

Milena (Olena)
User documentation 1.0a Id

Generated by Doxygen 1.5.6

Fri May 7 19:43:40 2010

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	5
2.2	Requirements	5
2.2.1	To compile the user examples	5
2.2.2	To compile the documentation (Optional)	5
2.2.3	To develop in Olena	5
2.3	Getting Olena	5
2.4	Building Olena	5
2.5	Foreword	5
2.6	Generality	5
2.7	Directory hierarchy	5
2.8	Writing and compiling a program with Olena	5
2.9	Site	5
2.10	Site set	5
2.11	Basic interface	5
2.12	Optional interface	5
2.13	Image	6
2.14	Definition	6
2.15	Possible image types	6
2.16	Possible value types	6
2.17	Domain	6
2.18	Border and extension	7
2.18.1	Image border	7

2.18.2	Generality on image extension	7
2.18.3	Different extensions	7
2.18.3.1	Extension with a value	7
2.18.3.2	Extension with a function	8
2.18.3.3	Extension with an image	8
2.19	Interface	9
2.20	Load and save images	9
2.21	Create an image	9
2.22	Access and modify values	10
2.23	Image size	10
2.24	Structural elements: Window and neighborhood	11
2.25	Define an element	11
2.25.1	Window	11
2.25.2	Neighborhood	11
2.25.3	Custom structural elements	11
2.25.4	Conversion between Neighborhoods and Windows	11
2.26	Sites, psites and dpoints	11
2.27	Need for site	11
2.28	Need for psite	11
2.29	From psite to site	12
2.30	Dpoint	12
2.31	Iterators	12
2.32	Memory management	13
2.33	Basic routines	13
2.34	Fill	13
2.35	Paste	13
2.36	Blobs	14
2.37	Logical not	14
2.38	Compute	15
2.38.1	Accumulators	15
2.38.2	Example with labeling::compute()	15
2.39	Working with parts of an image	15
2.39.1	Restrict an image with a site set	16
2.39.2	Restrict an image with a predicate	16
2.39.3	Restrict an image with a C function	16
2.40	Input / Output	17

2.41	ImageMagick	17
2.42	GDCM	17
2.43	Graphs and images	17
2.44	Description	17
2.45	Example	17
2.46	Useful global variables	22
2.47	Useful macros	22
2.48	Variable declaration macros	22
2.49	Iterator type macros	22
2.49.1	Default iterator types	22
2.49.2	Forward iterator types	22
2.49.3	Backward iterators	22
2.49.4	Graph iterators	22
2.50	Common Compilation Errors	22
2.51	Installation	22
2.52	Requirements	22
2.52.1	To compile the user examples	22
2.52.2	To compile the documentation (Optional)	22
2.52.3	To develop in Olena	22
2.53	Getting Olena	22
2.54	Building Olena	22
2.55	Foreword	22
2.56	Generality	22
2.57	Directory hierarchy	22
2.58	Writing and compiling a program with Olena	22
2.59	Site	22
2.60	Site set	22
2.61	Basic interface	23
2.62	Optional interface	23
2.63	Image	23
2.64	Definition	23
2.65	Possible image types	23
2.66	Possible value types	23
2.67	Domain	23
2.68	Border and extension	24
2.68.1	Image border	24

2.68.2	Generality on image extension	25
2.68.3	Different extensions	25
2.68.3.1	Extension with a value	25
2.68.3.2	Extension with a function	25
2.68.3.3	Extension with an image	25
2.69	Interface	27
2.70	Load and save images	27
2.71	Create an image	27
2.72	Access and modify values	27
2.73	Image size	28
2.74	Structural elements: Window and neighborhood	28
2.75	Define an element	28
2.75.1	Window	28
2.75.2	Neighborhood	28
2.75.3	Custom structural elements	28
2.75.4	Conversion between Neighborhoods and Windows	29
2.76	Sites, psites and dpoints	29
2.77	Need for site	29
2.78	Need for psite	29
2.79	From psite to site	29
2.80	Dpoint	29
2.81	Iterators	30
2.82	Memory management	30
2.83	Basic routines	31
2.84	Fill	31
2.85	Paste	31
2.86	Blobs	31
2.87	Logical not	32
2.88	Compute	32
2.88.1	Accumulators	32
2.88.2	Example with labeling::compute()	32
2.89	Working with parts of an image	33
2.89.1	Restrict an image with a site set	33
2.89.2	Restrict an image with a predicate	33
2.89.3	Restrict an image with a C function	34
2.90	Input / Output	35

2.91	ImageMagick	35
2.92	GDCM	35
2.93	Graphs and images	35
2.94	Description	35
2.95	Example	35
2.96	Useful global variables	38
2.97	Useful macros	38
2.98	Variable declaration macros	38
2.99	Iterator type macros	38
2.99.1	Default iterator types	38
2.99.2	Forward iterator types	38
2.99.3	Backward iterators	38
2.99.4	Graph iterators	38
2.100	Common Compilation Errors	38
3	Tutorial	39
4	Module Index	41
4.1	Modules	41
5	Namespace Index	43
5.1	Namespace List	43
6	Class Index	47
6.1	Class Hierarchy	47
7	Class Index	83
7.1	Class List	83
8	Module Documentation	93
8.1	On site sets	93
8.1.1	Detailed Description	93
8.2	On images	94
8.2.1	Detailed Description	94
8.3	On values	95
8.3.1	Detailed Description	96
8.4	Multiple accumulators	97
8.4.1	Detailed Description	97
8.5	Graphes	98

8.5.1	Detailed Description	98
8.6	Images	99
8.6.1	Detailed Description	99
8.7	Basic types	100
8.7.1	Detailed Description	100
8.8	Image morphers	101
8.9	Values morphers	102
8.9.1	Detailed Description	102
8.10	Domain morphers	103
8.10.1	Detailed Description	103
8.11	Identity morphers	104
8.11.1	Detailed Description	104
8.12	Types	105
8.12.1	Detailed Description	105
8.13	Accumulators	106
8.13.1	Detailed Description	106
8.14	Routines	107
8.15	Canvas	108
8.16	Functions	109
8.16.1	Detailed Description	110
8.17	Neighborhoods	111
8.17.1	Detailed Description	111
8.18	1D neighborhoods	112
8.18.1	Detailed Description	112
8.18.2	Typedef Documentation	112
8.18.2.1	neighb1d	112
8.18.3	Function Documentation	112
8.18.3.1	c2	112
8.19	2D neighborhoods	113
8.19.1	Detailed Description	113
8.19.2	Typedef Documentation	113
8.19.2.1	neighb2d	113
8.19.3	Function Documentation	113
8.19.3.1	c2_col	113
8.19.3.2	c2_row	114
8.19.3.3	c4	114

8.19.3.4	c8	114
8.20	3D neighborhoods	115
8.20.1	Detailed Description	115
8.20.2	Typedef Documentation	115
8.20.2.1	neighb3d	115
8.20.3	Function Documentation	115
8.20.3.1	c18	115
8.20.3.2	c26	116
8.20.3.3	c4_3d	116
8.20.3.4	c6	117
8.20.3.5	c8_3d	117
8.21	Site sets	118
8.21.1	Detailed Description	118
8.22	Basic types	119
8.22.1	Detailed Description	119
8.23	Graph based	120
8.23.1	Detailed Description	120
8.24	Complex based	121
8.24.1	Detailed Description	121
8.25	Sparse types	122
8.25.1	Detailed Description	122
8.26	Queue based	123
8.26.1	Detailed Description	123
8.27	Utilities	124
8.27.1	Detailed Description	124
8.28	Windows	125
8.28.1	Detailed Description	125
8.29	1D windows	126
8.29.1	Detailed Description	126
8.29.2	Typedef Documentation	126
8.29.2.1	segment1d	126
8.29.2.2	window1d	126
8.30	2D windows	127
8.30.1	Detailed Description	127
8.30.2	Typedef Documentation	128
8.30.2.1	disk2d	128

8.30.2.2	hline2d	128
8.30.2.3	vline2d	128
8.30.2.4	window2d	128
8.30.3	Function Documentation	128
8.30.3.1	win_c4p	128
8.30.3.2	win_c8p	129
8.31	3D windows	130
8.31.1	Detailed Description	130
8.31.2	Typedef Documentation	130
8.31.2.1	sphere3d	130
8.31.2.2	window3d	130
8.31.3	Function Documentation	131
8.31.3.1	win_c4p_3d	131
8.31.3.2	win_c8p_3d	131
8.32	N-D windows	132
8.32.1	Detailed Description	132
8.33	Multiple windows	133
8.33.1	Detailed Description	133
8.34	v2w2v functions	134
8.35	v2w_w2v functions	135
8.36	vv2b functions	136
9	Namespace Documentation	137
9.1	mln Namespace Reference	137
9.1.1	Detailed Description	159
9.1.2	Typedef Documentation	161
9.1.2.1	bin_1complex_image2d	161
9.1.2.2	bin_2complex_image3df	161
9.1.2.3	box1d	161
9.1.2.4	box2d	162
9.1.2.5	box2d_h	162
9.1.2.6	box3d	162
9.1.2.7	discrete_plane_1complex_geometry	162
9.1.2.8	discrete_plane_2complex_geometry	162
9.1.2.9	dpoint1d	162
9.1.2.10	dpoint2d	162
9.1.2.11	dpoint2d_h	162

9.1.2.12	dpoint3d	162
9.1.2.13	float_2complex_image3df	163
9.1.2.14	int_u8_1complex_image2d	163
9.1.2.15	int_u8_2complex_image2d	163
9.1.2.16	int_u8_2complex_image3df	163
9.1.2.17	p_run2d	163
9.1.2.18	p_runs2d	163
9.1.2.19	point1d	163
9.1.2.20	point1df	163
9.1.2.21	point2d	163
9.1.2.22	point2d_h	163
9.1.2.23	point2df	164
9.1.2.24	point3d	164
9.1.2.25	point3df	164
9.1.2.26	rgb8_2complex_image3df	164
9.1.2.27	space_2complex_geometry	164
9.1.2.28	unsigned_2complex_image3df	164
9.1.2.29	vec2d_d	164
9.1.2.30	vec2d_f	164
9.1.2.31	vec3d_d	164
9.1.2.32	vec3d_f	164
9.1.2.33	w_window1d_float	165
9.1.2.34	w_window1d_int	165
9.1.2.35	w_window2d_float	165
9.1.2.36	w_window2d_int	165
9.1.2.37	w_window3d_float	165
9.1.2.38	w_window3d_int	165
9.1.3	Function Documentation	165
9.1.3.1	a_point_of	165
9.1.3.2	apply_p2p	165
9.1.3.3	apply_p2p	165
9.1.3.4	compose	166
9.1.3.5	duplicate	166
9.1.3.6	extend	166
9.1.3.7	extend	166
9.1.3.8	extend	166

9.1.3.9	<code>implies</code>	167
9.1.3.10	<code>initialize</code>	167
9.1.3.11	<code>is_simple_2d</code>	167
9.1.3.12	<code>larger_than</code>	167
9.1.3.13	<code>make_debug_graph_image</code>	167
9.1.3.14	<code>mln_exact</code>	168
9.1.3.15	<code>mln_gen_complex_neighborhood</code>	168
9.1.3.16	<code>mln_gen_complex_neighborhood</code>	168
9.1.3.17	<code>mln_gen_complex_neighborhood</code>	168
9.1.3.18	<code>mln_gen_complex_neighborhood</code>	168
9.1.3.19	<code>mln_gen_complex_neighborhood</code>	168
9.1.3.20	<code>mln_gen_complex_neighborhood</code>	168
9.1.3.21	<code>mln_gen_complex_window</code>	169
9.1.3.22	<code>mln_gen_complex_window</code>	169
9.1.3.23	<code>mln_gen_complex_window</code>	169
9.1.3.24	<code>mln_gen_complex_window</code>	169
9.1.3.25	<code>mln_gen_complex_window</code>	169
9.1.3.26	<code>mln_gen_complex_window</code>	169
9.1.3.27	<code>mln_gen_complex_window_p</code>	169
9.1.3.28	<code>mln_gen_complex_window_p</code>	169
9.1.3.29	<code>mln_gen_complex_window_p</code>	170
9.1.3.30	<code>mln_gen_complex_window_p</code>	170
9.1.3.31	<code>mln_gen_complex_window_p</code>	170
9.1.3.32	<code>mln_gen_complex_window_p</code>	170
9.1.3.33	<code>mln_regular</code>	170
9.1.3.34	<code>mln_trait_op_geq</code>	170
9.1.3.35	<code>mln_trait_op_greater</code>	170
9.1.3.36	<code>mln_trait_op_leq</code>	171
9.1.3.37	<code>mln_trait_op_neq</code>	171
9.1.3.38	<code>operator"!="</code>	171
9.1.3.39	<code>operator"!="</code>	172
9.1.3.40	<code>operator*</code>	172
9.1.3.41	<code>operator++</code>	172
9.1.3.42	<code>operator-</code>	172
9.1.3.43	<code>operator-</code>	172
9.1.3.44	<code>operator-</code>	173

9.1.3.45	operator<	173
9.1.3.46	operator<	173
9.1.3.47	operator<	173
9.1.3.48	operator<<	173
9.1.3.49	operator<<	173
9.1.3.50	operator<<	174
9.1.3.51	operator<<	174
9.1.3.52	operator<=	174
9.1.3.53	operator<=	174
9.1.3.54	operator<=	174
9.1.3.55	operator<=	174
9.1.3.56	operator<=	175
9.1.3.57	operator==	175
9.1.3.58	operator==	175
9.1.3.59	operator==	175
9.1.3.60	operator==	175
9.1.3.61	operator==	175
9.1.3.62	operator==	176
9.1.3.63	operator==	176
9.1.3.64	operator"	176
9.1.3.65	operator"	176
9.1.3.66	operator"	177
9.1.3.67	operator"	177
9.1.3.68	operator"	177
9.1.3.69	operator"	177
9.1.3.70	primary	177
9.1.3.71	ptransform	177
9.1.4	Variable Documentation	177
9.1.4.1	before	177
9.1.4.2	sagittal_dec	178
9.1.4.3	up	178
9.2	mln::accu Namespace Reference	179
9.2.1	Detailed Description	180
9.2.2	Function Documentation	181
9.2.2.1	compute	181
9.2.2.2	line	181

9.2.2.3	mln_meta_accu_result	181
9.2.2.4	take	182
9.3	mln::accu::image Namespace Reference	183
9.3.1	Detailed Description	183
9.4	mln::accu::impl Namespace Reference	184
9.4.1	Detailed Description	184
9.5	mln::accu::logic Namespace Reference	185
9.5.1	Detailed Description	185
9.6	mln::accu::math Namespace Reference	186
9.6.1	Detailed Description	186
9.7	mln::accu::meta::logic Namespace Reference	187
9.7.1	Detailed Description	187
9.8	mln::accu::meta::math Namespace Reference	188
9.8.1	Detailed Description	188
9.9	mln::accu::meta::shape Namespace Reference	189
9.9.1	Detailed Description	189
9.10	mln::accu::meta::stat Namespace Reference	190
9.10.1	Detailed Description	190
9.11	mln::accu::shape Namespace Reference	191
9.11.1	Detailed Description	191
9.12	mln::accu::stat Namespace Reference	192
9.12.1	Detailed Description	193
9.13	mln::algebra Namespace Reference	194
9.13.1	Detailed Description	194
9.13.2	Function Documentation	194
9.13.2.1	ldlt_decomp	194
9.13.2.2	ldlt_solve	195
9.13.2.3	operator*	195
9.13.2.4	vprod	195
9.14	mln::arith Namespace Reference	196
9.14.1	Detailed Description	198
9.14.2	Function Documentation	198
9.14.2.1	diff_abs	198
9.14.2.2	div	198
9.14.2.3	div_cst	199
9.14.2.4	div_inplace	199

9.14.2.5	<code>min</code>	199
9.14.2.6	<code>min_inplace</code>	200
9.14.2.7	<code>minus</code>	200
9.14.2.8	<code>minus</code>	200
9.14.2.9	<code>minus_cst</code>	201
9.14.2.10	<code>minus_cst</code>	201
9.14.2.11	<code>minus_cst_inplace</code>	202
9.14.2.12	<code>minus_inplace</code>	202
9.14.2.13	<code>plus</code>	202
9.14.2.14	<code>plus</code>	203
9.14.2.15	<code>plus_cst</code>	204
9.14.2.16	<code>plus_cst</code>	204
9.14.2.17	<code>plus_cst_inplace</code>	204
9.14.2.18	<code>plus_inplace</code>	205
9.14.2.19	<code>revert</code>	205
9.14.2.20	<code>revert_inplace</code>	205
9.14.2.21	<code>times</code>	206
9.14.2.22	<code>times_cst</code>	206
9.14.2.23	<code>times_inplace</code>	206
9.15	<code>mln::arith::impl</code> Namespace Reference	208
9.15.1	Detailed Description	208
9.16	<code>mln::arith::impl::generic</code> Namespace Reference	209
9.16.1	Detailed Description	209
9.17	<code>mln::binarization</code> Namespace Reference	210
9.17.1	Detailed Description	210
9.17.2	Function Documentation	210
9.17.2.1	<code>binarization</code>	210
9.17.2.2	<code>threshold</code>	210
9.18	<code>mln::border</code> Namespace Reference	211
9.18.1	Detailed Description	211
9.18.2	Function Documentation	211
9.18.2.1	<code>adjust</code>	211
9.18.2.2	<code>duplicate</code>	212
9.18.2.3	<code>equalize</code>	212
9.18.2.4	<code>fill</code>	212
9.18.2.5	<code>find</code>	213

9.18.2.6	get	213
9.18.2.7	mirror	213
9.18.2.8	resize	214
9.19	mln::border::impl Namespace Reference	215
9.19.1	Detailed Description	215
9.20	mln::border::impl::generic Namespace Reference	216
9.20.1	Detailed Description	216
9.21	mln::canvas Namespace Reference	217
9.21.1	Detailed Description	217
9.21.2	Function Documentation	218
9.21.2.1	distance_front	218
9.21.2.2	distance_geodesic	218
9.22	mln::canvas::browsing Namespace Reference	219
9.22.1	Detailed Description	219
9.23	mln::canvas::impl Namespace Reference	220
9.23.1	Detailed Description	220
9.24	mln::canvas::labeling Namespace Reference	221
9.24.1	Detailed Description	221
9.24.2	Function Documentation	221
9.24.2.1	blobs	221
9.25	mln::canvas::labeling::impl Namespace Reference	222
9.25.1	Detailed Description	222
9.26	mln::canvas::morpho Namespace Reference	223
9.26.1	Detailed Description	223
9.27	mln::convert Namespace Reference	224
9.27.1	Detailed Description	226
9.27.2	Function Documentation	226
9.27.2.1	from_to	226
9.27.2.2	from_to	226
9.27.2.3	from_to	226
9.27.2.4	from_to	226
9.27.2.5	mln_image_from_grid	226
9.27.2.6	mln_image_from_grid	227
9.27.2.7	mln_image_from_grid	227
9.27.2.8	mln_image_from_grid	227
9.27.2.9	mln_window	227

9.27.2.10	to	227
9.27.2.11	to_dpoint	227
9.27.2.12	to_fun	227
9.27.2.13	to_fun	227
9.27.2.14	to_image	227
9.27.2.15	to_p_array	228
9.27.2.16	to_p_array	228
9.27.2.17	to_p_array	228
9.27.2.18	to_p_set	228
9.27.2.19	to_p_set	228
9.27.2.20	to_p_set	228
9.27.2.21	to_p_set	228
9.27.2.22	to_p_set	229
9.27.2.23	to_upper_window	229
9.27.2.24	to_upper_window	229
9.27.2.25	to_window	229
9.27.2.26	to_window	229
9.27.2.27	to_window	229
9.28	mln::data Namespace Reference	230
9.28.1	Detailed Description	232
9.28.2	Function Documentation	232
9.28.2.1	abs	232
9.28.2.2	abs_inplace	232
9.28.2.3	apply	232
9.28.2.4	compute	233
9.28.2.5	compute	233
9.28.2.6	convert	234
9.28.2.7	fast_median	234
9.28.2.8	fill	234
9.28.2.9	fill_with_image	235
9.28.2.10	fill_with_value	235
9.28.2.11	median	236
9.28.2.12	mln_meta_accu_result	236
9.28.2.13	paste	236
9.28.2.14	paste_without_localization	237
9.28.2.15	replace	237

9.28.2.16	saturate	237
9.28.2.17	saturate	238
9.28.2.18	saturate_inplace	238
9.28.2.19	sort_offsets_increasing	238
9.28.2.20	sort_psites_decreasing	238
9.28.2.21	sort_psites_increasing	239
9.28.2.22	stretch	239
9.28.2.23	to_enc	239
9.28.2.24	transform	240
9.28.2.25	transform	240
9.28.2.26	transform_inplace	241
9.28.2.27	transform_inplace	241
9.28.2.28	update	242
9.28.2.29	wrap	242
9.29	mln::data::approx Namespace Reference	243
9.29.1	Detailed Description	243
9.29.2	Function Documentation	243
9.29.2.1	median	243
9.29.2.2	median	243
9.29.2.3	median	244
9.30	mln::data::approx::impl Namespace Reference	245
9.30.1	Detailed Description	245
9.31	mln::data::impl Namespace Reference	246
9.31.1	Detailed Description	246
9.31.2	Function Documentation	246
9.31.2.1	stretch	246
9.31.2.2	transform_inplace_lowq	247
9.31.2.3	update_fastest	247
9.32	mln::data::impl::generic Namespace Reference	248
9.32.1	Detailed Description	249
9.32.2	Function Documentation	249
9.32.2.1	convert	249
9.32.2.2	fill_with_image	249
9.32.2.3	fill_with_value	249
9.32.2.4	median	250
9.32.2.5	paste	250

9.32.2.6	sort_offsets_increasing	250
9.32.2.7	transform	250
9.32.2.8	transform	251
9.32.2.9	transform_inplace	251
9.32.2.10	transform_inplace	251
9.32.2.11	update	251
9.33	mln::data::naive Namespace Reference	253
9.33.1	Detailed Description	253
9.33.2	Function Documentation	253
9.33.2.1	median	253
9.34	mln::data::naive::impl Namespace Reference	254
9.34.1	Detailed Description	254
9.35	mln::debug Namespace Reference	255
9.35.1	Detailed Description	256
9.35.2	Function Documentation	256
9.35.2.1	draw_graph	256
9.35.2.2	draw_graph	256
9.35.2.3	draw_graph	257
9.35.2.4	filename	257
9.35.2.5	format	257
9.35.2.6	format	257
9.35.2.7	format	257
9.35.2.8	format	257
9.35.2.9	iota	257
9.35.2.10	println	258
9.35.2.11	println	258
9.35.2.12	println_with_border	258
9.35.2.13	put_word	258
9.35.2.14	slices_2d	258
9.35.2.15	slices_2d	258
9.35.2.16	superpose	259
9.36	mln::debug::impl Namespace Reference	260
9.36.1	Detailed Description	260
9.37	mln::def Namespace Reference	261
9.37.1	Detailed Description	261
9.37.2	Typedef Documentation	261

9.37.2.1	coord	261
9.37.2.2	coordf	261
9.37.3	Enumeration Type Documentation	261
9.37.3.1	"@21	261
9.38	mln::display Namespace Reference	262
9.38.1	Detailed Description	262
9.39	mln::display::impl Namespace Reference	263
9.39.1	Detailed Description	263
9.40	mln::display::impl::generic Namespace Reference	264
9.40.1	Detailed Description	264
9.41	mln::doc Namespace Reference	265
9.41.1	Detailed Description	266
9.42	mln::draw Namespace Reference	267
9.42.1	Detailed Description	267
9.42.2	Function Documentation	267
9.42.2.1	box	267
9.42.2.2	line	267
9.42.2.3	plot	268
9.43	mln::estim Namespace Reference	269
9.43.1	Detailed Description	269
9.43.2	Function Documentation	269
9.43.2.1	mean	269
9.43.2.2	mean	270
9.43.2.3	min_max	270
9.43.2.4	sum	270
9.43.2.5	sum	270
9.44	mln::extension Namespace Reference	271
9.44.1	Detailed Description	271
9.44.2	Function Documentation	271
9.44.2.1	adjust	271
9.44.2.2	adjust	272
9.44.2.3	adjust	272
9.44.2.4	adjust	272
9.44.2.5	adjust_duplicate	272
9.44.2.6	adjust_fill	272
9.44.2.7	duplicate	272

9.44.2.8	fill	272
9.45	mln::fun Namespace Reference	274
9.45.1	Detailed Description	275
9.46	mln::fun::access Namespace Reference	276
9.46.1	Detailed Description	276
9.47	mln::fun::i2v Namespace Reference	277
9.47.1	Detailed Description	277
9.47.2	Function Documentation	277
9.47.2.1	operator<<	277
9.48	mln::fun::p2b Namespace Reference	278
9.48.1	Detailed Description	278
9.49	mln::fun::p2p Namespace Reference	279
9.49.1	Detailed Description	279
9.50	mln::fun::p2v Namespace Reference	280
9.50.1	Detailed Description	280
9.51	mln::fun::stat Namespace Reference	281
9.51.1	Detailed Description	281
9.52	mln::fun::v2b Namespace Reference	282
9.52.1	Detailed Description	282
9.53	mln::fun::v2i Namespace Reference	283
9.53.1	Detailed Description	283
9.54	mln::fun::v2v Namespace Reference	284
9.54.1	Detailed Description	284
9.54.2	Variable Documentation	285
9.54.2.1	f_hsi_to_rgb_3x8	285
9.54.2.2	f_hsl_to_rgb_3x8	285
9.54.2.3	f_rgb_to_hsi_f	285
9.54.2.4	f_rgb_to_hsl_f	285
9.55	mln::fun::v2w2v Namespace Reference	286
9.55.1	Detailed Description	286
9.56	mln::fun::v2w_w2v Namespace Reference	287
9.56.1	Detailed Description	287
9.57	mln::fun::vv2b Namespace Reference	288
9.57.1	Detailed Description	288
9.58	mln::fun::vv2v Namespace Reference	289
9.58.1	Detailed Description	289

9.59	mln::fun::x2p Namespace Reference	290
9.59.1	Detailed Description	290
9.60	mln::fun::x2v Namespace Reference	291
9.60.1	Detailed Description	291
9.61	mln::fun::x2x Namespace Reference	292
9.61.1	Detailed Description	292
9.62	mln::geom Namespace Reference	293
9.62.1	Detailed Description	296
9.62.2	Function Documentation	296
9.62.2.1	bbox	296
9.62.2.2	bbox	297
9.62.2.3	bbox	297
9.62.2.4	bbox	297
9.62.2.5	chamfer	297
9.62.2.6	delta	297
9.62.2.7	delta	297
9.62.2.8	delta	297
9.62.2.9	max_col	298
9.62.2.10	max_col	298
9.62.2.11	max_ind	298
9.62.2.12	max_row	298
9.62.2.13	max_row	298
9.62.2.14	max_sli	298
9.62.2.15	mesh_corner_point_area	298
9.62.2.16	mesh_curvature	299
9.62.2.17	mesh_normal	299
9.62.2.18	min_col	299
9.62.2.19	min_col	300
9.62.2.20	min_ind	300
9.62.2.21	min_row	300
9.62.2.22	min_row	300
9.62.2.23	min_sli	300
9.62.2.24	ncols	300
9.62.2.25	ncols	300
9.62.2.26	ninds	301
9.62.2.27	nrows	301

9.62.2.28	nrows	301
9.62.2.29	nsites	301
9.62.2.30	nslis	301
9.62.2.31	pmin_pmax	301
9.62.2.32	pmin_pmax	301
9.62.2.33	pmin_pmax	301
9.62.2.34	pmin_pmax	302
9.62.2.35	rotate	302
9.62.2.36	rotate	302
9.62.2.37	seeds2tiling	302
9.62.2.38	seeds2tiling_roundness	303
9.62.2.39	translate	303
9.62.2.40	translate	304
9.63	mln::geom::impl Namespace Reference	305
9.63.1	Detailed Description	305
9.63.2	Function Documentation	305
9.63.2.1	seeds2tiling	305
9.63.2.2	seeds2tiling_roundness	305
9.64	mln::graph Namespace Reference	307
9.64.1	Detailed Description	307
9.64.2	Function Documentation	307
9.64.2.1	compute	307
9.64.2.2	labeling	308
9.64.2.3	to_neighb	308
9.64.2.4	to_win	308
9.65	mln::grid Namespace Reference	310
9.65.1	Detailed Description	310
9.66	mln::histo Namespace Reference	311
9.66.1	Detailed Description	311
9.66.2	Function Documentation	311
9.66.2.1	compute	311
9.67	mln::histo::impl Namespace Reference	312
9.67.1	Detailed Description	312
9.68	mln::histo::impl::generic Namespace Reference	313
9.68.1	Detailed Description	313
9.69	mln::impl Namespace Reference	314

9.69.1 Detailed Description	314
9.70 mln::io Namespace Reference	315
9.70.1 Detailed Description	316
9.71 mln::io::cloud Namespace Reference	317
9.71.1 Detailed Description	317
9.71.2 Function Documentation	317
9.71.2.1 load	317
9.71.2.2 save	317
9.72 mln::io::dicom Namespace Reference	318
9.72.1 Detailed Description	318
9.72.2 Function Documentation	318
9.72.2.1 load	318
9.72.2.2 load	318
9.73 mln::io::dump Namespace Reference	319
9.73.1 Detailed Description	319
9.73.2 Function Documentation	319
9.73.2.1 load	319
9.73.2.2 save	319
9.74 mln::io::fits Namespace Reference	320
9.74.1 Detailed Description	320
9.74.2 Function Documentation	320
9.74.2.1 load	320
9.74.2.2 load	320
9.75 mln::io::fld Namespace Reference	321
9.75.1 Detailed Description	321
9.75.2 Function Documentation	321
9.75.2.1 load	321
9.75.2.2 read_header	321
9.75.2.3 write_header	322
9.76 mln::io::magick Namespace Reference	323
9.76.1 Detailed Description	323
9.76.2 Function Documentation	323
9.76.2.1 do_it	323
9.76.2.2 get_color	323
9.76.2.3 load	323
9.76.2.4 save	324

9.77	mln::io::off Namespace Reference	325
9.77.1	Detailed Description	325
9.77.2	Function Documentation	325
9.77.2.1	load	325
9.77.2.2	save	325
9.77.2.3	save_bin_alt	326
9.78	mln::io::pbm Namespace Reference	327
9.78.1	Detailed Description	327
9.78.2	Function Documentation	327
9.78.2.1	load	327
9.78.2.2	load	328
9.78.2.3	save	328
9.79	mln::io::pbm::impl Namespace Reference	329
9.79.1	Detailed Description	329
9.80	mln::io::pbms Namespace Reference	330
9.80.1	Detailed Description	330
9.80.2	Function Documentation	330
9.80.2.1	load	330
9.81	mln::io::pbms::impl Namespace Reference	331
9.81.1	Detailed Description	331
9.82	mln::io::pfm Namespace Reference	332
9.82.1	Detailed Description	332
9.82.2	Function Documentation	332
9.82.2.1	load	332
9.82.2.2	load	333
9.82.2.3	save	333
9.83	mln::io::pfm::impl Namespace Reference	334
9.83.1	Detailed Description	334
9.84	mln::io::pgm Namespace Reference	335
9.84.1	Detailed Description	335
9.84.2	Function Documentation	335
9.84.2.1	load	335
9.84.2.2	load	336
9.84.2.3	save	336
9.85	mln::io::pgms Namespace Reference	337
9.85.1	Detailed Description	337

9.85.2	Function Documentation	337
9.85.2.1	load	337
9.86	mln::io::plot Namespace Reference	338
9.86.1	Detailed Description	338
9.86.2	Function Documentation	338
9.86.2.1	load	338
9.86.2.2	save	338
9.86.2.3	save	339
9.87	mln::io::pnm Namespace Reference	340
9.87.1	Detailed Description	340
9.87.2	Function Documentation	340
9.87.2.1	load	340
9.87.2.2	load	341
9.87.2.3	load_ascii_builtin	341
9.87.2.4	load_ascii_value	341
9.87.2.5	load_raw_2d	341
9.87.2.6	max_component	341
9.87.2.7	save	341
9.88	mln::io::pnm::impl Namespace Reference	342
9.88.1	Detailed Description	342
9.89	mln::io::pnms Namespace Reference	343
9.89.1	Detailed Description	343
9.89.2	Function Documentation	343
9.89.2.1	load	343
9.90	mln::io::ppm Namespace Reference	344
9.90.1	Detailed Description	344
9.90.2	Function Documentation	344
9.90.2.1	load	344
9.90.2.2	load	345
9.90.2.3	save	345
9.91	mln::io::ppms Namespace Reference	346
9.91.1	Detailed Description	346
9.91.2	Function Documentation	346
9.91.2.1	load	346
9.92	mln::io::tiff Namespace Reference	347
9.92.1	Detailed Description	347

9.92.2	Function Documentation	347
9.92.2.1	load	347
9.93	mln::io::txt Namespace Reference	348
9.93.1	Detailed Description	348
9.93.2	Function Documentation	348
9.93.2.1	save	348
9.94	mln::labeling Namespace Reference	349
9.94.1	Detailed Description	351
9.94.2	Function Documentation	351
9.94.2.1	background	351
9.94.2.2	blobs	352
9.94.2.3	blobs_and_compute	352
9.94.2.4	colorize	353
9.94.2.5	compute	353
9.94.2.6	compute	354
9.94.2.7	compute	354
9.94.2.8	compute	355
9.94.2.9	compute	355
9.94.2.10	compute_image	356
9.94.2.11	compute_image	356
9.94.2.12	compute_image	357
9.94.2.13	fill_holes	357
9.94.2.14	flat_zones	357
9.94.2.15	foreground	358
9.94.2.16	pack	358
9.94.2.17	pack_inplace	359
9.94.2.18	regional_maxima	359
9.94.2.19	regional_minima	359
9.94.2.20	relabel	360
9.94.2.21	relabel	360
9.94.2.22	relabel_inplace	360
9.94.2.23	relabel_inplace	361
9.94.2.24	superpose	361
9.94.2.25	value	361
9.94.2.26	wrap	362
9.94.2.27	wrap	362

9.95	<code>mln::labeling::impl</code> Namespace Reference	363
9.95.1	Detailed Description	363
9.96	<code>mln::labeling::impl::generic</code> Namespace Reference	364
9.96.1	Detailed Description	364
9.96.2	Function Documentation	364
9.96.2.1	<code>compute</code>	364
9.96.2.2	<code>compute</code>	365
9.96.2.3	<code>compute</code>	365
9.97	<code>mln::linear</code> Namespace Reference	366
9.97.1	Detailed Description	366
9.97.2	Function Documentation	367
9.97.2.1	<code>gaussian</code>	367
9.97.2.2	<code>gaussian</code>	367
9.97.2.3	<code>gaussian_1st_derivative</code>	367
9.97.2.4	<code>gaussian_1st_derivative</code>	367
9.97.2.5	<code>gaussian_2nd_derivative</code>	368
9.97.2.6	<code>gaussian_2nd_derivative</code>	368
9.97.2.7	<code>mln_ch_convolve</code>	368
9.97.2.8	<code>mln_ch_convolve</code>	368
9.97.2.9	<code>mln_ch_convolve_grad</code>	369
9.98	<code>mln::linear::impl</code> Namespace Reference	370
9.98.1	Detailed Description	370
9.99	<code>mln::linear::local</code> Namespace Reference	371
9.99.1	Detailed Description	371
9.99.2	Function Documentation	371
9.99.2.1	<code>convolve</code>	371
9.99.2.2	<code>convolve</code>	371
9.100	<code>mln::linear::local::impl</code> Namespace Reference	372
9.100.1	Detailed Description	372
9.101	<code>mln::literal</code> Namespace Reference	373
9.101.1	Detailed Description	376
9.101.2	Variable Documentation	376
9.101.2.1	<code>black</code>	376
9.101.2.2	<code>blue</code>	376
9.101.2.3	<code>brown</code>	376
9.101.2.4	<code>cyan</code>	376

9.101.2.5	dark_gray	376
9.101.2.6	green	376
9.101.2.7	identity	376
9.101.2.8	light_gray	376
9.101.2.9	lime	377
9.101.2.10	magenta	377
9.101.2.11	lmax	377
9.101.2.12	medium_gray	377
9.101.2.13	min	377
9.101.2.14	olive	377
9.101.2.15	one	377
9.101.2.16	orange	377
9.101.2.17	origin	377
9.101.2.18	pink	377
9.101.2.19	purple	377
9.101.2.20	red	378
9.101.2.21	teal	378
9.101.2.22	violet	378
9.101.2.23	white	378
9.101.2.24	yellow	378
9.101.2.25	zero	378
9.102	mln::logical Namespace Reference	379
9.102.1	Detailed Description	379
9.102.2	Function Documentation	379
9.102.2.1	and_inplace	379
9.102.2.2	and_not	380
9.102.2.3	and_not_inplace	380
9.102.2.4	not_inplace	380
9.102.2.5	or_inplace	381
9.102.2.6	xor_inplace	381
9.103	mln::logical::impl Namespace Reference	382
9.103.1	Detailed Description	382
9.104	mln::logical::impl::generic Namespace Reference	383
9.104.1	Detailed Description	383
9.105	mln::make Namespace Reference	384
9.105.1	Detailed Description	389

9.105.2 Function Documentation	389
9.105.2.1 attachment	389
9.105.2.2 box1d	389
9.105.2.3 box1d	389
9.105.2.4 box2d	390
9.105.2.5 box2d	390
9.105.2.6 box2d_h	391
9.105.2.7 box2d_h	391
9.105.2.8 box3d	391
9.105.2.9 box3d	392
9.105.2.10 cell	392
9.105.2.11 couple	393
9.105.2.12 detachment	393
9.105.2.13 dpoint2d_h	393
9.105.2.14 dummy_p_edges	393
9.105.2.15 dummy_p_edges	394
9.105.2.16 dummy_p_vertices	394
9.105.2.17 dummy_p_vertices	394
9.105.2.18 edge_image	395
9.105.2.19 edge_image	395
9.105.2.20 edge_image	395
9.105.2.21 edge_image	396
9.105.2.22 edge_image	396
9.105.2.23 edge_image	396
9.105.2.24 h_mat	396
9.105.2.25 image	397
9.105.2.26 image	397
9.105.2.27 image	397
9.105.2.28 image2d	397
9.105.2.29 image3d	398
9.105.2.30 image3d	398
9.105.2.31 influence_zone_adjacency_graph	398
9.105.2.32 mat	398
9.105.2.33 ord_pair	399
9.105.2.34 p_edges_with_mass_centers	399
9.105.2.35 p_vertices_with_mass_centers	399

9.105.2.36	pix	400
9.105.2.37	pixel	400
9.105.2.38	pixel	400
9.105.2.39	point2d_h	400
9.105.2.40	rag_and_labeled_wsl	400
9.105.2.41	region_adjacency_graph	401
9.105.2.42	relabelfun	401
9.105.2.43	relabelfun	402
9.105.2.44	vec	402
9.105.2.45	vec	403
9.105.2.46	vec	403
9.105.2.47	vec	403
9.105.2.48	vertex_image	403
9.105.2.49	vertex_image	404
9.105.2.50	voronoi	404
9.105.2.51	w_window	404
9.105.2.52	w_window1d	405
9.105.2.53	w_window1d_int	405
9.105.2.54	w_window2d	405
9.105.2.55	w_window2d_int	406
9.105.2.56	w_window3d	406
9.105.2.57	w_window3d_int	406
9.105.2.58	w_window_directional	407
9.106	mln::math Namespace Reference	408
9.106.1	Detailed Description	408
9.106.2	Function Documentation	408
9.106.2.1	abs	408
9.106.2.2	abs	408
9.106.2.3	abs	408
9.107	mln::metal Namespace Reference	409
9.107.1	Detailed Description	409
9.108	mln::metal::impl Namespace Reference	410
9.108.1	Detailed Description	410
9.109	mln::metal::math Namespace Reference	411
9.109.1	Detailed Description	411
9.110	mln::metal::math::impl Namespace Reference	412

9.110.1 Detailed Description	412
9.111 <code>mln::morpho</code> Namespace Reference	413
9.111.1 Detailed Description	415
9.111.2 Function Documentation	416
9.111.2.1 <code>complementation</code>	416
9.111.2.2 <code>complementation_inplace</code>	416
9.111.2.3 <code>contrast</code>	416
9.111.2.4 <code>dilation</code>	416
9.111.2.5 <code>erosion</code>	416
9.111.2.6 <code>general</code>	417
9.111.2.7 <code>gradient</code>	417
9.111.2.8 <code>gradient_external</code>	417
9.111.2.9 <code>gradient_internal</code>	417
9.111.2.10 <code>hit_or_miss</code>	417
9.111.2.11 <code>hit_or_miss_background_closing</code>	417
9.111.2.12 <code>hit_or_miss_background_opening</code>	418
9.111.2.13 <code>hit_or_miss_closing</code>	418
9.111.2.14 <code>hit_or_miss_opening</code>	418
9.111.2.15 <code>laplacian</code>	418
9.111.2.16 <code>line_gradient</code>	418
9.111.2.17 <code>meyer_wst</code>	419
9.111.2.18 <code>meyer_wst</code>	419
9.111.2.19 <code>min</code>	419
9.111.2.20 <code>min_inplace</code>	419
9.111.2.21 <code>minus</code>	420
9.111.2.22 <code>plus</code>	420
9.111.2.23 <code>rank_filter</code>	420
9.111.2.24 <code>thick_miss</code>	420
9.111.2.25 <code>thickening</code>	420
9.111.2.26 <code>thin_fit</code>	421
9.111.2.27 <code>thinning</code>	421
9.111.2.28 <code>top_hat_black</code>	421
9.111.2.29 <code>top_hat_self_complementary</code>	421
9.111.2.30 <code>top_hat_white</code>	421
9.112 <code>mln::morpho::approx</code> Namespace Reference	422
9.112.1 Detailed Description	422

9.113	mln::morpho::attribute Namespace Reference	423
9.113.1	Detailed Description	423
9.114	mln::morpho::closing::approx Namespace Reference	424
9.114.1	Detailed Description	424
9.114.2	Function Documentation	424
9.114.2.1	structural	424
9.115	mln::morpho::elementary Namespace Reference	425
9.115.1	Detailed Description	425
9.115.2	Function Documentation	425
9.115.2.1	closing	425
9.115.2.2	mln_trait_op_minus_twice	426
9.115.2.3	opening	426
9.115.2.4	top_hat_black	426
9.115.2.5	top_hat_self_complementary	426
9.115.2.6	top_hat_white	426
9.116	mln::morpho::impl Namespace Reference	427
9.116.1	Detailed Description	427
9.117	mln::morpho::impl::generic Namespace Reference	428
9.117.1	Detailed Description	428
9.117.2	Function Documentation	428
9.117.2.1	hit_or_miss	428
9.117.2.2	rank_filter	428
9.118	mln::morpho::opening::approx Namespace Reference	429
9.118.1	Detailed Description	429
9.118.2	Function Documentation	429
9.118.2.1	structural	429
9.119	mln::morpho::reconstruction Namespace Reference	430
9.119.1	Detailed Description	430
9.120	mln::morpho::reconstruction::by_dilation Namespace Reference	431
9.120.1	Detailed Description	431
9.121	mln::morpho::reconstruction::by_erosion Namespace Reference	432
9.121.1	Detailed Description	432
9.122	mln::morpho::tree Namespace Reference	433
9.122.1	Detailed Description	434
9.122.2	Function Documentation	434
9.122.2.1	compute_attribute_image	434

9.122.2.2	compute_attribute_image_from	435
9.122.2.3	compute_parent	435
9.122.2.4	dual_input_max_tree	436
9.122.2.5	max_tree	436
9.122.2.6	min_tree	437
9.122.2.7	propagate_if	437
9.122.2.8	propagate_if_value	437
9.122.2.9	propagate_node_to_ancestors	438
9.122.2.10	propagate_node_to_ancestors	438
9.122.2.11	propagate_node_to_descendants	438
9.122.2.12	propagate_node_to_descendants	439
9.122.2.13	propagate_representative	439
9.123	mln::morpho::tree::filter Namespace Reference	440
9.123.1	Detailed Description	440
9.123.2	Function Documentation	440
9.123.2.1	direct	440
9.123.2.2	filter	441
9.123.2.3	max	441
9.123.2.4	min	441
9.123.2.5	subtractive	441
9.124	mln::morpho::watershed Namespace Reference	443
9.124.1	Detailed Description	443
9.124.2	Function Documentation	443
9.124.2.1	flooding	443
9.124.2.2	flooding	444
9.124.2.3	superpose	444
9.124.2.4	superpose	444
9.124.2.5	topological	445
9.125	mln::morpho::watershed::watershed Namespace Reference	446
9.125.1	Detailed Description	446
9.126	mln::morpho::watershed::watershed::generic Namespace Reference	447
9.126.1	Detailed Description	447
9.127	mln::norm Namespace Reference	448
9.127.1	Detailed Description	449
9.127.2	Function Documentation	449
9.127.2.1	ll	449

9.127.2.2 l1_distance	449
9.127.2.3 l2	449
9.127.2.4 l2_distance	449
9.127.2.5 linfty	449
9.127.2.6 linfty_distance	449
9.127.2.7 sqr_l2	449
9.128mln::norm::impl Namespace Reference	450
9.128.1 Detailed Description	450
9.129mln::opt Namespace Reference	451
9.129.1 Detailed Description	451
9.129.2 Function Documentation	451
9.129.2.1 at	451
9.129.2.2 at	452
9.129.2.3 at	452
9.129.2.4 at	452
9.129.2.5 at	452
9.129.2.6 at	452
9.130mln::opt::impl Namespace Reference	453
9.130.1 Detailed Description	453
9.131mln::pw Namespace Reference	454
9.131.1 Detailed Description	454
9.132mln::registration Namespace Reference	455
9.132.1 Detailed Description	455
9.132.2 Function Documentation	456
9.132.2.1 get_rot	456
9.132.2.2 icp	456
9.132.2.3 icp	456
9.132.2.4 registration1	457
9.132.2.5 registration2	457
9.132.2.6 registration3	457
9.133mln::select Namespace Reference	458
9.133.1 Detailed Description	458
9.134mln::set Namespace Reference	459
9.134.1 Detailed Description	459
9.134.2 Function Documentation	459
9.134.2.1 card	459

9.134.2.2	compute	460
9.134.2.3	compute_with_weights	460
9.134.2.4	compute_with_weights	460
9.134.2.5	get	461
9.134.2.6	has	461
9.134.2.7	mln_meta_accu_result	461
9.134.2.8	mln_meta_accu_result	461
9.135mln::	subsampling Namespace Reference	462
9.135.1	Detailed Description	462
9.135.2	Function Documentation	462
9.135.2.1	gaussian_subsampling	462
9.135.2.2	subsampling	462
9.136mln::	tag Namespace Reference	463
9.136.1	Detailed Description	463
9.137mln::	test Namespace Reference	464
9.137.1	Detailed Description	464
9.137.2	Function Documentation	464
9.137.2.1	positive	464
9.137.2.2	predicate	464
9.137.2.3	predicate	465
9.137.2.4	predicate	465
9.138mln::	test::impl Namespace Reference	466
9.138.1	Detailed Description	466
9.139mln::	topo Namespace Reference	467
9.139.1	Detailed Description	471
9.139.2	Function Documentation	471
9.139.2.1	detach	471
9.139.2.2	edge	471
9.139.2.3	is_facet	472
9.139.2.4	make_algebraic_face	472
9.139.2.5	make_algebraic_n_face	472
9.139.2.6	operator"!="	472
9.139.2.7	operator"!="	472
9.139.2.8	operator"!="	472
9.139.2.9	operator"!="	473
9.139.2.10	operator+	473

9.139.2.1	operator-	473
9.139.2.12	operator-	473
9.139.2.13	operator-	473
9.139.2.14	operator<	473
9.139.2.15	operator<	474
9.139.2.16	operator<	474
9.139.2.17	operator<	474
9.139.2.18	operator<<	474
9.139.2.19	operator<<	474
9.139.2.20	operator<<	474
9.139.2.21	operator<<	475
9.139.2.22	operator<<	475
9.139.2.23	operator==	475
9.139.2.24	operator==	475
9.139.2.25	operator==	475
9.139.2.26	operator==	475
9.139.2.27	operator==	476
9.140	mln::trace Namespace Reference	477
9.140.1	Detailed Description	477
9.141	mln::trait Namespace Reference	478
9.141.1	Detailed Description	478
9.142	mln::transform Namespace Reference	479
9.142.1	Detailed Description	480
9.142.2	Function Documentation	480
9.142.2.1	distance_and_closest_point_geodesic	480
9.142.2.2	distance_and_closest_point_geodesic	480
9.142.2.3	distance_and_influence_zone_geodesic	481
9.142.2.4	distance_front	481
9.142.2.5	distance_geodesic	481
9.142.2.6	hough	482
9.142.2.7	influence_zone_front	482
9.142.2.8	influence_zone_front	482
9.142.2.9	influence_zone_geodesic	482
9.142.2.10	influence_zone_geodesic_saturated	483
9.143	mln::util Namespace Reference	484
9.143.1	Detailed Description	487

9.143.2 Typedef Documentation	487
9.143.2.1 vertex_id_t	487
9.143.3 Function Documentation	487
9.143.3.1 display_branch	487
9.143.3.2 display_tree	488
9.143.3.3 lemmings	488
9.143.3.4 make_greater_point	488
9.143.3.5 make_greater_psite	488
9.143.3.6 operator<	488
9.143.3.7 operator<<	489
9.143.3.8 operator<<	489
9.143.3.9 operator==	489
9.143.3.10operator==	489
9.143.3.11ord_strict	489
9.143.3.12ord_weak	489
9.143.3.13tree_fast_to_image	489
9.143.3.14tree_to_fast	490
9.143.3.15tree_to_image	490
9.144mln::util::impl Namespace Reference	491
9.144.1 Detailed Description	491
9.144.2 Function Documentation	491
9.144.2.1 tree_fast_to_image	491
9.145mln::value Namespace Reference	492
9.145.1 Detailed Description	496
9.145.2 Typedef Documentation	496
9.145.2.1 float01_16	496
9.145.2.2 float01_8	496
9.145.2.3 gl16	496
9.145.2.4 gl8	497
9.145.2.5 glf	497
9.145.2.6 int_s16	497
9.145.2.7 int_s32	497
9.145.2.8 int_s8	497
9.145.2.9 int_u12	497
9.145.2.10int_u16	497
9.145.2.11int_u32	497

9.145.2.12	int_u8	497
9.145.2.13	label_16	497
9.145.2.14	label_32	497
9.145.2.15	label_8	498
9.145.2.16	rgb16	498
9.145.2.17	rgb8	498
9.145.3	Function Documentation	498
9.145.3.1	cast	498
9.145.3.2	equiv	498
9.145.3.3	operator*	498
9.145.3.4	operator*	498
9.145.3.5	operator+	498
9.145.3.6	operator+	498
9.145.3.7	operator-	499
9.145.3.8	operator-	499
9.145.3.9	operator/	499
9.145.3.10	operator/	499
9.145.3.11	operator<<	499
9.145.3.12	operator<<	499
9.145.3.13	operator<<	499
9.145.3.14	operator<<	500
9.145.3.15	operator<<	500
9.145.3.16	operator<<	500
9.145.3.17	operator<<	501
9.145.3.18	operator<<	501
9.145.3.19	operator<<	501
9.145.3.20	operator<<	501
9.145.3.21	operator<<	501
9.145.3.22	operator==	501
9.145.3.23	operator==	502
9.145.3.24	other	502
9.145.3.25	stack	502
9.146	mln::value::impl Namespace Reference	503
9.146.1	Detailed Description	503
9.147	mln::win Namespace Reference	504
9.147.1	Detailed Description	505

9.147.2 Function Documentation	505
9.147.2.1 diff	505
9.147.2.2 mln_regular	505
9.147.2.3 mln_regular	506
9.147.2.4 sym	506
9.147.2.5 sym	506
10 Class Documentation	507
10.1 mln::accu::center< P, V > Struct Template Reference	507
10.1.1 Detailed Description	507
10.1.2 Member Function Documentation	508
10.1.2.1 init	508
10.1.2.2 is_valid	508
10.1.2.3 take_as_init	508
10.1.2.4 take_n_times	508
10.1.2.5 to_result	508
10.2 mln::accu::convolve< T1, T2, R > Struct Template Reference	509
10.2.1 Detailed Description	509
10.2.2 Member Function Documentation	509
10.2.2.1 init	509
10.2.2.2 is_valid	509
10.2.2.3 take_as_init	510
10.2.2.4 take_n_times	510
10.2.2.5 to_result	510
10.3 mln::accu::count_adjacent_vertices< F, S > Struct Template Reference	511
10.3.1 Detailed Description	511
10.3.2 Member Function Documentation	511
10.3.2.1 init	511
10.3.2.2 is_valid	512
10.3.2.3 set_value	512
10.3.2.4 take_as_init	512
10.3.2.5 take_n_times	512
10.3.2.6 to_result	512
10.4 mln::accu::count_labels< L > Struct Template Reference	513
10.4.1 Detailed Description	513
10.4.2 Member Function Documentation	513
10.4.2.1 init	513

10.4.2.2	is_valid	513
10.4.2.3	set_value	514
10.4.2.4	take_as_init	514
10.4.2.5	take_n_times	514
10.4.2.6	to_result	514
10.5	mln::accu::count_value< V > Struct Template Reference	515
10.5.1	Detailed Description	515
10.5.2	Member Function Documentation	515
10.5.2.1	init	515
10.5.2.2	is_valid	515
10.5.2.3	set_value	516
10.5.2.4	take_as_init	516
10.5.2.5	take_n_times	516
10.5.2.6	to_result	516
10.6	mln::accu::histo< V > Struct Template Reference	517
10.6.1	Detailed Description	517
10.6.2	Member Function Documentation	517
10.6.2.1	is_valid	517
10.6.2.2	take	517
10.6.2.3	take_as_init	518
10.6.2.4	take_n_times	518
10.6.2.5	vect	518
10.7	mln::accu::label_used< L > Struct Template Reference	519
10.7.1	Detailed Description	519
10.7.2	Member Function Documentation	519
10.7.2.1	init	519
10.7.2.2	is_valid	519
10.7.2.3	take	520
10.7.2.4	take_as_init	520
10.7.2.5	take_n_times	520
10.7.2.6	to_result	520
10.8	mln::accu::logic::land Struct Reference	521
10.8.1	Detailed Description	521
10.8.2	Member Function Documentation	521
10.8.2.1	init	521
10.8.2.2	is_valid	521

10.8.2.3	take_as_init	521
10.8.2.4	take_n_times	522
10.8.2.5	to_result	522
10.9	mln::accu::logic::land_basic Struct Reference	523
10.9.1	Detailed Description	523
10.9.2	Member Function Documentation	523
10.9.2.1	can_stop	523
10.9.2.2	init	523
10.9.2.3	is_valid	524
10.9.2.4	take_as_init	524
10.9.2.5	take_n_times	524
10.9.2.6	to_result	524
10.10	mln::accu::logic::lor Struct Reference	525
10.10.1	Detailed Description	525
10.10.2	Member Function Documentation	525
10.10.2.1	init	525
10.10.2.2	is_valid	525
10.10.2.3	take_as_init	525
10.10.2.4	take_n_times	526
10.10.2.5	to_result	526
10.11	mln::accu::logic::lor_basic Struct Reference	527
10.11.1	Detailed Description	527
10.11.2	Member Function Documentation	527
10.11.2.1	can_stop	527
10.11.2.2	init	527
10.11.2.3	is_valid	528
10.11.2.4	take_as_init	528
10.11.2.5	take_n_times	528
10.11.2.6	to_result	528
10.12	mln::accu::maj_h< T > Struct Template Reference	529
10.12.1	Detailed Description	529
10.12.2	Member Function Documentation	529
10.12.2.1	init	529
10.12.2.2	is_valid	529
10.12.2.3	take_as_init	530
10.12.2.4	take_n_times	530

