Example :

```
1 0 0 2 0 0 3 0 0  
4 0 0 5 0 0 6 0 0  
7 0 0 8 0 0 9 0 0
```

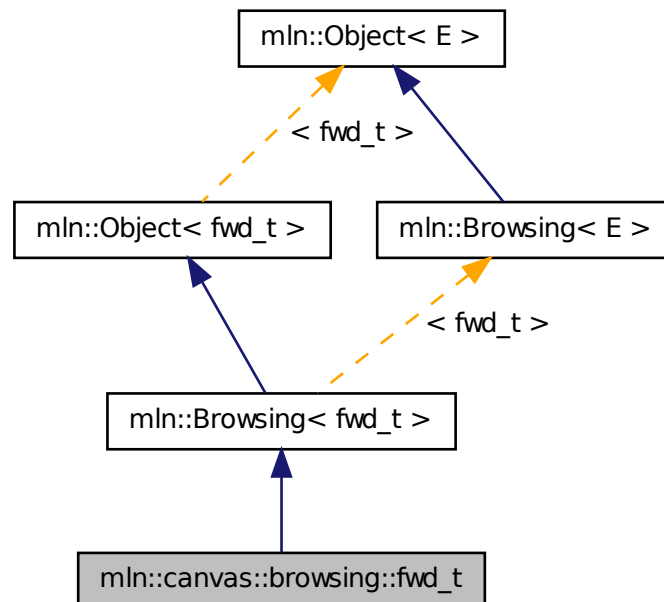
Definition at line 90 of file directional.hh.

10.94 mln::canvas::browsing::fwd_t Struct Reference

Canvas for forward browsing.

```
#include <fwd.hh>
```

Inheritance diagram for mln::canvas::browsing::fwd_t:



10.94.1 Detailed Description

Canvas for forward browsing. This canvas browse all the points of an image 'input' of type 'I' from left to right and from top to bottom

The functor should provide (In addition of 'I' and 'input') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall feature:

```
{  
--- as typedef:  
I;  
--as attributes:  
input;  
p;  
--- as method:  
void init();  
void next();  
void final();  
}
```

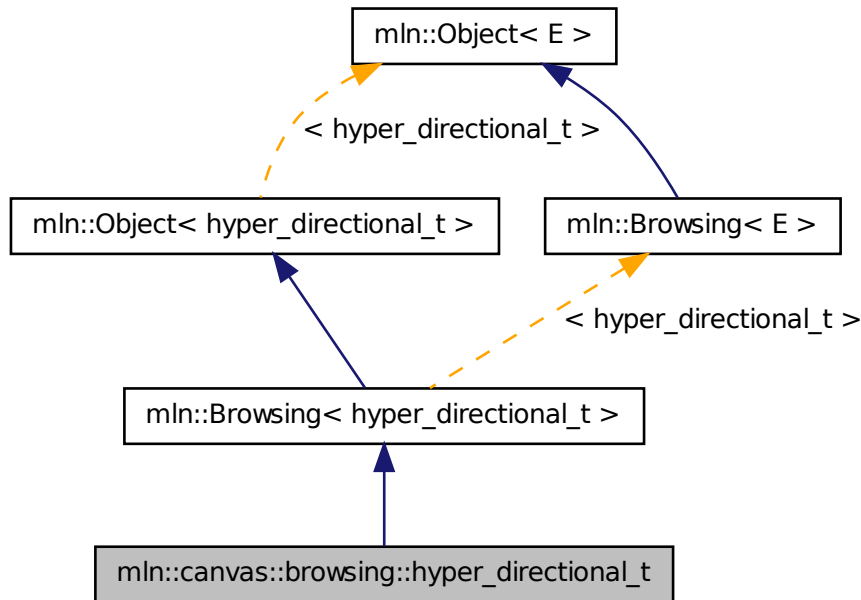
Definition at line 72 of file fwd.hh.

10.95 `mln::canvas::browsing::hyper_directional_t` Struct Reference

[Browsing](#) in a certain direction.

```
#include <hyper_directional.hh>
```

Inheritance diagram for mln::canvas::browsing::hyper_directional_t:



10.95.1 Detailed Description

Browsing in a certain direction. This canvas browse all the point of an image 'input' of type 'I' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'I', 'dim' and 'dir') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```

{
--- as types:
I;
--- as attributes:
dim;
dir; // and test dir < dim
input;

```

```

p;
--- as methods:
void init();
void next();
void final();
}

```

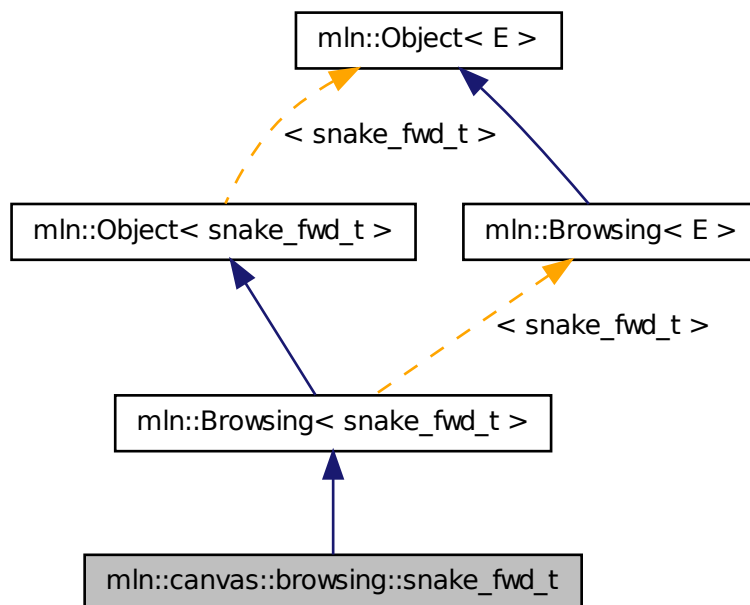
Definition at line 74 of file hyper_directional.hh.

10.96 mln::canvas::browsing::snake_fwd_t Struct Reference

Browsing in a snake-way, forward.

```
#include <snake_fwd.hh>
```

Inheritance diagram for mln::canvas::browsing::snake_fwd_t:



10.96.1 Detailed Description

Browsing in a snake-way, forward. This canvas browse all the point of an image 'input' like this :

```
-----> <-----' '----->
```

The functor should provide (In addition to 'input') four methods :

- `init()` : Will be called at the beginning.
- `down()` : Will be called after each moving down. (will also be called once at the first point).
- `fwd()` : Will be called after each moving right.
- `bwd()` : Will be called after each moving left.

This methods should access to the current working point 'p' also provided by the functor.

Warning: This canvas works only on 2D.

F shall feature:

```
{  
--- as attributes:  
input;  
p;  
--- as methods:  
void init();  
void down();  
void fwd();  
void bkd();  
}
```

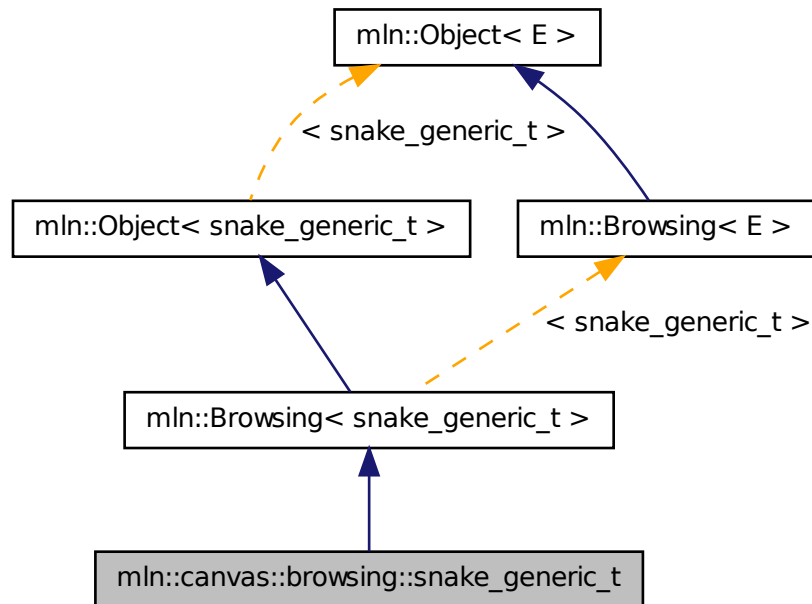
Definition at line 84 of file snake_fwd.hh.

10.97 mln::canvas::browsing::snake_generic_t Struct Reference

Multidimensional [Browsing](#) in a given-way.

```
#include <snake_generic.hh>
```

Inheritance diagram for mln::canvas::browsing::snake_generic_t:



10.97.1 Detailed Description

Multidimensional [Browsing](#) in a given-way. F shall feature:

```

{
--- as attributes:
input;
p;
--- as methods:
void init();
void *() moves[];
dpsite dps[];
}

```

init is called before browsing

The snake follow dimension using the delta point site of dps. dps[0] = delta psite following the global dimension (forward) dps[1] = delta psite following the 2nd dimension to follow (forward). dps[2] = delta psite following the 2nd dimension to follow (backward). dps[3] = delta psite following the 3rd dimension to follow (forward). dps[3] = delta psite following the 3rd dimension to follow (backward).

moves contains pointer to f's members. These members will be call in each time the snake progress in the

correct dimension :

moves[i] is called at each move following the delta psite dps[i]

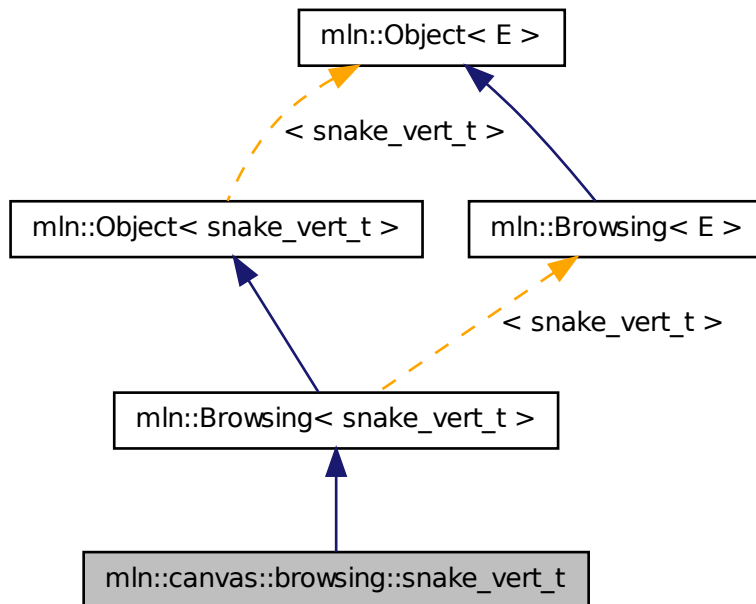
Definition at line 76 of file snake_generic.hh.

10.98 mln::canvas::browsing::snake_vert_t Struct Reference

Browsing in a snake-way, forward.

```
#include <snake_vert.hh>
```

Inheritance diagram for mln::canvas::browsing::snake_vert_t:



10.98.1 Detailed Description

Browsing in a snake-way, forward. This canvas browse all the point of an image 'input' like this :

```
| ^ | | | | \ | \
```

The functor should provide (In addition to 'input') four methods :

- `init()` : Will be called at the beginning.
- `down()` : Will be called after each moving down.
- `up()` : Will be called after each moving up.

- fwd() : Will be called after each moving right. (will also be called once at the first point).

This methods should acces to the current working point 'p' also provided by the functor.

Warning: This canvas works only on 2D.

F shall feature:

```
{
--- as attributes:
input;
p;
--- as methods:
void init();
void down();
void up();
void fwd();
}
```

Definition at line 83 of file snake_vert.hh.

10.99 mln::canvas::chamfer< F > Struct Template Reference

Compute chamfer distance.

```
#include <chamfer.hh>
```

10.99.1 Detailed Description

```
template<typename F> struct mln::canvas::chamfer< F >
```

Compute chamfer distance.

Definition at line 47 of file canvas/chamfer.hh.

10.100 mln::category< R(*)(&A) > Struct Template Reference

Category declaration for a unary C function.

```
#include <c.hh>
```

10.100.1 Detailed Description

```
template<typename R, typename A> struct mln::category< R(*)(&A) >
```

Category declaration for a unary C function.

Definition at line 51 of file c.hh.

10.101 mln::complex_image< D, G, V > Class Template Reference

[Image](#) based on a complex.

```
#include <complex_image.hh>
```

Inherits [image_primary](#)< V, p_complex< D, G >, [complex_image](#)< D, G, V > >.

Public Types

- typedef G [geom](#)
The geometry type of the complex.
- typedef V & [lvalue](#)
Return type of read-write access.
- typedef const V & [rvalue](#)
Return type of read-only access.
- typedef [complex_image](#)< D, tag::psite_< G >, tag::value_< V > > [skeleton](#)
Skeleton.
- typedef V [value](#)
Value associated type.

Public Member Functions

- [rvalue operator\(\)](#) (const [complex_psite](#)< D, G > &p) const
Read-only access of face value at point site p.
- [lvalue operator\(\)](#) (const [complex_psite](#)< D, G > &p)
Read-write access of face value at point site p.
- [complex_image](#) ()
Constructors.
- const [p_complex](#)< D, G > & [domain](#) () const
Accessors.
- const [metal::vec](#)< D+1, [std::vector](#)< [mlc_unbool](#)(V) > > & [values](#) () const
Return the array of values associated to the faces.

Static Public Attributes

- static const unsigned [dim](#) = D
The dimension of the complex.

10.101.1 Detailed Description

`template<unsigned D, typename G, typename V> class mln::complex_image< D, G, V >`

[Image](#) based on a complex. Values attached to each face of the complex.

Template Parameters

- D* The dimension of the complex.
- G* The geometry type of the complex.
- V* The value type of the image.

Definition at line 164 of file mln/core/image/complex_image.hh.

10.101.2 Member Typedef Documentation

10.101.2.1 `template<unsigned D, typename G, typename V> typedef G mln::complex_image< D, G, V >::geom`

The geometry type of the complex.

Definition at line 172 of file mln/core/image/complex_image.hh.

10.101.2.2 `template<unsigned D, typename G, typename V> typedef V& mln::complex_image< D, G, V >::lvalue`

Return type of read-write access.

Definition at line 177 of file mln/core/image/complex_image.hh.

10.101.2.3 `template<unsigned D, typename G, typename V> typedef const V& mln::complex_image< D, G, V >::rvalue`

Return type of read-only access.

Definition at line 180 of file mln/core/image/complex_image.hh.

10.101.2.4 `template<unsigned D, typename G, typename V> typedef complex_image< D, tag::psite_<G>, tag::value_<V> > mln::complex_image< D, G, V >::skeleton`

Skeleton.

Definition at line 183 of file mln/core/image/complex_image.hh.

10.101.2.5 `template<unsigned D, typename G, typename V> typedef V mln::complex_image< D, G, V >::value`

[Value](#) associated type.

Definition at line 174 of file mln/core/image/complex_image.hh.

10.101.3 Constructor & Destructor Documentation

10.101.3.1 `template<unsigned D, typename G, typename V > mln::complex_image< D, G, V >::complex_image() [inline]`

Constructors.

Definition at line 276 of file mln/core/image/complex_image.hh.

10.101.4 Member Function Documentation

10.101.4.1 `template<unsigned D, typename G, typename V > const p_complex< D, G > & mln::complex_image< D, G, V >::domain() const [inline]`

Accessors.

Return the domain of psites of the image.

Definition at line 343 of file mln/core/image/complex_image.hh.

10.101.4.2 `template<unsigned D, typename G, typename V > complex_image< D, G, V >::lvalue mln::complex_image< D, G, V >::operator() (const complex_psite< D, G > & p) [inline]`

Read-write access of face value at point site p.

Definition at line 326 of file mln/core/image/complex_image.hh.

References mln::complex_psite< D, G >::face_id(), and mln::complex_psite< D, G >::n().

10.101.4.3 `template<unsigned D, typename G, typename V > complex_image< D, G, V >::rvalue mln::complex_image< D, G, V >::operator() (const complex_psite< D, G > & p) const [inline]`

Read-only access of face value at point site p.

Definition at line 317 of file mln/core/image/complex_image.hh.

References mln::complex_psite< D, G >::face_id(), and mln::complex_psite< D, G >::n().

10.101.4.4 `template<unsigned D, typename G, typename V > const metal::vec< D+1, std::vector< mlc_unbool(V) > > & mln::complex_image< D, G, V >::values() const [inline]`

Return the array of values associated to the faces.

Definition at line 335 of file mln/core/image/complex_image.hh.

10.101.5 Member Data Documentation

10.101.5.1 `template<unsigned D, typename G, typename V > const unsigned mln::complex_image< D, G, V >::dim = D [static]`

The dimension of the complex.

Definition at line 170 of file mln/core/image/complex_image.hh.

10.102 mln::complex_neighborhood_bkd_piter< I, G, N > Class Template Reference

Backward iterator on complex neighborhood.

```
#include <complex_neighborhood_piter.hh>
```

Inherits site_relative_iterator_base< N, complex_neighborhood_bkd_piter< I, G, N > >.

Public Types

- typedef N::complex_bkd_iter [iter_type](#)
The type of the underlying complex iterator.
- typedef N::psite [psite](#)
The [Pseudo_Site](#) type.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [complex_neighborhood_bkd_piter](#) ()
Construction.
- const [iter_type](#) & [iter](#) () const
Accessors.

10.102.1 Detailed Description

```
template<typename I, typename G, typename N> class mln::complex_neighborhood_bkd_piter< I, G, N >
```

Backward iterator on complex neighborhood.

Definition at line 124 of file complex_neighborhood_piter.hh.

10.102.2 Member Typedef Documentation

10.102.2.1 `template<typename I, typename G, typename N> typedef N::complex_bkd_iter mln::complex_neighborhood_bkd_piter< I, G, N >::iter_type`

The type of the underlying complex iterator.

Definition at line 135 of file complex_neighborhood_piter.hh.

10.102.2.2 `template<typename I, typename G, typename N> typedef N ::psite
mln::complex_neighborhood_bkd_piter< I, G, N >::psite`

The [Pseudo_Site](#) type.

Definition at line 133 of file `complex_neighborhood_piter.hh`.

10.102.3 Constructor & Destructor Documentation

10.102.3.1 `template<typename I , typename G , typename N > mln::complex_-
neighborhood_bkd_piter< I, G, N >::complex_neighborhood_bkd_piter ()
[inline]`

Construction.

Definition at line 305 of file `complex_neighborhood_piter.hh`.

10.102.4 Member Function Documentation

10.102.4.1 `template<typename I , typename G , typename N > const N::complex_bkd_iter &
mln::complex_neighborhood_bkd_piter< I, G, N >::iter () const [inline]`

Accessors.

Definition at line 382 of file `complex_neighborhood_piter.hh`.

10.102.4.2 `void mln::Site_Iterator< complex_neighborhood_bkd_piter< I, G, N > >::next ()
[inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.103 mln::complex_neighborhood_fwd_piter< I, G, N > Class Template Reference

Forward iterator on complex neighborhood.

```
#include <complex_neighborhood_piter.hh>
```

Inherits `site_relative_iterator_base< N, complex_neighborhood_fwd_piter< I, G, N > >`.

Public Types

- `typedef N::complex_fwd_iter` [iter_type](#)

The type of the underlying complex iterator.

- typedef N::psite [psite](#)
The *Pseudo_Site* type.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [complex_neighborhood_fwd_piter](#) ()
Construction.
- const [iter_type](#) & [iter](#) () const
Accessors.

10.103.1 Detailed Description

template<typename I, typename G, typename N> class mln::complex_neighborhood_fwd_piter< I, G, N >

Forward iterator on complex neighborhood.

Definition at line 53 of file complex_neighborhood_piter.hh.

10.103.2 Member Typedef Documentation

10.103.2.1 template<typename I, typename G, typename N> typedef N::complex_fwd_iter mln::complex_neighborhood_fwd_piter< I, G, N >::iter_type

The type of the underlying complex iterator.

Definition at line 64 of file complex_neighborhood_piter.hh.

10.103.2.2 template<typename I, typename G, typename N> typedef N ::psite mln::complex_neighborhood_fwd_piter< I, G, N >::psite

The [Pseudo_Site](#) type.

Definition at line 62 of file complex_neighborhood_piter.hh.

10.103.3 Constructor & Destructor Documentation

**10.103.3.1 template<typename I , typename G , typename N > mln::complex_neighborhood_fwd_piter< I, G, N >::complex_neighborhood_fwd_piter ()
[inline]**

Construction.

Definition at line 198 of file complex_neighborhood_piter.hh.

10.103.4 Member Function Documentation

10.103.4.1 `template<typename I , typename G , typename N > const N::complex_fwd_iter & mln::complex_neighborhood_fwd_piter< I, G, N >::iter () const [inline]`

Accessors.

Definition at line 275 of file complex_neighborhood_piter.hh.

10.103.4.2 `void mln::Site_Iterator< complex_neighborhood_fwd_piter< I, G, N > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.104 mln::complex_psite< D, G > Class Template Reference

[Point](#) site associated to a [mln::p_complex](#).

```
#include <complex_psite.hh>
```

Inherits `pseudo_site_base_< const G::site &, complex_psite< D, G > >`.

Public Member Functions

- `complex_psite ()`
Construction and assignment.
- `complex_psite (const p_complex< D, G > &pc, const topo::face< D > &face)`
- `bool is_valid () const`
Psite manipulators.
- `void invalidate ()`
Invalidate this psite.
- `const target & site_set () const`
Site set manipulators.
- `void change_target (const target &new_target)`
Set the target site_set.

- const `topo::face< D > & face ()` const
Face handle manipulators.
- unsigned `n ()` const
Return the dimension of the face of this psite.
- unsigned `face_id ()` const
Return the id of the face of this psite.

10.104.1 Detailed Description

`template<unsigned D, typename G> class mln::complex_psite< D, G >`

Point site associated to a `mln::p_complex`.

Template Parameters

- D* The dimension of the complex this psite belongs to.
- G* The geometry of the complex.

Definition at line 60 of file `complex_psite.hh`.

10.104.2 Constructor & Destructor Documentation

10.104.2.1 `template<unsigned D, typename G > mln::complex_psite< D, G >::complex_psite ()` [`inline`]

Construction and assignment.

Definition at line 203 of file `complex_psite.hh`.

References `mln::complex_psite< D, G >::invalidate()`.

10.104.2.2 `template<unsigned D, typename G > mln::complex_psite< D, G >::complex_psite (const p_complex< D, G > & pc, const topo::face< D > & face)` [`inline`]

Precondition

`pc.cplx() == face.cplx()`.

Definition at line 211 of file `complex_psite.hh`.

References `mln::topo::face< D >::cplx()`, `mln::p_complex< D, G >::cplx()`, and `mln::complex_psite< D, G >::is_valid()`.

10.104.3 Member Function Documentation

10.104.3.1 `template<unsigned D, typename G > void mln::complex_psite< D, G >::change_target (const target & new_target)` [`inline`]

Set the target site_set.

Definition at line 280 of file complex_psite.hh.

References mln::p_complex< D, G >::cplx(), and mln::complex_psite< D, G >::invalidate().

10.104.3.2 `template<unsigned D, typename G > const topo::face< D > & mln::complex_psite< D, G >::face () const [inline]`

Face handle manipulators.

Return the face handle of this point site.

Definition at line 301 of file complex_psite.hh.

Referenced by mln::operator!=(), and mln::operator==().

10.104.3.3 `template<unsigned D, typename G > unsigned mln::complex_psite< D, G >::face_id () const [inline]`

Return the id of the face of this psite.

Definition at line 317 of file complex_psite.hh.

Referenced by mln::complex_image< D, G, V >::operator()().

10.104.3.4 `template<unsigned D, typename G > void mln::complex_psite< D, G >::invalidate () [inline]`

Invalidate this psite.

Definition at line 251 of file complex_psite.hh.

Referenced by mln::complex_psite< D, G >::change_target(), and mln::complex_psite< D, G >::complex_psite().

10.104.3.5 `template<unsigned D, typename G > bool mln::complex_psite< D, G >::is_valid () const [inline]`

Psite manipulators.

Is this psite valid?

Definition at line 239 of file complex_psite.hh.

Referenced by mln::complex_psite< D, G >::complex_psite(), and mln::p_complex< D, G >::has().

10.104.3.6 `template<unsigned D, typename G > unsigned mln::complex_psite< D, G >::n () const [inline]`

Return the dimension of the face of this psite.

Definition at line 309 of file complex_psite.hh.

Referenced by mln::make::cell(), and mln::complex_image< D, G, V >::operator()().

10.104.3.7 `template<unsigned D, typename G > const p_complex< D, G > & mln::complex_psite< D, G >::site_set() const [inline]`

[Site](#) set manipulators.

Return the [mln::p_complex](#) this site is built on. (shortcut for `*target()`).

Precondition

Member `face_` is valid.

Definition at line 259 of file `complex_psite.hh`.

Referenced by `mln::p_complex< D, G >::has()`, `mln::operator!=()`, and `mln::operator==()`.

10.105 `mln::complex_window_bkd_piter< I, G, W >` Class Template Reference

Backward iterator on complex window.

```
#include <complex_window_piter.hh>
```

Inherits `site_relative_iterator_base< W, complex_window_bkd_piter< I, G, W > >`.

Public Types

- typedef `W::complex_bkd_iter` [iter_type](#)
The type of the underlying complex iterator.
- typedef `W::psite` [psite](#)
The [Pseudo_Site](#) type.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [complex_window_bkd_piter](#) ()
Construction.
- const [iter_type](#) & [iter](#) () const
Accessors.

10.105.1 Detailed Description

```
template<typename I, typename G, typename W> class mln::complex_window_bkd_piter< I, G, W >
```

Backward iterator on complex window.

Definition at line 124 of file complex_window_piter.hh.

10.105.2 Member Typedef Documentation

10.105.2.1 `template<typename I, typename G, typename W> typedef W::complex_bkd_iter
mln::complex_window_bkd_piter< I, G, W >::iter_type`

The type of the underlying complex iterator.

Definition at line 135 of file complex_window_piter.hh.

10.105.2.2 `template<typename I, typename G, typename W> typedef W ::psite
mln::complex_window_bkd_piter< I, G, W >::psite`

The [Pseudo_Site](#) type.

Definition at line 133 of file complex_window_piter.hh.

10.105.3 Constructor & Destructor Documentation

10.105.3.1 `template<typename I , typename G , typename W > mln::complex_
window_bkd_piter< I, G, W >::complex_window_bkd_piter ()
[inline]`

Construction.

Definition at line 305 of file complex_window_piter.hh.

10.105.4 Member Function Documentation

10.105.4.1 `template<typename I , typename G , typename W > const W::complex_bkd_iter &
mln::complex_window_bkd_piter< I, G, W >::iter () const [inline]`

Accessors.

Definition at line 384 of file complex_window_piter.hh.

10.105.4.2 `void mln::Site_Iterator< complex_window_bkd_piter< I, G, W > >::next ()
[inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.106 mln::complex_window_fwd_piter< I, G, W > Class Template Reference

Forward iterator on complex window.

```
#include <complex_window_piter.hh>
```

Inherits `site_relative_iterator_base< W, complex_window_fwd_piter< I, G, W > >`.

Public Types

- typedef `W::complex_fwd_iter` `iter_type`
The type of the underlying complex iterator.
- typedef `W::psite` `psite`
The `Pseudo_Site` type.

Public Member Functions

- void `next ()`
Go to the next element.
- `complex_window_fwd_piter ()`
Construction.
- const `iter_type & iter () const`
Accessors.

10.106.1 Detailed Description

```
template<typename I, typename G, typename W> class mln::complex_window_fwd_piter< I, G, W >
```

Forward iterator on complex window.

Definition at line 54 of file `complex_window_piter.hh`.

10.106.2 Member Typedef Documentation

10.106.2.1 `template<typename I, typename G, typename W> typedef W::complex_fwd_iter mln::complex_window_fwd_piter< I, G, W >::iter_type`

The type of the underlying complex iterator.

Definition at line 65 of file `complex_window_piter.hh`.

10.106.2.2 `template<typename I, typename G, typename W> typedef W ::psite
mln::complex_window_fwd_piter< I, G, W >::psite`

The [Pseudo_Site](#) type.

Definition at line 63 of file complex_window_piter.hh.

10.106.3 Constructor & Destructor Documentation

10.106.3.1 `template<typename I , typename G , typename W > mln::complex_
window_fwd_piter< I, G, W >::complex_window_fwd_piter ()
[inline]`

Construction.

Definition at line 197 of file complex_window_piter.hh.

10.106.4 Member Function Documentation

10.106.4.1 `template<typename I , typename G , typename W > const W::complex_fwd_iter &
mln::complex_window_fwd_piter< I, G, W >::iter () const [inline]`

Accessors.

Definition at line 275 of file complex_window_piter.hh.

10.106.4.2 `void mln::Site_Iterator< complex_window_fwd_piter< I, G, W > >::next ()
[inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.107 mln::decorated_image< I, D > Struct Template Reference

[Image](#) that can have additional features.

```
#include <decorated_image.hh>
```

Inherits `decorated_image_impl< I, decorated_image< I, D > >`, and `image_identity< I, I::domain_t, decorated_image< I, D > >`.

Public Types

- typedef [impl_::lvalue](#) lvalue

Return type of read-write access.

- typedef I::psite [psite](#)
Type of the psite.
- typedef I::rvalue [rvalue](#)
Return type of read-only access.
- typedef [decorated_image](#)< tag::image_< I >, tag::data_< D > > [skeleton](#)
Skeleton.

Public Member Functions

- [decorated_image](#) ()
Ctors.
- D & [decoration](#) ()
Give the decoration.
- const D & [decoration](#) () const
Give the decoration.
- [operator decorated_image](#)< const I, D > () const
Const promotion via conversion.
- [rvalue operator](#)() (const [psite](#) &p) const
Read-only access of pixel value at point site p.
- [lvalue operator](#)() (const [psite](#) &p)
Read-write access of pixel value at point site p.
- [~decorated_image](#) ()
Dtor.

10.107.1 Detailed Description

`template<typename I, typename D> struct mln::decorated_image< I, D >`

[Image](#) that can have additional features.

Definition at line 81 of file `decorated_image.hh`.

10.107.2 Member Typedef Documentation

10.107.2.1 `template<typename I, typename D> typedef impl_::lvalue mln::decorated_image< I, D >::lvalue`

Return type of read-write access.

Definition at line 95 of file decorated_image.hh.

10.107.2.2 `template<typename I, typename D> typedef I ::psite mln::decorated_image< I, D >::psite`

Type of the psite.

Definition at line 90 of file decorated_image.hh.

10.107.2.3 `template<typename I, typename D> typedef I ::rvalue mln::decorated_image< I, D >::rvalue`

Return type of read-only access.

Definition at line 93 of file decorated_image.hh.

10.107.2.4 `template<typename I, typename D> typedef decorated_image< tag::image_<I>, tag::data_<D> > mln::decorated_image< I, D >::skeleton`

Skeleton.

Definition at line 108 of file decorated_image.hh.

10.107.3 Constructor & Destructor Documentation

10.107.3.1 `template<typename I , typename D > mln::decorated_image< I, D >::decorated_image () [inline]`

Ctors.

Definition at line 161 of file decorated_image.hh.

10.107.3.2 `template<typename I , typename D > mln::decorated_image< I, D >::~~decorated_image () [inline]`

Dtor.

Definition at line 184 of file decorated_image.hh.

10.107.4 Member Function Documentation

10.107.4.1 `template<typename I , typename D > const D & mln::decorated_image< I, D >::decoration () const [inline]`

Give the decoration.

Definition at line 249 of file decorated_image.hh.

10.107.4.2 `template<typename I , typename D > D & mln::decorated_image< I, D >::decoration () [inline]`

Give the decoration.

Definition at line 257 of file decorated_image.hh.

10.107.4.3 `template<typename I , typename D > mln::decorated_image< I, D >::operator decorated_image< const I, D > () const [inline]`

Const promotion via conversion.

Definition at line 239 of file decorated_image.hh.

10.107.4.4 `template<typename I , typename D > decorated_image< I, D >::rvalue mln::decorated_image< I, D >::operator() (const psite & p) const [inline]`

Read-only access of pixel value at point site p.

Definition at line 197 of file decorated_image.hh.

10.107.4.5 `template<typename I , typename D > decorated_image< I, D >::lvalue mln::decorated_image< I, D >::operator() (const psite & p) [inline]`

Read-write access of pixel value at point site p.

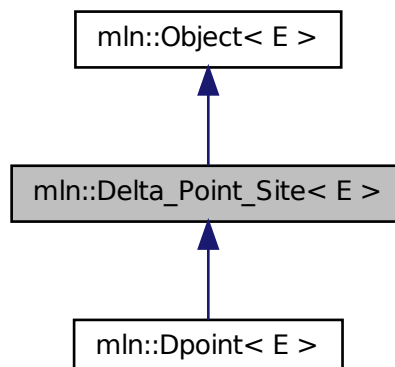
Definition at line 206 of file decorated_image.hh.

10.108 mln::Delta_Point_Site< E > Struct Template Reference

FIXME: Doc!

```
#include <delta_point_site.hh>
```

Inheritance diagram for mln::Delta_Point_Site< E >:



10.108.1 Detailed Description

```
template<typename E> struct mln::Delta_Point_Site< E >
```

FIXME: Doc!

Definition at line 79 of file delta_point_site.hh.

10.109 mln::Delta_Point_Site< void > Struct Template Reference

Delta point site category flag type.

```
#include <delta_point_site.hh>
```

10.109.1 Detailed Description

```
template<> struct mln::Delta_Point_Site< void >
```

Delta point site category flag type.

Definition at line 70 of file delta_point_site.hh.

10.110 mln::doc::Accumulator< E > Struct Template Reference

Documentation class for [mln::Accumulator](#).

```
#include <accumulator.hh>
```

Public Types

- typedef void [argument](#)

The argument type of elements to accumulate.

Public Member Functions

- void [init](#) ()

Initialize the accumulator.

- void [take](#) (const E &other)

Take into account another accumulator `other`.

- void [take](#) (const [argument](#) &t)

Take into account a argument `t` (an element).

10.110.1 Detailed Description

`template<typename E> struct mln::doc::Accumulator< E >`

Documentation class for [mln::Accumulator](#).

See also

[mln::Accumulator](#)

Definition at line 36 of file doc/accumulator.hh.

10.110.2 Member Typedef Documentation

10.110.2.1 `template<typename E > typedef void mln::doc::Accumulator< E >::argument`

The argument type of elements to accumulate.

Definition at line 39 of file doc/accumulator.hh.

10.110.3 Member Function Documentation

10.110.3.1 `template<typename E > void mln::doc::Accumulator< E >::init ()`

Initialize the accumulator.

10.110.3.2 `template<typename E > void mln::doc::Accumulator< E >::take (const E & other)`

Take into account another accumulator `other`.

10.110.3.3 `template<typename E > void mln::doc::Accumulator< E >::take (const argument & t)`

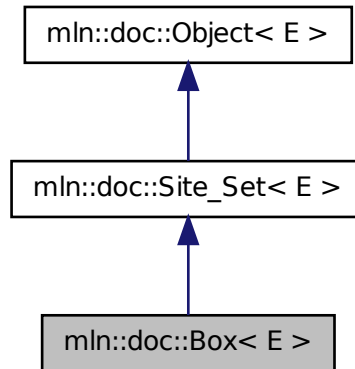
Take into account a argument `t` (an element).

10.111 mln::doc::Box< E > Struct Template Reference

Documentation class for [mln::Box](#).

```
#include <box.hh>
```

Inheritance diagram for mln::doc::Box< E >:



Public Types

- typedef void `bkd_piter`
Backward `Site_Iterator` associated type.
- typedef void `fwd_piter`
Forward `Site_Iterator` associated type.
- typedef void `psite`
PSite associated type.
- typedef void `site`
Site associated type.

Public Member Functions

- const E & `bbox` () const
Return the bounding box of this point set.
- bool `has` (const `psite` &p) const
Test if `p` belongs to this site set.
- unsigned `nsites` () const
Return the number of points of this box.
- const `site` & `pmax` () const
Give the box "maximum" point.

- const [site](#) & [pmin](#) () const
Give the box "minimum" point.

10.111.1 Detailed Description

template<typename E> struct mln::doc::Box< E >

Documentation class for [mln::Box](#).

See also

[mln::Box](#)

Definition at line 36 of file core/concept/doc/box.hh.

10.111.2 Member Typedef Documentation

10.111.2.1 template<typename E > typedef void mln::doc::Site_Set< E >::bkd_piter [inherited]

Backward [Site_Iterator](#) associated type.

Definition at line 53 of file mln/core/concept/doc/site_set.hh.

10.111.2.2 template<typename E > typedef void mln::doc::Site_Set< E >::fwd_piter [inherited]

Forward [Site_Iterator](#) associated type.

Definition at line 49 of file mln/core/concept/doc/site_set.hh.

10.111.2.3 template<typename E > typedef void mln::doc::Site_Set< E >::psite [inherited]

PSite associated type.

Definition at line 45 of file mln/core/concept/doc/site_set.hh.

10.111.2.4 template<typename E > typedef void mln::doc::Site_Set< E >::site [inherited]

[Site](#) associated type.

Definition at line 41 of file mln/core/concept/doc/site_set.hh.

10.111.3 Member Function Documentation

10.111.3.1 template<typename E > const E& mln::doc::Box< E >::bbox () const

Return the bounding box of this point set.

Return the bounding box of this point set, so that is itself. This method is declared by the [mln::Site_Set](#) concept.

Warning

This method is final for all box classes.

10.111.3.2 `template<typename E> bool mln::doc::Site_Set< E >::has (const psite & p) const`
[inherited]

Test if `p` belongs to this site set.

Parameters

[in] `p` A site.

Returns

True if `p` is an element of the site set.

10.111.3.3 `template<typename E> unsigned mln::doc::Box< E >::nsites () const`

Return the number of points of this box.

Return the number of points of this box. This method is declared by the `mln::Site_Set` concept.

Warning

This method is final for all box classes.

10.111.3.4 `template<typename E> const site& mln::doc::Box< E >::pmax () const`

Give the box "maximum" point.

Return the "maximum" point w.r.t. the ordering between points. For instance, with `mln::box2d`, this maximum is the bottom right point of the box.

10.111.3.5 `template<typename E> const site& mln::doc::Box< E >::pmin () const`

Give the box "minimum" point.

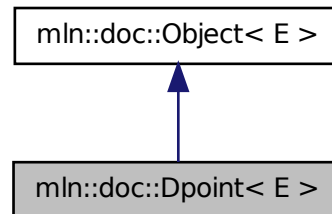
Return the "minimum" point w.r.t. the ordering between points. For instance, with `mln::box2d`, this minimum is the top left point of the box.

10.112 `mln::doc::Dpoint< E >` Struct Template Reference

Documentation class for `mln::Dpoint`.

```
#include <dpoint.hh>
```

Inheritance diagram for `mln::doc::Dpoint< E >`:



Public Types

- enum { [dim](#) }
- typedef void [coord](#)
- typedef void [dpoint](#)
Dpsite associated type.
- typedef void [point](#)
Site associated type.

Public Member Functions

- [coord operator\[\]](#) (unsigned i) const
*Read-only access to the *i*-th coordinate value.*

10.112.1 Detailed Description

`template<typename E> struct mln::doc::Dpoint< E >`

Documentation class for [mln::Dpoint](#).

See also

[mln::Dpoint](#)

Definition at line 36 of file `concept/doc/dpoint.hh`.

10.112.2 Member Typedef Documentation

10.112.2.1 `template<typename E > typedef void mln::doc::Dpoint< E >::coord`

Coordinate associated type.

Definition at line 56 of file concept/doc/dpoint.hh.

10.112.2.2 `template<typename E > typedef void mln::doc::Dpoint< E >::dpoint`

Dpsite associated type.

Invariant

This type has to derive from [mln::Dpoint](#).

Definition at line 52 of file concept/doc/dpoint.hh.

10.112.2.3 `template<typename E > typedef void mln::doc::Dpoint< E >::point`

[Site](#) associated type.

Invariant

This type has to derive from [mln::Point](#).

Definition at line 47 of file concept/doc/dpoint.hh.

10.112.3 Member Enumeration Documentation

10.112.3.1 `template<typename E > anonymous enum`

Enumerator:

dim Dimension of the space.

Invariant

$dim > 0$

Definition at line 42 of file concept/doc/dpoint.hh.

10.112.4 Member Function Documentation

10.112.4.1 `template<typename E > coord mln::doc::Dpoint< E >::operator[] (unsigned i) const`

Read-only access to the *i*-th coordinate value.

Parameters

[in] *i* The coordinate index.

Precondition

$i < dim$

Returns

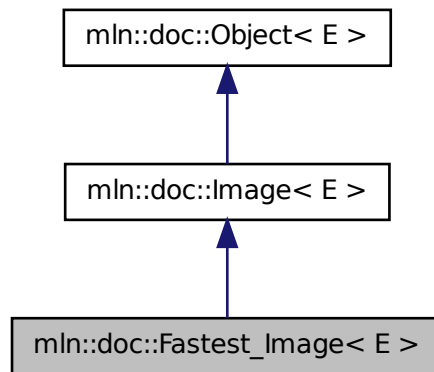
The value of the *i*-th coordinate.

10.113 mln::doc::Fastest_Image< E > Struct Template Reference

Documentation class for the concept of images that have the speed property set to "fastest".

```
#include <image_fastest.hh>
```

Inheritance diagram for mln::doc::Fastest_Image< E >:



Public Types

- typedef void [bkd_piter](#)
Backward point iterator associated type.
- typedef void [coord](#)
Coordinate associated type.
- typedef void [dpoint](#)
Dpsite associated type.
- typedef void [fwd_piter](#)
Forward point iterator associated type.
- typedef void [lvalue](#)
Type returned by the read-write pixel value operator.
- typedef void [point](#)
Site associated type.
- typedef void [pset](#)
Point set associated type.
- typedef void [psite](#)

Point_Site associated type.

- typedef void [rvalue](#)
Type returned by the read pixel value operator.
- typedef void [skeleton](#)
Associate type that describes how this type of image is constructed.
- typedef void [value](#)
Value associated type.
- typedef void [vset](#)
Value set associated type.

Public Member Functions

- const [box](#)< [point](#) > & [bbox](#) () const
Give a bounding box of the image domain.
- unsigned [border](#) ()
Give the border thickness.
- const [value](#) * [buffer](#) () const
Give a hook to the value buffer.
- int [delta_index](#) (const [dpoint](#) &dp)
Give the offset corresponding to the delta-point dp.
- const [pset](#) & [domain](#) () const
Give the definition domain of the image.
- bool [has](#) (const [psite](#) &p) const
Test if the image owns the point site p.
- bool [has](#) (const [psite](#) &p) const
Test if p belongs to the image domain.
- bool [is_valid](#) () const
Test if the image have been initialized.
- unsigned [nelements](#) () const
Give the number of pixels of the image including those of the virtual border.
- unsigned [nsites](#) () const
Give the number of points of the image domain.
- [lvalue operator](#)() (const [psite](#) &p)
Read-write access to the image value located at p.

- [rvalue operator\(\)](#) (const [psite](#) &p) const
Read-only access to the image value located at p.
- [rvalue operator\[\]](#) (unsigned o) const
Read-only access to the image value at offset o.
- [lvalue operator\[\]](#) (unsigned o)
Read-write access to the image value at offset o.
- [point point_at_index](#) (unsigned o) const
Give the point at offset o.
- const [vset](#) & [values](#) () const
Give the set of values of the image.

10.113.1 Detailed Description

template<typename E> struct mln::doc::Fastest_Image< E >

Documentation class for the concept of images that have the speed property set to "fastest".

Definition at line 36 of file concept/doc/image_fastest.hh.

10.113.2 Member Typedef Documentation

10.113.2.1 template<typename E > typedef void mln::doc::Image< E >::bkd_piter [inherited]

Backward point iterator associated type.

Invariant

This type has to derive from [mln::Site_Iterator](#).

Definition at line 147 of file core/concept/doc/image.hh.

10.113.2.2 template<typename E > typedef void mln::doc::Image< E >::coord [inherited]

Coordinate associated type.

Definition at line 131 of file core/concept/doc/image.hh.

10.113.2.3 template<typename E > typedef void mln::doc::Image< E >::dpoint [inherited]

Dpsite associated type.

Invariant

This type has to derive from [mln::Dpoint](#).

Definition at line 136 of file core/concept/doc/image.hh.

10.113.2.4 `template<typename E > typedef void mln::doc::Image< E >::fwd_piter`
[inherited]

Forward point iterator associated type.

Invariant

This type has to derive from [mln::Site_Iterator](#).

Definition at line 142 of file core/concept/doc/image.hh.

10.113.2.5 `template<typename E > typedef void mln::doc::Image< E >::lvalue` [inherited]

Type returned by the read-write pixel value operator.

Definition at line 52 of file core/concept/doc/image.hh.

10.113.2.6 `template<typename E > typedef void mln::doc::Image< E >::point` [inherited]

[Site](#) associated type.

Invariant

This type has to derive from [mln::Point](#).

Definition at line 121 of file core/concept/doc/image.hh.

10.113.2.7 `template<typename E > typedef void mln::doc::Image< E >::pset` [inherited]

[Point](#) set associated type.

Invariant

This type has to derive from [mln::Site_Set](#).

Definition at line 116 of file core/concept/doc/image.hh.

10.113.2.8 `template<typename E > typedef void mln::doc::Image< E >::psite` [inherited]

[Point_Site](#) associated type.

Invariant

This type has to derive from [mln::Point_Site](#).

Definition at line 126 of file core/concept/doc/image.hh.

10.113.2.9 `template<typename E > typedef void mln::doc::Image< E >::rvalue` [inherited]

Type returned by the read pixel value operator.

Definition at line 48 of file core/concept/doc/image.hh.

10.113.2.10 `template<typename E > typedef void mln::doc::Image< E >::skeleton`
`[inherited]`

Associate type that describes how this type of image is constructed.

Definition at line 64 of file core/concept/doc/image.hh.

10.113.2.11 `template<typename E > typedef void mln::doc::Image< E >::value` `[inherited]`

[Value](#) associated type.

Invariant

This type is neither qualified by const, nor by reference.

Definition at line 44 of file core/concept/doc/image.hh.

10.113.2.12 `template<typename E > typedef void mln::doc::Image< E >::vset` `[inherited]`

[Value](#) set associated type.

Invariant

This type has to derive from [mln::Value_Set](#).

Definition at line 57 of file core/concept/doc/image.hh.

10.113.3 Member Function Documentation**10.113.3.1** `template<typename E > const box<point>& mln::doc::Image< E >::bbox () const`
`[inherited]`

Give a bounding box of the image domain.

This bounding box may be larger than the smallest bounding box (the optimal one). Practically an image type is not obliged to update its bounding box so that it is always optimal.

Returns

A bounding box of the image domain.

10.113.3.2 `template<typename E > unsigned mln::doc::Fastest_Image< E >::border ()`

Give the border thickness.

Precondition

The image has to be initialized.

10.113.3.3 `template<typename E> const value* mln::doc::Fastest_Image< E >::buffer () const`

Give a hook to the value buffer.

Precondition

The image has to be initialized.

10.113.3.4 `template<typename E> int mln::doc::Fastest_Image< E >::delta_index (const dpoint & dp)`

Give the offset corresponding to the delta-point `dp`.

Parameters

[in] `dp` A delta-point.

Precondition

The image has to be initialized.

10.113.3.5 `template<typename E> const pset& mln::doc::Image< E >::domain () const [inherited]`

Give the definition domain of the image.

Returns

A reference to the domain point set.

10.113.3.6 `template<typename E> bool mln::doc::Image< E >::has (const psite & p) const [inherited]`

Test if the image owns the point site `p`.

Returns

True if accessing the image value at `p` is possible, that is, does not abort the execution.

10.113.3.7 `template<typename E> bool mln::doc::Image< E >::has (const psite & p) const [inherited]`

Test if `p` belongs to the image domain.

Parameters

[in] `p` A point site.

Returns

True if `p` belongs to the image domain.

Invariant

`has(p)` is true => `has(p)` is also true.

10.113.3.8 `template<typename E> bool mln::doc::Image< E >::is_valid () const`
`[inherited]`

Test if the image have been initialized.

10.113.3.9 `template<typename E> unsigned mln::doc::Fastest_Image< E >::nelements ()`
`const`

Give the number of pixels of the image including those of the virtual border.

Precondition

The image has to be initialized.

10.113.3.10 `template<typename E> unsigned mln::doc::Image< E >::nsites () const`
`[inherited]`

Give the number of points of the image domain.

10.113.3.11 `template<typename E> lvalue mln::doc::Image< E >::operator() (const psite & p`
`) [inherited]`

Read-write access to the image value located at *p*.

Parameters

[in] *p* A point site.

Precondition

The image has to own the site *p*.

Returns

The value at *p* (assignable).

10.113.3.12 `template<typename E> rvalue mln::doc::Image< E >::operator() (const psite & p`
`) const [inherited]`

Read-only access to the image value located at *p*.

Parameters

[in] *p* A point site.

Precondition

The image has to own the site *p*.

Returns

The value at *p* (not assignable).

10.113.3.13 `template<typename E > rvalue mln::doc::Fastest_Image< E >::operator[] (unsigned o) const`

Read-only access to the image value at offset `o`.

Parameters

[in] `o` An offset.

Precondition

`o < nelements()`

Returns

The value at `o` (not assignable).

10.113.3.14 `template<typename E > lvalue mln::doc::Fastest_Image< E >::operator[] (unsigned o)`

Read-write access to the image value at offset `o`.

Parameters

[in] `o` An offset.

Precondition

`o < nelements()`

Returns

The value at `o` (assignable).

10.113.3.15 `template<typename E > point mln::doc::Fastest_Image< E >::point_at_index (unsigned o) const`

Give the point at offset `o`.

Parameters

[in] `o` An offset.

Precondition

The image has to be initialized.

`o < nelements()`

10.113.3.16 `template<typename E > const vset& mln::doc::Image< E >::values () const [inherited]`

Give the set of values of the image.

Returns

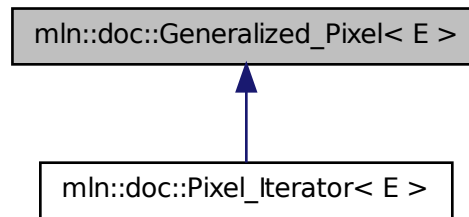
A reference to the value set.

10.114 mln::doc::Generalized_Pixel< E > Struct Template Reference

Documentation class for [mln::Generalized_Pixel](#).

```
#include <generalized_pixel.hh>
```

Inheritance diagram for mln::doc::Generalized_Pixel< E >:



Public Types

- typedef void [image](#)
Image associated type (with possible const qualification).
- typedef void [rvalue](#)
Read-only value associated type.
- typedef void [value](#)
Value associated type.

Public Member Functions

- [image](#) & [ima](#) () const
Give the image of this generalized pixel.
- [rvalue](#) [val](#) () const
Give the value of this generalized pixel.

10.114.1 Detailed Description

```
template<typename E> struct mln::doc::Generalized_Pixel< E >
```

Documentation class for [mln::Generalized_Pixel](#).

See also

[mln::Generalized_Pixel](#)

Definition at line 45 of file doc/generalized_pixel.hh.

10.114.2 Member Typedef Documentation

10.114.2.1 `template<typename E > typedef void mln::doc::Generalized_Pixel< E >::image`

[Image](#) associated type (with possible const qualification).

Definition at line 49 of file doc/generalized_pixel.hh.

10.114.2.2 `template<typename E > typedef void mln::doc::Generalized_Pixel< E >::rvalue`

Read-only value associated type.

Definition at line 55 of file doc/generalized_pixel.hh.

10.114.2.3 `template<typename E > typedef void mln::doc::Generalized_Pixel< E >::value`

[Value](#) associated type.

Definition at line 52 of file doc/generalized_pixel.hh.

10.114.3 Member Function Documentation

10.114.3.1 `template<typename E > image& mln::doc::Generalized_Pixel< E >::ima () const`

Give the image of this generalized pixel.

The constness of a pixel object is not transmitted to the underlying image.

10.114.3.2 `template<typename E > rvalue mln::doc::Generalized_Pixel< E >::val () const`

Give the value of this generalized pixel.

Returns

A read-only value.

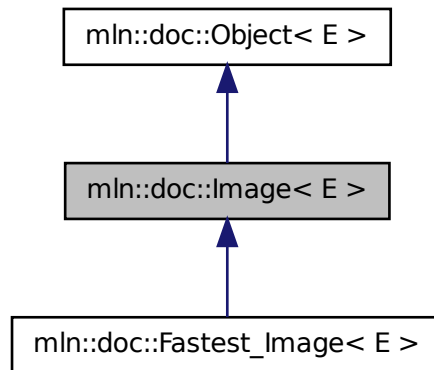
Reimplemented in [mln::doc::Pixel_Iterator< E >](#).

10.115 mln::doc::Image< E > Struct Template Reference

Documentation class for [mln::Image](#).

```
#include <image.hh>
```

Inheritance diagram for mln::doc::Image< E >:



Public Types

- typedef void [bkd_piter](#)
Backward point iterator associated type.
- typedef void [coord](#)
Coordinate associated type.
- typedef void [dpoint](#)
Dpsite associated type.
- typedef void [fwd_piter](#)
Forward point iterator associated type.
- typedef void [lvalue](#)
Type returned by the read-write pixel value operator.
- typedef void [point](#)
Site associated type.
- typedef void [pset](#)
Point set associated type.
- typedef void [psite](#)
Point_Site associated type.
- typedef void [rvalue](#)
Type returned by the read pixel value operator.

- typedef void [skeleton](#)
Associate type that describes how this type of image is constructed.
- typedef void [value](#)
Value associated type.
- typedef void [vset](#)
Value set associated type.

Public Member Functions

- const [box](#)< [point](#) > & [bbox](#) () const
Give a bounding box of the image domain.
- const [pset](#) & [domain](#) () const
Give the definition domain of the image.
- bool [has](#) (const [psite](#) &p) const
Test if the image owns the point site p.
- bool [has](#) (const [psite](#) &p) const
Test if p belongs to the image domain.
- bool [is_valid](#) () const
Test if the image have been initialized.
- unsigned [nsites](#) () const
Give the number of points of the image domain.
- [rvalue operator](#)() (const [psite](#) &p) const
Read-only access to the image value located at p.
- [lvalue operator](#)() (const [psite](#) &p)
Read-write access to the image value located at p.
- const [vset](#) & [values](#) () const
Give the set of values of the image.

10.115.1 Detailed Description

`template<typename E> struct mln::doc::Image< E >`

Documentation class for [mln::Image](#).

See also

[mln::Image](#)

Definition at line 36 of file core/concept/doc/image.hh.

10.115.2 Member Typedef Documentation

10.115.2.1 `template<typename E > typedef void mln::doc::Image< E >::bkd_piter`

Backward point iterator associated type.

Invariant

This type has to derive from [mln::Site_Iterator](#).

Definition at line 147 of file core/concept/doc/image.hh.

10.115.2.2 `template<typename E > typedef void mln::doc::Image< E >::coord`

Coordinate associated type.

Definition at line 131 of file core/concept/doc/image.hh.

10.115.2.3 `template<typename E > typedef void mln::doc::Image< E >::dpsite`

Dpsite associated type.

Invariant

This type has to derive from [mln::Dpoint](#).

Definition at line 136 of file core/concept/doc/image.hh.

10.115.2.4 `template<typename E > typedef void mln::doc::Image< E >::fwd_piter`

Forward point iterator associated type.

Invariant

This type has to derive from [mln::Site_Iterator](#).

Definition at line 142 of file core/concept/doc/image.hh.

10.115.2.5 `template<typename E > typedef void mln::doc::Image< E >::lvalue`

Type returned by the read-write pixel value operator.

Definition at line 52 of file core/concept/doc/image.hh.

10.115.2.6 `template<typename E > typedef void mln::doc::Image< E >::point`

[Site](#) associated type.

Invariant

This type has to derive from [mln::Point](#).

Definition at line 121 of file core/concept/doc/image.hh.

10.115.2.7 `template<typename E > typedef void mln::doc::Image< E >::pset`

[Point](#) set associated type.

Invariant

This type has to derive from [mln::Site_Set](#).

Definition at line 116 of file core/concept/doc/image.hh.

10.115.2.8 `template<typename E > typedef void mln::doc::Image< E >::psite`

[Point_Site](#) associated type.

Invariant

This type has to derive from [mln::Point_Site](#).

Definition at line 126 of file core/concept/doc/image.hh.

10.115.2.9 `template<typename E > typedef void mln::doc::Image< E >::rvalue`

Type returned by the read pixel value operator.

Definition at line 48 of file core/concept/doc/image.hh.

10.115.2.10 `template<typename E > typedef void mln::doc::Image< E >::skeleton`

Associate type that describes how this type of image is constructed.

Definition at line 64 of file core/concept/doc/image.hh.

10.115.2.11 `template<typename E > typedef void mln::doc::Image< E >::value`

[Value](#) associated type.

Invariant

This type is neither qualified by const, nor by reference.

Definition at line 44 of file core/concept/doc/image.hh.

10.115.2.12 `template<typename E > typedef void mln::doc::Image< E >::vset`

[Value](#) set associated type.

Invariant

This type has to derive from [mln::Value_Set](#).

Definition at line 57 of file core/concept/doc/image.hh.

10.115.3 Member Function Documentation

10.115.3.1 `template<typename E> const box<point>& mln::doc::Image< E >::bbox () const`

Give a bounding box of the image domain.

This bounding box may be larger than the smallest bounding box (the optimal one). Practically an image type is not obliged to update its bounding box so that it is always optimal.

Returns

A bounding box of the image domain.

10.115.3.2 `template<typename E> const pset& mln::doc::Image< E >::domain () const`

Give the definition domain of the image.

Returns

A reference to the domain point set.

10.115.3.3 `template<typename E> bool mln::doc::Image< E >::has (const psite & p) const`

Test if the image owns the point site *p*.

Returns

True if accessing the image value at *p* is possible, that is, does not abort the execution.

10.115.3.4 `template<typename E> bool mln::doc::Image< E >::has (const psite & p) const`

Test if *p* belongs to the image domain.

Parameters

[in] *p* A point site.

Returns

True if *p* belongs to the image domain.

Invariant

has(*p*) is true => has(*p*) is also true.

10.115.3.5 `template<typename E> bool mln::doc::Image< E >::is_valid () const`

Test if the image have been initialized.

10.115.3.6 `template<typename E> unsigned mln::doc::Image< E >::nsites () const`

Give the number of points of the image domain.

10.115.3.7 `template<typename E> rvalue mln::doc::Image< E >::operator() (const psite & p) const`

Read-only access to the image value located at *p*.

Parameters

[in] *p* A point site.

Precondition

The image has to own the site *p*.

Returns

The value at *p* (not assignable).

10.115.3.8 `template<typename E> lvalue mln::doc::Image< E >::operator() (const psite & p)`

Read-write access to the image value located at *p*.

Parameters

[in] *p* A point site.

Precondition

The image has to own the site *p*.

Returns

The value at *p* (assignable).

10.115.3.9 `template<typename E> const vset& mln::doc::Image< E >::values () const`

Give the set of values of the image.

Returns

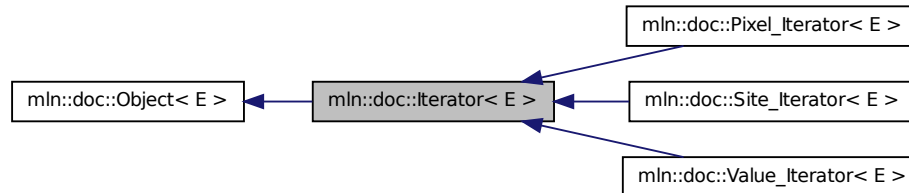
A reference to the value set.

10.116 `mln::doc::Iterator< E > Struct Template Reference`

Documentation class for [mln::Iterator](#).

```
#include <iterator.hh>
```

Inheritance diagram for `mln::doc::Iterator< E >`:



Public Member Functions

- void [invalidate](#) ()
Invalidate the iterator.
- bool [is_valid](#) () const
Returns true if the iterator is valid, that is, designates an element.
- void [start](#) ()
Start an iteration.

10.116.1 Detailed Description

`template<typename E> struct mln::doc::Iterator< E >`

Documentation class for [mln::Iterator](#).

See also

[mln::Iterator](#)

Definition at line 36 of file `doc/iterator.hh`.

10.116.2 Member Function Documentation

10.116.2.1 `template<typename E > void mln::doc::Iterator< E >::invalidate ()`

Invalidate the iterator.

10.116.2.2 `template<typename E > bool mln::doc::Iterator< E >::is_valid () const`

Returns true if the iterator is valid, that is, designates an element.

10.116.2.3 `template<typename E> void mln::doc::Iterator< E >::start ()`

Start an iteration.

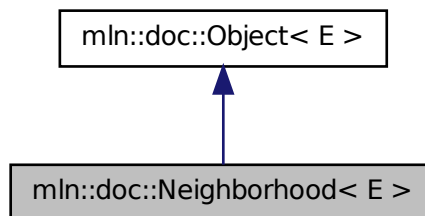
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

10.117 `mln::doc::Neighborhood< E >` Struct Template Reference

Documentation class for `mln::Neighborhood`.

```
#include <neighborhood.hh>
```

Inheritance diagram for `mln::doc::Neighborhood< E >`:

**Public Types**

- typedef void `bkd_niter`
Site_Iterator type associated to this neighborhood to browse neighbors in a backward way.
- typedef void `dpoint`
Dpsite associated type.
- typedef void `fwd_niter`
Site_Iterator type associated to this neighborhood to browse neighbors in a forward way.
- typedef void `niter`
Site_Iterator type associated to this neighborhood to browse neighbors.
- typedef void `point`
Site associated type.

10.117.1 Detailed Description

template<typename E> struct mln::doc::Neighborhood< E >

Documentation class for [mln::Neighborhood](#).

See also

[mln::Neighborhood](#)

Definition at line 37 of file core/concept/doc/neighborhood.hh.

10.117.2 Member Typedef Documentation

10.117.2.1 template<typename E > typedef void mln::doc::Neighborhood< E >::bkd_niter

[Site_Iterator](#) type associated to this neighborhood to browse neighbors in a backward way.

Definition at line 52 of file core/concept/doc/neighborhood.hh.

10.117.2.2 template<typename E > typedef void mln::doc::Neighborhood< E >::dpoint

Dpsite associated type.

Definition at line 55 of file core/concept/doc/neighborhood.hh.

10.117.2.3 template<typename E > typedef void mln::doc::Neighborhood< E >::fwd_niter

[Site_Iterator](#) type associated to this neighborhood to browse neighbors in a forward way.

Definition at line 47 of file core/concept/doc/neighborhood.hh.

10.117.2.4 template<typename E > typedef void mln::doc::Neighborhood< E >::niter

[Site_Iterator](#) type associated to this neighborhood to browse neighbors.

Definition at line 42 of file core/concept/doc/neighborhood.hh.

10.117.2.5 template<typename E > typedef void mln::doc::Neighborhood< E >::point

[Site](#) associated type.

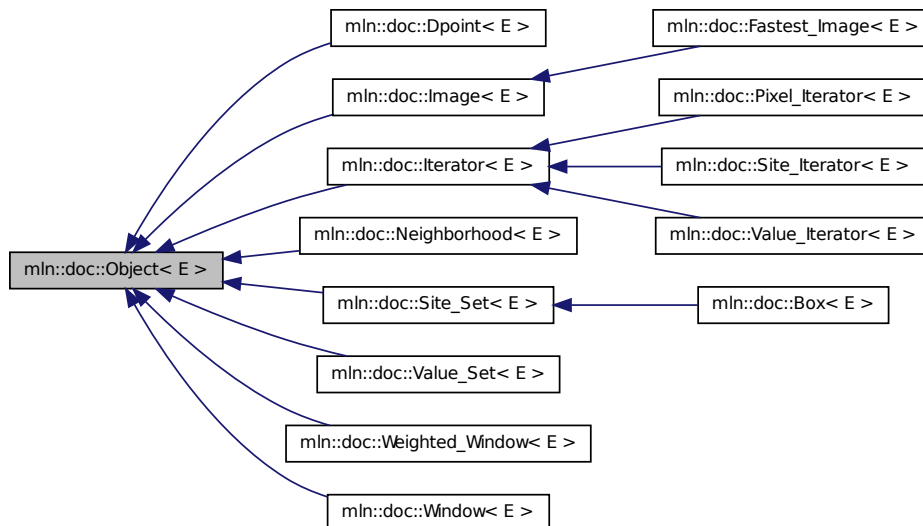
Definition at line 58 of file core/concept/doc/neighborhood.hh.

10.118 mln::doc::Object< E > Struct Template Reference

Documentation class for [mln::Object](#).

```
#include <object.hh>
```

Inheritance diagram for mln::doc::Object< E >:



10.118.1 Detailed Description

```
template<typename E> struct mln::doc::Object< E >
```

Documentation class for [mln::Object](#).

See also

[mln::Object](#)

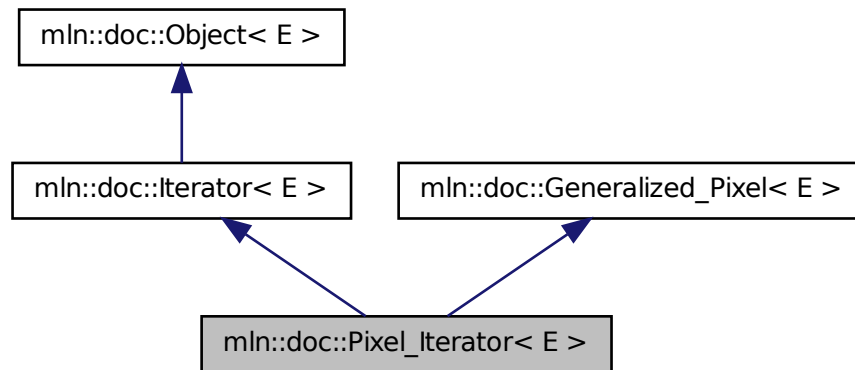
Definition at line 46 of file doc/object.hh.

10.119 mln::doc::Pixel_Iterator< E > Struct Template Reference

Documentation class for [mln::Iterator](#).

```
#include <pixel_iterator.hh>
```

Inheritance diagram for `mln::doc::Pixel_Iterator< E >`:



Public Types

- typedef void `image`
Image associated type (with possible const qualification).
- typedef void `lvalue`
Type returned by the read-write dereference operator.
- typedef void `rvalue`
Read-only value associated type.
- typedef void `value`
Value associated type.

Public Member Functions

- `image & ima () const`
Give the image of this generalized pixel.
- void `invalidate ()`
Invalidate the iterator.
- bool `is_valid () const`
Returns true if the iterator is valid, that is, designates an element.
- void `start ()`
Start an iteration.

- [lvalue val \(\) const](#)
Give the pixel value.

10.119.1 Detailed Description

template<typename E> struct mln::doc::Pixel_Iterator< E >

Documentation class for [mln::Iterator](#).

See also

[mln::Pixel_Iterator](#)

Definition at line 36 of file doc/pixel_iterator.hh.

10.119.2 Member Typedef Documentation

**10.119.2.1 template<typename E > typedef void mln::doc::Generalized_Pixel< E >::image
[inherited]**

[Image](#) associated type (with possible const qualification).

Definition at line 49 of file doc/generalized_pixel.hh.

10.119.2.2 template<typename E > typedef void mln::doc::Pixel_Iterator< E >::lvalue

Type returned by the read-write dereference operator.

Definition at line 41 of file doc/pixel_iterator.hh.

**10.119.2.3 template<typename E > typedef void mln::doc::Generalized_Pixel< E >::rvalue
[inherited]**

Read-only value associated type.

Definition at line 55 of file doc/generalized_pixel.hh.

**10.119.2.4 template<typename E > typedef void mln::doc::Generalized_Pixel< E >::value
[inherited]**

[Value](#) associated type.

Definition at line 52 of file doc/generalized_pixel.hh.

10.119.3 Member Function Documentation

**10.119.3.1 template<typename E > image& mln::doc::Generalized_Pixel< E >::ima () const
[inherited]**

Give the image of this generalized pixel.

The constness of a pixel object is not transmitted to the underlying image.

10.119.3.2 `template<typename E > void mln::doc::Iterator< E >::invalidate ()`
[inherited]

Invalidate the iterator.

10.119.3.3 `template<typename E > bool mln::doc::Iterator< E >::is_valid () const`
[inherited]

Returns true if the iterator is valid, that is, designates an element.

10.119.3.4 `template<typename E > void mln::doc::Iterator< E >::start ()` **[inherited]**

Start an iteration.

Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

10.119.3.5 `template<typename E > lvalue mln::doc::Pixel_Iterator< E >::val () const`

Give the pixel value.

Returns

The current pixel value; this value cannot be modified.

Reimplemented from [mln::doc::Generalized_Pixel< E >](#).

10.120 mln::doc::Point_Site< E > Struct Template Reference

Documentation class for [mln::Point_Site](#).

```
#include <point_site.hh>
```

Public Types

- enum { [dim](#) }
- typedef void [coord](#)
- typedef void [dpoint](#)
Dpsite associated type.
- typedef void [mesh](#)
Mesh associated type.
- typedef void [point](#)
Site associated type.

Public Member Functions

- `coord operator[]` (unsigned i) const
Read-only access to the i -th coordinate value.
- `const point & to_point` () const
Give a reference to the corresponding point.

10.120.1 Detailed Description

`template<typename E> struct mln::doc::Point_Site< E >`

Documentation class for `mln::Point_Site`.

See also

[mln::Point_Site](#)

Definition at line 37 of file `doc/point_site.hh`.

10.120.2 Member Typedef Documentation

10.120.2.1 `template<typename E > typedef void mln::doc::Point_Site< E >::coord`

Coordinate associated type.

Definition at line 62 of file `doc/point_site.hh`.

10.120.2.2 `template<typename E > typedef void mln::doc::Point_Site< E >::dpoint`

Dpsite associated type.

Invariant

This type has to derive from [mln::Dpoint](#).

Definition at line 58 of file `doc/point_site.hh`.

10.120.2.3 `template<typename E > typedef void mln::doc::Point_Site< E >::mesh`

[Mesh](#) associated type.

Invariant

This type has to derive from [mln::Mesh](#).

Definition at line 48 of file `doc/point_site.hh`.

10.120.2.4 `template<typename E > typedef void mln::doc::Point_Site< E >::point`

[Site](#) associated type.

Invariant

This type has to derive from [mln::Point](#).

Definition at line 53 of file doc/point_site.hh.

10.120.3 Member Enumeration Documentation

10.120.3.1 `template<typename E > anonymous enum`

Enumerator:

dim Dimension of the space.

Invariant

`dim > 0`

Definition at line 43 of file doc/point_site.hh.

10.120.4 Member Function Documentation

10.120.4.1 `template<typename E > coord mln::doc::Point_Site< E >::operator[] (unsigned i) const`

Read-only access to the `i`-th coordinate value.

Parameters

[in] *i* The coordinate index.

Precondition

`i < dim`

Returns

The value of the `i`-th coordinate.

10.120.4.2 `template<typename E > const point& mln::doc::Point_Site< E >::to_point () const`

Give a reference to the corresponding point.

This method allows for iterators to refer to a point.

Returns

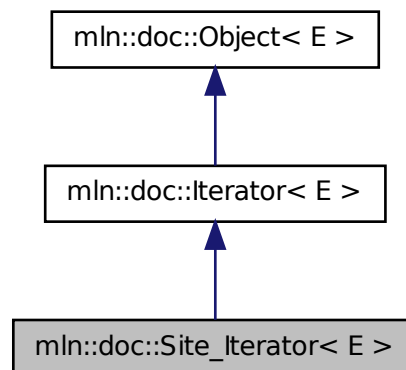
A point constant reference.

10.121 mln::doc::Site_Iterator< E > Struct Template Reference

Documentation class for [mln::Site_Iterator](#).

```
#include <point_iterator.hh>
```

Inheritance diagram for mln::doc::Site_Iterator< E >:



Public Types

- typedef void [psite](#)
Point_Site associated type.

Public Member Functions

- void [invalidate](#) ()
Invalidate the iterator.
- bool [is_valid](#) () const
Returns true if the iterator is valid, that is, designates an element.
- [operator psite](#) () const
Conversion into a point-site.
- void [start](#) ()
Start an iteration.

10.121.1 Detailed Description

`template<typename E> struct mln::doc::Site_Iterator< E >`

Documentation class for [mln::Site_Iterator](#).

See also

[mln::Site_Iterator](#)

Definition at line 37 of file `point_iterator.hh`.

10.121.2 Member Typedef Documentation

10.121.2.1 `template<typename E > typedef void mln::doc::Site_Iterator< E >::psite`

[Point_Site](#) associated type.

Invariant

This type has to derive from [mln::Point_Site](#).

Definition at line 43 of file `point_iterator.hh`.

10.121.3 Member Function Documentation

10.121.3.1 `template<typename E > void mln::doc::Iterator< E >::invalidate ()`
[**inherited**]

Invalidate the iterator.

10.121.3.2 `template<typename E > bool mln::doc::Iterator< E >::is_valid () const`
[**inherited**]

Returns true if the iterator is valid, that is, designates an element.

10.121.3.3 `template<typename E > mln::doc::Site_Iterator< E >::operator psite () const`

Conversion into a point-site.

Returns

A point site.

10.121.3.4 `template<typename E > void mln::doc::Iterator< E >::start ()` [**inherited**]

Start an iteration.

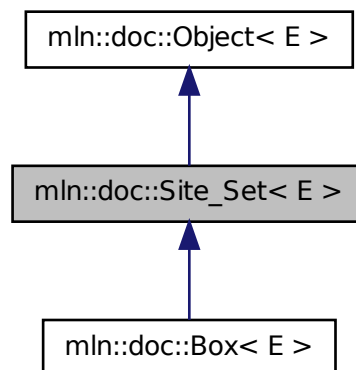
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

10.122 mln::doc::Site_Set< E > Struct Template Reference

Documentation class for [mln::Site_Set](#).

```
#include <site_set.hh>
```

Inheritance diagram for mln::doc::Site_Set< E >:



Public Types

- typedef void [bkd_piter](#)
Backward [Site_Iterator](#) associated type.
- typedef void [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef void [psite](#)
PSite associated type.
- typedef void [site](#)
Site associated type.

Public Member Functions

- bool [has](#) (const [psite](#) &p) const
*Test if *p* belongs to this site set.*

10.122.1 Detailed Description

`template<typename E> struct mln::doc::Site_Set< E >`

Documentation class for [mln::Site_Set](#).

See also

[mln::Site_Set](#)

Definition at line 37 of file `mln/core/concept/doc/site_set.hh`.

10.122.2 Member Typedef Documentation

10.122.2.1 `template<typename E > typedef void mln::doc::Site_Set< E >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 53 of file `mln/core/concept/doc/site_set.hh`.

10.122.2.2 `template<typename E > typedef void mln::doc::Site_Set< E >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 49 of file `mln/core/concept/doc/site_set.hh`.

10.122.2.3 `template<typename E > typedef void mln::doc::Site_Set< E >::psite`

PSite associated type.

Definition at line 45 of file `mln/core/concept/doc/site_set.hh`.

10.122.2.4 `template<typename E > typedef void mln::doc::Site_Set< E >::site`

[Site](#) associated type.

Definition at line 41 of file `mln/core/concept/doc/site_set.hh`.

10.122.3 Member Function Documentation

10.122.3.1 `template<typename E > bool mln::doc::Site_Set< E >::has (const psite & p) const`

Test if `p` belongs to this site set.

Parameters

`[in]` `p` A psite.

Returns

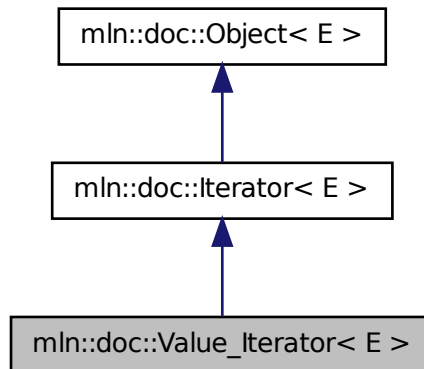
True if `p` is an element of the site set.

10.123 mln::doc::Value_Iterator< E > Struct Template Reference

Documentation class for [mln::Value_Iterator](#).

```
#include <value_iterator.hh>
```

Inheritance diagram for mln::doc::Value_Iterator< E >:



Public Types

- typedef void [value](#)
Value associated type.

Public Member Functions

- void [invalidate](#) ()
Invalidate the iterator.
- bool [is_valid](#) () const
Returns true if the iterator is valid, that is, designates an element.
- [operator value](#) () const
Conversion into a value.
- void [start](#) ()
Start an iteration.

10.123.1 Detailed Description

template<typename E> struct mln::doc::Value_Iterator< E >

Documentation class for [mln::Value_Iterator](#).

See also

[mln::Value_Iterator](#)

Definition at line 37 of file doc/value_iterator.hh.

10.123.2 Member Typedef Documentation

10.123.2.1 template<typename E > typedef void mln::doc::Value_Iterator< E >::value

[Value](#) associated type.

Definition at line 41 of file doc/value_iterator.hh.

10.123.3 Member Function Documentation

10.123.3.1 template<typename E > void mln::doc::Iterator< E >::invalidate ()
[*inherited*]

Invalidate the iterator.

10.123.3.2 template<typename E > bool mln::doc::Iterator< E >::is_valid () const
[*inherited*]

Returns true if the iterator is valid, that is, designates an element.

10.123.3.3 template<typename E > mln::doc::Value_Iterator< E >::operator value () const

Conversion into a value.

Returns

A value.

10.123.3.4 template<typename E > void mln::doc::Iterator< E >::start () [*inherited*]

Start an iteration.

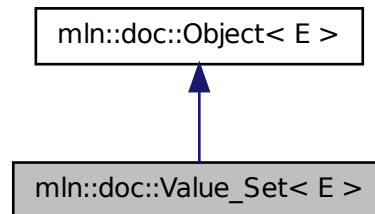
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

10.124 mln::doc::Value_Set< E > Struct Template Reference

Documentation class for [mln::Value_Set](#).

```
#include <value_set.hh>
```

Inheritance diagram for mln::doc::Value_Set< E >:



Public Types

- typedef void [bkd_viter](#)
Backward [Value_Iterator](#) associated type.
- typedef void [fwd_viter](#)
Forward [Value_Iterator](#) associated type.
- typedef void [value](#)
[Value](#) associated type.

Public Member Functions

- bool [has](#) (const [value](#) &v) const
Test if v belongs to this set of values.
- unsigned [index_of](#) (const [value](#) &v) const
Give the index of value v in this set.
- unsigned [nvalues](#) () const
Give the number of values in this set.
- [value operator\[\]](#) (unsigned i) const
Give the i-th value of this set.

10.124.1 Detailed Description

```
template<typename E> struct mln::doc::Value_Set< E >
```

Documentation class for [mln::Value_Set](#).

See also

[mln::Value_Set](#)

Definition at line 37 of file doc/value_set.hh.

10.124.2 Member Typedef Documentation

10.124.2.1 `template<typename E> typedef void mln::doc::Value_Set< E >::bkd_viter`

Backward [Value_Iterator](#) associated type.

Definition at line 49 of file doc/value_set.hh.

10.124.2.2 `template<typename E> typedef void mln::doc::Value_Set< E >::fwd_viter`

Forward [Value_Iterator](#) associated type.

Definition at line 45 of file doc/value_set.hh.

10.124.2.3 `template<typename E> typedef void mln::doc::Value_Set< E >::value`

[Value](#) associated type.

Definition at line 41 of file doc/value_set.hh.

10.124.3 Member Function Documentation

10.124.3.1 `template<typename E> bool mln::doc::Value_Set< E >::has (const value & v) const`

Test if v belongs to this set of values.

Parameters

[in] v A value.

Returns

True if v is an element of the set of values.

10.124.3.2 `template<typename E> unsigned mln::doc::Value_Set< E >::index_of (const value & v) const`

Give the index of value v in this set.

10.124.3.3 `template<typename E> unsigned mln::doc::Value_Set< E >::nvalues () const`

Give the number of values in this set.

10.124.3.4 `template<typename E> value mln::doc::Value_Set< E >::operator[] (unsigned i) const`

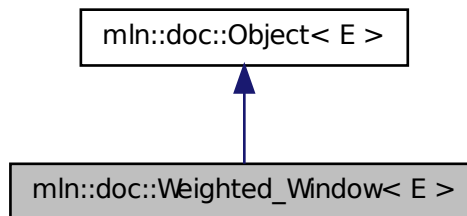
Give the `i`-th value of this set.

10.125 mln::doc::Weighted_Window< E > Struct Template Reference

Documentation class for [mln::Weighted_Window](#).

```
#include <weighted_window.hh>
```

Inheritance diagram for mln::doc::Weighted_Window< E >:



Public Types

- typedef void [bkd_qiter](#)
Site_Iterator type associated to this *weighted_window* to browse its points in a backward way.
- typedef void [dpoint](#)
Dpsite associated type.
- typedef void [fwd_qiter](#)
Site_Iterator type associated to this *weighted_window* to browse its points in a forward way.
- typedef void [point](#)
Site associated type.
- typedef void [weight](#)
Weight associated type.
- typedef void [window](#)
Window associated type.

Public Member Functions

- unsigned `delta` () const
Give the maximum coordinate gap between the window center and a window point.
- bool `is_centered` () const
Test if the `weighted_window` is centered.
- bool `is_empty` () const
Test if the `weighted window` is empty.
- E & `sym` ()
Apply a central symmetry to the target `weighted window`.
- const `window` & `win` () const
Give the corresponding window.

10.125.1 Detailed Description

`template<typename E> struct mln::doc::Weighted_Window< E >`

Documentation class for `mln::Weighted_Window`. A `weighted_window` is the definition of a set of points located around a central point, with a weight associated to each point.

See also

[mln::Weighted_Window](#)

Definition at line 40 of file `doc/weighted_window.hh`.

10.125.2 Member Typedef Documentation

10.125.2.1 `template<typename E > typedef void mln::doc::Weighted_Window< E >::bkd_qiter`

[Site_Iterator](#) type associated to this `weighted_window` to browse its points in a backward way.

Definition at line 51 of file `doc/weighted_window.hh`.

10.125.2.2 `template<typename E > typedef void mln::doc::Weighted_Window< E >::dpoint`

Dpsite associated type.

Definition at line 57 of file `doc/weighted_window.hh`.

10.125.2.3 `template<typename E > typedef void mln::doc::Weighted_Window< E >::fwd_qiter`

[Site_Iterator](#) type associated to this `weighted_window` to browse its points in a forward way.

Definition at line 46 of file `doc/weighted_window.hh`.

10.125.2.4 `template<typename E > typedef void mln::doc::Weighted_Window< E >::point`

[Site](#) associated type.

Definition at line 54 of file doc/weighted_window.hh.

10.125.2.5 `template<typename E > typedef void mln::doc::Weighted_Window< E >::weight`

Weight associated type.

Definition at line 60 of file doc/weighted_window.hh.

10.125.2.6 `template<typename E > typedef void mln::doc::Weighted_Window< E >::window`

[Window](#) associated type.

Definition at line 63 of file doc/weighted_window.hh.

10.125.3 Member Function Documentation**10.125.3.1** `template<typename E > unsigned mln::doc::Weighted_Window< E >::delta () const`

Give the maximum coordinate gap between the window center and a window point.

10.125.3.2 `template<typename E > bool mln::doc::Weighted_Window< E >::is_centered () const`

Test if the `weighted_window` is centered.

A weighted window is centered is the origin belongs to it.

10.125.3.3 `template<typename E > bool mln::doc::Weighted_Window< E >::is_empty () const`

Test if the weighted window is empty.

A `weighted_window` of null size is empty.

10.125.3.4 `template<typename E > E& mln::doc::Weighted_Window< E >::sym ()`

Apply a central symmetry to the target weighted window.

10.125.3.5 `template<typename E > const window& mln::doc::Weighted_Window< E >::win () const`

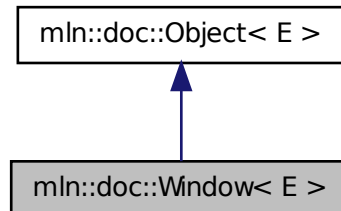
Give the corresponding window.

10.126 mln::doc::Window< E > Struct Template Reference

Documentation class for [mln::Window](#).

```
#include <window.hh>
```

Inheritance diagram for `mln::doc::Window< E >`:



Public Types

- typedef void [bkd_qiter](#)
Site_Iterator type associated to this window to browse its points in a backward way.
- typedef void [fwd_qiter](#)
Site_Iterator type associated to this window to browse its points in a forward way.
- typedef void [qiter](#)
Site_Iterator type associated to this window to browse its points.

10.126.1 Detailed Description

```
template<typename E> struct mln::doc::Window< E >
```

Documentation class for [mln::Window](#). A window is the definition of a set of points located around a central point.

See also

[mln::Window](#)

Definition at line 40 of file `concept/doc/window.hh`.

10.126.2 Member Typedef Documentation

10.126.2.1 `template<typename E > typedef void mln::doc::Window< E >::bkd_qiter`

[Site_Iterator](#) type associated to this window to browse its points in a backward way.

Definition at line 55 of file `concept/doc/window.hh`.

10.126.2.2 `template<typename E > typedef void mln::doc::Window< E >::fwd_qiter`

[Site_Iterator](#) type associated to this window to browse its points in a forward way.

Definition at line 50 of file concept/doc/window.hh.

10.126.2.3 `template<typename E > typedef void mln::doc::Window< E >::qiter`

[Site_Iterator](#) type associated to this window to browse its points.

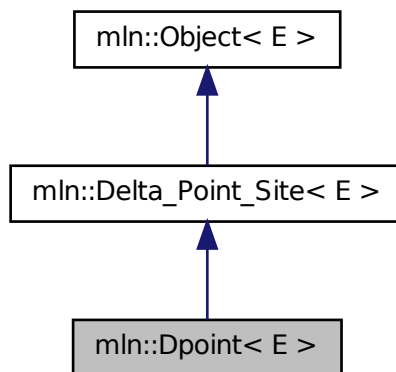
Definition at line 45 of file concept/doc/window.hh.

10.127 mln::Dpoint< E > Struct Template Reference

Base class for implementation of delta-point classes.

```
#include <dpoint.hh>
```

Inheritance diagram for mln::Dpoint< E >:

**Public Member Functions**

- `const E & to_dpoint () const`

It is a [Dpoint](#) so it returns itself.

10.127.1 Detailed Description

```
template<typename E> struct mln::Dpoint< E >
```

Base class for implementation of delta-point classes. A delta-point is a vector defined by a couple of points.

Given two points, A and B, the vector AB is mapped into the delta-point $D = AB$. Practically one can write: $D = B - A$.

See also

[mln::doc::Dpoint](#) for a complete documentation of this class contents.

Definition at line 63 of file concept/dpoint.hh.

10.127.2 Member Function Documentation

10.127.2.1 `template<typename E > const E & mln::Dpoint< E >::to_dpoint () const [inline]`

It is a [Dpoint](#) so it returns itself.

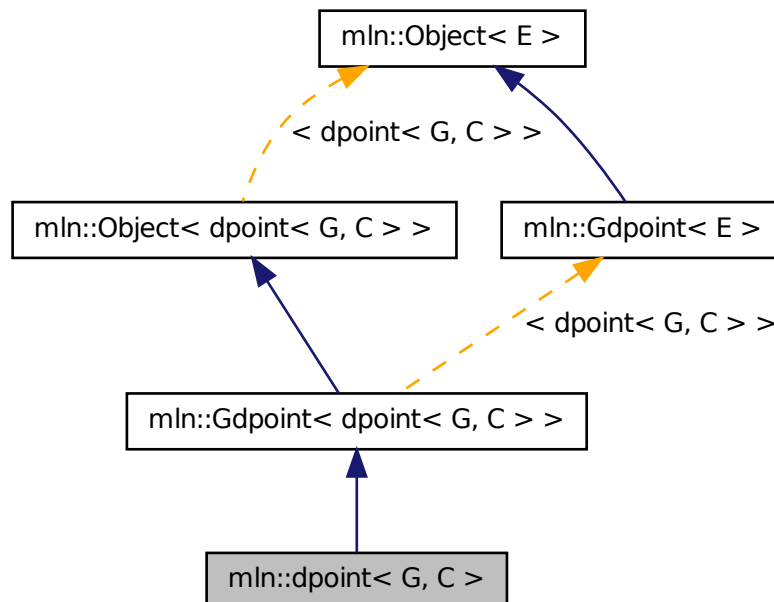
Definition at line 88 of file concept/dpoint.hh.

10.128 `mln::dpoint< G, C >` Struct Template Reference

Generic delta-point class.

```
#include <dpoint.hh>
```

Inheritance diagram for `mln::dpoint< G, C >`:



Public Types

- enum { `dim` = `G::dim` }
- typedef `C` `coord`
Coordinate associated type.
- typedef `G` `grid`
Grid associated type.
- typedef `point`< `G`, `C` > `psite`
Psite associated type.
- typedef `point`< `G`, `C` > `site`
Site associated type.
- typedef `algebra::vec`< `G::dim`, `C` > `vec`
Algebra vector (vec) associated type.

Public Member Functions

- `dpoint` ()
Constructor without argument.
- `template`<typename `C2` >
`dpoint` (const `algebra::vec`< `dim`, `C2` > &`v`)
Constructor from an algebra vector.
- `template`<typename `F` >
`dpoint` (const `Function_v2v`< `F` > &`f`)
Constructor; coordinates are set by function f .
- `template`<typename `Q` >
`operator mln::algebra::vec`< `dpoint`< `G`, `C` >::`dim`, `Q` > () const
Conversion towards a algebra::vec.
- `C operator`[] (unsigned `i`) const
Read-only access to the i -th coordinate value.
- `C & operator`[] (unsigned `i`)
Read-write access to the i -th coordinate value.
- void `set_all` (`C` `c`)
Set all coordinates to the value c .
- `vec to_vec` () const
Explicit conversion.
- `dpoint` (`C` `ind`)

- [dpoint](#) (const [literal::zero_t](#) &)
Constructors/assignments with literals.

10.128.1 Detailed Description

template<typename G, typename C> struct mln::dpoint< G, C >

Generic delta-point class. Parameters are G the dimension of the space and C the coordinate type in this space.

Definition at line 58 of file dpoint.hh.

10.128.2 Member Typedef Documentation

10.128.2.1 template<typename G, typename C> typedef C mln::dpoint< G, C >::coord

Coordinate associated type.

Definition at line 76 of file dpoint.hh.

10.128.2.2 template<typename G, typename C> typedef G mln::dpoint< G, C >::grid

Grid associated type.

Definition at line 67 of file dpoint.hh.

10.128.2.3 template<typename G, typename C> typedef point<G,C> mln::dpoint< G, C >::psite

Psite associated type.

Definition at line 70 of file dpoint.hh.

10.128.2.4 template<typename G, typename C> typedef point<G,C> mln::dpoint< G, C >::site

[Site](#) associated type.

Definition at line 73 of file dpoint.hh.

10.128.2.5 template<typename G, typename C> typedef algebra::vec<G::dim, C> mln::dpoint< G, C >::vec

Algebra vector (vec) associated type.

Definition at line 79 of file dpoint.hh.

10.128.3 Member Enumeration Documentation

10.128.3.1 template<typename G, typename C> anonymous enum

Enumerator:

dim Dimension of the space.

Invariant

`dim > 0`

Definition at line 64 of file dpoint.hh.

10.128.4 Constructor & Destructor Documentation

10.128.4.1 template<typename G, typename C> mln::dpoint< G, C >::dpoint () [inline]

Constructor without argument.

Definition at line 152 of file dpoint.hh.

10.128.4.2 template<typename G, typename C> template<typename C2> mln::dpoint< G, C >::dpoint (const algebra::vec< dim, C2 > & v) [inline]

Constructor from an algebra vector.

Definition at line 159 of file dpoint.hh.

References mln::dpoint< G, C >::dim.

10.128.4.3 template<typename G, typename C> mln::dpoint< G, C >::dpoint (C ind) [inline]

Constructors with different numbers of arguments (coordinates) w.r.t. the dimension.

Definition at line 176 of file dpoint.hh.

10.128.4.4 template<typename G, typename C> mln::dpoint< G, C >::dpoint (const literal::zero_t &) [inline]

Constructors/assignments with literals.

Definition at line 203 of file dpoint.hh.

10.128.4.5 template<typename G, typename C> template<typename F> mln::dpoint< G, C >::dpoint (const Function_v2v< F > & f) [inline]

Constructor; coordinates are set by function *f*.

Definition at line 238 of file dpoint.hh.

10.128.5 Member Function Documentation

10.128.5.1 `template<typename G , typename C > template<typename Q > mln::dpoint< G, C >::operator mln::algebra::vec< dpoint< G, C >::dim, Q >() const [inline]`

Conversion towards a algebra::vec.

Definition at line 257 of file dpoint.hh.

References mln::dpoint< G, C >::to_vec().

10.128.5.2 `template<typename G , typename C > C & mln::dpoint< G, C >::operator[] (unsigned i) [inline]`

Read-write access to the *i*-th coordinate value.

Parameters

[in] *i* The coordinate index.

Precondition

$i < \text{dim}$

Definition at line 144 of file dpoint.hh.

References mln::dpoint< G, C >::dim.

10.128.5.3 `template<typename G , typename C > C mln::dpoint< G, C >::operator[] (unsigned i) const [inline]`

Read-only access to the *i*-th coordinate value.

Parameters

[in] *i* The coordinate index.

Precondition

$i < \text{dim}$

Definition at line 136 of file dpoint.hh.

References mln::dpoint< G, C >::dim.

10.128.5.4 `template<typename G , typename C > void mln::dpoint< G, C >::set_all (C c) [inline]`

Set all coordinates to the value *c*.

Definition at line 248 of file dpoint.hh.

Referenced by mln::win::line< M, i, C >::line().

10.128.5.5 `template<typename G , typename C > dpoint< G, C >::vec mln::dpoint< G, C >::to_vec () const [inline]`

Explicit conversion.

Definition at line 265 of file dpoint.hh.

References `mln::dpoint< G, C >::dim`.

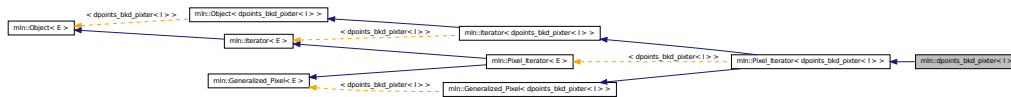
Referenced by `mln::dpoint< G, C >::operator mln::algebra::vec< dpoint< G, C >::dim, Q >()`.

10.129 mln::dpoints_bkd_pixter< I > Class Template Reference

A generic backward iterator on the pixels of a dpoint-based window or neighborhood.

```
#include <dpoints_pixter.hh>
```

Inheritance diagram for `mln::dpoints_bkd_pixter< I >`:



Public Member Functions

- `const I::value & center_val () const`
The value around which this iterator moves.
- `template<typename Dps , typename Pref > dpoints_bkd_pixter (const Generalized_Pixel< Pref > &pxl_ref, const Dps &dps)`
Constructor (using a generalized pixel).
- `template<typename Dps , typename Pref > dpoints_bkd_pixter (I &image, const Dps &dps, const Pref &p_ref)`
Constructor (using an image).
- `void next ()`
Go to the next element.
- `void start ()`
Manipulation.
- `void invalidate ()`
Invalidate the iterator.
- `bool is_valid () const`
Test the iterator validity.
- `void update ()`
Force this iterator to update its location to take into account that its center point may have moved.

10.129.1 Detailed Description

template<typename I> class mln::dpoints_bkd_pixter< I >

A generic backward iterator on the pixels of a dpoint-based window or neighborhood. Parameter *I* is the image type.

Definition at line 140 of file `dpoints_pixter.hh`.

10.129.2 Constructor & Destructor Documentation

10.129.2.1 template<typename I > template<typename Dps , typename Pref > mln::dpoints_bkd_pixter< I >::dpoints_bkd_pixter (I & *image*, const Dps & *dps*, const Pref & *p_ref*) [inline]

Constructor (using an image).

Parameters

- [in] *image* The image to iterate over.
- [in] *dps* An object (neighborhood or window) that can provide a set of delta-points.
- [in] *p_ref* Center (resp. reference) point of the neighborhood (resp. window).

Definition at line 338 of file `dpoints_pixter.hh`.

10.129.2.2 template<typename I > template<typename Dps , typename Pref > mln::dpoints_bkd_pixter< I >::dpoints_bkd_pixter (const Generalized_Pixel< Pref > & *pxl_ref*, const Dps & *dps*) [inline]

Constructor (using a generalized pixel).

Parameters

- [in] *pxl_ref* Center (generalized) pixel to iterate around.
- [in] *dps* An object (neighborhood or window) that can provide a set of delta-points.

Definition at line 352 of file `dpoints_pixter.hh`.

10.129.3 Member Function Documentation

10.129.3.1 template<typename I > const I::value & mln::dpoints_bkd_pixter< I >::center_val () const [inline]

The value around which this iterator moves.

Definition at line 367 of file `dpoints_pixter.hh`.

10.129.3.2 template<typename I > void mln::dpoints_bkd_pixter< I >::invalidate () [inline]

Invalidate the iterator.

Definition at line 436 of file `dpoints_pixter.hh`.

10.129.3.3 `template<typename I> bool mln::dpoints_bkd_pixter< I >::is_valid () const` `[inline]`

Test the iterator validity.

Definition at line 428 of file `dpoints_pixter.hh`.

Referenced by `mln::dpoints_bkd_pixter< I >::update()`.

10.129.3.4 `void mln::Iterator< dpoints_bkd_pixter< I > >::next ()` `[inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.129.3.5 `template<typename I> void mln::dpoints_bkd_pixter< I >::start ()` `[inline]`

Manipulation.

Start an iteration.

Definition at line 409 of file `dpoints_pixter.hh`.

References `mln::dpoints_bkd_pixter< I >::update()`.

10.129.3.6 `template<typename I> void mln::dpoints_bkd_pixter< I >::update ()` `[inline]`

Force this iterator to update its location to take into account that its center point may have moved.

Definition at line 395 of file `dpoints_pixter.hh`.

References `mln::dpoints_bkd_pixter< I >::is_valid()`.

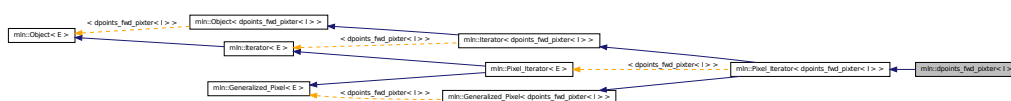
Referenced by `mln::dpoints_bkd_pixter< I >::start()`.

10.130 `mln::dpoints_fwd_pixter< I >` Class Template Reference

A generic forward iterator on the pixels of a dpoint-based window or neighborhood.

```
#include <dpoints_pixter.hh>
```

Inheritance diagram for `mln::dpoints_fwd_pixter< I >`:



Public Member Functions

- const I::value & [center_val](#) () const
The value around which this iterator moves.
- template<typename Dps , typename Pref >
[dpoints_fwd_pixter](#) (const [Generalized_Pixel](#)< Pref > &pxl_ref, const Dps &dps)
Constructor (using a generalized pixel).
- template<typename Dps , typename Pref >
[dpoints_fwd_pixter](#) (I &image, const Dps &dps, const Pref &p_ref)
Constructor (using an image).
- void [next](#) ()
Go to the next element.
- void [start](#) ()
Manipulation.
- void [invalidate](#) ()
Invalidate the iterator.
- bool [is_valid](#) () const
Test the iterator validity.
- void [update](#) ()
Force this iterator to update its location to take into account that its center point may have moved.

10.130.1 Detailed Description

template<typename I> class mln::dpoints_fwd_pixter< I >

A generic forward iterator on the pixels of a dpoint-based window or neighborhood. Parameter I is the image type.

Definition at line 57 of file dpoints_pixter.hh.

10.130.2 Constructor & Destructor Documentation

**10.130.2.1 template<typename I > template<typename Dps , typename Pref >
mln::dpoints_fwd_pixter< I >::dpoints_fwd_pixter (I & image, const Dps & dps,
const Pref & p_ref) [inline]**

Constructor (using an image).

Parameters

- [in] *image* The image to iterate over.
- [in] *dps* An object (neighborhood or window) that can provide a set of delta-points.
- [in] *p_ref* Center (resp. reference) point of the neighborhood (resp. window).

Definition at line 224 of file dpoints_pixter.hh.

10.130.2.2 `template<typename I > template<typename Dps , typename Pref > mln::dpoints_fwd_pixter< I >::dpoints_fwd_pixter (const Generalized_Pixel< Pref > & pxl_ref, const Dps & dps) [inline]`

Constructor (using a generalized pixel).

Parameters

[in] *pxl_ref* Center (generalized) pixel to iterate around.

[in] *dps* An object (neighborhood or window) that can provide a set of delta-points.

Definition at line 241 of file dpoints_pixter.hh.

10.130.3 Member Function Documentation

10.130.3.1 `template<typename I > const I::value & mln::dpoints_fwd_pixter< I >::center_val () const [inline]`

The value around which this iterator moves.

Definition at line 256 of file dpoints_pixter.hh.

10.130.3.2 `template<typename I > void mln::dpoints_fwd_pixter< I >::invalidate () [inline]`

Invalidate the iterator.

Definition at line 325 of file dpoints_pixter.hh.

10.130.3.3 `template<typename I > bool mln::dpoints_fwd_pixter< I >::is_valid () const [inline]`

Test the iterator validity.

Definition at line 317 of file dpoints_pixter.hh.

Referenced by mln::dpoints_fwd_pixter< I >::update().

10.130.3.4 `void mln::Iterator< dpoints_fwd_pixter< I > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.130.3.5 `template<typename I > void mln::dpoints_fwd_paxter< I >::start () [inline]`

Manipulation.

Start an iteration.

Definition at line 298 of file `dpoints_paxter.hh`.

References `mln::dpoints_fwd_paxter< I >::update()`.

10.130.3.6 `template<typename I > void mln::dpoints_fwd_paxter< I >::update () [inline]`

Force this iterator to update its location to take into account that its center point may have moved.

Definition at line 284 of file `dpoints_paxter.hh`.

References `mln::dpoints_fwd_paxter< I >::is_valid()`.

Referenced by `mln::dpoints_fwd_paxter< I >::start()`.

10.131 mln::dpsites_bkd_piter< V > Class Template Reference

A generic backward iterator on points of windows and of neighborhoods.

```
#include <dpsites_piter.hh>
```

Inherits `site_relative_iterator_base< V, dpsites_bkd_piter< V > >`.

Public Member Functions

- `template<typename P > dpsites_bkd_piter (const V &v, const P &c)`

Constructor.

- `dpsites_bkd_piter ()`

Constructor without argument.

- `void next ()`

Go to the next element.

10.131.1 Detailed Description

```
template<typename V> class mln::dpsites_bkd_piter< V >
```

A generic backward iterator on points of windows and of neighborhoods. The parameter `V` is the type of `std::vector` enclosing structure.

Definition at line 94 of file `dpsites_piter.hh`.

10.131.2 Constructor & Destructor Documentation

10.131.2.1 `template<typename V > template<typename P > mln::dpsites_bkd_piter< V >::dpsites_bkd_piter (const V & v, const P & c) [inline]`

Constructor.

Parameters

- [in] `v` [Object](#) that can provide an array of delta-points.
- [in] `c` Center point to iterate around.

Definition at line 217 of file `dpsites_piter.hh`.

10.131.2.2 `template<typename V > mln::dpsites_bkd_piter< V >::dpsites_bkd_piter () [inline]`

Constructor without argument.

Definition at line 210 of file `dpsites_piter.hh`.

10.131.3 Member Function Documentation

10.131.3.1 `void mln::Site_Iterator< dpsites_bkd_piter< V > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.132 mln::dpsites_fwd_piter< V > Class Template Reference

A generic forward iterator on points of windows and of neighborhoods.

```
#include <dpsites_piter.hh>
```

Inherits `site_relative_iterator_base< V, dpsites_fwd_piter< V > >`.

Public Member Functions

- `template<typename P > dpsites_fwd_piter (const V &v, const P &c)`
Constructor.
- `dpsites_fwd_piter ()`
Constructor without argument.

- void [next](#) ()

Go to the next element.

10.132.1 Detailed Description

template<typename V> class mln::dpsites_fwd_piter< V >

A generic forward iterator on points of windows and of neighborhoods. The parameter V is the type of std::vector enclosing structure.

Definition at line 48 of file dpsites_piter.hh.

10.132.2 Constructor & Destructor Documentation

10.132.2.1 template<typename V > template<typename P > mln::dpsites_fwd_piter< V >::dpsites_fwd_piter (const V & v, const P & c) [inline]

Constructor.

Parameters

[in] *v* [Object](#) that can provide an array of delta-points.

[in] *c* Center point to iterate around.

Definition at line 149 of file dpsites_piter.hh.

10.132.2.2 template<typename V > mln::dpsites_fwd_piter< V >::dpsites_fwd_piter () [inline]

Constructor without argument.

Definition at line 142 of file dpsites_piter.hh.

10.132.3 Member Function Documentation

10.132.3.1 void mln::Site_Iterator< dpsites_fwd_piter< V > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.133 mln::Edge< E > Struct Template Reference

edge category flag type.

```
#include <edge.hh>
```

10.133.1 Detailed Description

template<typename E> struct mln::Edge< E >

edge category flag type.

Definition at line 52 of file edge.hh.

10.134 mln::edge_image< P, V, G > Class Template Reference

[Image](#) based on graph edges.

```
#include <edge_image.hh>
```

Inherits [image_base< fun::i2v::array< V >, p_edges< G, internal::efsite_selector< P, G >::site_function_t>, edge_image< P, V, G >>](#).

Public Types

- typedef [graph_elt_neighborhood< G, p_edges< G, site_function_t >> edge_nbh_t](#)
Neighborhood type.
- typedef [graph_elt_window< G, p_edges< G, site_function_t >> edge_win_t](#)
Edge Window type.
- typedef G [graph_t](#)
The type of the underlying graph.
- typedef [edge_nbh_t nbh_t](#)
Default Neighborhood type.
- typedef [internal::efsite_selector< P, G >::site_function_t site_function_t](#)
Function mapping graph elements to sites.
- typedef [edge_image< tag::psite_< P >, tag::value_< V >, tag::graph_< G >> skeleton](#)
Skeleton type.
- typedef [edge_win_t win_t](#)
Default Window type.

Public Member Functions

- [edge_image](#) ()
Constructors.
- rvalue [operator\(\)](#) (unsigned e_id) const
Value accessors/operators overloads.

10.134.1 Detailed Description

`template<typename P, typename V, typename G = util::graph> class mln::edge_image< P, V, G >`

[Image](#) based on graph edges.

Definition at line 123 of file core/image/edge_image.hh.

10.134.2 Member Typedef Documentation

10.134.2.1 `template<typename P, typename V, typename G = util::graph> typedef graph_elt_neighborhood<G,p_edges<G,site_function_t> > mln::edge_image< P, V, G >::edge_nbh_t`

[Neighborhood](#) type.

Definition at line 153 of file core/image/edge_image.hh.

10.134.2.2 `template<typename P, typename V, typename G = util::graph> typedef graph_elt_window<G,p_edges<G,site_function_t> > mln::edge_image< P, V, G >::edge_win_t`

[Edge Window](#) type.

Definition at line 151 of file core/image/edge_image.hh.

10.134.2.3 `template<typename P, typename V, typename G = util::graph> typedef G mln::edge_image< P, V, G >::graph_t`

The type of the underlying graph.

Definition at line 138 of file core/image/edge_image.hh.

10.134.2.4 `template<typename P, typename V, typename G = util::graph> typedef edge_nbh_t mln::edge_image< P, V, G >::nbh_t`

Default [Neighborhood](#) type.

Definition at line 159 of file core/image/edge_image.hh.

10.134.2.5 `template<typename P, typename V, typename G = util::graph> typedef internal::efsite_selector<P,G>::site_function_t mln::edge_image< P, V, G >::site_function_t`

[Function](#) mapping graph elements to sites.

Definition at line 147 of file core/image/edge_image.hh.

10.134.2.6 `template<typename P, typename V, typename G = util::graph> typedef edge_image< tag::psite_<P>, tag::value_<V>, tag::graph_<G> > mln::edge_image< P, V, G >::skeleton`

Skeleton type.

Definition at line 143 of file core/image/edge_image.hh.

10.134.2.7 `template<typename P, typename V, typename G = util::graph> typedef edge_win_t mln::edge_image< P, V, G >::win_t`

Default [Window](#) type.

Definition at line 156 of file core/image/edge_image.hh.

10.134.3 Constructor & Destructor Documentation

10.134.3.1 `template<typename P, typename V, typename G > mln::edge_image< P, V, G >::edge_image() [inline]`

Constructors.

Definition at line 248 of file core/image/edge_image.hh.

10.134.4 Member Function Documentation

10.134.4.1 `template<typename P, typename V, typename G > edge_image< P, V, G >::rvalue mln::edge_image< P, V, G >::operator()(unsigned e_id) const`

[Value](#) accessors/operators overloads.

Definition at line 302 of file core/image/edge_image.hh.

10.135 mln::extended< I > Struct Template Reference

Makes an image become restricted by a point set.

```
#include <extended.hh>
```

Inherits `image_domain_morpher< I, box< I::site >, extended< I > >`.

Public Types

- `typedef tag::image_< I > skeleton`

Skeleton.

- typedef I::value [value](#)
Value type.

Public Member Functions

- const [box](#)< typename I::site > & [domain](#) () const
Give the definition domain.
- [extended](#) ()
Constructor without argument.
- [extended](#) (I &ima, const [box](#)< typename I::site > &b)
Constructor.

10.135.1 Detailed Description

template<typename I> struct mln::extended< I >

Makes an image become restricted by a point set.

Definition at line 92 of file extended.hh.

10.135.2 Member Typedef Documentation

10.135.2.1 template<typename I> typedef tag::image_<I> mln::extended< I >::skeleton

Skeleton.

Definition at line 102 of file extended.hh.

10.135.2.2 template<typename I> typedef I ::value mln::extended< I >::value

[Value](#) type.

Definition at line 99 of file extended.hh.

10.135.3 Constructor & Destructor Documentation

10.135.3.1 template<typename I > mln::extended< I >::extended () [inline]

Constructor without argument.

Definition at line 169 of file extended.hh.

10.135.3.2 `template<typename I > mln::extended< I >::extended (I & ima, const box< typename I::site > & b) [inline]`

Constructor.

Definition at line 175 of file extended.hh.

10.135.4 Member Function Documentation

10.135.4.1 `template<typename I > const box< typename I::site > & mln::extended< I >::domain () const [inline]`

Give the definition domain.

Definition at line 192 of file extended.hh.

10.136 mln::extension_fun< I, F > Class Template Reference

Extends the domain of an image with a function.

```
#include <extension_fun.hh>
```

Inherits image_identity< I, I::domain_t, extension_fun< I, F > >.

Public Types

- typedef I::value [rvalue](#)
Return type of read-only access.
- typedef [extension_fun](#)< tag::image_< I >, tag::function_< F > > [skeleton](#)
Skeleton.
- typedef I::value [value](#)
Image value type.

Public Member Functions

- const F & [extension](#) () const
Give the extension function.
- [extension_fun](#) (I &ima, const F &fun)
*Constructor from an image *ima* and a function *fun*.*
- [extension_fun](#) ()
Constructor without argument.
- template<typename P >
bool [has](#) (const P &p) const
*Test if *p* is valid.*

- `internal::morpher_lvalue_< I >::ret operator()` (const typename I::psite &p)
Read-write access to the image value located at site p.
- `I::value operator()` (const typename I::psite &p) const
Read-only access to the image value located at site p.

10.136.1 Detailed Description

`template<typename I, typename F> class mln::extension_fun< I, F >`

Extends the domain of an image with a function.

Definition at line 99 of file `extension_fun.hh`.

10.136.2 Member Typedef Documentation

10.136.2.1 `template<typename I, typename F> typedef I ::value mln::extension_fun< I, F >::rvalue`

Return type of read-only access.

Definition at line 112 of file `extension_fun.hh`.

10.136.2.2 `template<typename I, typename F> typedef extension_fun< tag::image_<I>, tag::function_<F> > mln::extension_fun< I, F >::skeleton`

Skeleton.

Definition at line 106 of file `extension_fun.hh`.

10.136.2.3 `template<typename I, typename F> typedef I ::value mln::extension_fun< I, F >::value`

[Image](#) value type.

Definition at line 109 of file `extension_fun.hh`.

10.136.3 Constructor & Destructor Documentation

10.136.3.1 `template<typename I, typename F > mln::extension_fun< I, F >::extension_fun ()`
`[inline]`

Constructor without argument.

Definition at line 178 of file `extension_fun.hh`.

10.136.3.2 `template<typename I, typename F > mln::extension_fun< I, F >::extension_fun (I & ima, const F & fun) [inline]`

Constructor from an image `ima` and a function `fun`.

Definition at line 184 of file `extension_fun.hh`.

10.136.4 Member Function Documentation

10.136.4.1 `template<typename I, typename F > const F & mln::extension_fun< I, F >::extension () const [inline]`

Give the extension function.

Definition at line 243 of file `extension_fun.hh`.

10.136.4.2 `template<typename I, typename F > template<typename P > bool mln::extension_fun< I, F >::has (const P & p) const [inline]`

Test if `p` is valid.

It returns always true, assuming that the function is valid for any `p`.

Definition at line 201 of file `extension_fun.hh`.

10.136.4.3 `template<typename I, typename F > internal::morpher_lvalue_< I >::ret mln::extension_fun< I, F >::operator() (const typename I::psite & p) [inline]`

Read-write access to the image value located at site `p`.

Definition at line 223 of file `extension_fun.hh`.

10.136.4.4 `template<typename I, typename F > I::value mln::extension_fun< I, F >::operator() (const typename I::psite & p) const [inline]`

Read-only access to the image value located at site `p`.

Definition at line 209 of file `extension_fun.hh`.

10.137 mln::extension_ima< I, J > Class Template Reference

Extends the domain of an image with an image.

```
#include <extension_ima.hh>
```

Inherits `image_identity< I, I::domain_t, extension_ima< I, J > >`.

Public Types

- typedef `I::value` [rvalue](#)

Return type of read-only access.

- typedef `extension_ima`< tag::image_< I >, tag::ext_< J > > `skeleton`
Skeleton.
- typedef I::value `value`
Image value type.

Public Member Functions

- const J & `extension` () const
Read-only access to the extension domain (image).
- `extension_ima` (I &ima, const J &ext)
Constructor from an image `ima` and a function `ext`.
- `extension_ima` ()
Constructor without argument.
- template<typename P >
bool `has` (const P &p) const
Test if `p` is valid.
- internal::morpher_ivalue_< I >::ret `operator`() (const typename I::psite &p)
Read-write access to the image value located at site `p`.
- I::value `operator`() (const typename I::psite &p) const
Read-only access to the image value located at site `p`.

10.137.1 Detailed Description

`template<typename I, typename J> class mln::extension_ima< I, J >`

Extends the domain of an image with an image.

Definition at line 97 of file `extension_ima.hh`.

10.137.2 Member Typedef Documentation

10.137.2.1 `template<typename I, typename J> typedef I ::value mln::extension_ima< I, J >::rvalue`

Return type of read-only access.

Definition at line 111 of file `extension_ima.hh`.

10.137.2.2 `template<typename I, typename J> typedef extension_ima< tag::image_<I>, tag::ext_<J> > mln::extension_ima< I, J >::skeleton`

Skeleton.

Definition at line 105 of file extension_ima.hh.

10.137.2.3 `template<typename I, typename J> typedef I ::value mln::extension_ima< I, J >::value`

[Image](#) value type.

Definition at line 108 of file extension_ima.hh.

10.137.3 Constructor & Destructor Documentation

10.137.3.1 `template<typename I, typename J > mln::extension_ima< I, J >::extension_ima () [inline]`

Constructor without argument.

Definition at line 173 of file extension_ima.hh.

10.137.3.2 `template<typename I, typename J > mln::extension_ima< I, J >::extension_ima (I & ima, const J & ext) [inline]`

Constructor from an image `ima` and a function `ext`.

Definition at line 179 of file extension_ima.hh.

10.137.4 Member Function Documentation

10.137.4.1 `template<typename I, typename J > const J & mln::extension_ima< I, J >::extension () const [inline]`

Read-only access to the extension domain (image).

Definition at line 244 of file extension_ima.hh.

10.137.4.2 `template<typename I, typename J > template<typename P > bool mln::extension_ima< I, J >::has (const P & p) const [inline]`

Test if `p` is valid.

Definition at line 196 of file extension_ima.hh.

10.137.4.3 `template<typename I, typename J > internal::morpher_lvalue_< I >::ret mln::extension_ima< I, J >::operator() (const typename I::psite & p) [inline]`

Read-write access to the image value located at site `p`.

Definition at line 223 of file extension_ima.hh.

10.137.4.4 `template<typename I , typename J > I::value mln::extension_ima< I, J >::operator()
(const typename I::psite & p) const [inline]`

Read-only access to the image value located at site *p*;

Definition at line 208 of file `extension_ima.hh`.

10.138 `mln::extension_val< I >` Class Template Reference

Extends the domain of an image with a value.

```
#include <extension_val.hh>
```

Inherits `image_identity< I, I::domain_t, extension_val< I > >`.

Public Types

- typedef `I::value` `rvalue`
Return type of read-only access.
- typedef `extension_val< tag::image_< I > >` `skeleton`
Skeleton.
- typedef `I::value` `value`
Image value type.

Public Member Functions

- void `change_extension` (const typename `I::value` &val)
Change the value of the extension domain.
- const `I::value` & `extension` () const
Read-only access to the value of the extension domain.
- `extension_val` (I &ima, const typename `I::value` &val)
*Constructor from an image *ima* and a value *val*.*
- `extension_val` ()
Constructor without argument.
- template<typename `P` >
bool `has` (const `P` &p) const
*Test if *p* is valid. It returns always true.*
- internal::mopher_lvalue_< I >::ret `operator()` (const typename `I::psite` &p)
*Read-write access to the image value located at site *p*.*
- `I::value` `operator()` (const typename `I::psite` &p) const
*Read-only access to the image value located at site *p*;*

10.138.1 Detailed Description

template<typename I> class mln::extension_val< I >

Extends the domain of an image with a value.

Definition at line 98 of file extension_val.hh.

10.138.2 Member Typedef Documentation

10.138.2.1 template<typename I> typedef I ::value mln::extension_val< I >::rvalue

Return type of read-only access.

Definition at line 110 of file extension_val.hh.

10.138.2.2 template<typename I> typedef extension_val< tag::image_<I> > mln::extension_val< I >::skeleton

Skeleton.

Definition at line 104 of file extension_val.hh.

10.138.2.3 template<typename I> typedef I ::value mln::extension_val< I >::value

[Image](#) value type.

Definition at line 107 of file extension_val.hh.

10.138.3 Constructor & Destructor Documentation

10.138.3.1 template<typename I> mln::extension_val< I >::extension_val () [inline]

Constructor without argument.

Definition at line 176 of file extension_val.hh.

10.138.3.2 template<typename I> mln::extension_val< I >::extension_val (I & ima, const typename I::value & val) [inline]

Constructor from an image *ima* and a value *val*.

Definition at line 182 of file extension_val.hh.

10.138.4 Member Function Documentation

10.138.4.1 template<typename I> void mln::extension_val< I >::change_extension (const typename I::value & val) [inline]

Change the value of the extension domain.

Definition at line 247 of file extension_val.hh.

10.138.4.2 `template<typename I> const I::value & mln::extension_val< I >::extension () const [inline]`

Read-only access to the value of the extension domain.

Definition at line 238 of file extension_val.hh.

10.138.4.3 `template<typename I> template<typename P> bool mln::extension_val< I >::has (const P & p) const [inline]`

Test if `p` is valid. It returns always true.

Definition at line 199 of file extension_val.hh.

10.138.4.4 `template<typename I> internal::morpher_lvalue_< I >::ret mln::extension_val< I >::operator() (const typename I::psite & p) [inline]`

Read-write access to the image value located at site `p`.

Definition at line 220 of file extension_val.hh.

10.138.4.5 `template<typename I> I::value mln::extension_val< I >::operator() (const typename I::psite & p) const [inline]`

Read-only access to the image value located at site `p`;

Definition at line 207 of file extension_val.hh.

10.139 mln::faces_psite< N, D, P > Class Template Reference

[Point](#) site associated to a [mln::p_faces](#).

```
#include <faces_psite.hh>
```

Inherits `pseudo_site_base_< const P &, faces_psite< N, D, P > >`.

Public Member Functions

- [faces_psite](#) ()
Construction and assignment.
- [faces_psite](#) (const [p_faces](#)< N, D, P > &pf, const [topo::n_face](#)< N, D > &face)
- bool [is_valid](#) () const
Psite manipulators.
- void [invalidate](#) ()
Invalidate this psite.
- const [target](#) & [site_set](#) () const
Site set manipulators.

- void `change_target` (const `target` &new_target)
Set the target site_set.
- `topo::n_face`< N, D > `face` () const
Face handle manipulators.
- unsigned `n` () const
Return the dimension of the face of this psite.
- unsigned `face_id` () const
Return the id of the face of this psite.

10.139.1 Detailed Description

`template<unsigned N, unsigned D, typename P> class mln::faces_psite< N, D, P >`

`Point` site associated to a `mln::p_faces`.

Template Parameters

- N* The dimension of the face associated to this psite.
- D* The dimension of the complex this psite belongs to.
- P* The type of point associated to this psite.

Definition at line 56 of file `faces_psite.hh`.

10.139.2 Constructor & Destructor Documentation

10.139.2.1 `template<unsigned N, unsigned D, typename P > mln::faces_psite< N, D, P >::faces_psite () [inline]`

Construction and assignment.

Definition at line 190 of file `faces_psite.hh`.

References `mln::faces_psite< N, D, P >::invalidate()`.

10.139.2.2 `template<unsigned N, unsigned D, typename P > mln::faces_psite< N, D, P >::faces_psite (const p_faces< N, D, P > & pf, const topo::n_face< N, D > & face) [inline]`

Precondition

`pf.cplx() == face.cplx()`.

Definition at line 201 of file `faces_psite.hh`.

10.139.3 Member Function Documentation

10.139.3.1 `template<unsigned N, unsigned D, typename P > void mln::faces_psite< N, D, P >::change_target (const target & new_target) [inline]`

Set the target site_set.

Definition at line 265 of file faces_psite.hh.

References `mln::p_faces< N, D, P >::cplx()`, and `mln::faces_psite< N, D, P >::invalidate()`.

10.139.3.2 `template<unsigned N, unsigned D, typename P > topo::n_face< N, D > mln::faces_psite< N, D, P >::face () const [inline]`

Face handle manipulators.

Return the face handle of this point site.

Definition at line 289 of file faces_psite.hh.

Referenced by `mln::operator!=()`, and `mln::operator==()`.

10.139.3.3 `template<unsigned N, unsigned D, typename P > unsigned mln::faces_psite< N, D, P >::face_id () const [inline]`

Return the id of the face of this psite.

Definition at line 305 of file faces_psite.hh.

10.139.3.4 `template<unsigned N, unsigned D, typename P > void mln::faces_psite< N, D, P >::invalidate () [inline]`

Invalidate this psite.

Definition at line 239 of file faces_psite.hh.

Referenced by `mln::faces_psite< N, D, P >::change_target()`, and `mln::faces_psite< N, D, P >::faces_psite()`.

10.139.3.5 `template<unsigned N, unsigned D, typename P > bool mln::faces_psite< N, D, P >::is_valid () const [inline]`

Psite manipulators.

Is this psite valid?

Definition at line 230 of file faces_psite.hh.

10.139.3.6 `template<unsigned N, unsigned D, typename P > unsigned mln::faces_psite< N, D, P >::n () const [inline]`

Return the dimension of the face of this psite.

Definition at line 297 of file faces_psite.hh.

10.139.3.7 `template<unsigned N, unsigned D, typename P > const p_faces< N, D, P > & mln::faces_psite< N, D, P >::site_set () const [inline]`

[Site](#) set manipulators.

Return the [p_faces](#) this site is built on. (shortcut for *target()).

Precondition

Member `face_` is valid.

Definition at line 247 of file `faces_psite.hh`.

Referenced by `mln::operator!=()`, and `mln::operator==()`.

10.140 mln::flat_image< T, S > Struct Template Reference

[Image](#) with a single value.

```
#include <flat_image.hh>
```

Inherits `image_primary< T, S, flat_image< T, S > >`.

Public Types

- typedef T & [lvalue](#)
Return type of read-write access.
- typedef const T & [rvalue](#)
Return type of read-only access.
- typedef [flat_image](#)< tag::value_< T >, tag::domain_< S > > [skeleton](#)
Skeleton.
- typedef T [value](#)
Value associated type.

Public Member Functions

- const S & [domain](#) () const
Give the definition domain.
- [flat_image](#) (const T &val, const S &pset)
Constructor.
- [flat_image](#) ()
Constructor without argument.
- bool [has](#) (const typename S::psite &p) const
Test if p is valid: always return true.

- `const T & operator()` (`const typename S::psite &p`) `const`
Read-only access to the image value located at point p .
- `T & operator()` (`const typename S::psite &p`)
Read-write access to the image value located at point p .

10.140.1 Detailed Description

`template<typename T, typename S> struct mln::flat_image< T, S >`

[Image](#) with a single value.

Definition at line 105 of file `flat_image.hh`.

10.140.2 Member Typedef Documentation

10.140.2.1 `template<typename T, typename S> typedef T& mln::flat_image< T, S >::lvalue`

Return type of read-write access.

Definition at line 118 of file `flat_image.hh`.

10.140.2.2 `template<typename T, typename S> typedef const T& mln::flat_image< T, S >::rvalue`

Return type of read-only access.

Definition at line 115 of file `flat_image.hh`.

10.140.2.3 `template<typename T, typename S> typedef flat_image< tag::value_<T>, tag::domain_<S> > mln::flat_image< T, S >::skeleton`

Skeleton.

Definition at line 108 of file `flat_image.hh`.

10.140.2.4 `template<typename T, typename S> typedef T mln::flat_image< T, S >::value`

[Value](#) associated type.

Definition at line 112 of file `flat_image.hh`.

10.140.3 Constructor & Destructor Documentation

10.140.3.1 `template<typename T , typename S > mln::flat_image< T, S >::flat_image ()`
`[inline]`

Constructor without argument.

Definition at line 191 of file `flat_image.hh`.

10.140.3.2 `template<typename T , typename S > mln::flat_image< T, S >::flat_image (const T & val, const S & pset) [inline]`

Constructor.

Definition at line 197 of file flat_image.hh.

10.140.4 Member Function Documentation

10.140.4.1 `template<typename T , typename S > const S & mln::flat_image< T, S >::domain () const [inline]`

Give the definition domain.

Definition at line 214 of file flat_image.hh.

10.140.4.2 `template<typename T , typename S > bool mln::flat_image< T, S >::has (const typename S::psite & p) const [inline]`

Test if p is valid: always return true.

Definition at line 222 of file flat_image.hh.

10.140.4.3 `template<typename T , typename S > const T & mln::flat_image< T, S >::operator() (const typename S::psite & p) const [inline]`

Read-only access to the image value located at point p.

Definition at line 230 of file flat_image.hh.

10.140.4.4 `template<typename T , typename S > T & mln::flat_image< T, S >::operator() (const typename S::psite & p) [inline]`

Read-write access to the image value located at point p.

Definition at line 239 of file flat_image.hh.

10.141 mln::fun::from_accu< A > Struct Template Reference

Wrap an accumulator into a function.

```
#include <from_accu.hh>
```

Inherits mln::fun::unary_param< from_accu< A >, A * >.

10.141.1 Detailed Description

`template<typename A> struct mln::fun::from_accu< A >`

Wrap an accumulator into a function.

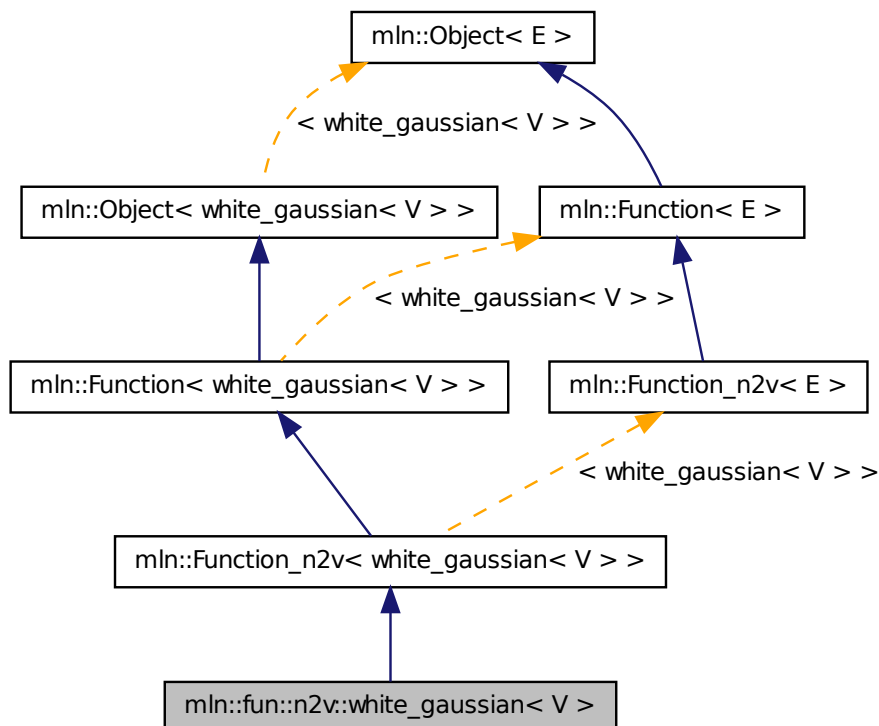
Definition at line 49 of file from_accu.hh.

10.142 mln::fun::n2v::white_gaussian< V > Struct Template Reference

Generate a White Gaussian Noise.