10.12.2.5 to_result	530
10.13 <code>mln::accu::math::count< T ></code> Struct Template Reference	531
10.13.1 Detailed Description	531
10.13.2 Member Function Documentation	531
10.13.2.1 init	531
10.13.2.2 is_valid	531
10.13.2.3 set_value	532
10.13.2.4 take_as_init	532
10.13.2.5 take_n_times	532
10.13.2.6 to_result	532
10.14 <code>mln::accu::math::inf< T ></code> Struct Template Reference	533
10.14.1 Detailed Description	533
10.14.2 Member Function Documentation	533
10.14.2.1 init	533
10.14.2.2 is_valid	533
10.14.2.3 take_as_init	534
10.14.2.4 take_n_times	534
10.14.2.5 to_result	534
10.15 <code>mln::accu::math::sum< T, S ></code> Struct Template Reference	535
10.15.1 Detailed Description	535
10.15.2 Member Function Documentation	535
10.15.2.1 init	535
10.15.2.2 is_valid	535
10.15.2.3 take_as_init	536
10.15.2.4 take_n_times	536
10.15.2.5 to_result	536
10.16 <code>mln::accu::math::sup< T ></code> Struct Template Reference	537
10.16.1 Detailed Description	537
10.16.2 Member Function Documentation	537
10.16.2.1 init	537
10.16.2.2 is_valid	537
10.16.2.3 take_as_init	538
10.16.2.4 take_n_times	538
10.16.2.5 to_result	538
10.17 <code>mln::accu::max_site< I ></code> Struct Template Reference	539
10.17.1 Detailed Description	539

10.17.2 Member Function Documentation	539
10.17.2.1 init	539
10.17.2.2 is_valid	539
10.17.2.3 take_as_init	540
10.17.2.4 take_n_times	540
10.17.2.5 to_result	540
10.18 mln::accu::meta::center Struct Reference	541
10.18.1 Detailed Description	541
10.19 mln::accu::meta::count_adjacent_vertices Struct Reference	542
10.19.1 Detailed Description	542
10.20 mln::accu::meta::count_labels Struct Reference	543
10.20.1 Detailed Description	543
10.21 mln::accu::meta::count_value Struct Reference	544
10.21.1 Detailed Description	544
10.22 mln::accu::meta::histo Struct Reference	545
10.22.1 Detailed Description	545
10.23 mln::accu::meta::label_used Struct Reference	546
10.23.1 Detailed Description	546
10.24 mln::accu::meta::logic::land Struct Reference	547
10.24.1 Detailed Description	547
10.25 mln::accu::meta::logic::land_basic Struct Reference	548
10.25.1 Detailed Description	548
10.26 mln::accu::meta::logic::lor Struct Reference	549
10.26.1 Detailed Description	549
10.27 mln::accu::meta::logic::lor_basic Struct Reference	550
10.27.1 Detailed Description	550
10.28 mln::accu::meta::maj_h Struct Reference	551
10.28.1 Detailed Description	551
10.29 mln::accu::meta::math::count Struct Reference	552
10.29.1 Detailed Description	552
10.30 mln::accu::meta::math::inf Struct Reference	553
10.30.1 Detailed Description	553
10.31 mln::accu::meta::math::sum Struct Reference	554
10.31.1 Detailed Description	554
10.32 mln::accu::meta::math::sup Struct Reference	555
10.32.1 Detailed Description	555

10.33	mln::accu::meta::max_site Struct Reference	556
10.33.1	Detailed Description	556
10.34	mln::accu::meta::nil Struct Reference	557
10.34.1	Detailed Description	557
10.35	mln::accu::meta::p< mA > Struct Template Reference	558
10.35.1	Detailed Description	558
10.36	mln::accu::meta::pair< A1, A2 > Struct Template Reference	559
10.36.1	Detailed Description	559
10.37	mln::accu::meta::rms Struct Reference	560
10.37.1	Detailed Description	560
10.38	mln::accu::meta::shape::bbox Struct Reference	561
10.38.1	Detailed Description	561
10.39	mln::accu::meta::shape::height Struct Reference	562
10.39.1	Detailed Description	562
10.40	mln::accu::meta::shape::volume Struct Reference	563
10.40.1	Detailed Description	563
10.41	mln::accu::meta::stat::max Struct Reference	564
10.41.1	Detailed Description	564
10.42	mln::accu::meta::stat::max_h Struct Reference	565
10.42.1	Detailed Description	565
10.43	mln::accu::meta::stat::mean Struct Reference	566
10.43.1	Detailed Description	566
10.44	mln::accu::meta::stat::median_alt< T > Struct Template Reference	567
10.44.1	Detailed Description	567
10.45	mln::accu::meta::stat::median_h Struct Reference	568
10.45.1	Detailed Description	568
10.46	mln::accu::meta::stat::min Struct Reference	569
10.46.1	Detailed Description	569
10.47	mln::accu::meta::stat::min_h Struct Reference	570
10.47.1	Detailed Description	570
10.48	mln::accu::meta::stat::rank Struct Reference	571
10.48.1	Detailed Description	571
10.49	mln::accu::meta::stat::rank_high_quant Struct Reference	572
10.49.1	Detailed Description	572
10.50	mln::accu::meta::tuple< n, > Struct Template Reference	573
10.50.1	Detailed Description	573

10.51	<code>mln::accu::meta::val< mA ></code> Struct Template Reference	574
10.51.1	Detailed Description	574
10.52	<code>mln::accu::nil< T ></code> Struct Template Reference	575
10.52.1	Detailed Description	575
10.52.2	Member Function Documentation	575
10.52.2.1	<code>init</code>	575
10.52.2.2	<code>is_valid</code>	575
10.52.2.3	<code>take_as_init</code>	576
10.52.2.4	<code>take_n_times</code>	576
10.52.2.5	<code>to_result</code>	576
10.53	<code>mln::accu::p< A ></code> Struct Template Reference	577
10.53.1	Detailed Description	577
10.53.2	Member Function Documentation	577
10.53.2.1	<code>init</code>	577
10.53.2.2	<code>is_valid</code>	577
10.53.2.3	<code>take_as_init</code>	578
10.53.2.4	<code>take_n_times</code>	578
10.53.2.5	<code>to_result</code>	578
10.54	<code>mln::accu::pair< A1, A2, T ></code> Struct Template Reference	579
10.54.1	Detailed Description	579
10.54.2	Member Function Documentation	579
10.54.2.1	<code>init</code>	579
10.54.2.2	<code>is_valid</code>	580
10.54.2.3	<code>take_as_init</code>	580
10.54.2.4	<code>take_n_times</code>	580
10.54.2.5	<code>to_result</code>	580
10.55	<code>mln::accu::rms< T, V ></code> Struct Template Reference	581
10.55.1	Detailed Description	581
10.55.2	Member Function Documentation	581
10.55.2.1	<code>init</code>	581
10.55.2.2	<code>is_valid</code>	581
10.55.2.3	<code>take_as_init</code>	582
10.55.2.4	<code>take_n_times</code>	582
10.55.2.5	<code>to_result</code>	582
10.56	<code>mln::accu::shape::bbox< P ></code> Struct Template Reference	583
10.56.1	Detailed Description	583

10.56.2 Member Function Documentation	583
10.56.2.1 init	583
10.56.2.2 is_valid	583
10.56.2.3 take_as_init	584
10.56.2.4 take_n_times	584
10.56.2.5 to_result	584
10.57mln::accu::shape::height< I > Struct Template Reference	585
10.57.1 Detailed Description	585
10.57.2 Member Typedef Documentation	586
10.57.2.1 argument	586
10.57.2.2 value	586
10.57.3 Member Function Documentation	586
10.57.3.1 init	586
10.57.3.2 is_valid	586
10.57.3.3 set_value	586
10.57.3.4 take_as_init	586
10.57.3.5 take_n_times	586
10.57.3.6 to_result	587
10.58mln::accu::shape::volume< I > Struct Template Reference	588
10.58.1 Detailed Description	588
10.58.2 Member Typedef Documentation	589
10.58.2.1 argument	589
10.58.2.2 value	589
10.58.3 Member Function Documentation	589
10.58.3.1 init	589
10.58.3.2 is_valid	589
10.58.3.3 set_value	589
10.58.3.4 take_as_init	589
10.58.3.5 take_n_times	589
10.58.3.6 to_result	590
10.59mln::accu::site_set::rectangularity< P > Class Template Reference	591
10.59.1 Detailed Description	591
10.59.2 Constructor & Destructor Documentation	591
10.59.2.1 rectangularity	591
10.59.3 Member Function Documentation	592
10.59.3.1 area	592

10.59.3.2	bbox	592
10.59.3.3	take_as_init	592
10.59.3.4	take_n_times	592
10.59.3.5	to_result	592
10.60	mln::accu::stat::deviation< T, S, M > Struct Template Reference	593
10.60.1	Detailed Description	593
10.60.2	Member Function Documentation	593
10.60.2.1	init	593
10.60.2.2	is_valid	594
10.60.2.3	take_as_init	594
10.60.2.4	take_n_times	594
10.60.2.5	to_result	594
10.61	mln::accu::stat::max< T > Struct Template Reference	595
10.61.1	Detailed Description	595
10.61.2	Member Function Documentation	595
10.61.2.1	init	595
10.61.2.2	is_valid	595
10.61.2.3	set_value	596
10.61.2.4	take_as_init	596
10.61.2.5	take_n_times	596
10.61.2.6	to_result	596
10.62	mln::accu::stat::max_h< V > Struct Template Reference	597
10.62.1	Detailed Description	597
10.62.2	Member Function Documentation	597
10.62.2.1	init	597
10.62.2.2	is_valid	597
10.62.2.3	take_as_init	598
10.62.2.4	take_n_times	598
10.62.2.5	to_result	598
10.63	mln::accu::stat::mean< T, S, M > Struct Template Reference	599
10.63.1	Detailed Description	599
10.63.2	Member Function Documentation	599
10.63.2.1	count	599
10.63.2.2	init	600
10.63.2.3	is_valid	600
10.63.2.4	sum	600

10.63.2.5	take_as_init	600
10.63.2.6	take_n_times	600
10.63.2.7	to_result	600
10.64	mln::accu::stat::median_alt< S > Struct Template Reference	601
10.64.1	Detailed Description	601
10.64.2	Member Function Documentation	601
10.64.2.1	is_valid	601
10.64.2.2	take	602
10.64.2.3	take_as_init	602
10.64.2.4	take_n_times	602
10.64.2.5	to_result	602
10.65	mln::accu::stat::median_h< V > Struct Template Reference	603
10.65.1	Detailed Description	603
10.65.2	Member Function Documentation	603
10.65.2.1	init	603
10.65.2.2	is_valid	604
10.65.2.3	take_as_init	604
10.65.2.4	take_n_times	604
10.65.2.5	to_result	604
10.66	mln::accu::stat::meta::deviation Struct Reference	605
10.66.1	Detailed Description	605
10.67	mln::accu::stat::min< T > Struct Template Reference	606
10.67.1	Detailed Description	606
10.67.2	Member Function Documentation	606
10.67.2.1	init	606
10.67.2.2	is_valid	606
10.67.2.3	set_value	607
10.67.2.4	take_as_init	607
10.67.2.5	take_n_times	607
10.67.2.6	to_result	607
10.68	mln::accu::stat::min_h< V > Struct Template Reference	608
10.68.1	Detailed Description	608
10.68.2	Member Function Documentation	608
10.68.2.1	init	608
10.68.2.2	is_valid	608
10.68.2.3	take_as_init	609

10.68.2.4	take_n_times	609
10.68.2.5	to_result	609
10.69	mln::accu::stat::min_max< V > Struct Template Reference	610
10.69.1	Detailed Description	610
10.69.2	Member Function Documentation	611
10.69.2.1	init	611
10.69.2.2	is_valid	611
10.69.2.3	take_as_init	611
10.69.2.4	take_n_times	611
10.69.2.5	to_result	611
10.70	mln::accu::stat::rank< T > Struct Template Reference	612
10.70.1	Detailed Description	612
10.70.2	Member Function Documentation	612
10.70.2.1	init	612
10.70.2.2	is_valid	612
10.70.2.3	k	613
10.70.2.4	take_as_init	613
10.70.2.5	take_n_times	613
10.70.2.6	to_result	613
10.71	mln::accu::stat::rank< bool > Struct Template Reference	614
10.71.1	Detailed Description	614
10.71.2	Member Function Documentation	614
10.71.2.1	init	614
10.71.2.2	is_valid	614
10.71.2.3	take_as_init	615
10.71.2.4	take_n_times	615
10.71.2.5	to_result	615
10.72	mln::accu::stat::rank_high_quant< T > Struct Template Reference	616
10.72.1	Detailed Description	616
10.72.2	Member Function Documentation	616
10.72.2.1	init	616
10.72.2.2	is_valid	616
10.72.2.3	take_as_init	617
10.72.2.4	take_n_times	617
10.72.2.5	to_result	617
10.73	mln::accu::stat::var< T > Struct Template Reference	618

10.73.1 Detailed Description	618
10.73.2 Member Typedef Documentation	619
10.73.2.1 mean_t	619
10.73.3 Member Function Documentation	619
10.73.3.1 init	619
10.73.3.2 is_valid	619
10.73.3.3 mean	619
10.73.3.4 n_items	619
10.73.3.5 take_as_init	619
10.73.3.6 take_n_times	619
10.73.3.7 to_result	620
10.73.3.8 variance	620
10.74mln::accu::stat::variance< T, S, R > Struct Template Reference	621
10.74.1 Detailed Description	621
10.74.2 Member Function Documentation	622
10.74.2.1 init	622
10.74.2.2 is_valid	622
10.74.2.3 mean	622
10.74.2.4 n_items	622
10.74.2.5 standard_deviation	622
10.74.2.6 sum	622
10.74.2.7 take_as_init	622
10.74.2.8 take_n_times	623
10.74.2.9 to_result	623
10.74.2.10var	623
10.75mln::accu::tuple< A, n, > Struct Template Reference	624
10.75.1 Detailed Description	624
10.75.2 Member Function Documentation	624
10.75.2.1 init	624
10.75.2.2 is_valid	624
10.75.2.3 take_as_init	625
10.75.2.4 take_n_times	625
10.75.2.5 to_result	625
10.76mln::accu::val< A > Struct Template Reference	626
10.76.1 Detailed Description	626
10.76.2 Member Function Documentation	626

10.76.2.1	init	626
10.76.2.2	is_valid	626
10.76.2.3	take_as_init	627
10.76.2.4	take_n_times	627
10.76.2.5	to_result	627
10.77	mln::Accumulator< E > Struct Template Reference	628
10.77.1	Detailed Description	628
10.77.2	Member Function Documentation	628
10.77.2.1	take_as_init	628
10.77.2.2	take_n_times	628
10.78	mln::algebra::h_mat< d, T > Struct Template Reference	629
10.78.1	Detailed Description	629
10.78.2	Member Enumeration Documentation	629
10.78.2.1	"@7	629
10.78.3	Constructor & Destructor Documentation	629
10.78.3.1	h_mat	629
10.78.3.2	h_mat	630
10.78.4	Member Function Documentation	630
10.78.4.1	_1	630
10.78.4.2	t	630
10.79	mln::algebra::h_vec< d, C > Struct Template Reference	631
10.79.1	Detailed Description	631
10.79.2	Member Enumeration Documentation	632
10.79.2.1	"@8	632
10.79.3	Constructor & Destructor Documentation	632
10.79.3.1	h_vec	632
10.79.3.2	h_vec	632
10.79.4	Member Function Documentation	632
10.79.4.1	operator mat< n, 1, U >	632
10.79.4.2	t	632
10.79.4.3	to_vec	632
10.79.5	Member Data Documentation	632
10.79.5.1	origin	632
10.79.5.2	zero	632
10.80	mln::bkd_pixterId< I > Class Template Reference	633
10.80.1	Detailed Description	633

10.80.2 Member Typedef Documentation	633
10.80.2.1 image	633
10.80.3 Constructor & Destructor Documentation	633
10.80.3.1 bkd_pixter1d	633
10.80.4 Member Function Documentation	634
10.80.4.1 next	634
10.81 mln::bkd_pixter2d< I > Class Template Reference	635
10.81.1 Detailed Description	635
10.81.2 Member Typedef Documentation	635
10.81.2.1 image	635
10.81.3 Constructor & Destructor Documentation	635
10.81.3.1 bkd_pixter2d	635
10.81.4 Member Function Documentation	636
10.81.4.1 next	636
10.82 mln::bkd_pixter3d< I > Class Template Reference	637
10.82.1 Detailed Description	637
10.82.2 Member Typedef Documentation	637
10.82.2.1 image	637
10.82.3 Constructor & Destructor Documentation	637
10.82.3.1 bkd_pixter3d	637
10.82.4 Member Function Documentation	638
10.82.4.1 next	638
10.83 mln::box< P > Struct Template Reference	639
10.83.1 Detailed Description	642
10.83.2 Member Typedef Documentation	642
10.83.2.1 bkd_piter	642
10.83.2.2 element	642
10.83.2.3 fwd_piter	642
10.83.2.4 piter	642
10.83.2.5 psite	642
10.83.2.6 site	642
10.83.3 Member Enumeration Documentation	642
10.83.3.1 "@31	642
10.83.4 Constructor & Destructor Documentation	642
10.83.4.1 box	642
10.83.4.2 box	643

10.83.4.3	box	643
10.83.5	Member Function Documentation	643
10.83.5.1	bbox	643
10.83.5.2	center	643
10.83.5.3	crop_wrt	643
10.83.5.4	enlarge	643
10.83.5.5	enlarge	644
10.83.5.6	has	644
10.83.5.7	is_empty	644
10.83.5.8	is_valid	644
10.83.5.9	len	644
10.83.5.10	memory_size	644
10.83.5.11	nsites	645
10.83.5.12	pmax	645
10.83.5.13	pmax	645
10.83.5.14	pmin	645
10.83.5.15	pmin	645
10.83.5.16	to_larger	645
10.83.6	Friends And Related Function Documentation	645
10.83.6.1	diff	645
10.83.6.2	inter	646
10.83.6.3	operator<	646
10.83.6.4	operator<	646
10.83.6.5	operator<<	646
10.83.6.6	operator<<	646
10.83.6.7	operator<=	647
10.83.6.8	operator<=	647
10.83.6.9	operator==	647
10.83.6.10	sym_diff	647
10.83.6.11	luni	647
10.83.6.12	unique	647
10.84	mln::Box< E > Struct Template Reference	648
10.84.1	Detailed Description	649
10.84.2	Member Function Documentation	649
10.84.2.1	bbox	649
10.84.2.2	is_empty	650

10.84.2.3 len	650
10.84.2.4 nsites	650
10.84.3 Friends And Related Function Documentation	650
10.84.3.1 diff	650
10.84.3.2 inter	650
10.84.3.3 operator<	650
10.84.3.4 operator<	651
10.84.3.5 operator<<	651
10.84.3.6 operator<=	651
10.84.3.7 operator<=	651
10.84.3.8 operator==	652
10.84.3.9 sym_diff	652
10.84.3.10 uni	652
10.84.3.11 unique	652
10.85mln::box_runend_piter< P > Class Template Reference	653
10.85.1 Detailed Description	653
10.85.2 Constructor & Destructor Documentation	653
10.85.2.1 box_runend_piter	653
10.85.3 Member Function Documentation	653
10.85.3.1 next	653
10.85.3.2 run_length	654
10.86mln::box_runstart_piter< P > Class Template Reference	655
10.86.1 Detailed Description	655
10.86.2 Constructor & Destructor Documentation	655
10.86.2.1 box_runstart_piter	655
10.86.3 Member Function Documentation	655
10.86.3.1 next	655
10.86.3.2 run_length	656
10.87mln::Browsing< E > Struct Template Reference	657
10.87.1 Detailed Description	657
10.88mln::canvas::browsing::backdiagonal2d_t Struct Reference	658
10.88.1 Detailed Description	658
10.89mln::canvas::browsing::breadth_first_search_t Struct Reference	659
10.89.1 Detailed Description	659
10.90mln::canvas::browsing::depth_first_search_t Struct Reference	660
10.90.1 Detailed Description	660

10.91	<code>mln::canvas::browsing::diagonal2d_t</code> Struct Reference	661
10.91.1	Detailed Description	661
10.92	<code>mln::canvas::browsing::dir_struct_elt_incr_update_t</code> Struct Reference	662
10.92.1	Detailed Description	662
10.93	<code>mln::canvas::browsing::directional_t</code> Struct Reference	664
10.93.1	Detailed Description	664
10.94	<code>mln::canvas::browsing::fwd_t</code> Struct Reference	666
10.94.1	Detailed Description	666
10.95	<code>mln::canvas::browsing::hyper_directional_t</code> Struct Reference	667
10.95.1	Detailed Description	667
10.96	<code>mln::canvas::browsing::snake_fwd_t</code> Struct Reference	668
10.96.1	Detailed Description	668
10.97	<code>mln::canvas::browsing::snake_generic_t</code> Struct Reference	669
10.97.1	Detailed Description	669
10.98	<code>mln::canvas::browsing::snake_vert_t</code> Struct Reference	670
10.98.1	Detailed Description	670
10.99	<code>mln::canvas::chamfer< F ></code> Struct Template Reference	671
10.99.1	Detailed Description	671
10.100	<code>mln::category< R(*) (A) ></code> Struct Template Reference	672
10.100.	Detailed Description	672
10.101	<code>mln::complex_image< D, G, V ></code> Class Template Reference	673
10.101.	Detailed Description	674
10.101.2	Member Typedef Documentation	674
10.101.2.1	<code>geom</code>	674
10.101.2.2	<code>value</code>	674
10.101.2.3	<code>value</code>	674
10.101.2.4	<code>skeleton</code>	674
10.101.2.5	<code>value</code>	674
10.101.3	Constructor & Destructor Documentation	674
10.101.3.1	<code>complex_image</code>	674
10.101.4	Member Function Documentation	675
10.101.4.1	<code>domain</code>	675
10.101.4.2	<code>operator()</code>	675
10.101.4.3	<code>operator()</code>	675
10.101.4.4	<code>values</code>	675
10.101.5	Member Data Documentation	675

10.101.5. <code>ldim</code>	675
10.102. <code>mln::complex_neighborhood_bkd_piter< I, G, N ></code> Class Template Reference	676
10.102. Detailed Description	676
10.102. Member Typedef Documentation	676
10.102.2. <code>liter_type</code>	676
10.102.2. <code>psite</code>	677
10.102. Constructor & Destructor Documentation	677
10.102.3. <code>lcomplex_neighborhood_bkd_piter</code>	677
10.102. Member Function Documentation	677
10.102.4. <code>liter</code>	677
10.102.4. <code>next</code>	677
10.103. <code>mln::complex_neighborhood_fwd_piter< I, G, N ></code> Class Template Reference	678
10.103. Detailed Description	678
10.103. Member Typedef Documentation	678
10.103.2. <code>liter_type</code>	678
10.103.2. <code>psite</code>	679
10.103. Constructor & Destructor Documentation	679
10.103.3. <code>lcomplex_neighborhood_fwd_piter</code>	679
10.103. Member Function Documentation	679
10.103.4. <code>liter</code>	679
10.103.4. <code>next</code>	679
10.104. <code>mln::complex_psite< D, G ></code> Class Template Reference	680
10.104. Detailed Description	680
10.104. Constructor & Destructor Documentation	681
10.104.2. <code>lcomplex_psite</code>	681
10.104.2. <code>complex_psite</code>	681
10.104. Member Function Documentation	681
10.104.3. <code>lchange_target</code>	681
10.104.3. <code>face</code>	681
10.104.3. <code>face_id</code>	681
10.104.3. <code>invalidate</code>	681
10.104.3. <code>is_valid</code>	682
10.104.3. <code>in</code>	682
10.104.3. <code>site_set</code>	682
10.105. <code>mln::complex_window_bkd_piter< I, G, W ></code> Class Template Reference	683
10.105. Detailed Description	683

10.105.2	Member Typedef Documentation	683
10.105.2.1	liter_type	683
10.105.2.2	psite	683
10.105.3	Constructor & Destructor Documentation	684
10.105.3.1	lcomplex_window_bkd_piter	684
10.105.4	Member Function Documentation	684
10.105.4.1	liter	684
10.105.4.2	next	684
10.106	ln::complex_window_fwd_piter< I, G, W > Class Template Reference	685
10.106.1	Detailed Description	685
10.106.2	Member Typedef Documentation	685
10.106.2.1	liter_type	685
10.106.2.2	psite	685
10.106.3	Constructor & Destructor Documentation	686
10.106.3.1	lcomplex_window_fwd_piter	686
10.106.4	Member Function Documentation	686
10.106.4.1	liter	686
10.106.4.2	next	686
10.107	ln::decorated_image< I, D > Struct Template Reference	687
10.107.1	Detailed Description	688
10.107.2	Member Typedef Documentation	688
10.107.2.1	lvalue	688
10.107.2.2	psite	688
10.107.2.3	rvalue	688
10.107.2.4	skeleton	688
10.107.3	Constructor & Destructor Documentation	688
10.107.3.1	ldecorated_image	688
10.107.3.2	~decorated_image	688
10.107.4	Member Function Documentation	688
10.107.4.1	ldecoration	688
10.107.4.2	decoration	689
10.107.4.3	operator decorated_image< const I, D >	689
10.107.4.4	operator()	689
10.107.4.5	operator()	689
10.108	ln::Delta_Point_Site< E > Struct Template Reference	690
10.108.1	Detailed Description	690

10.109	<code>ln::Delta_Point_Site< void ></code> Struct Template Reference	691
10.109	Detailed Description	691
10.110	<code>ln::doc::Accumulator< E ></code> Struct Template Reference	692
10.110	Detailed Description	692
10.110	Member Typedef Documentation	692
10.110.2	<code>largument</code>	692
10.110	Member Function Documentation	692
10.110.3	<code>limit</code>	692
10.110.3	<code>take</code>	692
10.110.3	<code>take</code>	693
10.111	<code>ln::doc::Box< E ></code> Struct Template Reference	694
10.111	Detailed Description	695
10.111	Member Typedef Documentation	695
10.111.2	<code>lbkd_piter</code>	695
10.111.2	<code>2fwd_piter</code>	695
10.111.2	<code>3psite</code>	695
10.111.2	<code>4site</code>	695
10.111	Member Function Documentation	695
10.111.3	<code>lbbox</code>	695
10.111.3	<code>2has</code>	695
10.111.3	<code>3nsites</code>	696
10.111.3	<code>4pmax</code>	696
10.111.3	<code>5pmin</code>	696
10.112	<code>ln::doc::Dpoint< E ></code> Struct Template Reference	697
10.112	Detailed Description	697
10.112	Member Typedef Documentation	697
10.112.2	<code>lcoord</code>	697
10.112.2	<code>2dpoint</code>	698
10.112.2	<code>3point</code>	698
10.112	Member Enumeration Documentation	698
10.112.3	<code>l"@19</code>	698
10.112	Member Function Documentation	698
10.112.4	<code>loperator[</code>	698
10.113	<code>ln::doc::Fastest_Image< E ></code> Struct Template Reference	699
10.113	Detailed Description	701
10.113	Member Typedef Documentation	701

10.113.2.1	<code>bkd_piter</code>	701
10.113.2.2	<code>coord</code>	701
10.113.2.3	<code>dpoint</code>	701
10.113.2.4	<code>fd_piter</code>	701
10.113.2.5	<code>value</code>	701
10.113.2.6	<code>point</code>	702
10.113.2.7	<code>pset</code>	702
10.113.2.8	<code>psite</code>	702
10.113.2.9	<code>rvalue</code>	702
10.113.2.10	<code>skeleton</code>	702
10.113.2.11	<code>value</code>	702
10.113.2.12	<code>set</code>	702
10.113.3	Member Function Documentation	703
10.113.3.1	<code>bbox</code>	703
10.113.3.2	<code>border</code>	703
10.113.3.3	<code>buffer</code>	703
10.113.3.4	<code>delta_index</code>	703
10.113.3.5	<code>domain</code>	703
10.113.3.6	<code>has</code>	704
10.113.3.7	<code>has</code>	704
10.113.3.8	<code>is_valid</code>	704
10.113.3.9	<code>elements</code>	704
10.113.3.10	<code>sites</code>	704
10.113.3.11	<code>operator()</code>	704
10.113.3.12	<code>operator()</code>	705
10.113.3.13	<code>operator[]</code>	705
10.113.3.14	<code>operator[]</code>	705
10.113.3.15	<code>point_at_index</code>	706
10.113.3.16	<code>values</code>	706
10.114	<code>nl::doc::Generalized_Pixel< E ></code> Struct Template Reference	707
10.114.1	Detailed Description	707
10.114.2	Member Typedef Documentation	707
10.114.2.1	<code>image</code>	707
10.114.2.2	<code>rvalue</code>	708
10.114.2.3	<code>value</code>	708
10.114.3	Member Function Documentation	708

10.114.3.lima	708
10.114.3.2val	708
10.115. Inn::doc::Image< E > Struct Template Reference	709
10.115.1. Detailed Description	710
10.115.2. Member Typedef Documentation	711
10.115.2.1. bkd_piter	711
10.115.2.2. coord	711
10.115.2.3. dpoint	711
10.115.2.4. fwd_piter	711
10.115.2.5. value	711
10.115.2.6. point	711
10.115.2.7. pset	711
10.115.2.8. psite	712
10.115.2.9. rvalue	712
10.115.2.10. skeleton	712
10.115.2.11. value	712
10.115.2.12. set	712
10.115.3. Member Function Documentation	712
10.115.3.1. bbox	712
10.115.3.2. domain	712
10.115.3.3. has	713
10.115.3.4. has	713
10.115.3.5. is_valid	713
10.115.3.6. sites	713
10.115.3.7. operator()	713
10.115.3.8. operator()	714
10.115.3.9. values	714
10.116. Inn::doc::Iterator< E > Struct Template Reference	715
10.116.1. Detailed Description	715
10.116.2. Member Function Documentation	715
10.116.2.1. invalidate	715
10.116.2.2. is_valid	715
10.116.2.3. start	716
10.117. Inn::doc::Neighborhood< E > Struct Template Reference	717
10.117.1. Detailed Description	717
10.117.2. Member Typedef Documentation	717

10.117.2.1bkd_niter	717
10.117.2.2dpoint	717
10.117.2.3fwd_niter	718
10.117.2.4niter	718
10.117.2.5point	718
10.118. In::doc::Object< E > Struct Template Reference	719
10.118. Detailed Description	719
10.119. In::doc::Pixel_Iterator< E > Struct Template Reference	720
10.119. Detailed Description	721
10.119. Member Typedef Documentation	721
10.119.2. limage	721
10.119.2.2value	721
10.119.2.3rvalue	721
10.119.2.4value	721
10.119.3. Member Function Documentation	721
10.119.3.1ima	721
10.119.3.2invalidate	721
10.119.3.3is_valid	721
10.119.3.4start	722
10.119.3.5val	722
10.120. In::doc::Point_Site< E > Struct Template Reference	723
10.120. Detailed Description	723
10.120. Member Typedef Documentation	723
10.120.2. lcoord	723
10.120.2.2dpoint	723
10.120.2.3mesh	724
10.120.2.4point	724
10.120.3. Member Enumeration Documentation	724
10.120.3.1"@20	724
10.120.4. Member Function Documentation	724
10.120.4. loperator[.	724
10.120.4.2to_point	725
10.121. In::doc::Site_Iterator< E > Struct Template Reference	726
10.121. Detailed Description	726
10.121. Member Typedef Documentation	727
10.121.2. lpsite	727

10.121.3	Member Function Documentation	727
10.121.3.1	invalidate	727
10.121.3.2	is_valid	727
10.121.3.3	operator psite	727
10.121.3.4	start	727
10.121	in::doc::Site_Set< E > Struct Template Reference	728
10.122	Detailed Description	728
10.122.2	Member Typedef Documentation	729
10.122.2.1	l_bkd_piter	729
10.122.2.2	2fwd_piter	729
10.122.2.3	psite	729
10.122.2.4	site	729
10.122.3	Member Function Documentation	729
10.122.3.1	lhas	729
10.121	in::doc::Value_Iterator< E > Struct Template Reference	730
10.123	Detailed Description	730
10.123.2	Member Typedef Documentation	731
10.123.2.1	lvalue	731
10.123.3	Member Function Documentation	731
10.123.3.1	invalidate	731
10.123.3.2	is_valid	731
10.123.3.3	operator value	731
10.123.3.4	start	731
10.121	in::doc::Value_Set< E > Struct Template Reference	732
10.124	Detailed Description	732
10.124.2	Member Typedef Documentation	733
10.124.2.1	l_bkd_viter	733
10.124.2.2	2fwd_viter	733
10.124.2.3	value	733
10.124.3	Member Function Documentation	733
10.124.3.1	lhas	733
10.124.3.2	index_of	733
10.124.3.3	nvalues	733
10.124.3.4	operator[733
10.121	in::doc::Weighted_Window< E > Struct Template Reference	734
10.125	Detailed Description	735

10.125.2	Member Typedef Documentation	735
10.125.2.1	l <code>bkd_qiter</code>	735
10.125.2.2	d <code>point</code>	735
10.125.2.3	f <code>wd_qiter</code>	735
10.125.2.4	p <code>oint</code>	735
10.125.2.5	w <code>eight</code>	735
10.125.2.6	w <code>indow</code>	735
10.125.3	Member Function Documentation	735
10.125.3.1	d <code>elta</code>	735
10.125.3.2	s <code>_centered</code>	736
10.125.3.3	s <code>_empty</code>	736
10.125.3.4	s <code>ym</code>	736
10.125.3.5	w <code>in</code>	736
10.126	Window< E > Struct Template Reference	737
10.126.1	Detailed Description	737
10.126.2	Member Typedef Documentation	737
10.126.2.1	l <code>bkd_qiter</code>	737
10.126.2.2	f <code>wd_qiter</code>	737
10.126.2.3	q <code>iter</code>	737
10.127	Dpoint< E > Struct Template Reference	738
10.127.1	Detailed Description	738
10.127.2	Member Function Documentation	738
10.127.2.1	l <code>to_dpoint</code>	738
10.128	Gpoint< G, C > Struct Template Reference	739
10.128.1	Detailed Description	740
10.128.2	Member Typedef Documentation	740
10.128.2.1	l <code>coord</code>	740
10.128.2.2	g <code>rid</code>	740
10.128.2.3	p <code>site</code>	740
10.128.2.4	s <code>ite</code>	740
10.128.2.5	v <code>ec</code>	741
10.128.3	Member Enumeration Documentation	741
10.128.3.1	"@22	741
10.128.4	Constructor & Destructor Documentation	741
10.128.4.1	l <code>dpoint</code>	741
10.128.4.2	d <code>point</code>	741

10.128.4.3dpoint	741
10.128.4.4dpoint	741
10.128.4.5dpoint	741
10.128.5Member Function Documentation	742
10.128.5.1operator mln::algebra::vec< dpoint< G, C >::dim, Q >	742
10.128.5.2operator[742
10.128.5.3operator[742
10.128.5.4set_all	742
10.128.5.5to_vec	742
10.129mln::dpoints_bkd_pixter< I > Class Template Reference	744
10.129.1Detailed Description	745
10.129.2Constructor & Destructor Documentation	745
10.129.2.1dpoints_bkd_pixter	745
10.129.2.2dpoints_bkd_pixter	745
10.129.3Member Function Documentation	745
10.129.3.1center_val	745
10.129.3.2invalidate	745
10.129.3.3is_valid	745
10.129.3.4next	746
10.129.3.5start	746
10.129.3.6update	746
10.130mln::dpoints_fwd_pixter< I > Class Template Reference	747
10.130.1Detailed Description	748
10.130.2Constructor & Destructor Documentation	748
10.130.2.1dpoints_fwd_pixter	748
10.130.2.2dpoints_fwd_pixter	748
10.130.3Member Function Documentation	748
10.130.3.1center_val	748
10.130.3.2invalidate	748
10.130.3.3is_valid	748
10.130.3.4next	749
10.130.3.5start	749
10.130.3.6update	749
10.131mln::dpsites_bkd_piter< V > Class Template Reference	750
10.131.1Detailed Description	750
10.131.2Constructor & Destructor Documentation	750

10.131.2.1	dpsites_bkd_piter	750
10.131.2.2	dpsites_bkd_piter	750
10.131.3	Member Function Documentation	751
10.131.3.1	next	751
10.132	mln::dpsites_fwd_piter< V > Class Template Reference	752
10.132.1	Detailed Description	752
10.132.2	Constructor & Destructor Documentation	752
10.132.2.1	dpsites_fwd_piter	752
10.132.2.2	dpsites_fwd_piter	752
10.132.3	Member Function Documentation	753
10.132.3.1	next	753
10.133	mln::Edge< E > Struct Template Reference	754
10.133.1	Detailed Description	754
10.134	mln::edge_image< P, V, G > Class Template Reference	755
10.134.1	Detailed Description	755
10.134.2	Member Typedef Documentation	756
10.134.2.1	edge_nbh_t	756
10.134.2.2	edge_win_t	756
10.134.2.3	graph_t	756
10.134.2.4	nbh_t	756
10.134.2.5	site_function_t	756
10.134.2.6	skeleton	756
10.134.2.7	win_t	756
10.134.3	Constructor & Destructor Documentation	756
10.134.3.1	edge_image	756
10.134.4	Member Function Documentation	757
10.134.4.1	operator()	757
10.135	mln::extended< I > Struct Template Reference	758
10.135.1	Detailed Description	758
10.135.2	Member Typedef Documentation	758
10.135.2.1	skeleton	758
10.135.2.2	value	758
10.135.3	Constructor & Destructor Documentation	759
10.135.3.1	extended	759
10.135.3.2	extended	759
10.135.4	Member Function Documentation	759

10.135.4.1domain	759
10.136.1.1ln::extension_fun< I, F > Class Template Reference	760
10.136.1.1.1Detailed Description	760
10.136.1.1.2Member Typedef Documentation	761
10.136.1.1.2.1rvalue	761
10.136.1.1.2.2skeleton	761
10.136.1.1.2.3value	761
10.136.1.1.3Constructor & Destructor Documentation	761
10.136.1.1.3.1extension_fun	761
10.136.1.1.3.2extension_fun	761
10.136.1.1.4Member Function Documentation	761
10.136.1.1.4.1extension	761
10.136.1.1.4.2has	761
10.136.1.1.4.3operator()	761
10.136.1.1.4.4operator()	762
10.137.1.1ln::extension_ima< I, J > Class Template Reference	763
10.137.1.1.1Detailed Description	763
10.137.1.1.2Member Typedef Documentation	764
10.137.1.1.2.1rvalue	764
10.137.1.1.2.2skeleton	764
10.137.1.1.2.3value	764
10.137.1.1.3Constructor & Destructor Documentation	764
10.137.1.1.3.1extension_ima	764
10.137.1.1.3.2extension_ima	764
10.137.1.1.4Member Function Documentation	764
10.137.1.1.4.1extension	764
10.137.1.1.4.2has	764
10.137.1.1.4.3operator()	764
10.137.1.1.4.4operator()	765
10.138.1.1ln::extension_val< I > Class Template Reference	766
10.138.1.1.1Detailed Description	766
10.138.1.1.2Member Typedef Documentation	767
10.138.1.1.2.1rvalue	767
10.138.1.1.2.2skeleton	767
10.138.1.1.2.3value	767
10.138.1.1.3Constructor & Destructor Documentation	767

10.138.3.1extension_val	767
10.138.3.2extension_val	767
10.138.4Member Function Documentation	767
10.138.4.1change_extension	767
10.138.4.2extension	767
10.138.4.3has	767
10.138.4.4operator()	767
10.138.4.5operator()	768
10.139Inl::faces_psite< N, D, P > Class Template Reference	769
10.139.Detailed Description	769
10.139.Constructor & Destructor Documentation	770
10.139.2.1faces_psite	770
10.139.2.2faces_psite	770
10.139.3Member Function Documentation	770
10.139.3.1change_target	770
10.139.3.2face	770
10.139.3.3face_id	770
10.139.3.4invalidate	770
10.139.3.5is_valid	770
10.139.3.6	771
10.139.3.7site_set	771
10.140Inl::flat_image< T, S > Struct Template Reference	772
10.140.Detailed Description	772
10.140.2Member Typedef Documentation	773
10.140.2.1lvalue	773
10.140.2.2rvalue	773
10.140.2.3skeleton	773
10.140.2.4value	773
10.140.3Constructor & Destructor Documentation	773
10.140.3.1flat_image	773
10.140.3.2flat_image	773
10.140.4Member Function Documentation	773
10.140.4.1domain	773
10.140.4.2has	773
10.140.4.3operator()	773
10.140.4.4operator()	774

10.141	<code>ln::fun::from_accu< A ></code> Struct Template Reference	775
	10.141. Detailed Description	775
10.142	<code>ln::fun::p2b::antilogy</code> Struct Reference	776
	10.142. Detailed Description	776
10.143	<code>ln::fun::p2b::tautology</code> Struct Reference	777
	10.143. Detailed Description	777
10.144	<code>ln::fun::v2b::lnot< V ></code> Struct Template Reference	778
	10.144. Detailed Description	778
10.145	<code>ln::fun::v2b::threshold< V ></code> Struct Template Reference	779
	10.145. Detailed Description	779
10.146	<code>ln::fun::v2v::ch_function_value< F, V ></code> Class Template Reference	780
	10.146. Detailed Description	780
10.147	<code>ln::fun::v2v::component< T, i ></code> Struct Template Reference	781
	10.147. Detailed Description	781
10.148	<code>ln::fun::v2v::l1_norm< V, R ></code> Struct Template Reference	782
	10.148. Detailed Description	782
10.149	<code>ln::fun::v2v::l2_norm< V, R ></code> Struct Template Reference	783
	10.149. Detailed Description	783
10.150	<code>ln::fun::v2v::linear< V, T, R ></code> Struct Template Reference	784
	10.150. Detailed Description	784
10.151	<code>ln::fun::v2v::linfty_norm< V, R ></code> Struct Template Reference	785
	10.151. Detailed Description	785
10.152	<code>ln::fun::v2w2v::cos< V ></code> Struct Template Reference	786
	10.152. Detailed Description	786
10.153	<code>ln::fun::v2w_w2v::l1_norm< V, R ></code> Struct Template Reference	787
	10.153. Detailed Description	787
10.154	<code>ln::fun::v2w_w2v::l2_norm< V, R ></code> Struct Template Reference	788
	10.154. Detailed Description	788
10.155	<code>ln::fun::v2w_w2v::linfty_norm< V, R ></code> Struct Template Reference	789
	10.155. Detailed Description	789
10.156	<code>ln::fun::vv2b::eq< L, R ></code> Struct Template Reference	790
	10.156. Detailed Description	790
10.157	<code>ln::fun::vv2b::ge< L, R ></code> Struct Template Reference	791
	10.157. Detailed Description	791
10.158	<code>ln::fun::vv2b::gt< L, R ></code> Struct Template Reference	792
	10.158. Detailed Description	792

10.159	<code>ln::fun::vv2b::implies< L, R ></code> Struct Template Reference	793
	10.159. Detailed Description	793
10.160	<code>ln::fun::vv2b::le< L, R ></code> Struct Template Reference	794
	10.160. Detailed Description	794
10.161	<code>ln::fun::vv2b::lt< L, R ></code> Struct Template Reference	795
	10.161. Detailed Description	795
10.162	<code>ln::fun::vv2v::diff_abs< V ></code> Struct Template Reference	796
	10.162. Detailed Description	796
10.163	<code>ln::fun::vv2v::land< L, R ></code> Struct Template Reference	797
	10.163. Detailed Description	797
10.164	<code>ln::fun::vv2v::land_not< L, R ></code> Struct Template Reference	798
	10.164. Detailed Description	798
10.165	<code>ln::fun::vv2v::lor< L, R ></code> Struct Template Reference	799
	10.165. Detailed Description	799
10.166	<code>ln::fun::vv2v::lxor< L, R ></code> Struct Template Reference	800
	10.166. Detailed Description	800
10.167	<code>ln::fun::vv2v::max< V ></code> Struct Template Reference	801
	10.167. Detailed Description	801
10.168	<code>ln::fun::vv2v::min< L, R ></code> Struct Template Reference	802
	10.168. Detailed Description	802
10.169	<code>ln::fun::vv2v::vec< V ></code> Struct Template Reference	803
	10.169. Detailed Description	803
10.170	<code>ln::fun::x2p::closest_point< P ></code> Struct Template Reference	804
	10.170. Detailed Description	804
10.171	<code>ln::fun::x2v::bilinear< I ></code> Struct Template Reference	805
	10.171. Detailed Description	805
	10.171. Member Function Documentation	805
	10.171.2. <code>operator()</code>	805
	10.171.2. <code>operator()</code>	805
10.172	<code>ln::fun::x2v::trilinear< I ></code> Struct Template Reference	806
	10.172. Detailed Description	806
10.173	<code>ln::fun::x2x::composed< T2, T1 ></code> Struct Template Reference	807
	10.173. Detailed Description	807
	10.173. Constructor & Destructor Documentation	807
	10.173.2. <code>lcomposed</code>	807
	10.173.2. <code>composed</code>	807

10.174	<code>ln::fun::x2x::linear< I ></code> Struct Template Reference	808
10.174.1	Detailed Description	808
10.174.2	Constructor & Destructor Documentation	808
10.174.2.1	<code>linear</code>	808
10.174.3	Member Function Documentation	808
10.174.3.1	<code>operator()</code>	808
10.174.4	Member Data Documentation	809
10.174.4.1	<code>lima</code>	809
10.175	<code>ln::fun::x2x::rotation< n, C ></code> Struct Template Reference	810
10.175.1	Detailed Description	811
10.175.2	Member Typedef Documentation	811
10.175.2.1	<code>linvert</code>	811
10.175.3	Constructor & Destructor Documentation	811
10.175.3.1	<code>rotation</code>	811
10.175.3.2	<code>rotation</code>	811
10.175.3.3	<code>rotation</code>	811
10.175.3.4	<code>rotation</code>	811
10.175.4	Member Function Documentation	811
10.175.4.1	<code>linv</code>	811
10.175.4.2	<code>operator()</code>	811
10.175.4.3	<code>set_alpha</code>	812
10.175.4.4	<code>set_axis</code>	812
10.176	<code>ln::fun::x2x::translation< n, C ></code> Struct Template Reference	813
10.176.1	Detailed Description	814
10.176.2	Member Typedef Documentation	814
10.176.2.1	<code>linvert</code>	814
10.176.3	Constructor & Destructor Documentation	814
10.176.3.1	<code>translation</code>	814
10.176.3.2	<code>translation</code>	814
10.176.4	Member Function Documentation	814
10.176.4.1	<code>linv</code>	814
10.176.4.2	<code>operator()</code>	814
10.176.4.3	<code>set_t</code>	814
10.176.4.4	<code>t</code>	814
10.177	<code>ln::fun_image< F, I ></code> Struct Template Reference	815
10.177.1	Detailed Description	815

10.177.2	Member Typedef Documentation	816
10.177.2.1	lvalue	816
10.177.2.2	rvalue	816
10.177.2.3	skeleton	816
10.177.2.4	value	816
10.177.3	Constructor & Destructor Documentation	816
10.177.3.1	fun_image	816
10.177.3.2	fun_image	816
10.177.3.3	fun_image	816
10.177.4	Member Function Documentation	816
10.177.4.1	operator()	816
10.177.4.2	operator()	816
10.178	mln::Function< E > Struct Template Reference	817
10.178.1	Detailed Description	817
10.178.2	Constructor & Destructor Documentation	817
10.178.2.1	Function	817
10.179	mln::Function< void > Struct Template Reference	818
10.179.1	Detailed Description	818
10.180	mln::Function_v2b< E > Struct Template Reference	819
10.180.1	Detailed Description	819
10.181	mln::Function_v2v< E > Struct Template Reference	820
10.181.1	Detailed Description	820
10.182	mln::Function_vv2b< E > Struct Template Reference	821
10.182.1	Detailed Description	821
10.183	mln::Function_vv2v< E > Struct Template Reference	822
10.183.1	Detailed Description	822
10.184	mln::fwd_pixter1d< I > Class Template Reference	823
10.184.1	Detailed Description	823
10.184.2	Member Typedef Documentation	823
10.184.2.1	limage	823
10.184.3	Constructor & Destructor Documentation	823
10.184.3.1	fwd_pixter1d	823
10.184.4	Member Function Documentation	824
10.184.4.1	lnext	824
10.185	mln::fwd_pixter2d< I > Class Template Reference	825
10.185.1	Detailed Description	825

10.185.2	Member Typedef Documentation	825
10.185.2.1	image	825
10.185.3	Constructor & Destructor Documentation	825
10.185.3.1	ifwd_pixter2d	825
10.185.4	Member Function Documentation	826
10.185.4.1	lnext	826
10.186	ln::fwd_pixter3d< I > Class Template Reference	827
10.186.1	Detailed Description	827
10.186.2	Member Typedef Documentation	827
10.186.2.1	image	827
10.186.3	Constructor & Destructor Documentation	827
10.186.3.1	ifwd_pixter3d	827
10.186.4	Member Function Documentation	828
10.186.4.1	lnext	828
10.187	ln::Gdpoint< E > Struct Template Reference	829
10.187.1	Detailed Description	829
10.188	ln::Gdpoint< void > Struct Template Reference	830
10.188.1	Detailed Description	830
10.189	ln::Generalized_Pixel< E > Struct Template Reference	831
10.189.1	Detailed Description	831
10.190	ln::geom::complex_geometry< D, P > Class Template Reference	832
10.190.1	Detailed Description	832
10.190.2	Constructor & Destructor Documentation	832
10.190.2.1	lcomplex_geometry	832
10.190.3	Member Function Documentation	833
10.190.3.1	ladd_location	833
10.190.3.2	operator()	833
10.191	ln::Gpoint< E > Struct Template Reference	834
10.191.1	Detailed Description	835
10.191.2	Friends And Related Function Documentation	835
10.191.2.1	operator+	835
10.191.2.2	operator+=	835
10.191.2.3	operator-	836
10.191.2.4	operator-=	836
10.191.2.5	operator/	837
10.191.2.6	operator<<	837

10.191.2.7operator==	837
10.192ln::Graph< E > Struct Template Reference	838
10.192. Detailed Description	838
10.193ln::graph::attribute::card_t Struct Reference	839
10.193. Detailed Description	839
10.193. Member Typedef Documentation	839
10.193.2. lresult	839
10.194ln::graph::attribute::representative_t Struct Reference	840
10.194. Detailed Description	840
10.194. Member Typedef Documentation	840
10.194.2. lresult	840
10.195ln::graph_elt_mixed_neighborhood< G, S, S2 > Struct Template Reference	841
10.195. Detailed Description	841
10.195. Member Typedef Documentation	841
10.195.2. lbkd_niter	841
10.195.2.2fwd_niter	841
10.195.2.3niter	842
10.196ln::graph_elt_mixed_window< G, S, S2 > Class Template Reference	843
10.196. Detailed Description	844
10.196. Member Typedef Documentation	844
10.196.2. lbkd_qiter	844
10.196.2.2center_t	844
10.196.2.3fwd_qiter	844
10.196.2.4graph_element	844
10.196.2.5psite	844
10.196.2.6qiter	845
10.196.2.7site	845
10.196.2.8target	845
10.196. Member Function Documentation	845
10.196.3. ldelta	845
10.196.3.2is_centered	845
10.196.3.3is_empty	845
10.196.3.4is_symmetric	845
10.196.3.5is_valid	845
10.196.3.6sym	846
10.197ln::graph_elt_neighborhood< G, S > Struct Template Reference	847

10.197. Detailed Description	847
10.197. Member Typedef Documentation	847
10.197.2. lbkd_niter	847
10.197.2.2 fwd_niter	847
10.197.2.3 niter	848
10.198. In::graph_elt_neighborhood_if< G, S, I > Struct Template Reference	849
10.198. Detailed Description	849
10.198. Member Typedef Documentation	849
10.198.2. lbkd_niter	849
10.198.2.2 fwd_niter	850
10.198.2.3 niter	850
10.198. Constructor & Destructor Documentation	850
10.198.3.1 graph_elt_neighborhood_if	850
10.198.3.2 graph_elt_neighborhood_if	850
10.198. Member Function Documentation	850
10.198.4. lmask	850
10.199. In::graph_elt_window< G, S > Class Template Reference	851
10.199. Detailed Description	852
10.199. Member Typedef Documentation	852
10.199.2. lbkd_qiter	852
10.199.2.2 center_t	852
10.199.2.3 fwd_qiter	852
10.199.2.4 graph_element	853
10.199.2.5 psite	853
10.199.2.6 qiter	853
10.199.2.7 site	853
10.199.2.8 target	853
10.199. Member Function Documentation	853
10.199.3. ldelta	853
10.199.3.2 is_centered	853
10.199.3.3 is_empty	853
10.199.3.4 is_symmetric	854
10.199.3.5 is_valid	854
10.199.3.6 sym	854
10.200. In::graph_elt_window_if< G, S, I > Class Template Reference	855
10.200. Detailed Description	856

10.200.2	Member Typedef Documentation	856
10.200.2.1	bkd_qiter	856
10.200.2.2	fwd_qiter	857
10.200.2.3	mask_t	857
10.200.2.4	psite	857
10.200.2.5	qiter	857
10.200.2.6	site	857
10.200.2.7	target	857
10.200.3	Constructor & Destructor Documentation	857
10.200.3.1	graph_elt_window_if	857
10.200.3.2	graph_elt_window_if	858
10.200.4	Member Function Documentation	858
10.200.4.1	change_mask	858
10.200.4.2	delta	858
10.200.4.3	is_centered	858
10.200.4.4	is_empty	858
10.200.4.5	is_symmetric	858
10.200.4.6	is_valid	858
10.200.4.7	mask	859
10.200.4.8	sym	859
10.201	mln::graph_window_base< P, E > Class Template Reference	860
10.201.1	Detailed Description	860
10.201.2	Member Typedef Documentation	861
10.201.2.1	site	861
10.201.3	Member Function Documentation	861
10.201.3.1	delta	861
10.201.3.2	is_centered	861
10.201.3.3	is_empty	861
10.201.3.4	is_symmetric	861
10.201.3.5	is_valid	861
10.201.3.6	sym	861
10.202	mln::graph_window_if_piter< S, W, I > Class Template Reference	862
10.202.1	Detailed Description	862
10.202.2	Member Typedef Documentation	862
10.202.2.1	IP	862
10.202.3	Constructor & Destructor Documentation	863

10.202.3.1graph_window_if_piter	863
10.202.4Member Function Documentation	863
10.202.4.1element	863
10.202.4.2id	863
10.202.4.3next	863
10.203Inn::graph_window_piter< S, W, I > Class Template Reference	864
10.203.1Detailed Description	865
10.203.2Member Typedef Documentation	865
10.203.2.1center_t	865
10.203.2.2graph_element	865
10.203.2.3P	865
10.203.3Constructor & Destructor Documentation	865
10.203.3.1graph_window_piter	865
10.203.3.2graph_window_piter	865
10.203.3.3graph_window_piter	866
10.203.4Member Function Documentation	866
10.203.4.1change_target_site_set	866
10.203.4.2element	866
10.203.4.3id	866
10.203.4.4next	866
10.203.4.5target_site_set	867
10.204Inn::hexa< I > Struct Template Reference	868
10.204.1Detailed Description	869
10.204.2Member Typedef Documentation	869
10.204.2.1bkd_piter	869
10.204.2.2fwd_piter	869
10.204.2.3value	869
10.204.2.4psite	869
10.204.2.5value	869
10.204.2.6skeleton	870
10.204.2.7value	870
10.204.3Constructor & Destructor Documentation	870
10.204.3.1hexa	870
10.204.3.2hexa	870
10.204.4Member Function Documentation	870
10.204.4.1domain	870

10.204.4.2has	870
10.204.4.3operator()	870
10.204.4.4operator()	870
10.205ln::histo::array< T > Struct Template Reference	871
10.205.Detailed Description	871
10.206ln::Image< E > Struct Template Reference	872
10.206.Detailed Description	874
10.207ln::image1d< T > Struct Template Reference	875
10.207.Detailed Description	876
10.207.2Member Typedef Documentation	876
10.207.2.1lvalue	876
10.207.2.2rvalue	876
10.207.2.3skeleton	876
10.207.2.4value	877
10.207.3Constructor & Destructor Documentation	877
10.207.3.1image1d	877
10.207.3.2image1d	877
10.207.3.3image1d	877
10.207.4Member Function Documentation	877
10.207.4.1bbox	877
10.207.4.2border	877
10.207.4.3buffer	877
10.207.4.4buffer	877
10.207.4.5delta_index	877
10.207.4.6domain	878
10.207.4.7element	878
10.207.4.8element	878
10.207.4.9has	878
10.207.4.10elements	878
10.207.4.11finds	878
10.207.4.12operator()	878
10.207.4.13operator()	878
10.207.4.14point_at_index	879
10.208ln::image2d< T > Class Template Reference	880
10.208.Detailed Description	881
10.208.2Member Typedef Documentation	881

10.208.2.1lvalue	881
10.208.2.2rvalue	881
10.208.2.3skeleton	882
10.208.2.4value	882
10.208.3.Constructor & Destructor Documentation	882
10.208.3.1image2d	882
10.208.3.2image2d	882
10.208.3.3image2d	882
10.208.4.Member Function Documentation	882
10.208.4.1bbox	882
10.208.4.2border	882
10.208.4.3buffer	882
10.208.4.4buffer	882
10.208.4.5delta_index	883
10.208.4.6domain	883
10.208.4.7element	883
10.208.4.8element	883
10.208.4.9has	883
10.208.4.10cols	883
10.208.4.11elements	883
10.208.4.12rows	883
10.208.4.13operator()	883
10.208.4.14operator()	884
10.208.4.15point_at_index	884
10.209.In::image2d_h< V > Struct Template Reference	885
10.209.1.Detailed Description	886
10.209.2.Member Typedef Documentation	886
10.209.2.1bkd_piter	886
10.209.2.2fwd_piter	886
10.209.2.3value	886
10.209.2.4psite	886
10.209.2.5value	886
10.209.2.6skeleton	886
10.209.2.7value	886
10.209.3.Constructor & Destructor Documentation	887
10.209.3.1image2d_h	887

10.209.4	Member Function Documentation	887
10.209.4.1	domain	887
10.209.4.2	has	887
10.209.4.3	operator()	887
10.209.4.4	operator()	887
10.210.1	ln::image3d< T > Struct Template Reference	888
10.210.1	Detailed Description	889
10.210.2	Member Typedef Documentation	889
10.210.2.1	lvalue	889
10.210.2.2	rvalue	890
10.210.2.3	skeleton	890
10.210.2.4	value	890
10.210.3	Constructor & Destructor Documentation	890
10.210.3.1	image3d	890
10.210.3.2	image3d	890
10.210.3.3	image3d	890
10.210.4	Member Function Documentation	890
10.210.4.1	lbbox	890
10.210.4.2	border	890
10.210.4.3	buffer	890
10.210.4.4	buffer	891
10.210.4.5	delta_index	891
10.210.4.6	domain	891
10.210.4.7	element	891
10.210.4.8	element	891
10.210.4.9	has	891
10.210.4.10	cols	891
10.210.4.11	elements	891
10.210.4.12	rows	892
10.210.4.13	slices	892
10.210.4.14	operator()	892
10.210.4.15	operator()	892
10.210.4.16	point_at_index	892
10.211.1	ln::image_if< I, F > Struct Template Reference	893
10.211.1	Detailed Description	893
10.211.2	Member Typedef Documentation	893

10.211.2. Iskeleton	893
10.211.3. Constructor & Destructor Documentation	893
10.211.3.1 image_if	893
10.211.3.2 image_if	894
10.211.4. Member Function Documentation	894
10.211.4.1 domain	894
10.211.4.2 operator image_if< const I, F >	894
10.212. In::interpolated< I, F > Struct Template Reference	895
10.212.1. Detailed Description	895
10.212.2. Member Typedef Documentation	895
10.212.2.1 lvalue	895
10.212.2.2 psite	896
10.212.2.3 rvalue	896
10.212.2.4 skeleton	896
10.212.2.5 value	896
10.212.3. Constructor & Destructor Documentation	896
10.212.3.1 interpolated	896
10.212.4. Member Function Documentation	896
10.212.4.1 lhas	896
10.212.4.2 is_valid	896
10.213. In::io::fld::fld_header Struct Reference	897
10.213.1. Detailed Description	897
10.214. In::Iterator< E > Struct Template Reference	898
10.214.1. Detailed Description	899
10.214.2. Member Function Documentation	899
10.214.2.1 lnext	899
10.215. In::labeled_image< I > Class Template Reference	900
10.215.1. Detailed Description	901
10.215.2. Member Typedef Documentation	901
10.215.2.1 lbbox_t	901
10.215.2.2 skeleton	901
10.215.3. Constructor & Destructor Documentation	901
10.215.3.1 labeled_image	901
10.215.3.2 labeled_image	902
10.215.3.3 labeled_image	902
10.215.4. Member Function Documentation	902

10.215.4.1 bbox	902
10.215.4.2 bboxes	902
10.215.4.3 labels	902
10.215.4.4 relabel	902
10.215.4.5 relabel	902
10.215.4.6 subdomain	903
10.215.4.7 update_data	903
10.216.1 ln::labeled_image_base< I, E > Class Template Reference	904
10.216.1.1 Detailed Description	905
10.216.1.2 Member Typedef Documentation	905
10.216.1.2.1 bbox_t	905
10.216.1.3 Constructor & Destructor Documentation	905
10.216.1.3.1 labeled_image_base	905
10.216.1.4 Member Function Documentation	905
10.216.1.4.1 bbox	905
10.216.1.4.2 bboxes	905
10.216.1.4.3 labels	906
10.216.1.4.4 relabel	906
10.216.1.4.5 relabel	906
10.216.1.4.6 subdomain	906
10.216.1.4.7 update_data	906
10.217.1 ln::lazy_image< I, F, B > Struct Template Reference	907
10.217.1.1 Detailed Description	908
10.217.1.2 Member Typedef Documentation	908
10.217.1.2.1 lvalue	908
10.217.1.2.2 rvalue	908
10.217.1.2.3 skeleton	908
10.217.1.3 Constructor & Destructor Documentation	908
10.217.1.3.1 lazy_image	908
10.217.1.3.2 lazy_image	908
10.217.1.4 Member Function Documentation	908
10.217.1.4.1 ldomain	908
10.217.1.4.2 has	909
10.217.1.4.3 operator()	909
10.217.1.4.4 operator()	909
10.217.1.4.5 operator()	909

10.217.4.operator()	909
10.218.ln::Literal< E > Struct Template Reference	910
10.218.Detailed Description	912
10.219.ln::literal::black_t Struct Reference	913
10.219.Detailed Description	913
10.220.ln::literal::blue_t Struct Reference	914
10.220.Detailed Description	914
10.221.ln::literal::brown_t Struct Reference	915
10.221.Detailed Description	915
10.222.ln::literal::cyan_t Struct Reference	916
10.222.Detailed Description	916
10.223.ln::literal::green_t Struct Reference	917
10.223.Detailed Description	917
10.224.ln::literal::identity_t Struct Reference	918
10.224.Detailed Description	918
10.225.ln::literal::light_gray_t Struct Reference	919
10.225.Detailed Description	919
10.226.ln::literal::lime_t Struct Reference	920
10.226.Detailed Description	920
10.227.ln::literal::magenta_t Struct Reference	921
10.227.Detailed Description	921
10.228.ln::literal::max_t Struct Reference	922
10.228.Detailed Description	922
10.229.ln::literal::min_t Struct Reference	923
10.229.Detailed Description	923
10.230.ln::literal::olive_t Struct Reference	924
10.230.Detailed Description	924
10.231.ln::literal::one_t Struct Reference	925
10.231.Detailed Description	925
10.232.ln::literal::orange_t Struct Reference	926
10.232.Detailed Description	926
10.233.ln::literal::origin_t Struct Reference	927
10.233.Detailed Description	927
10.234.ln::literal::pink_t Struct Reference	928
10.234.Detailed Description	928
10.235.ln::literal::purple_t Struct Reference	929

10.235. Detailed Description	929
10.236. <code>mln::literal::red_t</code> Struct Reference	930
10.236. Detailed Description	930
10.237. <code>mln::literal::teal_t</code> Struct Reference	931
10.237. Detailed Description	931
10.238. <code>mln::literal::violet_t</code> Struct Reference	932
10.238. Detailed Description	932
10.239. <code>mln::literal::white_t</code> Struct Reference	933
10.239. Detailed Description	933
10.240. <code>mln::literal::yellow_t</code> Struct Reference	934
10.240. Detailed Description	934
10.241. <code>mln::literal::zero_t</code> Struct Reference	935
10.241. Detailed Description	935
10.242. <code>mln::Mesh< E ></code> Struct Template Reference	936
10.242. Detailed Description	936
10.243. <code>mln::Meta_Accumulator< E ></code> Struct Template Reference	937
10.243. Detailed Description	937
10.244. <code>mln::Meta_Function< E ></code> Struct Template Reference	938
10.244. Detailed Description	938
10.245. <code>mln::Meta_Function_v2v< E ></code> Struct Template Reference	939
10.245. Detailed Description	939
10.246. <code>mln::Meta_Function_vv2v< E ></code> Struct Template Reference	940
10.246. Detailed Description	940
10.247. <code>mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 ></code> Struct Template Reference	941
10.247. Detailed Description	941
10.248. <code>mln::metal::converts_to< T, U ></code> Struct Template Reference	942
10.248. Detailed Description	942
10.249. <code>mln::metal::equal< T1, T2 ></code> Struct Template Reference	943
10.249. Detailed Description	943
10.250. <code>mln::metal::goes_to< T, U ></code> Struct Template Reference	944
10.250. Detailed Description	944
10.251. <code>mln::metal::is< T, U ></code> Struct Template Reference	945
10.251. Detailed Description	945
10.252. <code>mln::metal::is_a< T, M ></code> Struct Template Reference	946
10.252. Detailed Description	946
10.253. <code>mln::metal::is_not< T, U ></code> Struct Template Reference	947

10.253. Detailed Description	947
10.254. <code>ln::metal::is_not_a< T, M ></code> Struct Template Reference	948
10.254. Detailed Description	948
10.255. <code>ln::mixed_neighb< W ></code> Class Template Reference	949
10.255. Detailed Description	949
10.255. Member Typedef Documentation	949
10.255.2. <code>l_bkd_niter</code>	949
10.255.2.2. <code>2fwd_niter</code>	949
10.255.2.3. <code>3niter</code>	950
10.255.3. Constructor & Destructor Documentation	950
10.255.3.1. <code>l_mixed_neighb</code>	950
10.255.3.2. <code>2mixed_neighb</code>	950
10.256. <code>ln::morpho::attribute::card< I ></code> Class Template Reference	951
10.256. Detailed Description	951
10.256. Member Function Documentation	951
10.256.2. <code>linit</code>	951
10.256.2.2. <code>2is_valid</code>	951
10.256.2.3. <code>3take_as_init</code>	952
10.256.2.4. <code>4take_n_times</code>	952
10.256.2.5. <code>5to_result</code>	952
10.257. <code>ln::morpho::attribute::count_adjacent_vertices< I ></code> Struct Template Reference	953
10.257. Detailed Description	953
10.257. Member Function Documentation	953
10.257.2. <code>linit</code>	953
10.257.2.2. <code>2is_valid</code>	953
10.257.2.3. <code>3take_as_init</code>	954
10.257.2.4. <code>4take_n_times</code>	954
10.257.2.5. <code>5to_result</code>	954
10.258. <code>ln::morpho::attribute::height< I ></code> Struct Template Reference	955
10.258. Detailed Description	955
10.258. Member Function Documentation	955
10.258.2. <code>lbase_level</code>	955
10.258.2.2. <code>2init</code>	955
10.258.2.3. <code>3is_valid</code>	956
10.258.2.4. <code>4take_as_init</code>	956
10.258.2.5. <code>5take_n_times</code>	956

10.258.2.6to_result	956
10.259. In::morpho::attribute::sharpness< I > Struct Template Reference	957
10.259. Detailed Description	957
10.259. Member Function Documentation	958
10.259.2. larea	958
10.259.2.2height	958
10.259.2.3init	958
10.259.2.4is_valid	958
10.259.2.5take_as_init	958
10.259.2.6take_n_times	958
10.259.2.7to_result	958
10.259.2.8volume	959
10.260. In::morpho::attribute::sum< I, S > Class Template Reference	960
10.260. Detailed Description	960
10.260. Member Function Documentation	960
10.260.2. linit	960
10.260.2.2is_valid	961
10.260.2.3set_value	961
10.260.2.4take_as_init	961
10.260.2.5take_n_times	961
10.260.2.6to_result	961
10.260.2.7untake	961
10.261. In::morpho::attribute::volume< I > Struct Template Reference	962
10.261. Detailed Description	962
10.261. Member Function Documentation	962
10.261.2. larea	962
10.261.2.2init	963
10.261.2.3is_valid	963
10.261.2.4take_as_init	963
10.261.2.5take_n_times	963
10.261.2.6to_result	963
10.262. In::neighb< W > Class Template Reference	964
10.262. Detailed Description	964
10.262. Member Typedef Documentation	965
10.262.2. lbkd_niter	965
10.262.2.2fwd_niter	965

10.262.2.3niter	965
10.262.3.Constructor & Destructor Documentation	965
10.262.3.1neighb	965
10.262.3.2neighb	965
10.263.mln::Neighborhood< E > Struct Template Reference	966
10.263. Detailed Description	966
10.264.mln::Neighborhood< void > Struct Template Reference	967
10.264. Detailed Description	967
10.265.mln::Object< E > Struct Template Reference	968
10.265. Detailed Description	968
10.266.mln::p2p_image< I, F > Struct Template Reference	969
10.266. Detailed Description	969
10.266.2.Member Typedef Documentation	969
10.266.2.1skeleton	969
10.266.3.Constructor & Destructor Documentation	970
10.266.3.1p2p_image	970
10.266.3.2p2p_image	970
10.266.4.Member Function Documentation	970
10.266.4.1domain	970
10.266.4.2fun	970
10.266.4.3operator()	970
10.266.4.4operator()	970
10.267.mln::p_array< P > Class Template Reference	971
10.267. Detailed Description	973
10.267.2.Member Typedef Documentation	973
10.267.2.1bkd_piter	973
10.267.2.2element	973
10.267.2.3fwd_piter	973
10.267.2.4_element	974
10.267.2.5piter	974
10.267.2.6psite	974
10.267.3.Constructor & Destructor Documentation	974
10.267.3.1p_array	974
10.267.3.2p_array	974
10.267.4.Member Function Documentation	974
10.267.4.1append	974

10.267.4.2	append	974
10.267.4.3	change	974
10.267.4.4	clear	974
10.267.4.5	has	975
10.267.4.6	has	975
10.267.4.7	insert	975
10.267.4.8	is_valid	975
10.267.4.9	memory_size	975
10.267.4.10	sites	975
10.267.4.11	operator[]	975
10.267.4.12	operator[]	975
10.267.4.13	operator[]	976
10.267.4.14	reserve	976
10.267.4.15	size	976
10.267.4.16	id_vector	976
10.267.5	Friends And Related Function Documentation	976
10.267.5.1	diff	976
10.267.5.2	inter	976
10.267.5.3	operator<	976
10.267.5.4	operator<<	977
10.267.5.5	operator<=	977
10.267.5.6	operator==	977
10.267.5.7	sym_diff	977
10.267.5.8	uni	977
10.267.5.9	unique	977
10.268	ln::p_centered< W > Class Template Reference	978
10.268.1	Detailed Description	979
10.268.2	Member Typedef Documentation	980
10.268.2.1	l_bkd_piter	980
10.268.2.2	element	980
10.268.2.3	fwd_piter	980
10.268.2.4	piter	980
10.268.2.5	psite	980
10.268.2.6	site	980
10.268.3	Constructor & Destructor Documentation	980
10.268.3.1	lp_centered	980

10.268.3.2p_centered	980
10.268.4Member Function Documentation	980
10.268.4.1center	980
10.268.4.2has	981
10.268.4.3is_valid	981
10.268.4.4memory_size	981
10.268.4.5window	981
10.268.5Friends And Related Function Documentation	981
10.268.5.1diff	981
10.268.5.2inter	981
10.268.5.3operator<	981
10.268.5.4operator<<	981
10.268.5.5operator<=	982
10.268.5.6operator==	982
10.268.5.7sym_diff	982
10.268.5.8uni	982
10.268.5.9unique	982
10.269In::p_complex< D, G > Class Template Reference	983
10.269.1Detailed Description	984
10.269.2Member Typedef Documentation	985
10.269.2.1bkd_piter	985
10.269.2.2element	985
10.269.2.3fwd_piter	985
10.269.2.4piter	985
10.269.2.5psite	985
10.269.3Constructor & Destructor Documentation	985
10.269.3.1p_complex	985
10.269.4Member Function Documentation	986
10.269.4.1cplx	986
10.269.4.2cplx	986
10.269.4.3geom	986
10.269.4.4has	986
10.269.4.5is_valid	986
10.269.4.6faces	986
10.269.4.7nfaces_of_dim	986
10.269.4.8sites	987

10.269.5	Friends And Related Function Documentation	987
10.269.5.1	diff	987
10.269.5.2	inter	987
10.269.5.3	operator<	987
10.269.5.4	operator<<	987
10.269.5.5	operator<=	987
10.269.5.6	operator==	988
10.269.5.7	sym_diff	988
10.269.5.8	uni	988
10.269.5.9	unique	988
10.270	mln::p_edges< G, F > Class Template Reference	989
10.270.1	Detailed Description	991
10.270.2	Member Typedef Documentation	991
10.270.2.1	l_bkd_piter	991
10.270.2.2	edge	991
10.270.2.3	element	991
10.270.2.4	fun_t	992
10.270.2.5	fwd_piter	992
10.270.2.6	graph_element	992
10.270.2.7	graph_t	992
10.270.2.8	piter	992
10.270.2.9	psite	992
10.270.3	Constructor & Destructor Documentation	992
10.270.3.1	lp_edges	992
10.270.3.2	p_edges	992
10.270.3.3	p_edges	993
10.270.3.4	p_edges	993
10.270.4	Member Function Documentation	993
10.270.4.1	function	993
10.270.4.2	graph	993
10.270.4.3	has	993
10.270.4.4	has	994
10.270.4.5	invalidate	994
10.270.4.6	is_valid	994
10.270.4.7	memory_size	994
10.270.4.8	nedges	994

10.270.4.9nsites	994
10.270.5.Friends And Related Function Documentation	994
10.270.5.1diff	994
10.270.5.2inter	994
10.270.5.3operator<	995
10.270.5.4operator<<	995
10.270.5.5operator<=	995
10.270.5.6operator==	995
10.270.5.7sym_diff	995
10.270.5.8uni	996
10.270.5.9unique	996
10.271.mn::p_faces< N, D, P > Struct Template Reference	997
10.271.1.Detailed Description	998
10.271.2.Member Typedef Documentation	999
10.271.2.1bkd_piter	999
10.271.2.2element	999
10.271.2.3fwd_piter	999
10.271.2.4piter	999
10.271.2.5psite	999
10.271.3.Constructor & Destructor Documentation	999
10.271.3.1p_faces	999
10.271.3.2p_faces	999
10.271.4.Member Function Documentation	1000
10.271.4.1cplx	1000
10.271.4.2cplx	1000
10.271.4.3is_valid	1000
10.271.4.4nfaces	1000
10.271.4.5nsites	1000
10.271.5.Friends And Related Function Documentation	1000
10.271.5.1diff	1000
10.271.5.2inter	1001
10.271.5.3operator<	1001
10.271.5.4operator<<	1001
10.271.5.5operator<=	1001
10.271.5.6operator==	1001
10.271.5.7sym_diff	1002

10.273.5.7sym_diff	1009
10.273.5.8uni	1009
10.273.5.9unique	1009
10.274.1nl::p_image< I > Class Template Reference	1010
10.274.1.Detailed Description	1012
10.274.2.Member Typedef Documentation	1012
10.274.2.1bkd_piter	1012
10.274.2.2element	1012
10.274.2.3fwd_piter	1012
10.274.2.4i_element	1012
10.274.2.5piter	1012
10.274.2.6psite	1012
10.274.2.7r_element	1012
10.274.2.8S	1013
10.274.3.Constructor & Destructor Documentation	1013
10.274.3.1p_image	1013
10.274.3.2p_image	1013
10.274.4.Member Function Documentation	1013
10.274.4.1clear	1013
10.274.4.2has	1013
10.274.4.3insert	1013
10.274.4.4is_valid	1013
10.274.4.5memory_size	1013
10.274.4.6nsites	1014
10.274.4.7operator typename internal::p_image_site_set< I >::ret	1014
10.274.4.8remove	1014
10.274.4.9toggle	1014
10.274.5.Friends And Related Function Documentation	1014
10.274.5.1diff	1014
10.274.5.2inter	1014
10.274.5.3operator<	1014
10.274.5.4operator<<	1015
10.274.5.5operator<=	1015
10.274.5.6operator==	1015
10.274.5.7sym_diff	1015
10.274.5.8uni	1015

10.274.5.9unique	1015
10.275 ln::p_indexed_bkd_piter< S > Class Template Reference	1016
10.275.1 Detailed Description	1016
10.275.2 Constructor & Destructor Documentation	1016
10.275.2.1p_indexed_bkd_piter	1016
10.275.2.2p_indexed_bkd_piter	1016
10.275.3 Member Function Documentation	1016
10.275.3.1index	1016
10.275.3.2next	1017
10.276 ln::p_indexed_fwd_piter< S > Class Template Reference	1018
10.276.1 Detailed Description	1018
10.276.2 Constructor & Destructor Documentation	1018
10.276.2.1p_indexed_fwd_piter	1018
10.276.2.2p_indexed_fwd_piter	1018
10.276.3 Member Function Documentation	1018
10.276.3.1index	1018
10.276.3.2next	1019
10.277 ln::p_indexed_psite< S > Class Template Reference	1020
10.277.1 Detailed Description	1020
10.278 ln::p_key< K, P > Class Template Reference	1021
10.278.1 Detailed Description	1023
10.278.2 Member Typedef Documentation	1023
10.278.2.1bkd_piter	1023
10.278.2.2element	1023
10.278.2.3fwd_piter	1024
10.278.2.4i_element	1024
10.278.2.5piter	1024
10.278.2.6psite	1024
10.278.2.7r_element	1024
10.278.3 Constructor & Destructor Documentation	1024
10.278.3.1p_key	1024
10.278.4 Member Function Documentation	1024
10.278.4.1change_key	1024
10.278.4.2change_keys	1024
10.278.4.3clear	1024
10.278.4.4exists_key	1025

10.278.4.5	has	1025
10.278.4.6	has	1025
10.278.4.7	insert	1025
10.278.4.8	insert	1025
10.278.4.9	is_valid	1025
10.278.4.10	key	1025
10.278.4.11	keys	1025
10.278.4.12	memory_size	1025
10.278.4.13	sites	1026
10.278.4.14	operator()	1026
10.278.4.15	move	1026
10.278.4.16	move_key	1026
10.278.5	Friends And Related Function Documentation	1026
10.278.5.1	diff	1026
10.278.5.2	inter	1026
10.278.5.3	operator<	1026
10.278.5.4	operator<<	1027
10.278.5.5	operator<=	1027
10.278.5.6	operator==	1027
10.278.5.7	sym_diff	1027
10.278.5.8	uni	1027
10.278.5.9	unique	1027
10.279	nl::p_line2d Class Reference	1028
10.279.1	Detailed Description	1030
10.279.2	Member Typedef Documentation	1030
10.279.2.1	bkd_piter	1030
10.279.2.2	element	1030
10.279.2.3	fwd_piter	1030
10.279.2.4	piter	1030
10.279.2.5	psite	1030
10.279.2.6	q_box	1030
10.279.3	Constructor & Destructor Documentation	1030
10.279.3.1	lp_line2d	1030
10.279.3.2	p_line2d	1031
10.279.4	Member Function Documentation	1031
10.279.4.1	lbox	1031

10.279.4.2	<code>begin</code>	1031
10.279.4.3	<code>end</code>	1031
10.279.4.4	<code>has</code>	1031
10.279.4.5	<code>has</code>	1031
10.279.4.6	<code>is_valid</code>	1031
10.279.4.7	<code>memory_size</code>	1031
10.279.4.8	<code>nsites</code>	1032
10.279.4.9	<code>operator[]</code>	1032
10.279.4.10	<code>id_vector</code>	1032
10.279.5	Friends And Related Function Documentation	1032
10.279.5.1	<code>diff</code>	1032
10.279.5.2	<code>inter</code>	1032
10.279.5.3	<code>operator<</code>	1032
10.279.5.4	<code>operator<<</code>	1032
10.279.5.5	<code>operator<=</code>	1033
10.279.5.6	<code>operator==</code>	1033
10.279.5.7	<code>sym_diff</code>	1033
10.279.5.8	<code>uni</code>	1033
10.279.5.9	<code>unique</code>	1033
10.280	<code>std::p_mutable_array_of< S ></code> Class Template Reference	1034
10.280.1	Detailed Description	1036
10.280.2	Member Typedef Documentation	1036
10.280.2.1	<code>l_bkd_piter</code>	1036
10.280.2.2	<code>element</code>	1036
10.280.2.3	<code>fwd_piter</code>	1036
10.280.2.4	<code>i_element</code>	1036
10.280.2.5	<code>piter</code>	1036
10.280.2.6	<code>psite</code>	1036
10.280.3	Constructor & Destructor Documentation	1036
10.280.3.1	<code>p_mutable_array_of</code>	1036
10.280.4	Member Function Documentation	1037
10.280.4.1	<code>clear</code>	1037
10.280.4.2	<code>has</code>	1037
10.280.4.3	<code>insert</code>	1037
10.280.4.4	<code>is_valid</code>	1037
10.280.4.5	<code>memory_size</code>	1037

10.280.4.6elements	1037
10.280.4.7operator[1037
10.280.4.8operator[1037
10.280.4.9reserve	1038
10.280.5Friends And Related Function Documentation	1038
10.280.5.1diff	1038
10.280.5.2inter	1038
10.280.5.3operator<	1038
10.280.5.4operator<<	1038
10.280.5.5operator<=	1038
10.280.5.6operator==	1039
10.280.5.7sym_diff	1039
10.280.5.8uni	1039
10.280.5.9unique	1039
10.281mln::p_n_faces_bkd_piter< D, P > Class Template Reference	1040
10.281.1Detailed Description	1040
10.281.2Constructor & Destructor Documentation	1040
10.281.2.1p_n_faces_bkd_piter	1040
10.281.3Member Function Documentation	1040
10.281.3.1n	1040
10.281.3.2next	1040
10.282mln::p_n_faces_fwd_piter< D, P > Class Template Reference	1042
10.282.1Detailed Description	1042
10.282.2Constructor & Destructor Documentation	1042
10.282.2.1p_n_faces_fwd_piter	1042
10.282.3Member Function Documentation	1042
10.282.3.1n	1042
10.282.3.2next	1042
10.283mln::p_priority< P, Q > Class Template Reference	1044
10.283.1Detailed Description	1046
10.283.2Member Typedef Documentation	1046
10.283.2.1bkd_piter	1046
10.283.2.2element	1046
10.283.2.3fwd_piter	1047
10.283.2.4element	1047
10.283.2.5piter	1047

10.283.2.6psite	1047
10.283.3.Constructor & Destructor Documentation	1047
10.283.3.1p_priority	1047
10.283.4.Member Function Documentation	1047
10.283.4.1clear	1047
10.283.4.2exists_priority	1047
10.283.4.3front	1047
10.283.4.4has	1048
10.283.4.5highest_priority	1048
10.283.4.6insert	1048
10.283.4.7insert	1048
10.283.4.8s_valid	1048
10.283.4.9lowest_priority	1048
10.283.4.10memory_size	1049
10.283.4.11sites	1049
10.283.4.12operator()	1049
10.283.4.13pop	1049
10.283.4.14pop_front	1049
10.283.4.15priorities	1049
10.283.4.16push	1050
10.283.5.Friends And Related Function Documentation	1050
10.283.5.1diff	1050
10.283.5.2inter	1050
10.283.5.3operator<	1050
10.283.5.4operator<<	1050
10.283.5.5operator<=	1050
10.283.5.6operator==	1051
10.283.5.7sym_diff	1051
10.283.5.8uni	1051
10.283.5.9unique	1051
10.284.in::p_queue< P > Class Template Reference	1052
10.284.1.Detailed Description	1054
10.284.2.Member Typedef Documentation	1054
10.284.2.1bkd_piter	1054
10.284.2.2element	1054
10.284.2.3fwd_piter	1054

10.284.2.4i_element	1054
10.284.2.5piter	1054
10.284.2.6psite	1055
10.284.3Constructor & Destructor Documentation	1055
10.284.3.1p_queue	1055
10.284.4Member Function Documentation	1055
10.284.4.1clear	1055
10.284.4.2front	1055
10.284.4.3has	1055
10.284.4.4has	1055
10.284.4.5insert	1055
10.284.4.6is_valid	1055
10.284.4.7memory_size	1055
10.284.4.8nsites	1056
10.284.4.9operator[]	1056
10.284.4.10pop	1056
10.284.4.11pop_front	1056
10.284.4.12push	1056
10.284.4.13std_deque	1056
10.284.5Friends And Related Function Documentation	1056
10.284.5.1diff	1056
10.284.5.2inter	1057
10.284.5.3operator<	1057
10.284.5.4operator<<	1057
10.284.5.5operator<=	1057
10.284.5.6operator==	1057
10.284.5.7sym_diff	1058
10.284.5.8uni	1058
10.284.5.9unique	1058
10.285Inl::p_queue_fast< P > Class Template Reference	1059
10.285.1Detailed Description	1061
10.285.2Member Typedef Documentation	1061
10.285.2.1bkd_piter	1061
10.285.2.2element	1061
10.285.2.3fwd_piter	1062
10.285.2.4i_element	1062

10.285.2.5	<code>piter</code>	1062
10.285.2.6	<code>psite</code>	1062
10.285.3	Constructor & Destructor Documentation	1062
10.285.3.1	<code>p_queue_fast</code>	1062
10.285.4	Member Function Documentation	1062
10.285.4.1	<code>clear</code>	1062
10.285.4.2	<code>compute_has</code>	1062
10.285.4.3	<code>empty</code>	1062
10.285.4.4	<code>front</code>	1062
10.285.4.5	<code>has</code>	1063
10.285.4.6	<code>has</code>	1063
10.285.4.7	<code>insert</code>	1063
10.285.4.8	<code>is_valid</code>	1063
10.285.4.9	<code>memory_size</code>	1063
10.285.4.10	<code>sites</code>	1063
10.285.4.11	<code>operator[]</code>	1063
10.285.4.12	<code>pop</code>	1063
10.285.4.13	<code>pop_front</code>	1064
10.285.4.14	<code>purge</code>	1064
10.285.4.15	<code>push</code>	1064
10.285.4.16	<code>reserve</code>	1064
10.285.4.17	<code>std_vector</code>	1064
10.285.5	Friends And Related Function Documentation	1064
10.285.5.1	<code>diff</code>	1064
10.285.5.2	<code>inter</code>	1064
10.285.5.3	<code>operator<</code>	1064
10.285.5.4	<code>operator<<</code>	1065
10.285.5.5	<code>operator<=</code>	1065
10.285.5.6	<code>operator==</code>	1065
10.285.5.7	<code>sym_diff</code>	1065
10.285.5.8	<code>uni</code>	1065
10.285.5.9	<code>unique</code>	1065
10.286	<code>mln::p_run< P ></code> Class Template Reference	1066
10.286.1	Detailed Description	1068
10.286.2	Member Typedef Documentation	1068
10.286.2.1	<code>bkd_piter</code>	1068

10.286.2.2element	1068
10.286.2.3fwd_piter	1068
10.286.2.4piter	1068
10.286.2.5psite	1068
10.286.2.6q_box	1069
10.286.3.Constructor & Destructor Documentation	1069
10.286.3.1p_run	1069
10.286.3.2p_run	1069
10.286.3.3p_run	1069
10.286.4.Member Function Documentation	1069
10.286.4.1bbox	1069
10.286.4.2end	1069
10.286.4.3has	1069
10.286.4.4has	1069
10.286.4.5has_index	1070
10.286.4.6nit	1070
10.286.4.7is_valid	1070
10.286.4.8length	1070
10.286.4.9memory_size	1070
10.286.4.10sites	1070
10.286.4.11operator[]	1070
10.286.4.12start	1070
10.286.5.Friends And Related Function Documentation	1071
10.286.5.1diff	1071
10.286.5.2inter	1071
10.286.5.3operator<	1071
10.286.5.4operator<<	1071
10.286.5.5operator<=	1071
10.286.5.6operator==	1072
10.286.5.7sym_diff	1072
10.286.5.8uni	1072
10.286.5.9unique	1072
10.287.Inl::p_set< P > Class Template Reference	1073
10.287.1.Detailed Description	1075
10.287.2.Member Typedef Documentation	1075
10.287.2.1bkd_piter	1075

10.287.2.2element	1075
10.287.2.3fwd_piter	1075
10.287.2.4i_element	1075
10.287.2.5piter	1075
10.287.2.6psite	1076
10.287.2.7r_element	1076
10.287.3Constructor & Destructor Documentation	1076
10.287.3.1p_set	1076
10.287.4Member Function Documentation	1076
10.287.4.1clear	1076
10.287.4.2has	1076
10.287.4.3has	1076
10.287.4.4has	1076
10.287.4.5insert	1076
10.287.4.6is_valid	1076
10.287.4.7memory_size	1077
10.287.4.8nsites	1077
10.287.4.9operator[1077
10.287.4.10move	1077
10.287.4.11std_vector	1077
10.287.4.12til_set	1077
10.287.5Friends And Related Function Documentation	1077
10.287.5.1diff	1077
10.287.5.2inter	1077
10.287.5.3operator<	1077
10.287.5.4operator<<	1078
10.287.5.5operator<=	1078
10.287.5.6operator==	1078
10.287.5.7sym_diff	1078
10.287.5.8uni	1078
10.287.5.9unique	1079
10.288In::p_set_of< S > Class Template Reference	1080
10.288.1Detailed Description	1082
10.288.2Member Typedef Documentation	1082
10.288.2.1bkd_piter	1082
10.288.2.2element	1082

10.288.2.3	10.288.2.3	1082
10.288.2.4	10.288.2.4	1082
10.288.2.5	10.288.2.5	1082
10.288.2.6	10.288.2.6	1082
10.288.3	10.288.3	1082
10.288.3.1	10.288.3.1	1082
10.288.4	10.288.4	1082
10.288.4.1	10.288.4.1	1082
10.288.4.2	10.288.4.2	1083
10.288.4.3	10.288.4.3	1083
10.288.4.4	10.288.4.4	1083
10.288.4.5	10.288.4.5	1083
10.288.4.6	10.288.4.6	1083
10.288.4.7	10.288.4.7	1083
10.288.5	10.288.5	1083
10.288.5.1	10.288.5.1	1083
10.288.5.2	10.288.5.2	1083
10.288.5.3	10.288.5.3	1083
10.288.5.4	10.288.5.4	1084
10.288.5.5	10.288.5.5	1084
10.288.5.6	10.288.5.6	1084
10.288.5.7	10.288.5.7	1084
10.288.5.8	10.288.5.8	1084
10.288.5.9	10.288.5.9	1084
10.289	10.289	1085
10.289.1	10.289.1	1085
10.289.2	10.289.2	1086
10.289.2.1	10.289.2.1	1087
10.289.2.2	10.289.2.2	1087
10.289.2.3	10.289.2.3	1087
10.289.2.4	10.289.2.4	1087
10.289.2.5	10.289.2.5	1087
10.289.3	10.289.3	1087
10.289.3.1	10.289.3.1	1087
10.289.3.2	10.289.3.2	1087
10.289.4	10.289.4	1087

10.289.4.1function	1087
10.289.4.2has	1088
10.289.4.3is_valid	1088
10.289.4.4memory_size	1088
10.289.4.5primary_set	1088
10.289.5Friends And Related Function Documentation	1088
10.289.5.1diff	1088
10.289.5.2inter	1088
10.289.5.3operator<	1088
10.289.5.4operator<<	1088
10.289.5.5operator<=	1089
10.289.5.6operator==	1089
10.289.5.7sym_diff	1089
10.289.5.8uni	1089
10.289.5.9unique	1089
10.290mln::p_transformed_piter< Pi, S, F > Struct Template Reference	1090
10.290.1Detailed Description	1090
10.290.2Constructor & Destructor Documentation	1090
10.290.2.1p_transformed_piter	1090
10.290.2.2p_transformed_piter	1090
10.290.3Member Function Documentation	1091
10.290.3.1change_target	1091
10.290.3.2next	1091
10.291mln::p_vaccess< V, S > Class Template Reference	1092
10.291.1Detailed Description	1094
10.291.2Member Typedef Documentation	1094
10.291.2.1bkd_piter	1094
10.291.2.2element	1094
10.291.2.3fwd_piter	1094
10.291.2.4_element	1094
10.291.2.5piter	1094
10.291.2.6pset	1094
10.291.2.7psite	1095
10.291.2.8value	1095
10.291.2.9vset	1095
10.291.3Constructor & Destructor Documentation	1095

10.291.3.1p_vaccess	1095
10.291.4Member Function Documentation	1095
10.291.4.1has	1095
10.291.4.2has	1095
10.291.4.3insert	1095
10.291.4.4insert	1095
10.291.4.5is_valid	1095
10.291.4.6memory_size	1096
10.291.4.7operator()	1096
10.291.4.8values	1096
10.291.5Friends And Related Function Documentation	1096
10.291.5.1diff	1096
10.291.5.2inter	1096
10.291.5.3operator<	1096
10.291.5.4operator<<	1096
10.291.5.5operator<=	1097
10.291.5.6operator==	1097
10.291.5.7sym_diff	1097
10.291.5.8uni	1097
10.291.5.9unique	1097
10.292mln::p_vertices< G, F > Class Template Reference	1098
10.292.1Detailed Description	1100
10.292.2Member Typedef Documentation	1100
10.292.2.1bkd_piter	1100
10.292.2.2element	1100
10.292.2.3fun_t	1101
10.292.2.4fwd_piter	1101
10.292.2.5graph_element	1101
10.292.2.6graph_t	1101
10.292.2.7piter	1101
10.292.2.8psite	1101
10.292.2.9vertex	1101
10.292.3Constructor & Destructor Documentation	1101
10.292.3.1p_vertices	1101
10.292.3.2p_vertices	1102
10.292.3.3p_vertices	1102

10.292.3.4	p_vertices	1102
10.292.3.5	sp_vertices	1102
10.292.4	Member Function Documentation	1102
10.292.4.1	function	1102
10.292.4.2	graph	1103
10.292.4.3	has	1103
10.292.4.4	has	1103
10.292.4.5	invalidate	1103
10.292.4.6	is_valid	1103
10.292.4.7	memory_size	1103
10.292.4.8	nsites	1103
10.292.4.9	nvertices	1104
10.292.4.10	operator()	1104
10.292.5	Friends And Related Function Documentation	1104
10.292.5.1	diff	1104
10.292.5.2	inter	1104
10.292.5.3	operator<	1104
10.292.5.4	operator<<	1104
10.292.5.5	operator<=	1105
10.292.5.6	operator==	1105
10.292.5.7	sym_diff	1105
10.292.5.8	uni	1105
10.292.5.9	unique	1105
10.293	mln::pixel< I > Struct Template Reference	1106
10.293.1	Detailed Description	1106
10.293.2	Constructor & Destructor Documentation	1106
10.293.2.1	pixel	1106
10.293.2.2	pixel	1106
10.293.3	Member Function Documentation	1107
10.293.3.1	lchange_to	1107
10.293.3.2	is_valid	1107
10.294	mln::Pixel_Iterator< E > Struct Template Reference	1108
10.294.1	Detailed Description	1108
10.294.2	Member Function Documentation	1108
10.294.2.1	lnext	1108
10.295	mln::plain< I > Class Template Reference	1110

10.295.1	Detailed Description	1110
10.295.2	Member Typedef Documentation	1110
10.295.2.1	lskeleton	1110
10.295.3	Constructor & Destructor Documentation	1111
10.295.3.1	lplain	1111
10.295.3.2	2plain	1111
10.295.3.3	3plain	1111
10.295.4	Member Function Documentation	1111
10.295.4.1	loperator I	1111
10.295.4.2	2operator=	1111
10.295.4.3	3operator=	1111
10.296	mln::Point< P > Struct Template Reference	1112
10.296.1	Detailed Description	1113
10.296.2	Member Typedef Documentation	1113
10.296.2.1	lpoint	1113
10.296.3	Member Function Documentation	1113
10.296.3.1	lto_point	1113
10.296.4	Friends And Related Function Documentation	1113
10.296.4.1	loperator+=	1113
10.296.4.2	2operator-=	1113
10.296.4.3	3operator/	1114
10.297	mln::point< G, C > Struct Template Reference	1115
10.297.1	Detailed Description	1117
10.297.2	Member Typedef Documentation	1118
10.297.2.1	lcoord	1118
10.297.2.2	2delta	1118
10.297.2.3	3dpsite	1118
10.297.2.4	4grid	1118
10.297.2.5	5h_vec	1118
10.297.2.6	6vec	1118
10.297.3	Member Enumeration Documentation	1118
10.297.3.1	l"@30	1118
10.297.4	Constructor & Destructor Documentation	1118
10.297.4.1	lpoint	1118
10.297.4.2	2point	1119
10.297.4.3	3point	1119

10.297.4.4point	1119
10.297.4.5point	1119
10.297.5Member Function Documentation	1119
10.297.5.1last_coord	1119
10.297.5.2last_coord	1119
10.297.5.3minus_infty	1119
10.297.5.4operator+=	1119
10.297.5.5operator-=	1120
10.297.5.6operator[1120
10.297.5.7operator[1120
10.297.5.8plus_infty	1120
10.297.5.9set_all	1120
10.297.5.10_h_vec	1121
10.297.5.11b_vec	1121
10.297.6Friends And Related Function Documentation	1121
10.297.6.1operator+	1121
10.297.6.2operator+=	1121
10.297.6.3operator-	1122
10.297.6.4operator-=	1122
10.297.6.5operator/	1123
10.297.6.6operator<<	1123
10.297.6.7operator==	1123
10.297.7Member Data Documentation	1123
10.297.7.1origin	1123
10.298In::Point_Site< E > Struct Template Reference	1124
10.298.1Detailed Description	1124
10.298.2Friends And Related Function Documentation	1125
10.298.2.1operator+	1125
10.298.2.2operator-	1125
10.298.2.3operator-	1126
10.298.2.4operator<<	1126
10.298.2.5operator==	1126
10.299In::Point_Site< void > Struct Template Reference	1128
10.299.1Detailed Description	1128
10.300In::Proxy< E > Struct Template Reference	1129
10.300.1Detailed Description	1129

10.301	Inl::Proxy< void > Struct Template Reference	1130
10.301	Detailed Description	1130
10.302	Inl::Pseudo_Site< E > Struct Template Reference	1131
10.302	Detailed Description	1131
10.303	Inl::Pseudo_Site< void > Struct Template Reference	1132
10.303	Detailed Description	1132
10.304	Inl::pw::image< F, S > Class Template Reference	1133
10.304	Detailed Description	1133
10.304	Member Typedef Documentation	1133
10.304.2	Iskeleton	1133
10.304	Constructor & Destructor Documentation	1133
10.304.3	limage	1133
10.304.3	2image	1133
10.305	Inl::registration::closest_point_basic< P > Class Template Reference	1134
10.305	Detailed Description	1134
10.306	Inl::registration::closest_point_with_map< P > Class Template Reference	1135
10.306	Detailed Description	1135
10.307	Inl::Regular_Grid< E > Struct Template Reference	1136
10.307	Detailed Description	1136
10.308	Inl::safe_image< I > Class Template Reference	1137
10.308	Detailed Description	1137
10.308	Member Typedef Documentation	1137
10.308.2	Iskeleton	1137
10.308	Member Function Documentation	1137
10.308.3	operator safe_image< const I >	1137
10.309	Inl::select::p_of< P > Struct Template Reference	1138
10.309	Detailed Description	1138
10.310	Inl::Site< E > Struct Template Reference	1139
10.310	Detailed Description	1139
10.311	Inl::Site< void > Struct Template Reference	1140
10.311	Detailed Description	1140
10.312	Inl::Site_Iterator< E > Struct Template Reference	1141
10.312	Detailed Description	1142
10.312	Member Function Documentation	1142
10.312.2	Inext	1142
10.313	Inl::Site_Proxy< E > Struct Template Reference	1143

10.313. Detailed Description	1143
10.314. <code>ln::Site_Proxy< void > Struct Template Reference</code>	1144
10.314. Detailed Description	1144
10.315. <code>ln::Site_Set< E > Struct Template Reference</code>	1145
10.315. Detailed Description	1146
10.315. Friends And Related Function Documentation	1146
10.315.2. <code>ldiff</code>	1146
10.315.2. <code>linter</code>	1146
10.315.2. <code>loperator<</code>	1147
10.315.2. <code>loperator<<</code>	1147
10.315.2. <code>loperator<=</code>	1147
10.315.2. <code>loperator==</code>	1147
10.315.2. <code>lsym_diff</code>	1147
10.315.2. <code>luni</code>	1148
10.315.2. <code>lunique</code>	1148
10.316. <code>ln::Site_Set< void > Struct Template Reference</code>	1149
10.316. Detailed Description	1149
10.317. <code>ln::slice_image< I > Struct Template Reference</code>	1150
10.317. Detailed Description	1150
10.317. Member Typedef Documentation	1150
10.317.2. <code>lskeleton</code>	1150
10.317. Constructor & Destructor Documentation	1151
10.317.3. <code>lslice_image</code>	1151
10.317.3. <code>lslice_image</code>	1151
10.317. Member Function Documentation	1151
10.317.4. <code>ldomain</code>	1151
10.317.4. <code>loperator slice_image< const I ></code>	1151
10.317.4. <code>loperator()</code>	1151
10.317.4. <code>loperator()</code>	1151
10.317.4. <code>lsli</code>	1151
10.318. <code>ln::sub_image< I, S > Struct Template Reference</code>	1152
10.318. Detailed Description	1152
10.318. Member Typedef Documentation	1152
10.318.2. <code>lskeleton</code>	1152
10.318. Constructor & Destructor Documentation	1152
10.318.3. <code>lsub_image</code>	1152

10.318.3.2sub_image	1153
10.318.4Member Function Documentation	1153
10.318.4.1domain	1153
10.318.4.2operator sub_image< const I, S >	1153
10.319Inl::sub_image_if< I, S > Struct Template Reference	1154
10.319.1Detailed Description	1154
10.319.2Member Typedef Documentation	1154
10.319.2.1skeleton	1154
10.319.3Constructor & Destructor Documentation	1154
10.319.3.1sub_image_if	1154
10.319.3.2sub_image_if	1155
10.319.4Member Function Documentation	1155
10.319.4.1domain	1155
10.320Inl::thru_image< I, F > Class Template Reference	1156
10.320.1Detailed Description	1156
10.320.2Member Function Documentation	1156
10.320.2.1operator thru_image< const I, F >	1156
10.321Inl::thrubin_image< I1, I2, F > Class Template Reference	1157
10.321.1Detailed Description	1157
10.321.2Member Typedef Documentation	1157
10.321.2.1psite	1157
10.321.2.2rvalue	1157
10.321.2.3skeleton	1158
10.321.2.4value	1158
10.321.3Member Function Documentation	1158
10.321.3.1operator thrubin_image< const I1, const I2, F >	1158
10.322Inl::topo::adj_higher_dim_connected_n_face_bkd_iter< D > Class Template Reference	1159
10.322.1Detailed Description	1159
10.322.2Constructor & Destructor Documentation	1159
10.322.2.1adj_higher_dim_connected_n_face_bkd_iter	1159
10.322.3Member Function Documentation	1159
10.322.3.1next	1159
10.323Inl::topo::adj_higher_dim_connected_n_face_fwd_iter< D > Class Template Reference	1161
10.323.1Detailed Description	1161
10.323.2Constructor & Destructor Documentation	1161
10.323.2.1adj_higher_dim_connected_n_face_fwd_iter	1161

10.323.3	Member Function Documentation	1161
10.323.3.1	next	1161
10.324	nl::topo::adj_higher_face_bkd_iter< D > Class Template Reference	1163
10.324.1	Detailed Description	1163
10.324.2	Constructor & Destructor Documentation	1163
10.324.2.1	adj_higher_face_bkd_iter	1163
10.324.3	Member Function Documentation	1163
10.324.3.1	next	1163
10.325	nl::topo::adj_higher_face_fwd_iter< D > Class Template Reference	1164
10.325.1	Detailed Description	1164
10.325.2	Constructor & Destructor Documentation	1164
10.325.2.1	adj_higher_face_fwd_iter	1164
10.325.3	Member Function Documentation	1164
10.325.3.1	next	1164
10.326	nl::topo::adj_lower_dim_connected_n_face_bkd_iter< D > Class Template Reference	1165
10.326.1	Detailed Description	1165
10.326.2	Constructor & Destructor Documentation	1165
10.326.2.1	adj_lower_dim_connected_n_face_bkd_iter	1165
10.326.3	Member Function Documentation	1165
10.326.3.1	next	1165
10.327	nl::topo::adj_lower_dim_connected_n_face_fwd_iter< D > Class Template Reference	1167
10.327.1	Detailed Description	1167
10.327.2	Constructor & Destructor Documentation	1167
10.327.2.1	adj_lower_dim_connected_n_face_fwd_iter	1167
10.327.3	Member Function Documentation	1167
10.327.3.1	next	1167
10.328	nl::topo::adj_lower_face_bkd_iter< D > Class Template Reference	1169
10.328.1	Detailed Description	1169
10.328.2	Constructor & Destructor Documentation	1169
10.328.2.1	adj_lower_face_bkd_iter	1169
10.328.3	Member Function Documentation	1169
10.328.3.1	next	1169
10.329	nl::topo::adj_lower_face_fwd_iter< D > Class Template Reference	1170
10.329.1	Detailed Description	1170
10.329.2	Constructor & Destructor Documentation	1170
10.329.2.1	adj_lower_face_fwd_iter	1170

10.329.3	Member Function Documentation	1170
10.329.3	lnext	1170
10.330	nl::topo::adj_lower_higher_face_bkd_iter< D > Class Template Reference	1171
10.330	Detailed Description	1171
10.330	Constructor & Destructor Documentation	1171
10.330.2	ladj_lower_higher_face_bkd_iter	1171
10.330	Member Function Documentation	1171
10.330.3	lnext	1171
10.331	nl::topo::adj_lower_higher_face_fwd_iter< D > Class Template Reference	1172
10.331	Detailed Description	1172
10.331	Constructor & Destructor Documentation	1172
10.331.2	ladj_lower_higher_face_fwd_iter	1172
10.331	Member Function Documentation	1172
10.331.3	lnext	1172
10.332	nl::topo::adj_m_face_bkd_iter< D > Class Template Reference	1173
10.332	Detailed Description	1173
10.332	Constructor & Destructor Documentation	1173
10.332.2	ladj_m_face_bkd_iter	1173
10.332.2	2adj_m_face_bkd_iter	1173
10.332	Member Function Documentation	1174
10.332.3	lnext	1174
10.333	nl::topo::adj_m_face_fwd_iter< D > Class Template Reference	1175
10.333	Detailed Description	1175
10.333	Constructor & Destructor Documentation	1175
10.333.2	ladj_m_face_fwd_iter	1175
10.333.2	2adj_m_face_fwd_iter	1175
10.333	Member Function Documentation	1176
10.333.3	lnext	1176
10.334	nl::topo::algebraic_face< D > Struct Template Reference	1177
10.334	Detailed Description	1178
10.334	Constructor & Destructor Documentation	1178
10.334.2	lalgebraic_face	1178
10.334.2	2algebraic_face	1178
10.334.2	3algebraic_face	1179
10.334.2	4algebraic_face	1179
10.334	Member Function Documentation	1179

10.334.3.1cplx	1179
10.334.3.2data	1179
10.334.3.3dec_face_id	1179
10.334.3.4dec_n	1179
10.334.3.5face_id	1179
10.334.3.6higher_dim_adj_faces	1180
10.334.3.7nc_face_id	1180
10.334.3.8nc_n	1180
10.334.3.9invalidate	1180
10.334.3.10s_valid	1180
10.334.3.11bwer_dim_adj_faces	1180
10.334.3.12	1180
10.334.3.13set_cplx	1180
10.334.3.14set_face_id	1181
10.334.3.15set_n	1181
10.334.3.16set_sign	1181
10.334.3.17gn	1181
10.335mln::topo::algebraic_n_face< N, D > Class Template Reference	1182
10.335.1Detailed Description	1183
10.335.2Constructor & Destructor Documentation	1183
10.335.2.1algebraic_n_face	1183
10.335.2.2algebraic_n_face	1183
10.335.2.3algebraic_n_face	1183
10.335.3Member Function Documentation	1184
10.335.3.1cplx	1184
10.335.3.2data	1184
10.335.3.3dec_face_id	1184
10.335.3.4face_id	1184
10.335.3.5higher_dim_adj_faces	1184
10.335.3.6nc_face_id	1184
10.335.3.7invalidate	1184
10.335.3.8s_valid	1185
10.335.3.9lower_dim_adj_faces	1185
10.335.3.10	1185
10.335.3.11set_cplx	1185
10.335.3.12set_face_id	1185

10.335.3.1	set_sign	1185
10.335.3.1	sign	1185
10.336	nl::topo::center_only_iter< D > Class Template Reference	1186
10.336.1	Detailed Description	1186
10.336.2	Constructor & Destructor Documentation	1186
10.336.2.1	center_only_iter	1186
10.336.3	Member Function Documentation	1187
10.336.3.1	next	1187
10.337	nl::topo::centered_bkd_iter_adapter< D, I > Class Template Reference	1188
10.337.1	Detailed Description	1188
10.337.2	Constructor & Destructor Documentation	1188
10.337.2.1	centered_bkd_iter_adapter	1188
10.337.3	Member Function Documentation	1188
10.337.3.1	next	1188
10.338	nl::topo::centered_fwd_iter_adapter< D, I > Class Template Reference	1189
10.338.1	Detailed Description	1189
10.338.2	Constructor & Destructor Documentation	1189
10.338.2.1	centered_fwd_iter_adapter	1189
10.338.3	Member Function Documentation	1189
10.338.3.1	next	1189
10.339	nl::topo::complex< D > Class Template Reference	1190
10.339.1	Detailed Description	1191
10.339.2	Member Typedef Documentation	1191
10.339.2.1	bkd_citer	1191
10.339.2.2	fwd_citer	1191
10.339.3	Constructor & Destructor Documentation	1191
10.339.3.1	complex	1191
10.339.4	Member Function Documentation	1191
10.339.4.1	add_face	1191
10.339.4.2	add_face	1191
10.339.4.3	addr	1191
10.339.4.4	nfaces	1192
10.339.4.5	nfaces_of_dim	1192
10.339.4.6	nfaces_of_static_dim	1192
10.339.4.7	print	1192
10.339.4.8	print_faces	1192

10.340	<code>ln::topo::face< D ></code> Struct Template Reference	1193
10.340.	Detailed Description	1194
10.340.	Constructor & Destructor Documentation	1194
10.340.2.	<code>iface</code>	1194
10.340.2.2	<code>face</code>	1194
10.340.2.3	<code>face</code>	1194
10.340.3	Member Function Documentation	1194
10.340.3.1	<code>cplx</code>	1194
10.340.3.2	<code>data</code>	1195
10.340.3.3	<code>dec_face_id</code>	1195
10.340.3.4	<code>dec_n</code>	1195
10.340.3.5	<code>face_id</code>	1195
10.340.3.6	<code>higher_dim_adj_faces</code>	1195
10.340.3.7	<code>inc_face_id</code>	1195
10.340.3.8	<code>inc_n</code>	1195
10.340.3.9	<code>invalidate</code>	1195
10.340.3.10	<code>valid</code>	1195
10.340.3.11	<code>bwer_dim_adj_faces</code>	1196
10.340.3.12		1196
10.340.3.13	<code>set_cplx</code>	1196
10.340.3.14	<code>set_face_id</code>	1196
10.340.3.15	<code>set_n</code>	1196
10.341	<code>ln::topo::face_bkd_iter< D ></code> Class Template Reference	1197
10.341.	Detailed Description	1197
10.341.	Constructor & Destructor Documentation	1197
10.341.2.	<code>iface_bkd_iter</code>	1197
10.341.3	Member Function Documentation	1197
10.341.3.1	<code>next</code>	1197
10.341.3.2	<code>start</code>	1198
10.342	<code>ln::topo::face_fwd_iter< D ></code> Class Template Reference	1199
10.342.	Detailed Description	1199
10.342.	Constructor & Destructor Documentation	1199
10.342.2.	<code>iface_fwd_iter</code>	1199
10.342.3	Member Function Documentation	1199
10.342.3.1	<code>next</code>	1199
10.342.3.2	<code>start</code>	1200