```
#include <white_gaussian.hh>
```

Inheritance diagram for mln::fun::n2v::white_gaussian< V >:



10.142.1 Detailed Description

```
template<typename V> struct mln::fun::n2v::white_gaussian< V >
```

Generate a White Gaussian Noise. Reference: <http://www.dspguru.com/dsp/howtos/how-to-generate-white-gaussian-noise>

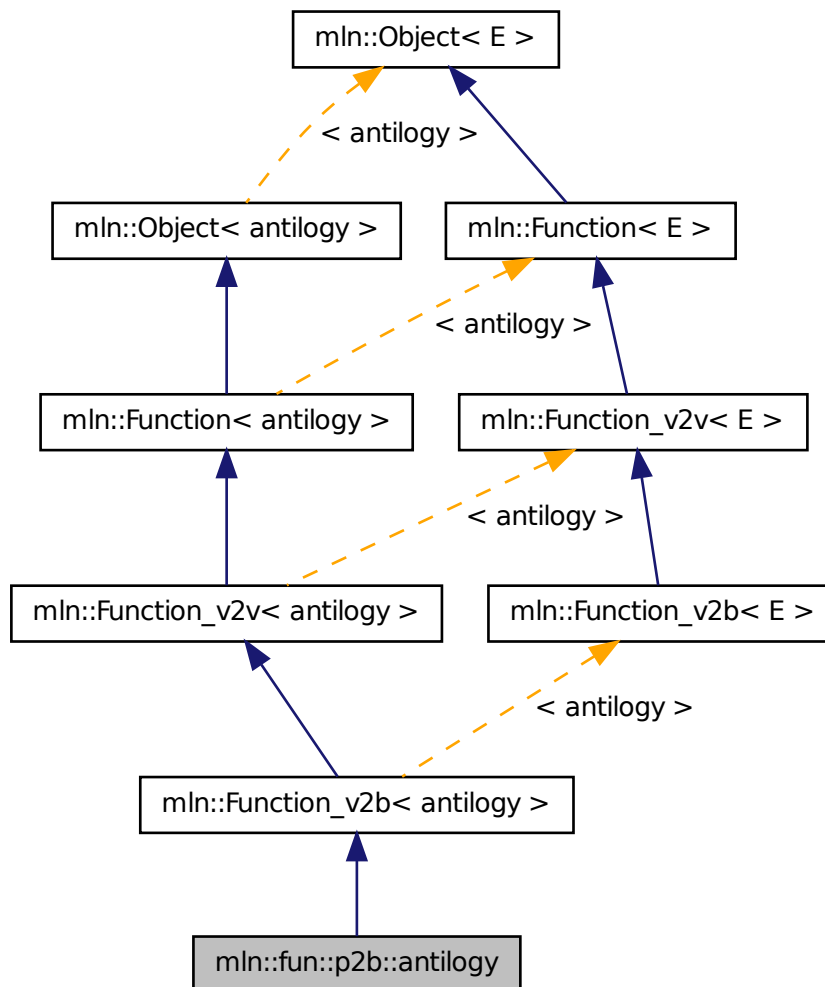
Definition at line 56 of file `white_gaussian.hh`.

10.143 mln::fun::p2b::antilogy Struct Reference

A `p2b` function always returning `false`.

```
#include <antilogy.hh>
```

Inheritance diagram for mln::fun::p2b::antilogy:



10.143.1 Detailed Description

A [p2b](#) function always returning `false`. A simpler name would be `false`, but this is not a valid C++ identifier, as `false` is a keyword of the language.

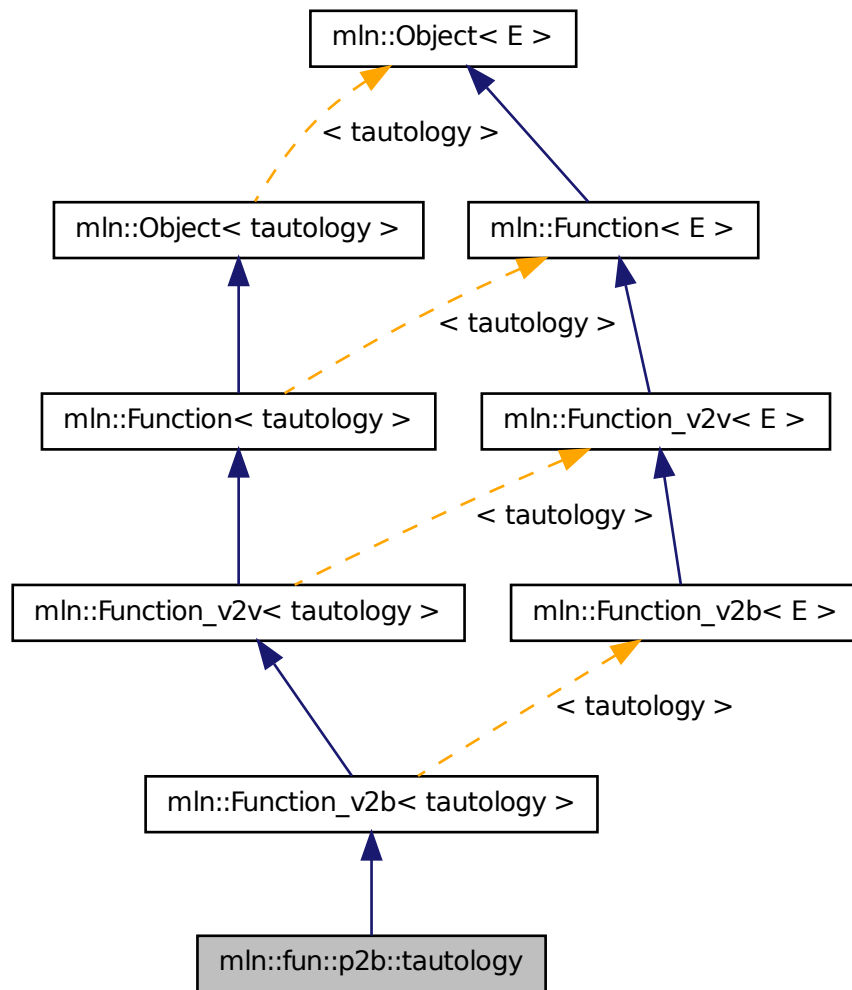
Definition at line 50 of file `antilogy.hh`.

10.144 mln::fun::p2b::tautology Struct Reference

A [p2b](#) function always returning `true`.

```
#include <tautology.hh>
```

Inheritance diagram for `mln::fun::p2b::tautology`:



10.144.1 Detailed Description

A `p2b` function always returning `true`. A simpler name would be `'true'`, but this is not a valid C++ identifier, as `true` is a keyword of the language.

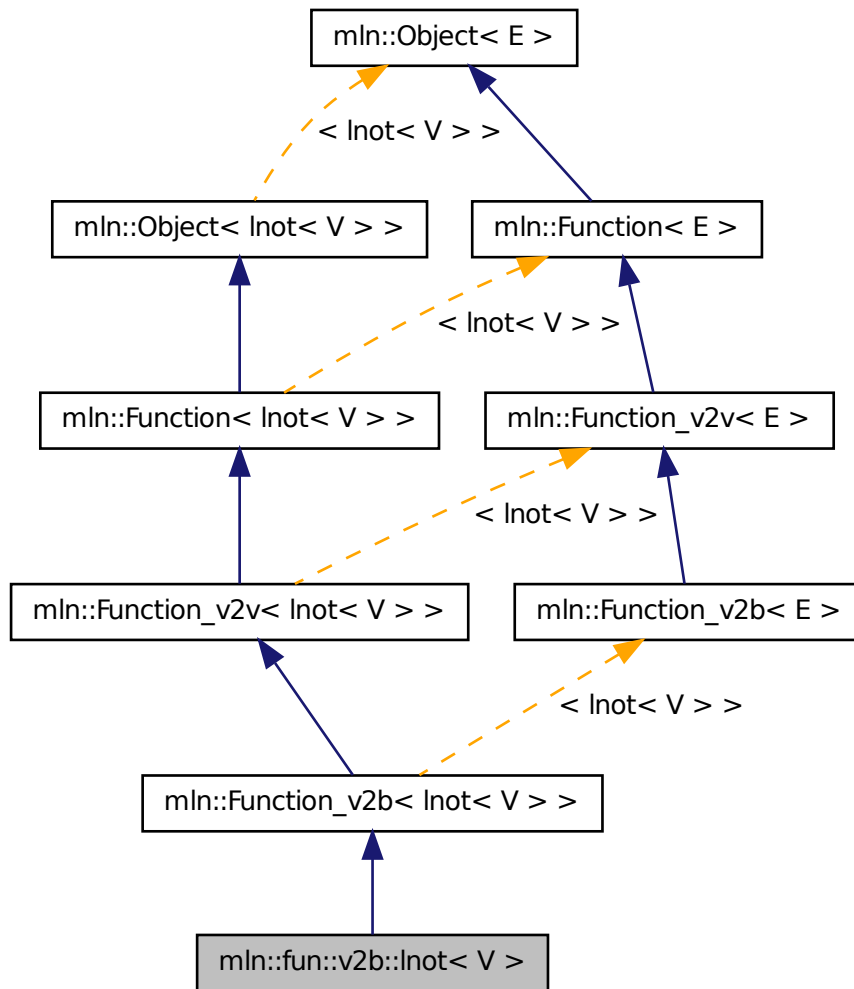
Definition at line 50 of file `tautology.hh`.

10.145 mln::fun::v2b::lnot< V > Struct Template Reference

Functor computing logical-not on a value.

```
#include <lnot.hh>
```

Inheritance diagram for mln::fun::v2b::lnot< V >:



10.145.1 Detailed Description

```
template<typename V> struct mln::fun::v2b::lnot< V >
```

Functor computing logical-not on a value.

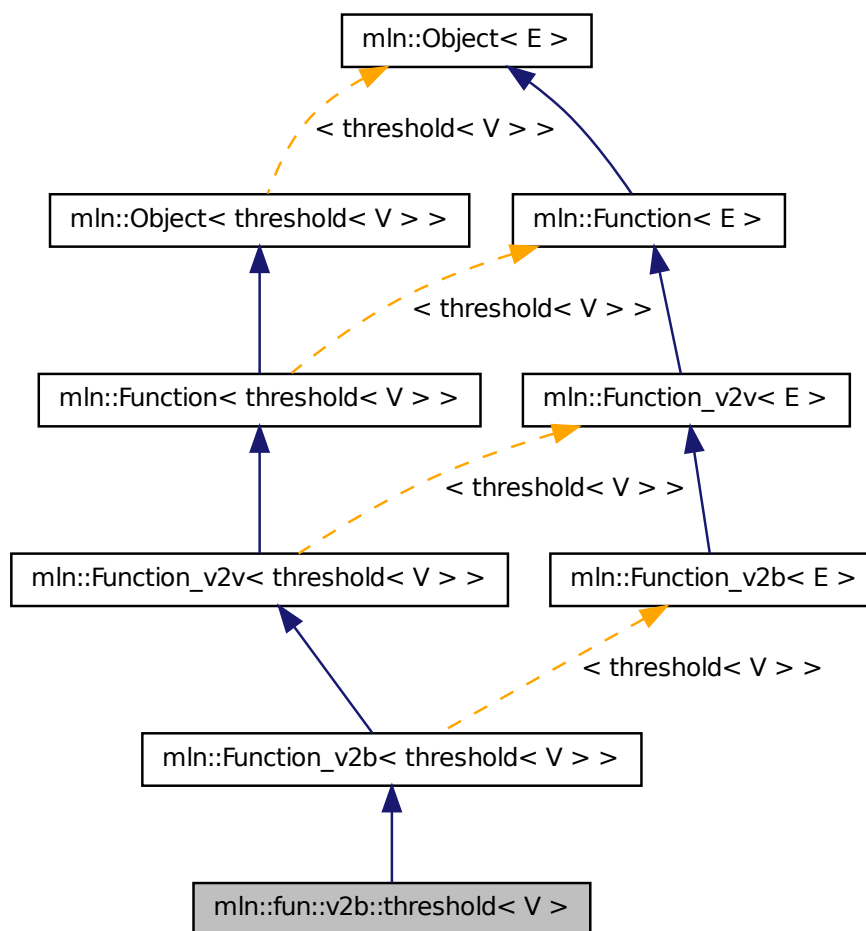
Definition at line 47 of file lnot.hh.

10.146 mln::fun::v2b::threshold< V > Struct Template Reference

Threshold function.

```
#include <threshold.hh>
```

Inheritance diagram for mln::fun::v2b::threshold< V >:



10.146.1 Detailed Description

```
template<typename V> struct mln::fun::v2b::threshold< V >
```

Threshold function. $f(v) = (v \geq \text{threshold})$.

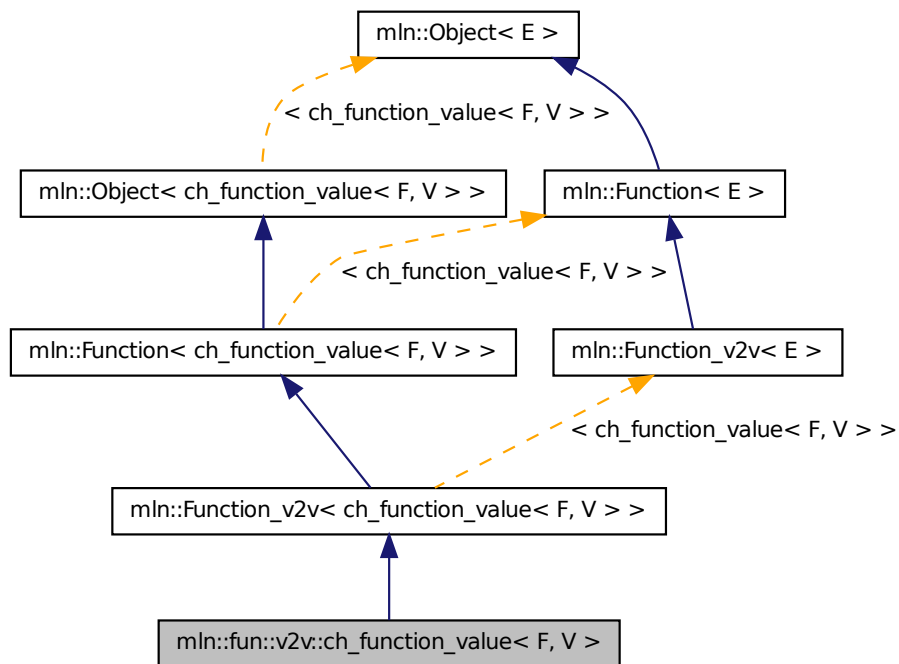
Definition at line 48 of file fun/v2b/threshold.hh.

10.147 mln::fun::v2v::ch_function_value< F, V > Class Template Reference

Wrap a function `v2v` and convert its result to another type.

```
#include <ch_function_value.hh>
```

Inheritance diagram for `mln::fun::v2v::ch_function_value< F, V >`:



10.147.1 Detailed Description

```
template<typename F, typename V> class mln::fun::v2v::ch_function_value< F, V >
```

Wrap a function `v2v` and convert its result to another type.

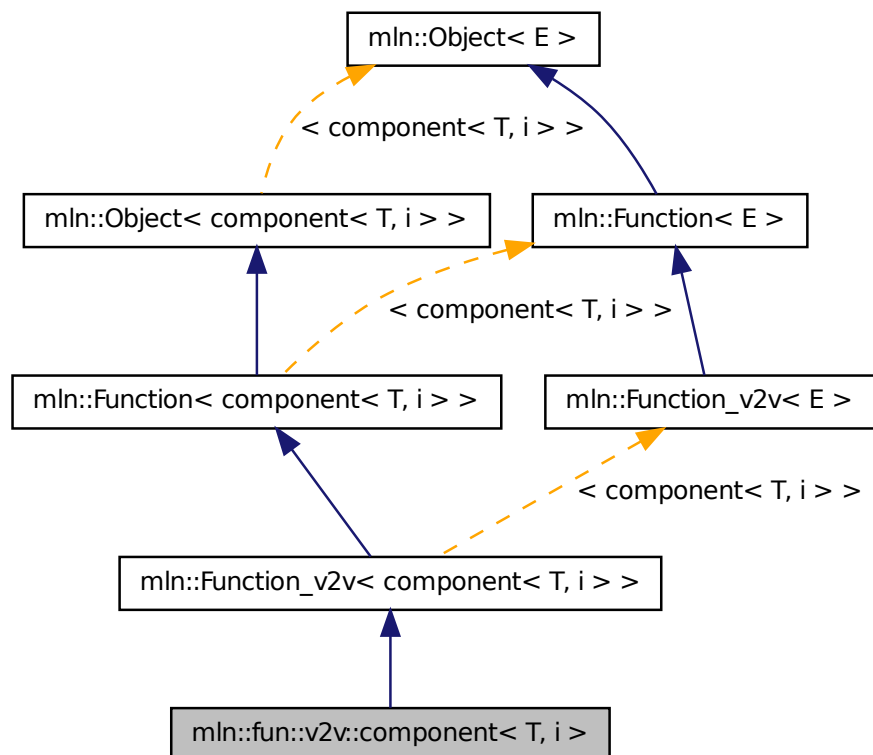
Definition at line 52 of file fun/v2v/ch_function_value.hh.

10.148 mln::fun::v2v::component< T, i > Struct Template Reference

Functor that accesses the i-th component of a value.

```
#include <component.hh>
```

Inheritance diagram for mln::fun::v2v::component< T, i >:



10.148.1 Detailed Description

```
template<typename T, unsigned i> struct mln::fun::v2v::component< T, i >
```

Functor that accesses the i-th component of a value.

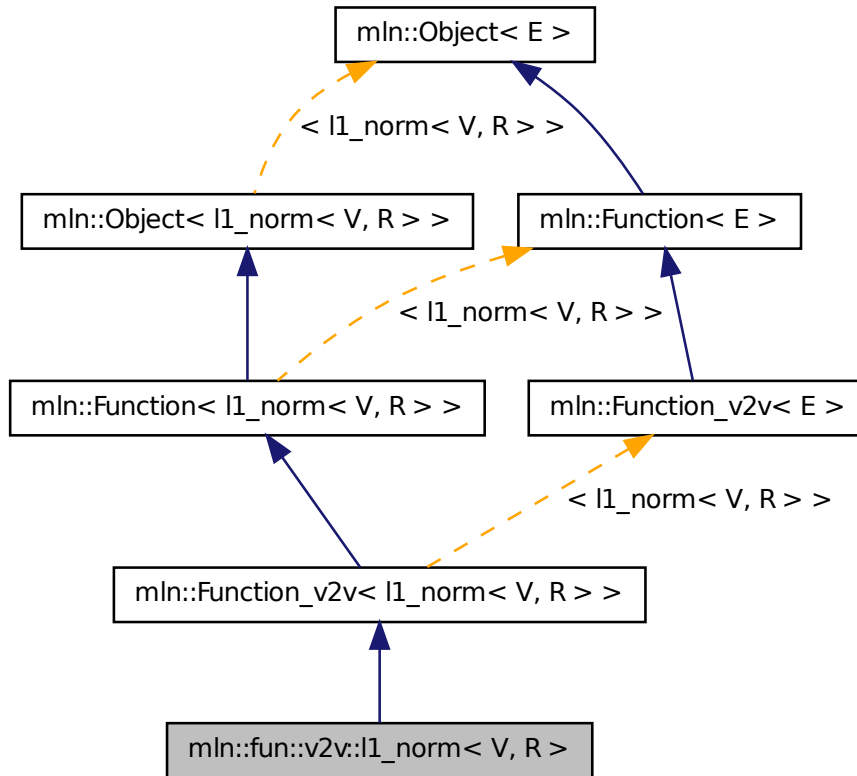
Definition at line 51 of file component.hh.

10.149 mln::fun::v2v::l1_norm< V, R > Struct Template Reference

L1-norm.

```
#include <norm.hh>
```

Inheritance diagram for `mln::fun::v2v::l1_norm< V, R >`:



10.149.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2v::l1_norm< V, R >
```

L1-norm. `V` is the type of input values; `R` is the result type.

See also

[mln::norm::l1](#).

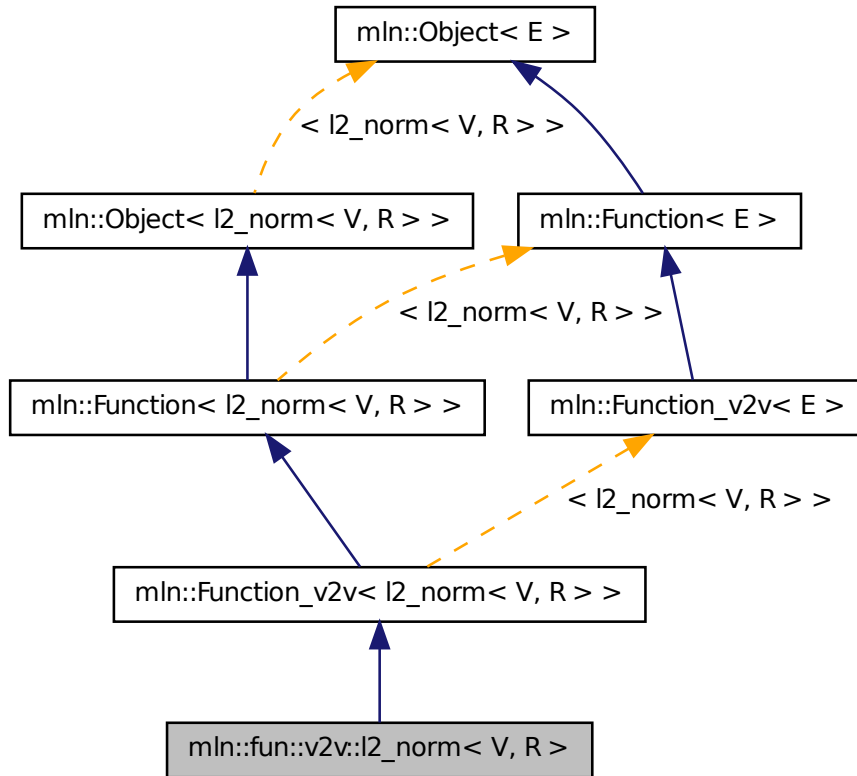
Definition at line 57 of file `v2v/norm.hh`.

10.150 `mln::fun::v2v::l2_norm< V, R >` Struct Template Reference

L2-norm.

```
#include <norm.hh>
```

Inheritance diagram for `mln::fun::v2v::l2_norm< V, R >`:



10.150.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2v::l2_norm< V, R >
```

L2-norm. V is the type of input values; R is the result type.

See also

`mln::norm::l2`.

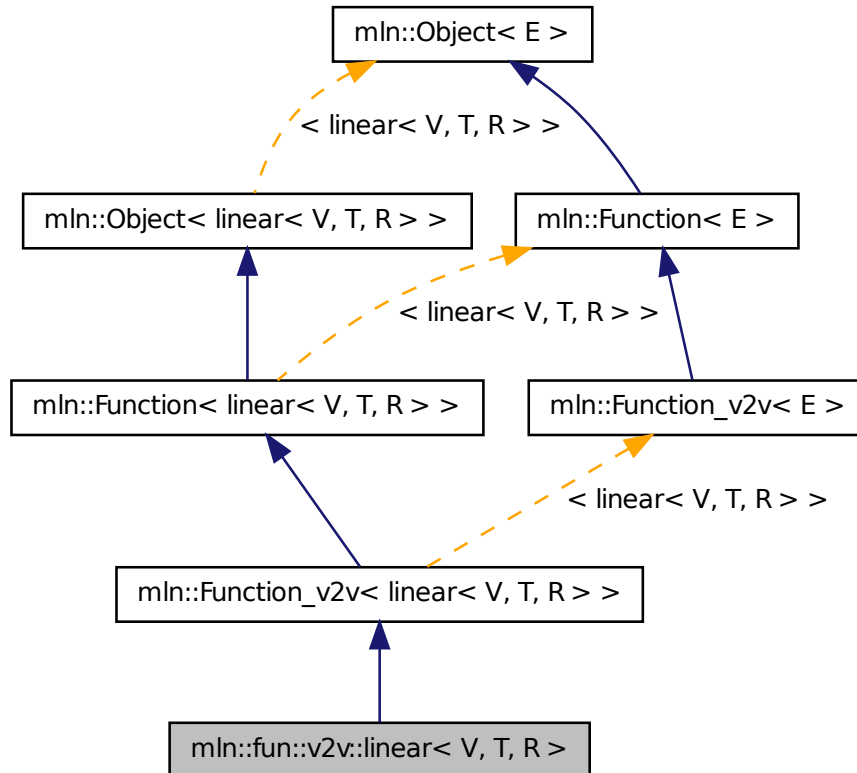
Definition at line 69 of file `v2v/norm.hh`.

10.151 `mln::fun::v2v::linear< V, T, R >` Struct Template Reference

Linear function. $f(v) = a * v + b$. V is the type of input values; T is the type used to compute the result; R is the result type.

```
#include <linear.hh>
```

Inheritance diagram for mln::fun::v2v::linear< V, T, R >:



10.151.1 Detailed Description

```
template<typename V, typename T = V, typename R = T> struct mln::fun::v2v::linear< V, T, R >
```

Linear function. $f(v) = a * v + b$. V is the type of input values; T is the type used to compute the result; R is the result type. By default, T is V and R is T .

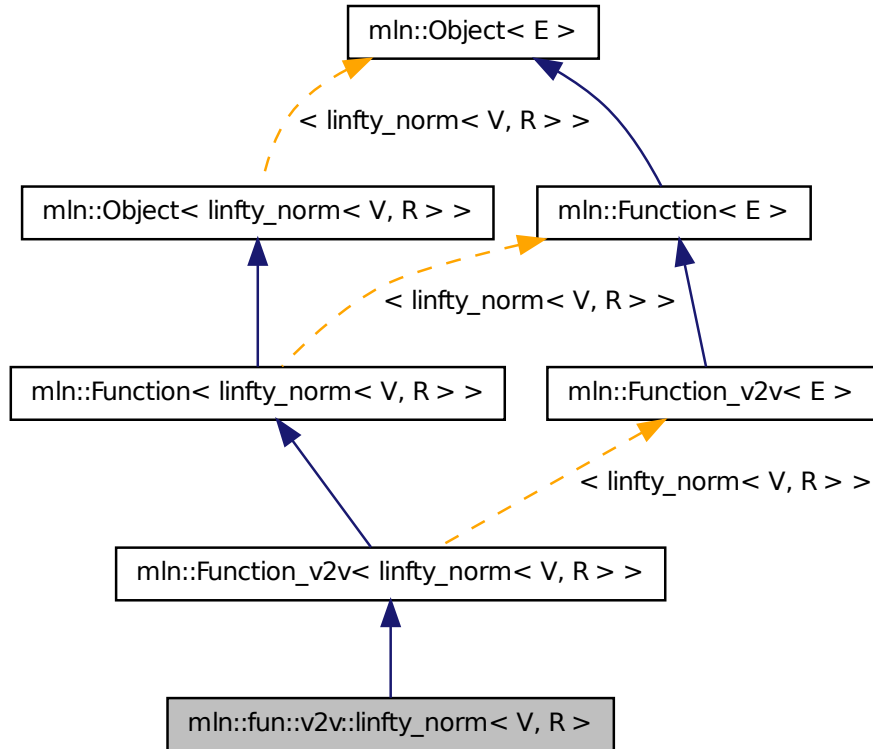
Definition at line 55 of file v2v/linear.hh.

10.152 mln::fun::v2v::linfty_norm< V, R > Struct Template Reference

L-infty norm.

```
#include <norm.hh>
```

Inheritance diagram for `mln::fun::v2v::linfty_norm< V, R >`:



10.152.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2v::linfty_norm< V, R >
```

L-infty norm. V is the type of input values; R is the result type.

See also

[mln::norm::linfty](#).

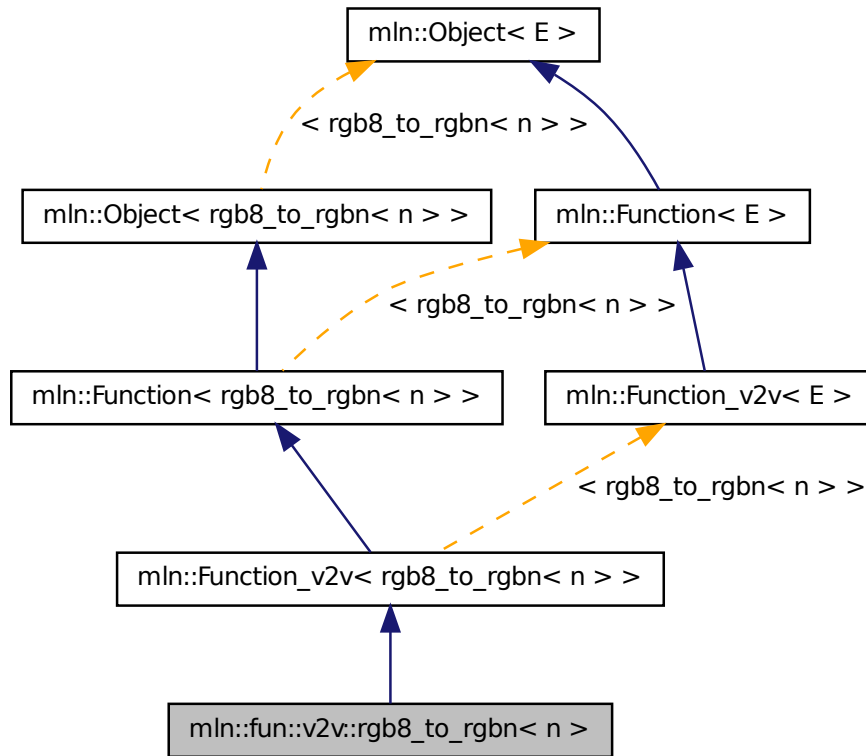
Definition at line 81 of file `v2v/norm.hh`.

10.153 `mln::fun::v2v::rgb8_to_rgbn< n >` Struct Template Reference

Convert a `rgb8` value to a `rgn`, $n < 8$.

```
#include <rgb8_to_rgbn.hh>
```

Inheritance diagram for mln::fun::v2v::rgb8_to_rgbn< n >:



Public Member Functions

- `result operator()` (const `argument` &c) const

Convert a rgb8 value to a rgn, $n < 8$.

10.153.1 Detailed Description

```
template<unsigned n> struct mln::fun::v2v::rgb8_to_rgbn< n >
```

Convert a rgb8 value to a rgn, $n < 8$.

Parameters

n defines the output quantification used for the transformation.

Definition at line 56 of file `rgb8_to_rgbn.hh`.

10.153.2 Member Function Documentation

10.153.2.1 `template<unsigned n> rgb8_to_rgn< n >::result mln::fun::v2v::rgb8_to_rgn< n >::operator() (const argument & c) const`

Convert a rgb8 value to a rgn, $n < 8$.

Parameters

[in] `v` the rgb8 value to convert.

Conversion is done by computing the size by which we divide each rgb component.

Parameters

`n` defines the output quantification used for the transformation.

Definition at line 83 of file `rgb8_to_rgn.hh`.

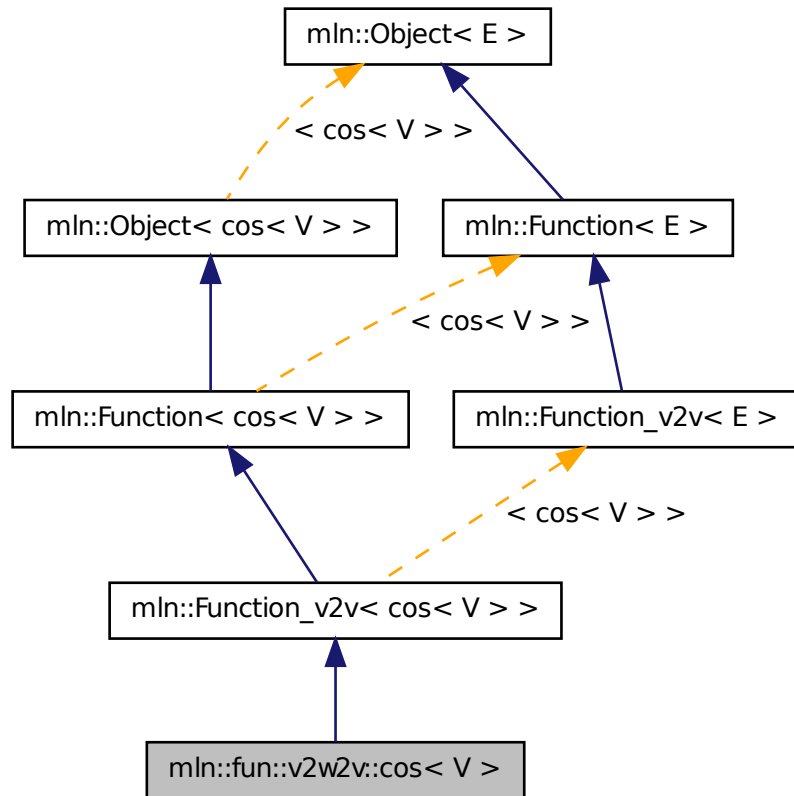
References `mln::value::rgb< n >::red()`.

10.154 `mln::fun::v2w2v::cos< V >` Struct Template Reference

Cosinus bijective functor.

```
#include <cos.hh>
```


Inheritance diagram for mln::fun::v2w2v::cos< V >:



10.154.1 Detailed Description

```
template<typename V> struct mln::fun::v2w2v::cos< V >
```

Cosinus bijective functor. `V` is the type of input values and the result type.

See also

`mln::math::cos`.

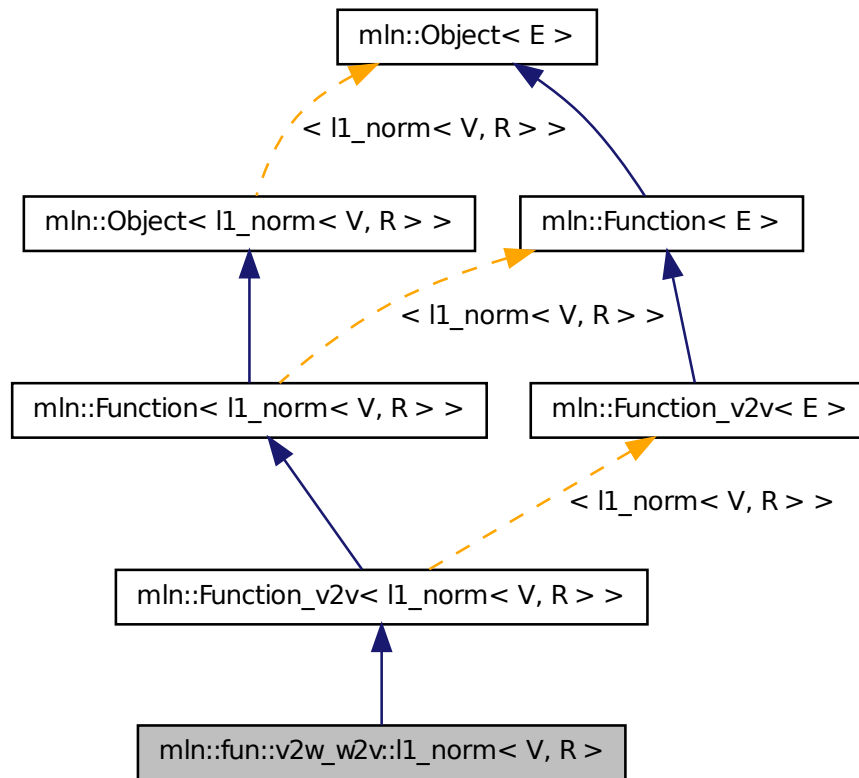
Definition at line 59 of file `fun/v2w2v/cos.hh`.

10.155 mln::fun::v2w_w2v::l1_norm< V, R > Struct Template Reference

L1-norm.

```
#include <norm.hh>
```

Inheritance diagram for `mln::fun::v2w_w2v::l1_norm< V, R >`:



10.155.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2w_w2v::l1_norm< V, R >
```

L1-norm. `V` is the type of input values; `R` is the result type.

See also

[mln::norm::l1](#).

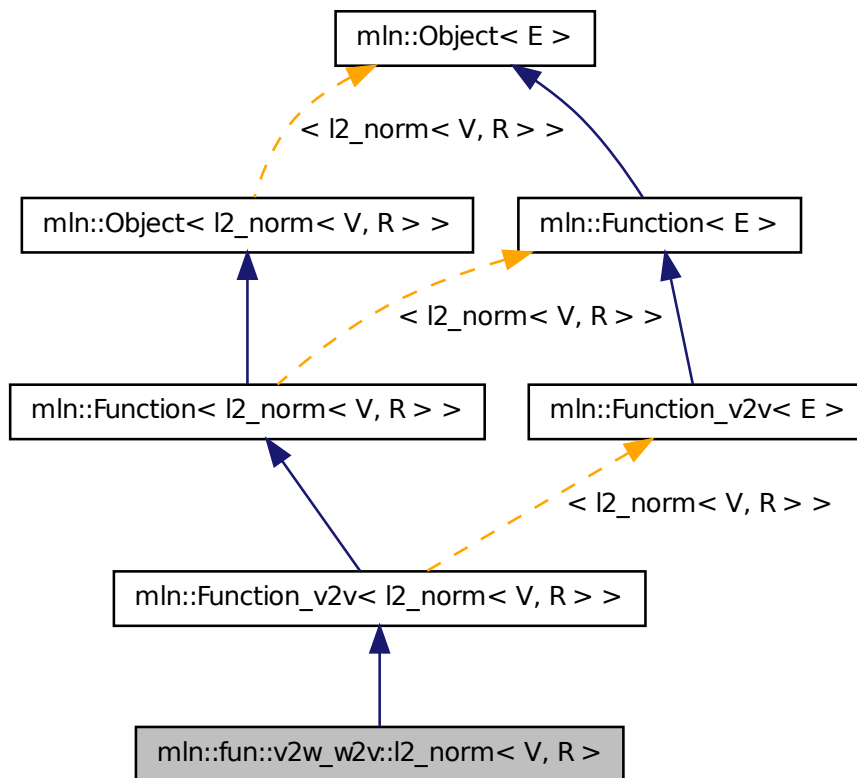
Definition at line 56 of file `v2w_w2v/norm.hh`.

10.156 mln::fun::v2w_w2v::l2_norm< V, R > Struct Template Reference

L2-norm.

```
#include <norm.hh>
```

Inheritance diagram for mln::fun::v2w_w2v::l2_norm< V, R >:



10.156.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2w_w2v::l2_norm< V, R >
```

L2-norm. V is the type of input values; R is the result type.

See also

`mln::norm::l2`.

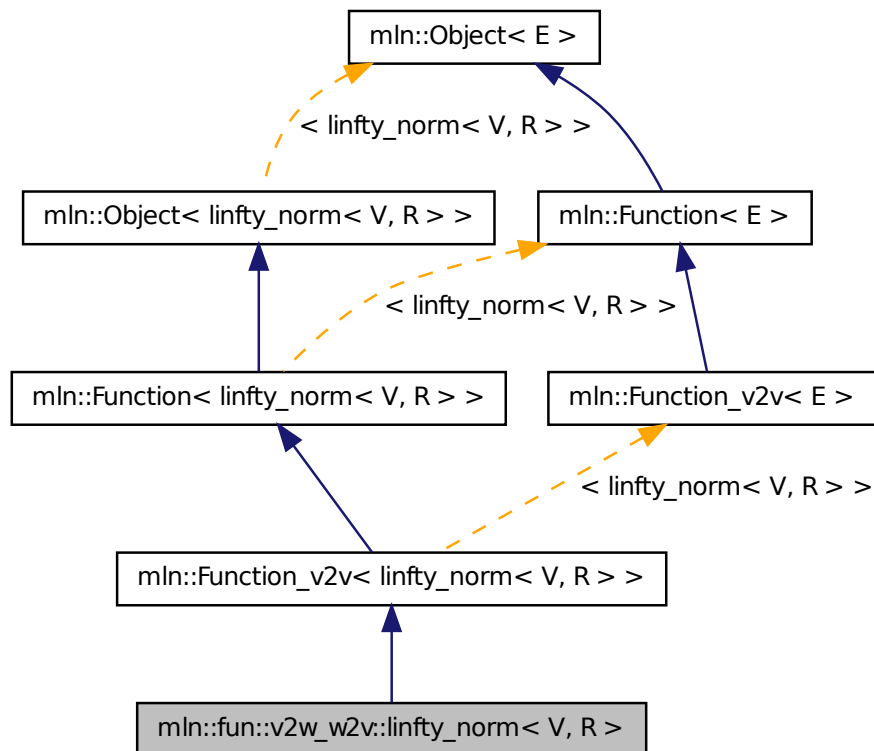
Definition at line 69 of file `v2w_w2v/norm.hh`.

10.157 mln::fun::v2w_w2v::linfty_norm< V, R > Struct Template Reference

L-infty norm.

```
#include <norm.hh>
```

Inheritance diagram for mln::fun::v2w_w2v::linfty_norm< V, R >:



10.157.1 Detailed Description

```
template<typename V, typename R> struct mln::fun::v2w_w2v::linfty_norm< V, R >
```

L-infty norm. V is the type of input values; R is the result type.

See also

[mln::norm::linfty](#).

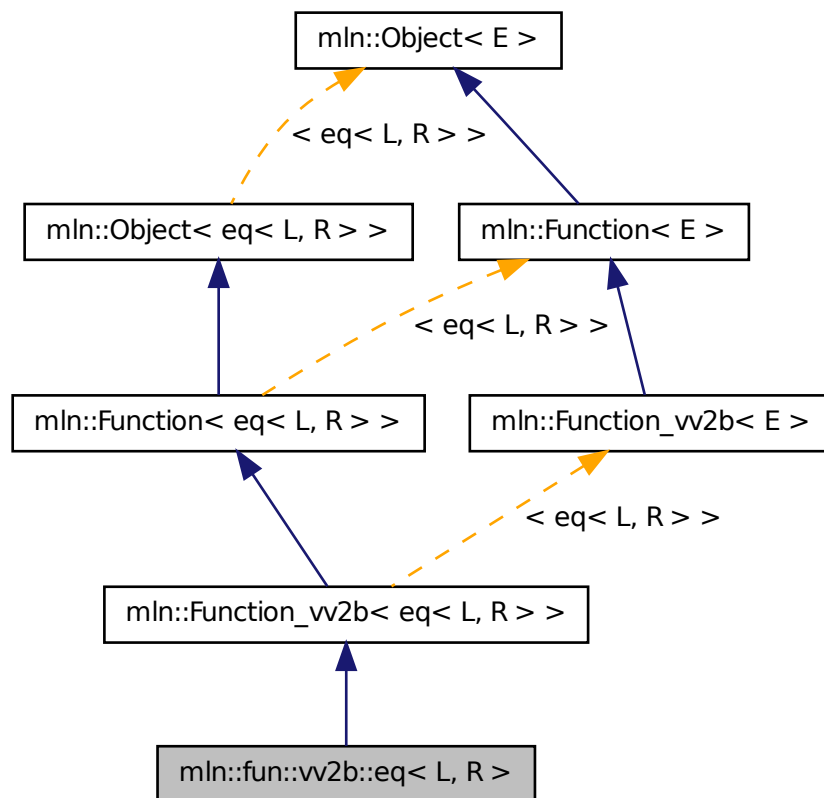
Definition at line 82 of file `v2w_w2v/norm.hh`.

10.158 mln::fun::vv2b::eq< L, R > Struct Template Reference

Functor computing equal between two values.

```
#include <eq.hh>
```

Inheritance diagram for mln::fun::vv2b::eq< L, R >:



10.158.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::eq< L, R >
```

Functor computing equal between two values.

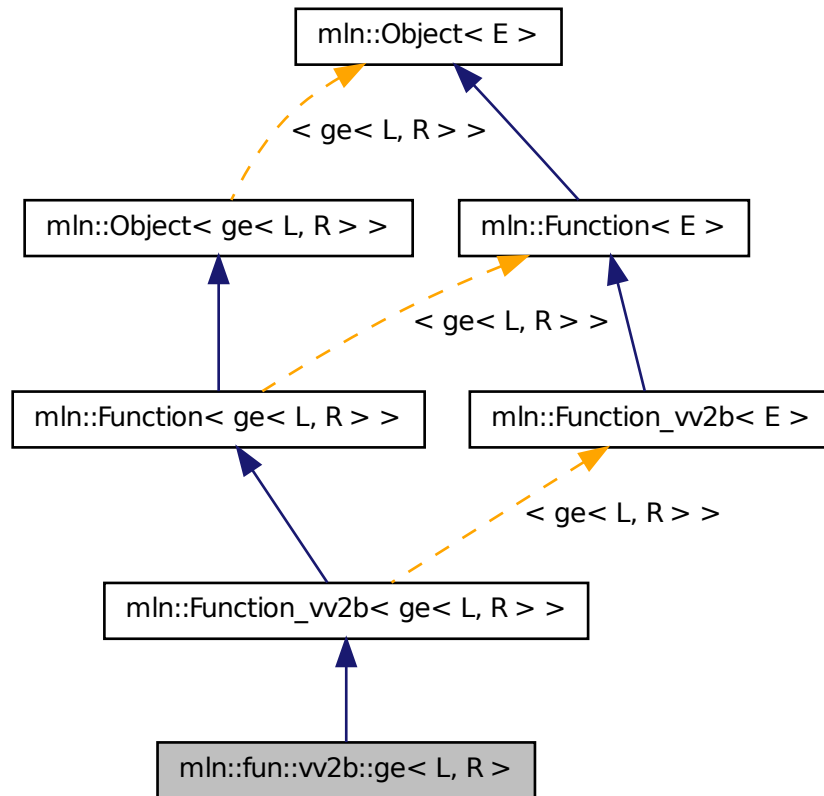
Definition at line 48 of file fun/vv2b/eq.hh.

10.159 mln::fun::vv2b::ge< L, R > Struct Template Reference

Functor computing "greater or equal than" between two values.

```
#include <ge.hh>
```

Inheritance diagram for `mln::fun::vv2b::ge< L, R >`:



10.159.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::ge< L, R >
```

Functor computing "greater or equal than" between two values.

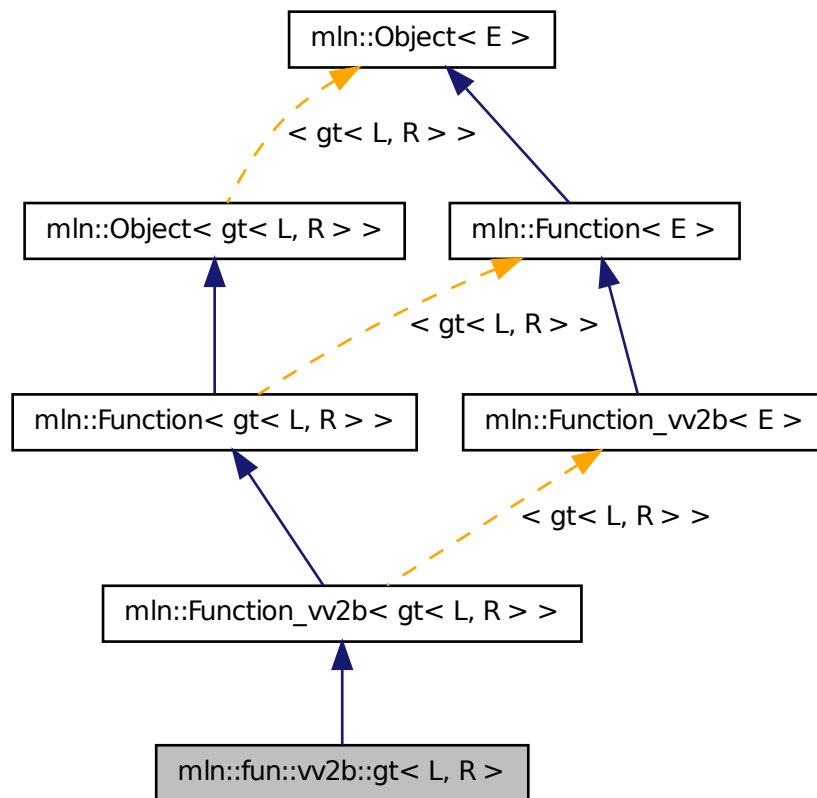
Definition at line 48 of file `ge.hh`.

10.160 `mln::fun::vv2b::gt< L, R >` Struct Template Reference

Functor computing "greater than" between two values.

```
#include <gt.hh>
```

Inheritance diagram for mln::fun::vv2b::gt< L, R >:



10.160.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::gt< L, R >
```

Functor computing "greater than" between two values.

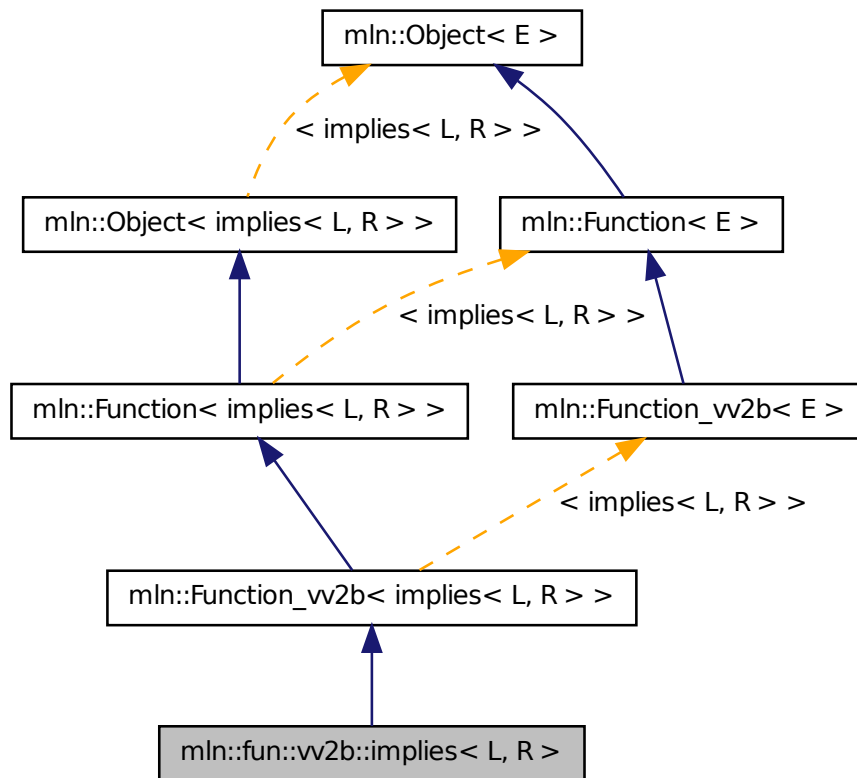
Definition at line 48 of file gt.hh.

10.161 mln::fun::vv2b::implies< L, R > Struct Template Reference

Functor computing logical-implies between two values.

```
#include <implies.hh>
```

Inheritance diagram for `mln::fun::vv2b::implies< L, R >`:



10.161.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::implies< L, R >
```

Functor computing logical-implies between two values.

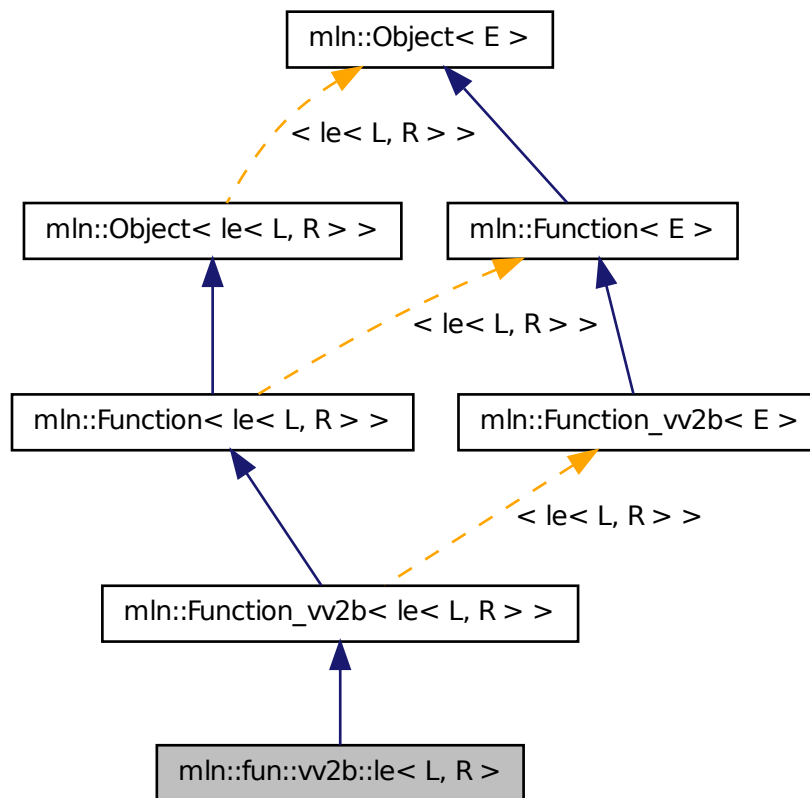
Definition at line 48 of file `implies.hh`.

10.162 `mln::fun::vv2b::le< L, R >` Struct Template Reference

Functor computing "lower or equal than" between two values.

```
#include <le.hh>
```


Inheritance diagram for mln::fun::vv2b::le< L, R >:



10.162.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::le< L, R >
```

Functor computing "lower or equal than" between two values.

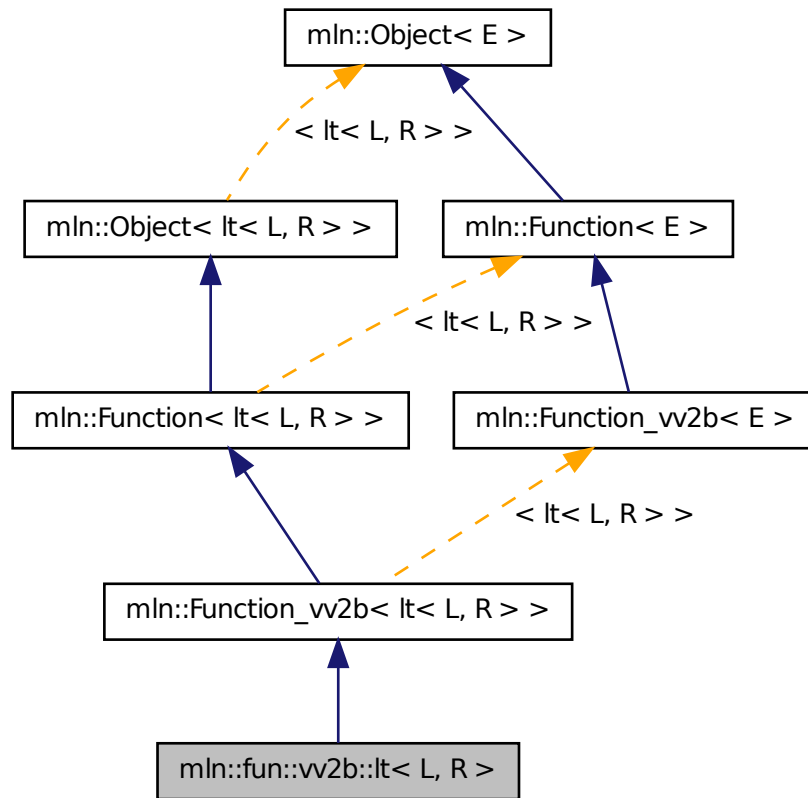
Definition at line 48 of file le.hh.

10.163 mln::fun::vv2b::lt< L, R > Struct Template Reference

Functor computing "lower than" between two values.

```
#include <lt.hh>
```

Inheritance diagram for `mln::fun::vv2b::lt< L, R >`:



10.163.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::lt< L, R >
```

Functor computing "lower than" between two values.

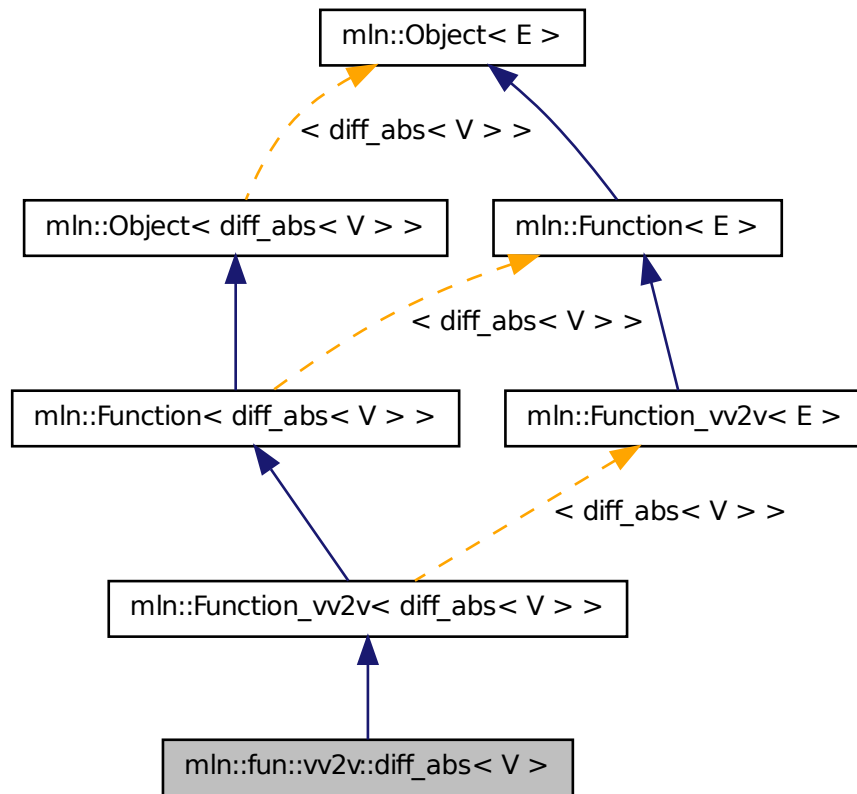
Definition at line 48 of file `lt.hh`.

10.164 `mln::fun::vv2v::diff_abs< V >` Struct Template Reference

A functor computing the `diff_absimum` of two values.

```
#include <diff_abs.hh>
```

Inheritance diagram for mln::fun::vv2v::diff_abs< V >:



10.164.1 Detailed Description

```
template<typename V> struct mln::fun::vv2v::diff_abs< V >
```

A functor computing the `diff_abs`imum of two values.

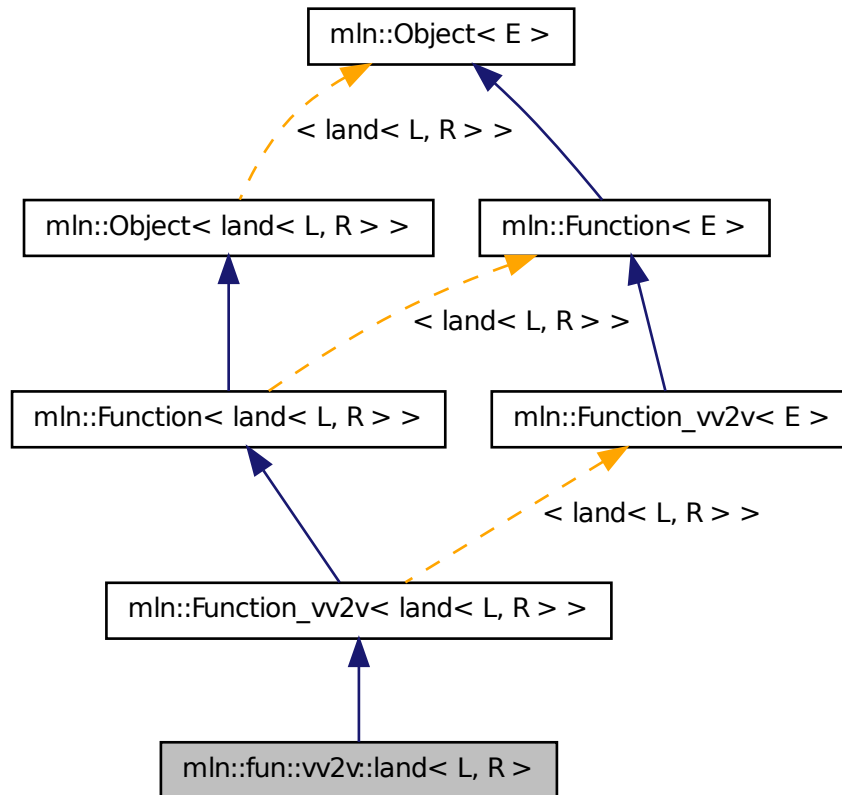
Definition at line 50 of file `fun/vv2v/diff_abs.hh`.

10.165 mln::fun::vv2v::land< L, R > Struct Template Reference

Functor computing logical-and between two values.

```
#include <land.hh>
```

Inheritance diagram for `mln::fun::vv2v::land< L, R >`:



10.165.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::land< L, R >
```

Functor computing logical-and between two values.

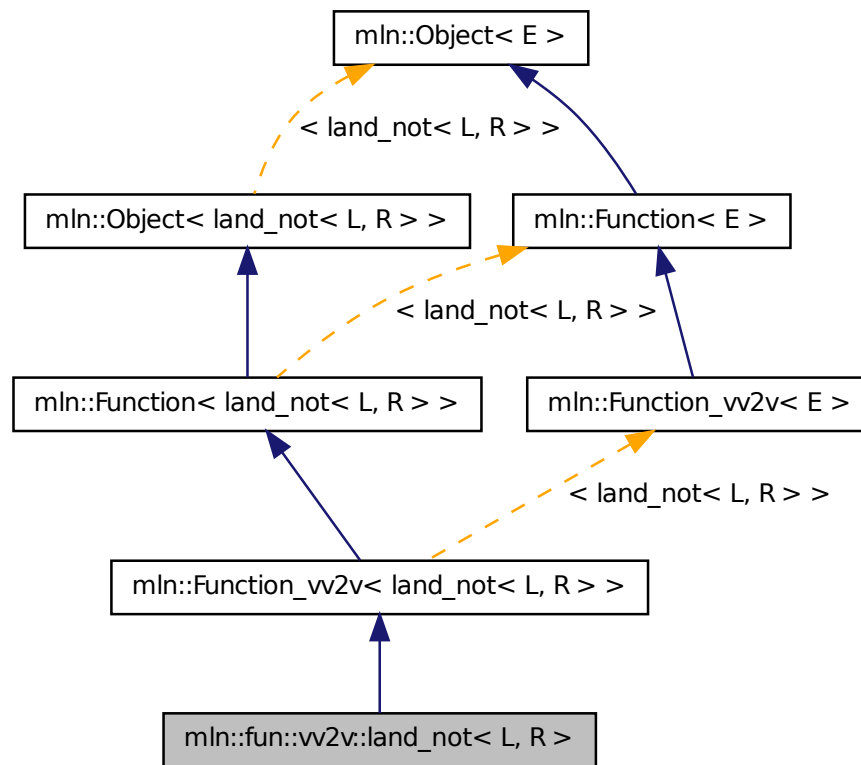
Definition at line 48 of file `fun/vv2v/land.hh`.

10.166 `mln::fun::vv2v::land_not< L, R >` Struct Template Reference

Functor computing logical and-not between two values.

```
#include <land_not.hh>
```

Inheritance diagram for mln::fun::vv2v::land_not< L, R >:



10.166.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::land_not< L, R >
```

Functor computing logical and-not between two values.

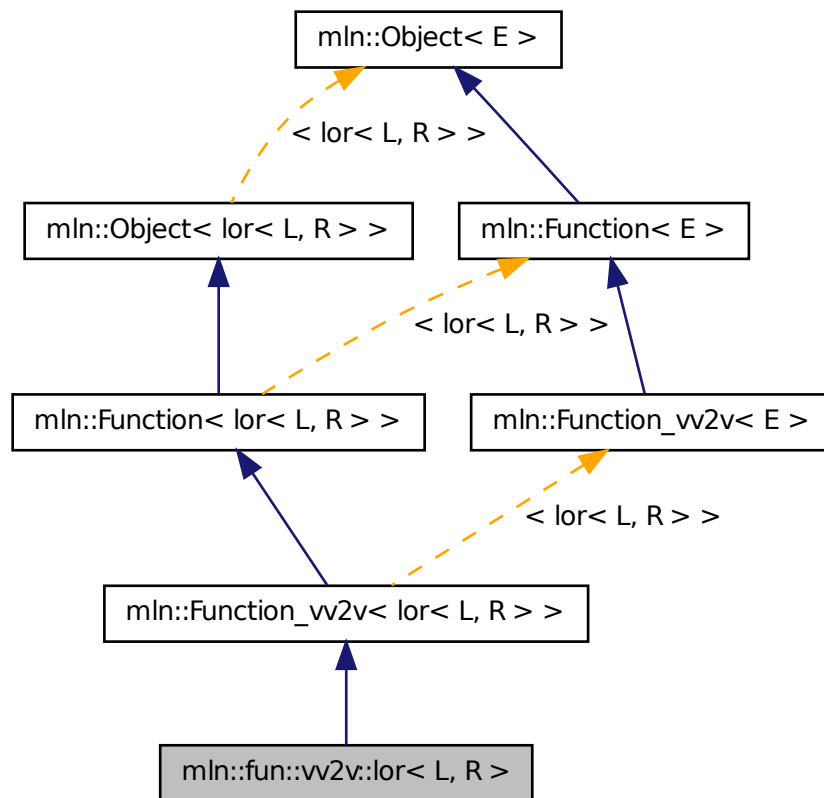
Definition at line 49 of file land_not.hh.

10.167 mln::fun::vv2v::lor< L, R > Struct Template Reference

Functor computing logical-or between two values.

```
#include <lor.hh>
```

Inheritance diagram for `mln::fun::vv2v::lor< L, R >`:



10.167.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::lor< L, R >
```

Functor computing logical-or between two values.

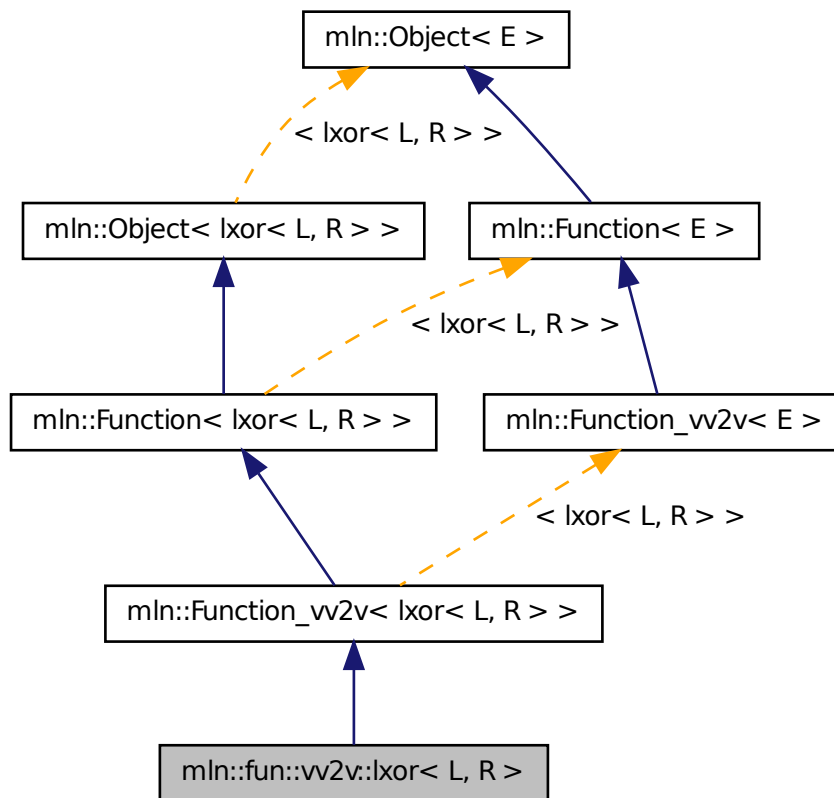
Definition at line 48 of file `fun/vv2v/lor.hh`.

10.168 `mln::fun::vv2v::lxor< L, R >` Struct Template Reference

Functor computing logical-xor between two values.

```
#include <lxor.hh>
```

Inheritance diagram for mln::fun::vv2v::lxor< L, R >:



10.168.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::lxor< L, R >
```

Functor computing logical-xor between two values.

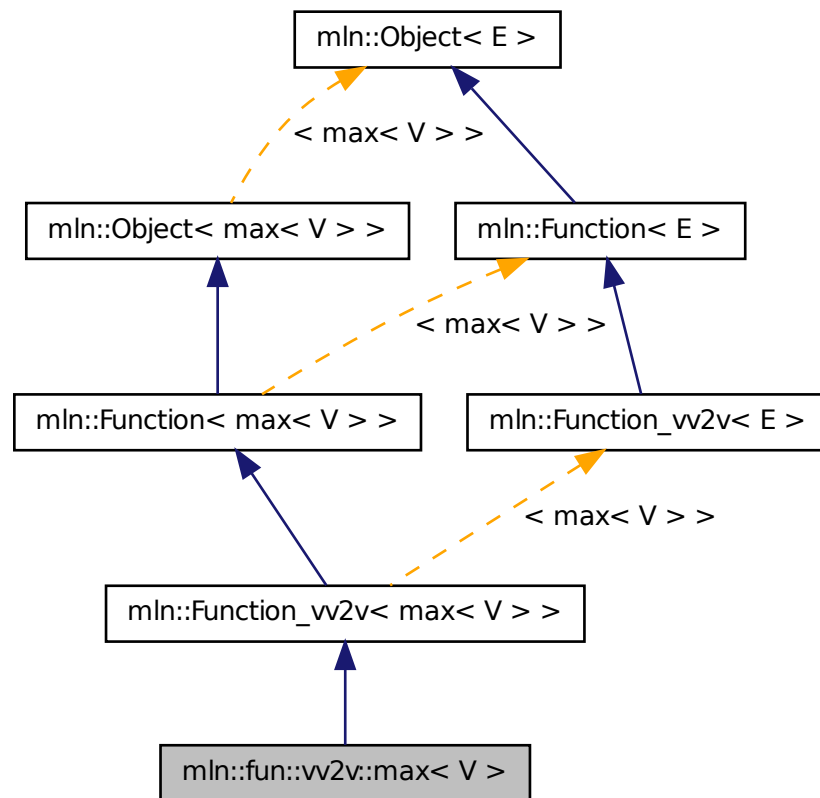
Definition at line 48 of file lxor.hh.

10.169 mln::fun::vv2v::max< V > Struct Template Reference

A functor computing the maximum of two values.

```
#include <max.hh>
```

Inheritance diagram for `mln::fun::vv2v::max< V >`:



10.169.1 Detailed Description

```
template<typename V> struct mln::fun::vv2v::max< V >
```

A functor computing the maximum of two values.

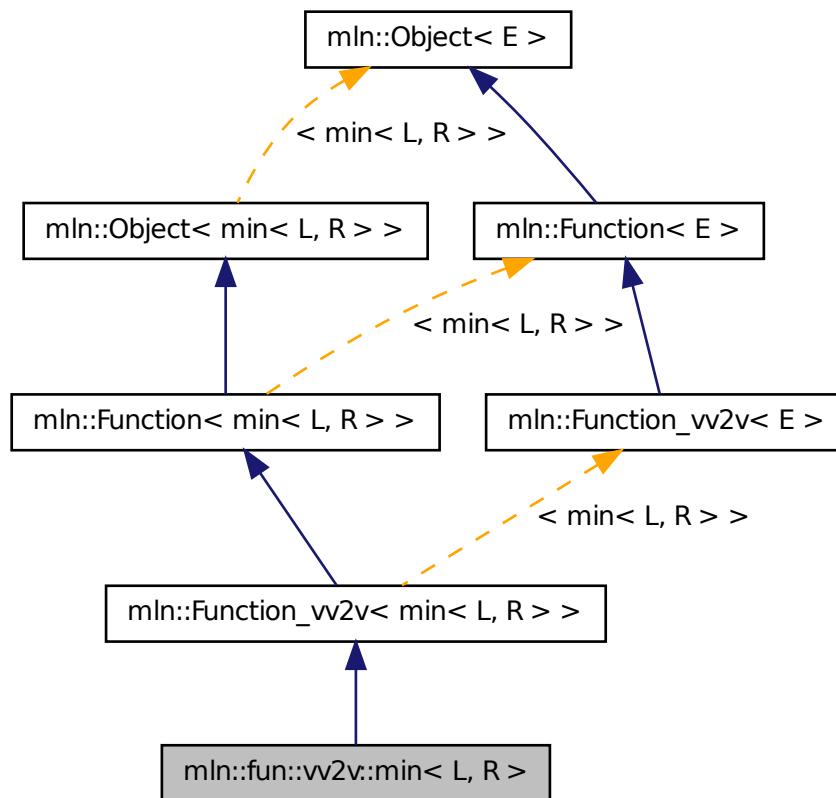
Definition at line 49 of file `fun/vv2v/max.hh`.

10.170 `mln::fun::vv2v::min< L, R >` Struct Template Reference

A functor computing the minimum of two values.

```
#include <min.hh>
```


Inheritance diagram for mln::fun::vv2v::min< L, R >:



10.170.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::min< L, R >
```

A functor computing the minimum of two values.

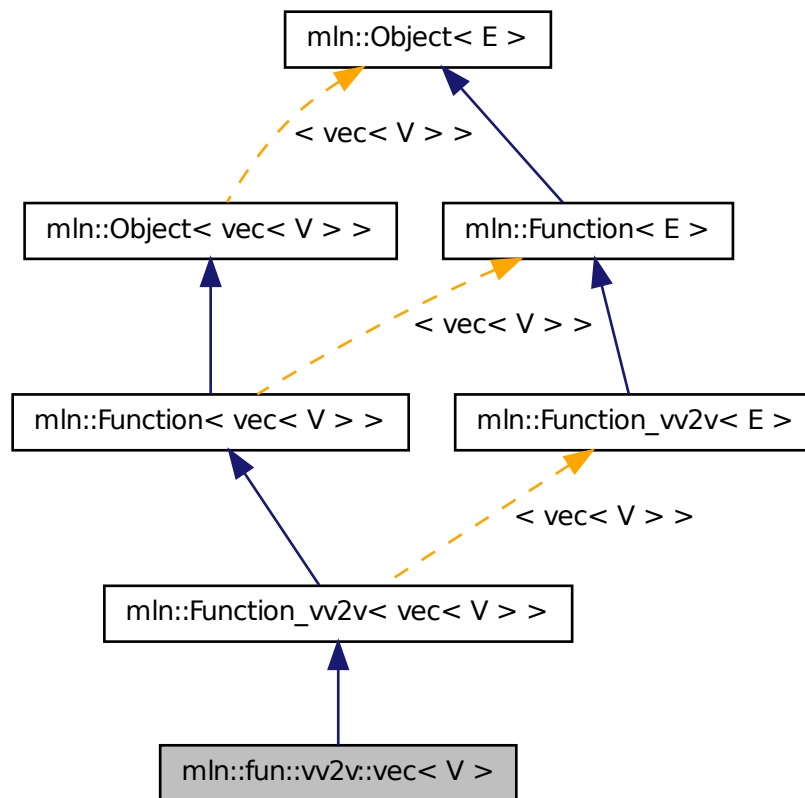
Definition at line 50 of file `fun/vv2v/min.hh`.

10.171 mln::fun::vv2v::vec< V > Struct Template Reference

A functor computing the vecimum of two values.

```
#include <vec.hh>
```

Inheritance diagram for `mln::fun::vv2v::vec< V >`:



10.171.1 Detailed Description

```
template<typename V> struct mln::fun::vv2v::vec< V >
```

A functor computing the vecimum of two values.

Definition at line 50 of file `fun/vv2v/vec.hh`.

10.172 `mln::fun::x2p::closest_point< P >` Struct Template Reference

FIXME: doxygen + concept checking.

```
#include <closest_point.hh>
```

10.172.1 Detailed Description

`template<typename P> struct mln::fun::x2p::closest_point< P >`

FIXME: doxygen + concept checking.

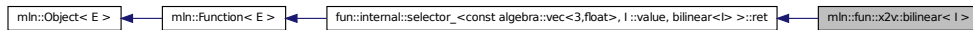
Definition at line 45 of file closest_point.hh.

10.173 mln::fun::x2v::bilinear< I > Struct Template Reference

Represent a bilinear interolation of values from an underlying image.

```
#include <bilinear.hh>
```

Inheritance diagram for mln::fun::x2v::bilinear< I >:



Public Member Functions

- `template<typename T >`
`I::value operator() (const algebra::vec< 3, T > &v) const`
Bilinear filtering on 3d images. Work on slices.
- `template<typename T >`
`I::value operator() (const algebra::vec< 2, T > &v) const`
Bilinear filtering on 2d images.

10.173.1 Detailed Description

`template<typename I> struct mln::fun::x2v::bilinear< I >`

Represent a bilinear interolation of values from an underlying image.

Definition at line 52 of file bilinear.hh.

10.173.2 Member Function Documentation

10.173.2.1 `template<typename I > template<typename T > I::value mln::fun::x2v::bilinear< I >::operator() (const algebra::vec< 2, T > & v) const`

Bilinear filtering on 2d images.

Definition at line 86 of file bilinear.hh.

10.173.2 `template<typename I> template<typename T> I::value mln::fun::x2v::bilinear< I >::operator() (const algebra::vec< 3, T > & v) const`

Bilinear filtering on 3d images. Work on slices.

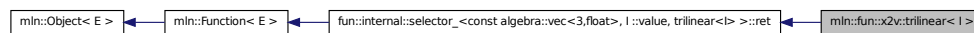
Definition at line 132 of file bilinear.hh.

10.174 `mln::fun::x2v::trilinear< I >` Struct Template Reference

Represent a trilinear interolation of values from an underlying image.

```
#include <trilinear.hh>
```

Inheritance diagram for `mln::fun::x2v::trilinear< I >`:



10.174.1 Detailed Description

```
template<typename I> struct mln::fun::x2v::trilinear< I >
```

Represent a trilinear interolation of values from an underlying image.

Definition at line 53 of file trilinear.hh.

10.175 `mln::fun::x2x::composed< T2, T1 >` Struct Template Reference

Represent a composition of two transformations.

```
#include <composed.hh>
```

Public Member Functions

- `composed ()`
Constructor without argument.
- `composed (const T2 &f, const T1 &g)`
Constructor with the two transformation to be composed.

10.175.1 Detailed Description

```
template<typename T2, typename T1> struct mln::fun::x2x::composed< T2, T1 >
```

Represent a composition of two transformations.

Definition at line 144 of file composed.hh.

10.175.2 Constructor & Destructor Documentation

10.175.2.1 `template<typename T2, typename T1> mln::fun::x2x::composed< T2, T1 >::composed () [inline]`

Constructor without argument.

Definition at line 153 of file composed.hh.

10.175.2.2 `template<typename T2, typename T1> mln::fun::x2x::composed< T2, T1 >::composed (const T2 & f, const T1 & g) [inline]`

Constructor with the two transformation to be composed.

Definition at line 156 of file composed.hh.

10.176 mln::fun::x2x::linear< I > Struct Template Reference

Represent a linear interolation of values from an underlying image.

```
#include <linear.hh>
```

Inheritance diagram for mln::fun::x2x::linear< I >:



Public Member Functions

- `linear` (const I & *ima*)

Constructor with the underlying image.

- `template<typename C > I::value operator() (const algebra::vec< 1, C > &v) const`

Return the interpolated value in the underlying image at the given 'point' v.

Public Attributes

- const I & *ima*

Underlying image.

10.176.1 Detailed Description

template<typename I> struct mln::fun::x2x::linear< I >

Represent a linear interpolation of values from an underlying image.

Definition at line 53 of file x2v/linear.hh.

10.176.2 Constructor & Destructor Documentation

10.176.2.1 template<typename I > mln::fun::x2x::linear< I >::linear (const I & *ima*)

Constructor with the underlying image.

Definition at line 77 of file x2v/linear.hh.

10.176.3 Member Function Documentation

10.176.3.1 template<typename I > template<typename C > I::value mln::fun::x2x::linear< I >::operator() (const algebra::vec< 1, C > & *v*) const

Return the interpolated value in the underlying image at the given 'point' *v*.

Definition at line 85 of file x2v/linear.hh.

10.176.4 Member Data Documentation

10.176.4.1 template<typename I > const I& mln::fun::x2x::linear< I >::ima

Underlying image.

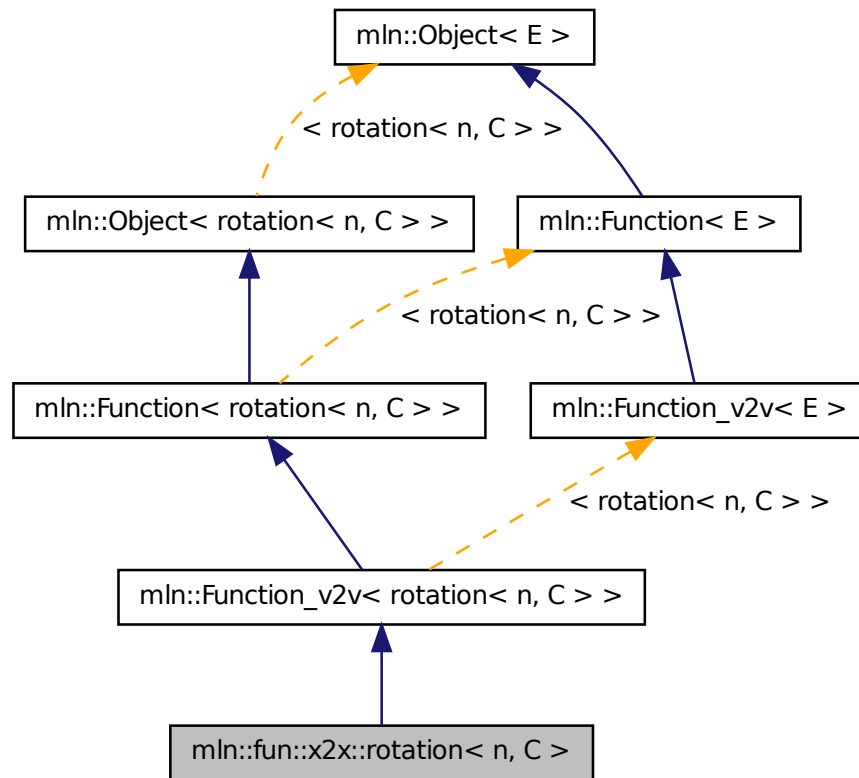
Definition at line 70 of file x2v/linear.hh.

10.177 mln::fun::x2x::rotation< n, C > Struct Template Reference

Represent a rotation function.

```
#include <rotation.hh>
```

Inheritance diagram for mln::fun::x2x::rotation< n, C >:



Public Types

- typedef C [data_t](#)
Type of the underlying data stored in vectors and matrices.
- typedef [rotation< n, C >](#) [invert](#)
Type of the inverse function.

Public Member Functions

- [invert inv \(\)](#) const
Return the inverse function.
- [algebra::vec< n, C >](#) [operator\(\)](#) (const [algebra::vec< n, C >](#) &v) const
Perform the rotation of the given vector.

- [rotation](#) ()
Constructor without argument.
- [rotation](#) (const algebra::quat &q)
Constructor with quaternion.
- [rotation](#) (const algebra::h_mat< n, C > &m)
Constructor with h_mat.
- [rotation](#) (C alpha, const algebra::vec< n, C > &axis)
Constructor with radian alpha and a facultative direction (rotation axis).
- void [set_alpha](#) (C alpha)
Set a new grade alpha.
- void [set_axis](#) (const algebra::vec< n, C > &axis)
Set a new rotation axis.

10.177.1 Detailed Description

`template<unsigned n, typename C> struct mln::fun::x2x::rotation< n, C >`

Represent a rotation function.

Definition at line 147 of file rotation.hh.

10.177.2 Member Typedef Documentation

10.177.2.1 `template<unsigned n, typename C > typedef C mln::fun::x2x::rotation< n, C >::data_t`

Type of the underlying data stored in vectors and matrices.

Definition at line 152 of file rotation.hh.

10.177.2.2 `template<unsigned n, typename C > typedef rotation<n,C> mln::fun::x2x::rotation< n, C >::invert`

Type of the inverse function.

Definition at line 155 of file rotation.hh.

10.177.3 Constructor & Destructor Documentation

10.177.3.1 `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation ()`
`[inline]`

Constructor without argument.

Definition at line 192 of file rotation.hh.

10.177.3.2 `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation (C alpha, const algebra::vec< n, C > & axis) [inline]`

Constructor with radian alpha and a facultative direction (rotation axis).

Definition at line 198 of file rotation.hh.

10.177.3.3 `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation (const algebra::quat & q) [inline]`

Constructor with quaternion.

Definition at line 208 of file rotation.hh.

References mln::make::h_mat().

10.177.3.4 `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation (const algebra::h_mat< n, C > & m) [inline]`

Constructor with h_mat.

Definition at line 238 of file rotation.hh.

10.177.4 Member Function Documentation

10.177.4.1 `template<unsigned n, typename C > rotation< n, C > mln::fun::x2x::rotation< n, C >::inv () const [inline]`

Return the inverse function.

Definition at line 265 of file rotation.hh.

10.177.4.2 `template<unsigned n, typename C > algebra::vec< n, C > mln::fun::x2x::rotation< n, C >::operator() (const algebra::vec< n, C > & v) const [inline]`

Perform the rotation of the given vector.

Definition at line 247 of file rotation.hh.

10.177.4.3 `template<unsigned n, typename C > void mln::fun::x2x::rotation< n, C >::set_alpha (C alpha) [inline]`

Set a new grade alpha.

Definition at line 274 of file rotation.hh.

10.177.4.4 `template<unsigned n, typename C > void mln::fun::x2x::rotation< n, C >::set_axis (const algebra::vec< n, C > & axis) [inline]`

Set a new rotation axis.

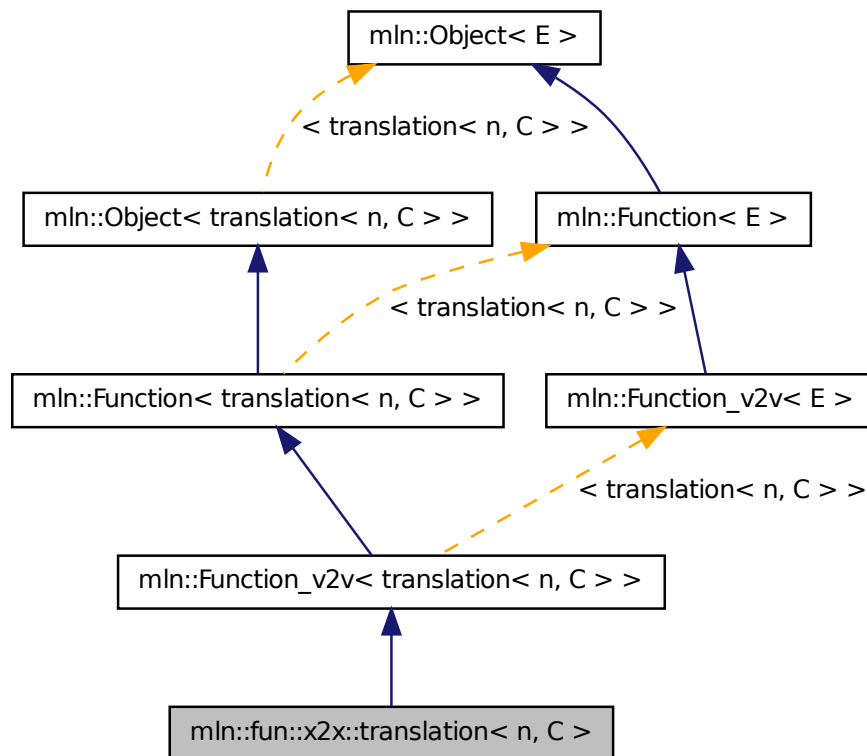
Definition at line 283 of file rotation.hh.

10.178 mln::fun::x2x::translation< n, C > Struct Template Reference

Translation function-object.

```
#include <translation.hh>
```

Inheritance diagram for mln::fun::x2x::translation< n, C >:



Public Types

- typedef C `data_t`
Type of the underlying data stored in vectors and matrices.
- typedef `translation< n, C >` `invert`
Type of the inverse function.

Public Member Functions

- `invert inv () const`

Return the inverse function.

- algebra::vec< n, C > [operator\(\)](#) (const algebra::vec< n, C > &v) const
Perform the translation of the given vector.
- void [set_t](#) (const algebra::vec< n, C > &t)
Set a net translation vector.
- const algebra::vec< n, C > & [t](#) () const
Return the translation vector.
- [translation](#) (const algebra::vec< n, C > &t)
Constructor with the translation vector.
- [translation](#) ()
Constructor without argument.

10.178.1 Detailed Description

template<unsigned n, typename C> struct mln::fun::x2x::translation< n, C >

Translation function-object.

Definition at line 52 of file x2x/translation.hh.

10.178.2 Member Typedef Documentation

10.178.2.1 template<unsigned n, typename C > typedef C mln::fun::x2x::translation< n, C >::data_t

Type of the underlying data stored in vectors and matrices.

Definition at line 59 of file x2x/translation.hh.

10.178.2.2 template<unsigned n, typename C > typedef translation<n,C> mln::fun::x2x::translation< n, C >::invert

Type of the inverse function.

Definition at line 62 of file x2x/translation.hh.

10.178.3 Constructor & Destructor Documentation

10.178.3.1 template<unsigned n, typename C > mln::fun::x2x::translation< n, C >::translation () [inline]

Constructor without argument.

Definition at line 93 of file x2x/translation.hh.

10.178.3.2 `template<unsigned n, typename C > mln::fun::x2x::translation< n, C >::translation (const algebra::vec< n, C > & t) [inline]`

Constructor with the translation vector.

Definition at line 99 of file x2x/translation.hh.

10.178.4 Member Function Documentation

10.178.4.1 `template<unsigned n, typename C > translation< n, C > mln::fun::x2x::translation< n, C >::inv () const [inline]`

Return the inverse function.

Definition at line 124 of file x2x/translation.hh.

10.178.4.2 `template<unsigned n, typename C > algebra::vec< n, C > mln::fun::x2x::translation< n, C >::operator() (const algebra::vec< n, C > & v) const [inline]`

Perform the translation of the given vector.

Definition at line 108 of file x2x/translation.hh.

10.178.4.3 `template<unsigned n, typename C > void mln::fun::x2x::translation< n, C >::set_t (const algebra::vec< n, C > & t) [inline]`

Set a net translation vector.

Definition at line 134 of file x2x/translation.hh.

10.178.4.4 `template<unsigned n, typename C > const algebra::vec< n, C > & mln::fun::x2x::translation< n, C >::t () const [inline]`

Return the translation vector.

Definition at line 143 of file x2x/translation.hh.

10.179 mln::fun_image< F, I > Struct Template Reference

[Image](#) read through a function.

```
#include <fun_image.hh>
```

Inherits `image_value_morpher< I, F::result, fun_image< F, I > >`.

Public Types

- typedef `F::result` [lvalue](#)
Return type of read-write access.
- typedef `F::result` [rvalue](#)

Return type of read-only access.

- typedef [fun_image](#)< tag::value_< typename F::result >, tag::image_< I > > [skeleton](#)
Skeleton.
- typedef F::result [value](#)
Value associated type.

Public Member Functions

- [fun_image](#) ()
Constructor.
- [fun_image](#) (const [Function_v2v](#)< F > &f, const [Image](#)< I > &ima)
Constructor.
- [fun_image](#) (const [Image](#)< I > &ima)
Constructor.
- F::result [operator](#)() (const typename I::psite &p) const
Read-only access of pixel value at point site p.
- F::result [operator](#)() (const typename I::psite &p)
Mutable access is for reading only.

10.179.1 Detailed Description

template<typename F, typename I> struct mln::fun_image< F, I >

[Image](#) read through a function.

Definition at line 101 of file fun_image.hh.

10.179.2 Member Typedef Documentation

10.179.2.1 template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::lvalue

Return type of read-write access.

Definition at line 111 of file fun_image.hh.

10.179.2.2 template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::rvalue

Return type of read-only access.

Definition at line 108 of file fun_image.hh.

10.179.2.3 `template<typename F, typename I> typedef fun_image< tag::value_<typename F
::result>, tag::image_<I> > mln::fun_image< F, I >::skeleton`

Skeleton.

Definition at line 115 of file fun_image.hh.

10.179.2.4 `template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::value`

Value associated type.

Definition at line 105 of file fun_image.hh.

10.179.3 Constructor & Destructor Documentation

10.179.3.1 `template<typename F , typename I > mln::fun_image< F, I >::fun_image ()
[inline]`

Constructor.

Definition at line 177 of file fun_image.hh.

10.179.3.2 `template<typename F , typename I > mln::fun_image< F, I >::fun_image (const
Function_v2v< F > & f, const Image< I > & ima) [inline]`

Constructor.

Definition at line 184 of file fun_image.hh.

10.179.3.3 `template<typename F , typename I > mln::fun_image< F, I >::fun_image (const
Image< I > & ima) [inline]`

Constructor.

Definition at line 191 of file fun_image.hh.

10.179.4 Member Function Documentation

10.179.4.1 `template<typename F , typename I > F::result mln::fun_image< F, I >::operator() (const
typename I::psite & p) const [inline]`

Read-only access of pixel value at point site p.

Definition at line 209 of file fun_image.hh.

10.179.4.2 `template<typename F , typename I > F::result mln::fun_image< F, I >::operator() (const
typename I::psite & p) [inline]`

Mutable access is for reading only.

Definition at line 218 of file fun_image.hh.

10.180 `mln::Function< E >` Struct Template Reference

Base class for implementation of function-objects.

```
#include <function.hh>
```

Inherits [mln::Object< E >](#).

Inherited by [mln::Function_n2v< E >](#), [mln::Function_v2v< E >](#), [mln::Function_vv2b< E >](#), and [mln::Function_vv2v< E >](#).

Protected Member Functions

- [Function \(\)](#)

An operator() has to be provided.

10.180.1 Detailed Description

```
template<typename E> struct mln::Function< E >
```

Base class for implementation of function-objects. The parameter *E* is the exact type.

Definition at line 64 of file `function.hh`.

10.180.2 Constructor & Destructor Documentation

10.180.2.1 `template<typename E > mln::Function< E >::Function () [inline, protected]`

An operator() has to be provided.

Its signature depends on the particular function-object one considers.

Definition at line 219 of file `function.hh`.

10.181 `mln::Function< void >` Struct Template Reference

[Function](#) category flag type.

```
#include <function.hh>
```

10.181.1 Detailed Description

```
template<> struct mln::Function< void >
```

[Function](#) category flag type.

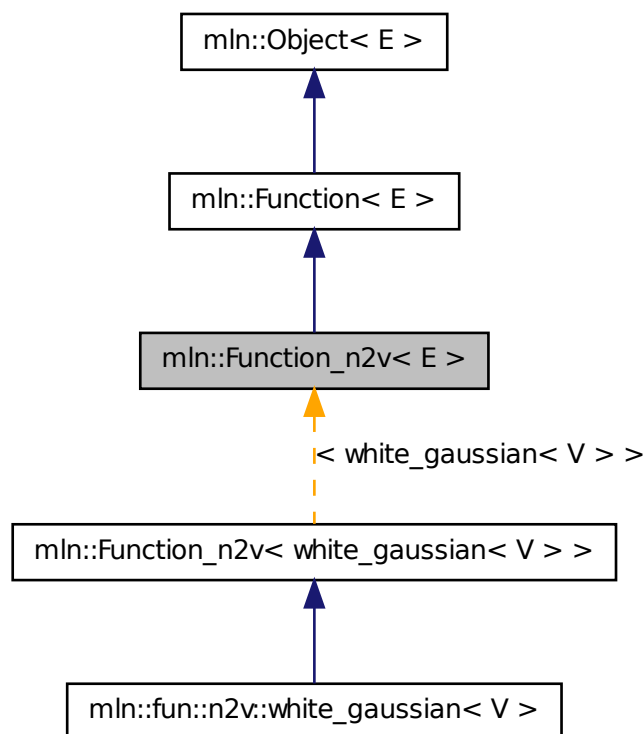
Definition at line 51 of file `function.hh`.

10.182 mln::Function_n2v< E > Struct Template Reference

Base class for implementation of function-objects from Nil to value.

```
#include <function.hh>
```

Inheritance diagram for mln::Function_n2v< E >:



10.182.1 Detailed Description

```
template<typename E> struct mln::Function_n2v< E >
```

Base class for implementation of function-objects from Nil to value. The parameter *E* is the exact type.

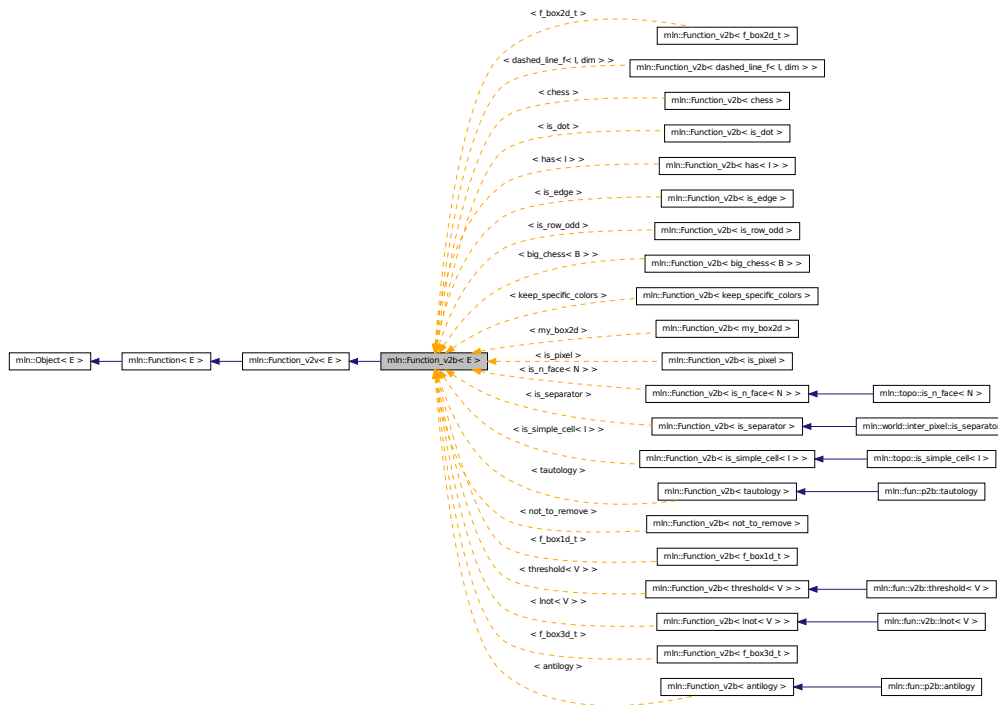
Definition at line 99 of file `function.hh`.

10.183 mln::Function_v2b< E > Struct Template Reference

Base class for implementation of function-objects from a value to a Boolean.

```
#include <function.hh>
```


Inheritance diagram for mln::Function_v2b< E >:



10.183.1 Detailed Description

`template<typename E> struct mln::Function_v2b< E >`

Base class for implementation of function-objects from a value to a Boolean. The parameter *E* is the exact type.

Definition at line 150 of file function.hh.

10.184 mln::Function_v2v< E > Struct Template Reference

Base class for implementation of function-objects from value to value.

```
#include <function.hh>
```

Inherits [mln::Function< E >](#).

Inherited by [mln::fun::C< R\(*\) \(A\) >](#), [mln::fun::v2v::dec< T >](#), [mln::fun::v2v::id< T >](#), [mln::fun::v2v::inc< T >](#), [mln::fun::x2v::bilinear< I >](#), [mln::fun::x2v::trilinear< I >](#), [mln::fun::x2x::internal::helper_composed_< T2, T1, E, false >](#), [mln::fun::x2x::internal::helper_composed_< T2, T1, E, true >](#), [mln::fun::x2x::linear< I >](#), [mln::fun::x2x::neighbor< I >](#), and [mln::Function_v2b< E >](#) [virtual].

10.184.1 Detailed Description

```
template<typename E> struct mln::Function_v2v< E >
```

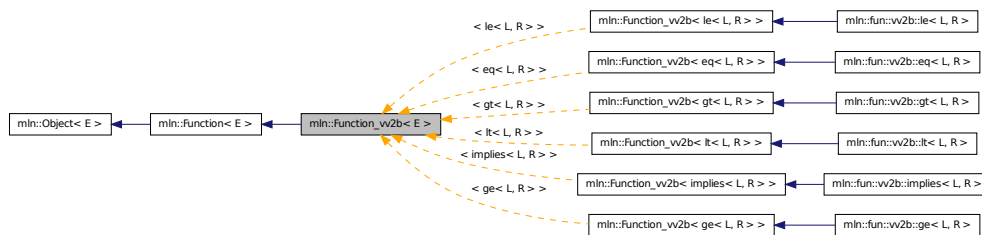
Base class for implementation of function-objects from value to value. The parameter *E* is the exact type. Definition at line 124 of file function.hh.

10.185 mln::Function_vv2b< E > Struct Template Reference

Base class for implementation of function-objects from a couple of values to a Boolean.

```
#include <function.hh>
```

Inheritance diagram for mln::Function_vv2b< E >:



10.185.1 Detailed Description

```
template<typename E> struct mln::Function_vv2b< E >
```

Base class for implementation of function-objects from a couple of values to a Boolean. The parameter *E* is the exact type.

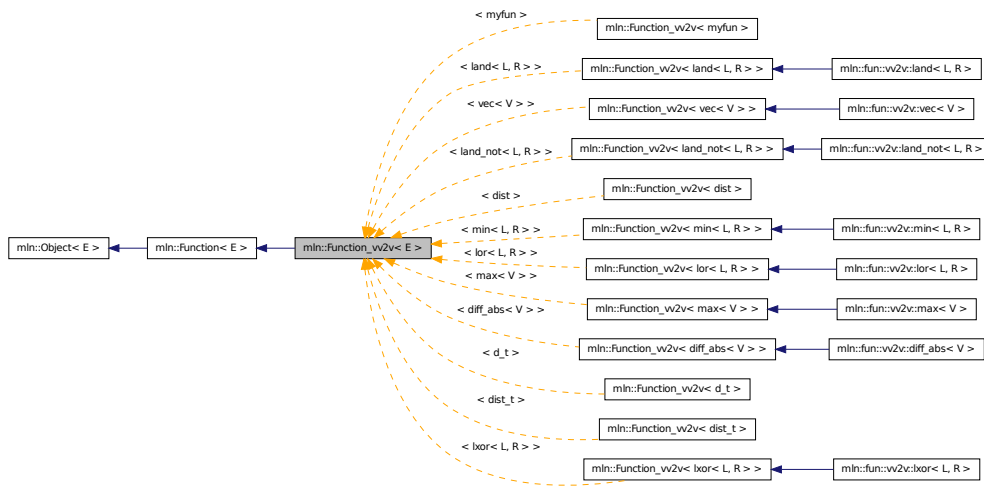
Definition at line 202 of file function.hh.

10.186 mln::Function_vv2v< E > Struct Template Reference

Base class for implementation of function-objects from a couple of values to a value.

```
#include <function.hh>
```

Inheritance diagram for mln::Function_vv2v< E >:



10.186.1 Detailed Description

template<typename E> struct mln::Function_vv2v< E >

Base class for implementation of function-objects from a couple of values to a value. The parameter *E* is the exact type.

Definition at line 177 of file function.hh.

10.187 mln::fwd_pixter1d< I > Class Template Reference

Forward pixel iterator on a 1-D image with border.

```
#include <pixter1d.hh>
```

Inherits forward_pixel_iterator_base_< I, fwd_pixter1d< I > >.

Public Types

- typedef I [image](#)
Image type.

Public Member Functions

- [fwd_pixter1d](#) (I &[image](#))
Constructor.
- void [next](#) ()

Go to the next element.

10.187.1 Detailed Description

`template<typename I> class mln::fwd_pixter1d< I >`

Forward pixel iterator on a 1-D image with border.

Definition at line 45 of file `pixter1d.hh`.

10.187.2 Member Typedef Documentation

10.187.2.1 `template<typename I > typedef I mln::fwd_pixter1d< I >::image`

[Image](#) type.

Definition at line 52 of file `pixter1d.hh`.

10.187.3 Constructor & Destructor Documentation

10.187.3.1 `template<typename I > mln::fwd_pixter1d< I >::fwd_pixter1d (I & image)`
[`inline`]

Constructor.

Parameters

[`in`] *image* The image this pixel iterator is bound to.

Definition at line 96 of file `pixter1d.hh`.

10.187.4 Member Function Documentation

10.187.4.1 `void mln::Iterator< fwd_pixter1d< I > >::next ()` [`inherited`]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.188 mln::fwd_pixter2d< I > Class Template Reference

Forward pixel iterator on a 2-D image with border.

```
#include <pixter2d.hh>
```

Inherits `forward_pixel_iterator_base_< I, fwd_pixter2d< I > >`.

Public Types

- typedef `I` `image`
Image type.

Public Member Functions

- `fwd_pixter2d` (`I &image`)
Constructor.
- void `next` ()
Go to the next element.

10.188.1 Detailed Description

```
template<typename I> class mln::fwd_pixter2d< I >
```

Forward pixel iterator on a 2-D image with border.

Definition at line 47 of file `pixter2d.hh`.

10.188.2 Member Typedef Documentation

10.188.2.1 `template<typename I > typedef I mln::fwd_pixter2d< I >::image`

`Image` type.

Definition at line 54 of file `pixter2d.hh`.

10.188.3 Constructor & Destructor Documentation

10.188.3.1 `template<typename I > mln::fwd_pixter2d< I >::fwd_pixter2d (I & image)`
`[inline]`

Constructor.

Parameters

`[in]` `image` The image this pixel iterator is bound to.

Definition at line 130 of file `pixter2d.hh`.

10.188.4 Member Function Documentation

10.188.4.1 void mln::Iterator< fwd_pixter2d< I > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.189 mln::fwd_pixter3d< I > Class Template Reference

Forward pixel iterator on a 3-D image with border.

```
#include <pixter3d.hh>
```

Inherits forward_pixel_iterator_base_< I, fwd_pixter3d< I > >.

Public Types

- typedef I [image](#)
Image type.

Public Member Functions

- [fwd_pixter3d](#) (I &image)
Constructor.
- void [next](#) ()
Go to the next element.

10.189.1 Detailed Description

```
template<typename I> class mln::fwd_pixter3d< I >
```

Forward pixel iterator on a 3-D image with border.

Definition at line 48 of file pixter3d.hh.

10.189.2 Member Typedef Documentation

10.189.2.1 template<typename I > typedef I mln::fwd_pixter3d< I >::image

[Image](#) type.

Definition at line 55 of file pixter3d.hh.

10.189.3 Constructor & Destructor Documentation

10.189.3.1 `template<typename I > mln::fwd_pixter3d< I >::fwd_pixter3d (I & image)`
[inline]

Constructor.

Parameters

[in] *image* The image this pixel iterator is bound to.

Definition at line 154 of file pixter3d.hh.

10.189.4 Member Function Documentation

10.189.4.1 `void mln::Iterator< fwd_pixter3d< I > >::next ()` [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

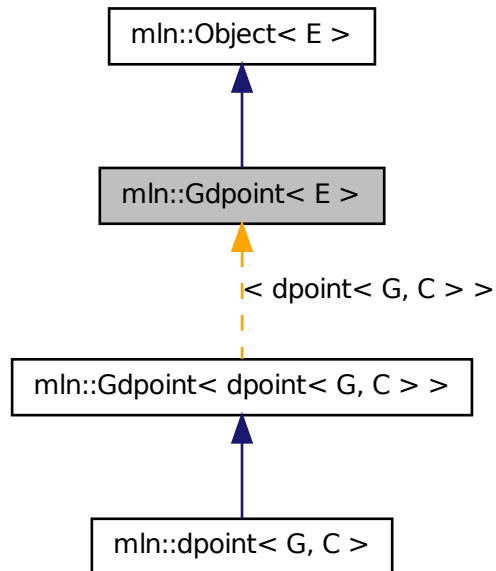
The iterator is valid.

10.190 mln::Gdpoint< E > Struct Template Reference

FIXME: Doc!

```
#include <gdpoint.hh>
```

Inheritance diagram for `mln::Gdpoint< E >`:



10.190.1 Detailed Description

```
template<typename E> struct mln::Gdpoint< E >
```

FIXME: Doc!

Definition at line 95 of file `gdpoint.hh`.

10.191 `mln::Gdpoint< void >` Struct Template Reference

Delta point site category flag type.

```
#include <gdpoint.hh>
```

10.191.1 Detailed Description

```
template<> struct mln::Gdpoint< void >
```

Delta point site category flag type.

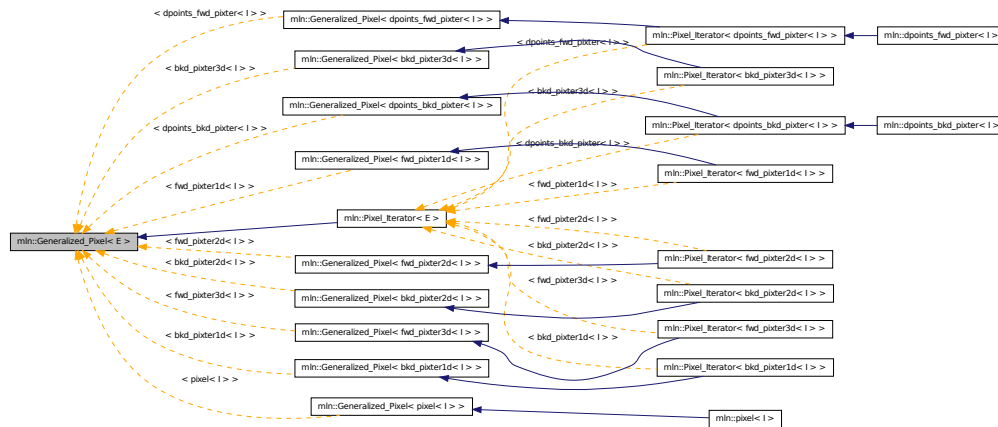
Definition at line 87 of file `gdpoint.hh`.

10.192 mln::Generalized_Pixel< E > Struct Template Reference

Base class for implementation classes that are pixels or that have the behavior of pixels.

```
#include <generalized_pixel.hh>
```

Inheritance diagram for mln::Generalized_Pixel< E >:



10.192.1 Detailed Description

```
template<typename E> struct mln::Generalized_Pixel< E >
```

Base class for implementation classes that are pixels or that have the behavior of pixels.

Warning

This class does *not* derive from [mln::Object](#); it is for use as a parallel hierarchy.

See also

[mln::doc::Generalized_Pixel](#) for a complete documentation of this class contents.

Definition at line 53 of file `generalized_pixel.hh`.

10.193 mln::geom::complex_geometry< D, P > Class Template Reference

A functor returning the sites of the faces of a complex where the locations of each 0-face is stored.

```
#include <complex_geometry.hh>
```

Public Member Functions

- unsigned [add_location](#) (const P &p)

Populate the set of locations.

- `complex_geometry ()`

Build a complex geometry object.

- `site operator() (const mln::topo::face< D > &f) const`

Retrieve the site associated to f .

10.193.1 Detailed Description

`template<unsigned D, typename P> class mln::geom::complex_geometry< D, P >`

A functor returning the sites of the faces of a complex where the locations of each 0-face is stored. Faces of higher dimensions are computed.

Template Parameters

D The dimension of the complex.

P The type of the location of a 0-face.

Locations of 0-face are usually points (hence the P above), but can possibly be any (default-constructible) values.

The functor returns a `std::vector` of locations: 0-faces are singletons, 1-faces are (usually) pairs, faces of higher dimensions are arrays of locations.

Note that for consistency reasons w.r.t. the return type of `operator()`, returned sites are always *arrays* of locations attached to 0-faces; hence the returned singletons (of locations) for 0-faces.

Definition at line 88 of file `geom/complex_geometry.hh`.

10.193.2 Constructor & Destructor Documentation

10.193.2.1 `template<unsigned D, typename P > mln::geom::complex_geometry< D, P >::complex_geometry () [inline]`

Build a complex geometry object.

Definition at line 132 of file `geom/complex_geometry.hh`.

10.193.3 Member Function Documentation

10.193.3.1 `template<unsigned D, typename P > unsigned mln::geom::complex_geometry< D, P >::add_location (const P & p) [inline]`

Populate the set of locations.

Append a new location p . Return the index of the newly created location (which should semantically match the id of the corresponding 0-face in the complex).

Definition at line 140 of file `geom/complex_geometry.hh`.

10.193.3.2 `template<unsigned D, typename P > util::multi_site< P > mln::geom::complex_geometry< D, P >::operator() (const mln::topo::face< D > & f) const`
[inline]

Retrieve the site associated to *f*.

Definition at line 151 of file geom/complex_geometry.hh.

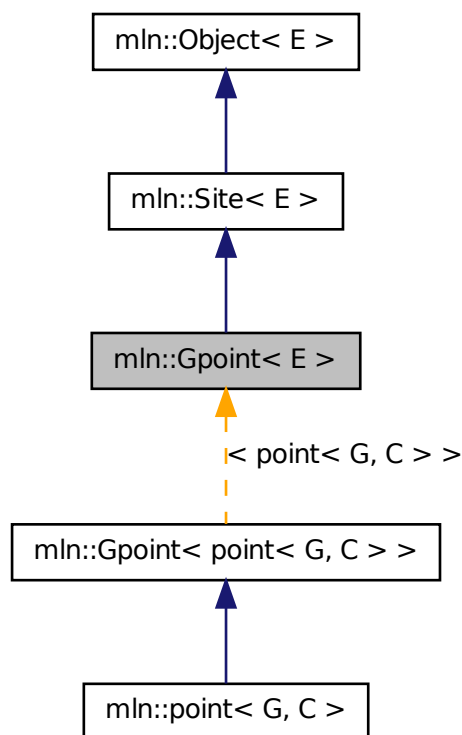
References `mln::topo::face< D >::face_id()`, and `mln::topo::face< D >::n()`.

10.194 mln::Gpoint< E > Struct Template Reference

Base class for implementation of point classes.

```
#include <gpoint.hh>
```

Inheritance diagram for `mln::Gpoint< E >`:



Related Functions

(Note that these are not member functions.)

- `template<typename P, typename D >`
`P operator+ (const Gpoint< P > &p, const Gdpoint< D > &dp)`
Add a delta-point rhs to a grid point lhs.
- `template<typename P, typename D >`
`P & operator+= (Gpoint< P > &p, const Gdpoint< D > &dp)`
Shift a point by a delta-point dp.
- `template<typename L, typename R >`
`L::delta operator- (const Gpoint< L > &lhs, const Gpoint< R > &rhs)`
Difference between a couple of grid point lhs and rhs.
- `template<typename P, typename D >`
`P & operator-= (Gpoint< P > &p, const Gdpoint< D > &dp)`
Shift a point by the negate of a delta-point dp.
- `template<typename P, typename D >`
`P operator/ (const Gpoint< P > &p, const value::scalar_< D > &dp)`
Divide a point by a scalar s.
- `template<typename P >`
`std::ostream & operator<< (std::ostream &ostr, const Gpoint< P > &p)`
Print a grid point p into the output stream ostr.
- `template<typename L, typename R >`
`bool operator== (const Gpoint< L > &lhs, const Gpoint< R > &rhs)`
Equality comparison between a couple of grid point lhs and rhs.

10.194.1 Detailed Description

`template<typename E> struct mln::Gpoint< E >`

Base class for implementation of point classes. A point is an element of a space.

For instance, `mln::point2d` is the type of elements defined on the discrete square grid of the 2D plane.

Definition at line 115 of file `gpoint.hh`.

10.194.2 Friends And Related Function Documentation

10.194.2.1 `template<typename P, typename D > P operator+ (const Gpoint< P > & p, const Gdpoint< D > & dp)` [**related**]

Add a delta-point `rhs` to a grid point `lhs`.

Parameters

- [in] `p` A grid point.
- [in] `dp` A delta-point.

The type of `dp` has to compatible with the type of `p`.

Returns

A point (temporary object).

See also

[mln::Gdpoint](#)

Definition at line 385 of file gpoint.hh.

10.194.2.2 `template<typename P , typename D > P & operator+= (Gpoint< P > & p, const Gdpoint< D > & dp) [related]`

Shift a point by a delta-point `dp`.

Parameters

[in, out] *p* The targeted point.

[in] *dp* A delta-point.

Returns

A reference to the point `p` once translated by `dp`.

Precondition

The type of `dp` has to be compatible with the type of `p`.

Definition at line 428 of file gpoint.hh.

10.194.2.3 `template<typename L , typename R > L::delta operator- (const Gpoint< L > & lhs, const Gpoint< R > & rhs) [related]`

Difference between a couple of grid point `lhs` and `rhs`.

Parameters

[in] *lhs* A first grid point.

[in] *rhs* A second grid point.

Warning

There is no type promotion in Milena so the client has to make sure that both points are defined with the same type of coordinates.

Precondition

Both `lhs` and `rhs` have to be defined on the same topology and with the same type of coordinates; otherwise this test does not compile.

Postcondition

The result, `dp`, is such as `lhs == rhs + dp`.

Returns

A delta point (temporary object).

See also

[mln::Gdpoint](#)

Definition at line 374 of file gpoint.hh.

10.194.2.4 `template<typename P , typename D > P & operator-= (Gpoint< P > & p, const Gdpoint< D > & dp) [related]`

Shift a point by the negate of a delta-point `dp`.

Parameters

[in, out] *p* The targeted point.
 [in] *dp* A delta-point.

Returns

A reference to the point `p` once translated by `- dp`.

Precondition

The type of `dp` has to be compatible with the type of `p`.

Definition at line 436 of file gpoint.hh.

10.194.2.5 `template<typename P , typename D > P operator/ (const Gpoint< P > & p, const value::scalar_< D > & dp) [related]`

Divise a point by a scalar `s`.

Parameters

[in, out] *p* The targeted point.
 [in] *dp* A scalar.

Returns

A reference to the point `p` once divided by `s`.

10.194.2.6 `template<typename P > std::ostream & operator<< (std::ostream & ostr, const Gpoint< P > & p) [related]`

Print a grid point `p` into the output stream `ostr`.

Parameters

[in, out] *ostr* An output stream.
 [in] *p* A grid point.

Returns

The modified output stream `ostr`.

Definition at line 417 of file gpoint.hh.

References `mln::debug::format()`.

10.194.2.7 `template<typename L , typename R > bool operator==(const Gpoint< L > & lhs, const Gpoint< R > & rhs)` [related]

Equality comparison between a couple of grid point `lhs` and `rhs`.

Parameters

- [in] *lhs* A first grid point.
- [in] *rhs* A second grid point.

Precondition

Both `lhs` and `rhs` have to be defined on the same topology; otherwise this test does not compile.

Returns

True if both grid points have the same coordinates, otherwise false.

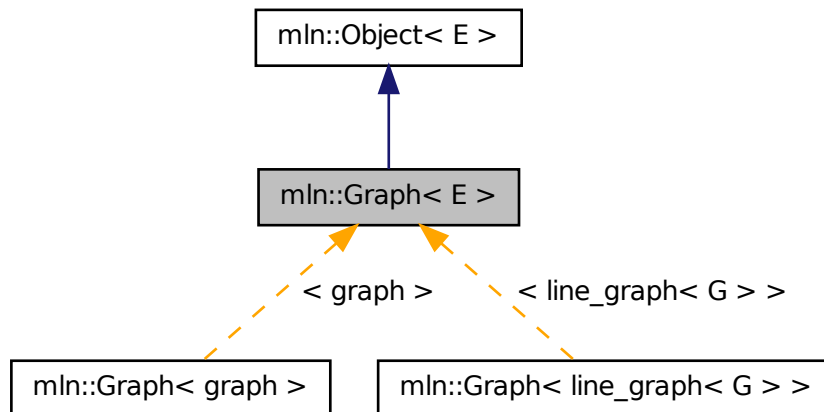
Definition at line 365 of file `gpoint.hh`.

10.195 mln::Graph< E > Struct Template Reference

Base class for implementation of graph classes.

```
#include <graph.hh>
```

Inheritance diagram for `mln::Graph< E >`:



10.195.1 Detailed Description

```
template<typename E> struct mln::Graph< E >
```

Base class for implementation of graph classes.

See also

`mln::doc::Graph` for a complete documentation of this class contents.

Definition at line 57 of file `mln/core/concept/graph.hh`.

10.196 `mln::graph::attribute::card_t` Struct Reference

Compute the cardinality of every component in a graph.

```
#include <card.hh>
```

Public Types

- typedef `util::array< unsigned > result`

Type of the computed value.

10.196.1 Detailed Description

Compute the cardinality of every component in a graph.

Returns

An array with the cardinality for each component. Components are labeled from 0.

Definition at line 61 of file `graph/attribute/card.hh`.

10.196.2 Member Typedef Documentation

10.196.2.1 `typedef util::array<unsigned> mln::graph::attribute::card_t::result`

Type of the computed value.

Definition at line 64 of file `graph/attribute/card.hh`.

10.197 `mln::graph::attribute::representative_t` Struct Reference

Compute the representative vertex of every component in a graph.

```
#include <representative.hh>
```

Public Types

- typedef `util::array< unsigned > result`

Type of the computed value.

10.197.1 Detailed Description

Compute the representative vertex of every component in a graph.

Returns

An array with the representative for each component. Components are labeled from 0.

Definition at line 63 of file representative.hh.

10.197.2 Member Typedef Documentation

10.197.2.1 typedef util::array<unsigned> mln::graph::attribute::representative_t::result

Type of the computed value.

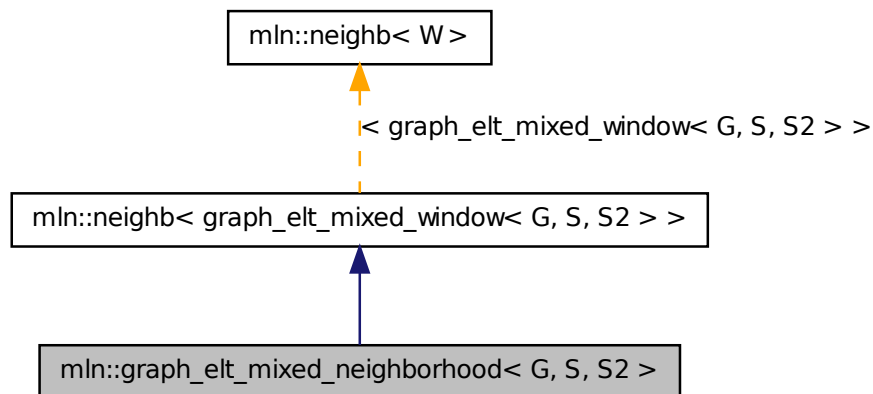
Definition at line 66 of file representative.hh.

10.198 mln::graph_elt_mixed_neighborhood< G, S, S2 > Struct Template Reference

Elementary neighborhood on graph class.

```
#include <graph_elt_mixed_neighborhood.hh>
```

Inheritance diagram for mln::graph_elt_mixed_neighborhood< G, S, S2 >:



Public Types

- typedef `neighb_bkd_niter< graph_elt_mixed_window< G, S, S2 > > bkd_niter`
Backward site iterator associated type.

- typedef `neighb_fwd_niter`< `graph_elt_mixed_window`< `G`, `S`, `S2` > > `fwd_niter`
Forward site iterator associated type.
- typedef `fwd_niter` `niter`
Site iterator associated type.

10.198.1 Detailed Description

```
template<typename G, typename S, typename S2> struct mln::graph_elt_mixed_neighborhood<
G, S, S2 >
```

Elementary neighborhood on graph class.

Template Parameters

- `G` is a graph type.
- `S` is a site set type.
- `S2` is the site set type of the neighbors.

Definition at line 48 of file `graph_elt_mixed_neighborhood.hh`.

10.198.2 Member Typedef Documentation

10.198.2.1 typedef `neighb_bkd_niter`<`graph_elt_mixed_window`< `G`, `S`, `S2` > > `mln::neighb`<`graph_elt_mixed_window`< `G`, `S`, `S2` > >::`bkd_niter` [`inherited`]

Backward site iterator associated type.

Definition at line 87 of file `mln/core/neighb.hh`.

10.198.2.2 typedef `neighb_fwd_niter`<`graph_elt_mixed_window`< `G`, `S`, `S2` > > `mln::neighb`<`graph_elt_mixed_window`< `G`, `S`, `S2` > >::`fwd_niter` [`inherited`]

Forward site iterator associated type.

Definition at line 84 of file `mln/core/neighb.hh`.

10.198.2.3 typedef `fwd_niter` `mln::neighb`< `graph_elt_mixed_window`< `G`, `S`, `S2` > >::`niter` [`inherited`]

Site iterator associated type.

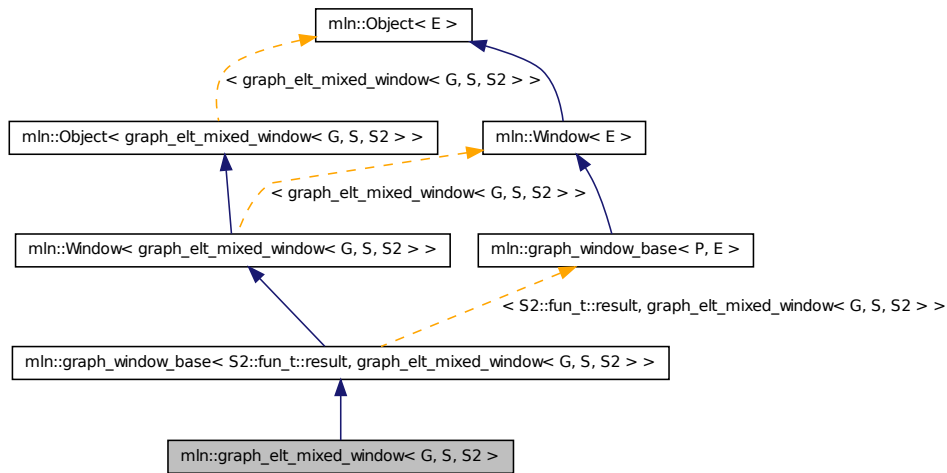
Definition at line 90 of file `mln/core/neighb.hh`.

10.199 mln::graph_elt_mixed_window< G, S, S2 > Class Template Reference

Elementary window on graph class.

```
#include <graph_elt_mixed_window.hh>
```

Inheritance diagram for mln::graph_elt_mixed_window< G, S, S2 >:



Public Types

- typedef `super_::target` [target](#)
Associated types.
- typedef `target::psite` [psite](#)
The type of psite corresponding to the window.
- typedef `S::psite` [center_t](#)
Type of the window center element.
- typedef `target::graph_element` [graph_element](#)
Type of the graph element pointed by this iterator.
- typedef `graph_window_piter< target, self_, nbh_fwd_iter_ >` [fwd_qiter](#)
Site_Iterator type to browse the psites of the window w.r.t.
- typedef `graph_window_piter< target, self_, nbh_bkd_iter_ >` [bkd_qiter](#)
Site_Iterator type to browse the psites of the window w.r.t.
- typedef `fwd_qiter` [qiter](#)
The default qiter type.
- typedef `S2::fun_t::result` [site](#)
Associated types.

Public Member Functions

- bool [is_valid](#) () const
Return true by default.
- bool [is_empty](#) () const
Interface of the concept Window.
- bool [is_centered](#) () const
Is the window centered?
- bool [is_symmetric](#) () const
Is the window symmetric?
- unsigned [delta](#) () const
Return the maximum coordinate gap between the window center and a window point.
- [self_ & sym](#) ()
Apply a central symmetry to the target window.

10.199.1 Detailed Description

```
template<typename G, typename S, typename S2> class mln::graph_elt_mixed_window< G, S, S2
>
```

Elementary window on graph class. *G* is the graph type. *S* is an image site set from where the center is extracted. *S2* is an image site set from where the neighbors are extracted.

Definition at line 109 of file `graph_elt_mixed_window.hh`.

10.199.2 Member Typedef Documentation

10.199.2.1 `template<typename G , typename S , typename S2 > typedef graph_window_piter<target,self,nbh_bkd_iter_> mln::graph_elt_mixed_window< G, S, S2 >::bkd_qiter`

[Site_Iterator](#) type to browse the psites of the window w.r.t.

the reverse ordering of vertices.

Definition at line 139 of file `graph_elt_mixed_window.hh`.

10.199.2.2 `template<typename G , typename S , typename S2 > typedef S ::psite mln::graph_elt_mixed_window< G, S, S2 >::center_t`

Type of the window center element.

Definition at line 128 of file `graph_elt_mixed_window.hh`.

10.199.2.3 `template<typename G , typename S , typename S2 > typedef graph_window_piter<target,self,nbh_fwd_iter_> mln::graph_elt_mixed_window< G, S, S2 >::fwd_qiter`

[Site_Iterator](#) type to browse the psites of the window w.r.t. the ordering of vertices.

Definition at line 135 of file graph_elt_mixed_window.hh.

10.199.2.4 `template<typename G , typename S , typename S2 > typedef target ::graph_element mln::graph_elt_mixed_window< G, S, S2 >::graph_element`

Type of the graph element pointed by this iterator.

Definition at line 131 of file graph_elt_mixed_window.hh.

10.199.2.5 `template<typename G , typename S , typename S2 > typedef target ::psite mln::graph_elt_mixed_window< G, S, S2 >::psite`

The type of psite corresponding to the window.

Definition at line 125 of file graph_elt_mixed_window.hh.

10.199.2.6 `template<typename G , typename S , typename S2 > typedef fwd_qiter mln::graph_elt_mixed_window< G, S, S2 >::qiter`

The default qiter type.

Definition at line 142 of file graph_elt_mixed_window.hh.

10.199.2.7 `typedef S2::fun_t::result mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::site [inherited]`

Associated types.

The type of site corresponding to the window.

Definition at line 48 of file graph_window_base.hh.

10.199.2.8 `template<typename G , typename S , typename S2 > typedef super_::target mln::graph_elt_mixed_window< G, S, S2 >::target`

Associated types.

Definition at line 123 of file graph_elt_mixed_window.hh.

10.199.3 Member Function Documentation

10.199.3.1 `unsigned mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::delta () const [inherited]`

Return the maximum coordinate gap between the window center and a window point.

10.199.3.2 `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_centered () const [inherited]`

Is the window centered?

10.199.3.3 `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_empty () const [inherited]`

Interface of the concept Window.

Is the window is empty?

10.199.3.4 `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_symmetric () const [inherited]`

Is the window symmetric?

10.199.3.5 `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_valid () const [inherited]`

Return true by default.

10.199.3.6 `self_& mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::sym () [inherited]`

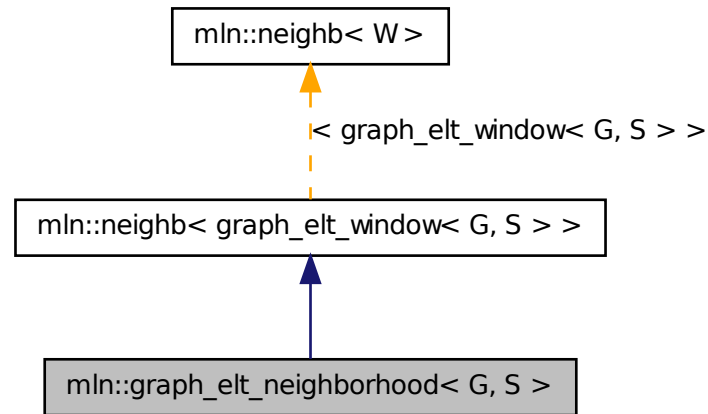
Apply a central symmetry to the target window.

10.200 mln::graph_elt_neighborhood< G, S > Struct Template Reference

Elementary neighborhood on graph class.