10.343	mIn::topo::is_n_face< N > Struct Template Reference	1201
10.343	Detailed Description	1201
10.344	mIn::topo::is_simple_cell< I > Class Template Reference	1202
10.344	Detailed Description	1203
10.344	Member Typedef Documentation	1203
10.344.2	lpsite	1203
10.344.2	result	1203
10.344	Member Function Documentation	1203
10.344.3	lmln_geom	1203
10.344.3	operator()	1203
10.344.3	set_image	1203
10.344	Member Data Documentation	1203
10.344.4	ID	1203
10.345	mIn::topo::n_face< N, D > Class Template Reference	1204
10.345	Detailed Description	1205
10.345	Constructor & Destructor Documentation	1205
10.345.2	ln_face	1205
10.345.2	n_face	1205
10.345	Member Function Documentation	1205
10.345.3	lcplx	1205
10.345.3	data	1205
10.345.3	dec_face_id	1205
10.345.3	face_id	1206
10.345.3	higher_dim_adj_faces	1206
10.345.3	inc_face_id	1206
10.345.3	invalidate	1206
10.345.3	is_valid	1206
10.345.3	lower_dim_adj_faces	1206
10.345.3	l0	1206
10.345.3	lset_cplx	1207
10.345.3	lset_face_id	1207
10.346	mIn::topo::n_face_bkd_iter< D > Class Template Reference	1208
10.346	Detailed Description	1208
10.346	Constructor & Destructor Documentation	1208
10.346.2	ln_face_bkd_iter	1208
10.346	Member Function Documentation	1208

10.346.3.1n	1208
10.346.3.2next	1209
10.346.3.3start	1209
10.347.1n::topo::n_face_fwd_iter< D > Class Template Reference	1210
10.347.1.Detailed Description	1210
10.347.2.Constructor & Destructor Documentation	1210
10.347.2.1n_face_fwd_iter	1210
10.347.3.Member Function Documentation	1210
10.347.3.1n	1210
10.347.3.2next	1211
10.347.3.3start	1211
10.348.1n::topo::n_faces_set< N, D > Class Template Reference	1212
10.348.1.Detailed Description	1212
10.348.2.Member Typedef Documentation	1212
10.348.2.1faces_type	1212
10.348.3.Member Function Documentation	1212
10.348.3.1add	1212
10.348.3.2faces	1213
10.348.3.3reserve	1213
10.349.1n::topo::static_n_face_bkd_iter< N, D > Class Template Reference	1214
10.349.1.Detailed Description	1214
10.349.2.Constructor & Destructor Documentation	1214
10.349.2.1static_n_face_bkd_iter	1214
10.349.3.Member Function Documentation	1214
10.349.3.1next	1214
10.349.3.2start	1215
10.350.1n::topo::static_n_face_fwd_iter< N, D > Class Template Reference	1216
10.350.1.Detailed Description	1216
10.350.2.Constructor & Destructor Documentation	1216
10.350.2.1static_n_face_fwd_iter	1216
10.350.3.Member Function Documentation	1216
10.350.3.1next	1216
10.350.3.2start	1217
10.351.1n::tr_image< S, I, T > Struct Template Reference	1218
10.351.1.Detailed Description	1219
10.351.2.Member Typedef Documentation	1219

10.351.2.1lvalue	1219
10.351.2.2psite	1219
10.351.2.3rvalue	1219
10.351.2.4site	1219
10.351.2.5skeleton	1219
10.351.2.6value	1219
10.351.3Constructor & Destructor Documentation	1219
10.351.3.1tr_image	1219
10.351.4Member Function Documentation	1220
10.351.4.1domain	1220
10.351.4.2has	1220
10.351.4.3is_valid	1220
10.351.4.4operator()	1220
10.351.4.5set_tr	1220
10.351.4.6tr	1220
10.352mln::transformed_image< I, F > Struct Template Reference	1221
10.352.1Detailed Description	1221
10.352.2Member Typedef Documentation	1221
10.352.2.1skeleton	1221
10.352.3Constructor & Destructor Documentation	1222
10.352.3.1transformed_image	1222
10.352.3.2transformed_image	1222
10.352.4Member Function Documentation	1222
10.352.4.1domain	1222
10.352.4.2operator transformed_image< const I, F >	1222
10.352.4.3operator()	1222
10.352.4.4operator()	1222
10.353mln::unproject_image< I, D, F > Struct Template Reference	1223
10.353.1Detailed Description	1223
10.353.2Constructor & Destructor Documentation	1223
10.353.2.1unproject_image	1223
10.353.2.2unproject_image	1223
10.353.3Member Function Documentation	1223
10.353.3.1domain	1223
10.353.3.2operator()	1224
10.353.3.3operator()	1224

10.354	<code>std::adjacency_matrix< V ></code> Class Template Reference	1225
10.354.1	Detailed Description	1225
10.354.2	Constructor & Destructor Documentation	1225
10.354.2.1	<code>adjacency_matrix</code>	1225
10.354.2.2	<code>adjacency_matrix</code>	1225
10.355	<code>std::array< T ></code> Class Template Reference	1226
10.355.1	Detailed Description	1228
10.355.2	Member Typedef Documentation	1228
10.355.2.1	<code>lback_ite</code>	1228
10.355.2.2	<code>back_ite</code>	1228
10.355.2.3	<code>element</code>	1228
10.355.2.4	<code>front_ite</code>	1228
10.355.2.5	<code>result</code>	1228
10.355.3	Constructor & Destructor Documentation	1228
10.355.3.1	<code>larray</code>	1228
10.355.3.2	<code>array</code>	1228
10.355.3.3	<code>array</code>	1229
10.355.4	Member Function Documentation	1229
10.355.4.1	<code>lappend</code>	1229
10.355.4.2	<code>append</code>	1229
10.355.4.3	<code>clear</code>	1229
10.355.4.4	<code>fill</code>	1229
10.355.4.5	<code>is_empty</code>	1229
10.355.4.6	<code>memory_size</code>	1229
10.355.4.7	<code>elements</code>	1230
10.355.4.8	<code>operator()</code>	1230
10.355.4.9	<code>operator()</code>	1230
10.355.4.10	<code>operator[]</code>	1230
10.355.4.11	<code>operator[]</code>	1230
10.355.4.12	<code>reserve</code>	1231
10.355.4.13	<code>size</code>	1231
10.355.4.14	<code>size</code>	1231
10.355.4.15	<code>size</code>	1231
10.355.4.16	<code>id_vector</code>	1231
10.356	<code>std::branch< T ></code> Class Template Reference	1232
10.356.1	Detailed Description	1232

10.356.2	Constructor & Destructor Documentation	1232
10.356.2.1	branch	1232
10.356.3	Member Function Documentation	1232
10.356.3.1	lapex	1232
10.356.3.2	util_tree	1233
10.357	util::branch_iter< T > Class Template Reference	1234
10.357.1	Detailed Description	1234
10.357.2	Member Function Documentation	1234
10.357.2.1	deepness	1234
10.357.2.2	invalidate	1234
10.357.2.3	is_valid	1235
10.357.2.4	next	1235
10.357.2.5	operator util::tree_node< T > &	1235
10.357.2.6	start	1235
10.358	util::branch_iter_ind< T > Class Template Reference	1236
10.358.1	Detailed Description	1236
10.358.2	Member Function Documentation	1236
10.358.2.1	deepness	1236
10.358.2.2	invalidate	1236
10.358.2.3	is_valid	1237
10.358.2.4	next	1237
10.358.2.5	operator util::tree_node< T > &	1237
10.358.2.6	start	1237
10.359	util::couple< T, U > Class Template Reference	1238
10.359.1	Detailed Description	1238
10.359.2	Member Function Documentation	1238
10.359.2.1	change_both	1238
10.359.2.2	change_first	1238
10.359.2.3	change_second	1239
10.359.2.4	first	1239
10.359.2.5	second	1239
10.360	util::eat Struct Reference	1240
10.360.1	Detailed Description	1240
10.361	util::edge< G > Class Template Reference	1241
10.361.1	Detailed Description	1242
10.361.2	Member Typedef Documentation	1242

10.361.2.1category	1242
10.361.2.2graph_t	1242
10.361.2.3id_t	1242
10.361.2.4id_value_t	1242
10.361.3Constructor & Destructor Documentation	1242
10.361.3.1edge	1242
10.361.4Member Function Documentation	1243
10.361.4.1change_graph	1243
10.361.4.2graph	1243
10.361.4.3id	1243
10.361.4.4invalidate	1243
10.361.4.5is_valid	1243
10.361.4.6th_nbh_edge	1243
10.361.4.7nmax_nbh_edges	1243
10.361.4.8operator edge_id_t	1243
10.361.4.9update_id	1244
10.361.4.10l	1244
10.361.4.112	1244
10.361.4.12other	1244
10.362In::util::fibonacci_heap< P, T > Class Template Reference	1245
10.362.1Detailed Description	1246
10.362.2Constructor & Destructor Documentation	1246
10.362.2.1fibonacci_heap	1246
10.362.2.2fibonacci_heap	1246
10.362.3Member Function Documentation	1246
10.362.3.1clear	1246
10.362.3.2front	1246
10.362.3.3is_empty	1246
10.362.3.4is_valid	1246
10.362.3.5elements	1247
10.362.3.6operator=	1247
10.362.3.7pop_front	1247
10.362.3.8push	1247
10.362.3.9push	1247
10.363In::util::graph Class Reference	1248
10.363.1Detailed Description	1250

10.363.2	Member Typedef Documentation	1250
10.363.2.1	edge_fwd_iter	1250
10.363.2.2	edge_nbh_edge_fwd_iter	1250
10.363.2.3	edges_set_t	1250
10.363.2.4	edges_t	1250
10.363.2.5	vertex_fwd_iter	1250
10.363.2.6	vertex_nbh_edge_fwd_iter	1250
10.363.2.7	vertex_nbh_vertex_fwd_iter	1250
10.363.2.8	vertices_t	1251
10.363.3	Constructor & Destructor Documentation	1251
10.363.3.1	lgraph	1251
10.363.3.2	graph	1251
10.363.4	Member Function Documentation	1251
10.363.4.1	add_edge	1251
10.363.4.2	add_vertex	1251
10.363.4.3	add_vertices	1251
10.363.4.4	e_ith_nbh_edge	1252
10.363.4.5	e_nmax	1252
10.363.4.6	e_nmax_nbh_edges	1252
10.363.4.7	edge	1252
10.363.4.8	edge	1252
10.363.4.9	edges	1252
10.363.4.10	has_e	1252
10.363.4.11	has_v	1253
10.363.4.12	is_subgraph_of	1253
10.363.4.13	l1	1253
10.363.4.14	l2	1253
10.363.4.15	l5_ith_nbh_edge	1253
10.363.4.16	l6_ith_nbh_vertex	1253
10.363.4.17	l7_nmax	1253
10.363.4.18	l8_nmax_nbh_edges	1253
10.363.4.19	l9_nmax_nbh_vertices	1254
10.363.4.20	lvertex	1254
10.364	util::greater_point< I > Class Template Reference	1255
10.364.1	Detailed Description	1255
10.364.2	Member Function Documentation	1255

10.364.2.operator()	1255
10.365.inl::util::greater_site< I > Class Template Reference	1256
10.365.Detailed Description	1256
10.365.Member Function Documentation	1256
10.365.2.operator()	1256
10.366.inl::util::head< T, R > Class Template Reference	1257
10.366.Detailed Description	1257
10.367.inl::util::ignore Struct Reference	1258
10.367.Detailed Description	1258
10.368.inl::util::ilcell< T > Struct Template Reference	1259
10.368.Detailed Description	1259
10.369.inl::util::line_graph< G > Class Template Reference	1260
10.369.Detailed Description	1262
10.369.Member Typedef Documentation	1262
10.369.2.ledge_fwd_iter	1262
10.369.2.2edge_nbh_edge_fwd_iter	1262
10.369.2.3edges_t	1262
10.369.2.4vertex_fwd_iter	1262
10.369.2.5vertex_nbh_edge_fwd_iter	1262
10.369.2.6vertex_nbh_vertex_fwd_iter	1262
10.369.2.7vertices_t	1262
10.369.3.Member Function Documentation	1263
10.369.3.1e_ith_nbh_edge	1263
10.369.3.2e_nmax	1263
10.369.3.3e_nmax_nbh_edges	1263
10.369.3.4edge	1263
10.369.3.5graph	1263
10.369.3.6has	1263
10.369.3.7has	1264
10.369.3.8has_e	1264
10.369.3.9has_v	1264
10.369.3.10_subgraph_of	1264
10.369.3.11	1264
10.369.3.12	1264
10.369.3.13_ith_nbh_edge	1265
10.369.3.14_ith_nbh_vertex	1265

10.369.3.15	<code>l5_nmax</code>	1265
10.369.3.16	<code>l6_nmax_nbh_edges</code>	1265
10.369.3.17	<code>l7_nmax_nbh_vertices</code>	1265
10.369.3.18	<code>l8_vertex</code>	1265
10.370	<code>l9_nil</code> Struct Reference	1266
10.370.1	Detailed Description	1266
10.371	<code>l10_node</code> < T, R > Class Template Reference	1267
10.371.1	Detailed Description	1267
10.372	<code>l11_object_id</code> < Tag, V > Class Template Reference	1268
10.372.1	Detailed Description	1268
10.372.2	Member Typedef Documentation	1268
10.372.2.1	<code>lvalue_t</code>	1268
10.372.3	Constructor & Destructor Documentation	1268
10.372.3.1	<code>lobject_id</code>	1268
10.373	<code>l12_ord</code> < T > Struct Template Reference	1269
10.373.1	Detailed Description	1269
10.374	<code>l13_ord_pair</code> < T > Struct Template Reference	1270
10.374.1	Detailed Description	1270
10.374.2	Member Function Documentation	1270
10.374.2.1	<code>lchange_both</code>	1270
10.374.2.2	<code>lchange_first</code>	1271
10.374.2.3	<code>lchange_second</code>	1271
10.374.2.4	<code>lfirst</code>	1271
10.374.2.5	<code>lsecond</code>	1271
10.375	<code>l14_pix</code> < I > Struct Template Reference	1272
10.375.1	Detailed Description	1272
10.375.2	Member Typedef Documentation	1272
10.375.2.1	<code>lpsite</code>	1272
10.375.2.2	<code>lvalue</code>	1272
10.375.3	Constructor & Destructor Documentation	1273
10.375.3.1	<code>lpix</code>	1273
10.375.4	Member Function Documentation	1273
10.375.4.1	<code>llima</code>	1273
10.375.4.2	<code>lp</code>	1273
10.375.4.3	<code>lv</code>	1273
10.376	<code>l15_set</code> < T > Class Template Reference	1274

10.376. Detailed Description	1275
10.376. Member Typedef Documentation	1275
10.376.2. lbkd_eiter	1275
10.376.2. 2eiter	1276
10.376.2. 3element	1276
10.376.2. 4fwd_eiter	1276
10.376. Constructor & Destructor Documentation	1276
10.376.3. lset	1276
10.376. Member Function Documentation	1276
10.376.4. lclear	1276
10.376.4. 2first_element	1276
10.376.4. 3has	1276
10.376.4. 4insert	1277
10.376.4. 5insert	1277
10.376.4. 6is_empty	1277
10.376.4. 7last_element	1277
10.376.4. 8memory_size	1278
10.376.4. 9elements	1278
10.376.4. 10operator[.	1278
10.376.4. 11remove	1278
10.376.4. 12std_vector	1278
10.377. mln::util::site_pair< P > Class Template Reference	1280
10.377. Detailed Description	1280
10.377. Member Function Documentation	1280
10.377.2. lfirst	1280
10.377.2. 2pair	1280
10.377.2. 3second	1280
10.378. mln::util::soft_heap< T, R > Class Template Reference	1281
10.378. Detailed Description	1282
10.378. Member Typedef Documentation	1282
10.378.2. lelement	1282
10.378. Constructor & Destructor Documentation	1282
10.378.3. lsoft_heap	1282
10.378.3. 2~soft_heap	1282
10.378. Member Function Documentation	1282
10.378.4. lclear	1282

10.378.4.2	is_empty	1282
10.378.4.3	is_valid	1282
10.378.4.4	elements	1283
10.378.4.5	pop_front	1283
10.378.4.6	push	1283
10.378.4.7	push	1283
10.379	nl::util::timer Class Reference	1284
10.379	Detailed Description	1284
10.380	nl::util::tracked_ptr< T > Struct Template Reference	1285
10.380	Detailed Description	1285
10.380	Constructor & Destructor Documentation	1285
10.380.2.1	tracked_ptr	1285
10.380.2.2	tracked_ptr	1286
10.380.2.3	~tracked_ptr	1286
10.380.3	Member Function Documentation	1286
10.380.3.1	operator bool	1286
10.380.3.2	operator"	1286
10.380.3.3	operator->	1286
10.380.3.4	operator->	1286
10.380.3.5	operator=	1286
10.380.3.6	operator=	1286
10.381	nl::util::tree< T > Class Template Reference	1287
10.381	Detailed Description	1287
10.381	Constructor & Destructor Documentation	1287
10.381.2.1	tree	1287
10.381.2.2	tree	1287
10.381.3	Member Function Documentation	1288
10.381.3.1	add_tree_down	1288
10.381.3.2	add_tree_up	1288
10.381.3.3	check_consistency	1288
10.381.3.4	main_branch	1288
10.381.3.5	root	1288
10.382	nl::util::tree_node< T > Class Template Reference	1289
10.382	Detailed Description	1290
10.382	Constructor & Destructor Documentation	1290
10.382.2.1	tree_node	1290

10.382.2	<code>tree_node</code>	1290
10.382.3	Member Function Documentation	1290
10.382.3.1	<code>add_child</code>	1290
10.382.3.2	<code>add_child</code>	1290
10.382.3.3	<code>check_consistency</code>	1291
10.382.3.4	<code>children</code>	1291
10.382.3.5	<code>children</code>	1291
10.382.3.6	<code>delete_tree_node</code>	1291
10.382.3.7	<code>elt</code>	1291
10.382.3.8	<code>elt</code>	1291
10.382.3.9	<code>parent</code>	1292
10.382.3.10	<code>print</code>	1292
10.382.3.11	<code>search</code>	1292
10.382.3.12	<code>search_rec</code>	1292
10.382.3.13	<code>set_parent</code>	1292
10.383	<code>util::vertex< G ></code> Class Template Reference	1293
10.383.1	Detailed Description	1294
10.383.2	Member Typedef Documentation	1294
10.383.2.1	<code>Category</code>	1294
10.383.2.2	<code>graph_t</code>	1294
10.383.2.3	<code>id_t</code>	1294
10.383.2.4	<code>id_value_t</code>	1295
10.383.3	Constructor & Destructor Documentation	1295
10.383.3.1	<code>vertex</code>	1295
10.383.4	Member Function Documentation	1295
10.383.4.1	<code>change_graph</code>	1295
10.383.4.2	<code>edge_with</code>	1295
10.383.4.3	<code>graph</code>	1295
10.383.4.4	<code>id</code>	1295
10.383.4.5	<code>invalidate</code>	1295
10.383.4.6	<code>is_valid</code>	1295
10.383.4.7	<code>nth_nbh_edge</code>	1296
10.383.4.8	<code>nth_nbh_vertex</code>	1296
10.383.4.9	<code>nmax_nbh_edges</code>	1296
10.383.4.10	<code>nmax_nbh_vertices</code>	1296
10.383.4.11	<code>operator vertex_id_t</code>	1296

10.383.4.1other	1296
10.383.4.1update_id	1296
10.384.1In::util::yes Struct Reference	1297
10.384.1Detailed Description	1297
10.385.1In::Value< E > Struct Template Reference	1298
10.385.1Detailed Description	1298
10.386.1In::value::float01 Class Reference	1299
10.386.1Detailed Description	1300
10.386.2Member Typedef Documentation	1300
10.386.2.1enc	1300
10.386.2.2equiv	1300
10.386.3Constructor & Destructor Documentation	1300
10.386.3.1float01	1300
10.386.3.2float01	1300
10.386.3.3float01	1300
10.386.4Member Function Documentation	1300
10.386.4.1Inbits	1300
10.386.4.2operator float	1300
10.386.4.3set_nbits	1300
10.386.4.4to_nbits	1301
10.386.4.5value	1301
10.386.4.6value_ind	1301
10.387.1In::value::float01_f Struct Reference	1302
10.387.1Detailed Description	1302
10.387.2Constructor & Destructor Documentation	1302
10.387.2.1float01_f	1302
10.387.2.2float01_f	1302
10.387.3Member Function Documentation	1302
10.387.3.1operator float	1302
10.387.3.2operator=	1303
10.387.3.3value	1303
10.388.1In::value::graylevel< n > Struct Template Reference	1304
10.388.1Detailed Description	1305
10.388.2Constructor & Destructor Documentation	1305
10.388.2.1graylevel	1305
10.388.2.2graylevel	1305

10.388.2.3graylevel	1305
10.388.2.4graylevel	1305
10.388.2.5graylevel	1305
10.388.3Member Function Documentation	1305
10.388.3.1operator=	1305
10.388.3.2operator=	1305
10.388.3.3operator=	1306
10.388.3.4operator=	1306
10.388.3.5to_float	1306
10.388.3.6value	1306
10.389In::value::graylevel_f Struct Reference	1307
10.389.Detailed Description	1308
10.389.Constructor & Destructor Documentation	1308
10.389.2.1graylevel_f	1308
10.389.2.2graylevel_f	1308
10.389.2.3graylevel_f	1308
10.389.2.4graylevel_f	1308
10.389.2.5graylevel_f	1308
10.389.3Member Function Documentation	1308
10.389.3.1operator graylevel< n >	1308
10.389.3.2operator=	1308
10.389.3.3operator=	1308
10.389.3.4operator=	1309
10.389.3.5operator=	1309
10.389.3.6value	1309
10.390In::value::int_s< n > Struct Template Reference	1310
10.390.Detailed Description	1310
10.390.Constructor & Destructor Documentation	1311
10.390.2.1int_s	1311
10.390.2.2int_s	1311
10.390.2.3int_s	1311
10.390.3Member Function Documentation	1311
10.390.3.1operator int	1311
10.390.3.2operator=	1311
10.390.4Member Data Documentation	1311
10.390.4.1one	1311

10.395.3.3label	1319
10.395.4Member Function Documentation	1319
10.395.4.1next	1319
10.395.4.2operator unsigned	1319
10.395.4.3operator++	1319
10.395.4.4operator--	1320
10.395.4.5operator=	1320
10.395.4.6operator=	1320
10.395.4.7prev	1320
10.396mln::value::lut_vec< S, T > Struct Template Reference	1321
10.396.1Detailed Description	1322
10.396.2Member Typedef Documentation	1322
10.396.2.1bkd_viter	1322
10.396.2.2fwd_viter	1322
10.396.2.3value	1322
10.396.3Constructor & Destructor Documentation	1322
10.396.3.1lut_vec	1322
10.396.3.2lut_vec	1322
10.396.3.3lut_vec	1323
10.396.4Member Function Documentation	1323
10.396.4.1has	1323
10.396.4.2index_of	1323
10.396.4.3nvalues	1323
10.396.4.4operator[.	1323
10.397mln::value::proxy< I > Class Template Reference	1324
10.397.1Detailed Description	1325
10.397.2Member Typedef Documentation	1325
10.397.2.1enc	1325
10.397.2.2equiv	1325
10.397.3Constructor & Destructor Documentation	1325
10.397.3.1proxy	1325
10.397.3.2proxy	1325
10.397.3.3~proxy	1325
10.397.4Member Function Documentation	1325
10.397.4.1operator=	1325
10.397.4.2operator=	1325

10.397.4.3to_value	1326
10.398.1.1ln::value::rgb< n > Struct Template Reference	1327
10.398.1.1.1Detailed Description	1327
10.398.1.1.2Constructor & Destructor Documentation	1327
10.398.1.1.2.1rgb	1327
10.398.1.1.2.2rgb	1328
10.398.1.1.2.3rgb	1328
10.398.1.1.2.4rgb	1328
10.398.1.1.3Member Function Documentation	1328
10.398.1.1.3.1operator=	1328
10.398.1.1.3.2red	1328
10.398.1.1.4Member Data Documentation	1328
10.398.1.1.4.1zero	1328
10.399.1.1ln::value::set< T > Struct Template Reference	1329
10.399.1.1.1Detailed Description	1329
10.399.1.1.2Member Function Documentation	1329
10.399.1.1.2.1the	1329
10.400.1.1ln::value::sign Class Reference	1330
10.400.1.1.1Detailed Description	1330
10.400.1.1.2Member Typedef Documentation	1331
10.400.1.1.2.1lenc	1331
10.400.1.1.2.2equiv	1331
10.400.1.1.3Constructor & Destructor Documentation	1331
10.400.1.1.3.1sign	1331
10.400.1.1.3.2sign	1331
10.400.1.1.3.3sign	1331
10.400.1.1.4Member Function Documentation	1331
10.400.1.1.4.1operator int	1331
10.400.1.1.4.2operator=	1331
10.400.1.1.5Member Data Documentation	1331
10.400.1.1.5.1one	1331
10.400.1.1.5.2zero	1331
10.401.1.1ln::value::stack_image< n, I > Struct Template Reference	1332
10.401.1.1.1Detailed Description	1333
10.401.1.1.2Member Typedef Documentation	1333
10.401.1.1.2.1ldomain_t	1333

10.401.2.2	value	1333
10.401.2.3	psite	1333
10.401.2.4	rvalue	1333
10.401.2.5	skeleton	1333
10.401.2.6	value	1333
10.401.3	Constructor & Destructor Documentation	1334
10.401.3.1	stack_image	1334
10.401.4	Member Function Documentation	1334
10.401.4.1	lis_valid	1334
10.401.4.2	operator()	1334
10.401.4.3	operator()	1334
10.402	mln::value::super_value< sign > Struct Template Reference	1335
10.402.1	Detailed Description	1335
10.403	mln::value::value_array< T, V > Struct Template Reference	1336
10.403.1	Detailed Description	1336
10.403.2	Constructor & Destructor Documentation	1336
10.403.2.1	value_array	1336
10.403.3	Member Function Documentation	1336
10.403.3.1	operator()	1336
10.403.3.2	operator[1336
10.403.3.3	vset	1337
10.404	mln::Value_Iterator< E > Struct Template Reference	1338
10.404.1	Detailed Description	1338
10.404.2	Member Function Documentation	1338
10.404.2.1	next	1338
10.404.3	Friends And Related Function Documentation	1339
10.404.3.1	operator<<	1339
10.405	mln::Value_Set< E > Struct Template Reference	1340
10.405.1	Detailed Description	1340
10.406	mln::Vertex< E > Struct Template Reference	1341
10.406.1	Detailed Description	1341
10.407	mln::vertex_image< P, V, G > Class Template Reference	1342
10.407.1	Detailed Description	1342
10.407.2	Member Typedef Documentation	1343
10.407.2.1	lgraph_t	1343
10.407.2.2	nbh_t	1343

10.407.2.3site_function_t	1343
10.407.2.4skeleton	1343
10.407.2.5vertex_nbh_t	1343
10.407.2.6vertex_win_t	1343
10.407.2.7win_t	1343
10.407.3Constructor & Destructor Documentation	1343
10.407.3.1vertex_image	1343
10.407.4Member Function Documentation	1344
10.407.4.1operator()	1344
10.408In::violent_cast_image< T, I > Struct Template Reference	1345
10.408.1Detailed Description	1345
10.408.2Member Typedef Documentation	1345
10.408.2.1lvalue	1345
10.408.2.2rvalue	1346
10.408.2.3skeleton	1346
10.408.2.4value	1346
10.408.3Constructor & Destructor Documentation	1346
10.408.3.1violent_cast_image	1346
10.408.4Member Function Documentation	1346
10.408.4.1operator()	1346
10.408.4.2operator()	1346
10.409In::w_window< D, W > Struct Template Reference	1347
10.409.1Detailed Description	1348
10.409.2Member Typedef Documentation	1348
10.409.2.1bkd_qiter	1348
10.409.2.2dpsite	1348
10.409.2.3fwd_qiter	1348
10.409.2.4weight	1348
10.409.3Constructor & Destructor Documentation	1349
10.409.3.1w_window	1349
10.409.4Member Function Documentation	1349
10.409.4.1clear	1349
10.409.4.2insert	1349
10.409.4.3is_symmetric	1349
10.409.4.4std_vector	1349
10.409.4.5sym	1349

10.409.4.6w	1349
10.409.4.7weights	1350
10.409.4.8win	1350
10.409.5.Friends And Related Function Documentation	1350
10.409.5.1operator-	1350
10.409.5.2operator<<	1350
10.409.5.3operator==	1350
10.410.mln::Weighted_Window< E > Struct Template Reference	1351
10.410.1.Detailed Description	1351
10.410.2.Friends And Related Function Documentation	1351
10.410.2.1operator-	1351
10.411.mln::win::backdiag2d Struct Reference	1352
10.411.1.Detailed Description	1352
10.411.2.Constructor & Destructor Documentation	1352
10.411.2.1backdiag2d	1352
10.411.3.Member Function Documentation	1352
10.411.3.1length	1352
10.412.mln::win::ball< G, C > Struct Template Reference	1353
10.412.1.Detailed Description	1353
10.412.2.Constructor & Destructor Documentation	1353
10.412.2.1ball	1353
10.412.3.Member Function Documentation	1353
10.412.3.1diameter	1353
10.413.mln::win::cube3d Struct Reference	1354
10.413.1.Detailed Description	1354
10.413.2.Constructor & Destructor Documentation	1354
10.413.2.1cube3d	1354
10.413.3.Member Function Documentation	1355
10.413.3.1length	1355
10.414.mln::win::cuboid3d Struct Reference	1356
10.414.1.Detailed Description	1356
10.414.2.Constructor & Destructor Documentation	1357
10.414.2.1cuboid3d	1357
10.414.3.Member Function Documentation	1357
10.414.3.1depth	1357
10.414.3.2height	1357

10.414.3.volume	1357
10.414.3.4width	1357
10.415 <code>ln::win::diag2d</code> Struct Reference	1358
10.415. Detailed Description	1358
10.415. Constructor & Destructor Documentation	1358
10.415.2. <code>diag2d</code>	1358
10.415. Member Function Documentation	1358
10.415.3. <code>length</code>	1358
10.416 <code>ln::win::line< M, i, C ></code> Struct Template Reference	1359
10.416. Detailed Description	1359
10.416. Member Enumeration Documentation	1359
10.416.2.1" @86	1359
10.416. Constructor & Destructor Documentation	1360
10.416.3. <code>line</code>	1360
10.416. Member Function Documentation	1360
10.416.4. <code>length</code>	1360
10.416.4.2. <code>size</code>	1360
10.417 <code>ln::win::multiple< W, F ></code> Class Template Reference	1361
10.417. Detailed Description	1361
10.418 <code>ln::win::multiple_size< n, W, F ></code> Class Template Reference	1362
10.418. Detailed Description	1362
10.419 <code>ln::win::octagon2d</code> Struct Reference	1363
10.419. Detailed Description	1363
10.419. Constructor & Destructor Documentation	1363
10.419.2. <code>loctagon2d</code>	1363
10.419. Member Function Documentation	1364
10.419.3. <code>larea</code>	1364
10.419.3.2. <code>length</code>	1364
10.420 <code>ln::win::rectangle2d</code> Struct Reference	1365
10.420. Detailed Description	1365
10.420. Constructor & Destructor Documentation	1365
10.420.2. <code>rectangle2d</code>	1365
10.420. Member Function Documentation	1366
10.420.3. <code>larea</code>	1366
10.420.3.2. <code>height</code>	1366
10.420.3.3. <code>std_vector</code>	1366

10.420.3.4width	1366
10.421mln::Window< E > Struct Template Reference	1367
10.421. Detailed Description	1367
10.422mln::window< D > Class Template Reference	1368
10.422. Detailed Description	1369
10.422.2Member Typedef Documentation	1369
10.422.2.1bkd_qiter	1369
10.422.2.2fwd_qiter	1370
10.422.2.3qiter	1370
10.422.2.4regular	1370
10.422.3Constructor & Destructor Documentation	1370
10.422.3.1window	1370
10.422.4Member Function Documentation	1370
10.422.4.1clear	1370
10.422.4.2delta	1370
10.422.4.3dp	1370
10.422.4.4has	1370
10.422.4.5insert	1371
10.422.4.6insert	1371
10.422.4.7insert	1371
10.422.4.8s_centered	1371
10.422.4.9s_empty	1371
10.422.4.10s_symmetric	1371
10.422.4.11print	1371
10.422.4.12size	1372
10.422.4.13std_vector	1372
10.422.4.14sym	1372
10.422.5Friends And Related Function Documentation	1372
10.422.5.1operator==	1372
10.423mln::world::inter_pixel::is_separator Struct Reference	1373
10.423. Detailed Description	1373
10.424trait::graph< I > Struct Template Reference	1374
10.424. Detailed Description	1374
10.425trait::graph< mln::complex_image< 1, G, V > > Struct Template Reference	1375
10.425. Detailed Description	1375
10.426trait::graph< mln::image2d< T > > Struct Template Reference	1376

10.426. Detailed Description 1376

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 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
-
- 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

2.2 Requirements

2.2.1 To compile the user examples

2.2.2 To compile the documentation (Optional)

2.2.3 To develop in Olena

2.3 Getting Olena

2.4 Building Olena

2.5 Foreword

2.6 Generality

2.7 Directory hierarchy

2.8 Writing and compiling a program with Olena

2.9 Site

2.10 Site set

[Iterators](#)

2.11 Basic interface

2.12 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

```

2.13 Image

2.14 Definition

2.15 Possible image types

2.16 Possible value types

2.17 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

2.18 Border and extension

2.18.1 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);

```

```

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

```

======"

```

- | |
| - -
| | -

```

2.18.2 Generality on image extension

imamorphed

2.18.3 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));

```

2.18.3.1 Extension with a value

```

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

```

2.18.3.2 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));

```

2.18.3.3 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());

```

2.19 Interface

2.20 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");

```

2.21 Create an image

```

// Build an empty image;
image2d<value::int_u8> img1a;

// Build an image with 2 rows
// and 3 columns sites
image2d<value::int_u8> img1b(box2d(2, 3));
image2d<value::int_u8> img1c(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

2.22 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

2.23 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
```

2.24 Structural elements: Window and neighborhood

2.25 Define an element

2.25.1 Window

2.25.2 Neighborhood

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

2.25.3 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);
```

2.25.4 Conversion between Neighborhoods and Windows

2.26 Sites, psites and dpoints

2.27 Need for site

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

2.28 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| | | |
        +---+

```

2.29 From psite to site

2.30 Dpoint

```

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

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

(0,1)

```

2.31 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

2.32 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

```

2.33 Basic routines

2.34 Fill

```

image2d<char> imga(5, 5);

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

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

```

2.35 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;

```

2.36 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);

```

2.37 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);

```

2.38 Compute

2.38.1 Accumulators

2.38.2 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);

```

2.39 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);

```

2.39.1 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);

-

-

-

-

```

2.39.2 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);

```

2.39.3 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, imab1 | 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, imab1 | b2);
debug::println(imad);

ima / sub_D

```

2.40 Input / Output

2.41 ImageMagick

2.42 GDCM

2.43 Graphs and images

2.44 Description

2.45 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.
mIn_vertex_iter_(util::graph) v(g);

// Iterator on v's edges.
mIn_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

2.46 Useful global variables**2.47 Useful macros****2.48 Variable declaration macros****2.49 Iterator type macros****2.49.1 Default iterator types****2.49.2 Forward iterator types****2.49.3 Backward iterators****2.49.4 Graph iterators****2.50 Common Compilation Errors****2.51 Installation****2.52 Requirements****2.52.1 To compile the user examples****2.52.2 To compile the documentation (Optional)****2.52.3 To develop in Olena****2.53 Getting Olena****2.54 Building Olena****2.55 Foreword****2.56 Generality****2.57 Directory hierarchy****2.58 Writing and compiling a program with Olena****2.59 Site****2.60 Site set**

2.61 Basic interface

2.62 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

```

2.63 Image

2.64 Definition

2.65 Possible image types

2.66 Possible value types

2.67 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

```

2.68 Border and extension

2.68.1 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);

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

=====

- | |
| - -
| | -

```

2.68.2 Generality on image extension

imamorphed

2.68.3 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));
```

2.68.3.1 Extension with a value

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

2.68.3.2 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));
```

2.68.3.3 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());

```

2.69 Interface

2.70 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");
```

2.71 Create an image

```
// Build an empty image;
image2d<value::int_u8> img1a;

// Build an image with 2 rows
// and 3 columns sites
image2d<value::int_u8> img1b(box2d(2, 3));
image2d<value::int_u8> img1c(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

2.72 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

2.73 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
```

2.74 Structural elements: Window and neighborhood

2.75 Define an element

2.75.1 Window

2.75.2 Neighborhood

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

2.75.3 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);

```

2.75.4 Conversion between Neighborhoods and Windows

2.76 Sites, psites and dpoints

2.77 Need for site

```

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

```

2.78 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| | | |
      +---+---+---+

```

2.79 From psite to site

2.80 Dpoint

```

dpoint2d dp(-1,0);

```

```

point2d p(1,1);

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

(0,1)

```

2.81 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

2.82 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

```

2.83 Basic routines

2.84 Fill

```
image2d<char> imga(5, 5);

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

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

2.85 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;
```

2.86 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);
```

2.87 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);
```

2.88 Compute

2.88.1 Accumulators

2.88.2 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);
```

2.89 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);
```

2.89.1 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);

-

-

-

-
```

2.89.2 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);

```

2.89.3 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, imab1 | 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, imab1 | b2);
debug::println(imad);

ima / sub_D

```

2.90 Input / Output

2.91 ImageMagick

2.92 GDCM

2.93 Graphs and images

2.94 Description

2.95 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

```

2.96 Useful global variables**2.97 Useful macros****2.98 Variable declaration macros****2.99 Iterator type macros****2.99.1 Default iterator types****2.99.2 Forward iterator types****2.99.3 Backward iterators****2.99.4 Graph iterators****2.100 Common Compilation Errors**

Chapter 3

Tutorial

- tuto1
- tuto2
- tuto3
- tuto4
- tuto5
- tuto6
- tuto7
- tuto8

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

Types	105
Graphes	98
Images	99
Basic types	100
Image morphers	101
Values morphers	102
Domain morphers	103
Identity morphers	104
Neighborhoods	111
1D neighborhoods	112
2D neighborhoods	113
3D neighborhoods	115
Site sets	118
Basic types	119
Graph based	120
Complex based	121
Sparse types	122
Queue based	123
Utilities	124
Windows	125
1D windows	126
2D windows	127
3D windows	130
N-D windows	132
Multiple windows	133
Accumulators	106
On site sets	93
On images	94
On values	95
Multiple accumulators	97
Routines	107
Canvas	108

Functions	109
v2w2v functions	134
v2w_w2v functions	135
vv2b functions	136

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)	137
mln::accu (Namespace of accumulators)	179
mln::accu::image (Namespace of accumulator image routines)	183
mln::accu::impl (Implementation namespace of accumulator namespace)	184
mln::accu::logic (Namespace of logical accumulators)	185
mln::accu::math (Namespace of mathematic accumulators)	186
mln::accu::meta::logic (Namespace of logical meta-accumulators)	187
mln::accu::meta::math (Namespace of mathematic meta-accumulators)	188
mln::accu::meta::shape (Namespace of shape meta-accumulators)	189
mln::accu::meta::stat (Namespace of statistical meta-accumulators)	190
mln::accu::shape (Namespace of shape accumulators)	191
mln::accu::stat (Namespace of statistical accumulators)	192
mln::algebra (Namespace of algebraic structure)	194
mln::arith (Namespace of arithmetic)	196
mln::arith::impl (Implementation namespace of arith namespace)	208
mln::arith::impl::generic (Generic implementation namespace of arith namespace)	209
mln::binarization (Namespace of "point-wise" expression tools)	210
mln::border (Namespace of routines related to image virtual (outer) border)	211
mln::border::impl (Implementation namespace of border namespace)	215
mln::border::impl::generic (Generic implementation namespace of border namespace)	216
mln::canvas (Namespace of canvas)	217
mln::canvas::browsing (Namespace of browsing canvas)	219
mln::canvas::impl (Implementation namespace of canvas namespace)	220
mln::canvas::labeling (Namespace of labeling canvas)	221
mln::canvas::labeling::impl (Implementation namespace of labeling canvas namespace)	222
mln::canvas::morpho (Namespace of morphological canvas)	223
mln::convert (Namespace of conversion routines)	224
mln::data (Namespace of image processing routines related to pixel data)	230
mln::data::approx (Namespace of image processing routines related to pixel levels with approxi- mation)	243
mln::data::approx::impl (Implementation namespace of data::approx namespace)	245
mln::data::impl (Implementation namespace of data namespace)	246
mln::data::impl::generic (Generic implementation namespace of data namespace)	248

<code>mln::data::naive</code> (Namespace of image processing routines related to <code>pixel</code> levels with <code>naive</code> approach)	253
<code>mln::data::naive::impl</code> (Implementation namespace of <code>data::naive</code> namespace)	254
<code>mln::debug</code> (Namespace of routines that help to <code>debug</code>)	255
<code>mln::debug::impl</code> (Implementation namespace of <code>debug</code> namespace)	260
<code>mln::def</code> (Namespace for core definitions)	261
<code>mln::display</code> (Namespace of routines that help to <code>display</code> images)	262
<code>mln::display::impl</code> (Implementation namespace of <code>display</code> namespace)	263
<code>mln::display::impl::generic</code> (Generic implementation namespace of <code>display</code> namespace)	264
<code>mln::doc</code> (The namespace <code>mln::doc</code> is only for documentation purpose)	265
<code>mln::draw</code> (Namespace of drawing routines)	267
<code>mln::estim</code> (Namespace of estimation materials)	269
<code>mln::extension</code> (Namespace of <code>extension</code> tools)	271
<code>mln::fun</code> (Namespace of functions)	274
<code>mln::fun::access</code> (Namespace for <code>access</code> functions)	276
<code>mln::fun::i2v</code> (Namespace of integer-to-value functions)	277
<code>mln::fun::p2b</code> (Namespace of functions from <code>point</code> to boolean)	278
<code>mln::fun::p2p</code> (Namespace of functions from <code>grid point</code> to <code>grid point</code>)	279
<code>mln::fun::p2v</code> (Namespace of functions from <code>point</code> to <code>value</code>)	280
<code>mln::fun::stat</code> (Namespace of statistical functions)	281
<code>mln::fun::v2b</code> (Namespace of functions from <code>value</code> to logic <code>value</code>)	282
<code>mln::fun::v2i</code> (Namespace of value-to-integer functions)	283
<code>mln::fun::v2v</code> (Namespace of functions from <code>value</code> to <code>value</code>)	284
<code>mln::fun::v2w2v</code> (Namespace of bijective functions)	286
<code>mln::fun::v2w_w2v</code> (Namespace of functions from <code>value</code> to <code>value</code>)	287
<code>mln::fun::vv2v</code> (Namespace of functions from <code>value</code> to <code>value</code>)	288
<code>mln::fun::vv2v</code> (Namespace of functions from a couple of values to a <code>value</code>)	289
<code>mln::fun::x2p</code> (Namespace of functions from <code>point</code> to <code>value</code>)	290
<code>mln::fun::x2v</code> (Namespace of functions from vector to <code>value</code>)	291
<code>mln::fun::x2x</code> (Namespace of functions from vector to vector)	292
<code>mln::geom</code> (Namespace of all things related to geometry)	293
<code>mln::geom::impl</code> (Implementation namespace of <code>geom</code> namespace)	305
<code>mln::graph</code> (Namespace of <code>graph</code> related routines)	307
<code>mln::grid</code> (Namespace of grids definitions)	310
<code>mln::histo</code> (Namespace of histograms)	311
<code>mln::histo::impl</code> (Implementation namespace of <code>histo</code> namespace)	312
<code>mln::histo::impl::generic</code> (Generic implementation namespace of <code>histo</code> namespace)	313
<code>mln::impl</code> (Implementation namespace of <code>mln</code> namespace)	314
<code>mln::io</code> (Namespace of input/output handling)	315
<code>mln::io::cloud</code> (Namespace of <code>cloud</code> input/output handling)	317
<code>mln::io::dicom</code> (Namespace of DICOM input/output handling)	318
<code>mln::io::dump</code> (Namespace of <code>dump</code> input/output handling)	319
<code>mln::io::fits</code> (Namespace of <code>fits</code> input/output handling)	320
<code>mln::io::fld</code> (Namespace of <code>pgm</code> input/output handling)	321
<code>mln::io::magick</code> (Namespace of <code>magick</code> input/output handling)	323
<code>mln::io::off</code> (Namespace of <code>off</code> input/output handling)	325
<code>mln::io::pbm</code> (Namespace of <code>pbm</code> input/output handling)	327
<code>mln::io::pbm::impl</code> (Namespace of <code>pbm</code> implementation details)	329
<code>mln::io::pbms</code> (Namespace of <code>pbms</code> input/output handling)	330
<code>mln::io::pbms::impl</code> (Namespace of <code>pbms</code> implementation details)	331
<code>mln::io::pfm</code> (Namespace of <code>pfm</code> input/output handling)	332
<code>mln::io::pfm::impl</code> (Implementation namespace of <code>pfm</code> namespace)	334
<code>mln::io::pgm</code> (Namespace of <code>pgm</code> input/output handling)	335
<code>mln::io::pgms</code> (Namespace of <code>pgms</code> input/output handling)	337

<code>mln::io::plot</code> (Namespace of <code>plot</code> input/output handling)	338
<code>mln::io::pnm</code> (Namespace of <code>pnm</code> input/output handling)	340
<code>mln::io::pnm::impl</code> (Namespace of <code>pnm</code> 's implementation details)	342
<code>mln::io::pnms</code> (Namespace of <code>pnms</code> input/output handling)	343
<code>mln::io::ppm</code> (Namespace of <code>ppm</code> input/output handling)	344
<code>mln::io::ppms</code> (Namespace of <code>ppms</code> input/output handling)	346
<code>mln::io::tiff</code> (Namespace of <code>tiff</code> input/output handling)	347
<code>mln::io::txt</code> (Namespace of <code>txt</code> input/output handling)	348
<code>mln::labeling</code> (Namespace of <code>labeling</code> routines)	349
<code>mln::labeling::impl</code> (Implementation namespace of <code>labeling</code> namespace)	363
<code>mln::labeling::impl::generic</code> (Generic implementation namespace of <code>labeling</code> namespace)	364
<code>mln::linear</code> (Namespace of <code>linear</code> image processing routines)	366
<code>mln::linear::impl</code> (Namespace of <code>linear</code> image processing routines implementation details)	370
<code>mln::linear::local</code> (Specializations of <code>local linear</code> routines)	371
<code>mln::linear::local::impl</code> (Namespace of <code>local linear</code> routines implementation details)	372
<code>mln::literal</code> (Namespace of literals)	373
<code>mln::logical</code> (Namespace of logic)	379
<code>mln::logical::impl</code> (Implementation namespace of <code>logical</code> namespace)	382
<code>mln::logical::impl::generic</code> (Generic implementation namespace of <code>logical</code> namespace)	383
<code>mln::make</code> (Namespace of routines that help to <code>make</code> Milena's objects)	384
<code>mln::math</code> (Namespace of mathematical routines)	408
<code>mln::metal</code> (Namespace of meta-programming tools)	409
<code>mln::metal::impl</code> (Implementation namespace of <code>metal</code> namespace)	410
<code>mln::metal::math</code> (Namespace of static mathematical functions)	411
<code>mln::metal::math::impl</code> (Implementation namespace of <code>metal::math</code> namespace)	412
<code>mln::morpho</code> (Namespace of mathematical morphology routines)	413
<code>mln::morpho::approx</code> (Namespace of approximate mathematical morphology routines)	422
<code>mln::morpho::attribute</code> (Namespace of attributes used in mathematical morphology)	423
<code>mln::morpho::closing::approx</code> (Namespace of approximate mathematical morphology closing routines)	424
<code>mln::morpho::elementary</code> (Namespace of image processing routines of <code>elementary</code> mathematical morphology)	425
<code>mln::morpho::impl</code> (Namespace of mathematical morphology routines implementations)	427
<code>mln::morpho::impl::generic</code> (Namespace of mathematical morphology routines <code>generic</code> implementations)	428
<code>mln::morpho::opening::approx</code> (Namespace of approximate mathematical morphology opening routines)	429
<code>mln::morpho::reconstruction</code> (Namespace of morphological <code>reconstruction</code> routines)	430
<code>mln::morpho::reconstruction::by_dilation</code> (Namespace of morphological <code>reconstruction</code> by dilation routines)	431
<code>mln::morpho::reconstruction::by_erosion</code> (Namespace of morphological <code>reconstruction</code> by erosion routines)	432
<code>mln::morpho::tree</code> (Namespace of morphological tree-related routines)	433
<code>mln::morpho::tree::filter</code> (Namespace for <code>attribute</code> filtering)	440
<code>mln::morpho::watershed</code> (Namespace of morphological <code>watershed</code> routines)	443
<code>mln::morpho::watershed::watershed</code> (Namespace of morphological <code>watershed</code> routines implementations)	446
<code>mln::morpho::watershed::watershed::generic</code> (Namespace of morphological <code>watershed</code> routines <code>generic</code> implementations)	447
<code>mln::norm</code> (Namespace of norms)	448
<code>mln::norm::impl</code> (Implementation namespace of <code>norm</code> namespace)	450
<code>mln::opt</code> (Namespace of optional routines)	451
<code>mln::opt::impl</code> (Implementation namespace of <code>opt</code> namespace)	453
<code>mln::pw</code> (Namespace of "point-wise" expression tools)	454

mln::registration (Namespace of "point-wise" expression tools)	455
mln::select (Select namespace (FIXME doc))	458
mln::set (Namespace of image processing routines related to pixel sets)	459
mln::subsampling (Namespace of "point-wise" expression tools)	462
mln::tag (Namespace of image processing routines related to tags)	463
mln::test (Namespace of image processing routines related to pixel tests)	464
mln::test::impl (Implementation namespace of test namespace)	466
mln::topo (Namespace of "point-wise" expression tools)	467
mln::trace (Namespace of routines related to the trace mechanism)	477
mln::trait (Namespace where traits are defined)	478
mln::transform (Namespace of transforms)	479
mln::util (Namespace of tools using for more complex algorithm)	484
mln::util::impl (Implementation namespace of util namespace)	491
mln::value (Namespace of materials related to pixel value types)	492
mln::value::impl (Implementation namespace of value namespace)	503
mln::win (Namespace of image processing routines related to win)	504

Chapter 6

Class Index

6.1 Class Hierarchy

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

mln::Generalized_Pixel< mln::bkd_pixter1d< I > >	831
mln::Pixel_Iterator< mln::bkd_pixter1d< I > >	1108
mln::Generalized_Pixel< mln::bkd_pixter2d< I > >	831
mln::Pixel_Iterator< mln::bkd_pixter2d< I > >	1108
mln::Generalized_Pixel< mln::bkd_pixter3d< I > >	831
mln::Pixel_Iterator< mln::bkd_pixter3d< I > >	1108
mln::Generalized_Pixel< mln::dpoints_bkd_pixter< I > >	831
mln::Pixel_Iterator< mln::dpoints_bkd_pixter< I > >	1108
mln::Generalized_Pixel< mln::dpoints_fwd_pixter< I > >	831
mln::Pixel_Iterator< mln::dpoints_fwd_pixter< I > >	1108
mln::Generalized_Pixel< mln::fwd_pixter1d< I > >	831
mln::Pixel_Iterator< mln::fwd_pixter1d< I > >	1108
mln::Generalized_Pixel< mln::fwd_pixter2d< I > >	831
mln::Pixel_Iterator< mln::fwd_pixter2d< I > >	1108
mln::Generalized_Pixel< mln::fwd_pixter3d< I > >	831
mln::Pixel_Iterator< mln::fwd_pixter3d< I > >	1108
mln::Generalized_Pixel< mln::pixel< I > >	831
mln::internal::image_base< F::result, S, E >	
image_primary< F::result, S, E >	
mln::pw::internal::image_base	
mln::edge_image< P, V, G >	755
mln::pw::image< F, S >	1133
mln::vertex_image< P, V, G >	1342
mln::internal::image_base< I::value, S, E >	
image_morpher< I, I::value, S, E >	
mln::internal::image_domain_morpher	
mln::hexa< mln::image2d< V > >	868
mln::extended< I >	758
mln::hexa< I >	868
mln::image2d_h< V >	885
mln::image_if< I, F >	893

mln::p2p_image< I, F >	969
mln::slice_image< I >	1150
mln::sub_image< I, S >	1152
mln::sub_image_if< I, S >	1154
mln::transformed_image< I, F >	1221
mln::unproject_image< I, D, F >	1223
mln::internal::image_identity	
mln::labeled_image_base< I, mln::labeled_image< I > >	904
mln::decorated_image< I, D >	687
mln::extension_fun< I, F >	760
mln::extension_ima< I, J >	763
mln::extension_val< I >	766
mln::interpolated< I, F >	895
mln::labeled_image_base< I, E >	904
mln::labeled_image< I >	900
mln::lazy_image< I, F, B >	907
mln::plain< I >	1110
mln::safe_image< I >	1137
mln::tr_image< S, I, T >	1218
mln::internal::image_base< T, I::domain_t, E >	
image_morpher< I, T, I::domain_t, E >	
mln::internal::image_value_morpher	
mln::fun_image< F, I >	815
mln::thrubin_image< I1, I2, F >	1157
mln::value::stack_image< n, I >	1332
mln::violent_cast_image< T, I >	1345
mln::value::Integer< mln::util::object_id< Tag, V > >	1316
mln::value::Integer< mln::value::graylevel< n > >	1316
mln::value::Integer< mln::value::int_s< n > >	1316
mln::value::Integer< mln::value::int_u< n > >	1316
mln::value::Integer< mln::value::int_u_sat< n > >	1316
mln::algebra::h_mat< d, T >	629
mln::algebra::h_vec< d, C >	631
mln::canvas::chamfer< F >	671
mln::category< R(*)>(A) >	672
mln::Delta_Point_Site< void >	691
mln::doc::Accumulator< E >	692
mln::doc::Generalized_Pixel< E >	707
mln::doc::Pixel_Iterator< E >	720
mln::doc::Object< E >	719
mln::doc::Dpoint< E >	697
mln::doc::Image< E >	709
mln::doc::Fastest_Image< E >	699
mln::doc::Iterator< E >	715
mln::doc::Pixel_Iterator< E >	720
mln::doc::Site_Iterator< E >	726
mln::doc::Value_Iterator< E >	730
mln::doc::Neighborhood< E >	717
mln::doc::Site_Set< E >	728
mln::doc::Box< E >	694
mln::doc::Value_Set< E >	732
mln::doc::Weighted_Window< E >	734
mln::doc::Window< E >	737

mln::doc::Point_Site< E >	723
mln::Edge< E >	754
mln::fun::from_accu< A >	775
mln::fun::internal::ch_function_value_impl< F, V >	
mln::fun::v2v::ch_function_value< F, V >	780
mln::fun::x2p::closest_point< P >	804
mln::fun::x2x::composed< T2, T1 >	807
mln::Function< void >	818
mln::Gdpoint< void >	830
mln::Generalized_Pixel< E >	831
mln::pixel< I >	1106
mln::Pixel_Iterator< E >	1108
mln::dpoints_bkd_pixter< I >	744
mln::dpoints_fwd_pixter< I >	747
mln::internal::pixel_iterator_base_	
mln::internal::backward_pixel_iterator_base_	
mln::bkd_pixter1d< I >	633
mln::bkd_pixter2d< I >	635
mln::bkd_pixter3d< I >	637
mln::internal::forward_pixel_iterator_base_	
mln::fwd_pixter1d< I >	823
mln::fwd_pixter2d< I >	825
mln::fwd_pixter3d< I >	827
mln::geom::complex_geometry< D, P >	832
mln::graph::attribute::card_t	839
mln::graph::attribute::representative_t	840
mln::histo::array< T >	871
mln::internal::check::image_fastest< E, B >	
mln::internal::image_base< T, S, E >	
mln::internal::image_primary	
mln::complex_image< D, G, V >	673
mln::flat_image< T, S >	772
mln::image1d< T >	875
mln::image2d< T >	880
mln::image3d< T >	888
mln::internal::impl_selector< C, P, E >	
mln::graph_window_piter< S, W, I >	864
mln::internal::is_masked_impl_selector< S, D, E >	
mln::graph_window_if_piter< S, W, I >	862
mln::internal::neighborhood_base< W, E >	
mln::internal::neighb_base	
mln::mixed_neighb< W >	949
mln::neighb< W >	964
mln::graph_elt_mixed_neighborhood< G, S, S2 >	841
mln::graph_elt_neighborhood< G, S >	847
mln::graph_elt_neighborhood_if< G, S, I >	849
mln::neighb< mln::graph_elt_mixed_window< G, S, S2 > >	964
mln::neighb< mln::graph_elt_window< G, S > >	964
mln::neighb< mln::graph_elt_window_if< G, S, I > >	964
mln::internal::pixel_impl< I, E >	
mln::dpoints_bkd_pixter< I >	744
mln::dpoints_fwd_pixter< I >	747
mln::internal::pixel_iterator_base_	

mln::pixel< I >	1106
mln::io::fld::fld_header	897
mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 >	941
mln::metal::bool_< false >	
mln::metal::equal< T1::coord, T2::coord >	943
mln::metal::equal< T1::point, T2::point >	943
mln::metal::equal< T1, T2 >	943
mln::metal::converts_to< T, U >	942
mln::metal::goes_to< T, U >	944
mln::metal::is< T, U >	945
mln::metal::is_a< T, M >	946
mln::metal::is_not< T, U >	947
mln::metal::is_not_a< T, M >	948
mln::Neighborhood< void >	967
mln::Object< E >	968
mln::Function< function< meta::blue< mln::value::mln::value::rgb::mln::value::mln::value::rgb< n > > > >	817
mln::Function< function< meta::green< mln::value::mln::value::rgb::mln::value::mln::value::rgb< n > > > >	817
mln::Function< function< meta::red< mln::value::mln::value::rgb::mln::value::mln::value::rgb< n > > > >	817
mln::Meta_Function< composition< mln::mln::mln::mln::Meta_Function_v2v, F, mln::mln::mln::mln::Meta_Function_v2v, G > >	938
mln::Meta_Function< composition< mln::mln::mln::mln::Meta_Function_v2v, F, mln::mln::mln::mln::Meta_Function_vv2v, G > >	938
mln::Browsing< E >	657
mln::canvas::browsing::backdiagonal2d_t	658
mln::canvas::browsing::diagonal2d_t	661
mln::canvas::browsing::dir_struct_elt_incr_update_t	662
mln::canvas::browsing::directional_t	664
mln::canvas::browsing::fwd_t	666
mln::canvas::browsing::hyper_directional_t	667
mln::canvas::browsing::internal::graph_first_search_t mln::canvas::browsing::breadth_first_search_t	659
mln::canvas::browsing::depth_first_search_t	660
mln::canvas::browsing::snake_fwd_t	668
mln::canvas::browsing::snake_generic_t	669
mln::canvas::browsing::snake_vert_t	670
mln::Delta_Point_Site< E >	690
mln::Dpoint< E >	738
mln::Function< E >	817
mln::Function_v2v< function< meta::blue< mln::value::rgb::mln::value::rgb< n > > > >	820
mln::Function_v2v< function< meta::green< mln::value::rgb::mln::value::rgb< n > > > >	820
mln::Function_v2v< function< meta::red< mln::value::rgb::mln::value::rgb< n > > > >	820
mln::Function_v2v< E >	820
mln::fun::v2v::ch_function_value< F, V >	780
mln::fun::v2v::component< T, i >	781
mln::fun::v2v::l1_norm< V, R >	782
mln::fun::v2v::l2_norm< V, R >	783
mln::fun::v2v::linear< V, T, R >	784
mln::fun::v2v::linfty_norm< V, R >	785
mln::fun::v2w2v::cos< V >	786

mln::fun::v2w_w2v::l1_norm< V, R >	787
mln::fun::v2w_w2v::l2_norm< V, R >	788
mln::fun::v2w_w2v::linfty_norm< V, R >	789
mln::fun::x2v::bilinear< I >	805
mln::fun::x2v::trilinear< I >	806
mln::fun::x2x::linear< I >	808
mln::fun::x2x::rotation< n, C >	810
mln::fun::x2x::translation< n, C >	813
mln::Function_v2b< E >	819
mln::fun::p2b::antilogy	776
mln::fun::p2b::tautology	777
mln::fun::v2b::lnot< V >	778
mln::fun::v2b::threshold< V >	779
mln::topo::is_n_face< N >	1201
mln::topo::is_simple_cell< I >	1202
mln::world::inter_pixel::is_separator	1373
mln::Function_vv2b< E >	821
mln::fun::vv2b::eq< L, R >	790
mln::fun::vv2b::ge< L, R >	791
mln::fun::vv2b::gt< L, R >	792
mln::fun::vv2b::implies< L, R >	793
mln::fun::vv2b::le< L, R >	794
mln::fun::vv2b::lt< L, R >	795
mln::Function_vv2v< E >	822
mln::fun::vv2v::diff_abs< V >	796
mln::fun::vv2v::land< L, R >	797
mln::fun::vv2v::land_not< L, R >	798
mln::fun::vv2v::lor< L, R >	799
mln::fun::vv2v::lxor< L, R >	800
mln::fun::vv2v::max< V >	801
mln::fun::vv2v::min< L, R >	802
mln::fun::vv2v::vec< V >	803
mln::Gdpoint< E >	829
mln::dpoint< G, C >	739
mln::Graph< E >	838
mln::util::internal::graph_base	
mln::util::graph	1248
mln::util::line_graph< G >	1260
mln::Image< E >	872
mln::Iterator< E >	898
mln::Pixel_Iterator< E >	1108
mln::topo::internal::complex_iterator_base	
mln::topo::internal::complex_relative_iterator_base	
mln::topo::internal::backward_complex_relative_iterator_base	
mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D >	1159
mln::topo::adj_higher_face_bkd_iter< D >	1163
mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D >	1165
mln::topo::adj_lower_face_bkd_iter< D >	1169
mln::topo::adj_m_face_bkd_iter< D >	1173
mln::topo::internal::forward_complex_relative_iterator_base	
mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D >	1161
mln::topo::adj_higher_face_fwd_iter< D >	1164
mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D >	1167

mln::topo::adj_lower_face_fwd_iter< D >	1170
mln::topo::adj_m_face_fwd_iter< D >	1175
mln::topo::center_only_iter< D >	1186
mln::topo::internal::complex_set_iterator_base	
mln::topo::face_bkd_iter< D >	1197
mln::topo::face_fwd_iter< D >	1199
mln::topo::n_face_bkd_iter< D >	1208
mln::topo::n_face_fwd_iter< D >	1210
mln::topo::static_n_face_bkd_iter< N, D >	1214
mln::topo::static_n_face_fwd_iter< N, D >	1216
mln::topo::internal::complex_relative_iterator_sequence	
mln::topo::adj_lower_higher_face_bkd_iter< D >	1171
mln::topo::adj_lower_higher_face_fwd_iter< D >	1172
mln::topo::centered_bkd_iter_adapter< D, I >	1188
mln::topo::centered_fwd_iter_adapter< D, I >	1189
mln::Value_Iterator< E >	1338
mln::Literal< E >	910
mln::literal::black_t	913
mln::literal::blue_t	914
mln::literal::brown_t	915
mln::literal::cyan_t	916
mln::literal::green_t	917
mln::literal::identity_t	918
mln::literal::light_gray_t	919
mln::literal::lime_t	920
mln::literal::magenta_t	921
mln::literal::max_t	922
mln::literal::min_t	923
mln::literal::olive_t	924
mln::literal::one_t	925
mln::literal::orange_t	926
mln::literal::origin_t	927
mln::literal::pink_t	928
mln::literal::purple_t	929
mln::literal::red_t	930
mln::literal::teal_t	931
mln::literal::violet_t	932
mln::literal::white_t	933
mln::literal::yellow_t	934
mln::literal::zero_t	935
mln::Mesh< E >	936
mln::Regular_Grid< E >	1136
mln::Meta_Accumulator< E >	937
mln::accu::meta::center	541
mln::accu::meta::count_adjacent_vertices	542
mln::accu::meta::count_labels	543
mln::accu::meta::count_value	544
mln::accu::meta::histo	545
mln::accu::meta::label_used	546
mln::accu::meta::logic::land	547
mln::accu::meta::logic::land_basic	548
mln::accu::meta::logic::lor	549
mln::accu::meta::logic::lor_basic	550

mln::accu::meta::maj_h	551
mln::accu::meta::math::count	552
mln::accu::meta::math::inf	553
mln::accu::meta::math::sum	554
mln::accu::meta::math::sup	555
mln::accu::meta::max_site	556
mln::accu::meta::nil	557
mln::accu::meta::p< mA >	558
mln::accu::meta::pair< A1, A2 >	559
mln::accu::meta::rms	560
mln::accu::meta::shape::bbox	561
mln::accu::meta::shape::height	562
mln::accu::meta::shape::volume	563
mln::accu::meta::stat::max	564
mln::accu::meta::stat::max_h	565
mln::accu::meta::stat::mean	566
mln::accu::meta::stat::median_alt< T >	567
mln::accu::meta::stat::median_h	568
mln::accu::meta::stat::min	569
mln::accu::meta::stat::min_h	570
mln::accu::meta::stat::rank	571
mln::accu::meta::stat::rank_high_quant	572
mln::accu::meta::tuple< n, >	573
mln::accu::meta::val< mA >	574
mln::accu::stat::meta::deviation	605
mln::Meta_Function< E >	938
mln::Meta_Function_v2v< composition< mln::mln::mln::Meta_Function_v2v, F, mln::mln::mln::Meta_Function_v2v, G > >	939
mln::Meta_Function_vv2v< composition< mln::mln::mln::Meta_Function_v2v, F, mln::mln::mln::Meta_Function_vv2v, G > >	940
mln::Meta_Function_v2v< E >	939
mln::Meta_Function_vv2v< E >	940
mln::Neighborhood< E >	966
mln::pixel< I >	1106
mln::Point_Site< E >	1124
mln::Proxy< E >	1129
mln::Accumulator< E >	628
mln::accu::internal::base	
mln::accu::stat::median_alt< mln::value::set< T > >	601
mln::accu::center< P, V >	507
mln::accu::convolve< T1, T2, R >	509
mln::accu::count_adjacent_vertices< F, S >	511
mln::accu::count_labels< L >	513
mln::accu::count_value< V >	515
mln::accu::histo< V >	517
mln::accu::internal::couple	
mln::accu::site_set::rectangularity< P >	591
mln::accu::label_used< L >	519
mln::accu::logic::land	521
mln::accu::logic::land_basic	523
mln::accu::logic::lor	525
mln::accu::logic::lor_basic	527
mln::accu::maj_h< T >	529

mln::accu::math::count< T >	531
mln::accu::math::inf< T >	533
mln::accu::math::sum< T, S >	535
mln::accu::math::sup< T >	537
mln::accu::max_site< I >	539
mln::accu::nil< T >	575
mln::accu::p< A >	577
mln::accu::pair< A1, A2, T >	579
mln::accu::stat::min_max< V >	610
mln::accu::rms< T, V >	581
mln::accu::shape::bbox< P >	583
mln::accu::shape::height< I >	585
mln::accu::shape::volume< I >	588
mln::accu::stat::deviation< T, S, M >	593
mln::accu::stat::max< T >	595
mln::accu::stat::max_h< V >	597
mln::accu::stat::mean< T, S, M >	599
mln::accu::stat::median_alt< S >	601
mln::accu::stat::median_h< V >	603
mln::accu::stat::min< T >	606
mln::accu::stat::min_h< V >	608
mln::accu::stat::rank< T >	612
mln::accu::stat::rank< bool >	614
mln::accu::stat::rank_high_quant< T >	616
mln::accu::stat::var< T >	618
mln::accu::stat::variance< T, S, R >	621
mln::accu::tuple< A, n, >	624
mln::accu::val< A >	626
mln::morpho::attribute::card< I >	951
mln::morpho::attribute::count_adjacent_vertices< I >	953
mln::morpho::attribute::height< I >	955
mln::morpho::attribute::sharpness< I >	957
mln::morpho::attribute::sum< I, S >	960
mln::morpho::attribute::volume< I >	962
mln::accu::pair< mln::accu::stat::min< V >, mln::accu::stat::max< V > >	579
mln::Site_Proxy< E >	1143
mln::Pseudo_Site< E >	1131
mln::internal::pseudo_site_base_	
mln::complex_psite< D, G >	680
mln::faces_psite< N, D, P >	769
mln::p_indexed_psite< S >	1020
mln::Site_Iterator< E >	1141
mln::internal::site_iterator_base	
mln::internal::site_relative_iterator_base	
mln::complex_neighborhood_bkd_piter< I, G, N >	676
mln::complex_neighborhood_fwd_piter< I, G, N >	678
mln::complex_window_bkd_piter< I, G, W >	683
mln::complex_window_fwd_piter< I, G, W >	685
mln::dpsites_bkd_piter< V >	750
mln::dpsites_fwd_piter< V >	752
mln::graph_window_if_piter< S, W, I >	862
mln::graph_window_piter< S, W, I >	864
mln::internal::site_set_iterator_base	

mln::box_runend_piter< P >	653
mln::box_runstart_piter< P >	655
mln::internal::p_complex_piter_base_	
mln::p_n_faces_bkd_piter< D, P >	1040
mln::p_n_faces_fwd_piter< D, P >	1042
mln::p_graph_piter< S, I >	1003
mln::p_indexed_bkd_piter< S >	1016
mln::p_indexed_fwd_piter< S >	1018
mln::p_transformed_piter< Pi, S, F >	1090
mln::util::timer	1284
mln::value::proxy< I >	1324
mln::Site< E >	1139
mln::Gpoint< E >	834
mln::point< G, C >	1115
mln::util::vertex< G >	1293
mln::Site_Set< E >	1145
mln::Box< E >	648
mln::box< P >	639
mln::internal::site_set_base_	
mln::p_array< P >	971
mln::p_centered< W >	978
mln::p_complex< D, G >	983
mln::p_edges< G, F >	989
mln::p_faces< N, D, P >	997
mln::p_if< S, F >	1005
mln::p_image< I >	1010
mln::p_key< K, P >	1021
mln::p_line2d	1028
mln::p_mutable_array_of< S >	1034
mln::p_priority< P, Q >	1044
mln::p_queue< P >	1052
mln::p_queue_fast< P >	1059
mln::p_run< P >	1066
mln::p_set< P >	1073
mln::p_set_of< S >	1080
mln::p_transformed< S, F >	1085
mln::p_vaccess< V, S >	1092
mln::p_vertices< G, F >	1098
mln::util::couple< T, U >	1238
mln::util::eat	1240
mln::util::fibonacci_heap< P, T >	1245
mln::util::ignore	1258
mln::util::nil	1266
mln::util::ord_pair< T >	1270
mln::util::site_pair< P >	1280
mln::util::soft_heap< T, R >	1281
mln::util::yes	1297
mln::Value< E >	1298
mln::Value_Set< E >	1340
mln::value::lut_vec< S, T >	1321
mln::Weighted_Window< E >	1351
mln::internal::weighted_window_base	
mln::w_window< D, W >	1347

mln::Window< E >	1367
mln::graph_window_base< P, E >	860
mln::graph_elt_mixed_window< G, S, S2 >	843
mln::graph_elt_window< G, S >	851
mln::graph_elt_window_if< G, S, I >	855
mln::internal::window_base	
mln::internal::classical_window_base	
mln::win::backdiag2d	1352
mln::win::ball< G, C >	1353
mln::win::cube3d	1354
mln::win::cuboid3d	1356
mln::win::diag2d	1358
mln::win::line< M, i, C >	1359
mln::win::octagon2d	1363
mln::win::rectangle2d	1365
mln::win::multiple< W, F >	1361
mln::win::multiple_size< n, W, F >	1362
mln::window< D >	1368
mln::Point_Site< void >	1128
mln::Proxy< void >	1130
mln::Pseudo_Site< void >	1132
mln::registration::closest_point_basic< P >	1134
mln::registration::closest_point_with_map< P >	1135
mln::select::p_of< P >	1138
mln::Site< void >	1140
mln::Site_Proxy< void >	1144
mln::Site_Set< void >	1149
mln::thru_image< I, F >	1156
mln::topo::complex< D >	1190
mln::topo::face< D >	1193
mln::topo::algebraic_face< D >	1177
mln::topo::n_face< N, D >	1204
mln::topo::algebraic_n_face< N, D >	1182
mln::topo::n_faces_set< N, D >	1212
mln::util::adjacency_matrix< V >	1225
mln::util::array< T >	1226
mln::util::branch< T >	1232
mln::util::branch_iter< T >	1234
mln::util::branch_iter_ind< T >	1236
mln::util::greater_point< I >	1255
mln::util::greater_psite< I >	1256
mln::util::head< T, R >	1257
mln::util::ilcell< T >	1259
mln::util::internal::edge_impl_< G >	
mln::util::edge< G >	1241
mln::util::internal::vertex_impl_< G >	
mln::util::vertex< G >	1293
mln::util::node< T, R >	1267
mln::util::ord< T >	1269
mln::util::pix< I >	1272
mln::util::tracked_ptr< T >	1285
mln::util::tree< T >	1287
mln::util::tree_node< T >	1289

mln::value::float01	1299
mln::value::Integer< E >	1316
mln::util::object_id< Tag, V >	1268
mln::value::graylevel< n >	1304
mln::value::int_s< n >	1310
mln::value::int_u< n >	1312
mln::value::int_u_sat< n >	1314
mln::value::Integer< void >	1317
mln::value::internal::value_like_< V, C, N, E >	
mln::value::float01_f	1302
mln::value::graylevel< n >	1304
mln::value::graylevel_f	1307
mln::value::int_s< n >	1310
mln::value::int_u< n >	1312
mln::value::int_u_sat< n >	1314
mln::value::label< n >	1318
mln::value::rgb< n >	1327
mln::value::set< T >	1329
mln::value::sign	1330
mln::value::super_value< sign >	1335
mln::value::value_array< T, V >	1336
mln::Vertex< E >	1341
mln::Object< colorize >	968
mln::Function< colorize >	817
mln::Function_v2v< colorize >	820
mln::Object< composition< mln::mln::mln::mln::Meta_Function_v2v, F, mln::mln::mln::mln::Meta_Function_v2v, G >>	968
mln::Object< composition< mln::mln::mln::mln::Meta_Function_v2v, F, mln::mln::mln::mln::Meta_Function_vv2v, G >>	968
mln::Object< d_t >	968
mln::Function< d_t >	817
mln::Function_vv2v< d_t >	822
mln::Object< dist_t >	968
mln::Function< dist_t >	817
mln::Function_vv2v< dist_t >	822
mln::Object< f_16_to_8 >	968
mln::Function< f_16_to_8 >	817
mln::Function_v2v< f_16_to_8 >	820
mln::Object< f_box1d_t >	968
mln::Function< f_box1d_t >	817
mln::Function_v2v< f_box1d_t >	820
mln::Function_v2b< f_box1d_t >	819
mln::Object< f_box2d_t >	968
mln::Function< f_box2d_t >	817
mln::Function_v2v< f_box2d_t >	820
mln::Function_v2b< f_box2d_t >	819
mln::Object< f_box3d_t >	968
mln::Function< f_box3d_t >	817
mln::Function_v2v< f_box3d_t >	820
mln::Function_v2b< f_box3d_t >	819

mln::Object< function< meta::blue< mln::value::mln::value::mln::value::rgb::mln::value::mln::value::mln::value::rgb< n > > > >	968
mln::Object< function< meta::first< util::couple< T, U > > > >	968
mln::Function< function< meta::first< util::couple< T, U > > > >	817
mln::Function_v2v< function< meta::first< util::couple< T, U > > > >	820
mln::Object< function< meta::green< mln::value::mln::value::mln::value::rgb::mln::value::mln::value::mln::value::rgb< n > > > >	968
mln::Object< function< meta::red< mln::value::mln::value::mln::value::rgb::mln::value::mln::value::mln::value::rgb< n > > > >	968
mln::Object< function< meta::second< util::couple< T, U > > > >	968
mln::Function< function< meta::second< util::couple< T, U > > > >	817
mln::Function_v2v< function< meta::second< util::couple< T, U > > > >	820
mln::Object< function< meta::to_enc< T > > >	968
mln::Function< function< meta::to_enc< T > > >	817
mln::Function_v2v< function< meta::to_enc< T > > >	820
mln::Object< keep_specific_colors >	968
mln::Function< keep_specific_colors >	817
mln::Function_v2v< keep_specific_colors >	820
mln::Function_v2b< keep_specific_colors >	819
mln::Object< mln::accu::center< P, V > >	968
mln::Proxy< mln::accu::center< P, V > >	1129
mln::Accumulator< mln::accu::center< P, V > >	628
mln::Object< mln::accu::convolve< T1, T2, R > >	968
mln::Proxy< mln::accu::convolve< T1, T2, R > >	1129
mln::Accumulator< mln::accu::convolve< T1, T2, R > >	628
mln::Object< mln::accu::count_adjacent_vertices< F, S > >	968
mln::Proxy< mln::accu::count_adjacent_vertices< F, S > >	1129
mln::Accumulator< mln::accu::count_adjacent_vertices< F, S > >	628
mln::Object< mln::accu::count_labels< L > >	968
mln::Proxy< mln::accu::count_labels< L > >	1129
mln::Accumulator< mln::accu::count_labels< L > >	628
mln::Object< mln::accu::count_value< V > >	968
mln::Proxy< mln::accu::count_value< V > >	1129
mln::Accumulator< mln::accu::count_value< V > >	628
mln::Object< mln::accu::histo< V > >	968
mln::Proxy< mln::accu::histo< V > >	1129
mln::Accumulator< mln::accu::histo< V > >	628
mln::Object< mln::accu::label_used< L > >	968
mln::Proxy< mln::accu::label_used< L > >	1129
mln::Accumulator< mln::accu::label_used< L > >	628
mln::Object< mln::accu::logic::land >	968
mln::Proxy< mln::accu::logic::land >	1129
mln::Accumulator< mln::accu::logic::land >	628
mln::Object< mln::accu::logic::land_basic >	968
mln::Proxy< mln::accu::logic::land_basic >	1129
mln::Accumulator< mln::accu::logic::land_basic >	628
mln::Object< mln::accu::logic::lor >	968
mln::Proxy< mln::accu::logic::lor >	1129

mln::Accumulator< mln::accu::logic::lor >	628
mln::Object< mln::accu::logic::lor_basic >	968
mln::Proxy< mln::accu::logic::lor_basic >	1129
mln::Accumulator< mln::accu::logic::lor_basic >	628
mln::Object< mln::accu::maj_h< T > >	968
mln::Proxy< mln::accu::maj_h< T > >	1129
mln::Accumulator< mln::accu::maj_h< T > >	628
mln::Object< mln::accu::math::count< T > >	968
mln::Proxy< mln::accu::math::count< T > >	1129
mln::Accumulator< mln::accu::math::count< T > >	628
mln::Object< mln::accu::math::inf< T > >	968
mln::Proxy< mln::accu::math::inf< T > >	1129
mln::Accumulator< mln::accu::math::inf< T > >	628
mln::Object< mln::accu::math::sum< T, S > >	968
mln::Proxy< mln::accu::math::sum< T, S > >	1129
mln::Accumulator< mln::accu::math::sum< T, S > >	628
mln::Object< mln::accu::math::sup< T > >	968
mln::Proxy< mln::accu::math::sup< T > >	1129
mln::Accumulator< mln::accu::math::sup< T > >	628
mln::Object< mln::accu::max_site< I > >	968
mln::Proxy< mln::accu::max_site< I > >	1129
mln::Accumulator< mln::accu::max_site< I > >	628
mln::Object< mln::accu::meta::center >	968
mln::Meta_Accumulator< mln::accu::meta::center >	937
mln::Object< mln::accu::meta::count_adjacent_vertices >	968
mln::Meta_Accumulator< mln::accu::meta::count_adjacent_vertices >	937
mln::Object< mln::accu::meta::count_labels >	968
mln::Meta_Accumulator< mln::accu::meta::count_labels >	937
mln::Object< mln::accu::meta::count_value >	968
mln::Meta_Accumulator< mln::accu::meta::count_value >	937
mln::Object< mln::accu::meta::histo >	968
mln::Meta_Accumulator< mln::accu::meta::histo >	937
mln::Object< mln::accu::meta::label_used >	968
mln::Meta_Accumulator< mln::accu::meta::label_used >	937
mln::Object< mln::accu::meta::logic::land >	968
mln::Meta_Accumulator< mln::accu::meta::logic::land >	937
mln::Object< mln::accu::meta::logic::land_basic >	968
mln::Meta_Accumulator< mln::accu::meta::logic::land_basic >	937
mln::Object< mln::accu::meta::logic::lor >	968
mln::Meta_Accumulator< mln::accu::meta::logic::lor >	937
mln::Object< mln::accu::meta::logic::lor_basic >	968
mln::Meta_Accumulator< mln::accu::meta::logic::lor_basic >	937
mln::Object< mln::accu::meta::maj_h >	968
mln::Meta_Accumulator< mln::accu::meta::maj_h >	937
mln::Object< mln::accu::meta::math::count >	968
mln::Meta_Accumulator< mln::accu::meta::math::count >	937
mln::Object< mln::accu::meta::math::inf >	968

mln::Meta_Accumulator< mln::accu::meta::math::inf >	937
mln::Object< mln::accu::meta::math::sum >	968
mln::Meta_Accumulator< mln::accu::meta::math::sum >	937
mln::Object< mln::accu::meta::math::sup >	968
mln::Meta_Accumulator< mln::accu::meta::math::sup >	937
mln::Object< mln::accu::meta::max_site >	968
mln::Meta_Accumulator< mln::accu::meta::max_site >	937
mln::Object< mln::accu::meta::nil >	968
mln::Meta_Accumulator< mln::accu::meta::nil >	937
mln::Object< mln::accu::meta::p< mA > >	968
mln::Meta_Accumulator< mln::accu::meta::p< mA > >	937
mln::Object< mln::accu::meta::pair< A1, A2 > >	968
mln::Meta_Accumulator< mln::accu::meta::pair< A1, A2 > >	937
mln::Object< mln::accu::meta::rms >	968
mln::Meta_Accumulator< mln::accu::meta::rms >	937
mln::Object< mln::accu::meta::shape::bbox >	968
mln::Meta_Accumulator< mln::accu::meta::shape::bbox >	937
mln::Object< mln::accu::meta::shape::height >	968
mln::Meta_Accumulator< mln::accu::meta::shape::height >	937
mln::Object< mln::accu::meta::shape::volume >	968
mln::Meta_Accumulator< mln::accu::meta::shape::volume >	937
mln::Object< mln::accu::meta::stat::max >	968
mln::Meta_Accumulator< mln::accu::meta::stat::max >	937
mln::Object< mln::accu::meta::stat::max_h >	968
mln::Meta_Accumulator< mln::accu::meta::stat::max_h >	937
mln::Object< mln::accu::meta::stat::mean >	968
mln::Meta_Accumulator< mln::accu::meta::stat::mean >	937
mln::Object< mln::accu::meta::stat::median_alt< T > >	968
mln::Meta_Accumulator< mln::accu::meta::stat::median_alt< T > >	937
mln::Object< mln::accu::meta::stat::median_h >	968
mln::Meta_Accumulator< mln::accu::meta::stat::median_h >	937
mln::Object< mln::accu::meta::stat::min >	968
mln::Meta_Accumulator< mln::accu::meta::stat::min >	937
mln::Object< mln::accu::meta::stat::min_h >	968
mln::Meta_Accumulator< mln::accu::meta::stat::min_h >	937
mln::Object< mln::accu::meta::stat::rank >	968
mln::Meta_Accumulator< mln::accu::meta::stat::rank >	937
mln::Object< mln::accu::meta::stat::rank_high_quant >	968
mln::Meta_Accumulator< mln::accu::meta::stat::rank_high_quant >	937
mln::Object< mln::accu::meta::tuple< n, BOOST_PP_ENUM_PARAMS(10, T)> >	968
mln::Meta_Accumulator< mln::accu::meta::tuple< n, BOOST_PP_ENUM_PARAMS(10, T)> >	937
mln::Object< mln::accu::meta::val< mA > >	968
mln::Meta_Accumulator< mln::accu::meta::val< mA > >	937
mln::Object< mln::accu::nil< T > >	968
mln::Proxy< mln::accu::nil< T > >	1129
mln::Accumulator< mln::accu::nil< T > >	628

mln::Object< mln::accu::p< A > >	968
mln::Proxy< mln::accu::p< A > >	1129
mln::Accumulator< mln::accu::p< A > >	628
mln::Object< mln::accu::pair< A1, A2, T > >	968
mln::Proxy< mln::accu::pair< A1, A2, T > >	1129
mln::Accumulator< mln::accu::pair< A1, A2, T > >	628
mln::Object< mln::accu::pair< mln::accu::stat::min< V >, mln::accu::stat::max< V >, mln_	
argument(mln::accu::stat::min< V >) > >	968
mln::Proxy< mln::accu::pair< mln::accu::stat::min< V >, mln::accu::stat::max< V >,	
mln_argument(mln::accu::stat::min< V >) > >	1129
mln::Accumulator< mln::accu::pair< mln::accu::stat::min< V >, mln::accu::stat::max<	
V >, mln_argument(mln::accu::stat::min< V >) > >	628
mln::Object< mln::accu::rms< T, V > >	968
mln::Proxy< mln::accu::rms< T, V > >	1129
mln::Accumulator< mln::accu::rms< T, V > >	628
mln::Object< mln::accu::shape::bbox< P > >	968
mln::Proxy< mln::accu::shape::bbox< P > >	1129
mln::Accumulator< mln::accu::shape::bbox< P > >	628
mln::Object< mln::accu::shape::height< I > >	968
mln::Proxy< mln::accu::shape::height< I > >	1129
mln::Accumulator< mln::accu::shape::height< I > >	628
mln::Object< mln::accu::shape::volume< I > >	968
mln::Proxy< mln::accu::shape::volume< I > >	1129
mln::Accumulator< mln::accu::shape::volume< I > >	628
mln::Object< mln::accu::site_set::rectangularity< P > >	968
mln::Proxy< mln::accu::site_set::rectangularity< P > >	1129
mln::Accumulator< mln::accu::site_set::rectangularity< P > >	628
mln::Object< mln::accu::stat::deviation< T, S, M > >	968
mln::Proxy< mln::accu::stat::deviation< T, S, M > >	1129
mln::Accumulator< mln::accu::stat::deviation< T, S, M > >	628
mln::Object< mln::accu::stat::max< T > >	968
mln::Proxy< mln::accu::stat::max< T > >	1129
mln::Accumulator< mln::accu::stat::max< T > >	628
mln::Object< mln::accu::stat::max_h< V > >	968
mln::Proxy< mln::accu::stat::max_h< V > >	1129
mln::Accumulator< mln::accu::stat::max_h< V > >	628
mln::Object< mln::accu::stat::mean< T, S, M > >	968
mln::Proxy< mln::accu::stat::mean< T, S, M > >	1129
mln::Accumulator< mln::accu::stat::mean< T, S, M > >	628
mln::Object< mln::accu::stat::median_alt< mln::value::set< T > > >	968
mln::Proxy< mln::accu::stat::median_alt< mln::value::set< T > > >	1129
mln::Accumulator< mln::accu::stat::median_alt< mln::value::set< T > > >	628
mln::Object< mln::accu::stat::median_alt< S > >	968
mln::Proxy< mln::accu::stat::median_alt< S > >	1129
mln::Accumulator< mln::accu::stat::median_alt< S > >	628
mln::Object< mln::accu::stat::median_h< V > >	968
mln::Proxy< mln::accu::stat::median_h< V > >	1129
mln::Accumulator< mln::accu::stat::median_h< V > >	628

mln::Object< mln::accu::stat::meta::deviation >	968
mln::Meta_Accumulator< mln::accu::stat::meta::deviation >	937
mln::Object< mln::accu::stat::min< T > >	968
mln::Proxy< mln::accu::stat::min< T > >	1129
mln::Accumulator< mln::accu::stat::min< T > >	628
mln::Object< mln::accu::stat::min_h< V > >	968
mln::Proxy< mln::accu::stat::min_h< V > >	1129
mln::Accumulator< mln::accu::stat::min_h< V > >	628
mln::Object< mln::accu::stat::rank< bool > >	968
mln::Proxy< mln::accu::stat::rank< bool > >	1129
mln::Accumulator< mln::accu::stat::rank< bool > >	628
mln::Object< mln::accu::stat::rank< T > >	968
mln::Proxy< mln::accu::stat::rank< T > >	1129
mln::Accumulator< mln::accu::stat::rank< T > >	628
mln::Object< mln::accu::stat::rank_high_quant< T > >	968
mln::Proxy< mln::accu::stat::rank_high_quant< T > >	1129
mln::Accumulator< mln::accu::stat::rank_high_quant< T > >	628
mln::Object< mln::accu::stat::var< T > >	968
mln::Proxy< mln::accu::stat::var< T > >	1129
mln::Accumulator< mln::accu::stat::var< T > >	628
mln::Object< mln::accu::stat::variance< T, S, R > >	968
mln::Proxy< mln::accu::stat::variance< T, S, R > >	1129
mln::Accumulator< mln::accu::stat::variance< T, S, R > >	628
mln::Object< mln::accu::tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >	968
mln::Proxy< mln::accu::tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >	1129
mln::Accumulator< mln::accu::tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >	628
mln::Object< mln::accu::val< A > >	968
mln::Proxy< mln::accu::val< A > >	1129
mln::Accumulator< mln::accu::val< A > >	628
mln::Object< mln::algebra::mat< n, m, T > >	968
mln::Object< mln::algebra::quat >	968
mln::Value< mln::algebra::quat >	1298
mln::Object< mln::algebra::vec< 1, T > >	968
mln::Object< mln::algebra::vec< 2, T > >	968
mln::Object< mln::algebra::vec< 3, T > >	968
mln::Object< mln::algebra::vec< 4, T > >	968
mln::Object< mln::algebra::vec< n, C > >	968
mln::Object< mln::algebra::vec< n, T > >	968
mln::Object< mln::bkd_pixter1d< I > >	968
mln::Iterator< mln::bkd_pixter1d< I > >	898
mln::Pixel_Iterator< mln::bkd_pixter1d< I > >	1108
mln::Object< mln::bkd_pixter2d< I > >	968
mln::Iterator< mln::bkd_pixter2d< I > >	898
mln::Pixel_Iterator< mln::bkd_pixter2d< I > >	1108
mln::Object< mln::bkd_pixter3d< I > >	968
mln::Iterator< mln::bkd_pixter3d< I > >	898
mln::Pixel_Iterator< mln::bkd_pixter3d< I > >	1108
mln::Object< mln::box< P > >	968

mln::Site_Set< mln::box< P > >	1145
mln::Box< mln::box< P > >	648
mln::Object< mln::box_runend_piter< P > >	968
mln::Proxy< mln::box_runend_piter< P > >	1129
mln::Site_Proxy< mln::box_runend_piter< P > >	1143
mln::Site_Iterator< mln::box_runend_piter< P > >	1141
mln::Object< mln::box_runstart_piter< P > >	968
mln::Proxy< mln::box_runstart_piter< P > >	1129
mln::Site_Proxy< mln::box_runstart_piter< P > >	1143
mln::Site_Iterator< mln::box_runstart_piter< P > >	1141
mln::Object< mln::canvas::browsing::backdiagonal2d_t >	968
mln::Browsing< mln::canvas::browsing::backdiagonal2d_t >	657
mln::Object< mln::canvas::browsing::breadth_first_search_t >	968
mln::Browsing< mln::canvas::browsing::breadth_first_search_t >	657
mln::Object< mln::canvas::browsing::depth_first_search_t >	968
mln::Browsing< mln::canvas::browsing::depth_first_search_t >	657
mln::Object< mln::canvas::browsing::diagonal2d_t >	968
mln::Browsing< mln::canvas::browsing::diagonal2d_t >	657
mln::Object< mln::canvas::browsing::dir_struct_elt_incr_update_t >	968
mln::Browsing< mln::canvas::browsing::dir_struct_elt_incr_update_t >	657
mln::Object< mln::canvas::browsing::directional_t >	968
mln::Browsing< mln::canvas::browsing::directional_t >	657
mln::Object< mln::canvas::browsing::fwd_t >	968
mln::Browsing< mln::canvas::browsing::fwd_t >	657
mln::Object< mln::canvas::browsing::hyper_directional_t >	968
mln::Browsing< mln::canvas::browsing::hyper_directional_t >	657
mln::Object< mln::canvas::browsing::snake_fwd_t >	968
mln::Browsing< mln::canvas::browsing::snake_fwd_t >	657
mln::Object< mln::canvas::browsing::snake_generic_t >	968
mln::Browsing< mln::canvas::browsing::snake_generic_t >	657
mln::Object< mln::canvas::browsing::snake_vert_t >	968
mln::Browsing< mln::canvas::browsing::snake_vert_t >	657
mln::Object< mln::ch_piter_image< I, Fwd > >	968
mln::Image< mln::ch_piter_image< I, Fwd > >	872
mln::Object< mln::complex_image< D, G, V > >	968
mln::Image< mln::complex_image< D, G, V > >	872
mln::Object< mln::complex_neighborhood_bkd_piter< I, G, N > >	968
mln::Proxy< mln::complex_neighborhood_bkd_piter< I, G, N > >	1129
mln::Site_Proxy< mln::complex_neighborhood_bkd_piter< I, G, N > >	1143
mln::Site_Iterator< mln::complex_neighborhood_bkd_piter< I, G, N > >	1141
mln::Object< mln::complex_neighborhood_fwd_piter< I, G, N > >	968
mln::Proxy< mln::complex_neighborhood_fwd_piter< I, G, N > >	1129
mln::Site_Proxy< mln::complex_neighborhood_fwd_piter< I, G, N > >	1143
mln::Site_Iterator< mln::complex_neighborhood_fwd_piter< I, G, N > >	1141
mln::Object< mln::complex_psite< D, G > >	968
mln::Proxy< mln::complex_psite< D, G > >	1129
mln::Site_Proxy< mln::complex_psite< D, G > >	1143

mln::Pseudo_Site< mln::complex_psite< D, G > >	1131
mln::Object< mln::complex_window_bkd_piter< I, G, W > >	968
mln::Proxy< mln::complex_window_bkd_piter< I, G, W > >	1129
mln::Site_Proxy< mln::complex_window_bkd_piter< I, G, W > >	1143
mln::Site_Iterator< mln::complex_window_bkd_piter< I, G, W > >	1141
mln::Object< mln::complex_window_fwd_piter< I, G, W > >	968
mln::Proxy< mln::complex_window_fwd_piter< I, G, W > >	1129
mln::Site_Proxy< mln::complex_window_fwd_piter< I, G, W > >	1143
mln::Site_Iterator< mln::complex_window_fwd_piter< I, G, W > >	1141
mln::Object< mln::concrete >	968
mln::Object< mln::decorated_image< I, D > >	968
mln::Image< mln::decorated_image< I, D > >	872
mln::Object< mln::dist >	968
mln::Function< mln::dist >	817
mln::Function_vv2v< mln::dist >	822
mln::Object< mln::dpoint< G, C > >	968
mln::Gdpoint< mln::dpoint< G, C > >	829
mln::Object< mln::dpoints_bkd_pixter< I > >	968
mln::Iterator< mln::dpoints_bkd_pixter< I > >	898
mln::Pixel_Iterator< mln::dpoints_bkd_pixter< I > >	1108
mln::Object< mln::dpoints_fwd_pixter< I > >	968
mln::Iterator< mln::dpoints_fwd_pixter< I > >	898
mln::Pixel_Iterator< mln::dpoints_fwd_pixter< I > >	1108
mln::Object< mln::dpsites_bkd_piter< V > >	968
mln::Proxy< mln::dpsites_bkd_piter< V > >	1129
mln::Site_Proxy< mln::dpsites_bkd_piter< V > >	1143
mln::Site_Iterator< mln::dpsites_bkd_piter< V > >	1141
mln::Object< mln::dpsites_fwd_piter< V > >	968
mln::Proxy< mln::dpsites_fwd_piter< V > >	1129
mln::Site_Proxy< mln::dpsites_fwd_piter< V > >	1143
mln::Site_Iterator< mln::dpsites_fwd_piter< V > >	1141
mln::Object< mln::edge_image< P, V, G > >	968
mln::Image< mln::edge_image< P, V, G > >	872
mln::Object< mln::edge_to_color< I, V > >	968
mln::Function< mln::edge_to_color< I, V > >	817
mln::Function_v2v< mln::edge_to_color< I, V > >	820
mln::Object< mln::extended< I > >	968
mln::Image< mln::extended< I > >	872
mln::Object< mln::extension_fun< I, F > >	968
mln::Image< mln::extension_fun< I, F > >	872
mln::Object< mln::extension_ima< I, J > >	968
mln::Image< mln::extension_ima< I, J > >	872
mln::Object< mln::extension_val< I > >	968
mln::Image< mln::extension_val< I > >	872
mln::Object< mln::faces_psite< N, D, P > >	968
mln::Proxy< mln::faces_psite< N, D, P > >	1129
mln::Site_Proxy< mln::faces_psite< N, D, P > >	1143
mln::Pseudo_Site< mln::faces_psite< N, D, P > >	1131

mln::Object< mln::flat_image< T, S > >	968
mln::Image< mln::flat_image< T, S > >	872
mln::Object< mln::fun::abs >	968
mln::Meta_Function< mln::fun::abs >	938
mln::Meta_Function_v2v< mln::fun::abs >	939
mln::Object< mln::fun::access::mean >	968
mln::Meta_Function< mln::fun::access::mean >	938
mln::Meta_Function_v2v< mln::fun::access::mean >	939
mln::Object< mln::fun::accu_result >	968
mln::Meta_Function< mln::fun::accu_result >	938
mln::Meta_Function_v2v< mln::fun::accu_result >	939
mln::Object< mln::fun::blue >	968
mln::Meta_Function< mln::fun::blue >	938
mln::Meta_Function_v2v< mln::fun::blue >	939
mln::Object< mln::fun::col >	968
mln::Meta_Function< mln::fun::col >	938
mln::Meta_Function_v2v< mln::fun::col >	939
mln::Object< mln::fun::comp >	968
mln::Meta_Function< mln::fun::comp >	938
mln::Meta_Function_v2v< mln::fun::comp >	939
mln::Object< mln::fun::comp_count >	968
mln::Meta_Function< mln::fun::comp_count >	938
mln::Meta_Function_v2v< mln::fun::comp_count >	939
mln::Object< mln::fun::compose >	968
mln::Meta_Function< mln::fun::compose >	938
mln::Meta_Function_vv2v< mln::fun::compose >	940
mln::Object< mln::fun::cos >	968
mln::Meta_Function< mln::fun::cos >	938
mln::Meta_Function_v2v< mln::fun::cos >	939
mln::Object< mln::fun::from_accu< A > >	968
mln::Meta_Function< mln::fun::from_accu< A > >	938
mln::Meta_Function_v2v< mln::fun::from_accu< A > >	939
mln::Object< mln::fun::green >	968
mln::Meta_Function< mln::fun::green >	938
mln::Meta_Function_v2v< mln::fun::green >	939
mln::Object< mln::fun::i2v::all_to< T > >	968
mln::Function< mln::fun::i2v::all_to< T > >	817
mln::Function_v2v< mln::fun::i2v::all_to< T > >	820
mln::Object< mln::fun::i2v::value_at_index< bool > >	968
mln::Function< mln::fun::i2v::value_at_index< bool > >	817
mln::Function_v2v< mln::fun::i2v::value_at_index< bool > >	820
mln::Object< mln::fun::i2v::value_at_index< T > >	968
mln::Function< mln::fun::i2v::value_at_index< T > >	817
mln::Function_v2v< mln::fun::i2v::value_at_index< T > >	820
mln::Object< mln::fun::inf >	968
mln::Meta_Function< mln::fun::inf >	938
mln::Meta_Function_vv2v< mln::fun::inf >	940
mln::Object< mln::fun::ithcomp >	968

mln::Meta_Function< mln::fun::ithcomp >	938
mln::Meta_Function_v2v< mln::fun::ithcomp >	940
mln::Object< mln::fun::norm::l1 >	968
mln::Meta_Function< mln::fun::norm::l1 >	938
mln::Meta_Function_v2v< mln::fun::norm::l1 >	939
mln::Object< mln::fun::norm::l2 >	968
mln::Meta_Function< mln::fun::norm::l2 >	938
mln::Meta_Function_v2v< mln::fun::norm::l2 >	939
mln::Object< mln::fun::norm::linfty >	968
mln::Meta_Function< mln::fun::norm::linfty >	938
mln::Meta_Function_v2v< mln::fun::norm::linfty >	939
mln::Object< mln::fun::p2b::antilogy >	968
mln::Function< mln::fun::p2b::antilogy >	817
mln::Function_v2v< mln::fun::p2b::antilogy >	820
mln::Function_v2b< mln::fun::p2b::antilogy >	819
mln::Object< mln::fun::p2b::big_chess< B > >	968
mln::Function< mln::fun::p2b::big_chess< B > >	817
mln::Function_v2v< mln::fun::p2b::big_chess< B > >	820
mln::Function_v2b< mln::fun::p2b::big_chess< B > >	819
mln::Object< mln::fun::p2b::chess >	968
mln::Function< mln::fun::p2b::chess >	817
mln::Function_v2v< mln::fun::p2b::chess >	820
mln::Function_v2b< mln::fun::p2b::chess >	819
mln::Object< mln::fun::p2b::has< I > >	968
mln::Function< mln::fun::p2b::has< I > >	817
mln::Function_v2v< mln::fun::p2b::has< I > >	820
mln::Function_v2b< mln::fun::p2b::has< I > >	819
mln::Object< mln::fun::p2b::tautology >	968
mln::Function< mln::fun::p2b::tautology >	817
mln::Function_v2v< mln::fun::p2b::tautology >	820
mln::Function_v2b< mln::fun::p2b::tautology >	819
mln::Object< mln::fun::p2p::fold< P, dir_0, dir_1, dir_2 > >	968
mln::Function< mln::fun::p2p::fold< P, dir_0, dir_1, dir_2 > >	817
mln::Function_v2v< mln::fun::p2p::fold< P, dir_0, dir_1, dir_2 > >	820
mln::Object< mln::fun::p2p::mirror< B > >	968
mln::Function< mln::fun::p2p::mirror< B > >	817
mln::Function_v2v< mln::fun::p2p::mirror< B > >	820
mln::Object< mln::fun::p2p::translation_t< P > >	968
mln::Function< mln::fun::p2p::translation_t< P > >	817
mln::Function_v2v< mln::fun::p2p::translation_t< P > >	820
mln::Object< mln::fun::p2v::iota >	968
mln::Function< mln::fun::p2v::iota >	817
mln::Function_v2v< mln::fun::p2v::iota >	820
mln::Object< mln::fun::red >	968
mln::Meta_Function< mln::fun::red >	938
mln::Meta_Function_v2v< mln::fun::red >	939
mln::Object< mln::fun::row >	968
mln::Meta_Function< mln::fun::row >	938

mln::Meta_Function_v2v< mln::fun::row >	939
mln::Object< mln::fun::scomp< ith > >	968
mln::Meta_Function< mln::fun::scomp< ith > >	938
mln::Meta_Function_v2v< mln::fun::scomp< ith > >	939
mln::Object< mln::fun::sli >	968
mln::Meta_Function< mln::fun::sli >	938
mln::Meta_Function_v2v< mln::fun::sli >	939
mln::Object< mln::fun::spe::binary< Fun, T1, T2 > >	968
mln::Function< mln::fun::spe::binary< Fun, T1, T2 > >	817
mln::Function_v2v< mln::fun::spe::binary< Fun, T1, T2 > >	820
mln::Object< mln::fun::spe::unary< Fun, T > >	968
mln::Function< mln::fun::spe::unary< Fun, T > >	817
mln::Function_v2v< mln::fun::spe::unary< Fun, T > >	820
mln::Object< mln::fun::stat::mahalanobis< V > >	968
mln::Function< mln::fun::stat::mahalanobis< V > >	817
mln::Function_v2v< mln::fun::stat::mahalanobis< V > >	820
mln::Object< mln::fun::sup >	968
mln::Meta_Function< mln::fun::sup >	938
mln::Meta_Function_vv2v< mln::fun::sup >	940
mln::Object< mln::fun::v2b::lnot< V > >	968
mln::Function< mln::fun::v2b::lnot< V > >	817
mln::Function_v2v< mln::fun::v2b::lnot< V > >	820
mln::Function_v2b< mln::fun::v2b::lnot< V > >	819
mln::Object< mln::fun::v2b::threshold< V > >	968
mln::Function< mln::fun::v2b::threshold< V > >	817
mln::Function_v2v< mln::fun::v2b::threshold< V > >	820
mln::Function_v2b< mln::fun::v2b::threshold< V > >	819
mln::Object< mln::fun::v2i::index_of_value< bool > >	968
mln::Function< mln::fun::v2i::index_of_value< bool > >	817
mln::Function_v2v< mln::fun::v2i::index_of_value< bool > >	820
mln::Object< mln::fun::v2i::index_of_value< T > >	968
mln::Function< mln::fun::v2i::index_of_value< T > >	817
mln::Function_v2v< mln::fun::v2i::index_of_value< T > >	820
mln::Object< mln::fun::v2v::abs< V > >	968
mln::Function< mln::fun::v2v::abs< V > >	817
mln::Function_v2v< mln::fun::v2v::abs< V > >	820
mln::Object< mln::fun::v2v::cast< V > >	968
mln::Function< mln::fun::v2v::cast< V > >	817
mln::Function_v2v< mln::fun::v2v::cast< V > >	820
mln::Object< mln::fun::v2v::ch_function_value< F, V > >	968
mln::Function< mln::fun::v2v::ch_function_value< F, V > >	817
mln::Function_v2v< mln::fun::v2v::ch_function_value< F, V > >	820
mln::Object< mln::fun::v2v::component< T, i > >	968
mln::Function< mln::fun::v2v::component< T, i > >	817
mln::Function_v2v< mln::fun::v2v::component< T, i > >	820
mln::Object< mln::fun::v2v::convert< V > >	968
mln::Function< mln::fun::v2v::convert< V > >	817
mln::Function_v2v< mln::fun::v2v::convert< V > >	820

mln::Object< mln::fun::v2v::enc< V > >	968
mln::Function< mln::fun::v2v::enc< V > >	817
mln::Function_v2v< mln::fun::v2v::enc< V > >	820
mln::Object< mln::fun::v2v::f_hsi_to_rgb< T_rgb > >	968
mln::Function< mln::fun::v2v::f_hsi_to_rgb< T_rgb > >	817
mln::Function_v2v< mln::fun::v2v::f_hsi_to_rgb< T_rgb > >	820
mln::Object< mln::fun::v2v::f_hsl_to_rgb< T_rgb > >	968
mln::Function< mln::fun::v2v::f_hsl_to_rgb< T_rgb > >	817
mln::Function_v2v< mln::fun::v2v::f_hsl_to_rgb< T_rgb > >	820
mln::Object< mln::fun::v2v::f_rgb_to_hsi< T_hsi > >	968
mln::Function< mln::fun::v2v::f_rgb_to_hsi< T_hsi > >	817
mln::Function_v2v< mln::fun::v2v::f_rgb_to_hsi< T_hsi > >	820
mln::Object< mln::fun::v2v::f_rgb_to_hsl< T_hsl > >	968
mln::Function< mln::fun::v2v::f_rgb_to_hsl< T_hsl > >	817
mln::Function_v2v< mln::fun::v2v::f_rgb_to_hsl< T_hsl > >	820
mln::Object< mln::fun::v2v::l1_norm< V, R > >	968
mln::Function< mln::fun::v2v::l1_norm< V, R > >	817
mln::Function_v2v< mln::fun::v2v::l1_norm< V, R > >	820
mln::Object< mln::fun::v2v::l2_norm< V, R > >	968
mln::Function< mln::fun::v2v::l2_norm< V, R > >	817
mln::Function_v2v< mln::fun::v2v::l2_norm< V, R > >	820
mln::Object< mln::fun::v2v::linear< V, T, R > >	968
mln::Function< mln::fun::v2v::linear< V, T, R > >	817
mln::Function_v2v< mln::fun::v2v::linear< V, T, R > >	820
mln::Object< mln::fun::v2v::linear_sat< V, T, R > >	968
mln::Function< mln::fun::v2v::linear_sat< V, T, R > >	817
mln::Function_v2v< mln::fun::v2v::linear_sat< V, T, R > >	820
mln::Object< mln::fun::v2v::linfty_norm< V, R > >	968
mln::Function< mln::fun::v2v::linfty_norm< V, R > >	817
mln::Function_v2v< mln::fun::v2v::linfty_norm< V, R > >	820
mln::Object< mln::fun::v2v::projection< P, dir > >	968
mln::Function< mln::fun::v2v::projection< P, dir > >	817
mln::Function_v2v< mln::fun::v2v::projection< P, dir > >	820
mln::Object< mln::fun::v2v::saturate< V > >	968
mln::Function< mln::fun::v2v::saturate< V > >	817
mln::Function_v2v< mln::fun::v2v::saturate< V > >	820
mln::Object< mln::fun::v2v::wrap< L > >	968
mln::Function< mln::fun::v2v::wrap< L > >	817
mln::Function_v2v< mln::fun::v2v::wrap< L > >	820
mln::Object< mln::fun::v2w2v::cos< V > >	968
mln::Function< mln::fun::v2w2v::cos< V > >	817
mln::Function_v2v< mln::fun::v2w2v::cos< V > >	820
mln::Object< mln::fun::v2w_w2v::l1_norm< V, R > >	968
mln::Function< mln::fun::v2w_w2v::l1_norm< V, R > >	817
mln::Function_v2v< mln::fun::v2w_w2v::l1_norm< V, R > >	820
mln::Object< mln::fun::v2w_w2v::l2_norm< V, R > >	968
mln::Function< mln::fun::v2w_w2v::l2_norm< V, R > >	817
mln::Function_v2v< mln::fun::v2w_w2v::l2_norm< V, R > >	820

mln::Object< mln::fun::v2w_w2v::linfty_norm< V, R > >	968
mln::Function< mln::fun::v2w_w2v::linfty_norm< V, R > >	817
mln::Function_v2v< mln::fun::v2w_w2v::linfty_norm< V, R > >	820
mln::Object< mln::fun::vv2b::eq< L, R > >	968
mln::Function< mln::fun::vv2b::eq< L, R > >	817
mln::Function_vv2b< mln::fun::vv2b::eq< L, R > >	821
mln::Object< mln::fun::vv2b::ge< L, R > >	968
mln::Function< mln::fun::vv2b::ge< L, R > >	817
mln::Function_vv2b< mln::fun::vv2b::ge< L, R > >	821
mln::Object< mln::fun::vv2b::gt< L, R > >	968
mln::Function< mln::fun::vv2b::gt< L, R > >	817
mln::Function_vv2b< mln::fun::vv2b::gt< L, R > >	821
mln::Object< mln::fun::vv2b::implies< L, R > >	968
mln::Function< mln::fun::vv2b::implies< L, R > >	817
mln::Function_vv2b< mln::fun::vv2b::implies< L, R > >	821
mln::Object< mln::fun::vv2b::le< L, R > >	968
mln::Function< mln::fun::vv2b::le< L, R > >	817
mln::Function_vv2b< mln::fun::vv2b::le< L, R > >	821
mln::Object< mln::fun::vv2b::lt< L, R > >	968
mln::Function< mln::fun::vv2b::lt< L, R > >	817
mln::Function_vv2b< mln::fun::vv2b::lt< L, R > >	821
mln::Object< mln::fun::vv2v::diff_abs< V > >	968
mln::Function< mln::fun::vv2v::diff_abs< V > >	817
mln::Function_vv2v< mln::fun::vv2v::diff_abs< V > >	822
mln::Object< mln::fun::vv2v::land< L, R > >	968
mln::Function< mln::fun::vv2v::land< L, R > >	817
mln::Function_vv2v< mln::fun::vv2v::land< L, R > >	822
mln::Object< mln::fun::vv2v::land_not< L, R > >	968
mln::Function< mln::fun::vv2v::land_not< L, R > >	817
mln::Function_vv2v< mln::fun::vv2v::land_not< L, R > >	822
mln::Object< mln::fun::vv2v::lor< L, R > >	968
mln::Function< mln::fun::vv2v::lor< L, R > >	817
mln::Function_vv2v< mln::fun::vv2v::lor< L, R > >	822
mln::Object< mln::fun::vv2v::lxor< L, R > >	968
mln::Function< mln::fun::vv2v::lxor< L, R > >	817
mln::Function_vv2v< mln::fun::vv2v::lxor< L, R > >	822
mln::Object< mln::fun::vv2v::max< V > >	968
mln::Function< mln::fun::vv2v::max< V > >	817
mln::Function_vv2v< mln::fun::vv2v::max< V > >	822
mln::Object< mln::fun::vv2v::min< L, R > >	968
mln::Function< mln::fun::vv2v::min< L, R > >	817
mln::Function_vv2v< mln::fun::vv2v::min< L, R > >	822
mln::Object< mln::fun::vv2v::vec< V > >	968
mln::Function< mln::fun::vv2v::vec< V > >	817
mln::Function_vv2v< mln::fun::vv2v::vec< V > >	822
mln::Object< mln::fun::x2v::l1_norm< V > >	968
mln::Function< mln::fun::x2v::l1_norm< V > >	817
mln::Function_v2v< mln::fun::x2v::l1_norm< V > >	820

mln::Object< mln::fun::x2x::rotation< n, C > >	968
mln::Function< mln::fun::x2x::rotation< n, C > >	817
mln::Function_v2v< mln::fun::x2x::rotation< n, C > >	820
mln::Object< mln::fun::x2x::translation< n, C > >	968
mln::Function< mln::fun::x2x::translation< n, C > >	817
mln::Function_v2v< mln::fun::x2x::translation< n, C > >	820
mln::Object< mln::fun_image< F, I > >	968
mln::Image< mln::fun_image< F, I > >	872
mln::Object< mln::fwd_pixter1d< I > >	968
mln::Iterator< mln::fwd_pixter1d< I > >	898
mln::Pixel_Iterator< mln::fwd_pixter1d< I > >	1108
mln::Object< mln::fwd_pixter2d< I > >	968
mln::Iterator< mln::fwd_pixter2d< I > >	898
mln::Pixel_Iterator< mln::fwd_pixter2d< I > >	1108
mln::Object< mln::fwd_pixter3d< I > >	968
mln::Iterator< mln::fwd_pixter3d< I > >	898
mln::Pixel_Iterator< mln::fwd_pixter3d< I > >	1108
mln::Object< mln::graph_elt_mixed_window< G, S, S2 > >	968
mln::Window< mln::graph_elt_mixed_window< G, S, S2 > >	1367
mln::graph_window_base< S2::fun_t::result, mln::graph_elt_mixed_window< G, S, S2 > >	860
mln::Object< mln::graph_elt_window< G, S > >	968
mln::Window< mln::graph_elt_window< G, S > >	1367
mln::graph_window_base< S::fun_t::result, mln::graph_elt_window< G, S > >	860
mln::Object< mln::graph_elt_window_if< G, S, I > >	968
mln::Window< mln::graph_elt_window_if< G, S, I > >	1367
mln::graph_window_base< S::fun_t::result, mln::graph_elt_window_if< G, S, I > >	860
mln::Object< mln::graph_window_if_piter< S, W, I > >	968
mln::Proxy< mln::graph_window_if_piter< S, W, I > >	1129
mln::Site_Proxy< mln::graph_window_if_piter< S, W, I > >	1143
mln::Site_Iterator< mln::graph_window_if_piter< S, W, I > >	1141
mln::Object< mln::graph_window_piter< S, W, I > >	968
mln::Proxy< mln::graph_window_piter< S, W, I > >	1129
mln::Site_Proxy< mln::graph_window_piter< S, W, I > >	1143
mln::Site_Iterator< mln::graph_window_piter< S, W, I > >	1141
mln::Object< mln::grid::cube >	968
mln::Mesh< mln::grid::cube >	936