```
#include <graph_elt_neighborhood.hh>
```

Inheritance diagram for mln::graph_elt_neighborhood< G, S >:



Public Types

- typedef `neighb_bkd_niter< graph_elt_window< G, S > > bkd_niter`
Backward site iterator associated type.
- typedef `neighb_fwd_niter< graph_elt_window< G, S > > fwd_niter`
Forward site iterator associated type.
- typedef `fwd_niter niter`
Site iterator associated type.

10.200.1 Detailed Description

```
template<typename G, typename S> struct mln::graph_elt_neighborhood< G, S >
```

Elementary neighborhood on graph class.

Template Parameters

G is a graph type.

S is a site set type.

Definition at line 47 of file `graph_elt_neighborhood.hh`.

10.200.2 Member Typedef Documentation

10.200.2.1 `typedef neighb_bkd_niter<graph_elt_window< G, S > > mln::neighb<graph_elt_window< G, S > >::bkd_niter` `[inherited]`

Backward site iterator associated type.

Definition at line 87 of file mln/core/neighb.hh.

10.200.2.2 `typedef neighb_fwd_niter<graph_elt_window< G, S > > mln::neighb<graph_elt_window< G, S > >::fwd_niter` `[inherited]`

Forward site iterator associated type.

Definition at line 84 of file mln/core/neighb.hh.

10.200.2.3 `typedef fwd_niter mln::neighb< graph_elt_window< G, S > >::niter` `[inherited]`

Site iterator associated type.

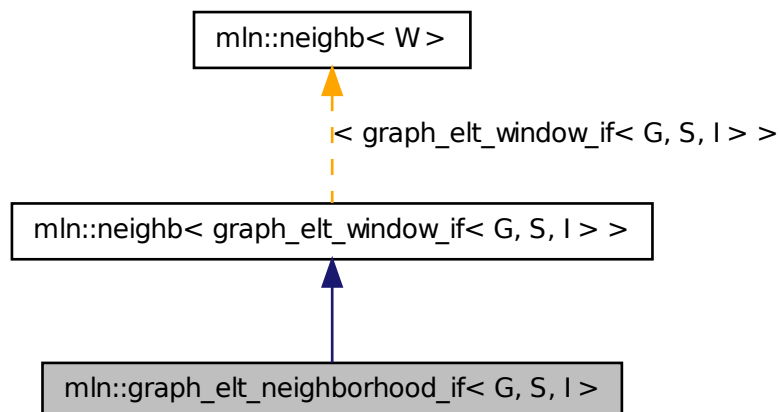
Definition at line 90 of file mln/core/neighb.hh.

10.201 mln::graph_elt_neighborhood_if< G, S, I > Struct Template Reference

Elementary neighborhood_if on graph class.

```
#include <graph_elt_neighborhood_if.hh>
```

Inheritance diagram for mln::graph_elt_neighborhood_if< G, S, I >:



Public Types

- typedef neighb_bkd_niter< graph_elt_window_if< G, S, I > > bkd_niter
Backward site iterator associated type.
- typedef neighb_fwd_niter< graph_elt_window_if< G, S, I > > fwd_niter
Forward site iterator associated type.
- typedef fwd_niter niter
Site iterator associated type.

Public Member Functions

- graph_elt_neighborhood_if ()
Constructors @{ Construct an invalid neighborhood.
- graph_elt_neighborhood_if (const Image< I > &mask)
- const I & mask () const
@/

10.201.1 Detailed Description

```
template<typename G, typename S, typename I> struct mln::graph_elt_neighborhood_if< G, S, I >
```

Elementary neighborhood_if on graph class.

Definition at line 43 of file graph_elt_neighborhood_if.hh.

10.201.2 Member Typedef Documentation

10.201.2.1 typedef neighb_bkd_niter<graph_elt_window_if< G, S, I > > mln::neighb<graph_elt_window_if< G, S, I > >::bkd_niter **[inherited]**

Backward site iterator associated type.

Definition at line 87 of file mln/core/neighb.hh.

10.201.2.2 typedef neighb_fwd_niter<graph_elt_window_if< G, S, I > > mln::neighb<graph_elt_window_if< G, S, I > >::fwd_niter **[inherited]**

Forward site iterator associated type.

Definition at line 84 of file mln/core/neighb.hh.

10.201.2.3 `typedef fwd_niter mln::neighb< graph_elt_window_if< G, S, I > >::niter`
`[inherited]`

Site iterator associated type.

Definition at line 90 of file mln/core/neighb.hh.

10.201.3 Constructor & Destructor Documentation

10.201.3.1 `template<typename G , typename S , typename I > mln::graph_elt_neighborhood_if<`
`G, S, I >::graph_elt_neighborhood_if() [inline]`

Constructors @ { Construct an invalid neighborhood.

Definition at line 67 of file graph_elt_neighborhood_if.hh.

10.201.3.2 `template<typename G , typename S , typename I > mln::graph_elt_neighborhood_if<`
`G, S, I >::graph_elt_neighborhood_if(const Image< I > & mask) [inline]`

Parameters

[in] *mask* A graph image of Boolean.

Definition at line 74 of file graph_elt_neighborhood_if.hh.

10.201.4 Member Function Documentation

10.201.4.1 `template<typename G , typename S , typename I > const I &`
`mln::graph_elt_neighborhood_if< G, S, I >::mask() const [inline]`

@ }

Return the graph image used as mask.

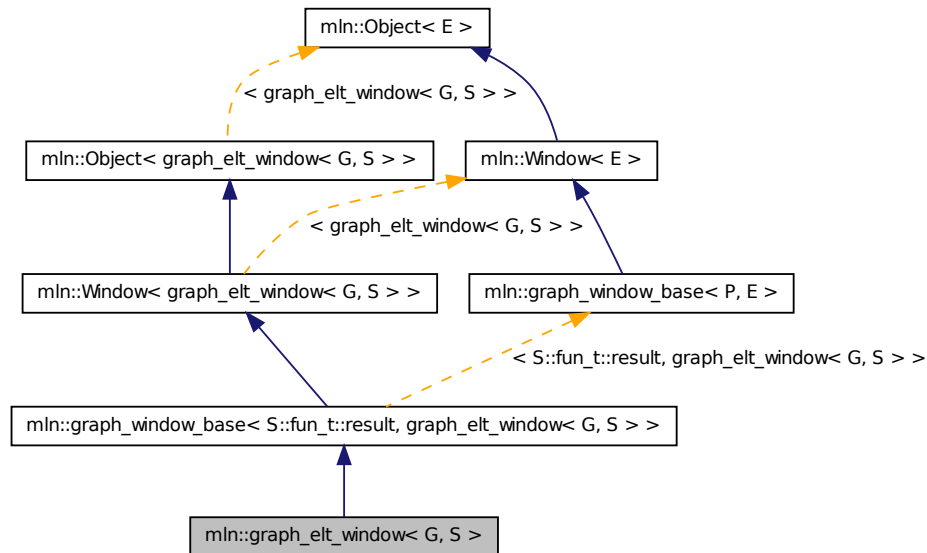
Definition at line 83 of file graph_elt_neighborhood_if.hh.

10.202 mln::graph_elt_window< G, S > Class Template Reference

Elementary window on graph class.

`#include <graph_elt_window.hh>`

Inheritance diagram for mln::graph_elt_window< G, S >:



Public Types

- typedef `S target`
Associated types.
- typedef `S::psite psite`
The type of psite corresponding to the window.
- typedef `S::psite center_t`
Type of the window center element.
- typedef `S::graph_element graph_element`
Type of the graph element pointed by this iterator.
- typedef `graph_window_piter< S, self_, nbh_fwd_iter_ > fwd_qiter`
Site_Iterator type to browse the psites of the window w.r.t.
- typedef `graph_window_piter< S, self_, nbh_bkd_iter_ > bkd_qiter`
Site_Iterator type to browse the psites of the window w.r.t.
- typedef `fwd_qiter qiter`
The default qiter type.
- typedef `S::fun_t::result site`
Associated types.

Public Member Functions

- bool [is_valid](#) () const
Return true by default.
- bool [is_empty](#) () const
Interface of the concept Window.
- bool [is_centered](#) () const
Is the window centered?
- bool [is_symmetric](#) () const
Is the window symmetric?
- unsigned [delta](#) () const
Return the maximum coordinate gap between the window center and a window point.
- [self_ & sym](#) ()
Apply a central symmetry to the target window.

10.202.1 Detailed Description

template<typename G, typename S> class mln::graph_elt_window< G, S >

Elementary window on graph class. *G* is the graph type. *S* is an image site set from where the center is extracted. *S2* is an image site set from where the neighbors are extracted.

Definition at line 111 of file graph_elt_window.hh.

10.202.2 Member Typedef Documentation

10.202.2.1 template<typename G , typename S > typedef graph_window_piter<S,self_nbh_bkd_iter_> mln::graph_elt_window< G, S >::bkd_qiter

[Site_Iterator](#) type to browse the psites of the window w.r.t.

the reverse ordering of vertices.

Definition at line 142 of file graph_elt_window.hh.

10.202.2.2 template<typename G , typename S > typedef S ::psite mln::graph_elt_window< G, S >::center_t

Type of the window center element.

Definition at line 131 of file graph_elt_window.hh.

10.202.2.3 `template<typename G , typename S > typedef graph_window_riter<S,self_,nbh_fwd_iter_> mln::graph_elt_window< G, S >::fwd_qiter`

[Site_Iterator](#) type to browse the psites of the window w.r.t. the ordering of vertices.

Definition at line 138 of file graph_elt_window.hh.

10.202.2.4 `template<typename G , typename S > typedef S ::graph_element mln::graph_elt_window< G, S >::graph_element`

Type of the graph element pointed by this iterator.

Definition at line 134 of file graph_elt_window.hh.

10.202.2.5 `template<typename G , typename S > typedef S ::psite mln::graph_elt_window< G, S >::psite`

The type of psite corresponding to the window.

Definition at line 128 of file graph_elt_window.hh.

10.202.2.6 `template<typename G , typename S > typedef fwd_qiter mln::graph_elt_window< G, S >::qiter`

The default qiter type.

Definition at line 145 of file graph_elt_window.hh.

10.202.2.7 `typedef S::fun_t::result mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::site [inherited]`

Associated types.

The type of site corresponding to the window.

Definition at line 48 of file graph_window_base.hh.

10.202.2.8 `template<typename G , typename S > typedef S mln::graph_elt_window< G, S >::target`

Associated types.

Definition at line 125 of file graph_elt_window.hh.

10.202.3 Member Function Documentation

10.202.3.1 `unsigned mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::delta () const [inherited]`

Return the maximum coordinate gap between the window center and a window point.

10.202.3.2 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S >
>::is_centered () const [inherited]`

Is the window centered?

10.202.3.3 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S >
>::is_empty () const [inherited]`

Interface of the concept Window.

Is the window is empty?

10.202.3.4 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S >
>::is_symmetric () const [inherited]`

Is the window symmetric?

10.202.3.5 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S >
>::is_valid () const [inherited]`

Return true by default.

10.202.3.6 `self_& mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S >
>::sym () [inherited]`

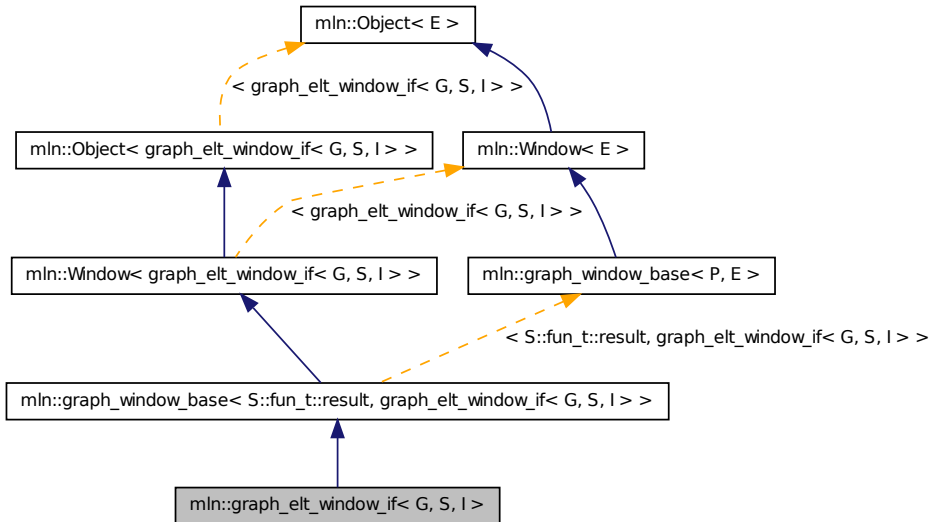
Apply a central symmetry to the target window.

10.203 mln::graph_elt_window_if< G, S, I > Class Template Reference

Custom window on graph class.

```
#include <graph_elt_window_if.hh>
```

Inheritance diagram for mln::graph_elt_window_if< G, S, I >:



Public Types

- typedef I [mask_t](#)

The type of the image used as mask.

- typedef S [target](#)

@}

- typedef target::psite [psite](#)

The type of psite corresponding to the window.

- typedef [graph_window_if_piter](#)< [target](#), [self_](#), [nbh_fwd_iter_](#) > [fwd_qiter](#)

Site_Iterator type to browse the psites of the window w.r.t.

- typedef [graph_window_if_piter](#)< [target](#), [self_](#), [nbh_bkd_iter_](#) > [bkd_qiter](#)

Site_Iterator type to browse the psites of the window w.r.t.

- typedef [fwd_qiter](#) [qiter](#)

The default qiter type.

- typedef S::fun_t::result [site](#)

Associated types.

Public Member Functions

- void [change_mask](#) (const [Image](#)< I > &mask)
Change mask image.
- [graph_elt_window_if](#) ()
Constructor.
- [graph_elt_window_if](#) (const [Image](#)< I > &mask)
- bool [is_valid](#) () const
Return true by default.
- const I & [mask](#) () const
Return the graph image used as mask.
- bool [is_empty](#) () const
Interface of the concept Window.
- bool [is_centered](#) () const
Is the window centered?
- bool [is_symmetric](#) () const
Is the window symmetric?
- unsigned [delta](#) () const
Return the maximum coordinate gap between the window center and a window point.
- [self_ & sym](#) ()
Apply a central symmetry to the target window.

10.203.1 Detailed Description

template<typename G, typename S, typename I> class mln::graph_elt_window_if< G, S, I >

Custom window on graph class. It is defined thanks to a mask.

G is the graph type. S is the image site set. I is the graph image the type used as mask.

Definition at line 105 of file graph_elt_window_if.hh.

10.203.2 Member Typedef Documentation

10.203.2.1 template<typename G , typename S , typename I > typedef graph_window_if_piter<target,self_nbh_bkd_iter_> mln::graph_elt_window_if< G, S, I >::bkd_qiter

[Site_Iterator](#) type to browse the psites of the window w.r.t.

the reverse ordering of vertices.

Definition at line 147 of file graph_elt_window_if.hh.

10.203.2.2 `template<typename G , typename S , typename I > typedef graph_window_if_piter<target,self,nbh_fwd_iter_> mln::graph_elt_window_if< G, S, I >::fwd_qiter`

[Site_Iterator](#) type to browse the psites of the window w.r.t.

the ordering of vertices.

Definition at line 143 of file graph_elt_window_if.hh.

10.203.2.3 `template<typename G , typename S , typename I > typedef I mln::graph_elt_window_if< G, S, I >::mask_t`

The type of the image used as mask.

Definition at line 119 of file graph_elt_window_if.hh.

10.203.2.4 `template<typename G , typename S , typename I > typedef target ::psite mln::graph_elt_window_if< G, S, I >::psite`

The type of psite corresponding to the window.

Definition at line 139 of file graph_elt_window_if.hh.

10.203.2.5 `template<typename G , typename S , typename I > typedef fwd_qiter mln::graph_elt_window_if< G, S, I >::qiter`

The default qiter type.

Definition at line 150 of file graph_elt_window_if.hh.

10.203.2.6 `typedef S::fun_t::result mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::site [inherited]`

Associated types.

The type of site corresponding to the window.

Definition at line 48 of file graph_window_base.hh.

10.203.2.7 `template<typename G , typename S , typename I > typedef S mln::graph_elt_window_if< G, S, I >::target`

@ }

Associated types. The image domain on which this window iterates on.

Definition at line 136 of file graph_elt_window_if.hh.

10.203.3 Constructor & Destructor Documentation

10.203.3.1 `template<typename G , typename S , typename I > mln::graph_elt_window_if< G, S, I >::graph_elt_window_if() [inline]`

Constructor.

@{ Default. Construct an invalid window.

Definition at line 174 of file `graph_elt_window_if.hh`.

10.203.3.2 `template<typename G , typename S , typename I > mln::graph_elt_window_if< G, S, I >::graph_elt_window_if(const Image< I > & mask) [inline]`

Parameters

[in] *mask* A graph image of bool.

See also

[vertex_image](#), [edge_image](#).

Definition at line 181 of file `graph_elt_window_if.hh`.

10.203.4 Member Function Documentation

10.203.4.1 `template<typename G , typename S , typename I > void mln::graph_elt_window_if< G, S, I >::change_mask(const Image< I > & mask) [inline]`

Change mask image.

Definition at line 199 of file `graph_elt_window_if.hh`.

References `mln::graph_elt_window_if< G, S, I >::is_valid()`.

10.203.4.2 `unsigned mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::delta() const [inherited]`

Return the maximum coordinate gap between the window center and a window point.

10.203.4.3 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::is_centered() const [inherited]`

Is the window centered?

10.203.4.4 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::is_empty() const [inherited]`

Interface of the concept Window.

Is the window is empty?

10.203.4.5 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::is_symmetric () const` [*inherited*]

Is the window symmetric?

10.203.4.6 `template<typename G , typename S , typename I > bool mln::graph_elt_window_if< G, S, I >::is_valid () const` [*inline*]

Return true by default.

Reimplemented from `mln::graph_window_base< S::fun_t::result, graph_elt_window_if< G, S, I > >`.

Definition at line 208 of file `graph_elt_window_if.hh`.

Referenced by `mln::graph_elt_window_if< G, S, I >::change_mask()`.

10.203.4.7 `template<typename G , typename S , typename I > const I & mln::graph_elt_window_if< G, S, I >::mask () const` [*inline*]

Return the graph image used as mask.

Definition at line 190 of file `graph_elt_window_if.hh`.

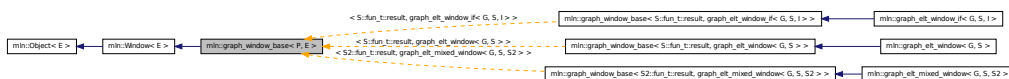
10.203.4.8 `self_& mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::sym ()` [*inherited*]

Apply a central symmetry to the target window.

10.204 mln::graph_window_base< P, E > Class Template Reference

```
#include <graph_window_base.hh>
```

Inheritance diagram for `mln::graph_window_base< P, E >`:



Public Types

- typedef `P` `site`
Associated types.

Public Member Functions

- bool `is_valid` () const
Return true by default.

- bool `is_empty` () const
Interface of the concept [Window](#).
- bool `is_centered` () const
Is the window centered?
- bool `is_symmetric` () const
Is the window symmetric?
- unsigned `delta` () const
Return the maximum coordinate gap between the window center and a window point.
- `self_ & sym` ()
Apply a central symmetry to the target window.

10.204.1 Detailed Description

template<typename P, typename E> class mln::graph_window_base< P, E >

Template Parameters

P [Site](#) type.

Definition at line 40 of file `graph_window_base.hh`.

10.204.2 Member Typedef Documentation

10.204.2.1 template<typename P, typename E> typedef P mln::graph_window_base< P, E >::site

Associated types.

The type of site corresponding to the window.

Definition at line 48 of file `graph_window_base.hh`.

10.204.3 Member Function Documentation

10.204.3.1 template<typename P , typename E > unsigned mln::graph_window_base< P, E >::delta () const [inline]

Return the maximum coordinate gap between the window center and a window point.

Definition at line 128 of file `graph_window_base.hh`.

10.204.3.2 template<typename P , typename E > bool mln::graph_window_base< P, E >::is_centered () const [inline]

Is the window centered?

Definition at line 112 of file `graph_window_base.hh`.

10.204.3.3 `template<typename P , typename E > bool mln::graph_window_base< P, E >::is_empty () const [inline]`

Interface of the concept [Window](#).

Is the window is empty?

Definition at line 104 of file graph_window_base.hh.

10.204.3.4 `template<typename P , typename E > bool mln::graph_window_base< P, E >::is_symmetric () const [inline]`

Is the window symmetric?

Definition at line 120 of file graph_window_base.hh.

10.204.3.5 `template<typename P , typename E > bool mln::graph_window_base< P, E >::is_valid () const [inline]`

Return true by default.

Reimplemented in [mln::graph_elt_window_if< G, S, I >](#).

Definition at line 153 of file graph_window_base.hh.

10.204.3.6 `template<typename P , typename E > graph_window_base< P, E > & mln::graph_window_base< P, E >::sym () [inline]`

Apply a central symmetry to the target window.

Definition at line 137 of file graph_window_base.hh.

10.205 mln::graph_window_if_piter< S, W, I > Class Template Reference

Forward iterator on line graph window.

```
#include <graph_window_if_piter.hh>
```

Inherits [site_relative_iterator_base< W, graph_window_if_piter< S, W, I > >](#), and [is_masked_impl_selector< S, W::mask_t::domain_t, graph_window_if_piter< S, W, I > >](#).

Public Types

- typedef `S::fun_t::result P`
Associated types.

Public Member Functions

- void [next](#) ()
Go to the next element.

- [graph_window_if_piter](#) ()
Construction.
- const S::graph_element & [element](#) () const
Return the graph element pointed by this iterator.
- unsigned [id](#) () const
Return the graph element id.

10.205.1 Detailed Description

`template<typename S, typename W, typename I> class mln::graph_window_if_piter< S, W, I >`

Forward iterator on line graph window.

Definition at line 47 of file `graph_window_if_piter.hh`.

10.205.2 Member Typedef Documentation

10.205.2.1 `template<typename S , typename W , typename I > typedef S::fun_t ::result
mln::graph_window_if_piter< S, W, I >::P`

Associated types.

Definition at line 60 of file `graph_window_if_piter.hh`.

10.205.3 Constructor & Destructor Documentation

10.205.3.1 `template<typename S , typename W , typename I > mln::graph_window_if_piter< S,
W, I >::graph_window_if_piter () [inline]`

Construction.

Definition at line 122 of file `graph_window_if_piter.hh`.

10.205.4 Member Function Documentation

10.205.4.1 `template<typename S , typename W , typename I > const S::graph_element &
mln::graph_window_if_piter< S, W, I >::element () const [inline]`

Return the graph element pointed by this iterator.

Definition at line 213 of file `graph_window_if_piter.hh`.

10.205.4.2 `template<typename S , typename W , typename I > unsigned
mln::graph_window_if_piter< S, W, I >::id () const [inline]`

Return the graph element id.

FIXME: we do not want to have this member since there is an automatic conversion to the graph element. C++ does not seem to use this conversion operator.

Definition at line 221 of file graph_window_if_piter.hh.

10.205.4.3 void mln::Site_Iterator< graph_window_if_piter< S, W, I > >::next ()
[inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.206 mln::graph_window_piter< S, W, I > Class Template Reference

Forward iterator on line graph window.

```
#include <graph_window_piter.hh>
```

Inherits site_relative_iterator_base< W, graph_window_piter< S, W, I >, W::center_t >, and impl_selector< W::center_t, W::psite, graph_window_piter< S, W, I > >.

Public Types

- typedef S::fun_t::result **P**
Associated types
Type of the window elements.
- typedef W::center_t **center_t**
Type of the window center.
- typedef W::graph_element **graph_element**
Type of the graph element pointed by this iterator.

Public Member Functions

- void **change_target_site_set** (const S &s)
Change the target site set.
- void **next** ()
Go to the next element.
- const S & **target_site_set** () const

Return the target site set.

- [graph_window_piter](#) ()
Construction.
- `template<typename Pref >`
`graph_window_piter (const Window< W > &win, const Pref &p_ref)`
To be used in case the center and neighbor sites have the same type and belong to the same site set.
- `template<typename Pref >`
`graph_window_piter (const Window< W > &win, const Site_Set< S > &target_site_set, const Pref &p_ref)`
To be used in case center and neighbors sites do not have the same type and do not belong to the same site set.
- `const graph_element & element () const`
Return the graph element pointed by this iterator.
- `unsigned id () const`
Return the graph element id.

10.206.1 Detailed Description

`template<typename S, typename W, typename I> class mln::graph_window_piter< S, W, I >`

Forward iterator on line graph window.

Template Parameters

S is the site set type.

W is the window type.

I is the underlying iterator type.

Definition at line 99 of file `graph_window_piter.hh`.

10.206.2 Member Typedef Documentation

10.206.2.1 `template<typename S , typename W , typename I > typedef W::center_t mln::graph_window_piter< S, W, I >::center_t`

Type of the window center.

Definition at line 120 of file `graph_window_piter.hh`.

10.206.2.2 `template<typename S , typename W , typename I > typedef W::graph_element mln::graph_window_piter< S, W, I >::graph_element`

Type of the graph element pointed by this iterator.

Definition at line 122 of file `graph_window_piter.hh`.

10.206.2.3 `template<typename S , typename W , typename I > typedef S::fun_t ::result
mln::graph_window_piter< S, W, I >::P`

Associated types

Type of the window elements.

Definition at line 118 of file graph_window_piter.hh.

10.206.3 Constructor & Destructor Documentation

10.206.3.1 `template<typename S , typename W , typename I > mln::graph_window_piter< S, W,
I >::graph_window_piter () [inline]`

Construction.

Definition at line 226 of file graph_window_piter.hh.

10.206.3.2 `template<typename S , typename W , typename I > template<typename Pref >
mln::graph_window_piter< S, W, I >::graph_window_piter (const Window< W > &
win, const Pref & p_ref) [inline]`

To be used in case the center and neighbor sites have the same type and belong to the same site set.

Parameters

win The underlying window.

p_ref [Window](#) center.

Definition at line 235 of file graph_window_piter.hh.

10.206.3.3 `template<typename S , typename W , typename I > template<typename Pref >
mln::graph_window_piter< S, W, I >::graph_window_piter (const Window< W > &
win, const Site_Set< S > & target_site_set, const Pref & p_ref) [inline]`

To be used in case center and neighbors sites do not have the same type and do not belong to the same site set.

Parameters

win The underlying window.

target_site_set [Site](#) set in which neighbor sites are extracted.

p_ref [Window](#) center.

Definition at line 249 of file graph_window_piter.hh.

10.206.4 Member Function Documentation

10.206.4.1 `template<typename S , typename W , typename I > void mln::graph_window_piter<
S, W, I >::change_target_site_set (const S & s) [inline]`

Change the target site set.

[Window](#) elements different from the center come from the target site set.

Definition at line 357 of file `graph_window_piter.hh`.

10.206.4.2 `template<typename S , typename W , typename I > const graph_window_piter< S, W, I >::graph_element & mln::graph_window_piter< S, W, I >::element () const [inline]`

Return the graph element pointed by this iterator.

Definition at line 341 of file `graph_window_piter.hh`.

10.206.4.3 `template<typename S , typename W , typename I > unsigned mln::graph_window_piter< S, W, I >::id () const [inline]`

Return the graph element id.

FIXME: we do not want to have this member since there is an automatic conversion to the graph element. C++ does not seem to use this conversion operator.

Definition at line 349 of file `graph_window_piter.hh`.

10.206.4.4 `void mln::Site_Iterator< graph_window_piter< S, W, I > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.206.4.5 `template<typename S , typename W , typename I > const S & mln::graph_window_piter< S, W, I >::target_site_set () const [inline]`

Return the target site set.

[Window](#) elements different from the center come from the target site set.

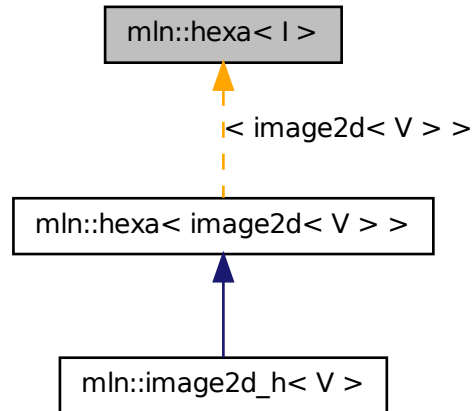
Definition at line 366 of file `graph_window_piter.hh`.

10.207 mln::hexa< I > Struct Template Reference

hexagonal image class.

```
#include <hexa.hh>
```

Inheritance diagram for mln::hexa< I >:



Public Types

- typedef hexa_bkd_piter_< box2d > bkd_piter
FIXME : should it be in box2d_h? Backward Site_Iterator associated type.
- typedef hexa_fwd_piter_< box2d > fwd_piter
FIXME : should it be in box2d_h? Forward Site_Iterator associated type.
- typedef I::lvalue lvalue
Lvalue associated type.
- typedef point2d_h psite
Point site type.
- typedef I::rvalue rvalue
Return type of read-only access.
- typedef hexa< tag::image_< I > > skeleton
Skeleton.
- typedef I::value value
Value associated type.

Public Member Functions

- const box2d_h & domain () const

Give the definition domain.

- `bool has (const psite &p) const`
Test if p belongs to the image domain.
- `hexa (I &ima)`
Constructor with an base image.
- `hexa ()`
Constructor without argument.
- `lvalue operator() (const point2d_h &p)`
Read-write access of pixel value at hexa point site p .
- `rvalue operator() (const point2d_h &p) const`
Read-only access of pixel value at hexa point site p .

10.207.1 Detailed Description

`template<typename I> struct mln::hexa< I >`

hexagonal image class. The parameter I is the type of the base image. This image class which handles hexagonal grid.

```
Ex : 1 3 5 7 9 11 0 2 4 6 8 10 ----- 0 XX| | | | |XX ----- 2 XX| | | | |XX
----- 4 XX| | | | |XX ----- 6 XX| | | | |XX ----- 8 XX| | | | |
|XX -----
```

Definition at line 116 of file hexa.hh.

10.207.2 Member Typedef Documentation

10.207.2.1 `template<typename I> typedef hexa_bkd_piter_<box2d> mln::hexa< I >::bkd_piter`

FIXME : should it be in box2d_h? Backward [Site_Iterator](#) associated type.

Definition at line 140 of file hexa.hh.

10.207.2.2 `template<typename I> typedef hexa_fwd_piter_<box2d> mln::hexa< I >::fwd_piter`

FIXME : should it be in box2d_h? Forward [Site_Iterator](#) associated type.

Definition at line 136 of file hexa.hh.

10.207.2.3 `template<typename I> typedef I ::lvalue mln::hexa< I >::lvalue`

Lvalue associated type.

Definition at line 126 of file hexa.hh.

10.207.2.4 `template<typename I> typedef point2d_h mln::hexa< I >::psite`

[Point](#) site type.

Reimplemented in [mln::image2d_h< V >](#).

Definition at line 132 of file hexa.hh.

10.207.2.5 `template<typename I> typedef I ::rvalue mln::hexa< I >::rvalue`

Return type of read-only access.

Definition at line 129 of file hexa.hh.

10.207.2.6 `template<typename I> typedef hexa< tag::image_<I> > mln::hexa< I >::skeleton`

Skeleton.

Definition at line 120 of file hexa.hh.

10.207.2.7 `template<typename I> typedef I ::value mln::hexa< I >::value`

[Value](#) associated type.

Definition at line 123 of file hexa.hh.

10.207.3 **Constructor & Destructor Documentation****10.207.3.1** `template<typename I> mln::hexa< I >::hexa () [inline]`

Constructor without argument.

Definition at line 215 of file hexa.hh.

10.207.3.2 `template<typename I> mln::hexa< I >::hexa (I & ima) [inline]`

Constructor with an base image.

Definition at line 222 of file hexa.hh.

10.207.4 **Member Function Documentation****10.207.4.1** `template<typename I> const box2d_h & mln::hexa< I >::domain () const [inline]`

Give the definition domain.

Definition at line 250 of file hexa.hh.

10.207.4.2 `template<typename I> bool mln::hexa< I >::has (const psite & p) const [inline]`

Test if *p* belongs to the image domain.

Definition at line 259 of file hexa.hh.

Referenced by `mln::hexa< I >::operator()`.

10.207.4.3 `template<typename I> hexa< I >::rvalue mln::hexa< I >::operator() (const point2d_h & p) const [inline]`

Read-only access of pixel value at hexa point site `p`.

Definition at line 230 of file hexa.hh.

References `mln::hexa< I >::has()`.

10.207.4.4 `template<typename I> hexa< I >::lvalue mln::hexa< I >::operator() (const point2d_h & p) [inline]`

Read-write access of pixel value at hexa point site `p`.

Definition at line 240 of file hexa.hh.

References `mln::hexa< I >::has()`.

10.208 `mln::histo::array< T >` Struct Template Reference

Generic histogram class over a value set with type `T`.

```
#include <array.hh>
```

10.208.1 Detailed Description

```
template<typename T> struct mln::histo::array< T >
```

Generic histogram class over a value set with type `T`.

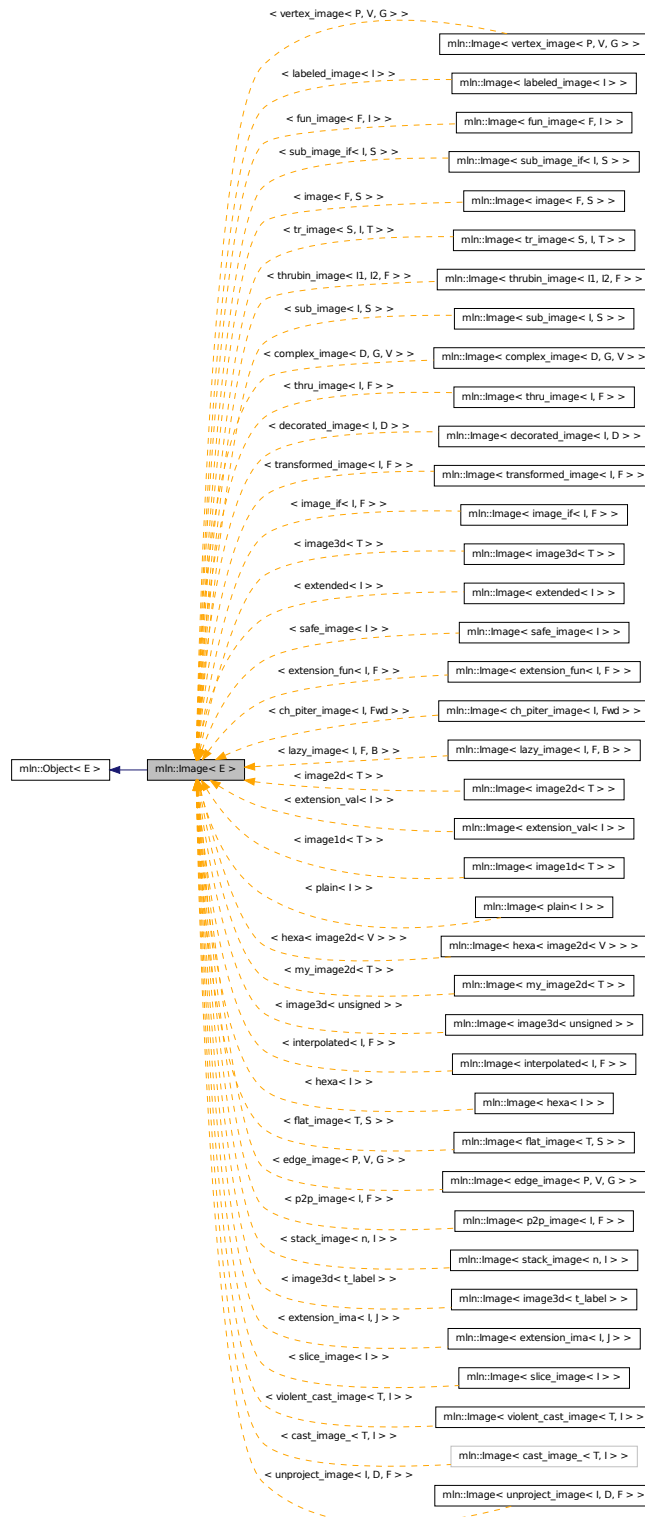
Definition at line 48 of file `histo/array.hh`.

10.209 `mln::Image< E >` Struct Template Reference

Base class for implementation of image classes.

```
#include <image.hh>
```

Inheritance diagram for mln::Image< E >:



10.209.1 Detailed Description

`template<typename E> struct mln::Image< E >`

Base class for implementation of image classes.

See also

[mln::doc::Image](#) for a complete documentation of this class contents.

Definition at line 73 of file core/concept/image.hh.

10.210 mln::image1d< T > Struct Template Reference

Basic 1D image class.

```
#include <image1d.hh>
```

Inherits `image_primary< T, box1d, image1d< T > >`.

Public Types

- typedef T & [lvalue](#)
Return type of read-write access.
- typedef const T & [rvalue](#)
Return type of read-only access.
- typedef [image1d< tag::value_< T > >](#) [skeleton](#)
Skeleton.
- typedef T [value](#)
Value associated type.

Public Member Functions

- const [box1d](#) & [bbox](#) () const
Give the bounding box domain.
- unsigned [border](#) () const
Give the border thickness.
- const T * [buffer](#) () const
Give a hook to the value buffer.
- T * [buffer](#) ()
Give a hook to the value buffer.
- int [delta_index](#) (const [dpoint1d](#) &dp) const

Give the offset corresponding to the delta-point Δp .

- const `box1d` & `domain` () const
Give the definition domain.
- const T & `element` (unsigned i) const
Read-only access to the i -th image value (including the border).
- T & `element` (unsigned i)
Read-write access to the i -th image value (including the border).
- bool `has` (const `point1d` &p) const
Test if p is valid.
- `image1d` ()
Constructor without argument.
- `image1d` (const `box1d` &b, unsigned bdr=border::thickness)
Constructor with a box and the border thickness.
- `image1d` (unsigned ninds, unsigned bdr=border::thickness)
Constructor with the number of indices and the border thickness.
- unsigned `nelements` () const
Give the number of cells (points including border ones).
- unsigned `ninds` () const
Give the number of indexes.
- T & `operator()` (const `point1d` &p)
Read-write access to the image value located at point p .
- const T & `operator()` (const `point1d` &p) const
Read-only access to the image value located at point p .
- `point1d` `point_at_index` (unsigned i) const
Give the point corresponding to the offset o .
- const `box1d` & `vbbox` () const
virtual box, i.e., box including the virtual border

10.210.1 Detailed Description

`template<typename T> struct mln::image1d< T >`

Basic 1D image class. The parameter T is the type of pixel values. This image class stores data in memory and has a virtual border with constant thickness before and after data.

Definition at line 155 of file `image1d.hh`.

10.210.2 Member Typedef Documentation

10.210.2.1 `template<typename T> typedef T& mln::image1d< T >::lvalue`

Return type of read-write access.

Definition at line 167 of file image1d.hh.

10.210.2.2 `template<typename T> typedef const T& mln::image1d< T >::rvalue`

Return type of read-only access.

Definition at line 164 of file image1d.hh.

10.210.2.3 `template<typename T> typedef image1d< tag::value_<T> > mln::image1d< T >::skeleton`

Skeleton.

Definition at line 170 of file image1d.hh.

10.210.2.4 `template<typename T> typedef T mln::image1d< T >::value`

[Value](#) associated type.

Definition at line 161 of file image1d.hh.

10.210.3 Constructor & Destructor Documentation

10.210.3.1 `template<typename T > mln::image1d< T >::image1d () [inline]`

Constructor without argument.

Definition at line 371 of file image1d.hh.

10.210.3.2 `template<typename T > mln::image1d< T >::image1d (unsigned ninds, unsigned bdr = border::thickness) [inline]`

Constructor with the number of indices and the border thickness.

Definition at line 384 of file image1d.hh.

References `mln::make::box1d()`.

10.210.3.3 `template<typename T > mln::image1d< T >::image1d (const box1d & b, unsigned bdr = border::thickness) [inline]`

Constructor with a box and the border thickness.

Definition at line 377 of file image1d.hh.

10.210.4 Member Function Documentation

10.210.4.1 `template<typename T > const box1d & mln::image1d< T >::bbox () const [inline]`

Give the bounding box domain.

Definition at line 411 of file image1d.hh.

10.210.4.2 `template<typename T > unsigned mln::image1d< T >::border () const [inline]`

Give the border thickness.

Definition at line 429 of file image1d.hh.

10.210.4.3 `template<typename T > const T * mln::image1d< T >::buffer () const [inline]`

Give a hook to the value buffer.

Definition at line 520 of file image1d.hh.

10.210.4.4 `template<typename T > T * mln::image1d< T >::buffer () [inline]`

Give a hook to the value buffer.

Definition at line 529 of file image1d.hh.

10.210.4.5 `template<typename T > int mln::image1d< T >::delta_index (const dpoint1d & dp) const [inline]`

Give the offset corresponding to the delta-point *dp*.

Definition at line 538 of file image1d.hh.

10.210.4.6 `template<typename T > const box1d & mln::image1d< T >::domain () const [inline]`

Give the definition domain.

Definition at line 402 of file image1d.hh.

10.210.4.7 `template<typename T > const T & mln::image1d< T >::element (unsigned i) const [inline]`

Read-only access to the *i*-th image value (including the border).

Definition at line 502 of file image1d.hh.

References `mln::image1d< T >::nelements()`.

10.210.4.8 `template<typename T> T & mln::image1d< T >::element (unsigned i)`
[inline]

Read-write access to the *i*-th image value (including the border).

Definition at line 511 of file image1d.hh.

References mln::image1d< T >::nelements().

10.210.4.9 `template<typename T> bool mln::image1d< T >::has (const point1d & p) const`
[inline]

Test if *p* is valid.

Definition at line 447 of file image1d.hh.

Referenced by mln::image1d< T >::operator()().

10.210.4.10 `template<typename T> unsigned mln::image1d< T >::nelements () const`
[inline]

Give the number of cells (points including border ones).

Definition at line 438 of file image1d.hh.

Referenced by mln::image1d< T >::element(), and mln::image1d< T >::point_at_index().

10.210.4.11 `template<typename T> unsigned mln::image1d< T >::ninds () const` **[inline]**

Give the number of indexes.

Definition at line 483 of file image1d.hh.

10.210.4.12 `template<typename T> const T & mln::image1d< T >::operator() (const point1d`
& p) const **[inline]**

Read-only access to the image value located at point *p*.

Definition at line 456 of file image1d.hh.

References mln::image1d< T >::has().

10.210.4.13 `template<typename T> T & mln::image1d< T >::operator() (const point1d & p)`
[inline]

Read-write access to the image value located at point *p*.

Definition at line 465 of file image1d.hh.

References mln::image1d< T >::has().

10.210.4.14 `template<typename T> point1d mln::image1d< T >::point_at_index (unsigned i)`
const **[inline]**

Give the point corresponding to the offset *o*.

Definition at line 548 of file image1d.hh.

References mln::image1d< T >::nelements().

10.210.4.15 `template<typename T > const box1d & mln::image1d< T >::vbbox () const` [inline]

virtual box, i.e., box including the virtual border

Definition at line 420 of file image1d.hh.

10.211 mln::image2d< T > Class Template Reference

Basic 2D image class.

```
#include <image2d.hh>
```

Inherits image_primary< T, mln::box2d, image2d< T > >.

Public Types

- typedef T & [lvalue](#)
Return type of read-write access.
- typedef const T & [rvalue](#)
Return type of read-only access.
- typedef [image2d](#)< tag::value_< T > > [skeleton](#)
Skeleton.
- typedef T [value](#)
Value associated type.

Public Member Functions

- const [box2d](#) & [bbox](#) () const
Give the bounding box domain.
- unsigned [border](#) () const
Give the border thickness.
- const T * [buffer](#) () const
Give a hook to the value buffer.
- T * [buffer](#) ()
Give a hook to the value buffer.
- int [delta_index](#) (const [dpoint2d](#) &dp) const
Give the delta-index corresponding to the delta-point dp.

- `const box2d & domain () const`
Give the definition domain.
- `T & element (unsigned i)`
Read-write access to the image value located at index i .
- `const T & element (unsigned i) const`
Read-only access to the image value located at index i .
- `bool has (const point2d &p) const`
Test if p is valid.
- `image2d (int nrows, int ncols, unsigned bdr=border::thickness)`
Constructor with the numbers of rows and columns and the border thickness.
- `image2d ()`
Constructor without argument.
- `image2d (const box2d &b, unsigned bdr=border::thickness)`
Constructor with a box and the border thickness (default is 3).
- `unsigned ncols () const`
Give the number of columns.
- `unsigned nelements () const`
Give the number of elements (points including border ones).
- `unsigned nrows () const`
Give the number of rows.
- `const T & operator() (const point2d &p) const`
Read-only access to the image value located at point p .
- `T & operator() (const point2d &p)`
Read-write access to the image value located at point p .
- `point2d point_at_index (unsigned i) const`
Give the point corresponding to the index i .

10.211.1 Detailed Description

```
template<typename T> class mln::image2d< T >
```

Basic 2D image class. The parameter `T` is the type of pixel values. This image class stores data in memory and has a virtual border with constant thickness around data.

Definition at line 136 of file `core/image/image2d.hh`.

10.211.2 Member Typedef Documentation

10.211.2.1 `template<typename T> typedef T& mln::image2d< T >::lvalue`

Return type of read-write access.

Definition at line 148 of file core/image/image2d.hh.

10.211.2.2 `template<typename T> typedef const T& mln::image2d< T >::rvalue`

Return type of read-only access.

Definition at line 145 of file core/image/image2d.hh.

10.211.2.3 `template<typename T> typedef image2d< tag::value_<T> > mln::image2d< T >::skeleton`

Skeleton.

Definition at line 152 of file core/image/image2d.hh.

10.211.2.4 `template<typename T> typedef T mln::image2d< T >::value`

[Value](#) associated type.

Definition at line 142 of file core/image/image2d.hh.

10.211.3 Constructor & Destructor Documentation

10.211.3.1 `template<typename T > mln::image2d< T >::image2d () [inline]`

Constructor without argument.

Definition at line 394 of file core/image/image2d.hh.

10.211.3.2 `template<typename T > mln::image2d< T >::image2d (int nrows, int ncols, unsigned bdr = border::thickness) [inline]`

Constructor with the numbers of rows and columns and the border thickness.

Definition at line 400 of file core/image/image2d.hh.

References `mln::make::box2d()`.

10.211.3.3 `template<typename T > mln::image2d< T >::image2d (const box2d & b, unsigned bdr = border::thickness) [inline]`

Constructor with a box and the border thickness (default is 3).

Definition at line 407 of file core/image/image2d.hh.

10.211.4 Member Function Documentation

10.211.4.1 `template<typename T> const box2d & mln::image2d<T>::bbox () const [inline]`

Give the bounding box domain.

Definition at line 433 of file core/image/image2d.hh.

10.211.4.2 `template<typename T> unsigned mln::image2d<T>::border () const [inline]`

Give the border thickness.

Definition at line 520 of file core/image/image2d.hh.

10.211.4.3 `template<typename T> const T * mln::image2d<T>::buffer () const [inline]`

Give a hook to the value buffer.

Definition at line 556 of file core/image/image2d.hh.

10.211.4.4 `template<typename T> T * mln::image2d<T>::buffer () [inline]`

Give a hook to the value buffer.

Definition at line 565 of file core/image/image2d.hh.

10.211.4.5 `template<typename T> int mln::image2d<T>::delta_index (const dpoint2d & dp) const [inline]`

Give the delta-index corresponding to the delta-point dp.

Definition at line 574 of file core/image/image2d.hh.

10.211.4.6 `template<typename T> const box2d & mln::image2d<T>::domain () const [inline]`

Give the definition domain.

Definition at line 424 of file core/image/image2d.hh.

Referenced by mln::morpho::line_gradient(), mln::make_debug_graph_image(), and mln::io::txt::save().

10.211.4.7 `template<typename T> const T & mln::image2d<T>::element (unsigned i) const [inline]`

Read-only access to the image value located at index i.

Definition at line 538 of file core/image/image2d.hh.

References mln::image2d<T>::nelements().

10.211.4.8 `template<typename T> T & mln::image2d< T >::element (unsigned i)`
`[inline]`

Read-write access to the image value located at index *i*.

Definition at line 547 of file core/image/image2d.hh.

References mln::image2d< T >::nelements().

10.211.4.9 `template<typename T> bool mln::image2d< T >::has (const point2d & p) const`
`[inline]`

Test if *p* is valid.

Definition at line 451 of file core/image/image2d.hh.

Referenced by mln::image2d< T >::operator>(), and mln::debug::put_word().

10.211.4.10 `template<typename T> unsigned mln::image2d< T >::ncols () const` `[inline]`

Give the number of columns.

Definition at line 508 of file core/image/image2d.hh.

10.211.4.11 `template<typename T> unsigned mln::image2d< T >::nelements () const`
`[inline]`

Give the number of elements (points including border ones).

Definition at line 529 of file core/image/image2d.hh.

Referenced by mln::image2d< T >::element(), and mln::image2d< T >::point_at_index().

10.211.4.12 `template<typename T> unsigned mln::image2d< T >::nrows () const` `[inline]`

Give the number of rows.

Definition at line 499 of file core/image/image2d.hh.

10.211.4.13 `template<typename T> T & mln::image2d< T >::operator() (const point2d & p)`
`[inline]`

Read-write access to the image value located at point *p*.

Definition at line 469 of file core/image/image2d.hh.

References mln::image2d< T >::has().

10.211.4.14 `template<typename T> const T & mln::image2d< T >::operator() (const point2d`
`& p) const` `[inline]`

Read-only access to the image value located at point *p*.

Definition at line 460 of file core/image/image2d.hh.

References mln::image2d< T >::has().

10.211.4.15 `template<typename T> point2d mln::image2d< T >::point_at_index (unsigned i) const [inline]`

Give the point corresponding to the index *i*.

Definition at line 584 of file core/image/image2d.hh.

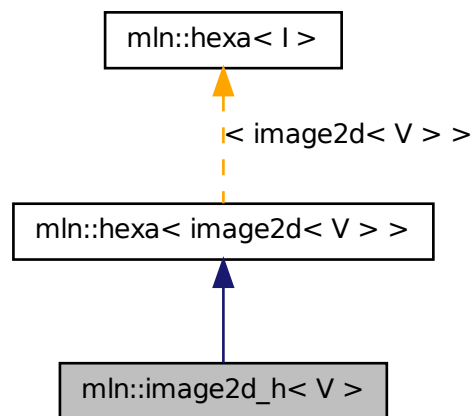
References `mln::image2d< T >::nelements()`.

10.212 mln::image2d_h< V > Struct Template Reference

2d image based on an hexagonal mesh.

```
#include <image2d_h.hh>
```

Inheritance diagram for `mln::image2d_h< V >`:



Public Types

- typedef `hexa_bkd_piter_< box2d > bkd_piter`
FIXME : should it be in box2d_h? Backward Site_Iterator associated type.
- typedef `hexa_fwd_piter_< box2d > fwd_piter`
FIXME : should it be in box2d_h? Forward Site_Iterator associated type.
- typedef `image2d< V >::lvalue lvalue`
Lvalue associated type.
- typedef `point2d_h psite`
Point site type.

- typedef `image2d< V >::rvalue rvalue`
Return type of read-only access.
- typedef `hexa< tag::image_< image2d< V > > > skeleton`
Skeleton.
- typedef `image2d< V >::value value`
Value associated type.

Public Member Functions

- const `box2d_h & domain () const`
Give the definition domain.
- bool `has (const psite &p) const`
*Test if *p* belongs to the image domain.*
- `image2d_h (int nrows, int ncols, unsigned bdr=border::thickness)`
Constructor with the numbers of rows and columns border thickness.
- `lvalue operator() (const point2d_h &p)`
*Read-write access of pixel value at hexa point site *p*.*
- `rvalue operator() (const point2d_h &p) const`
*Read-only access of pixel value at hexa point site *p*.*

10.212.1 Detailed Description

`template<typename V> struct mln::image2d_h< V >`

2d image based on an hexagonal mesh.

Definition at line 50 of file `image2d_h.hh`.

10.212.2 Member Typedef Documentation

10.212.2.1 `typedef hexa_bkd_piter_<box2d> mln::hexa< image2d< V > >::bkd_piter`
`[inherited]`

FIXME : should it be in `box2d_h`? Backward Site_Iterator associated type.

Definition at line 140 of file `hexa.hh`.

10.212.2.2 `typedef hexa_fwd_piter_<box2d> mln::hexa< image2d< V > >::fwd_piter`
`[inherited]`

FIXME : should it be in `box2d_h`? Forward Site_Iterator associated type.

Definition at line 136 of file `hexa.hh`.

10.212.2.3 typedef image2d< V > ::lvalue mln::hexa< image2d< V > >::lvalue [inherited]

Lvalue associated type.

Definition at line 126 of file hexa.hh.

10.212.2.4 template<typename V > typedef point2d_h mln::image2d_h< V >::psite

[Point](#) site type.

Reimplemented from [mln::hexa< image2d< V > >](#).

Definition at line 56 of file image2d_h.hh.

10.212.2.5 typedef image2d< V > ::rvalue mln::hexa< image2d< V > >::rvalue [inherited]

Return type of read-only access.

Definition at line 129 of file hexa.hh.

10.212.2.6 typedef hexa< tag::image_<image2d< V > > > mln::hexa< image2d< V > >::skeleton [inherited]

Skeleton.

Definition at line 120 of file hexa.hh.

10.212.2.7 typedef image2d< V > ::value mln::hexa< image2d< V > >::value [inherited]

Value associated type.

Definition at line 123 of file hexa.hh.

10.212.3 Constructor & Destructor Documentation**10.212.3.1 template<typename V > mln::image2d_h< V >::image2d_h (int *nrows*, int *ncols*, unsigned *bdr = border::thickness*) [inline]**

Constructor with the numbers of rows and columns border thickness.

image2d_h(3,6) will build this hexa image :

```
1 3 5 0 2 4 ----- 0| x x x | 2| x x x | 4| x x x
```

Definition at line 82 of file image2d_h.hh.

10.212.4 Member Function Documentation**10.212.4.1 const box2d_h& mln::hexa< image2d< V > >::domain () const [inherited]**

Give the definition domain.

10.212.4.2 `bool mln::hexa< image2d< V > >::has (const psite & p) const` `[inherited]`

Test if `p` belongs to the image domain.

10.212.4.3 `rvalue mln::hexa< image2d< V > >::operator() (const point2d_h & p) const` `[inherited]`

Read-only access of pixel value at hexa point site `p`.

10.212.4.4 `lvalue mln::hexa< image2d< V > >::operator() (const point2d_h & p)` `[inherited]`

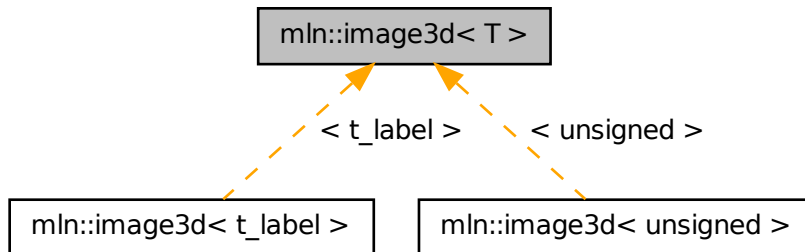
Read-write access of pixel value at hexa point site `p`.

10.213 `mln::image3d< T >` Struct Template Reference

Basic 3D image class.

```
#include <image3d.hh>
```

Inheritance diagram for `mln::image3d< T >`:

**Public Types**

- typedef `T` & `lvalue`
Return type of read-write access.
- typedef `const T` & `rvalue`
Return type of read-only access.
- typedef `image3d< tag::value_< T > >` `skeleton`
Skeleton.
- typedef `T` `value`

Value associated type.

Public Member Functions

- const `box3d` & `bbox` () const
Give the bounding box domain.
- unsigned `border` () const
Give the border thickness.
- const T * `buffer` () const
Give a hook to the value buffer.
- T * `buffer` ()
Give a hook to the value buffer.
- int `delta_index` (const `dpoint3d` &dp) const
Fast Image method.
- const `box3d` & `domain` () const
Give the definition domain.
- const T & `element` (unsigned i) const
*Read-only access to the image value located at index *i*.*
- T & `element` (unsigned i)
*Read-write access to the image value located at index *i*.*
- bool `has` (const `point3d` &p) const
*Test if *p* is valid.*
- `image3d` (const `box3d` &b, unsigned bdr=`border::thickness`)
Constructor with a box and the border thickness (default is 3).
- `image3d` ()
Constructor without argument.
- `image3d` (int nslis, int nrows, int ncols, unsigned bdr=`border::thickness`)
Constructor with the numbers of indexes and the border thickness.
- unsigned `ncols` () const
Give the number of columns.
- unsigned `nelements` () const
Give the number of cells (points including border ones).
- unsigned `nrows` () const
Give the number of rows.

- unsigned `nslis` () const
Give the number of slices.
- const T & `operator()` (const `point3d` &p) const
Read-only access to the image value located at point p.
- T & `operator()` (const `point3d` &p)
Read-write access to the image value located at point p.
- `point3d point_at_index` (unsigned o) const
Give the point corresponding to the offset o.
- const `box3d` & `vbbox` () const
virtual box, i.e., box including the virtual border

10.213.1 Detailed Description

template<typename T> struct mln::image3d< T >

Basic 3D image class. The parameter T is the type of pixel values. This image class stores data in memory and has a virtual border with constant thickness around data.

Definition at line 130 of file core/image/image3d.hh.

10.213.2 Member Typedef Documentation

10.213.2.1 template<typename T> typedef T& mln::image3d< T >::lvalue

Return type of read-write access.

Definition at line 153 of file core/image/image3d.hh.

10.213.2.2 template<typename T> typedef const T& mln::image3d< T >::rvalue

Return type of read-only access.

Definition at line 150 of file core/image/image3d.hh.

10.213.2.3 template<typename T> typedef image3d< tag::value_<T> > mln::image3d< T >::skeleton

Skeleton.

Definition at line 157 of file core/image/image3d.hh.

10.213.2.4 template<typename T> typedef T mln::image3d< T >::value

[Value](#) associated type.

Definition at line 147 of file core/image/image3d.hh.

10.213.3 Constructor & Destructor Documentation

10.213.3.1 `template<typename T> mln::image3d< T >::image3d () [inline]`

Constructor without argument.

Definition at line 389 of file core/image/image3d.hh.

10.213.3.2 `template<typename T> mln::image3d< T >::image3d (const box3d & b, unsigned bdr = border::thickness) [inline]`

Constructor with a box and the border thickness (default is 3).

Definition at line 395 of file core/image/image3d.hh.

10.213.3.3 `template<typename T> mln::image3d< T >::image3d (int nslis, int nrows, int ncols, unsigned bdr = border::thickness) [inline]`

Constructor with the numbers of indexes and the border thickness.

Definition at line 402 of file core/image/image3d.hh.

References `mln::make::box3d()`.

10.213.4 Member Function Documentation

10.213.4.1 `template<typename T> const box3d & mln::image3d< T >::bbox () const [inline]`

Give the bounding box domain.

Definition at line 428 of file core/image/image3d.hh.

10.213.4.2 `template<typename T> unsigned mln::image3d< T >::border () const [inline]`

Give the border thickness.

Definition at line 446 of file core/image/image3d.hh.

10.213.4.3 `template<typename T> const T * mln::image3d< T >::buffer () const [inline]`

Give a hook to the value buffer.

Definition at line 554 of file core/image/image3d.hh.

10.213.4.4 `template<typename T> T * mln::image3d< T >::buffer () [inline]`

Give a hook to the value buffer.

Definition at line 563 of file core/image/image3d.hh.

10.213.4.5 `template<typename T> int mln::image3d< T >::delta_index (const dpoint3d & dp) const [inline]`

Fast [Image](#) method.

Give the offset corresponding to the delta-point `dp`.

Definition at line 572 of file `core/image/image3d.hh`.

10.213.4.6 `template<typename T> const box3d & mln::image3d< T >::domain () const [inline]`

Give the definition domain.

Definition at line 419 of file `core/image/image3d.hh`.

Referenced by `mln::accu::stat::operator==()`.

10.213.4.7 `template<typename T> const T & mln::image3d< T >::element (unsigned i) const [inline]`

Read-only access to the image value located at index `i`.

Definition at line 491 of file `core/image/image3d.hh`.

References `mln::image3d< T >::nelements()`.

10.213.4.8 `template<typename T> T & mln::image3d< T >::element (unsigned i) [inline]`

Read-write access to the image value located at index `i`.

Definition at line 500 of file `core/image/image3d.hh`.

References `mln::image3d< T >::nelements()`.

10.213.4.9 `template<typename T> bool mln::image3d< T >::has (const point3d & p) const [inline]`

Test if `p` is valid.

Definition at line 464 of file `core/image/image3d.hh`.

Referenced by `mln::image3d< T >::operator()()`.

10.213.4.10 `template<typename T> unsigned mln::image3d< T >::ncols () const [inline]`

Give the number of columns.

Definition at line 545 of file `core/image/image3d.hh`.

10.213.4.11 `template<typename T> unsigned mln::image3d< T >::nelements () const [inline]`

Give the number of cells (points including border ones).

Definition at line 455 of file core/image/image3d.hh.

Referenced by `mln::image3d< T >::element()`, and `mln::image3d< T >::point_at_index()`.

10.213.4.12 `template<typename T> unsigned mln::image3d< T >::nrows () const [inline]`

Give the number of rows.

Definition at line 536 of file core/image/image3d.hh.

10.213.4.13 `template<typename T> unsigned mln::image3d< T >::nslis () const [inline]`

Give the number of slices.

Definition at line 527 of file core/image/image3d.hh.

10.213.4.14 `template<typename T> T & mln::image3d< T >::operator() (const point3d & p) [inline]`

Read-write access to the image value located at point `p`.

Definition at line 482 of file core/image/image3d.hh.

References `mln::image3d< T >::has()`.

10.213.4.15 `template<typename T> const T & mln::image3d< T >::operator() (const point3d & p) const [inline]`

Read-only access to the image value located at point `p`.

Definition at line 473 of file core/image/image3d.hh.

References `mln::image3d< T >::has()`.

10.213.4.16 `template<typename T> point3d mln::image3d< T >::point_at_index (unsigned o) const [inline]`

Give the point corresponding to the offset `o`.

Definition at line 583 of file core/image/image3d.hh.

References `mln::image3d< T >::nelements()`.

10.213.4.17 `template<typename T> const box3d & mln::image3d< T >::vbbox () const [inline]`

virtual box, i.e., box including the virtual border

Definition at line 437 of file core/image/image3d.hh.

10.214 `mln::image_if< I, F > Struct Template Reference`

[Image](#) which domain is restricted by a function 'site -> Boolean'.

```
#include <image_if.hh>
```

Inherits image_domain_morpher< I, p_if< I::domain_t, F >, image_if< I, F > >.

Public Types

- typedef [image_if](#)< tag::image_< I >, tag::function_< F > > [skeleton](#)
Skeleton.

Public Member Functions

- const [p_if](#)< typename I::domain_t, F > & [domain](#) () const
Give the definition domain.
- [image_if](#) (I &ima, const F &f)
*Constructor from an image *ima* and a predicate *f*.*
- [image_if](#) ()
Constructor without argument.
- operator [image_if](#)< const I, F > () const
Const promotion via conversion.

10.214.1 Detailed Description

```
template<typename I, typename F> struct mln::image_if< I, F >
```

[Image](#) which domain is restricted by a function 'site -> Boolean'.

Definition at line 93 of file image_if.hh.

10.214.2 Member Typedef Documentation

10.214.2.1 `template<typename I, typename F> typedef image_if< tag::image_<I>, tag::function_<F> > mln::image_if< I, F >::skeleton`

Skeleton.

Definition at line 98 of file image_if.hh.

10.214.3 Constructor & Destructor Documentation

10.214.3.1 `template<typename I, typename F > mln::image_if< I, F >::image_if ()`
`[inline]`

Constructor without argument.

Definition at line 188 of file image_if.hh.

10.214.3.2 `template<typename I, typename F> mln::image_if< I, F >::image_if (I & ima, const F & f) [inline]`

Constructor from an image *ima* and a predicate *f*.

Definition at line 194 of file `image_if.hh`.

10.214.4 Member Function Documentation

10.214.4.1 `template<typename I, typename F> const p_if< typename I::domain_t, F > & mln::image_if< I, F >::domain () const [inline]`

Give the definition domain.

Definition at line 211 of file `image_if.hh`.

10.214.4.2 `template<typename I, typename F> mln::image_if< I, F >::operator image_if< const I, F > () const [inline]`

Const promotion via conversion.

Definition at line 219 of file `image_if.hh`.

10.215 mln::interpolated< I, F > Struct Template Reference

Makes the underlying image being accessed with floating coordinates.

```
#include <interpolated.hh>
```

Inherits `image_identity< I, I::domain_t, interpolated< I, F > >`.

Public Types

- typedef `I::lvalue` [lvalue](#)
Return type of read-write access.
- typedef `I::psite` [psite](#)
Point_Site associated type.
- typedef `I::rvalue` [rvalue](#)
Return type of read-only access.
- typedef `interpolated< tag::image_< I >, F >` [skeleton](#)
Skeleton.
- typedef `I::value` [value](#)
Value associated type.

Public Member Functions

- `template<typename C >`
`bool has (const mln::algebra::vec< I::psite::dim, C > &v) const`
Test if a pixel value is accessible at v.
- `interpolated (I &ima)`
Constructors.
- `bool is_valid () const`
Test if this image has been initialized.

10.215.1 Detailed Description

`template<typename I, template< class > class F> struct mln::interpolated< I, F >`

Makes the underlying image being accessed with floating coordinates.

Definition at line 84 of file interpolated.hh.

10.215.2 Member Typedef Documentation

10.215.2.1 `template<typename I , template< class > class F> typedef I ::lvalue`
`mln::interpolated< I, F >::lvalue`

Return type of read-write access.

Definition at line 98 of file interpolated.hh.

10.215.2.2 `template<typename I , template< class > class F> typedef I ::psite mln::interpolated<`
`I, F >::psite`

[Point_Site](#) associated type.

Definition at line 92 of file interpolated.hh.

10.215.2.3 `template<typename I , template< class > class F> typedef I ::rvalue`
`mln::interpolated< I, F >::rvalue`

Return type of read-only access.

Definition at line 101 of file interpolated.hh.

10.215.2.4 `template<typename I , template< class > class F> typedef interpolated<`
`tag::image_<I>, F > mln::interpolated< I, F >::skeleton`

Skeleton.

Definition at line 104 of file interpolated.hh.

10.215.2.5 `template<typename I , template< class > class F> typedef I ::value mln::interpolated< I, F >::value`

[Value](#) associated type.

Definition at line 95 of file interpolated.hh.

10.215.3 Constructor & Destructor Documentation

10.215.3.1 `template<typename I , template< class > class F> mln::interpolated< I, F >::interpolated (I & ima) [inline]`

Constructors.

FIXME: don't we want a 'const' here?

Definition at line 156 of file interpolated.hh.

10.215.4 Member Function Documentation

10.215.4.1 `template<typename I , template< class > class F> template<typename C > bool mln::interpolated< I, F >::has (const mln::algebra::vec< I::psite::dim, C > & v) const [inline]`

Test if a pixel value is accessible at v.

Definition at line 189 of file interpolated.hh.

10.215.4.2 `template<typename I , template< class > class F> bool mln::interpolated< I, F >::is_valid () const [inline]`

Test if this image has been initialized.

Definition at line 180 of file interpolated.hh.

10.216 mln::io::dicom::dicom_header Struct Reference

Store dicom file header.

```
#include <get_header.hh>
```

10.216.1 Detailed Description

Store dicom file header.

Definition at line 59 of file dicom/get_header.hh.

10.217 mln::io::dump::dump_header Struct Reference

Store dump file header.

```
#include <get_header.hh>
```

10.217.1 Detailed Description

Store dump file header.

Definition at line 53 of file dump/get_header.hh.

10.218 mln::io::fld::fld_header Struct Reference

Define the header structure of an AVS field data file.

```
#include <header.hh>
```

10.218.1 Detailed Description

Define the header structure of an AVS field data file.

Definition at line 45 of file header.hh.

10.219 mln::io::raw::raw_header Struct Reference

Store raw file header.

```
#include <get_header.hh>
```

10.219.1 Detailed Description

Store raw file header.

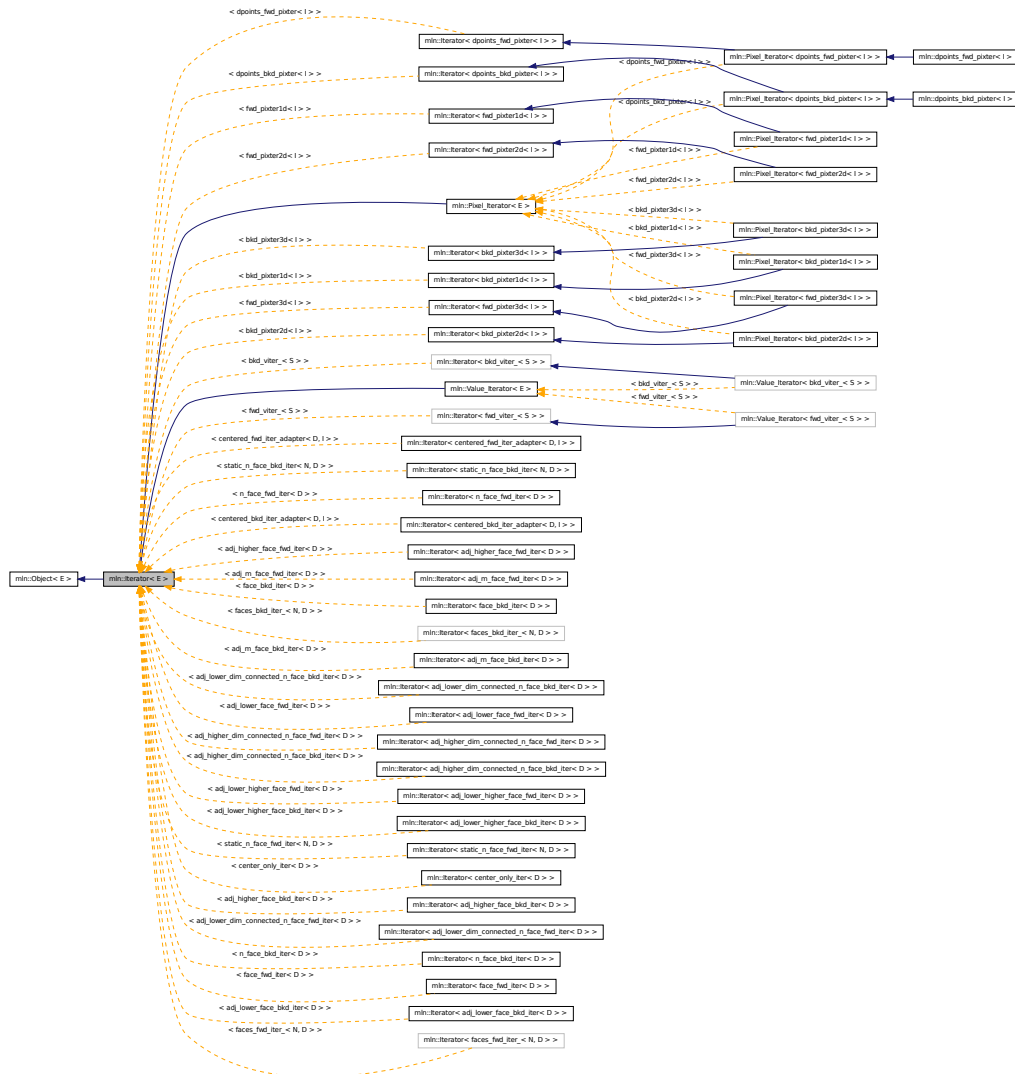
Definition at line 53 of file raw/get_header.hh.

10.220 mln::Iterator< E > Struct Template Reference

Base class for implementation classes that are iterators.

```
#include <iterator.hh>
```

Inheritance diagram for `mln::Iterator< E >`:



Public Member Functions

- void [next](#) ()

Go to the next element.

10.220.1 Detailed Description

`template<typename E> struct mln::Iterator< E >`

Base class for implementation classes that are iterators.

See also

[mln::doc::Iterator](#) for a complete documentation of this class contents.

Definition at line 75 of file iterator.hh.

10.220.2 Member Function Documentation**10.220.2.1** `template<typename E > void mln::Iterator< E >::next ()`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

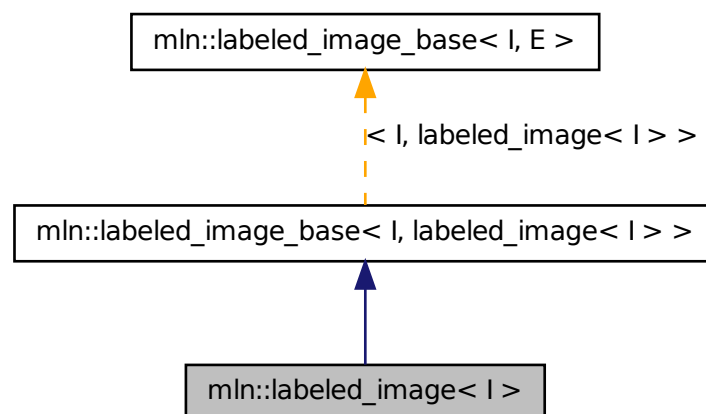
Definition at line 102 of file iterator.hh.

10.221 mln::labeled_image< I > Class Template Reference

Morpher providing an improved interface for labeled image.

```
#include <labeled_image.hh>
```

Inheritance diagram for mln::labeled_image< I >:



Public Types

- typedef `accu::shape::bbox`< typename I::psite >::result `bbox_t`
Type of the bounding component bounding boxes.
- typedef `labeled_image`< tag::image_< I > > `skeleton`
Skeleton.

Public Member Functions

- const `bbox_t` & `bbox` (const typename I::value &label) const
Return the bounding box of the component `label`.
- const `util::array`< `bbox_t` > & `bboxes` () const
Return the component bounding boxes.
- I::value `nlabels` () const
Return the number of labels;.
- `p_if`< mln_box(I), fun::eq_v2b_expr_< pw::value_< I >, pw::cst_< typename I::value > > >
`subdomain` (const typename I::value &label) const
Return the domain of the component with label `label`.
- `labeled_image` ()
Constructors
Constructor without argument.
- `labeled_image` (const I &ima, const typename I::value &nlabels)
Constructor from an image `ima` and the number of labels `nlabels`.
- `labeled_image` (const I &ima, const typename I::value &nlabels, const `util::array`< mln_box(I)> &bboxes)
Constructor from an image `ima`, the number of labels `nlabels` and the object bounding boxes.
- void `relabel` (const `Function_v2v`< F > &f)
Relabel according to a function.
- void `relabel` (const `Function_v2b`< F > &f)
Labels may be removed.

Protected Member Functions

- void `update_data` (const fun::i2v::array< typename I::value > &relabel_fun)
Update bounding boxes information.

10.221.1 Detailed Description

template<typename I> class mln::labeled_image< I >

Morpher providing an improved interface for labeled image.

Template Parameters

I The label image type.

This image type allows to access every site set at a given label.

This image type guaranties that labels are contiguous (from 1 to n).

Definition at line 105 of file labeled_image.hh.

10.221.2 Member Typedef Documentation

10.221.2.1 typedef accu::shape::bbox<typename I ::psite>::result mln::labeled_image_base< I, labeled_image< I > >::bbox_t [inherited]

Type of the bounding component bounding boxes.

Definition at line 124 of file labeled_image_base.hh.

10.221.2.2 template<typename I> typedef labeled_image< tag::image_<I> > mln::labeled_image< I >::skeleton

Skeleton.

Definition at line 113 of file labeled_image.hh.

10.221.3 Constructor & Destructor Documentation

10.221.3.1 template<typename I> mln::labeled_image< I >::labeled_image () [inline]

Constructors

Constructor without argument.

Definition at line 193 of file labeled_image.hh.

10.221.3.2 template<typename I> mln::labeled_image< I >::labeled_image (const I & ima, const typename I::value & nlabels) [inline]

Constructor from an image *ima* and the number of labels *nlabels*.

Definition at line 199 of file labeled_image.hh.

10.221.3.3 template<typename I> mln::labeled_image< I >::labeled_image (const I & ima, const typename I::value & nlabels, const util::array< mln_box(I)> & bboxes) [inline]

Constructor from an image *ima*, the number of labels *nlabels* and the object bounding boxes.

Definition at line 206 of file `labeled_image.hh`.

References `mln::data::compute()`.

10.221.4 Member Function Documentation

10.221.4.1 `const bbox_t& mln::labeled_image_base< I, labeled_image< I > >::bbox (const typename I::value & label) const` `[inherited]`

Return the bounding box of the component `label`.

10.221.4.2 `const util::array< bbox_t > & mln::labeled_image_base< I, labeled_image< I > >::bboxes () const` `[inherited]`

Return the component bounding boxes.

10.221.4.3 `I::value mln::labeled_image_base< I, labeled_image< I > >::nlabels () const` `[inherited]`

Return the number of labels;

10.221.4.4 `void mln::labeled_image_base< I, labeled_image< I > >::relabel (const Function_v2b< F > & f)` `[inherited]`

Labels may be removed.

This overload make sure the labeling is still contiguous.

10.221.4.5 `void mln::labeled_image_base< I, labeled_image< I > >::relabel (const Function_v2v< F > & f)` `[inherited]`

Relabel according to a function.

Merge or delete labels according to the given function. This method ensures that the labeling remains contiguous.

10.221.4.6 `p_if< mln_box(I), fun::eq_v2b_expr_< pw::value_< I >, pw::cst_< typename I::value > > > mln::labeled_image_base< I, labeled_image< I > >::subdomain (const typename I::value & label) const` `[inherited]`

Return the domain of the component with label `label`.

10.221.4.7 `void mln::labeled_image_base< I, labeled_image< I > >::update_data (const fun::i2v::array< typename I::value > & relabel_fun)` `[protected, inherited]`

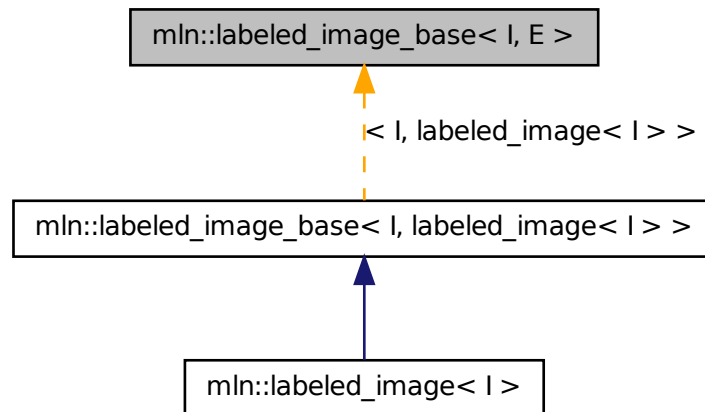
Update bounding boxes information.

10.222 mln::labeled_image_base< I, E > Class Template Reference

Base class Morpher providing an improved interface for labeled image.

```
#include <labeled_image_base.hh>
```

Inheritance diagram for mln::labeled_image_base< I, E >:



Public Types

- typedef [accu::shape::bbox](#)< typename I::psite >::result [bbox_t](#)

Type of the bounding component bounding boxes.

Public Member Functions

- const [bbox_t](#) & [bbox](#) (const typename I::value &[label](#)) const
Return the bounding box of the component [label](#).
- const [util::array](#)< [bbox_t](#) > & [bboxes](#) () const
Return the component bounding boxes.
- I::value [nlabels](#) () const
Return the number of labels;.
- [p_if](#)< mln_box(I), fun::eq_v2b_expr_< pw::value_< I >, pw::cst_< typename I::value > > > [subdomain](#) (const typename I::value &[label](#)) const
Return the domain of the component with label [label](#).
- [labeled_image_base](#) ()

*Constructors**Constructor without argument.*

- `template<typename F >`
`void relabel (const Function_v2v< F > &f)`
Relabel according to a function.
- `template<typename F >`
`void relabel (const Function_v2b< F > &f)`
Labels may be removed.

Protected Member Functions

- `void update_data (const fun::i2v::array< typename I::value > &relabel_fun)`
Update bounding boxes information.

10.222.1 Detailed Description

`template<typename I, typename E> class mln::labeled_image_base< I, E >`

Base class Morpher providing an improved interface for labeled image.

Template Parameters

I The label image type.

This image type allows to access every site set at a given label.

This image type guaranties that labels are contiguous (from 1 to n).

Definition at line 116 of file labeled_image_base.hh.

10.222.2 Member Typedef Documentation

10.222.2.1 `template<typename I, typename E> typedef accu::shape::bbox<typename I
::psite>::result mln::labeled_image_base< I, E >::bbox_t`

Type of the bounding component bounding boxes.

Definition at line 124 of file labeled_image_base.hh.

10.222.3 Constructor & Destructor Documentation

10.222.3.1 `template<typename I, typename E > mln::labeled_image_base< I, E
>::labeled_image_base () [inline]`

Constructors

Constructor without argument.

Definition at line 217 of file labeled_image_base.hh.

10.222.4 Member Function Documentation

10.222.4.1 `template<typename I, typename E > const labeled_image_base< I, E >::bbox_t & mln::labeled_image_base< I, E >::bbox (const typename I::value & label) const`

Return the bounding box of the component `label`.

Definition at line 305 of file `labeled_image_base.hh`.

Referenced by `mln::labeled_image_base< I, E >::subdomain()`.

10.222.4.2 `template<typename I , typename E > const util::array< typename labeled_image_base< I, E >::bbox_t > & mln::labeled_image_base< I, E >::bboxes () const`

Return the component bounding boxes.

Definition at line 313 of file `labeled_image_base.hh`.

10.222.4.3 `template<typename I , typename E > I::value mln::labeled_image_base< I, E >::nlabels () const [inline]`

Return the number of labels;

Definition at line 273 of file `labeled_image_base.hh`.

10.222.4.4 `template<typename I , typename E > template<typename F > void mln::labeled_image_base< I, E >::relabel (const Function_v2b< F > & f) [inline]`

Labels may be removed.

This overload make sure the labeling is still contiguous.

Definition at line 252 of file `labeled_image_base.hh`.

References `mln::labeling::relabel_inplace()`, `mln::make::relabelfun()`, and `mln::labeled_image_base< I, E >::update_data()`.

10.222.4.5 `template<typename I , typename E > template<typename F > void mln::labeled_image_base< I, E >::relabel (const Function_v2v< F > & f) [inline]`

Relabel according to a function.

Merge or delete labels according to the given function. This method ensures that the labeling remains contiguous.

Definition at line 226 of file `labeled_image_base.hh`.

References `mln::labeling::relabel_inplace()`, `mln::make::relabelfun()`, and `mln::labeled_image_base< I, E >::update_data()`.

10.222.4.6 `template<typename I, typename E > p_if< mln_box(I), fun::eq_v2b_expr_< pw::value_< I >, pw::cst_< typename I::value > > > mln::labeled_image_base< I, E >::subdomain (const typename I::value & label) const`

Return the domain of the component with label `label`.

Definition at line 322 of file `labeled_image_base.hh`.

References `mln::labeled_image_base< I, E >::bbox()`.

10.222.4.7 `template<typename I, typename E > void mln::labeled_image_base< I, E >::update_data (const fun::i2v::array< typename I::value > & relabel_fun) [protected]`

Update bounding boxes information.

Definition at line 281 of file `labeled_image_base.hh`.

References `mln::util::array< T >::size()`.

Referenced by `mln::labeled_image_base< I, E >::relabel()`.

10.223 mln::lazy_image< I, F, B > Struct Template Reference

[Image](#) values are computed on the fly.

```
#include <lazy_image.hh>
```

Inherits `image_identity< mln::trait::ch_value< I, F::result >::ret, I::domain_t, lazy_image< I, F, B > >`.

Public Types

- typedef `F::result` [lvalue](#)
Return type of read-write access.
- typedef `F::result` [rvalue](#)
Return type of read access.
- typedef `lazy_image< tag::image_< I >, F, B >` [skeleton](#)
Skeleton.

Public Member Functions

- const `box< typename I::psite > & domain ()` const
Return domain of lazyd_image.
- bool `has (const typename I::psite &)` const
Test if a pixel value is accessible at p.
- `lazy_image (const F &fun, const B &box)`
Constructors.

- [lazy_image](#) ()
Constructors.
- `F::result operator()` (const typename F::input &x) const
Read-only access of pixel value at F::input x.
- `lvalue operator()` (const typename I::psite &p)
Read and "write if possible" access of pixel value at point site p.
- `F::result operator()` (const typename F::input &x)
Read and "write if possible" access of pixel value at F::input x.
- `rvalue operator()` (const typename I::psite &p) const
Read-only access of pixel value at point site p.

10.223.1 Detailed Description

`template<typename I, typename F, typename B> struct mln::lazy_image< I, F, B >`

[Image](#) values are computed on the fly. The parameter `I` is the type of image. The parameter `F` is the type of function. The parameter `B` is the type of box.

This image class take a functor `fun` and a box `box`. Access to `ima(p)` where `p` include `box` return `fun(b)` lazily.

Definition at line 92 of file `lazy_image.hh`.

10.223.2 Member Typedef Documentation

10.223.2.1 `template<typename I, typename F, typename B> typedef F ::result mln::lazy_image< I, F, B >::lvalue`

Return type of read-write access.

Definition at line 104 of file `lazy_image.hh`.

10.223.2.2 `template<typename I, typename F, typename B> typedef F ::result mln::lazy_image< I, F, B >::rvalue`

Return type of read access.

Definition at line 101 of file `lazy_image.hh`.

10.223.2.3 `template<typename I, typename F, typename B> typedef lazy_image< tag::image_<I>, F, B > mln::lazy_image< I, F, B >::skeleton`

Skeleton.

Definition at line 107 of file `lazy_image.hh`.

10.223.3 Constructor & Destructor Documentation

10.223.3.1 `template<typename I, typename F, typename B> mln::lazy_image< I, F, B >::lazy_image ()`

Constructors.

10.223.3.2 `template<typename I, typename F, typename B> mln::lazy_image< I, F, B >::lazy_image (const F & fun, const B & box) [inline]`

Constructors.

Definition at line 161 of file lazy_image.hh.

10.223.4 Member Function Documentation

10.223.4.1 `template<typename I, typename F, typename B > const box< typename I::psite > & mln::lazy_image< I, F, B >::domain () const [inline]`

Return domain of lazyd_image.

Definition at line 226 of file lazy_image.hh.

10.223.4.2 `template<typename I, typename F, typename B > bool mln::lazy_image< I, F, B >::has (const typename I::psite & p) const [inline]`

Test if a pixel value is accessible at p.

Definition at line 175 of file lazy_image.hh.

10.223.4.3 `template<typename I, typename F, typename B > lazy_image< I, F, B >::rvalue mln::lazy_image< I, F, B >::operator() (const typename I::psite & p) const [inline]`

Read-only access of pixel value at point site p.

Definition at line 210 of file lazy_image.hh.

10.223.4.4 `template<typename I, typename F, typename B > F::result mln::lazy_image< I, F, B >::operator() (const typename F::input & x) [inline]`

Read and "write if possible" access of pixel value at F::input x.

Definition at line 197 of file lazy_image.hh.

10.223.4.5 `template<typename I, typename F, typename B > F::result mln::lazy_image< I, F, B >::operator() (const typename F::input & x) const [inline]`

Read-only access of pixel value at F::input x.

Definition at line 183 of file lazy_image.hh.

10.223.4.6 `template<typename I, typename F, typename B > lazy_image< I, F, B >::lvalue
mln::lazy_image< I, F, B >::operator() (const typename I::psite & p) [inline]`

Read and "write if possible" access of pixel value at point site p.

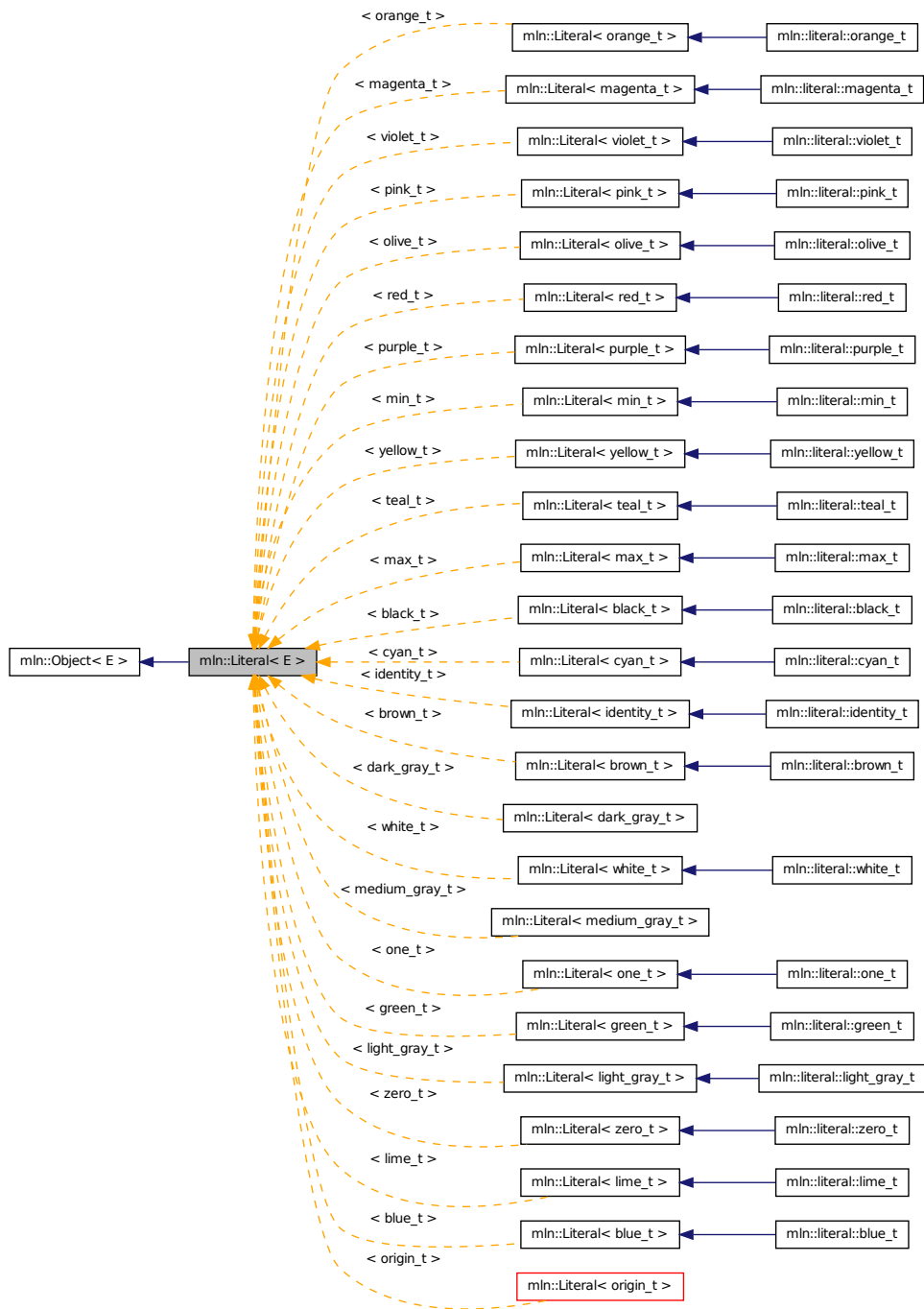
Definition at line 218 of file lazy_image.hh.

10.224 mln::Literal< E > Struct Template Reference

Base class for implementation classes of literals.

```
#include <literal.hh>
```

Inheritance diagram for mln::Literal< E >:



10.224.1 Detailed Description

```
template<typename E> struct mln::Literal< E >
```

Base class for implementation classes of literals.

See also

`mln::doc::Literal` for a complete documentation of this class contents.

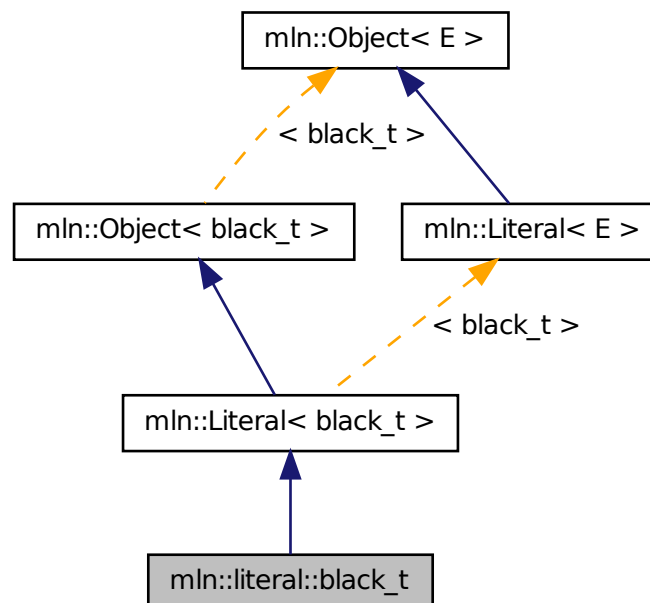
Definition at line 56 of file `literal.hh`.

10.225 mln::literal::black_t Struct Reference

Type of literal black.

```
#include <black.hh>
```

Inheritance diagram for `mln::literal::black_t`:



10.225.1 Detailed Description

Type of literal black.

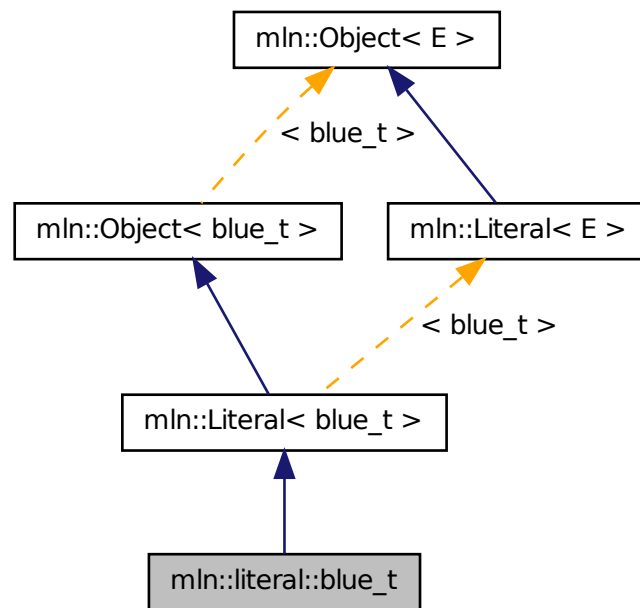
Definition at line 43 of file `black.hh`.

10.226 mln::literal::blue_t Struct Reference

Type of literal blue.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::blue_t:



10.226.1 Detailed Description

Type of literal blue.

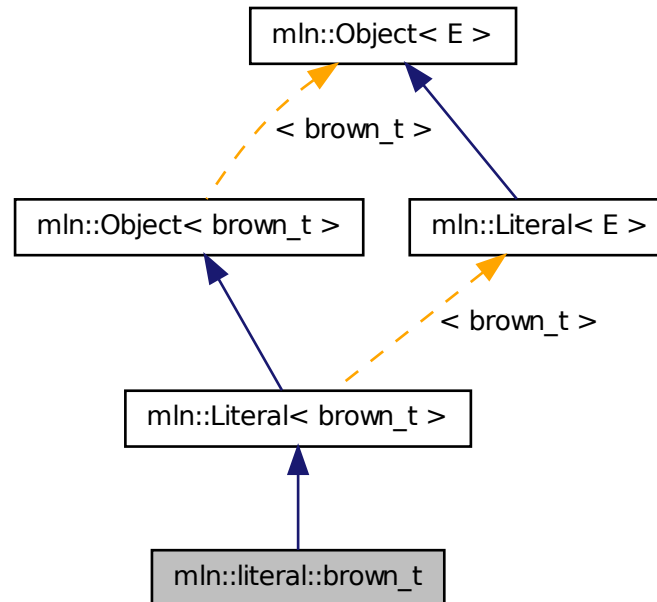
Definition at line 53 of file colors.hh.

10.227 mln::literal::brown_t Struct Reference

Type of literal brown.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::brown_t:



10.227.1 Detailed Description

Type of literal brown.

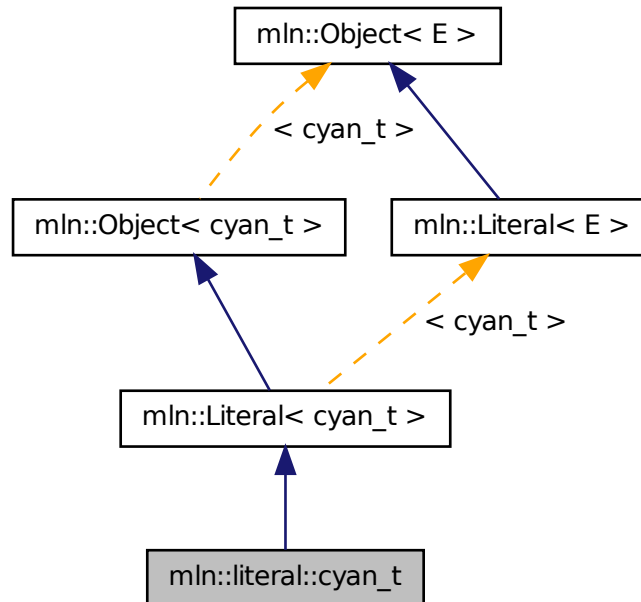
Definition at line 58 of file colors.hh.

10.228 mln::literal::cyan_t Struct Reference

Type of literal cyan.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::cyan_t:



10.228.1 Detailed Description

Type of literal cyan.

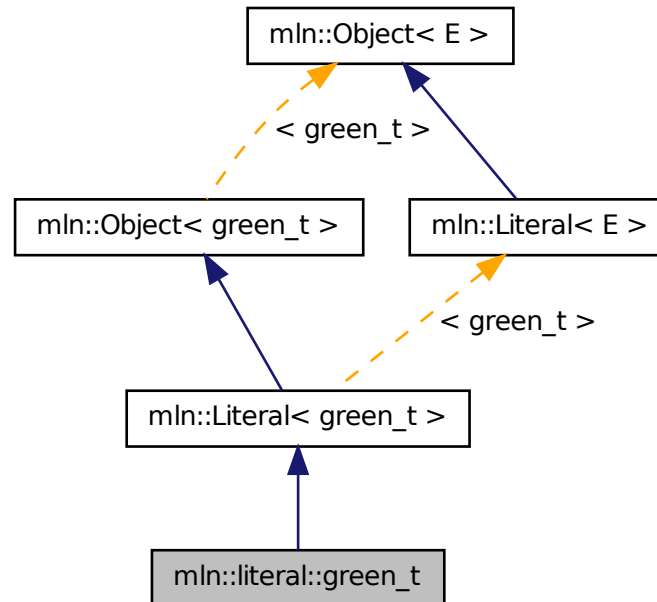
Definition at line 93 of file colors.hh.

10.229 mln::literal::green_t Struct Reference

Type of literal green.

```
#include <colors.hh>
```


Inheritance diagram for mln::literal::green_t:



10.229.1 Detailed Description

Type of literal green.

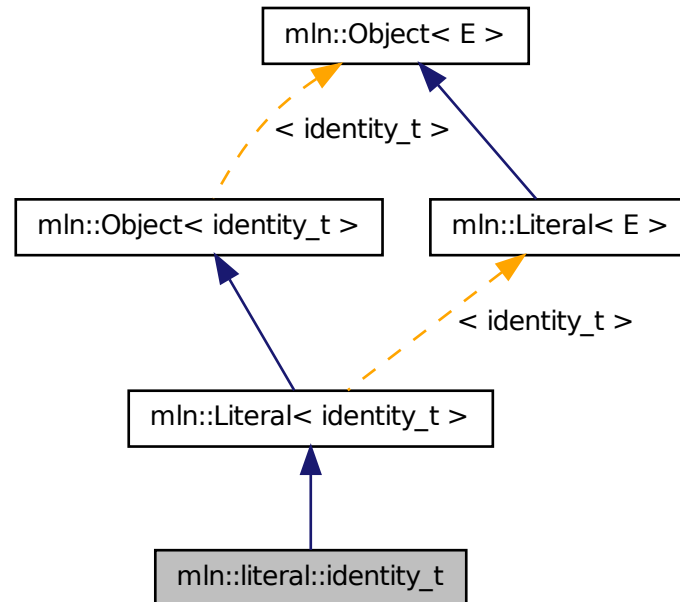
Definition at line 48 of file colors.hh.

10.230 mln::literal::identity_t Struct Reference

Type of literal identity.

```
#include <identity.hh>
```

Inheritance diagram for mln::literal::identity_t:



10.230.1 Detailed Description

Type of literal identity.

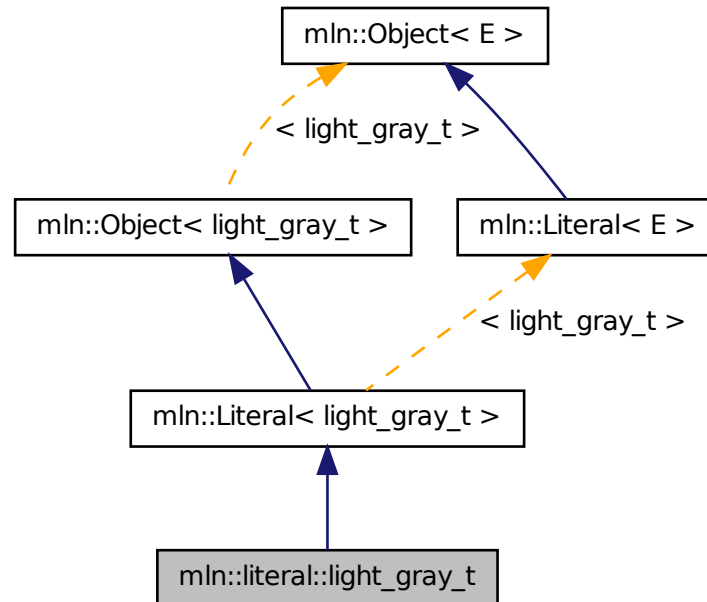
Definition at line 43 of file identity.hh.

10.231 mln::literal::light_gray_t Struct Reference

Type of literal grays.

```
#include <grays.hh>
```

Inheritance diagram for mln::literal::light_gray_t:



10.231.1 Detailed Description

Type of literal grays.

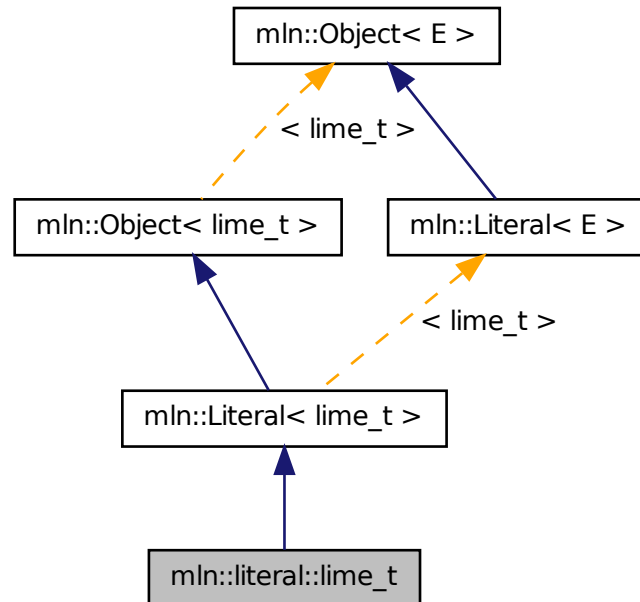
Definition at line 43 of file `grays.hh`.

10.232 mln::literal::lime_t Struct Reference

Type of literal lime.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::lime_t:



10.232.1 Detailed Description

Type of literal lime.

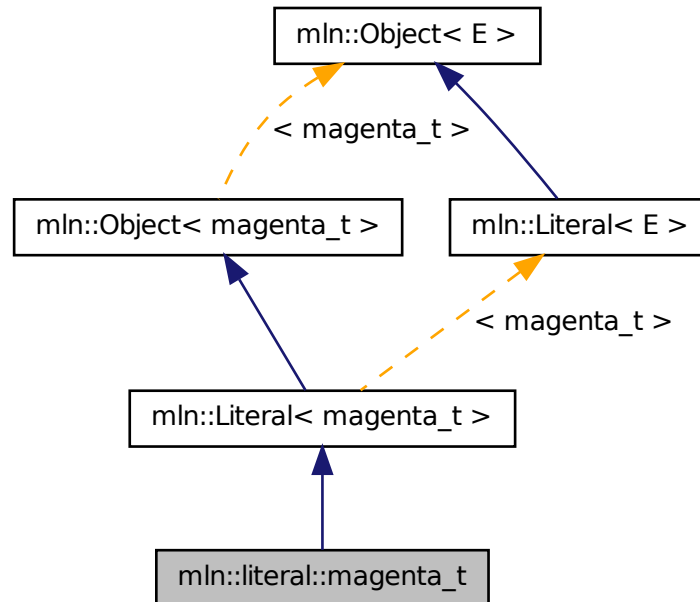
Definition at line 63 of file colors.hh.

10.233 mln::literal::magenta_t Struct Reference

Type of literal magenta.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::magenta_t:



10.233.1 Detailed Description

Type of literal magenta.

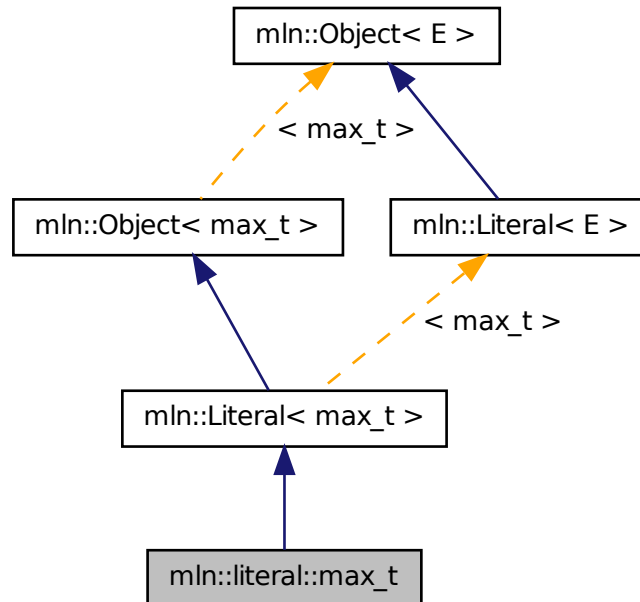
Definition at line 98 of file colors.hh.

10.234 mln::literal::max_t Struct Reference

Type of literal max.

```
#include <max.hh>
```

Inheritance diagram for mln::literal::max_t:



10.234.1 Detailed Description

Type of literal max.

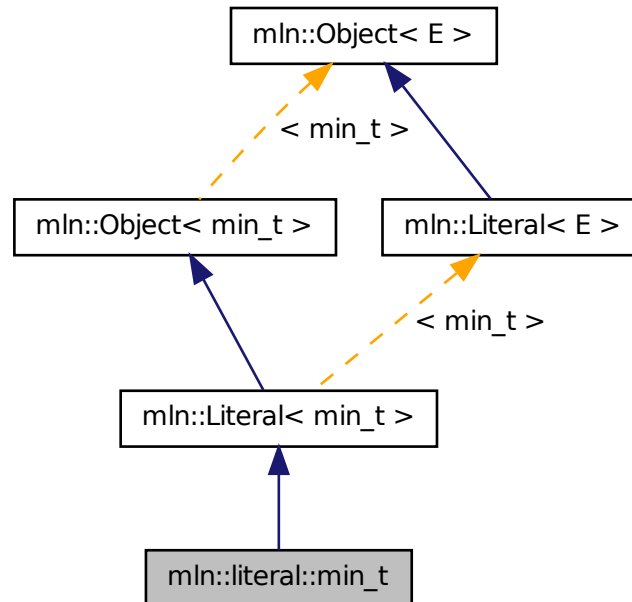
Definition at line 44 of file literal/max.hh.

10.235 mln::literal::min_t Struct Reference

Type of literal min.

```
#include <min.hh>
```

Inheritance diagram for mln::literal::min_t:



10.235.1 Detailed Description

Type of literal min.

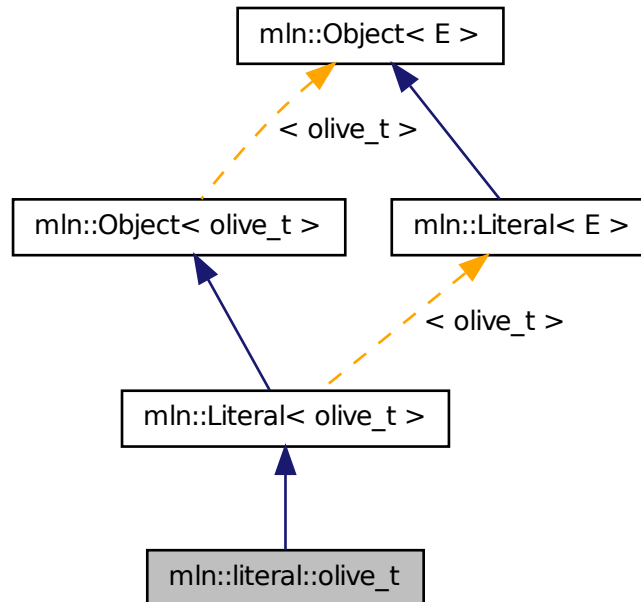
Definition at line 44 of file literal/min.hh.

10.236 mln::literal::olive_t Struct Reference

Type of literal olive.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::olive_t:



10.236.1 Detailed Description

Type of literal olive.

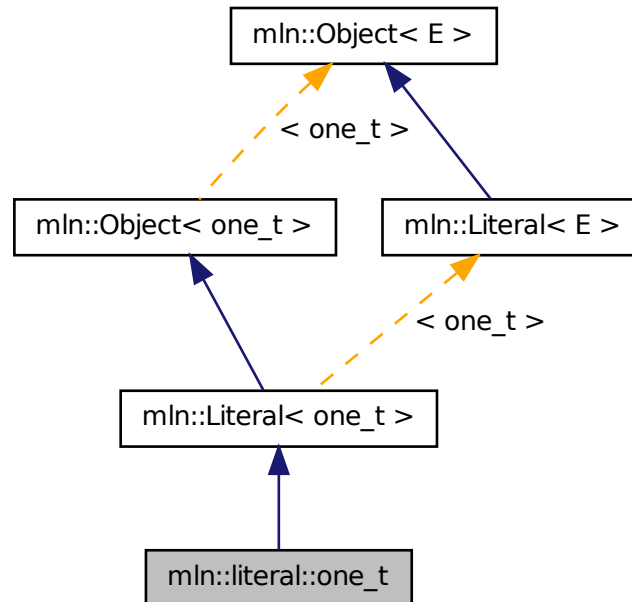
Definition at line 108 of file colors.hh.

10.237 mln::literal::one_t Struct Reference

Type of literal one.

```
#include <one.hh>
```


Inheritance diagram for mln::literal::one_t:



10.237.1 Detailed Description

Type of literal one.

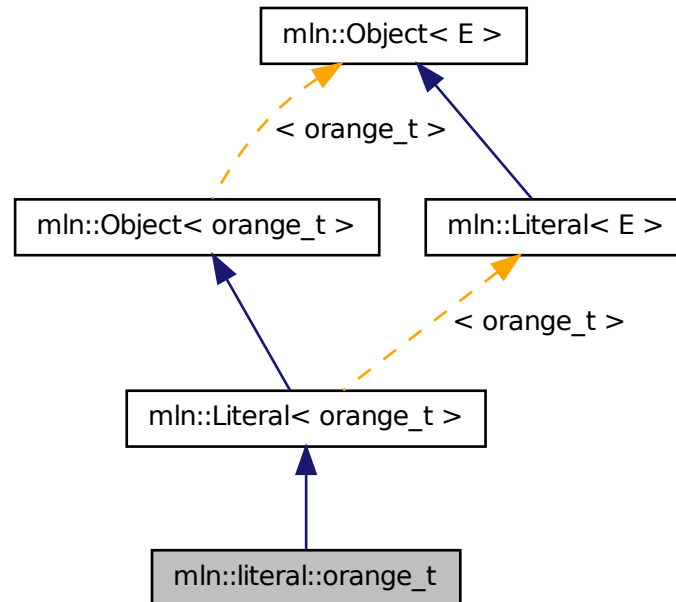
Definition at line 44 of file one.hh.

10.238 mln::literal::orange_t Struct Reference

Type of literal orange.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::orange_t:



10.238.1 Detailed Description

Type of literal orange.

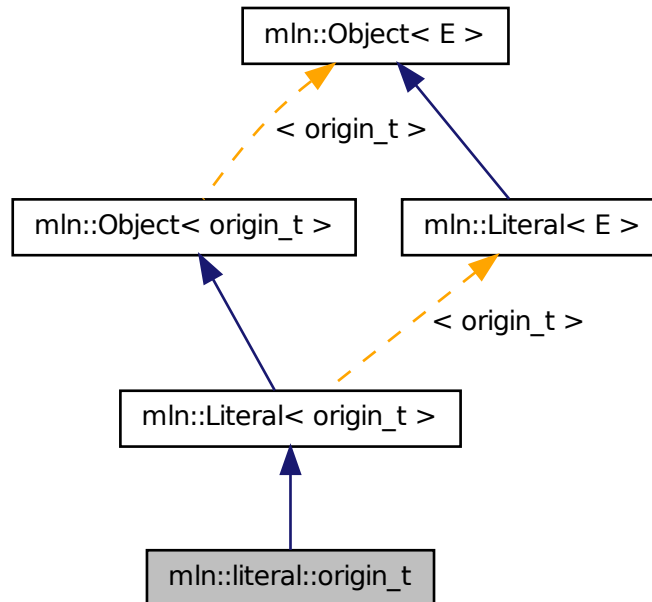
Definition at line 68 of file colors.hh.

10.239 mln::literal::origin_t Struct Reference

Type of literal origin.

```
#include <origin.hh>
```

Inheritance diagram for mln::literal::origin_t:



10.239.1 Detailed Description

Type of literal origin.

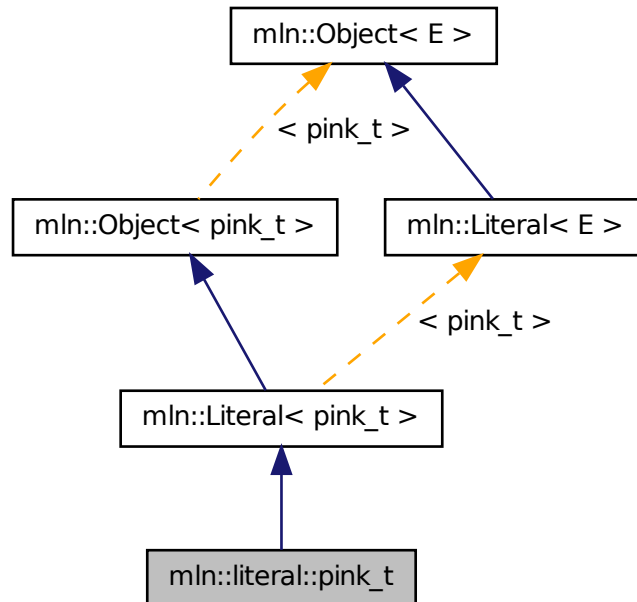
Definition at line 44 of file origin.hh.

10.240 mln::literal::pink_t Struct Reference

Type of literal pink.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::pink_t:



10.240.1 Detailed Description

Type of literal pink.

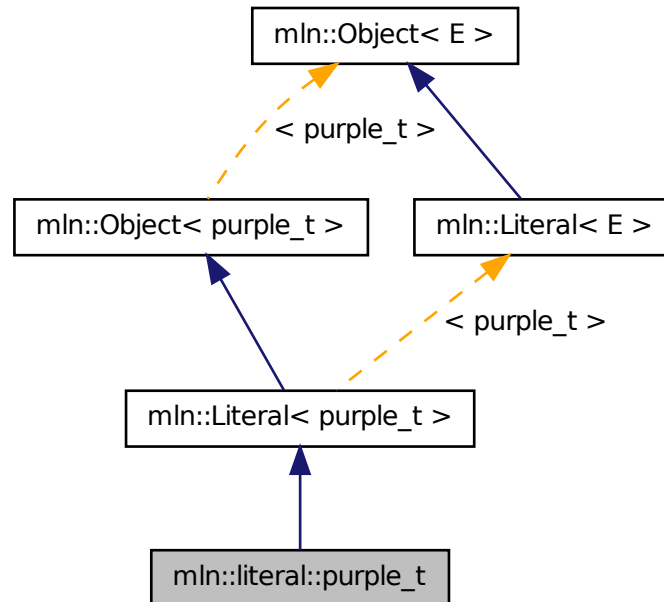
Definition at line 73 of file colors.hh.

10.241 mln::literal::purple_t Struct Reference

Type of literal purple.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::purple_t:



10.241.1 Detailed Description

Type of literal purple.

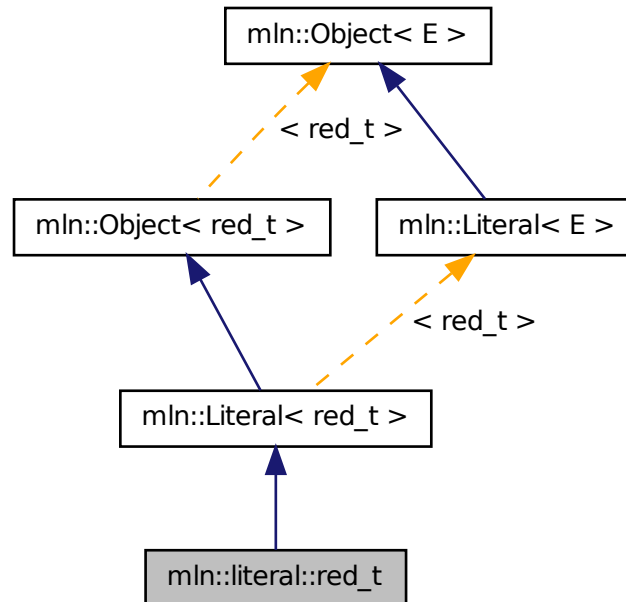
Definition at line 78 of file colors.hh.

10.242 mln::literal::red_t Struct Reference

Type of literal red.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::red_t:



10.242.1 Detailed Description

Type of literal red.

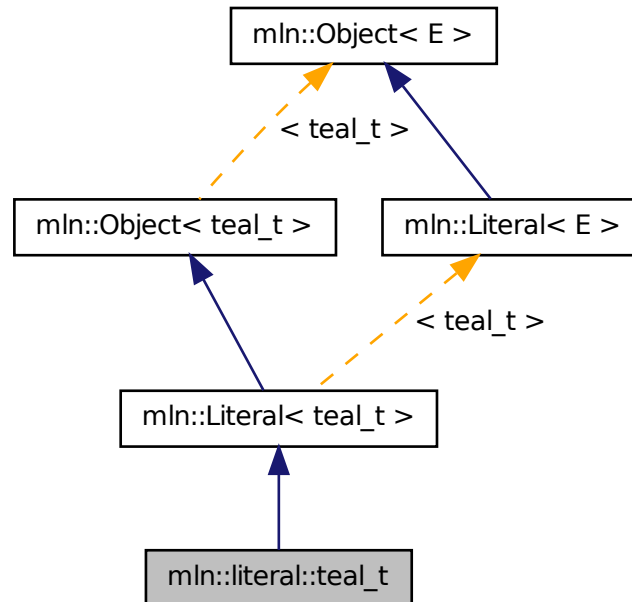
Definition at line 43 of file colors.hh.

10.243 mln::literal::teal_t Struct Reference

Type of literal teal.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::teal_t:



10.243.1 Detailed Description

Type of literal teal.

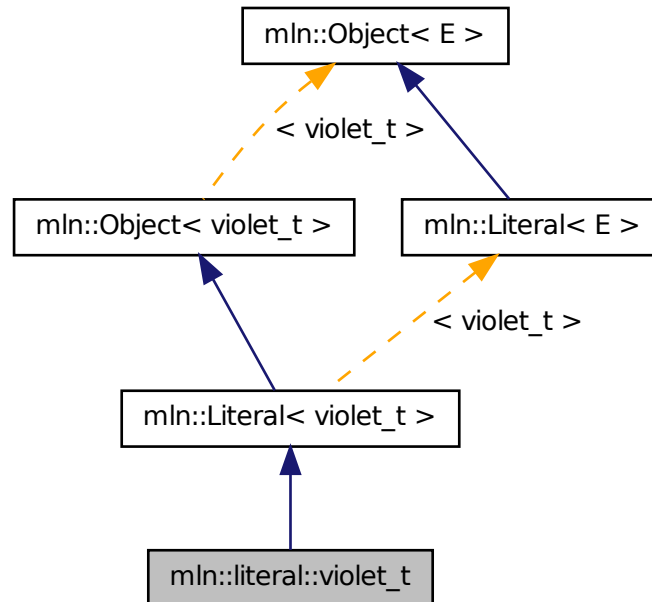
Definition at line 83 of file colors.hh.

10.244 mln::literal::violet_t Struct Reference

Type of literal violet.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::violet_t:



10.244.1 Detailed Description

Type of literal violet.

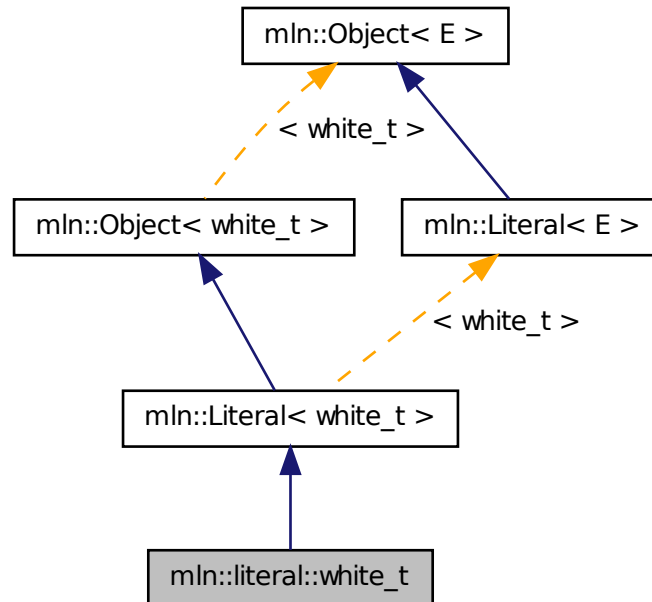
Definition at line 88 of file colors.hh.

10.245 mln::literal::white_t Struct Reference

Type of literal white.

```
#include <white.hh>
```


Inheritance diagram for mln::literal::white_t:



10.245.1 Detailed Description

Type of literal white.

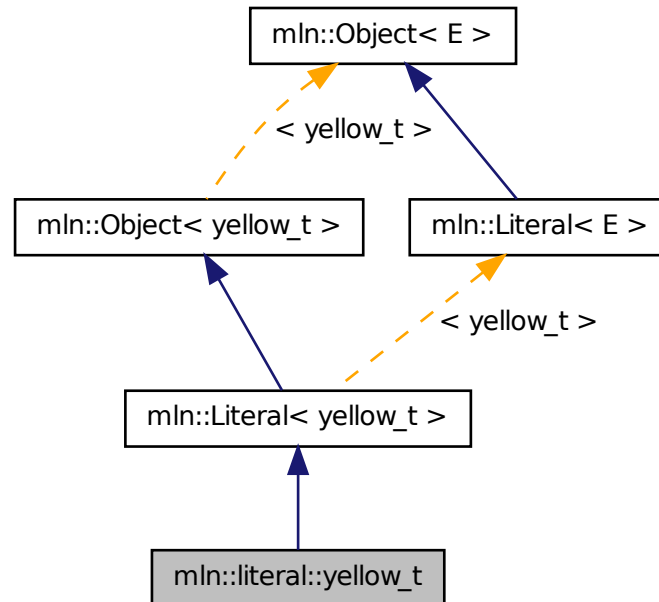
Definition at line 43 of file `white.hh`.

10.246 mln::literal::yellow_t Struct Reference

Type of literal yellow.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::yellow_t:



10.246.1 Detailed Description

Type of literal yellow.

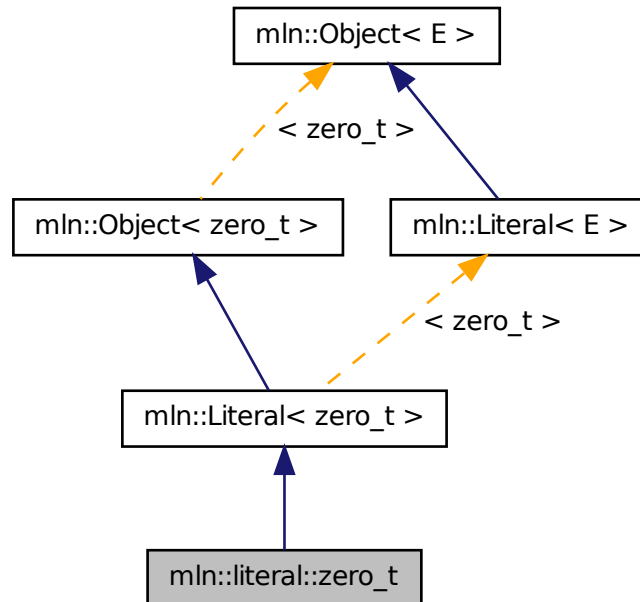
Definition at line 103 of file colors.hh.

10.247 mln::literal::zero_t Struct Reference

Type of literal zero.

```
#include <zero.hh>
```

Inheritance diagram for mln::literal::zero_t:



10.247.1 Detailed Description

Type of literal zero.

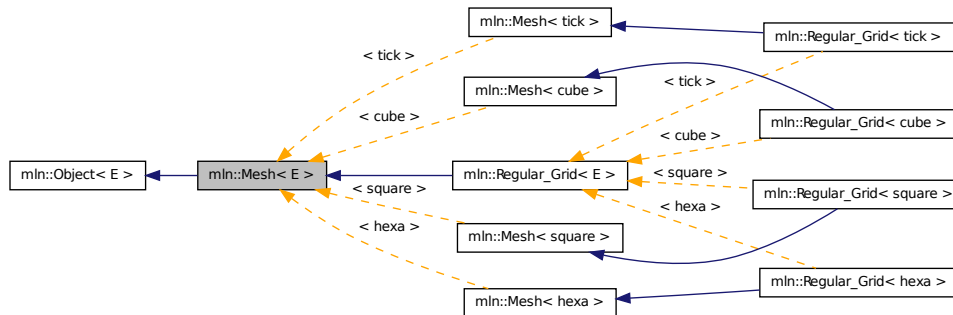
Definition at line 46 of file zero.hh.

10.248 mln::Mesh< E > Struct Template Reference

Base class for implementation classes of meshes.

```
#include <mesh.hh>
```

Inheritance diagram for `mln::Mesh< E >`:



10.248.1 Detailed Description

template<typename E> struct mln::Mesh< E >

Base class for implementation classes of meshes.

See also

`mln::doc::Mesh` for a complete documentation of this class contents.

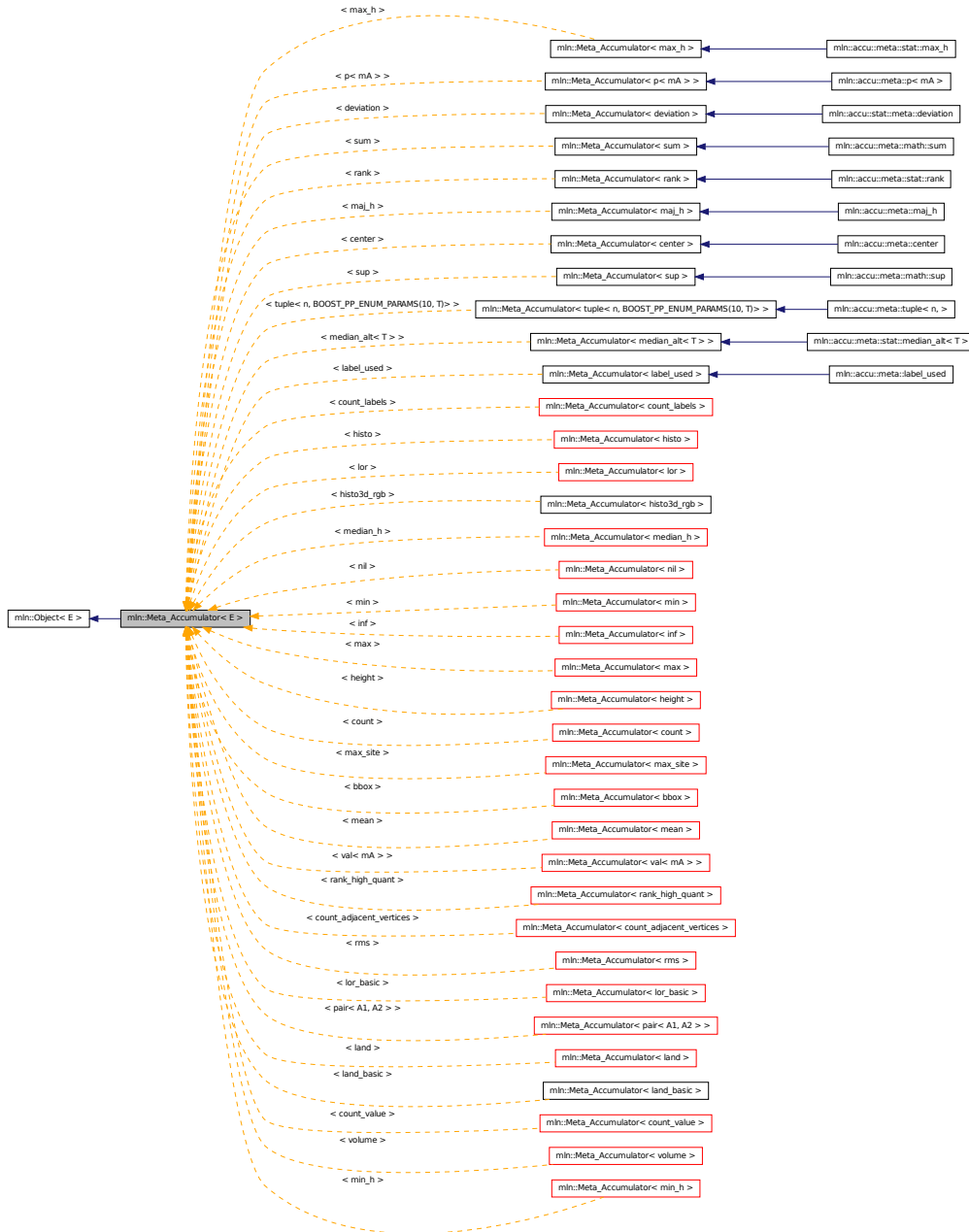
Definition at line 45 of file `mesh.hh`.

10.249 mln::Meta_Accumulator< E > Struct Template Reference

Base class for implementation of meta accumulators.

```
#include <meta_accumulator.hh>
```

Inheritance diagram for mln::Meta_Accumulator< E >:



10.249.1 Detailed Description

`template<typename E> struct mln::Meta_Accumulator< E >`

Base class for implementation of meta accumulators. The parameter *E* is the exact type.

See also

`mln::doc::Meta_Accumulator` for a complete documentation of this class contents.

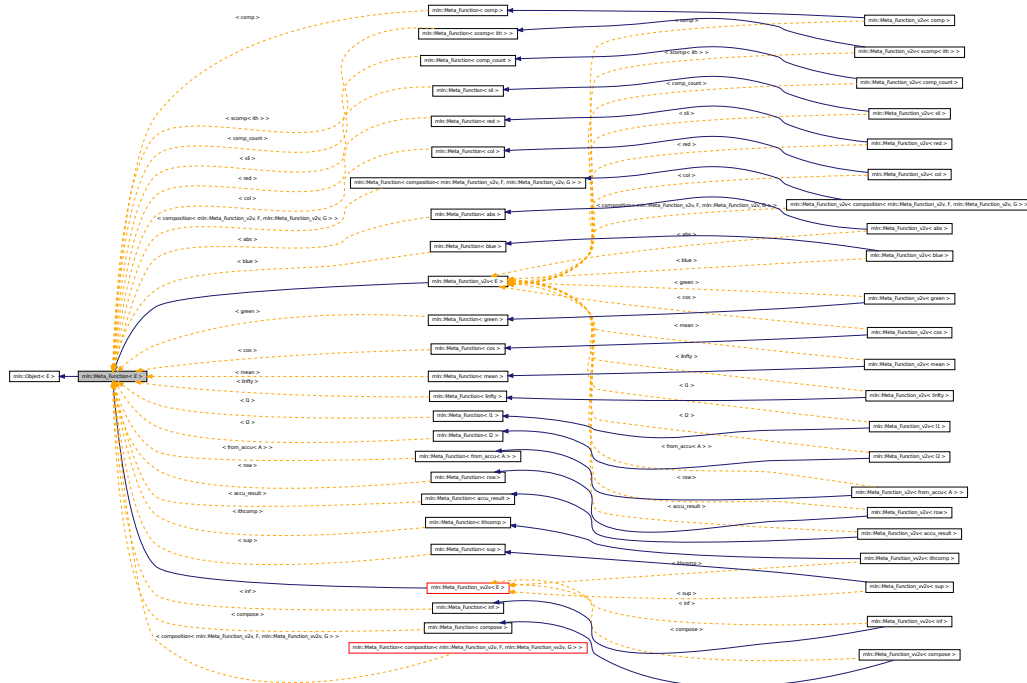
Definition at line 103 of file `meta_accumulator.hh`.

10.250 mln::Meta_Function< E > Struct Template Reference

Base class for implementation of meta functions.

```
#include <meta_function.hh>
```

Inheritance diagram for `mln::Meta_Function< E >`:



10.250.1 Detailed Description

```
template<typename E> struct mln::Meta_Function< E >
```

Base class for implementation of meta functions. The parameter *E* is the exact type.

See also

`mln::doc::Meta_Function` for a complete documentation of this class contents.

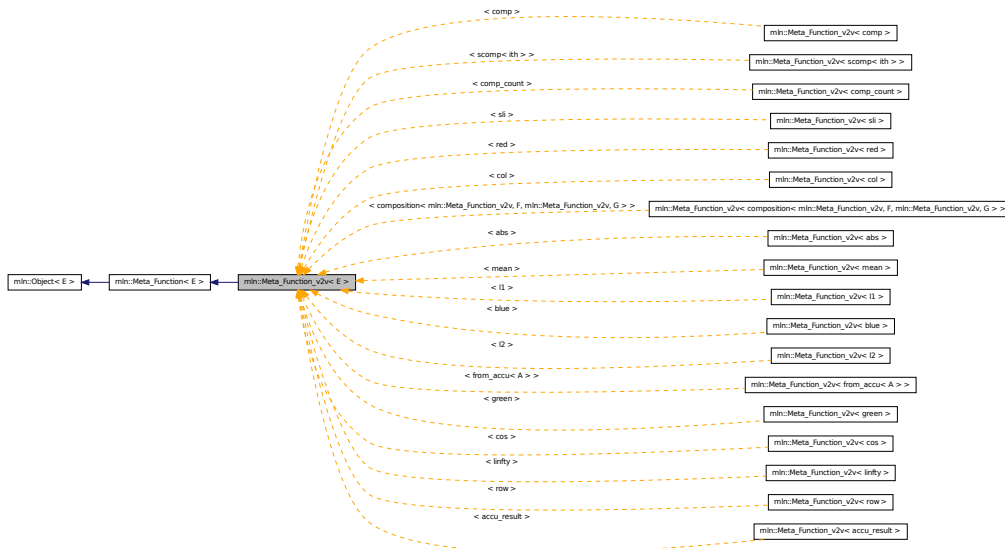
Definition at line 78 of file `meta_function.hh`.

10.251 mln::Meta_Function_v2v< E > Struct Template Reference

Base class for implementation of function-objects from value to value.

```
#include <meta_function.hh>
```

Inheritance diagram for mln::Meta_Function_v2v< E >:



10.251.1 Detailed Description

```
template<typename E> struct mln::Meta_Function_v2v< E >
```

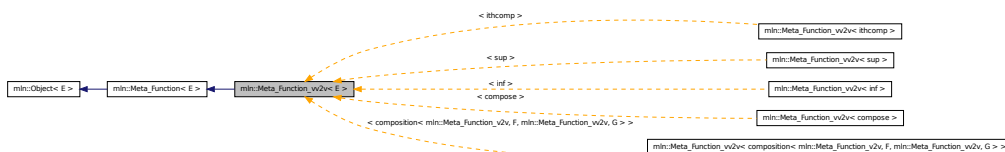
Base class for implementation of function-objects from value to value. The parameter *E* is the exact type. Definition at line 98 of file meta_function.hh.

10.252 mln::Meta_Function_vv2v< E > Struct Template Reference

Base class for implementation of function-objects from value to value.

```
#include <meta_function.hh>
```

Inheritance diagram for mln::Meta_Function_vv2v< E >:



10.252.1 Detailed Description

```
template<typename E> struct mln::Meta_Function_vv2v< E >
```

Base class for implementation of function-objects from value to value. The parameter *E* is the exact type.
Definition at line 120 of file meta_function.hh.

10.253 mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 > Struct Template Reference

Ands type.

```
#include <ands.hh>
```

10.253.1 Detailed Description

```
template<typename E1, typename E2, typename E3, typename E4 = true_, typename E5 = true_,  
typename E6 = true_, typename E7 = true_, typename E8 = true_> struct mln::metal::ands< E1,  
E2, E3, E4, E5, E6, E7, E8 >
```

Ands type.

Definition at line 51 of file ands.hh.

10.254 mln::metal::converts_to< T, U > Struct Template Reference

"converts-to" check.

```
#include <converts_to.hh>
```

Inherited by mln::metal::converts_to< T *, U * >.

10.254.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::converts_to< T, U >
```

"converts-to" check.

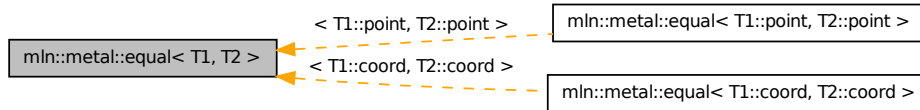
Definition at line 78 of file converts_to.hh.

10.255 mln::metal::equal< T1, T2 > Struct Template Reference

Definition of a static 'equal' test.

```
#include <equal.hh>
```


Inheritance diagram for mln::metal::equal< T1, T2 >:



10.255.1 Detailed Description

```
template<typename T1, typename T2> struct mln::metal::equal< T1, T2 >
```

Definition of a static 'equal' test. Check whether type T1 is exactly type T2.

Definition at line 49 of file equal.hh.

10.256 mln::metal::goes_to< T, U > Struct Template Reference

"goes-to" check.

```
#include <goes_to.hh>
```

10.256.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::goes_to< T, U >
```

"goes-to" check. FIXME: Doc!

Definition at line 53 of file goes_to.hh.

10.257 mln::metal::is< T, U > Struct Template Reference

"is" check.

```
#include <is.hh>
```

10.257.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::is< T, U >
```

"is" check. Check whether T inherits from U.

Definition at line 64 of file is.hh.

10.258 `mln::metal::is_a< T, M >` Struct Template Reference

"is_a" check.

```
#include <is_a.hh>
```

10.258.1 Detailed Description

`template<typename T, template< class > class M> struct mln::metal::is_a< T, M >`

"is_a" check. Check whether T inherits from `_CONCEPT_M`.

Definition at line 95 of file `is_a.hh`.

10.259 `mln::metal::is_not< T, U >` Struct Template Reference

"is_not" check.

```
#include <is_not.hh>
```

10.259.1 Detailed Description

`template<typename T, typename U> struct mln::metal::is_not< T, U >`

"is_not" check. FIXME: Doc!

Definition at line 52 of file `is_not.hh`.

10.260 `mln::metal::is_not_a< T, M >` Struct Template Reference

"is_not_a" static Boolean expression.

```
#include <is_not_a.hh>
```

10.260.1 Detailed Description

`template<typename T, template< class > class M> struct mln::metal::is_not_a< T, M >`

"is_not_a" static Boolean expression.

Definition at line 48 of file `is_not_a.hh`.

10.261 `mln::mixed_neighb< W >` Class Template Reference

Adapter class from window to neighborhood.

```
#include <mixed_neighb.hh>
```

Inherits `neighb_base< W, mixed_neighb< W > >`, and `mlc_is_aW`.

Public Types

- typedef `mixed_neighb_bkd_niter< W >` [bkd_niter](#)
Backward site iterator associated type.
- typedef `mixed_neighb_fwd_niter< W >` [fwd_niter](#)
Forward site iterator associated type.
- typedef `fwd_niter` [niter](#)
Site iterator associated type.

Public Member Functions

- `mixed_neighb ()`
Constructor without argument.
- `mixed_neighb (const W &win)`
*Constructor from a window *win*.*

10.261.1 Detailed Description

`template<typename W> class mln::mixed_neighb< W >`

Adapter class from window to neighborhood.

Definition at line 79 of file `mixed_neighb.hh`.

10.261.2 Member Typedef Documentation

10.261.2.1 `template<typename W> typedef mixed_neighb_bkd_niter<W> mln::mixed_neighb< W >::bkd_niter`

Backward site iterator associated type.

Definition at line 91 of file `mixed_neighb.hh`.

10.261.2.2 `template<typename W> typedef mixed_neighb_fwd_niter<W> mln::mixed_neighb< W >::fwd_niter`

Forward site iterator associated type.

Definition at line 88 of file `mixed_neighb.hh`.

10.261.2.3 `template<typename W> typedef fwd_niter mln::mixed_neighb< W >::niter`

[Site](#) iterator associated type.

Definition at line 94 of file `mixed_neighb.hh`.

10.261.3 Constructor & Destructor Documentation

10.261.3.1 `template<typename W> mln::mixed_neigh<W>::mixed_neigh () [inline]`

Constructor without argument.

Definition at line 158 of file `mixed_neighb.hh`.

10.261.3.2 `template<typename W> mln::mixed_neigh<W>::mixed_neigh (const W & win) [inline]`

Constructor from a window `win`.

Definition at line 164 of file `mixed_neighb.hh`.

10.262 `mln::morpho::attribute::card<I>` Class Template Reference

Cardinality accumulator class.

```
#include <card.hh>
```

Inherits `base<unsigned, card<I>>`.

Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value `t`.
- `void take_n_times (unsigned n, const T &t)`
Take `n` times the value `t`.
- `unsigned to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.262.1 Detailed Description

```
template<typename I> class mln::morpho::attribute::card<I>
```

Cardinality accumulator class.

Definition at line 80 of file `morpho/attribute/card.hh`.

10.262.2 Member Function Documentation

10.262.2.1 `template<typename I> void mln::morpho::attribute::card< I >::init ()`
[inline]

Manipulators.

Definition at line 128 of file morpho/attribute/card.hh.

10.262.2.2 `template<typename I> bool mln::morpho::attribute::card< I >::is_valid () const`
[inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 197 of file morpho/attribute/card.hh.

10.262.2.3 `void mln::Accumulator< card< I > >::take_as_init (const T & t)` **[inherited]**

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.262.2.4 `void mln::Accumulator< card< I > >::take_n_times (unsigned n, const T & t)`
[inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.262.2.5 `template<typename I> unsigned mln::morpho::attribute::card< I >::to_result ()`
const [inline]

Get the value of the accumulator.

Definition at line 189 of file morpho/attribute/card.hh.

10.263 mln::morpho::attribute::count_adjacent_vertices< I > Struct Template Reference

Count_Adjacent_Vertices accumulator class.

```
#include <count_adjacent_vertices.hh>
```

Inherits base< unsigned, count_adjacent_vertices< I > >.

Public Member Functions

- bool `is_valid` () const

Check whether this accu is able to return a result.

- void `take_as_init` (const T &t)
Take as initialization the value t .
- void `take_n_times` (unsigned n, const T &t)
Take n times the value t .
- unsigned `to_result` () const
Get the value of the accumulator.
- void `init` ()
Manipulators.

10.263.1 Detailed Description

template<typename I> struct mln::morpho::attribute::count_adjacent_vertices< I >

Count_Adjacent_Vertices accumulator class. The parameter `I` is the image type on which the accumulator of pixels is built.

Definition at line 83 of file `morpho/attribute/count_adjacent_vertices.hh`.

10.263.2 Member Function Documentation

10.263.2.1 template<typename I > void mln::morpho::attribute::count_adjacent_vertices< I >::init () [inline]

Manipulators.

Definition at line 132 of file `morpho/attribute/count_adjacent_vertices.hh`.

10.263.2.2 template<typename I > bool mln::morpho::attribute::count_adjacent_vertices< I >::is_valid () const [inline]

Check whether this accu is able to return a result.

Definition at line 185 of file `morpho/attribute/count_adjacent_vertices.hh`.

10.263.2.3 void mln::Accumulator< count_adjacent_vertices< I > >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.263.2.4 void mln::Accumulator< count_adjacent_vertices< I > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.263.2.5 `template<typename I > unsigned mln::morpho::attribute::count_adjacent_vertices< I >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 169 of file morpho/attribute/count_adjacent_vertices.hh.

10.264 mln::morpho::attribute::height< I > Struct Template Reference

Height accumulator class.

```
#include <height.hh>
```

Inherits base< unsigned, height< I > >.

Public Member Functions

- unsigned `base_level` () const
Get base & current level of the accumulator.
- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
Take as initialization the value τ .
- void `take_n_times` (unsigned n, const T &t)
Take n times the value τ .
- unsigned `to_result` () const
Get the value of the accumulator.
- void `init` ()
Manipulators.

10.264.1 Detailed Description

`template<typename I > struct mln::morpho::attribute::height< I >`

Height accumulator class. The parameter I is the image type on which the accumulator of pixels is built.

Definition at line 80 of file morpho/attribute/height.hh.

10.264.2 Member Function Documentation

10.264.2.1 `template<typename I > unsigned mln::morpho::attribute::height< I >::base_level () const [inline]`

Get base & current level of the accumulator.

Definition at line 214 of file morpho/attribute/height.hh.

10.264.2.2 `template<typename I > void mln::morpho::attribute::height< I >::init () [inline]`

Manipulators.

Definition at line 131 of file morpho/attribute/height.hh.

10.264.2.3 `template<typename I > bool mln::morpho::attribute::height< I >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 231 of file morpho/attribute/height.hh.

Referenced by `mln::morpho::attribute::height< I >::to_result()`.

10.264.2.4 `void mln::Accumulator< height< I > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.264.2.5 `void mln::Accumulator< height< I > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.264.2.6 `template<typename I > unsigned mln::morpho::attribute::height< I >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 204 of file morpho/attribute/height.hh.

References `mln::morpho::attribute::height< I >::is_valid()`.

10.265 mln::morpho::attribute::sharpness< I > Struct Template Reference

Sharpness accumulator class.


```
#include <sharpness.hh>
Inherits base< double, sharpness< I > >.
```

Public Member Functions

- unsigned [area](#) () const
Give the area of the component.
- unsigned [height](#) () const
Give the height.
- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- double [to_result](#) () const
Get the value of the accumulator.
- unsigned [volume](#) () const
Give the volume of the component.

- void [init](#) ()
Manipulators.

10.265.1 Detailed Description

template<typename I> struct mln::morpho::attribute::sharpness< I >

Sharpness accumulator class. The parameter I is the image type on which the accumulator of pixels is built.

Definition at line 80 of file sharpness.hh.

10.265.2 Member Function Documentation

**10.265.2.1 template<typename I > unsigned mln::morpho::attribute::sharpness< I >::area ()
const [inline]**

Give the area of the component.

Definition at line 190 of file sharpness.hh.

10.265.2.2 `template<typename I > unsigned mln::morpho::attribute::sharpness< I >::height () const [inline]`

Give the height.

Definition at line 206 of file sharpness.hh.

10.265.2.3 `template<typename I > void mln::morpho::attribute::sharpness< I >::init () [inline]`

Manipulators.

Definition at line 134 of file sharpness.hh.

10.265.2.4 `template<typename I > bool mln::morpho::attribute::sharpness< I >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 214 of file sharpness.hh.

10.265.2.5 `void mln::Accumulator< sharpness< I > >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.265.2.6 `void mln::Accumulator< sharpness< I > >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.265.2.7 `template<typename I > double mln::morpho::attribute::sharpness< I >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 175 of file sharpness.hh.

10.265.2.8 `template<typename I > unsigned mln::morpho::attribute::sharpness< I >::volume () const [inline]`

Give the volume of the component.

Definition at line 198 of file sharpness.hh.

10.266 mln::morpho::attribute::sum< I, S > Class Template Reference

Suminality accumulator class.

```
#include <sum.hh>
```

Inherits base< S, sum< I, S > >.

Public Member Functions

- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `set_value` (const argument &v)
Set the return value of the accumulator.
- void `take_as_init` (const T &t)
Take as initialization the value t.
- void `take_n_times` (unsigned n, const T &t)
Take n times the value t.
- S `to_result` () const
Get the value of the accumulator.
- void `untake` (const argument &v)
Untake a value from the accumulator.
- void `init` ()
Manipulators.

10.266.1 Detailed Description

```
template<typename I, typename S = typename mln::value::props< typename I ::value >::sum>  
class mln::morpho::attribute::sum< I, S >
```

Suminality accumulator class.

Definition at line 80 of file morpho/attribute/sum.hh.

10.266.2 Member Function Documentation

10.266.2.1 `template<typename I, typename S > void mln::morpho::attribute::sum< I, S >::init () [inline]`

Manipulators.

Definition at line 137 of file morpho/attribute/sum.hh.

References mln::literal::zero.

10.266.2.2 `template<typename I , typename S > bool mln::morpho::attribute::sum< I, S >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Return always true.

Definition at line 229 of file morpho/attribute/sum.hh.

10.266.2.3 `template<typename I , typename S > void mln::morpho::attribute::sum< I, S >::set_value (const argument & v) [inline]`

Set the return value of the accumulalator.

Definition at line 205 of file morpho/attribute/sum.hh.

10.266.2.4 `void mln::Accumulator< sum< I, S > >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.266.2.5 `void mln::Accumulator< sum< I, S > >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.266.2.6 `template<typename I , typename S > S mln::morpho::attribute::sum< I, S >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 221 of file morpho/attribute/sum.hh.

10.266.2.7 `template<typename I , typename S > void mln::morpho::attribute::sum< I, S >::untake (const argument & v) [inline]`

Untake a value from the accumulator.

Definition at line 189 of file morpho/attribute/sum.hh.

10.267 mln::morpho::attribute::volume< I > Struct Template Reference

Volume accumulator class.

```
#include <volume.hh>
```

Inherits base< unsigned, volume< I > >.

Public Member Functions

- unsigned `area` () const
Give the area.
- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
Take as initialization the value τ .
- void `take_n_times` (unsigned n, const T &t)
Take n times the value τ .
- unsigned `to_result` () const
Get the value of the accumulator.

- void `init` ()
Manipulators.

10.267.1 Detailed Description

`template<typename I> struct mln::morpho::attribute::volume< I >`

Volume accumulator class. The parameter I is the image type on which the accumulator of pixels is built. Definition at line 78 of file morpho/attribute/volume.hh.

10.267.2 Member Function Documentation

10.267.2.1 `template<typename I> unsigned mln::morpho::attribute::volume< I >::area ()`
`const [inline]`

Give the area.

Definition at line 202 of file morpho/attribute/volume.hh.

10.267.2.2 `template<typename I> void mln::morpho::attribute::volume< I >::init ()`
`[inline]`

Manipulators.

Definition at line 130 of file morpho/attribute/volume.hh.

10.267.2.3 `template<typename I> bool mln::morpho::attribute::volume< I >::is_valid ()`
`const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 210 of file morpho/attribute/volume.hh.

10.267.2.4 `void mln::Accumulator< volume< I > >::take_as_init (const T & t)`
[**inherited**]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.267.2.5 `void mln::Accumulator< volume< I > >::take_n_times (unsigned n, const T & t)`
[**inherited**]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.267.2.6 `template<typename I > unsigned mln::morpho::attribute::volume< I >::to_result () const` [**inline**]

Get the value of the accumulator.

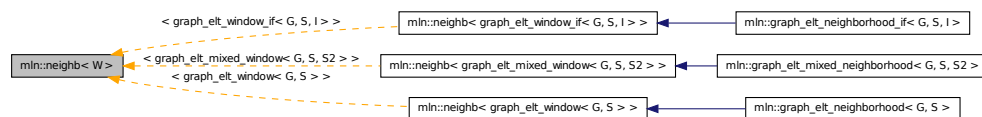
Definition at line 194 of file morpho/attribute/volume.hh.

10.268 mln::neighb< W > Class Template Reference

Adapter class from window to neighborhood.

```
#include <neighb.hh>
```

Inheritance diagram for `mln::neighb< W >`:



Public Types

- typedef `neighb_bkd_niter< W >` [bkd_niter](#)
Backward site iterator associated type.
- typedef `neighb_fwd_niter< W >` [fwd_niter](#)
Forward site iterator associated type.
- typedef `fwd_niter` [niter](#)
Site iterator associated type.

Public Member Functions

- [neighb](#) ()
Constructor without argument.
- [neighb](#) (const W &win)
Constructor from a window win.

10.268.1 Detailed Description

`template<typename W> class mln::neighb< W >`

Adapter class from window to neighborhood.

Definition at line 76 of file mln/core/neighb.hh.

10.268.2 Member Typedef Documentation

10.268.2.1 `template<typename W> typedef neighb_bkd_niter<W> mln::neighb< W >::bkd_niter`

Backward site iterator associated type.

Definition at line 87 of file mln/core/neighb.hh.

10.268.2.2 `template<typename W> typedef neighb_fwd_niter<W> mln::neighb< W >::fwd_niter`

Forward site iterator associated type.

Definition at line 84 of file mln/core/neighb.hh.

10.268.2.3 `template<typename W> typedef fwd_niter mln::neighb< W >::niter`

[Site](#) iterator associated type.

Definition at line 90 of file mln/core/neighb.hh.

10.268.3 Constructor & Destructor Documentation

10.268.3.1 `template<typename W > mln::neighb< W >::neighb () [inline]`

Constructor without argument.

Definition at line 150 of file mln/core/neighb.hh.

10.268.3.2 `template<typename W> mln::neighb< W >::neighb (const W & win) [inline]`

Constructor from a window win.

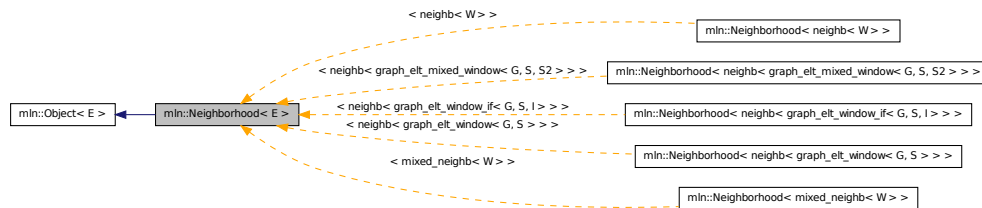
Definition at line 156 of file mln/core/neighb.hh.

10.269 mln::Neighborhood< E > Struct Template Reference

Base class for implementation classes that are neighborhoods.

```
#include <neighborhood.hh>
```

Inheritance diagram for mln::Neighborhood< E >:



10.269.1 Detailed Description

```
template<typename E> struct mln::Neighborhood< E >
```

Base class for implementation classes that are neighborhoods.

See also

[mln::doc::Neighborhood](#) for a complete documentation of this class contents.

Definition at line 66 of file core/concept/neighborhood.hh.

10.270 mln::Neighborhood< void > Struct Template Reference

[Neighborhood](#) category flag type.

```
#include <neighborhood.hh>
```

10.270.1 Detailed Description

```
template<> struct mln::Neighborhood< void >
```

[Neighborhood](#) category flag type.

Definition at line 54 of file core/concept/neighborhood.hh.

10.271 mln::Object< E > Struct Template Reference

Base class for almost every class defined in Milena.

```
#include <object.hh>
```


Inherited by mln::Base< E >, mln::Browsing< E >, mln::Delta_Point_Site< E >, mln::Function< E >, mln::Gdpoint< E >, mln::Graph< E >, mln::Image< E >, mln::io::off::internal::off_loader< I, E >, mln::io::off::internal::off_saver< I, E >, mln::Iterator< E >, mln::Literal< E >, mln::Mesh< E >, mln::Meta_Accumulator< E >, mln::Meta_Function< E >, mln::Neighborhood< E >, mln::Point_Site< E >, mln::Proxy< E >, mln::Site< E >, mln::Site_Set< E >, mln::Value< E >, mln::value::HSL< E >, mln::Value_Set< E >, mln::Weighted_Window< E >, and mln::Window< E >.

10.271.1 Detailed Description

`template<typename E> struct mln::Object< E >`

Base class for almost every class defined in Milena. The parameter *E* is the exact type.

Definition at line 171 of file object.hh.

10.272 mln::p2p_image< I, F > Struct Template Reference

FIXME: Doc!

```
#include <p2p_image.hh>
```

Inherits image_domain_morpher< I, I::domain_t, p2p_image< I, F > >.

Public Types

- typedef `p2p_image< tag::image_< I >, tag::function_< F > >` `skeleton`
Skeleton.

Public Member Functions

- `const I::domain_t & domain () const`
Give the definition domain.
- `const F & fun () const`
Give the p2p function.
- `I::rvalue operator() (const typename I::psite &p) const`
Read-only access to the image value located at point p.
- `internal::morpher_lvalue_< I >::ret operator() (const typename I::psite &p)`
Read-write access to the image value located at point p.
- `p2p_image (I &ima, const F &f)`
*Constructor from an image *ima* and a predicate *f*.*
- `p2p_image ()`
Constructor without argument.

10.272.1 Detailed Description

`template<typename I, typename F> struct mln::p2p_image< I, F >`

FIXME: Doc!

Definition at line 90 of file p2p_image.hh.

10.272.2 Member Typedef Documentation

10.272.2.1 `template<typename I, typename F> typedef p2p_image< tag::image_<I>, tag::function_<F> > mln::p2p_image< I, F >::skeleton`

Skeleton.

Definition at line 95 of file p2p_image.hh.

10.272.3 Constructor & Destructor Documentation

10.272.3.1 `template<typename I, typename F > mln::p2p_image< I, F >::p2p_image ()`
`[inline]`

Constructor without argument.

Definition at line 178 of file p2p_image.hh.

10.272.3.2 `template<typename I, typename F > mln::p2p_image< I, F >::p2p_image (I & ima, const F & f)`
`[inline]`

Constructor from an image *ima* and a predicate *f*.

Definition at line 184 of file p2p_image.hh.

10.272.4 Member Function Documentation

10.272.4.1 `template<typename I, typename F > const I::domain_t & mln::p2p_image< I, F >::domain () const`
`[inline]`

Give the definition domain.

Definition at line 201 of file p2p_image.hh.

10.272.4.2 `template<typename I, typename F > const F & mln::p2p_image< I, F >::fun () const`
`[inline]`

Give the p2p function.

Definition at line 210 of file p2p_image.hh.

10.272.4.3 `template<typename I, typename F > I::rvalue mln::p2p_image< I, F >::operator() (const typename I::psite & p) const [inline]`

Read-only access to the image value located at point `p`.

Definition at line 219 of file `p2p_image.hh`.

10.272.4.4 `template<typename I, typename F > internal::morpher_lvalue< I >::ret mln::p2p_image< I, F >::operator() (const typename I::psite & p) [inline]`

Read-write access to the image value located at point `p`.

Definition at line 229 of file `p2p_image.hh`.

10.273 mln::p_array< P > Class Template Reference

Multi-set of sites.

```
#include <p_array.hh>
```

Inherits `site_set_base< P, p_array< P >>`.

Public Types

- typedef `p_indexed_bkd_piter< self_ > bkd_piter`
Backward Site_Iterator associated type.
- typedef `P element`
Element associated type.
- typedef `p_indexed_fwd_piter< self_ > fwd_piter`
Forward Site_Iterator associated type.
- typedef `P i_element`
Insertion element associated type.
- typedef `fwd_piter piter`
Site_Iterator associated type.
- typedef `p_indexed_psite< self_ > psite`
Psite associated type.

Public Member Functions

- `p_array< P > & append (const P &p)`
Append a point `p`.
- `p_array< P > & append (const p_array< P > &other)`
Append an array `other` of points.

- void **change** (const **psite** &p, const P &new_p)
Change site p into new_p.
- void **clear** ()
Clear this set.
- bool **has** (const util::index &i) const
Test is index i belongs to this site set.
- bool **has** (const **psite** &p) const
Test is p belongs to this site set.
- void **insert** (const P &p)
Insert a point p (equivalent as 'append').
- bool **is_valid** () const
Test this set validity so returns always true.
- std::size_t **memory_size** () const
Return the size of this site set in memory.
- unsigned **nsites** () const
Give the number of sites.
- const P & **operator[]** (const util::index &i) const
Return the i-th element.
- P & **operator[]** (unsigned i)
Return the i-th site (mutable).
- const P & **operator[]** (unsigned i) const
Return the i-th site (constant).
- **p_array** ()
Constructor.
- **p_array** (const std::vector< P > &vect)
Constructor from a vector vect.
- void **reserve** (size_type n)
Reserve n cells.
- void **resize** (size_t size)
Update the size of this array.
- const std::vector< P > & **std_vector** () const
Return the corresponding std::vector of points.

10.273.1 Detailed Description

template<typename P> class mln::p_array< P >

Multi-set of sites. [Site](#) set class based on std::vector.

Definition at line 84 of file p_array.hh.

10.273.2 Member Typedef Documentation

10.273.2.1 template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_array< P >::bkd_piter

Backward [Site_Iterator](#) associated type.

Definition at line 100 of file p_array.hh.

10.273.2.2 template<typename P> typedef P mln::p_array< P >::element

Element associated type.

Definition at line 91 of file p_array.hh.

10.273.2.3 template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_array< P >::fwd_piter

Forward [Site_Iterator](#) associated type.

Definition at line 97 of file p_array.hh.

10.273.2.4 template<typename P> typedef P mln::p_array< P >::i_element

Insertion element associated type.

Definition at line 141 of file p_array.hh.

10.273.2.5 template<typename P> typedef fwd_piter mln::p_array< P >::piter

[Site_Iterator](#) associated type.

Definition at line 103 of file p_array.hh.

10.273.2.6 template<typename P> typedef p_indexed_psite<self_> mln::p_array< P >::psite

Psite associated type.

Definition at line 94 of file p_array.hh.

10.273.3 Constructor & Destructor Documentation

10.273.3.1 `template<typename P> mln::p_array<P>::p_array() [inline]`

Constructor.

Definition at line 340 of file p_array.hh.

10.273.3.2 `template<typename P> mln::p_array<P>::p_array(const std::vector<P> & vect) [inline]`

Constructor from a vector `vect`.

Definition at line 346 of file p_array.hh.

10.273.4 Member Function Documentation

10.273.4.1 `template<typename P> p_array<P> & mln::p_array<P>::append(const P & p) [inline]`

Append a point `p`.

Definition at line 408 of file p_array.hh.

Referenced by `mln::convert::to_p_array()`.

10.273.4.2 `template<typename P> p_array<P> & mln::p_array<P>::append(const p_array<P> & other) [inline]`

Append an array `other` of points.

Definition at line 425 of file p_array.hh.

References `mln::p_array<P>::std_vector()`.

10.273.4.3 `template<typename P> void mln::p_array<P>::change(const psite & p, const P & new_p) [inline]`

Change site `p` into `new_p`.

Definition at line 472 of file p_array.hh.

References `mln::p_array<P>::has()`.

10.273.4.4 `template<typename P> void mln::p_array<P>::clear() [inline]`

Clear this set.

Definition at line 436 of file p_array.hh.

10.273.4.5 `template<typename P> bool mln::p_array<P>::has(const psite & p) const [inline]`

Test if `p` belongs to this site set.

Definition at line 362 of file p_array.hh.

Referenced by mln::p_array< P >::change(), and mln::p_array< P >::operator[]().

10.273.4.6 `template<typename P > bool mln::p_array< P >::has (const util::index & i) const [inline]`

Test is index *i* belongs to this site set.

Definition at line 375 of file p_array.hh.

References mln::p_array< P >::nsites().

10.273.4.7 `template<typename P > void mln::p_array< P >::insert (const P & p) [inline]`

Insert a point *p* (equivalent as 'append').

Definition at line 417 of file p_array.hh.

10.273.4.8 `template<typename P > bool mln::p_array< P >::is_valid () const [inline]`

Test this set validity so returns always true.

Definition at line 383 of file p_array.hh.

10.273.4.9 `template<typename P > std::size_t mln::p_array< P >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 481 of file p_array.hh.

References mln::p_array< P >::nsites().

10.273.4.10 `template<typename P > unsigned mln::p_array< P >::nsites () const [inline]`

Give the number of sites.

Definition at line 400 of file p_array.hh.

Referenced by mln::registration::get_rot(), mln::p_array< P >::has(), mln::p_array< P >::memory_size(), and mln::p_array< P >::operator[]().

10.273.4.11 `template<typename P > P & mln::p_array< P >::operator[] (unsigned i) [inline]`

Return the *i*-th site (mutable).

Definition at line 463 of file p_array.hh.

References mln::p_array< P >::nsites().

10.273.4.12 `template<typename P> const P & mln::p_array< P >::operator[] (const util::index & i) const [inline]`

Return the *i*-th element.

Definition at line 391 of file `p_array.hh`.

References `mln::p_array< P >::has()`.

10.273.4.13 `template<typename P> const P & mln::p_array< P >::operator[] (unsigned i) const [inline]`

Return the *i*-th site (constant).

Definition at line 454 of file `p_array.hh`.

References `mln::p_array< P >::nsites()`.

10.273.4.14 `template<typename P> void mln::p_array< P >::reserve (size_type n) [inline]`

Reserve *n* cells.

Definition at line 354 of file `p_array.hh`.

Referenced by `mln::convert::to_p_array()`.

10.273.4.15 `template<typename P> void mln::p_array< P >::resize (size_t size) [inline]`

Update the size of this array.

Definition at line 445 of file `p_array.hh`.

10.273.4.16 `template<typename P> const std::vector< P > & mln::p_array< P >::std_vector () const [inline]`

Return the corresponding `std::vector` of points.

Definition at line 489 of file `p_array.hh`.

Referenced by `mln::p_array< P >::append()`.

10.274 mln::p_centered< W > Class Template Reference

[Site](#) set corresponding to a window centered on a site.

```
#include <p_centered.hh>
```

Inherits `site_set_base< W::psite, p_centered< W >>`, and `mlc_is_aW`.

Public Types

- `typedef p_centered_piter< W > bkd_piter`
Backward [Site Iterator](#) associated type.

- typedef [psite element](#)
Element associated type.
- typedef [p_centered_piter](#)< W > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef [fwd_piter piter](#)
[Site_Iterator](#) associated type.
- typedef W::psite [psite](#)
Psite associated type.
- typedef W::site [site](#)
[Site](#) associated type.

Public Member Functions

- const W::psite & [center](#) () const
Give the center of this site set.
- template<typename P >
bool [has](#) (const P &p) const
*Test if *p* belongs to the box.*
- bool [is_valid](#) () const
Test if this site set is initialized.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- [p_centered](#) (const W &win, const typename W::psite &c)
*Constructor from a window *win* and a center *c*.*
- [p_centered](#) ()
Constructor without argument.
- const W & [window](#) () const
Give the window this site set is defined upon.

10.274.1 Detailed Description

`template<typename W> class mln::p_centered< W >`

[Site](#) set corresponding to a window centered on a site.

Definition at line 77 of file `p_centered.hh`.

10.274.2 Member Typedef Documentation

10.274.2.1 `template<typename W> typedef p_centered_piter<W> mln::p_centered< W >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 97 of file `p_centered.hh`.

10.274.2.2 `template<typename W> typedef psite mln::p_centered< W >::element`

Element associated type.

Definition at line 90 of file `p_centered.hh`.

10.274.2.3 `template<typename W> typedef p_centered_piter<W> mln::p_centered< W >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 94 of file `p_centered.hh`.

10.274.2.4 `template<typename W> typedef fwd_piter mln::p_centered< W >::piter`

[Site_Iterator](#) associated type.

Definition at line 100 of file `p_centered.hh`.

10.274.2.5 `template<typename W> typedef W ::psite mln::p_centered< W >::psite`

Psite associated type.

Definition at line 83 of file `p_centered.hh`.

10.274.2.6 `template<typename W> typedef W ::site mln::p_centered< W >::site`

[Site](#) associated type.

Definition at line 86 of file `p_centered.hh`.

10.274.3 Constructor & Destructor Documentation

10.274.3.1 `template<typename W > mln::p_centered< W >::p_centered () [inline]`

Constructor without argument.

Definition at line 182 of file `p_centered.hh`.

10.274.3.2 `template<typename W > mln::p_centered< W >::p_centered (const W & win, const typename W::psite & c) [inline]`

Constructor from a window `win` and a center `c`.

Definition at line 188 of file p_centered.hh.

References mln::p_centered< W >::is_valid().

10.274.4 Member Function Documentation

10.274.4.1 `template<typename W > const W::psite & mln::p_centered< W >::center () const [inline]`

Give the center of this site set.

Definition at line 216 of file p_centered.hh.

10.274.4.2 `template<typename W > template<typename P > bool mln::p_centered< W >::has (const P & p) const [inline]`

Test if *p* belongs to the box.

Definition at line 199 of file p_centered.hh.

References mln::p_centered< W >::is_valid().

10.274.4.3 `template<typename W > bool mln::p_centered< W >::is_valid () const [inline]`

Test if this site set is initialized.

Definition at line 175 of file p_centered.hh.

Referenced by mln::p_centered< W >::has(), and mln::p_centered< W >::p_centered().

10.274.4.4 `template<typename W > std::size_t mln::p_centered< W >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 208 of file p_centered.hh.

10.274.4.5 `template<typename W > const W & mln::p_centered< W >::window () const [inline]`

Give the window this site set is defined upon.

Definition at line 224 of file p_centered.hh.

10.275 mln::p_complex< D, G > Class Template Reference

A complex psite set based on the *N*-faces of a complex of dimension *D* (a *D*-complex).

```
#include <p_complex.hh>
```

Inherits site_set_base_< complex_psite< D, G >, p_complex< D, G > >.

Public Types

- typedef `super_::site element`
Associated types.
- typedef `complex_psite< D, G > psite`
Point_Site associated type.
- typedef `p_complex_fwd_piter_< D, G > fwd_piter`
Forward Site_Iterator associated type.
- typedef `p_complex_bkd_piter_< D, G > bkd_piter`
Backward Site_Iterator associated type.
- typedef `fwd_piter piter`
Site_Iterator associated type.

Public Member Functions

- bool `has (const psite &p) const`
Does this site set has p?
- bool `is_valid () const`
Is this site set valid?
- unsigned `nfaces () const`
Return the number of faces in the complex.
- unsigned `nfaces_of_dim (unsigned n) const`
Return the number of n-faces in the complex.
- unsigned `nsites () const`
Return The number of sites of the set, i.e., the number of faces.
- `p_complex (const topo::complex< D > &cplx, const G &geom)`
Construct a complex psite set from a complex.
- `topo::complex< D > & cplx () const`
Accessors.
- `topo::complex< D > & cplx ()`
Return the complex associated to the p_complex domain (mutable version).
- `const G & geom () const`
Return the geometry of the complex.

10.275.1 Detailed Description

template<unsigned D, typename G> class mln::p_complex< D, G >

A complex psite set based on the N-faces of a complex of dimension D (a `D-complex`).

Template Parameters

D The dimension of the complex.

G A function object type, associating localization information (geometry) to each face of the complex.

See also

[mln::geom::complex_geometry](#). A complex [psite set](#) based on the N-faces of a complex.

Definition at line 116 of file `p_complex.hh`.

10.275.2 Member Typedef Documentation

**10.275.2.1 template<unsigned D, typename G> typedef p_complex_bkd_piter_<D, G>
mln::p_complex< D, G >::bkd_piter**

Backward [Site_Iterator](#) associated type.

Definition at line 141 of file `p_complex.hh`.

**10.275.2.2 template<unsigned D, typename G> typedef super_::site mln::p_complex< D, G
>::element**

Associated types.

Element associated type.

Definition at line 132 of file `p_complex.hh`.

**10.275.2.3 template<unsigned D, typename G> typedef p_complex_fwd_piter_<D, G>
mln::p_complex< D, G >::fwd_piter**

Forward [Site_Iterator](#) associated type.

Definition at line 138 of file `p_complex.hh`.

**10.275.2.4 template<unsigned D, typename G> typedef fwd_piter mln::p_complex< D, G
>::piter**

[Site_Iterator](#) associated type.

Definition at line 144 of file `p_complex.hh`.

**10.275.2.5 template<unsigned D, typename G> typedef complex_psite<D, G> mln::p_complex<
D, G >::psite**

[Point_Site](#) associated type.

Definition at line 135 of file p_complex.hh.

10.275.3 Constructor & Destructor Documentation

10.275.3.1 `template<unsigned D, typename G > mln::p_complex< D, G >::p_complex (const topo::complex< D > & cplx, const G & geom) [inline]`

Construct a complex psite set from a complex.

Parameters

cplx The complex upon which the complex psite set is built.

geom FIXME

Definition at line 231 of file p_complex.hh.

10.275.4 Member Function Documentation

10.275.4.1 `template<unsigned D, typename G > topo::complex< D > & mln::p_complex< D, G >::cplx () const`

Accessors.

Return the complex associated to the [p_complex](#) domain (const version)

Definition at line 293 of file p_complex.hh.

References `mln::p_complex< D, G >::is_valid()`.

Referenced by `mln::complex_psite< D, G >::change_target()`, `mln::complex_psite< D, G >::complex_psite()`, and `mln::operator==()`.

10.275.4.2 `template<unsigned D, typename G > topo::complex< D > & mln::p_complex< D, G >::cplx ()`

Return the complex associated to the [p_complex](#) domain (mutable version).

Definition at line 301 of file p_complex.hh.

References `mln::p_complex< D, G >::is_valid()`.

10.275.4.3 `template<unsigned D, typename G > const G & mln::p_complex< D, G >::geom () const`

Return the geometry of the complex.

Definition at line 309 of file p_complex.hh.

10.275.4.4 `template<unsigned D, typename G > bool mln::p_complex< D, G >::has (const psite & p) const [inline]`

Does this site set has *p*?

Definition at line 271 of file p_complex.hh.

References `mln::complex_psite< D, G >::is_valid()`, `mln::p_complex< D, G >::is_valid()`, and `mln::complex_psite< D, G >::site_set()`.

10.275.4.5 `template<unsigned D, typename G > bool mln::p_complex< D, G >::is_valid ()`
`const [inline]`

Is this site set valid?

Definition at line 263 of file `p_complex.hh`.

Referenced by `mln::p_complex< D, G >::cplx()`, and `mln::p_complex< D, G >::has()`.

10.275.4.6 `template<unsigned D, typename G > unsigned mln::p_complex< D, G >::nfaces ()`
`const [inline]`

Return the number of faces in the complex.

Definition at line 247 of file `p_complex.hh`.

Referenced by `mln::p_complex< D, G >::nsites()`.

10.275.4.7 `template<unsigned D, typename G > unsigned mln::p_complex< D, G >::nfaces_of_dim (unsigned n)`
`const [inline]`

Return the number of *n-faces* in the complex.

Definition at line 255 of file `p_complex.hh`.

10.275.4.8 `template<unsigned D, typename G > unsigned mln::p_complex< D, G >::nsites ()`
`const [inline]`

Return The number of sites of the set, i.e., the number of *faces*.

(Required by the `mln::Site_Set` concept, since the property `trait::site_set::nsites::known` of this site set is set to 'known'.)

Definition at line 239 of file `p_complex.hh`.

References `mln::p_complex< D, G >::nfaces()`.

10.276 `mln::p_edges< G, F >` Class Template Reference

[Site](#) set mapping graph edges and image sites.

```
#include <p_edges.hh>
```

Inherits `site_set_base< F::result, p_edges< G, F > >`.

Public Types

- typedef `util::edge< G > edge`

Type of graph edge.

- typedef F `fun_t`
Function associated type.
- typedef `util::edge< G >` `graph_element`
Type of graph element this site set focuses on.
- typedef G `graph_t`
Graph associated type.

- typedef `super_::site` `element`
Associated types.
- typedef `p_edges_psite< G, F >` `psite`
Point_Site associated type.
- typedef `p_graph_piter< self_, mln_edge_fwd_iter(G) >` `fwd_piter`
Forward Site_Iterator associated type.
- typedef `p_graph_piter< self_, mln_edge_bkd_iter(G) >` `bkd_piter`
Backward Site_Iterator associated type.
- typedef `fwd_piter` `piter`
Site_Iterator associated type.

Public Member Functions

- bool `has` (const `psite` &p) const
Does this site set has site p?
- template<typename G2 >
bool `has` (const `util::edge< G2 >` &e) const
Does this site set has edge e?
- void `invalidate` ()
Invalidate this site set.
- bool `is_valid` () const
Is this site set valid?
- `std::size_t` `memory_size` () const
Does this site set has vertex_id? FIXME: causes ambiguities while calling has(mln::neighb_fwd_niter<>); bool has(unsigned vertex_id) const;.
- unsigned `nedges` () const
Return The number of edges in the graph.
- unsigned `nsites` () const
Return The number of points (sites) of the set, i.e., the number of edges.

- [p_edges](#) ()
Constructors
Default constructor.
- [p_edges](#) (const [Graph](#)< G > &gr)
Construct a graph edge psite set from a graph.
- [p_edges](#) (const [Graph](#)< G > &gr, const [Function](#)< F > &f)
Construct a graph edge psite set from a graph and a function.
- `template<typename F2 >`
[p_edges](#) (const [Graph](#)< G > &gr, const [Function](#)< F2 > &f)
Construct a graph edge psite set from a graph and a function.
- const G & [graph](#) () const
Accessors.
- const F & [function](#) () const
Return the mapping function.

10.276.1 Detailed Description

`template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> class mln::p_edges< G, F >`

[Site](#) set mapping graph edges and image sites.

Definition at line 70 of file p_edges.hh.

10.276.2 Member Typedef Documentation

10.276.2.1 `template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> typedef p_graph_piter< self_, mln_edge_bkd_iter(G) > mln::p_edges< G, F >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 128 of file p_edges.hh.

10.276.2.2 `template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> typedef util::edge<G> mln::p_edges< G, F >::edge`

Type of graph edge.

Definition at line 86 of file p_edges.hh.

10.276.2.3 `template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> typedef super_::site mln::p_edges< G, F >::element`

Associated types.

Element associated type.

Definition at line 119 of file p_edges.hh.

10.276.2.4 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>>
typedef F mln::p_edges< G, F >::fun_t`

[Function](#) associated type.

Definition at line 83 of file p_edges.hh.

10.276.2.5 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>>
>> typedef p_graph_piter< self_, mln_edge_fwd_iter(G) > mln::p_edges< G, F
>::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 125 of file p_edges.hh.

10.276.2.6 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>>
typedef util::edge<G> mln::p_edges< G, F >::graph_element`

Type of graph element this site set focuses on.

Definition at line 89 of file p_edges.hh.

10.276.2.7 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>>
typedef G mln::p_edges< G, F >::graph_t`

[Graph](#) associated type.

Definition at line 80 of file p_edges.hh.

10.276.2.8 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>>
typedef fwd_piter mln::p_edges< G, F >::piter`

[Site_Iterator](#) associated type.

Definition at line 131 of file p_edges.hh.

10.276.2.9 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>>
typedef p_edges_psite<G, F> mln::p_edges< G, F >::psite`

[Point_Site](#) associated type.

Definition at line 122 of file p_edges.hh.

10.276.3 Constructor & Destructor Documentation

10.276.3.1 `template<typename G , typename F > mln::p_edges< G, F >::p_edges ()`
[inline]

Constructors

Default constructor.

Definition at line 203 of file p_edges.hh.

10.276.3.2 `template<typename G , typename F > mln::p_edges< G, F >::p_edges (const`
Graph< G > & gr) **[inline]**

Construct a graph edge psite set from a graph.

Parameters

gr The graph upon which the graph edge psite set is built.

Definition at line 209 of file p_edges.hh.

References mln::p_edges< G, F >::is_valid().

10.276.3.3 `template<typename G , typename F > mln::p_edges< G, F >::p_edges (const`
Graph< G > & gr, const Function< F > & f) **[inline]**

Construct a graph edge psite set from a graph and a function.

Parameters

gr The graph upon which the graph edge psite set is built.

f the function mapping edges and sites.

Definition at line 221 of file p_edges.hh.

References mln::p_edges< G, F >::is_valid().

10.276.3.4 `template<typename G , typename F > template<typename F2 > mln::p_edges< G, F`
>::p_edges (const Graph< G > & gr, const Function< F2 > & f) **[inline]**

Construct a graph edge psite set from a graph and a function.

Parameters

gr The graph upon which the graph edge psite set is built.

f the function mapping edges and sites. It must be convertible towards the function type F.

Definition at line 231 of file p_edges.hh.

References mln::p_edges< G, F >::is_valid().

10.276.4 Member Function Documentation

10.276.4.1 `template<typename G , typename F > const F & mln::p_edges< G, F >::function () const [inline]`

Return the mapping function.

Definition at line 324 of file p_edges.hh.

10.276.4.2 `template<typename G , typename F > const G & mln::p_edges< G, F >::graph () const [inline]`

Accessors.

Return the graph associated to this site set

Definition at line 315 of file p_edges.hh.

References `mln::p_edges< G, F >::is_valid()`.

Referenced by `mln::operator==()`.

10.276.4.3 `template<typename G , typename F > bool mln::p_edges< G, F >::has (const psite & p) const [inline]`

Does this site set has site *p*?

Definition at line 276 of file p_edges.hh.

References `mln::p_edges< G, F >::is_valid()`.

10.276.4.4 `template<typename G , typename F > template<typename G2 > bool mln::p_edges< G, F >::has (const util::edge< G2 > & e) const [inline]`

Does this site set has edge *e*?

Definition at line 286 of file p_edges.hh.

References `mln::util::edge< G >::graph()`, `mln::util::edge< G >::is_valid()`, and `mln::p_edges< G, F >::is_valid()`.

10.276.4.5 `template<typename G , typename F > void mln::p_edges< G, F >::invalidate () [inline]`

Invalidate this site set.

Definition at line 268 of file p_edges.hh.

10.276.4.6 `template<typename G , typename F > bool mln::p_edges< G, F >::is_valid () const [inline]`

Is this site set valid?

Definition at line 260 of file p_edges.hh.

Referenced by mln::p_edges< G, F >::graph(), mln::p_edges< G, F >::has(), and mln::p_edges< G, F >::p_edges().

10.276.4.7 `template<typename G , typename F > std::size_t mln::p_edges< G, F >::memory_size () const [inline]`

Does this site set has *vertex_id*? FIXME: causes ambiguities while calling has(mln::neighb_fwd_niter<>);
bool has(unsigned vertex_id) const;.

Definition at line 305 of file p_edges.hh.

10.276.4.8 `template<typename G , typename F > unsigned mln::p_edges< G, F >::nedges () const [inline]`

Return The number of edges in the graph.

Definition at line 252 of file p_edges.hh.

Referenced by mln::p_edges< G, F >::nsites().

10.276.4.9 `template<typename G , typename F > unsigned mln::p_edges< G, F >::nsites () const [inline]`

Return The number of points (sites) of the set, i.e., the number of *edges*.

Definition at line 244 of file p_edges.hh.

References mln::p_edges< G, F >::nedges().

10.277 mln::p_faces< N, D, P > Struct Template Reference

A complex psite set based on a the N-faces of a complex of dimension D (a D-complex).

```
#include <p_faces.hh>
```

Inherits site_set_base_< faces_psite< N, D, P >, p_faces< N, D, P > >.

Public Types

- typedef super_::site [element](#)
Associated types.
- typedef [faces_psite](#)< N, D, P > [psite](#)
Point_Site associated type.
- typedef [p_faces_fwd_piter](#)< N, D, P > [fwd_piter](#)
Forward Site_Iterator associated type.
- typedef [p_faces_bkd_piter](#)< N, D, P > [bkd_piter](#)
Backward Site_Iterator associated type.
- typedef [fwd_piter](#) [piter](#)
Site_Iterator associated type.

Public Member Functions

- `bool is_valid () const`
Is this site set valid?
- `unsigned nfaces () const`
Return The number of faces in the complex.
- `unsigned nsites () const`
Return The number of sites of the set, i.e., the number of faces.
- `p_faces (const topo::complex< D > &cplx)`
Construct a faces psite set from an mln::complex.
- `p_faces (const p_complex< D, P > &pc)`
Construct a faces psite set from an mln::p_complex.
- `topo::complex< D > & cplx () const`
Accessors.
- `topo::complex< D > & cplx ()`
Return the complex associated to the p_faces domain (mutable version).

10.277.1 Detailed Description

`template<unsigned N, unsigned D, typename P> struct mln::p_faces< N, D, P >`

A complex psite set based on a the N-faces of a complex of dimension D (a D-complex).

Definition at line 77 of file p_faces.hh.

10.277.2 Member Typedef Documentation

10.277.2.1 `template<unsigned N, unsigned D, typename P> typedef p_faces_bkd_piter_<N, D, P> mln::p_faces< N, D, P >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 110 of file p_faces.hh.

10.277.2.2 `template<unsigned N, unsigned D, typename P> typedef super_::site mln::p_faces< N, D, P >::element`

Associated types.

Element associated type.

Definition at line 99 of file p_faces.hh.

10.277.2.3 `template<unsigned N, unsigned D, typename P> typedef p_faces_fwd_piter_<N, D, P> mln::p_faces< N, D, P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 106 of file p_faces.hh.

10.277.2.4 `template<unsigned N, unsigned D, typename P> typedef fwd_piter mln::p_faces< N, D, P >::piter`

[Site_Iterator](#) associated type.

Definition at line 113 of file p_faces.hh.

10.277.2.5 `template<unsigned N, unsigned D, typename P> typedef faces_psite<N, D, P> mln::p_faces< N, D, P >::psite`

[Point_Site](#) associated type.

Definition at line 102 of file p_faces.hh.

10.277.3 Constructor & Destructor Documentation

10.277.3.1 `template<unsigned N, unsigned D, typename P > mln::p_faces< N, D, P >::p_faces (const topo::complex< D > & cplx) [inline]`

Construct a faces psite set from an mln::complex.

Parameters

cplx The complex upon which the complex psite set is built.

Definition at line 192 of file p_faces.hh.

10.277.3.2 `template<unsigned N, unsigned D, typename P > mln::p_faces< N, D, P >::p_faces (const p_complex< D, P > & pc) [inline]`

Construct a faces psite set from an mln::p_complex.

Parameters

pc The complex upon which the complex psite set is built.

Definition at line 201 of file p_faces.hh.

10.277.4 Member Function Documentation

10.277.4.1 `template<unsigned N, unsigned D, typename P > topo::complex< D > & mln::p_faces< N, D, P >::cplx () const`

Accessors.

Return the complex associated to the [p_faces](#) domain (const version).

Definition at line 257 of file `p_faces.hh`.

References `mln::p_faces< N, D, P >::is_valid()`.

Referenced by `mln::faces_psite< N, D, P >::change_target()`, and `mln::operator==()`.

10.277.4.2 `template<unsigned N, unsigned D, typename P > topo::complex< D > & mln::p_faces< N, D, P >::cplx ()`

Return the complex associated to the [p_faces](#) domain (mutable version).

Definition at line 265 of file `p_faces.hh`.

References `mln::p_faces< N, D, P >::is_valid()`.

10.277.4.3 `template<unsigned N, unsigned D, typename P > bool mln::p_faces< N, D, P >::is_valid () const [inline]`

Is this site set valid?

Definition at line 227 of file `p_faces.hh`.

Referenced by `mln::p_faces< N, D, P >::cplx()`.

10.277.4.4 `template<unsigned N, unsigned D, typename P > unsigned mln::p_faces< N, D, P >::nfaces () const [inline]`

Return The number of faces in the complex.

Definition at line 219 of file `p_faces.hh`.

Referenced by `mln::p_faces< N, D, P >::nsites()`.

10.277.4.5 `template<unsigned N, unsigned D, typename P > unsigned mln::p_faces< N, D, P >::nsites () const [inline]`

Return The number of sites of the set, i.e., the number of *faces*.

(Required by the [mln::Site_Set](#) concept, since the property trait::site_set::nsites::known of this site set is set to 'known'.)

Definition at line 211 of file `p_faces.hh`.

References `mln::p_faces< N, D, P >::nfaces()`.

10.278 `mln::p_graph_piter< S, I > Class Template Reference`

Generic iterator on point sites of a `mln::S`.

```
#include <p_graph_piter.hh>
```

Inherits `site_set_iterator_base< S, p_graph_piter< S, I > >`.

Public Member Functions

- `const S::graph_t & graph () const`
Return the graph associated to the target S.
 - `unsigned id () const`
Return the graph element id.
 - `mln_q_subject (iter) element()`
Return the underlying graph element.
 - `void next ()`
Go to the next element.
-
- `p_graph_piter ()`
Constructors.

10.278.1 Detailed Description

`template<typename S, typename I> class mln::p_graph_piter< S, I >`

Generic iterator on point sites of a mln::S.

Definition at line 55 of file p_graph_piter.hh.

10.278.2 Constructor & Destructor Documentation

10.278.2.1 `template<typename S, typename I> mln::p_graph_piter< S, I >::p_graph_piter ()`
[inline]

Constructors.

Definition at line 151 of file p_graph_piter.hh.

10.278.3 Member Function Documentation

10.278.3.1 `template<typename S, typename I> const S::graph_t & mln::p_graph_piter< S, I >::graph () const` **[inline]**

Return the graph associated to the target S.

Definition at line 212 of file p_graph_piter.hh.

10.278.3.2 `template<typename S, typename I> unsigned mln::p_graph_piter< S, I >::id () const` **[inline]**

Return the graph element id.

Definition at line 228 of file p_graph_piter.hh.

10.278.3.3 `template<typename S , typename I > mln::p_graph_piter< S, I >::mln_q_subject (iter)`

Return the underlying graph element.

10.278.3.4 `void mln::Site_Iterator< p_graph_piter< S, I > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.279 `mln::p_if< S, F > Class Template Reference`

[Site](#) set restricted w.r.t.

```
#include <p_if.hh>
```

Inherits `site_set_base_< S::psite, p_if< S, F > >`.

Public Types

- `typedef p_if_piter_< typename S::bkd_piter, S, F > bkd_piter`
Backward [Site_Iterator](#) associated type.
- `typedef S::element element`
Element associated type.
- `typedef p_if_piter_< typename S::fwd_piter, S, F > fwd_piter`
Forward [Site_Iterator](#) associated type.
- `typedef fwd_piter piter`
[Site_Iterator](#) associated type.
- `typedef S::psite psite`
Psite associated type.

Public Member Functions

- `bool has (const psite &p) const`
*Test if *p* belongs to the subset.*
- `bool is_valid () const`

Test if this site set is valid.

- `std::size_t memory_size () const`
Return the size of this site set in memory.
- `const S & overset () const`
Give the primary overset.
- `p_if ()`
Constructor without argument.
- `p_if (const S &s, const F &f)`
*Constructor with a site set *s* and a predicate *f*.*
- `bool pred (const psite &p) const`
*Test predicate on point site *p*.*
- `const F & predicate () const`
Give the predicate function.

10.279.1 Detailed Description

`template<typename S, typename F> class mln::p_if< S, F >`

[Site](#) set restricted w.r.t. a predicate.

Parameter *S* is a site set type; parameter *F* is a function from point to Boolean.

Definition at line 83 of file `p_if.hh`.

10.279.2 Member Typedef Documentation

10.279.2.1 `template<typename S, typename F> typedef p_if_piter_<typename S ::bkd_piter, S, F> mln::p_if< S, F >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 100 of file `p_if.hh`.

10.279.2.2 `template<typename S, typename F> typedef S ::element mln::p_if< S, F >::element`

Element associated type.

Definition at line 90 of file `p_if.hh`.

10.279.2.3 `template<typename S, typename F> typedef p_if_piter_<typename S ::fwd_piter, S, F> mln::p_if< S, F >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 97 of file `p_if.hh`.

10.279.2.4 `template<typename S, typename F> typedef fwd_piter mln::p_if< S, F >::piter`

[Site_Iterator](#) associated type.

Definition at line 103 of file p_if.hh.

10.279.2.5 `template<typename S, typename F> typedef S ::psite mln::p_if< S, F >::psite`

Psite associated type.

Definition at line 94 of file p_if.hh.

10.279.3 Constructor & Destructor Documentation**10.279.3.1** `template<typename S, typename F > mln::p_if< S, F >::p_if (const S & s, const F & f) [inline]`

Constructor with a site set *s* and a predicate *f*.

Definition at line 190 of file p_if.hh.

10.279.3.2 `template<typename S, typename F > mln::p_if< S, F >::p_if () [inline]`

Constructor without argument.

Definition at line 198 of file p_if.hh.

10.279.4 Member Function Documentation**10.279.4.1** `template<typename S, typename F > bool mln::p_if< S, F >::has (const psite & p) const [inline]`

Test if *p* belongs to the subset.

Definition at line 159 of file p_if.hh.

References `mln::p_if< S, F >::has()`.

Referenced by `mln::p_if< S, F >::has()`.

10.279.4.2 `template<typename S, typename F > bool mln::p_if< S, F >::is_valid () const [inline]`

Test if this site set is valid.

Definition at line 167 of file p_if.hh.

10.279.4.3 `template<typename S, typename F > std::size_t mln::p_if< S, F >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 213 of file p_if.hh.

10.279.4.4 `template<typename S, typename F > const S & mln::p_if< S, F >::overset () const [inline]`

Give the primary overset.

Definition at line 175 of file p_if.hh.

10.279.4.5 `template<typename S, typename F > bool mln::p_if< S, F >::pred (const psite & p) const [inline]`

Test predicate on point site p.

Definition at line 183 of file p_if.hh.

10.279.4.6 `template<typename S, typename F > const F & mln::p_if< S, F >::predicate () const [inline]`

Give the predicate function.

Definition at line 205 of file p_if.hh.

10.280 mln::p_image< I > Class Template Reference

[Site](#) set based on an image of Booleans.

```
#include <p_image.hh>
```

Inherits `site_set_base_< I::psite, p_image< I > >`.

Public Types

- `typedef S::bkd_piter bkd_piter`
Backward [Site_Iterator](#) associated type.
- `typedef I::psite element`
Element associated type.
- `typedef S::fwd_piter fwd_piter`
Forward [Site_Iterator](#) associated type.
- `typedef psite i_element`
Insertion element associated type.
- `typedef S::piter piter`
[Site_Iterator](#) associated type.
- `typedef I::psite psite`
Psite associated type.
- `typedef psite r_element`
Removal element associated type.

- typedef `internal::p_image_site_set< I >::ret S`
Equivalent `site_set` type.

Public Member Functions

- void `clear ()`
Clear this set.
- bool `has (const psite &) const`
Test if the psite `p` belongs to this site set.
- void `insert (const psite &p)`
Insert a site `p`.
- bool `is_valid () const`
Test if this site set is valid, i.e., initialized.
- `std::size_t memory_size () const`
Return the size of this site set in memory.
- unsigned `nsites () const`
Give the number of sites.
- operator `typename internal::p_image_site_set< I >::ret () const`
Conversion towards the equivalent site set.
- `p_image ()`
Constructor without argument.
- `p_image (const I &ima)`
Constructor.
- void `remove (const psite &p)`
Remove a site `p`.
- void `toggle (const psite &p)`
Change the status in/out of a site `p`.

10.280.1 Detailed Description

`template<typename I> class mln::p_image< I >`

[Site](#) set based on an image of Booleans.

Definition at line 88 of file `p_image.hh`.

10.280.2 Member Typedef Documentation

10.280.2.1 `template<typename I > typedef S ::bkd_piter mln::p_image< I >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 110 of file p_image.hh.

10.280.2.2 `template<typename I > typedef I ::psite mln::p_image< I >::element`

Element associated type.

Definition at line 100 of file p_image.hh.

10.280.2.3 `template<typename I > typedef S ::fwd_piter mln::p_image< I >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 107 of file p_image.hh.

10.280.2.4 `template<typename I > typedef psite mln::p_image< I >::i_element`

Insertion element associated type.

Definition at line 136 of file p_image.hh.

10.280.2.5 `template<typename I > typedef S ::piter mln::p_image< I >::piter`

[Site_Iterator](#) associated type.

Definition at line 113 of file p_image.hh.

10.280.2.6 `template<typename I > typedef I ::psite mln::p_image< I >::psite`

Psite associated type.

Definition at line 104 of file p_image.hh.

10.280.2.7 `template<typename I > typedef psite mln::p_image< I >::r_element`

Removal element associated type.

Definition at line 142 of file p_image.hh.

10.280.2.8 `template<typename I > typedef internal::p_image_site_set<I>::ret mln::p_image< I >::S`

Equivalent site_set type.

Definition at line 93 of file p_image.hh.

10.280.3 Constructor & Destructor Documentation

10.280.3.1 `template<typename I> mln::p_image<I>::p_image() [inline]`

Constructor without argument.

Definition at line 182 of file `p_image.hh`.

10.280.3.2 `template<typename I> mln::p_image<I>::p_image(const I & ima) [inline]`

Constructor.

Definition at line 189 of file `p_image.hh`.

References `mln::p_image<I>::clear()`.

10.280.4 Member Function Documentation

10.280.4.1 `template<typename I> void mln::p_image<I>::clear() [inline]`

Clear this set.

Definition at line 283 of file `p_image.hh`.

References `mln::data::fill_with_value()`, and `mln::p_image<I>::is_valid()`.

Referenced by `mln::p_image<I>::p_image()`.

10.280.4.2 `template<typename I> bool mln::p_image<I>::has(const psite & p) const [inline]`

Test if the `psite p` belongs to this site set.

Definition at line 199 of file `p_image.hh`.

References `mln::p_image<I>::is_valid()`.

10.280.4.3 `template<typename I> void mln::p_image<I>::insert(const psite & p) [inline]`

Insert a site `p`.

Definition at line 224 of file `p_image.hh`.

References `mln::p_image<I>::is_valid()`.

10.280.4.4 `template<typename I> bool mln::p_image<I>::is_valid() const [inline]`

Test if this site set is valid, i.e., initialized.

Definition at line 208 of file `p_image.hh`.

Referenced by `mln::p_image<I>::clear()`, `mln::p_image<I>::has()`, `mln::p_image<I>::insert()`, `mln::p_image<I>::memory_size()`, `mln::p_image<I>::remove()`, and `mln::p_image<I>::toggle()`.

10.280.4.5 `template<typename I> std::size_t mln::p_image< I >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 273 of file p_image.hh.

References mln::p_image< I >::is_valid().

10.280.4.6 `template<typename I> unsigned mln::p_image< I >::nsites () const [inline]`

Give the number of sites.

Definition at line 216 of file p_image.hh.

10.280.4.7 `template<typename I> mln::p_image< I >::operator typename internal::p_image_site_set< I >::ret () const [inline]`

Conversion towards the equivalent site set.

Definition at line 174 of file p_image.hh.

10.280.4.8 `template<typename I> void mln::p_image< I >::remove (const psite & p) [inline]`

Remove a site p.

Definition at line 237 of file p_image.hh.

References mln::p_image< I >::is_valid().

10.280.4.9 `template<typename I> void mln::p_image< I >::toggle (const psite & p) [inline]`

Change the status in/out of a site p.

Definition at line 251 of file p_image.hh.

References mln::p_image< I >::is_valid().

10.281 mln::p_indexed_bkd_piter< S > Class Template Reference

Backward iterator on sites of an indexed site set.

```
#include <p_array.hh>
```

Inherits site_set_iterator_base< S, p_indexed_bkd_piter< S > >.

Public Member Functions

- `int index () const`

Return the current index.

- void `next()`
Go to the next element.
- `p_indexed_bkd_piter` (const S &s)
Constructor.
- `p_indexed_bkd_piter` ()
Constructor with no argument.

10.281.1 Detailed Description

`template<typename S> class mln::p_indexed_bkd_piter< S >`

Backward iterator on sites of an indexed site set.

Definition at line 276 of file `p_array.hh`.

10.281.2 Constructor & Destructor Documentation

10.281.2.1 `template<typename S > mln::p_indexed_bkd_piter< S >::p_indexed_bkd_piter ()`
[inline]

Constructor with no argument.

Definition at line 686 of file `p_array.hh`.

10.281.2.2 `template<typename S > mln::p_indexed_bkd_piter< S >::p_indexed_bkd_piter (const S & s)` [inline]

Constructor.

Definition at line 692 of file `p_array.hh`.

10.281.3 Member Function Documentation

10.281.3.1 `template<typename S > int mln::p_indexed_bkd_piter< S >::index () const`
[inline]

Return the current index.

Definition at line 733 of file `p_array.hh`.

10.281.3.2 `void mln::Site_Iterator< p_indexed_bkd_piter< S > >::next ()` [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.282 mln::p_indexed_fwd_piter< S > Class Template Reference

Forward iterator on sites of an indexed site set.

```
#include <p_array.hh>
```

Inherits site_set_iterator_base< S, p_indexed_fwd_piter< S > >.

Public Member Functions

- int [index](#) () const
Return the current index.
- void [next](#) ()
Go to the next element.
- [p_indexed_fwd_piter](#) (const S &s)
Constructor.
- [p_indexed_fwd_piter](#) ()
Constructor with no argument.

10.282.1 Detailed Description

```
template<typename S> class mln::p_indexed_fwd_piter< S >
```

Forward iterator on sites of an indexed site set.

Definition at line 235 of file p_array.hh.

10.282.2 Constructor & Destructor Documentation

10.282.2.1 `template<typename S > mln::p_indexed_fwd_piter< S >::p_indexed_fwd_piter ()`
`[inline]`

Constructor with no argument.

Definition at line 629 of file p_array.hh.

10.282.2.2 `template<typename S > mln::p_indexed_fwd_piter< S >::p_indexed_fwd_piter (`
`const S & s) [inline]`

Constructor.

Definition at line 635 of file p_array.hh.

10.282.3 Member Function Documentation

10.282.3.1 `template<typename S> int mln::p_indexed_fwd_piter< S >::index () const`
`[inline]`

Return the current index.

Definition at line 676 of file p_array.hh.

10.282.3.2 `void mln::Site_Iterator< p_indexed_fwd_piter< S > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.283 mln::p_indexed_psite< S > Class Template Reference

Psite class for indexed site sets such as [p_array](#).

```
#include <p_array.hh>
```

Inherits `pseudo_site_base_< const S::element &, p_indexed_psite< S > >`.

10.283.1 Detailed Description

`template<typename S> class mln::p_indexed_psite< S >`

Psite class for indexed site sets such as [p_array](#).

Definition at line 183 of file p_array.hh.

10.284 mln::p_key< K, P > Class Template Reference

Priority queue class.

```
#include <p_key.hh>
```

Inherits `site_set_base_< P, p_key< K, P > >`.

Public Types

- `typedef p_double_piter< self_, mln_bkd_eiter(util::set< K >), typename p_set< P >::bkd_piter >`
`bkd_piter`

Backward [Site_Iterator](#) associated type.

- typedef P [element](#)
Element associated type.
- typedef p_double_piter< [self_](#), mln_fwd_eiter(util::set< K >), typename p_set< P >::fwd_piter > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef std::pair< K, P > [i_element](#)
Insertion element associated type.
- typedef [fwd_piter](#) [piter](#)
[Site_Iterator](#) associated type.
- typedef p_double_psite< [self_](#), p_set< P > > [psite](#)
Psite associated type.
- typedef P [r_element](#)
Removal element associated type.

Public Member Functions

- void [change_key](#) (const K &k, const K &new_k)
*Change the key *k* into a new value *new_k*.*
- template<typename F >
void [change_keys](#) (const [Function_v2v](#)< F > &f)
*Change the keys by applying the function *f*.*
- void [clear](#) ()
Clear this site set.
- bool [exists_key](#) (const K &key) const
*Test if the *priority* exists.*
- bool [has](#) (const [psite](#) &) const
*Test is the *psite p* belongs to this site set.*
- bool [has](#) (const P &p) const
*Test is the *psite p* belongs to this site set.*
- void [insert](#) (const [i_element](#) &k_p)
*Insert a pair *k_p* (key *k*, site *p*).*
- void [insert](#) (const K &k, const P &p)
*Insert a pair (key *k*, site *p*).*
- bool [is_valid](#) () const
Test this set validity so returns always true.

- `const K & key (const P &p) const`
Give the key associated with site `p`.
- `const util::set< K > & keys () const`
Give the set of keys.
- `std::size_t memory_size () const`
Return the size of this site set in memory.
- `unsigned nsites () const`
Give the number of sites.
- `const p_set< P > & operator() (const K &key) const`
Give the queue with the priority `priority`.
- `p_key ()`
Constructor.
- `void remove (const P &p)`
Remove a site `p`.
- `void remove_key (const K &k)`
Remove all sites with key `k`.

10.284.1 Detailed Description

`template<typename K, typename P> class mln::p_key< K, P >`

Priority queue class.

Definition at line 72 of file `p_key.hh`.

10.284.2 Member Typedef Documentation

10.284.2.1 `template<typename K , typename P > typedef p_double_piter<self_, mln_bkd_eiter(util::set<K>), typename p_set<P>::bkd_piter> mln::p_key< K, P >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 93 of file `p_key.hh`.

10.284.2.2 `template<typename K , typename P > typedef P mln::p_key< K, P >::element`

Element associated type.

Definition at line 79 of file `p_key.hh`.

10.284.2.3 `template<typename K , typename P > typedef p_double_piter<self_, mln_fwd_eiter(util::set<K>), typename p_set<P>::fwd_piter> mln::p_key< K, P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 88 of file p_key.hh.

10.284.2.4 `template<typename K , typename P > typedef std::pair<K,P> mln::p_key< K, P >::i_element`

Insertion element associated type.

Definition at line 118 of file p_key.hh.

10.284.2.5 `template<typename K , typename P > typedef fwd_piter mln::p_key< K, P >::piter`

[Site_Iterator](#) associated type.

Definition at line 96 of file p_key.hh.

10.284.2.6 `template<typename K , typename P > typedef p_double_psite< self_, p_set<P> > mln::p_key< K, P >::psite`

Psite associated type.

Definition at line 83 of file p_key.hh.

10.284.2.7 `template<typename K , typename P > typedef P mln::p_key< K, P >::r_element`

Removal element associated type.

Definition at line 132 of file p_key.hh.

10.284.3 Constructor & Destructor Documentation

10.284.3.1 `template<typename K , typename P > mln::p_key< K, P >::p_key () [inline]`

Constructor.

Definition at line 208 of file p_key.hh.

10.284.4 Member Function Documentation

10.284.4.1 `template<typename K , typename P > void mln::p_key< K, P >::change_key (const K & k, const K & new_k) [inline]`

Change the key *k* into a new value *new_k*.

Definition at line 382 of file p_key.hh.

References `mln::p_set< P >::nsites()`.

10.284.4.2 `template<typename K , typename P > template<typename F > void mln::p_key< K, P >::change_keys (const Function_v2v< F > & f) [inline]`

Change the keys by applying the function *f*.

Definition at line 428 of file `p_key.hh`.

References `mln::util::set< T >::insert()`.

10.284.4.3 `template<typename K , typename P > void mln::p_key< K, P >::clear () [inline]`

Clear this site set.

Definition at line 462 of file `p_key.hh`.

10.284.4.4 `template<typename K , typename P > bool mln::p_key< K, P >::exists_key (const K & key) const [inline]`

Test if the `priority` exists.

Definition at line 520 of file `p_key.hh`.

Referenced by `mln::p_key< K, P >::operator()()`.

10.284.4.5 `template<typename K , typename P > bool mln::p_key< K, P >::has (const P & p) const [inline]`

Test is the psite *p* belongs to this site set.

Definition at line 227 of file `p_key.hh`.

10.284.4.6 `template<typename K , typename P > bool mln::p_key< K, P >::has (const psite &) const [inline]`

Test is the psite *p* belongs to this site set.

Definition at line 217 of file `p_key.hh`.

Referenced by `mln::p_key< K, P >::insert()`.

10.284.4.7 `template<typename K , typename P > void mln::p_key< K, P >::insert (const i_element & k_p) [inline]`

Insert a pair `k_p` (key *k*, site *p*).

Definition at line 301 of file `p_key.hh`.

10.284.4.8 `template<typename K , typename P > void mln::p_key< K, P >::insert (const K & k, const P & p) [inline]`

Insert a pair (key *k*, site *p*).

Definition at line 268 of file `p_key.hh`.

References mln::p_key< K, P >::has().

10.284.4.9 `template<typename K , typename P > bool mln::p_key< K, P >::is_valid () const [inline]`

Test this set validity so returns always true.

Definition at line 236 of file p_key.hh.

10.284.4.10 `template<typename K , typename P > const K & mln::p_key< K, P >::key (const P & p) const [inline]`

Give the key associated with site p.

Definition at line 501 of file p_key.hh.

10.284.4.11 `template<typename K , typename P > const util::set< K > & mln::p_key< K, P >::keys () const [inline]`

Give the set of keys.

Definition at line 511 of file p_key.hh.

10.284.4.12 `template<typename K , typename P > std::size_t mln::p_key< K, P >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 475 of file p_key.hh.

10.284.4.13 `template<typename K , typename P > unsigned mln::p_key< K, P >::nsites () const [inline]`

Give the number of sites.

Definition at line 245 of file p_key.hh.

10.284.4.14 `template<typename K , typename P > const p_set< P > & mln::p_key< K, P >::operator() (const K & key) const [inline]`

Give the queue with the priority *priority*.

This method always works: if the priority is not in this set, an empty queue is returned.

Definition at line 489 of file p_key.hh.

References mln::p_key< K, P >::exists_key().

10.284.4.15 `template<typename K , typename P > void mln::p_key< K, P >::remove (const P & p) [inline]`

Remove a site p.

Definition at line 309 of file p_key.hh.

10.284.4.16 `template<typename K , typename P > void mln::p_key< K, P >::remove_key (const K & k) [inline]`

Remove all sites with key *k*.

Definition at line 351 of file `p_key.hh`.

References `mln::p_set< P >::nsites()`.

10.285 mln::p_line2d Class Reference

2D discrete line of points.

```
#include <p_line2d.hh>
```

Inherits `site_set_base_< point2d, p_line2d >`.

Public Types

- typedef `p_indexed_bkd_piter< self_ > bkd_piter`
Backward Site_Iterator associated type.
- typedef `point2d element`
Element associated type.
- typedef `p_indexed_fwd_piter< self_ > fwd_piter`
Forward Site_Iterator associated type.
- typedef `p_indexed_fwd_piter< self_ > piter`
Site_Iterator associated type.
- typedef `p_indexed_psite< self_ > psite`
Psite associated type.
- typedef const `box2d & q_box`
Box (qualified) associated type.

Public Member Functions

- const `box2d & bbox () const`
Give the exact bounding box.
- const `point2d & begin () const`
Give the point that begins the line.
- const `point2d & end () const`
Give the point that ends the line.
- bool `has (const util::index &i) const`

Test if index i belongs to this point set.

- bool `has` (const `psite` &p) const
Test if p belongs to this point set.
- bool `is_valid` () const
Test if this line is valid, i.e., initialized.
- `std::size_t` `memory_size` () const
Return the size of this site set in memory.
- unsigned `nsites` () const
Give the number of points.
- const `point2d` & `operator[]` (unsigned i) const
Return the i -th point of the line.
- `p_line2d` (const `point2d` &beg, const `point2d` &end, bool is_end_excluded=false)
Constructor from point beg to point end.
- `p_line2d` ()
Constructor without argument.
- const `std::vector< point2d >` & `std_vector` () const
Return the corresponding `std::vector` of points.

10.285.1 Detailed Description

2D discrete line of points. It is based on `p_array`.

Definition at line 79 of file `p_line2d.hh`.

10.285.2 Member Typedef Documentation

10.285.2.1 typedef `p_indexed_bkd_piter`<self_> `mln::p_line2d::bkd_piter`

Backward `Site_Iterator` associated type.

Definition at line 97 of file `p_line2d.hh`.

10.285.2.2 typedef `point2d` `mln::p_line2d::element`

Element associated type.

Definition at line 85 of file `p_line2d.hh`.

10.285.2.3 typedef `p_indexed_fwd_piter`<self_> `mln::p_line2d::fwd_piter`

Forward `Site_Iterator` associated type.

Definition at line 94 of file `p_line2d.hh`.

10.285.2.4 `typedef p_indexed_fwd_piter<self_> mln::p_line2d::piter`

[Site_Iterator](#) associated type.

Definition at line 91 of file `p_line2d.hh`.

10.285.2.5 `typedef p_indexed_psite<self_> mln::p_line2d::psite`

`Psite` associated type.

Definition at line 88 of file `p_line2d.hh`.

10.285.2.6 `typedef const box2d& mln::p_line2d::q_box`

[Box](#) (qualified) associated type.

Definition at line 132 of file `p_line2d.hh`.

10.285.3 Constructor & Destructor Documentation

10.285.3.1 `mln::p_line2d::p_line2d () [inline]`

Constructor without argument.

Definition at line 161 of file `p_line2d.hh`.

References `is_valid()`.

10.285.3.2 `mln::p_line2d::p_line2d (const point2d & beg, const point2d & end, bool is_end_excluded = false) [inline]`

Constructor from point `beg` to point `end`.

Definition at line 167 of file `p_line2d.hh`.

References `is_valid()`.

10.285.4 Member Function Documentation

10.285.4.1 `const box2d & mln::p_line2d::bbox () const [inline]`

Give the exact bounding box.

Definition at line 273 of file `p_line2d.hh`.

References `is_valid()`.

10.285.4.2 `const point2d & mln::p_line2d::begin () const [inline]`

Give the point that begins the line.

Definition at line 307 of file `p_line2d.hh`.

References `is_valid()`.

Referenced by mln::debug::draw_graph().

10.285.4.3 const point2d & mln::p_line2d::end () const [inline]

Give the point that ends the line.

Definition at line 315 of file p_line2d.hh.

References is_valid(), and nsites().

Referenced by mln::debug::draw_graph().

10.285.4.4 bool mln::p_line2d::has (const psite & p) const [inline]

Test if *p* belongs to this point set.

Definition at line 240 of file p_line2d.hh.

10.285.4.5 bool mln::p_line2d::has (const util::index & i) const [inline]

Test if index *i* belongs to this point set.

Definition at line 251 of file p_line2d.hh.

References nsites().

10.285.4.6 bool mln::p_line2d::is_valid () const [inline]

Test if this line is valid, i.e., initialized.

Definition at line 258 of file p_line2d.hh.

References mln::implies().

Referenced by bbox(), begin(), end(), and p_line2d().

10.285.4.7 std::size_t mln::p_line2d::memory_size () const [inline]

Return the size of this site set in memory.

Definition at line 323 of file p_line2d.hh.

10.285.4.8 unsigned mln::p_line2d::nsites () const [inline]

Give the number of points.

Definition at line 266 of file p_line2d.hh.

Referenced by end(), has(), and operator[]().

10.285.4.9 const point2d & mln::p_line2d::operator[] (unsigned i) const [inline]

Return the *i*-th point of the line.

Definition at line 299 of file p_line2d.hh.

References `nsites()`.

10.285.4.10 `const std::vector< point2d > & mln::p_line2d::std_vector () const [inline]`

Return the corresponding `std::vector` of points.

Definition at line 281 of file `p_line2d.hh`.

10.286 `mln::p_mutable_array_of< S >` Class Template Reference

`p_mutable_array_of` is a mutable array of site sets.

```
#include <p_mutable_array_of.hh>
```

Inherits `site_set_base< S::site, p_mutable_array_of< S >>`.

Public Types

- typedef `p_double_piter< self_, mln_bkd_eiter(array_), typename S::bkd_piter > bkd_piter`
Backward `Site_Iterator` associated type.
- typedef `S element`
Element associated type.
- typedef `p_double_piter< self_, mln_fwd_eiter(array_), typename S::fwd_piter > fwd_piter`
Forward `Site_Iterator` associated type.
- typedef `S i_element`
Insertion element associated type.
- typedef `fwd_piter piter`
`Site_Iterator` associated type.
- typedef `p_double_psite< self_, element > psite`
Psite associated type.

Public Member Functions

- void `clear ()`
Clear this set.
- bool `has (const psite &p) const`
Test if `p` belongs to this point set.
- void `insert (const S &s)`
Insert a site set `s`.
- bool `is_valid () const`

Test this set validity so returns always true.

- `std::size_t memory_size () const`
Return the size of this site set in memory.
- `unsigned nelements () const`
Give the number of elements (site sets) of this composite.
- `S & operator[] (unsigned i)`
*Return the *i*-th site set (mutable version).*
- `const S & operator[] (unsigned i) const`
*Return the *i*-th site set (const version).*
- `p_mutable_array_of ()`
Constructor without arguments.
- `void reserve (unsigned n)`
*Reserve memory for *n* elements.*

10.286.1 Detailed Description

`template<typename S> class mln::p_mutable_array_of< S >`

`p_mutable_array_of` is a mutable array of site sets. Parameter `S` is the type of the contained site sets. Definition at line 76 of file `p_mutable_array_of.hh`.

10.286.2 Member Typedef Documentation

10.286.2.1 `template<typename S > typedef p_double_piter<self_, mln_bkd_eiter(array_), typename S ::bkd_piter> mln::p_mutable_array_of< S >::bkd_piter`

Backward `Site_Iterator` associated type.

Definition at line 99 of file `p_mutable_array_of.hh`.

10.286.2.2 `template<typename S > typedef S mln::p_mutable_array_of< S >::element`

Element associated type.

Definition at line 85 of file `p_mutable_array_of.hh`.

10.286.2.3 `template<typename S > typedef p_double_piter<self_, mln_fwd_eiter(array_), typename S ::fwd_piter> mln::p_mutable_array_of< S >::fwd_piter`

Forward `Site_Iterator` associated type.

Definition at line 94 of file `p_mutable_array_of.hh`.

10.286.2.4 `template<typename S > typedef S mln::p_mutable_array_of< S >::i_element`

Insertion element associated type.

Definition at line 121 of file `p_mutable_array_of.hh`.

10.286.2.5 `template<typename S > typedef fwd_piter mln::p_mutable_array_of< S >::piter`

[Site_Iterator](#) associated type.

Definition at line 102 of file `p_mutable_array_of.hh`.

10.286.2.6 `template<typename S > typedef p_double_psite<self_, element>
mln::p_mutable_array_of< S >::psite`

Psite associated type.

Definition at line 89 of file `p_mutable_array_of.hh`.

10.286.3 Constructor & Destructor Documentation**10.286.3.1** `template<typename S > mln::p_mutable_array_of< S >::p_mutable_array_of ()
[inline]`

Constructor without arguments.

Definition at line 175 of file `p_mutable_array_of.hh`.

10.286.4 Member Function Documentation**10.286.4.1** `template<typename S > void mln::p_mutable_array_of< S >::clear () [inline]`

Clear this set.

Definition at line 241 of file `p_mutable_array_of.hh`.

10.286.4.2 `template<typename S > bool mln::p_mutable_array_of< S >::has (const psite & p)
const [inline]`

Test if `p` belongs to this point set.

Definition at line 190 of file `p_mutable_array_of.hh`.

10.286.4.3 `template<typename S > void mln::p_mutable_array_of< S >::insert (const S & s)
[inline]`

Insert a site set `s`.

Precondition

`s` is valid.

Definition at line 206 of file `p_mutable_array_of.hh`.

10.286.4.4 `template<typename S> bool mln::p_mutable_array_of< S >::is_valid () const [inline]`

Test this set validity so returns always true.

Definition at line 198 of file p_mutable_array_of.hh.

10.286.4.5 `template<typename S> std::size_t mln::p_mutable_array_of< S >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 258 of file p_mutable_array_of.hh.

10.286.4.6 `template<typename S> unsigned mln::p_mutable_array_of< S >::nelements () const [inline]`

Give the number of elements (site sets) of this composite.

Definition at line 233 of file p_mutable_array_of.hh.

10.286.4.7 `template<typename S> S & mln::p_mutable_array_of< S >::operator[] (unsigned i) [inline]`

Return the *i*-th site set (mutable version).

Definition at line 224 of file p_mutable_array_of.hh.

10.286.4.8 `template<typename S> const S & mln::p_mutable_array_of< S >::operator[] (unsigned i) const [inline]`

Return the *i*-th site set (const version).

Definition at line 215 of file p_mutable_array_of.hh.

10.286.4.9 `template<typename S> void mln::p_mutable_array_of< S >::reserve (unsigned n) [inline]`

Reserve memory for *n* elements.

Definition at line 182 of file p_mutable_array_of.hh.

10.287 mln::p_n_faces_bkd_piter< D, G > Class Template Reference

Backward iterator on the *n*-faces sites of an mln::p_complex<D, G>.

```
#include <p_n_faces_piter.hh>
```

Inherits p_complex_piter_base_< topo::n_face_bkd_iter< D >, p_complex< D, G >, G::site, p_n_faces_bkd_piter< D, G > >.

Public Member Functions

- [p_n_faces_bkd_piter](#) ()
Construction and assignment.
- unsigned [n](#) () const
Accessors.

10.287.1 Detailed Description

`template<unsigned D, typename G> class mln::p_n_faces_bkd_piter< D, G >`

Backward iterator on the n-faces sites of an `mln::p_complex<D, G>`.

Definition at line 92 of file `p_n_faces_piter.hh`.

10.287.2 Constructor & Destructor Documentation

10.287.2.1 `template<unsigned D, typename G > mln::p_n_faces_bkd_piter< D, G >::p_n_faces_bkd_piter () [inline]`

Construction and assignment.

Definition at line 169 of file `p_n_faces_piter.hh`.

10.287.3 Member Function Documentation

10.287.3.1 `template<unsigned D, typename G > unsigned mln::p_n_faces_bkd_piter< D, G >::n () const [inline]`

Accessors.

Shortcuts to `face_`'s accessors.

Definition at line 186 of file `p_n_faces_piter.hh`.

10.288 mln::p_n_faces_fwd_piter< D, G > Class Template Reference

Forward iterator on the n-faces sites of an `mln::p_complex<D, G>`.

```
#include <p_n_faces_piter.hh>
```

Inherits `p_complex_piter_base_< topo::n_face_fwd_iter< D >, p_complex< D, G >, G::site, p_n_faces_fwd_piter< D, G > >`.

Public Member Functions

- void [next](#) ()

Go to the next element.

- [p_n_faces_fwd_piter \(\)](#)
Construction and assignment.
- `unsigned n () const`
Accessors.

10.288.1 Detailed Description

`template<unsigned D, typename G> class mln::p_n_faces_fwd_piter< D, G >`

Forward iterator on the n-faces sites of an `mln::p_complex<D, G>`.

Definition at line 56 of file `p_n_faces_piter.hh`.

10.288.2 Constructor & Destructor Documentation

10.288.2.1 `template<unsigned D, typename G > mln::p_n_faces_fwd_piter< D, G >::p_n_faces_fwd_piter () [inline]`

Construction and assignment.

Definition at line 132 of file `p_n_faces_piter.hh`.

10.288.3 Member Function Documentation

10.288.3.1 `template<unsigned D, typename G > unsigned mln::p_n_faces_fwd_piter< D, G >::n () const [inline]`

Accessors.

Shortcuts to `face_`'s accessors.

Definition at line 149 of file `p_n_faces_piter.hh`.

10.288.3.2 `void mln::Site_Iterator< p_n_faces_fwd_piter< D, G > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.289 `mln::p_priority< P, Q >` Class Template Reference

Priority queue.

```
#include <p_priority.hh>
```

Inherits `site_set_base_< Q::site, p_priority< P, Q > >`.

Public Types

- typedef `p_double_piter< self_, mln_fwd_eiter(util::set< P >), typename Q::bkd_piter >` `bkd_piter`
Backward [Site_Iterator](#) associated type.
- typedef `Q::element` `element`
Element associated type.
- typedef `p_double_piter< self_, mln_bkd_eiter(util::set< P >), typename Q::fwd_piter >` `fwd_piter`
Forward [Site_Iterator](#) associated type.
- typedef `std::pair< P, element >` `i_element`
Insertion element associated type.
- typedef `fwd_piter` `piter`
[Site_Iterator](#) associated type.
- typedef `p_double_psite< self_, Q >` `psite`
Psite associated type.

Public Member Functions

- void `clear ()`
Clear the queue.
- bool `exists_priority (const P &priority) const`
Test if the `priority` exists.
- const `Q::element & front () const`
Give an element with highest priority.
- bool `has (const psite &) const`
Test is the psite `p` belongs to this site set.
- const `P highest_priority () const`
Give the highest priority.
- void `insert (const i_element &p_e)`
Insert a pair `p_e` (priority `p`, element `e`).

- void [insert](#) (const [p_priority](#)< P, Q > &other)
Insert elements from another priority queue.
- bool [is_valid](#) () const
Test this set validity so returns always true.
- const P [lowest_priority](#) () const
Give the lowest priority.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- unsigned [nsites](#) () const
Give the number of sites.
- const Q & [operator](#)() (const P &priority) const
Give the queue with the priority priority.
- [p_priority](#) ()
Constructor.
- void [pop](#) ()
Pop (remove) from the queue an element with highest priority.
- Q::element [pop_front](#) ()
Return an element with highest priority and remove it from the set.
- const [util::set](#)< P > & [priorities](#) () const
Give the set of priorities.
- void [push](#) (const P &priority, const [element](#) &e)
Push in the queue with priority the element e.

10.289.1 Detailed Description

template<typename P, typename Q> **class** mln::p_priority< P, Q >

Priority queue. The parameter P is the type of the priorities (for instance unsigned).

The parameter Q is a type of queue (for instance p_queue<point2d>).

Definition at line 76 of file p_priority.hh.

10.289.2 Member Typedef Documentation

10.289.2.1 **template**<typename P, typename Q> **typedef** p_double_piter< self_, mln_fwd_eiter(util::set<P>), typename Q ::bkd_piter > mln::p_priority< P, Q >::bkd_piter

Backward [Site_Iterator](#) associated type.

Definition at line 98 of file p_priority.hh.

10.289.2.2 `template<typename P, typename Q> typedef Q ::element mln::p_priority< P, Q >::element`

Element associated type.

Definition at line 84 of file p_priority.hh.

10.289.2.3 `template<typename P, typename Q> typedef p_double_piter< self_, mln_bkd_eiter(util::set<P>), typename Q ::fwd_piter > mln::p_priority< P, Q >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 93 of file p_priority.hh.

10.289.2.4 `template<typename P, typename Q> typedef std::pair<P, element> mln::p_priority< P, Q >::i_element`

Insertion element associated type.

Definition at line 121 of file p_priority.hh.

10.289.2.5 `template<typename P, typename Q> typedef fwd_piter mln::p_priority< P, Q >::piter`

[Site_Iterator](#) associated type.

Definition at line 101 of file p_priority.hh.

10.289.2.6 `template<typename P, typename Q> typedef p_double_psite<self_, Q> mln::p_priority< P, Q >::psite`

Psite associated type.

Definition at line 88 of file p_priority.hh.

10.289.3 Constructor & Destructor Documentation

10.289.3.1 `template<typename P , typename Q > mln::p_priority< P, Q >::p_priority () [inline]`

Constructor.

Definition at line 202 of file p_priority.hh.

10.289.4 Member Function Documentation

10.289.4.1 `template<typename P , typename Q > void mln::p_priority< P, Q >::clear () [inline]`

Clear the queue.

Definition at line 316 of file p_priority.hh.

10.289.4.2 `template<typename P , typename Q > bool mln::p_priority< P, Q >::exists_priority (const P & priority) const [inline]`

Test if the `priority` exists.

Definition at line 366 of file p_priority.hh.

Referenced by `mln::p_priority< P, Q >::operator()`.

10.289.4.3 `template<typename P , typename Q > const Q::element & mln::p_priority< P, Q >::front () const [inline]`

Give an element with highest priority.

If several elements have this priority, the least recently inserted is chosen.

Precondition

`! is_empty()`

Definition at line 294 of file p_priority.hh.

References `mln::p_priority< P, Q >::highest_priority()`.

Referenced by `mln::morpho::meyer_wst()`, and `mln::morpho::watershed::topological()`.

10.289.4.4 `template<typename P , typename Q > bool mln::p_priority< P, Q >::has (const psite &) const [inline]`

Test is the psite `p` belongs to this site set.

Definition at line 211 of file p_priority.hh.

10.289.4.5 `template<typename P , typename Q > const P mln::p_priority< P, Q >::highest_priority () const [inline]`

Give the highest priority.

Precondition

`! is_empty()`

Definition at line 375 of file p_priority.hh.

Referenced by `mln::p_priority< P, Q >::front()`, and `mln::p_priority< P, Q >::pop()`.

10.289.4.6 `template<typename P , typename Q > void mln::p_priority< P, Q >::insert (const i_element & p_e) [inline]`

Insert a pair `p_e` (priority `p`, element `e`).

Definition at line 251 of file p_priority.hh.

References `mln::p_priority< P, Q >::push()`.

10.289.4.7 `template<typename P , typename Q > void mln::p_priority< P, Q >::insert (const p_priority< P, Q > & other) [inline]`

Insert elements from another priority queue.

Definition at line 259 of file p_priority.hh.

10.289.4.8 `template<typename P , typename Q > bool mln::p_priority< P, Q >::is_valid () const [inline]`

Test this set validity so returns always true.

Definition at line 221 of file p_priority.hh.

10.289.4.9 `template<typename P , typename Q > const P mln::p_priority< P, Q >::lowest_priority () const [inline]`

Give the lowest priority.

Precondition

`! is_empty()`

Definition at line 384 of file p_priority.hh.

10.289.4.10 `template<typename P , typename Q > std::size_t mln::p_priority< P, Q >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 328 of file p_priority.hh.

10.289.4.11 `template<typename P , typename Q > unsigned mln::p_priority< P, Q >::nsites () const [inline]`

Give the number of sites.

Definition at line 230 of file p_priority.hh.

Referenced by `mln::p_priority< P, Q >::operator()`.

10.289.4.12 `template<typename P , typename Q > const Q & mln::p_priority< P, Q >::operator() (const P & priority) const [inline]`

Give the queue with the priority `priority`.

This method always works: if the priority is not in this set, an empty queue is returned.

Definition at line 341 of file p_priority.hh.

References `mln::p_priority< P, Q >::exists_priority()`, and `mln::p_priority< P, Q >::nsites()`.

10.289.4.13 `template<typename P , typename Q > void mln::p_priority< P, Q >::pop ()`
[inline]

Pop (remove) from the queue an element with highest priority.

If several elements have this priority, the least recently inserted is chosen.

Precondition

`! is_empty()`

Definition at line 277 of file `p_priority.hh`.

References `mln::p_priority< P, Q >::highest_priority()`.

Referenced by `mln::morpho::meyer_wst()`, and `mln::morpho::watershed::topological()`.

10.289.4.14 `template<typename P , typename Q > Q::element mln::p_priority< P, Q >::pop_front ()`
[inline]

Return an element with highest priority and remove it from the set.

If several elements have this priority, the least recently inserted is chosen.

Precondition

`! is_empty()`

Definition at line 304 of file `p_priority.hh`.

10.289.4.15 `template<typename P , typename Q > const util::set< P > & mln::p_priority< P, Q >::priorities () const`
[inline]

Give the set of priorities.

Definition at line 357 of file `p_priority.hh`.

10.289.4.16 `template<typename P , typename Q > void mln::p_priority< P, Q >::push (const P & priority, const element & e)`
[inline]

Push in the queue with `priority` the element `e`.

Definition at line 239 of file `p_priority.hh`.

Referenced by `mln::p_priority< P, Q >::insert()`, `mln::morpho::meyer_wst()`, and `mln::morpho::watershed::topological()`.

10.290 `mln::p_queue< P >` Class Template Reference

Queue of sites (based on `std::deque`).

`#include <p_queue.hh>`

Inherits `site_set_base< P, p_queue< P >>`.

Public Types

- typedef `p_indexed_bkd_piter< self_ > bkd_piter`
Backward `Site_Iterator` associated type.
- typedef `P element`
Element associated type.
- typedef `p_indexed_fwd_piter< self_ > fwd_piter`
Forward `Site_Iterator` associated type.
- typedef `P i_element`
Insertion element associated type.
- typedef `fwd_piter piter`
`Site_Iterator` associated type.
- typedef `p_indexed_psite< self_ > psite`
Psite associated type.

Public Member Functions

- void `clear ()`
Clear the queue.
- const `P & front () const`
Give the front site `p` of the queue; `p` is the least recently inserted site.
- bool `has (const psite &p) const`
Test if `p` belongs to this site set.
- bool `has (const util::index &i) const`
Test if index `i` belongs to this site set.
- void `insert (const P &p)`
Insert a site `p` (equivalent as 'push').
- bool `is_valid () const`
This set is always valid so it returns true.
- `std::size_t memory_size () const`
Return the size of this site set in memory.
- unsigned `nsites () const`
Give the number of sites.
- const `P & operator[] (unsigned i) const`
Return the `i`-th site.

- [p_queue](#) ()
Constructor without argument.
- void [pop](#) ()
Pop (remove) the front site p from the queue; p is the least recently inserted site.
- P [pop_front](#) ()
Pop (remove) the front site p from the queue; p is the least recently inserted site and give the front site p of the queue; p is the least recently inserted site.
- void [push](#) (const P &p)
Push a site p in the queue.
- const std::deque< P > & [std_deque](#) () const
Return the corresponding std::deque of sites.

10.290.1 Detailed Description

template<typename P> class mln::p_queue< P >

Queue of sites (based on std::deque). The parameter P shall be a site or pseudo-site type.

Definition at line 74 of file p_queue.hh.

10.290.2 Member Typedef Documentation

10.290.2.1 template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_queue< P >::bkd_piter

Backward [Site_Iterator](#) associated type.

Definition at line 90 of file p_queue.hh.

10.290.2.2 template<typename P> typedef P mln::p_queue< P >::element

Element associated type.

Definition at line 80 of file p_queue.hh.

10.290.2.3 template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_queue< P >::fwd_piter

Forward [Site_Iterator](#) associated type.

Definition at line 87 of file p_queue.hh.

10.290.2.4 template<typename P> typedef P mln::p_queue< P >::i_element

Insertion element associated type.

Definition at line 118 of file p_queue.hh.

10.290.2.5 `template<typename P> typedef fwd_piter mln::p_queue< P >::piter`

[Site_Iterator](#) associated type.

Definition at line 93 of file p_queue.hh.

10.290.2.6 `template<typename P> typedef p_indexed_psite<self_> mln::p_queue< P >::psite`

Psite associated type.

Definition at line 84 of file p_queue.hh.

10.290.3 Constructor & Destructor Documentation**10.290.3.1** `template<typename P> mln::p_queue< P >::p_queue () [inline]`

Constructor without argument.

Definition at line 163 of file p_queue.hh.

10.290.4 Member Function Documentation**10.290.4.1** `template<typename P> void mln::p_queue< P >::clear () [inline]`

Clear the queue.

Definition at line 244 of file p_queue.hh.

10.290.4.2 `template<typename P> const P & mln::p_queue< P >::front () const [inline]`

Give the front site *p* of the queue; *p* is the least recently inserted site.

Definition at line 224 of file p_queue.hh.

Referenced by `mln::p_queue< P >::pop_front()`, and `mln::geom::impl::seeds2tiling()`.

10.290.4.3 `template<typename P> bool mln::p_queue< P >::has (const util::index & i) const [inline]`

Test if index *i* belongs to this site set.

Definition at line 183 of file p_queue.hh.

References `mln::p_queue< P >::nsites()`.

10.290.4.4 `template<typename P> bool mln::p_queue< P >::has (const psite & p) const [inline]`

Test if *p* belongs to this site set.

Definition at line 170 of file p_queue.hh.

References `mln::p_queue< P >::nsites()`.

10.290.4.5 `template<typename P> void mln::p_queue< P >::insert (const P & p) [inline]`

Insert a site `p` (equivalent as 'push').

Definition at line 261 of file `p_queue.hh`.

References `mln::p_queue< P >::push()`.

10.290.4.6 `template<typename P> bool mln::p_queue< P >::is_valid () const [inline]`

This set is always valid so it returns true.

Definition at line 191 of file `p_queue.hh`.

10.290.4.7 `template<typename P> std::size_t mln::p_queue< P >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 277 of file `p_queue.hh`.

References `mln::p_queue< P >::nsites()`.

10.290.4.8 `template<typename P> unsigned mln::p_queue< P >::nsites () const [inline]`

Give the number of sites.

Definition at line 199 of file `p_queue.hh`.

Referenced by `mln::p_queue< P >::has()`, `mln::p_queue< P >::memory_size()`, and `mln::p_queue< P >::operator[]()`.

10.290.4.9 `template<typename P> const P & mln::p_queue< P >::operator[] (unsigned i) const [inline]`

Return the `i`-th site.

Definition at line 252 of file `p_queue.hh`.

References `mln::p_queue< P >::nsites()`.

10.290.4.10 `template<typename P> void mln::p_queue< P >::pop () [inline]`

Pop (remove) the front site `p` from the queue; `p` is the least recently inserted site.

Definition at line 215 of file `p_queue.hh`.

Referenced by `mln::p_queue< P >::pop_front()`, and `mln::geom::impl::seeds2tiling()`.

10.290.4.11 `template<typename P> P mln::p_queue< P >::pop_front () [inline]`

Pop (remove) the front site `p` from the queue; `p` is the least recently inserted site and give the front site `p` of the queue; `p` is the least recently inserted site.

Definition at line 233 of file `p_queue.hh`.

References `mln::p_queue< P >::front()`, and `mln::p_queue< P >::pop()`.

10.290.4.12 `template<typename P > void mln::p_queue< P >::push (const P & p) [inline]`

Push a site `p` in the queue.

Definition at line 207 of file `p_queue.hh`.

Referenced by `mln::p_queue< P >::insert()`, and `mln::geom::impl::seeds2tiling()`.

10.290.4.13 `template<typename P > const std::deque< P > & mln::p_queue< P >::std_deque () const [inline]`

Return the corresponding `std::deque` of sites.

Definition at line 269 of file `p_queue.hh`.

10.291 `mln::p_queue_fast< P >` Class Template Reference

Queue of sites class (based on [p_array](#)).

```
#include <p_queue_fast.hh>
```

Inherits `site_set_base_< P, p_queue_fast< P > >`.

Public Types

- typedef `p_indexed_bkd_piter< self_ > bkd_piter`
Backward [Site_Iterator](#) associated type.
- typedef `P element`
Element associated type.
- typedef `p_indexed_fwd_piter< self_ > fwd_piter`
Forward [Site_Iterator](#) associated type.
- typedef `P i_element`
Insertion element associated type.
- typedef `fwd_piter piter`
[Site_Iterator](#) associated type.
- typedef `p_indexed_psite< self_ > psite`
Psite associated type.

Public Member Functions

- void `clear ()`
Clear the queue.

- bool `compute_has` (const P &p) const
Test if p belongs to this site set.
- bool `empty` () const
Test if the queue is empty.
- const P & `front` () const
Give the front site p of the queue; p is the least recently inserted site.
- bool `has` (const util::index &i) const
Test if index i belongs to this site set.
- bool `has` (const psite &p) const
Test if p belongs to this site set.
- void `insert` (const P &p)
Insert a site p (equivalent as 'push').
- bool `is_valid` () const
This set is always valid so it returns true.
- std::size_t `memory_size` () const
Return the size of this site set in memory.
- unsigned `nsites` () const
Give the number of sites.
- const P & `operator[]` (unsigned i) const
Return the i -th site.
- `p_queue_fast` ()
Constructor without argument.
- void `pop` ()
Pop (remove) the front site p from the queue; p is the least recently inserted site.
- const P & `pop_front` ()
Pop (remove) the front site p from the queue; p is the least recently inserted site and give the front site p of the queue; p is the least recently inserted site.
- void `purge` ()
Purge the queue to save (free) some memory.
- void `push` (const P &p)
Push a site p in the queue.
- void `reserve` (typename p_array< P >::size_type n)
Reserve n cells.

- `const std::vector< P > & std_vector () const`

Return the corresponding std::vector of sites.

10.291.1 Detailed Description

template<typename P> class mln::p_queue_fast< P >

Queue of sites class (based on [p_array](#)).

This container is efficient; FIXME: explain...

The parameter P shall be a site or pseudo-site type.

Definition at line 72 of file p_queue_fast.hh.

10.291.2 Member Typedef Documentation

10.291.2.1 template<typename P > typedef p_indexed_bkd_piter<self_> mln::p_queue_fast< P >::bkd_piter

Backward [Site_Iterator](#) associated type.

Definition at line 87 of file p_queue_fast.hh.

10.291.2.2 template<typename P > typedef P mln::p_queue_fast< P >::element

Element associated type.

Definition at line 78 of file p_queue_fast.hh.

10.291.2.3 template<typename P > typedef p_indexed_fwd_piter<self_> mln::p_queue_fast< P >::fwd_piter

Forward [Site_Iterator](#) associated type.

Definition at line 84 of file p_queue_fast.hh.

10.291.2.4 template<typename P > typedef P mln::p_queue_fast< P >::i_element

Insertion element associated type.

Definition at line 121 of file p_queue_fast.hh.

10.291.2.5 template<typename P > typedef fwd_piter mln::p_queue_fast< P >::piter

[Site_Iterator](#) associated type.

Definition at line 90 of file p_queue_fast.hh.

10.291.2.6 `template<typename P > typedef p_indexed_psite<self_> mln::p_queue_fast< P >::psite`

Psite associated type.

Definition at line 81 of file p_queue_fast.hh.

10.291.3 Constructor & Destructor Documentation**10.291.3.1** `template<typename P > mln::p_queue_fast< P >::p_queue_fast () [inline]`

Constructor without argument.

Definition at line 170 of file p_queue_fast.hh.

10.291.4 Member Function Documentation**10.291.4.1** `template<typename P > void mln::p_queue_fast< P >::clear () [inline]`

Clear the queue.

Definition at line 297 of file p_queue_fast.hh.

10.291.4.2 `template<typename P > bool mln::p_queue_fast< P >::compute_has (const P & p) const [inline]`

Test if *p* belongs to this site set.

Definition at line 222 of file p_queue_fast.hh.

10.291.4.3 `template<typename P > bool mln::p_queue_fast< P >::empty () const [inline]`

Test if the queue is empty.

Definition at line 250 of file p_queue_fast.hh.

10.291.4.4 `template<typename P > const P & mln::p_queue_fast< P >::front () const [inline]`

Give the front site *p* of the queue; *p* is the least recently inserted site.

Definition at line 277 of file p_queue_fast.hh.

Referenced by mln::p_queue_fast< P >::pop_front().

10.291.4.5 `template<typename P > bool mln::p_queue_fast< P >::has (const psite & p) const [inline]`

Test if *p* belongs to this site set.

Definition at line 201 of file p_queue_fast.hh.

References mln::p_queue_fast< P >::nsites().

10.291.4.6 `template<typename P> bool mln::p_queue_fast< P >::has (const util::index & i) const [inline]`

Test if index *i* belongs to this site set.

Definition at line 214 of file `p_queue_fast.hh`.

References `mln::p_queue_fast< P >::nsites()`.

10.291.4.7 `template<typename P> void mln::p_queue_fast< P >::insert (const P & p) [inline]`

Insert a site *p* (equivalent as 'push').

Definition at line 314 of file `p_queue_fast.hh`.

References `mln::p_queue_fast< P >::push()`.

10.291.4.8 `template<typename P> bool mln::p_queue_fast< P >::is_valid () const [inline]`

This set is always valid so it returns true.

Definition at line 233 of file `p_queue_fast.hh`.

10.291.4.9 `template<typename P> std::size_t mln::p_queue_fast< P >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 330 of file `p_queue_fast.hh`.

10.291.4.10 `template<typename P> unsigned mln::p_queue_fast< P >::nsites () const [inline]`

Give the number of sites.

Definition at line 241 of file `p_queue_fast.hh`.

Referenced by `mln::p_queue_fast< P >::has()`, and `mln::p_queue_fast< P >::operator[]()`.

10.291.4.11 `template<typename P> const P & mln::p_queue_fast< P >::operator[] (unsigned i) const [inline]`

Return the *i*-th site.

Definition at line 305 of file `p_queue_fast.hh`.

References `mln::p_queue_fast< P >::nsites()`.

10.291.4.12 `template<typename P> void mln::p_queue_fast< P >::pop () [inline]`

Pop (remove) the front site *p* from the queue; *p* is the least recently inserted site.

Definition at line 268 of file `p_queue_fast.hh`.

Referenced by mln::p_queue_fast< P >::pop_front().

10.291.4.13 `template<typename P > const P & mln::p_queue_fast< P >::pop_front ()`
[inline]

Pop (remove) the front site `p` from the queue; `p` is the least recently inserted site and give the front site `p` of the queue; `p` is the least recently inserted site.

Definition at line 286 of file `p_queue_fast.hh`.

References mln::p_queue_fast< P >::front(), and mln::p_queue_fast< P >::pop().

10.291.4.14 `template<typename P > void mln::p_queue_fast< P >::purge ()` **[inline]**

Purge the queue to save (free) some memory.

Definition at line 187 of file `p_queue_fast.hh`.

10.291.4.15 `template<typename P > void mln::p_queue_fast< P >::push (const P & p)`
[inline]

Push a site `p` in the queue.

Definition at line 259 of file `p_queue_fast.hh`.

Referenced by mln::p_queue_fast< P >::insert().

10.291.4.16 `template<typename P > void mln::p_queue_fast< P >::reserve (typename`
`p_array< P >::size_type n)` **[inline]**

Reserve `n` cells.

Definition at line 179 of file `p_queue_fast.hh`.

10.291.4.17 `template<typename P > const std::vector< P > & mln::p_queue_fast< P`
`>::std_vector () const` **[inline]**

Return the corresponding `std::vector` of sites.

Definition at line 322 of file `p_queue_fast.hh`.

10.292 mln::p_run< P > Class Template Reference

[Point](#) set class in run.

```
#include <p_run.hh>
```

Inherits `site_set_base_< P, p_run< P > >`.

Public Types

- `typedef p_run_bkd_piter_< P > bkd_piter`

Backward [Site_Iterator](#) associated type.

- typedef P [element](#)
Element associated type.
- typedef p_run_fwd_piter_< P > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef [fwd_piter](#) piter
[Site_Iterator](#) associated type.
- typedef p_run_psite< P > [psite](#)
Psite associated type.
- typedef [mln::box](#)< P > [q_box](#)
Box associated type.

Public Member Functions

- [mln::box](#)< P > [bbox](#) () const
Give the exact bounding box.
- P [end](#) () const
Return (compute) the ending point.
- bool [has](#) (const P &p) const
*Test if *p* belongs to this point set.*
- bool [has](#) (const [psite](#) &p) const
*Test if *p* belongs to this point set.*
- bool [has_index](#) (unsigned short i) const
*Test if index *i* belongs to this point set.*
- void [init](#) (const P &start, unsigned short len)
Set the starting point.
- bool [is_valid](#) () const
*Test if this run is valid, i.e., with *length* > 0.*
- unsigned short [length](#) () const
Give the length of the run.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- unsigned [nsites](#) () const
Give the number of sites.

- `P operator[]` (unsigned short i) const
Return the i -th point.
- `p_run()`
Constructor without argument.
- `p_run` (const P &start, unsigned short len)
Constructor.
- `p_run` (const P &start, const P &end)
Constructor.
- const P & `start` () const
Return the starting point.

10.292.1 Detailed Description

template<typename P> class mln::p_run< P >

Point set class in run. This is a mathematical set of points (not a multi-set). The parameter `P` shall be a **Point** type.

Definition at line 86 of file p_run.hh.

10.292.2 Member Typedef Documentation

10.292.2.1 template<typename P> typedef p_run_bkd_piter_<P> mln::p_run< P >::bkd_piter

Backward **Site_Iterator** associated type.

Definition at line 101 of file p_run.hh.

10.292.2.2 template<typename P> typedef P mln::p_run< P >::element

Element associated type.

Definition at line 91 of file p_run.hh.

10.292.2.3 template<typename P> typedef p_run_fwd_piter_<P> mln::p_run< P >::fwd_piter

Forward **Site_Iterator** associated type.

Definition at line 98 of file p_run.hh.

10.292.2.4 template<typename P> typedef fwd_piter mln::p_run< P >::piter

Site_Iterator associated type.

Definition at line 104 of file p_run.hh.

10.292.2.5 `template<typename P> typedef p_run_psite<P> mln::p_run< P >::psite`

Psite associated type.

Definition at line 95 of file p_run.hh.

10.292.2.6 `template<typename P> typedef mln::box<P> mln::p_run< P >::q_box`

[Box](#) associated type.

Definition at line 149 of file p_run.hh.

10.292.3 Constructor & Destructor Documentation**10.292.3.1** `template<typename P > mln::p_run< P >::p_run () [inline]`

Constructor without argument.

Definition at line 223 of file p_run.hh.

10.292.3.2 `template<typename P > mln::p_run< P >::p_run (const P & start, unsigned short len) [inline]`

Constructor.

Definition at line 230 of file p_run.hh.

References `mln::p_run< P >::init()`.

10.292.3.3 `template<typename P > mln::p_run< P >::p_run (const P & start, const P & end) [inline]`

Constructor.

Definition at line 238 of file p_run.hh.

10.292.4 Member Function Documentation**10.292.4.1** `template<typename P > mln::box< P > mln::p_run< P >::bbox () const [inline]`

Give the exact bounding box.

Definition at line 267 of file p_run.hh.

References `mln::p_run< P >::end()`.

10.292.4.2 `template<typename P > P mln::p_run< P >::end () const [inline]`

Return (compute) the ending point.

Definition at line 348 of file p_run.hh.

References `mln::point< G, C >::last_coord()`.

Referenced by mln::p_run< P >::bbox().

10.292.4.3 `template<typename P > bool mln::p_run< P >::has (const psite & p) const [inline]`

Test if *p* belongs to this point set.

Definition at line 276 of file p_run.hh.

10.292.4.4 `template<typename P > bool mln::p_run< P >::has (const P & p) const [inline]`

Test if *p* belongs to this point set.

Definition at line 289 of file p_run.hh.

References mln::p_run< P >::is_valid().

10.292.4.5 `template<typename P > bool mln::p_run< P >::has_index (unsigned short i) const [inline]`

Test if index *i* belongs to this point set.

Definition at line 302 of file p_run.hh.

10.292.4.6 `template<typename P > void mln::p_run< P >::init (const P & start, unsigned short len) [inline]`

Set the starting point.

Definition at line 249 of file p_run.hh.

Referenced by mln::p_run< P >::p_run().

10.292.4.7 `template<typename P > bool mln::p_run< P >::is_valid () const [inline]`

Test if this run is valid, i.e., with length > 0.

Definition at line 259 of file p_run.hh.

Referenced by mln::p_run< P >::has(), mln::p_run< P >::length(), mln::p_run< P >::nsites(), and mln::p_run< P >::operator[]().

10.292.4.8 `template<typename P > unsigned short mln::p_run< P >::length () const [inline]`

Give the length of the run.

Definition at line 319 of file p_run.hh.

References mln::p_run< P >::is_valid().

10.292.4.9 `template<typename P> std::size_t mln::p_run<P>::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 358 of file p_run.hh.

10.292.4.10 `template<typename P> unsigned mln::p_run<P>::nsites () const [inline]`

Give the number of sites.

Definition at line 310 of file p_run.hh.

References `mln::p_run<P>::is_valid()`.

10.292.4.11 `template<typename P> P mln::p_run<P>::operator[] (unsigned short i) const [inline]`

Return the *i*-th point.

Definition at line 328 of file p_run.hh.

References `mln::p_run<P>::is_valid()`, and `mln::point<G, C>::last_coord()`.

10.292.4.12 `template<typename P> const P & mln::p_run<P>::start () const [inline]`

Return the starting point.

Definition at line 340 of file p_run.hh.

10.293 mln::p_set<P> Class Template Reference

Mathematical set of sites (based on [util::set](#)).

```
#include <p_set.hh>
```

Inherits `site_set_base<P, p_set<P>>`.

Public Types

- typedef `p_indexed_bkd_piter<self_> bkd_piter`
Backward [Site_Iterator](#) associated type.
- typedef `P element`
Element associated type.
- typedef `p_indexed_fwd_piter<self_> fwd_piter`
Forward [Site_Iterator](#) associated type.
- typedef `P i_element`
Insertion element associated type.

- typedef `fwd_piter` `piter`
Site_iterator associated type.
- typedef `p_indexed_psite< self_ >` `psite`
Psite associated type.
- typedef `P_r_element`
Removal element associated type.

Public Member Functions

- void `clear` ()
Clear this set.
- bool `has` (const `psite` &p) const
Test if psite p belongs to this point set.
- bool `has` (const util::index &i) const
Test if index i belongs to this point set.
- bool `has` (const P &p) const
Test if p belongs to this point set.
- void `insert` (const P &p)
Insert a site p.
- bool `is_valid` () const
Test this set validity so returns always true.
- std::size_t `memory_size` () const
Return the size of this site set in memory.
- unsigned `nsites` () const
Give the number of sites.
- const P & `operator[]` (unsigned i) const
Return the i-th site.
- `p_set` ()
Constructor.
- void `remove` (const P &p)
Remove a site p.
- const std::vector< P > & `std_vector` () const
Return the corresponding std::vector of sites.
- const util::set< P > & `util_set` () const
Return the corresponding util::set of sites.

10.293.1 Detailed Description

template<typename P> class mln::p_set< P >

Mathematical set of sites (based on [util::set](#)). This is a mathematical set of sites (not a multi-set).

The parameter P shall be a site or pseudo-site type.

Definition at line 70 of file p_set.hh.

10.293.2 Member Typedef Documentation

10.293.2.1 template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_set< P >::bkd_piter

Backward [Site_Iterator](#) associated type.

Definition at line 85 of file p_set.hh.

10.293.2.2 template<typename P> typedef P mln::p_set< P >::element

Element associated type.

Definition at line 76 of file p_set.hh.

10.293.2.3 template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_set< P >::fwd_piter

Forward [Site_Iterator](#) associated type.

Definition at line 82 of file p_set.hh.

10.293.2.4 template<typename P> typedef P mln::p_set< P >::i_element

Insertion element associated type.

Definition at line 113 of file p_set.hh.

10.293.2.5 template<typename P> typedef fwd_piter mln::p_set< P >::piter

[Site_Iterator](#) associated type.

Definition at line 88 of file p_set.hh.

10.293.2.6 template<typename P> typedef p_indexed_psite<self_> mln::p_set< P >::psite

Psite associated type.

Definition at line 79 of file p_set.hh.

10.293.2.7 template<typename P> typedef P mln::p_set< P >::r_element

Removal element associated type.

Definition at line 119 of file p_set.hh.

10.293.3 Constructor & Destructor Documentation

10.293.3.1 `template<typename P> mln::p_set< P >::p_set () [inline]`

Constructor.

Definition at line 152 of file p_set.hh.

10.293.4 Member Function Documentation

10.293.4.1 `template<typename P> void mln::p_set< P >::clear () [inline]`

Clear this set.

Definition at line 219 of file p_set.hh.

10.293.4.2 `template<typename P> bool mln::p_set< P >::has (const psite & p) const [inline]`

Test if psite *p* belongs to this point set.

Definition at line 167 of file p_set.hh.

10.293.4.3 `template<typename P> bool mln::p_set< P >::has (const P & p) const [inline]`

Test if *p* belongs to this point set.

Definition at line 159 of file p_set.hh.

10.293.4.4 `template<typename P> bool mln::p_set< P >::has (const util::index & i) const [inline]`

Test if index *i* belongs to this point set.

Definition at line 179 of file p_set.hh.

References mln::p_set< P >::nsites().

10.293.4.5 `template<typename P> void mln::p_set< P >::insert (const P & p) [inline]`

Insert a site *p*.

Definition at line 203 of file p_set.hh.

Referenced by mln::convert::to_p_set().

10.293.4.6 `template<typename P> bool mln::p_set< P >::is_valid () const [inline]`

Test this set validity so returns always true.

Definition at line 187 of file p_set.hh.

10.293.4.7 `template<typename P> std::size_t mln::p_set< P >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 236 of file p_set.hh.

10.293.4.8 `template<typename P> unsigned mln::p_set< P >::nsites () const [inline]`

Give the number of sites.

Definition at line 195 of file p_set.hh.

Referenced by `mln::p_key< K, P >::change_key()`, `mln::p_set< P >::has()`, `mln::p_set< P >::operator[]()`, and `mln::p_key< K, P >::remove_key()`.

10.293.4.9 `template<typename P> const P & mln::p_set< P >::operator[] (unsigned i) const [inline]`

Return the *i*-th site.

Definition at line 227 of file p_set.hh.

References `mln::p_set< P >::nsites()`.

10.293.4.10 `template<typename P> void mln::p_set< P >::remove (const P & p) [inline]`

Remove a site *p*.

Definition at line 211 of file p_set.hh.

10.293.4.11 `template<typename P> const std::vector< P > & mln::p_set< P >::std_vector () const [inline]`

Return the corresponding `std::vector` of sites.

Definition at line 244 of file p_set.hh.

10.293.4.12 `template<typename P> const util::set< P > & mln::p_set< P >::util_set () const [inline]`

Return the corresponding `util::set` of sites.

Definition at line 252 of file p_set.hh.

10.294 `mln::p_set_of< S >` Class Template Reference

`p_set_of` is a set of site sets.

```
#include <p_set_of.hh>
```

Inherits `site_set_base< S::site, p_set_of< S >>`, and `site_set_impl< S >`.

Public Types

- typedef p_double_piter< self_, mln_bkd_eiter(set_), typename S::bkd_piter > **bkd_piter**
Backward Site_Iterator associated type.
- typedef S **element**
Element associated type.
- typedef p_double_piter< self_, mln_fwd_eiter(set_), typename S::fwd_piter > **fwd_piter**
Forward Site_Iterator associated type.
- typedef S **i_element**
Insertion element associated type.
- typedef fwd_piter **piter**
Site_Iterator associated type.
- typedef p_double_psite< self_, element > **psite**
Psite associated type.

Public Member Functions

- void **clear** ()
Clear this set.
- bool **has** (const **psite** &p) const
Test if p belongs to this point set.
- void **insert** (const S &s)
Insert a site set s.
- bool **is_valid** () const
Test if this set of runs is valid.
- std::size_t **memory_size** () const
Return the size of this site set in memory.
- unsigned **nelements** () const
Give the number of elements (site sets) of this composite.
- const S & **operator[]** (unsigned i) const
Return the i-th site set.
- **p_set_of** ()
Constructor without arguments.

10.294.1 Detailed Description

template<typename S> class mln::p_set_of< S >

[p_set_of](#) is a set of site sets. Parameter *S* is the type of the contained site sets.

Definition at line 71 of file `p_set_of.hh`.

10.294.2 Member Typedef Documentation

10.294.2.1 template<typename S > typedef p_double_piter<self_, mln_bkd_eiter(set_), typename S ::bkd_piter> mln::p_set_of< S >::bkd_piter

Backward [Site_Iterator](#) associated type.

Definition at line 95 of file `p_set_of.hh`.

10.294.2.2 template<typename S > typedef S mln::p_set_of< S >::element

Element associated type.