mln::Regular_Grid< mln::grid::cube >	1136
mln::Object< mln::grid::hexa >	968
mln::Mesh< mln::grid::hexa >	936
mln::Regular_Grid< mln::grid::hexa >	1136
mln::Object< mln::grid::square >	968
mln::Mesh< mln::grid::square >	936
mln::Regular_Grid< mln::grid::square >	1136
mln::Object< mln::grid::tick >	968
mln::Mesh< mln::grid::tick >	936
mln::Regular_Grid< mln::grid::tick >	1136
mln::Object< mln::hexa< I > >	968

mln::Image< mln::hexa< I > >	872
mln::Object< mln::hexa< mln::image2d< V > > >	968
mln::Image< mln::hexa< mln::image2d< V > > >	872
mln::Object< mln::histo::point_from_value< T > >	968
mln::Function< mln::histo::point_from_value< T > >	817
mln::Function_v2v< mln::histo::point_from_value< T > >	820
mln::Object< mln::image1d< T > >	968
mln::Image< mln::image1d< T > >	872
mln::Object< mln::image2d< T > >	968
mln::Image< mln::image2d< T > >	872
mln::Object< mln::image3d< T > >	968
mln::Image< mln::image3d< T > >	872
mln::Object< mln::image_if< I, F > >	968
mln::Image< mln::image_if< I, F > >	872
mln::Object< mln::interpolated< I, F > >	968
mln::Image< mln::interpolated< I, F > >	872
mln::Object< mln::labeled_image< I > >	968
mln::Image< mln::labeled_image< I > >	872
mln::Object< mln::lazy_image< I, F, B > >	968
mln::Image< mln::lazy_image< I, F, B > >	872
mln::Object< mln::literal::black_t >	968
mln::Literal< mln::literal::black_t >	910
mln::Object< mln::literal::blue_t >	968
mln::Literal< mln::literal::blue_t >	910
mln::Object< mln::literal::brown_t >	968
mln::Literal< mln::literal::brown_t >	910
mln::Object< mln::literal::cyan_t >	968
mln::Literal< mln::literal::cyan_t >	910
mln::Object< mln::literal::dark_gray_t >	968
mln::Literal< mln::literal::dark_gray_t >	910
mln::Object< mln::literal::green_t >	968
mln::Literal< mln::literal::green_t >	910
mln::Object< mln::literal::identity_t >	968
mln::Literal< mln::literal::identity_t >	910
mln::Object< mln::literal::light_gray_t >	968
mln::Literal< mln::literal::light_gray_t >	910
mln::Object< mln::literal::lime_t >	968
mln::Literal< mln::literal::lime_t >	910
mln::Object< mln::literal::magenta_t >	968
mln::Literal< mln::literal::magenta_t >	910
mln::Object< mln::literal::max_t >	968
mln::Literal< mln::literal::max_t >	910
mln::Object< mln::literal::medium_gray_t >	968
mln::Literal< mln::literal::medium_gray_t >	910
mln::Object< mln::literal::min_t >	968
mln::Literal< mln::literal::min_t >	910
mln::Object< mln::literal::olive_t >	968

mln::Literal< mln::literal::olive_t >	910
mln::Object< mln::literal::one_t >	968
mln::Literal< mln::literal::one_t >	910
mln::Object< mln::literal::orange_t >	968
mln::Literal< mln::literal::orange_t >	910
mln::Object< mln::literal::origin_t >	968
mln::Literal< mln::literal::origin_t >	910
mln::Object< mln::literal::pink_t >	968
mln::Literal< mln::literal::pink_t >	910
mln::Object< mln::literal::purple_t >	968
mln::Literal< mln::literal::purple_t >	910
mln::Object< mln::literal::red_t >	968
mln::Literal< mln::literal::red_t >	910
mln::Object< mln::literal::teal_t >	968
mln::Literal< mln::literal::teal_t >	910
mln::Object< mln::literal::violet_t >	968
mln::Literal< mln::literal::violet_t >	910
mln::Object< mln::literal::white_t >	968
mln::Literal< mln::literal::white_t >	910
mln::Object< mln::literal::yellow_t >	968
mln::Literal< mln::literal::yellow_t >	910
mln::Object< mln::literal::zero_t >	968
mln::Literal< mln::literal::zero_t >	910
mln::Object< mln::math::round< R > >	968
mln::Function< mln::math::round< R > >	817
mln::Function_v2v< mln::math::round< R > >	820
mln::Object< mln::metal::array1d< T, Size > >	968
mln::Object< mln::metal::array2d< T, r, c > >	968
mln::Object< mln::metal::array3d< T, s, r, c > >	968
mln::Object< mln::metal::mat< n, m, T > >	968
mln::Object< mln::metal::vec< 1, T > >	968
mln::Object< mln::metal::vec< 2, T > >	968
mln::Object< mln::metal::vec< 3, T > >	968
mln::Object< mln::metal::vec< 4, T > >	968
mln::Object< mln::metal::vec< n, T > >	968
mln::Object< mln::mixed_neighb< W > >	968
mln::Neighborhood< mln::mixed_neighb< W > >	966
mln::Object< mln::morpho::attribute::card< I > >	968
mln::Proxy< mln::morpho::attribute::card< I > >	1129
mln::Accumulator< mln::morpho::attribute::card< I > >	628
mln::Object< mln::morpho::attribute::count_adjacent_vertices< I > >	968
mln::Proxy< mln::morpho::attribute::count_adjacent_vertices< I > >	1129
mln::Accumulator< mln::morpho::attribute::count_adjacent_vertices< I > >	628
mln::Object< mln::morpho::attribute::height< I > >	968
mln::Proxy< mln::morpho::attribute::height< I > >	1129
mln::Accumulator< mln::morpho::attribute::height< I > >	628
mln::Object< mln::morpho::attribute::sharpness< I > >	968
mln::Proxy< mln::morpho::attribute::sharpness< I > >	1129

mln::Accumulator< mln::morpho::attribute::sharpness< I > >	628
mln::Object< mln::morpho::attribute::sum< I, S > >	968
mln::Proxy< mln::morpho::attribute::sum< I, S > >	1129
mln::Accumulator< mln::morpho::attribute::sum< I, S > >	628
mln::Object< mln::morpho::attribute::volume< I > >	968
mln::Proxy< mln::morpho::attribute::volume< I > >	1129
mln::Accumulator< mln::morpho::attribute::volume< I > >	628
mln::Object< mln::morpho::tree::asc_propagation >	968
mln::Object< mln::morpho::tree::depth1st_piter< T > >	968
mln::Proxy< mln::morpho::tree::depth1st_piter< T > >	1129
mln::Site_Proxy< mln::morpho::tree::depth1st_piter< T > >	1143
mln::Site_Iterator< mln::morpho::tree::depth1st_piter< T > >	1141
mln::Object< mln::morpho::tree::desc_propagation >	968
mln::Object< mln::morpho::tree::dn_leaf_piter< T > >	968
mln::Proxy< mln::morpho::tree::dn_leaf_piter< T > >	1129
mln::Site_Proxy< mln::morpho::tree::dn_leaf_piter< T > >	1143
mln::Site_Iterator< mln::morpho::tree::dn_leaf_piter< T > >	1141
mln::Object< mln::morpho::tree::dn_node_piter< T > >	968
mln::Proxy< mln::morpho::tree::dn_node_piter< T > >	1129
mln::Site_Proxy< mln::morpho::tree::dn_node_piter< T > >	1143
mln::Site_Iterator< mln::morpho::tree::dn_node_piter< T > >	1141
mln::Object< mln::morpho::tree::dn_site_piter< T > >	968
mln::Proxy< mln::morpho::tree::dn_site_piter< T > >	1129
mln::Site_Proxy< mln::morpho::tree::dn_site_piter< T > >	1143
mln::Site_Iterator< mln::morpho::tree::dn_site_piter< T > >	1141
mln::Object< mln::morpho::tree::up_leaf_piter< T > >	968
mln::Proxy< mln::morpho::tree::up_leaf_piter< T > >	1129
mln::Site_Proxy< mln::morpho::tree::up_leaf_piter< T > >	1143
mln::Site_Iterator< mln::morpho::tree::up_leaf_piter< T > >	1141
mln::Object< mln::morpho::tree::up_node_piter< T > >	968
mln::Proxy< mln::morpho::tree::up_node_piter< T > >	1129
mln::Site_Proxy< mln::morpho::tree::up_node_piter< T > >	1143
mln::Site_Iterator< mln::morpho::tree::up_node_piter< T > >	1141
mln::Object< mln::morpho::tree::up_site_piter< T > >	968
mln::Proxy< mln::morpho::tree::up_site_piter< T > >	1129
mln::Site_Proxy< mln::morpho::tree::up_site_piter< T > >	1143
mln::Site_Iterator< mln::morpho::tree::up_site_piter< T > >	1141
mln::Object< mln::my_ext >	968
mln::Function< mln::my_ext >	817
mln::Function_v2v< mln::my_ext >	820
mln::Object< mln::my_image2d< T > >	968
mln::Image< mln::my_image2d< T > >	872
mln::Object< mln::myfun >	968
mln::Function< mln::myfun >	817
mln::Function_vv2v< mln::myfun >	822
mln::Object< mln::neighb< mln::graph_elt_mixed_window< G, S, S2 > > >	968
mln::Neighborhood< mln::neighb< mln::graph_elt_mixed_window< G, S, S2 > > >	966
mln::Object< mln::neighb< mln::graph_elt_window< G, S > > >	968

mln::Neighborhood< mln::neighb< mln::graph_elt_window< G, S > > >	966
mln::Object< mln::neighb< mln::graph_elt_window_if< G, S, I > > >	968
mln::Neighborhood< mln::neighb< mln::graph_elt_window_if< G, S, I > > >	966
mln::Object< mln::neighb< W > >	968
mln::Neighborhood< mln::neighb< W > >	966
mln::Object< mln::neighb_bkd_niter< W > >	968
mln::Proxy< mln::neighb_bkd_niter< W > >	1129
mln::Site_Proxy< mln::neighb_bkd_niter< W > >	1143
mln::Site_Iterator< mln::neighb_bkd_niter< W > >	1141
mln::Object< mln::neighb_fwd_niter< W > >	968
mln::Proxy< mln::neighb_fwd_niter< W > >	1129
mln::Site_Proxy< mln::neighb_fwd_niter< W > >	1143
mln::Site_Iterator< mln::neighb_fwd_niter< W > >	1141
mln::Object< mln::p2p_image< I, F > >	968
mln::Image< mln::p2p_image< I, F > >	872
mln::Object< mln::p_array< P > >	968
mln::Site_Set< mln::p_array< P > >	1145
mln::Object< mln::p_centered< W > >	968
mln::Site_Set< mln::p_centered< W > >	1145
mln::Object< mln::p_centered_piter< W > >	968
mln::Proxy< mln::p_centered_piter< W > >	1129
mln::Site_Proxy< mln::p_centered_piter< W > >	1143
mln::Site_Iterator< mln::p_centered_piter< W > >	1141
mln::Object< mln::p_complex< D, G > >	968
mln::Site_Set< mln::p_complex< D, G > >	1145
mln::Object< mln::p_double_piter< S, I1, I2 > >	968
mln::Proxy< mln::p_double_piter< S, I1, I2 > >	1129
mln::Site_Proxy< mln::p_double_piter< S, I1, I2 > >	1143
mln::Site_Iterator< mln::p_double_piter< S, I1, I2 > >	1141
mln::Object< mln::p_double_psite< S, Sp > >	968
mln::Proxy< mln::p_double_psite< S, Sp > >	1129
mln::Site_Proxy< mln::p_double_psite< S, Sp > >	1143
mln::Pseudo_Site< mln::p_double_psite< S, Sp > >	1131
mln::Object< mln::p_edges< G, F > >	968
mln::Site_Set< mln::p_edges< G, F > >	1145
mln::Object< mln::p_edges_psite< G, F > >	968
mln::Proxy< mln::p_edges_psite< G, F > >	1129
mln::Site_Proxy< mln::p_edges_psite< G, F > >	1143
mln::Pseudo_Site< mln::p_edges_psite< G, F > >	1131
mln::Object< mln::p_faces< N, D, P > >	968
mln::Site_Set< mln::p_faces< N, D, P > >	1145
mln::Object< mln::p_graph_piter< S, I > >	968
mln::Proxy< mln::p_graph_piter< S, I > >	1129
mln::Site_Proxy< mln::p_graph_piter< S, I > >	1143
mln::Site_Iterator< mln::p_graph_piter< S, I > >	1141
mln::Object< mln::p_if< S, F > >	968
mln::Site_Set< mln::p_if< S, F > >	1145
mln::Object< mln::p_image< I > >	968

mln::Site_Set< mln::p_image< I > >	1145
mln::Object< mln::p_indexed_bkd_piter< S > >	968
mln::Proxy< mln::p_indexed_bkd_piter< S > >	1129
mln::Site_Proxy< mln::p_indexed_bkd_piter< S > >	1143
mln::Site_Iterator< mln::p_indexed_bkd_piter< S > >	1141
mln::Object< mln::p_indexed_fwd_piter< S > >	968
mln::Proxy< mln::p_indexed_fwd_piter< S > >	1129
mln::Site_Proxy< mln::p_indexed_fwd_piter< S > >	1143
mln::Site_Iterator< mln::p_indexed_fwd_piter< S > >	1141
mln::Object< mln::p_indexed_psite< S > >	968
mln::Proxy< mln::p_indexed_psite< S > >	1129
mln::Site_Proxy< mln::p_indexed_psite< S > >	1143
mln::Pseudo_Site< mln::p_indexed_psite< S > >	1131
mln::Object< mln::p_key< K, P > >	968
mln::Site_Set< mln::p_key< K, P > >	1145
mln::Object< mln::p_line2d >	968
mln::Site_Set< mln::p_line2d >	1145
mln::Object< mln::p_mutable_array_of< S > >	968
mln::Site_Set< mln::p_mutable_array_of< S > >	1145
mln::Object< mln::p_n_faces_bkd_piter< D, P > >	968
mln::Proxy< mln::p_n_faces_bkd_piter< D, P > >	1129
mln::Site_Proxy< mln::p_n_faces_bkd_piter< D, P > >	1143
mln::Site_Iterator< mln::p_n_faces_bkd_piter< D, P > >	1141
mln::Object< mln::p_n_faces_fwd_piter< D, P > >	968
mln::Proxy< mln::p_n_faces_fwd_piter< D, P > >	1129
mln::Site_Proxy< mln::p_n_faces_fwd_piter< D, P > >	1143
mln::Site_Iterator< mln::p_n_faces_fwd_piter< D, P > >	1141
mln::Object< mln::p_priority< P, Q > >	968
mln::Site_Set< mln::p_priority< P, Q > >	1145
mln::Object< mln::p_queue< P > >	968
mln::Site_Set< mln::p_queue< P > >	1145
mln::Object< mln::p_queue_fast< P > >	968
mln::Site_Set< mln::p_queue_fast< P > >	1145
mln::Object< mln::p_run< P > >	968
mln::Site_Set< mln::p_run< P > >	1145
mln::Object< mln::p_run_psite< P > >	968
mln::Proxy< mln::p_run_psite< P > >	1129
mln::Site_Proxy< mln::p_run_psite< P > >	1143
mln::Pseudo_Site< mln::p_run_psite< P > >	1131
mln::Object< mln::p_set< P > >	968
mln::Site_Set< mln::p_set< P > >	1145
mln::Object< mln::p_set_of< S > >	968
mln::Site_Set< mln::p_set_of< S > >	1145
mln::Object< mln::p_transformed< S, F > >	968
mln::Site_Set< mln::p_transformed< S, F > >	1145
mln::Object< mln::p_transformed_piter< Pi, S, F > >	968
mln::Proxy< mln::p_transformed_piter< Pi, S, F > >	1129

mln::Site_Proxy< mln::p_transformed_piter< Pi, S, F > >	1143
mln::Site_Iterator< mln::p_transformed_piter< Pi, S, F > >	1141
mln::Object< mln::p_vaccess< V, S > >	968
mln::Site_Set< mln::p_vaccess< V, S > >	1145
mln::Object< mln::p_vertices< G, F > >	968
mln::Site_Set< mln::p_vertices< G, F > >	1145
mln::Object< mln::p_vertices_psite< G, F > >	968
mln::Proxy< mln::p_vertices_psite< G, F > >	1129
mln::Site_Proxy< mln::p_vertices_psite< G, F > >	1143
mln::Pseudo_Site< mln::p_vertices_psite< G, F > >	1131
mln::Object< mln::pixel< I > >	968
mln::Object< mln::plain< I > >	968
mln::Image< mln::plain< I > >	872
mln::Object< mln::point< G, C > >	968
mln::Site< mln::point< G, C > >	1139
mln::Gpoint< mln::point< G, C > >	834
mln::Object< mln::pw::image< F, S > >	968
mln::Image< mln::pw::image< F, S > >	872
mln::Object< mln::ref_data >	968
mln::Function< mln::ref_data >	817
mln::Function_v2v< mln::ref_data >	820
mln::Object< mln::safe_image< I > >	968
mln::Image< mln::safe_image< I > >	872
mln::Object< mln::saturate_rgb8 >	968
mln::Function< mln::saturate_rgb8 >	817
mln::Function_v2v< mln::saturate_rgb8 >	820
mln::Object< mln::slice_image< I > >	968
mln::Image< mln::slice_image< I > >	872
mln::Object< mln::sub_image< I, S > >	968
mln::Image< mln::sub_image< I, S > >	872
mln::Object< mln::sub_image_if< I, S > >	968
mln::Image< mln::sub_image_if< I, S > >	872
mln::Object< mln::thru_image< I, F > >	968
mln::Image< mln::thru_image< I, F > >	872
mln::Object< mln::thrubin_image< I1, I2, F > >	968
mln::Image< mln::thrubin_image< I1, I2, F > >	872
mln::Object< mln::to8bits >	968
mln::Function< mln::to8bits >	817
mln::Function_v2v< mln::to8bits >	820
mln::Object< mln::tfloat01 >	968
mln::Function< mln::tfloat01 >	817
mln::Function_v2v< mln::tfloat01 >	820
mln::Object< mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D > >	968
mln::Iterator< mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D > >	898
mln::Object< mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D > >	968
mln::Iterator< mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D > >	898
mln::Object< mln::topo::adj_higher_face_bkd_iter< D > >	968

mln::Iterator< mln::topo::adj_higher_face_bkd_iter< D > >	898
mln::Object< mln::topo::adj_higher_face_fwd_iter< D > >	968
mln::Iterator< mln::topo::adj_higher_face_fwd_iter< D > >	898
mln::Object< mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D > >	968
mln::Iterator< mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D > >	898
mln::Object< mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D > >	968
mln::Iterator< mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D > >	898
mln::Object< mln::topo::adj_lower_face_bkd_iter< D > >	968
mln::Iterator< mln::topo::adj_lower_face_bkd_iter< D > >	898
mln::Object< mln::topo::adj_lower_face_fwd_iter< D > >	968
mln::Iterator< mln::topo::adj_lower_face_fwd_iter< D > >	898
mln::Object< mln::topo::adj_lower_higher_face_bkd_iter< D > >	968
mln::Iterator< mln::topo::adj_lower_higher_face_bkd_iter< D > >	898
mln::Object< mln::topo::adj_lower_higher_face_fwd_iter< D > >	968
mln::Iterator< mln::topo::adj_lower_higher_face_fwd_iter< D > >	898
mln::Object< mln::topo::adj_m_face_bkd_iter< D > >	968
mln::Iterator< mln::topo::adj_m_face_bkd_iter< D > >	898
mln::Object< mln::topo::adj_m_face_fwd_iter< D > >	968
mln::Iterator< mln::topo::adj_m_face_fwd_iter< D > >	898
mln::Object< mln::topo::center_only_iter< D > >	968
mln::Iterator< mln::topo::center_only_iter< D > >	898
mln::Object< mln::topo::centered_bkd_iter_adapter< D, I > >	968
mln::Iterator< mln::topo::centered_bkd_iter_adapter< D, I > >	898
mln::Object< mln::topo::centered_fwd_iter_adapter< D, I > >	968
mln::Iterator< mln::topo::centered_fwd_iter_adapter< D, I > >	898
mln::Object< mln::topo::face_bkd_iter< D > >	968
mln::Iterator< mln::topo::face_bkd_iter< D > >	898
mln::Object< mln::topo::face_fwd_iter< D > >	968
mln::Iterator< mln::topo::face_fwd_iter< D > >	898
mln::Object< mln::topo::is_n_face< N > >	968
mln::Function< mln::topo::is_n_face< N > >	817
mln::Function_v2v< mln::topo::is_n_face< N > >	820
mln::Function_v2b< mln::topo::is_n_face< N > >	819
mln::Object< mln::topo::is_simple_cell< I > >	968
mln::Function< mln::topo::is_simple_cell< I > >	817
mln::Function_v2v< mln::topo::is_simple_cell< I > >	820
mln::Function_v2b< mln::topo::is_simple_cell< I > >	819
mln::Object< mln::topo::n_face_bkd_iter< D > >	968
mln::Iterator< mln::topo::n_face_bkd_iter< D > >	898
mln::Object< mln::topo::n_face_fwd_iter< D > >	968
mln::Iterator< mln::topo::n_face_fwd_iter< D > >	898
mln::Object< mln::topo::static_n_face_bkd_iter< N, D > >	968
mln::Iterator< mln::topo::static_n_face_bkd_iter< N, D > >	898
mln::Object< mln::topo::static_n_face_fwd_iter< N, D > >	968
mln::Iterator< mln::topo::static_n_face_fwd_iter< N, D > >	898
mln::Object< mln::tr_image< S, I, T > >	968
mln::Image< mln::tr_image< S, I, T > >	872

mln::Object< mln::transformed_image< I, F > >	968
mln::Image< mln::transformed_image< I, F > >	872
mln::Object< mln::unproject_image< I, D, F > >	968
mln::Image< mln::unproject_image< I, D, F > >	872
mln::Object< mln::util::array_bkd_iter< T > >	968
mln::Proxy< mln::util::array_bkd_iter< T > >	1129
mln::Object< mln::util::array_fwd_iter< T > >	968
mln::Proxy< mln::util::array_fwd_iter< T > >	1129
mln::Object< mln::util::couple< T, U > >	968
mln::Object< mln::util::eat >	968
mln::Object< mln::util::fibonacci_heap< P, T > >	968
mln::Object< mln::util::graph >	968
mln::Graph< mln::util::graph >	838
mln::Object< mln::util::ignore >	968
mln::Object< mln::util::line_graph< G > >	968
mln::Graph< mln::util::line_graph< G > >	838
mln::Object< mln::util::multi_site< P > >	968
mln::Object< mln::util::nil >	968
mln::Object< mln::util::object_id< Tag, V > >	968
mln::Value< mln::util::object_id< Tag, V > >	1298
mln::Object< mln::util::ord_pair< T > >	968
mln::Object< mln::util::set< T > >	968
mln::util::set< T >	1274
mln::Object< mln::util::set_bkd_iter< T > >	968
mln::Proxy< mln::util::set_bkd_iter< T > >	1129
mln::Object< mln::util::set_fwd_iter< T > >	968
mln::Proxy< mln::util::set_fwd_iter< T > >	1129
mln::Object< mln::util::site_pair< P > >	968
mln::Object< mln::util::soft_heap< T, R > >	968
mln::Object< mln::util::timer >	968
mln::Proxy< mln::util::timer >	1129
mln::Object< mln::util::vertex< G > >	968
mln::Site< mln::util::vertex< G > >	1139
mln::Object< mln::util::yes >	968
mln::Object< mln::value::float01 >	968
mln::Value< mln::value::float01 >	1298
mln::Object< mln::value::float01_f >	968
mln::Value< mln::value::float01_f >	1298
mln::Object< mln::value::graylevel< n > >	968
mln::Value< mln::value::graylevel< n > >	1298
mln::Object< mln::value::graylevel_f >	968
mln::Value< mln::value::graylevel_f >	1298
mln::Object< mln::value::int_s< n > >	968
mln::Value< mln::value::int_s< n > >	1298
mln::Object< mln::value::int_u< n > >	968
mln::Value< mln::value::int_u< n > >	1298
mln::Object< mln::value::int_u_sat< n > >	968
mln::Value< mln::value::int_u_sat< n > >	1298

mln::Object< mln::value::label< n > >	968
mln::Value< mln::value::label< n > >	1298
mln::Object< mln::value::lut_vec< S, T > >	968
mln::Value_Set< mln::value::lut_vec< S, T > >	1340
mln::Object< mln::value::proxy< I > >	968
mln::Proxy< mln::value::proxy< I > >	1129
mln::Object< mln::value::rgb< n > >	968
mln::Value< mln::value::rgb< n > >	1298
mln::Object< mln::value::shell< F, I > >	968
mln::Proxy< mln::value::shell< F, I > >	1129
mln::Object< mln::value::sign >	968
mln::Value< mln::value::sign >	1298
mln::Object< mln::value::stack_image< n, I > >	968
mln::Image< mln::value::stack_image< n, I > >	872
mln::Object< mln::vertex_image< P, V, G > >	968
mln::Object< mln::violent_cast_image< T, I > >	968
mln::Image< mln::violent_cast_image< T, I > >	872
mln::Object< mln::w_window< D, W > >	968
mln::Weighted_Window< mln::w_window< D, W > >	1351
mln::Object< mln::win::backdiag2d >	968
mln::Window< mln::win::backdiag2d >	1367
mln::Object< mln::win::ball< G, C > >	968
mln::Window< mln::win::ball< G, C > >	1367
mln::Object< mln::win::cube3d >	968
mln::Window< mln::win::cube3d >	1367
mln::Object< mln::win::cuboid3d >	968
mln::Window< mln::win::cuboid3d >	1367
mln::Object< mln::win::diag2d >	968
mln::Window< mln::win::diag2d >	1367
mln::Object< mln::win::line< M, i, C > >	968
mln::Window< mln::win::line< M, i, C > >	1367
mln::Object< mln::win::multiple< W, F > >	968
mln::Window< mln::win::multiple< W, F > >	1367
mln::Object< mln::win::multiple_qiter< W, F > >	968
mln::Proxy< mln::win::multiple_qiter< W, F > >	1129
mln::Site_Proxy< mln::win::multiple_qiter< W, F > >	1143
mln::Site_Iterator< mln::win::multiple_qiter< W, F > >	1141
mln::Object< mln::win::multiple_size< n, W, F > >	968
mln::Window< mln::win::multiple_size< n, W, F > >	1367
mln::Object< mln::win::multiple_size_qiter< n, W, F > >	968
mln::Proxy< mln::win::multiple_size_qiter< n, W, F > >	1129
mln::Site_Proxy< mln::win::multiple_size_qiter< n, W, F > >	1143
mln::Site_Iterator< mln::win::multiple_size_qiter< n, W, F > >	1141
mln::Object< mln::win::octagon2d >	968
mln::Window< mln::win::octagon2d >	1367
mln::Object< mln::win::rectangle2d >	968
mln::Window< mln::win::rectangle2d >	1367

mln::Object< mln::window< D > >	968
mln::Window< mln::window< D > >	1367
mln::Object< mln::world::inter_pixel::dim2::is_dot >	968
mln::Function< mln::world::inter_pixel::dim2::is_dot >	817
mln::Function_v2v< mln::world::inter_pixel::dim2::is_dot >	820
mln::Function_v2b< mln::world::inter_pixel::dim2::is_dot >	819
mln::Object< mln::world::inter_pixel::dim2::is_edge >	968
mln::Function< mln::world::inter_pixel::dim2::is_edge >	817
mln::Function_v2v< mln::world::inter_pixel::dim2::is_edge >	820
mln::Function_v2b< mln::world::inter_pixel::dim2::is_edge >	819
mln::Object< mln::world::inter_pixel::dim2::is_pixel >	968
mln::Function< mln::world::inter_pixel::dim2::is_pixel >	817
mln::Function_v2v< mln::world::inter_pixel::dim2::is_pixel >	820
mln::Function_v2b< mln::world::inter_pixel::dim2::is_pixel >	819
mln::Object< mln::world::inter_pixel::dim2::is_row_odd >	968
mln::Function< mln::world::inter_pixel::dim2::is_row_odd >	817
mln::Function_v2v< mln::world::inter_pixel::dim2::is_row_odd >	820
mln::Function_v2b< mln::world::inter_pixel::dim2::is_row_odd >	819
mln::Object< mln::world::inter_pixel::is_pixel >	968
mln::Function< mln::world::inter_pixel::is_pixel >	817
mln::Function_v2v< mln::world::inter_pixel::is_pixel >	820
mln::Function_v2b< mln::world::inter_pixel::is_pixel >	819
mln::Object< mln::world::inter_pixel::is_separator >	968
mln::Function< mln::world::inter_pixel::is_separator >	817
mln::Function_v2v< mln::world::inter_pixel::is_separator >	820
mln::Function_v2b< mln::world::inter_pixel::is_separator >	819
mln::Object< my::sqrt >	968
mln::Function< my::sqrt >	817
mln::Function_v2v< my::sqrt >	820
mln::Object< my_box2d >	968
mln::Function< my_box2d >	817
mln::Function_v2v< my_box2d >	820
mln::Function_v2b< my_box2d >	819
mln::Object< my_fun< G > >	968
mln::Function< my_fun< G > >	817
mln::Object< my_values_t >	968
mln::Function< my_values_t >	817
mln::Function_v2v< my_values_t >	820
mln::Object< mysqrt >	968
mln::Function< mysqrt >	817
mln::Function_v2v< mysqrt >	820
mln::Object< not_to_remove >	968
mln::Function< not_to_remove >	817
mln::Function_v2v< not_to_remove >	820
mln::Function_v2b< not_to_remove >	819
mln::Object< P >	968
mln::Point_Site< P >	1124
mln::Point< P >	1112

mln::Object< qrde >	968
mln::Function< qrde >	817
mln::Function_v2v< qrde >	820
mln::Object< test< T > >	968
mln::Function< test< T > >	817
mln::Function_v2v< test< T > >	820
mln::Object< to16bits >	968
mln::Function< to16bits >	817
mln::Function_v2v< to16bits >	820
mln::Object< to19bits >	968
mln::Function< to19bits >	817
mln::Function_v2v< to19bits >	820
mln::Object< to23bits >	968
mln::Function< to23bits >	817
mln::Function_v2v< to23bits >	820
mln::Object< to27bits >	968
mln::Function< to27bits >	817
mln::Function_v2v< to27bits >	820
mln::Object< viota_t< S > >	968
mln::Function< viota_t< S > >	817
mln::Function_v2v< viota_t< S > >	820
mln::Object< W >	968
mln::Object< wrap >	968
mln::Function< wrap >	817
mln::Function_v2v< wrap >	820
trait::graph< I >	1374
trait::graph< mln::complex_image< 1, G, V > >	1375
trait::graph< mln::image2d< T > >	1376

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)	507
mln::accu::convolve< T1, T2, R > (Generic convolution accumulator class)	509
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))	511
mln::accu::count_labels< L > (Count the number of different labels in an image)	513
mln::accu::count_value< V > (Count a given value)	515
mln::accu::histo< V > (Generic histogram class over a value set with type V)	517
mln::accu::label_used< L > (References all the labels used)	519
mln::accu::logic::land ("Logical-and" accumulator)	521
mln::accu::logic::land_basic ("Logical-and" accumulator)	523
mln::accu::logic::lor ("Logical-or" accumulator)	525
mln::accu::logic::lor_basic ("Logical-or" accumulator class)	527
mln::accu::maj_h< T > (Compute the majority value)	529
mln::accu::math::count< T > (Generic counter accumulator)	531
mln::accu::math::inf< T > (Generic inf accumulator class)	533
mln::accu::math::sum< T, S > (Generic sum accumulator class)	535
mln::accu::math::sup< T > (Generic sup accumulator class)	537
mln::accu::max_site< I > (Define an accumulator that computes the first site with the maximum value in an image)	539
mln::accu::meta::center (Meta accumulator for center)	541
mln::accu::meta::count_adjacent_vertices (Meta accumulator for count_adjacent_vertices)	542
mln::accu::meta::count_labels (Meta accumulator for count_labels)	543
mln::accu::meta::count_value (FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for count_value)	544
mln::accu::meta::histo (Meta accumulator for histo)	545
mln::accu::meta::label_used (Meta accumulator for label_used)	546
mln::accu::meta::logic::land (Meta accumulator for land)	547
mln::accu::meta::logic::land_basic (Meta accumulator for land_basic)	548
mln::accu::meta::logic::lor (Meta accumulator for lor)	549
mln::accu::meta::logic::lor_basic (Meta accumulator for lor_basic)	550
mln::accu::meta::maj_h (Meta accumulator for maj_h)	551
mln::accu::meta::math::count (Meta accumulator for count)	552
mln::accu::meta::math::inf (Meta accumulator for inf)	553

<code>mln::accu::meta::math::sum</code> (Meta accumulator for <code>sum</code>)	554
<code>mln::accu::meta::math::sup</code> (Meta accumulator for <code>sup</code>)	555
<code>mln::accu::meta::max_site</code> (Meta accumulator for <code>max_site</code>)	556
<code>mln::accu::meta::nil</code> (Meta accumulator for <code>nil</code>)	557
<code>mln::accu::meta::p< mA ></code> (Meta accumulator for <code>p</code>)	558
<code>mln::accu::meta::pair< A1, A2 ></code> (Meta accumulator for <code>pair</code>)	559
<code>mln::accu::meta::rms</code> (Meta accumulator for <code>rms</code>)	560
<code>mln::accu::meta::shape::bbox</code> (Meta accumulator for <code>bbox</code>)	561
<code>mln::accu::meta::shape::height</code> (Meta accumulator for <code>height</code>)	562
<code>mln::accu::meta::shape::volume</code> (Meta accumulator for <code>volume</code>)	563
<code>mln::accu::meta::stat::max</code> (Meta accumulator for <code>max</code>)	564
<code>mln::accu::meta::stat::max_h</code> (Meta accumulator for <code>max</code>)	565
<code>mln::accu::meta::stat::mean</code> (Meta accumulator for <code>mean</code>)	566
<code>mln::accu::meta::stat::median_alt< T ></code> (Meta accumulator for <code>median_alt</code>)	567
<code>mln::accu::meta::stat::median_h</code> (Meta accumulator for <code>median_h</code>)	568
<code>mln::accu::meta::stat::min</code> (Meta accumulator for <code>min</code>)	569
<code>mln::accu::meta::stat::min_h</code> (Meta accumulator for <code>min</code>)	570
<code>mln::accu::meta::stat::rank</code> (Meta accumulator for <code>rank</code>)	571
<code>mln::accu::meta::stat::rank_high_quant</code> (Meta accumulator for <code>rank_high_quant</code>)	572
<code>mln::accu::meta::tuple< n, ></code> (Meta accumulator for <code>tuple</code>)	573
<code>mln::accu::meta::val< mA ></code> (Meta accumulator for <code>val</code>)	574
<code>mln::accu::nil< T ></code> (Define an accumulator that does nothing)	575
<code>mln::accu::p< A ></code> (Generic <code>p</code> of accumulators)	577
<code>mln::accu::pair< A1, A2, T ></code> (Generic <code>pair</code> of accumulators)	579
<code>mln::accu::rms< T, V ></code> (Generic root mean square accumulator class)	581
<code>mln::accu::shape::bbox< P ></code> (Generic bounding <code>box</code> accumulator class)	583
<code>mln::accu::shape::height< I ></code> (Height accumulator)	585
<code>mln::accu::shape::volume< I ></code> (Volume accumulator class)	588
<code>mln::accu::site_set::rectangularity< P ></code> (Compute the <code>rectangularity</code> of a site <code>set</code>)	591
<code>mln::accu::stat::deviation< T, S, M ></code> (Generic standard <code>deviation</code> accumulator class)	593
<code>mln::accu::stat::max< T ></code> (Generic <code>max</code> accumulator class)	595
<code>mln::accu::stat::max_h< V ></code> (Generic <code>max</code> function based on histogram over a <code>value set</code> with type <code>V</code>)	597
<code>mln::accu::stat::mean< T, S, M ></code> (Generic <code>mean</code> accumulator class)	599
<code>mln::accu::stat::median_alt< S ></code> (Generic <code>median_alt</code> function based on histogram over a <code>value set</code> with type <code>S</code>)	601
<code>mln::accu::stat::median_h< V ></code> (Generic <code>median</code> function based on histogram over a <code>value set</code> with type <code>V</code>)	603
<code>mln::accu::stat::meta::deviation</code> (Meta accumulator for <code>deviation</code>)	605
<code>mln::accu::stat::min< T ></code> (Generic <code>min</code> accumulator class)	606
<code>mln::accu::stat::min_h< V ></code> (Generic <code>min</code> function based on histogram over a <code>value set</code> with type <code>V</code>)	608
<code>mln::accu::stat::min_max< V ></code> (Generic <code>min</code> and <code>max</code> accumulator class)	610
<code>mln::accu::stat::rank< T ></code> (Generic <code>rank</code> accumulator class)	612
<code>mln::accu::stat::rank< bool ></code> (Rank accumulator class for Boolean)	614
<code>mln::accu::stat::rank_high_quant< T ></code> (Generic <code>rank</code> accumulator class)	616
<code>mln::accu::stat::var< T ></code> (Var accumulator class)	618
<code>mln::accu::stat::variance< T, S, R ></code> (Variance accumulator class)	621
<code>mln::accu::tuple< A, n, ></code> (Generic <code>tuple</code> of accumulators)	624
<code>mln::accu::val< A ></code> (Generic <code>val</code> of accumulators)	626
<code>mln::Accumulator< E ></code> (Base class for implementation of accumulators)	628
<code>mln::algebra::h_mat< d, T ></code> (N-Dimensional matrix with homogeneous coordinates)	629
<code>mln::algebra::h_vec< d, C ></code> (N-Dimensional vector with homogeneous coordinates)	631
<code>mln::bkd_pixterId< I ></code> (Backward <code>pixel</code> iterator on a 1-D image with <code>border</code>)	633

mln::bkd_paxter2d< I > (Backward pixel iterator on a 2-D image with border)	635
mln::bkd_paxter3d< I > (Backward pixel iterator on a 3-D image with border)	637
mln::box< P > (Generic box class: site set containing points of a regular grid)	639
mln::Box< E > (Base class for implementation classes of boxes)	648
mln::box_runend_piter< P > (A generic backward iterator on points by lines)	653
mln::box_runstart_piter< P > (A generic forward iterator on points by lines)	655
mln::Browsing< E > (Base class for implementation classes that are browsings)	657
mln::canvas::browsing::backdiagonal2d_t (Browsing in a certain direction)	658
mln::canvas::browsing::breadth_first_search_t (Breadth-first search algorithm for graph , on vertices)	659
mln::canvas::browsing::depth_first_search_t (Breadth-first search algorithm for graph , on vertices)	660
mln::canvas::browsing::diagonal2d_t (Browsing in a certain direction)	661
mln::canvas::browsing::dir_struct_elt_incr_update_t (Browsing in a certain direction with a segment)	662
mln::canvas::browsing::directional_t (Browsing in a certain direction)	664
mln::canvas::browsing::fwd_t (Canvas for forward browsing)	666
mln::canvas::browsing::hyper_directional_t (Browsing in a certain direction)	667
mln::canvas::browsing::snake_fwd_t (Browsing in a snake-way, forward)	668
mln::canvas::browsing::snake_generic_t (Multidimensional Browsing in a given-way)	669
mln::canvas::browsing::snake_vert_t (Browsing in a snake-way, forward)	670
mln::canvas::chamfer< F > (Compute chamfer distance)	671
mln::category< R(*)(&A) > (Category declaration for a unary C function)	672
mln::complex_image< D, G, V > (Image based on a complex)	673
mln::complex_neighborhood_bkd_piter< I, G, N > (Backward iterator on complex neighborhood)	676
mln::complex_neighborhood_fwd_piter< I, G, N > (Forward iterator on complex neighborhood)	678
mln::complex_psite< D, G > (Point site associated to a mln::p_complex)	680
mln::complex_window_bkd_piter< I, G, W > (Backward iterator on complex window)	683
mln::complex_window_fwd_piter< I, G, W > (Forward iterator on complex window)	685
mln::decorated_image< I, D > (Image that can have additional features)	687
mln::Delta_Point_Site< E > (FIXME: Doc!)	690
mln::Delta_Point_Site< void > (Delta point site category flag type)	691
mln::doc::Accumulator< E > (Documentation class for mln::Accumulator)	692
mln::doc::Box< E > (Documentation class for mln::Box)	694
mln::doc::Dpoint< E > (Documentation class for mln::Dpoint)	697
mln::doc::Fastest_Image< E > (Documentation class for the concept of images that have the speed property set to "fastest")	699
mln::doc::Generalized_Pixel< E > (Documentation class for mln::Generalized_Pixel)	707
mln::doc::Image< E > (Documentation class for mln::Image)	709
mln::doc::Iterator< E > (Documentation class for mln::Iterator)	715
mln::doc::Neighborhood< E > (Documentation class for mln::Neighborhood)	717
mln::doc::Object< E > (Documentation class for mln::Object)	719
mln::doc::Pixel_Iterator< E > (Documentation class for mln::Iterator)	720
mln::doc::Point_Site< E > (Documentation class for mln::Point_Site)	723
mln::doc::Site_Iterator< E > (Documentation class for mln::Site_Iterator)	726
mln::doc::Site_Set< E > (Documentation class for mln::Site_Set)	728
mln::doc::Value_Iterator< E > (Documentation class for mln::Value_Iterator)	730
mln::doc::Value_Set< E > (Documentation class for mln::Value_Set)	732
mln::doc::Weighted_Window< E > (Documentation class for mln::Weighted_Window)	734
mln::doc::Window< E > (Documentation class for mln::Window)	737
mln::Dpoint< E > (Base class for implementation of delta-point classes)	738
mln::dpoint< G, C > (Generic delta-point class)	739

<code>mln::dpoints_bkd_pixter< I ></code> (A generic backward iterator on the pixels of a dpoint-based <code>window</code> or neighborhood)	744
<code>mln::dpoints_fwd_pixter< I ></code> (A generic forward iterator on the pixels of a dpoint-based <code>window</code> or neighborhood)	747
<code>mln::dpsites_bkd_piter< V ></code> (A generic backward iterator on points of windows and of neighborhoods)	750
<code>mln::dpsites_fwd_piter< V ></code> (A generic forward iterator on points of windows and of neighborhoods)	752
<code>mln::Edge< E ></code> (Edge category flag type)	754
<code>mln::edge_image< P, V, G ></code> (Image based on <code>graph</code> edges)	755
<code>mln::extended< I ></code> (Makes an image become restricted by a <code>point set</code>)	758
<code>mln::extension_fun< I, F ></code> (Extends the domain of an image with a function)	760
<code>mln::extension_ima< I, J ></code> (Extends the domain of an image with an image)	763
<code>mln::extension_val< I ></code> (Extends the domain of an image with a <code>value</code>)	766
<code>mln::faces_psite< N, D, P ></code> (Point site associated to a <code>mln::p_faces</code>)	769
<code>mln::flat_image< T, S ></code> (Image with a single <code>value</code>)	772
<code>mln::fun::from_accu< A ></code> (Wrap an accumulator into a function)	775
<code>mln::fun::p2b::antilogy</code> (A <code>p2b</code> function always returning <code>false</code>)	776
<code>mln::fun::p2b::tautology</code> (A <code>p2b</code> function always returning <code>true</code>)	777
<code>mln::fun::v2b::lnot< V ></code> (Functor computing logical-not on a <code>value</code>)	778
<code>mln::fun::v2b::threshold< V ></code> (Threshold function)	779
<code>mln::fun::v2v::ch_function_value< F, V ></code> (Wrap a function <code>v2v</code> and <code>convert</code> its result to another type)	780
<code>mln::fun::v2v::component< T, i ></code> (Functor that accesses the <code>i</code> -th <code>component</code> of a <code>value</code>)	781
<code>mln::fun::v2v::l1_norm< V, R ></code> (L1-norm)	782
<code>mln::fun::v2v::l2_norm< V, R ></code> (L2-norm)	783
<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)	784
<code>mln::fun::v2v::linfty_norm< V, R ></code> (L-infty norm)	785
<code>mln::fun::v2w2v::cos< V ></code> (Cosinus bijective functor)	786
<code>mln::fun::v2w_w2v::l1_norm< V, R ></code> (L1-norm)	787
<code>mln::fun::v2w_w2v::l2_norm< V, R ></code> (L2-norm)	788
<code>mln::fun::v2w_w2v::linfty_norm< V, R ></code> (L-infty norm)	789
<code>mln::fun::vv2b::eq< L, R ></code> (Functor computing equal between two values)	790
<code>mln::fun::vv2b::ge< L, R ></code> (Functor computing "greater or equal than" between two values)	791
<code>mln::fun::vv2b::gt< L, R ></code> (Functor computing "greater than" between two values)	792
<code>mln::fun::vv2b::implies< L, R ></code> (Functor computing logical-implies between two values)	793
<code>mln::fun::vv2b::le< L, R ></code> (Functor computing "lower or equal than" between two values)	794
<code>mln::fun::vv2b::lt< L, R ></code> (Functor computing "lower than" between two values)	795
<code>mln::fun::vv2v::diff_abs< V ></code> (A functor computing the <code>diff_absimum</code> of two values)	796
<code>mln::fun::vv2v::land< L, R ></code> (Functor computing logical-and between two values)	797
<code>mln::fun::vv2v::land_not< L, R ></code> (Functor computing <code>logical</code> and-not between two values)	798
<code>mln::fun::vv2v::lor< L, R ></code> (Functor computing logical-or between two values)	799
<code>mln::fun::vv2v::lxor< L, R ></code> (Functor computing logical-xor between two values)	800
<code>mln::fun::vv2v::max< V ></code> (A functor computing the maximum of two values)	801
<code>mln::fun::vv2v::min< L, R ></code> (A functor computing the minimum of two values)	802
<code>mln::fun::vv2v::vec< V ></code> (A functor computing the <code>vecimum</code> of two values)	803
<code>mln::fun::x2p::closest_point< P ></code> (FIXME: doxygen + concept checking)	804
<code>mln::fun::x2v::bilinear< I ></code> (Represent a <code>bilinear</code> interolation of values from an underlying image)	805
<code>mln::fun::x2v::trilinear< I ></code> (Represent a <code>trilinear</code> interolation of values from an underlying image)	806
<code>mln::fun::x2x::composed< T2, T1 ></code> (Represent a composition of two transformations)	807
<code>mln::fun::x2x::linear< I ></code> (Represent a <code>linear</code> interolation of values from an underlying image)	808

<code>mln::fun::x2x::rotation< n, C ></code> (Represent a rotation function)	810
<code>mln::fun::x2x::translation< n, C ></code> (Translation function-object)	813
<code>mln::fun_image< F, I ></code> (Image read through a function)	815
<code>mln::Function< E ></code> (Base class for implementation of function-objects)	817
<code>mln::Function< void ></code> (Function category flag type)	818
<code>mln::Function_v2b< E ></code> (Base class for implementation of function-objects from a value to a Boolean)	819
<code>mln::Function_v2v< E ></code> (Base class for implementation of function-objects from value to value)	820
<code>mln::Function_vv2b< E ></code> (Base class for implementation of function-objects from a couple of values to a Boolean)	821
<code>mln::Function_vv2v< E ></code> (Base class for implementation of function-objects from a couple of values to a value)	822
<code>mln::fwd_pixter1d< I ></code> (Forward pixel iterator on a 1-D image with border)	823
<code>mln::fwd_pixter2d< I ></code> (Forward pixel iterator on a 2-D image with border)	825
<code>mln::fwd_pixter3d< I ></code> (Forward pixel iterator on a 3-D image with border)	827
<code>mln::Gdpoint< E ></code> (FIXME: Doc!)	829
<code>mln::Gdpoint< void ></code> (Delta point site category flag type)	830
<code>mln::Generalized_Pixel< E ></code> (Base class for implementation classes that are pixels or that have the behavior of pixels)	831
<code>mln::geom::complex_geometry< D, P ></code> (A functor returning the sites of the faces of a complex where the locations of each 0-face is stored)	832
<code>mln::Gpoint< E ></code> (Base class for implementation of point classes)	834
<code>mln::Graph< E ></code> (Base class for implementation of graph classes)	838
<code>mln::graph::attribute::card_t</code> (Compute the cardinality of every component in a graph)	839
<code>mln::graph::attribute::representative_t</code> (Compute the representative vertex of every component in a graph)	840
<code>mln::graph_elt_mixed_neighborhood< G, S, S2 ></code> (Elementary neighborhood on graph class)	841
<code>mln::graph_elt_mixed_window< G, S, S2 ></code> (Elementary window on graph class)	843
<code>mln::graph_elt_neighborhood< G, S ></code> (Elementary neighborhood on graph class)	847
<code>mln::graph_elt_neighborhood_if< G, S, I ></code> (Elementary neighborhood_if on graph class)	849
<code>mln::graph_elt_window< G, S ></code> (Elementary window on graph class)	851
<code>mln::graph_elt_window_if< G, S, I ></code> (Custom window on graph class)	855
<code>mln::graph_window_base< P, E ></code>	860
<code>mln::graph_window_if_piter< S, W, I ></code> (Forward iterator on line graph window)	862
<code>mln::graph_window_piter< S, W, I ></code> (Forward iterator on line graph window)	864
<code>mln::hexa< I ></code> (Hexagonal image class)	868
<code>mln::histo::array< T ></code> (Generic histogram class over a value set with type T)	871
<code>mln::Image< E ></code> (Base class for implementation of image classes)	872
<code>mln::image1d< T ></code> (Basic 1D image class)	875
<code>mln::image2d< T ></code> (Basic 2D image class)	880
<code>mln::image2d_h< V ></code> (2d image based on an hexagonal mesh)	885
<code>mln::image3d< T ></code> (Basic 3D image class)	888
<code>mln::image_if< I, F ></code> (Image which domain is restricted by a function 'site -> Boolean')	893
<code>mln::interpolated< I, F ></code> (Makes the underlying image being accessed with floating coordinates)	895
<code>mln::io::fld::fld_header</code> (Define the header structure of an AVS field data file)	897
<code>mln::Iterator< E ></code> (Base class for implementation classes that are iterators)	898
<code>mln::labeled_image< I ></code> (Morpher providing an improved interface for labeled image)	900
<code>mln::labeled_image_base< I, E ></code> (Base class Morpher providing an improved interface for labeled image)	904
<code>mln::lazy_image< I, F, B ></code> (Image values are computed on the fly)	907
<code>mln::Literal< E ></code> (Base class for implementation classes of literals)	910
<code>mln::literal::black_t</code> (Type of literal black)	913
<code>mln::literal::blue_t</code> (Type of literal blue)	914
<code>mln::literal::brown_t</code> (Type of literal brown)	915

<code>mln::literal::cyan_t</code> (Type of <code>literal</code> cyan)	916
<code>mln::literal::green_t</code> (Type of <code>literal</code> green)	917
<code>mln::literal::identity_t</code> (Type of <code>literal</code> identity)	918
<code>mln::literal::light_gray_t</code> (Type of <code>literal</code> grays)	919
<code>mln::literal::lime_t</code> (Type of <code>literal</code> lime)	920
<code>mln::literal::magenta_t</code> (Type of <code>literal</code> magenta)	921
<code>mln::literal::max_t</code> (Type of <code>literal</code> max)	922
<code>mln::literal::min_t</code> (Type of <code>literal</code> min)	923
<code>mln::literal::olive_t</code> (Type of <code>literal</code> olive)	924
<code>mln::literal::one_t</code> (Type of <code>literal</code> one)	925
<code>mln::literal::orange_t</code> (Type of <code>literal</code> orange)	926
<code>mln::literal::origin_t</code> (Type of <code>literal</code> origin)	927
<code>mln::literal::pink_t</code> (Type of <code>literal</code> pink)	928
<code>mln::literal::purple_t</code> (Type of <code>literal</code> purple)	929
<code>mln::literal::red_t</code> (Type of <code>literal</code> red)	930
<code>mln::literal::teal_t</code> (Type of <code>literal</code> teal)	931
<code>mln::literal::violet_t</code> (Type of <code>literal</code> violet)	932
<code>mln::literal::white_t</code> (Type of <code>literal</code> white)	933
<code>mln::literal::yellow_t</code> (Type of <code>literal</code> yellow)	934
<code>mln::literal::zero_t</code> (Type of <code>literal</code> zero)	935
<code>mln::Mesh< E ></code> (Base class for implementation classes of meshes)	936
<code>mln::Meta_Accumulator< E ></code> (Base class for implementation of meta accumulators)	937
<code>mln::Meta_Function< E ></code> (Base class for implementation of meta functions)	938
<code>mln::Meta_Function_v2v< E ></code> (Base class for implementation of function-objects from <code>value</code> to <code>value</code>)	939
<code>mln::Meta_Function_vv2v< E ></code> (Base class for implementation of function-objects from <code>value</code> to <code>value</code>)	940
<code>mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 ></code> (Ands type)	941
<code>mln::metal::converts_to< T, U ></code> ("converts-to" check)	942
<code>mln::metal::equal< T1, T2 ></code> (Definition of a static 'equal' <code>test</code>)	943
<code>mln::metal::goes_to< T, U ></code> ("goes-to" check)	944
<code>mln::metal::is< T, U ></code> ("is" check)	945
<code>mln::metal::is_a< T, M ></code> ("is_a" check)	946
<code>mln::metal::is_not< T, U ></code> ("is_not" check)	947
<code>mln::metal::is_not_a< T, M ></code> ("is_not_a" static Boolean expression)	948
<code>mln::mixed_neighb< W ></code> (Adapter class from <code>window</code> to neighborhood)	949
<code>mln::morpho::attribute::card< I ></code> (Cardinality accumulator class)	951
<code>mln::morpho::attribute::count_adjacent_vertices< I ></code> (Count_Adjacent_Vertices accumulator class)	953
<code>mln::morpho::attribute::height< I ></code> (Height accumulator class)	955
<code>mln::morpho::attribute::sharpness< I ></code> (Sharpness accumulator class)	957
<code>mln::morpho::attribute::sum< I, S ></code> (Suminality accumulator class)	960
<code>mln::morpho::attribute::volume< I ></code> (Volume accumulator class)	962
<code>mln::neighb< W ></code> (Adapter class from <code>window</code> to neighborhood)	964
<code>mln::Neighborhood< E ></code> (Base class for implementation classes that are neighborhoods)	966
<code>mln::Neighborhood< void ></code> (<code>Neighborhood</code> category flag type)	967
<code>mln::Object< E ></code> (Base class for almost every class defined in Milena)	968
<code>mln::p2p_image< I, F ></code> (FIXME: Doc!)	969
<code>mln::p_array< P ></code> (Multi-set of sites)	971
<code>mln::p_centered< W ></code> (Site set corresponding to a <code>window</code> centered on a site)	978
<code>mln::p_complex< D, G ></code> (A complex psite <code>set</code> based on the N-faces of a complex of dimension D (a D-complex))	983
<code>mln::p_edges< G, F ></code> (Site set mapping <code>graph</code> edges and image sites)	989

<code>mln::p_faces< N, D, P ></code> (A complex psite <code>set</code> based on a the N-faces of a complex of dimension D (a D-complex))	997
<code>mln::p_graph_piter< S, I ></code> (Generic iterator on <code>point</code> sites of a <code>mln::S</code>)	1003
<code>mln::p_if< S, F ></code> (<code>Site set</code> restricted w.r.t)	1005
<code>mln::p_image< I ></code> (<code>Site set</code> based on an image of Booleans)	1010
<code>mln::p_indexed_bkd_piter< S ></code> (Backward iterator on sites of an indexed site <code>set</code>)	1016
<code>mln::p_indexed_fwd_piter< S ></code> (Forward iterator on sites of an indexed site <code>set</code>)	1018
<code>mln::p_indexed_psite< S ></code> (Psite class for indexed site sets such as <code>p_array</code>)	1020
<code>mln::p_key< K, P ></code> (Priority queue class)	1021
<code>mln::p_line2d</code> (2D discrete line of points)	1028
<code>mln::p_mutable_array_of< S ></code> (<code>P_mutable_array_of</code> is a mutable array of site sets)	1034
<code>mln::p_n_faces_bkd_piter< D, P ></code> (Backward iterator on the n-faces sites of an <code>mln::p-complex<D, P></code>)	1040
<code>mln::p_n_faces_fwd_piter< D, P ></code> (Forward iterator on the n-faces sites of an <code>mln::p-complex<D, P></code>)	1042
<code>mln::p_priority< P, Q ></code> (Priority queue)	1044
<code>mln::p_queue< P ></code> (Queue of sites (based on <code>std::deque</code>))	1052
<code>mln::p_queue_fast< P ></code> (Queue of sites class (based on <code>p_array</code>))	1059
<code>mln::p_run< P ></code> (<code>Point set</code> class in run)	1066
<code>mln::p_set< P ></code> (Mathematical <code>set</code> of sites (based on <code>util::set</code>))	1073
<code>mln::p_set_of< S ></code> (<code>P_set_of</code> is a <code>set</code> of site sets)	1080
<code>mln::p_transformed< S, F ></code> (<code>Site set</code> transformed through a function)	1085
<code>mln::p_transformed_piter< Pi, S, F ></code> (<code>Iterator</code> on <code>p_transformed<S,F></code>)	1090
<code>mln::p_vaccess< V, S ></code> (<code>Site set</code> in which sites are grouped by their associated <code>value</code>)	1092
<code>mln::p_vertices< G, F ></code> (<code>Site set</code> based mapping <code>graph</code> vertices to sites)	1098
<code>mln::pixel< I ></code> (Generic <code>pixel</code> class)	1106
<code>mln::Pixel_Iterator< E ></code> (Base class for the implementation of <code>pixel</code> iterator classes)	1108
<code>mln::plain< I ></code> (Prevents an image from sharing its <code>data</code>)	1110
<code>mln::Point< P ></code> (Base class for implementation of <code>point</code> classes)	1112
<code>mln::point< G, C ></code> (Generic <code>point</code> class)	1115
<code>mln::Point_Site< E ></code> (Base class for implementation classes of the notion of "point site")	1124
<code>mln::Point_Site< void ></code> (<code>Point</code> site category flag type)	1128
<code>mln::Proxy< E ></code> (Base class for implementation classes of the notion of "proxy")	1129
<code>mln::Proxy< void ></code> (<code>Proxy</code> category flag type)	1130
<code>mln::Pseudo_Site< E ></code> (Base class for implementation classes of the notion of "pseudo site")	1131
<code>mln::Pseudo_Site< void ></code> (<code>Pseudo_Site</code> category flag type)	1132
<code>mln::pw::image< F, S ></code> (A generic point-wise <code>image</code> implementation)	1133
<code>mln::registration::closest_point_basic< P ></code> (Closest <code>point</code> functor based on map distance)	1134
<code>mln::registration::closest_point_with_map< P ></code> (Closest <code>point</code> functor based on map distance)	1135
<code>mln::Regular_Grid< E ></code> (Base class for implementation classes of regular grids)	1136
<code>mln::safe_image< I ></code> (Makes an image accessible at undefined location)	1137
<code>mln::select::p_of< P ></code> (Structure <code>p_of</code>)	1138
<code>mln::Site< E ></code> (Base class for classes that are explicitly sites)	1139
<code>mln::Site< void ></code> (<code>Site</code> category flag type)	1140
<code>mln::Site_Iterator< E ></code> (Base class for implementation of classes of iterator on points)	1141
<code>mln::Site_Proxy< E ></code> (Base class for implementation classes of the notion of "site proxy")	1143
<code>mln::Site_Proxy< void ></code> (<code>Site_Proxy</code> category flag type)	1144
<code>mln::Site_Set< E ></code> (Base class for implementation classes of site sets)	1145
<code>mln::Site_Set< void ></code> (<code>Site_Set</code> category flag type)	1149
<code>mln::slice_image< I ></code> (2D image extracted from a slice of a 3D image)	1150
<code>mln::sub_image< I, S ></code> (<code>Image</code> having its domain restricted by a site <code>set</code>)	1152
<code>mln::sub_image_if< I, S ></code> (<code>Image</code> having its domain restricted by a site <code>set</code> and a function)	1154
<code>mln::thru_image< I, F ></code> (Morph image values through a function)	1156
<code>mln::thrubin_image< I1, I2, F ></code> (Morphes values from two images through a binary function)	1157

<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>)	1159
<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>)	1161
<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>)	1163
<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>)	1164
<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>)	1165
<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>)	1167
<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>)	1169
<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>)	1170
<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>)	1171
<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>)	1172
<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 <code>complex</code>)	1173
<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 <code>complex</code>)	1175
<code>mln::topo::algebraic_face< D ></code> (Algebraic <code>face</code> handle in a <code>complex</code> ; the <code>face</code> dimension is dynamic)	1177
<code>mln::topo::algebraic_n_face< N, D ></code> (Algebraic N-face handle in a <code>complex</code>)	1182
<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>)	1186
<code>mln::topo::centered_bkd_iter_adapter< D, I ></code> (Forward <code>complex</code> relative iterator adapters adding the central (reference) <code>point</code> to the <code>set</code> of iterated faces)	1188
<code>mln::topo::centered_fwd_iter_adapter< D, I ></code> (Backward <code>complex</code> relative iterator adapters adding the central (reference) <code>point</code> to the <code>set</code> of iterated faces)	1189
<code>mln::topo::complex< D ></code> (General <code>complex</code> of dimension D)	1190
<code>mln::topo::face< D ></code> (Face handle in a <code>complex</code> ; the <code>face</code> dimension is dynamic)	1193
<code>mln::topo::face_bkd_iter< D ></code> (Backward iterator on all the faces of an <code>mln::complex<D></code>)	1197
<code>mln::topo::face_fwd_iter< D ></code> (Forward iterator on all the faces of an <code>mln::complex<D></code>)	1199
<code>mln::topo::is_n_face< N ></code> (A functor testing wheter a <code>mln::complex_psite</code> is an N-face)	1201
<code>mln::topo::is_simple_cell< I ></code> (A predicate for the simplicity of a <code>point</code> based on the collapse property of the attachment)	1202
<code>mln::topo::n_face< N, D ></code> (N-face handle in a <code>complex</code>)	1204
<code>mln::topo::n_face_bkd_iter< D ></code> (Backward iterator on all the faces of an <code>mln::complex<D></code>)	1208
<code>mln::topo::n_face_fwd_iter< D ></code> (Forward iterator on all the faces of an <code>mln::complex<D></code>)	1210
<code>mln::topo::n_faces_set< N, D ></code> (Set of <code>face</code> handles of dimension N)	1212
<code>mln::topo::static_n_face_bkd_iter< N, D ></code> (Backward iterator on all the N-faces of a <code>mln::complex<D></code>)	1214
<code>mln::topo::static_n_face_fwd_iter< N, D ></code> (Forward iterator on all the N-faces of a <code>mln::complex<D></code>)	1216
<code>mln::tr_image< S, I, T ></code> (Transform an image by a given transformation)	1218
<code>mln::transformed_image< I, F ></code> (Image having its domain restricted by a site <code>set</code>)	1221

<code>mln::unproject_image< I, D, F ></code> (Un-projects an image)	1223
<code>mln::util::adjacency_matrix< V ></code> (A class of adjacency matrix)	1225
<code>mln::util::array< T ></code> (A dynamic <code>array</code> class)	1226
<code>mln::util::branch< T ></code> (Class of generic <code>branch</code>)	1232
<code>mln::util::branch_iter< T ></code> (Basic 2D image class)	1234
<code>mln::util::branch_iter_ind< T ></code> (Basic 2D image class)	1236
<code>mln::util::couple< T, U ></code> (Definition of a <code>couple</code>)	1238
<code>mln::util::eat</code> (Eat structure)	1240
<code>mln::util::edge< G ></code> (Edge of a <code>graph</code> <code>G</code>)	1241
<code>mln::util::fibonacci_heap< P, T ></code> (Fibonacci heap)	1245
<code>mln::util::graph</code> (Undirected <code>graph</code>)	1248
<code>mln::util::greater_point< I ></code> (A “greater than” functor comparing points w.r.t)	1255
<code>mln::util::greater_psite< I ></code> (A “greater than” functor comparing psites w.r.t)	1256
<code>mln::util::head< T, R ></code> (Top structure of the soft heap)	1257
<code>mln::util::ignore</code> (Ignore structure)	1258
<code>mln::util::ilcell< T ></code> (Element of an item list. Store the <code>data</code> (key) used in <code>soft_heap</code>)	1259
<code>mln::util::line_graph< G ></code> (Undirected line <code>graph</code> of a <code>graph</code> of type <code>G</code>)	1260
<code>mln::util::nil</code> (Nil structure)	1266
<code>mln::util::node< T, R ></code> (Meta-data of an element in the heap)	1267
<code>mln::util::object_id< Tag, V ></code> (Base class of an object id)	1268
<code>mln::util::ord< T ></code> (Function-object that defines an ordering between objects with type <code>T</code> : <i>lhs</i> <i>R rhs</i>)	1269
<code>mln::util::ord_pair< T ></code> (Ordered pair structure s.a)	1270
<code>mln::util::pix< I ></code> (Structure <code>pix</code>)	1272
<code>mln::util::set< T ></code> (An "efficient" mathematical <code>set</code> class)	1274
<code>mln::util::site_pair< P ></code> (A pair of sites)	1280
<code>mln::util::soft_heap< T, R ></code> (Soft heap)	1281
<code>mln::util::timer</code> (Timer structure)	1284
<code>mln::util::tracked_ptr< T ></code> (Smart pointer for shared <code>data</code> with tracking)	1285
<code>mln::util::tree< T ></code> (Class of generic <code>tree</code>)	1287
<code>mln::util::tree_node< T ></code> (Class of generic <code>tree_node</code> for <code>tree</code>)	1289
<code>mln::util::vertex< G ></code> (Vertex of a <code>graph</code> <code>G</code>)	1293
<code>mln::util::yes</code> (Object that always says "yes")	1297
<code>mln::Value< E ></code> (Base class for implementation classes of values)	1298
<code>mln::value::float01</code> (Class for floating values restricted to the interval [0)	1299
<code>mln::value::float01_f</code> (Class for floating values restricted to the interval [0..1])	1302
<code>mln::value::graylevel< n ></code> (General gray-level class on n bits)	1304
<code>mln::value::graylevel_f</code> (General gray-level class on n bits)	1307
<code>mln::value::int_s< n ></code> (Signed integer <code>value</code> class)	1310
<code>mln::value::int_u< n ></code> (Unsigned integer <code>value</code> class)	1312
<code>mln::value::int_u_sat< n ></code> (Unsigned integer <code>value</code> class with saturation behavior)	1314
<code>mln::value::Integer< E ></code> (Concept of integer)	1316
<code>mln::value::Integer< void ></code> (Category flag type)	1317
<code>mln::value::label< n ></code> (Label <code>value</code> class)	1318
<code>mln::value::lut_vec< S, T ></code> (Class that defines FIXME)	1321
<code>mln::value::proxy< I ></code> (Generic <code>proxy</code> class for an image <code>pixel value</code>)	1324
<code>mln::value::rgb< n ></code> (Color class for red-green-blue where every component is n-bit encoded)	1327
<code>mln::value::set< T ></code> (Class that defines the <code>set</code> of values of type <code>T</code>)	1329
<code>mln::value::sign</code> (Value type composed by the <code>set</code> (-1, 0, 1) <code>sign value</code> type is a subset of the int value type)	1330
<code>mln::value::stack_image< n, I ></code> (Stack image class)	1332
<code>mln::value::super_value< sign ></code> (Specializations:)	1335
<code>mln::value::value_array< T, V ></code> (Generic array class over indexed by a <code>value set</code> with type <code>T</code>)	1336
<code>mln::Value_Iterator< E ></code> (Base class for implementation of classes of iterator on values)	1338

<code>mln::Value_Set< E ></code> (Base class for implementation classes of sets of values)	1340
<code>mln::Vertex< E ></code> (<code>Vertex</code> category flag type)	1341
<code>mln::vertex_image< P, V, G ></code> (<code>Image</code> based on <code>graph</code> vertices)	1342
<code>mln::violent_cast_image< T, I ></code> (Violently cast image values to a given type)	1345
<code>mln::w_window< D, W ></code> (Generic <code>w_window</code> class)	1347
<code>mln::Weighted_Window< E ></code> (Base class for implementation classes that are weighted_ windows)	1351
<code>mln::win::backdiag2d</code> (Diagonal <code>line window</code> defined on the 2D square <code>grid</code>)	1352
<code>mln::win::ball< G, C ></code> (Generic <code>ball window</code> defined on a given <code>grid</code>)	1353
<code>mln::win::cube3d</code> (Cube <code>window</code> defined on the 3D <code>grid</code>)	1354
<code>mln::win::cuboid3d</code> (Cuboid defined on the 3-D square <code>grid</code>)	1356
<code>mln::win::diag2d</code> (Diagonal <code>line window</code> defined on the 2D square <code>grid</code>)	1358
<code>mln::win::line< M, i, C ></code> (Generic <code>line window</code> defined on a given <code>grid</code> in the given dimension)	1359
<code>mln::win::multiple< W, F ></code> (Multiple <code>window</code>)	1361
<code>mln::win::multiple_size< n, W, F ></code> (Definition of a multiple-size <code>window</code>)	1362
<code>mln::win::octagon2d</code> (Octagon <code>window</code> defined on the 2D square <code>grid</code>)	1363
<code>mln::win::rectangle2d</code> (Rectangular <code>window</code> defined on the 2D square <code>grid</code>)	1365
<code>mln::Window< E ></code> (Base class for implementation classes that are windows)	1367
<code>mln::window< D ></code> (Generic <code>window</code> class)	1368
<code>mln::world::inter_pixel::is_separator</code> (Functor returning whether a site is a separator in an inter- pixel image)	1373
<code>trait::graph< I ></code> (Graph traits)	1374
<code>trait::graph< mln::complex_image< I, G, V >></code> (Graph traits for 1-complexes images)	1375
<code>trait::graph< mln::image2d< T >></code> (Graph traits for <code>mln::image2d</code>)	1376

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 >`
*Count 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::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::image1d< 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.

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::thru_bin_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.
- struct `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::transformed_image< I, F >`
[Image](#) having its domain restricted by a [site set](#).
- 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](#)
Miscalleneous 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.

8.15 Canvas

All canvas.

8.16 Functions

All predefined functions.

Namespaces

- namespace `mln::fun::i2v`
Namespace of integer-to-value functions.
- 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*.*

Classes

- struct `mln::Function< E >`
Base class for implementation of function-objects.
- 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*.*

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.

8.18.3 Function Documentation

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

2-connectivity neighborhood on the 1D [grid](#).

○ × ○

Returns:

A `neighb1d`.

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

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.

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.

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

Horizontal 2-connectivity neighborhood on the 2D [grid](#).

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

Returns:

A `neighb2d`.

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

4-connectivity neighborhood on the 2D [grid](#).

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

Returns:

A `neighb2d`.

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

8-connectivity neighborhood on the 2D [grid](#).