Definition at line 81 of file `p_set_of.hh`.

10.294.2.3 template<typename S > typedef p_double_piter<self_, mln_fwd_eiter(set_), typename S ::fwd_piter> mln::p_set_of< S >::fwd_piter

Forward [Site_Iterator](#) associated type.

Definition at line 90 of file `p_set_of.hh`.

10.294.2.4 template<typename S > typedef S mln::p_set_of< S >::i_element

Insertion element associated type.

Definition at line 113 of file `p_set_of.hh`.

10.294.2.5 template<typename S > typedef fwd_piter mln::p_set_of< S >::piter

[Site_Iterator](#) associated type.

Definition at line 98 of file `p_set_of.hh`.

10.294.2.6 template<typename S > typedef p_double_psite<self_, element> mln::p_set_of< S >::psite

Psite associated type.

Definition at line 85 of file `p_set_of.hh`.

10.294.3 Constructor & Destructor Documentation

10.294.3.1 `template<typename S > mln::p_set_of< S >::p_set_of () [inline]`

Constructor without arguments.

Definition at line 160 of file p_set_of.hh.

10.294.4 Member Function Documentation

10.294.4.1 `template<typename S > void mln::p_set_of< S >::clear () [inline]`

Clear this set.

Definition at line 211 of file p_set_of.hh.

10.294.4.2 `template<typename S > bool mln::p_set_of< S >::has (const psite & p) const [inline]`

Test if *p* belongs to this point set.

Definition at line 167 of file p_set_of.hh.

10.294.4.3 `template<typename S > void mln::p_set_of< S >::insert (const S & s) [inline]`

Insert a site set *s*.

Definition at line 183 of file p_set_of.hh.

10.294.4.4 `template<typename S > bool mln::p_set_of< S >::is_valid () const [inline]`

Test if this set of runs is valid.

Definition at line 175 of file p_set_of.hh.

10.294.4.5 `template<typename S > std::size_t mln::p_set_of< S >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 220 of file p_set_of.hh.

10.294.4.6 `template<typename S > unsigned mln::p_set_of< S >::nelements () const [inline]`

Give the number of elements (site sets) of this composite.

Definition at line 203 of file p_set_of.hh.

10.294.4.7 `template<typename S> const S & mln::p_set_of< S >::operator[] (unsigned i) const [inline]`

Return the `i`-th site set.

Definition at line 194 of file `p_set_of.hh`.

10.295 `mln::p_transformed< S, F >` Class Template Reference

[Site](#) set transformed through a function.

```
#include <p_transformed.hh>
```

Inherits `site_set_base_< S::psite, p_transformed< S, F >>`.

Public Types

- typedef `p_transformed_piter< typename S::bkd_piter, S, F > bkd_piter`
Backward [Site_Iterator](#) associated type.
- typedef `S::element element`
Element associated type.
- typedef `p_transformed_piter< typename S::fwd_piter, S, F > fwd_piter`
Forward [Site_Iterator](#) associated type.
- typedef `fwd_piter piter`
[Site_Iterator](#) associated type.
- typedef `S::psite psite`
[Psite](#) associated type.

Public Member Functions

- `const F & function () const`
Return the transformation function.
- `bool has (const psite &p) const`
Test if `p` belongs to the subset.
- `bool is_valid () const`
Test if this site set is valid.
- `std::size_t memory_size () const`
Return the size of this site set in memory.
- `p_transformed ()`
Constructor without argument.

- `p_transformed` (const S &s, const F &f)
Constructor with a site set s and a predicate f .
- const S & `primary_set` () const
Return the primary set.

10.295.1 Detailed Description

`template<typename S, typename F> class mln::p_transformed< S, F >`

[Site](#) set transformed through a function. Parameter S is a site set type; parameter F is a function from site to site.

Definition at line 82 of file `p_transformed.hh`.

10.295.2 Member Typedef Documentation

10.295.2.1 `template<typename S, typename F> typedef p_transformed_piter<typename S
::bkd_piter, S, F> mln::p_transformed< S, F >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 101 of file `p_transformed.hh`.

10.295.2.2 `template<typename S, typename F> typedef S ::element mln::p_transformed< S, F
>::element`

Element associated type.

Definition at line 91 of file `p_transformed.hh`.

10.295.2.3 `template<typename S, typename F> typedef p_transformed_piter<typename S
::fwd_piter, S, F> mln::p_transformed< S, F >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 98 of file `p_transformed.hh`.

10.295.2.4 `template<typename S, typename F> typedef fwd_piter mln::p_transformed< S, F
>::piter`

[Site_Iterator](#) associated type.

Definition at line 104 of file `p_transformed.hh`.

10.295.2.5 `template<typename S, typename F> typedef S ::psite mln::p_transformed< S, F
>::psite`

Psite associated type.

Definition at line 95 of file `p_transformed.hh`.

10.295.3 Constructor & Destructor Documentation

10.295.3.1 `template<typename S, typename F > mln::p_transformed< S, F >::p_transformed (const S & s, const F & f) [inline]`

Constructor with a site set *s* and a predicate *f*.

Definition at line 163 of file `p_transformed.hh`.

10.295.3.2 `template<typename S, typename F > mln::p_transformed< S, F >::p_transformed () [inline]`

Constructor without argument.

Definition at line 157 of file `p_transformed.hh`.

10.295.4 Member Function Documentation

10.295.4.1 `template<typename S, typename F > const F & mln::p_transformed< S, F >::function () const [inline]`

Return the transformation function.

Definition at line 206 of file `p_transformed.hh`.

10.295.4.2 `template<typename S, typename F > bool mln::p_transformed< S, F >::has (const psite & p) const [inline]`

Test if *p* belongs to the subset.

Definition at line 172 of file `p_transformed.hh`.

10.295.4.3 `template<typename S, typename F > bool mln::p_transformed< S, F >::is_valid () const [inline]`

Test if this site set is valid.

Definition at line 182 of file `p_transformed.hh`.

10.295.4.4 `template<typename S, typename F > std::size_t mln::p_transformed< S, F >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 190 of file `p_transformed.hh`.

10.295.4.5 `template<typename S, typename F > const S & mln::p_transformed< S, F >::primary_set () const [inline]`

Return the primary set.

Definition at line 198 of file `p_transformed.hh`.

Referenced by `mln::p_transformed_piter< Pi, S, F >::change_target()`.

10.296 `mln::p_transformed_piter< Pi, S, F >` Struct Template Reference

[Iterator](#) on `p_transformed<S,F>`.

```
#include <p_transformed_piter.hh>
```

Inherits `mln::internal::site_set_iterator_base< p_transformed< S, F >,p_transformed_piter< Pi, S, F >`.

Public Member Functions

- void [change_target](#) (const [p_transformed](#)< S, F > &s)
Change the set site targeted by this iterator.
- void [next](#) ()
Go to the next element.
- [p_transformed_piter](#) (const [p_transformed](#)< S, F > &s)
Constructor from a site set.
- [p_transformed_piter](#) ()
Constructor without argument.

10.296.1 Detailed Description

```
template<typename Pi, typename S, typename F> struct mln::p_transformed_piter< Pi, S, F >
```

[Iterator](#) on `p_transformed<S,F>`. Parameter `S` is a site set type; parameter `F` is a function from point to Boolean.

See also

[mln::p_transformed](#)

Definition at line 50 of file `p_transformed_piter.hh`.

10.296.2 Constructor & Destructor Documentation

10.296.2.1 `template<typename Pi, typename S, typename F > mln::p_transformed_piter< Pi, S, F >::p_transformed_piter () [inline]`

Constructor without argument.

Definition at line 93 of file `p_transformed_piter.hh`.

10.296.2.2 `template<typename Pi, typename S, typename F > mln::p_transformed_piter< Pi, S, F >::p_transformed_piter (const p_transformed< S, F > & s) [inline]`

Constructor from a site set.

Definition at line 99 of file p_transformed_piter.hh.

References mln::p_transformed_piter< Pi, S, F >::change_target().

10.296.3 Member Function Documentation

10.296.3.1 `template<typename Pi , typename S , typename F > void mln::p_transformed_piter< Pi, S, F >::change_target (const p_transformed< S, F > & s) [inline]`

Change the set site targeted by this iterator.

Definition at line 143 of file p_transformed_piter.hh.

References mln::p_transformed< S, F >::primary_set().

Referenced by mln::p_transformed_piter< Pi, S, F >::p_transformed_piter().

10.296.3.2 `template<typename E > void mln::Site_Iterator< E >::next () [inline, inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

Definition at line 92 of file site_iterator.hh.

10.297 mln::p_vaccess< V, S > Class Template Reference

[Site](#) set in which sites are grouped by their associated value.

```
#include <p_vaccess.hh>
```

Inherits site_set_base_< S::site, p_vaccess< V, S > >, and site_set_impl< S >.

Public Types

- typedef p_double_piter< [self_](#), typename vset::bkd_viter, typename S::bkd_piter > [bkd_piter](#)
Backward [Site_Iterator](#) associated type.
- typedef S::element [element](#)
Element associated type.
- typedef p_double_piter< [self_](#), typename vset::fwd_viter, typename S::fwd_piter > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef std::pair< V, [element](#) > [i_element](#)

Insertion element associated type.

- typedef fwd_piter piter
Site_Iterator associated type.
- typedef S pset
Inner site set associated type.
- typedef p_double_psite< self_, S > psite
Psite associated type.
- typedef V value
Value associated type.
- typedef mln::value::set< V > vset
Value_Set associated type.

Public Member Functions

- bool has (const psite &p) const
Test if p belongs to this site set.
- bool has (const V &v, const typename S::psite &p) const
Test if the couple (value v, psite p) belongs to this site set.
- void insert (const i_element &v_e)
Insert a pair v_e (value v, element e).
- void insert (const V &v, const element &e)
Insert e at value v.
- bool is_valid () const
Test if this site set is valid.
- std::size_t memory_size () const
Return the size of this site set in memory.
- const S & operator() (const V &v) const
Return the site set at value v.
- p_vaccess ()
Constructor.
- const mln::value::set< V > & values () const
Give the set of values.

10.297.1 Detailed Description

template<typename V, typename S> class mln::p_vaccess< V, S >

[Site](#) set in which sites are grouped by their associated value.

Definition at line 70 of file p_vaccess.hh.

10.297.2 Member Typedef Documentation

**10.297.2.1 template<typename V , typename S > typedef p_double_piter<self_, typename vset
::bkd_viter, typename S ::bkd_piter> mln::p_vaccess< V, S >::bkd_piter**

Backward [Site_Iterator](#) associated type.

Definition at line 94 of file p_vaccess.hh.

**10.297.2.2 template<typename V , typename S > typedef S ::element mln::p_vaccess< V, S
>::element**

Element associated type.

Definition at line 117 of file p_vaccess.hh.

**10.297.2.3 template<typename V , typename S > typedef p_double_piter<self_, typename vset
::fwd_viter, typename S ::fwd_piter> mln::p_vaccess< V, S >::fwd_piter**

Forward [Site_Iterator](#) associated type.

Definition at line 91 of file p_vaccess.hh.

**10.297.2.4 template<typename V , typename S > typedef std::pair<V, element> mln::p_vaccess<
V, S >::i_element**

Insertion element associated type.

Definition at line 120 of file p_vaccess.hh.

10.297.2.5 template<typename V , typename S > typedef fwd_piter mln::p_vaccess< V, S >::piter

[Site_Iterator](#) associated type.

Definition at line 97 of file p_vaccess.hh.

10.297.2.6 template<typename V , typename S > typedef S mln::p_vaccess< V, S >::pset

Inner site set associated type.

Definition at line 85 of file p_vaccess.hh.

10.297.2.7 `template<typename V , typename S > typedef p_double_psite<self_, S>
mln::p_vaccess< V, S >::psite`

Psite associated type.

Definition at line 88 of file p_vaccess.hh.

10.297.2.8 `template<typename V , typename S > typedef V mln::p_vaccess< V, S >::value`

Value associated type.

Definition at line 78 of file p_vaccess.hh.

10.297.2.9 `template<typename V , typename S > typedef mln::value::set<V> mln::p_vaccess<
V, S >::vset`

Value_Set associated type.

Definition at line 81 of file p_vaccess.hh.

10.297.3 Constructor & Destructor Documentation

10.297.3.1 `template<typename V , typename S > mln::p_vaccess< V, S >::p_vaccess ()
[inline]`

Constructor.

Definition at line 163 of file p_vaccess.hh.

10.297.4 Member Function Documentation

10.297.4.1 `template<typename V , typename S > bool mln::p_vaccess< V, S >::has (const V &
v, const typename S::psite & p) const [inline]`

Test if the couple (value v, psite p) belongs to this site set.

Definition at line 189 of file p_vaccess.hh.

10.297.4.2 `template<typename V , typename S > bool mln::p_vaccess< V, S >::has (const psite
& p) const [inline]`

Test if p belongs to this site set.

Definition at line 180 of file p_vaccess.hh.

10.297.4.3 `template<typename V , typename S > void mln::p_vaccess< V, S >::insert (const
i_element & v_e) [inline]`

Insert a pair v_e (value v, element e).

Definition at line 216 of file p_vaccess.hh.

10.297.4.4 `template<typename V , typename S > void mln::p_vaccess< V, S >::insert (const V & v, const element & e) [inline]`

Insert *e* at value *v*.

Definition at line 206 of file `p_vaccess.hh`.

10.297.4.5 `template<typename V , typename S > bool mln::p_vaccess< V, S >::is_valid () const [inline]`

Test if this site set is valid.

Definition at line 197 of file `p_vaccess.hh`.

10.297.4.6 `template<typename V , typename S > std::size_t mln::p_vaccess< V, S >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 242 of file `p_vaccess.hh`.

10.297.4.7 `template<typename V , typename S > const S & mln::p_vaccess< V, S >::operator() (const V & v) const [inline]`

Return the site set at value *v*.

Definition at line 234 of file `p_vaccess.hh`.

10.297.4.8 `template<typename V , typename S > const mln::value::set< V > & mln::p_vaccess< V, S >::values () const [inline]`

Give the set of values.

Definition at line 254 of file `p_vaccess.hh`.

10.298 mln::p_vertices< G, F > Class Template Reference

[Site](#) set based mapping graph vertices to sites.

```
#include <p_vertices.hh>
```

Inherits `site_set_base_< F::result, p_vertices< G, F >>`.

Public Types

- typedef `F` `fun_t`
Function associated type.
- typedef `util::vertex< G >` `graph_element`
Type of graph element this site set focuses on.
- typedef `G` `graph_t`

Graph associated type.

- typedef util::vertex< G > vertex
Type of graph vertex.
- typedef super_::site element
Associated types.
- typedef p_vertices_psite< G, F > psite
Point_Site associated type.
- typedef p_graph_piter< self_, mln_vertex_fwd_iter(G) > fwd_piter
Forward Site_Iterator associated type.
- typedef p_graph_piter< self_, mln_vertex_bkd_iter(G) > bkd_piter
Backward Site_Iterator associated type.
- typedef fwd_piter piter
Site_Iterator associated type.

Public Member Functions

- bool has (const psite &p) const
Does this site set has p?
- template<typename G2 >
bool has (const util::vertex< G2 > &v) const
Does this site set has v?
- void invalidate ()
Invalidate this site set.
- bool is_valid () const
Test this site set validity.
- std::size_t memory_size () const
*Does this site set has vertex_id? FIXME: causes ambiguities while calling has(mln::neighb_fwd_niter<>);
bool has(unsigned vertex_id) const;.*
- unsigned nsites () const
Return The number of points (sites) of the set, i.e., the number of vertices.
- unsigned nvertices () const
Return The number of vertices in the graph.
- p_vertices (const Graph< G > &gr)
Construct a graph psite set from a graph of points.
- p_vertices (const Graph< G > &gr, const Function< F > &f)

Construct a graph psite set from a graph of points.

- `template<typename F2 >`
`p_vertices` (const `p_vertices`< G, F2 > &other)
Copy constructor.
- `p_vertices` ()
Constructor without argument.
- `template<typename F2 >`
`p_vertices` (const `Graph`< G > &gr, const `Function`< F2 > &f)
Construct a graph psite set from a graph of points.
- `F::result operator()` (const `psite` &p) const
Return the value associated to an element of this site set.
- const G & `graph` () const
Accessors.
- const F & `function` () const
Return the association function.

10.298.1 Detailed Description

```
template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> class
mln::p_vertices< G, F >
```

[Site](#) set based mapping graph vertices to sites.

Definition at line 71 of file `p_vertices.hh`.

10.298.2 Member Typedef Documentation

10.298.2.1 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>`
`>> typedef p_graph_piter< self_, mln_vertex_bkd_iter(G) > mln::p_vertices< G, F`
`>::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 132 of file `p_vertices.hh`.

10.298.2.2 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>`
`>> typedef super_::site mln::p_vertices< G, F >::element`

Associated types.

Element associated type.

Definition at line 123 of file `p_vertices.hh`.

10.298.2.3 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef F mln::p_vertices< G, F >::fun_t`

[Function](#) associated type.

Definition at line 84 of file p_vertices.hh.

10.298.2.4 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef p_graph_piter< self_, mln_vertex_fwd_iter(G) > mln::p_vertices< G, F >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 129 of file p_vertices.hh.

10.298.2.5 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef util::vertex<G> mln::p_vertices< G, F >::graph_element`

Type of graph element this site set focuses on.

Definition at line 91 of file p_vertices.hh.

10.298.2.6 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef G mln::p_vertices< G, F >::graph_t`

[Graph](#) associated type.

Definition at line 81 of file p_vertices.hh.

10.298.2.7 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef fwd_piter mln::p_vertices< G, F >::piter`

[Site_Iterator](#) associated type.

Definition at line 135 of file p_vertices.hh.

10.298.2.8 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef p_vertices_psite<G,F> mln::p_vertices< G, F >::psite`

[Point_Site](#) associated type.

Definition at line 126 of file p_vertices.hh.

10.298.2.9 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef util::vertex<G> mln::p_vertices< G, F >::vertex`

Type of graph vertex.

Definition at line 87 of file p_vertices.hh.

10.298.3 Constructor & Destructor Documentation

10.298.3.1 `template<typename G , typename F > mln::p_vertices< G, F >::p_vertices ()`
[inline]

Constructor without argument.

Definition at line 220 of file p_vertices.hh.

10.298.3.2 `template<typename G , typename F > mln::p_vertices< G, F >::p_vertices (const`
Graph< G > & gr) [inline]

Construct a graph psite set from a graph of points.

Parameters

gr The graph upon which the graph psite set is built. The identity function is used.

Definition at line 226 of file p_vertices.hh.

References mln::p_vertices< G, F >::is_valid().

10.298.3.3 `template<typename G , typename F > mln::p_vertices< G, F >::p_vertices (const`
Graph< G > & gr, const Function< F > & f) [inline]

Construct a graph psite set from a graph of points.

Parameters

gr The graph upon which the graph psite set is built.

f the function which maps a vertex to a site.

Definition at line 238 of file p_vertices.hh.

References mln::p_vertices< G, F >::is_valid().

10.298.3.4 `template<typename G , typename F > template<typename F2 > mln::p_vertices< G,`
F >::p_vertices (const Graph< G > & gr, const Function< F2 > & f) [inline]

Construct a graph psite set from a graph of points.

Parameters

gr The graph upon which the graph psite set is built.

f the function which maps a vertex to a site. It must be convertible to the function type F.

Definition at line 248 of file p_vertices.hh.

References mln::p_vertices< G, F >::is_valid().

10.298.3.5 `template<typename G , typename F > template<typename F2 > mln::p_vertices< G, F >::p_vertices (const p_vertices< G, F2 > & other) [inline]`

Copy constructor.

Definition at line 260 of file p_vertices.hh.

References mln::p_vertices< G, F >::function(), mln::p_vertices< G, F >::graph(), and mln::p_vertices< G, F >::is_valid().

10.298.4 Member Function Documentation

10.298.4.1 `template<typename G , typename F > const F & mln::p_vertices< G, F >::function () const [inline]`

Return the association function.

Definition at line 385 of file p_vertices.hh.

Referenced by mln::p_vertices< G, F >::p_vertices().

10.298.4.2 `template<typename G , typename F > const G & mln::p_vertices< G, F >::graph () const [inline]`

Accessors.

Return the graph associated to this site set (const version)

Definition at line 376 of file p_vertices.hh.

References mln::p_vertices< G, F >::is_valid().

Referenced by mln::debug::draw_graph(), mln::operator==(), and mln::p_vertices< G, F >::p_vertices().

10.298.4.3 `template<typename G , typename F > bool mln::p_vertices< G, F >::has (const psite & p) const [inline]`

Does this site set has *p*?

Definition at line 304 of file p_vertices.hh.

References mln::p_vertices< G, F >::is_valid().

10.298.4.4 `template<typename G , typename F > template<typename G2 > bool mln::p_vertices< G, F >::has (const util::vertex< G2 > & v) const [inline]`

Does this site set has *v*?

Definition at line 314 of file p_vertices.hh.

References mln::util::vertex< G >::graph(), mln::util::vertex< G >::is_valid(), and mln::p_vertices< G, F >::is_valid().

10.298.4.5 `template<typename G , typename F > void mln::p_vertices< G, F >::invalidate ()`
`[inline]`

Invalidate this site set.

Definition at line 296 of file p_vertices.hh.

10.298.4.6 `template<typename G , typename F > bool mln::p_vertices< G, F >::is_valid ()`
`const [inline]`

Test this site set validity.

Definition at line 288 of file p_vertices.hh.

Referenced by `mln::p_vertices< G, F >::graph()`, `mln::p_vertices< G, F >::has()`, and `mln::p_vertices< G, F >::p_vertices()`.

10.298.4.7 `template<typename G , typename F > std::size_t mln::p_vertices< G, F`
`>::memory_size () const [inline]`

Does this site set has *vertex_id*? FIXME: causes ambiguities while calling `has(mln::neighb_fwd_niter<>)`; `bool has(unsigned vertex_id) const;`

Definition at line 339 of file p_vertices.hh.

10.298.4.8 `template<typename G , typename F > unsigned mln::p_vertices< G, F >::nsites ()`
`const [inline]`

Return The number of points (sites) of the set, i.e., the number of *vertices*.

Required by the `mln::Point_Set` concept.

Definition at line 272 of file p_vertices.hh.

References `mln::p_vertices< G, F >::nvertices()`.

10.298.4.9 `template<typename G , typename F > unsigned mln::p_vertices< G, F >::nvertices (`
`) const [inline]`

Return The number of vertices in the graph.

Definition at line 280 of file p_vertices.hh.

Referenced by `mln::p_vertices< G, F >::nsites()`.

10.298.4.10 `template<typename G , typename F > F::result mln::p_vertices< G, F >::operator()`
`(const psite & p) const [inline]`

Return the value associated to an element of this site set.

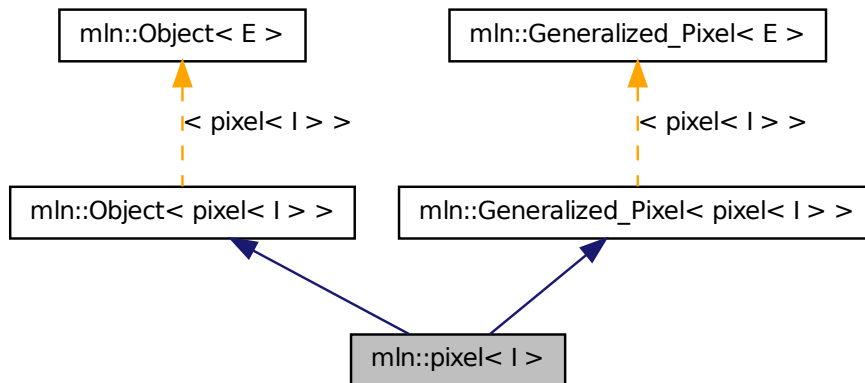
Definition at line 349 of file p_vertices.hh.

10.299 mln::pixel< I > Struct Template Reference

Generic pixel class.

```
#include <pixel.hh>
```

Inheritance diagram for mln::pixel< I >:



Public Member Functions

- void [change_to](#) (const typename I::psite &p)
Change the pixel to the one at point p.
- bool [is_valid](#) () const
Test if this pixel is valid.
- [pixel](#) (I &image)
Constructor.
- [pixel](#) (I &image, const typename I::psite &p)
Constructor.

10.299.1 Detailed Description

```
template<typename I> struct mln::pixel< I >
```

Generic pixel class. The parameter is I the type of the image it belongs to.

Definition at line 50 of file core/pixel.hh.

10.299.2 Constructor & Destructor Documentation

10.299.2.1 `template<typename I> mln::pixel< I >::pixel (I & image) [inline]`

Constructor.

Definition at line 75 of file core/pixel.hh.

10.299.2.2 `template<typename I> mln::pixel< I >::pixel (I & image, const typename I::psite & p) [inline]`

Constructor.

Definition at line 82 of file core/pixel.hh.

References `mln::pixel< I >::change_to()`.

10.299.3 Member Function Documentation

10.299.3.1 `template<typename I> void mln::pixel< I >::change_to (const typename I::psite & p) [inline]`

Change the pixel to the one at point p.

Definition at line 92 of file core/pixel.hh.

Referenced by `mln::pixel< I >::pixel()`.

10.299.3.2 `template<typename I> bool mln::pixel< I >::is_valid () const [inline]`

Test if this pixel is valid.

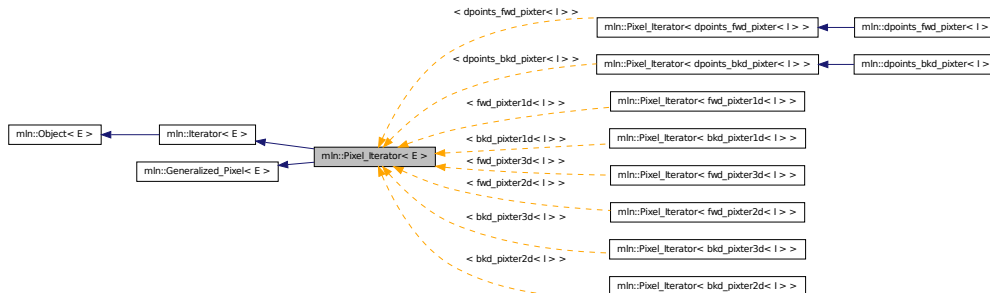
Definition at line 101 of file core/pixel.hh.

10.300 mln::Pixel_Iterator< E > Struct Template Reference

Base class for the implementation of pixel iterator classes.

```
#include <pixel_iterator.hh>
```

Inheritance diagram for `mln::Pixel_Iterator< E >`:



Public Member Functions

- void [next](#) ()

Go to the next element.

10.300.1 Detailed Description

`template<typename E> struct mln::Pixel_Iterator< E >`

Base class for the implementation of pixel iterator classes. An iterator on pixels is an iterator that is bound to a particular image and that browses over a set of image pixels.

See also

[mln::doc::Pixel_Iterator](#) for a complete documentation of this class contents.

Definition at line 51 of file `pixel_iterator.hh`.

10.300.2 Member Function Documentation

10.300.2.1 `template<typename E > void mln::Iterator< E >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

Definition at line 102 of file `iterator.hh`.

10.301 `mln::plain< I >` Class Template Reference

Prevents an image from sharing its data.

```
#include <plain.hh>
```

Inherits `image_identity< I, I::domain_t, plain< I > >`.

Public Types

- typedef [plain](#)< tag::image_< I > > [skeleton](#)

Skeleton.

Public Member Functions

- `operator I () const`
Conversion into an image with type I .
- `plain< I > & operator= (const I &ima)`
Assignment operator from an image ima .
- `plain< I > & operator= (const plain< I > &rhs)`
Assignment operator.
- `plain (const plain< I > &rhs)`
Copy constructor.
- `plain ()`
Constructor without argument.
- `plain (const I &ima)`
Copy constructor from an image ima .

10.301.1 Detailed Description

`template<typename I> class mln::plain< I >`

Prevents an image from sharing its data. While assigned to another image, its data is duplicated.

Definition at line 82 of file plain.hh.

10.301.2 Member Typedef Documentation

10.301.2.1 `template<typename I> typedef plain< tag::image_<I> > mln::plain< I >::skeleton`

Skeleton.

Definition at line 93 of file plain.hh.

10.301.3 Constructor & Destructor Documentation

10.301.3.1 `template<typename I> mln::plain< I >::plain () [inline]`

Constructor without argument.

Definition at line 141 of file plain.hh.

10.301.3.2 `template<typename I> mln::plain< I >::plain (const plain< I > & rhs) [inline]`

Copy constructor.

Definition at line 147 of file plain.hh.

10.301.3.3 `template<typename I > mln::plain< I >::plain (const I & ima) [inline]`

Copy constructor from an image *ima*.

Definition at line 156 of file plain.hh.

10.301.4 Member Function Documentation**10.301.4.1** `template<typename I > mln::plain< I >::operator I () const [inline]`

Conversion into an image with type *I*.

Definition at line 197 of file plain.hh.

References mln::duplicate().

10.301.4.2 `template<typename I > plain< I > & mln::plain< I >::operator= (const plain< I > & rhs) [inline]`

Assignment operator.

Definition at line 174 of file plain.hh.

10.301.4.3 `template<typename I > plain< I > & mln::plain< I >::operator= (const I & ima) [inline]`

Assignment operator from an image *ima*.

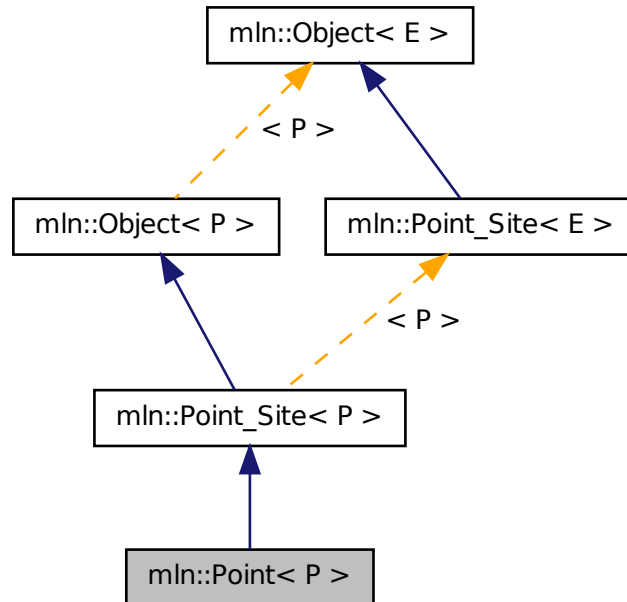
Definition at line 187 of file plain.hh.

10.302 mln::Point< P > Struct Template Reference

Base class for implementation of point classes.

```
#include <point.hh>
```

Inheritance diagram for `mln::Point< P >`:



Public Types

- typedef `P` `point`

The associated point type is itself.

Public Member Functions

- `const P & to_point () const`

It is a `Point` so it returns itself.

Related Functions

(Note that these are not member functions.)

- `template<typename P , typename D >`
`P & operator+= (Point< P > &p, const Dpoint< D > &dp)`

Shift a point by a delta-point `dp`.

- `template<typename P , typename D >`
`P & operator-= (Point< P > &p, const Dpoint< D > &dp)`
Shift a point by the negate of a delta-point dp.
- `template<typename P , typename D >`
`P & operator/ (Point< P > &p, const value::Scalar< D > &dp)`
Divide a point by a scalar s.

10.302.1 Detailed Description

`template<typename P> struct mln::Point< P >`

Base class for implementation of point classes. A point is an element of a space.

For instance, `mln::point2d` is the type of elements defined on the discrete square grid of the 2D plane.

Definition at line 62 of file `concept/point.hh`.

10.302.2 Member Typedef Documentation

10.302.2.1 `template<typename P > typedef P mln::Point< P >::point`

The associated point type is itself.

Definition at line 66 of file `concept/point.hh`.

10.302.3 Member Function Documentation

10.302.3.1 `template<typename P > const P & mln::Point< P >::to_point () const [inline]`

It is a `Point` so it returns itself.

Definition at line 130 of file `concept/point.hh`.

10.302.4 Friends And Related Function Documentation

10.302.4.1 `template<typename P , typename D > P & operator+= (Point< P > & p, const Dpoint< D > & dp) [related]`

Shift a point by a delta-point dp.

Parameters

[in, out] *p* The targeted point.

[in] *dp* A delta-point.

Returns

A reference to the point *p* once translated by *dp*.

Precondition

The type of `dp` has to be compatible with the type of `p`.

Definition at line 137 of file `concept/point.hh`.

10.302.4.2 `template<typename P , typename D > P & operator-= (Point< P > & p, const Dpoint< D > & dp) [related]`

Shift a point by the negate of a delta-point `dp`.

Parameters

[in, out] `p` The targeted point.

[in] `dp` A delta-point.

Returns

A reference to the point `p` once translated by `- dp`.

Precondition

The type of `dp` has to be compatible with the type of `p`.

Definition at line 149 of file `concept/point.hh`.

10.302.4.3 `template<typename P , typename D > P & operator/ (Point< P > & p, const value::Scalar< D > & dp) [related]`

Divide a point by a scalar `s`.

Parameters

[in, out] `p` The targeted point.

[in] `dp` A scalar.

Returns

A reference to the point `p` once divided by `s`.

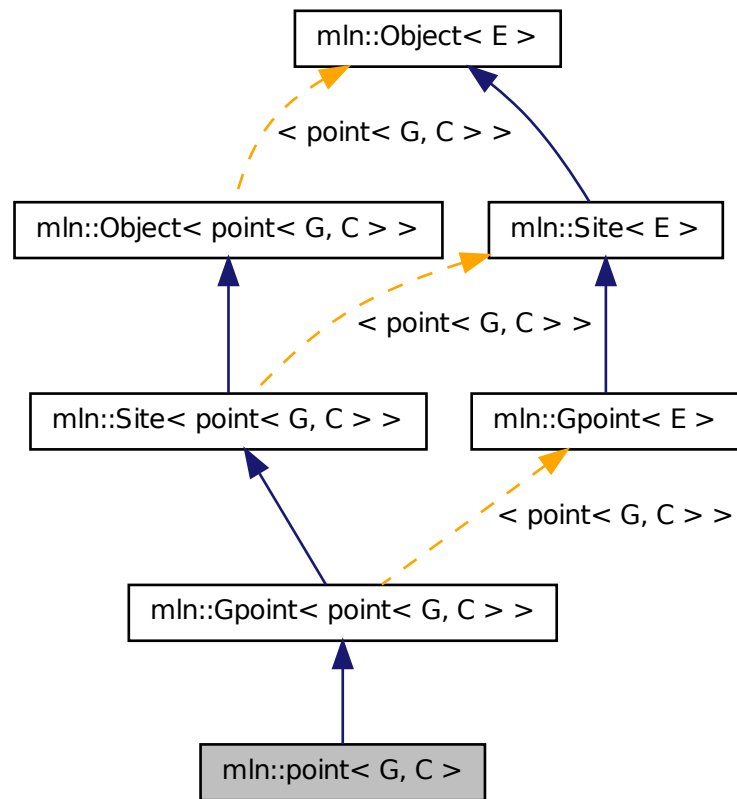
Definition at line 163 of file `concept/point.hh`.

10.303 `mln::point< G, C >` Struct Template Reference

Generic point class.

```
#include <point.hh>
```

Inheritance diagram for mln::point< G, C >:



Public Types

- enum { `dim = G::dim` }
- typedef `C coord`
Coordinate associated type.
- typedef `dpoint< G, C > delta`
Delta associated type.
- typedef `dpoint< G, C > dpsite`
DPsite associated type.
- typedef `G grid`
Grid associated type.
- typedef `mln::algebra::h_vec< G::dim, float > h_vec`

Algebra hexagonal vector (hvec) associated type.

- `typedef mln::algebra::vec< G::dim, float > vec`
Algebra vector (vec) associated type.

Public Member Functions

- `const C & last_coord () const`
Read-only access to the last coordinate.
- `C & last_coord ()`
Read-write access to the last coordinate.
- `point< G, C > & operator+= (const delta &dp)`
Shifting by dp.
- `point< G, C > & operator-= (const delta &dp)`
Shifting by the inverse of dp.
- `C & operator[] (unsigned i)`
Read-write access to the i-th coordinate value.
- `const C & operator[] (unsigned i) const`
Read-only access to the i-th coordinate value.
- `template<typename F >`
`point (const Function_v2v< F > &f)`
Constructor; coordinates are set by function f.
- `point ()`
Constructor without argument.
- `template<typename C2 >`
`point (const mln::algebra::vec< dim, C2 > &v)`
Constructor from an algebra vector.
- `void set_all (C c)`
Set all coordinates to the value c.
- `h_vec to_h_vec () const`
Transform to point in homogeneous coordinate system.
- `vec to_vec () const`
Explicit conversion towards mln::algebra::vec.
- `point (C ind)`
- `point (const literal::origin_t &)`
Constructors/assignments with literals.

Static Public Member Functions

- static const `point< G, C >` & `minus_infty` ()
Point with all coordinates set to the minimum value.
- static const `point< G, C >` & `plus_infty` ()
Point with all coordinates set to the maximum value.

Static Public Attributes

- static const `point< G, C >` `origin` = `all_to(0)`
Origin point (all coordinates are 0).

10.303.1 Detailed Description

`template<typename G, typename C> struct mln::point< G, C >`

Generic point class. Parameters are n the dimension of the space and C the coordinate type in this space.

Definition at line 108 of file point.hh.

10.303.2 Member Typedef Documentation

10.303.2.1 `template<typename G, typename C> typedef C mln::point< G, C >::coord`

Coordinate associated type.

Definition at line 131 of file point.hh.

10.303.2.2 `template<typename G, typename C> typedef dpoint<G,C> mln::point< G, C >::delta`

Delta associated type.

Definition at line 125 of file point.hh.

10.303.2.3 `template<typename G, typename C> typedef dpoint<G,C> mln::point< G, C >::dpsite`

DPsite associated type.

Definition at line 128 of file point.hh.

10.303.2.4 `template<typename G, typename C> typedef G mln::point< G, C >::grid`

Grid associated type.

Definition at line 122 of file point.hh.

10.303.2.5 `template<typename G, typename C> typedef mln::algebra::h_vec<G::dim, float>
mln::point< G, C >::h_vec`

Algebra hexagonal vector (hvec) associated type.

Definition at line 137 of file point.hh.

10.303.2.6 `template<typename G, typename C> typedef mln::algebra::vec<G::dim, float>
mln::point< G, C >::vec`

Algebra vector (vec) associated type.

Definition at line 134 of file point.hh.

10.303.3 Member Enumeration Documentation

10.303.3.1 `template<typename G, typename C> anonymous enum`

Enumerator:

dim Dimension of the space.

Invariant

`dim > 0`

Definition at line 119 of file point.hh.

10.303.4 Constructor & Destructor Documentation

10.303.4.1 `template<typename G , typename C > mln::point< G, C >::point () [inline]`

Constructor without argument.

Definition at line 420 of file point.hh.

10.303.4.2 `template<typename G , typename C > template<typename C2 > mln::point< G, C
>::point (const mln::algebra::vec< dim, C2 > & v) [inline]`

Constructor from an algebra vector.

Definition at line 427 of file point.hh.

10.303.4.3 `template<typename G , typename C> mln::point< G, C >::point (C ind)
[inline, explicit]`

Constructors with different numbers of arguments (coordinates) w.r.t. the dimension.

Definition at line 443 of file point.hh.

10.303.4.4 `template<typename G , typename C> mln::point< G, C >::point (const
literal::origin_t &) [inline]`

Constructors/assignments with literals.

Definition at line 481 of file point.hh.

10.303.4.5 `template<typename G , typename C > template<typename F > mln::point< G, C >::point (const Function_v2v< F > & f) [inline]`

Constructor; coordinates are set by function *f*.

Definition at line 471 of file point.hh.

10.303.5 Member Function Documentation

10.303.5.1 `template<typename G , typename C > const C & mln::point< G, C >::last_coord () const [inline]`

Read-only access to the last coordinate.

Definition at line 402 of file point.hh.

Referenced by `mln::p_run< P >::end()`, `mln::p_run< P >::operator[]()`, and `mln::debug::put_word()`.

10.303.5.2 `template<typename G , typename C > C & mln::point< G, C >::last_coord () [inline]`

Read-write access to the last coordinate.

Definition at line 410 of file point.hh.

10.303.5.3 `template<typename G , typename C > const point< G, C > & mln::point< G, C >::minus_infty () [inline, static]`

[Point](#) with all coordinates set to the minimum value.

Definition at line 627 of file point.hh.

10.303.5.4 `template<typename G , typename C > point< G, C > & mln::point< G, C >::operator+=(const delta & dp) [inline]`

Shifting by *dp*.

Definition at line 544 of file point.hh.

10.303.5.5 `template<typename G , typename C > point< G, C > & mln::point< G, C >::operator-=(const delta & dp) [inline]`

Shifting by the inverse of *dp*.

Definition at line 554 of file point.hh.

10.303.5.6 `template<typename G , typename C > C & mln::point< G, C >::operator[] (unsigned i) [inline]`

Read-write access to the *i*-th coordinate value.

Parameters

[in] *i* The coordinate index.

Precondition

$i < \text{dim}$

Definition at line 393 of file point.hh.

10.303.5.7 `template<typename G , typename C > const C & mln::point< G, C >::operator[] (unsigned i) const [inline]`

Read-only access to the *i*-th coordinate value.

Parameters

[in] *i* The coordinate index.

Precondition

$i < \text{dim}$

Definition at line 385 of file point.hh.

10.303.5.8 `template<typename G , typename C > const point< G, C > & mln::point< G, C >::plus_infty () [inline, static]`

[Point](#) with all coordinates set to the maximum value.

Definition at line 618 of file point.hh.

10.303.5.9 `template<typename G , typename C> void mln::point< G, C >::set_all (C c) [inline]`

Set all coordinates to the value *c*.

Definition at line 533 of file point.hh.

10.303.5.10 `template<typename G , typename C > point< G, C >::h_vec mln::point< G, C >::to_h_vec () const [inline]`

Transform to point in homogeneous coordinate system.

Definition at line 592 of file point.hh.

10.303.5.11 `template<typename G , typename C > point< G, C >::vec mln::point< G, C >::to_vec () const [inline]`

Explicit conversion towards `mln::algebra::vec`.

Definition at line 571 of file point.hh.

10.303.6 Member Data Documentation

10.303.6.1 `template<typename G, typename C> const point< G, C > mln::point< G, C >::origin = all_to(0) [static]`

Origin point (all coordinates are 0).

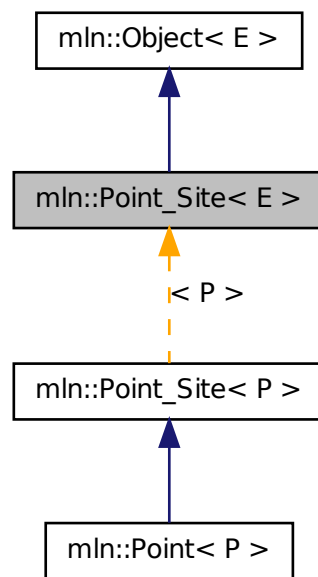
Definition at line 192 of file point.hh.

10.304 mln::Point_Site< E > Struct Template Reference

Base class for implementation classes of the notion of "point site".

```
#include <point_site.hh>
```

Inheritance diagram for mln::Point_Site< E >:



Related Functions

(Note that these are not member functions.)

- `template<typename L, typename R >`
`L::dpoint operator- (const Point_Site< L > &lhs, const Point_Site< R > &rhs)`

Difference between a couple of point site lhs and rhs.

- `template<typename P >`
`std::ostream & operator<< (std::ostream &ostr, const Point_Site< P > &p)`
Print a point site `p` into the output stream `ostr`.
- `template<typename L , typename R >`
`bool operator== (const Point_Site< L > &lhs, const Point_Site< R > &rhs)`
Equality comparison between a couple of point site `lhs` and `rhs`.
- `template<typename P , typename D >`
`P::point operator+ (const Point_Site< P > &p, const Delta_Point_Site< D > &dp)`
Add a delta-point `rhs` to a point site `lhs`.
- `template<typename P , typename D >`
`P::point operator- (const Point_Site< P > &p, const Delta_Point_Site< D > &dp)`
`}`

10.304.1 Detailed Description

`template<typename E> struct mln::Point_Site< E >`

Base class for implementation classes of the notion of "point site". A point site ("psite" for short) is an object that allows an efficient access to data associated with a point. A point site is either a point or designates a point.

When a point site is not really a point, it is automatically convertible to the point it designates.

Let us take the example of a 2D image encoded as an array of runs of values. With a point, a pair (row index, column index), retrieving the corresponding pixel value would mean to browse the array of runs to find the value location. That would not be efficient. Conversely, a point site dedicated to this image structure allows for value access in constant time; precisely the proper point site is a pair (index of run, index within the run).

Definition at line 106 of file `point_site.hh`.

10.304.2 Friends And Related Function Documentation

10.304.2.1 `template<typename P , typename D > P::point operator+ (const Point_Site< P > & p, const Delta_Point_Site< D > & dp)` [[related](#)]

Add a delta-point `rhs` to a point site `lhs`.

Parameters

[in] `p` A point site.

[in] `dp` A delta-point.

The type of `dp` has to be compatible with the type of `p`.

Returns

A point (temporary object).

See also

[mln::Delta_Point_Site](#)

Definition at line 282 of file point_site.hh.

10.304.2.2 `template<typename P , typename D > P::point operator- (const Point_Site< P > & p, const Delta_Point_Site< D > & dp) [related]`

}

Subtract a delta-point `dp` to a point site `p`.

Parameters

[in] *p* A point site.

[in] *dp* A delta-point.

The type of `dp` has to compatible with the type of `p`.

Returns

A point (temporary object).

See also

[mln::Dpoint](#)

[mln::Delta_Point_Site](#)

Definition at line 296 of file point_site.hh.

10.304.2.3 `template<typename L , typename R > L::dpoint operator- (const Point_Site< L > & lhs, const Point_Site< R > & rhs) [related]`

Difference between a couple of point site `lhs` and `rhs`.

Parameters

[in] *lhs* A first point site.

[in] *rhs* A second point site.

Warning

There is no type promotion in Milena so the client has to make sure that both points are defined with the same type of coordinates.

Precondition

Both `lhs` and `rhs` have to be defined on the same topology and with the same type of coordinates; otherwise this test does not compile.

Postcondition

The result, `dp`, is such as `lhs == rhs + dp`.

Returns

A delta point (temporary object).

See also

[mln::Delta_Point_Site](#)

Definition at line 267 of file point_site.hh.

10.304.2.4 `template<typename P> std::ostream & operator<< (std::ostream & ostr, const Point_Site< P> & p)` **[related]**

Print a point site `p` into the output stream `ostr`.

Parameters

[in, out] `ostr` An output stream.

[in] `p` A point site.

Returns

The modified output stream `ostr`.

Definition at line 311 of file point_site.hh.

10.304.2.5 `template<typename L, typename R> bool operator== (const Point_Site< L> & lhs, const Point_Site< R> & rhs)` **[related]**

Equality comparison between a couple of point site `lhs` and `rhs`.

Parameters

[in] `lhs` A first point site.

[in] `rhs` A second point site.

Precondition

Both `lhs` and `rhs` have to be defined on the same topology; otherwise this test does not compile.

Returns

True if both point sites have the same coordinates, otherwise false.

Definition at line 251 of file point_site.hh.

10.305 `mln::Point_Site< void>` Struct Template Reference

[Point](#) site category flag type.

```
#include <point_site.hh>
```


10.305.1 Detailed Description

```
template<> struct mln::Point_Site< void >
```

[Point](#) site category flag type.

Definition at line 80 of file point_site.hh.

10.306 mln::Proxy< E > Struct Template Reference

Base class for implementation classes of the notion of "proxy".

```
#include <proxy.hh>
```

Inherits [mln::Object< E >](#).

Inherited by [mln::Accumulator< E >](#), [mln::internal::graph_iter_base< G, Elt, E >](#), [mln::internal::nbh_iterator_base< G, C, Elt, E >](#), and [mln::Site_Proxy< E >](#).

10.306.1 Detailed Description

```
template<typename E> struct mln::Proxy< E >
```

Base class for implementation classes of the notion of "proxy".

Definition at line 232 of file core/concept/proxy.hh.

10.307 mln::Proxy< void > Struct Template Reference

[Proxy](#) category flag type.

```
#include <proxy.hh>
```

10.307.1 Detailed Description

```
template<> struct mln::Proxy< void >
```

[Proxy](#) category flag type.

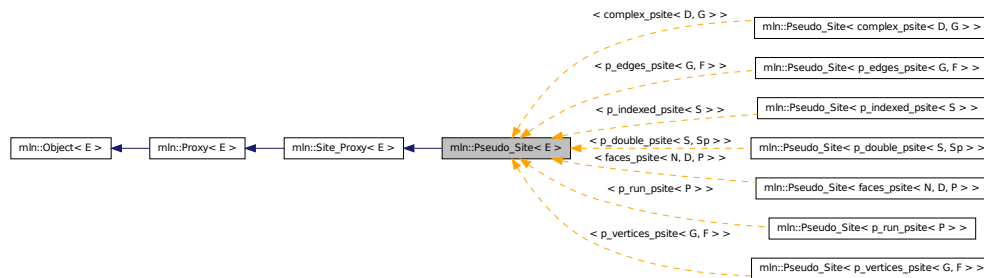
Definition at line 222 of file core/concept/proxy.hh.

10.308 mln::Pseudo_Site< E > Struct Template Reference

Base class for implementation classes of the notion of "pseudo site".

```
#include <pseudo_site.hh>
```

Inheritance diagram for `mln::Pseudo_Site< E >`:



10.308.1 Detailed Description

```
template<typename E> struct mln::Pseudo_Site< E >
```

Base class for implementation classes of the notion of "pseudo site". FIXME: Explain...

Definition at line 64 of file `pseudo_site.hh`.

10.309 mln::Pseudo_Site< void > Struct Template Reference

[Pseudo_Site](#) category flag type.

```
#include <pseudo_site.hh>
```

10.309.1 Detailed Description

```
template<> struct mln::Pseudo_Site< void >
```

[Pseudo_Site](#) category flag type.

Definition at line 52 of file `pseudo_site.hh`.

10.310 mln::pw::image< F, S > Class Template Reference

A generic point-wise image implementation.

```
#include <image.hh>
```

Inherits `image_base< F, S, image< F, S >>`.

Public Types

- typedef `image< tag::function_< F >, tag::domain_< S >>` [skeleton](#)

Skeleton.

Public Member Functions

- `image ()`
Constructor without argument.
- `image (const Function_v2v< F > &f, const Site_Set< S > &ps)`
Constructor.

10.310.1 Detailed Description

`template<typename F, typename S> class mln::pw::image< F, S >`

A generic point-wise image implementation. Parameter `F` is a function restricting the domain. Parameter `S` is the domain type.

Definition at line 92 of file `pw/image.hh`.

10.310.2 Member Typedef Documentation

10.310.2.1 `template<typename F, typename S> typedef image< tag::function_<F>, tag::domain_<S> > mln::pw::image< F, S >::skeleton`

Skeleton.

Definition at line 99 of file `pw/image.hh`.

10.310.3 Constructor & Destructor Documentation

10.310.3.1 `template<typename F, typename S > mln::pw::image< F, S >::image ()`
`[inline]`

Constructor without argument.

Definition at line 169 of file `pw/image.hh`.

10.310.3.2 `template<typename F, typename S > mln::pw::image< F, S >::image (const Function_v2v< F > & f, const Site_Set< S > & ps)`
`[inline]`

Constructor.

Definition at line 175 of file `pw/image.hh`.

10.311 mln::registration::closest_point_basic< P > Class Template Reference

Closest point functor based on map distance.

```
#include <icp.hh>
```

10.311.1 Detailed Description

`template<typename P> class mln::registration::closest_point_basic< P >`

Closest point functor based on map distance.

Definition at line 240 of file icp.hh.

10.312 mln::registration::closest_point_with_map< P > Class Template Reference

Closest point functor based on map distance.

```
#include <icp.hh>
```

10.312.1 Detailed Description

`template<typename P> class mln::registration::closest_point_with_map< P >`

Closest point functor based on map distance.

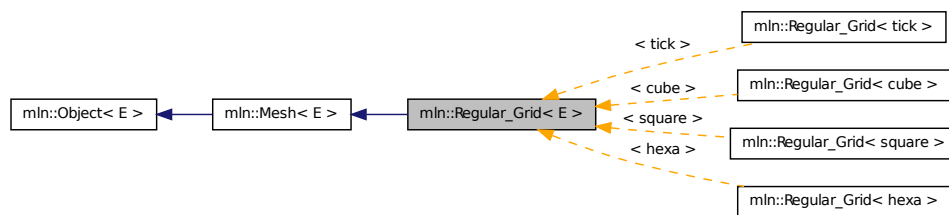
Definition at line 145 of file icp.hh.

10.313 mln::Regular_Grid< E > Struct Template Reference

Base class for implementation classes of regular grids.

```
#include <regular_grid.hh>
```

Inheritance diagram for `mln::Regular_Grid< E >`:



10.313.1 Detailed Description

`template<typename E> struct mln::Regular_Grid< E >`

Base class for implementation classes of regular grids.

Definition at line 42 of file regular_grid.hh.

10.314 mln::safe_image< I > Class Template Reference

Makes an image accessible at undefined location.

```
#include <safe.hh>
```

Inherits image_identity< I, I::domain_t, safe_image< I > >.

Public Types

- typedef [safe_image](#)< tag::image_< I > > [skeleton](#)
Skeleton.

Public Member Functions

- [operator safe_image](#)< const I > () const
Const promotion via conversion.

10.314.1 Detailed Description

```
template<typename I> class mln::safe_image< I >
```

Makes an image accessible at undefined location.

Definition at line 83 of file safe.hh.

10.314.2 Member Typedef Documentation

10.314.2.1 `template<typename I> typedef safe_image< tag::image_<I> > mln::safe_image< I >::skeleton`

Skeleton.

Definition at line 88 of file safe.hh.

10.314.3 Member Function Documentation

10.314.3.1 `template<typename I> mln::safe_image< I >::operator safe_image< const I > () const [inline]`

Const promotion via conversion.

Definition at line 195 of file safe.hh.

10.315 mln::select::p_of< P > Struct Template Reference

Structure [p_of](#).

```
#include <pix.hh>
```

10.315.1 Detailed Description

`template<typename P> struct mln::select::p_of< P >`

Structure [p_of](#).

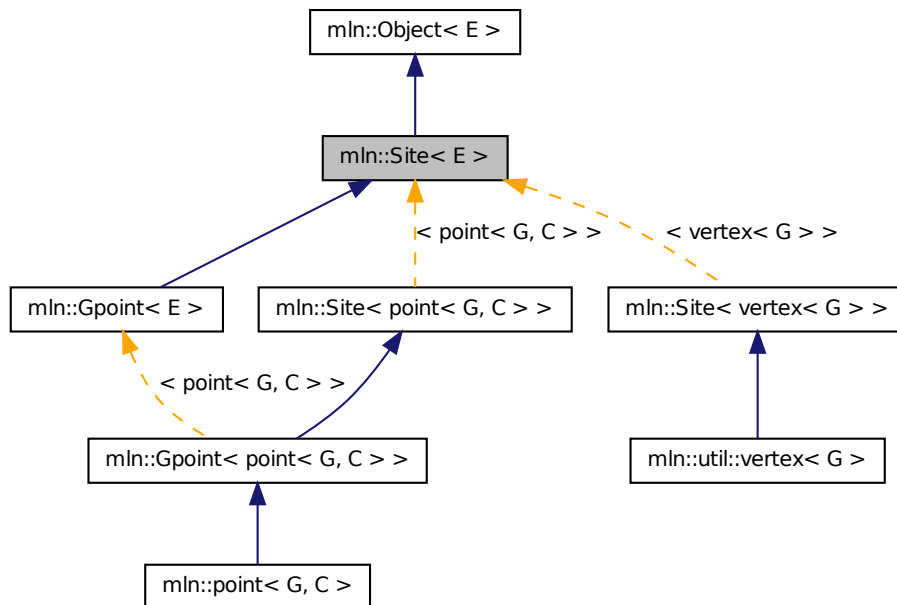
Definition at line 52 of file util/pix.hh.

10.316 mln::Site< E > Struct Template Reference

Base class for classes that are explicitly sites.

```
#include <site.hh>
```

Inheritance diagram for `mln::Site< E >`:



10.316.1 Detailed Description

`template<typename E> struct mln::Site< E >`

Base class for classes that are explicitly sites.

Definition at line 55 of file site.hh.

10.317 mln::Site< void > Struct Template Reference

[Site](#) category flag type.

```
#include <site.hh>
```

10.317.1 Detailed Description

template<> struct mln::Site< void >

[Site](#) category flag type.

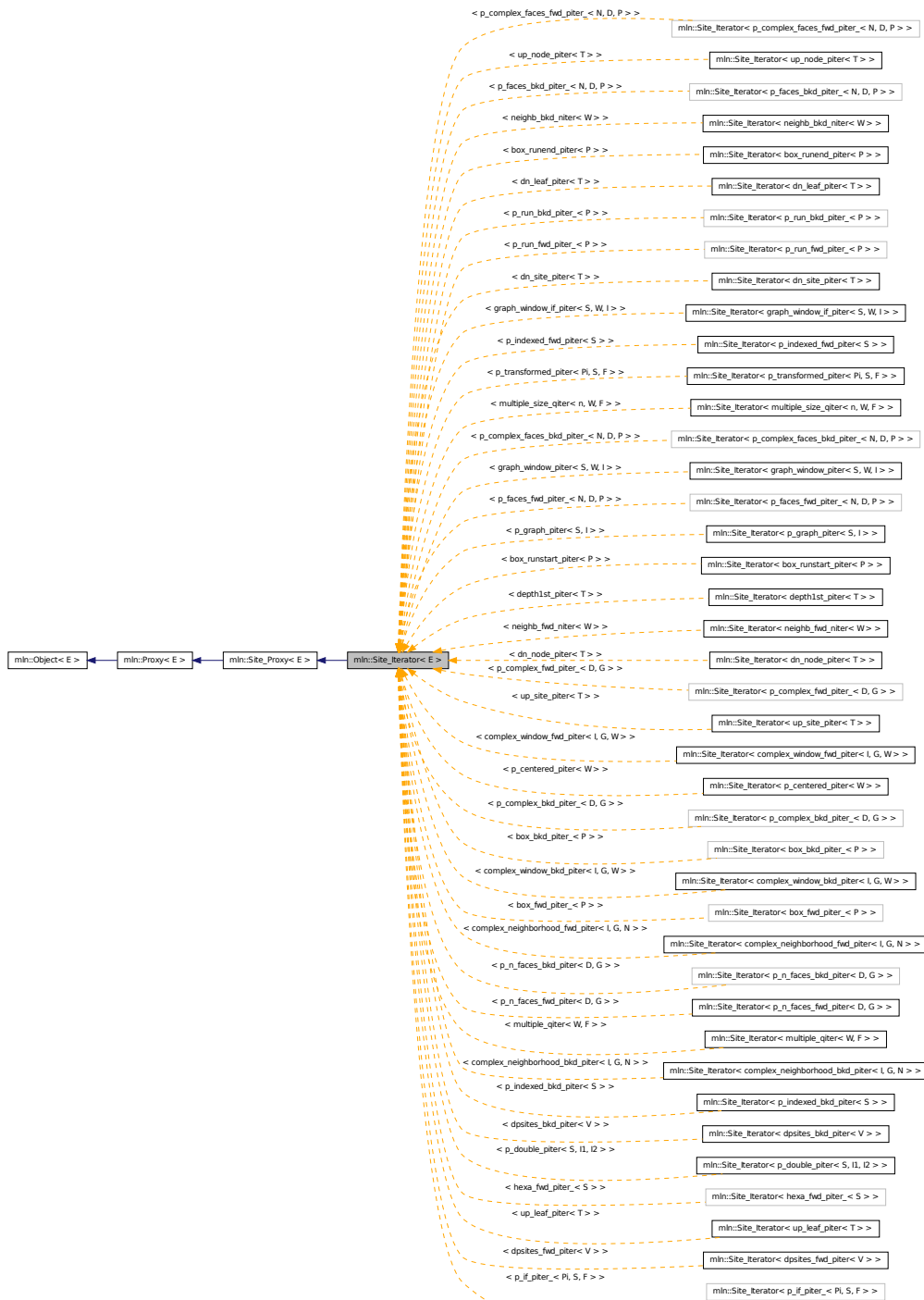
Definition at line 46 of file site.hh.

10.318 mln::Site_Iterator< E > Struct Template Reference

Base class for implementation of classes of iterator on points.

```
#include <site_iterator.hh>
```

Inheritance diagram for mln::Site_Iterator< E >:



Public Member Functions

- void [next](#) ()
Go to the next element.

10.318.1 Detailed Description

`template<typename E> struct mln::Site_Iterator< E >`

Base class for implementation of classes of iterator on points. An iterator on points is an iterator that browse over a set of points.

See also

[mln::doc::Site_Iterator](#) for a complete documentation of this class contents.

Definition at line 53 of file `site_iterator.hh`.

10.318.2 Member Function Documentation

10.318.2.1 `template<typename E > void mln::Site_Iterator< E >::next () [inline]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

Definition at line 92 of file `site_iterator.hh`.

10.319 mln::Site_Proxy< E > Struct Template Reference

Base class for implementation classes of the notion of "site proxy".

```
#include <site_proxy.hh>
```

Inherits [mln::Proxy< E >](#).

Inherited by [mln::Pseudo_Site< E >](#), and [mln::Site_Iterator< E >](#).

10.319.1 Detailed Description

`template<typename E> struct mln::Site_Proxy< E >`

Base class for implementation classes of the notion of "site proxy". FIXME: Explain...

Definition at line 61 of file `site_proxy.hh`.

10.320 mln::Site_Proxy< void > Struct Template Reference

[Site_Proxy](#) category flag type.

```
#include <site_proxy.hh>
```

10.320.1 Detailed Description

`template<> struct mln::Site_Proxy< void >`

[Site_Proxy](#) category flag type.

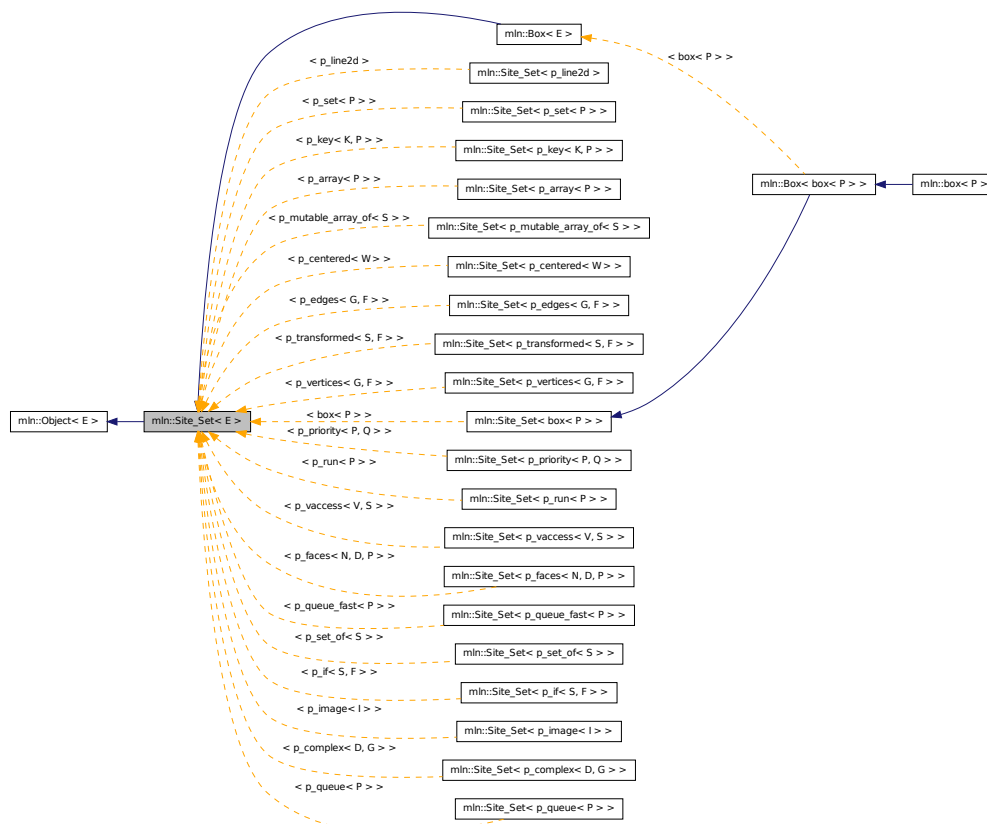
Definition at line 49 of file `site_proxy.hh`.

10.321 mln::Site_Set< E > Struct Template Reference

Base class for implementation classes of site sets.

```
#include <site_set.hh>
```

Inheritance diagram for `mln::Site_Set< E >`:



Related Functions

(Note that these are not member functions.)

- `template<typename SI , typename Sr >`
`p_set< typename SI::site > diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic difference of lhs and rhs.
- `template<typename SI , typename Sr >`
`p_set< typename SI::site > inter (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Intersection between a couple of point sets.
- `template<typename SI , typename Sr >`
`bool operator< (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Strict inclusion test between site sets lhs and rhs.
- `template<typename S >`
`std::ostream & operator<< (std::ostream &ostr, const Site_Set< S > &set)`
Print a site set set into the output stream ostr.
- `template<typename SI , typename Sr >`
`bool operator<= (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Inclusion test between site sets lhs and rhs.
- `template<typename SI , typename Sr >`
`bool operator== (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Equality test between site sets lhs and rhs.
- `template<typename SI , typename Sr >`
`p_set< typename SI::site > sym_diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic symmetrical difference of lhs and rhs.
- `template<typename SI , typename Sr >`
`p_set< typename SI::site > uni (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Union of a couple of point sets.
- `template<typename S >`
`p_set< typename S::site > unique (const Site_Set< S > &s)`
Give the unique set of s.

10.321.1 Detailed Description

`template<typename E> struct mln::Site_Set< E >`

Base class for implementation classes of site sets.

See also

[mln::doc::Site_Set](#) for a complete documentation of this class contents.

Definition at line 65 of file mln/core/concept/site_set.hh.

10.321.2 Friends And Related Function Documentation

10.321.2.1 `template<typename SI, typename Sr > p_set< typename SI::site > diff (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related]`

Set theoretic difference of `lhs` and `rhs`.

Definition at line 66 of file `set/diff.hh`.

10.321.2.2 `template<typename SI, typename Sr > p_set< typename SI::site > inter (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related]`

Intersection between a couple of point sets.

Definition at line 62 of file `set/inter.hh`.

10.321.2.3 `template<typename SI, typename Sr > bool operator< (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related]`

Strict inclusion test between site sets `lhs` and `rhs`.

Parameters

[in] *lhs* A site set (strictly included?).

[in] *rhs* Another site set (includer?).

Definition at line 479 of file `operators.hh`.

10.321.2.4 `template<typename S > std::ostream & operator<< (std::ostream & ostr, const Site_Set< S > & set) [related]`

Print a site set `set` into the output stream `ostr`.

Parameters

[in, out] *ostr* An output stream.

[in] *set* A site set.

Returns

The modified output stream `ostr`.

Definition at line 505 of file `operators.hh`.

10.321.2.5 `template<typename SI, typename Sr > bool operator<= (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related]`

Inclusion test between site sets `lhs` and `rhs`.

Parameters

[in] *lhs* A site set (included?).

[in] *rhs* Another site set (includer?).

Definition at line 491 of file `operators.hh`.

10.321.2.6 `template<typename SI , typename Sr > bool operator==(const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related]`

Equality test between site sets `lhs` and `rhs`.

Parameters

[in] `lhs` A site set.

[in] `rhs` Another site set.

Definition at line 467 of file operators.hh.

10.321.2.7 `template<typename SI , typename Sr > p_set< typename SI::site > sym_diff (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related]`

Set theoretic symmetrical difference of `lhs` and `rhs`.

Definition at line 65 of file sym_diff.hh.

10.321.2.8 `template<typename SI , typename Sr > p_set< typename SI::site > uni (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs) [related]`

Union of a couple of point sets.

Definition at line 61 of file uni.hh.

10.321.2.9 `template<typename S > p_set< typename S::site > unique (const Site_Set< S > & s) [related]`

Give the unique set of `s`.

Definition at line 61 of file unique.hh.

10.322 mln::Site_Set< void > Struct Template Reference

[Site_Set](#) category flag type.

```
#include <site_set.hh>
```

10.322.1 Detailed Description

`template<> struct mln::Site_Set< void >`

[Site_Set](#) category flag type.

Definition at line 54 of file mln/core/concept/site_set.hh.

10.323 mln::slice_image< I > Struct Template Reference

2D image extracted from a slice of a 3D image.

```
#include <slice_image.hh>
```

Inherits `image_domain_morpher< I, box2d, slice_image< I > >`.

Public Types

- typedef `slice_image< tag::image_< I > > skeleton`
Skeleton.

Public Member Functions

- const `box2d & domain () const`
Give the definition domain.
- `operator slice_image< const I > () const`
Const promotion via conversion.
- `I::rvalue operator() (const point2d &p) const`
Read-only access to the image value located at point p.
- `internal::morpher_lvalue_< I >::ret operator() (const point2d &p)`
Read-write access to the image value located at point p.
- `def::coord sli () const`
Give the slice number.
- `slice_image ()`
Constructor without argument.
- `slice_image (I &ima, def::coord sli)`
Constructor from an image ima and a predicate f .

10.323.1 Detailed Description

```
template<typename I> struct mln::slice_image< I >
```

2D image extracted from a slice of a 3D image.

Definition at line 95 of file `slice_image.hh`.

10.323.2 Member Typedef Documentation

10.323.2.1 `template<typename I> typedef slice_image< tag::image_<I> > mln::slice_image< I >::skeleton`

Skeleton.

Definition at line 100 of file `slice_image.hh`.

10.323.3 Constructor & Destructor Documentation

10.323.3.1 `template<typename I > mln::slice_image< I >::slice_image () [inline]`

Constructor without argument.

Definition at line 193 of file slice_image.hh.

10.323.3.2 `template<typename I > mln::slice_image< I >::slice_image (I & ima, def::coord sli) [inline]`

Constructor from an image *ima* and a predicate *f*.

Definition at line 199 of file slice_image.hh.

10.323.4 Member Function Documentation

10.323.4.1 `template<typename I > const box2d & mln::slice_image< I >::domain () const [inline]`

Give the definition domain.

Definition at line 216 of file slice_image.hh.

10.323.4.2 `template<typename I > mln::slice_image< I >::operator slice_image< const I > () const [inline]`

Const promotion via conversion.

Definition at line 255 of file slice_image.hh.

10.323.4.3 `template<typename I > internal::morpher_lvalue_< I >::ret mln::slice_image< I >::operator() (const point2d & p) [inline]`

Read-write access to the image value located at point *p*.

Definition at line 245 of file slice_image.hh.

10.323.4.4 `template<typename I > I::rvalue mln::slice_image< I >::operator() (const point2d & p) const [inline]`

Read-only access to the image value located at point *p*.

Definition at line 234 of file slice_image.hh.

10.323.4.5 `template<typename I > def::coord mln::slice_image< I >::sli () const [inline]`

Give the slice number.

Definition at line 225 of file slice_image.hh.

10.324 mln::sub_image< I, S > Class Template Reference

[Image](#) having its domain restricted by a site set.

```
#include <sub_image.hh>
```

Inherits `image_domain_morpher< I, S, sub_image< I, S > >`.

Public Types

- typedef `sub_image< tag::image_< I >, tag::domain_< S > >` [skeleton](#)

Skeleton.

Public Member Functions

- `const S & domain () const`
Give the definition domain.
- `operator sub_image< const I, S > () const`
Const promotion via conversion.
- `sub_image ()`
Constructor without argument.
- `sub_image (const I &ima, const S &pset)`
Constructor.

10.324.1 Detailed Description

```
template<typename I, typename S> class mln::sub_image< I, S >
```

[Image](#) having its domain restricted by a site set.

Definition at line 102 of file `sub_image.hh`.

10.324.2 Member Typedef Documentation

10.324.2.1 `template<typename I, typename S> typedef sub_image< tag::image_<I>, tag::domain_<S> > mln::sub_image< I, S >::skeleton`

Skeleton.

Definition at line 108 of file `sub_image.hh`.

10.324.3 Constructor & Destructor Documentation

10.324.3.1 `template<typename I, typename S > mln::sub_image< I, S >::sub_image ()`
`[inline]`

Constructor without argument.

Definition at line 182 of file sub_image.hh.

10.324.3.2 `template<typename I, typename S > mln::sub_image< I, S >::sub_image (const I & ima, const S & pset)`
`[inline]`

Constructor.

Definition at line 188 of file sub_image.hh.

10.324.4 Member Function Documentation

10.324.4.1 `template<typename I, typename S > const S & mln::sub_image< I, S >::domain ()`
`const [inline]`

Give the definition domain.

Definition at line 205 of file sub_image.hh.

10.324.4.2 `template<typename I, typename S > mln::sub_image< I, S >::operator sub_image< const I, S > () const`
`[inline]`

Const promotion via conversion.

Definition at line 212 of file sub_image.hh.

10.325 mln::sub_image_if< I, S > Struct Template Reference

[Image](#) having its domain restricted by a site set and a function.

```
#include <sub_image_if.hh>
```

Inherits `image_domain_morpher< I, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > >`.

Public Types

- typedef `sub_image_if< tag::image_< I >, tag::domain_< S > >` [skeleton](#)
Skeleton.

Public Member Functions

- const `p_if< S, fun::p2b::has< I > > & domain () const`
Give the definition domain.

- `sub_image_if()`
Constructor without argument.
- `sub_image_if(I &ima, const S &s)`
Constructor.

10.325.1 Detailed Description

`template<typename I, typename S> struct mln::sub_image_if< I, S >`

[Image](#) having its domain restricted by a site set and a function.

Definition at line 101 of file `sub_image_if.hh`.

10.325.2 Member Typedef Documentation

10.325.2.1 `template<typename I, typename S> typedef sub_image_if< tag::image_<I>, tag::domain_<S> > mln::sub_image_if< I, S >::skeleton`

Skeleton.

Definition at line 106 of file `sub_image_if.hh`.

10.325.3 Constructor & Destructor Documentation

10.325.3.1 `template<typename I, typename S > mln::sub_image_if< I, S >::sub_image_if ()`
`[inline]`

Constructor without argument.

Definition at line 181 of file `sub_image_if.hh`.

10.325.3.2 `template<typename I, typename S > mln::sub_image_if< I, S >::sub_image_if (I & ima, const S & s)`
`[inline]`

Constructor.

Definition at line 187 of file `sub_image_if.hh`.

10.325.4 Member Function Documentation

10.325.4.1 `template<typename I, typename S > const p_if< S, fun::p2b::has< I > > & mln::sub_image_if< I, S >::domain () const`
`[inline]`

Give the definition domain.

Definition at line 204 of file `sub_image_if.hh`.

10.326 mln::thru_image< I, F > Class Template Reference

Morph image values through a function.

```
#include <thru_image.hh>
```

Public Member Functions

- [operator thru_image< const I, F > \(\) const](#)
Const promotion via conversion.

10.326.1 Detailed Description

```
template<typename I, typename F> class mln::thru_image< I, F >
```

Morph image values through a function.

Definition at line 156 of file thru_image.hh.

10.326.2 Member Function Documentation

10.326.2.1 `template<typename I, typename F > mln::thru_image< I, F >::operator thru_image< const I, F > () const` [[inline](#)]

Const promotion via conversion.

Definition at line 239 of file thru_image.hh.

10.327 mln::thrubin_image< I1, I2, F > Class Template Reference

Morphes values from two images through a binary function.

```
#include <thrubin_image.hh>
```

Inherits `image_value_morpher< I1, F::result, thrubin_image< I1, I2, F > >`.

Public Types

- typedef `I1::psite` [psite](#)
Point_Site associated type.
- typedef `value` [rvalue](#)
Return type of read-only access.
- typedef `thrubin_image< tag::image_< I1 >, tag::image_< I2 >, F >` [skeleton](#)
Skeleton.
- typedef `F::result` [value](#)
Value associated type.

10.327.1 Detailed Description

template<typename I1, typename I2, typename F> class mln::thrubin_image< I1, I2, F >

Morphes values from two images through a binary function.

Definition at line 82 of file thrubin_image.hh.

10.327.2 Member Typedef Documentation

10.327.2.1 **template**<typename I1, typename I2, typename F> **typedef** I1 ::psite
mln::thrubin_image< I1, I2, F >::psite

[Point_Site](#) associated type.

Definition at line 94 of file thrubin_image.hh.

10.327.2.2 **template**<typename I1, typename I2, typename F> **typedef** value
mln::thrubin_image< I1, I2, F >::rvalue

Return type of read-only access.

Definition at line 100 of file thrubin_image.hh.

10.327.2.3 **template**<typename I1, typename I2, typename F> **typedef** thrubin_
image<tag::image_<I1>, tag::image_<I2>, F> mln::thrubin_image< I1, I2, F
>::skeleton

Skeleton.

Definition at line 91 of file thrubin_image.hh.

10.327.2.4 **template**<typename I1, typename I2, typename F> **typedef** F ::result
mln::thrubin_image< I1, I2, F >::value

[Value](#) associated type.

Definition at line 97 of file thrubin_image.hh.

10.328 mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D > Class Template Reference

Backward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.

```
#include <adj_higher_dim_connected_n_face_iter.hh>
```

Inherits backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_bkd_iter< D > >, and mln::topo::internal::adj_higher_dim_connected_n_face_iterator< D >.

Public Member Functions

- void [next](#) ()

Go to the next element.

- [adj_higher_dim_connected_n_face_bkd_iter](#) ()

Construction.

10.328.1 Detailed Description

`template<unsigned D> class mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D >`

Backward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 104 of file adj_higher_dim_connected_n_face_iter.hh.

10.328.2 Constructor & Destructor Documentation

10.328.2.1 `template<unsigned D> mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D >::adj_higher_dim_connected_n_face_bkd_iter () [inline]`

Construction.

Definition at line 196 of file adj_higher_dim_connected_n_face_iter.hh.

10.328.3 Member Function Documentation

10.328.3.1 `void mln::Iterator< adj_higher_dim_connected_n_face_bkd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.329 mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D > > Class Template Reference

Forward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.

```
#include <adj_higher_dim_connected_n_face_iter.hh>
```

Inherits forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_fwd_iter< D > >, and mln::topo::internal::adj_higher_dim_connected_n_face_iterator< D >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_higher_dim_connected_n_face_fwd_iter](#) ()
Construction.

10.329.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D >
```

Forward iterator on all the n-faces sharing an adjacent (n+1)-face with a (reference) n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 65 of file adj_higher_dim_connected_n_face_iter.hh.

10.329.2 Constructor & Destructor Documentation

10.329.2.1 `template<unsigned D> mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D >::adj_higher_dim_connected_n_face_fwd_iter () [inline]`

Construction.

Definition at line 162 of file adj_higher_dim_connected_n_face_iter.hh.

10.329.3 Member Function Documentation

10.329.3.1 `void mln::Iterator< adj_higher_dim_connected_n_face_fwd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-define this method. The actual "next" operation has to be defined through the *next*_method.

Precondition

The iterator is valid.

10.330 mln::topo::adj_higher_face_bkd_iter< D > Class Template Reference

Backward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_higher_face_iter.hh>
```

Inherits backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_bkd_iter< D > >.

Public Member Functions

- void [next](#) ()

Go to the next element.

- [adj_higher_face_bkd_iter](#) ()

Construction.

10.330.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_higher_face_bkd_iter< D >
```

Backward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 106 of file adj_higher_face_iter.hh.

10.330.2 Constructor & Destructor Documentation

```
10.330.2.1 template<unsigned D> mln::topo::adj_higher_face_bkd_iter< D  
>::adj_higher_face_bkd_iter ( ) [inline]
```

Construction.

Definition at line 167 of file adj_higher_face_iter.hh.

10.330.3 Member Function Documentation

10.330.3.1 void mln::Iterator< adj_higher_face_bkd_iter< D > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-define this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.331 mln::topo::adj_higher_face_fwd_iter< D > Class Template Reference

Forward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_higher_face_iter.hh>
```

Inherits forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_fwd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_higher_face_fwd_iter](#) ()
Construction.

10.331.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_higher_face_fwd_iter< D >
```

Forward iterator on all the adjacent (n+1)-faces of the n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 72 of file adj_higher_face_iter.hh.

10.331.2 Constructor & Destructor Documentation

10.331.2.1 template<unsigned D> mln::topo::adj_higher_face_fwd_iter< D >::adj_higher_face_fwd_iter () [inline]

Construction.

10.332 mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D > Class Template Reference 983

Definition at line 139 of file adj_higher_face_iter.hh.

10.331.3 Member Function Documentation

10.331.3.1 void mln::Iterator< adj_higher_face_fwd_iter< D > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.332 mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D > Class Template Reference

Backward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an mln::complex<D>.

```
#include <adj_lower_dim_connected_n_face_iter.hh>
```

Inherits backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_bkd_iter< D > >, and mln::topo::internal::adj_lower_dim_connected_n_face_iterator< D >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_lower_dim_connected_n_face_bkd_iter](#) ()
Construction.

10.332.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D >
```

Backward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 104 of file adj_lower_dim_connected_n_face_iter.hh.

10.332.2 Constructor & Destructor Documentation

10.332.2.1 `template<unsigned D> mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D >::adj_lower_dim_connected_n_face_bkd_iter () [inline]`

Construction.

Definition at line 196 of file `adj_lower_dim_connected_n_face_iter.hh`.

10.332.3 Member Function Documentation

10.332.3.1 `void mln::Iterator< adj_lower_dim_connected_n_face_bkd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.333 mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D > Class Template Reference

Forward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an `mln::complex<D>`.

```
#include <adj_lower_dim_connected_n_face_iter.hh>
```

Inherits `forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_fwd_iter< D > >`, and `mln::topo::internal::adj_lower_dim_connected_n_face_iterator< D >`.

Public Member Functions

- void [next](#) ()

Go to the next element.

- [adj_lower_dim_connected_n_face_fwd_iter](#) ()

Construction.

10.333.1 Detailed Description

template<unsigned D> class mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D >

Forward iterator on all the n-faces sharing an adjacent (n-1)-face with a (reference) n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 65 of file adj_lower_dim_connected_n_face_iter.hh.

10.333.2 Constructor & Destructor Documentation

10.333.2.1 template<unsigned D> mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D >::adj_lower_dim_connected_n_face_fwd_iter () [inline]

Construction.

Definition at line 162 of file adj_lower_dim_connected_n_face_iter.hh.

10.333.3 Member Function Documentation

10.333.3.1 void mln::Iterator< adj_lower_dim_connected_n_face_fwd_iter< D > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.334 mln::topo::adj_lower_face_bkd_iter< D > Class Template Reference

Backward iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_lower_face_iter.hh>
```

Inherits backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_face_bkd_iter< D > >.

Public Member Functions

- void [next](#) ()

Go to the next element.

- [adj_lower_face_bkd_iter\(\)](#)
Construction.

10.334.1 Detailed Description

template<unsigned D> class mln::topo::adj_lower_face_bkd_iter< D >

Backward iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 108 of file adj_lower_face_iter.hh.

10.334.2 Constructor & Destructor Documentation

10.334.2.1 template<unsigned D> mln::topo::adj_lower_face_bkd_iter< D >::adj_lower_face_bkd_iter() [inline]

Construction.

Definition at line 169 of file adj_lower_face_iter.hh.

10.334.3 Member Function Documentation

10.334.3.1 void mln::Iterator< adj_lower_face_bkd_iter< D > >::next() [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.335 mln::topo::adj_lower_face_fwd_iter< D > Class Template Reference

Forward iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_lower_face_iter.hh>
```

Inherits forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_face_fwd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.

- [adj_lower_face_fwd_iter](#) ()
Construction.

10.335.1 Detailed Description

`template<unsigned D> class mln::topo::adj_lower_face_fwd_iter< D >`

Forward iterator on all the adjacent (n-1)-faces of the n-face of an `mln::complex<D>`.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 73 of file `adj_lower_face_iter.hh`.

10.335.2 Constructor & Destructor Documentation

10.335.2.1 `template<unsigned D> mln::topo::adj_lower_face_fwd_iter< D >::adj_lower_face_fwd_iter () [inline]`

Construction.

Definition at line 141 of file `adj_lower_face_iter.hh`.

10.335.3 Member Function Documentation

10.335.3.1 `void mln::Iterator< adj_lower_face_fwd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.336 mln::topo::adj_lower_higher_face_bkd_iter< D > Class Template Reference

Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an `mln::complex<D>`.

```
#include <adj_lower_higher_face_iter.hh>
```

Inherits `complex_relative_iterator_sequence< adj_higher_face_bkd_iter< D >, adj_lower_face_bkd_iter< D >, adj_lower_higher_face_bkd_iter< D >>`.

Public Member Functions

- `void next ()`
Go to the next element.
- `adj_lower_higher_face_bkd_iter ()`
Construction.

10.336.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_lower_higher_face_bkd_iter< D >
```

Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an `mln::complex<D>`.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 102 of file `adj_lower_higher_face_iter.hh`.

10.336.2 Constructor & Destructor Documentation

10.336.2.1 `template<unsigned D> mln::topo::adj_lower_higher_face_bkd_iter< D >::adj_lower_higher_face_bkd_iter () [inline]`

Construction.

Definition at line 152 of file `adj_lower_higher_face_iter.hh`.

10.336.3 Member Function Documentation

10.336.3.1 `void mln::Iterator< adj_lower_higher_face_bkd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.337 mln::topo::adj_lower_higher_face_fwd_iter< D > Class Template Reference

Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an mln::complex<D>.

```
#include <adj_lower_higher_face_iter.hh>
```

Inherits complex_relative_iterator_sequence< adj_lower_face_fwd_iter< D >, adj_higher_face_fwd_iter< D >, adj_lower_higher_face_fwd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_lower_higher_face_fwd_iter](#) ()
Construction.

10.337.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_lower_higher_face_fwd_iter< D >
```

Forward iterator on all the adjacent (n-1)-faces and (n+1)-faces of the n-face of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 71 of file adj_lower_higher_face_iter.hh.

10.337.2 Constructor & Destructor Documentation

10.337.2.1 `template<unsigned D> mln::topo::adj_lower_higher_face_fwd_iter< D >::adj_lower_higher_face_fwd_iter () [inline]`

Construction.

Definition at line 133 of file adj_lower_higher_face_iter.hh.

10.337.3 Member Function Documentation

10.337.3.1 `void mln::Iterator< adj_lower_higher_face_fwd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.338 mln::topo::adj_m_face_bkd_iter< D > Class Template Reference

Backward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex.

```
#include <adj_m_face_iter.hh>
```

Inherits backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_bkd_iter< D >>, and mln::topo::internal::adj_m_face_iterator< D >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_m_face_bkd_iter](#) ()
Construction.
- template<typename Fref >
[adj_m_face_bkd_iter](#) (const Fref &f_ref, unsigned m)
Constructs an iterator, with f_ref as reference face, and a target dimension equal to m.

10.338.1 Detailed Description

```
template<unsigned D> class mln::topo::adj_m_face_bkd_iter< D >
```

Backward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex.

Template Parameters

D The dimension of the complex this iterator belongs to.

The dimension parameter (*m_*) must be lower or equal to *D*.

If *m_* is equal to the dimension of the reference face, then the iterated set is empty.

Definition at line 118 of file adj_m_face_iter.hh.

10.338.2 Constructor & Destructor Documentation

10.338.2.1 template<unsigned D> mln::topo::adj_m_face_bkd_iter< D >::adj_m_face_bkd_iter () [[inline](#)]

Construction.

Construct an iterator, with an invalid reference face, and a target dimension equal to 0.

Definition at line 223 of file adj_m_face_iter.hh.

10.338.2.2 `template<unsigned D> template<typename Fref > mln::topo::adj_m_face_bkd_iter< D >::adj_m_face_bkd_iter (const Fref & f_ref, unsigned m) [inline]`

Constructs an iterator, with *f_ref* as reference face, and a target dimension equal to *m*.

Definition at line 230 of file adj_m_face_iter.hh.

10.338.3 Member Function Documentation

10.338.3.1 `void mln::Iterator< adj_m_face_bkd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.339 mln::topo::adj_m_face_fwd_iter< D > Class Template Reference

Forward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex.

```
#include <adj_m_face_iter.hh>
```

Inherits `forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_fwd_iter< D > >`, and `mln::topo::internal::adj_m_face_iterator< D >`.

Public Member Functions

- `void next ()`
Go to the next element.
- `adj_m_face_fwd_iter ()`
Construction.
- `template<typename Fref > adj_m_face_fwd_iter (const Fref &f_ref, unsigned m)`
Constructs an iterator, with f_ref as reference face, and a target dimension equal to m.

10.339.1 Detailed Description

`template<unsigned D> class mln::topo::adj_m_face_fwd_iter< D >`

Forward iterator on all the m-faces transitively adjacent to a (reference) n-face in a complex.

Template Parameters

D The dimension of the complex this iterator belongs to.

The dimension parameter (*m_*) must be lower or equal to *D*.

If *m_* is equal to the dimension of the reference face, then the iterated set is empty.

Definition at line 70 of file `adj_m_face_iter.hh`.

10.339.2 Constructor & Destructor Documentation

10.339.2.1 `template<unsigned D> mln::topo::adj_m_face_fwd_iter< D >::adj_m_face_fwd_iter () [inline]`

Construction.

Construct an iterator, with an invalid reference face, and a target dimension equal to 0.

Definition at line 194 of file `adj_m_face_iter.hh`.

10.339.2.2 `template<unsigned D> template<typename Fref > mln::topo::adj_m_face_fwd_iter< D >::adj_m_face_fwd_iter (const Fref & f_ref, unsigned m) [inline]`

Constructs an iterator, with *f_ref* as reference face, and a target dimension equal to *m*.

Definition at line 201 of file `adj_m_face_iter.hh`.

10.339.3 Member Function Documentation

10.339.3.1 `void mln::Iterator< adj_m_face_fwd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

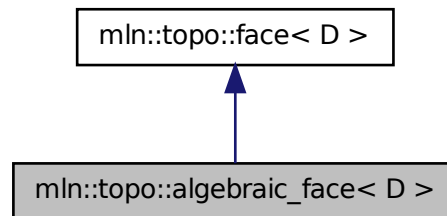
The iterator is valid.

10.340 mln::topo::algebraic_face< D > Class Template Reference

Algebraic face handle in a complex; the face dimension is dynamic.

```
#include <algebraic_face.hh>
```

Inheritance diagram for mln::topo::algebraic_face< D >:



Public Member Functions

- [algebraic_face](#) ()
Build a non-initialized algebraic face handle.
- [algebraic_face](#) ([complex](#)< D > &[complex](#), unsigned n, unsigned face_id, bool [sign](#))
Build an algebraic face handle from complex and face_id.
- [template](#)<unsigned N>
[algebraic_face](#) (const [algebraic_n_face](#)< N, D > &f)
Build a face handle from an mln::topo::algebraic_n_face.
- [algebraic_face](#) (const [face](#)< D > &f, bool [sign](#))
Build an algebraic face handle from an mln::face.
- void [invalidate](#) ()
Invalidate this handle.
- bool [is_valid](#) () const
Is this handle valid?
- bool [sign](#) () const
Accessors.
- void [set_sign](#) (bool [sign](#))
Set the sign of this face.
- [complex](#)< D > [cplx](#) () const
Accessors.
- unsigned [n](#) () const
Return the dimension of the face.

- unsigned `face_id` () const
Return the id of the face.
- void `set_cplx` (const `complex`< D > &cplx)
Set the complex the face belongs to.
- void `set_n` (unsigned n)
Set the dimension of the face.
- void `inc_n` ()
Increment the dimension of the face.
- void `dec_n` ()
Decrement the dimension of the face.
- void `set_face_id` (unsigned face_id)
Set the id of the face.
- void `inc_face_id` ()
Increment the id of the face.
- void `dec_face_id` ()
Decrement the id of the face.
- template<unsigned N>
face_data< N, D > & `data` () const
Return the mln::topo::face_data pointed by this handle.
- std::vector< `algebraic_face`< D > > `lower_dim_adj_faces` () const
Return an array of face handles pointing to adjacent (n-1)-faces.
- std::vector< `algebraic_face`< D > > `higher_dim_adj_faces` () const
Return an array of face handles pointing to adjacent (n+1)-faces.

10.340.1 Detailed Description

template<unsigned D> class mln::topo::algebraic_face< D >

Algebraic face handle in a complex; the face dimension is dynamic. Contrary to an `mln::topo::algebraic_n_face`, the dimension of an `mln::topo::algebraic_face` is not fixed.

Definition at line 60 of file `algebraic_face.hh`.

10.340.2 Constructor & Destructor Documentation

10.340.2.1 template<unsigned D> mln::topo::algebraic_face< D >::algebraic_face ()
[inline]

Build a non-initialized algebraic face handle.

Definition at line 157 of file `algebraic_face.hh`.

10.340.2.2 `template<unsigned D> mln::topo::algebraic_face< D >::algebraic_face (complex< D > & complex, unsigned n, unsigned face_id, bool sign) [inline]`

Build an algebraic face handle from *complex* and *face_id*.

Definition at line 164 of file algebraic_face.hh.

10.340.2.3 `template<unsigned D> mln::topo::algebraic_face< D >::algebraic_face (const face< D > & f, bool sign) [inline]`

Build an algebraic face handle from an mln::face.

Definition at line 174 of file algebraic_face.hh.

References mln::topo::face< D >::n().

10.340.2.4 `template<unsigned D> template<unsigned N> mln::topo::algebraic_face< D >::algebraic_face (const algebraic_n_face< N, D > & f) [inline]`

Build a face handle from an mln::topo::algebraic_n_face.

Definition at line 184 of file algebraic_face.hh.

10.340.3 Member Function Documentation

10.340.3.1 `template<unsigned D> complex< D > mln::topo::face< D >::cplx () const [inline, inherited]`

Accessors.

Return the complex the face belongs to.

Definition at line 224 of file face.hh.

Referenced by mln::complex_psite< D, G >::complex_psite(), mln::topo::operator!==(), and mln::topo::operator==().

10.340.3.2 `template<unsigned D> template<unsigned N> face_data< N, D > & mln::topo::face< D >::data () const [inline, inherited]`

Return the mln::topo::face_data pointed by this handle.

Definition at line 305 of file face.hh.

References mln::topo::face< D >::is_valid().

10.340.3.3 `template<unsigned D> void mln::topo::face< D >::dec_face_id () [inline, inherited]`

Decrement the id of the face.

Definition at line 296 of file face.hh.

10.340.3.4 `template<unsigned D> void mln::topo::face< D >::dec_n () [inline, inherited]`

Decrement the dimension of the face.

Definition at line 272 of file face.hh.

10.340.3.5 `template<unsigned D> unsigned mln::topo::face< D >::face_id () const [inline, inherited]`

Return the id of the face.

Definition at line 240 of file face.hh.

Referenced by `mln::geom::complex_geometry< D, P >::operator()`, and `mln::topo::operator==()`.

10.340.3.6 `template<unsigned D> std::vector< algebraic_face< D > > mln::topo::face< D >::higher_dim_adj_faces () const [inline, inherited]`

Return an array of face handles pointing to adjacent (n+1)-faces.

Definition at line 370 of file face.hh.

10.340.3.7 `template<unsigned D> void mln::topo::face< D >::inc_face_id () [inline, inherited]`

Increment the id of the face.

Definition at line 288 of file face.hh.

10.340.3.8 `template<unsigned D> void mln::topo::face< D >::inc_n () [inline, inherited]`

Increment the dimension of the face.

Definition at line 264 of file face.hh.

10.340.3.9 `template<unsigned D> void mln::topo::face< D >::invalidate () [inline, inherited]`

Invalidate this handle.

Definition at line 215 of file face.hh.

References `mln::topo::face< D >::set_face_id()`, and `mln::topo::face< D >::set_n()`.

10.340.3.10 `template<unsigned D> bool mln::topo::face< D >::is_valid () const [inline, inherited]`

Is this handle valid?

Definition at line 207 of file face.hh.

Referenced by `mln::topo::face< D >::data()`.

10.340.3.11 `template<unsigned D> std::vector< algebraic_face< D > > mln::topo::face< D >::lower_dim_adj_faces () const [inline, inherited]`

Return an array of face handles pointing to adjacent (n-1)-faces.

Definition at line 357 of file face.hh.

10.340.3.12 `template<unsigned D> unsigned mln::topo::face< D >::n () const [inline, inherited]`

Return the dimension of the face.

Definition at line 232 of file face.hh.

Referenced by `mln::topo::algebraic_face< D >::algebraic_face()`, `mln::geom::complex_geometry< D, P >::operator()`, and `mln::topo::operator==()`.

10.340.3.13 `template<unsigned D> void mln::topo::face< D >::set_cplx (const complex< D > & cplx) [inline, inherited]`

Set the complex the face belongs to.

Definition at line 248 of file face.hh.

10.340.3.14 `template<unsigned D> void mln::topo::face< D >::set_face_id (unsigned face_id) [inline, inherited]`

Set the id of the face.

Definition at line 280 of file face.hh.

Referenced by `mln::topo::face< D >::invalidate()`.

10.340.3.15 `template<unsigned D> void mln::topo::face< D >::set_n (unsigned n) [inline, inherited]`

Set the dimension of the face.

Definition at line 256 of file face.hh.

Referenced by `mln::topo::face< D >::invalidate()`.

10.340.3.16 `template<unsigned D> void mln::topo::algebraic_face< D >::set_sign (bool sign) [inline]`

Set the sign of this face.

Definition at line 203 of file algebraic_face.hh.

10.340.3.17 `template<unsigned D> bool mln::topo::algebraic_face< D >::sign () const [inline]`

Accessors.

Return the sign of this face.

Definition at line 195 of file algebraic_face.hh.

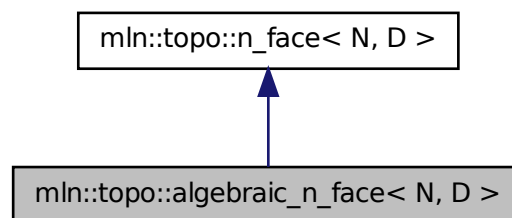
Referenced by mln::topo::operator==().

10.341 mln::topo::algebraic_n_face< N, D > Class Template Reference

Algebraic N-face handle in a complex.

```
#include <algebraic_n_face.hh>
```

Inheritance diagram for mln::topo::algebraic_n_face< N, D >:



Public Member Functions

- [algebraic_n_face](#) ()
Build a non-initialized algebraic face handle.
- [algebraic_n_face](#) ([complex](#)< D > &[complex](#), unsigned [face_id](#), bool [sign](#))
Build an algebraic face handle from complex and face_id.
- [algebraic_n_face](#) (const [n_face](#)< N, D > &[f](#), bool [sign](#))
Build an algebraic face handle from an mln::n_face.
- void [invalidate](#) ()
Invalidate this handle.
- bool [is_valid](#) () const
Is this handle valid?
- bool [sign](#) () const
Accessors.
- void [set_sign](#) (bool [sign](#))
Set the sign of this face.

- `complex< D > cplx () const`
Accessors.
- `unsigned face_id () const`
Return the id of the face.
- `void set_cplx (const complex< D > &cplx)`
Set the complex the face belongs to.
- `unsigned n () const`
Return the dimension of the face.
- `void set_face_id (unsigned face_id)`
Set the id of the face.
- `void inc_face_id ()`
Increment the id of the face.
- `void dec_face_id ()`
Decrement the id of the face.
- `face_data< N, D > & data () const`
Return the mln::topo::face_data pointed by this handle.
- `std::vector< algebraic_n_face< N-1, D > > lower_dim_adj_faces () const`
Return an array of face handles pointing to adjacent (n-1)-faces.
- `std::vector< algebraic_n_face< N+1, D > > higher_dim_adj_faces () const`
Return an array of face handles pointing to adjacent (n+1)-faces.

10.341.1 Detailed Description

`template<unsigned N, unsigned D> class mln::topo::algebraic_n_face< N, D >`

Algebraic `N-face` handle in a complex. Contrary to an `mln::topo::algebraic_face`, the dimension of an `mln::topo::algebraic_n_face` is fixed.

Definition at line 50 of file `algebraic_n_face.hh`.

10.341.2 Constructor & Destructor Documentation

10.341.2.1 `template<unsigned N, unsigned D> mln::topo::algebraic_n_face< N, D >::algebraic_n_face () [inline]`

Build a non-initialized algebraic face handle.

Definition at line 166 of file `algebraic_n_face.hh`.

References `mln::topo::n_face< N, D >::is_valid()`.

10.341.2.2 `template<unsigned N, unsigned D> mln::topo::algebraic_n_face< N, D >::algebraic_n_face (complex< D > & complex, unsigned face_id, bool sign) [inline]`

Build an algebraic face handle from *complex* and *face_id*.

Definition at line 176 of file algebraic_n_face.hh.

10.341.2.3 `template<unsigned N, unsigned D> mln::topo::algebraic_n_face< N, D >::algebraic_n_face (const n_face< N, D > & f, bool sign) [inline]`

Build an algebraic face handle from an mln::n_face.

Definition at line 186 of file algebraic_n_face.hh.

10.341.3 Member Function Documentation

10.341.3.1 `template<unsigned N, unsigned D> complex< D > mln::topo::n_face< N, D >::cplx () const [inline, inherited]`

Accessors.

Return the complex the face belongs to.

Definition at line 195 of file n_face.hh.

Referenced by mln::topo::n_faces_set< N, D >::add(), mln::topo::operator!=(), and mln::topo::operator==().

10.341.3.2 `template<unsigned N, unsigned D> face_data< N, D > & mln::topo::n_face< N, D >::data () const [inline, inherited]`

Return the mln::topo::face_data pointed by this handle.

Definition at line 251 of file n_face.hh.

References mln::topo::n_face< N, D >::is_valid().

10.341.3.3 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::dec_face_id () [inline, inherited]`

Decrement the id of the face.

Definition at line 243 of file n_face.hh.

10.341.3.4 `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::face_id () const [inline, inherited]`

Return the id of the face.

Definition at line 211 of file n_face.hh.

Referenced by mln::topo::operator==().

10.341.3.5 `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N+1, D >
> mln::topo::n_face< N, D >::higher_dim_adj_faces () const [inline,
inherited]`

Return an array of face handles pointing to adjacent (n+1)-faces.

Definition at line 270 of file n_face.hh.

References mln::topo::n_face< N, D >::is_valid().

Referenced by mln::topo::edge().

10.341.3.6 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::inc_face_id ()
[inline, inherited]`

Increment the id of the face.

Definition at line 235 of file n_face.hh.

10.341.3.7 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::invalidate ()
[inline, inherited]`

Invalidate this handle.

Definition at line 187 of file n_face.hh.

References mln::topo::n_face< N, D >::set_face_id().

10.341.3.8 `template<unsigned N, unsigned D> bool mln::topo::n_face< N, D >::is_valid ()
const [inline, inherited]`

Is this handle valid?

Definition at line 179 of file n_face.hh.

Referenced by mln::topo::algebraic_n_face< N, D >::algebraic_n_face(), mln::topo::n_face< N, D >::data(), mln::topo::n_face< N, D >::higher_dim_adj_faces(), mln::topo::n_face< N, D >::lower_dim_adj_faces(), and mln::topo::n_face< N, D >::n_face().

10.341.3.9 `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N-1, D >
> > mln::topo::n_face< N, D >::lower_dim_adj_faces () const [inline,
inherited]`

Return an array of face handles pointing to adjacent (n-1)-faces.

Definition at line 260 of file n_face.hh.

References mln::topo::n_face< N, D >::is_valid().

10.341.3.10 `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::n ()
const [inline, inherited]`

Return the dimension of the face.

Definition at line 203 of file n_face.hh.

10.341.3.11 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::set_cplx (const complex< D > & cplx) [inline, inherited]`

Set the complex the face belongs to.

Definition at line 219 of file n_face.hh.

10.341.3.12 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::set_face_id (unsigned face_id) [inline, inherited]`

Set the id of the face.

Definition at line 227 of file n_face.hh.

Referenced by `mln::topo::n_face< N, D >::invalidate()`.

10.341.3.13 `template<unsigned N, unsigned D> void mln::topo::algebraic_n_face< N, D >::set_sign (bool sign) [inline]`

Set the sign of this face.

Definition at line 205 of file algebraic_n_face.hh.

10.341.3.14 `template<unsigned N, unsigned D> bool mln::topo::algebraic_n_face< N, D >::sign () const [inline]`

Accessors.

Return the sign of this face.

Definition at line 197 of file algebraic_n_face.hh.

Referenced by `mln::topo::operator==()`.

10.342 `mln::topo::center_only_iter< D >` Class Template Reference

[Iterator](#) on all the adjacent (n-1)-faces of the n-face of an `mln::complex<D>`.

```
#include <center_only_iter.hh>
```

Inherits `forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, center_only_iter< D >>`.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [center_only_iter](#) ()
Construction.

10.342.1 Detailed Description

`template<unsigned D> class mln::topo::center_only_iter< D >`

[Iterator](#) on all the adjacent (n-1)-faces of the n-face of an `mln::complex<D>`.

Template Parameters

D The dimension of the complex this iterator belongs to.

`mln::topo::center_only_iter` inherits from `mln::topo::internal::forward_complex_relative_iterator_base`, but it could inherit from `mln::topo::internal::backward_complex_relative_iterator_base` as well, since it always contains a single element, the center/reference face (and the traversal order is meaningless).

This iterator is essentially used to implement other iterators.

See also

`mln::topo::centered_iter_adapter`
`mln::complex_lower_window`
`mln::complex_higher_window`
`mln::complex_lower_higher_window`

Definition at line 73 of file `center_only_iter.hh`.

10.342.2 Constructor & Destructor Documentation

10.342.2.1 `template<unsigned D> mln::topo::center_only_iter< D >::center_only_iter ()`
[inline]

Construction.

Definition at line 107 of file `center_only_iter.hh`.

10.342.3 Member Function Documentation

10.342.3.1 `void mln::Iterator< center_only_iter< D > >::next ()` **[inherited]**

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.343 mln::topo::centered_bkd_iter_adapter< D, I > Class Template Reference

Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

```
#include <centered_iter_adapter.hh>
```

Inherits `complex_relative_iterator_sequence< I, center_only_iter< D >, centered_bkd_iter_adapter< D, I >>`.

Public Member Functions

- `void next ()`
Go to the next element.
- `centered_bkd_iter_adapter ()`
Construction.

10.343.1 Detailed Description

```
template<unsigned D, typename I> class mln::topo::centered_bkd_iter_adapter< D, I >
```

Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

Template Parameters

- D* The dimension of the complex this iterator belongs to.
- I* The adapted complex relative iterator.

Definition at line 91 of file `centered_iter_adapter.hh`.

10.343.2 Constructor & Destructor Documentation

```
10.343.2.1 template<unsigned D, typename I > mln::topo::centered_bkd_iter_adapter< D, I >::centered_bkd_iter_adapter ( ) [inline]
```

Construction.

Definition at line 141 of file `centered_iter_adapter.hh`.

10.343.3 Member Function Documentation

```
10.343.3.1 void mln::Iterator< centered_bkd_iter_adapter< D, I > >::next ( ) [inherited]
```

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.344 mln::topo::centered_fwd_iter_adapter< D, I > Class Template Reference

Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

```
#include <centered_iter_adapter.hh>
```

Inherits complex_relative_iterator_sequence< center_only_iter< D >, I, centered_fwd_iter_adapter< D, I >>.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [centered_fwd_iter_adapter](#) ()
Construction.

10.344.1 Detailed Description

```
template<unsigned D, typename I> class mln::topo::centered_fwd_iter_adapter< D, I >
```

Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.

Template Parameters

- D* The dimension of the complex this iterator belongs to.
- I* The adapted complex relative iterator.

Definition at line 57 of file centered_iter_adapter.hh.

10.344.2 Constructor & Destructor Documentation

10.344.2.1 `template<unsigned D, typename I> mln::topo::centered_fwd_iter_adapter< D, I >::centered_fwd_iter_adapter () [inline]`

Construction.

Definition at line 122 of file centered_iter_adapter.hh.

10.344.3 Member Function Documentation

10.344.3.1 `void mln::Iterator< centered_fwd_iter_adapter< D, I >>::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.345 mln::topo::complex< D > Class Template Reference

General complex of dimension D.

```
#include <complex.hh>
```

Public Types

- typedef [face_bkd_iter](#)< D > [bkd_citer](#)
Backward [mln::Iterator](#) type iterating on all faces.
- typedef [face_fwd_iter](#)< D > [fwd_citer](#)
Forward [mln::Iterator](#) type iterating on all faces.

Public Member Functions

- const void * [addr](#) () const
Get the address of the data of this complex.
- [complex](#) ()
Complex construction.
- [n_face](#)< 0u, D > [add_face](#) ()
Add a 0-face to the complex.
- template<unsigned N>
[n_face](#)< N+1, D > [add_face](#) (const [n_faces_set](#)< N, D > &adjacent_faces)
Add a (N+1)-face to the complex (with N >= 0).
- unsigned [nfaces](#) () const
Static manipulators.
- template<unsigned N>
unsigned [nfaces_of_static_dim](#) () const
Return the number of N-faces.
- unsigned [nfaces_of_dim](#) (unsigned n) const
Dynamic manipulators.
- void [print](#) (std::ostream &ostr) const
Pretty-printing.

- `template<unsigned N>`
`void print_faces (std::ostream &ostr) const`
Print the faces of dimension N.

10.345.1 Detailed Description

`template<unsigned D> class mln::topo::complex< D >`

General complex of dimension D.

Definition at line 87 of file complex.hh.

10.345.2 Member Typedef Documentation

10.345.2.1 `template<unsigned D> typedef face_bkd_iter<D> mln::topo::complex< D >::bkd_citer`

Backward `mln::Iterator` type iterating on all faces.

Definition at line 93 of file complex.hh.

10.345.2.2 `template<unsigned D> typedef face_fwd_iter<D> mln::topo::complex< D >::fwd_citer`

Forward `mln::Iterator` type iterating on all faces.

Definition at line 91 of file complex.hh.

10.345.3 Constructor & Destructor Documentation

10.345.3.1 `template<unsigned D> mln::topo::complex< D >::complex () [inline]`

Complex construction.

Create a new D-complex.

Definition at line 471 of file complex.hh.

10.345.4 Member Function Documentation

10.345.4.1 `template<unsigned D> n_face< 0u, D > mln::topo::complex< D >::add_face () [inline]`

Add a 0-face to the complex.

Definition at line 480 of file complex.hh.

10.345.4.2 `template<unsigned D> template<unsigned N> n_face< N+1, D >`
`mln::topo::complex< D >::add_face (const n_faces_set< N, D > & adjacent_faces)`
`[inline]`

Add a (N+1)-face to the complex (with $N \geq 0$).

Parameters

adjacent_faces The (N-1)-faces adjacent to the new N-face.

Definition at line 493 of file complex.hh.

References `mln::topo::n_faces_set< N, D >::faces()`.

10.345.4.3 `template<unsigned D> const void * mln::topo::complex< D >::addr () const`
`[inline]`

Get the address of the data of this complex.

This address is a concise and useful information to print and track the actual content of this complex.

Definition at line 699 of file complex.hh.

10.345.4.4 `template<unsigned D> unsigned mln::topo::complex< D >::nfaces () const`
`[inline]`

Static manipulators.

These methods use statically-known input.

Return the total number of faces, whatever their dimension.

Definition at line 579 of file complex.hh.

10.345.4.5 `template<unsigned D> unsigned mln::topo::complex< D >::nfaces_of_dim (`
`unsigned n) const [inline]`

Dynamic manipulators.

These methods use input know as run time.

Return the number of *n-faces*.

Warning, this function has a complexity linear in term of N, since each `n_faces_set` is checked (the present implementation does not provide a direct access to `n_faces_set` through a dynamic value of the dimension).

Definition at line 601 of file complex.hh.

10.345.4.6 `template<unsigned D> template<unsigned N> unsigned mln::topo::complex< D`
`>::nfaces_of_static_dim () const [inline]`

Return the number of N-faces.

Definition at line 588 of file complex.hh.

10.345.4.7 `template<unsigned D> void mln::topo::complex< D >::print (std::ostream & ostr) const [inline]`

Pretty-printing.

Print the complex.

Definition at line 679 of file complex.hh.

Referenced by mln::topo::operator<<().

10.345.4.8 `template<unsigned D> template<unsigned N> void mln::topo::complex< D >::print_faces (std::ostream & ostr) const [inline]`

Print the faces of dimension N.

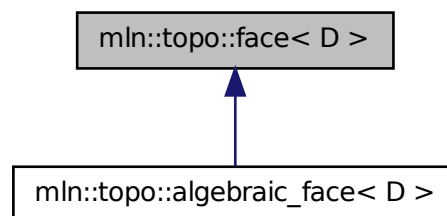
Definition at line 688 of file complex.hh.

10.346 mln::topo::face< D > Class Template Reference

Face handle in a complex; the face dimension is dynamic.

```
#include <face.hh>
```

Inheritance diagram for mln::topo::face< D >:



Public Member Functions

- `face ()`
Build a non-initialized face handle.
- `face (complex< D > &complex, unsigned n, unsigned face_id)`
Build a face handle from complex and face_id.
- `template<unsigned N> face (const n_face< N, D > &f)`
Build a face handle from an mln::topo::n_face.

- void [invalidate](#) ()
Invalidate this handle.
- bool [is_valid](#) () const
Is this handle valid?
- [complex](#)< D > [cplx](#) () const
Accessors.
- unsigned [n](#) () const
Return the dimension of the face.
- unsigned [face_id](#) () const
Return the id of the face.
- void [set_cplx](#) (const [complex](#)< D > &cplx)
Set the complex the face belongs to.
- void [set_n](#) (unsigned n)
Set the dimension of the face.
- void [inc_n](#) ()
Increment the dimension of the face.
- void [dec_n](#) ()
Decrement the dimension of the face.
- void [set_face_id](#) (unsigned face_id)
Set the id of the face.
- void [inc_face_id](#) ()
Increment the id of the face.
- void [dec_face_id](#) ()
Decrement the id of the face.
- template<unsigned N>
[face_data](#)< N, D > & [data](#) () const
Return the mln::topo::face_data pointed by this handle.
- std::vector< [algebraic_face](#)< D > > [lower_dim_adj_faces](#) () const
Return an array of face handles pointing to adjacent (n-1)-faces.
- std::vector< [algebraic_face](#)< D > > [higher_dim_adj_faces](#) () const
Return an array of face handles pointing to adjacent (n+1)-faces.

10.346.1 Detailed Description

`template<unsigned D> class mln::topo::face< D >`

Face handle in a complex; the face dimension is dynamic. Contrary to an [mln::topo::n_face](#), the dimension of an [mln::topo::face](#) is not fixed.

Definition at line 64 of file face.hh.

10.346.2 Constructor & Destructor Documentation

10.346.2.1 `template<unsigned D> mln::topo::face< D >::face () [inline]`

Build a non-initialized face handle.

Definition at line 178 of file face.hh.

10.346.2.2 `template<unsigned D> mln::topo::face< D >::face (complex< D > & complex, unsigned n, unsigned face_id) [inline]`

Build a face handle from *complex* and *face_id*.

Definition at line 187 of file face.hh.

10.346.2.3 `template<unsigned D> template<unsigned N> mln::topo::face< D >::face (const n_face< N, D > & f) [inline]`

Build a face handle from an [mln::topo::n_face](#).

Definition at line 197 of file face.hh.

10.346.3 Member Function Documentation

10.346.3.1 `template<unsigned D> complex< D > mln::topo::face< D >::cplx () const [inline]`

Accessors.

Return the complex the face belongs to.

Definition at line 224 of file face.hh.

Referenced by `mln::complex_psite< D, G >::complex_psite()`, `mln::topo::operator!=()`, and `mln::topo::operator==()`.

10.346.3.2 `template<unsigned D> template<unsigned N> face_data< N, D > & mln::topo::face< D >::data () const [inline]`

Return the `mln::topo::face_data` pointed by this handle.

Definition at line 305 of file face.hh.

References `mln::topo::face< D >::is_valid()`.

10.346.3.3 `template<unsigned D> void mln::topo::face< D >::dec_face_id () [inline]`

Decrement the id of the face.

Definition at line 296 of file face.hh.

10.346.3.4 `template<unsigned D> void mln::topo::face< D >::dec_n () [inline]`

Decrement the dimension of the face.

Definition at line 272 of file face.hh.

10.346.3.5 `template<unsigned D> unsigned mln::topo::face< D >::face_id () const [inline]`

Return the id of the face.

Definition at line 240 of file face.hh.

Referenced by `mln::geom::complex_geometry< D, P >::operator()`, and `mln::topo::operator==()`.

10.346.3.6 `template<unsigned D> std::vector< algebraic_face< D > > mln::topo::face< D >::higher_dim_adj_faces () const [inline]`

Return an array of face handles pointing to adjacent (n+1)-faces.

Definition at line 370 of file face.hh.

10.346.3.7 `template<unsigned D> void mln::topo::face< D >::inc_face_id () [inline]`

Increment the id of the face.

Definition at line 288 of file face.hh.

10.346.3.8 `template<unsigned D> void mln::topo::face< D >::inc_n () [inline]`

Increment the dimension of the face.

Definition at line 264 of file face.hh.

10.346.3.9 `template<unsigned D> void mln::topo::face< D >::invalidate () [inline]`

Invalidate this handle.

Definition at line 215 of file face.hh.

References `mln::topo::face< D >::set_face_id()`, and `mln::topo::face< D >::set_n()`.

10.346.3.10 `template<unsigned D> bool mln::topo::face< D >::is_valid () const [inline]`

Is this handle valid?

Definition at line 207 of file face.hh.

Referenced by `mln::topo::face< D >::data()`.

10.346.3.11 `template<unsigned D> std::vector< algebraic_face< D > > mln::topo::face< D >::lower_dim_adj_faces () const [inline]`

Return an array of face handles pointing to adjacent (n-1)-faces.

Definition at line 357 of file face.hh.

10.346.3.12 `template<unsigned D> unsigned mln::topo::face< D >::n () const [inline]`

Return the dimension of the face.

Definition at line 232 of file face.hh.

Referenced by mln::topo::algebraic_face< D >::algebraic_face(), mln::geom::complex_geometry< D, P >::operator()(), and mln::topo::operator==().

10.346.3.13 `template<unsigned D> void mln::topo::face< D >::set_cplx (const complex< D > & cplx) [inline]`

Set the complex the face belongs to.

Definition at line 248 of file face.hh.

10.346.3.14 `template<unsigned D> void mln::topo::face< D >::set_face_id (unsigned face_id) [inline]`

Set the id of the face.

Definition at line 280 of file face.hh.

Referenced by mln::topo::face< D >::invalidate().

10.346.3.15 `template<unsigned D> void mln::topo::face< D >::set_n (unsigned n) [inline]`

Set the dimension of the face.

Definition at line 256 of file face.hh.

Referenced by mln::topo::face< D >::invalidate().

10.347 mln::topo::face_bkd_iter< D > Class Template Reference

Backward iterator on all the faces of an mln::complex<D>.

```
#include <face_iter.hh>
```

Inherits complex_set_iterator_base< topo::face< D >, face_bkd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [face_bkd_iter](#) ()
Construction and assignment.
- void [start](#) ()

Manipulation.

10.347.1 Detailed Description

template<unsigned D> class mln::topo::face_bkd_iter< D >

Backward iterator on all the faces of an mln::complex<D>.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 112 of file face_iter.hh.

10.347.2 Constructor & Destructor Documentation

10.347.2.1 template<unsigned D> mln::topo::face_bkd_iter< D >::face_bkd_iter ()
[inline]

Construction and assignment.

Definition at line 207 of file face_iter.hh.

10.347.3 Member Function Documentation

10.347.3.1 void mln::Iterator< face_bkd_iter< D > >::next () **[inherited]**

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.347.3.2 template<unsigned D> void mln::topo::face_bkd_iter< D >::start () **[inline]**

Manipulation.

Start an iteration.

Definition at line 224 of file face_iter.hh.

10.348 mln::topo::face_fwd_iter< D > Class Template Reference

Forward iterator on all the faces of an mln::complex<D>.


```
#include <face_iter.hh>
```

Inherits `complex_set_iterator_base< topo::face< D >, face_fwd_iter< D > >`.

Public Member Functions

- void `next()`
Go to the next element.
- `face_fwd_iter()`
Construction and assignment.
- void `start()`
Manipulation.

10.348.1 Detailed Description

`template<unsigned D> class mln::topo::face_fwd_iter< D >`

Forward iterator on all the faces of an `mln::complex<D>`.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 69 of file `face_iter.hh`.

10.348.2 Constructor & Destructor Documentation

10.348.2.1 `template<unsigned D> mln::topo::face_fwd_iter< D >::face_fwd_iter ()`
`[inline]`

Construction and assignment.

Definition at line 155 of file `face_iter.hh`.

10.348.3 Member Function Documentation

10.348.3.1 `void mln::Iterator< face_fwd_iter< D > >::next ()` `[inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_method`.

Precondition

The iterator is valid.

10.348.3.2 `template<unsigned D> void mln::topo::face_fwd_iter< D >::start () [inline]`

Manipulation.

Test if the iterator is valid.

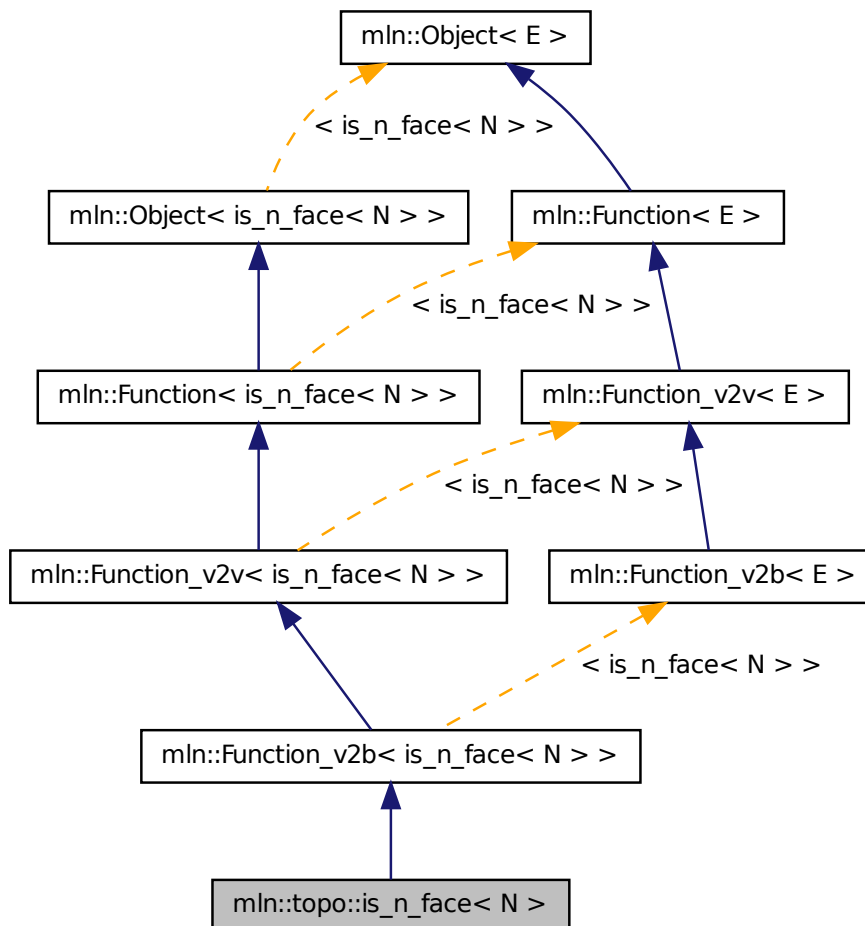
Definition at line 172 of file `face_iter.hh`.

10.349 `mln::topo::is_n_face< N >` Struct Template Reference

A functor testing whether a `mln::complex_psite` is an N-face.

```
#include <is_n_face.hh>
```

Inheritance diagram for `mln::topo::is_n_face< N >`:



10.349.1 Detailed Description

```
template<unsigned N> struct mln::topo::is_n_face< N >
```

A functor testing wheter a [mln::complex_psite](#) is an N -face.

Definition at line 48 of file is_n_face.hh.

10.350 mln::topo::is_simple_2d_t< N > Struct Template Reference

Test if a point is simple or not.

```
#include <is_simple_2d.hh>
```

10.350.1 Detailed Description

```
template<typename N> struct mln::topo::is_simple_2d_t< N >
```

Test if a point is simple or not. A point of an object is simple if in its c8 neighborhood, there is exactly one connected component of the object, and only one connected component of the background Examples : (| == object, - = background)

- - | | P | Here p is simple in the c4 and c8 case. | | |
- | - | P | Here p is never simple. | | |

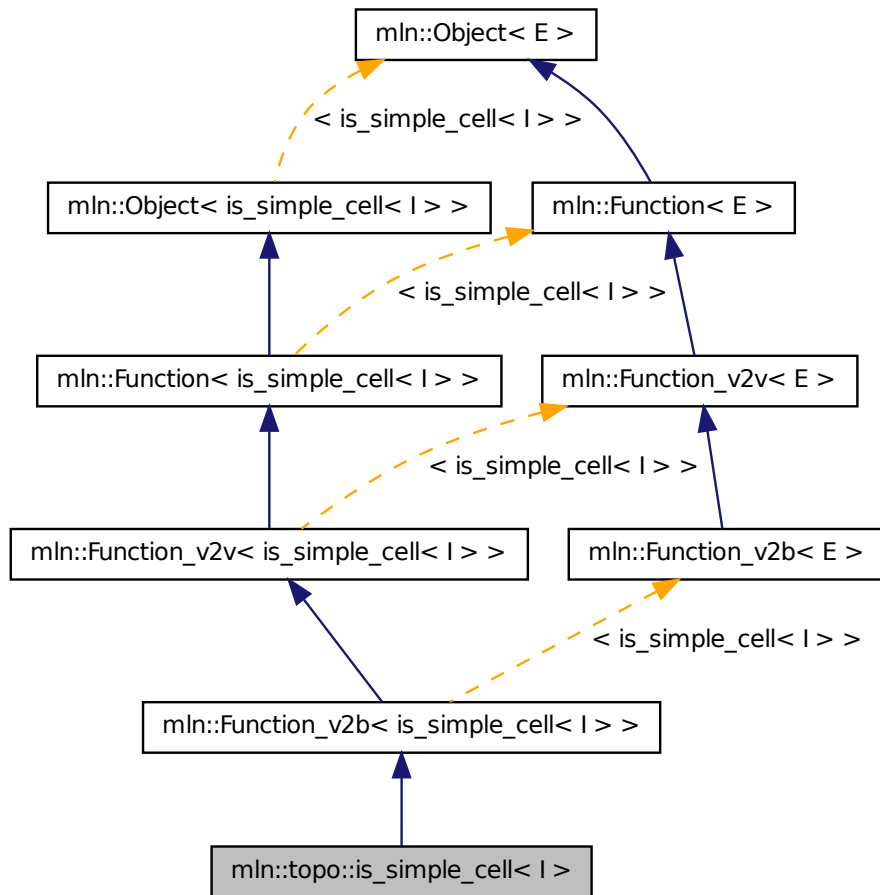
Definition at line 66 of file is_simple_2d.hh.

10.351 mln::topo::is_simple_cell< I > Class Template Reference

A predicate for the simplicity of a point based on the collapse property of the attachment.

```
#include <is_simple_cell.hh>
```

Inheritance diagram for `mln::topo::is_simple_cell< I >`:



Public Types

- typedef `mln::complex_psite< D, G > psite`
Psite type.
- typedef `bool result`
Result type of the functor.

Public Member Functions

- typedef `mln_geom (I) G`
Geometry of the image.

- bool `operator()` (const `mln::complex_psite`< I::dim, `mln_geom`(I)> &p) const
Based on the algorithm A2 from couprie.08.pami.

- void `set_image` (const `mln::Image`< I > &ima)
Set the underlying image.

Static Public Attributes

- static const unsigned `D` = I::dim
Dimension of the image (and therefore of the complex).

10.351.1 Detailed Description

`template<typename I> class mln::topo::is_simple_cell< I >`

A predicate for the simplicity of a point based on the collapse property of the attachment. The functor does not actually take a cell as input, but a face that is expected to be a D-facet.

Definition at line 57 of file `is_simple_cell.hh`.

10.351.2 Member Typedef Documentation

10.351.2.1 `template<typename I > typedef mln::complex_psite<D, G>
mln::topo::is_simple_cell< I >::psite`

Psite type.

Definition at line 65 of file `is_simple_cell.hh`.

10.351.2.2 `template<typename I > typedef bool mln::topo::is_simple_cell< I >::result`

Result type of the functor.

Reimplemented from `mln::Function_v2b< is_simple_cell< I > >`.

Definition at line 68 of file `is_simple_cell.hh`.

10.351.3 Member Function Documentation

10.351.3.1 `template<typename I > typedef mln::topo::is_simple_cell< I >::mln_geom (I)`

Geometry of the image.

10.351.3.2 `template<typename I > bool mln::topo::is_simple_cell< I >::operator() (const
mln::complex_psite< I::dim, mln_geom(I)> & p) const [inline]`

Based on the algorithm A2 from couprie.08.pami.

Definition at line 115 of file `is_simple_cell.hh`.

References `mln::make::attachment()`.

10.351.3.3 `template<typename I> void mln::topo::is_simple_cell< I >::set_image (const mln::Image< I > & ima) [inline]`

Set the underlying image.

Definition at line 107 of file `is_simple_cell.hh`.

10.351.4 Member Data Documentation

10.351.4.1 `template<typename I> const unsigned mln::topo::is_simple_cell< I >::D = I::dim [static]`

Dimension of the image (and therefore of the complex).

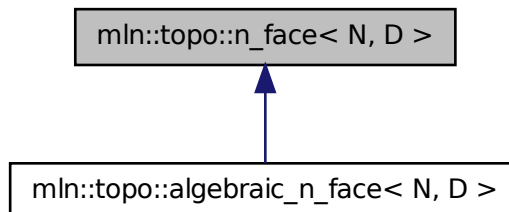
Definition at line 61 of file `is_simple_cell.hh`.

10.352 `mln::topo::n_face< N, D >` Class Template Reference

N-face handle in a complex.

```
#include <n_face.hh>
```

Inheritance diagram for `mln::topo::n_face< N, D >`:



Public Member Functions

- void `invalidate ()`
Invalidate this handle.
- bool `is_valid () const`
Is this handle valid?
- `n_face ()`
Build a non-initialized face handle.

- [n_face](#) ([complex](#)< D > &[complex](#), unsigned face_id)
Build a face handle from complex and face_id.
- [complex](#)< D > [cplx](#) () const
Accessors.
- unsigned [face_id](#) () const
Return the id of the face.
- void [set_cplx](#) (const [complex](#)< D > &cplx)
Set the complex the face belongs to.
- unsigned [n](#) () const
Return the dimension of the face.
- void [set_face_id](#) (unsigned face_id)
Set the id of the face.
- void [inc_face_id](#) ()
Increment the id of the face.
- void [dec_face_id](#) ()
Decrement the id of the face.
- [face_data](#)< N, D > & [data](#) () const
Return the mln::topo::face_data pointed by this handle.
- std::vector< [algebraic_n_face](#)< N-1, D > > [lower_dim_adj_faces](#) () const
Return an array of face handles pointing to adjacent (n-1)-faces.
- std::vector< [algebraic_n_face](#)< N+1, D > > [higher_dim_adj_faces](#) () const
Return an array of face handles pointing to adjacent (n+1)-faces.

10.352.1 Detailed Description

template<unsigned N, unsigned D> class mln::topo::n_face< N, D >

N-face handle in a complex. Contrary to an [mln::topo::face](#), the dimension of an [mln::topo::n_face](#) is fixed.

Definition at line 61 of file n_face.hh.

10.352.2 Constructor & Destructor Documentation

10.352.2.1 template<unsigned N, unsigned D> mln::topo::n_face< N, D >::n_face ()
[inline]

Build a non-initialized face handle.

Definition at line 159 of file n_face.hh.

References [mln::topo::n_face< N, D >::is_valid\(\)](#).

10.352.2.2 `template<unsigned N, unsigned D> mln::topo::n_face< N, D >::n_face (complex< D > & complex, unsigned face_id) [inline]`

Build a face handle from *complex* and *face_id*.

Definition at line 169 of file *n_face.hh*.

10.352.3 Member Function Documentation

10.352.3.1 `template<unsigned N, unsigned D> complex< D > mln::topo::n_face< N, D >::cplx () const [inline]`

Accessors.

Return the complex the face belongs to.

Definition at line 195 of file *n_face.hh*.

Referenced by `mln::topo::n_faces_set< N, D >::add()`, `mln::topo::operator!=()`, and `mln::topo::operator==()`.

10.352.3.2 `template<unsigned N, unsigned D> face_data< N, D > & mln::topo::n_face< N, D >::data () const [inline]`

Return the `mln::topo::face_data` pointed by this handle.

Definition at line 251 of file *n_face.hh*.

References `mln::topo::n_face< N, D >::is_valid()`.

10.352.3.3 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::dec_face_id () [inline]`

Decrement the id of the face.

Definition at line 243 of file *n_face.hh*.

10.352.3.4 `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::face_id () const [inline]`

Return the id of the face.

Definition at line 211 of file *n_face.hh*.

Referenced by `mln::topo::operator==()`.

10.352.3.5 `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N+1, D > > mln::topo::n_face< N, D >::higher_dim_adj_faces () const [inline]`

Return an array of face handles pointing to adjacent (n+1)-faces.

Definition at line 270 of file *n_face.hh*.

References `mln::topo::n_face< N, D >::is_valid()`.

Referenced by `mln::topo::edge()`.

10.352.3.6 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::inc_face_id ()`
`[inline]`

Increment the id of the face.

Definition at line 235 of file n_face.hh.

10.352.3.7 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::invalidate ()`
`[inline]`

Invalidate this handle.

Definition at line 187 of file n_face.hh.

References mln::topo::n_face< N, D >::set_face_id().

10.352.3.8 `template<unsigned N, unsigned D> bool mln::topo::n_face< N, D >::is_valid ()`
`const [inline]`

Is this handle valid?

Definition at line 179 of file n_face.hh.

Referenced by mln::topo::algebraic_n_face< N, D >::algebraic_n_face(), mln::topo::n_face< N, D >::data(), mln::topo::n_face< N, D >::higher_dim_adj_faces(), mln::topo::n_face< N, D >::lower_dim_adj_faces(), and mln::topo::n_face< N, D >::n_face().

10.352.3.9 `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N-1, D > >`
`mln::topo::n_face< N, D >::lower_dim_adj_faces () const [inline]`

Return an array of face handles pointing to adjacent (n-1)-faces.

Definition at line 260 of file n_face.hh.

References mln::topo::n_face< N, D >::is_valid().

10.352.3.10 `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::n ()`
`const [inline]`

Return the dimension of the face.

Definition at line 203 of file n_face.hh.

10.352.3.11 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::set_cplx (const`
`complex< D > & cplx) [inline]`

Set the complex the face belongs to.

Definition at line 219 of file n_face.hh.

10.352.3.12 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::set_face_id (`
`unsigned face_id) [inline]`

Set the id of the face.

Definition at line 227 of file `n_face.hh`.

Referenced by `mln::topo::n_face< N, D >::invalidate()`.

10.353 `mln::topo::n_face_bkd_iter< D >` Class Template Reference

Backward iterator on all the faces of an `mln::complex<D>`.

```
#include <n_face_iter.hh>
```

Inherits `complex_set_iterator_base< topo::face< D >, n_face_bkd_iter< D > >`.

Public Member Functions

- void `next()`
Go to the next element.
- `n_face_bkd_iter()`
Construction and assignment.
- void `start()`
Manipulation.
- unsigned `n()` const
Accessors.

10.353.1 Detailed Description

```
template<unsigned D> class mln::topo::n_face_bkd_iter< D >
```

Backward iterator on all the faces of an `mln::complex<D>`.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 127 of file `n_face_iter.hh`.

10.353.2 Constructor & Destructor Documentation

10.353.2.1 `template<unsigned D> mln::topo::n_face_bkd_iter< D >::n_face_bkd_iter ()`
`[inline]`

Construction and assignment.

Definition at line 256 of file `n_face_iter.hh`.

10.353.3 Member Function Documentation

10.353.3.1 `template<unsigned D> unsigned mln::topo::n_face_bkd_iter< D >::n () const` `[inline]`

Accessors.

Shortcuts to face_'s accessors.

Definition at line 308 of file n_face_iter.hh.

Referenced by mln::topo::n_face_bkd_iter< D >::start().

10.353.3.2 `void mln::Iterator< n_face_bkd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.353.3.3 `template<unsigned D> void mln::topo::n_face_bkd_iter< D >::start () [inline]`

Manipulation.

Start an iteration.

Definition at line 275 of file n_face_iter.hh.

References mln::topo::n_face_bkd_iter< D >::n().

10.354 mln::topo::n_face_fwd_iter< D > Class Template Reference

Forward iterator on all the faces of an mln::complex<D>.

```
#include <n_face_iter.hh>
```

Inherits complex_set_iterator_base< topo::face< D >, n_face_fwd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [n_face_fwd_iter](#) ()
Construction and assignment.

- void `start()`
Manipulation.
- unsigned `n()` const
Accessors.

10.354.1 Detailed Description

`template<unsigned D> class mln::topo::n_face_fwd_iter< D >`

Forward iterator on all the faces of an `mln::complex<D>`.

Template Parameters

D The dimension of the complex this iterator belongs to.

Definition at line 72 of file `n_face_iter.hh`.

10.354.2 Constructor & Destructor Documentation

10.354.2.1 `template<unsigned D> mln::topo::n_face_fwd_iter< D >::n_face_fwd_iter ()`
[`inline`]

Construction and assignment.

Definition at line 183 of file `n_face_iter.hh`.

10.354.3 Member Function Documentation

10.354.3.1 `template<unsigned D> unsigned mln::topo::n_face_fwd_iter< D >::n () const`
[`inline`]

Accessors.

Shortcuts to `face_`'s accessors.

Definition at line 235 of file `n_face_iter.hh`.

10.354.3.2 `void mln::Iterator< n_face_fwd_iter< D > >::next ()` [`inherited`]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.354.3.3 `template<unsigned D> void mln::topo::n_face_fwd_iter< D >::start () [inline]`

Manipulation.

Test if the iterator is valid.

Definition at line 202 of file n_face_iter.hh.

10.355 `mln::topo::n_faces_set< N, D >` Class Template Reference

Set of face handles of dimension N.

```
#include <n_faces_set.hh>
```

Public Types

- typedef `std::vector< algebraic_n_face< N, D > >` `faces_type`
The type of the set of face handles.

Public Member Functions

- void `add` (const `algebraic_n_face< N, D >` &f)
Append an algebraic face f to the set.
- void `reserve` (size_t n)
Reserve n cells in the set.
- const `faces_type` & `faces` () const
Accessors.

10.355.1 Detailed Description

```
template<unsigned N, unsigned D> class mln::topo::n_faces_set< N, D >
```

Set of face handles of dimension N.

Definition at line 56 of file n_faces_set.hh.

10.355.2 Member Typedef Documentation

10.355.2.1 `template<unsigned N, unsigned D> typedef std::vector< algebraic_n_face<N, D> > mln::topo::n_faces_set< N, D >::faces_type`

The type of the set of face handles.

Definition at line 70 of file n_faces_set.hh.

10.355.3 Member Function Documentation

10.355.3.1 `template<unsigned N, unsigned D> void mln::topo::n_faces_set< N, D >::add (const algebraic_n_face< N, D > & f) [inline]`

Append an algebraic face f to the set.

Definition at line 171 of file `n_faces_set.hh`.

References `mln::topo::n_face< N, D >::cplx()`.

Referenced by `mln::topo::operator+()`, and `mln::topo::operator-()`.

10.355.3.2 `template<unsigned N, unsigned D> const std::vector< algebraic_n_face< N, D > > & mln::topo::n_faces_set< N, D >::faces () const [inline]`

Accessors.

Return the set of handles.

Definition at line 190 of file `n_faces_set.hh`.

Referenced by `mln::topo::complex< D >::add_face()`.

10.355.3.3 `template<unsigned N, unsigned D> void mln::topo::n_faces_set< N, D >::reserve (size_t n) [inline]`

Reserve n cells in the set.

This methods does not change the content of `faces_;` it only pre-allocate memory. Method `reserve` is provided for efficiency purpose, and its use is completely optional.

Definition at line 182 of file `n_faces_set.hh`.

10.356 mln::topo::skeleton::is_simple_point< N > Struct Template Reference

```
#include <is_simple_point.hh>
```

10.356.1 Detailed Description

`template<typename N> struct mln::topo::skeleton::is_simple_point< N >`

Tell if a point is simple or not. A point of an object is simple if in its c_8 neighborhood, there is exactly one connected component of the object, and only one connected component of the background Examples : (| == object, - = background)

```
- - |
| P | Here p is simple in the c4 and c8 case.
| | |

- | -
| P | Here p is never simple.
| | |
```

Definition at line 68 of file is_simple_point.hh.

10.357 mln::topo::static_n_face_bkd_iter< N, D > Class Template Reference

Backward iterator on all the *N*-faces of a mln::complex<D>.

```
#include <static_n_face_iter.hh>
```

Inherits complex_set_iterator_base< topo::face< D >, static_n_face_bkd_iter< N, D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [static_n_face_bkd_iter](#) ()
Construction and assignment.
- void [start](#) ()
Manipulation.

10.357.1 Detailed Description

```
template<unsigned N, unsigned D> class mln::topo::static_n_face_bkd_iter< N, D >
```

Backward iterator on all the *N*-faces of a mln::complex<D>.

Template Parameters

- N* The dimension of the face associated to this iterator.
- D* The dimension of the complex this iterator belongs to.

Definition at line 101 of file static_n_face_iter.hh.

10.357.2 Constructor & Destructor Documentation

10.357.2.1 `template<unsigned N, unsigned D> mln::topo::static_n_face_bkd_iter< N, D >::static_n_face_bkd_iter () [inline]`

Construction and assignment.

Definition at line 194 of file static_n_face_iter.hh.

10.357.3 Member Function Documentation

10.357.3.1 `void mln::Iterator< static_n_face_bkd_iter< N, D > >::next ()` [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.357.3.2 `template<unsigned N, unsigned D> void mln::topo::static_n_face_bkd_iter< N, D >::start ()` [inline]

Manipulation.

Start an iteration.

Definition at line 217 of file `static_n_face_iter.hh`.

10.358 `mln::topo::static_n_face_fwd_iter< N, D >` Class Template Reference

Forward iterator on all the `N`-faces of a `mln::complex<D>`.

```
#include <static_n_face_iter.hh>
```

Inherits `complex_set_iterator_base< topo::face< D >, static_n_face_fwd_iter< N, D > >`.

Public Member Functions

- `void next ()`
Go to the next element.
- `static_n_face_fwd_iter ()`
Construction and assignment.
- `void start ()`
Manipulation.

10.358.1 Detailed Description

```
template<unsigned N, unsigned D> class mln::topo::static_n_face_fwd_iter< N, D >
```

Forward iterator on all the `N`-faces of a `mln::complex<D>`.

Template Parameters

- N* The dimension of the face associated to this iterator.
- D* The dimension of the complex this iterator belongs to.

Definition at line 55 of file static_n_face_iter.hh.

10.358.2 Constructor & Destructor Documentation

10.358.2.1 `template<unsigned N, unsigned D> mln::topo::static_n_face_fwd_iter< N, D >::static_n_face_fwd_iter () [inline]`

Construction and assignment.

Definition at line 145 of file static_n_face_iter.hh.

10.358.3 Member Function Documentation

10.358.3.1 `void mln::Iterator< static_n_face_fwd_iter< N, D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.358.3.2 `template<unsigned N, unsigned D> void mln::topo::static_n_face_fwd_iter< N, D >::start () [inline]`

Manipulation.

Test if the iterator is valid.

Definition at line 168 of file static_n_face_iter.hh.

10.359 mln::tr_image< S, I, T > Struct Template Reference

Transform an image by a given transformation.

```
#include <tr_image.hh>
```

Inherits image_identity< I, S, tr_image< S, I, T > >.

Public Types

- typedef I::value [lvalue](#)

Return type of read-write access.

- typedef I::psite [psite](#)
Point_Site associated type.
- typedef I::value [rvalue](#)
Return type of read-only access.
- typedef I::site [site](#)
Site associated type.
- typedef [tr_image](#)< S, tag::image_< I >, T > [skeleton](#)
Skeleton.
- typedef I::value [value](#)
Value associated type.

Public Member Functions

- const S & [domain](#) () const
Return the domain morpher.
- bool [has](#) (const vec_t &v) const
Test if a pixel value is accessible at v.
- bool [is_valid](#) () const
Test if this image has been initialized.
- I::value [operator\(\)](#) (const [psite](#) &p) const
Read-only access of pixel value at point site p.
- void [set_tr](#) (T &tr)
Set the transformation.
- const T & [tr](#) () const
Return the underlying transformation.
- [tr_image](#) (const S &s, const I &ima, const T &tr)
Constructors.

10.359.1 Detailed Description

template<typename S, typename I, typename T> struct mln::tr_image< S, I, T >

Transform an image by a given transformation.

Definition at line 83 of file tr_image.hh.

10.359.2 Member Typedef Documentation

10.359.2.1 `template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::lvalue`

Return type of read-write access.

Definition at line 101 of file tr_image.hh.

10.359.2.2 `template<typename S, typename I, typename T> typedef I ::psite mln::tr_image< S, I, T >::psite`

[Point_Site](#) associated type.

Definition at line 92 of file tr_image.hh.

10.359.2.3 `template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::rvalue`

Return type of read-only access.

Definition at line 104 of file tr_image.hh.

10.359.2.4 `template<typename S, typename I, typename T> typedef I ::site mln::tr_image< S, I, T >::site`

[Site](#) associated type.

Definition at line 95 of file tr_image.hh.

10.359.2.5 `template<typename S, typename I, typename T> typedef tr_image< S, tag::image_<I>, T> mln::tr_image< S, I, T >::skeleton`

Skeleton.

Definition at line 107 of file tr_image.hh.

10.359.2.6 `template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::value`

[Value](#) associated type.

Definition at line 98 of file tr_image.hh.

10.359.3 Constructor & Destructor Documentation

10.359.3.1 `template<typename S, typename I, typename T > mln::tr_image< S, I, T >::tr_image (const S & s, const I & ima, const T & tr) [inline]`

Constructors.

Definition at line 174 of file tr_image.hh.

10.359.4 Member Function Documentation

10.359.4.1 `template<typename S , typename I , typename T > const S & mln::tr_image< S, I, T >::domain () const [inline]`

Return the domain morpher.

Definition at line 247 of file tr_image.hh.

10.359.4.2 `template<typename S , typename I , typename T > bool mln::tr_image< S, I, T >::has (const vec_t & v) const [inline]`

Test if a pixel value is accessible at v.

Definition at line 200 of file tr_image.hh.

10.359.4.3 `template<typename S , typename I , typename T > bool mln::tr_image< S, I, T >::is_valid () const [inline]`

Test if this image has been initialized.

Definition at line 191 of file tr_image.hh.

10.359.4.4 `template<typename S , typename I , typename T > I::value mln::tr_image< S, I, T >::operator() (const psite & p) const [inline]`

Read-only access of pixel value at point site p.

Mutable access is only OK for reading (not writing).

Definition at line 213 of file tr_image.hh.

10.359.4.5 `template<typename S , typename I , typename T > void mln::tr_image< S, I, T >::set_tr (T & tr) [inline]`

Set the transformation.

Definition at line 231 of file tr_image.hh.

10.359.4.6 `template<typename S , typename I , typename T > const T & mln::tr_image< S, I, T >::tr () const [inline]`

Return the underlying transformation.

Definition at line 239 of file tr_image.hh.

10.360 mln::unproject_image< I, D, F > Struct Template Reference

Un-projects an image.

```
#include <unproject_image.hh>
```

Inherits image_domain_morpher< I, D, unproject_image< I, D, F > >.

Public Member Functions

- const D & [domain](#) () const
Give the definition domain.
- internal::morpher_lvalue_< I >::ret [operator](#)() (const typename D::psite &p)
Read-write access to the image value located at point p.
- I::rvalue [operator](#)() (const typename D::psite &p) const
Read-only access to the image value located at point p.
- [unproject_image](#) ()
Constructor without argument.
- [unproject_image](#) (I &ima, const D &dom, const F &f)
Constructor from an image ima, a domain dom, and a function f.

10.360.1 Detailed Description

template<typename I, typename D, typename F> struct mln::unproject_image< I, D, F >

Un-projects an image.

Definition at line 96 of file unproject_image.hh.

10.360.2 Constructor & Destructor Documentation

10.360.2.1 template<typename I, typename D, typename F > mln::unproject_image< I, D, F >::unproject_image () [inline]

Constructor without argument.

Definition at line 182 of file unproject_image.hh.

10.360.2.2 template<typename I, typename D, typename F > mln::unproject_image< I, D, F >::unproject_image (I & ima, const D & dom, const F & f) [inline]

Constructor from an image ima, a domain dom, and a function f.

Definition at line 188 of file unproject_image.hh.

10.360.3 Member Function Documentation

10.360.3.1 template<typename I, typename D, typename F > const D & mln::unproject_image< I, D, F >::domain () const [inline]

Give the definition domain.

Definition at line 205 of file unproject_image.hh.

10.360.3.2 `template<typename I, typename D, typename F > internal::morpher_lvalue_< I >::ret mln::unproject_image< I, D, F >::operator() (const typename D::psite & p) [inline]`

Read-write access to the image value located at point p.

Definition at line 225 of file unproject_image.hh.

10.360.3.3 `template<typename I, typename D, typename F > I::rvalue mln::unproject_image< I, D, F >::operator() (const typename D::psite & p) const [inline]`

Read-only access to the image value located at point p.

Definition at line 214 of file unproject_image.hh.

10.361 mln::util::adjacency_matrix< V > Class Template Reference

A class of adjacency matrix.

```
#include <adjacency_matrix.hh>
```

Inherits adjacency_matrix_impl_selector< V, mln::metal::equal< mln::trait::value_< V >::quant, trait::value::quant::low >::eval >.

Public Member Functions

- [adjacency_matrix \(\)](#)
Constructors.
- [adjacency_matrix \(const V &elements\)](#)
Construct an adjacency matrix with `n` elements elements maximum.

10.361.1 Detailed Description

```
template<typename V = def::coord> class mln::util::adjacency_matrix< V >
```

A class of adjacency matrix. Support low and high quantification value types. In case of low quantification value type, it uses an [image2d](#) to store adjacency information. In case of high quantification value type, it uses a [util::set](#) to store the adjacency information.

Definition at line 136 of file adjacency_matrix.hh.

10.361.2 Constructor & Destructor Documentation

10.361.2.1 `template<typename V > mln::util::adjacency_matrix< V >::adjacency_matrix ()`

Constructors.

```
@{
```

Default

Definition at line 308 of file adjacency_matrix.hh.

10.361.2.2 `template<typename V > mln::util::adjacency_matrix< V >::adjacency_matrix (const V & nelements)`

Construct an adjacency matrix with `nelements` elements maximum.

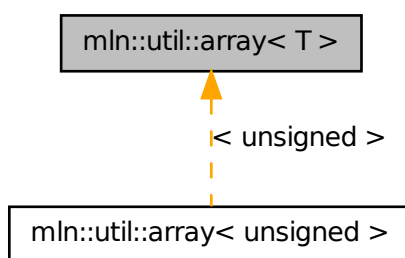
Definition at line 315 of file adjacency_matrix.hh.

10.362 mln::util::array< T > Class Template Reference

A dynamic array class.

```
#include <array.hh>
```

Inheritance diagram for `mln::util::array< T >`:



Public Types

- typedef T [element](#)
Element associated type.
- typedef T [result](#)
Returned value types.
- typedef `array_fwd_iter< T >` [fwd_eiter](#)
Iterator types
Forward iterator associated type.
- typedef `array_bkd_iter< T >` [bkd_eiter](#)
Backward iterator associated type.

- typedef `fwd_eiter eiter`
Iterator associated type.

Public Member Functions

- `array< T > & append (const T &elt)`
Add the element `elt` at the end of this array.
- `template<typename U > array< T > & append (const array< U > &other)`
Add the elements of `other` at the end of this array.
- `void clear ()`
Empty the array.
- `void fill (const T &value)`
Fill the whole array with value `value`.
- `bool is_empty () const`
Test if the array is empty.
- `ro_result last () const`
Return the last element.
- `mutable_result last ()`
Return the last element.
- `std::size_t memory_size () const`
Return the size of this array in memory.
- `unsigned nelements () const`
Return the number of elements of the array.
- `ro_result operator() (unsigned i) const`
Return the `i`-th element of the array.
- `mutable_result operator() (unsigned i)`
Return the `i`-th element of the array.
- `ro_result operator[] (unsigned i) const`
Return the `i`-th element of the array.
- `mutable_result operator[] (unsigned i)`
Return the `i`-th element of the array.
- `void reserve (unsigned n)`
Reserve memory for `n` elements.
- `void resize (unsigned n)`

Resize this array to n elements.

- void [resize](#) (unsigned n , const T &value)
Resize this array to n elements with `value` as value.
- unsigned [size](#) () const
Return the number of elements of the array.
- const std::vector< T > & [std_vector](#) () const
Return the corresponding std::vector of elements.
- [array](#) ()
*Constructors
Constructor without arguments.*
- [array](#) (unsigned n)
Construct a new array and resize it to elements.
- [array](#) (unsigned n , const T &value)
Construct a new array, resize it to elements and fill it with `default_value`.

10.362.1 Detailed Description

template<typename T> class mln::util::array< T >

A dynamic array class. Elements are stored by copy. Implementation is lazy.

The parameter T is the element type, which shall not be const-qualified.

Definition at line 99 of file util/array.hh.

10.362.2 Member Typedef Documentation

10.362.2.1 template<typename T> typedef array_bkd_iter<T> mln::util::array< T >::bkd_eiter

Backward iterator associated type.

Definition at line 124 of file util/array.hh.

10.362.2.2 template<typename T> typedef fwd_eiter mln::util::array< T >::eiter

[Iterator](#) associated type.

Definition at line 127 of file util/array.hh.

10.362.2.3 `template<typename T> typedef T mln::util::array< T >::element`

Element associated type.

Definition at line 107 of file util/array.hh.

10.362.2.4 `template<typename T> typedef array_fwd_iter<T> mln::util::array< T >::fwd_eiter`

[Iterator](#) types

Forward iterator associated type.

Definition at line 121 of file util/array.hh.

10.362.2.5 `template<typename T> typedef T mln::util::array< T >::result`

Returned value types.

Related to the [Function_v2v](#) concept.

Definition at line 112 of file util/array.hh.

10.362.3 Constructor & Destructor Documentation**10.362.3.1** `template<typename T> mln::util::array< T >::array () [inline]`

Constructors

Constructor without arguments.

Definition at line 427 of file util/array.hh.

10.362.3.2 `template<typename T> mln::util::array< T >::array (unsigned n) [inline]`

Construct a new array and resize it to elements.

Definition at line 433 of file util/array.hh.

10.362.3.3 `template<typename T> mln::util::array< T >::array (unsigned n, const T & value) [inline]`

Construct a new array, resize it to elements and fill it with `default_value`.

Definition at line 440 of file util/array.hh.

10.362.4 Member Function Documentation**10.362.4.1** `template<typename T> array< T > & mln::util::array< T >::append (const T & elt) [inline]`

Add the element `elt` at the end of this array.

Definition at line 472 of file util/array.hh.

Referenced by mln::io::dicom::get_header(), and mln::io::plot::load().

10.362.4.2 `template<typename T > template<typename U > array< T > & mln::util::array< T >::append (const array< U > & other) [inline]`

Add the elements of `other` at the end of this array.

Definition at line 482 of file util/array.hh.

References mln::util::array< T >::is_empty(), and mln::util::array< T >::std_vector().

10.362.4.3 `template<typename T > void mln::util::array< T >::clear () [inline]`

Empty the array.

All elements contained in the array are destroyed.

Postcondition

`is_empty() == true`

Definition at line 495 of file util/array.hh.

References mln::util::array< T >::is_empty().

Referenced by mln::io::plot::load().

10.362.4.4 `template<typename T> void mln::util::array< T >::fill (const T & value) [inline]`

Fill the whole array with value `value`.

Definition at line 504 of file util/array.hh.

10.362.4.5 `template<typename T > bool mln::util::array< T >::is_empty () const [inline]`

Test if the array is empty.

Definition at line 578 of file util/array.hh.

References mln::util::array< T >::nelements().

Referenced by mln::util::array< T >::append(), mln::util::array< T >::clear(), mln::make::image3d(), and mln::io::pnms::load().

10.362.4.6 `template<typename T > array< T >::ro_result mln::util::array< T >::last () const [inline]`

Return the last element.

Definition at line 562 of file util/array.hh.

References mln::util::array< T >::nelements().

10.362.4.7 `template<typename T> array< T >::mutable_result mln::util::array< T >::last ()`
[inline]

Return the last element.

Definition at line 570 of file util/array.hh.

References `mln::util::array< T >::nelements()`.

10.362.4.8 `template<typename T> std::size_t mln::util::array< T >::memory_size () const`
[inline]

Return the size of this array in memory.

Definition at line 602 of file util/array.hh.

References `mln::util::array< T >::nelements()`.

10.362.4.9 `template<typename T> unsigned mln::util::array< T >::nelements () const`
[inline]

Return the number of elements of the array.

Definition at line 520 of file util/array.hh.

Referenced by `mln::labeling::fill_holes()`, `mln::make::image3d()`, `mln::util::array< T >::is_empty()`, `mln::util::array< T >::last()`, `mln::io::pnms::load()`, `mln::util::array< T >::memory_size()`, `mln::util::operator<<()`, `mln::util::array< T >::operator[]()`, `mln::io::plot::save()`, and `mln::util::array< T >::size()`.

10.362.4.10 `template<typename T> array< T >::ro_result mln::util::array< T >::operator() (unsigned i) const`
[inline]

Return the *i*-th element of the array.

Precondition

`i < nelements()`

Definition at line 528 of file util/array.hh.

10.362.4.11 `template<typename T> array< T >::mutable_result mln::util::array< T >::operator() (unsigned i)`
[inline]

Return the *i*-th element of the array.

Precondition

`i < nelements()`

Definition at line 536 of file util/array.hh.

10.362.4.12 `template<typename T> array< T >::ro_result mln::util::array< T >::operator[] (unsigned i) const [inline]`

Return the *i*-th element of the array.

Precondition

i < `nelements()`

Definition at line 544 of file util/array.hh.

References `mln::util::array< T >::nelements()`.

10.362.4.13 `template<typename T> array< T >::mutable_result mln::util::array< T >::operator[] (unsigned i) [inline]`

Return the *i*-th element of the array.

Precondition

i < `nelements()`

Definition at line 553 of file util/array.hh.

References `mln::util::array< T >::nelements()`.

10.362.4.14 `template<typename T> void mln::util::array< T >::reserve (unsigned n) [inline]`

Reserve memory for *n* elements.

Definition at line 448 of file util/array.hh.

10.362.4.15 `template<typename T> void mln::util::array< T >::resize (unsigned n) [inline]`

Resize this array to *n* elements.

Definition at line 456 of file util/array.hh.

Referenced by `mln::labeling::impl::generic::compute()`, `mln::labeling::impl::compute_fastest()`, `mln::io::raw::get_header()`, and `mln::io::dump::get_header()`.

10.362.4.16 `template<typename T> void mln::util::array< T >::resize (unsigned n, const T & value) [inline]`

Resize this array to *n* elements with `value` as value.

Definition at line 464 of file util/array.hh.

10.362.4.17 `template<typename T> unsigned mln::util::array< T >::size () const [inline]`

Return the number of elements of the array.

Added for compatibility with `fun::i2v::array`.

See also

[nelements](#)

Definition at line 512 of file util/array.hh.

References `mln::util::array< T >::nelements()`.

Referenced by `mln::labeling::impl::generic::compute()`, `mln::labeling::impl::compute_fastest()`, `mln::value::lut_vec< S, T >::lut_vec()`, and `mln::labeled_image_base< I, E >::update_data()`.

10.362.4.18 `template<typename T> const std::vector< T > & mln::util::array< T >::std_vector () const [inline]`

Return the corresponding `std::vector` of elements.

Definition at line 586 of file util/array.hh.

Referenced by `mln::util::array< T >::append()`, `mln::value::lut_vec< S, T >::lut_vec()`, and `mln::util::operator==()`.

10.363 `mln::util::branch< T >` Class Template Reference

Class of generic branch.

```
#include <tree.hh>
```

Public Member Functions

- [tree_node< T > & apex \(\)](#)
The getter of the appex.
- [branch \(tree< T > &tree, tree_node< T > &apex\)](#)
Constructor.
- [tree< T > & util_tree \(\)](#)
The getter of the tree.

10.363.1 Detailed Description

```
template<typename T> class mln::util::branch< T >
```

Class of generic branch.

Definition at line 249 of file tree.hh.

10.363.2 Constructor & Destructor Documentation

10.363.2.1 `template<typename T> mln::util::branch< T >::branch (util::tree< T > & tree, util::tree_node< T > & apex) [inline]`

Constructor.

Parameters

- [in] *tree* The tree of the branch.
- [in] *apex* The apex of the branch.

Definition at line 537 of file tree.hh.

10.363.3 Member Function Documentation**10.363.3.1** `template<typename T > util::tree_node< T > & mln::util::branch< T >::apex ()`
`[inline]`

The getter of the appex.

Returns

The [tree_node](#) appex of the current branch.

Definition at line 548 of file tree.hh.

10.363.3.2 `template<typename T > mln::util::tree< T > & mln::util::branch< T >::util_tree ()`
`[inline]`

The getter of the tree.

Returns

The tree of the current branch.

Definition at line 556 of file tree.hh.

10.364 mln::util::branch_iter< T > Class Template Reference

Basic 2D image class.

```
#include <branch_iter.hh>
```

Public Member Functions

- unsigned [deepness](#) () const
Give how deep is the iterator in the branch.
- void [invalidate](#) ()
Invalidate the iterator.
- bool [is_valid](#) () const
Test the iterator validity.
- void [next](#) ()
Go to the next point.

- `operator util::tree_node< T > & () const`

Conversion to node.

- `void start ()`

Start an iteration.

10.364.1 Detailed Description

`template<typename T> class mln::util::branch_iter< T >`

Basic 2D image class. The parameter T is the type of node's data. `branch_iter` is used to pre-order walk a branch.

Definition at line 52 of file `branch_iter.hh`.

10.364.2 Member Function Documentation

10.364.2.1 `template<typename T > unsigned mln::util::branch_iter< T >::deepness () const`
[inline]

Give how deep is the iterator in the branch.

Definition at line 119 of file `branch_iter.hh`.

References `mln::util::branch_iter< T >::is_valid()`, and `mln::util::tree_node< T >::parent()`.

10.364.2.2 `template<typename T > void mln::util::branch_iter< T >::invalidate ()`
[inline]

Invalidate the iterator.

Definition at line 143 of file `branch_iter.hh`.

Referenced by `mln::util::branch_iter< T >::next()`.

10.364.2.3 `template<typename T > bool mln::util::branch_iter< T >::is_valid () const`
[inline]

Test the iterator validity.

Definition at line 135 of file `branch_iter.hh`.

Referenced by `mln::util::branch_iter< T >::deepness()`.

10.364.2.4 `template<typename T > void mln::util::branch_iter< T >::next () [inline]`

Go to the next point.

Definition at line 162 of file `branch_iter.hh`.

References `mln::util::branch_iter< T >::invalidate()`.

10.364.2.5 `template<typename T > mln::util::branch_iter< T >::operator util::tree_node< T > & () const [inline]`

Conversion to node.

Definition at line 101 of file `branch_iter.hh`.

10.364.2.6 `template<typename T > void mln::util::branch_iter< T >::start () [inline]`

Start an iteration.

Definition at line 152 of file `branch_iter.hh`.

10.365 `mln::util::branch_iter_ind< T > Class Template Reference`

Basic 2D image class.

```
#include <branch_iter_ind.hh>
```

Public Member Functions

- unsigned `deepness` () const
Give how deep is the iterator in the branch.
- void `invalidate` ()
Invalidate the iterator.
- bool `is_valid` () const
Test the iterator validity.
- void `next` ()
Go to the next point.
- `operator util::tree_node< T > & ()` const
Conversion to node.
- void `start` ()
Start an iteration.

10.365.1 Detailed Description

```
template<typename T> class mln::util::branch_iter_ind< T >
```

Basic 2D image class. The parameter `T` is the type of node's data. `branch_iter_ind` is used to pre-order walk a branch.

Definition at line 66 of file `branch_iter_ind.hh`.

10.365.2 Member Function Documentation

10.365.2.1 `template<typename T> unsigned mln::util::branch_iter_ind< T >::deepness () const [inline]`

Give how deep is the iterator in the branch.

Definition at line 131 of file `branch_iter_ind.hh`.

References `mln::util::branch_iter_ind< T >::is_valid()`, and `mln::util::tree_node< T >::parent()`.

10.365.2.2 `template<typename T> void mln::util::branch_iter_ind< T >::invalidate () [inline]`

Invalidate the iterator.

Definition at line 155 of file `branch_iter_ind.hh`.

Referenced by `mln::util::branch_iter_ind< T >::next()`.

10.365.2.3 `template<typename T> bool mln::util::branch_iter_ind< T >::is_valid () const [inline]`

Test the iterator validity.

Definition at line 147 of file `branch_iter_ind.hh`.

Referenced by `mln::util::branch_iter_ind< T >::deepness()`.

10.365.2.4 `template<typename T> void mln::util::branch_iter_ind< T >::next () [inline]`

Go to the next point.

Definition at line 174 of file `branch_iter_ind.hh`.

References `mln::util::branch_iter_ind< T >::invalidate()`.

10.365.2.5 `template<typename T> mln::util::branch_iter_ind< T >::operator util::tree_node< T > &() const [inline]`

Conversion to node.

Definition at line 113 of file `branch_iter_ind.hh`.

10.365.2.6 `template<typename T> void mln::util::branch_iter_ind< T >::start () [inline]`

Start an iteration.

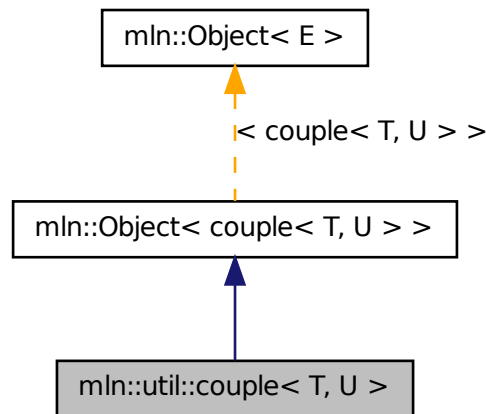
Definition at line 164 of file `branch_iter_ind.hh`.

10.366 mln::util::couple< T, U > Class Template Reference

Definition of a couple.

```
#include <couple.hh>
```

Inheritance diagram for mln::util::couple< T, U >:



Public Member Functions

- void `change_both` (const T &first, const U &second)
Replace both members of the couple by val.
- void `change_first` (const T &val)
Replace the first member of the couple by val.
- void `change_second` (const U &val)
Replace the second member of the couple by val.
- const T & `first` () const
Get the first member of the couple.
- const U & `second` () const
Get the second member of the couple.

10.366.1 Detailed Description

```
template<typename T, typename U> class mln::util::couple< T, U >
```

Definition of a couple.

Definition at line 48 of file util/couple.hh.

10.366.2 Member Function Documentation

10.366.2.1 `template<typename T , typename U > void mln::util::couple< T, U >::change_both (const T & first, const U & second) [inline]`

Replace both members of the couple by *val*.

Definition at line 182 of file util/couple.hh.

10.366.2.2 `template<typename T , typename U > void mln::util::couple< T, U >::change_first (const T & val) [inline]`

Replace the first member of the couple by *val*.

Definition at line 166 of file util/couple.hh.

10.366.2.3 `template<typename T , typename U > void mln::util::couple< T, U >::change_second (const U & val) [inline]`

Replace the second member of the couple by *val*.

Definition at line 174 of file util/couple.hh.

10.366.2.4 `template<typename T , typename U > const T & mln::util::couple< T, U >::first () const [inline]`

Get the first member of the couple.

Definition at line 134 of file util/couple.hh.

10.366.2.5 `template<typename T , typename U > const U & mln::util::couple< T, U >::second () const [inline]`

Get the second member of the couple.

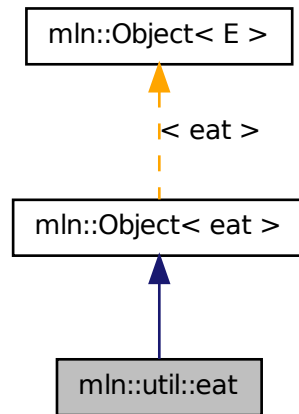
Definition at line 150 of file util/couple.hh.

10.367 mln::util::eat Struct Reference

Eat structure.

```
#include <eat.hh>
```

Inheritance diagram for mln::util::eat:



10.367.1 Detailed Description

Eat structure.

Definition at line 46 of file eat.hh.

10.368 mln::util::edge< G > Class Template Reference

[Edge](#) of a graph G.

```
#include <edge.hh>
```

Inherits mln::util::internal::edge_impl_< G >.

Public Types

- typedef [Edge](#)< void > [category](#)
Object category.
- typedef G [graph_t](#)
Graph associated type.
- typedef [edge_id_t](#) [id_t](#)
The edge type id.
- typedef [edge_id_t::value_t](#) [id_value_t](#)
The underlying type used to store edge ids.

Public Member Functions

- [edge](#) ()
Constructors.
- bool [is_valid](#) () const
Misc.
- void [invalidate](#) ()
Invalidate that vertex.
- [edge_id_t id](#) () const
Return the edge id.
- void [update_id](#) (const [edge_id_t](#) &id)
Set id_ with id;
- operator [edge_id_t](#) () const
Conversion to the edge id.
- const G & [graph](#) () const
Return a reference to the graph holding this edge.
- void [change_graph](#) (const G &g)
Set g_ with g;
- [vertex_id_t v_other](#) (const [vertex_id_t](#) &id_v) const
Vertex and edges oriented.
- [vertex_id_t v1](#) () const
Edge oriented.
- [vertex_id_t v2](#) () const
Return the highest vertex id adjacent to this edge.
- [size_t nmax_nbh_edges](#) () const
Return the number max of adjacent edges.
- [edge_id_t ith_nbh_edge](#) (unsigned i) const
Return the i th adjacent edge.

10.368.1 Detailed Description

```
template<typename G> class mln::util::edge< G >
```

[Edge](#) of a graph G.

Definition at line 69 of file edge.hh.

10.368.2 Member Typedef Documentation

10.368.2.1 `template<typename G> typedef Edge<void> mln::util::edge< G >::category`

[Object](#) category.

Definition at line 73 of file edge.hh.

10.368.2.2 `template<typename G> typedef G mln::util::edge< G >::graph_t`

[Graph](#) associated type.

Definition at line 82 of file edge.hh.

10.368.2.3 `template<typename G> typedef edge_id_t mln::util::edge< G >::id_t`

The edge type id.

Definition at line 79 of file edge.hh.

10.368.2.4 `template<typename G> typedef edge_id_t::value_t mln::util::edge< G >::id_value_t`

The underlying type used to store edge ids.

Definition at line 76 of file edge.hh.

10.368.3 Constructor & Destructor Documentation

10.368.3.1 `template<typename G> mln::util::edge< G >::edge() [inline]`

Constructors.

Definition at line 227 of file edge.hh.

References `mln::util::edge< G >::invalidate()`.

10.368.4 Member Function Documentation

10.368.4.1 `template<typename G> void mln::util::edge< G >::change_graph(const G & g) [inline]`

Set `g_` with `g`;

Definition at line 290 of file edge.hh.

10.368.4.2 `template<typename G> const G & mln::util::edge< G >::graph() const [inline]`

Return a reference to the graph holding this edge.

Definition at line 282 of file edge.hh.

Referenced by `mln::p_edges< G, F >::has()`, and `mln::util::line_graph< G >::has()`.

10.368.4.3 `template<typename G> edge_id_t mln::util::edge< G >::id () const [inline]`

Return the edge id.

Definition at line 259 of file edge.hh.

Referenced by `mln::util::line_graph< G >::has()`.

10.368.4.4 `template<typename G> void mln::util::edge< G >::invalidate () [inline]`

Invalidate that vertex.

Definition at line 306 of file edge.hh.

Referenced by `mln::util::edge< G >::edge()`.

10.368.4.5 `template<typename G> bool mln::util::edge< G >::is_valid () const [inline]`

Misc.

Return whether is points to a known edge.

Definition at line 298 of file edge.hh.

Referenced by `mln::p_edges< G, F >::has()`.

10.368.4.6 `template<typename G> edge_id_t mln::util::edge< G >::ith_nbh_edge (unsigned i) const [inline]`

Return the *i* th adjacent edge.

Definition at line 351 of file edge.hh.

10.368.4.7 `template<typename G> size_t mln::util::edge< G >::nmax_nbh_edges () const [inline]`

Return the number max of adjacent edges.

Definition at line 342 of file edge.hh.

10.368.4.8 `template<typename G> mln::util::edge< G >::operator edge_id_t () const [inline]`

Conversion to the edge id.

Definition at line 274 of file edge.hh.

10.368.4.9 `template<typename G> void mln::util::edge< G >::update_id (const edge_id_t & id) [inline]`

Set `id_` with `id`;

Definition at line 267 of file edge.hh.

10.368.4.10 `template<typename G> vertex_id_t mln::util::edge< G >::v1 () const [inline]`

[Edge](#) oriented.

Return the lowest vertex id adjacent to this edge.

Definition at line 324 of file edge.hh.

Referenced by mln::util::edge< G >::v_other().

10.368.4.11 `template<typename G> vertex_id_t mln::util::edge< G >::v2 () const [inline]`

Return the highest vertex id adjacent to this edge.

Definition at line 333 of file edge.hh.

Referenced by mln::util::edge< G >::v_other().

10.368.4.12 `template<typename G> vertex_id_t mln::util::edge< G >::v_other (const vertex_id_t & id_v) const [inline]`

[Vertex](#) and edges oriented.

Return the vertex id of this edge which is different from `id_v`.

Definition at line 315 of file edge.hh.

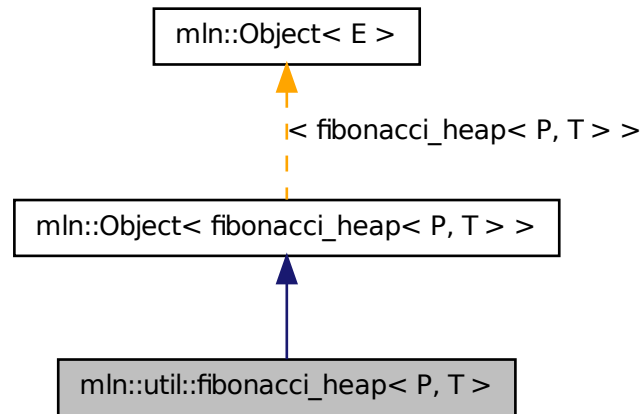
References mln::util::edge< G >::v1(), and mln::util::edge< G >::v2().

10.369 `mln::util::fibonacci_heap< P, T > Class Template Reference`

Fibonacci heap.

```
#include <fibonacci_heap.hh>
```

Inheritance diagram for `mln::util::fibonacci_heap< P, T >`:



Public Member Functions

- void `clear ()`
Clear all elements in the heap and make the heap empty.
- `fibonacci_heap ()`
Default constructor.
- `fibonacci_heap (const fibonacci_heap< P, T > &node)`
Copy constructor Be ware that once this heap is constructed, the argument node is cleared and all its elements are part of this new heap.
- const T & `front () const`
Return the minimum value in the heap.
- bool `is_empty () const`
Is it empty?
- bool `is_valid () const`
return false if it is empty.
- unsigned `nelements () const`
Return the number of elements.
- `fibonacci_heap< P, T > & operator= (fibonacci_heap< P, T > &rhs)`
Assignment operator.
- T `pop_front ()`

Return and remove the minimum value in the heap.

- void `push` (const P &priority, const T &value)
Push a new element in the heap.
- void `push` (fibonacci_heap< P, T > &other_heap)
Take other_heap's elements and insert them in this heap.

10.369.1 Detailed Description

`template<typename P, typename T> class mln::util::fibonacci_heap< P, T >`

Fibonacci heap.

Definition at line 117 of file fibonacci_heap.hh.

10.369.2 Constructor & Destructor Documentation

10.369.2.1 `template<typename P , typename T > mln::util::fibonacci_heap< P, T >::fibonacci_heap () [inline]`

Default constructor.

Definition at line 472 of file fibonacci_heap.hh.

10.369.2.2 `template<typename P , typename T > mln::util::fibonacci_heap< P, T >::fibonacci_heap (const fibonacci_heap< P, T > & node) [inline]`

Copy constructor Be ware that once this heap is constructed, the argument `node` is cleared and all its elements are part of this new heap.

Definition at line 480 of file fibonacci_heap.hh.

10.369.3 Member Function Documentation

10.369.3.1 `template<typename P , typename T > void mln::util::fibonacci_heap< P, T >::clear () [inline]`

Clear all elements in the heap and make the heap empty.

Definition at line 723 of file fibonacci_heap.hh.

References `mln::util::fibonacci_heap< P, T >::pop_front()`.

10.369.3.2 `template<typename P , typename T > const T & mln::util::fibonacci_heap< P, T >::front () const [inline]`

Return the minimum value in the heap.

Definition at line 569 of file fibonacci_heap.hh.

10.369.3.3 `template<typename P , typename T > bool mln::util::fibonacci_heap< P, T >::is_empty () const [inline]`

Is it empty?

Definition at line 705 of file fibonacci_heap.hh.

Referenced by `mln::util::fibonacci_heap< P, T >::pop_front()`, and `mln::util::fibonacci_heap< P, T >::push()`.

10.369.3.4 `template<typename P , typename T > bool mln::util::fibonacci_heap< P, T >::is_valid () const [inline]`

return false if it is empty.

Definition at line 714 of file fibonacci_heap.hh.

Referenced by `mln::util::fibonacci_heap< P, T >::pop_front()`.

10.369.3.5 `template<typename P , typename T > unsigned mln::util::fibonacci_heap< P, T >::nelements () const [inline]`

Return the number of elements.

Definition at line 745 of file fibonacci_heap.hh.

10.369.3.6 `template<typename P , typename T > fibonacci_heap< P, T > & mln::util::fibonacci_heap< P, T >::operator= (fibonacci_heap< P, T > & rhs) [inline]`

Assignment operator.

Be ware that this operator do **not** copy the data from `rhs` to this heap. It moves all elements which means that afterwards, `rhs` is is cleared and all its elements are part of this new heap.

Definition at line 754 of file fibonacci_heap.hh.

10.369.3.7 `template<typename P , typename T > T mln::util::fibonacci_heap< P, T >::pop_front () [inline]`

Return and remove the minimum value in the heap.

Definition at line 578 of file fibonacci_heap.hh.

References `mln::util::fibonacci_heap< P, T >::is_empty()`, `mln::util::fibonacci_heap< P, T >::is_valid()`, and `mln::util::fibonacci_heap< P, T >::push()`.

Referenced by `mln::util::fibonacci_heap< P, T >::clear()`.

10.369.3.8 `template<typename P , typename T > void mln::util::fibonacci_heap< P, T >::push (const P & priority, const T & value) [inline]`

Push a new element in the heap.

See also

insert

Definition at line 508 of file fibonacci_heap.hh.

Referenced by mln::util::fibonacci_heap< P, T >::pop_front().

10.369.3.9 `template<typename P, typename T> void mln::util::fibonacci_heap< P, T >::push (fibonacci_heap< P, T > & other_heap) [inline]`

Take `other_heap`'s elements and insert them in this heap.

After this call `other_heap` is cleared.

Definition at line 520 of file fibonacci_heap.hh.

References mln::util::fibonacci_heap< P, T >::is_empty().

10.370 mln::util::graph Class Reference

Undirected graph.

```
#include <graph.hh>
```

Inherits graph_base< graph >.

Public Types

- typedef std::set< edge_data_t > [edges_set_t](#)
A set to test the presence of a given edge.
- typedef std::vector< edge_data_t > [edges_t](#)
The type of the set of edges.
- typedef std::vector< vertex_data_t > [vertices_t](#)
The type of the set of vertices.
- typedef mln::internal::vertex_fwd_iterator< [graph](#) > [vertex_fwd_iter](#)
Iterator types
Vertex iterators.
- typedef mln::internal::vertex_nbh_edge_fwd_iterator< [graph](#) > [vertex_nbh_edge_fwd_iter](#)
Vertex centered edge iterators.
- typedef mln::internal::vertex_nbh_vertex_fwd_iterator< [graph](#) > [vertex_nbh_vertex_fwd_iter](#)
Vertex centered vertex iterators.
- typedef mln::internal::edge_fwd_iterator< [graph](#) > [edge_fwd_iter](#)

Edge iterators.

- typedef mln::internal::edge_nbh_edge_fwd_iterator< [graph](#) > [edge_nbh_edge_fwd_iter](#)
Edge centered edge iterators.

Public Member Functions

- [graph](#) ()
- [graph](#) (unsigned nvertices)
Construct a graph with nvertices vertices.
- bool [has_v](#) (const [vertex_id_t](#) &id_v) const
Check whether a vertex id id_v exists in the graph.
- [edge_id_t v_ith_nbh_edge](#) (const [vertex_id_t](#) &id_v, unsigned i) const
Returns the i th edge adjacent to the vertex id_v.
- [vertex_id_t v_ith_nbh_vertex](#) (const [vertex_id_t](#) &id_v, unsigned i) const
Returns the i th vertex adjacent to the vertex id_v.
- [size_t v_nmax](#) () const
Return the number of vertices in the graph.
- [size_t v_nmax_nbh_edges](#) (const [vertex_id_t](#) &id_v) const
Return the number of adjacent edges of vertex id_v.
- [size_t v_nmax_nbh_vertices](#) (const [vertex_id_t](#) &id_v) const
Return the number of adjacent vertices of vertex id_v.
- unsigned [add_vertex](#) ()
Vertex oriented.
- std::pair< [vertex_id_t](#), [vertex_id_t](#) > [add_vertices](#) (unsigned n)
Add n vertices to the graph.
- [vertex_t vertex](#) ([vertex_id_t](#) id_v) const
Return the vertex whose id is v.
- [edge_id_t add_edge](#) (const [vertex_id_t](#) &id_v1, const [vertex_id_t](#) &id_v2)
Edge oriented.
- [edge_t edge](#) (const [edge_id_t](#) &e) const
Return the edge whose id is e.
- const std::vector< [util::ord_pair](#)< [vertex_id_t](#) > > & [edges](#) () const
Return the list of all edges.

- `size_t e_nmax () const`
Return the number of edges in the graph.
- `bool has_e (const edge_id_t &id_e) const`
Return whether `id_e` is in the graph.
- `edge_t edge (const vertex_t &v1, const vertex_t &v2) const`
Return the corresponding edge id if exists.
- `vertex_id_t v1 (const edge_id_t &id_e) const`
Return the first vertex associated to the edge `id_e`.
- `vertex_id_t v2 (const edge_id_t &id_e) const`
Return the second vertex associated to edge `id_e`.
- `size_t e_nmax_nbh_edges (const edge_id_t &id_e) const`
Return the number max of adjacent edge, given an edge `id_e`.
- `edge_id_t e_ith_nbh_edge (const edge_id_t &id_e, unsigned i) const`
Return the `i` th edge adjacent to the edge `id_e`.
- `template<typename G2 >`
`bool is_subgraph_of (const G2 &g) const`
*Return whether this graph is a subgraph Return true if `g` and `*this` have the same `graph_id`.*

10.370.1 Detailed Description

Undirected graph.

Definition at line 87 of file `mln/util/graph.hh`.

10.370.2 Member Typedef Documentation

10.370.2.1 `typedef mln::internal::edge_fwd_iterator<graph> mln::util::graph::edge_fwd_iter`

Edge iterators.

Definition at line 129 of file `mln/util/graph.hh`.

10.370.2.2 `typedef mln::internal::edge_nbh_edge_fwd_iterator<graph>` `mln::util::graph::edge_nbh_edge_fwd_iter`

Edge centered edge iterators.

Definition at line 136 of file `mln/util/graph.hh`.

10.370.2.3 `typedef std::set<edge_data_t> mln::util::graph::edges_set_t`

A set to test the presence of a given edge.

Definition at line 102 of file `mln/util/graph.hh`.

10.370.2.4 `typedef std::vector<edge_data_t> mln::util::graph::edges_t`

The type of the set of edges.

Definition at line 100 of file mln/util/graph.hh.

10.370.2.5 `typedef mln::internal::vertex_fwd_iterator<graph> mln::util::graph::vertex_fwd_iter`

[Iterator](#) types

[Vertex](#) iterators.

Definition at line 108 of file mln/util/graph.hh.

10.370.2.6 `typedef mln::internal::vertex_nbh_edge_fwd_iterator<graph> mln::util::graph::vertex_nbh_edge_fwd_iter`

[Vertex](#) centered edge iterators.

Definition at line 115 of file mln/util/graph.hh.

10.370.2.7 `typedef mln::internal::vertex_nbh_vertex_fwd_iterator<graph> mln::util::graph::vertex_nbh_vertex_fwd_iter`

[Vertex](#) centered vertex iterators.

Definition at line 122 of file mln/util/graph.hh.

10.370.2.8 `typedef std::vector<vertex_data_t> mln::util::graph::vertices_t`

The type of the set of vertices.

Definition at line 97 of file mln/util/graph.hh.

10.370.3 Constructor & Destructor Documentation

10.370.3.1 `mln::util::graph::graph () [inline]`

Constructor.

Definition at line 282 of file mln/util/graph.hh.

10.370.3.2 `mln::util::graph::graph (unsigned nvertices) [inline]`

Construct a graph with `nvertices` vertices.

Definition at line 288 of file mln/util/graph.hh.

10.370.4 Member Function Documentation

10.370.4.1 `edge_id_t mln::util::graph::add_edge (const vertex_id_t & id_v1, const vertex_id_t & id_v2) [inline]`

[Edge](#) oriented.

Add an edge.

Returns

The id of the new edge if it does not exist yet; otherwise, return `mln_max(unsigned)`.

Definition at line 386 of file `mln/util/graph.hh`.

References `edge()`, and `has_v()`.

Referenced by `mln::make::voronoi()`.

10.370.4.2 `unsigned mln::util::graph::add_vertex () [inline]`

[Vertex](#) oriented.

Shortcuts factoring the insertion of vertices and edges. Add a vertex.

Returns

The id of the new vertex.

Definition at line 299 of file `mln/util/graph.hh`.

References `v_nmax()`.

Referenced by `mln::make::voronoi()`.

10.370.4.3 `std::pair< vertex_id_t, vertex_id_t > mln::util::graph::add_vertices (unsigned n) [inline]`

Add `n` vertices to the graph.

Returns

A range of vertex ids.

Definition at line 310 of file `mln/util/graph.hh`.

References `v_nmax()`.

10.370.4.4 `edge_id_t mln::util::graph::e_ith_nbh_edge (const edge_id_t & id_e, unsigned i) const [inline]`

Return the `i` th edge adjacent to the edge `id_e`.

Definition at line 503 of file `mln/util/graph.hh`.

References `e_nmax()`, `e_nmax_nbh_edges()`, `has_e()`, `v1()`, `v2()`, `v_ith_nbh_edge()`, and `v_nmax_nbh_edges()`.

10.370.4.5 `size_t mln::util::graph::e_nmax () const [inline]`

Return the number of edges in the graph.

Definition at line 443 of file mln/util/graph.hh.

Referenced by `e_ith_nbh_edge()`, and `edge()`.

10.370.4.6 `size_t mln::util::graph::e_nmax_nbh_edges (const edge_id_t & id_e) const [inline]`

Return the number max of adjacent edge, given an edge `id_e`.

Definition at line 495 of file mln/util/graph.hh.

References `has_e()`, `v1()`, `v2()`, and `v_nmax_nbh_edges()`.

Referenced by `e_ith_nbh_edge()`.

10.370.4.7 `graph::edge_t mln::util::graph::edge (const edge_id_t & e) const [inline]`

Return the edge whose id is `e`.

Definition at line 435 of file mln/util/graph.hh.

References `e_nmax()`.

Referenced by `add_edge()`.

10.370.4.8 `graph::edge_t mln::util::graph::edge (const vertex_t & v1, const vertex_t & v2) const [inline]`

Return the corresponding edge id if exists.

If it is not, returns an invalid edge.

Definition at line 457 of file mln/util/graph.hh.

References `has_v()`.

10.370.4.9 `const std::vector< util::ord_pair< vertex_id_t > > & mln::util::graph::edges () const [inline]`

Return the list of all edges.

Definition at line 428 of file mln/util/graph.hh.

10.370.4.10 `bool mln::util::graph::has_e (const edge_id_t & id_e) const [inline]`

Return whether `id_e` is in the graph.

Definition at line 450 of file mln/util/graph.hh.

Referenced by `e_ith_nbh_edge()`, `e_nmax_nbh_edges()`, `v1()`, and `v2()`.

10.370.4.11 `bool mln::util::graph::has_v (const vertex_id_t & id_v) const [inline]`

Check whether a vertex id `id_v` exists in the graph.

Definition at line 338 of file `mln/util/graph.hh`.

Referenced by `add_edge()`, `edge()`, `v_ith_nbh_edge()`, `v_ith_nbh_vertex()`, `v_nmax_nbh_edges()`, `v_nmax_nbh_vertices()`, and `vertex()`.

10.370.4.12 `template<typename G2 > bool mln::util::graph::is_subgraph_of (const G2 & g) const [inline]`

Return whether this graph is a subgraph Return true if `g` and `*this` have the same `graph_id`.

Definition at line 519 of file `mln/util/graph.hh`.

10.370.4.13 `vertex_id_t mln::util::graph::v1 (const edge_id_t & id_e) const [inline]`

Return the first vertex associated to the edge `id_e`.

Definition at line 479 of file `mln/util/graph.hh`.

References `has_e()`.

Referenced by `e_ith_nbh_edge()`, and `e_nmax_nbh_edges()`.

10.370.4.14 `vertex_id_t mln::util::graph::v2 (const edge_id_t & id_e) const [inline]`

Return the second vertex associated to edge `id_e`.

Definition at line 487 of file `mln/util/graph.hh`.

References `has_e()`.

Referenced by `e_ith_nbh_edge()`, and `e_nmax_nbh_edges()`.

10.370.4.15 `edge_id_t mln::util::graph::v_ith_nbh_edge (const vertex_id_t & id_v, unsigned i) const [inline]`

Returns the `i` th edge adjacent to the vertex `id_v`.

Definition at line 353 of file `mln/util/graph.hh`.

References `has_v()`, and `v_nmax_nbh_edges()`.

Referenced by `e_ith_nbh_edge()`, and `v_ith_nbh_vertex()`.

10.370.4.16 `vertex_id_t mln::util::graph::v_ith_nbh_vertex (const vertex_id_t & id_v, unsigned i) const [inline]`

Returns the `i` th vertex adjacent to the vertex `id_v`.

Definition at line 371 of file `mln/util/graph.hh`.

References `has_v()`, and `v_ith_nbh_edge()`.

10.370.4.17 `size_t mln::util::graph::v_nmax () const [inline]`

Return the number of vertices in the graph.

Definition at line 331 of file mln/util/graph.hh.

Referenced by `add_vertex()`, and `add_vertices()`.

10.370.4.18 `size_t mln::util::graph::v_nmax_nbh_edges (const vertex_id_t & id_v) const [inline]`

Return the number of adjacent edges of vertex `id_v`.

Definition at line 345 of file mln/util/graph.hh.

References `has_v()`.

Referenced by `e_ith_nbh_edge()`, `e_nmax_nbh_edges()`, `v_ith_nbh_edge()`, and `v_nmax_nbh_vertices()`.

10.370.4.19 `size_t mln::util::graph::v_nmax_nbh_vertices (const vertex_id_t & id_v) const [inline]`

Return the number of adjacent vertices of vertex `id_v`.

Definition at line 363 of file mln/util/graph.hh.

References `has_v()`, and `v_nmax_nbh_edges()`.

10.370.4.20 `graph::vertex_t mln::util::graph::vertex (vertex_id_t id_v) const [inline]`

Return the vertex whose id is `v`.

Definition at line 322 of file mln/util/graph.hh.

References `has_v()`.

10.371 mln::util::greater_point< I > Class Template Reference

A “greater than” functor comparing points w.r.t.

```
#include <greater_point.hh>
```

Public Member Functions

- `bool operator() (const point &x, const point &y)`
Is x greater than y?

10.371.1 Detailed Description

```
template<typename I> class mln::util::greater_point< I >
```

A “greater than” functor comparing points w.r.t. the values they refer to in an image.

This functor used in useful to implement ordered queues of points.

Definition at line 42 of file greater_point.hh.

10.371.2 Member Function Documentation

10.371.2.1 `template<typename I> bool mln::util::greater_point< I >::operator() (const point & x, const point & y)`

Is *x* greater than *y*?

Definition at line 74 of file greater_point.hh.

10.372 mln::util::greater_psite< I > Class Template Reference

A “greater than” functor comparing psites w.r.t.

```
#include <greater_psite.hh>
```

Public Member Functions

- `bool operator() (const psite &x, const psite &y)`

Is x greater than y?

10.372.1 Detailed Description

```
template<typename I> class mln::util::greater_psite< I >
```

A “greater than” functor comparing psites w.r.t. the values they refer to in an image.

This functor used in useful to implement ordered queues of psites.

Definition at line 42 of file greater_psite.hh.

10.372.2 Member Function Documentation

10.372.2.1 `template<typename I> bool mln::util::greater_psite< I >::operator() (const psite & x, const psite & y)`

Is *x* greater than *y*?

Definition at line 74 of file greater_psite.hh.

10.373 mln::util::head< T, R > Class Template Reference

Top structure of the soft heap.

```
#include <soft_heap.hh>
```

10.373.1 Detailed Description

```
template<typename T, typename R> class mln::util::head< T, R >
```

Top structure of the soft heap.

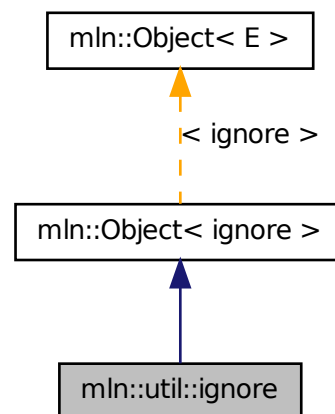
Definition at line 138 of file soft_heap.hh.

10.374 mln::util::ignore Struct Reference

Ignore structure.

```
#include <ignore.hh>
```

Inheritance diagram for mln::util::ignore:



10.374.1 Detailed Description

Ignore structure.

Definition at line 46 of file ignore.hh.

10.375 mln::util::ilcell< T > Struct Template Reference

Element of an item list. Store the data (key) used in [soft_heap](#).

```
#include <soft_heap.hh>
```

10.375.1 Detailed Description

`template<typename T> struct mln::util::ilcell< T >`

Element of an item list. Store the data (key) used in [soft_heap](#).

Definition at line 76 of file `soft_heap.hh`.

10.376 mln::util::line_graph< G > Class Template Reference

Undirected line graph of a graph of type G.

`#include <line_graph.hh>`

Inherits `graph_base< line_graph< G > >`.

Public Types

- `typedef std::vector< edge_data_t > edges_t`
The type of the set of edges.
- `typedef std::vector< vertex_data_t > vertices_t`
The type of the set of vertices.
- `typedef mln::internal::vertex_fwd_iterator< line_graph< G > > vertex_fwd_iter`
Iterator types
Vertex iterators.
- `typedef mln::internal::edge_fwd_iterator< line_graph< G > > edge_fwd_iter`
Edge iterators.
- `typedef mln::internal::edge_nbh_edge_fwd_iterator< line_graph< G > > edge_nbh_edge_fwd_iter`
Edge nbh edge iterators.
- `typedef mln::internal::vertex_nbh_vertex_fwd_iterator< line_graph< G > > vertex_nbh_vertex_fwd_iter`
Vertex nbh vertex iterators.
- `typedef mln::internal::vertex_nbh_edge_fwd_iterator< line_graph< G > > vertex_nbh_edge_fwd_iter`
Vertex nbh edge iterators.

Public Member Functions

- `template<typename G2 >`
`bool has (const util::vertex< G2 > &v) const`
Check whether a vertex `v` exists in the line graph.
- `bool has_v (const vertex_id_t &id_v) const`
Check whether a vertex id `id_v` exists in the line graph.
- `edge_id_t v_ith_nbh_edge (const vertex_id_t &id_v, unsigned i) const`
Returns the `i` th edge adjacent to the vertex `id_v`.
- `vertex_id_t v_ith_nbh_vertex (const vertex_id_t &id_v, unsigned i) const`
Returns the `i` th vertex adjacent to the vertex `id_v`.
- `size_t v_nmax () const`
Return the number of vertices in the graph.
- `size_t v_nmax_nbh_edges (const vertex_id_t &id_v) const`
Return the number of adjacent edges of vertex `id_v`.
- `size_t v_nmax_nbh_vertices (const vertex_id_t &id_v) const`
Return the number of adjacent vertices of vertex `id_v`.

- `vertex_t vertex (const vertex_id_t &id_v) const`
Vertex oriented.

- `edge_t edge (const edge_id_t &e) const`
Edge oriented.
- `size_t e_nmax () const`
Return the number of edges in the graph.
- `bool has_e (const util::edge_id_t &id_e) const`
Return whether `id_e` is in the line graph.
- `template<typename G2 >`
`bool has (const util::edge< G2 > &e) const`
Return whether `e` is in the line graph.
- `vertex_id_t v1 (const edge_id_t &id_e) const`
Return the first vertex associated to the edge `id_e`.
- `vertex_id_t v2 (const edge_id_t &id_e) const`
Return the second vertex associated to edge `id_e`.
- `size_t e_nmax_nbh_edges (const edge_id_t &id_e) const`
Return the number max of adjacent edge, given an edge `id_e`.
- `edge_id_t e_ith_nbh_edge (const edge_id_t &id_e, unsigned i) const`

Return the i th edge adjacent to the edge id_e .

- `template<typename G2 >`
`bool is_subgraph_of (const G2 &g) const`
 Return whether this graph is a subgraph Return true if g and $*this$ have the same $graph_id$.
- `const G & graph () const`
 Return the underlying graph.

10.376.1 Detailed Description

`template<typename G> class mln::util::line_graph< G >`

Undirected line graph of a graph of type G .

Definition at line 82 of file `line_graph.hh`.

10.376.2 Member Typedef Documentation

10.376.2.1 `template<typename G> typedef mln::internal::edge_fwd_iterator< line_graph<G> > mln::util::line_graph< G >::edge_fwd_iter`

[Edge](#) iterators.

Definition at line 114 of file `line_graph.hh`.

10.376.2.2 `template<typename G> typedef mln::internal::edge_nbh_edge_fwd_iterator< line_graph<G> > mln::util::line_graph< G >::edge_nbh_edge_fwd_iter`

[Edge](#) nbh edge iterators.

Definition at line 123 of file `line_graph.hh`.

10.376.2.3 `template<typename G> typedef std::vector<edge_data_t> mln::util::line_graph< G >::edges_t`

The type of the set of edges.

Definition at line 98 of file `line_graph.hh`.

10.376.2.4 `template<typename G> typedef mln::internal::vertex_fwd_iterator< line_graph<G> > mln::util::line_graph< G >::vertex_fwd_iter`

[Iterator](#) types

[Vertex](#) iterators.

Definition at line 105 of file `line_graph.hh`.

10.376.2.5 `template<typename G> typedef mln::internal::vertex_nbh_edge_fwd_iterator< line_graph<G> > mln::util::line_graph< G >::vertex_nbh_edge_fwd_iter`

[Vertex](#) nbh edge iterators.

Definition at line 141 of file line_graph.hh.

10.376.2.6 `template<typename G> typedef mln::internal::vertex_nbh_vertex_fwd_iterator< line_graph<G> > mln::util::line_graph< G >::vertex_nbh_vertex_fwd_iter`

[Vertex](#) nbh vertex iterators.

Definition at line 132 of file line_graph.hh.

10.376.2.7 `template<typename G> typedef std::vector<vertex_data_t> mln::util::line_graph< G >::vertices_t`

The type of the set of vertices.

Definition at line 95 of file line_graph.hh.

10.376.3 Member Function Documentation

10.376.3.1 `template<typename G > edge_id_t mln::util::line_graph< G >::e_ith_nbh_edge (const edge_id_t & id_e, unsigned i) const [inline]`

Return the *i* th edge adjacent to the edge *id_e*.

Definition at line 460 of file line_graph.hh.

References `mln::util::line_graph< G >::e_nmax()`, `mln::util::line_graph< G >::e_nmax_nbh_edges()`, `mln::util::line_graph< G >::has_e()`, `mln::util::line_graph< G >::v1()`, `mln::util::line_graph< G >::v2()`, `mln::util::line_graph< G >::v_ith_nbh_edge()`, and `mln::util::line_graph< G >::v_nmax_nbh_edges()`.

10.376.3.2 `template<typename G > size_t mln::util::line_graph< G >::e_nmax () const [inline]`

Return the number of edges in the graph.

Definition at line 408 of file line_graph.hh.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`, and `mln::util::line_graph< G >::edge()`.

10.376.3.3 `template<typename G > size_t mln::util::line_graph< G >::e_nmax_nbh_edges (const edge_id_t & id_e) const [inline]`

Return the number max of adjacent edge, given an edge *id_e*.

Definition at line 451 of file line_graph.hh.

References `mln::util::line_graph< G >::has_e()`, `mln::util::line_graph< G >::v1()`, `mln::util::line_graph< G >::v2()`, and `mln::util::line_graph< G >::v_nmax_nbh_edges()`.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`.

10.376.3.4 `template<typename G > line_graph< G >::edge_t mln::util::line_graph< G >::edge (const edge_id_t & e) const [inline]`

Edge oriented.

Return the edge whose id is *e*.

Definition at line 399 of file line_graph.hh.

References mln::util::line_graph< G >::e_nmax().

10.376.3.5 `template<typename G > const G & mln::util::line_graph< G >::graph () const [inline]`

Return the underlying graph.

Definition at line 485 of file line_graph.hh.

10.376.3.6 `template<typename G > template<typename G2 > bool mln::util::line_graph< G >::has (const util::vertex< G2 > & v) const [inline]`

Check whether a vertex *v* exists in the line graph.

Definition at line 345 of file line_graph.hh.

References mln::util::vertex< G >::graph(), mln::util::line_graph< G >::has_v(), and mln::util::vertex< G >::id().

10.376.3.7 `template<typename G > template<typename G2 > bool mln::util::line_graph< G >::has (const util::edge< G2 > & e) const [inline]`

Return whether *e* is in the line graph.

Definition at line 425 of file line_graph.hh.

References mln::util::edge< G >::graph(), mln::util::line_graph< G >::has_e(), and mln::util::edge< G >::id().

10.376.3.8 `template<typename G > bool mln::util::line_graph< G >::has_e (const util::edge_id_t & id_e) const [inline]`

Return whether *id_e* is in the line graph.

Definition at line 416 of file line_graph.hh.

Referenced by mln::util::line_graph< G >::e_ith_nbh_edge(), mln::util::line_graph< G >::e_nmax_nbh_edges(), mln::util::line_graph< G >::has(), mln::util::line_graph< G >::v1(), and mln::util::line_graph< G >::v2().

10.376.3.9 `template<typename G > bool mln::util::line_graph< G >::has_v (const vertex_id_t & id_v) const [inline]`

Check whether a vertex id *id_v* exists in the line graph.

Definition at line 336 of file line_graph.hh.

Referenced by `mln::util::line_graph< G >::has()`, `mln::util::line_graph< G >::v_ith_nbh_edge()`, `mln::util::line_graph< G >::v_ith_nbh_vertex()`, `mln::util::line_graph< G >::v_nmax_nbh_edges()`, `mln::util::line_graph< G >::v_nmax_nbh_vertices()`, and `mln::util::line_graph< G >::vertex()`.

10.376.3.10 `template<typename G > template<typename G2 > bool mln::util::line_graph< G >::is_subgraph_of (const G2 & g) const [inline]`

Return whether this graph is a subgraph Return true if `g` and `*this` have the same `graph_id`.

Definition at line 477 of file `line_graph.hh`.

10.376.3.11 `template<typename G > vertex_id_t mln::util::line_graph< G >::v1 (const edge_id_t & id_e) const [inline]`

Return the first vertex associated to the edge `id_e`.

Definition at line 433 of file `line_graph.hh`.

References `mln::util::line_graph< G >::has_e()`.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`, and `mln::util::line_graph< G >::e_nmax_nbh_edges()`.

10.376.3.12 `template<typename G > vertex_id_t mln::util::line_graph< G >::v2 (const edge_id_t & id_e) const [inline]`

Return the second vertex associated to edge `id_e`.

Definition at line 442 of file `line_graph.hh`.

References `mln::util::line_graph< G >::has_e()`.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`, and `mln::util::line_graph< G >::e_nmax_nbh_edges()`.

10.376.3.13 `template<typename G > edge_id_t mln::util::line_graph< G >::v_ith_nbh_edge (const vertex_id_t & id_v, unsigned i) const [inline]`

Returns the `i` th edge adjacent to the vertex `id_v`.

Definition at line 363 of file `line_graph.hh`.

References `mln::util::line_graph< G >::has_v()`, `mln::util::line_graph< G >::v_nmax()`, and `mln::util::line_graph< G >::v_nmax_nbh_edges()`.

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`, and `mln::util::line_graph< G >::v_ith_nbh_vertex()`.

10.376.3.14 `template<typename G > vertex_id_t mln::util::line_graph< G >::v_ith_nbh_vertex (const vertex_id_t & id_v, unsigned i) const [inline]`

Returns the `i` th vertex adjacent to the vertex `id_v`.

Definition at line 383 of file `line_graph.hh`.

References `mln::util::line_graph< G >::has_v()`, and `mln::util::line_graph< G >::v_ith_nbh_edge()`.

10.376.3.15 `template<typename G > size_t mln::util::line_graph< G >::v_nmax () const [inline]`

Return the number of vertices in the graph.

Definition at line 328 of file line_graph.hh.

Referenced by mln::util::line_graph< G >::v_ith_nbh_edge().

10.376.3.16 `template<typename G > size_t mln::util::line_graph< G >::v_nmax_nbh_edges (const vertex_id_t & id_v) const [inline]`

Return the number of adjacent edges of vertex `id_v`.

Definition at line 354 of file line_graph.hh.

References mln::util::line_graph< G >::has_v().

Referenced by mln::util::line_graph< G >::e_ith_nbh_edge(), mln::util::line_graph< G >::e_nmax_nbh_edges(), mln::util::line_graph< G >::v_ith_nbh_edge(), and mln::util::line_graph< G >::v_nmax_nbh_vertices().

10.376.3.17 `template<typename G > size_t mln::util::line_graph< G >::v_nmax_nbh_vertices (const vertex_id_t & id_v) const [inline]`

Return the number of adjacent vertices of vertex `id_v`.

Definition at line 374 of file line_graph.hh.

References mln::util::line_graph< G >::has_v(), and mln::util::line_graph< G >::v_nmax_nbh_edges().

10.376.3.18 `template<typename G > line_graph< G >::vertex_t mln::util::line_graph< G >::vertex (const vertex_id_t & id_v) const [inline]`

[Vertex](#) oriented.

Shortcuts factoring the insertion of vertices and edges.

Return the vertex whose id is `v`.

Definition at line 318 of file line_graph.hh.

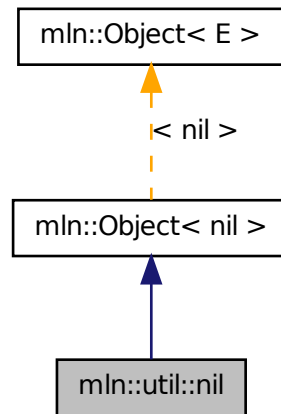
References mln::util::line_graph< G >::has_v().

10.377 mln::util::nil Struct Reference

Nil structure.

```
#include <nil.hh>
```

Inheritance diagram for mln::util::nil:



10.377.1 Detailed Description

Nil structure.

Definition at line 46 of file util/nil.hh.

10.378 mln::util::node< T, R > Class Template Reference

Meta-data of an element in the heap.

```
#include <soft_heap.hh>
```

10.378.1 Detailed Description

```
template<typename T, typename R> class mln::util::node< T, R >
```

Meta-data of an element in the heap.

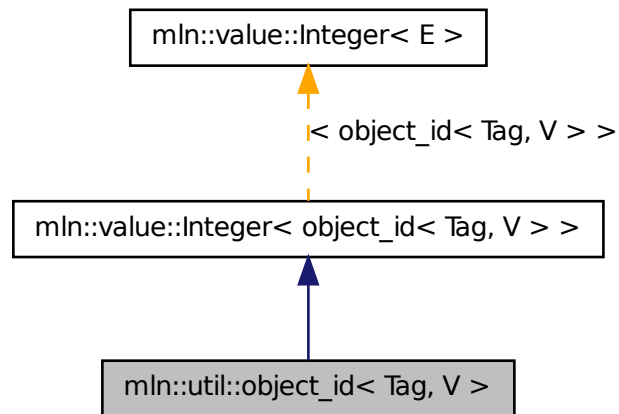
Definition at line 97 of file soft_heap.hh.

10.379 mln::util::object_id< Tag, V > Class Template Reference

Base class of an object id.

```
#include <object_id.hh>
```

Inheritance diagram for mln::util::object_id< Tag, V >:



Public Types

- typedef V [value_t](#)

The underlying type id.

Public Member Functions

- [object_id\(\)](#)

Constructors.

10.379.1 Detailed Description

```
template<typename Tag, typename V> class mln::util::object_id< Tag, V >
```

Base class of an object id.

Template Parameters

Tag the tag type

Equiv the equivalent value.

Definition at line 66 of file object_id.hh.

10.379.2 Member Typedef Documentation

10.379.2.1 `template<typename Tag, typename V> typedef V mln::util::object_id< Tag, V >::value_t`

The underlying type id.

Definition at line 70 of file object_id.hh.

10.379.3 Constructor & Destructor Documentation

10.379.3.1 `template<typename Tag , typename V > mln::util::object_id< Tag, V >::object_id () [inline]`

Constructors.

Definition at line 120 of file object_id.hh.

10.380 mln::util::ord< T > Struct Template Reference

Function-object that defines an ordering between objects with type `T`: *lhs R rhs*.

```
#include <ord.hh>
```

10.380.1 Detailed Description

`template<typename T> struct mln::util::ord< T >`

Function-object that defines an ordering between objects with type `T`: *lhs R rhs*. Its meaning is "lhs less-than rhs."

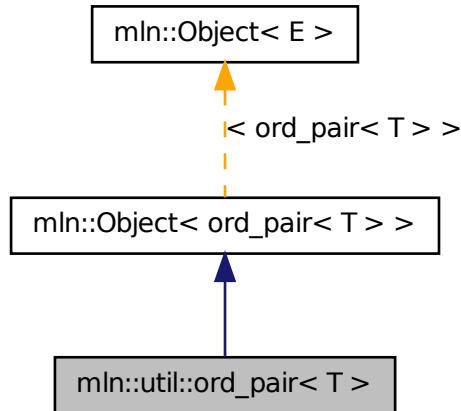
Definition at line 48 of file util/ord.hh.

10.381 mln::util::ord_pair< T > Struct Template Reference

Ordered pair structure s.a.

```
#include <ord_pair.hh>
```


Inheritance diagram for mln::util::ord_pair< T >:



Public Member Functions

- void `change_both` (const T &first, const T &second)
Replace both members of the pair by val, while keeping the relative order.
- void `change_first` (const T &val)
Replace the first member of the pair by val, while keeping the relative order.
- void `change_second` (const T &val)
Replace the second member of the pair by val, while keeping the relative order.
- const T & `first` () const
Get the first (lowest) member of the pair.
- const T & `second` () const
Get the second (highest) member of the pair.

10.381.1 Detailed Description

`template<typename T> struct mln::util::ord_pair< T >`

Ordered pair structure s.a. `this->first <= this->second`; ordered pairs are partially ordered using lexicographical ordering.

Definition at line 50 of file `ord_pair.hh`.

10.381.2 Member Function Documentation

10.381.2.1 `template<typename T> void mln::util::ord_pair< T >::change_both (const T & first, const T & second) [inline]`

Replace both members of the pair by *val*, while keeping the relative order.

Postcondition

first_ <= second_ (with *<=* being the [mln::util::ord_weak](#) relationship).

Definition at line 211 of file `ord_pair.hh`.

References `mln::util::ord_strict()`, and `mln::util::ord_weak()`.

10.381.2.2 `template<typename T> void mln::util::ord_pair< T >::change_first (const T & val) [inline]`

Replace the first member of the pair by *val*, while keeping the relative order.

Postcondition

first_ <= second_ (with *<=* being the [mln::util::ord_weak](#) relationship).

Definition at line 181 of file `ord_pair.hh`.

References `mln::util::ord_strict()`, and `mln::util::ord_weak()`.

10.381.2.3 `template<typename T> void mln::util::ord_pair< T >::change_second (const T & val) [inline]`

Replace the second member of the pair by *val*, while keeping the relative order.

Postcondition

first_ <= second_ (with *<=* being the [mln::util::ord_weak](#) relationship).

Definition at line 196 of file `ord_pair.hh`.

References `mln::util::ord_strict()`, and `mln::util::ord_weak()`.

10.381.2.4 `template<typename T> const T & mln::util::ord_pair< T >::first () const [inline]`

Get the first (lowest) member of the pair.

Definition at line 149 of file `ord_pair.hh`.

10.381.2.5 `template<typename T> const T & mln::util::ord_pair< T >::second () const [inline]`

Get the second (highest) member of the pair.

Definition at line 165 of file `ord_pair.hh`.

10.382 mln::util::pix< I > Struct Template Reference

Structure pix.

```
#include <pix.hh>
```

Public Types

- typedef I::psite [psite](#)
Point_Site associated type.
- typedef I::value [value](#)
Value associated type.

Public Member Functions

- const I & [ima](#) () const
The getter of the image associate to pix structure.
- const I::psite & [p](#) () const
The getter of psite associate to pix structure.
- [pix](#) (const [Image](#)< I > &ima, const typename I::psite &p)
Constructor.
- I::rvalue [v](#) () const
The getter of value associate to pix structure.

10.382.1 Detailed Description

```
template<typename I> struct mln::util::pix< I >
```

Structure pix.

Definition at line 69 of file util/pix.hh.

10.382.2 Member Typedef Documentation

10.382.2.1 template<typename I> typedef I ::psite mln::util::pix< I >::psite

[Point_Site](#) associated type.

Definition at line 73 of file util/pix.hh.

10.382.2.2 template<typename I> typedef I ::value mln::util::pix< I >::value

[Value](#) associated type.

Definition at line 76 of file util/pix.hh.

10.382.3 Constructor & Destructor Documentation

10.382.3.1 `template<typename I> mln::util::pix<I>::pix (const Image<I> & ima, const typename I::psite & p) [inline]`

Constructor.

Parameters

[in] *ima* The image.

[in] *p* The p_site.

Definition at line 121 of file util/pix.hh.

10.382.4 Member Function Documentation

10.382.4.1 `template<typename I> const I & mln::util::pix<I>::ima () const [inline]`

The getter of the image associate to pix structure.

Returns

The image *ima_*.

Definition at line 131 of file util/pix.hh.

10.382.4.2 `template<typename I> const I::psite & mln::util::pix<I>::p () const [inline]`

The getter of psite associate to pix structure.

Returns

The psite *p_*.

Definition at line 140 of file util/pix.hh.

10.382.4.3 `template<typename I> I::rvalue mln::util::pix<I>::v () const [inline]`

The getter of value associate to pix structure.

Returns

The value of *pix*.

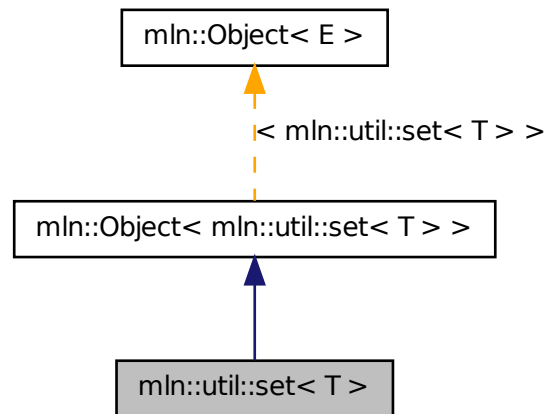
Definition at line 150 of file util/pix.hh.

10.383 mln::util::set< T > Class Template Reference

An "efficient" mathematical set class.

```
#include <set.hh>
```

Inheritance diagram for mln::util::set< T >:



Public Types

- typedef set_bkd_iter< T > [bkd_eiter](#)
Backward iterator associated type.
- typedef [fwd_eiter](#) eiter
Iterator associated type.
- typedef T [element](#)
Element associated type.
- typedef set_fwd_iter< T > [fwd_eiter](#)
Forward iterator associated type.

Public Member Functions

- void [clear](#) ()
Empty the set.
- const T [first_element](#) () const
Return the first element of the set.
- bool [has](#) (const T &elt) const
Test if the object elt belongs to the set.
- set< T > & [insert](#) (const T &elt)

Insert an element `elt` into the set.

- `template<typename U >`
`set< T > & insert (const set< U > &other)`
Insert the elements of `other` into the set.
- `bool is_empty () const`
Test if the set is empty.
- `const T last_element () const`
Return the last element of the set.
- `std::size_t memory_size () const`
Return the size of this set in memory.
- `unsigned nelements () const`
Return the number of elements of the set.
- `const T & operator[] (unsigned i) const`
*Return the *i*-th element of the set.*
- `set< T > & remove (const T &elt)`
Remove an element `elt` into the set.
- `set ()`
Constructor without arguments.
- `const std::vector< T > & std_vector () const`
Give access to the set elements.

10.383.1 Detailed Description

template<typename T> class mln::util::set< T >

An "efficient" mathematical set class. This set class is designed to store a mathematical set and to present it to the user as a linear array (`std::vector`).

Elements are stored by copy. Implementation is lazy.

The set has two states: frozen or not. There is an automatic switch of state when the user modifies its contents (insert, remove, or clear) or access to its contents (`op[i]`).

The parameter `T` is the element type, which shall not be const-qualified.

The unicity of set elements is handled by the `mln::util::ord` mechanism.

See also

[mln::util::ord](#)

Definition at line 81 of file `util/set.hh`.

10.383.2 Member Typedef Documentation

10.383.2.1 `template<typename T> typedef set_bkd_iter<T> mln::util::set< T >::bkd_eiter`

Backward iterator associated type.

Definition at line 93 of file util/set.hh.

10.383.2.2 `template<typename T> typedef fwd_eiter mln::util::set< T >::eiter`

[Iterator](#) associated type.

Definition at line 96 of file util/set.hh.

10.383.2.3 `template<typename T> typedef T mln::util::set< T >::element`

Element associated type.

Definition at line 86 of file util/set.hh.

10.383.2.4 `template<typename T> typedef set_fwd_iter<T> mln::util::set< T >::fwd_eiter`

Forward iterator associated type.

Definition at line 90 of file util/set.hh.

10.383.3 Constructor & Destructor Documentation

10.383.3.1 `template<typename T> mln::util::set< T >::set () [inline]`

Constructor without arguments.

Definition at line 348 of file util/set.hh.

10.383.4 Member Function Documentation

10.383.4.1 `template<typename T> void mln::util::set< T >::clear () [inline]`

Empty the set.

All elements contained in the set are destroyed so the set is emptied.

Postcondition

`is_empty() == true`

Definition at line 390 of file util/set.hh.

References `mln::util::set< T >::is_empty()`.

10.383.4.2 `template<typename T> const T mln::util::set< T >::first_element () const [inline]`

Return the first element of the set.

Precondition

not `is_empty()`

Definition at line 427 of file `util/set.hh`.

References `mln::util::set< T >::is_empty()`.

10.383.4.3 `template<typename T > bool mln::util::set< T >::has (const T & elt) const` `[inline]`

Test if the object `elt` belongs to the set.

Parameters

`[in]` *elt* A possible element of the set.

Returns

True is `elt` is in the set.

Definition at line 445 of file `util/set.hh`.

10.383.4.4 `template<typename T > set< T > & mln::util::set< T >::insert (const T & elt)` `[inline]`

Insert an element `elt` into the set.

Parameters

`[in]` *elt* The element to be inserted.

If `elt` is already in the set, this method is a no-op.

Returns

The set itself after insertion.

Definition at line 356 of file `util/set.hh`.

Referenced by `mln::p_key< K, P >::change_keys()`.

10.383.4.5 `template<typename T > template<typename U > set< T > & mln::util::set< T >::insert (const set< U > & other)` `[inline]`

Insert the elements of `other` into the set.

Parameters

`[in]` *other* The set containing the elements to be inserted.

Returns

The set itself after insertion.

Definition at line 367 of file `util/set.hh`.

References `mln::util::set< T >::is_empty()`, and `mln::util::set< T >::std_vector()`.

10.383.4.6 `template<typename T> bool mln::util::set< T >::is_empty () const [inline]`

Test if the set is empty.

Definition at line 453 of file util/set.hh.

References `mln::util::set< T >::nelements()`.

Referenced by `mln::util::set< T >::clear()`, `mln::util::set< T >::first_element()`, `mln::util::set< T >::insert()`, and `mln::util::set< T >::last_element()`.

10.383.4.7 `template<typename T> const T mln::util::set< T >::last_element () const [inline]`

Return the last element of the set.

Precondition

not `is_empty()`

Definition at line 436 of file util/set.hh.

References `mln::util::set< T >::is_empty()`.

10.383.4.8 `template<typename T> std::size_t mln::util::set< T >::memory_size () const [inline]`

Return the size of this set in memory.

Definition at line 494 of file util/set.hh.

References `mln::util::set< T >::nelements()`.

10.383.4.9 `template<typename T> unsigned mln::util::set< T >::nelements () const [inline]`

Return the number of elements of the set.

Definition at line 409 of file util/set.hh.

Referenced by `mln::util::set< T >::is_empty()`, `mln::util::set< T >::memory_size()`, and `mln::util::set< T >::operator[]()`.

10.383.4.10 `template<typename T> const T & mln::util::set< T >::operator[] (unsigned i) const [inline]`

Return the *i*-th element of the set.

Parameters

[in] *i* Index of the element to retrieve.

Precondition

i < `nelements()`

The element is returned by reference and is constant.

Definition at line 417 of file util/set.hh.

References `mln::util::set< T >::nelements()`.

10.383.4.11 `template<typename T > set< T > & mln::util::set< T >::remove (const T & elt)` [inline]

Remove an element `elt` into the set.

Parameters

[in] *elt* The element to be inserted.

If `elt` is already in the set, this method is a no-op.

Returns

The set itself after suppression.

Definition at line 380 of file util/set.hh.

10.383.4.12 `template<typename T > const std::vector< T > & mln::util::set< T >::std_vector () const` [inline]

Give access to the set elements.

The complexity of this method is O(1).

Postcondition

The set is frozen.

Returns

An array (`std::vector`) of elements.

Definition at line 461 of file util/set.hh.

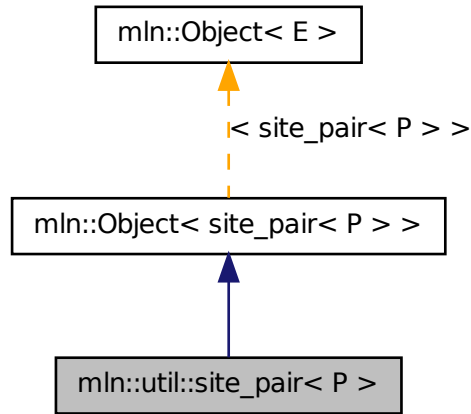
Referenced by `mln::util::set< T >::insert()`.

10.384 `mln::util::site_pair< P >` Class Template Reference

A pair of sites.

```
#include <site_pair.hh>
```

Inheritance diagram for mln::util::site_pair< P >:



Public Member Functions

- const P & [first](#) () const
Return the first site.
- const [util::ord_pair](#)< P > & [pair](#) () const
Return the underlying pair.
- const P & [second](#) () const
Return the second site.

10.384.1 Detailed Description

template<typename P> class mln::util::site_pair< P >

A pair of sites. It can be used as site.

Definition at line 52 of file site_pair.hh.

10.384.2 Member Function Documentation

10.384.2.1 **template<typename P > const P & mln::util::site_pair< P >::first () const**
[inline]

Return the first site.

Definition at line 142 of file site_pair.hh.

10.384.2.2 `template<typename P> const util::ord_pair< P> & mln::util::site_pair< P>::pair () const [inline]`

Return the underlying pair.

Definition at line 158 of file site_pair.hh.

10.384.2.3 `template<typename P> const P & mln::util::site_pair< P>::second () const [inline]`

Return the second site.

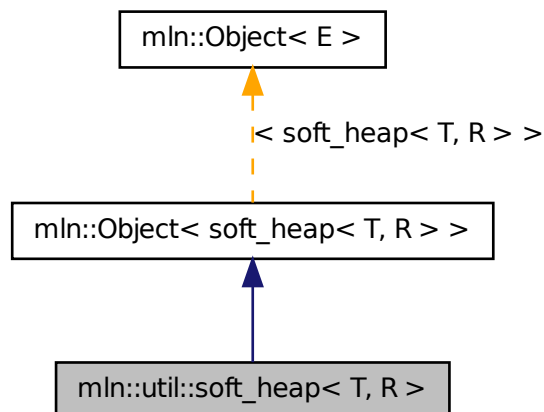
Definition at line 150 of file site_pair.hh.

10.385 mln::util::soft_heap< T, R > Class Template Reference

Soft heap.

```
#include <soft_heap.hh>
```

Inheritance diagram for mln::util::soft_heap< T, R >:



Public Types

- typedef T [element](#)
Element associated type.

Public Member Functions

- void [clear](#) ()

Clear the heap.

- bool `is_empty () const`
Return true if there is at least one element.
- bool `is_valid () const`
Return true if there is at least one element.
- int `nelements () const`
Return the number of element in the heap.
- T `pop_front ()`
Returns the element with the lowest priority and remove it from the heap.
- void `push (soft_heap< T, R > &sh)`
Merge sh with this heap.
- void `push (const T &element)`
Add a new element element.
- `soft_heap (unsigned r=20)`
Default constructor.
- `~soft_heap ()`
Destructor.

10.385.1 Detailed Description

`template<typename T, typename R> class mln::util::soft_heap< T, R >`

Soft heap. T key, the data to store in the heap. For instance a point 2d. R rank, for instance int_u8

Definition at line 178 of file soft_heap.hh.

10.385.2 Member Typedef Documentation

10.385.2.1 `template<typename T, typename R> typedef T mln::util::soft_heap< T, R >::element`

Element associated type.

Definition at line 185 of file soft_heap.hh.

10.385.3 Constructor & Destructor Documentation

10.385.3.1 `template<typename T, typename R > mln::util::soft_heap< T, R >::soft_heap (unsigned r = 20) [inline]`

Default constructor.

A corruption threshold x can be specified. This threshold means that if nodes have a rank higher than this threshold they can be "corrupted" and therefore their rank can be reduced.

Definition at line 619 of file soft_heap.hh.

10.385.3.2 `template<typename T , typename R > mln::util::soft_heap< T, R >::~~soft_heap ()`
[inline]

Destructor.

Definition at line 631 of file soft_heap.hh.

10.385.4 Member Function Documentation

10.385.4.1 `template<typename T , typename R > void mln::util::soft_heap< T, R >::clear ()`
[inline]

Clear the heap.

Definition at line 771 of file soft_heap.hh.

10.385.4.2 `template<typename T , typename R > bool mln::util::soft_heap< T, R >::is_empty () const` **[inline]**

Return true if there is at least one element.

Definition at line 753 of file soft_heap.hh.

10.385.4.3 `template<typename T , typename R > bool mln::util::soft_heap< T, R >::is_valid () const` **[inline]**

Return true if there is at least one element.

Definition at line 744 of file soft_heap.hh.

Referenced by `mln::util::soft_heap< T, R >::pop_front()`.

10.385.4.4 `template<typename T , typename R > int mln::util::soft_heap< T, R >::nelements () const` **[inline]**

Return the number of element in the heap.

Definition at line 762 of file soft_heap.hh.

Referenced by `mln::util::soft_heap< T, R >::push()`.

10.385.4.5 `template<typename T , typename R > T mln::util::soft_heap< T, R >::pop_front ()`
[inline]

Returns the element with the lowest priority and remove it from the heap.

Definition at line 675 of file soft_heap.hh.

References `mln::util::soft_heap< T, R >::is_valid()`.

10.385.4.6 `template<typename T, typename R > void mln::util::soft_heap< T, R >::push (const T & element) [inline]`

Add a new element `element`.

Definition at line 646 of file `soft_heap.hh`.

10.385.4.7 `template<typename T, typename R > void mln::util::soft_heap< T, R >::push (soft_heap< T, R > & sh) [inline]`

Merge `sh` with this heap.

Be ware that after this call, `sh` will be empty. This heap will hold the elements which were part of `sh`.

Definition at line 658 of file `soft_heap.hh`.

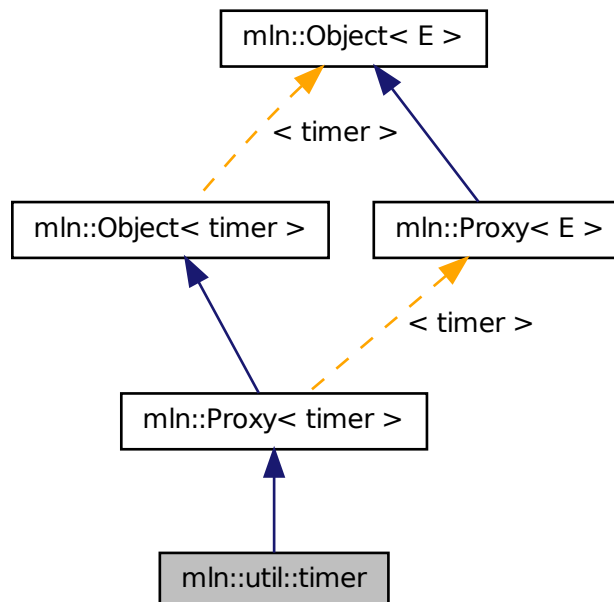
References `mln::util::soft_heap< T, R >::nelements()`.

10.386 mln::util::timer Class Reference

Timer structure.

```
#include <timer.hh>
```

Inheritance diagram for `mln::util::timer`:



10.386.1 Detailed Description

Timer structure.

Definition at line 45 of file mln/util/timer.hh.

10.387 mln::util::tracked_ptr< T > Struct Template Reference

Smart pointer for shared data with tracking.

```
#include <tracked_ptr.hh>
```

Public Member Functions

- `operator bool () const`
Coercion towards Boolean (for arithmetical tests).
- `bool operator! () const`
Negation (for arithmetical tests).
- `T * operator-> ()`
Mimics the behavior of op-> for a pointer in the mutable case.
- `const T * operator-> () const`
Mimics the behavior of op-> for a pointer in the const case.
- `tracked_ptr< T > & operator= (T *ptr)`
Assignment.
- `tracked_ptr< T > & operator= (const tracked_ptr< T > &rhs)`
Assignment.
- `~tracked_ptr ()`
Destructor.
- `tracked_ptr ()`
Constructors.
- `tracked_ptr (const tracked_ptr< T > &rhs)`
Copy constructor.

10.387.1 Detailed Description

```
template<typename T> struct mln::util::tracked_ptr< T >
```

Smart pointer for shared data with tracking.

Definition at line 52 of file tracked_ptr.hh.

10.387.2 Constructor & Destructor Documentation

10.387.2.1 `template<typename T > mln::util::tracked_ptr< T >::tracked_ptr () [inline]`

Constructors.

Definition at line 140 of file tracked_ptr.hh.

10.387.2.2 `template<typename T > mln::util::tracked_ptr< T >::tracked_ptr (const tracked_ptr< T > & rhs) [inline]`

Copy constructor.

Definition at line 164 of file tracked_ptr.hh.

10.387.2.3 `template<typename T > mln::util::tracked_ptr< T >::~~tracked_ptr () [inline]`

Destructor.

Definition at line 216 of file tracked_ptr.hh.

10.387.3 Member Function Documentation

10.387.3.1 `template<typename T > mln::util::tracked_ptr< T >::operator bool () const [inline]`

Coercion towards Boolean (for arithmetical tests).

Definition at line 106 of file tracked_ptr.hh.

10.387.3.2 `template<typename T > bool mln::util::tracked_ptr< T >::operator! () const [inline]`

Negation (for arithmetical tests).

Definition at line 114 of file tracked_ptr.hh.

10.387.3.3 `template<typename T > T * mln::util::tracked_ptr< T >::operator-> () [inline]`

Mimics the behavior of `op->` for a pointer in the mutable case.

Invariant

Pointer proxy exists.

Definition at line 131 of file tracked_ptr.hh.

10.387.3.4 `template<typename T > const T * mln::util::tracked_ptr< T >::operator-> () const [inline]`

Mimics the behavior of `op->` for a pointer in the const case.

Invariant

Pointer proxy exists.

Definition at line 122 of file tracked_ptr.hh.

10.387.3.5 `template<typename T > tracked_ptr< T > & mln::util::tracked_ptr< T >::operator=(T * ptr) [inline]`

Assignment.

Definition at line 195 of file tracked_ptr.hh.

10.387.3.6 `template<typename T > tracked_ptr< T > & mln::util::tracked_ptr< T >::operator=(const tracked_ptr< T > & rhs) [inline]`

Assignment.

Definition at line 176 of file tracked_ptr.hh.

10.388 mln::util::tree< T > Class Template Reference

Class of generic tree.

```
#include <tree.hh>
```

Public Member Functions

- void [add_tree_down](#) (T &elt)
Bind a new tree downer the current.
- void [add_tree_up](#) (T &elt)
Bind a new tree upper the current.
- bool [check_consistency](#) ()
Check the consistency of the tree.
- [branch](#)< T > [main_branch](#) ()
Convert the tree into brach.
- [tree_node](#)< T > * [root](#) ()
The getter of the root.
- [tree](#) ()
Constructor.
- [tree](#) ([tree_node](#)< T > *root)
Constructor.

10.388.1 Detailed Description

template<typename T> class mln::util::tree< T >

Class of generic tree.

Definition at line 187 of file tree.hh.

10.388.2 Constructor & Destructor Documentation

10.388.2.1 template<typename T > mln::util::tree< T >::tree () [inline]

Constructor.

Definition at line 285 of file tree.hh.

10.388.2.2 template<typename T > mln::util::tree< T >::tree (tree_node< T > * root) [inline]

Constructor.

Parameters

[in] *root* The root of the tree.

Definition at line 292 of file tree.hh.

10.388.3 Member Function Documentation

10.388.3.1 template<typename T > void mln::util::tree< T >::add_tree_down (T & elt) [inline]

Bind a new tree downer the current.

Parameters

[in] *elt* The new value of the new [tree_node](#) of the new tree add downer the current.

Definition at line 328 of file tree.hh.

10.388.3.2 template<typename T > void mln::util::tree< T >::add_tree_up (T & elt) [inline]

Bind a new tree upper the current.

Parameters

[in] *elt* The new value of the new [tree_node](#) of the new tree add upper the current.

Definition at line 317 of file tree.hh.

References [mln::util::tree_node< T >::children\(\)](#).

10.388.3.3 `template<typename T> bool mln::util::tree< T >::check_consistency ()` `[inline]`

Check the consistency of the tree.

Returns

true if no error, else false.

Definition at line 338 of file tree.hh.

References `mln::util::tree< T >::root()`.

10.388.3.4 `template<typename T> branch< T > mln::util::tree< T >::main_branch ()` `[inline]`

Convert the tree into brach.

Returns

The root's `tree_node` of the the current tree.

Definition at line 309 of file tree.hh.

References `mln::util::tree< T >::root()`.

10.388.3.5 `template<typename T> tree_node< T > * mln::util::tree< T >::root ()` `[inline]`

The getter of the root.

Returns

The root's `tree_node` of the the current tree.

Definition at line 301 of file tree.hh.

Referenced by `mln::util::tree< T >::check_consistency()`, `mln::util::display_tree()`, `mln::util::tree< T >::main_branch()`, and `mln::util::tree_to_fast()`.

10.389 `mln::util::tree_node< T >` Class Template Reference

Class of generic `tree_node` for tree.

```
#include <tree.hh>
```

Public Member Functions

- `tree_node< T > * add_child (T elt)`
Create a `tree_node` with `elt` which become the child of the current `tree_node`.
- `tree_node< T > * add_child (tree_node< T > *tree_node)`

Bind *tree_node* to the current *tree_node* and become its child.

- bool `check_consistency` ()
Check the consistency of the *tree_node*.
- children_t & `children` ()
The getter of the children.
- const children_t & `children` () const
The getter of the children.
- `tree_node`< T > * `delete_tree_node` ()
Delete the current *tree_node*.
- T & `elt` ()
The getter of the element.
- const T & `elt` () const
The const getter of the element.
- `tree_node`< T > * `parent` ()
The getter of the parent.
- void `print` (std::ostream &ostr, int level=0)
Print on *ostr* the arborescence with the current *tree_node* as root.
- `tree_node`< T > * `search` (T &elt)
Search the *tree_node* with value *elt* in the arborescence of the current *tree_node*.
- int `search_rec` (`tree_node`< T > **res, T &elt)
The using method for method search.
- void `set_parent` (`tree_node`< T > *parent)
Bind *tree_node* to the current *tree_node* and become its parent.
- `tree_node` ()
Constructor.
- `tree_node` (T elt)
Constructor.

10.389.1 Detailed Description

`template<typename T> class mln::util::tree_node< T >`

Class of generic `tree_node` for tree.

Definition at line 58 of file tree.hh.

10.389.2 Constructor & Destructor Documentation

10.389.2.1 `template<typename T> mln::util::tree_node< T >::tree_node () [inline]`

Constructor.

Definition at line 345 of file tree.hh.

10.389.2.2 `template<typename T> mln::util::tree_node< T >::tree_node (T elt) [inline]`

Constructor.

Parameters

[in] *elt* The element of [tree_node](#).

Definition at line 352 of file tree.hh.

10.389.3 Member Function Documentation

10.389.3.1 `template<typename T> tree_node< T > * mln::util::tree_node< T >::add_child (T elt) [inline]`

Create a [tree_node](#) with *elt* which become the child of the current [tree_node](#).

Parameters

[in] *elt* The element of the new child to add.

Returns

The new [tree_node](#) created.

Definition at line 394 of file tree.hh.

10.389.3.2 `template<typename T> tree_node< T > * mln::util::tree_node< T >::add_child (tree_node< T > * tree_node) [inline]`

Bind [tree_node](#) to the current [tree_node](#) and become its child.

Parameters

[in] *tree_node* The new child [tree_node](#).

Returns

The child [tree_node](#).

Definition at line 407 of file tree.hh.

References [mln::util::tree_node< T >::children\(\)](#), and [mln::util::tree_node< T >::parent\(\)](#).

10.389.3.3 `template<typename T> bool mln::util::tree_node< T >::check_consistency ()`
`[inline]`

Check the consistency of the [tree_node](#).

Returns

true if no error, else false.

Definition at line 519 of file tree.hh.

10.389.3.4 `template<typename T> const std::vector< tree_node< T > * > &`
`mln::util::tree_node< T >::children () const [inline]`

The getter of the children.

Returns

The children of the [tree_node](#) in const.

Definition at line 386 of file tree.hh.

10.389.3.5 `template<typename T> std::vector< tree_node< T > * > & mln::util::tree_node< T`
`>::children () [inline]`

The getter of the children.

Returns

The children of the [tree_node](#).

Definition at line 378 of file tree.hh.

Referenced by `mln::util::tree_node< T >::add_child()`, and `mln::util::tree< T >::add_tree_up()`.

10.389.3.6 `template<typename T> tree_node< T > * mln::util::tree_node< T`
`>::delete_tree_node () [inline]`

Delete the current [tree_node](#).

Definition at line 427 of file tree.hh.

10.389.3.7 `template<typename T> const T & mln::util::tree_node< T >::elt () const`
`[inline]`

The const getter of the element.

Returns

The element of the [tree_node](#) in const.

Definition at line 361 of file tree.hh.

10.389.3.8 `template<typename T> T & mln::util::tree_node< T >::elt () [inline]`

The getter of the element.

Returns

The element of the [tree_node](#).

Definition at line 369 of file tree.hh.

Referenced by `mln::util::tree_node< T >::print()`.

10.389.3.9 `template<typename T> tree_node< T > * mln::util::tree_node< T >::parent () [inline]`

The getter of the parent.

Returns

The parent of the [tree_node](#).

Definition at line 477 of file tree.hh.

Referenced by `mln::util::tree_node< T >::add_child()`, `mln::util::branch_iter_ind< T >::deepness()`, and `mln::util::branch_iter< T >::deepness()`.

10.389.3.10 `template<typename T> void mln::util::tree_node< T >::print (std::ostream & ostr, int level = 0) [inline]`

Print on `ostr` the arborescence with the current [tree_node](#) as root.

Parameters

[in] *ostr* The output stream.

[in] *level* The deep level

Definition at line 449 of file tree.hh.

References `mln::util::tree_node< T >::elt()`.

10.389.3.11 `template<typename T> tree_node< T > * mln::util::tree_node< T >::search (T & elt) [inline]`

Search the [tree_node](#) with value `elt` in the arborescence of the current [tree_node](#).

Parameters

[in] *elt* The value of the searched [tree_node](#).

Returns

If not found 0 else the [tree_node](#) with `elt` value.

Definition at line 507 of file tree.hh.

References `mln::util::tree_node< T >::search_rec()`.

10.389.3.12 `template<typename T> int mln::util::tree_node< T >::search_rec (tree_node< T > ** res, T & elt) [inline]`

The using method for method search.

Definition at line 485 of file tree.hh.

Referenced by `mln::util::tree_node< T >::search()`.

10.389.3.13 `template<typename T> void mln::util::tree_node< T >::set_parent (tree_node< T > * parent) [inline]`

Bind `tree_node` to the current `tree_node` and become its parent.

Parameters

[in] *parent* The new parent `tree_node`.

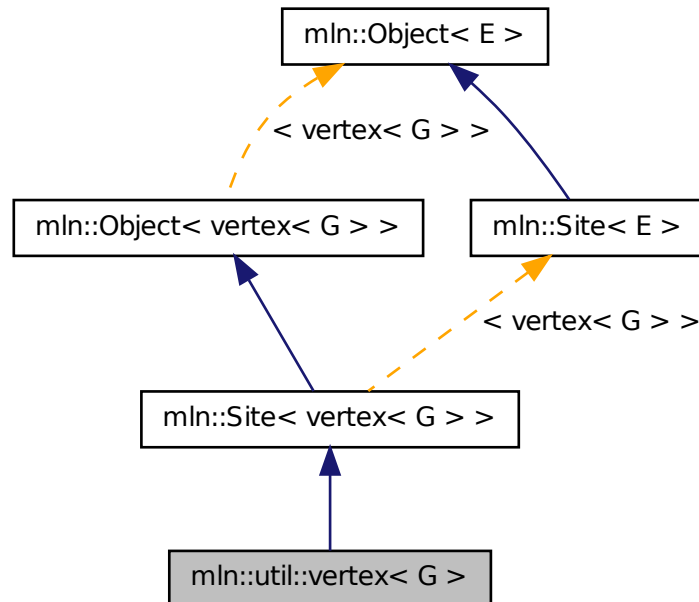
Definition at line 467 of file tree.hh.

10.390 mln::util::vertex< G > Class Template Reference

`Vertex` of a graph `G`.

```
#include <vertex.hh>
```

Inheritance diagram for `mln::util::vertex< G >`:



Public Types

- typedef [Vertex](#)< void > [Category](#)
Object category.
- typedef G [graph_t](#)
Graph associated type.
- typedef [vertex_id_t](#) [id_t](#)
The vertex type id.
- typedef [vertex_id_t::value_t](#) [id_value_t](#)
The underlying type used to store vertex ids.

Public Member Functions

- void [change_graph](#) (const G &g)
Change the parent graph of that vertex.
- [edge](#)< G > [edge_with](#) (const [vertex](#)< G > &v_id) const
Returns true if this vertex has an edge with the given vertex.
- const G & [graph](#) () const
Returns the graph pointer this vertex belongs to.
- const [vertex_id_t](#) & [id](#) () const
Returns the vertex id.
- void [invalidate](#) ()
Invalidate that vertex.
- bool [is_valid](#) () const
Check whether the vertex is still part of the graph.
- [edge_id_t](#) [ith_nbh_edge](#) (unsigned i) const
Returns the ith edge starting from this vertex.
- [vertex_id_t](#) [ith_nbh_vertex](#) (unsigned i) const
Returns the ith vertex adjacent to this vertex.
- unsigned [nmax_nbh_edges](#) () const
Returns the number max of edges starting from this vertex.
- unsigned [nmax_nbh_vertices](#) () const
Returns the number max of vertices adjacent to this vertex.
- operator [vertex_id_t](#) () const
Conversion to the vertex id.

- [vertex_id_t other](#) (const [edge_id_t](#) &id_e) const
Returns the other vertex located on edge id_e.
- void [update_id](#) (const [vertex_id_t](#) &id)
Update the vertex id.
- [vertex](#) ()
Constructors.

10.390.1 Detailed Description

template<typename G> class mln::util::vertex< G >

[Vertex](#) of a graph G.

Definition at line 71 of file vertex.hh.

10.390.2 Member Typedef Documentation

10.390.2.1 template<typename G> typedef Vertex<void> mln::util::vertex< G >::Category

[Object](#) category.

Definition at line 77 of file vertex.hh.

10.390.2.2 template<typename G> typedef G mln::util::vertex< G >::graph_t

[Graph](#) associated type.

Definition at line 86 of file vertex.hh.

10.390.2.3 template<typename G> typedef vertex_id_t mln::util::vertex< G >::id_t

The vertex type id.

Definition at line 83 of file vertex.hh.

10.390.2.4 template<typename G> typedef vertex_id_t::value_t mln::util::vertex< G >::id_value_t

The underlying type used to store vertex ids.

Definition at line 80 of file vertex.hh.

10.390.3 Constructor & Destructor Documentation

10.390.3.1 template<typename G> mln::util::vertex< G >::vertex () [inline]

Constructors.

Definition at line 226 of file vertex.hh.

References mln::util::vertex< G >::invalidate().

10.390.4 Member Function Documentation

10.390.4.1 `template<typename G > void mln::util::vertex< G >::change_graph (const G & g) [inline]`

Change the parent graph of that vertex.

Definition at line 331 of file vertex.hh.

10.390.4.2 `template<typename G > edge< G > mln::util::vertex< G >::edge_with (const vertex< G > & v_id) const [inline]`

Returns true if this vertex has an edge with the given vertex.

Definition at line 321 of file vertex.hh.

10.390.4.3 `template<typename G > const G & mln::util::vertex< G >::graph () const [inline]`

Returns the graph pointer this vertex belongs to.

Definition at line 348 of file vertex.hh.

Referenced by mln::p_vertices< G, F >::has(), mln::util::line_graph< G >::has(), and mln::util::operator==().

10.390.4.4 `template<typename G > const vertex_id_t & mln::util::vertex< G >::id () const [inline]`

Returns the vertex id.

Definition at line 356 of file vertex.hh.

Referenced by mln::util::line_graph< G >::has(), and mln::util::operator==().

10.390.4.5 `template<typename G > void mln::util::vertex< G >::invalidate () [inline]`

Invalidate that vertex.

Definition at line 266 of file vertex.hh.

Referenced by mln::util::vertex< G >::vertex().

10.390.4.6 `template<typename G > bool mln::util::vertex< G >::is_valid () const [inline]`

Check whether the vertex is still part of the graph.

Definition at line 258 of file vertex.hh.

Referenced by mln::p_vertices< G, F >::has().

10.390.4.7 `template<typename G > edge_id_t mln::util::vertex< G >::ith_nbh_edge (unsigned i) const [inline]`

Returns the ith edge starting from this vertex.

Definition at line 285 of file vertex.hh.

10.390.4.8 `template<typename G > vertex_id_t mln::util::vertex< G >::ith_nbh_vertex (unsigned i) const [inline]`

Returns the ith vertex adjacent to this vertex.

Definition at line 303 of file vertex.hh.

10.390.4.9 `template<typename G > unsigned mln::util::vertex< G >::nmax_nbh_edges () const [inline]`

Returns the number max of edges starting from this vertex.

If `g_` is a sub graph of another graph, `nmax` will be retrieved from the initial graph.

Definition at line 294 of file vertex.hh.

10.390.4.10 `template<typename G > unsigned mln::util::vertex< G >::nmax_nbh_vertices () const [inline]`

Returns the number max of vertices adjacent to this vertex.

Definition at line 312 of file vertex.hh.

10.390.4.11 `template<typename G > mln::util::vertex< G >::operator vertex_id_t () const [inline]`

Conversion to the vertex id.

FIXME: May cause ambiguities... :(

Definition at line 363 of file vertex.hh.

10.390.4.12 `template<typename G > vertex_id_t mln::util::vertex< G >::other (const edge_id_t & id_e) const [inline]`

Returns the other vertex located on edge `id_e`.

Definition at line 274 of file vertex.hh.

10.390.4.13 `template<typename G > void mln::util::vertex< G >::update_id (const vertex_id_t & id) [inline]`

Update the vertex id.

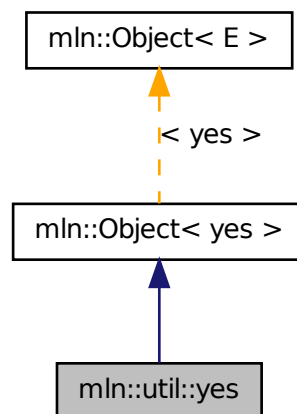
Definition at line 340 of file vertex.hh.

10.391 mln::util::yes Struct Reference

[Object](#) that always says "yes".

```
#include <yes.hh>
```

Inheritance diagram for mln::util::yes:



10.391.1 Detailed Description

[Object](#) that always says "yes".

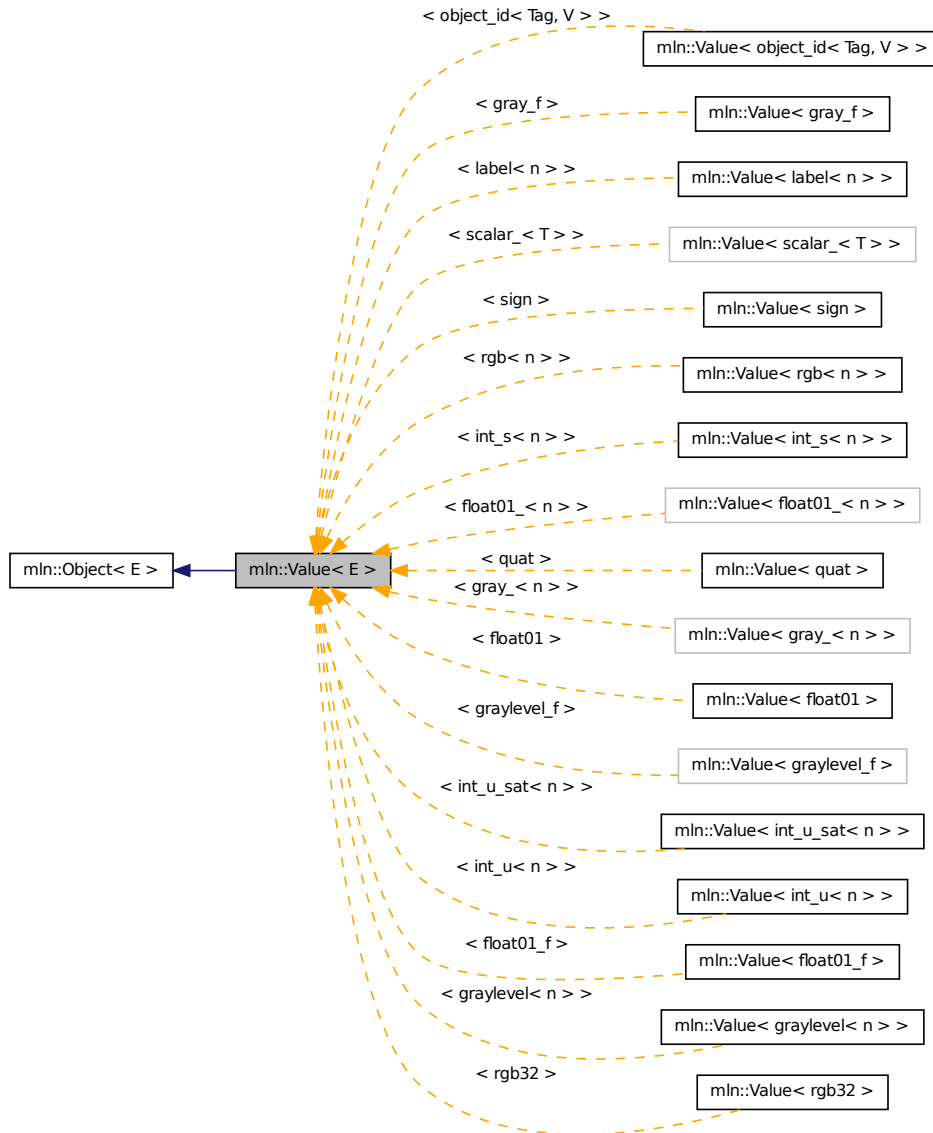
Definition at line 76 of file `yes.hh`.

10.392 mln::Value< E > Struct Template Reference

Base class for implementation classes of values.

```
#include <value.hh>
```

Inheritance diagram for mln::Value< E >:



10.392.1 Detailed Description

`template<typename E> struct mln::Value< E >`

Base class for implementation classes of values.

See also

`mln::doc::Value` for a complete documentation of this class contents.

Definition at line 57 of file core/concept/value.hh.

10.393 mln::value::float01 Class Reference

Class for floating values restricted to the interval [0..1] and discretized with n bits.

```
#include <float01.hh>
```

Inherits mln::value::Floating< float01 >.

Public Types

- typedef std::pair< unsigned, unsigned long > [enc](#)
Encoding associated type.
- typedef float [equiv](#)
Equivalent associated type.

Public Member Functions

- [float01](#) ()
Ctor.
- template<unsigned n>
[float01](#) (const float01_< n > &val)
Ctor.
- [float01](#) (unsigned nbits, float val)
Ctor.
- unsigned [nbits](#) () const
Access to the encoding size.
- operator float () const
Conversion to float.
- [float01](#) & [set_nbits](#) (unsigned nbits)
Set the encoding size to nbits.
- const [float01](#) [to_nbits](#) (unsigned nbits) const
Return an equivalent gray encoded on nbits bits.
- float [value](#) () const
Access to std type.
- unsigned long [value_ind](#) () const
Access to the position in the quantized interval.

10.393.1 Detailed Description

Class for floating values restricted to the interval [0..1] and discretized with n bits.

Definition at line 56 of file float01.hh.

10.393.2 Member Typedef Documentation

10.393.2.1 `typedef std::pair<unsigned, unsigned long> mln::value::float01::enc`

Encoding associated type.

Definition at line 61 of file float01.hh.

10.393.2.2 `typedef float mln::value::float01::equiv`

Equivalent associated type.

Definition at line 64 of file float01.hh.

10.393.3 Constructor & Destructor Documentation

10.393.3.1 `mln::value::float01::float01 () [inline]`

Ctor.

Definition at line 151 of file float01.hh.

10.393.3.2 `template<unsigned n> mln::value::float01::float01 (const float01_<n> & val) [inline]`

Ctor.

Definition at line 158 of file float01.hh.

10.393.3.3 `mln::value::float01::float01 (unsigned nbits, float val) [inline]`

Ctor.

Definition at line 165 of file float01.hh.

10.393.4 Member Function Documentation

10.393.4.1 `unsigned mln::value::float01::nbits () const [inline]`

Access to the encoding size.

Definition at line 186 of file float01.hh.

10.393.4.2 `mln::value::float01::operator float () const [inline]`

Conversion to float.

Definition at line 224 of file float01.hh.

10.393.4.3 float01 & mln::value::float01::set_nbits (unsigned *nbits*) [inline]

Set the encoding size to *nbits*.

Definition at line 193 of file float01.hh.

Referenced by to_nbits().

10.393.4.4 const float01 mln::value::float01::to_nbits (unsigned *nbits*) const [inline]

Return an equivalent gray encoded on *nbits* bits.

Definition at line 214 of file float01.hh.

References set_nbits().

10.393.4.5 float mln::value::float01::value () const [inline]

Access to std type.

Definition at line 172 of file float01.hh.

10.393.4.6 unsigned long mln::value::float01::value_ind () const [inline]

Access to the position in the quantized interval.

Definition at line 179 of file float01.hh.

10.394 mln::value::float01_f Struct Reference

Class for floating values restricted to the interval [0..1].

```
#include <float01_f.hh>
```

Inherits mln::value::Floating< float01_f >, and mln::value::internal::value_like_< float,float,float,float01_f >.

Public Member Functions

- [float01_f\(\)](#)
Constructor without argument.
- [float01_f\(float val\)](#)
Constructor from a float.
- [operator float\(\)](#) const
Conversion to a float.
- [float01_f & operator=\(const float val\)](#)
Assignment from a float.

- float `value` () const
Access to float value.

10.394.1 Detailed Description

Class for floating values restricted to the interval [0..1].

Definition at line 84 of file float01_f.hh.

10.394.2 Constructor & Destructor Documentation

10.394.2.1 mln::value::float01_f::float01_f() [inline]

Constructor without argument.

Definition at line 115 of file float01_f.hh.

10.394.2.2 mln::value::float01_f::float01_f(float val) [inline]

Constructor from a float.

Definition at line 120 of file float01_f.hh.

10.394.3 Member Function Documentation

10.394.3.1 mln::value::float01_f::operator float() const [inline]

Conversion to a float.

Definition at line 145 of file float01_f.hh.

10.394.3.2 float01_f & mln::value::float01_f::operator=(const float val) [inline]

Assignment from a float.

Definition at line 136 of file float01_f.hh.

10.394.3.3 float mln::value::float01_f::value() const [inline]

Access to float value.

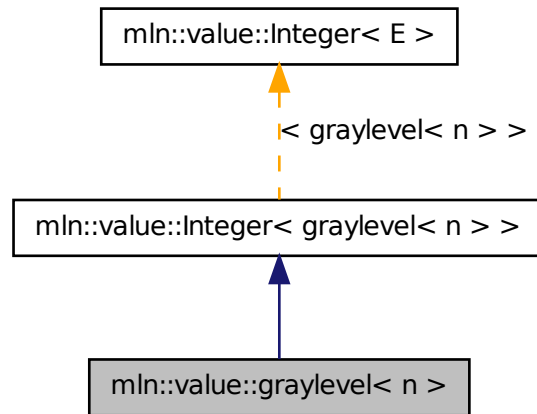
Definition at line 129 of file float01_f.hh.

10.395 mln::value::graylevel< n > Struct Template Reference

General gray-level class on n bits.

```
#include <graylevel.hh>
```

Inheritance diagram for `mln::value::graylevel< n >`:



Public Member Functions

- [graylevel \(\)](#)
Constructor without argument.
- [graylevel \(const \[graylevel< n >\]\(#\) &rhs\)](#)
Copy constructor.
- [graylevel \(int val\)](#)
Constructor from int.
- `template<unsigned m>`
[graylevel \(const \[graylevel< m >\]\(#\) &rhs\)](#)
Constructor from any graylevel.
- [graylevel< n > & operator= \(const \[graylevel< n >\]\(#\) &rhs\)](#)
Assignment.
- [graylevel< n > & operator= \(int val\)](#)
Assignment with int.
- `template<unsigned m>`
[graylevel< n > & operator= \(const \[graylevel< m >\]\(#\) &rhs\)](#)
Assignment with any graylevel.
- `float to_float \(\) const`
Conversion to float between 0 and 1.

- unsigned `value` () const
Access to std type.
- `graylevel` (const `mln::literal::black_t` &)
Ctors with literals.
- `graylevel`< n > & `operator=` (const `mln::literal::black_t` &)
Assigment with literals.

10.395.1 Detailed Description

`template<unsigned n> struct mln::value::graylevel< n >`

General gray-level class on n bits.

Definition at line 257 of file graylevel.hh.

10.395.2 Constructor & Destructor Documentation

10.395.2.1 `template<unsigned n> mln::value::graylevel< n >::graylevel () [inline]`

Constructor without argument.

Definition at line 436 of file graylevel.hh.

10.395.2.2 `template<unsigned n> mln::value::graylevel< n >::graylevel (const graylevel< n > & rhs) [inline]`

Copy constructor.

Definition at line 463 of file graylevel.hh.

10.395.2.3 `template<unsigned n> mln::value::graylevel< n >::graylevel (int val) [inline]`

Constructor from int.

Definition at line 443 of file graylevel.hh.

10.395.2.4 `template<unsigned n> template<unsigned m> mln::value::graylevel< n >::graylevel (const graylevel< m > & rhs) [inline]`

Constructor from any graylevel.

Definition at line 481 of file graylevel.hh.

References `mln::value::graylevel< n >::value()`.

10.395.2.5 `template<unsigned n> mln::value::graylevel<n>::graylevel (const mln::literal::black_t &) [inline]`

ctors with literals.

Definition at line 499 of file graylevel.hh.

10.395.3 Member Function Documentation

10.395.3.1 `template<unsigned n> graylevel<n> & mln::value::graylevel<n>::operator= (const graylevel<n> & rhs) [inline]`

Assignment.

Definition at line 472 of file graylevel.hh.

10.395.3.2 `template<unsigned n> graylevel<n> & mln::value::graylevel<n>::operator= (int val) [inline]`

Assignment with int.

Definition at line 453 of file graylevel.hh.

10.395.3.3 `template<unsigned n> graylevel<n> & mln::value::graylevel<n>::operator= (const mln::literal::black_t &) [inline]`

Assignment with literals.

Definition at line 507 of file graylevel.hh.

10.395.3.4 `template<unsigned n> template<unsigned m> graylevel<n> & mln::value::graylevel<n>::operator= (const graylevel<m> & rhs) [inline]`

Assignment with any graylevel.

Definition at line 490 of file graylevel.hh.

References `mln::value::graylevel<n>::value()`.

10.395.3.5 `template<unsigned n> float mln::value::graylevel<n>::to_float () const [inline]`

Conversion to float between 0 and 1.

Definition at line 557 of file graylevel.hh.

Referenced by `mln::value::graylevel_f::graylevel_f()`, and `mln::value::graylevel_f::operator=()`.

10.395.3.6 `template<unsigned n> unsigned mln::value::graylevel<n>::value () const [inline]`

Access to std type.

Definition at line 549 of file graylevel.hh.

Referenced by `mln::value::graylevel<n>::graylevel()`, and `mln::value::graylevel<n>::operator=()`.

10.396 mln::value::graylevel_f Struct Reference

General gray-level class on n bits.

```
#include <graylevel_f.hh>
```

Inherits `mln::value::Floating<graylevel_f>`, and `mln::value::internal::value_like_<float01_f, float01_f::enc, internal::gray_f, graylevel_f>`.

Public Member Functions

- [graylevel_f\(\)](#)
Constructor without argument.
- [graylevel_f\(const graylevel_f &rhs\)](#)
Copy constructor.
- [graylevel_f\(float val\)](#)
Constructor from float.
- `template<unsigned n>`
[graylevel_f\(const graylevel<n> &rhs\)](#)
Constructor from graylevel.
- `template<unsigned n>`
[operator graylevel<n>\(\) const](#)
Conversion to graylevel<n>.
- [graylevel_f & operator=\(float val\)](#)
Assignment with float.
- [graylevel_f & operator=\(const graylevel_f &rhs\)](#)
Assignment.
- `template<unsigned n>`
[graylevel_f & operator=\(const graylevel<n> &rhs\)](#)
Assignment with graylevel.
- `float value() const`
Access to std type.
- [graylevel_f\(const mln::literal::black_t &\)](#)
Ctors with literals.
- [graylevel_f & operator=\(const mln::literal::black_t &\)](#)
Assignment with literals.

10.396.1 Detailed Description

General gray-level class on n bits.

Definition at line 193 of file graylevel_f.hh.

10.396.2 Constructor & Destructor Documentation

10.396.2.1 `mln::value::graylevel_f::graylevel_f() [inline]`

Constructor without argument.

Definition at line 341 of file graylevel_f.hh.

10.396.2.2 `mln::value::graylevel_f::graylevel_f(const graylevel_f & rhs) [inline]`

Copy constructor.

Definition at line 383 of file graylevel_f.hh.

10.396.2.3 `mln::value::graylevel_f::graylevel_f(float val) [inline]`

Constructor from float.

Definition at line 347 of file graylevel_f.hh.

10.396.2.4 `template<unsigned n> mln::value::graylevel_f::graylevel_f(const graylevel<n> & rhs)`

Constructor from graylevel.

Definition at line 365 of file graylevel_f.hh.

References `mln::value::graylevel<n>::to_float()`.

10.396.2.5 `mln::value::graylevel_f::graylevel_f(const mln::literal::black_t &) [inline]`

Ctors with literals.

Definition at line 400 of file graylevel_f.hh.

10.396.3 Member Function Documentation

10.396.3.1 `template<unsigned n> mln::value::graylevel_f::operator graylevel<n>() const [inline]`

Conversion to `graylevel<n>`.

Definition at line 443 of file graylevel_f.hh.

10.396.3.2 `graylevel_f & mln::value::graylevel_f::operator=(float val) [inline]`

Assignment with float.

Definition at line 356 of file graylevel_f.hh.

10.396.3.3 `template<unsigned n> graylevel_f & mln::value::graylevel_f::operator= (const graylevel< n > & rhs)`

Assignment with graylevel.

Definition at line 374 of file graylevel_f.hh.

References mln::value::graylevel< n >::to_float().

10.396.3.4 `graylevel_f & mln::value::graylevel_f::operator= (const mln::literal::black_t &)`
`[inline]`

Assignment with literals.

Definition at line 407 of file graylevel_f.hh.

10.396.3.5 `graylevel_f & mln::value::graylevel_f::operator= (const graylevel_f & rhs)`
`[inline]`

Assignment.

Definition at line 391 of file graylevel_f.hh.

10.396.3.6 `float mln::value::graylevel_f::value () const` `[inline]`

Access to std type.

Definition at line 450 of file graylevel_f.hh.

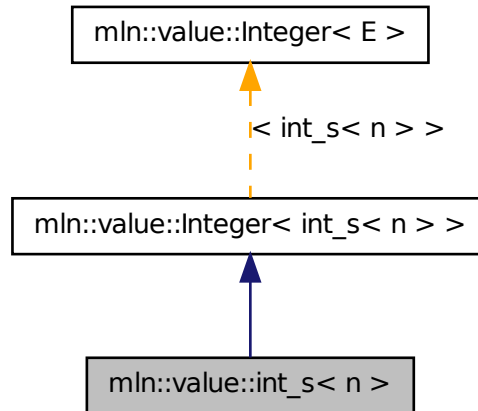
Referenced by mln::value::operator<<().

10.397 mln::value::int_s< n > Struct Template Reference

Signed integer value class.

```
#include <int_s.hh>
```

Inheritance diagram for `mln::value::int_s< n >`:



Public Member Functions

- `int_s ()`
Constructor without argument.
- `int_s (int i)`
Constructor from an integer.
- `operator int () const`
Conversion to an integer.
- `int_s< n > & operator= (int i)`
Assignment from an integer.
- `int_s (const mln::literal::zero_t &)`
Constructors/assignments with literals.

Static Public Attributes

- static const `int_s< n > one = 1`
Unit value.
- static const `int_s< n > zero = 0`
Zero value.

10.397.1 Detailed Description

`template<unsigned n> struct mln::value::int_s< n >`

Signed integer value class. The parameter is `n` the number of encoding bits.

Definition at line 115 of file `int_s.hh`.

10.397.2 Constructor & Destructor Documentation

10.397.2.1 `template<unsigned n> mln::value::int_s< n >::int_s () [inline]`

Constructor without argument.

Definition at line 179 of file `int_s.hh`.

10.397.2.2 `template<unsigned n> mln::value::int_s< n >::int_s (int i) [inline]`

Constructor from an integer.

Definition at line 192 of file `int_s.hh`.

10.397.2.3 `template<unsigned n> mln::value::int_s< n >::int_s (const mln::literal::zero_t &) [inline]`

Constructors/assignments with literals.

Definition at line 222 of file `int_s.hh`.

10.397.3 Member Function Documentation

10.397.3.1 `template<unsigned n> mln::value::int_s< n >::operator int () const [inline]`

Conversion to an integer.

Definition at line 185 of file `int_s.hh`.

10.397.3.2 `template<unsigned n> int_s< n > & mln::value::int_s< n >::operator= (int i) [inline]`

Assignment from an integer.

Definition at line 207 of file `int_s.hh`.

10.397.4 Member Data Documentation

10.397.4.1 `template<unsigned n> const int_s< n > mln::value::int_s< n >::one = 1 [static]`

Unit value.

Definition at line 149 of file `int_s.hh`.

10.397.4.2 `template<unsigned n> const int_s< n > mln::value::int_s< n >::zero = 0` [static]

Zero value.

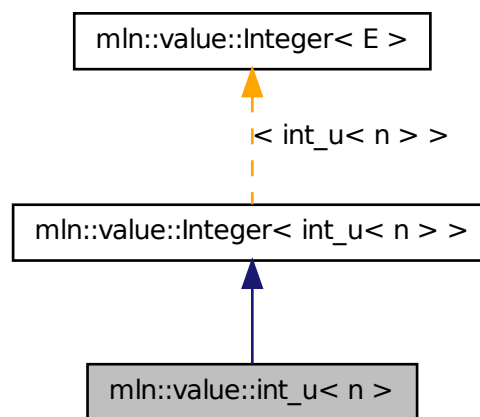
Definition at line 146 of file int_s.hh.

10.398 `mln::value::int_u< n >` Struct Template Reference

Unsigned integer value class.

```
#include <int_u.hh>
```

Inheritance diagram for `mln::value::int_u< n >`:



Public Member Functions

- `int_u ()`
Constructor without argument.
- `int_u (int i)`
Constructor from an integer.
- `int_u< n > next () const`
Give the next value (i.e., $i + 1$).
- `operator unsigned () const`
Conversion to an unsigned integer.
- `int operator- () const`
Unary operator minus.

- `int_u< n > & operator= (int i)`
Assignment from an integer.
- `int_u (const mln::literal::zero_t &)`
Constructors/assignments with literals.

10.398.1 Detailed Description

`template<unsigned n> struct mln::value::int_u< n >`

Unsigned integer value class. The parameter is `n` the number of encoding bits.

Definition at line 156 of file `int_u.hh`.

10.398.2 Constructor & Destructor Documentation

10.398.2.1 `template<unsigned n> mln::value::int_u< n >::int_u () [inline]`

Constructor without argument.

Definition at line 276 of file `int_u.hh`.

10.398.2.2 `template<unsigned n> mln::value::int_u< n >::int_u (int i) [inline]`

Constructor from an integer.

Definition at line 282 of file `int_u.hh`.

10.398.2.3 `template<unsigned n> mln::value::int_u< n >::int_u (const mln::literal::zero_t &) [inline]`

Constructors/assignments with literals.

Definition at line 291 of file `int_u.hh`.

10.398.3 Member Function Documentation

10.398.3.1 `template<unsigned n> int_u< n > mln::value::int_u< n >::next () const [inline]`

Give the next value (i.e., `i + 1`).

Definition at line 350 of file `int_u.hh`.

10.398.3.2 `template<unsigned n> mln::value::int_u< n >::operator unsigned () const [inline]`

Conversion to an unsigned integer.

Definition at line 323 of file int_u.hh.

10.398.3.3 `template<unsigned n> int mln::value::int_u< n >::operator-() const [inline]`

Unary operator minus.

Definition at line 331 of file int_u.hh.

10.398.3.4 `template<unsigned n> int_u< n > & mln::value::int_u< n >::operator=(int i) [inline]`

Assignment from an integer.

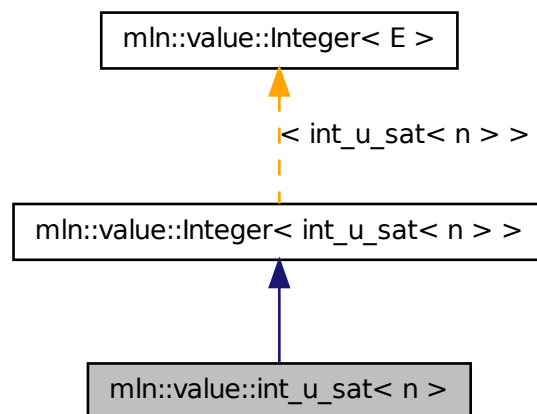
Definition at line 339 of file int_u.hh.

10.399 mln::value::int_u_sat< n > Struct Template Reference

Unsigned integer value class with saturation behavior.

```
#include <int_u_sat.hh>
```

Inheritance diagram for mln::value::int_u_sat< n >:



Public Member Functions

- `int_u_sat ()`
Constructor without argument.
- `int_u_sat (int i)`
Constructor from an integer.

- `operator int () const`
Conversion to an integer.
- `int_u_sat< n > & operator+= (int i)`
Self addition.
- `int_u_sat< n > & operator-= (int i)`
Self subtraction.
- `int_u_sat< n > & operator= (int i)`
Assignment from an integer.

Static Public Attributes

- static const `int_u_sat< n > one = 1`
Unit value.
- static const `int_u_sat< n > zero = 0`
Zero value.

10.399.1 Detailed Description

`template<unsigned n> struct mln::value::int_u_sat< n >`

Unsigned integer value class with saturation behavior. The parameter is `n` the number of encoding bits. Definition at line 90 of file `int_u_sat.hh`.

10.399.2 Constructor & Destructor Documentation

10.399.2.1 `template<unsigned n> mln::value::int_u_sat< n >::int_u_sat () [inline]`

Constructor without argument.

Definition at line 149 of file `int_u_sat.hh`.

10.399.2.2 `template<unsigned n> mln::value::int_u_sat< n >::int_u_sat (int i) [inline]`

Constructor from an integer.

Definition at line 155 of file `int_u_sat.hh`.

10.399.3 Member Function Documentation

10.399.3.1 `template<unsigned n> mln::value::int_u_sat< n >::operator int () const [inline]`

Conversion to an integer.

Definition at line 170 of file int_u_sat.hh.

10.399.3.2 `template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator+=(int i) [inline]`

Self addition.

Definition at line 195 of file int_u_sat.hh.

10.399.3.3 `template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator-=(int i) [inline]`

Self subtraction.

Definition at line 205 of file int_u_sat.hh.

10.399.3.4 `template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator=(int i) [inline]`

Assignment from an integer.

Definition at line 178 of file int_u_sat.hh.

10.399.4 Member Data Documentation

10.399.4.1 `template<unsigned n> const int_u_sat< n > mln::value::int_u_sat< n >::one = 1 [static]`

Unit value.

Definition at line 115 of file int_u_sat.hh.

10.399.4.2 `template<unsigned n> const int_u_sat< n > mln::value::int_u_sat< n >::zero = 0 [static]`

Zero value.

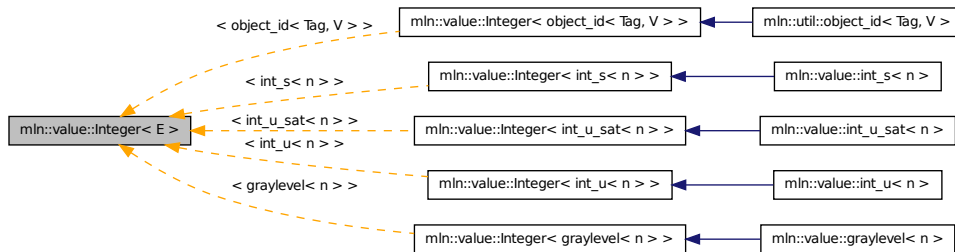
Definition at line 112 of file int_u_sat.hh.

10.400 mln::value::Integer< E > Struct Template Reference

Concept of integer.

```
#include <integer.hh>
```


Inheritance diagram for mln::value::Integer< E >:



10.400.1 Detailed Description

```
template<typename E> struct mln::value::Integer< E >
```

Concept of integer.

Definition at line 58 of file concept/integer.hh.

10.401 mln::value::Integer< void > Struct Template Reference

Category flag type.

```
#include <integer.hh>
```

10.401.1 Detailed Description

```
template<> struct mln::value::Integer< void >
```

Category flag type.

Definition at line 50 of file concept/integer.hh.

10.402 mln::value::label< n > Struct Template Reference

Label value class.

```
#include <label.hh>
```

Inherits mln::value::Symbolic< label< n > >, and mln::value::internal::value_like_< unsigned,internal::encoding_unsigned_< n >::ret,int,label< n > >.

Public Types

- typedef internal::encoding_unsigned_< n >::ret **enc**
Encoding associated type.

Public Member Functions

- [label \(\)](#)
Constructor without argument.
- [label \(unsigned i\)](#)
Constructor from an (unsigned) integer.
- [label \(const literal::zero_t &v\)](#)
Constructor from [literal::zero](#).
- [label< n > next \(\) const](#)
Return the next value.
- [operator unsigned \(\) const](#)
Conversion to an unsigned integer.
- [label< n > & operator++ \(\)](#)
Self increment.
- [label< n > & operator-- \(\)](#)
Self decrement.
- [label< n > & operator= \(unsigned i\)](#)
Assignment from an (unsigned) integer.
- [label< n > & operator= \(const literal::zero_t &v\)](#)
Assignment from [literal::zero](#).
- [label< n > prev \(\) const](#)
Return the previous value.

10.402.1 Detailed Description

template<unsigned n> struct mln::value::label< n >

Label value class. The parameter n is the number of encoding bits.

Definition at line 140 of file label.hh.

10.402.2 Member Typedef Documentation

10.402.2.1 template<unsigned n> typedef internal::encoding_unsigned_<n>::ret mln::value::label< n >::enc

Encoding associated type.

Definition at line 150 of file label.hh.

10.402.3 Constructor & Destructor Documentation

10.402.3.1 `template<unsigned n> mln::value::label< n >::label () [inline]`

Constructor without argument.

Definition at line 271 of file label.hh.

10.402.3.2 `template<unsigned n> mln::value::label< n >::label (unsigned i) [inline]`

Constructor from an (unsigned) integer.

Definition at line 277 of file label.hh.

10.402.3.3 `template<unsigned n> mln::value::label< n >::label (const literal::zero_t & v) [inline]`

Constructor from [literal::zero](#).

Definition at line 284 of file label.hh.

10.402.4 Member Function Documentation

10.402.4.1 `template<unsigned n> label< n > mln::value::label< n >::next () const [inline]`

Return the next value.

Definition at line 338 of file label.hh.

10.402.4.2 `template<unsigned n> mln::value::label< n >::operator unsigned () const [inline]`

Conversion to an unsigned integer.

Definition at line 291 of file label.hh.

10.402.4.3 `template<unsigned n> label< n > & mln::value::label< n >::operator++ () [inline]`

Self increment.

Definition at line 318 of file label.hh.

10.402.4.4 `template<unsigned n> label< n > & mln::value::label< n >::operator-- () [inline]`

Self decrement.

Definition at line 328 of file label.hh.

10.402.4.5 `template<unsigned n> label< n > & mln::value::label< n >::operator= (unsigned i) [inline]`

Assignment from an (unsigned) integer.

Definition at line 299 of file label.hh.

10.402.4.6 `template<unsigned n> label< n > & mln::value::label< n >::operator= (const literal::zero_t & v) [inline]`

Assignment from `literal::zero`.

Definition at line 309 of file label.hh.

10.402.4.7 `template<unsigned n> label< n > mln::value::label< n >::prev () const [inline]`

Return the previous value.

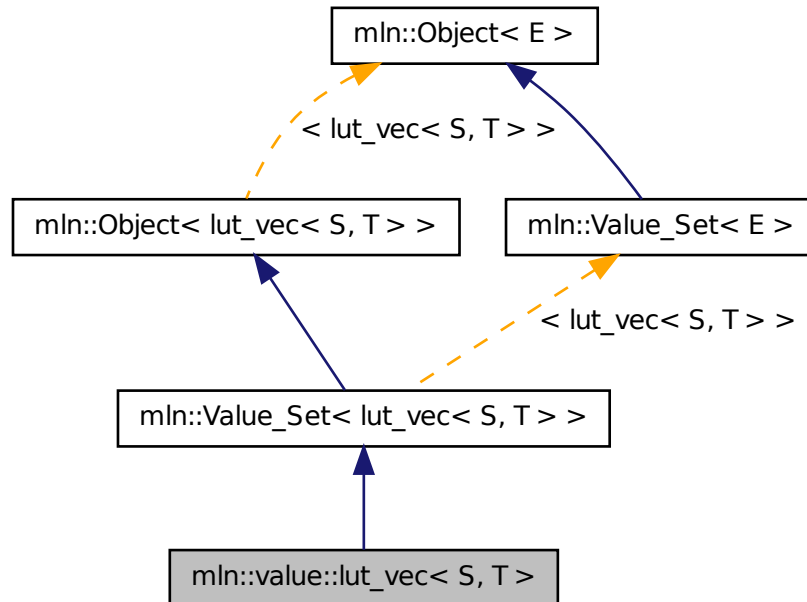
Definition at line 346 of file label.hh.

10.403 `mln::value::lut_vec< S, T >` Struct Template Reference

Class that defines FIXME.

```
#include <lut_vec.hh>
```

Inheritance diagram for mln::value::lut_vec< S, T >:



Public Types

- typedef `bkd_viter_< lut_vec< S, T > >` `bkd_viter`
Backward Value_Iterator associated type.
- typedef `fwd_viter_< lut_vec< S, T > >` `fwd_viter`
Forward Value_Iterator associated type.
- typedef `T` `value`
Value associated type.

Public Member Functions

- bool `has` (const `value` &`v`) const
Test if `v` belongs to this set.
- unsigned `index_of` (const `value` &`v`) const
Give the index of value `v` in this set.
- unsigned `nvalues` () const

Give the number of values.

- `T operator[] (unsigned i) const`
Give the i -th value.

- `template<typename F >`
`lut_vec (const S &vset, const Function_v2v< F > &f)`
Constructors
Constructor from a value set and any Function_v2v.

- `template<typename V >`
`lut_vec (const S &vset, const Function_v2v< fun::i2v::array< V > > &f)`
Constructor from a value set and any fun::i2v::array.

- `template<typename V >`
`lut_vec (const S &vset, const Function_v2v< util::array< V > > &f)`
Constructor from a value set and any util::array.

10.403.1 Detailed Description

`template<typename S, typename T> struct mln::value::lut_vec< S, T >`

Class that defines FIXME.

Warning

This is a multi-set!!! FIXME

Definition at line 71 of file lut_vec.hh.

10.403.2 Member Typedef Documentation

10.403.2.1 `template<typename S , typename T > typedef bkd_viter_< lut_vec<S,T> >`
`mln::value::lut_vec< S, T >::bkd_viter`

Backward [Value_Iterator](#) associated type.

Definition at line 80 of file lut_vec.hh.

10.403.2.2 `template<typename S , typename T > typedef fwd_viter_< lut_vec<S,T> >`
`mln::value::lut_vec< S, T >::fwd_viter`

Forward [Value_Iterator](#) associated type.

Definition at line 77 of file lut_vec.hh.

10.403.2.3 `template<typename S , typename T > typedef T mln::value::lut_vec< S, T >::value`

[Value](#) associated type.

Definition at line 74 of file lut_vec.hh.

10.403.3 Constructor & Destructor Documentation

10.403.3.1 `template<typename S, typename T> template<typename F> mln::value::lut_vec< S, T>::lut_vec(const S & vset, const Function_v2v< F> & f) [inline]`

Constructors

Constructor from a value set and any [Function_v2v](#).

Definition at line 148 of file lut_vec.hh.

10.403.3.2 `template<typename S, typename T> template<typename V> mln::value::lut_vec< S, T>::lut_vec(const S & vset, const Function_v2v< fun::i2v::array< V>> & f) [inline]`

Constructor from a value set and any [fun::i2v::array](#).

Definition at line 161 of file lut_vec.hh.

10.403.3.3 `template<typename S, typename T> template<typename V> mln::value::lut_vec< S, T>::lut_vec(const S & vset, const Function_v2v< util::array< V>> & f) [inline]`

Constructor from a value set and any [util::array](#).

Definition at line 172 of file lut_vec.hh.

References [mln::util::array< T>::size\(\)](#), and [mln::util::array< T>::std_vector\(\)](#).

10.403.4 Member Function Documentation

10.403.4.1 `template<typename S, typename T> bool mln::value::lut_vec< S, T>::has(const value & v) const`

Test if *v* belongs to this set.

10.403.4.2 `template<typename S, typename T> unsigned mln::value::lut_vec< S, T>::index_of(const value & v) const`

Give the index of value *v* in this set.

10.403.4.3 `template<typename S, typename T> unsigned mln::value::lut_vec< S, T>::nvalues() const [inline]`

Give the number of values.

Definition at line 202 of file lut_vec.hh.

Referenced by [mln::value::lut_vec< S, T>::operator\[\]\(\)](#).

10.403.4.4 `template<typename S, typename T> T mln::value::lut_vec< S, T >::operator[] (unsigned i) const [inline]`

Give the `i`-th value.

Definition at line 193 of file `lut_vec.hh`.

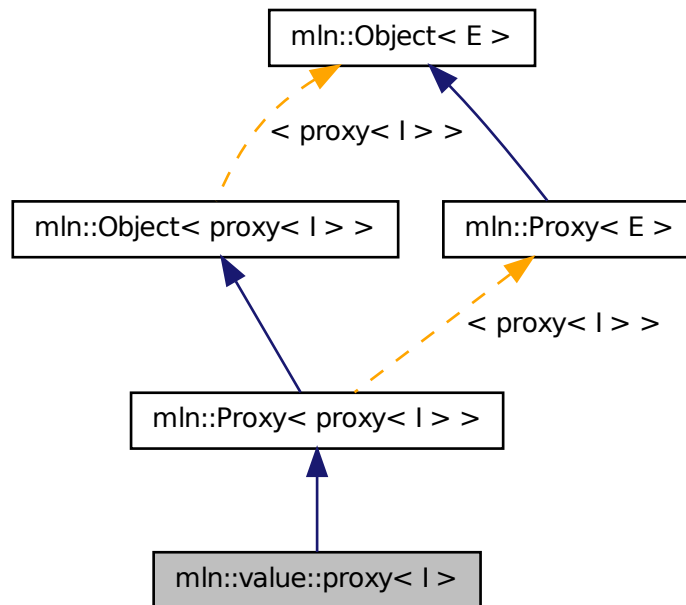
References `mln::value::lut_vec< S, T >::nvalues()`.

10.404 `mln::value::proxy< I >` Class Template Reference

Generic proxy class for an image pixel value.

```
#include <proxy.hh>
```

Inheritance diagram for `mln::value::proxy< I >`:



Public Types

- typedef void `enc`
Encoding associated type.
- typedef `I::value` `equiv`
Equivalent associated type.

Public Member Functions

- `proxy< I > & operator=` (const `proxy< I > &rhs`)
Assignment (write access); replacement for default op.
- `template<typename J > proxy< I > & operator=` (const `proxy< J > &rhs`)
Assignment (write access); with other proxy.
- `proxy ()`
Constructor.
- `proxy (I &ima, const typename I::psite &p)`
Constructor.
- `I::value to_value ()` const
Explicit read access.
- `~proxy ()`
Destructor.

10.404.1 Detailed Description

`template<typename I> class mln::value::proxy< I >`

Generic proxy class for an image pixel value. The parameter `I` is an image type.

Definition at line 85 of file `value/proxy.hh`.

10.404.2 Member Typedef Documentation

10.404.2.1 `template<typename I> typedef void mln::value::proxy< I >::enc`

Encoding associated type.

Definition at line 91 of file `value/proxy.hh`.

10.404.2.2 `template<typename I> typedef I::value mln::value::proxy< I >::equiv`

Equivalent associated type.

Definition at line 94 of file `value/proxy.hh`.

10.404.3 Constructor & Destructor Documentation

10.404.3.1 `template<typename I> mln::value::proxy< I >::proxy () [inline]`

Constructor.

Definition at line 150 of file `value/proxy.hh`.

10.404.3.2 `template<typename I> mln::value::proxy<I>::proxy (I & ima, const typename I::psite & p) [inline]`

Constructor.

Definition at line 157 of file value/proxy.hh.

10.404.3.3 `template<typename I> mln::value::proxy<I>::~~proxy () [inline]`

Destructor.

Definition at line 165 of file value/proxy.hh.

10.404.4 Member Function Documentation

10.404.4.1 `template<typename I> proxy<I> & mln::value::proxy<I>::operator= (const proxy<I> & rhs) [inline]`

Assignment (write access); replacement for default op.

Definition at line 186 of file value/proxy.hh.

References `mln::value::proxy<I>::to_value()`.

10.404.4.2 `template<typename I> template<typename J> proxy<I> & mln::value::proxy<I>::operator= (const proxy<J> & rhs) [inline]`

Assignment (write access); with other proxy.

Definition at line 199 of file value/proxy.hh.

References `mln::value::proxy<I>::to_value()`.

10.404.4.3 `template<typename I> I::value mln::value::proxy<I>::to_value () const [inline]`

Explicit read access.

Definition at line 226 of file value/proxy.hh.

Referenced by `mln::value::proxy<I>::operator=()`.

10.405 mln::value::qt::rgb32 Struct Reference

Color class for red-green-blue where every component is n-bit encoded.

```
#include <rgb32.hh>
```

Inherits `mln::value::Vectorial<rgb32>`, and `mln::value::internal::value_like_<algebra::vec<3, int_u<8>>, algebra::vec<3, int_u<8>>, algebra::vec<3, int>>, rgb32>`.

Public Member Functions

- `rgb32 & operator= (const rgb32 &rhs)`

Assignment.

- `rgb32` (const algebra::vec< 3, int > &rhs)
Constructor from a algebra::vec.
- `rgb32` (int r, int g, int b)
Constructor from component values.
- `rgb32` ()
Constructor without argument.
- `int_u< 8 > red` () const
Acces to red/green/blue component.
- `rgb32` (const mln::literal::zero_t &)
Constructors with literals.

Static Public Attributes

- static const `rgb32 zero`
Zero value.

10.405.1 Detailed Description

Color class for red-green-blue where every component is n-bit encoded.

Definition at line 197 of file rgb32.hh.

10.405.2 Constructor & Destructor Documentation

10.405.2.1 mln::value::qt::rgb32::rgb32 () [inline]

Constructor without argument.

Definition at line 385 of file rgb32.hh.

10.405.2.2 mln::value::qt::rgb32::rgb32 (int r, int g, int b) [inline]

Constructor from component values.

Definition at line 427 of file rgb32.hh.

10.405.2.3 mln::value::qt::rgb32::rgb32 (const algebra::vec< 3, int > & rhs) [inline]

Constructor from a algebra::vec.

Definition at line 391 of file rgb32.hh.

10.405.2.4 `mln::value::qt::rgb32::rgb32 (const mln::literal::zero_t &) [inline]`

Constructors with literals.

Definition at line 442 of file rgb32.hh.

10.405.3 Member Function Documentation**10.405.3.1** `rgb32 & mln::value::qt::rgb32::operator= (const rgb32 & rhs) [inline]`

Assignment.

Definition at line 623 of file rgb32.hh.

10.405.3.2 `int_u<8> mln::value::qt::rgb32::red () const [inline]`

Access to red/green/blue component.

Definition at line 212 of file rgb32.hh.

10.405.4 Member Data Documentation**10.405.4.1** `const rgb32 mln::value::qt::rgb32::zero [static]`

Zero value.

Definition at line 272 of file rgb32.hh.

10.406 `mln::value::rgb< n >` Struct Template Reference

Color class for red-green-blue where every component is n-bit encoded.

```
#include <rgb.hh>
```

Inherits `mln::value::Vectorial< rgb< n > >`, and `mln::value::internal::value_like_< algebra::vec< 3, int_u< n > >, algebra::vec< 3, int_u< n > >, algebra::vec< 3, int >, rgb< n > >`.

Public Member Functions

- `rgb< n > & operator= (const rgb< n > &rhs)`
Assignment.
- `rgb (const algebra::vec< 3, int > &rhs)`
Constructor from a algebra::vec.
- `rgb (int r, int g, int b)`
Constructor from component values.
- `rgb ()`
Constructor without argument.

- `int_u<n> red()` const
Acces to red/green/blue component.
- `rgb(const mln::literal::white_t &)`
Constructors with literals.

Static Public Attributes

- static const `rgb<n> zero`
Zero value.

10.406.1 Detailed Description

`template<unsigned n> struct mln::value::rgb<n>`

Color class for red-green-blue where every component is n-bit encoded.

Definition at line 248 of file `value/rgb.hh`.

10.406.2 Constructor & Destructor Documentation

10.406.2.1 `template<unsigned n> mln::value::rgb<n>::rgb()` [inline]

Constructor without argument.

Definition at line 422 of file `value/rgb.hh`.

10.406.2.2 `template<unsigned n> mln::value::rgb<n>::rgb(int r, int g, int b)` [inline]

Constructor from component values.

Definition at line 458 of file `value/rgb.hh`.

10.406.2.3 `template<unsigned n> mln::value::rgb<n>::rgb(const algebra::vec<3, int> & rhs)` [inline]

Constructor from a `algebra::vec`.

Definition at line 428 of file `value/rgb.hh`.

10.406.2.4 `template<unsigned n> mln::value::rgb<n>::rgb(const mln::literal::white_t &)` [inline]

Constructors with literals.

Definition at line 473 of file `value/rgb.hh`.

10.406.3 Member Function Documentation

10.406.3.1 `template<unsigned n> rgb< n > & mln::value::rgb< n >::operator= (const rgb< n > & rhs) [inline]`

Assignment.

Definition at line 645 of file value/rgb.hh.

10.406.3.2 `template<unsigned n> int_u<n> mln::value::rgb< n >::red () const [inline]`

Access to red/green/blue component.

Definition at line 264 of file value/rgb.hh.

Referenced by `mln::fun::v2v::rgb8_to_rgbn< n >::operator()`.

10.406.4 Member Data Documentation

10.406.4.1 `template<unsigned n> const rgb< n > mln::value::rgb< n >::zero [static]`

Zero value.

Definition at line 322 of file value/rgb.hh.

10.407 mln::value::set< T > Struct Template Reference

Class that defines the set of values of type T.

```
#include <set.hh>
```

Inherits `set_selector_< T, set< T >, mln::metal::equal< mln::trait::value_< T >::quant, mln::trait::value::quant::low >::value >`.

Static Public Member Functions

- static const `set< T > & the ()`

Return a singleton.

10.407.1 Detailed Description

`template<typename T> struct mln::value::set< T >`

Class that defines the set of values of type T. This is the exhaustive set of values obtainable from type T.

Definition at line 65 of file value/set.hh.

10.407.2 Member Function Documentation

10.407.2.1 `template<typename T> const set< T> & mln::value::set< T>::the () [inline, static]`

Return a singleton.

Definition at line 80 of file value/set.hh.

10.408 mln::value::sign Class Reference

The sign class represents the value type composed by the set (-1, 0, 1) sign value type is a subset of the int value type.

```
#include <sign.hh>
```

Inherits Integer< sign >.

Public Types

- typedef int `enc`
FIXME Are these typedefs correct?
- typedef int `equiv`
Define the equivalent type.

Public Member Functions

- `operator int () const`
Conversion to an integer.
- `sign & operator= (int i)`
Assignment from an integer.
- `sign ()`
Constructor without argument.
- `sign (int i)`
Constructor from an integer.
- `sign (const mln::literal::zero_t &)`
Constructors/assignments with literals.

Static Public Attributes

- static const `sign one = 1`
Unit value.

- static const `sign zero = 0`

Zero value.

10.408.1 Detailed Description

The `sign` class represents the value type composed by the set $(-1, 0, 1)$ `sign` value type is a subset of the `int` value type.

Definition at line 49 of file `value/sign.hh`.

10.408.2 Member Typedef Documentation

10.408.2.1 `typedef int mln::value::sign::enc`

FIXME Are these typedefs correct?

Define the encoding type

Definition at line 55 of file `value/sign.hh`.

10.408.2.2 `typedef int mln::value::sign::equiv`

Define the equivalent type.

Definition at line 58 of file `value/sign.hh`.

10.408.3 Constructor & Destructor Documentation

10.408.3.1 `mln::value::sign::sign () [inline]`

Constructor without argument.

Definition at line 119 of file `value/sign.hh`.

10.408.3.2 `mln::value::sign::sign (int i) [inline]`

Constructor from an integer.

Definition at line 137 of file `value/sign.hh`.

10.408.3.3 `mln::value::sign::sign (const mln::literal::zero_t &) [inline]`

Constructors/assignments with literals.

Definition at line 155 of file `value/sign.hh`.

10.408.4 Member Function Documentation

10.408.4.1 mln::value::sign::operator int () const [inline]

Conversion to an integer.

Definition at line 124 of file value/sign.hh.

10.408.4.2 sign & mln::value::sign::operator=(int i) [inline]

Assignment from an integer.

Definition at line 146 of file value/sign.hh.

10.408.5 Member Data Documentation

10.408.5.1 const sign mln::value::sign::one = 1 [static]

Unit value.

Definition at line 88 of file value/sign.hh.

10.408.5.2 const sign mln::value::sign::zero = 0 [static]

Zero value.

Definition at line 85 of file value/sign.hh.

10.409 mln::value::stack_image< n, I > Struct Template Reference

Stack image class.

```
#include <stack.hh>
```

Inherits image_value_morpher< I, algebra::vec< n, I::value >, stack_image< n, I > >.

Public Types

- typedef I::domain_t [domain_t](#)
Site_Set associated type.
- typedef internal::helper_stack_image_lvalue_< n, I >::ret [lvalue](#)
Return type of read-write access.
- typedef I::psite [psite](#)
Point_Site associated type.
- typedef [value](#) [rvalue](#)
Return type of read-only access.
- typedef [stack_image](#)< n, tag::image_< I > > [skeleton](#)

Skeleton.

- typedef algebra::vec< n, typename I::value > [value](#)
Value associated type.

Public Member Functions

- bool [is_valid](#) () const
Test if this image has been initialized.
- [lvalue operator](#)() (const [psite](#) &)
Read-write access of pixel value at point site p.
- [rvalue operator](#)() (const [psite](#) &p) const
Read-only access of pixel value at point site p.
- [stack_image](#) (const algebra::vec< n, I > &imas)
Constructors.

10.409.1 Detailed Description

template<unsigned n, typename I> struct mln::value::stack_image< n, I >

Stack image class. [mln::value::stack_image](#) stores a vector of n images of the same domain.

The parameter n is the number of images, I is the type of a stack element. Acces a value will compute a vector which contains n coordinates : [stack[0](p), stack[1](p), ... , stack[n](p)]

Definition at line 145 of file stack.hh.

10.409.2 Member Typedef Documentation

10.409.2.1 template<unsigned n, typename I> typedef I ::domain_t mln::value::stack_image< n, I >::domain_t

[Site_Set](#) associated type.

Definition at line 154 of file stack.hh.

10.409.2.2 template<unsigned n, typename I> typedef internal::helper_-stack_image_lvalue_<n,I>::ret mln::value::stack_image< n, I >::lvalue

Return type of read-write access.

Definition at line 166 of file stack.hh.

10.409.2.3 `template<unsigned n, typename I> typedef I ::psite mln::value::stack_image< n, I >::psite`

[Point_Site](#) associated type.

Definition at line 151 of file stack.hh.

10.409.2.4 `template<unsigned n, typename I> typedef value mln::value::stack_image< n, I >::rvalue`

Return type of read-only access.

The rvalue type is not a const reference, since the value type is built on the fly, and return by value (copy).

Definition at line 163 of file stack.hh.

10.409.2.5 `template<unsigned n, typename I> typedef stack_image< n, tag::image_<I> > mln::value::stack_image< n, I >::skeleton`

Skeleton.

Definition at line 170 of file stack.hh.

10.409.2.6 `template<unsigned n, typename I> typedef algebra::vec<n, typename I ::value> mln::value::stack_image< n, I >::value`

[Value](#) associated type.

Definition at line 157 of file stack.hh.

10.409.3 Constructor & Destructor Documentation

10.409.3.1 `template<unsigned n, typename I> mln::value::stack_image< n, I >::stack_image (const algebra::vec< n, I > & imas) [inline]`

Constructors.

Definition at line 236 of file stack.hh.

10.409.4 Member Function Documentation

10.409.4.1 `template<unsigned n, typename I > bool mln::value::stack_image< n, I >::is_valid () const [inline]`

Test if this image has been initialized.

Definition at line 255 of file stack.hh.

10.409.4.2 `template<unsigned n, typename I > stack_image< n, I >::lvalue mln::value::stack_image< n, I >::operator() (const psite & p) [inline]`

Read-write access of pixel value at point site *p*.

Definition at line 296 of file stack.hh.

10.409.4.3 `template<unsigned n, typename I > stack_image< n, I >::rvalue
mln::value::stack_image< n, I >::operator() (const psite & p) const` `[inline]`

Read-only access of pixel value at point site p.

Definition at line 277 of file stack.hh.

10.410 mln::value::super_value< sign > Struct Template Reference

Specializations:

```
#include <super_value.hh>
```

10.410.1 Detailed Description

```
template<> struct mln::value::super_value< sign >
```

Specializations: Sign type is a subset of the short value type.

Definition at line 56 of file super_value.hh.

10.411 mln::value::value_array< T, V > Struct Template Reference

Generic array class over indexed by a value set with type T.

```
#include <value_array.hh>
```

Public Member Functions

- `const V & operator() (const T &v) const`
}
- `const V & operator[] (unsigned i) const`
}
- `value_array ()`
Constructors.
- `const mln::value::set< T > & vset () const`
}

10.411.1 Detailed Description

```
template<typename T, typename V> struct mln::value::value_array< T, V >
```

Generic array class over indexed by a value set with type T.

Definition at line 45 of file value_array.hh.

10.411.2 Constructor & Destructor Documentation

10.411.2.1 `template<typename T , typename V > mln::value::value_array< T, V >::value_array () [inline]`

Constructors.

```
{
```

Definition at line 89 of file value_array.hh.

10.411.3 Member Function Documentation

10.411.3.1 `template<typename T , typename V > const V & mln::value::value_array< T, V >::operator()(const T & v) const [inline]`

```
}
```

Access elements through a value of T. {

Definition at line 128 of file value_array.hh.

10.411.3.2 `template<typename T , typename V > const V & mln::value::value_array< T, V >::operator[](unsigned i) const [inline]`

```
}
```

Access elements through array indexes. {

Definition at line 152 of file value_array.hh.

10.411.3.3 `template<typename T , typename V > const mln::value::set< T > & mln::value::value_array< T, V >::vset () const [inline]`

```
}
```

Reference to the set of T.

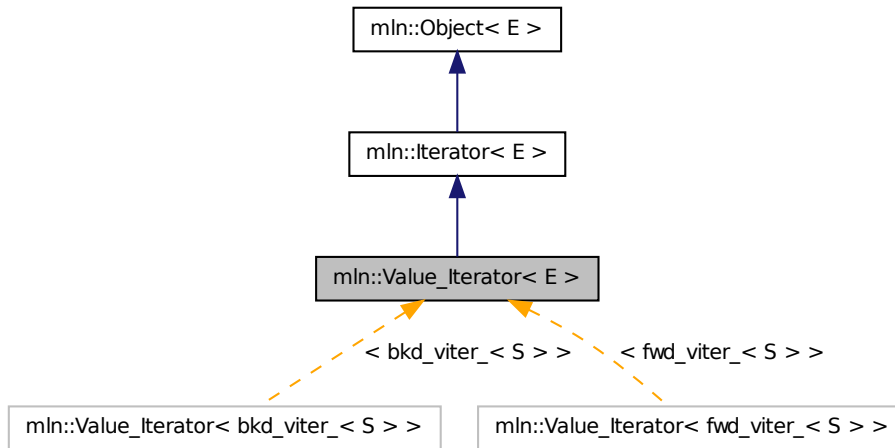
Definition at line 144 of file value_array.hh.

10.412 mln::Value_Iterator< E > Struct Template Reference

Base class for implementation of classes of iterator on values.

```
#include <value_iterator.hh>
```

Inheritance diagram for `mln::Value_Iterator< E >`:



Public Member Functions

- void [next](#) ()
Go to the next element.

Related Functions

(Note that these are not member functions.)

- `template<typename E >`
`std::ostream & operator<< (std::ostream &ostr, const Value_Iterator< E > &v)`
Print an iterator `v` on value set into the output stream `ostr`.

10.412.1 Detailed Description

`template<typename E> struct mln::Value_Iterator< E >`

Base class for implementation of classes of iterator on values. An iterator on values is an iterator that browse over a set of values.

See also

[mln::doc::Value_Iterator](#) for a complete documentation of this class contents.

Definition at line 50 of file `value_iterator.hh`.

10.412.2 Member Function Documentation

10.412.2.1 `template<typename E > void mln::Iterator< E >::next ()` [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

Definition at line 102 of file iterator.hh.

10.412.3 Friends And Related Function Documentation

10.412.3.1 `template<typename E > std::ostream & operator<< (std::ostream & ostr, const Value_Iterator< E > & v)` [related]

Print an iterator *v* on value set into the output stream *ostr*.

Parameters

[in, out] *ostr* An output stream.

[in] *v* An iterator on value set.

Precondition

v is a valid.

Returns

The modified output stream *ostr*.

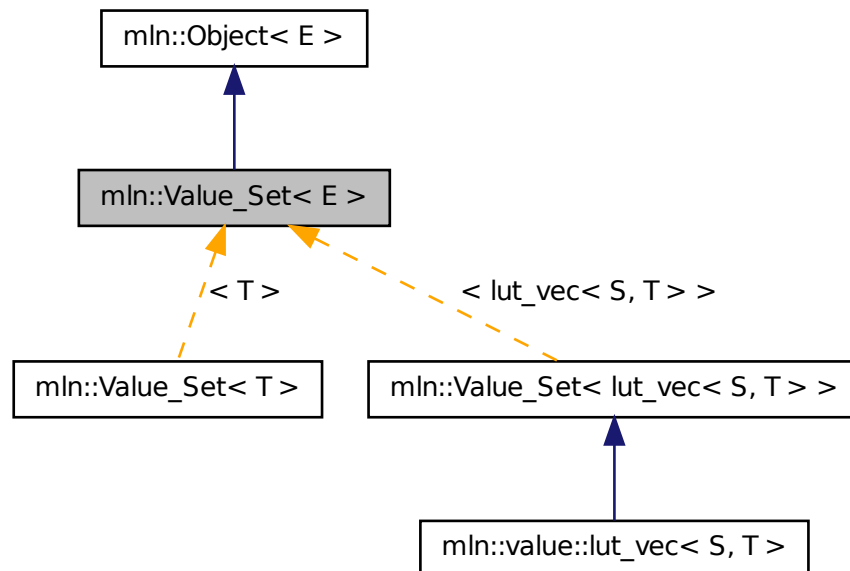
Definition at line 92 of file value_iterator.hh.

10.413 mln::Value_Set< E > Struct Template Reference

Base class for implementation classes of sets of values.

```
#include <value_set.hh>
```

Inheritance diagram for `mln::Value_Set< E >`:



10.413.1 Detailed Description

```
template<typename E> struct mln::Value_Set< E >
```

Base class for implementation classes of sets of values.

See also

[mln::doc::Value_Set](#) for a complete documentation of this class contents.

Definition at line 57 of file `value_set.hh`.

10.414 mln::Vertex< E > Struct Template Reference

[Vertex](#) category flag type.

```
#include <vertex.hh>
```

10.414.1 Detailed Description

```
template<typename E> struct mln::Vertex< E >
```

[Vertex](#) category flag type.

Definition at line 53 of file vertex.hh.

10.415 mln::vertex_image< P, V, G > Class Template Reference

[Image](#) based on graph vertices.

```
#include <vertex_image.hh>
```

Inherits [image_base< fun::i2v::array< V >, p_vertices< G, internal::vsite_selector< P, G >::site_function_t >, vertex_image< P, V, G > >](#).

Public Types

- typedef G [graph_t](#)
The type of the underlying graph.
- typedef [vertex_nbh_t](#) nbh_t
Neighborhood type.
- typedef internal::vsite_selector< P, G >::[site_function_t](#) site_function_t
Function mapping graph elements to sites.
- typedef [vertex_image](#)< tag::psite_< P >, tag::value_< V >, tag::graph_< G > > [skeleton](#)
Skeleton type.
- typedef [graph_elt_neighborhood](#)< G, S > [vertex_nbh_t](#)
Vertex Neighborhood type.
- typedef [graph_elt_window](#)< G, S > [vertex_win_t](#)
Vertex Window type.
- typedef [vertex_win_t](#) win_t
Window type.

Public Member Functions

- [vertex_image](#) ()
Constructors.
- rvalue [operator](#)() (unsigned v_id) const
Value accessors/operators overloads.

10.415.1 Detailed Description

`template<typename P, typename V, typename G = util::graph> class mln::vertex_image< P, V, G >`

[Image](#) based on graph vertices.

Definition at line 126 of file core/image/vertex_image.hh.

10.415.2 Member Typedef Documentation

10.415.2.1 `template<typename P, typename V, typename G = util::graph> typedef G mln::vertex_image< P, V, G >::graph_t`

The type of the underlying graph.

Definition at line 141 of file core/image/vertex_image.hh.

10.415.2.2 `template<typename P, typename V, typename G = util::graph> typedef vertex_nbh_t mln::vertex_image< P, V, G >::nbh_t`

[Neighborhood](#) type.

Definition at line 165 of file core/image/vertex_image.hh.

10.415.2.3 `template<typename P, typename V, typename G = util::graph> typedef internal::vfsite_selector<P,G>::site_function_t mln::vertex_image< P, V, G >::site_function_t`

[Function](#) mapping graph elements to sites.

Definition at line 145 of file core/image/vertex_image.hh.

10.415.2.4 `template<typename P, typename V, typename G = util::graph> typedef vertex_image< tag::psite_<P>, tag::value_<V>, tag::graph_<G> > mln::vertex_image< P, V, G >::skeleton`

[Skeleton](#) type.

Definition at line 152 of file core/image/vertex_image.hh.

10.415.2.5 `template<typename P, typename V, typename G = util::graph> typedef graph_elt_neighborhood<G,S> mln::vertex_image< P, V, G >::vertex_nbh_t`

[Vertex Neighborhood](#) type.

Definition at line 160 of file core/image/vertex_image.hh.

10.415.2.6 `template<typename P, typename V, typename G = util::graph> typedef graph_elt_window<G,S> mln::vertex_image< P, V, G >::vertex_win_t`

[Vertex Window](#) type.

Definition at line 157 of file core/image/vertex_image.hh.

10.415.2.7 `template<typename P, typename V, typename G = util::graph> typedef vertex_win_t mln::vertex_image< P, V, G >::win_t`

[Window](#) type.

Definition at line 163 of file core/image/vertex_image.hh.

10.415.3 Constructor & Destructor Documentation

10.415.3.1 `template<typename P, typename V, typename G > mln::vertex_image< P, V, G >::vertex_image() [inline]`

Constructors.

Definition at line 247 of file core/image/vertex_image.hh.

10.415.4 Member Function Documentation

10.415.4.1 `template<typename P, typename V, typename G > vertex_image< P, V, G >::rvalue mln::vertex_image< P, V, G >::operator() (unsigned v_id) const`

[Value](#) accessors/operators overloads.

Definition at line 292 of file core/image/vertex_image.hh.

10.416 mln::violent_cast_image< T, I > Struct Template Reference

Violently cast image values to a given type.

```
#include <violent_cast_image.hh>
```

Inherits `image_value_morpher< I, T, violent_cast_image< T, I > >`.

Public Types

- typedef T [lvalue](#)
Return type of read-write access.
- typedef T [rvalue](#)
Return type of read-only access.
- typedef [violent_cast_image](#)< tag::value_< T >, tag::image_< I > > [skeleton](#)
Skeleton.
- typedef T [value](#)
Value associated type.

Public Member Functions

- `T operator()` (const typename I::psite &p) const
Read-only access of pixel value at point site p.
- `T operator()` (const typename I::psite &p)
Mutable access is only OK for reading (not writing).
- `violent_cast_image` (const `Image< I >` &ima)
Constructor.

10.416.1 Detailed Description

`template<typename T, typename I> struct mln::violent_cast_image< T, I >`

Violently cast image values to a given type.

Definition at line 112 of file `violent_cast_image.hh`.

10.416.2 Member Typedef Documentation

10.416.2.1 `template<typename T, typename I> typedef T mln::violent_cast_image< T, I >::lvalue`

Return type of read-write access.

Definition at line 122 of file `violent_cast_image.hh`.

10.416.2.2 `template<typename T, typename I> typedef T mln::violent_cast_image< T, I >::rvalue`

Return type of read-only access.

Definition at line 119 of file `violent_cast_image.hh`.

10.416.2.3 `template<typename T, typename I> typedef violent_cast_image< tag::value_<T>, tag::image_<I> > mln::violent_cast_image< T, I >::skeleton`

Skeleton.

Definition at line 125 of file `violent_cast_image.hh`.

10.416.2.4 `template<typename T, typename I> typedef T mln::violent_cast_image< T, I >::value`

[Value](#) associated type.

Definition at line 116 of file `violent_cast_image.hh`.

10.416.3 Constructor & Destructor Documentation

10.416.3.1 `template<typename T , typename I > mln::violent_cast_image< T, I >::violent_cast_image(const Image< I > & ima) [inline]`

Constructor.

Definition at line 172 of file violent_cast_image.hh.

10.416.4 Member Function Documentation

10.416.4.1 `template<typename T , typename I > T mln::violent_cast_image< T, I >::operator()(const typename I::psite & p) const [inline]`

Read-only access of pixel value at point site p.

Definition at line 191 of file violent_cast_image.hh.

10.416.4.2 `template<typename T , typename I > T mln::violent_cast_image< T, I >::operator()(const typename I::psite & p) [inline]`

Mutable access is only OK for reading (not writing).

Definition at line 200 of file violent_cast_image.hh.

10.417 mln::w_window< D, W > Struct Template Reference

Generic [w_window](#) class.

```
#include <w_window.hh>
```

Inherits [weighted_window_base< mln::window< D >, w_window< D, W > >](#).

Public Types

- typedef with_w_< [dpsites_bkd_piter< w_window< D, W > >](#), W > [bkd_qiter](#)
Site_Iterator type to browse (backward) the points of a generic [w_window](#).
- typedef D [dpsite](#)
Dpsite associated type.
- typedef with_w_< [dpsites_fwd_piter< w_window< D, W > >](#), W > [fwd_qiter](#)
Site_Iterator type to browse (forward) the points of a generic [w_window](#).
- typedef W [weight](#)
Weight associated type.

Public Member Functions

- void [clear](#) ()

Clear this window.

- `w_window< D, W > & insert (const W &w, const D &d)`
Insert a couple of weight w and delta-point d .
- `bool is_symmetric () const`
Test if the window is symmetric.
- `const std::vector< D > & std_vector () const`
Give access to the vector of delta-points.
- `void sym ()`
Apply a central symmetry to the window.
- `W w (unsigned i) const`
Give the i -th weight.
- `w_window ()`
Constructor without argument.
- `const std::vector< W > & weights () const`
Give access to the vector of weights.
- `const mln::window< D > & win () const`
Give the corresponding window.

Related Functions

(Note that these are not member functions.)

- `template<typename D, typename W > std::ostream & operator<< (std::ostream &ostr, const w_window< D, W > &w_win)`
Print a weighted window w_win into an output stream $ostr$.
- `template<typename D, typename Wl, typename Wr > bool operator== (const w_window< D, Wl > &lhs, const w_window< D, Wr > &rhs)`
Equality test between two weighted windows lhs and rhs .

10.417.1 Detailed Description

`template<typename D, typename W> struct mln::w_window< D, W >`

Generic `w_window` class. This type of `w_window` is just like a set of delta-points. The parameter `D` is the type of delta-points; the parameter `W` is the type of weights.

Definition at line 100 of file `core/w_window.hh`.

10.417.2 Member Typedef Documentation

10.417.2.1 `template<typename D, typename W> typedef with_w_< dpsites_bkd_piter< w_window<D, W> >, W > mln::w_window< D, W >::bkd_qiter`

[Site_Iterator](#) type to browse (backward) the points of a generic [w_window](#).

Definition at line 114 of file core/w_window.hh.

10.417.2.2 `template<typename D, typename W> typedef D mln::w_window< D, W >::dpsite`

Dpsite associated type.

Definition at line 104 of file core/w_window.hh.

10.417.2.3 `template<typename D, typename W> typedef with_w_< dpsites_fwd_piter< w_window<D, W> >, W > mln::w_window< D, W >::fwd_qiter`

[Site_Iterator](#) type to browse (forward) the points of a generic [w_window](#).

Definition at line 111 of file core/w_window.hh.

10.417.2.4 `template<typename D, typename W> typedef W mln::w_window< D, W >::weight`

Weight associated type.

Definition at line 107 of file core/w_window.hh.

10.417.3 Constructor & Destructor Documentation

10.417.3.1 `template<typename D , typename W > mln::w_window< D, W >::w_window ()
[inline]`

Constructor without argument.

Definition at line 213 of file core/w_window.hh.

10.417.4 Member Function Documentation

10.417.4.1 `template<typename D , typename W > void mln::w_window< D, W >::clear ()
[inline]`

Clear this window.

Definition at line 308 of file core/w_window.hh.

10.417.4.2 `template<typename D , typename W > w_window< D, W > & mln::w_window< D,
W >::insert (const W & w, const D & d) [inline]`

Insert a couple of weight *w* and delta-point *d*.

Definition at line 254 of file core/w_window.hh.

Referenced by `mln::w_window< D, W >::sym()`, `mln::make::w_window()`, `mln::make::w_window1d()`, `mln::make::w_window3d()`, and `mln::make::w_window_directional()`.

10.417.4.3 `template<typename D , typename W > bool mln::w_window< D, W >::is_symmetric () const [inline]`

Test if the window is symmetric.

Definition at line 284 of file `core/w_window.hh`.

References `mln::w_window< D, W >::sym()`.

10.417.4.4 `template<typename D , typename W > const std::vector< D > & mln::w_window< D, W >::std_vector () const [inline]`

Give access to the vector of delta-points.

Definition at line 228 of file `core/w_window.hh`.

10.417.4.5 `template<typename D , typename W > void mln::w_window< D, W >::sym () [inline]`

Apply a central symmetry to the window.

Definition at line 296 of file `core/w_window.hh`.

References `mln::w_window< D, W >::insert()`.

Referenced by `mln::w_window< D, W >::is_symmetric()`.

10.417.4.6 `template<typename D , typename W > W mln::w_window< D, W >::w (unsigned i) const [inline]`

Give the *i*-th weight.

Definition at line 244 of file `core/w_window.hh`.

10.417.4.7 `template<typename D , typename W > const std::vector< W > & mln::w_window< D, W >::weights () const [inline]`

Give access to the vector of weights.

Definition at line 236 of file `core/w_window.hh`.

Referenced by `mln::w_window< D, W >::operator==()`.

10.417.4.8 `template<typename D , typename W > const mln::window< D > & mln::w_window< D, W >::win () const [inline]`

Give the corresponding window.

Definition at line 220 of file `core/w_window.hh`.

Referenced by `mln::w_window< D, W >::operator==()`.

10.417.5 Friends And Related Function Documentation

10.417.5.1 `template<typename D , typename W > std::ostream & operator<< (std::ostream & ostr, const w_window< D, W > & w_win) [related]`

Print a weighted window `w_win` into an output stream `ostr`.

Definition at line 420 of file `core/w_window.hh`.

10.417.5.2 `template<typename D , typename Wl , typename Wr > bool operator== (const w_window< D, Wl > & lhs, const w_window< D, Wr > & rhs) [related]`

Equality test between two weighted windows `lhs` and `rhs`.

Definition at line 430 of file `core/w_window.hh`.

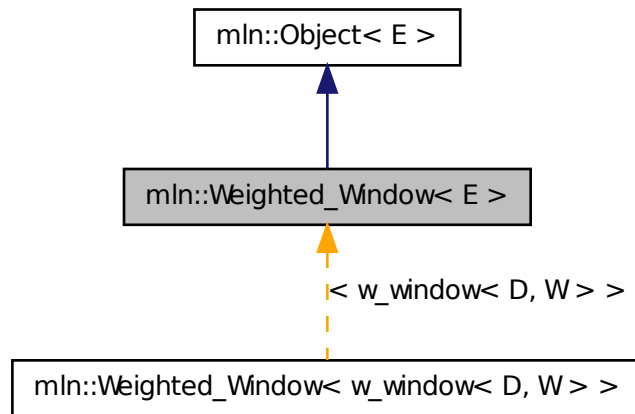
References `mln::w_window< D, W >::weights()`, and `mln::w_window< D, W >::win()`.

10.418 mln::Weighted_Window< E > Struct Template Reference

Base class for implementation classes that are `weighted_windows`.

```
#include <weighted_window.hh>
```

Inheritance diagram for `mln::Weighted_Window< E >`:



Related Functions

(Note that these are not member functions.)

- `template<typename W > W operator- (const Weighted_Window< W > &rhs)`

Compute the symmetrical weighted window of *rhs*.

10.418.1 Detailed Description

`template<typename E> struct mln::Weighted_Window< E >`

Base class for implementation classes that are `weighted_windows`.

See also

[mln::doc::Weighted_Window](#) for a complete documentation of this class contents.

Definition at line 68 of file `weighted_window.hh`.

10.418.2 Friends And Related Function Documentation

10.418.2.1 `template<typename W > W operator- (const Weighted_Window< W > & rhs)`
[related]

Compute the symmetrical weighted window of *rhs*.

10.419 mln::win::backdiag2d Struct Reference

Diagonal line window defined on the 2D square grid.

```
#include <backdiag2d.hh>
```

Inherits `classical_window_base< dpoint2d, backdiag2d >`.

Public Member Functions

- `backdiag2d` (unsigned length)
Constructor.
- unsigned `length` () const
Give the diagonal length, that is, its width.

10.419.1 Detailed Description

Diagonal line window defined on the 2D square grid. An `backdiag2d` is centered and symmetric. its width (length) is odd.

For instance:

```
*  o
*  o
*  x
*  o
*  o
*
```

is defined with length = 5.

Definition at line 63 of file backdiag2d.hh.

10.419.2 Constructor & Destructor Documentation

10.419.2.1 mln::win::backdiag2d::backdiag2d (unsigned length) [inline]

Constructor.

Parameters

[in] *length* Length, thus width, of the diagonal line.

Precondition

length is odd.

Definition at line 93 of file backdiag2d.hh.

10.419.3 Member Function Documentation

10.419.3.1 unsigned mln::win::backdiag2d::length () const [inline]

Give the diagonal length, that is, its width.

Definition at line 105 of file backdiag2d.hh.

10.420 mln::win::ball< G, C > Struct Template Reference

Generic ball window defined on a given grid.

```
#include <ball.hh>
```

Inherits classical_window_base< dpoint< G, C >, ball< G, C > >.

Public Member Functions

- [ball](#) (unsigned diameter)
Constructor.
- unsigned [diameter](#) () const
Give the ball diameter.

10.420.1 Detailed Description

```
template<typename G, typename C> struct mln::win::ball< G, C >
```

Generic ball window defined on a given grid. A ball is centered and symmetric; so its diameter is odd.

G is the given grid on which the ball is defined and C is the type of coordinates.

Definition at line 71 of file ball.hh.

10.420.2 Constructor & Destructor Documentation

10.420.2.1 `template<typename G , typename C > mln::win::ball< G, C >::ball (unsigned diameter) [inline]`

Constructor.

Parameters

[in] *diameter* Diameter of the ball.

Precondition

`diameter` is odd.

Definition at line 99 of file `ball.hh`.

References `mln::literal::origin`.

10.420.3 Member Function Documentation

10.420.3.1 `template<typename G , typename C > unsigned mln::win::ball< G, C >::diameter () const [inline]`

Give the ball diameter.

Definition at line 122 of file `ball.hh`.

10.421 mln::win::cube3d Struct Reference

Cube window defined on the 3D grid.

```
#include <cube3d.hh>
```

Inherits `classical_window_base< dpoint3d, cube3d >`.

Public Member Functions

- `cube3d` (unsigned length)
Constructor.
- unsigned `length` () const
Give the cube length, that is, its height.

10.421.1 Detailed Description

Cube window defined on the 3D grid. An `cube3d` is centered and symmetric; so its height (length) is odd.

For instance:

```
*   o o o
*   o o o
```

```

*  o  o  o
*   o  o  o
*  o  x  o
*  o  o  o

*   o  o  o
*  o  o  o
*  o  o  o
*
```

is defined with `length = 3`.

Definition at line 69 of file `cube3d.hh`.

10.421.2 Constructor & Destructor Documentation

10.421.2.1 mln::win::cube3d::cube3d (unsigned *length*) [inline]

Constructor.

Parameters

[in] *length* Length, thus height, of the `cube3d`.

Precondition

`length` is odd.

Definition at line 99 of file `cube3d.hh`.

10.421.3 Member Function Documentation

10.421.3.1 unsigned mln::win::cube3d::length () const [inline]

Give the cube length, that is, its height.

Definition at line 113 of file `cube3d.hh`.

10.422 mln::win::cuboid3d Struct Reference

Cuboid defined on the 3-D square grid.

```
#include <cube3d.hh>
```

Inherits `classical_window_base< dpoint3d, cuboid3d >`.

Public Member Functions

- `cube3d` (unsigned depth, unsigned height, unsigned width)
Constructor.
- unsigned `volume` () const
Return the volume of the cuboid.

- unsigned `depth ()` const
Accessors.
- unsigned `height ()` const
Return the height of the cuboid.
- unsigned `width ()` const
Return the width of the cuboid.

10.422.1 Detailed Description

Cuboid defined on the 3-D square grid. A `cuboid3d` is a 3-D window with cuboid (also known as rectangular prism or rectangular parallelepiped) shape. It is centered and symmetric.

For instance:

```

      o o o o o o o
      o o o o o o o
    o o o o o o o
    o o o o o o o
  o o o o o o o

      o o o o o o o
      o o o o o o o
    o o o x o o o
    o o o o o o o
  o o o o o o o

      o o o o o o o
      o o o o o o o
    o o o o o o o
    o o o o o o o
  o o o o o o o

```

is defined with `depth = 3`, `height = 5` and `width = 7`.

Reference: <http://en.wikipedia.org/wiki/Cuboid>

Definition at line 80 of file `cuboid3d.hh`.

10.422.2 Constructor & Destructor Documentation

10.422.2.1 `mln::win::cuboid3d::cuboid3d (unsigned depth, unsigned height, unsigned width)`
[inline]

Constructor.

Parameters

- [in] *depth* The depth of the `cuboid3d`.
- [in] *height* The height of the `cuboid3d`.
- [in] *width* The width of the `cuboid3d`.

Precondition

Argument *depth*, *height* and *width* must be odd.

Definition at line 125 of file cuboid3d.hh.

10.422.3 Member Function Documentation

10.422.3.1 unsigned mln::win::cuboid3d::depth () const [inline]

Accessors.

Return the depth of the cuboid.

Definition at line 146 of file cuboid3d.hh.

10.422.3.2 unsigned mln::win::cuboid3d::height () const [inline]

Return the height of the cuboid.

Definition at line 153 of file cuboid3d.hh.

10.422.3.3 unsigned mln::win::cuboid3d::volume () const [inline]

Return the volume of the cuboid.

Definition at line 167 of file cuboid3d.hh.

10.422.3.4 unsigned mln::win::cuboid3d::width () const [inline]

Return the width of the cuboid.

Definition at line 160 of file cuboid3d.hh.

10.423 mln::win::diag2d Struct Reference

Diagonal line window defined on the 2D square grid.

```
#include <diag2d.hh>
```

Inherits classical_window_base< dpoint2d, diag2d >.

Public Member Functions

- [diag2d](#) (unsigned length)
Constructor.
- unsigned [length](#) () const
Give the diagonal length, that is, its width.

10.423.1 Detailed Description

Diagonal line window defined on the 2D square grid. An [diag2d](#) is centered and symmetric. its width (length) is odd.

For instance:

```

*           o
*         o
*       x
*     o
*   o
* o
*

```

is defined with `length = 5`.

Definition at line 63 of file `diag2d.hh`.

10.423.2 Constructor & Destructor Documentation

10.423.2.1 `mln::win::diag2d::diag2d (unsigned length) [inline]`

Constructor.

Parameters

[in] *length* Length, thus width, of the diagonal line.

Precondition

`length` is odd.

Definition at line 93 of file `diag2d.hh`.

10.423.3 Member Function Documentation

10.423.3.1 `unsigned mln::win::diag2d::length () const [inline]`

Give the diagonal length, that is, its width.

Definition at line 106 of file `diag2d.hh`.

10.424 `mln::win::line< M, i, C >` Struct Template Reference

Generic line window defined on a given grid in the given dimension.

```
#include <line.hh>
```

Inherits `classical_window_base< dpoint< M, C >, line< M, i, C > >`.

Public Types

- enum

Direction.

Public Member Functions

- unsigned `length` () const
Give the line length.
- `line` (unsigned length)
Constructor.
- unsigned `size` () const
Give the line size, that is, its length.

10.424.1 Detailed Description

`template<typename M, unsigned i, typename C> struct mln::win::line< M, i, C >`

Generic line window defined on a given grid in the given dimension. An line is centered and symmetric; so its length is odd.

M is the given grid on which the line is defined, i is the given dimension of the line end C is the type of the coordinates.

See also

`mln::win::hline2d` for an exemple of his use.

Definition at line 73 of file win/line.hh.

10.424.2 Member Enumeration Documentation

10.424.2.1 `template<typename M , unsigned i, typename C > anonymous enum`

Direction.

Definition at line 76 of file win/line.hh.

10.424.3 Constructor & Destructor Documentation

10.424.3.1 `template<typename M , unsigned i, typename C > mln::win::line< M, i, C >::line (unsigned length) [inline]`

Constructor.

Parameters

[in] *length* Length of the line.

Precondition

`length` is odd.

Definition at line 106 of file win/line.hh.

References `mln::dpoint< G, C >::set_all()`.

10.424.4 Member Function Documentation

10.424.4.1 `template<typename M, unsigned i, typename C > unsigned mln::win::line< M, i, C >::length() const [inline]`

Give the line length.

Definition at line 125 of file win/line.hh.

10.424.4.2 `template<typename M, unsigned i, typename C > unsigned mln::win::line< M, i, C >::size() const [inline]`

Give the line size, that is, its length.

Definition at line 132 of file win/line.hh.

10.425 mln::win::multiple< W, F > Class Template Reference

Multiple window.

```
#include <multiple.hh>
```

Inherits window_base< W::dpsite, multiple< W, F > >.

10.425.1 Detailed Description

`template<typename W, typename F> class mln::win::multiple< W, F >`

Multiple window.

Definition at line 76 of file multiple.hh.

10.426 mln::win::multiple_size< n, W, F > Class Template Reference

Definition of a multiple-size window.

```
#include <multiple_size.hh>
```

Inherits window_base< W::dpsite, multiple_size< n, W, F > >.

10.426.1 Detailed Description

`template<unsigned n, typename W, typename F> class mln::win::multiple_size< n, W, F >`

Definition of a multiple-size window.

Definition at line 78 of file multiple_size.hh.

10.427 mln::win::octagon2d Struct Reference

Octagon window defined on the 2D square grid.

```
#include <octagon2d.hh>
```

Inherits classical_window_base< dpoint2d, octagon2d >.

Public Member Functions

- unsigned [area](#) () const
Give the area.
- unsigned [length](#) () const
Give the octagon length, that is, its width.
- [octagon2d](#) (unsigned length)
Constructor.

10.427.1 Detailed Description

Octagon window defined on the 2D square grid. An [octagon2d](#) is centered and symmetric.

The length L of the octagon is such as $L = 6 * l + 1$ where $l \geq 0$.

For instance:

```
*      o o o
*     o o o o o
*  o o o o o o o
* o o o x o o o
* o o o o o o o
*   o o o o o
*    o o o
*
```

is defined with $L = 7$ ($l = 1$).

Definition at line 67 of file octagon2d.hh.

10.427.2 Constructor & Destructor Documentation

10.427.2.1 mln::win::octagon2d::octagon2d (unsigned *length*) [inline]

Constructor.

Parameters

[in] *length* Length, of the octagon.

Precondition

length is such as $length = 6 * x + 1$ where $x \geq 0$.

Definition at line 101 of file octagon2d.hh.

10.427.3 Member Function Documentation

10.427.3.1 unsigned mln::win::octagon2d::area () const [inline]

Give the area.

Definition at line 157 of file octagon2d.hh.

10.427.3.2 unsigned mln::win::octagon2d::length () const [inline]

Give the octagon length, that is, its width.

Definition at line 145 of file octagon2d.hh.

10.428 mln::win::rectangle2d Struct Reference

Rectangular window defined on the 2D square grid.

```
#include <rectangle2d.hh>
```

Inherits classical_window_base< dpoint2d, rectangle2d >.

Public Member Functions

- unsigned [area](#) () const
Give the rectangle area.
- unsigned [height](#) () const
Give the rectangle height.
- [rectangle2d](#) (unsigned height, unsigned width)
Constructor.
- const std::vector< [dpoint2d](#) > & [std_vector](#) () const
Give the std vector of delta-points.
- unsigned [width](#) () const
Give the rectangle width.

10.428.1 Detailed Description

Rectangular window defined on the 2D square grid. A [rectangle2d](#) is a 2D window with rectangular shape. It is centered and symmetric.

For instance:

```
*  o o o o o
*  o o x o o
*  o o o o o
*
```

is defined with height = 3 and width = 5.

Definition at line 64 of file rectangle2d.hh.

10.428.2 Constructor & Destructor Documentation

10.428.2.1 mln::win::rectangle2d::rectangle2d (unsigned *height*, unsigned *width*) [inline]

Constructor.

Parameters

[in] *height* Height of the [rectangle2d](#).

[in] *width* Width of the [rectangle2d](#).

Precondition

Height and width are odd.

Definition at line 106 of file rectangle2d.hh.

10.428.3 Member Function Documentation

10.428.3.1 unsigned mln::win::rectangle2d::area () const [inline]

Give the rectangle area.

Definition at line 132 of file rectangle2d.hh.

10.428.3.2 unsigned mln::win::rectangle2d::height () const [inline]

Give the rectangle height.

Definition at line 120 of file rectangle2d.hh.

10.428.3.3 const std::vector< dpoint2d > & mln::win::rectangle2d::std_vector () const [inline]

Give the std vector of delta-points.

Definition at line 145 of file rectangle2d.hh.

10.428.3.4 unsigned mln::win::rectangle2d::width () const [inline]

Give the rectangle width.

Definition at line 126 of file rectangle2d.hh.

10.429 mln::Window< E > Struct Template Reference

Base class for implementation classes that are windows.

```
#include <window.hh>
```

Inheritance diagram for `mln::Window< E >`:



10.429.1 Detailed Description

```
template<typename E> struct mln::Window< E >
```

Base class for implementation classes that are windows.

See also

[mln::doc::Window](#) for a complete documentation of this class contents.

Definition at line 87 of file `concept/window.hh`.

10.430 mln::window< D > Class Template Reference

Generic window class.

```
#include <window.hh>
```

Inherits `window_base< D, window< D > >`.

Public Types

- typedef `dpsites_bkd_piter< window< D > > bkd_qiter`
Site Iterator type to browse the points of a basic window w.r.t. the reverse ordering of delta-points.
- typedef `dpsites_fwd_piter< window< D > > fwd_qiter`
Site Iterator type to browse the points of a basic window w.r.t. the ordering of delta-points.
- typedef `fwd_qiter qiter`
Site Iterator type to browse the points of a basic window whatever the ordering of delta-points.
- typedef `window< D > regular`
Regular window associated type.

Public Member Functions

- void `clear ()`
Clear the window.
- unsigned `delta () const`
Give the maximum coordinate gap between the window center and a window point.
- const D & `dp (unsigned i) const`
*Give the *i*-th delta-point.*
- bool `has (const D &dp) const`
*Test if *dp* is in this window definition.*
- `window< D > & insert (const D &dp)`
*Insert a delta-point *dp*.*
- template<typename W >
`window< D > & insert (const Window< W > &win)`
*Insert another window *win*.*
- bool `is_centered () const`
Test if the window is centered.
- bool `is_empty () const`
Test if the window is empty (null size; no delta-point).
- bool `is_symmetric () const`
- void `print (std::ostream &ostr) const`
*Print the window definition into *ostr*.*
- unsigned `size () const`
Give the window size, i.e., the number of delta-sites.
- const std::vector< D > & `std_vector () const`
Give the std vector of delta-points.
- void `sym ()`
Apply a central symmetry to the target window.
- `window ()`
Constructor without argument.
- `window< D > & insert (const typename D::coord &dind)`

Related Functions

(Note that these are not member functions.)

- `template<typename D >`
`bool operator== (const window< D > &lhs, const window< D > &rhs)`
Equality comparison between windows lhs and rhs.

10.430.1 Detailed Description

`template<typename D> class mln::window< D >`

Generic window class. This type of window is just like a set of delta-points. The parameter is D, type of delta-point.

Definition at line 85 of file window.hh.

10.430.2 Member Typedef Documentation

10.430.2.1 `template<typename D> typedef dpsites_bkd_piter< window<D> > mln::window< D >::bkd_qiter`

[Site_Iterator](#) type to browse the points of a basic window w.r.t. the reverse ordering of delta-points.

Definition at line 124 of file window.hh.

10.430.2.2 `template<typename D> typedef dpsites_fwd_piter< window<D> > mln::window< D >::fwd_qiter`

[Site_Iterator](#) type to browse the points of a basic window w.r.t. the ordering of delta-points.

Definition at line 119 of file window.hh.

10.430.2.3 `template<typename D> typedef fwd_qiter mln::window< D >::qiter`

[Site_Iterator](#) type to browse the points of a basic window whatever the ordering of delta-points.

Definition at line 129 of file window.hh.

10.430.2.4 `template<typename D> typedef window<D> mln::window< D >::regular`

Regular window associated type.

Definition at line 90 of file window.hh.

10.430.3 Constructor & Destructor Documentation

10.430.3.1 `template<typename D > mln::window< D >::window () [inline]`

Constructor without argument.

The constructed window is empty.

Definition at line 207 of file window.hh.

10.430.4 Member Function Documentation

10.430.4.1 `template<typename D > void mln::window< D >::clear () [inline]`

Clear the window.

Definition at line 254 of file window.hh.

10.430.4.2 `template<typename D > unsigned mln::window< D >::delta () const [inline]`

Give the maximum coordinate gap between the window center and a window point.

Definition at line 262 of file window.hh.

References `mln::window< D >::dp()`, and `mln::window< D >::size()`.

10.430.4.3 `template<typename D > const D & mln::window< D >::dp (unsigned i) const [inline]`

Give the *i*-th delta-point.

Definition at line 302 of file window.hh.

References `mln::window< D >::size()`.

Referenced by `mln::window< D >::delta()`, and `mln::window< D >::insert()`.

10.430.4.4 `template<typename D > bool mln::window< D >::has (const D & dp) const [inline]`

Test if `dp` is in this window definition.

Definition at line 311 of file window.hh.

10.430.4.5 `template<typename D > window< D > & mln::window< D >::insert (const D & dp) [inline]`

Insert a delta-point `dp`.

Definition at line 327 of file window.hh.

Referenced by `mln::c18()`, `mln::c26()`, `mln::c2_3d_sli()`, `mln::c4_3d()`, `mln::c6()`, `mln::window< D >::insert()`, `mln::morpho::line_gradient()`, `mln::window< D >::sym()`, `mln::convert::to_upper_window()`, `mln::convert::to_window()`, `mln::win_c4p()`, `mln::win_c4p_3d()`, `mln::win_c8p()`, and `mln::win_c8p_3d()`.

10.430.4.6 `template<typename D > template<typename W > window< D > & mln::window< D >::insert (const Window< W > & win) [inline]`

Insert another window `win`.

Definition at line 337 of file window.hh.

10.430.4.7 `template<typename D> window< D> & mln::window< D>::insert (const typename D::coord & dind) [inline]`

Insertion of a delta-point with different numbers of arguments (coordinates) w.r.t. the dimension.

Definition at line 349 of file window.hh.

References `mln::window< D>::dp()`, and `mln::window< D>::insert()`.

10.430.4.8 `template<typename D> bool mln::window< D>::is_centered () const [inline]`

Test if the window is centered.

Returns

True if the delta-point 0 belongs to the window.

Definition at line 226 of file window.hh.

References `mln::literal::zero`.

10.430.4.9 `template<typename D> bool mln::window< D>::is_empty () const [inline]`

Test if the window is empty (null size; no delta-point).

Definition at line 246 of file window.hh.

10.430.4.10 `template<typename D> bool mln::window< D>::is_symmetric () const [inline]`

Test if the window is symmetric.

Returns

True if for every `dp` of this window, `-dp` is also in this window.

Definition at line 216 of file window.hh.

References `mln::window< D>::sym()`.

10.430.4.11 `template<typename D> void mln::window< D>::print (std::ostream & ostr) const [inline]`

Print the window definition into `ostr`.

Definition at line 390 of file window.hh.

10.430.4.12 `template<typename D> unsigned mln::window< D>::size () const [inline]`

Give the window size, i.e., the number of delta-sites.

Definition at line 294 of file window.hh.

Referenced by `mln::window< D>::delta()`, `mln::window< D>::dp()`, `mln::window< D>::sym()`, `mln::win_c4p()`, `mln::win_c4p_3d()`, `mln::win_c8p()`, and `mln::win_c8p_3d()`.

10.430.4.13 `template<typename D> const std::vector< D> & mln::window< D>::std_vector () const [inline]`

Give the std vector of delta-points.

Definition at line 319 of file window.hh.

10.430.4.14 `template<typename D> void mln::window< D>::sym () [inline]`

Apply a central symmetry to the target window.

Definition at line 234 of file window.hh.

References `mln::window< D>::insert()`, and `mln::window< D>::size()`.

Referenced by `mln::window< D>::is_symmetric()`.

10.430.5 Friends And Related Function Documentation

10.430.5.1 `template<typename D> bool operator==(const window< D> & lhs, const window< D> & rhs) [related]`

Equality comparison between windows `lhs` and `rhs`.

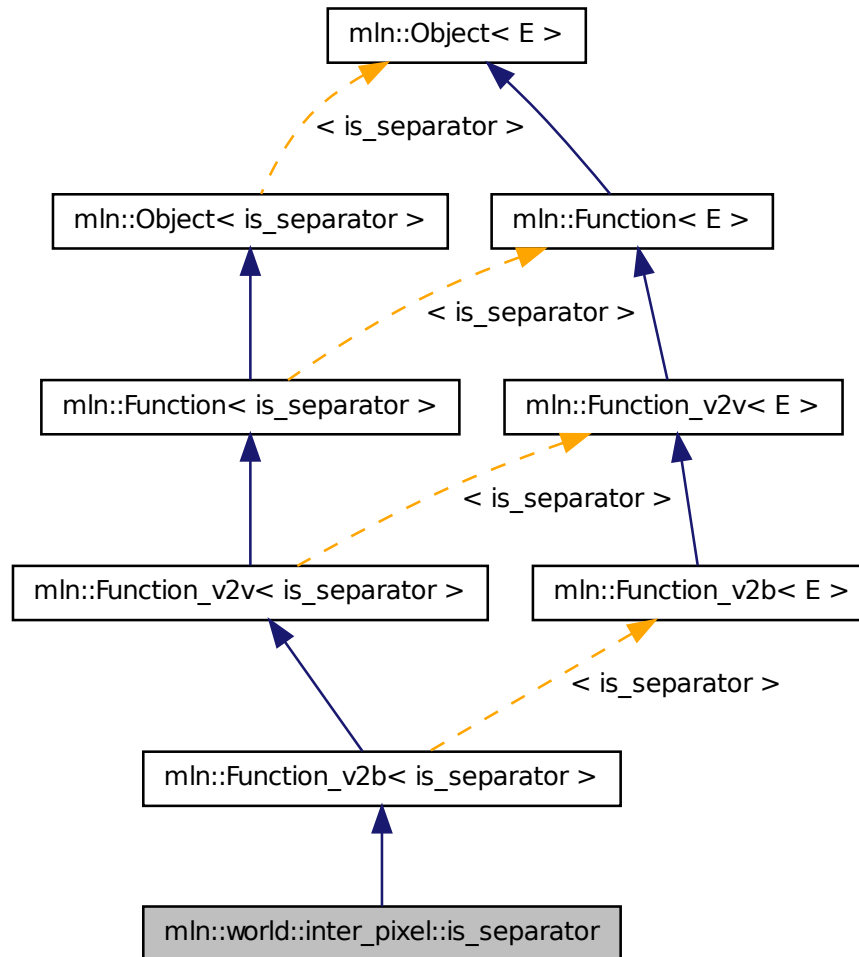
Definition at line 400 of file window.hh.

10.431 mln::world::inter_pixel::is_separator Struct Reference

Functor returning whether a site is a separator in an inter-pixel image.

```
#include <is_separator.hh>
```

Inheritance diagram for mln::world::inter_pixel::is_separator:



10.431.1 Detailed Description

Functor returning whether a site is a separator in an inter-pixel image.

Definition at line 52 of file is_separator.hh.

10.432 trait::graph< I > Struct Template Reference

Graph traits.

```
#include <morpho.hh>
```

10.432.1 Detailed Description

```
template<typename I> struct trait::graph< I >
```

Graph traits.

Definition at line 73 of file `morpho.hh`.

10.433 `trait::graph< mln::complex_image< 1, G, V > >` Struct Template Reference

Graph traits for 1-complexes images.

```
#include <morpho.hh>
```

10.433.1 Detailed Description

```
template<typename G, typename V> struct trait::graph< mln::complex_image< 1, G, V > >
```

Graph traits for 1-complexes images.

Definition at line 133 of file `morpho.hh`.

10.434 `trait::graph< mln::image2d< T > >` Struct Template Reference

Graph traits for [mln::image2d](#).

```
#include <morpho.hh>
```

10.434.1 Detailed Description

```
template<typename T> struct trait::graph< mln::image2d< T > >
```

Graph traits for [mln::image2d](#).

Definition at line 94 of file `morpho.hh`.

Index

- ~decorated_image
 - mln::decorated_image, 597
- ~proxy
 - mln::value::proxy, 1136
- ~soft_heap
 - mln::util::soft_heap, 1092
- ~tracked_ptr
 - mln::util::tracked_ptr, 1095
- _1
 - mln::algebra::h_mat, 545
- 1D neighborhoods, 101
- 1D windows, 112
- 2D neighborhoods, 102
- 2D windows, 113
- 3D neighborhoods, 104
- 3D windows, 115
- a_point_of
 - mln, 150
- abs
 - mln::data, 206
 - mln::math, 355
- abs_inplace
 - mln::data, 206
- Accumulators, 99
- add
 - mln::topo::n_faces_set, 1028
- add_child
 - mln::util::tree_node, 1100
- add_edge
 - mln::util::graph, 1063
- add_face
 - mln::topo::complex, 1007
- add_location
 - mln::geom::complex_geometry, 728
- add_tree_down
 - mln::util::tree, 1097
- add_tree_up
 - mln::util::tree, 1097
- add_vertex
 - mln::util::graph, 1063
- add_vertices
 - mln::util::graph, 1063
- addr
 - mln::topo::complex, 1008
- adj_higher_dim_connected_n_face_bkd_iter
 - mln::topo::adj_higher_dim_connected_n_face_bkd_iter, 979
- adj_higher_dim_connected_n_face_fwd_iter
 - mln::topo::adj_higher_dim_connected_n_face_fwd_iter, 980
- adj_higher_face_bkd_iter
 - mln::topo::adj_higher_face_bkd_iter, 981
- adj_higher_face_fwd_iter
 - mln::topo::adj_higher_face_fwd_iter, 982
- adj_lower_dim_connected_n_face_bkd_iter
 - mln::topo::adj_lower_dim_connected_n_face_bkd_iter, 984
- adj_lower_dim_connected_n_face_fwd_iter
 - mln::topo::adj_lower_dim_connected_n_face_fwd_iter, 985
- adj_lower_face_bkd_iter
 - mln::topo::adj_lower_face_bkd_iter, 986
- adj_lower_face_fwd_iter
 - mln::topo::adj_lower_face_fwd_iter, 987
- adj_lower_higher_face_bkd_iter
 - mln::topo::adj_lower_higher_face_bkd_iter, 988
- adj_lower_higher_face_fwd_iter
 - mln::topo::adj_lower_higher_face_fwd_iter, 989
- adj_m_face_bkd_iter
 - mln::topo::adj_m_face_bkd_iter, 990
- adj_m_face_fwd_iter
 - mln::topo::adj_m_face_fwd_iter, 992
- adjacency_matrix
 - mln::util::adjacency_matrix, 1036, 1037
- adjust
 - mln::border, 190
 - mln::extension, 240, 241
- adjust_duplicate
 - mln::extension, 241
- adjust_fill
 - mln::extension, 241
- algebraic_face
 - mln::topo::algebraic_face, 994, 995
- algebraic_n_face
 - mln::topo::algebraic_n_face, 999, 1000
- and_inplace
 - mln::logical, 327

- and_not
 - mln::logical, 327
- and_not_inplace
 - mln::logical, 328
- antialiased
 - mln::subsampling, 396, 397
- apex
 - mln::util::branch, 1045
- append
 - mln::p_array, 852
 - mln::util::array, 1040, 1041
- apply
 - mln::data, 207
- apply_p2p
 - mln, 150
- area
 - mln::accu::site_set::rectangularity, 509
 - mln::morpho::attribute::sharpness, 839
 - mln::morpho::attribute::volume, 843
 - mln::win::octagon2d, 1170
 - mln::win::rectangle2d, 1171
- argument
 - mln::accu::shape::height, 505
 - mln::accu::shape::volume, 507
 - mln::doc::Accumulator, 600
- array
 - mln::util::array, 1040
- at
 - mln::opt, 388, 389
- attachment
 - mln::make, 335
- backdiag2d
 - mln::win::backdiag2d, 1161
- background
 - mln::labeling, 298
- ball
 - mln::win::ball, 1162
- base_level
 - mln::morpho::attribute::height, 838
- Basic types, 95, 108
- bbox
 - mln::accu::site_set::rectangularity, 509
 - mln::Box, 562
 - mln::box, 556
 - mln::doc::Box, 602
 - mln::doc::Fastest_Image, 610
 - mln::doc::Image, 620
 - mln::geom, 255, 256
 - mln::image1d, 769
 - mln::image2d, 774
 - mln::image3d, 782
 - mln::labeled_image, 794
 - mln::labeled_image_base, 797
 - mln::p_line2d, 890
 - mln::p_run, 916
- bbox_t
 - mln::labeled_image, 793
 - mln::labeled_image_base, 796
- bboxes
 - mln::labeled_image, 794
 - mln::labeled_image_base, 797
- before
 - mln, 164
- begin
 - mln::p_line2d, 890
- bin_1complex_image2d
 - mln, 145
- bin_2complex_image3df
 - mln, 145
- binarization
 - mln::binarization, 189
- bkd_citer
 - mln::topo::complex, 1007
- bkd_eiter
 - mln::util::array, 1039
 - mln::util::set, 1085
- bkd_niter
 - mln::doc::Neighborhood, 624
 - mln::graph_elt_mixed_neighborhood, 736
 - mln::graph_elt_neighborhood, 742
 - mln::graph_elt_neighborhood_if, 743
 - mln::mixed_neighb, 833
 - mln::neighb, 845
- bkd_piter
 - mln::box, 554
 - mln::doc::Box, 602
 - mln::doc::Fastest_Image, 608
 - mln::doc::Image, 618
 - mln::doc::Site_Set, 634
 - mln::hexa, 762
 - mln::image2d_h, 777
 - mln::p_array, 851
 - mln::p_centered, 856
 - mln::p_complex, 859
 - mln::p_edges, 863
 - mln::p_faces, 868
 - mln::p_if, 873
 - mln::p_image, 877
 - mln::p_key, 884
 - mln::p_line2d, 889
 - mln::p_mutable_array_of, 893
 - mln::p_priority, 899
 - mln::p_queue, 905
 - mln::p_queue_fast, 910
 - mln::p_run, 915
 - mln::p_set, 920
 - mln::p_set_of, 924

- mln::p_transformed, 927
- mln::p_vaccess, 932
- mln::p_vertices, 936
- bkd_pixter1d
 - mln::bkd_pixter1d, 548
- bkd_pixter2d
 - mln::bkd_pixter2d, 549
- bkd_pixter3d
 - mln::bkd_pixter3d, 551
- bkd_qiter
 - mln::doc::Weighted_Window, 640
 - mln::doc::Window, 642
 - mln::graph_elt_mixed_window, 738
 - mln::graph_elt_window, 746
 - mln::graph_elt_window_if, 750
 - mln::w_window, 1157
 - mln::window, 1174
- bkd_viter
 - mln::doc::Value_Set, 638
 - mln::value::lut_vec, 1132
- black
 - mln::literal, 323
- blobs
 - mln::canvas::labeling, 197
 - mln::labeling, 299
- blobs_and_compute
 - mln::labeling, 299
- blue
 - mln::literal, 323
- border
 - mln::doc::Fastest_Image, 610
 - mln::image1d, 769
 - mln::image2d, 774
 - mln::image3d, 782
- box
 - mln::box, 555
 - mln::draw, 234
- box1d
 - mln, 145
 - mln::make, 335, 336
- box2d
 - mln, 145
 - mln::make, 336
- box2d_h
 - mln, 146
 - mln::make, 337
- box3d
 - mln, 146
 - mln::make, 338
- box_plain
 - mln::draw, 235
- box_runend_piter
 - mln::box_runend_piter, 565
- box_runstart_piter
 - mln::box_runstart_piter, 567
- branch
 - mln::util::branch, 1044
- brown
 - mln::literal, 323
- buffer
 - mln::doc::Fastest_Image, 610
 - mln::image1d, 769
 - mln::image2d, 774
 - mln::image3d, 782
- c18
 - modneighb3d, 105
- c2
 - modneighb1d, 102
- c26
 - modneighb3d, 105
- c2_3d_sli
 - modneighb3d, 106
- c2_col
 - modneighb2d, 103
- c2_row
 - modneighb2d, 103
- c4
 - modneighb2d, 103
- c4_3d
 - modneighb3d, 106
- c6
 - modneighb3d, 107
- c8
 - modneighb2d, 104
- c8_3d
 - modneighb3d, 107
- can_stop
 - mln::accu::logic::land_basic, 450
 - mln::accu::logic::lor_basic, 453
- Canvas, 99
- card
 - mln::set, 394
- cast
 - mln::value, 428
- Category
 - mln::util::vertex, 1105
- category
 - mln::util::edge, 1053
- cell
 - mln::make, 339
- center
 - mln::p_centered, 857
- center_only_iter
 - mln::topo::center_only_iter, 1003
- center_t
 - mln::graph_elt_mixed_window, 738
 - mln::graph_elt_window, 746

- mln::graph_window_piter, 758
- center_val
 - mln::dpoints_bkd_pixter, 650
 - mln::dpoints_fwd_pixter, 653
- centered_bkd_iter_adapter
 - mln::topo::centered_bkd_iter_adapter, 1004
- centered_fwd_iter_adapter
 - mln::topo::centered_fwd_iter_adapter, 1005
- chamfer
 - mln::geom, 256
- change
 - mln::p_array, 852
- change_both
 - mln::util::couple, 1050
 - mln::util::ord_pair, 1080
- change_extension
 - mln::extension_val, 667
- change_first
 - mln::util::couple, 1050
 - mln::util::ord_pair, 1080
- change_graph
 - mln::util::edge, 1053
 - mln::util::vertex, 1106
- change_key
 - mln::p_key, 885
- change_keys
 - mln::p_key, 885
- change_mask
 - mln::graph_elt_window_if, 752
- change_second
 - mln::util::couple, 1050
 - mln::util::ord_pair, 1080
- change_target
 - mln::complex_psite, 590
 - mln::faces_psite, 670
 - mln::p_transformed_piter, 930
- change_target_site_set
 - mln::graph_window_piter, 759
- change_to
 - mln::pixel, 942
- check_consistency
 - mln::util::tree, 1097
 - mln::util::tree_node, 1100
- children
 - mln::util::tree_node, 1101
- clear
 - mln::p_array, 852
 - mln::p_image, 878
 - mln::p_key, 886
 - mln::p_mutable_array_of, 894
 - mln::p_priority, 900
 - mln::p_queue, 906
 - mln::p_queue_fast, 911
 - mln::p_set, 921
- mln::p_set_of, 925
- mln::util::array, 1041
- mln::util::fibonacci_heap, 1057
- mln::util::set, 1085
- mln::util::soft_heap, 1092
- mln::w_window, 1157
- mln::window, 1175
- closing
 - mln::morpho::elementary, 369
- colorize
 - mln::labeling, 299, 300
- complementation
 - mln::morpho, 360
- complementation_inplace
 - mln::morpho, 360
- complex
 - mln::topo::complex, 1007
- Complex based, 109
- complex_geometry
 - mln::geom::complex_geometry, 728
- complex_image
 - mln::complex_image, 585
- complex_neighborhood_bkd_piter
 - mln::complex_neighborhood_bkd_piter, 587
- complex_neighborhood_fwd_piter
 - mln::complex_neighborhood_fwd_piter, 588
- complex_psite
 - mln::complex_psite, 590
- complex_window_bkd_piter
 - mln::complex_window_bkd_piter, 593
- complex_window_fwd_piter
 - mln::complex_window_fwd_piter, 595
- compose
 - mln, 151
- composed
 - mln::fun::x2x::composed, 707
- compute
 - mln::accu, 166
 - mln::data, 207
 - mln::graph, 266
 - mln::histo, 268
 - mln::labeling, 300–302
 - mln::labeling::impl::generic, 313, 314
 - mln::set, 394
- compute_attribute_image
 - mln::morpho::tree, 374
- compute_attribute_image_from
 - mln::morpho::tree, 374
- compute_fastest
 - mln::labeling::impl, 311
- compute_has
 - mln::p_queue_fast, 911
- compute_image
 - mln::labeling, 302, 303

- compute_parent
 - mln::morpho::tree, 375
- compute_with_weights
 - mln::set, 394
- contrast
 - mln::morpho, 361
- convert
 - mln::data, 208
- convolve
 - mln::linear::local, 319
- coord
 - mln::def, 231
 - mln::doc::Dpoint, 604
 - mln::doc::Fastest_Image, 608
 - mln::doc::Image, 618
 - mln::doc::Point_Site, 629
 - mln::dpoint, 646
 - mln::point, 951
- coordf
 - mln::def, 231
- count
 - mln::accu::stat::mean, 518
- couple
 - mln::make, 339
- cplx
 - mln::p_complex, 860
 - mln::p_faces, 869, 870
 - mln::topo::algebraic_face, 995
 - mln::topo::algebraic_n_face, 1000
 - mln::topo::face, 1011
 - mln::topo::n_face, 1022
- crop_wrt
 - mln::box, 556
- cube3d
 - mln::win::cube3d, 1163
- cuboid3d
 - mln::win::cuboid3d, 1164
- cyan
 - mln::literal, 323
- D
 - mln::topo::is_simple_cell, 1020
- dark_gray
 - mln::literal, 323
- dashed_line
 - mln::draw, 235
- data
 - mln::topo::algebraic_face, 995
 - mln::topo::algebraic_n_face, 1000
 - mln::topo::face, 1011
 - mln::topo::n_face, 1022
- data_t
 - mln::fun::x2x::rotation, 710
 - mln::fun::x2x::translation, 713
- dec_face_id
 - mln::topo::algebraic_face, 995
 - mln::topo::algebraic_n_face, 1000
 - mln::topo::face, 1011
 - mln::topo::n_face, 1022
- dec_n
 - mln::topo::algebraic_face, 995
 - mln::topo::face, 1011
- decorated_image
 - mln::decorated_image, 597
- decoration
 - mln::decorated_image, 597
- deepness
 - mln::util::branch_iter, 1046
 - mln::util::branch_iter_ind, 1048
- delete_tree_node
 - mln::util::tree_node, 1101
- delta
 - mln::doc::Weighted_Window, 641
 - mln::geom, 256
 - mln::graph_elt_mixed_window, 739
 - mln::graph_elt_window, 747
 - mln::graph_elt_window_if, 752
 - mln::graph_window_base, 754
 - mln::point, 951
 - mln::window, 1175
- delta_index
 - mln::doc::Fastest_Image, 611
 - mln::image1d, 769
 - mln::image2d, 774
 - mln::image3d, 782
- depth
 - mln::win::cuboid3d, 1165
- detach
 - mln::topo, 404
- detachment
 - mln::make, 339
- diag2d
 - mln::win::diag2d, 1166
- diameter
 - mln::win::ball, 1162
- diff
 - mln::Box, 563
 - mln::Site_Set, 970
 - mln::win, 435
- diff_abs
 - mln::arith, 178
- dilation
 - mln::morpho, 361
- dim
 - mln::complex_image, 585
 - mln::doc::Dpoint, 605
 - mln::doc::Point_Site, 630
 - mln::dpoint, 647

- mln::point, 952
- direct
 - mln::morpho::tree::filter, 381
- discrete_plane_1complex_geometry
 - mln, 146
- discrete_plane_2complex_geometry
 - mln, 146
- disk2d
 - modwin2d, 114
- display_branch
 - mln::util, 419
- display_tree
 - mln::util, 419
- distance_and_closest_point_geodesic
 - mln::transform, 411, 412
- distance_and_influence_zone_geodesic
 - mln::transform, 412
- distance_front
 - mln::canvas, 195
 - mln::transform, 413
- distance_geodesic
 - mln::canvas, 195
 - mln::transform, 413
- div
 - mln::arith, 178
- div_cst
 - mln::arith, 178
- div_inplace
 - mln::arith, 179
- domain
 - mln::complex_image, 585
 - mln::doc::Fastest_Image, 611
 - mln::doc::Image, 620
 - mln::extended, 661
 - mln::flat_image, 673
 - mln::hexa, 763
 - mln::image1d, 769
 - mln::image2d, 774
 - mln::image2d_h, 778
 - mln::image3d, 783
 - mln::image_if, 786
 - mln::lazy_image, 800
 - mln::p2p_image, 848
 - mln::slice_image, 973
 - mln::sub_image, 975
 - mln::sub_image_if, 976
 - mln::tr_image, 1034
 - mln::unproject_image, 1035
- Domain morphers, 97
- domain_t
 - mln::value::stack_image, 1144
- dp
 - mln::window, 1175
- dpoint
 - mln::doc::Dpoint, 605
 - mln::doc::Fastest_Image, 608
 - mln::doc::Image, 618
 - mln::doc::Neighborhood, 624
 - mln::doc::Point_Site, 629
 - mln::doc::Weighted_Window, 640
 - mln::dpoint, 647
- dpoint1d
 - mln, 146
- dpoint2d
 - mln, 146
- dpoint2d_h
 - mln, 146
 - mln::make, 340
- dpoint3d
 - mln, 147
- dpoints_bkd_pixter
 - mln::dpoints_bkd_pixter, 650
- dpoints_fwd_pixter
 - mln::dpoints_fwd_pixter, 652
- dpsite
 - mln::point, 951
 - mln::w_window, 1157
- dpsites_bkd_piter
 - mln::dpsites_bkd_piter, 655
- dpsites_fwd_piter
 - mln::dpsites_fwd_piter, 656
- draw_graph
 - mln::debug, 226, 227
- dual_input_max_tree
 - mln::morpho::tree, 376
- dummy_p_edges
 - mln::make, 340
- dummy_p_vertices
 - mln::make, 341
- duplicate
 - mln, 151
 - mln::border, 191
 - mln::extension, 241
- e_ith_nbh_edge
 - mln::util::graph, 1063
 - mln::util::line_graph, 1072
- e_nmax
 - mln::util::graph, 1063
 - mln::util::line_graph, 1072
- e_nmax_nbh_edges
 - mln::util::graph, 1064
 - mln::util::line_graph, 1072
- edge
 - mln::p_edges, 863
 - mln::topo, 404
 - mln::util::edge, 1053
 - mln::util::graph, 1064

- mln::util::line_graph, 1072
- edge_fwd_iter
 - mln::util::graph, 1061
 - mln::util::line_graph, 1071
- edge_image
 - mln::edge_image, 659
 - mln::make, 341–343
- edge_nbh_edge_fwd_iter
 - mln::util::graph, 1061
 - mln::util::line_graph, 1071
- edge_nbh_t
 - mln::edge_image, 658
- edge_win_t
 - mln::edge_image, 658
- edge_with
 - mln::util::vertex, 1106
- edges
 - mln::util::graph, 1064
- edges_set_t
 - mln::util::graph, 1061
- edges_t
 - mln::util::graph, 1061
 - mln::util::line_graph, 1071
- eiter
 - mln::util::array, 1039
 - mln::util::set, 1085
- element
 - mln::box, 554
 - mln::graph_window_if_piter, 756
 - mln::graph_window_piter, 760
 - mln::image1d, 769
 - mln::image2d, 774
 - mln::image3d, 783
 - mln::p_array, 851
 - mln::p_centered, 856
 - mln::p_complex, 859
 - mln::p_edges, 863
 - mln::p_faces, 868
 - mln::p_if, 873
 - mln::p_image, 877
 - mln::p_key, 884
 - mln::p_line2d, 889
 - mln::p_mutable_array_of, 893
 - mln::p_priority, 900
 - mln::p_queue, 905
 - mln::p_queue_fast, 910
 - mln::p_run, 915
 - mln::p_set, 920
 - mln::p_set_of, 924
 - mln::p_transformed, 927
 - mln::p_vaccess, 932
 - mln::p_vertices, 936
 - mln::util::array, 1039
 - mln::util::set, 1085
- mln::util::soft_heap, 1091
- elt
 - mln::util::tree_node, 1101
- empty
 - mln::p_queue_fast, 911
- enc
 - mln::value::float01, 1111
 - mln::value::label, 1128
 - mln::value::proxy, 1135
 - mln::value::sign, 1142
- end
 - mln::p_line2d, 891
 - mln::p_run, 916
- enlarge
 - mln::box, 556
- equalize
 - mln::border, 191
 - mln::histo, 268
- equiv
 - mln::value, 428
 - mln::value::float01, 1111
 - mln::value::proxy, 1135
 - mln::value::sign, 1142
- erosion
 - mln::morpho, 361
- exists_key
 - mln::p_key, 886
- exists_priority
 - mln::p_priority, 901
- extend
 - mln, 151, 152
- extended
 - mln::extended, 660
- extension
 - mln::extension_fun, 663
 - mln::extension_ima, 665
 - mln::extension_val, 667
- extension_fun
 - mln::extension_fun, 662
- extension_ima
 - mln::extension_ima, 665
- extension_val
 - mln::extension_val, 667
- f_hsi_to_rgb_3x8
 - mln::fun::v2v, 247
- f_hsl_to_rgb_3x8
 - mln::fun::v2v, 247
- f_rgb_to_hsi_f
 - mln::fun::v2v, 247
- f_rgb_to_hsl_f
 - mln::fun::v2v, 247
- face
 - mln::complex_psite, 591

- mln::faces_psite, 670
- mln::topo::face, 1011
- face_bkd_iter
 - mln::topo::face_bkd_iter, 1014
- face_fwd_iter
 - mln::topo::face_fwd_iter, 1015
- face_id
 - mln::complex_psite, 591
 - mln::faces_psite, 670
 - mln::topo::algebraic_face, 996
 - mln::topo::algebraic_n_face, 1000
 - mln::topo::face, 1012
 - mln::topo::n_face, 1022
- faces
 - mln::topo::n_faces_set, 1028
- faces_psite
 - mln::faces_psite, 669
- faces_type
 - mln::topo::n_faces_set, 1027
- fast_median
 - mln::data, 208
- fibonacci_heap
 - mln::util::fibonacci_heap, 1057
- filename
 - mln::debug, 227
- fill
 - mln::border, 191
 - mln::data, 208
 - mln::extension, 241
 - mln::util::array, 1041
- fill_holes
 - mln::labeling, 304
- fill_with_image
 - mln::data, 209
 - mln::data::impl::generic, 221
- fill_with_value
 - mln::data, 209
 - mln::data::impl::generic, 221
- filter
 - mln::morpho::tree::filter, 381
- find
 - mln::border, 192
- first
 - mln::accu::pair, 500
 - mln::accu::stat::min_max, 528
 - mln::util::couple, 1050
 - mln::util::ord_pair, 1080
 - mln::util::site_pair, 1089
- first_accu
 - mln::accu::pair, 500
 - mln::accu::stat::min_max, 528
- first_element
 - mln::util::set, 1085
- flat_image
 - mln::flat_image, 672
- flat_zones
 - mln::labeling, 304
- float01
 - mln::value::float01, 1111
- float01_16
 - mln::value, 426
- float01_8
 - mln::value, 426
- float01_f
 - mln::value::float01_f, 1113
- float_2complex_image3df
 - mln, 147
- flooding
 - mln::morpho::watershed, 383
- foreground
 - mln::labeling, 304
- format
 - mln::debug, 227, 228
- from_to
 - mln::convert, 200
- front
 - mln::p_priority, 901
 - mln::p_queue, 906
 - mln::p_queue_fast, 911
 - mln::util::fibonacci_heap, 1057
- fun
 - mln::p2p_image, 848
- fun_image
 - mln::fun_image, 716
- fun_t
 - mln::p_edges, 864
 - mln::p_vertices, 936
- Function
 - mln::Function, 717
- function
 - mln::p_edges, 866
 - mln::p_transformed, 928
 - mln::p_vertices, 939
- Functions, 100
- fwd_citer
 - mln::topo::complex, 1007
- fwd_eiter
 - mln::util::array, 1040
 - mln::util::set, 1085
- fwd_niter
 - mln::doc::Neighborhood, 624
 - mln::graph_elt_mixed_neighborhood, 736
 - mln::graph_elt_neighborhood, 742
 - mln::graph_elt_neighborhood_if, 743
 - mln::mixed_neighb, 833
 - mln::neighb, 845
- fwd_piter
 - mln::box, 554

- mln::doc::Box, 602
- mln::doc::Fastest_Image, 608
- mln::doc::Image, 618
- mln::doc::Site_Set, 634
- mln::hexa, 762
- mln::image2d_h, 777
- mln::p_array, 851
- mln::p_centered, 856
- mln::p_complex, 859
- mln::p_edges, 864
- mln::p_faces, 868
- mln::p_if, 873
- mln::p_image, 877
- mln::p_key, 884
- mln::p_line2d, 889
- mln::p_mutable_array_of, 893
- mln::p_priority, 900
- mln::p_queue, 905
- mln::p_queue_fast, 910
- mln::p_run, 915
- mln::p_set, 920
- mln::p_set_of, 924
- mln::p_transformed, 927
- mln::p_vaccess, 932
- mln::p_vertices, 937
- fwd_pixter1d
 - mln::fwd_pixter1d, 722
- fwd_pixter2d
 - mln::fwd_pixter2d, 723
- fwd_pixter3d
 - mln::fwd_pixter3d, 725
- fwd_qiter
 - mln::doc::Weighted_Window, 640
 - mln::doc::Window, 642
 - mln::graph_elt_mixed_window, 738
 - mln::graph_elt_window, 746
 - mln::graph_elt_window_if, 750
 - mln::w_window, 1157
 - mln::window, 1174
- fwd_viter
 - mln::doc::Value_Set, 638
 - mln::value::lut_vec, 1132
- gaussian
 - mln::linear, 315
- gaussian_1st_derivative
 - mln::linear, 316
- gaussian_2nd_derivative
 - mln::linear, 316, 317
- gaussian_subsampling
 - mln::subsampling, 397
- general
 - mln::morpho, 361
- geom
 - mln::complex_image, 584
 - mln::p_complex, 860
- get
 - mln::border, 192
 - mln::set, 395
- get_header
 - mln::io::dicom, 272
 - mln::io::dump, 273
 - mln::io::raw, 293
- get_rot
 - mln::registration, 391
- gl16
 - mln::value, 427
- gl8
 - mln::value, 427
- glf
 - mln::value, 427
- gradient
 - mln::morpho, 361
- gradient_external
 - mln::morpho, 362
- gradient_internal
 - mln::morpho, 362
- graph
 - mln::p_edges, 866
 - mln::p_graph_piter, 871
 - mln::p_vertices, 939
 - mln::util::edge, 1053
 - mln::util::graph, 1062
 - mln::util::line_graph, 1073
 - mln::util::vertex, 1106
- Graph based, 109
- graph_element
 - mln::graph_elt_mixed_window, 739
 - mln::graph_elt_window, 747
 - mln::graph_window_piter, 758
 - mln::p_edges, 864
 - mln::p_vertices, 937
- graph_elt_neighborhood_if
 - mln::graph_elt_neighborhood_if, 744
- graph_elt_window_if
 - mln::graph_elt_window_if, 752
- graph_t
 - mln::edge_image, 658
 - mln::p_edges, 864
 - mln::p_vertices, 937
 - mln::util::edge, 1053
 - mln::util::vertex, 1105
 - mln::vertex_image, 1152
- graph_window_if_piter
 - mln::graph_window_if_piter, 756
- graph_window_piter
 - mln::graph_window_piter, 759
- Graphes, 94

- graylevel
 - mln::value::graylevel, 1115
- graylevel_f
 - mln::value::graylevel_f, 1118
- green
 - mln::literal, 323
- grid
 - mln::dpoint, 646
 - mln::point, 951
- h_mat
 - mln::algebra::h_mat, 544
 - mln::make, 343
- h_vec
 - mln::algebra::h_vec, 546
 - mln::point, 951
- has
 - mln::box, 556
 - mln::doc::Box, 603
 - mln::doc::Fastest_Image, 611
 - mln::doc::Image, 620
 - mln::doc::Site_Set, 634
 - mln::doc::Value_Set, 638
 - mln::extension_fun, 663
 - mln::extension_ima, 665
 - mln::extension_val, 668
 - mln::flat_image, 673
 - mln::hexa, 763
 - mln::image1d, 770
 - mln::image2d, 775
 - mln::image2d_h, 778
 - mln::image3d, 783
 - mln::interpolated, 788
 - mln::lazy_image, 800
 - mln::p_array, 852, 853
 - mln::p_centered, 857
 - mln::p_complex, 860
 - mln::p_edges, 866
 - mln::p_if, 874
 - mln::p_image, 878
 - mln::p_key, 886
 - mln::p_line2d, 891
 - mln::p_mutable_array_of, 894
 - mln::p_priority, 901
 - mln::p_queue, 906
 - mln::p_queue_fast, 911
 - mln::p_run, 917
 - mln::p_set, 921
 - mln::p_set_of, 925
 - mln::p_transformed, 928
 - mln::p_vaccess, 933
 - mln::p_vertices, 939
 - mln::set, 395
 - mln::tr_image, 1034
 - mln::util::line_graph, 1073
 - mln::util::set, 1086
 - mln::value::lut_vec, 1133
 - mln::window, 1175
- has_e
 - mln::util::graph, 1064
 - mln::util::line_graph, 1073
- has_index
 - mln::p_run, 917
- has_v
 - mln::util::graph, 1064
 - mln::util::line_graph, 1073
- height
 - mln::morpho::attribute::sharpness, 839
 - mln::win::cuboid3d, 1165
 - mln::win::rectangle2d, 1171
- hexa
 - mln::hexa, 763
- higher_dim_adj_faces
 - mln::topo::algebraic_face, 996
 - mln::topo::algebraic_n_face, 1000
 - mln::topo::face, 1012
 - mln::topo::n_face, 1022
- highest_priority
 - mln::p_priority, 901
- histo3d_rgb
 - mln::accu::stat::histo3d_rgb, 513
- hit_or_miss
 - mln::morpho, 362
- hit_or_miss_background_closing
 - mln::morpho, 362
- hit_or_miss_background_opening
 - mln::morpho, 362
- hit_or_miss_closing
 - mln::morpho, 363
- hit_or_miss_opening
 - mln::morpho, 363
- hline2d
 - modwin2d, 114
- horizontal_symmetry
 - mln::geom, 256
- hough
 - mln::transform, 413
- i_element
 - mln::p_array, 851
 - mln::p_image, 877
 - mln::p_key, 885
 - mln::p_mutable_array_of, 893
 - mln::p_priority, 900
 - mln::p_queue, 905
 - mln::p_queue_fast, 910
 - mln::p_set, 920
 - mln::p_set_of, 924

- mln::p_vaccess, 932
- icp
 - mln::registration, 391, 392
- id
 - mln::graph_window_if_piter, 756
 - mln::graph_window_piter, 760
 - mln::p_graph_piter, 871
 - mln::util::edge, 1053
 - mln::util::vertex, 1106
- id_t
 - mln::util::edge, 1053
 - mln::util::vertex, 1105
- id_value_t
 - mln::util::edge, 1053
 - mln::util::vertex, 1105
- identity
 - mln::literal, 324
- Identity morphers, 98
- ima
 - mln::doc::Generalized_Pixel, 615
 - mln::doc::Pixel_Iterator, 627
 - mln::fun::x2x::linear, 708
 - mln::util::pix, 1082
- image
 - mln::bkd_pixter1d, 548
 - mln::bkd_pixter2d, 549
 - mln::bkd_pixter3d, 550
 - mln::doc::Generalized_Pixel, 615
 - mln::doc::Pixel_Iterator, 627
 - mln::fwd_pixter1d, 722
 - mln::fwd_pixter2d, 723
 - mln::fwd_pixter3d, 724
 - mln::make, 343, 344
 - mln::pw::image, 961
- Image morphers, 96
- image1d
 - mln::image1d, 768
- image2d
 - mln::image2d, 773
 - mln::make, 344
- image2d_h
 - mln::image2d_h, 778
- image3d
 - mln::image3d, 782
 - mln::make, 344, 345
- image_if
 - mln::image_if, 785
- Images, 95
- implies
 - mln, 152
- inc_face_id
 - mln::topo::algebraic_face, 996
 - mln::topo::algebraic_n_face, 1001
 - mln::topo::face, 1012
- mln::topo::n_face, 1022
- inc_n
 - mln::topo::algebraic_face, 996
 - mln::topo::face, 1012
- index
 - mln::p_indexed_bkd_piter, 880
 - mln::p_indexed_fwd_piter, 882
- index_of
 - mln::doc::Value_Set, 638
 - mln::value::lut_vec, 1133
- influence_zone_adjacency_graph
 - mln::make, 345
- influence_zone_front
 - mln::transform, 413, 414
- influence_zone_geodesic
 - mln::transform, 414
- influence_zone_geodesic_saturated
 - mln::transform, 414, 415
- init
 - mln::accu::center, 438
 - mln::accu::convolve, 439
 - mln::accu::count_adjacent_vertices, 441
 - mln::accu::count_labels, 443
 - mln::accu::count_value, 444
 - mln::accu::label_used, 447
 - mln::accu::logic::land, 449
 - mln::accu::logic::land_basic, 450
 - mln::accu::logic::lor, 452
 - mln::accu::logic::lor_basic, 453
 - mln::accu::maj_h, 455
 - mln::accu::math::count, 456
 - mln::accu::math::inf, 458
 - mln::accu::math::sum, 459
 - mln::accu::math::sup, 460
 - mln::accu::max_site, 462
 - mln::accu::nil, 496
 - mln::accu::p, 497
 - mln::accu::pair, 500
 - mln::accu::rms, 502
 - mln::accu::shape::bbox, 503
 - mln::accu::shape::height, 505
 - mln::accu::shape::volume, 507
 - mln::accu::stat::deviation, 511
 - mln::accu::stat::histo3d_rgb, 513
 - mln::accu::stat::max, 515
 - mln::accu::stat::max_h, 516
 - mln::accu::stat::mean, 518
 - mln::accu::stat::median_h, 522
 - mln::accu::stat::min, 524
 - mln::accu::stat::min_h, 526
 - mln::accu::stat::min_max, 528
 - mln::accu::stat::rank, 530
 - mln::accu::stat::rank < bool >, 531
 - mln::accu::stat::rank_high_quant, 533

- mln::accu::stat::var, 535
- mln::accu::stat::variance, 537
- mln::accu::tuple, 539
- mln::accu::val, 541
- mln::doc::Accumulator, 600
- mln::morpho::attribute::card, 835
- mln::morpho::attribute::count_adjacent_vertices, 836
- mln::morpho::attribute::height, 838
- mln::morpho::attribute::sharpness, 840
- mln::morpho::attribute::sum, 841
- mln::morpho::attribute::volume, 843
- mln::p_run, 917
- initialize
 - mln, 152
- insert
 - mln::p_array, 853
 - mln::p_image, 878
 - mln::p_key, 886
 - mln::p_mutable_array_of, 894
 - mln::p_priority, 901
 - mln::p_queue, 906
 - mln::p_queue_fast, 912
 - mln::p_set, 921
 - mln::p_set_of, 925
 - mln::p_vaccess, 933
 - mln::util::set, 1086
 - mln::w_window, 1157
 - mln::window, 1175
- int_s
 - mln::value::int_s, 1121
- int_s16
 - mln::value, 427
- int_s32
 - mln::value, 427
- int_s8
 - mln::value, 427
- int_u
 - mln::value::int_u, 1123
- int_u12
 - mln::value, 427
- int_u16
 - mln::value, 427
- int_u32
 - mln::value, 427
- int_u8
 - mln::value, 428
- int_u8_1complex_image2d
 - mln, 147
- int_u8_2complex_image2d
 - mln, 147
- int_u8_2complex_image3df
 - mln, 147
- int_u_sat
 - mln::value::int_u_sat, 1125
- inter
 - mln::Box, 563
 - mln::Site_Set, 970
- interpolated
 - mln::interpolated, 788
- inv
 - mln::fun::x2x::rotation, 711
 - mln::fun::x2x::translation, 714
- invalidate
 - mln::complex_psite, 591
 - mln::doc::Iterator, 622
 - mln::doc::Pixel_Iterator, 628
 - mln::doc::Site_Iterator, 632
 - mln::doc::Value_Iterator, 636
 - mln::dpoints_bkd_pixter, 650
 - mln::dpoints_fwd_pixter, 653
 - mln::faces_psite, 670
 - mln::p_edges, 866
 - mln::p_vertices, 939
 - mln::topo::algebraic_face, 996
 - mln::topo::algebraic_n_face, 1001
 - mln::topo::face, 1012
 - mln::topo::n_face, 1023
 - mln::util::branch_iter, 1046
 - mln::util::branch_iter_ind, 1048
 - mln::util::edge, 1054
 - mln::util::vertex, 1106
- invert
 - mln::fun::x2x::rotation, 710
 - mln::fun::x2x::translation, 713
- iota
 - mln::debug, 228
- is_centered
 - mln::doc::Weighted_Window, 641
 - mln::graph_elt_mixed_window, 739
 - mln::graph_elt_window, 747
 - mln::graph_elt_window_if, 752
 - mln::graph_window_base, 754
 - mln::window, 1176
- is_empty
 - mln::Box, 562
 - mln::box, 557
 - mln::doc::Weighted_Window, 641
 - mln::graph_elt_mixed_window, 740
 - mln::graph_elt_window, 748
 - mln::graph_elt_window_if, 752
 - mln::graph_window_base, 754
 - mln::util::array, 1041
 - mln::util::fibonacci_heap, 1057
 - mln::util::set, 1086
 - mln::util::soft_heap, 1092
 - mln::window, 1176
- is_facet

- mln::topo, 404
- is_subgraph_of
 - mln::util::graph, 1065
 - mln::util::line_graph, 1074
- is_symmetric
 - mln::graph_elt_mixed_window, 740
 - mln::graph_elt_window, 748
 - mln::graph_elt_window_if, 752
 - mln::graph_window_base, 755
 - mln::w_window, 1158
 - mln::window, 1176
- is_valid
 - mln::accu::center, 438
 - mln::accu::convolve, 439
 - mln::accu::count_adjacent_vertices, 441
 - mln::accu::count_labels, 443
 - mln::accu::count_value, 444
 - mln::accu::histo, 446
 - mln::accu::label_used, 447
 - mln::accu::logic::land, 449
 - mln::accu::logic::land_basic, 450
 - mln::accu::logic::lor, 452
 - mln::accu::logic::lor_basic, 453
 - mln::accu::maj_h, 455
 - mln::accu::math::count, 456
 - mln::accu::math::inf, 458
 - mln::accu::math::sum, 459
 - mln::accu::math::sup, 460
 - mln::accu::max_site, 462
 - mln::accu::nil, 496
 - mln::accu::p, 497
 - mln::accu::pair, 500
 - mln::accu::rms, 502
 - mln::accu::shape::bbox, 503
 - mln::accu::shape::height, 505
 - mln::accu::shape::volume, 507
 - mln::accu::stat::deviation, 511
 - mln::accu::stat::histo3d_rgb, 513
 - mln::accu::stat::max, 515
 - mln::accu::stat::max_h, 516
 - mln::accu::stat::mean, 518
 - mln::accu::stat::median_alt, 520
 - mln::accu::stat::median_h, 522
 - mln::accu::stat::min, 524
 - mln::accu::stat::min_h, 526
 - mln::accu::stat::min_max, 528
 - mln::accu::stat::rank, 530
 - mln::accu::stat::rank < bool >, 531
 - mln::accu::stat::rank_high_quant, 533
 - mln::accu::stat::var, 535
 - mln::accu::stat::variance, 537
 - mln::accu::tuple, 539
 - mln::accu::val, 541
 - mln::box, 557
 - mln::complex_psite, 591
 - mln::doc::Fastest_Image, 611
 - mln::doc::Image, 620
 - mln::doc::Iterator, 622
 - mln::doc::Pixel_Iterator, 628
 - mln::doc::Site_Iterator, 632
 - mln::doc::Value_Iterator, 636
 - mln::dpoints_bkd_pixter, 650
 - mln::dpoints_fwd_pixter, 653
 - mln::faces_psite, 670
 - mln::graph_elt_mixed_window, 740
 - mln::graph_elt_window, 748
 - mln::graph_elt_window_if, 753
 - mln::graph_window_base, 755
 - mln::interpolated, 788
 - mln::morpho::attribute::card, 835
 - mln::morpho::attribute::count_adjacent_vertices, 836
 - mln::morpho::attribute::height, 838
 - mln::morpho::attribute::sharpness, 840
 - mln::morpho::attribute::sum, 841
 - mln::morpho::attribute::volume, 843
 - mln::p_array, 853
 - mln::p_centered, 857
 - mln::p_complex, 861
 - mln::p_edges, 866
 - mln::p_faces, 870
 - mln::p_if, 874
 - mln::p_image, 878
 - mln::p_key, 887
 - mln::p_line2d, 891
 - mln::p_mutable_array_of, 894
 - mln::p_priority, 902
 - mln::p_queue, 907
 - mln::p_queue_fast, 912
 - mln::p_run, 917
 - mln::p_set, 921
 - mln::p_set_of, 925
 - mln::p_transformed, 928
 - mln::p_vaccess, 934
 - mln::p_vertices, 940
 - mln::pixel, 942
 - mln::topo::algebraic_face, 996
 - mln::topo::algebraic_n_face, 1001
 - mln::topo::face, 1012
 - mln::topo::n_face, 1023
 - mln::tr_image, 1034
 - mln::util::branch_iter, 1046
 - mln::util::branch_iter_ind, 1048
 - mln::util::edge, 1054
 - mln::util::fibonacci_heap, 1058
 - mln::util::soft_heap, 1092
 - mln::util::vertex, 1106
 - mln::value::stack_image, 1145

- iter
 - mln::complex_neighborhood_bkd_piter, 587
 - mln::complex_neighborhood_fwd_piter, 589
 - mln::complex_window_bkd_piter, 593
 - mln::complex_window_fwd_piter, 595
- iter_type
 - mln::complex_neighborhood_bkd_piter, 586
 - mln::complex_neighborhood_fwd_piter, 588
 - mln::complex_window_bkd_piter, 593
 - mln::complex_window_fwd_piter, 594
- ith_nbh_edge
 - mln::util::edge, 1054
 - mln::util::vertex, 1106
- ith_nbh_vertex
 - mln::util::vertex, 1107
- k
 - mln::accu::stat::rank, 530
- key
 - mln::p_key, 887
- keys
 - mln::p_key, 887
- l1
 - mln::norm, 386
- l1_distance
 - mln::norm, 386
- l2
 - mln::norm, 386
- l2_distance
 - mln::norm, 387
- label
 - mln::value::label, 1129
- label_16
 - mln::value, 428
- label_32
 - mln::value, 428
- label_8
 - mln::value, 428
- labeled_image
 - mln::labeled_image, 793
- labeled_image_base
 - mln::labeled_image_base, 796
- labeling
 - mln::graph, 266
- laplacian
 - mln::morpho, 363
- larger_than
 - mln, 152
- last
 - mln::util::array, 1041
- last_coord
 - mln::point, 953
- last_element
 - mln::util::set, 1087
- lazy_image
 - mln::lazy_image, 800
- ldlt_decomp
 - mln::algebra, 174
- ldlt_solve
 - mln::algebra, 174
- lemmings
 - mln::util, 419
- len
 - mln::Box, 562
 - mln::box, 557
- length
 - mln::p_run, 917
 - mln::win::backdiag2d, 1161
 - mln::win::cube3d, 1163
 - mln::win::diag2d, 1166
 - mln::win::line, 1168
 - mln::win::octagon2d, 1170
- light_gray
 - mln::literal, 324
- lime
 - mln::literal, 324
- line
 - mln::accu, 166
 - mln::draw, 236
 - mln::win::line, 1167
- line_gradient
 - mln::morpho, 363
- linear
 - mln::fun::x2x::linear, 708
- linfty
 - mln::norm, 387
- linfty_distance
 - mln::norm, 387
- load
 - mln::io::cloud, 271
 - mln::io::dicom, 272
 - mln::io::dump, 273
 - mln::io::fits, 274, 275
 - mln::io::fld, 276
 - mln::io::magick, 277
 - mln::io::off, 278
 - mln::io::pbm, 280
 - mln::io::pbms, 281
 - mln::io::pfm, 283
 - mln::io::pgm, 284
 - mln::io::pgms, 285
 - mln::io::plot, 286
 - mln::io::pnm, 288
 - mln::io::pnms, 290
 - mln::io::ppm, 291
 - mln::io::ppms, 292
 - mln::io::raw, 293

- mln::io::tiff, 294
- load_ascii_builtin
 - mln::io::pnm, 288
- load_ascii_value
 - mln::io::pnm, 288
- load_raw_2d
 - mln::io::pnm, 288
- lower_dim_adj_faces
 - mln::topo::algebraic_face, 996
 - mln::topo::algebraic_n_face, 1001
 - mln::topo::face, 1012
 - mln::topo::n_face, 1023
- lowest_priority
 - mln::p_priority, 902
- lut_vec
 - mln::value::lut_vec, 1133
- lvalue
 - mln::complex_image, 584
 - mln::decorated_image, 596
 - mln::doc::Fastest_Image, 609
 - mln::doc::Image, 618
 - mln::doc::Pixel_Iterator, 627
 - mln::flat_image, 672
 - mln::fun_image, 715
 - mln::hexa, 762
 - mln::image1d, 768
 - mln::image2d, 773
 - mln::image2d_h, 777
 - mln::image3d, 781
 - mln::interpolated, 787
 - mln::lazy_image, 799
 - mln::tr_image, 1033
 - mln::value::stack_image, 1144
 - mln::violent_cast_image, 1154
- magenta
 - mln::literal, 324
- main_branch
 - mln::util::tree, 1098
- make_algebraic_face
 - mln::topo, 405
- make_algebraic_n_face
 - mln::topo, 405
- make_debug_graph_image
 - mln, 152
- make_greater_point
 - mln::util, 419
- make_greater_psite
 - mln::util, 419
- mask
 - mln::graph_elt_neighborhood_if, 744
 - mln::graph_elt_window_if, 753
- mask_t
 - mln::graph_elt_window_if, 751
- mat
 - mln::make, 345
- max
 - mln::literal, 324
 - mln::morpho::tree::filter, 381
- max_col
 - mln::geom, 256, 257
- max_component
 - mln::io::pnm, 289
- max_ind
 - mln::geom, 257
- max_row
 - mln::geom, 257
- max_sli
 - mln::geom, 257
- max_tree
 - mln::morpho::tree, 376
- mean
 - mln::accu::stat::var, 535
 - mln::accu::stat::variance, 537
 - mln::estim, 238
- mean_t
 - mln::accu::stat::var, 535
- median
 - mln::data, 210
 - mln::data::approx, 216, 217
 - mln::data::naive, 224
- medium_gray
 - mln::literal, 324
- memory_size
 - mln::box, 557
 - mln::p_array, 853
 - mln::p_centered, 857
 - mln::p_edges, 867
 - mln::p_if, 874
 - mln::p_image, 878
 - mln::p_key, 887
 - mln::p_line2d, 891
 - mln::p_mutable_array_of, 895
 - mln::p_priority, 902
 - mln::p_queue, 907
 - mln::p_queue_fast, 912
 - mln::p_run, 917
 - mln::p_set, 921
 - mln::p_set_of, 925
 - mln::p_transformed, 928
 - mln::p_vaccess, 934
 - mln::p_vertices, 940
 - mln::util::array, 1042
 - mln::util::set, 1087
- merge
 - mln::box, 557
- mesh
 - mln::doc::Point_Site, 629

- mesh_corner_point_area
 - mln::geom, 257
- mesh_curvature
 - mln::geom, 258
- mesh_normal
 - mln::geom, 258
- meyer_wst
 - mln::morpho, 364
- min
 - mln::arith, 179
 - mln::literal, 324
 - mln::morpho, 364
 - mln::morpho::tree::filter, 381
- min_col
 - mln::geom, 259
- min_ind
 - mln::geom, 259
- min_inplace
 - mln::arith, 179
 - mln::morpho, 364
- min_max
 - mln::estim, 238
- min_row
 - mln::geom, 259
- min_sli
 - mln::geom, 259
- min_tree
 - mln::morpho::tree, 377
- minus
 - mln::arith, 180, 181
 - mln::morpho, 365
- minus_cst
 - mln::arith, 181
- minus_cst_inplace
 - mln::arith, 182
- minus_infty
 - mln::point, 953
- minus_inplace
 - mln::arith, 182
- mirror
 - mln::border, 192
- mixed_neighb
 - mln::mixed_neighb, 834
- mln, 121
 - a_point_of, 150
 - apply_p2p, 150
 - before, 164
 - bin_1complex_image2d, 145
 - bin_2complex_image3df, 145
 - box1d, 145
 - box2d, 145
 - box2d_h, 146
 - box3d, 146
 - compose, 151
 - discrete_plane_1complex_geometry, 146
 - discrete_plane_2complex_geometry, 146
 - dpoint1d, 146
 - dpoint2d, 146
 - dpoint2d_h, 146
 - dpoint3d, 147
 - duplicate, 151
 - extend, 151, 152
 - float_2complex_image3df, 147
 - implies, 152
 - initialize, 152
 - int_u8_1complex_image2d, 147
 - int_u8_2complex_image2d, 147
 - int_u8_2complex_image3df, 147
 - larger_than, 152
 - make_debug_graph_image, 152
 - mln_exact, 153
 - mln_gen_complex_neighborhood, 153, 154
 - mln_gen_complex_window, 154
 - mln_gen_complex_window_p, 154, 155
 - mln_regular, 155
 - mln_trait_op_geq, 155
 - mln_trait_op_greater, 155
 - mln_trait_op_leq, 156
 - mln_trait_op_neq, 156
 - operator<, 158
 - operator<<, 159
 - operator<=, 159, 160
 - operator*, 157
 - operator++, 157
 - operator-, 157
 - operator--, 158
 - operator==, 160–162
 - p_run2d, 147
 - p_runs2d, 147
 - point1d, 148
 - point1df, 148
 - point2d, 148
 - point2d_h, 148
 - point2df, 148
 - point3d, 148
 - point3df, 148
 - primary, 163
 - ptransform, 163
 - rgb8_2complex_image3df, 148
 - sagittal_dec, 164
 - space_2complex_geometry, 149
 - unsigned_2complex_image3df, 149
 - up, 164
 - vec2d_d, 149
 - vec2d_f, 149
 - vec3d_d, 149
 - vec3d_f, 149
 - w_window1d_float, 149

- w_window1d_int, 149
- w_window2d_float, 150
- w_window2d_int, 150
- w_window3d_float, 150
- w_window3d_int, 150
- mln::accu, 164
 - compute, 166
 - line, 166
 - mln_meta_accu_result, 167
 - take, 167
- mln::accu::center, 437
 - init, 438
 - is_valid, 438
 - nsites, 438
 - take_as_init, 438
 - take_n_times, 438
 - to_result, 438
- mln::accu::convolve, 439
 - init, 439
 - is_valid, 439
 - take_as_init, 440
 - take_n_times, 440
 - to_result, 440
- mln::accu::count_adjacent_vertices, 440
 - init, 441
 - is_valid, 441
 - set_value, 441
 - take_as_init, 441
 - take_n_times, 441
 - to_result, 442
- mln::accu::count_labels, 442
 - init, 443
 - is_valid, 443
 - set_value, 443
 - take_as_init, 443
 - take_n_times, 443
 - to_result, 443
- mln::accu::count_value, 443
 - init, 444
 - is_valid, 444
 - set_value, 444
 - take_as_init, 444
 - take_n_times, 445
 - to_result, 445
- mln::accu::histo, 445
 - is_valid, 446
 - take, 446
 - take_as_init, 446
 - take_n_times, 446
 - vect, 446
- mln::accu::image, 168
- mln::accu::impl, 168
- mln::accu::label_used, 446
 - init, 447
 - is_valid, 447
 - take, 447
 - take_as_init, 447
 - take_n_times, 448
 - to_result, 448
- mln::accu::logic, 168
- mln::accu::logic::land, 448
 - init, 449
 - is_valid, 449
 - take_as_init, 449
 - take_n_times, 449
 - to_result, 449
- mln::accu::logic::land_basic, 449
 - can_stop, 450
 - init, 450
 - is_valid, 450
 - take_as_init, 450
 - take_n_times, 451
 - to_result, 451
- mln::accu::logic::lor, 451
 - init, 452
 - is_valid, 452
 - take_as_init, 452
 - take_n_times, 452
 - to_result, 452
- mln::accu::logic::lor_basic, 452
 - can_stop, 453
 - init, 453
 - is_valid, 453
 - take_as_init, 453
 - take_n_times, 453
 - to_result, 454
- mln::accu::maj_h, 454
 - init, 455
 - is_valid, 455
 - take_as_init, 455
 - take_n_times, 455
 - to_result, 455
- mln::accu::math, 168
- mln::accu::math::count, 455
 - init, 456
 - is_valid, 456
 - set_value, 456
 - take_as_init, 456
 - take_n_times, 456
 - to_result, 457
- mln::accu::math::inf, 457
 - init, 458
 - is_valid, 458
 - take_as_init, 458
 - take_n_times, 458
 - to_result, 458
- mln::accu::math::sum, 458
 - init, 459

- is_valid, 459
- take_as_init, 459
- take_n_times, 459
- to_result, 459
- mln::accu::math::sup, 460
 - init, 460
 - is_valid, 460
 - take_as_init, 461
 - take_n_times, 461
 - to_result, 461
- mln::accu::max_site, 461
 - init, 462
 - is_valid, 462
 - take_as_init, 462
 - take_n_times, 462
 - to_result, 462
- mln::accu::meta::center, 463
- mln::accu::meta::count_adjacent_vertices, 463
- mln::accu::meta::count_labels, 464
- mln::accu::meta::count_value, 465
- mln::accu::meta::histo, 466
- mln::accu::meta::label_used, 467
- mln::accu::meta::logic, 169
- mln::accu::meta::logic::land, 468
- mln::accu::meta::logic::lor, 469
- mln::accu::meta::logic::lor_basic, 470
- mln::accu::meta::maj_h, 471
- mln::accu::meta::math, 169
- mln::accu::meta::math::count, 472
- mln::accu::meta::math::inf, 473
- mln::accu::meta::math::sum, 474
- mln::accu::meta::math::sup, 475
- mln::accu::meta::max_site, 476
- mln::accu::meta::nil, 477
- mln::accu::meta::p, 478
- mln::accu::meta::pair, 479
- mln::accu::meta::rms, 480
- mln::accu::meta::shape, 170
- mln::accu::meta::shape::bbox, 481
- mln::accu::meta::shape::height, 482
- mln::accu::meta::shape::volume, 483
- mln::accu::meta::stat, 170
- mln::accu::meta::stat::max, 484
- mln::accu::meta::stat::max_h, 485
- mln::accu::meta::stat::mean, 486
- mln::accu::meta::stat::median_alt, 487
- mln::accu::meta::stat::median_h, 488
- mln::accu::meta::stat::min, 489
- mln::accu::meta::stat::min_h, 490
- mln::accu::meta::stat::rank, 491
- mln::accu::meta::stat::rank_high_quant, 492
- mln::accu::meta::tuple, 493
- mln::accu::meta::val, 494
- mln::accu::nil, 495
- init, 496
- is_valid, 496
- take_as_init, 496
- take_n_times, 496
- to_result, 496
- mln::accu::p, 497
 - init, 497
 - is_valid, 497
 - take_as_init, 498
 - take_n_times, 498
 - to_result, 498
- mln::accu::pair, 498
 - first, 500
 - first_accu, 500
 - init, 500
 - is_valid, 500
 - second, 500
 - second_accu, 500
 - take_as_init, 501
 - take_n_times, 501
 - to_result, 501
- mln::accu::rms, 501
 - init, 502
 - is_valid, 502
 - take_as_init, 502
 - take_n_times, 502
 - to_result, 502
- mln::accu::shape, 171
- mln::accu::shape::bbox, 503
 - init, 503
 - is_valid, 503
 - take_as_init, 503
 - take_n_times, 504
 - to_result, 504
- mln::accu::shape::height, 504
 - argument, 505
 - init, 505
 - is_valid, 505
 - set_value, 505
 - take_as_init, 506
 - take_n_times, 506
 - to_result, 506
 - value, 505
- mln::accu::shape::volume, 506
 - argument, 507
 - init, 507
 - is_valid, 507
 - set_value, 508
 - take_as_init, 508
 - take_n_times, 508
 - to_result, 508
 - value, 507
- mln::accu::site_set::rectangularity, 508
 - area, 509

- bbox, 509
- rectangularity, 509
- take_as_init, 509
- take_n_times, 510
- to_result, 510
- mln::accu::stat, 172
 - operator==, 173
- mln::accu::stat::deviation, 510
 - init, 511
 - is_valid, 511
 - take_as_init, 511
 - take_n_times, 511
 - to_result, 511
- mln::accu::stat::histo3d_rgb, 512
 - histo3d_rgb, 513
 - init, 513
 - is_valid, 513
 - take, 513
 - take_as_init, 514
 - take_n_times, 514
 - to_result, 514
- mln::accu::stat::max, 514
 - init, 515
 - is_valid, 515
 - set_value, 515
 - take_as_init, 515
 - take_n_times, 515
 - to_result, 515
- mln::accu::stat::max_h, 516
 - init, 516
 - is_valid, 516
 - take_as_init, 517
 - take_n_times, 517
 - to_result, 517
- mln::accu::stat::mean, 517
 - count, 518
 - init, 518
 - is_valid, 518
 - sum, 518
 - take_as_init, 518
 - take_n_times, 519
 - to_result, 519
- mln::accu::stat::median_alt, 519
 - is_valid, 520
 - take, 520
 - take_as_init, 520
 - take_n_times, 520
 - to_result, 520
- mln::accu::stat::median_h, 521
 - init, 522
 - is_valid, 522
 - take_as_init, 522
 - take_n_times, 522
 - to_result, 522
- mln::accu::stat::meta::deviation, 522
- mln::accu::stat::min, 523
 - init, 524
 - is_valid, 524
 - set_value, 524
 - take_as_init, 524
 - take_n_times, 524
 - to_result, 525
- mln::accu::stat::min_h, 525
 - init, 526
 - is_valid, 526
 - take_as_init, 526
 - take_n_times, 526
 - to_result, 526
- mln::accu::stat::min_max, 526
 - first, 528
 - first_accu, 528
 - init, 528
 - is_valid, 528
 - second, 528
 - second_accu, 528
 - take_as_init, 528
 - take_n_times, 529
 - to_result, 529
- mln::accu::stat::rank, 529
 - init, 530
 - is_valid, 530
 - k, 530
 - take_as_init, 530
 - take_n_times, 530
 - to_result, 530
- mln::accu::stat::rank< bool >, 531
 - init, 531
 - is_valid, 531
 - take_as_init, 532
 - take_n_times, 532
 - to_result, 532
- mln::accu::stat::rank_high_quant, 532
 - init, 533
 - is_valid, 533
 - take_as_init, 533
 - take_n_times, 533
 - to_result, 533
- mln::accu::stat::var, 534
 - init, 535
 - is_valid, 535
 - mean, 535
 - mean_t, 535
 - n_items, 535
 - take_as_init, 535
 - take_n_times, 535
 - to_result, 535
 - variance, 536
- mln::accu::stat::variance, 536

- init, 537
- is_valid, 537
- mean, 537
- n_items, 537
- standard_deviation, 537
- sum, 538
- take_as_init, 538
- take_n_times, 538
- to_result, 538
- var, 538
- mln::accu::tuple, 538
 - init, 539
 - is_valid, 539
 - take_as_init, 539
 - take_n_times, 540
 - to_result, 540
- mln::accu::val, 540
 - init, 541
 - is_valid, 541
 - take_as_init, 541
 - take_n_times, 541
 - to_result, 541
- mln::Accumulator, 541
 - take_as_init, 543
 - take_n_times, 543
- mln::algebra, 173
 - ldlt_decomp, 174
 - ldlt_solve, 174
 - operator*, 174
 - vprod, 175
- mln::algebra::h_mat, 543
 - _1, 545
 - h_mat, 544
 - t, 545
- mln::algebra::h_vec, 545
 - h_vec, 546
 - operator mat< n, 1, U >, 546
 - origin, 547
 - t, 546
 - to_vec, 547
 - zero, 547
- mln::arith, 175
 - diff_abs, 178
 - div, 178
 - div_cst, 178
 - div_inplace, 179
 - min, 179
 - min_inplace, 179
 - minus, 180, 181
 - minus_cst, 181
 - minus_cst_inplace, 182
 - minus_inplace, 182
 - plus, 183
 - plus_cst, 184, 185
 - plus_cst_inplace, 185
 - plus_inplace, 185
 - revert, 186
 - revert_inplace, 186
 - times, 187
 - times_cst, 187
 - times_inplace, 187
- mln::arith::impl, 188
- mln::arith::impl::generic, 188
- mln::binarization, 189
 - binarization, 189
 - threshold, 189
- mln::bkd_pixter1d, 547
 - bkd_pixter1d, 548
 - image, 548
 - next, 548
- mln::bkd_pixter2d, 548
 - bkd_pixter2d, 549
 - image, 549
 - next, 549
- mln::bkd_pixter3d, 550
 - bkd_pixter3d, 551
 - image, 550
 - next, 551
- mln::border, 190
 - adjust, 190
 - duplicate, 191
 - equalize, 191
 - fill, 191
 - find, 192
 - get, 192
 - mirror, 192
 - resize, 193
- mln::border::impl, 193
- mln::border::impl::generic, 194
- mln::Box, 559
 - bbox, 562
 - diff, 563
 - inter, 563
 - is_empty, 562
 - len, 562
 - nsites, 562
 - operator<, 563
 - operator<<, 563
 - operator<=, 563, 564
 - operator==, 564
 - sym_diff, 564
 - uni, 564
 - unique, 564
- mln::box, 551
 - bbox, 556
 - bkd_piter, 554
 - box, 555
 - crop_wrt, 556

- element, 554
- enlarge, 556
- fwd_piter, 554
- has, 556
- is_empty, 557
- is_valid, 557
- len, 557
- memory_size, 557
- merge, 557
- nsites, 557
- operator<<, 559
- pcenter, 558
- piter, 555
- pmax, 558
- pmin, 558
- psite, 555
- site, 555
- to_larger, 558
- mln::box_runend_piter, 565
 - box_runend_piter, 565
 - next, 566
 - run_length, 566
- mln::box_runstart_piter, 566
 - box_runstart_piter, 567
 - next, 567
 - run_length, 567
- mln::Browsing, 567
- mln::canvas, 194
 - distance_front, 195
 - distance_geodesic, 195
- mln::canvas::browsing, 195
- mln::canvas::browsing::backdiagonal2d_t, 568
- mln::canvas::browsing::breadth_first_search_t, 570
- mln::canvas::browsing::depth_first_search_t, 570
- mln::canvas::browsing::diagonal2d_t, 570
- mln::canvas::browsing::dir_struct_elt_incr_update_t, 572
- mln::canvas::browsing::directional_t, 573
- mln::canvas::browsing::fwd_t, 575
- mln::canvas::browsing::hyper_directional_t, 576
- mln::canvas::browsing::snake_fwd_t, 578
- mln::canvas::browsing::snake_generic_t, 579
- mln::canvas::browsing::snake_vert_t, 581
- mln::canvas::chamfer, 582
- mln::canvas::impl, 196
- mln::canvas::labeling, 196
 - blobs, 197
- mln::canvas::labeling::impl, 197
- mln::canvas::morpho, 197
- mln::category< R(*) (A) >, 582
- mln::complex_image, 583
 - complex_image, 585
 - dim, 585
 - domain, 585
 - geom, 584
 - lvalue, 584
 - operator(), 585
 - rvalue, 584
 - skeleton, 584
 - value, 584
 - values, 585
- mln::complex_neighborhood_bkd_piter, 586
 - complex_neighborhood_bkd_piter, 587
 - iter, 587
 - iter_type, 586
 - next, 587
 - psite, 586
- mln::complex_neighborhood_fwd_piter, 587
 - complex_neighborhood_fwd_piter, 588
 - iter, 589
 - iter_type, 588
 - next, 589
 - psite, 588
- mln::complex_psite, 589
 - change_target, 590
 - complex_psite, 590
 - face, 591
 - face_id, 591
 - invalidate, 591
 - is_valid, 591
 - n, 591
 - site_set, 591
- mln::complex_window_bkd_piter, 592
 - complex_window_bkd_piter, 593
 - iter, 593
 - iter_type, 593
 - next, 593
 - psite, 593
- mln::complex_window_fwd_piter, 594
 - complex_window_fwd_piter, 595
 - iter, 595
 - iter_type, 594
 - next, 595
 - psite, 594
- mln::convert, 198
 - from_to, 200
 - mln_image_from_grid, 200, 201
 - mln_window, 201
 - to, 201
 - to_dpoint, 201
 - to_fun, 201, 204
 - to_image, 201
 - to_p_array, 202
 - to_p_set, 202, 203
 - to_qimage, 203
 - to_upper_window, 203
 - to_window, 203, 204
- mln::data, 204

- abs, 206
- abs_inplace, 206
- apply, 207
- compute, 207
- convert, 208
- fast_median, 208
- fill, 208
- fill_with_image, 209
- fill_with_value, 209
- median, 210
- mln_meta_accu_result, 210
- paste, 210
- paste_without_localization, 211
- replace, 211
- saturate, 211
- saturate_inplace, 212
- sort_offsets_increasing, 212
- sort_psites_decreasing, 212
- sort_psites_increasing, 212
- stretch, 213
- to_enc, 213
- transform, 213, 214
- transform_inplace, 214
- update, 215
- wrap, 215
- mln::data::approx, 216
 - median, 216, 217
- mln::data::approx::impl, 217
- mln::data::impl, 217
 - paste_without_localization_fast, 218
 - paste_without_localization_fastest, 218
 - paste_without_localization_lines, 219
 - stretch, 219
 - transform_inplace_lowq, 220
 - update_fastest, 220
- mln::data::impl::generic, 220
 - fill_with_image, 221
 - fill_with_value, 221
 - paste, 222
 - transform, 222
 - transform_inplace, 223
 - update, 223
- mln::data::naive, 223
 - median, 224
- mln::data::naive::impl, 224
- mln::debug, 225
 - draw_graph, 226, 227
 - filename, 227
 - format, 227, 228
 - iota, 228
 - mosaic, 228
 - println, 228, 229
 - println_with_border, 229
 - put_word, 229
 - slices_2d, 229
 - superpose, 229, 230
 - z_order, 230
- mln::debug::impl, 230
- mln::decorated_image, 595
 - ~decorated_image, 597
 - decorated_image, 597
 - decoration, 597
 - lvalue, 596
 - operator decorated_image< const I, D >, 598
 - operator(), 598
 - psite, 597
 - rvalue, 597
 - skeleton, 597
- mln::def, 231
 - coord, 231
 - coordf, 231
- mln::Delta_Point_Site, 598
- mln::Delta_Point_Site< void >, 599
- mln::display, 232
- mln::display::impl, 232
- mln::display::impl::generic, 232
- mln::doc, 232
- mln::doc::Accumulator, 599
 - argument, 600
 - init, 600
 - take, 600
- mln::doc::Box, 600
 - bbox, 602
 - bkd_piter, 602
 - fwd_piter, 602
 - has, 603
 - nsites, 603
 - pmax, 603
 - pmin, 603
 - psite, 602
 - site, 602
- mln::doc::Dpoint, 603
 - coord, 604
 - dim, 605
 - dpoint, 605
 - point, 605
- mln::doc::Fastest_Image, 606
 - bbox, 610
 - bkd_piter, 608
 - border, 610
 - buffer, 610
 - coord, 608
 - delta_index, 611
 - domain, 611
 - dpoint, 608
 - fwd_piter, 608
 - has, 611
 - is_valid, 611

- lvalue, 609
- nelements, 612
- nsites, 612
- operator(), 612
- point, 609
- point_at_index, 613
- pset, 609
- psite, 609
- rvalue, 609
- skeleton, 609
- value, 610
- values, 613
- vset, 610
- mln::doc::Generalized_Pixel, 614
 - ima, 615
 - image, 615
 - rvalue, 615
 - val, 615
 - value, 615
- mln::doc::Image, 615
 - bbox, 620
 - bkd_piter, 618
 - coord, 618
 - domain, 620
 - dpoint, 618
 - fwd_piter, 618
 - has, 620
 - is_valid, 620
 - lvalue, 618
 - nsites, 620
 - operator(), 621
 - point, 618
 - pset, 618
 - psite, 619
 - rvalue, 619
 - skeleton, 619
 - value, 619
 - values, 621
 - vset, 619
- mln::doc::Iterator, 621
 - invalidate, 622
 - is_valid, 622
 - start, 622
- mln::doc::Neighborhood, 623
 - bkd_niter, 624
 - dpoint, 624
 - fwd_niter, 624
 - niter, 624
 - point, 624
- mln::doc::Object, 624
- mln::doc::Pixel_Iterator, 625
 - ima, 627
 - image, 627
 - invalidate, 628
 - is_valid, 628
 - lvalue, 627
 - rvalue, 627
 - start, 628
 - val, 628
 - value, 627
- mln::doc::Point_Site
 - dim, 630
- mln::doc::Point_Site, 628
 - coord, 629
 - dpoint, 629
 - mesh, 629
 - point, 629
 - to_point, 630
- mln::doc::Site_Iterator, 631
 - invalidate, 632
 - is_valid, 632
 - operator psite, 632
 - psite, 632
 - start, 632
- mln::doc::Site_Set, 633
 - bkd_piter, 634
 - fwd_piter, 634
 - has, 634
 - psite, 634
 - site, 634
- mln::doc::Value_Iterator, 635
 - invalidate, 636
 - is_valid, 636
 - operator value, 636
 - start, 636
 - value, 636
- mln::doc::Value_Set, 636
 - bkd_viter, 638
 - fwd_viter, 638
 - has, 638
 - index_of, 638
 - nvalues, 638
 - value, 638
- mln::doc::Weighted_Window, 639
 - bkd_qiter, 640
 - delta, 641
 - dpoint, 640
 - fwd_qiter, 640
 - is_centered, 641
 - is_empty, 641
 - point, 640
 - sym, 641
 - weight, 641
 - win, 641
 - window, 641
- mln::doc::Window, 641
 - bkd_qiter, 642
 - fwd_qiter, 642

- qiter, 643
- mln::Dpoint, 643
 - to_dpoint, 644
- mln::dpoint, 644
 - coord, 646
 - dim, 647
 - dpoint, 647
 - grid, 646
 - operator mln::algebra::vec< dpoint< G, C >::dim, Q >, 648
 - psite, 646
 - set_all, 648
 - site, 646
 - to_vec, 648
 - vec, 646
- mln::dpoints_bkd_pixter, 649
 - center_val, 650
 - dpoints_bkd_pixter, 650
 - invalidate, 650
 - is_valid, 650
 - next, 651
 - start, 651
 - update, 651
- mln::dpoints_fwd_pixter, 651
 - center_val, 653
 - dpoints_fwd_pixter, 652
 - invalidate, 653
 - is_valid, 653
 - next, 653
 - start, 653
 - update, 654
- mln::dpsites_bkd_piter, 654
 - dpsites_bkd_piter, 655
 - next, 655
- mln::dpsites_fwd_piter, 655
 - dpsites_fwd_piter, 656
 - next, 656
- mln::draw, 234
 - box, 234
 - box_plain, 235
 - dashed_line, 235
 - line, 236
 - plot, 236
 - polygon, 236
 - site_set, 237
- mln::Edge, 657
- mln::edge_image, 657
 - edge_image, 659
 - edge_nbh_t, 658
 - edge_win_t, 658
 - graph_t, 658
 - nbh_t, 658
 - operator(), 659
 - site_function_t, 658
 - skeleton, 659
 - win_t, 659
- mln::estim, 237
 - mean, 238
 - min_max, 238
 - sum, 239
- mln::extended, 659
 - domain, 661
 - extended, 660
 - skeleton, 660
 - value, 660
- mln::extension, 239
 - adjust, 240, 241
 - adjust_duplicate, 241
 - adjust_fill, 241
 - duplicate, 241
 - fill, 241
- mln::extension_fun, 661
 - extension, 663
 - extension_fun, 662
 - has, 663
 - operator(), 663
 - rvalue, 662
 - skeleton, 662
 - value, 662
- mln::extension_ima, 663
 - extension, 665
 - extension_ima, 665
 - has, 665
 - operator(), 665
 - rvalue, 664
 - skeleton, 664
 - value, 665
- mln::extension_val, 666
 - change_extension, 667
 - extension, 667
 - extension_val, 667
 - has, 668
 - operator(), 668
 - rvalue, 667
 - skeleton, 667
 - value, 667
- mln::faces_psite, 668
 - change_target, 670
 - face, 670
 - face_id, 670
 - faces_psite, 669
 - invalidate, 670
 - is_valid, 670
 - n, 670
 - site_set, 670
- mln::flat_image, 671
 - domain, 673
 - flat_image, 672

- has, 673
- lvalue, 672
- operator(), 673
- rvalue, 672
- skeleton, 672
- value, 672
- mln::fun, 242
- mln::fun::access, 243
- mln::fun::from_accu, 673
- mln::fun::i2v, 244
 - operator<<, 244
- mln::fun::n2v, 244
- mln::fun::n2v::white_gaussian, 674
- mln::fun::p2b, 244
- mln::fun::p2b::antilog, 674
- mln::fun::p2b::tautology, 675
- mln::fun::p2p, 245
- mln::fun::p2v, 245
- mln::fun::stat, 245
- mln::fun::v2b, 245
- mln::fun::v2b::lnot, 677
- mln::fun::v2b::threshold, 678
- mln::fun::v2i, 246
- mln::fun::v2v, 246
 - f_hsi_to_rgb_3x8, 247
 - f_hsl_to_rgb_3x8, 247
 - f_rgb_to_hsi_f, 247
 - f_rgb_to_hsl_f, 247
- mln::fun::v2v::ch_function_value, 679
- mln::fun::v2v::component, 680
- mln::fun::v2v::l1_norm, 680
- mln::fun::v2v::l2_norm, 681
- mln::fun::v2v::linear, 682
- mln::fun::v2v::linfty_norm, 683
- mln::fun::v2v::rgb8_to_rgnb, 684
 - operator(), 686
- mln::fun::v2w2v, 248
- mln::fun::v2w2v::cos, 686
- mln::fun::v2w_w2v, 248
- mln::fun::v2w_w2v::l1_norm, 687
- mln::fun::v2w_w2v::l2_norm, 689
- mln::fun::v2w_w2v::linfty_norm, 690
- mln::fun::vv2b, 248
- mln::fun::vv2b::eq, 691
- mln::fun::vv2b::ge, 691
- mln::fun::vv2b::gt, 692
- mln::fun::vv2b::implies, 693
- mln::fun::vv2b::le, 694
- mln::fun::vv2b::lt, 695
- mln::fun::vv2v, 249
- mln::fun::vv2v::diff_abs, 696
- mln::fun::vv2v::land, 697
- mln::fun::vv2v::land_not, 698
- mln::fun::vv2v::lor, 699
- mln::fun::vv2v::lxor, 700
- mln::fun::vv2v::max, 701
- mln::fun::vv2v::min, 702
- mln::fun::vv2v::vec, 703
- mln::fun::x2p, 250
- mln::fun::x2p::closest_point, 704
- mln::fun::x2v, 250
- mln::fun::x2v::bilinear, 705
 - operator(), 705
- mln::fun::x2v::trilinear, 706
- mln::fun::x2x, 250
- mln::fun::x2x::composed, 706
 - composed, 707
- mln::fun::x2x::linear, 707
 - ima, 708
 - linear, 708
 - operator(), 708
- mln::fun::x2x::rotation, 708
 - data_t, 710
 - inv, 711
 - invert, 710
 - operator(), 711
 - rotation, 710, 711
 - set_alpha, 711
 - set_axis, 711
- mln::fun::x2x::translation, 712
 - data_t, 713
 - inv, 714
 - invert, 713
 - operator(), 714
 - set_t, 714
 - t, 714
 - translation, 713
- mln::fun_image, 714
 - fun_image, 716
 - lvalue, 715
 - operator(), 716
 - rvalue, 715
 - skeleton, 715
 - value, 716
- mln::Function, 717
 - Function, 717
- mln::Function< void >, 717
- mln::Function_n2v, 718
- mln::Function_v2b, 718
- mln::Function_v2v, 719
- mln::Function_vv2b, 720
- mln::Function_vv2v, 720
- mln::fwd_pixter1d, 721
 - fwd_pixter1d, 722
 - image, 722
 - next, 722
- mln::fwd_pixter2d, 722
 - fwd_pixter2d, 723

- image, 723
- next, 724
- mln::fwd_pixter3d, 724
 - fwd_pixter3d, 725
 - image, 724
 - next, 725
- mln::Gdpoint, 725
- mln::Gdpoint< void >, 726
- mln::Generalized_Pixel, 727
- mln::geom, 251
 - bbox, 255, 256
 - chamfer, 256
 - delta, 256
 - horizontal_symmetry, 256
 - max_col, 256, 257
 - max_ind, 257
 - max_row, 257
 - max_sli, 257
 - mesh_corner_point_area, 257
 - mesh_curvature, 258
 - mesh_normal, 258
 - min_col, 259
 - min_ind, 259
 - min_row, 259
 - min_sli, 259
 - ncols, 260
 - ninds, 260
 - nrows, 260
 - nsites, 260
 - nslis, 261
 - pmin_pmax, 261
 - rotate, 261, 262
 - seeds2tiling, 263
 - translate, 263, 264
 - vertical_symmetry, 264
- mln::geom::complex_geometry, 727
 - add_location, 728
 - complex_geometry, 728
 - operator(), 728
- mln::geom::impl, 264
 - seeds2tiling, 265
- mln::Gpoint, 729
 - operator<<, 732
 - operator+, 730
 - operator+=", 731
 - operator-, 731
 - operator-=, 732
 - operator/, 732
 - operator==, 732
- mln::Graph, 733
- mln::graph, 265
 - compute, 266
 - labeling, 266
 - to_neighb, 266
 - to_win, 267
- mln::graph::attribute::card_t, 734
 - result, 734
- mln::graph::attribute::representative_t, 734
 - result, 735
- mln::graph_elt_mixed_neighborhood, 735
 - bkd_niter, 736
 - fwd_niter, 736
 - niter, 736
- mln::graph_elt_mixed_window, 736
 - bkd_qiter, 738
 - center_t, 738
 - delta, 739
 - fwd_qiter, 738
 - graph_element, 739
 - is_centered, 739
 - is_empty, 740
 - is_symmetric, 740
 - is_valid, 740
 - psite, 739
 - qiter, 739
 - site, 739
 - sym, 740
 - target, 739
- mln::graph_elt_neighborhood, 740
 - bkd_niter, 742
 - fwd_niter, 742
 - niter, 742
- mln::graph_elt_neighborhood_if, 742
 - bkd_niter, 743
 - fwd_niter, 743
 - graph_elt_neighborhood_if, 744
 - mask, 744
 - niter, 743
- mln::graph_elt_window, 744
 - bkd_qiter, 746
 - center_t, 746
 - delta, 747
 - fwd_qiter, 746
 - graph_element, 747
 - is_centered, 747
 - is_empty, 748
 - is_symmetric, 748
 - is_valid, 748
 - psite, 747
 - qiter, 747
 - site, 747
 - sym, 748
 - target, 747
- mln::graph_elt_window_if, 748
 - bkd_qiter, 750
 - change_mask, 752
 - delta, 752
 - fwd_qiter, 750

- graph_elt_window_if, [752](#)
- is_centered, [752](#)
- is_empty, [752](#)
- is_symmetric, [752](#)
- is_valid, [753](#)
- mask, [753](#)
- mask_t, [751](#)
- psite, [751](#)
- qiter, [751](#)
- site, [751](#)
- sym, [753](#)
- target, [751](#)
- mln::graph_window_base, [753](#)
 - delta, [754](#)
 - is_centered, [754](#)
 - is_empty, [754](#)
 - is_symmetric, [755](#)
 - is_valid, [755](#)
 - site, [754](#)
 - sym, [755](#)
- mln::graph_window_if_piter, [755](#)
 - element, [756](#)
 - graph_window_if_piter, [756](#)
 - id, [756](#)
 - next, [757](#)
 - P, [756](#)
- mln::graph_window_piter, [757](#)
 - center_t, [758](#)
 - change_target_site_set, [759](#)
 - element, [760](#)
 - graph_element, [758](#)
 - graph_window_piter, [759](#)
 - id, [760](#)
 - next, [760](#)
 - P, [758](#)
 - target_site_set, [760](#)
- mln::grid, [267](#)
- mln::hexa, [760](#)
 - bkd_piter, [762](#)
 - domain, [763](#)
 - fwd_piter, [762](#)
 - has, [763](#)
 - hexa, [763](#)
 - lvalue, [762](#)
 - operator(), [764](#)
 - psite, [762](#)
 - rvalue, [763](#)
 - skeleton, [763](#)
 - value, [763](#)
- mln::histo, [267](#)
 - compute, [268](#)
 - equalize, [268](#)
- mln::histo::array, [764](#)
- mln::histo::impl, [269](#)
 - mln::histo::impl::generic, [269](#)
 - mln::Image, [764](#)
 - mln::image1d, [766](#)
 - bbox, [769](#)
 - border, [769](#)
 - buffer, [769](#)
 - delta_index, [769](#)
 - domain, [769](#)
 - element, [769](#)
 - has, [770](#)
 - image1d, [768](#)
 - lvalue, [768](#)
 - nelements, [770](#)
 - ninds, [770](#)
 - operator(), [770](#)
 - point_at_index, [770](#)
 - rvalue, [768](#)
 - skeleton, [768](#)
 - value, [768](#)
 - vbbox, [771](#)
 - mln::image2d, [771](#)
 - bbox, [774](#)
 - border, [774](#)
 - buffer, [774](#)
 - delta_index, [774](#)
 - domain, [774](#)
 - element, [774](#)
 - has, [775](#)
 - image2d, [773](#)
 - lvalue, [773](#)
 - ncols, [775](#)
 - nelements, [775](#)
 - nrows, [775](#)
 - operator(), [775](#)
 - point_at_index, [775](#)
 - rvalue, [773](#)
 - skeleton, [773](#)
 - value, [773](#)
 - mln::image2d_h, [776](#)
 - bkd_piter, [777](#)
 - domain, [778](#)
 - fwd_piter, [777](#)
 - has, [778](#)
 - image2d_h, [778](#)
 - lvalue, [777](#)
 - operator(), [779](#)
 - psite, [778](#)
 - rvalue, [778](#)
 - skeleton, [778](#)
 - value, [778](#)
 - mln::image3d, [779](#)
 - bbox, [782](#)
 - border, [782](#)
 - buffer, [782](#)

- delta_index, 782
- domain, 783
- element, 783
- has, 783
- image3d, 782
- lvalue, 781
- ncols, 783
- nelements, 783
- nrows, 784
- nslls, 784
- operator(), 784
- point_at_index, 784
- rvalue, 781
- skeleton, 781
- value, 781
- vbbox, 784
- mln::image_if, 784
 - domain, 786
 - image_if, 785
 - operator image_if< const I, F >, 786
 - skeleton, 785
- mln::impl, 269
- mln::interpolated, 786
 - has, 788
 - interpolated, 788
 - is_valid, 788
 - lvalue, 787
 - psite, 787
 - rvalue, 787
 - skeleton, 787
 - value, 787
- mln::io, 269
- mln::io::cloud, 271
 - load, 271
 - save, 271
- mln::io::dicom, 272
 - get_header, 272
 - load, 272
- mln::io::dicom::dicom_header, 788
- mln::io::dump, 273
 - get_header, 273
 - load, 273
 - save, 274
- mln::io::dump::dump_header, 788
- mln::io::fits, 274
 - load, 274, 275
- mln::io::fld, 275
 - load, 276
 - read_header, 276
 - write_header, 276
- mln::io::fld::fld_header, 789
- mln::io::magick, 276
 - load, 277
 - save, 277
- mln::io::off, 278
 - load, 278
 - save, 279
 - save_bin_alt, 279
- mln::io::pbm, 279
 - load, 280
 - save, 280
- mln::io::pbm::impl, 281
- mln::io::pbms, 281
 - load, 281
- mln::io::pbms::impl, 282
- mln::io::pfm, 282
 - load, 283
 - save, 283
- mln::io::pfm::impl, 283
- mln::io::pgm, 284
 - load, 284
 - save, 284
- mln::io::pgms, 285
 - load, 285
- mln::io::plot, 285
 - load, 286
 - save, 286, 287
- mln::io::pnm, 287
 - load, 288
 - load_ascii_builtin, 288
 - load_ascii_value, 288
 - load_raw_2d, 288
 - max_component, 289
 - save, 289
- mln::io::pnm::impl, 289
- mln::io::pnms, 289
 - load, 290
- mln::io::ppm, 290
 - load, 291
 - save, 291
- mln::io::ppms, 292
 - load, 292
- mln::io::raw, 292
 - get_header, 293
 - load, 293
 - save, 293
- mln::io::raw::raw_header, 789
- mln::io::tiff, 294
 - load, 294
- mln::io::txt, 294
 - save, 295
- mln::Iterator, 789
 - next, 791
- mln::labeled_image, 791
 - bbox, 794
 - bbox_t, 793
 - bboxes, 794
 - labeled_image, 793

- nlabels, 794
- relabel, 794
- skeleton, 793
- subdomain, 794
- update_data, 794
- mln::labeled_image_base, 795
 - bbox, 797
 - bbox_t, 796
 - bboxes, 797
 - labeled_image_base, 796
 - nlabels, 797
 - relabel, 797
 - subdomain, 797
 - update_data, 798
- mln::labeling, 295
 - background, 298
 - blobs, 299
 - blobs_and_compute, 299
 - colorize, 299, 300
 - compute, 300–302
 - compute_image, 302, 303
 - fill_holes, 304
 - flat_zones, 304
 - foreground, 304
 - pack, 305
 - pack_inplace, 306
 - regional_maxima, 306
 - regional_minima, 306
 - relabel, 307
 - relabel_inplace, 308
 - superpose, 308
 - value, 309
 - value_and_compute, 309
 - wrap, 310
- mln::labeling::impl, 310
 - compute_fastest, 311
- mln::labeling::impl::generic, 312
 - compute, 313, 314
- mln::lazy_image, 798
 - domain, 800
 - has, 800
 - lazy_image, 800
 - lvalue, 799
 - operator(), 800
 - rvalue, 799
 - skeleton, 799
- mln::linear, 314
 - gaussian, 315
 - gaussian_1st_derivative, 316
 - gaussian_2nd_derivative, 316, 317
 - mln_ch_convolve, 317, 318
 - mln_ch_convolve_grad, 318
- mln::linear::impl, 318
- mln::linear::local, 318
 - convolve, 319
- mln::linear::local::impl, 320
- mln::Literal, 801
- mln::literal, 320
 - black, 323
 - blue, 323
 - brown, 323
 - cyan, 323
 - dark_gray, 323
 - green, 323
 - identity, 324
 - light_gray, 324
 - lime, 324
 - magenta, 324
 - max, 324
 - medium_gray, 324
 - min, 324
 - olive, 324
 - one, 325
 - orange, 325
 - origin, 325
 - pink, 325
 - purple, 325
 - red, 325
 - teal, 325
 - violet, 325
 - white, 326
 - yellow, 326
 - zero, 326
- mln::literal::black_t, 803
- mln::literal::blue_t, 804
- mln::literal::brown_t, 804
- mln::literal::cyan_t, 805
- mln::literal::green_t, 806
- mln::literal::identity_t, 807
- mln::literal::light_gray_t, 808
- mln::literal::lime_t, 809
- mln::literal::magenta_t, 810
- mln::literal::max_t, 811
- mln::literal::min_t, 812
- mln::literal::olive_t, 813
- mln::literal::one_t, 814
- mln::literal::orange_t, 815
- mln::literal::origin_t, 816
- mln::literal::pink_t, 817
- mln::literal::purple_t, 818
- mln::literal::red_t, 819
- mln::literal::teal_t, 820
- mln::literal::violet_t, 821
- mln::literal::white_t, 822
- mln::literal::yellow_t, 823
- mln::literal::zero_t, 824
- mln::logical, 326
 - and_inplace, 327

- and_not, 327
- and_not_inplace, 328
- not_inplace, 328
- or_inplace, 328
- xor_inplace, 329
- mln::logical::impl, 329
- mln::logical::impl::generic, 330
- mln::make, 330
 - attachment, 335
 - box1d, 335, 336
 - box2d, 336
 - box2d_h, 337
 - box3d, 338
 - cell, 339
 - couple, 339
 - detachment, 339
 - dpoint2d_h, 340
 - dummy_p_edges, 340
 - dummy_p_vertices, 341
 - edge_image, 341–343
 - h_mat, 343
 - image, 343, 344
 - image2d, 344
 - image3d, 344, 345
 - influence_zone_adjacency_graph, 345
 - mat, 345
 - ord_pair, 345
 - p_edges_with_mass_centers, 346
 - p_vertices_with_mass_centers, 346
 - pix, 346
 - pixel, 347
 - point2d_h, 347
 - rag_and_labeled_wsl, 347
 - region_adjacency_graph, 348
 - relabelfun, 348, 349
 - vec, 349, 350
 - vertex_image, 350, 351
 - voronoi, 351
 - w_window, 351
 - w_window1d, 352
 - w_window1d_int, 352
 - w_window2d, 353
 - w_window2d_int, 353
 - w_window3d, 353
 - w_window3d_int, 354
 - w_window_directional, 354
- mln::math, 355
 - abs, 355
- mln::Mesh, 825
- mln::Meta_Accumulator, 826
- mln::Meta_Function, 828
- mln::Meta_Function_v2v, 829
- mln::Meta_Function_vv2v, 829
- mln::metal, 356
 - mln::metal::ands, 830
 - mln::metal::converts_to, 830
 - mln::metal::equal, 830
 - mln::metal::goes_to, 831
 - mln::metal::impl, 356
 - mln::metal::is, 831
 - mln::metal::is_a, 832
 - mln::metal::is_not, 832
 - mln::metal::is_not_a, 832
 - mln::metal::math, 357
 - mln::metal::math::impl, 357
 - mln::mixed_neighb, 832
 - bkd_niter, 833
 - fwd_niter, 833
 - mixed_neighb, 834
 - niter, 833
 - mln::morpho, 357
 - complementation, 360
 - complementation_inplace, 360
 - contrast, 361
 - dilation, 361
 - erosion, 361
 - general, 361
 - gradient, 361
 - gradient_external, 362
 - gradient_internal, 362
 - hit_or_miss, 362
 - hit_or_miss_background_closing, 362
 - hit_or_miss_background_opening, 362
 - hit_or_miss_closing, 363
 - hit_or_miss_opening, 363
 - laplacian, 363
 - line_gradient, 363
 - meyer_wst, 364
 - min, 364
 - min_inplace, 364
 - minus, 365
 - plus, 365
 - rank_filter, 365
 - thick_miss, 365
 - thickening, 365
 - thin_fit, 366
 - thinning, 366
 - top_hat_black, 366
 - top_hat_self_complementary, 366
 - top_hat_white, 367
 - mln::morpho::approx, 367
 - mln::morpho::attribute, 367
 - mln::morpho::attribute::card, 834
 - init, 835
 - is_valid, 835
 - take_as_init, 835
 - take_n_times, 835
 - to_result, 835

- mln::morpho::attribute::count_adjacent_vertices, 835
 - init, 836
 - is_valid, 836
 - take_as_init, 836
 - take_n_times, 836
 - to_result, 836
- mln::morpho::attribute::height, 837
 - base_level, 838
 - init, 838
 - is_valid, 838
 - take_as_init, 838
 - take_n_times, 838
 - to_result, 838
- mln::morpho::attribute::sharpness, 838
 - area, 839
 - height, 839
 - init, 840
 - is_valid, 840
 - take_as_init, 840
 - take_n_times, 840
 - to_result, 840
 - volume, 840
- mln::morpho::attribute::sum, 841
 - init, 841
 - is_valid, 841
 - set_value, 842
 - take_as_init, 842
 - take_n_times, 842
 - to_result, 842
 - untake, 842
- mln::morpho::attribute::volume, 842
 - area, 843
 - init, 843
 - is_valid, 843
 - take_as_init, 844
 - take_n_times, 844
 - to_result, 844
- mln::morpho::closing::approx, 368
 - structural, 368
- mln::morpho::elementary, 368
 - closing, 369
 - mln_trait_op_minus_twice, 369
 - opening, 369
 - top_hat_black, 370
 - top_hat_self_complementary, 370
 - top_hat_white, 370
- mln::morpho::impl, 370
- mln::morpho::impl::generic, 371
- mln::morpho::opening::approx, 371
 - structural, 371
- mln::morpho::reconstruction, 372
- mln::morpho::reconstruction::by_dilation, 372
- mln::morpho::reconstruction::by_erosion, 372
- mln::morpho::tree, 372
 - compute_attribute_image, 374
 - compute_attribute_image_from, 374
 - compute_parent, 375
 - dual_input_max_tree, 376
 - max_tree, 376
 - min_tree, 377
 - propagate_if, 377
 - propagate_if_value, 378
 - propagate_node_to_ancestors, 378, 379
 - propagate_node_to_descendants, 379
 - propagate_representative, 379
- mln::morpho::tree::filter, 380
 - direct, 381
 - filter, 381
 - max, 381
 - min, 381
 - subtractive, 382
- mln::morpho::watershed, 382
 - flooding, 383
 - superpose, 384
 - topological, 384
- mln::morpho::watershed::watershed, 385
- mln::morpho::watershed::watershed::generic, 385
- mln::neighb, 844
 - bkd_niter, 845
 - fwd_niter, 845
 - neighb, 845
 - niter, 845
- mln::Neighborhood, 846
- mln::Neighborhood< void >, 846
- mln::norm, 385
 - l1, 386
 - l1_distance, 386
 - l2, 386
 - l2_distance, 387
 - linfty, 387
 - linfty_distance, 387
 - sqr_l2, 387
- mln::norm::impl, 387
- mln::Object, 846
- mln::opt, 388
 - at, 388, 389
- mln::opt::impl, 389
- mln::p2p_image, 847
 - domain, 848
 - fun, 848
 - operator(), 848, 849
 - p2p_image, 848
 - skeleton, 848
- mln::p_array, 849
 - append, 852
 - bkd_piter, 851
 - change, 852

- clear, 852
- element, 851
- fwd_piter, 851
- has, 852, 853
- i_element, 851
- insert, 853
- is_valid, 853
- memory_size, 853
- nsites, 853
- p_array, 852
- piter, 851
- psite, 851
- reserve, 854
- resize, 854
- std_vector, 854
- mln::p_centered, 854
 - bkd_piter, 856
 - center, 857
 - element, 856
 - fwd_piter, 856
 - has, 857
 - is_valid, 857
 - memory_size, 857
 - p_centered, 856
 - piter, 856
 - psite, 856
 - site, 856
 - window, 857
- mln::p_complex, 857
 - bkd_piter, 859
 - cplx, 860
 - element, 859
 - fwd_piter, 859
 - geom, 860
 - has, 860
 - is_valid, 861
 - nfaces, 861
 - nfaces_of_dim, 861
 - nsites, 861
 - p_complex, 860
 - piter, 859
 - psite, 859
- mln::p_edges, 861
 - bkd_piter, 863
 - edge, 863
 - element, 863
 - fun_t, 864
 - function, 866
 - fwd_piter, 864
 - graph, 866
 - graph_element, 864
 - graph_t, 864
 - has, 866
 - invalidate, 866
 - is_valid, 866
 - memory_size, 867
 - nedges, 867
 - nsites, 867
 - p_edges, 865
 - piter, 864
 - psite, 864
- mln::p_faces, 867
 - bkd_piter, 868
 - cplx, 869, 870
 - element, 868
 - fwd_piter, 868
 - is_valid, 870
 - nfaces, 870
 - nsites, 870
 - p_faces, 869
 - piter, 869
 - psite, 869
- mln::p_graph_piter, 870
 - graph, 871
 - id, 871
 - mln_q_subject, 871
 - next, 872
 - p_graph_piter, 871
- mln::p_if, 872
 - bkd_piter, 873
 - element, 873
 - fwd_piter, 873
 - has, 874
 - is_valid, 874
 - memory_size, 874
 - overset, 874
 - p_if, 874
 - piter, 873
 - pred, 875
 - predicate, 875
 - psite, 874
- mln::p_image, 875
 - bkd_piter, 877
 - clear, 878
 - element, 877
 - fwd_piter, 877
 - has, 878
 - i_element, 877
 - insert, 878
 - is_valid, 878
 - memory_size, 878
 - nsites, 879
 - operator typename internal::p_image_site_-
set< I >::ret, 879
 - p_image, 878
 - piter, 877
 - psite, 877
 - r_element, 877

- remove, 879
- S, 877
- toggle, 879
- mln::p_indexed_bkd_piter, 879
 - index, 880
 - next, 880
 - p_indexed_bkd_piter, 880
- mln::p_indexed_fwd_piter, 881
 - index, 882
 - next, 882
 - p_indexed_fwd_piter, 881
- mln::p_indexed_psite, 882
- mln::p_key, 882
 - bkd_piter, 884
 - change_key, 885
 - change_keys, 885
 - clear, 886
 - element, 884
 - exists_key, 886
 - fwd_piter, 884
 - has, 886
 - i_element, 885
 - insert, 886
 - is_valid, 887
 - key, 887
 - keys, 887
 - memory_size, 887
 - nsites, 887
 - operator(), 887
 - p_key, 885
 - piter, 885
 - psite, 885
 - r_element, 885
 - remove, 887
 - remove_key, 887
- mln::p_line2d, 888
 - bbox, 890
 - begin, 890
 - bkd_piter, 889
 - element, 889
 - end, 891
 - fwd_piter, 889
 - has, 891
 - is_valid, 891
 - memory_size, 891
 - nsites, 891
 - p_line2d, 890
 - piter, 889
 - psite, 890
 - q_box, 890
 - std_vector, 892
- mln::p_mutable_array_of, 892
 - bkd_piter, 893
 - clear, 894
 - element, 893
 - fwd_piter, 893
 - has, 894
 - i_element, 893
 - insert, 894
 - is_valid, 894
 - memory_size, 895
 - nelements, 895
 - p_mutable_array_of, 894
 - piter, 894
 - psite, 894
 - reserve, 895
- mln::p_n_faces_bkd_piter, 895
 - n, 896
 - p_n_faces_bkd_piter, 896
- mln::p_n_faces_fwd_piter, 896
 - n, 897
 - next, 897
 - p_n_faces_fwd_piter, 897
- mln::p_priority, 898
 - bkd_piter, 899
 - clear, 900
 - element, 900
 - exists_priority, 901
 - front, 901
 - fwd_piter, 900
 - has, 901
 - highest_priority, 901
 - i_element, 900
 - insert, 901
 - is_valid, 902
 - lowest_priority, 902
 - memory_size, 902
 - nsites, 902
 - operator(), 902
 - p_priority, 900
 - piter, 900
 - pop, 902
 - pop_front, 903
 - priorities, 903
 - psite, 900
 - push, 903
- mln::p_queue, 903
 - bkd_piter, 905
 - clear, 906
 - element, 905
 - front, 906
 - fwd_piter, 905
 - has, 906
 - i_element, 905
 - insert, 906
 - is_valid, 907
 - memory_size, 907
 - nsites, 907

- p_queue, 906
- piter, 905
- pop, 907
- pop_front, 907
- psite, 906
- push, 908
- std_deque, 908
- mln::p_queue_fast, 908
 - bkd_piter, 910
 - clear, 911
 - compute_has, 911
 - element, 910
 - empty, 911
 - front, 911
 - fwd_piter, 910
 - has, 911
 - i_element, 910
 - insert, 912
 - is_valid, 912
 - memory_size, 912
 - nsites, 912
 - p_queue_fast, 911
 - piter, 910
 - pop, 912
 - pop_front, 913
 - psite, 910
 - purge, 913
 - push, 913
 - reserve, 913
 - std_vector, 913
- mln::p_run, 913
 - bbox, 916
 - bkd_piter, 915
 - element, 915
 - end, 916
 - fwd_piter, 915
 - has, 917
 - has_index, 917
 - init, 917
 - is_valid, 917
 - length, 917
 - memory_size, 917
 - nsites, 918
 - p_run, 916
 - piter, 915
 - psite, 915
 - q_box, 916
 - start, 918
- mln::p_set, 918
 - bkd_piter, 920
 - clear, 921
 - element, 920
 - fwd_piter, 920
 - has, 921
 - i_element, 920
 - insert, 921
 - is_valid, 921
 - memory_size, 921
 - nsites, 922
 - p_set, 921
 - piter, 920
 - psite, 920
 - r_element, 920
 - remove, 922
 - std_vector, 922
 - util_set, 922
- mln::p_set_of, 922
 - bkd_piter, 924
 - clear, 925
 - element, 924
 - fwd_piter, 924
 - has, 925
 - i_element, 924
 - insert, 925
 - is_valid, 925
 - memory_size, 925
 - nelements, 925
 - p_set_of, 925
 - piter, 924
 - psite, 924
- mln::p_transformed, 926
 - bkd_piter, 927
 - element, 927
 - function, 928
 - fwd_piter, 927
 - has, 928
 - is_valid, 928
 - memory_size, 928
 - p_transformed, 928
 - piter, 927
 - primary_set, 928
 - psite, 927
- mln::p_transformed_piter, 929
 - change_target, 930
 - next, 930
 - p_transformed_piter, 929
- mln::p_vaccess, 930
 - bkd_piter, 932
 - element, 932
 - fwd_piter, 932
 - has, 933
 - i_element, 932
 - insert, 933
 - is_valid, 934
 - memory_size, 934
 - operator(), 934
 - p_vaccess, 933
 - piter, 932

- pset, 932
- psite, 932
- value, 933
- values, 934
- vset, 933
- mln::p_vertices, 934
 - bkd_piter, 936
 - element, 936
 - fun_t, 936
 - function, 939
 - fwd_piter, 937
 - graph, 939
 - graph_element, 937
 - graph_t, 937
 - has, 939
 - invalidate, 939
 - is_valid, 940
 - memory_size, 940
 - nsites, 940
 - nvertices, 940
 - operator(), 940
 - p_vertices, 938
 - piter, 937
 - psite, 937
 - vertex, 937
- mln::pixel, 941
 - change_to, 942
 - is_valid, 942
 - pixel, 942
- mln::Pixel_Iterator, 942
 - next, 943
- mln::plain, 943
 - operator I, 945
 - operator=, 945
 - plain, 944
 - skeleton, 944
- mln::Point, 945
 - operator+=", 947
 - operator-=, 948
 - operator/, 948
 - point, 947
 - to_point, 947
- mln::point, 948
 - coord, 951
 - delta, 951
 - dim, 952
 - dpsite, 951
 - grid, 951
 - h_vec, 951
 - last_coord, 953
 - minus_infty, 953
 - operator+=", 953
 - operator-=, 953
 - origin, 955
 - plus_infty, 954
 - point, 952, 953
 - set_all, 954
 - to_h_vec, 954
 - to_vec, 954
 - vec, 952
- mln::Point_Site, 955
 - operator<<, 958
 - operator+, 956
 - operator-, 957
 - operator==, 958
- mln::Point_Site< void >, 958
- mln::Proxy, 959
 - mln::Proxy< void >, 959
- mln::Pseudo_Site, 959
- mln::Pseudo_Site< void >, 960
- mln::pw, 389
- mln::pw::image, 960
 - image, 961
 - skeleton, 961
- mln::registration, 390
 - get_rot, 391
 - icp, 391, 392
 - registration1, 392
 - registration2, 392
 - registration3, 392
- mln::registration::closest_point_basic, 961
- mln::registration::closest_point_with_map, 962
- mln::Regular_Grid, 962
- mln::safe_image, 963
 - operator safe_image< const I >, 963
 - skeleton, 963
- mln::select, 393
- mln::select::p_of, 963
- mln::set, 393
 - card, 394
 - compute, 394
 - compute_with_weights, 394
 - get, 395
 - has, 395
 - mln_meta_accu_result, 395
- mln::Site, 964
- mln::Site< void >, 965
- mln::Site_Iterator, 965
 - next, 967
- mln::Site_Proxy, 967
- mln::Site_Proxy< void >, 968
- mln::Site_Set, 968
 - diff, 970
 - inter, 970
 - operator<, 970
 - operator<<, 970
 - operator<=, 970
 - operator==, 970

- sym_diff, 971
- uni, 971
- unique, 971
- mln::Site_Set< void >, 971
- mln::slice_image, 971
 - domain, 973
 - operator slice_image< const I >, 973
 - operator(), 973
 - skeleton, 972
 - sli, 973
 - slice_image, 973
- mln::sub_image, 974
 - domain, 975
 - operator sub_image< const I, S >, 975
 - skeleton, 974
 - sub_image, 975
- mln::sub_image_if, 975
 - domain, 976
 - skeleton, 976
 - sub_image_if, 976
- mln::subsampling, 396
 - antialiased, 396, 397
 - gaussian_subsampling, 397
 - subsampling, 397
- mln::tag, 397
- mln::test, 397
 - positive, 398
 - predicate, 398, 399
- mln::test::impl, 399
- mln::thru_image, 977
 - operator thru_image< const I, F >, 977
- mln::thrubin_image, 977
 - psite, 978
 - rvalue, 978
 - skeleton, 978
 - value, 978
- mln::topo, 399
 - detach, 404
 - edge, 404
 - is_facet, 404
 - make_algebraic_face, 405
 - make_algebraic_n_face, 405
 - operator<, 406, 407
 - operator<<, 407, 408
 - operator+, 406
 - operator-, 406
 - operator==, 408, 409
- mln::topo::adj_higher_dim_connected_n_face_-
 bkd_iter, 978
 - adj_higher_dim_connected_n_face_bkd_iter, 979
 - next, 979
- mln::topo::adj_higher_dim_connected_n_face_-
 fwd_iter, 980
 - adj_higher_dim_connected_n_face_fwd_iter, 980
 - next, 980
- mln::topo::adj_higher_face_bkd_iter, 981
 - adj_higher_face_bkd_iter, 981
 - next, 982
- mln::topo::adj_higher_face_fwd_iter, 982
 - adj_higher_face_fwd_iter, 982
 - next, 983
- mln::topo::adj_lower_dim_connected_n_face_-
 bkd_iter, 983
 - adj_lower_dim_connected_n_face_bkd_iter, 984
 - next, 984
- mln::topo::adj_lower_dim_connected_n_face_-
 fwd_iter, 984
 - adj_lower_dim_connected_n_face_fwd_iter, 985
 - next, 985
- mln::topo::adj_lower_face_bkd_iter, 985
 - adj_lower_face_bkd_iter, 986
 - next, 986
- mln::topo::adj_lower_face_fwd_iter, 986
 - adj_lower_face_fwd_iter, 987
 - next, 987
- mln::topo::adj_lower_higher_face_bkd_iter, 987
 - adj_lower_higher_face_bkd_iter, 988
 - next, 988
- mln::topo::adj_lower_higher_face_fwd_iter, 989
 - adj_lower_higher_face_fwd_iter, 989
 - next, 989
- mln::topo::adj_m_face_bkd_iter, 990
 - adj_m_face_bkd_iter, 990
 - next, 991
- mln::topo::adj_m_face_fwd_iter, 991
 - adj_m_face_fwd_iter, 992
 - next, 992
- mln::topo::algebraic_face, 992
 - algebraic_face, 994, 995
 - cplx, 995
 - data, 995
 - dec_face_id, 995
 - dec_n, 995
 - face_id, 996
 - higher_dim_adj_faces, 996
 - inc_face_id, 996
 - inc_n, 996
 - invalidate, 996
 - is_valid, 996
 - lower_dim_adj_faces, 996
 - n, 997
 - set_cplx, 997
 - set_face_id, 997
 - set_n, 997

- set_sign, 997
- sign, 997
- mln::topo::algebraic_n_face, 998
 - algebraic_n_face, 999, 1000
 - cplx, 1000
 - data, 1000
 - dec_face_id, 1000
 - face_id, 1000
 - higher_dim_adj_faces, 1000
 - inc_face_id, 1001
 - invalidate, 1001
 - is_valid, 1001
 - lower_dim_adj_faces, 1001
 - n, 1001
 - set_cplx, 1001
 - set_face_id, 1002
 - set_sign, 1002
 - sign, 1002
- mln::topo::center_only_iter, 1002
 - center_only_iter, 1003
 - next, 1003
- mln::topo::centered_bkd_iter_adapter, 1003
 - centered_bkd_iter_adapter, 1004
 - next, 1004
- mln::topo::centered_fwd_iter_adapter, 1005
 - centered_fwd_iter_adapter, 1005
 - next, 1005
- mln::topo::complex, 1006
 - add_face, 1007
 - addr, 1008
 - bkd_citer, 1007
 - complex, 1007
 - fwd_citer, 1007
 - nfaces, 1008
 - nfaces_of_dim, 1008
 - nfaces_of_static_dim, 1008
 - print, 1008
 - print_faces, 1009
- mln::topo::face, 1009
 - cplx, 1011
 - data, 1011
 - dec_face_id, 1011
 - dec_n, 1011
 - face, 1011
 - face_id, 1012
 - higher_dim_adj_faces, 1012
 - inc_face_id, 1012
 - inc_n, 1012
 - invalidate, 1012
 - is_valid, 1012
 - lower_dim_adj_faces, 1012
 - n, 1012
 - set_cplx, 1013
 - set_face_id, 1013
 - set_n, 1013
- mln::topo::face_bkd_iter, 1013
 - face_bkd_iter, 1014
 - next, 1014
 - start, 1014
- mln::topo::face_fwd_iter, 1014
 - face_fwd_iter, 1015
 - next, 1015
 - start, 1015
- mln::topo::is_n_face, 1016
- mln::topo::is_simple_2d_t, 1017
- mln::topo::is_simple_cell, 1017
 - D, 1020
 - mln_geom, 1019
 - operator(), 1019
 - psite, 1019
 - result, 1019
 - set_image, 1020
- mln::topo::n_face, 1020
 - cplx, 1022
 - data, 1022
 - dec_face_id, 1022
 - face_id, 1022
 - higher_dim_adj_faces, 1022
 - inc_face_id, 1022
 - invalidate, 1023
 - is_valid, 1023
 - lower_dim_adj_faces, 1023
 - n, 1023
 - n_face, 1021
 - set_cplx, 1023
 - set_face_id, 1023
- mln::topo::n_face_bkd_iter, 1024
 - n, 1025
 - n_face_bkd_iter, 1024
 - next, 1025
 - start, 1025
- mln::topo::n_face_fwd_iter, 1025
 - n, 1026
 - n_face_fwd_iter, 1026
 - next, 1026
 - start, 1026
- mln::topo::n_faces_set, 1027
 - add, 1028
 - faces, 1028
 - faces_type, 1027
 - reserve, 1028
- mln::topo::skeleton::is_simple_point, 1028
- mln::topo::static_n_face_bkd_iter, 1029
 - next, 1030
 - start, 1030
 - static_n_face_bkd_iter, 1029
- mln::topo::static_n_face_fwd_iter, 1030
 - next, 1031

- start, 1031
- static_n_face_fwd_iter, 1031
- mln::tr_image, 1031
 - domain, 1034
 - has, 1034
 - is_valid, 1034
 - lvalue, 1033
 - operator(), 1034
 - psite, 1033
 - rvalue, 1033
 - set_tr, 1034
 - site, 1033
 - skeleton, 1033
 - tr, 1034
 - tr_image, 1033
 - value, 1033
- mln::trace, 410
- mln::trait, 410
- mln::transform, 410
 - distance_and_closest_point_geodesic, 411, 412
 - distance_and_influence_zone_geodesic, 412
 - distance_front, 413
 - distance_geodesic, 413
 - hough, 413
 - influence_zone_front, 413, 414
 - influence_zone_geodesic, 414
 - influence_zone_geodesic_saturated, 414, 415
- mln::unproject_image, 1034
 - domain, 1035
 - operator(), 1035, 1036
 - unproject_image, 1035
- mln::util, 415
 - display_branch, 419
 - display_tree, 419
 - lemmings, 419
 - make_greater_point, 419
 - make_greater_psite, 419
 - operator<, 420
 - operator<<, 420
 - operator==, 420
 - ord_strict, 420
 - ord_weak, 421
 - tree_fast_to_image, 421
 - tree_to_fast, 421
 - tree_to_image, 421
 - vertex_id_t, 418
- mln::util::adjacency_matrix, 1036
 - adjacency_matrix, 1036, 1037
- mln::util::array, 1037
 - append, 1040, 1041
 - array, 1040
 - bkd_eiter, 1039
 - clear, 1041
 - eiter, 1039
 - element, 1039
 - fill, 1041
 - fwd_eiter, 1040
 - is_empty, 1041
 - last, 1041
 - memory_size, 1042
 - nelements, 1042
 - operator(), 1042
 - reserve, 1043
 - resize, 1043
 - result, 1040
 - size, 1043
 - std_vector, 1044
- mln::util::branch, 1044
 - apex, 1045
 - branch, 1044
 - util_tree, 1045
- mln::util::branch_iter, 1045
 - deepness, 1046
 - invalidate, 1046
 - is_valid, 1046
 - next, 1046
 - operator util::tree_node< T > &, 1046
 - start, 1047
- mln::util::branch_iter_ind, 1047
 - deepness, 1048
 - invalidate, 1048
 - is_valid, 1048
 - next, 1048
 - operator util::tree_node< T > &, 1048
 - start, 1048
- mln::util::couple, 1048
 - change_both, 1050
 - change_first, 1050
 - change_second, 1050
 - first, 1050
 - second, 1050
- mln::util::eat, 1050
- mln::util::edge, 1051
 - category, 1053
 - change_graph, 1053
 - edge, 1053
 - graph, 1053
 - graph_t, 1053
 - id, 1053
 - id_t, 1053
 - id_value_t, 1053
 - invalidate, 1054
 - is_valid, 1054
 - ith_nbh_edge, 1054
 - nmax_nbh_edges, 1054
 - operator edge_id_t, 1054
 - update_id, 1054

- v1, 1054
- v2, 1055
- v_other, 1055
- mln::util::fibonacci_heap, 1055
 - clear, 1057
 - fibonacci_heap, 1057
 - front, 1057
 - is_empty, 1057
 - is_valid, 1058
 - nelements, 1058
 - operator=, 1058
 - pop_front, 1058
 - push, 1058, 1059
- mln::util::graph, 1059
 - add_edge, 1063
 - add_vertex, 1063
 - add_vertices, 1063
 - e_ith_nbh_edge, 1063
 - e_nmax, 1063
 - e_nmax_nbh_edges, 1064
 - edge, 1064
 - edge_fwd_iter, 1061
 - edge_nbh_edge_fwd_iter, 1061
 - edges, 1064
 - edges_set_t, 1061
 - edges_t, 1061
 - graph, 1062
 - has_e, 1064
 - has_v, 1064
 - is_subgraph_of, 1065
 - v1, 1065
 - v2, 1065
 - v_ith_nbh_edge, 1065
 - v_ith_nbh_vertex, 1065
 - v_nmax, 1065
 - v_nmax_nbh_edges, 1066
 - v_nmax_nbh_vertices, 1066
 - vertex, 1066
 - vertex_fwd_iter, 1062
 - vertex_nbh_edge_fwd_iter, 1062
 - vertex_nbh_vertex_fwd_iter, 1062
 - vertices_t, 1062
- mln::util::greater_point, 1066
 - operator(), 1067
- mln::util::greater_psite, 1067
 - operator(), 1067
- mln::util::head, 1067
- mln::util::ignore, 1068
- mln::util::ilcell, 1068
- mln::util::impl, 422
- mln::util::line_graph, 1069
 - e_ith_nbh_edge, 1072
 - e_nmax, 1072
 - e_nmax_nbh_edges, 1072
 - edge, 1072
 - edge_fwd_iter, 1071
 - edge_nbh_edge_fwd_iter, 1071
 - edges_t, 1071
 - graph, 1073
 - has, 1073
 - has_e, 1073
 - has_v, 1073
 - is_subgraph_of, 1074
 - v1, 1074
 - v2, 1074
 - v_ith_nbh_edge, 1074
 - v_ith_nbh_vertex, 1074
 - v_nmax, 1074
 - v_nmax_nbh_edges, 1075
 - v_nmax_nbh_vertices, 1075
 - vertex, 1075
 - vertex_fwd_iter, 1071
 - vertex_nbh_edge_fwd_iter, 1071
 - vertex_nbh_vertex_fwd_iter, 1072
 - vertices_t, 1072
- mln::util::nil, 1075
- mln::util::node, 1076
- mln::util::object_id, 1076
 - object_id, 1078
 - value_t, 1078
- mln::util::ord, 1078
- mln::util::ord_pair, 1078
 - change_both, 1080
 - change_first, 1080
 - change_second, 1080
 - first, 1080
 - second, 1080
- mln::util::pix, 1081
 - ima, 1082
 - p, 1082
 - pix, 1082
 - psite, 1081
 - v, 1082
 - value, 1081
- mln::util::set, 1082
 - bkd_eiter, 1085
 - clear, 1085
 - eiter, 1085
 - element, 1085
 - first_element, 1085
 - fwd_eiter, 1085
 - has, 1086
 - insert, 1086
 - is_empty, 1086
 - last_element, 1087
 - memory_size, 1087
 - nelements, 1087
 - remove, 1088

- set, 1085
- std_vector, 1088
- mln::util::site_pair, 1088
 - first, 1089
 - pair, 1089
 - second, 1090
- mln::util::soft_heap, 1090
 - ~soft_heap, 1092
 - clear, 1092
 - element, 1091
 - is_empty, 1092
 - is_valid, 1092
 - nelements, 1092
 - pop_front, 1092
 - push, 1092, 1093
 - soft_heap, 1091
- mln::util::timer, 1093
- mln::util::tracked_ptr, 1094
 - ~tracked_ptr, 1095
 - operator bool, 1095
 - operator->, 1095
 - operator=, 1096
 - tracked_ptr, 1095
- mln::util::tree, 1096
 - add_tree_down, 1097
 - add_tree_up, 1097
 - check_consistency, 1097
 - main_branch, 1098
 - root, 1098
 - tree, 1097
- mln::util::tree_node, 1098
 - add_child, 1100
 - check_consistency, 1100
 - children, 1101
 - delete_tree_node, 1101
 - elt, 1101
 - parent, 1102
 - print, 1102
 - search, 1102
 - search_rec, 1102
 - set_parent, 1103
 - tree_node, 1100
- mln::util::vertex, 1103
 - Category, 1105
 - change_graph, 1106
 - edge_with, 1106
 - graph, 1106
 - graph_t, 1105
 - id, 1106
 - id_t, 1105
 - id_value_t, 1105
 - invalidate, 1106
 - is_valid, 1106
 - ith_nbh_edge, 1106
 - ith_nbh_vertex, 1107
 - nmax_nbh_edges, 1107
 - nmax_nbh_vertices, 1107
 - operator vertex_id_t, 1107
 - other, 1107
 - update_id, 1107
 - vertex, 1105
- mln::util::yes, 1108
- mln::Value, 1108
- mln::value, 422
 - cast, 428
 - equiv, 428
 - float01_16, 426
 - float01_8, 426
 - gl16, 427
 - gl8, 427
 - glf, 427
 - int_s16, 427
 - int_s32, 427
 - int_s8, 427
 - int_u12, 427
 - int_u16, 427
 - int_u32, 427
 - int_u8, 428
 - label_16, 428
 - label_32, 428
 - label_8, 428
 - operator<<, 430–432
 - operator*, 429
 - operator+, 429
 - operator-, 429
 - operator/, 429, 430
 - operator==, 433
 - other, 433
 - rgb16, 428
 - rgb8, 428
 - stack, 433
- mln::value::float01, 1110
 - enc, 1111
 - equiv, 1111
 - float01, 1111
 - nbits, 1111
 - operator float, 1111
 - set_nbits, 1112
 - to_nbits, 1112
 - value, 1112
 - value_ind, 1112
- mln::value::float01_f, 1112
 - float01_f, 1113
 - operator float, 1113
 - operator=, 1113
 - value, 1113
- mln::value::graylevel, 1113
 - graylevel, 1115

- operator=, 1116
- to_float, 1116
- value, 1116
- mln::value::graylevel_f, 1117
 - graylevel_f, 1118
 - operator graylevel< n >, 1118
 - operator=, 1118, 1119
 - value, 1119
- mln::value::impl, 433
- mln::value::int_s, 1119
 - int_s, 1121
 - one, 1121
 - operator int, 1121
 - operator=, 1121
 - zero, 1121
- mln::value::int_u, 1122
 - int_u, 1123
 - next, 1123
 - operator unsigned, 1123
 - operator-, 1124
 - operator=, 1124
- mln::value::int_u_sat, 1124
 - int_u_sat, 1125
 - one, 1126
 - operator int, 1125
 - operator+=, 1126
 - operator=, 1126
 - operator=, 1126
 - zero, 1126
- mln::value::Integer, 1126
- mln::value::Integer< void >, 1127
- mln::value::label, 1127
 - enc, 1128
 - label, 1129
 - next, 1129
 - operator unsigned, 1129
 - operator++, 1129
 - operator--, 1129
 - operator=, 1129, 1130
 - prev, 1130
- mln::value::lut_vec, 1130
 - bkd_viter, 1132
 - fwd_viter, 1132
 - has, 1133
 - index_of, 1133
 - lut_vec, 1133
 - nvalues, 1133
 - value, 1132
- mln::value::proxy, 1134
 - ~proxy, 1136
 - enc, 1135
 - equiv, 1135
 - operator=, 1136
 - proxy, 1135
 - to_value, 1136
- mln::value::qt::rgb32, 1136
 - operator=, 1138
 - red, 1138
 - rgb32, 1137
 - zero, 1138
- mln::value::rgb, 1138
 - operator=, 1140
 - red, 1140
 - rgb, 1139
 - zero, 1140
- mln::value::set, 1140
 - the, 1141
- mln::value::sign, 1141
 - enc, 1142
 - equiv, 1142
 - one, 1143
 - operator int, 1143
 - operator=, 1143
 - sign, 1142
 - zero, 1143
- mln::value::stack_image, 1143
 - domain_t, 1144
 - is_valid, 1145
 - lvalue, 1144
 - operator(), 1145, 1146
 - psite, 1144
 - rvalue, 1145
 - skeleton, 1145
 - stack_image, 1145
 - value, 1145
- mln::value::super_value< sign >, 1146
- mln::value::value_array, 1146
 - operator(), 1147
 - value_array, 1147
 - vset, 1147
- mln::Value_Iterator, 1147
 - next, 1149
 - operator<<, 1149
- mln::Value_Set, 1149
- mln::Vertex, 1150
- mln::vertex_image, 1151
 - graph_t, 1152
 - nbh_t, 1152
 - operator(), 1153
 - site_function_t, 1152
 - skeleton, 1152
 - vertex_image, 1153
 - vertex_nbh_t, 1152
 - vertex_win_t, 1152
 - win_t, 1152
- mln::violent_cast_image, 1153
 - lvalue, 1154
 - operator(), 1155

- rvalue, 1154
- skeleton, 1154
- value, 1154
- violent_cast_image, 1155
- mln::w_window, 1155
 - bkd_qiter, 1157
 - clear, 1157
 - dpsite, 1157
 - fwd_qiter, 1157
 - insert, 1157
 - is_symmetric, 1158
 - operator<<, 1159
 - operator==, 1159
 - std_vector, 1158
 - sym, 1158
 - w, 1158
 - w_window, 1157
 - weight, 1157
 - weights, 1158
 - win, 1158
- mln::Weighted_Window, 1159
 - operator-, 1160
- mln::win, 433
 - diff, 435
 - mln_regular, 435
 - sym, 436
- mln::win::backdiag2d, 1160
 - backdiag2d, 1161
 - length, 1161
- mln::win::ball, 1161
 - ball, 1162
 - diameter, 1162
- mln::win::cube3d, 1162
 - cube3d, 1163
 - length, 1163
- mln::win::cuboid3d, 1163
 - cuboid3d, 1164
 - depth, 1165
 - height, 1165
 - volume, 1165
 - width, 1165
- mln::win::diag2d, 1165
 - diag2d, 1166
 - length, 1166
- mln::win::line, 1166
 - length, 1168
 - line, 1167
 - size, 1168
- mln::win::multiple, 1168
- mln::win::multiple_size, 1168
- mln::win::octagon2d, 1169
 - area, 1170
 - length, 1170
 - octagon2d, 1169
- mln::win::rectangle2d, 1170
 - area, 1171
 - height, 1171
 - rectangle2d, 1171
 - std_vector, 1171
 - width, 1171
- mln::Window, 1171
- mln::window, 1172
 - bkd_qiter, 1174
 - clear, 1175
 - delta, 1175
 - dp, 1175
 - fwd_qiter, 1174
 - has, 1175
 - insert, 1175
 - is_centered, 1176
 - is_empty, 1176
 - is_symmetric, 1176
 - operator==, 1177
 - print, 1176
 - qiter, 1174
 - regular, 1174
 - size, 1176
 - std_vector, 1176
 - sym, 1177
 - window, 1174
- mln::world::inter_pixel::is_separator, 1177
- mln_ch_convolve
 - mln::linear, 317, 318
- mln_ch_convolve_grad
 - mln::linear, 318
- mln_exact
 - mln, 153
- mln_gen_complex_neighborhood
 - mln, 153, 154
- mln_gen_complex_window
 - mln, 154
- mln_gen_complex_window_p
 - mln, 154, 155
- mln_geom
 - mln::topo::is_simple_cell, 1019
- mln_image_from_grid
 - mln::convert, 200, 201
- mln_meta_accu_result
 - mln::accu, 167
 - mln::data, 210
 - mln::set, 395
- mln_q_subject
 - mln::p_graph_piter, 871
- mln_regular
 - mln, 155
 - mln::win, 435
- mln_trait_op_geq
 - mln, 155

- mln_trait_op_greater
 - mln, 155
- mln_trait_op_leq
 - mln, 156
- mln_trait_op_minus_twice
 - mln::morpho::elementary, 369
- mln_trait_op_neq
 - mln, 156
- mln_window
 - mln::convert, 201
- modneighb1d
 - c2, 102
 - neighb1d, 102
- modneighb2d
 - c2_col, 103
 - c2_row, 103
 - c4, 103
 - c8, 104
 - neighb2d, 103
- modneighb3d
 - c18, 105
 - c26, 105
 - c2_3d_sli, 106
 - c4_3d, 106
 - c6, 107
 - c8_3d, 107
 - neighb3d, 105
- modwin1d
 - segment1d, 112
 - window1d, 113
- modwin2d
 - disk2d, 114
 - hline2d, 114
 - vline2d, 114
 - win_c4p, 115
 - win_c8p, 115
 - window2d, 114
- modwin3d
 - sline3d, 116
 - sphere3d, 116
 - win_c4p_3d, 117
 - win_c8p_3d, 117
 - window3d, 117
- mosaic
 - mln::debug, 228
- Multiple accumulators, 94
- Multiple windows, 118
- n
 - mln::complex_psite, 591
 - mln::faces_psite, 670
 - mln::p_n_faces_bkd_piter, 896
 - mln::p_n_faces_fwd_piter, 897
 - mln::topo::algebraic_face, 997
 - mln::topo::algebraic_n_face, 1001
 - mln::topo::face, 1012
 - mln::topo::n_face, 1023
 - mln::topo::n_face_bkd_iter, 1025
 - mln::topo::n_face_fwd_iter, 1026
- N-D windows, 118
- n_face
 - mln::topo::n_face, 1021
- n_face_bkd_iter
 - mln::topo::n_face_bkd_iter, 1024
- n_face_fwd_iter
 - mln::topo::n_face_fwd_iter, 1026
- n_items
 - mln::accu::stat::var, 535
 - mln::accu::stat::variance, 537
- nbh_t
 - mln::edge_image, 658
 - mln::vertex_image, 1152
- nbits
 - mln::value::float01, 1111
- ncols
 - mln::geom, 260
 - mln::image2d, 775
 - mln::image3d, 783
- nedges
 - mln::p_edges, 867
- neighb
 - mln::neighb, 845
- neighb1d
 - modneighb1d, 102
- neighb2d
 - modneighb2d, 103
- neighb3d
 - modneighb3d, 105
- Neighborhoods, 101
- nelements
 - mln::doc::Fastest_Image, 612
 - mln::image1d, 770
 - mln::image2d, 775
 - mln::image3d, 783
 - mln::p_mutable_array_of, 895
 - mln::p_set_of, 925
 - mln::util::array, 1042
 - mln::util::fibonacci_heap, 1058
 - mln::util::set, 1087
 - mln::util::soft_heap, 1092
- next
 - mln::bkd_pixter1d, 548
 - mln::bkd_pixter2d, 549
 - mln::bkd_pixter3d, 551
 - mln::box_runend_piter, 566
 - mln::box_runstart_piter, 567
 - mln::complex_neighborhood_bkd_piter, 587
 - mln::complex_neighborhood_fwd_piter, 589

- mln::complex_window_bkd_piter, 593
- mln::complex_window_fwd_piter, 595
- mln::dpoints_bkd_pixter, 651
- mln::dpoints_fwd_pixter, 653
- mln::dpsites_bkd_piter, 655
- mln::dpsites_fwd_piter, 656
- mln::fwd_pixter1d, 722
- mln::fwd_pixter2d, 724
- mln::fwd_pixter3d, 725
- mln::graph_window_if_piter, 757
- mln::graph_window_piter, 760
- mln::Iterator, 791
- mln::p_graph_piter, 872
- mln::p_indexed_bkd_piter, 880
- mln::p_indexed_fwd_piter, 882
- mln::p_n_faces_fwd_piter, 897
- mln::p_transformed_piter, 930
- mln::Pixel_Iterator, 943
- mln::Site_Iterator, 967
- mln::topo::adj_higher_dim_connected_n_-
face_bkd_iter, 979
- mln::topo::adj_higher_dim_connected_n_-
face_fwd_iter, 980
- mln::topo::adj_higher_face_bkd_iter, 982
- mln::topo::adj_higher_face_fwd_iter, 983
- mln::topo::adj_lower_dim_connected_n_-
face_bkd_iter, 984
- mln::topo::adj_lower_dim_connected_n_-
face_fwd_iter, 985
- mln::topo::adj_lower_face_bkd_iter, 986
- mln::topo::adj_lower_face_fwd_iter, 987
- mln::topo::adj_lower_higher_face_bkd_iter,
988
- mln::topo::adj_lower_higher_face_fwd_iter,
989
- mln::topo::adj_m_face_bkd_iter, 991
- mln::topo::adj_m_face_fwd_iter, 992
- mln::topo::center_only_iter, 1003
- mln::topo::centered_bkd_iter_adapter, 1004
- mln::topo::centered_fwd_iter_adapter, 1005
- mln::topo::face_bkd_iter, 1014
- mln::topo::face_fwd_iter, 1015
- mln::topo::n_face_bkd_iter, 1025
- mln::topo::n_face_fwd_iter, 1026
- mln::topo::static_n_face_bkd_iter, 1030
- mln::topo::static_n_face_fwd_iter, 1031
- mln::util::branch_iter, 1046
- mln::util::branch_iter_ind, 1048
- mln::value::int_u, 1123
- mln::value::label, 1129
- mln::Value_Iterator, 1149
- nfaces
 - mln::p_complex, 861
 - mln::p_faces, 870
- mln::topo::complex, 1008
- nfaces_of_dim
 - mln::p_complex, 861
 - mln::topo::complex, 1008
- nfaces_of_static_dim
 - mln::topo::complex, 1008
- ninds
 - mln::geom, 260
 - mln::image1d, 770
- niter
 - mln::doc::Neighborhood, 624
 - mln::graph_elt_mixed_neighborhood, 736
 - mln::graph_elt_neighborhood, 742
 - mln::graph_elt_neighborhood_if, 743
 - mln::mixed_neighb, 833
 - mln::neighb, 845
- nlabels
 - mln::labeled_image, 794
 - mln::labeled_image_base, 797
- nmax_nbh_edges
 - mln::util::edge, 1054
 - mln::util::vertex, 1107
- nmax_nbh_vertices
 - mln::util::vertex, 1107
- not_inplace
 - mln::logical, 328
- nrows
 - mln::geom, 260
 - mln::image2d, 775
 - mln::image3d, 784
- nsites
 - mln::accu::center, 438
 - mln::Box, 562
 - mln::box, 557
 - mln::doc::Box, 603
 - mln::doc::Fastest_Image, 612
 - mln::doc::Image, 620
 - mln::geom, 260
 - mln::p_array, 853
 - mln::p_complex, 861
 - mln::p_edges, 867
 - mln::p_faces, 870
 - mln::p_image, 879
 - mln::p_key, 887
 - mln::p_line2d, 891
 - mln::p_priority, 902
 - mln::p_queue, 907
 - mln::p_queue_fast, 912
 - mln::p_run, 918
 - mln::p_set, 922
 - mln::p_vertices, 940
- nslis
 - mln::geom, 261
 - mln::image3d, 784

- nvalues
 - mln::doc::Value_Set, 638
 - mln::value::lut_vec, 1133
- nvertices
 - mln::p_vertices, 940
- object_id
 - mln::util::object_id, 1078
- octagon2d
 - mln::win::octagon2d, 1169
- olive
 - mln::literal, 324
- On images, 91
- On site sets, 91
- On values, 92
- one
 - mln::literal, 325
 - mln::value::int_s, 1121
 - mln::value::int_u_sat, 1126
 - mln::value::sign, 1143
- opening
 - mln::morpho::elementary, 369
- operator bool
 - mln::util::tracked_ptr, 1095
- operator decorated_image< const I, D >
 - mln::decorated_image, 598
- operator edge_id_t
 - mln::util::edge, 1054
- operator float
 - mln::value::float01, 1111
 - mln::value::float01_f, 1113
- operator graylevel< n >
 - mln::value::graylevel_f, 1118
- operator I
 - mln::plain, 945
- operator image_if< const I, F >
 - mln::image_if, 786
- operator int
 - mln::value::int_s, 1121
 - mln::value::int_u_sat, 1125
 - mln::value::sign, 1143
- operator mat< n, l, U >
 - mln::algebra::h_vec, 546
- operator mln::algebra::vec< dpoint< G, C >::dim, Q >
 - mln::dpoint, 648
- operator psite
 - mln::doc::Site_Iterator, 632
- operator safe_image< const I >
 - mln::safe_image, 963
- operator slice_image< const I >
 - mln::slice_image, 973
- operator sub_image< const I, S >
 - mln::sub_image, 975
- operator thru_image< const I, F >
 - mln::thru_image, 977
- operator typename internal::p_image_site_set< I >::ret
 - mln::p_image, 879
- operator unsigned
 - mln::value::int_u, 1123
 - mln::value::label, 1129
- operator util::tree_node< T > &
 - mln::util::branch_iter, 1046
 - mln::util::branch_iter_ind, 1048
- operator value
 - mln::doc::Value_Iterator, 636
- operator vertex_id_t
 - mln::util::vertex, 1107
- operator<
 - mln, 158
 - mln::Box, 563
 - mln::Site_Set, 970
 - mln::topo, 406, 407
 - mln::util, 420
- operator<<
 - mln, 159
 - mln::Box, 563
 - mln::box, 559
 - mln::fun::i2v, 244
 - mln::Gpoint, 732
 - mln::Point_Site, 958
 - mln::Site_Set, 970
 - mln::topo, 407, 408
 - mln::util, 420
 - mln::value, 430–432
 - mln::Value_Iterator, 1149
 - mln::w_window, 1159
- operator<=
 - mln, 159, 160
 - mln::Box, 563, 564
 - mln::Site_Set, 970
- operator*
 - mln, 157
 - mln::algebra, 174
 - mln::value, 429
- operator()
 - mln::complex_image, 585
 - mln::decorated_image, 598
 - mln::doc::Fastest_Image, 612
 - mln::doc::Image, 621
 - mln::edge_image, 659
 - mln::extension_fun, 663
 - mln::extension_ima, 665
 - mln::extension_val, 668
 - mln::flat_image, 673
 - mln::fun::v2v::rgb8_to_rgbn, 686
 - mln::fun::x2v::bilinear, 705

- mln::fun::x2x::linear, 708
- mln::fun::x2x::rotation, 711
- mln::fun::x2x::translation, 714
- mln::fun_image, 716
- mln::geom::complex_geometry, 728
- mln::hexa, 764
- mln::image1d, 770
- mln::image2d, 775
- mln::image2d_h, 779
- mln::image3d, 784
- mln::lazy_image, 800
- mln::p2p_image, 848, 849
- mln::p_key, 887
- mln::p_priority, 902
- mln::p_vaccess, 934
- mln::p_vertices, 940
- mln::slice_image, 973
- mln::topo::is_simple_cell, 1019
- mln::tr_image, 1034
- mln::unproject_image, 1035, 1036
- mln::util::array, 1042
- mln::util::greater_point, 1067
- mln::util::greater_psite, 1067
- mln::value::stack_image, 1145, 1146
- mln::value::value_array, 1147
- mln::vertex_image, 1153
- mln::violent_cast_image, 1155
- operator+
 - mln::Gpoint, 730
 - mln::Point_Site, 956
 - mln::topo, 406
 - mln::value, 429
- operator++
 - mln, 157
 - mln::value::label, 1129
- operator+=
 - mln::Gpoint, 731
 - mln::Point, 947
 - mln::point, 953
 - mln::value::int_u_sat, 1126
- operator-
 - mln, 157
 - mln::Gpoint, 731
 - mln::Point_Site, 957
 - mln::topo, 406
 - mln::value, 429
 - mln::value::int_u, 1124
 - mln::Weighted_Window, 1160
- operator->
 - mln::util::tracked_ptr, 1095
- operator--
 - mln, 158
 - mln::value::label, 1129
- operator==
 - mln::Gpoint, 732
 - mln::Point, 948
 - mln::point, 953
 - mln::value::int_u_sat, 1126
- operator/
 - mln::Gpoint, 732
 - mln::Point, 948
 - mln::value, 429, 430
- operator=
 - mln::plain, 945
 - mln::util::fibonacci_heap, 1058
 - mln::util::tracked_ptr, 1096
 - mln::value::float01_f, 1113
 - mln::value::graylevel, 1116
 - mln::value::graylevel_f, 1118, 1119
 - mln::value::int_s, 1121
 - mln::value::int_u, 1124
 - mln::value::int_u_sat, 1126
 - mln::value::label, 1129, 1130
 - mln::value::proxy, 1136
 - mln::value::qt::rgb32, 1138
 - mln::value::rgb, 1140
 - mln::value::sign, 1143
- operator==
 - mln, 160–162
 - mln::accu::stat, 173
 - mln::Box, 564
 - mln::Gpoint, 732
 - mln::Point_Site, 958
 - mln::Site_Set, 970
 - mln::topo, 408, 409
 - mln::util, 420
 - mln::value, 433
 - mln::w_window, 1159
 - mln::window, 1177
- or_inplace
 - mln::logical, 328
- orange
 - mln::literal, 325
- ord_pair
 - mln::make, 345
- ord_strict
 - mln::util, 420
- ord_weak
 - mln::util, 421
- origin
 - mln::algebra::h_vec, 547
 - mln::literal, 325
 - mln::point, 955
- other
 - mln::util::vertex, 1107
 - mln::value, 433
- overset
 - mln::p_if, 874

- P
 - mln::graph_window_if_piter, 756
 - mln::graph_window_piter, 758
- p
 - mln::util::pix, 1082
- p2p_image
 - mln::p2p_image, 848
- p_array
 - mln::p_array, 852
- p_centered
 - mln::p_centered, 856
- p_complex
 - mln::p_complex, 860
- p_edges
 - mln::p_edges, 865
- p_edges_with_mass_centers
 - mln::make, 346
- p_faces
 - mln::p_faces, 869
- p_graph_piter
 - mln::p_graph_piter, 871
- p_if
 - mln::p_if, 874
- p_image
 - mln::p_image, 878
- p_indexed_bkd_piter
 - mln::p_indexed_bkd_piter, 880
- p_indexed_fwd_piter
 - mln::p_indexed_fwd_piter, 881
- p_key
 - mln::p_key, 885
- p_line2d
 - mln::p_line2d, 890
- p_mutable_array_of
 - mln::p_mutable_array_of, 894
- p_n_faces_bkd_piter
 - mln::p_n_faces_bkd_piter, 896
- p_n_faces_fwd_piter
 - mln::p_n_faces_fwd_piter, 897
- p_priority
 - mln::p_priority, 900
- p_queue
 - mln::p_queue, 906
- p_queue_fast
 - mln::p_queue_fast, 911
- p_run
 - mln::p_run, 916
- p_run2d
 - mln, 147
- p_runs2d
 - mln, 147
- p_set
 - mln::p_set, 921
- p_set_of
 - mln::p_set_of, 925
- p_transformed
 - mln::p_transformed, 928
- p_transformed_piter
 - mln::p_transformed_piter, 929
- p_vaccess
 - mln::p_vaccess, 933
- p_vertices
 - mln::p_vertices, 938
- p_vertices_with_mass_centers
 - mln::make, 346
- pack
 - mln::labeling, 305
- pack_inplace
 - mln::labeling, 306
- pair
 - mln::util::site_pair, 1089
- parent
 - mln::util::tree_node, 1102
- paste
 - mln::data, 210
 - mln::data::impl::generic, 222
- paste_without_localization
 - mln::data, 211
- paste_without_localization_fast
 - mln::data::impl, 218
- paste_without_localization_fastest
 - mln::data::impl, 218
- paste_without_localization_lines
 - mln::data::impl, 219
- pcenter
 - mln::box, 558
- pink
 - mln::literal, 325
- piter
 - mln::box, 555
 - mln::p_array, 851
 - mln::p_centered, 856
 - mln::p_complex, 859
 - mln::p_edges, 864
 - mln::p_faces, 869
 - mln::p_if, 873
 - mln::p_image, 877
 - mln::p_key, 885
 - mln::p_line2d, 889
 - mln::p_mutable_array_of, 894
 - mln::p_priority, 900
 - mln::p_queue, 905
 - mln::p_queue_fast, 910
 - mln::p_run, 915
 - mln::p_set, 920
 - mln::p_set_of, 924
 - mln::p_transformed, 927
 - mln::p_vaccess, 932

- mln::p_vertices, 937
- pix
 - mln::make, 346
 - mln::util::pix, 1082
- pixel
 - mln::make, 347
 - mln::pixel, 942
- plain
 - mln::plain, 944
- plot
 - mln::draw, 236
- plus
 - mln::arith, 183
 - mln::morpho, 365
- plus_cst
 - mln::arith, 184, 185
- plus_cst_inplace
 - mln::arith, 185
- plus_infty
 - mln::point, 954
- plus_inplace
 - mln::arith, 185
- pmax
 - mln::box, 558
 - mln::doc::Box, 603
- pmin
 - mln::box, 558
 - mln::doc::Box, 603
- pmin_pmax
 - mln::geom, 261
- point
 - mln::doc::Dpoint, 605
 - mln::doc::Fastest_Image, 609
 - mln::doc::Image, 618
 - mln::doc::Neighborhood, 624
 - mln::doc::Point_Site, 629
 - mln::doc::Weighted_Window, 640
 - mln::Point, 947
 - mln::point, 952, 953
- point1d
 - mln, 148
- point1df
 - mln, 148
- point2d
 - mln, 148
- point2d_h
 - mln, 148
 - mln::make, 347
- point2df
 - mln, 148
- point3d
 - mln, 148
- point3df
 - mln, 148
- point_at_index
 - mln::doc::Fastest_Image, 613
 - mln::image1d, 770
 - mln::image2d, 775
 - mln::image3d, 784
- polygon
 - mln::draw, 236
- pop
 - mln::p_priority, 902
 - mln::p_queue, 907
 - mln::p_queue_fast, 912
- pop_front
 - mln::p_priority, 903
 - mln::p_queue, 907
 - mln::p_queue_fast, 913
 - mln::util::fibonacci_heap, 1058
 - mln::util::soft_heap, 1092
- positive
 - mln::test, 398
- pred
 - mln::p_if, 875
- predicate
 - mln::p_if, 875
 - mln::test, 398, 399
- prev
 - mln::value::label, 1130
- primary
 - mln, 163
- primary_set
 - mln::p_transformed, 928
- print
 - mln::topo::complex, 1008
 - mln::util::tree_node, 1102
 - mln::window, 1176
- print_faces
 - mln::topo::complex, 1009
- println
 - mln::debug, 228, 229
- println_with_border
 - mln::debug, 229
- priorities
 - mln::p_priority, 903
- propagate_if
 - mln::morpho::tree, 377
- propagate_if_value
 - mln::morpho::tree, 378
- propagate_node_to_ancestors
 - mln::morpho::tree, 378, 379
- propagate_node_to_descendants
 - mln::morpho::tree, 379
- propagate_representative
 - mln::morpho::tree, 379
- proxy
 - mln::value::proxy, 1135

- pset
 - mln::doc::Fastest_Image, 609
 - mln::doc::Image, 618
 - mln::p_vaccess, 932
- psite
 - mln::box, 555
 - mln::complex_neighborhood_bkd_piter, 586
 - mln::complex_neighborhood_fwd_piter, 588
 - mln::complex_window_bkd_piter, 593
 - mln::complex_window_fwd_piter, 594
 - mln::decorated_image, 597
 - mln::doc::Box, 602
 - mln::doc::Fastest_Image, 609
 - mln::doc::Image, 619
 - mln::doc::Site_Iterator, 632
 - mln::doc::Site_Set, 634
 - mln::dpoint, 646
 - mln::graph_elt_mixed_window, 739
 - mln::graph_elt_window, 747
 - mln::graph_elt_window_if, 751
 - mln::hexa, 762
 - mln::image2d_h, 778
 - mln::interpolated, 787
 - mln::p_array, 851
 - mln::p_centered, 856
 - mln::p_complex, 859
 - mln::p_edges, 864
 - mln::p_faces, 869
 - mln::p_if, 874
 - mln::p_image, 877
 - mln::p_key, 885
 - mln::p_line2d, 890
 - mln::p_mutable_array_of, 894
 - mln::p_priority, 900
 - mln::p_queue, 906
 - mln::p_queue_fast, 910
 - mln::p_run, 915
 - mln::p_set, 920
 - mln::p_set_of, 924
 - mln::p_transformed, 927
 - mln::p_vaccess, 932
 - mln::p_vertices, 937
 - mln::thrubin_image, 978
 - mln::topo::is_simple_cell, 1019
 - mln::tr_image, 1033
 - mln::util::pix, 1081
 - mln::value::stack_image, 1144
- ptransform
 - mln, 163
- purge
 - mln::p_queue_fast, 913
- purple
 - mln::literal, 325
- push
 - mln::p_priority, 903
 - mln::p_queue, 908
 - mln::p_queue_fast, 913
 - mln::util::fibonacci_heap, 1058, 1059
 - mln::util::soft_heap, 1092, 1093
- put_word
 - mln::debug, 229
- q_box
 - mln::p_line2d, 890
 - mln::p_run, 916
- qiter
 - mln::doc::Window, 643
 - mln::graph_elt_mixed_window, 739
 - mln::graph_elt_window, 747
 - mln::graph_elt_window_if, 751
 - mln::window, 1174
- Queue based, 110
- r_element
 - mln::p_image, 877
 - mln::p_key, 885
 - mln::p_set, 920
- rag_and_labeled_wsl
 - mln::make, 347
- rank_filter
 - mln::morpho, 365
- read_header
 - mln::io::fld, 276
- rectangle2d
 - mln::win::rectangle2d, 1171
- rectangularity
 - mln::accu::site_set::rectangularity, 509
- red
 - mln::literal, 325
 - mln::value::qt::rgb32, 1138
 - mln::value::rgb, 1140
- region_adjacency_graph
 - mln::make, 348
- regional_maxima
 - mln::labeling, 306
- regional_minima
 - mln::labeling, 306
- registration1
 - mln::registration, 392
- registration2
 - mln::registration, 392
- registration3
 - mln::registration, 392
- regular
 - mln::window, 1174
- relabel
 - mln::labeled_image, 794
 - mln::labeled_image_base, 797

- mln::labeling, 307
 - relabel_inplace
 - mln::labeling, 308
 - relabelfun
 - mln::make, 348, 349
 - remove
 - mln::p_image, 879
 - mln::p_key, 887
 - mln::p_set, 922
 - mln::util::set, 1088
 - remove_key
 - mln::p_key, 887
 - replace
 - mln::data, 211
 - reserve
 - mln::p_array, 854
 - mln::p_mutable_array_of, 895
 - mln::p_queue_fast, 913
 - mln::topo::n_faces_set, 1028
 - mln::util::array, 1043
 - resize
 - mln::border, 193
 - mln::p_array, 854
 - mln::util::array, 1043
 - result
 - mln::graph::attribute::card_t, 734
 - mln::graph::attribute::representative_t, 735
 - mln::topo::is_simple_cell, 1019
 - mln::util::array, 1040
 - revert
 - mln::arith, 186
 - revert_inplace
 - mln::arith, 186
 - rgb
 - mln::value::rgb, 1139
 - rgb16
 - mln::value, 428
 - rgb32
 - mln::value::qt::rgb32, 1137
 - rgb8
 - mln::value, 428
 - rgb8_2complex_image3df
 - mln, 148
 - root
 - mln::util::tree, 1098
 - rotate
 - mln::geom, 261, 262
 - rotation
 - mln::fun::x2x::rotation, 710, 711
 - Routines, 99
 - run_length
 - mln::box_runend_piter, 566
 - mln::box_runstart_piter, 567
 - rvalue
 - mln::complex_image, 584
 - mln::decorated_image, 597
 - mln::doc::Fastest_Image, 609
 - mln::doc::Generalized_Pixel, 615
 - mln::doc::Image, 619
 - mln::doc::Pixel_Iterator, 627
 - mln::extension_fun, 662
 - mln::extension_ima, 664
 - mln::extension_val, 667
 - mln::flat_image, 672
 - mln::fun_image, 715
 - mln::hexa, 763
 - mln::image1d, 768
 - mln::image2d, 773
 - mln::image2d_h, 778
 - mln::image3d, 781
 - mln::interpolated, 787
 - mln::lazy_image, 799
 - mln::thrubin_image, 978
 - mln::tr_image, 1033
 - mln::value::stack_image, 1145
 - mln::violent_cast_image, 1154
- S
- mln::p_image, 877
 - sagittal_dec
 - mln, 164
 - saturate
 - mln::data, 211
 - saturate_inplace
 - mln::data, 212
 - save
 - mln::io::cloud, 271
 - mln::io::dump, 274
 - mln::io::magick, 277
 - mln::io::off, 279
 - mln::io::pbm, 280
 - mln::io::pfm, 283
 - mln::io::pgm, 284
 - mln::io::plot, 286, 287
 - mln::io::pnm, 289
 - mln::io::ppm, 291
 - mln::io::raw, 293
 - mln::io::txt, 295
 - save_bin_alt
 - mln::io::off, 279
 - search
 - mln::util::tree_node, 1102
 - search_rec
 - mln::util::tree_node, 1102
 - second
 - mln::accu::pair, 500
 - mln::accu::stat::min_max, 528
 - mln::util::couple, 1050

- mln::util::ord_pair, 1080
- mln::util::site_pair, 1090
- second_accu
 - mln::accu::pair, 500
 - mln::accu::stat::min_max, 528
- seeds2tiling
 - mln::geom, 263
 - mln::geom::impl, 265
- segment1d
 - modwin1d, 112
- set
 - mln::util::set, 1085
- set_all
 - mln::dpoint, 648
 - mln::point, 954
- set_alpha
 - mln::fun::x2x::rotation, 711
- set_axis
 - mln::fun::x2x::rotation, 711
- set_cplx
 - mln::topo::algebraic_face, 997
 - mln::topo::algebraic_n_face, 1001
 - mln::topo::face, 1013
 - mln::topo::n_face, 1023
- set_face_id
 - mln::topo::algebraic_face, 997
 - mln::topo::algebraic_n_face, 1002
 - mln::topo::face, 1013
 - mln::topo::n_face, 1023
- set_image
 - mln::topo::is_simple_cell, 1020
- set_n
 - mln::topo::algebraic_face, 997
 - mln::topo::face, 1013
- set_nbits
 - mln::value::float01, 1112
- set_parent
 - mln::util::tree_node, 1103
- set_sign
 - mln::topo::algebraic_face, 997
 - mln::topo::algebraic_n_face, 1002
- set_t
 - mln::fun::x2x::translation, 714
- set_tr
 - mln::tr_image, 1034
- set_value
 - mln::accu::count_adjacent_vertices, 441
 - mln::accu::count_labels, 443
 - mln::accu::count_value, 444
 - mln::accu::math::count, 456
 - mln::accu::shape::height, 505
 - mln::accu::shape::volume, 508
 - mln::accu::stat::max, 515
 - mln::accu::stat::min, 524
- mln::morpho::attribute::sum, 842
- sign
 - mln::topo::algebraic_face, 997
 - mln::topo::algebraic_n_face, 1002
 - mln::value::sign, 1142
- site
 - mln::box, 555
 - mln::doc::Box, 602
 - mln::doc::Site_Set, 634
 - mln::dpoint, 646
 - mln::graph_elt_mixed_window, 739
 - mln::graph_elt_window, 747
 - mln::graph_elt_window_if, 751
 - mln::graph_window_base, 754
 - mln::p_centered, 856
 - mln::tr_image, 1033
- Site sets, 108
- site_function_t
 - mln::edge_image, 658
 - mln::vertex_image, 1152
- site_set
 - mln::complex_psite, 591
 - mln::draw, 237
 - mln::faces_psite, 670
- size
 - mln::util::array, 1043
 - mln::win::line, 1168
 - mln::window, 1176
- skeleton
 - mln::complex_image, 584
 - mln::decorated_image, 597
 - mln::doc::Fastest_Image, 609
 - mln::doc::Image, 619
 - mln::edge_image, 659
 - mln::extended, 660
 - mln::extension_fun, 662
 - mln::extension_ima, 664
 - mln::extension_val, 667
 - mln::flat_image, 672
 - mln::fun_image, 715
 - mln::hexa, 763
 - mln::image1d, 768
 - mln::image2d, 773
 - mln::image2d_h, 778
 - mln::image3d, 781
 - mln::image_if, 785
 - mln::interpolated, 787
 - mln::labeled_image, 793
 - mln::lazy_image, 799
 - mln::p2p_image, 848
 - mln::plain, 944
 - mln::pw::image, 961
 - mln::safe_image, 963
 - mln::slice_image, 972

- mln::sub_image, 974
- mln::sub_image_if, 976
- mln::thrubin_image, 978
- mln::tr_image, 1033
- mln::value::stack_image, 1145
- mln::vertex_image, 1152
- mln::violent_cast_image, 1154
- sli
 - mln::slice_image, 973
- slice_image
 - mln::slice_image, 973
- slices_2d
 - mln::debug, 229
- sline3d
 - modwin3d, 116
- soft_heap
 - mln::util::soft_heap, 1091
- sort_offsets_increasing
 - mln::data, 212
- sort_psites_decreasing
 - mln::data, 212
- sort_psites_increasing
 - mln::data, 212
- space_2complex_geometry
 - mln, 149
- Sparse types, 109
- sphere3d
 - modwin3d, 116
- sqr_l2
 - mln::norm, 387
- stack
 - mln::value, 433
- stack_image
 - mln::value::stack_image, 1145
- standard_deviation
 - mln::accu::stat::variance, 537
- start
 - mln::doc::Iterator, 622
 - mln::doc::Pixel_Iterator, 628
 - mln::doc::Site_Iterator, 632
 - mln::doc::Value_Iterator, 636
 - mln::dpoints_bkd_pixter, 651
 - mln::dpoints_fwd_pixter, 653
 - mln::p_run, 918
 - mln::topo::face_bkd_iter, 1014
 - mln::topo::face_fwd_iter, 1015
 - mln::topo::n_face_bkd_iter, 1025
 - mln::topo::n_face_fwd_iter, 1026
 - mln::topo::static_n_face_bkd_iter, 1030
 - mln::topo::static_n_face_fwd_iter, 1031
 - mln::util::branch_iter, 1047
 - mln::util::branch_iter_ind, 1048
- static_n_face_bkd_iter
 - mln::topo::static_n_face_bkd_iter, 1029
- static_n_face_fwd_iter
 - mln::topo::static_n_face_fwd_iter, 1031
- std_deque
 - mln::p_queue, 908
- std_vector
 - mln::p_array, 854
 - mln::p_line2d, 892
 - mln::p_queue_fast, 913
 - mln::p_set, 922
 - mln::util::array, 1044
 - mln::util::set, 1088
 - mln::w_window, 1158
 - mln::win::rectangle2d, 1171
 - mln::window, 1176
- stretch
 - mln::data, 213
 - mln::data::impl, 219
- structural
 - mln::morpho::closing::approx, 368
 - mln::morpho::opening::approx, 371
- sub_image
 - mln::sub_image, 975
- sub_image_if
 - mln::sub_image_if, 976
- subdomain
 - mln::labeled_image, 794
 - mln::labeled_image_base, 797
- subsampling
 - mln::subsampling, 397
- subtractive
 - mln::morpho::tree::filter, 382
- sum
 - mln::accu::stat::mean, 518
 - mln::accu::stat::variance, 538
 - mln::estim, 239
- superpose
 - mln::debug, 229, 230
 - mln::labeling, 308
 - mln::morpho::watershed, 384
- sym
 - mln::doc::Weighted_Window, 641
 - mln::graph_elt_mixed_window, 740
 - mln::graph_elt_window, 748
 - mln::graph_elt_window_if, 753
 - mln::graph_window_base, 755
 - mln::w_window, 1158
 - mln::win, 436
 - mln::window, 1177
- sym_diff
 - mln::Box, 564
 - mln::Site_Set, 971
- t
 - mln::algebra::h_mat, 545

- mln::algebra::h_vec, 546
- mln::fun::x2x::translation, 714
- take
 - mln::accu, 167
 - mln::accu::histo, 446
 - mln::accu::label_used, 447
 - mln::accu::stat::histo3d_rgb, 513
 - mln::accu::stat::median_alt, 520
 - mln::doc::Accumulator, 600
- take_as_init
 - mln::accu::center, 438
 - mln::accu::convolve, 440
 - mln::accu::count_adjacent_vertices, 441
 - mln::accu::count_labels, 443
 - mln::accu::count_value, 444
 - mln::accu::histo, 446
 - mln::accu::label_used, 447
 - mln::accu::logic::land, 449
 - mln::accu::logic::land_basic, 450
 - mln::accu::logic::lor, 452
 - mln::accu::logic::lor_basic, 453
 - mln::accu::maj_h, 455
 - mln::accu::math::count, 456
 - mln::accu::math::inf, 458
 - mln::accu::math::sum, 459
 - mln::accu::math::sup, 461
 - mln::accu::max_site, 462
 - mln::accu::nil, 496
 - mln::accu::p, 498
 - mln::accu::pair, 501
 - mln::accu::rms, 502
 - mln::accu::shape::bbox, 503
 - mln::accu::shape::height, 506
 - mln::accu::shape::volume, 508
 - mln::accu::site_set::rectangularity, 509
 - mln::accu::stat::deviation, 511
 - mln::accu::stat::histo3d_rgb, 514
 - mln::accu::stat::max, 515
 - mln::accu::stat::max_h, 517
 - mln::accu::stat::mean, 518
 - mln::accu::stat::median_alt, 520
 - mln::accu::stat::median_h, 522
 - mln::accu::stat::min, 524
 - mln::accu::stat::min_h, 526
 - mln::accu::stat::min_max, 529
 - mln::accu::stat::rank, 530
 - mln::accu::stat::rank< bool >, 532
 - mln::accu::stat::rank_high_quant, 533
 - mln::accu::stat::var, 535
 - mln::accu::stat::variance, 538
 - mln::accu::tuple, 539
 - mln::accu::val, 541
 - mln::Accumulator, 543
 - mln::morpho::attribute::card, 835
 - mln::morpho::attribute::count_adjacent_vertices, 836
 - mln::morpho::attribute::height, 838
 - mln::morpho::attribute::sharpness, 840
 - mln::morpho::attribute::sum, 842
 - mln::morpho::attribute::volume, 844
- take_n_times
 - mln::accu::center, 438
 - mln::accu::convolve, 440
 - mln::accu::count_adjacent_vertices, 441
 - mln::accu::count_labels, 443
 - mln::accu::count_value, 445
 - mln::accu::histo, 446
 - mln::accu::label_used, 448
 - mln::accu::logic::land, 449
 - mln::accu::logic::land_basic, 451
 - mln::accu::logic::lor, 452
 - mln::accu::logic::lor_basic, 453
 - mln::accu::maj_h, 455
 - mln::accu::math::count, 456
 - mln::accu::math::inf, 458
 - mln::accu::math::sum, 459
 - mln::accu::math::sup, 461
 - mln::accu::max_site, 462
 - mln::accu::nil, 496
 - mln::accu::p, 498
 - mln::accu::pair, 501
 - mln::accu::rms, 502
 - mln::accu::shape::bbox, 504
 - mln::accu::shape::height, 506
 - mln::accu::shape::volume, 508
 - mln::accu::site_set::rectangularity, 510
 - mln::accu::stat::deviation, 511
 - mln::accu::stat::histo3d_rgb, 514
 - mln::accu::stat::max, 515
 - mln::accu::stat::max_h, 517
 - mln::accu::stat::mean, 519
 - mln::accu::stat::median_alt, 520
 - mln::accu::stat::median_h, 522
 - mln::accu::stat::min, 524
 - mln::accu::stat::min_h, 526
 - mln::accu::stat::min_max, 529
 - mln::accu::stat::rank, 530
 - mln::accu::stat::rank< bool >, 532
 - mln::accu::stat::rank_high_quant, 533
 - mln::accu::stat::var, 535
 - mln::accu::stat::variance, 538
 - mln::accu::tuple, 540
 - mln::accu::val, 541
 - mln::Accumulator, 543
 - mln::morpho::attribute::card, 835
 - mln::morpho::attribute::count_adjacent_vertices, 836
 - mln::morpho::attribute::height, 838

- mln::morpho::attribute::sharpness, 840
- mln::morpho::attribute::sum, 842
- mln::morpho::attribute::volume, 844
- target
 - mln::graph_elt_mixed_window, 739
 - mln::graph_elt_window, 747
 - mln::graph_elt_window_if, 751
- target_site_set
 - mln::graph_window_piter, 760
- teal
 - mln::literal, 325
- the
 - mln::value::set, 1141
- thick_miss
 - mln::morpho, 365
- thickening
 - mln::morpho, 365
- thin_fit
 - mln::morpho, 366
- thinning
 - mln::morpho, 366
- threshold
 - mln::binarization, 189
- times
 - mln::arith, 187
- times_cst
 - mln::arith, 187
- times_inplace
 - mln::arith, 187
- to
 - mln::convert, 201
- to_dpoint
 - mln::convert, 201
 - mln::Dpoint, 644
- to_enc
 - mln::data, 213
- to_float
 - mln::value::graylevel, 1116
- to_fun
 - mln::convert, 201, 204
- to_h_vec
 - mln::point, 954
- to_image
 - mln::convert, 201
- to_larger
 - mln::box, 558
- to_nbits
 - mln::value::float01, 1112
- to_neighb
 - mln::graph, 266
- to_p_array
 - mln::convert, 202
- to_p_set
 - mln::convert, 202, 203
- to_point
 - mln::doc::Point_Site, 630
 - mln::Point, 947
- to_qimage
 - mln::convert, 203
- to_result
 - mln::accu::center, 438
 - mln::accu::convolve, 440
 - mln::accu::count_adjacent_vertices, 442
 - mln::accu::count_labels, 443
 - mln::accu::count_value, 445
 - mln::accu::label_used, 448
 - mln::accu::logic::land, 449
 - mln::accu::logic::land_basic, 451
 - mln::accu::logic::lor, 452
 - mln::accu::logic::lor_basic, 454
 - mln::accu::maj_h, 455
 - mln::accu::math::count, 457
 - mln::accu::math::inf, 458
 - mln::accu::math::sum, 459
 - mln::accu::math::sup, 461
 - mln::accu::max_site, 462
 - mln::accu::nil, 496
 - mln::accu::p, 498
 - mln::accu::pair, 501
 - mln::accu::rms, 502
 - mln::accu::shape::bbox, 504
 - mln::accu::shape::height, 506
 - mln::accu::shape::volume, 508
 - mln::accu::site_set::rectangularity, 510
 - mln::accu::stat::deviation, 511
 - mln::accu::stat::histo3d_rgb, 514
 - mln::accu::stat::max, 515
 - mln::accu::stat::max_h, 517
 - mln::accu::stat::mean, 519
 - mln::accu::stat::median_alt, 520
 - mln::accu::stat::median_h, 522
 - mln::accu::stat::min, 525
 - mln::accu::stat::min_h, 526
 - mln::accu::stat::min_max, 529
 - mln::accu::stat::rank, 530
 - mln::accu::stat::rank < bool >, 532
 - mln::accu::stat::rank_high_quant, 533
 - mln::accu::stat::var, 535
 - mln::accu::stat::variance, 538
 - mln::accu::tuple, 540
 - mln::accu::val, 541
 - mln::morpho::attribute::card, 835
 - mln::morpho::attribute::count_adjacent_vertices, 836
 - mln::morpho::attribute::height, 838
 - mln::morpho::attribute::sharpness, 840
 - mln::morpho::attribute::sum, 842
 - mln::morpho::attribute::volume, 844

- to_upper_window
 - mln::convert, 203
- to_value
 - mln::value::proxy, 1136
- to_vec
 - mln::algebra::h_vec, 547
 - mln::dpoint, 648
 - mln::point, 954
- to_win
 - mln::graph, 267
- to_window
 - mln::convert, 203, 204
- toggle
 - mln::p_image, 879
- top_hat_black
 - mln::morpho, 366
 - mln::morpho::elementary, 370
- top_hat_self_complementary
 - mln::morpho, 366
 - mln::morpho::elementary, 370
- top_hat_white
 - mln::morpho, 367
 - mln::morpho::elementary, 370
- topological
 - mln::morpho::watershed, 384
- tr
 - mln::tr_image, 1034
- tr_image
 - mln::tr_image, 1033
- tracked_ptr
 - mln::util::tracked_ptr, 1095
- trait::graph, 1178
- trait::graph< mln::complex_image< 1, G, V > >, 1179
- trait::graph< mln::image2d< T > >, 1179
- transform
 - mln::data, 213, 214
 - mln::data::impl::generic, 222
- transform_inplace
 - mln::data, 214
 - mln::data::impl::generic, 223
- transform_inplace_lowq
 - mln::data::impl, 220
- translate
 - mln::geom, 263, 264
- translation
 - mln::fun::x2x::translation, 713
- tree
 - mln::util::tree, 1097
- tree_fast_to_image
 - mln::util, 421
- tree_node
 - mln::util::tree_node, 1100
- tree_to_fast
 - mln::util, 421
- tree_to_image
 - mln::util, 421
- Types, 98
- uni
 - mln::Box, 564
 - mln::Site_Set, 971
- unique
 - mln::Box, 564
 - mln::Site_Set, 971
- unproject_image
 - mln::unproject_image, 1035
- unsigned_2complex_image3df
 - mln, 149
- untake
 - mln::morpho::attribute::sum, 842
- up
 - mln, 164
- update
 - mln::data, 215
 - mln::data::impl::generic, 223
 - mln::dpoints_bkd_pixter, 651
 - mln::dpoints_fwd_pixter, 654
- update_data
 - mln::labeled_image, 794
 - mln::labeled_image_base, 798
- update_fastest
 - mln::data::impl, 220
- update_id
 - mln::util::edge, 1054
 - mln::util::vertex, 1107
- util_set
 - mln::p_set, 922
- util_tree
 - mln::util::branch, 1045
- Utilities, 110
- v
 - mln::util::pix, 1082
- v1
 - mln::util::edge, 1054
 - mln::util::graph, 1065
 - mln::util::line_graph, 1074
- v2
 - mln::util::edge, 1055
 - mln::util::graph, 1065
 - mln::util::line_graph, 1074
- v2w2v functions, 118
- v2w_w2v functions, 119
- v_ith_nbh_edge
 - mln::util::graph, 1065
 - mln::util::line_graph, 1074
- v_ith_nbh_vertex

- mln::util::graph, 1065
- mln::util::line_graph, 1074
- v_nmax
 - mln::util::graph, 1065
 - mln::util::line_graph, 1074
- v_nmax_nbh_edges
 - mln::util::graph, 1066
 - mln::util::line_graph, 1075
- v_nmax_nbh_vertices
 - mln::util::graph, 1066
 - mln::util::line_graph, 1075
- v_other
 - mln::util::edge, 1055
- val
 - mln::doc::Generalized_Pixel, 615
 - mln::doc::Pixel_Iterator, 628
- value
 - mln::accu::shape::height, 505
 - mln::accu::shape::volume, 507
 - mln::complex_image, 584
 - mln::doc::Fastest_Image, 610
 - mln::doc::Generalized_Pixel, 615
 - mln::doc::Image, 619
 - mln::doc::Pixel_Iterator, 627
 - mln::doc::Value_Iterator, 636
 - mln::doc::Value_Set, 638
 - mln::extended, 660
 - mln::extension_fun, 662
 - mln::extension_ima, 665
 - mln::extension_val, 667
 - mln::flat_image, 672
 - mln::fun_image, 716
 - mln::hexa, 763
 - mln::image1d, 768
 - mln::image2d, 773
 - mln::image2d_h, 778
 - mln::image3d, 781
 - mln::interpolated, 787
 - mln::labeling, 309
 - mln::p_vaccess, 933
 - mln::thruin_image, 978
 - mln::tr_image, 1033
 - mln::util::pix, 1081
 - mln::value::float01, 1112
 - mln::value::float01_f, 1113
 - mln::value::graylevel, 1116
 - mln::value::graylevel_f, 1119
 - mln::value::lut_vec, 1132
 - mln::value::stack_image, 1145
 - mln::violent_cast_image, 1154
- value_and_compute
 - mln::labeling, 309
- value_array
 - mln::value::value_array, 1147
- value_ind
 - mln::value::float01, 1112
- value_t
 - mln::util::object_id, 1078
- values
 - mln::complex_image, 585
 - mln::doc::Fastest_Image, 613
 - mln::doc::Image, 621
 - mln::p_vaccess, 934
- Values morphers, 96
- var
 - mln::accu::stat::variance, 538
- variance
 - mln::accu::stat::var, 536
- vbbox
 - mln::image1d, 771
 - mln::image3d, 784
- vec
 - mln::dpoint, 646
 - mln::make, 349, 350
 - mln::point, 952
- vec2d_d
 - mln, 149
- vec2d_f
 - mln, 149
- vec3d_d
 - mln, 149
- vec3d_f
 - mln, 149
- vect
 - mln::accu::histo, 446
- vertex
 - mln::p_vertices, 937
 - mln::util::graph, 1066
 - mln::util::line_graph, 1075
 - mln::util::vertex, 1105
- vertex_fwd_iter
 - mln::util::graph, 1062
 - mln::util::line_graph, 1071
- vertex_id_t
 - mln::util, 418
- vertex_image
 - mln::make, 350, 351
 - mln::vertex_image, 1153
- vertex_nbh_edge_fwd_iter
 - mln::util::graph, 1062
 - mln::util::line_graph, 1071
- vertex_nbh_t
 - mln::vertex_image, 1152
- vertex_nbh_vertex_fwd_iter
 - mln::util::graph, 1062
 - mln::util::line_graph, 1072
- vertex_win_t
 - mln::vertex_image, 1152

- vertical_symmetry
 - mln::geom, 264
- vertices_t
 - mln::util::graph, 1062
 - mln::util::line_graph, 1072
- violent_cast_image
 - mln::violent_cast_image, 1155
- violet
 - mln::literal, 325
- vline2d
 - modwin2d, 114
- volume
 - mln::morpho::attribute::sharpness, 840
 - mln::win::cuboid3d, 1165
- voronoi
 - mln::make, 351
- vprod
 - mln::algebra, 175
- vset
 - mln::doc::Fastest_Image, 610
 - mln::doc::Image, 619
 - mln::p_vaccess, 933
 - mln::value::value_array, 1147
- vv2b functions, 119
- w
 - mln::w_window, 1158
- w_window
 - mln::make, 351
 - mln::w_window, 1157
- w_window1d
 - mln::make, 352
- w_window1d_float
 - mln, 149
- w_window1d_int
 - mln, 149
 - mln::make, 352
- w_window2d
 - mln::make, 353
- w_window2d_float
 - mln, 150
- w_window2d_int
 - mln, 150
 - mln::make, 353
- w_window3d
 - mln::make, 353
- w_window3d_float
 - mln, 150
- w_window3d_int
 - mln, 150
 - mln::make, 354
- w_window_directional
 - mln::make, 354
- weight
 - mln::doc::Weighted_Window, 641
 - mln::w_window, 1157
- weights
 - mln::w_window, 1158
- white
 - mln::literal, 326
- width
 - mln::win::cuboid3d, 1165
 - mln::win::rectangle2d, 1171
- win
 - mln::doc::Weighted_Window, 641
 - mln::w_window, 1158
- win_c4p
 - modwin2d, 115
- win_c4p_3d
 - modwin3d, 117
- win_c8p
 - modwin2d, 115
- win_c8p_3d
 - modwin3d, 117
- win_t
 - mln::edge_image, 659
 - mln::vertex_image, 1152
- window
 - mln::doc::Weighted_Window, 641
 - mln::p_centered, 857
 - mln::window, 1174
- window1d
 - modwin1d, 113
- window2d
 - modwin2d, 114
- window3d
 - modwin3d, 117
- Windows, 111
- wrap
 - mln::data, 215
 - mln::labeling, 310
- write_header
 - mln::io::fld, 276
- xor_inplace
 - mln::logical, 329
- yellow
 - mln::literal, 326
- z_order
 - mln::debug, 230
- zero
 - mln::algebra::h_vec, 547
 - mln::literal, 326
 - mln::value::int_s, 1121
 - mln::value::int_u_sat, 1126
 - mln::value::qt::rgb32, 1138

`mln::value::rgb`, [1140](#)
`mln::value::sign`, [1143](#)