```
o o o  
o x o  
o o o
```

Returns:

A `neighb2d`.

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::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.

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`.

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`.

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

8.20.3.3 const neighb3d & mln::c4_3d () [inline]

4-connectivity neighborhood on the 3D [grid](#).

```

  . . .
  . . .
  . . .

  . o .
  o x o
  . o .

  . . .
  . . .
  . . .

```


Returns:

A `neighb3d`.

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

8.20.3.4 const neighb3d & mln::c6 () [inline]

6-connectivity neighborhood on the 3D [grid](#).

```

. . .
. o .
. . .

. o .
o x o
. o .

. . .
. o .
. . .

```

Returns:

A `neighb3d`.

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

Referenced by `mln::c18()`.

8.20.3.5 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`.

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

- struct `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 a 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.

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.

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](#).

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.

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.

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.

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.

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`.

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

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

8.31.2.2 typedef `window<mln::dpoint3d> mln::window3d`

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

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`.

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`.

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.

8.35 $v2w_w2v$ functions

All bijective function.

8.36 vv2b functions

All functions mapping two values to a [logical value](#).

Chapter 9

Namespace Documentation

9.1 mln Namespace Reference

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

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](#).
- struct [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_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 [imageId](#)
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, P>`.
- class [p_n_faces_fwd_piter](#)
Forward iterator on the n-faces sites of an `mln::p_complex<D, P>`.
- 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.
- struct [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 [transformed_image](#)
Image having its domain restricted by a site [set](#).
- 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.

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](#).

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](#)< const I, F > [apply_p2p](#) (const [Image](#)< I > &ima, const [Function_v2v](#)< F > &f)
FIXME: Doc!
- template<typename I, typename F>
[p2p_image](#)< I, F > [apply_p2p](#) ([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 [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>`
`extension_val< const I > extend (const Image< I > &ima, const typename I::value &val)`
Routines for domain extension with a value.
- `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, typename F>`
`extension_fun< const I, F > extend (const Image< I > &ima, const Function_v2v< F > &fun)`
Routines for domain extension with a function.
- `bool implies (bool lexpr, bool rexpr)`
Implication.
- `template<typename I, typename J>`
`void initialize (Image< I > &target, const Image< J > &model)`
- `template<typename I, typename N>`
`bool is_simple_2d (const Image< I > &ima, const Neighborhood< N > &nbh, const typename I::psite &p)`
Test if a point is simple or not.
- `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_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_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_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_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_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_neighborhood, complex_lower_window)

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

- [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](#) (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_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_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_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_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_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.

- [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_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_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_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_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.
- `template<typename W1, typename W2>`
`mln_regular` (W1) `operator-(const Window< W1 > &win1`
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`
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 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 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 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_bkd_piter< I, G, W > &p)
Print an [mln::complex_window_bkd_piter](#).
- `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 N>`
`std::ostream & operator<<` (std::ostream &ostr, const complex_neighborhood_bkd_piter< I, G, N > &p)
Print an [mln::complex_neighborhood_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 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<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<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<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 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<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<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_edges< G, F > &lhs, const p_edges< G, F > &rhs)`
Comparison between two `mln::p_edges`'s.
- `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<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 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 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 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 an edge image from a `fun::i2v::array` and a `p_edges`.
- `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 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 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<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)
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?
- 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 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<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`.

Variables

- const `dpoint1d before` = `dpoint1d`(-1)
Definition of a shortcut for delta `point` in 1d.

- const `dpoint3d sagittal_dec` = `dpoint3d`(0, 0, -1)

Definition of a shortcut for delta [point](#) in 3d.

- const [dpoint2d](#) up = [dpoint2d](#)(-1, 0)

Definition of a shortcut for delta [point](#) in 2d.

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.

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 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 hierachical 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.

Hierarchical queues are often used with connected operators (P. Salemebier'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.

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.

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`.

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](#).

9.1.2.5 typedef box<point2d_h> mln::box2d_h

FIXME.

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](#).

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).

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).

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.

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.

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.

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.

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.

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.

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.

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.

9.1.2.17 `typedef p_run<point2d> mln::p_run2d`

Type alias for a run of 2d points.

9.1.2.18 `typedef p_set_of<p_run2d> mln::p_runs2d`

Type alias for a [set](#) of runs of 2d points.

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.

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.

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.

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.

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.

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.

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.

**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.

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).

**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.

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

2D vector with double coordinates.

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

2D vector with float coordinates.

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

3D vector with double coordinates.

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

3D vector with float coordinates.

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.

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.

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.

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.

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.

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.

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.

9.1.3.2 `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!

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

FIXME: Doc!

Referenced by `mln::debug::slices_2d()`.

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:

- ← *f* The second transformation.
- ← *g* The first transformation.

Returns:

The composed transformation `fog`.

Referenced by `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:

- ← *model* The image to be duplicated.

Returns:

The duplicate.

Precondition:

`model.is_valid`

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

Referenced by `mln::registration::icp()`, `mln::plain< I >::operator I()`, `mln::geom::impl::seeds2tiling()`, `mln::geom::impl::seeds2tiling_roundness()`, and `mln::labeling::superpose()`.

9.1.3.6 `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`.

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)` [inline]

Routines for domain `extension` with an image.

9.1.3.8 `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.

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

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

Implication.

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:

- ↔ *target* The image to be initialized.
- ← *model* The image to provide `data` for the initialization.

Precondition:

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

Referenced by `duplicate()`, `mln::labeling::fill_holes()`, `mln::morpho::tree::filter::filter()`, `mln::linear::gaussian()`, `mln::linear::gaussian_1st_derivative()`, `mln::linear::gaussian_2nd_derivative()`, `mln::morpho::impl::generic::hit_or_miss()`, `mln::graph::labeling()`, `mln::io::magick::load()`, `mln::io::dicom::load()`, `make_debug_graph_image()`, `mln::morpho::tree::filter::max()`, `mln::data::impl::generic::median()`, `mln::morpho::meyer_wst()`, `mln::morpho::tree::filter::min()`, `mln::arith::min()`, `mln::arith::minus()`, `mln::arith::plus()`, `mln::morpho::impl::generic::rank_filter()`, `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 I, typename N> bool mln::is_simple_2d (const Image< I > & ima, const Neighborhood< N > & nbh, const typename I::psite & p)` [inline]

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. | | |

9.1.3.12 `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`.

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

9.1.3.13 `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`.

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()`.

9.1.3.14 `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).

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

9.1.3.15 `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.

9.1.3.16 `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.

9.1.3.17 `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.

9.1.3.18 `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.

9.1.3.19 `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.

9.1.3.20 `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.

9.1.3.21 `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.

9.1.3.22 `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.

9.1.3.23 `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_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.

9.1.3.25 `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.

9.1.3.26 `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.

9.1.3.27 `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.

9.1.3.28 `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.

9.1.3.29 `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.

9.1.3.30 `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.

9.1.3.31 `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.

9.1.3.32 `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.

9.1.3.33 `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`.

9.1.3.34 `template<typename O1, typename O2> mln::mln_trait_op_geq (O1, O2) [inline]`

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`.

9.1.3.35 `template<typename O1, typename O2> mln::mln_trait_op_greater (O1, O2) const [inline]`

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](#).

9.1.3.36 `template<typename O1, typename O2> mln::mln_trait_op_leq (O1, O2) [inline]`

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.

9.1.3.37 `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](#).

9.1.3.38 `template<unsigned D, typename G> bool mln::operator!=(const complex_psite< D, G > & lhs, const complex_psite< D, G > & rhs) [inline]`

Is *lhs* not equal to *rhs*?

Precondition:

Arguments *lhs* and *rhs* must belong to the same [mln::p_complex](#).

References `mln::complex_psite< D, G >::face()`, and `mln::complex_psite< D, G >::site_set()`.

9.1.3.39 `template<unsigned N, unsigned D, typename P> bool mln::operator!= (const faces_psite< N, D, P > & lhs, const faces_psite< N, D, P > & rhs) [inline]`

Is *lhs* equal to *rhs*?

Precondition:

Arguments *lhs* and *rhs* must belong to the same `mln::complex`.

References `mln::faces_psite< N, D, P >::face()`, and `mln::faces_psite< N, D, P >::site_set()`.

9.1.3.40 `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*.

9.1.3.41 `template<typename S> S & mln::operator++ (value::Scalar< S > & rhs) [inline]`

Pre-incrementation for any scalar type.

References `mln::literal::one`.

9.1.3.42 `template<typename N1, typename N2> N2 neighb< typename N1::window::regular > mln::operator- (const Neighborhood< N1 > & nbh1, const Neighborhood< N2 > & nbh2) [inline]`

Set difference between a couple of neighborhoods *nbh1* and *nbh2*.

It just calls `mln::win::diff`.

References `mln::win::diff()`.

9.1.3.43 `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:

← *p* A [grid point](#).

← *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](#)

9.1.3.44 `template<typename S> S & mln::operator- (value::Scalar< S > & rhs)` [inline]

Pre-decrementation for any scalar type.

References `mln::literal::one`.

9.1.3.45 `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:

← *lhs* A first image.

← *rhs* A second image.

Precondition:

`lhs.domain == rhs.domain`

References `mln::test::predicate()`.

9.1.3.46 `template<unsigned D, typename G> bool mln::operator< (const complex_psite< D, G > & lhs, const complex_psite< D, G > & rhs)` [inline]

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](#).

9.1.3.47 `template<unsigned N, unsigned D, typename P> bool mln::operator< (const faces_psite< N, D, P > & lhs, const faces_psite< N, D, P > & rhs)` [inline]

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`.

9.1.3.48 `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](#).

9.1.3.49 `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](#).

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](#).

9.1.3.51 `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](#).

9.1.3.52 `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:

← *lhs* A first image.

← *rhs* A second image.

Precondition:

`lhs.domain == rhs.domain`

References `mln::test::predicate()`.

9.1.3.53 `template<typename G, typename F> bool mln::operator<= (const p_vertices< G, F > & lhs, const p_vertices< G, F > & rhs)` `[inline]`

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.

9.1.3.54 `template<unsigned N, unsigned D, typename P> bool mln::operator<= (const p_faces< N, D, P > & lhs, const p_faces< N, D, P > & rhs)` `[inline]`

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.

9.1.3.55 `template<typename G, typename F> bool mln::operator<= (const p_edges< G, F > & lhs, const p_edges< G, F > & rhs)` `[inline]`

Inclusion of a [mln::p_edges](#) in another one.

9.1.3.56 `template<unsigned D, typename G> bool mln::operator<= (const p_complex< D, G > & lhs, const p_complex< D, G > & rhs) [inline]`

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.

9.1.3.57 `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:

← `lhs` A first image.

← `rhs` A second image.

Precondition:

`lhs.domain == rhs.domain`

References `mln::test::predicate()`.

9.1.3.58 `template<typename G, typename F> bool mln::operator== (const p_vertices< G, F > & lhs, const p_vertices< G, F > & rhs) [inline]`

Comparison between two [mln::p_vertices](#)'s.

Two [mln::p_vertices](#)'s are considered equal if they share the same [graph](#).

References `mln::p_vertices< G, F >::graph()`.

9.1.3.59 `template<unsigned N, unsigned D, typename P> bool mln::operator== (const p_faces< N, D, P > & lhs, const p_faces< N, D, P > & rhs) [inline]`

Comparison between two [mln::p_faces](#)'s.

Two [mln::p_faces](#)'s are considered equal if they share the same complex.

References `mln::p_faces< N, D, P >::cplx()`.

9.1.3.60 `template<typename G, typename F> bool mln::operator== (const p_edges< G, F > & lhs, const p_edges< G, F > & rhs) [inline]`

Comparison between two [mln::p_edges](#)'s.

Two [mln::p_edges](#)'s are considered equal if they share the same [graph](#).

References `mln::p_edges< G, F >::graph()`.

9.1.3.61 `template<unsigned D, typename G> bool mln::operator== (const p_complex< D, G > & lhs, const p_complex< D, G > & rhs) [inline]`

Comparison between two [mln::p_complex](#)'s.

Two `mln::p_complex`'s are considered equal if they share the same complex.

References `mln::p_complex< D, G >::cplx()`.

9.1.3.62 `template<unsigned D, typename G> bool mln::operator==(const complex_psite< D, G > & lhs, const complex_psite< D, G > & rhs) [inline]`

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`.

References `mln::complex_psite< D, G >::face()`, and `mln::complex_psite< D, G >::site_set()`.

9.1.3.63 `template<unsigned N, unsigned D, typename P> bool mln::operator==(const faces_psite< N, D, P > & lhs, const faces_psite< N, D, P > & rhs) [inline]`

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`.

References `mln::faces_psite< N, D, P >::face()`, and `mln::faces_psite< N, D, P >::site_set()`.

9.1.3.64 `template<typename F, typename S> pw::image< F, S > mln::operator|(const Function_v2v< F > & f, const Site_Set< S > & ps) [inline]`

Construct an image from a function and a site `set`.

`image = function | site_set`.

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) [inline]`

Restrict a site `set` *s* to points that verify *f*.

Parameters:

← *s* A site `set`.

← *f* A function from `point` to Boolean.

Returns:

A subset of points.

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) [inline]`

Construct a vertex image from a `fun::i2v::array` and a `p_vertices`.

`image = fun::i2v::array | p_vertices.`

9.1.3.67 `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) [inline]`

Construct a edge image from a `fun::i2v::array` and a `p_edges`.

`image = fun::i2v::array | p_edges.`

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) [inline]`

`ima | f` creates an `image_if` with the image `ima` and the function `f`.

9.1.3.69 `template<typename I, typename F> image_if< I, F > mln::operator| (Image< I > & ima, const Function_v2b< F > & f) [inline]`

`ima | f` creates an `image_if` with the image `ima` and the function `f`.

9.1.3.70 `template<typename I> const internal::primary_type< I >::ret & mln::primary (const Image< I > & input) [inline]`

FIXME: Doc!

Referenced by `mln::border::resize()`.

9.1.3.71 `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:

- ← `s` A site `set`.
- ← `f` A function from site to site.

Returns:

The transformed site `set`.

9.1.4 Variable Documentation

9.1.4.1 `const dpoint1d mln::before = dpoint1d(-1)`

Definition of a shortcut for delta `point` in 1d.

9.1.4.2 `const dpoint3d mln::sagittal_dec = dpoint3d(0, 0, -1)`

Definition of a shortcut for delta [point](#) in 3d.

9.1.4.3 `const dpoint2d mln::up = dpoint2d(-1, 0)`

Definition of a shortcut for delta [point](#) in 2d.

9.2 mln::accu Namespace Reference

Namespace of 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](#)
Count a given [value](#).
- struct [histo](#)
Generic histogram class over a [value set](#) with type \mathbb{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.

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.*

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:

- ← *input* The input `image`.
- ← *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()`.

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) [inline]`

Line an accumulator onto the `pixel` values of the `image` input.

Parameters:

- ← *input* The input `image`.
- ← *p_start* The starting site of the line.
- ← *len* The line length.
- ← *half_length* The half length of the line.
- ↔ *output* The resulting `image`.

This routine runs:

```
tmp = a
tmp.init()
accu::take(input, tmp)
return tmp.to_result()
```

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:

- ← *input* The input `image`.
- ← *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:

← *input* The input `image`.

↔ *a* The accumulator.

This routine runs:

for all `p` of `input`, `a.take(pix(input, p))`

Warning:

This routine does not perform `a.init()`.

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 [land_basic](#)
Meta accumulator for [land_basic](#).
- 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 [max](#)
Generic [max](#) accumulator class.
- struct [max_h](#)
Generic [max](#) function based on histogram over a [value set](#) with type \mathbb{V} .
- struct [mean](#)
Generic [mean](#) accumulator class.
- struct [median_alt](#)
Generic [median_alt](#) function based on histogram over a [value set](#) with type \mathbb{S} .
- struct [median_h](#)
Generic median function based on histogram over a [value set](#) with type \mathbb{V} .
- struct [min](#)
Generic [min](#) accumulator class.
- struct [min_h](#)
Generic [min](#) function based on histogram over a [value set](#) with type \mathbb{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.

9.12.1 Detailed Description

Namespace of statistical accumulators.

9.13 mln::algebra Namespace Reference

Namespace of algebraic structure.

Classes

- struct [h_mat](#)
N-Dimensional matrix with homogeneous coordinates.
- struct [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 `ldltdc` routine.

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](#).

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).

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).

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, 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 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 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>`
`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>`
`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 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, 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 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>`
`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:

- ← *lhs* First operand image.
- ← *rhs* Second operand image.

Returns:

The result image.

Precondition:

`lhs.domain == rhs.domain`

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:

- ← *lhs* First operand image.
- ← *rhs* Second operand image.
- *output* The result image.

Precondition:

`output.domain == lhs.domain == rhs.domain`

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:

- ← *input* The image.
- ← *val* The `value`.
- *output* The result image.

Precondition:

```
output.domain == input.domain
```

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:

- ← *lhs* First operand image (subject to division).
- ↔ *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
```

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:

- ← *lhs* First operand image.
- ← *rhs* Second operand image.

Returns:

The result image.

Precondition:

```
lhs.domain == rhs.domain
```

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:

- ↔ *lhs* First operand image.
- ← *rhs* Second operand image.

Precondition:

```
rhs.domain == lhs.domain
```

9.14.2.7 `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:

- ↔ *lhs* First operand image.
- ← *rhs* Second operand image.
- ← *f* [Function](#).

Returns:

The result image.

Precondition:

```
lhs.domain == rhs.domain
```

References `mln::initialize()`.

9.14.2.8 `template<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:

- ↔ *lhs* First operand image.
- ← *rhs* Second operand image.

Returns:

The result image.

Precondition:

```
lhs.domain == rhs.domain
```

Parameters:

- ← *lhs* First operand image.
- ← *rhs* Second operand image.

Returns:

The result image.

The free parameter V sets the destination [value](#) type.

Precondition:

```
lhs.domain == rhs.domain
```

References mln::initialize().

9.14.2.9 `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:

- ← *input* The image.
- ← *val* The [value](#).
- ← *f* [Function](#).

Returns:

The result image.

Precondition:

```
input.is_valid
```

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:

- ← *input* The image.
- ← *val* The [value](#).

Returns:

The result image.

Precondition:

`input.is_valid`

9.14.2.11 `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:

↔ *input* The image.

← *val* The [value](#).

Precondition:

`input.is_valid`

References `minus_cst_inplace()`, and `minus_inplace()`.

Referenced by `minus_cst_inplace()`.

9.14.2.12 `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:

↔ *lhs* First operand image (subject to addition).

← *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`

References `minus_inplace()`.

Referenced by `minus_cst_inplace()`, and `minus_inplace()`.

9.14.2.13 `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:

- ← *lhs* First operand image.
- ← *rhs* Second operand image.
- ← *f* [Function](#).

Returns:

The result image.

Precondition:

```
lhs.domain == rhs.domain
```

References `mln::initialize()`.

9.14.2.14 `template<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:

- ← *lhs* First operand image.
- ← *rhs* Second operand image.

Returns:

The result image.

Precondition:

```
lhs.domain == rhs.domain
```

Parameters:

- ← *lhs* First operand image.
- ← *rhs* Second operand image.

Returns:

The result image.

The free parameter `V` sets the destination [value](#) type.

Precondition:

```
lhs.domain == rhs.domain
```

References `mln::initialize()`.

Referenced by `mln::morpho::contrast()`.

9.14.2.15 `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:

← *input* The image.

← *val* The `value`.

← *f* Function.

Returns:

The result image.

Precondition:

`input.is_valid`

9.14.2.16 `template<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:

← *input* The image.

← *val* The `value`.

Returns:

The result image.

Precondition:

`input.is_valid`

9.14.2.17 `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:

↔ *input* The image.

← *val* The `value`.

Precondition:

`input.is_valid`

References `plus_cst_inplace()`, and `plus_inplace()`.

Referenced by `plus_cst_inplace()`.

9.14.2.18 `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:

- ↔ *lhs* First operand image (subject to addition).
- ← *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
```

Referenced by `plus_cst_inplace()`.

9.14.2.19 `template<typename I> mln::trait::concrete< I >::ret mln::arith::revert (const Image< I > & input) [inline]`

Point-wise reversion of image `input`.

Parameters:

- ← *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))`

References `mln::initialize()`.

9.14.2.20 `template<typename I> void mln::arith::revert_inplace (Image< I > & input) [inline]`

Point-wise in-place reversion of image `input`.

Parameters:

- ↔ *input* The target image.

Precondition:

```
input.is_valid
```

It performs:

for all p of `input.domain`

```
input(p) = min + (max - input(p))
```

9.14.2.21 `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:

← *lhs* First operand image.

← *rhs* Second operand image.

→ *output* The result image.

Precondition:

```
output.domain == lhs.domain == rhs.domain
```

9.14.2.22 `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:

← *input* The image.

← *val* The [value](#).

→ *output* The result image.

Precondition:

```
output.domain == input.domain
```

References `times_cst()`.

Referenced by `times_cst()`.

9.14.2.23 `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:

← *lhs* First operand image (subject to addition).

↔ *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`

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:

- ← *input* The input image.
- ← *fun* The thresholding function, from value(I) to bool.

`for_all(p), output(p) = fun(p)`

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:

- ← *input* The input image.
- ← *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.

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:

- ↔ `ima` The image whose [border](#) is to be adjusted.
- ← `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.

References `get()`, and `resize()`.

9.18.2.2 `template<typename I> void mln::border::duplicate (const Image< I > & ima)` [inline]

Assign the virtual (outer) `border` of image `ima` with the duplicate of the inner `border` of this image.

Parameters:

↔ *ima* The image whose `border` is to be duplicated.

Precondition:

`ima` has to be initialized.

References `get()`.

Referenced by `mln::extension::duplicate()`.

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:

↔ *ima1* The first image whose `border` is to be equalized.

↔ *ima2* The second image whose `border` is to be equalized.

← *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.

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:

↔ *ima* The image whose `border` is to be filled.

← *v* The [value](#) to assign to all [border](#) pixels.

Precondition:

ima has to be initialized.

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:

← *ima* The image.

Returns:

The [border](#) thickness (0 if there is no [border](#)).

Precondition:

ima has to be initialized.

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:

← *ima* The image.

Returns:

The [border](#) thickness (0 if there is no [border](#)).

Precondition:

ima has to be initialized.

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:

↔ *ima* The image whose [border](#) is to be mirrored.

Precondition:

ima has to be initialized.

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:

↔ *ima* The image whose `border` is to be resized.

← *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.

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](#).

Classes

- struct [chamfer](#)
Compute [chamfer](#) distance.

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](#).

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.

Referenced by `mln::transform::distance_front()`, and `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](#).

Referenced by `mln::transform::distance_and_closest_point_geodesic()`, `mln::transform::distance_and_influence_zone_geodesic()`, `mln::transform::distance_geodesic()`, and `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:

- ← *input* The input image.
- ← *nbh* The connexity of the objects.
- *nlabels* The Number of labels. Its [value](#) is [set](#) in the algorithms.
- ↔ *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).

Referenced by `mln::labeling::blobs()`, and `mln::labeling::blobs_and_compute()`.

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 unsigned &from, Value< V > &to)`
Conversion of an unsigned `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 float &from, Value< V > &to)`
Conversion of a float `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 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 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 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 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 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 R, typename A>`
`fun::C< R(*) (A)> to_fun (R(*) (A))`
Convert a C unary function into an mln::fun::C.
- `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 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 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 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 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 W>`
`p_set< typename W::psite > to_p_set (const Window< W > &win)`
*Convert a *Window win* 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 N>`
`p_set< typename N::psite > to_p_set (const Neighborhood< N > &nbh)`
*Convert a *neighborhood nbh* into a site *set*.*
- `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 S>`
`window< typename S::site::dpsite > to_window (const Site_Set< S > &pset)`
Convert a site `set` `pset` 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`.

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 unsigned &from, Value< V > &to) [inline]`

Conversion of an unsigned `from` towards a `value` `to`.

9.27.2.2 `template<typename V> void mln::convert::from_to (const int &from, Value< V > &to) [inline]`

Conversion of a `int` `from` towards a `value` `to`.

9.27.2.3 `template<typename V> void mln::convert::from_to (const float &from, Value< V > &to) [inline]`

Conversion of a `float` `from` towards a `value` `to`.

9.27.2.4 `template<typename V> void mln::convert::from_to (const double &from, Value< V > &to) [inline]`

Conversion of a `double` `from` towards a `value` `to`.

9.27.2.5 `template<typename N> mln::convert::mln_image_from_grid (typename N::site::grid, bool) const [inline]`

Convert a neighborhood `nbh` into a binary image.

9.27.2.6 `template<typename W> mln::convert::mln_image_from_grid (typename W::site::grid, mln_weight(W)) const` `[inline]`

Convert a weighted [window](#) `w_win` into an image.

9.27.2.7 `template<typename W> mln::convert::mln_image_from_grid (typename W::site::grid, bool) const` `[inline]`

Convert a [window](#) `win` into a binary image.

9.27.2.8 `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`.

9.27.2.9 `template<typename N> mln::convert::mln_window (N) const` `[inline]`

Convert a neighborhood `nbh` into a [window](#).

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`.

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.

9.27.2.12 `template<typename I> pw::value_< I > mln::convert::to_fun (const Image< I > & ima)` `[inline]`

Convert an image into a function.

9.27.2.13 `template<typename R, typename A> fun::C< R*(A)> mln::convert::to_fun (R*(A) f)` `[inline]`

Convert a `C` unary function into an `mln::fun::C`.

9.27.2.14 `template<typename T> image1d<unsigned> mln::convert::to_image (const histo::array< T > & h)` `[inline]`

Convert an [histo](#) `h` into an `image1d<unsigned>`.

9.27.2.15 `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`.

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).

References `mln::p_array< P >::append()`, and `mln::p_array< P >::reserve()`.

9.27.2.17 `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).

References `mln::p_array< P >::append()`.

9.27.2.18 `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`.

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.

References `mln::p_set< P >::insert()`.

9.27.2.20 `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`.

References `mln::p_set< P >::insert()`.

9.27.2.21 `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`.

References `mln::p_set< P >::insert()`.

9.27.2.22 `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](#).

References `mln::p_set< P >::insert()`.

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](#).

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](#).

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](#).

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

9.27.2.26 `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](#).

References `to_window()`.

9.27.2.27 `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](#).

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

Referenced by `to_window()`.

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 (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 (const 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)`
Generic implementation of [data::stretch](#).
- `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)`
Generic implementation of [data::transform](#).
- `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>`
`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`.
- `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:

- ← *input* The input image.
- *output* The output image.

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:

- ↔ *input* The input image.

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:

- ↔ *input* The input image.

← *f* The function-object.

This routine runs:

for all *p* of *input*, $\text{input}(p) = f(\text{input}(p))$

This routine is equivalent to `data::transform(input, f, input)` but it is faster since a single iterator is required.

Referenced by `abs_inplace()`, and `saturate_inplace()`.

9.28.2.4 `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:

↔ *a* An accumulator.

← *input* The input image.

Returns:

The accumulator result.

It fully relies on [data::update](#).

9.28.2.5 `template<typename A, typename I> A::result mln::data::compute (const Accumulator< 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:

← *a* An accumulator.

← *input* The input image.

Returns:

The accumulator result.

It fully relies on [data::update](#).

Compute an accumulator onto the [pixel](#) values of the image *input*.

Parameters:

← *input* The input image.

← *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()`.

Referenced by `mln::labeled_image< I >::labeled_image()`, `mln::estim::mean()`, `mln::estim::min_max()`, `mln::labeling::pack()`, `mln::labeling::pack_inplace()`, and `mln::estim::sum()`.

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:

- ← `v` A `value` of the destination type.
- ← `input` The input image.

References `transform()`.

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:

- ← `input` The image to be filtered.
- ← `win` The `window`.
- ↔ `output` The output image.

Precondition:

`input` and `output` have to be initialized.

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:

- ↔ `ima` The image to be filled.
- ← `data` The auxiliary `data` to fill the image `ima`.

Precondition:

`ima` has to be initialized.

Referenced by mln::topo::detach(), mln::util::display_branch(), mln::transform::distance_and_closest_point_geodesic(), mln::duplicate(), mln::make::edge_image(), mln::labeling::fill_holes(), mln::morpho::tree::filter::filter(), mln::morpho::impl::generic::hit_or_miss(), mln::transform::hough(), mln::registration::icp(), 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::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:

- ↔ `ima` The image to be filled.
- ← `data` The image.

Warning:

The definition domain of `ima` has to be included in the one of `data`.

Precondition:

```
ima.domain <= data.domain.
```

Fill the image `ima` with the values of the image `data`.

Parameters:

- ↔ `ima_` The image to be filled.
- ← `data_` The image.

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:

- ↔ `ima` The image to be filled.
- ← `val` The `value` to assign to all sites.

Precondition:

`ima` has to be initialized.

Parameters:

- ↔ `ima_` The image to be filled.
- ← `val` The `value` to assign to all sites.

Precondition:

`ima` has to be initialized.

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) [inline]`

Compute in `output` the median filter of image `input` by the `window win`.

Parameters:

← *input* The image to be filtered.

← *win* The `window`.

Precondition:

`input` have to be initialized.

References `mln::extension::adjust()`, and `mln::initialize()`.

Referenced by `mln::data::approx::median()`.

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:

← *a* A meta-accumulator.

← *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:

← *input* The input image providing pixels values.

↔ *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`

Paste the contents of image `input` into the image `output`.

Parameters:

- ← *input* The input image providing pixels values.
- ↔ *output* The image in which values are assigned.

Referenced by `mln::make::image3d()`, `mln::draw::line()`, `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:

- ← *input* The input image providing pixels values.
- ↔ *output* The image in which values are assigned.

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)` [inline]

Replace `old_value` by `new_value` in the image `input`

Parameters:

- ← *input* The input image.
- ← *old_value* The `value` to be replaced...
- ← *new_value* ...by this one.

9.28.2.16 `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:

- ← *input* The input image.
- ← *min* The minimum output `value`.
- ← *max* The maximum output `value`.

References `transform()`.

9.28.2.17 `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:

- ← *v* A [value](#) of the output type.
- ← *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.

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:

- ↔ *input* The input image.
- ← *min* The minimum output [value](#).
- ← *max* The maximum output [value](#)

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.

References `mln::util::array< T >::append()`, and `mln::util::array< T >::reserve()`.

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:

- ← *input* The input image.

Precondition:

`input.is_valid`

Referenced by `mln::morpho::tree::min_tree()`.

9.28.2.21 `template<typename I> p_array< typename I::psite > mln::data::sort_psites_increasing (const Image< I > & input) [inline]`

Sort psites the image `input` through a function `f` to [set](#) the `output` image in increasing way.

Parameters:

← *input* The input image.

Precondition:

`input.is_valid`

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]`

Generic implementation of [data::stretch](#).

Stretch the values of `input` so that they can be stored in `output`.

Parameters:

← *v* A [value](#) to [set](#) the output [value](#) type.

← *input* The input image.

Returns:

A stretch image with values of the same type as `v`.

Precondition:

`input.is_valid`

Parameters:

← *v* A [value](#) to [set](#) the output [value](#) type.

← *input* The input image.

Returns:

A stretch image with values of the same type as `v`.

References `mln::initialize()`, `mln::estim::min_max()`, `mln::data::impl::stretch()`, and `transform()`.

Referenced by `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:

← *input* The input image.

→ *output* The result image.

Precondition:

`output.domain >= input.domain`

References `transform()`.

9.28.2.24 `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]`

Generic implementation of `data::transform`.

Transform two images `input1 input2` through a function `f`.

Parameters:

- ← *input1* The 1st input image.
- ← *input2* The 2nd input image.
- ← *f* The function.

This routine runs:

for all `p` of `input`, `output (p) = f(input1 (p), input2 (p))`.

Parameters:

- ← *input1_* The 1st input image.
- ← *input2_* The 2nd input image.
- ← *f_* The function.

References `mln::initialize()`.

9.28.2.25 `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]`

Generic implementation of `data::transform`.

Transform the image `input` through a function `f`.

Parameters:

- ← *input* The input image.
- ← *f* The function.

This routine runs:

for all `p` of `input`, `output (p) = f(input (p))`.

Parameters:

- ← *input_* The input image.

← *f_* The function.

References mln::initialize().

Referenced by abs(), mln::logical::and_not(), mln::labeling::colorize(), mln::data::impl::generic::convert(), mln::arith::diff_abs(), 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().

9.28.2.26 `template<typename I1, typename I2, typename F> void mln::data::transform_inplace (Image< I1 > & ima_, const Image< I2 > & aux_, const Function_vv2v< F > & f_) [inline]`

Generic implementation of transform_inplace.

Transform inplace the image *ima* with the image *aux* through a function *f*.

Parameters:

← *ima* The image to be transformed.
 ← *aux* The auxiliary image.
 ← *f* The function.

This routine runs:

for all *p* of *ima*, $ima(p) = f(ima(p), aux(p))$.

Parameters:

← *ima_* The image to be transformed.
 ← *aux_* The auxiliary image.
 ← *f_* The function.

9.28.2.27 `template<typename I, typename F> void mln::data::transform_inplace (Image< I > & ima_, const Function_v2v< F > & f_) [inline]`

Generic implementation of transform_inplace.

Transform inplace the image *ima* through a function *f*.

Parameters:

↔ *ima* The image to be transformed.
 ← *f* The function.

This routine runs:

for all *p* of *ima*, $ima(p) = f(ima(p))$.

Parameters:

↔ *ima_* The image to be transformed.
 ← *f_* The function.

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]`

Generic implementation of [data::update](#).

Update an accumulator with the [pixel](#) values of the image `input_`.

Parameters:

- ← *a* The accumulator.
- ← *input_* The input image.

Returns:

The accumulator result.

Parameters:

- ← *a_* The accumulator.
- ← *input_* The input image.

Returns:

The accumulator result.

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) [inline]`

Routine to wrap values such as 0 -> 0 and [1, lmax] maps to [1, Lmax] (using modulus).

Parameters:

- ← *v* The target [value](#) type.
- ← *input* Input image.

Returns:

An image with wrapped values.

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::octagon2d &win)`
- `template<typename I>`
`mln::trait::concrete< I >::ret median (const Image< I > &input, const win::disk2d &win)`
- `template<typename I>`
`mln::trait::concrete< I >::ret median (const Image< I > &input, const win::rectangle2d &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::octagon2d &win) [inline]`

Compute in `output` an approximate of the median filter of image `input` by the 2D octagon `win`.

Parameters:

- ← `input` The image to be filtered.
- ← `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.

References `median()`.

9.29.2.2 `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:

- ← *input* The image to be filtered.
- ← *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.

References `mln::data::median()`.

9.29.2.3 `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:

- ← *input* The image to be filtered.
- ← *win* The rectangle.

The approximation is based on a vertical median ran after an horizontal median.

Precondition:

`input` and `output` have to be initialized.

References `mln::data::median()`.

Referenced by `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 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 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:

- ← *v* A [value](#) to [set](#) the output [value](#) type.
- ← *input* The input image.

Returns:

A stretch image with values of the same type as *v*.

References [mln::initialize\(\)](#), [mln::estim::min_max\(\)](#), [stretch\(\)](#), and [mln::data::transform\(\)](#).

Referenced by [mln::data::stretch\(\)](#).

9.31.2.2 `template<typename I, typename F> void mln::data::impl::transform_inplace_lowq
(Image< I > & input_, const Function_v2v< F > & f_) [inline]`

Specialized implementation.

9.31.2.3 `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:

← *a_* The accumulator.

← *input_* The input image.

Returns:

The accumulator result.

9.32 mln::data::impl::generic Namespace Reference

Generic implementation namespace of [data](#) namespace.

Functions

- `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 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 W>`
`mln::trait::concrete< I >::ret median (const Image< I > &input, const Window< W > &win)`
- `template<typename I, typename J>`
`void paste (const Image< I > &input_, Image< J > &output_)`
Generic implementation of [data::paste](#).
- `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 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 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>`
`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 V, typename I> mln::trait::ch_value< I, V >::ret
mln::data::impl::generic::convert (const V & v, const Image< I > & input) [inline]`

Convert the image `input` by changing the [value](#) type.

Parameters:

- ← `v` A [value](#) of the destination type.
- ← `input` The input image.

References `mln::data::transform()`.

Referenced by `mln::morpho::watershed::superpose()`, and `mln::debug::superpose()`.

9.32.2.2 `template<typename I, typename J> void mln::data::impl::generic::fill_with_image
(Image< I > & ima_, const Image< J > & data_) [inline]`

Generic implementation.

Fill the image `ima` with the values of the image `data`.

Parameters:

- ↔ `ima_` The image to be filled.
- ← `data_` The image.

9.32.2.3 `template<typename I, typename V> void mln::data::impl::generic::fill_with_value
(Image< I > & ima_, const V & val) [inline]`

Fill the whole image `ima` with the single [value](#) `v`.

Parameters:

- ↔ `ima_` The image to be filled.
- ← `val` The [value](#) to assign to all sites.

Precondition:

`ima` has to be initialized.

Referenced by `mln::p_image< I >::clear()`.

9.32.2.4 `template<typename I, typename W> mln::trait::concrete< I >::ret
mln::data::impl::generic::median (const Image< I > & input, const Window< W > &
win) [inline]`

Compute in `output` the median filter of image `input` by the `window win`.

Parameters:

- ← `input` The image to be filtered.
- ← `win` The `window`.

Precondition:

`input` have to be initialized.

References `mln::extension::adjust()`, and `mln::initialize()`.

Referenced by `mln::data::approx::median()`.

9.32.2.5 `template<typename I, typename J> void mln::data::impl::generic::paste (const Image<
I > & input_, Image< J > & output_) [inline]`

Generic implementation of `data::paste`.

Paste the contents of image `input` into the image `output`.

Parameters:

- ← `input_` The input image providing pixels values.
- ↔ `output_` The image in which values are assigned.

Referenced by `mln::make::image3d()`, `mln::draw::line()`, `mln::geom::rotate()`, `mln::debug::slices_2d()`, and `mln::labeling::superpose()`.

9.32.2.6 `template<typename I> util::array<unsigned> mln::data::impl::generic::sort_offsets_
increasing (const Image< I > & input_) [inline]`

Sort `pixel` offsets of the image `input` wrt increasing `pixel` values.

References `mln::util::array< T >::append()`, and `mln::util::array< T >::reserve()`.

9.32.2.7 `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_) [inline]`

Generic implementation of `data::transform`.

Parameters:

- ← `input1_` The 1st input image.
- ← `input2_` The 2nd input image.
- ← `f_` The function.

References `mln::initialize()`.

9.32.2.8 `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_) [inline]`

Generic implementation of [data::transform](#).

Parameters:

- ← *input_* The input image.
- ← *f_* The function.

References `mln::initialize()`.

Referenced by `mln::data::abs()`, `mln::logical::and_not()`, `mln::labeling::colorize()`, `convert()`, `mln::arith::diff_abs()`, `mln::linear::mln_ch_convolve_grad()`, `mln::labeling::pack()`, `mln::labeling::pack_inplace()`, `mln::labeling::relabel()`, `mln::data::saturate()`, `mln::data::impl::stretch()`, `mln::data::to_enc()`, `mln::labeling::wrap()`, and `mln::data::wrap()`.

9.32.2.9 `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_) [inline]`

Generic implementation of `transform_inplace`.

Parameters:

- ← *ima_* The image to be transformed.
- ← *aux_* The auxiliary image.
- ← *f_* The function.

9.32.2.10 `template<typename I, typename F> void mln::data::impl::generic::transform_inplace (Image< I > & ima_, const Function_v2v< F > & f_) [inline]`

Generic implementation of `transform_inplace`.

Parameters:

- ↔ *ima_* The image to be transformed.
- ← *f_* The function.

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()`.

9.32.2.11 `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:

- ← *a_* The accumulator.

← *input_* The input image.

Returns:

The accumulator result.

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:

- ← *input* The image to be filtered.
- ← *win* The [window](#).
- ↔ *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](#)

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, 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`.
- `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>`
`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.*
- `std::string filename (const std::string &filename, int id)`
Constructs and returns a formatted output file name.
- unsigned short [format](#) (unsigned char v)
*Format an unsigned char to print it properly, i.e., like an integer *value*.*
- signed short [format](#) (signed char v)
*Format a signed char to print it properly, i.e., like an integer *value*.*
- char [format](#) (bool v)
Format a Boolean to print it nicely: "|" for true and "-" for false.
- `template<typename T>`
`const T & format (const T &v)`
*Default version for formatting a *value* is a no-op.*
- `template<typename I>`
`void iota (Image< I > &input)`
- `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, float ratio_hv, 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, 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, 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.

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, 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_*.

References [mln::p_line2d::begin\(\)](#), [mln::p_line2d::end\(\)](#), [mln::p_vertices< G, F >::graph\(\)](#), and [mln::draw::line\(\)](#).

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_*.

References `mln::p_vertices< G, F >::graph()`, and `mln::draw::line()`.

9.35.2.3 `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.

References `mln::p_vertices< G, F >::graph()`, and `mln::draw::line()`.

Referenced by `mln::make_debug_graph_image()`.

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

9.35.2.5 `unsigned short mln::debug::format (unsigned char v)` `[inline]`

Format an unsigned char to print it properly, i.e., like an integer `value`.

9.35.2.6 `signed short mln::debug::format (signed char v)` `[inline]`

Format a signed char to print it properly, i.e., like an integer `value`.

9.35.2.7 `char mln::debug::format (bool v)` `[inline]`

Format a Boolean to print it nicely: `"|"` for true and `"-"` for false.

9.35.2.8 `template<typename T> const T & mln::debug::format (const T & v)` `[inline]`

Default version for formatting a `value` is a no-op.

Referenced by `mln::value::operator<<()`, and `mln::Gpoint< E >::operator<<()`.

9.35.2.9 `template<typename I> void mln::debug::iota (Image< I > & input)` `[inline]`

Fill the image `input` with successive values.

Parameters:

↔ *input* The image in which values are assigned.

9.35.2.10 `template<typename I> void mln::debug::println (const std::string & msg, const Image< I > & input)` [inline]

Print the message `msg` and the image `input` on the standard output.

References `println()`.

9.35.2.11 `template<typename I> void mln::debug::println (const Image< I > & input)` [inline]

Print the image `input` on the standard output.

References `mln::geom::bbox()`.

Referenced by `println()`.

9.35.2.12 `template<typename I> void mln::debug::println_with_border (const Image< I > & input)` [inline]

Print the image `input` on the standard output.

References `mln::geom::bbox()`.

9.35.2.13 `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`.

References `mln::image2d< T >::has()`, and `mln::point< G, C >::last_coord()`.

9.35.2.14 `template<typename I> image2d< typename I::value > mln::debug::slices_2d (const Image< I > & input, float ratio_hv, const typename I::value & bg)` [inline]

Create a 2D image of the slices of the 3D image `input`.

References `slices_2d()`.

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`.

References `mln::apply_p2p()`, `mln::data::fill()`, and `mln::data::paste()`.

Referenced by `slices_2d()`.

9.35.2.16 `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) [inline]`

Superpose two images.

Parameters:

- ← *input_* An image. Its `value` type must be convertible toward `value::rgb8` thanks to a conversion operator or `convert::from_to`.
- ← *object_* A scalar or labeled image. Objects used for superposition. have their `pixel` values different from 0.
- ← *object_color* The color used to `draw` the objects in `object_`.

Precondition:

`input_` and `object_` must have the same domain.

Returns:

A color image.

References `mln::data::convert()`, `mln::data::fill()`, and `mln::literal::zero`.

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'.

9.37.2.2 typedef float mln::def::coordf

Definition of the floating coordinate type.

9.37.3 Enumeration Type Documentation

9.37.3.1 anonymous enum

Definition of the number of bits of the low quantization threshold.

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>`
`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)`

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:

- ↔ `ima` The image to be drawn.
- ← `b` the box to `draw`.
- ← `v` The `value` to assign to all drawn pixels.

Precondition:

- `ima` has to be initialized.
- `ima` has `beg`.
- `ima` has `end`.

References `line()`.

9.42.2.2 `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:

- ↔ `ima` The image to be drawn.
- ← `beg` The start `point` to drawn line.
- ← `end` The end `point` to drawn line.

← *v* The [value](#) to assign to all drawn pixels.

Precondition:

ima has to be initialized.
ima has beg.
ima has end.

References `mln::data::paste()`.

Referenced by `box()`, and `mln::debug::draw_graph()`.

9.42.2.3 `template<typename I> void mln::draw::plot (Image< I > & ima, const typename I::point & p, const typename I::value & v)` [inline]

Plot a [point](#) at level *v* in image *ima*

Parameters:

↔ *ima* The image to be drawn.
← *p* The [point](#) to be plotted.
← *v* The [value](#) to assign to all drawn pixels.

Precondition:

ima has to be initialized.
ima has *p*.

9.43 mln::estim Namespace Reference

Namespace of estimation materials.

Functions

- `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>`
`mln::value::props< typename I::value >::sum mean (const Image< I > &input)`
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, typename S>`
`void sum (const Image< I > &input, S &result)`
Compute the sum [value](#) 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`.

9.43.1 Detailed Description

Namespace of estimation materials.

9.43.2 Function Documentation

9.43.2.1 `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:

- ← *`input`* The image.
- *`result`* The mean [value](#).

The free parameter `S` is the type used to compute the summation.

References `mln::data::compute()`.

9.43.2.2 `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:

← *input* The image.

Returns:

The mean [value](#).

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:

← *input* The image.

→ *min* The minimum [pixel value](#) of `input`.

→ *max* The maximum [pixel value](#) of `input`.

References `mln::data::compute()`.

Referenced by `mln::data::impl::stretch()`, and `mln::make::voronoi()`.

9.43.2.4 `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:

← *input* The image.

→ *result* The sum [value](#).

References `mln::data::compute()`.

9.43.2.5 `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:

← *input* The image.

Returns:

The sum [value](#).

References `mln::data::compute()`.

9.44 mln::extension Namespace Reference

Namespace of [extension](#) tools.

Functions

- `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 (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, 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_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> void mln::extension::adjust (const Image< I > & ima, unsigned delta)` `[inline]`

Adjust the domain [extension](#) of image `ima` with the size `delta`.

9.44.2.2 `template<typename I, typename N> void mln::extension::adjust (const Image< I > & ima, const Neighborhood< N > & nbh) [inline]`

Adjust the domain [extension](#) of image `ima` with the size of the neighborhood `nbh`.

References `mln::geom::delta()`.

9.44.2.3 `template<typename I, typename W> void mln::extension::adjust (const Image< I > & ima, const Weighted_Window< W > & wwin) [inline]`

Adjust the domain [extension](#) of image `ima` with the size of the weighted [window](#) `wwin`.

References `mln::geom::delta()`.

9.44.2.4 `template<typename I, typename W> void mln::extension::adjust (const Image< I > & ima, const Window< W > & win) [inline]`

Adjust the domain [extension](#) of image `ima` with the size of the [window](#) `win`.

References `mln::geom::delta()`.

Referenced by `adjust_duplicate()`, `adjust_fill()`, and `mln::data::impl::generic::median()`.

9.44.2.5 `template<typename I, typename W> void mln::extension::adjust_duplicate (const Image< I > & ima, const Window< W > & win) [inline]`

Adjust then duplicate.

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) [inline]`

Adjust then fill.

References `adjust()`, and `fill()`.

Referenced by `mln::morpho::impl::generic::rank_filter()`.

9.44.2.7 `template<typename I> void mln::extension::duplicate (const Image< I > & ima) [inline]`

Assign the contents of the domain [extension](#) by duplicating the values of the inner boundary of image `ima`.

References `mln::border::duplicate()`.

Referenced by `adjust_duplicate()`.

9.44.2.8 `template<typename I> void mln::extension::fill (const Image< I > & ima, const typename I::value & val) [inline]`

Fill the domain [extension](#) of image `ima` with the single [value](#) `v`.

Parameters:

- ↔ *ima* The image whose domain [extension](#) is to be filled.
- ← *val* The [value](#) to assign.

Precondition:

ima has to be initialized.

Referenced by `adjust_fill()`.

9.45 mln::fun Namespace Reference

Namespace of functions.

Classes

- struct [from_accu](#)
Wrap an accumulator into a function.

Namespaces

- namespace [access](#)
Namespace for [access](#) functions.
- namespace [i2v](#)
Namespace of integer-to-value functions.
- 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](#).

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) [inline]`

Operator<<.

9.48 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`.

9.48.1 Detailed Description

Namespace of functions from [point](#) to boolean.

9.49 mln::fun::p2p Namespace Reference

Namespace of functions from [grid point](#) to [grid point](#).

9.49.1 Detailed Description

Namespace of functions from [grid point](#) to [grid point](#).

9.50 mln::fun::p2v Namespace Reference

Namespace of functions from [point](#) to [value](#).

9.50.1 Detailed Description

Namespace of functions from [point](#) to [value](#).

9.51 mln::fun::stat Namespace Reference

Namespace of statistical functions.

9.51.1 Detailed Description

Namespace of statistical functions.

9.52 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.

9.52.1 Detailed Description

Namespace of functions from [value](#) to logic [value](#).

9.53 mln::fun::v2i Namespace Reference

Namespace of value-to-integer functions.

9.53.1 Detailed Description

Namespace of value-to-integer functions.

9.54 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. $f(v) = a * v + b$. \forall is the type of input values; \mathbb{T} is the type used to compute the result; \mathbb{R} is the result type.*
- struct [linfty_norm](#)
L-infty norm.

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.

9.54.1 Detailed Description

Namespace of functions from [value](#) to [value](#).

9.54.2 Variable Documentation

9.54.2.1 f_hsi_to_rgb_3x8_t mln::fun::v2v::f_hsi_to_rgb_3x8

Global variable.

9.54.2.2 f_hsl_to_rgb_3x8_t mln::fun::v2v::f_hsl_to_rgb_3x8

Global variables.

9.54.2.3 f_rgb_to_hsi_f_t mln::fun::v2v::f_rgb_to_hsi_f

Global variables.

9.54.2.4 f_rgb_to_hsl_f_t mln::fun::v2v::f_rgb_to_hsl_f

Global variables.

9.55 mln::fun::v2w2v Namespace Reference

Namespace of bijective functions.

Classes

- struct `cos`

Cosinus bijective functor.

9.55.1 Detailed Description

Namespace of bijective functions.

9.56 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.

9.56.1 Detailed Description

Namespace of functions from [value](#) to [value](#).

9.57 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.

9.57.1 Detailed Description

Namespace of functions from [value](#) to [value](#).

9.58 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.

9.58.1 Detailed Description

Namespace of functions from a couple of values to a [value](#).

9.59 mln::fun::x2p Namespace Reference

Namespace of functions from [point](#) to [value](#).

Classes

- struct [closest_point](#)
FIXME: doxygen + concept checking.

9.59.1 Detailed Description

Namespace of functions from [point](#) to [value](#).

9.60 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.

9.60.1 Detailed Description

Namespace of functions from vector to [value](#).

9.61 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.

9.61.1 Detailed Description

Namespace of functions from vector to vector.

9.62 mln::geom Namespace Reference

Namespace of all things related to geometry.

Classes

- class [complex_geometry](#)

A functor returning the sites of the faces of a complex where the locations of each 0-face is stored.

Namespaces

- namespace [impl](#)

Implementation namespace of [geom](#) namespace.

Functions

- `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 W>`
`box< typename W::psite > bbox (const Window< W > &win)`
Compute the precise bounding [box](#) of a [window](#) win.
- `template<typename I>`
`box< typename I::site > bbox (const Image< I > &ima)`
Compute the precise bounding [box](#) of a [point set](#) pset.
- `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, 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 N>`
`unsigned delta (const Neighborhood< N > &nbh)`
Compute the [delta](#) of a [neighborhood](#) nbh.
- `template<typename W>`
`unsigned delta (const Weighted_Window< W > &wwin)`
Compute the [delta](#) of a weighted [window](#) wwin.
- `template<typename W>`
`unsigned delta (const Window< W > &win)`

Compute the delta of a [window win](#).

- `template<typename B>`
`B::point::coord max_col (const Box< B > &b)`
Give the maximum col of an [box](#) 2d or 3d.
- `template<typename I>`
`I::site::coord max_col (const Image< I > &ima)`
Give the maximum column of an image.
- `template<typename I>`
`I::site::coord max_ind (const Image< I > &ima)`
Give the maximum ind of an image.
- `template<typename B>`
`B::point::coord max_row (const Box< B > &b)`
Give the maximum row of an [box](#) 2d or 3d.
- `template<typename I>`
`I::site::coord max_row (const Image< I > &ima)`
Give the maximum row of an image.
- `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 B>`
`B::point::coord min_col (const Box< B > &b)`
Give the minimum column of an [box](#) 2d or 3d.
- `template<typename I>`
`I::site::coord min_col (const Image< I > &ima)`
Give the minimum column of an image.
- `template<typename I>`
`I::site::coord min_ind (const Image< I > &ima)`
Give the minimum ind of an image.

- `template<typename B>`
`B::point::coord min_row` (const `Box< B >` &b)
Give the minimum row of an `box` 2d or 3d.
- `template<typename I>`
`I::site::coord min_row` (const `Image< I >` &ima)
Give the minimum row of an image.
- `template<typename I>`
`I::site::coord min_sli` (const `Image< I >` &ima)
Give the minimum sli 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 ncols` (const `Image< I >` &ima)
Give the number of columns of an image.
- `template<typename I>`
`unsigned ninds` (const `Image< I >` &ima)
Give the number of inds 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 nrows` (const `Image< I >` &ima)
Give the number of rows of an image.
- `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 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>`
`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 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>`
`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 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, 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>`
`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, 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 N>`
`I seeds2tiling_roundness (Image< I > &ima_, const w_window2d_int &w_win, unsigned max, const Neighborhood< N > &nbh_)`
 Take a labeled image `ima_` with seeds and extend them until creating tiles rounder than the primary version.

9.62.1 Detailed Description

Namespace of all things related to geometry.

Namespace of essential things related to geometry.

9.62.2 Function Documentation

9.62.2.1 `template<typename W> box< typename W::psite > mln::geom::bbox (const Weighted_Window< W > &win) [inline]`

Compute the precise bounding `box` of a weighted `window win`.

References `bbox()`.

9.62.2.2 `template<typename W> box< typename W::site > mln::geom::bbox (const Window< W > & win) [inline]`

Compute the precise bounding `box` of a `window win`.

References `mln::literal::origin`, and `mln::accu::shape::bbox< P >::take()`.

9.62.2.3 `template<typename I> box< typename I::site > mln::geom::bbox (const Image< I > & ima) [inline]`

Compute the precise bounding `box` of a `point set pset`.

References `bbox()`.

9.62.2.4 `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`.

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()`.

9.62.2.5 `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)) [inline]`

Apply chamfer algorithm to a binary image.

Referenced by `mln::geom::impl::seeds2tiling_roundness()`.

9.62.2.6 `template<typename N> unsigned mln::geom::delta (const Neighborhood< N > & nbh) [inline]`

Compute the delta of a neighborhood `nbh`.

References `delta()`.

9.62.2.7 `template<typename W> unsigned mln::geom::delta (const Weighted_Window< W > & wwin) [inline]`

Compute the delta of a weighted `window wwin`.

References `delta()`.

9.62.2.8 `template<typename W> unsigned mln::geom::delta (const Window< W > & win) [inline]`

Compute the delta of a `window win`.

Referenced by `mln::extension::adjust()`, `delta()`, and `mln::morpho::impl::generic::rank_filter()`.

9.62.2.9 `template<typename B> B::point::coord mln::geom::max_col (const Box< B > & b)`
`[inline]`

Give the maximum col of an `box` 2d or 3d.

9.62.2.10 `template<typename I> I::site::coord mln::geom::max_col (const Image< I > & ima)`
`[inline]`

Give the maximum column of an image.

References `bbox()`.

Referenced by `ncols()`.

9.62.2.11 `template<typename I> I::site::coord mln::geom::max_ind (const Image< I > & ima)`
`[inline]`

Give the maximum ind of an image.

Referenced by `ninds()`.

9.62.2.12 `template<typename B> B::point::coord mln::geom::max_row (const Box< B > & b)`
`[inline]`

Give the maximum row of an `box` 2d or 3d.

9.62.2.13 `template<typename I> I::site::coord mln::geom::max_row (const Image< I > & ima)`
`[inline]`

Give the maximum row of an image.

References `bbox()`.

Referenced by `nrows()`.

9.62.2.14 `template<typename I> I::site::coord mln::geom::max_sli (const Image< I > & ima)`
`[inline]`

Give the maximum sli of an image.

References `bbox()`.

Referenced by `nslis()`.

9.62.2.15 `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).”

References mln::data::fill(), mln::norm::sqr_l2(), mln::algebra::vprod(), and mln::literal::zero.

Referenced by mesh_curvature().

9.62.2.16 `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:

← *mesh* The surface (triangle mesh) on which the curvature is to be computed.

References mln::c2(), mln::algebra::ldlt_decomp(), mln::algebra::ldlt_solve(), mesh_corner_point_area(), mesh_normal(), mln::algebra::vprod(), and mln::literal::zero.

9.62.2.17 `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.

References mln::data::fill(), mln::norm::sqr_l2(), mln::algebra::vprod(), and mln::literal::zero.

Referenced by mesh_curvature().

9.62.2.18 `template<typename B> B::point::coord mln::geom::min_col (const Box< B > & b)`
 [inline]

Give the minimum column of an `box` 2d or 3d.

9.62.2.19 `template<typename I> I::site::coord mln::geom::min_col (const Image< I > & ima)`
`[inline]`

Give the minimum column of an image.

References `bbox()`.

Referenced by `mln::transform::hough()`, and `ncols()`.

9.62.2.20 `template<typename I> I::site::coord mln::geom::min_ind (const Image< I > & ima)`
`[inline]`

Give the minimum ind of an image.

Referenced by `ninds()`.

9.62.2.21 `template<typename B> B::point::coord mln::geom::min_row (const Box< B > & b)`
`[inline]`

Give the minimum row of an `box` 2d or 3d.

9.62.2.22 `template<typename I> I::site::coord mln::geom::min_row (const Image< I > & ima)`
`[inline]`

Give the minimum row of an image.

References `bbox()`.

Referenced by `mln::transform::hough()`, and `nrows()`.

9.62.2.23 `template<typename I> I::site::coord mln::geom::min_sli (const Image< I > & ima)`
`[inline]`

Give the minimum sli of an image.

References `bbox()`.

Referenced by `nslis()`.

9.62.2.24 `template<typename B> unsigned mln::geom::ncols (const Box< B > & b)` `[inline]`

Give the number of cols of a `box` 2d or 3d.

References `max_col()`, `min_col()`, and `ncols()`.

9.62.2.25 `template<typename I> unsigned mln::geom::ncols (const Image< I > & ima)`
`[inline]`

Give the number of columns of an image.

References `max_col()`, and `min_col()`.

Referenced by `mln::subsampling::gaussian_subsampling()`, `mln::transform::hough()`, `ncols()`, and `mln::subsampling::subsampling()`.

9.62.2.26 `template<typename I> unsigned mln::geom::ninds (const Image< I > & ima)`
[inline]

Give the number of inds of an image.

References `max_ind()`, and `min_ind()`.

9.62.2.27 `template<typename B> unsigned mln::geom::nrows (const Box< B > & b)` [inline]

Give the number of rows of a `box` 2d or 3d.

References `max_row()`, `min_row()`, and `nrows()`.

9.62.2.28 `template<typename I> unsigned mln::geom::nrows (const Image< I > & ima)`
[inline]

Give the number of rows of an image.

References `max_row()`, and `min_row()`.

Referenced by `mln::subsampling::gaussian_subsampling()`, `mln::transform::hough()`, `nrows()`, and `mln::subsampling::subsampling()`.

9.62.2.29 `template<typename I> unsigned mln::geom::nsites (const Image< I > & input)`
[inline]

Compute the number of sites of the image `input`.

Referenced by `pmin_pmax()`.

9.62.2.30 `template<typename I> unsigned mln::geom::nslis (const Image< I > & ima)`
[inline]

Give the number of slices of an image.

References `max_sli()`, and `min_sli()`.

9.62.2.31 `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`.

9.62.2.32 `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`.

References `pmin_pmax()`.

9.62.2.33 `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`.

References nsites().

9.62.2.34 `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*.

References nsites().

Referenced by pmin_pmax().

9.62.2.35 `template<typename I> mln::trait::concrete< I >::ret mln::geom::rotate (const Image<
I > & input, double angle) [inline]`

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](#).

References rotate(), and mln::literal::zero.

9.62.2.36 `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) [inline]`

Perform a rotation from the center of an image.

Parameters:

- ← *input* An image.
- ← *angle* An angle in degrees.
- ← *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.
- ← *output_domain* The domain of the output image. An invalid domain, causes the routine to use the rotated *input_domain*.

Returns:

An image with the same domain as *input*.

References bbox(), mln::compose(), mln::extend(), mln::initialize(), mln::mln_exact(), mln::literal::origin, mln::data::paste(), mln::accu::shape::bbox< P >::take(), and mln::accu::shape::bbox< P >::to_result().

Referenced by rotate().

9.62.2.37 `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:

- ↔ *ima_* The labeled image with seed.
- ← *nbh* The neighborhood to use on this algorithm.

Returns:

A tiled image.

Precondition:

`ima_` has to be initialized.

Take a labeled image `ima_` with seeds and extend them until creating tiles.

Parameters:

- ↔ `ima_` The labeled image with seed.
- ← `nbh_` The neighborhood to use on this algorithm.

References `mln::duplicate()`, `mln::p_queue< P >::front()`, `mln::p_queue< P >::pop()`, `mln::p_queue< P >::push()`, and `mln::geom::impl::seeds2tiling()`.

Referenced by `seeds2tiling()`.

9.62.2.38 `template<typename I, typename N> I mln::geom::seeds2tiling_roundness (Image< I > & ima_, const w_window2d_int & w_win, unsigned max, const Neighborhood< N > & nbh_) [inline]`

Take a labeled image `ima_` with seeds and extend them until creating tiles rounder than the primary version.

Parameters:

- ↔ `ima_` The labeled image with seed.
- ← `w_win` The weight `window` using by `geom::chamfer` to compute distance.
- ← `max` Unsigned using by `geom::chamfer` to compute the distance.
- ← `nbh_` The neighborhood to use on this algorithm.

Precondition:

`ima_` has to be initialized.

References `chamfer()`, `mln::duplicate()`, `mln::p_priority< P, Q >::pop_front()`, `mln::p_priority< P, Q >::push()`, `mln::geom::impl::seeds2tiling_roundness()`, and `mln::literal::zero`.

Referenced by `seeds2tiling_roundness()`.

9.62.2.39 `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) [inline]`

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`.

References `translate()`, and `mln::literal::zero`.

9.62.2.40 `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)`
[inline]

Perform a translation from the center of an image.

Parameters:

- ← *input* An image.
- ← *ref* The translation vector.
- ← *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.
- ← *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*.

References `mln::extend()`, `mln::data::fill()`, and `mln::mln_exact()`.

Referenced by `translate()`.

9.63 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.

- `template<typename I, typename N>`
`I seeds2tiling_roundness (Image< I > &ima_, const w_window2d_int &w_win, unsigned max, const Neighborhood< N > &nbh_)`

Take a labeled image ima_ with seeds and extend them until creating tiles rounder than the primary version.

9.63.1 Detailed Description

Implementation namespace of [geom](#) namespace.

9.63.2 Function Documentation

- 9.63.2.1** `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.

Take a labeled image ima_ with seeds and extend them until creating tiles.

Parameters:

- ↔ *ima_* The labeled image with seed.
- ← *nbh_* The neighborhood to use on this algorithm.

References `mln::duplicate()`, `mln::p_queue< P >::front()`, `mln::p_queue< P >::pop()`, `mln::p_queue< P >::push()`, and `seeds2tiling()`.

Referenced by `mln::geom::seeds2tiling()`.

- 9.63.2.2** `template<typename I, typename N> I mln::geom::impl::seeds2tiling_roundness`
`(Image< I > & ima_, const w_window2d_int & w_win, unsigned max, const Neighborhood< N > & nbh_) [inline]`

Take a labeled image ima_ with seeds and extend them until creating tiles rounder than the primary version.

Parameters:

- ↔ *ima_* The labeled image with seed.

- ← *w_win* The weight [window](#) using by [geom::chamfer](#) to compute distance.
- ← *max* Unsigned using by [geom::chamfer](#) to compute the distance.
- ← *nbh_* The neighborhood to use on this algorithm.

Precondition:

ima_ has to be initialized.

References [mln::geom::chamfer\(\)](#), [mln::duplicate\(\)](#), [mln::p_priority< P, Q >::pop_front\(\)](#), [mln::p_priority< P, Q >::push\(\)](#), [seeds2tiling_roundness\(\)](#), and [mln::literal::zero](#).

Referenced by [mln::geom::seeds2tiling_roundness\(\)](#).

9.64 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.

9.64.1 Detailed Description

Namespace of [graph](#) related routines.

9.64.2 Function Documentation

9.64.2.1 `template<typename G, typename F> F::result mln::graph::compute (const Graph< G > &g_, F &functor) [inline]`

Base routine to compute attributes on a [graph](#).

Parameters:

- ← `g_` A [graph](#).
- ← `functor` A functor implementing the right interface.

Returns:

The computed [data](#).

See also:

`canvas::browsing::depth_first_search`

9.64.2.2 `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) [inline]`

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:

← *graph_image_* A [graph](#) image (

See also:

[vertex_image](#), [edge_image](#)).

Parameters:

← *nbh_* A [graph](#) neighborhood.

↔ *nlabels* The number of labels found.

Returns:

a [Graph](#) image of labels.

References [mln::labeling::blobs\(\)](#), [mln::data::fill\(\)](#), and [mln::initialize\(\)](#).

9.64.2.3 `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_) [inline]`

Make a custom [graph](#) neighborhood from a mask image.

Parameters:

← *graph_image_* A [graph](#) image (

See also:

[vertex_image](#) and [edge_image](#)).

Parameters:

← *graph_mask_image_* A [graph](#) image of bool used as a mask.

Returns:

A masked neighborhood on [graph](#).

9.64.2.4 `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_) [inline]`

Make a custom [graph window](#) from a mask image.

Parameters:

← *graph_image_* A [graph](#) image (

See also:

[vertex_image](#) and [edge_image](#)).

Parameters:

← *graph_mask_image_* A [graph](#) image of bool used as a mask.

Returns:

A masked [window](#) on [graph](#).

9.65 mln::grid Namespace Reference

Namespace of grids definitions.

9.65.1 Detailed Description

Namespace of grids definitions.

Compute the image::space [trait](#) from a [point](#) type.

9.66 mln::histo Namespace Reference

Namespace of histograms.

Classes

- struct [array](#)
Generic histogram class over a [value set](#) with type T.

Namespaces

- namespace [impl](#)
Implementation namespace of [histo](#) namespace.

Functions

- `template<typename I>
array< typename I::value > compute (const Image< I > &input)`
Compute the histogram of image `input`.

9.66.1 Detailed Description

Namespace of histograms.

9.66.2 Function Documentation

9.66.2.1 `template<typename I> array< typename I::value > mln::histo::compute (const Image< I > &input)` [inline]

Compute the histogram of image `input`.

9.67 mln::histo::impl Namespace Reference

Implementation namespace of [histo](#) namespace.

Namespaces

- namespace [generic](#)
Generic implementation namespace of [histo](#) namespace.

9.67.1 Detailed Description

Implementation namespace of [histo](#) namespace.

9.68 mln::histo::impl::generic Namespace Reference

Generic implementation namespace of [histo](#) namespace.

9.68.1 Detailed Description

Generic implementation namespace of [histo](#) namespace.

9.69 mln::impl Namespace Reference

Implementation namespace of [mln](#) namespace.

9.69.1 Detailed Description

Implementation namespace of [mln](#) namespace.

9.70 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 [tiff](#)

Namespace of [tiff](#) input/output handling.

- namespace [txt](#)

Namespace of [txt](#) input/output handling.

9.70.1 Detailed Description

Namespace of input/output handling.

9.71 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.

9.71.1 Detailed Description

Namespace of [cloud](#) input/output handling.

9.71.2 Function Documentation

9.71.2.1 `template<typename P> void mln::io::cloud::load (p_array< P > &arr, const std::string &filename)` [inline]

Load a [cloud](#) of points.

Parameters:

- ↔ *arr* the site [set](#) where to load the [data](#).
- ← *filename* file to load.

9.71.2.2 `template<typename P> void mln::io::cloud::save (const p_array< P > &arr, const std::string &filename)` [inline]

Load a [cloud](#) of points.

Parameters:

- ← *arr* the [cloud](#) of points to save.
- ← *filename* the destination.

9.72 mln::io::dicom Namespace Reference

Namespace of DICOM input/output handling.

Functions

- `template<typename V>`
`image2d< V > load` (const std::string &filename)
Load a [fits](#) image in a image2d<float>.
- `template<typename I>`
`void load` (Image< I > &ima, const std::string &filename)
Load a DICOM file in a Milena image.

9.72.1 Detailed Description

Namespace of DICOM input/output handling.

9.72.2 Function Documentation

9.72.2.1 `template<typename V> image3d< V > mln::io::dicom::load` (const std::string &filename) [inline]

Load a [fits](#) image in a image2d<float>.

Load a [ppm](#) image in a Milena image.

Load a [pgm](#) image in a Milena image.

Load a [pfm](#) image in a image2d<float>.

Load a [pbm](#) image in a image2d<float>.

Parameters:

← *filename* The image source.

Returns:

An image2d<float> which contains loaded [data](#).

9.72.2.2 `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:

→ *ima* A reference to the image which will receive [data](#).

← *filename* The source.

References `mln::initialize()`, and `mln::point< G, C >::to_vec()`.

9.73 mln::io::dump Namespace Reference

Namespace of [dump](#) input/output handling.

Functions

- `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.

9.73.1 Detailed Description

Namespace of [dump](#) input/output handling.

9.73.2 Function Documentation

9.73.2.1 `template<typename I> void mln::io::dump::load (Image< I > &ima_, const std::string &filename)` [inline]

Load a Milena image by dumped into a file.

Parameters:

- ↔ *ima_* The image to load.
- ← *filename* the destination.

9.73.2.2 `template<typename I> void mln::io::dump::save (const Image< I > &ima_, const std::string &filename)` [inline]

Save a Milena image by dumping its [data](#) to a file.

Parameters:

- ← *ima_* The image to save.
- ← *filename* the destination.

9.74 mln::io::fits Namespace Reference

Namespace of [fits](#) input/output handling.

Functions

- [image2d](#)< float > [load](#) (const std::string &filename)
Load a [fits](#) image in a [image2d](#)<float>.
- void [load](#) ([image2d](#)< float > &ima, const std::string &filename)
Load a [fits](#) image in a Milena image.

9.74.1 Detailed Description

Namespace of [fits](#) input/output handling.

9.74.2 Function Documentation

9.74.2.1 [image2d](#)< float > [mln::io::fits::load](#) (const std::string & *filename*) [inline]

Load a [fits](#) image in a [image2d](#)<float>.

Parameters:

← *filename* The image source.

Returns:

An [image2d](#)<float> which contains loaded [data](#).

9.74.2.2 void [mln::io::fits::load](#) ([image2d](#)< float > & *ima*, const std::string & *filename*) [inline]

Load a [fits](#) image in a Milena image.

Parameters:

→ *ima* A reference to the [image2d](#)<float> which will receive [data](#).

← *filename* The source.

9.75 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.

9.75.1 Detailed Description

Namespace of [pgm](#) input/output handling.

9.75.2 Function Documentation

9.75.2.1 `template<typename I> void mln::io::fld::load (Image< I > &ima_, const char *filename) [inline]`

Load an image from an AVS field file.

Parameters:

- ↔ *ima_* The image to load.
- ← *filename* The path to the AVS file.

References `mln::io::fld::fld_header::data`, `mln::io::fld::fld_header::max_ext`, `mln::io::fld::fld_header::min_ext`, `mln::io::fld::fld_header::ndim`, `mln::io::fld::fld_header::nspc`, `mln::box< P >::pmax()`, `mln::box< P >::pmin()`, `read_header()`, and `mln::io::fld::fld_header::veclen`.

9.75.2.2 `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.

References `mln::io::fld::fld_header::data`, `mln::io::fld::fld_header::dim`, `mln::io::fld::fld_header::field`, `mln::io::fld::fld_header::max_ext`, `mln::io::fld::fld_header::min_ext`, `mln::io::fld::fld_header::ndim`, `mln::io::fld::fld_header::nspace`, and `mln::io::fld::fld_header::veclen`.

Referenced by `load()`.

9.75.2.3 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.

References `mln::io::fld::fld_header::data`, `mln::io::fld::fld_header::dim`, `mln::io::fld::fld_header::field`, `mln::io::fld::fld_header::max_ext`, `mln::io::fld::fld_header::min_ext`, `mln::io::fld::fld_header::ndim`, `mln::io::fld::fld_header::nspace`, and `mln::io::fld::fld_header::veclen`.

9.76 mln::io::magick Namespace Reference

Namespace of [magick](#) input/output handling.

Functions

- bool [do_it](#) (const [value::rgb8](#) &in, bool &out, const std::string &filename)
- [Magick::Color](#) [get_color](#) (bool value)
- template<typename I>
void [load](#) ([Image](#)< I > &ima, const std::string &filename)
- template<typename I>
void [save](#) (const [Image](#)< I > &ima, const std::string &filename)

9.76.1 Detailed Description

Namespace of [magick](#) input/output handling.

9.76.2 Function Documentation

9.76.2.1 bool mln::io::magick::do_it (const [value::rgb8](#) & *in*, bool & *out*, const std::string & *filename*) [inline]

Load a [magick](#) image in a tiled image.

Parameters:

- *ima* A reference to the image which will receive [data](#).
- ← *filename* The source.

References [mln::value::rgb< n >::blue\(\)](#), [mln::value::rgb< n >::green\(\)](#), and [mln::value::rgb< n >::red\(\)](#).

Referenced by [load\(\)](#).

9.76.2.2 [Magick::Color](#) mln::io::magick::get_color (bool *value*) [inline]

Save a Milena tiled image in a [magick](#) image.

Parameters:

- *ima* A reference to the image to save.
- ← *filename* The output.

Referenced by [save\(\)](#).

9.76.2.3 template<typename I> void mln::io::magick::load ([Image](#)< I > & *ima*, const std::string & *filename*) [inline]

Load a [magick](#) image in a Milena image.

Parameters:

- *ima* A reference to the image which will receive [data](#).
- ← *filename* The source.

References [do_it\(\)](#), [mln::initialize\(\)](#), and [mln::point< G, C >::to_vec\(\)](#).

9.76.2.4 `template<typename I> void mln::io::magick::save (const Image< I > & ima, const std::string & filename) [inline]`

Save a Milena image in a [magick](#) image.

Parameters:

- *ima* A reference to the image to save.
- ← *filename* The output.

References [get_color\(\)](#), and [mln::point< G, C >::to_vec\(\)](#).

9.77 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'.

9.77.1 Detailed Description

Namespace of [off](#) input/output handling.

9.77.2 Function Documentation

9.77.2.1 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:

- *ima* A reference to the image to construct.
- ← *filename* The name of the file to load.

The image is said binary since [data](#) only represent the existence of faces.

Parameters:

- *ima* A reference to the image to construct.
- ← *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.

9.77.2.2 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:

← *ima* The image to save.

← *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:

← *ima* The image to save.

← *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.

9.77.2.3 `template<typename I> void mln::io::off::save_bin_alt (const I & ima, const std::string & filename) [inline]`

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 mln::io::pbm Namespace Reference

Namespace of [pbm](#) input/output handling.

Namespaces

- namespace [impl](#)
Namespace of [pbm](#) implementation details.

Functions

- [image2d](#)< bool > [load](#) (const std::string &filename)
Load a [pbm](#) image in a [image2d](#)<float>.
- void [load](#) ([image2d](#)< bool > &ima, const std::string &filename)
Load a [pbm](#) image in a Milena image.
- template<typename I>
void [save](#) (const [Image](#)< I > &ima, const std::string &filename)

9.78.1 Detailed Description

Namespace of [pbm](#) input/output handling.

9.78.2 Function Documentation

9.78.2.1 [image2d](#)< bool > [mln::io::pbm::load](#) (const std::string & *filename*) [inline]

Load a [pbm](#) image in a [image2d](#)<float>.

Parameters:

← *filename* The image source.

Returns:

An [image2d](#)<float> which contains loaded [data](#).

Load a [pbm](#) image in a [image2d](#)<float>.

Parameters:

← *filename* The image source.

Returns:

An [image2d](#)<float> which contains loaded [data](#).

9.78.2.2 `void mln::io::pbm::load (image2d< bool > & ima, const std::string & filename)`
[inline]

Load a [pbm](#) image in a Milena image.

Parameters:

- *ima* A reference to the image2d<bool> which will receive [data](#).
- ← *filename* The source.

9.78.2.3 `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:

- ← *ima* The image to save.
- ↔ *filename* the destination.

9.79 mln::io::pbm::impl Namespace Reference

Namespace of [pbm](#) implementation details.

9.79.1 Detailed Description

Namespace of [pbm](#) implementation details.

9.80 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.

9.80.1 Detailed Description

Namespace of [pbms](#) input/output handling.

9.80.2 Function Documentation

9.80.2.1 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:

- *ima* A reference to the 3D image which will receive [data](#).
- ← *filenames* The list of 2D images to load..

References [mln::io::pnms::load\(\)](#).

9.81 mln::io::pbms::impl Namespace Reference

Namespace of [pbms](#) implementation details.

9.81.1 Detailed Description

Namespace of [pbms](#) implementation details.

9.82 mln::io::pfm Namespace Reference

Namespace of [pfm](#) input/output handling.

Namespaces

- namespace [impl](#)
Implementation namespace of [pfm](#) namespace.

Functions

- [image2d](#)< float > [load](#) (const std::string &filename)
Load a [pfm](#) image in a [image2d](#)<float>.
- void [load](#) ([image2d](#)< float > &ima, const std::string &filename)
Load a [pfm](#) image in a Milena image.
- template<typename I>
void [save](#) (const [Image](#)< I > &ima, const std::string &filename)
Save a Milena image as a [pfm](#) image.

9.82.1 Detailed Description

Namespace of [pfm](#) input/output handling.

9.82.2 Function Documentation

9.82.2.1 [image2d](#)< float > [mln::io::pfm::load](#) (const std::string & *filename*) [inline]

Load a [pfm](#) image in a [image2d](#)<float>.

Parameters:

← *filename* The image source.

Returns:

An [image2d](#)<float> which contains loaded [data](#).

Load a [pfm](#) image in a [image2d](#)<float>.

Load a [pbm](#) image in a [image2d](#)<float>.

Parameters:

← *filename* The image source.

Returns:

An [image2d](#)<float> which contains loaded [data](#).

9.82.2.2 `void mln::io::pfm::load (image2d< float > & ima, const std::string & filename)`
[inline]

Load a [pfm](#) image in a Milena image.

Parameters:

- *ima* A reference to the image2d<float> which will receive [data](#).
- ← *filename* The source.

9.82.2.3 `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:

- ← *ima* The image to save.
- ↔ *filename* the destination.

9.83 mln::io::pfm::impl Namespace Reference

Implementation namespace of [pfm](#) namespace.

9.83.1 Detailed Description

Implementation namespace of [pfm](#) namespace.

9.84 mln::io::pgm Namespace Reference

Namespace of [pgm](#) input/output handling.

Functions

- `template<typename V>`
`image2d< V > load` (const std::string &filename)
Load a [pgm](#) image in a Milena image.
- `template<typename I>`
`void load (Image< I > &ima, 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)

9.84.1 Detailed Description

Namespace of [pgm](#) input/output handling.

9.84.2 Function Documentation

9.84.2.1 `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:

← *filename* The image source.

Returns:

An `image2d` which contains loaded [data](#).

Load a [pgm](#) image in a Milena image.

Load a [pfm](#) image in a `image2d<float>`.

Load a [pbm](#) image in a `image2d<float>`.

Parameters:

← *filename* The image source.

Returns:

An `image2d<float>` which contains loaded [data](#).

9.84.2.2 `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:

- *ima* A reference to the image which will receive [data](#).
- ← *filename* The source.

9.84.2.3 `template<typename I> void mln::io::pgm::save (const Image< I > & ima, const std::string & filename) [inline]`

Save a Milena image as a [pgm](#) image.

Parameters:

- ← *ima* The image to save.
- ↔ *filename* the destination.

References `mln::io::pnm::save()`.

9.85 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.

9.85.1 Detailed Description

Namespace of [pgms](#) input/output handling.

9.85.2 Function Documentation

- 9.85.2.1** `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:

- *ima* A reference to the 3D image which will receive [data](#).
- ← *filenames* The list of 2D images to load..

9.86 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 (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.

9.86.1 Detailed Description

Namespace of [plot](#) input/output handling.

9.86.2 Function Documentation

9.86.2.1 `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:

- ← *ima* A reference to the image to load.
- *filename* The output file.
- ← *start_value* The start index [value](#) of the [plot](#) (optional).

Load a Milena array from a [plot](#) file.

Parameters:

- ← *arr* A reference to the array to load.
- *filename* The output file.

References `mln::util::array< T >::append()`, and `mln::util::array< T >::clear()`.

9.86.2.2 `template<typename T> void mln::io::plot::save (util::array< T > &arr, const std::string &filename, int start_value = 0)` [inline]

Save a Milena array in a [plot](#) file.

Parameters:

- ← *arr* A reference to the array to save.
- *filename* The output file.
- ← *start_value* The start index [value](#) of the [plot](#) (optional).

9.86.2.3 `template<typename I> void mln::io::plot::save (const image1d< I > & ima, const std::string & filename) [inline]`

Save a Milena 1D image in a [plot](#) file.

Parameters:

- ← *ima* A reference to the image to save.
- *filename* The output file.

9.87 mln::io::pnm Namespace Reference

Namespace of [pnm](#) input/output handling.

Namespaces

- namespace [impl](#)
Namespace of [pnm](#)'s implementation details.

Functions

- `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 V>`
`image2d< V > load (char type_, const std::string &filename)`
main function : load [pnm](#) format
- `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)`

9.87.1 Detailed Description

Namespace of [pnm](#) input/output handling.

9.87.2 Function Documentation

- 9.87.2.1** `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.

References mln::make::box2d(), load_raw_2d(), and max_component().

9.87.2.2 `template<typename V> image2d<V> mln::io::pnm::load (char type_, const std::string & filename) [inline]`

main function : load [pnm](#) format

References load_raw_2d(), and max_component().

9.87.2.3 `template<typename I> void mln::io::pnm::load_ascii_builtin (std::ifstream & file, I & ima) [inline]`

load_ascii for builtin [value](#) types.

9.87.2.4 `template<typename I> void mln::io::pnm::load_ascii_value (std::ifstream & file, I & ima) [inline]`

load_ascii for Milena [value](#) types.

9.87.2.5 `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

Referenced by load().

9.87.2.6 `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.

Referenced by load().

9.87.2.7 `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:

- ← *type* The type of the image to save (can be PPM, PGM, PBM).
- ← *ima_* The image to save.
- ↔ *filename* the destination.

Referenced by mln::io::ppm::save(), and mln::io::pgm::save().

9.88 mln::io::pnm::impl Namespace Reference

Namespace of pnm's implementation details.

9.88.1 Detailed Description

Namespace of pnm's implementation details.

9.89 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.

9.89.1 Detailed Description

Namespace of [pnms](#) input/output handling.

9.89.2 Function Documentation

- 9.89.2.1** `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:

- ← *type* The type of the [pnm](#) files.
- *ima* A reference to the 3D image which will receive [data](#).
- ← *filenames* The list of 2D images to load..

References [mln::make::image3d\(\)](#), [mln::util::array< T >::is_empty\(\)](#), and [mln::util::array< T >::nelements\(\)](#).

Referenced by [mln::io::pbms::load\(\)](#).

9.90 mln::io::ppm Namespace Reference

Namespace of [ppm](#) input/output handling.

Functions

- `template<typename V>`
[image2d](#)< V > [load](#) (const std::string &filename)
Load a [ppm](#) image in a Milena image.
- `template<typename I>`
void [load](#) ([Image](#)< I > &ima, 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)

9.90.1 Detailed Description

Namespace of [ppm](#) input/output handling.

9.90.2 Function Documentation

9.90.2.1 `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:

← *filename* The image source.

Returns:

An [image2d](#) which contains loaded [data](#).

Load a [ppm](#) image in a Milena image.

Load a [pgm](#) image in a Milena image.

Load a [pfm](#) image in a [image2d](#)<float>.

Load a [pbm](#) image in a [image2d](#)<float>.

Parameters:

← *filename* The image source.

Returns:

An [image2d](#)<float> which contains loaded [data](#).

9.90.2.2 `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:

- *ima* A reference to the image which will receive [data](#).
- ← *filename* The source.

9.90.2.3 `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:

- ← *ima* The image to save.
- ↔ *filename* the destination.

References `mln::io::ppm::save()`.

Referenced by `mln::registration::icp()`.

9.91 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.

9.91.1 Detailed Description

Namespace of [ppms](#) input/output handling.

9.91.2 Function Documentation

9.91.2.1 `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:

- *ima* A reference to the 3D image which will receive [data](#).
- ← *filenames* The list of 2D images to load..

9.92 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.

9.92.1 Detailed Description

Namespace of [tiff](#) input/output handling.

9.92.2 Function Documentation

- 9.92.2.1** `template<typename I> void mln::io::tiff::load (Image< I > & ima_, const std::string & filename)` `[inline]`

Load a TIFF image to a Milena image.

9.93 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.

9.93.1 Detailed Description

Namespace of `txt` input/output handling.

9.93.2 Function Documentation

9.93.2.1 void `mln::io::txt::save` (const `image2d`< char > &`ima`, const std::string &`filename`) [inline]

Save an image as `txt` file.

Parameters:

- ← *`ima`* The image to save. Must be an image of char.
- ← *`filename`* the destination.

References `mln::image2d`< T >::domain().

9.94 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::array< typename A::result > > 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 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 L>`
`util::array< typename A::result > compute (const 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< 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 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)`
Compute an accumulator onto the [pixel](#) values of the image input.

- `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>`
`mln::trait::ch_value< L, mln_meta_accu_result(A, typename I::value) >::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 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 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 I, typename N, typename L>`
`I 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>`
`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, const Function_v2v< F > &fv2v)`
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, typename I::value &new_nlabels, const Function_v2b< F > &fv2b)`
Remove components and relabel a labeled image.
- `template<typename I, typename F>`
`void relabel_inplace (Image< I > &label, 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, 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>`
`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).
- `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).

9.94.1 Detailed Description

Namespace of [labeling](#) routines.

9.94.2 Function Documentation

- 9.94.2.1** `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:

- ← *input* The input image.
- ← *nbh* The connexity of the background.
- *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](#)

References `value()`.

Referenced by `fill_holes()`.

9.94.2.2 `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:

- ← *input* The input image.
- ← *nbh* The connexity of the objects.
- *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).

References `mln::canvas::labeling::blobs()`.

Referenced by `mln::graph::labeling()`.

9.94.2.3 `template<typename I, typename N, typename L, typename A> util::couple<
mln::trait::ch_value< I, L >::ret, util::array< typename A::result > >
mln::labeling::blobs_and_compute (const Image< I > & input, const Neighborhood< N
> & nbh, L & nlabels, const Accumulator< A > & accu) [inline]`

Label an image and compute given accumulators.

Parameters:

- ← *input* A binary image.
- ← *nbh* A neighborhood used for [labeling](#).
- ↔ *nlabels* The number of labels found.
- ← *accu* An accumulator to be computed while [labeling](#).

References [mln::canvas::labeling::blobs\(\)](#), and [mln::make::couple\(\)](#).

9.94.2.4 `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:

- ← *value* *value* type used in the returned image.
- ← *labeled_image* A labeled image (

See also:

[labeling::blobs](#)).

Parameters:

- ← *nlabels* Number of labels.

References [mln::literal::black](#), and [mln::data::transform\(\)](#).

9.94.2.5 `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:

- ← *a* A meta-accumulator.
- ← *label* The labeled image.
- ← *nlabels* The number of labels in `label`.

Returns:

A [mln::p_array](#) of accumulator result (one result per label).

References [compute\(\)](#).

9.94.2.6 `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:

- ← *a* An accumulator.
- ← *label* The labeled image.
- ← *nlabels* The number of labels in `label`.

Returns:

A [mln::p_array](#) of accumulator result (one result per label).

Compute an accumulator onto the [pixel](#) sites of each component domain of `label`.

Parameters:

- ← *a_* An accumulator.
- ← *label_* The labeled image.
- ← *nlabels* The number of labels in `label`.

Returns:

A [mln::p_array](#) of accumulator result (one result per label).

9.94.2.7 `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:

- ← *a* A meta-accumulator.
- ← *input* The input image.
- ← *label* The labeled image.
- ← *nlabels* The number of labels in `label`.

Returns:

A [mln::p_array](#) of accumulator result (one result per label).

References `compute()`.

9.94.2.8 `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:

- ← *a* An accumulator.
- ← *input_* The input image.
- ← *label_* The labeled image.
- ← *nlabels* The number of labels in `label_`.

Returns:

A [mln::p_array](#) of accumulator result (one result per label).

Compute an accumulator onto the [pixel](#) values of the image `input_`.

Parameters:

- ← *a_* An accumulator.
- ← *input_* The input image.
- ← *label_* The labeled image.
- ← *nlabels* The number of labels in `label_`.

Returns:

A [mln::p_array](#) of accumulator result (one result per label).

9.94.2.9 `template<typename A, typename I, typename L> util::array< typename A::result > mln::labeling::compute (util::array< A > & accus, 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:

- ← *a* An array of accumulator.
- ← *input_* The input image.
- ← *label_* The labeled image.
- ← *nlabels* The number of labels in `label_`.

Returns:

A [mln::p_array](#) of accumulator result (one result per label).

Compute an accumulator onto the [pixel](#) values of the image `input_`.

Parameters:

- ← *accu* An array of accumulators.
- ← *input_* The input image.
- ← *label_* The labeled image.
- ← *nlabels* The number of labels in `label`.

Returns:

A `mln::p_array` of accumulator result (one result per label).

Referenced by `compute()`, `compute_image()`, `fill_holes()`, `mln::make::p_edges_with_mass_centers()`, and `mln::make::p_vertices_with_mass_centers()`.

9.94.2.10 `template<typename A, typename I, typename L> mln::trait::ch_value< L, mln_meta_accu_result(A, typename I::value) >::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:

- ← *accu* The meta-accumulator.
- ← *input* The input image (values).
- ← *labels* The label image.
- ← *nlabels* The count of labels.

Returns:

The image where labels are replaced by the result of the accumulator.

References `compute()`.

9.94.2.11 `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:

- ← *accu* The accumulator.
- ← *input* The input image (values).
- ← *labels* The label image.
- ← *nlabels* The count of labels.

Returns:

The image where labels are replaced by the result of the accumulator.

References `compute()`.

9.94.2.12 `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) [inline]`

Compute an accumulator onto the [pixel](#) values of the image `input`.
for each component of the image `label`.

Parameters:

- ← *a* The [mln::p_array](#) of accumulator result.
- ← *input* The input image (values).
- ← *labels* The label image.
- ← *nlabels* The count of labels.

Returns:

The image where labels are replaced by the result of the accumulator.

9.94.2.13 `template<typename I, typename N, typename L> I 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:

- ← *input* The input image.
- ← *nbh* The connexity of the background.
- *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](#)

References [background\(\)](#), [compute\(\)](#), [mln::data::fill\(\)](#), [mln::initialize\(\)](#), and [mln::util::array< T >::nelements\(\)](#).

9.94.2.14 `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) [inline]`

Connected component [labeling](#) of the flat zones of an image.

Parameters:

- ← *input* The input image.
- ← *nbh* The connexity of the flat zones.
- *nlabels* The number of labels.

Returns:

The label image.

9.94.2.15 `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:

- ← *input* The input image.
- ← *nbh* The connexity of the foreground.
- *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](#)

References [value\(\)](#).

9.94.2.16 `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) [inline]`

Relabel a labeled image in order to have a contiguous [labeling](#).

Parameters:

- ← *label* The labeled image.
- *new_nlabels* The number of labels after relabeling.
- *repack_fun* The function used to repack the labels.

Returns:

The relabeled image.

References [mln::data::compute\(\)](#), [mln::make::relabelfun\(\)](#), and [mln::data::transform\(\)](#).

9.94.2.17 `template<typename I> void mln::labeling::pack_inplace (Image< I > & label, typename I::value & new_nlabels, fun::i2v::array< typename I::value > & repack_fun) [inline]`

Relabel inplace a labeled image in order to have a contiguous [labeling](#).

Parameters:

- ← *label* The labeled image.
- *new_nlabels* The number of labels after relabeling.
- *repack_fun* The function used to repack the labels.

References `mln::data::compute()`, `mln::make::relabelfun()`, and `mln::data::transform()`.

9.94.2.18 `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) [inline]`

Connected component [labeling](#) of the regional maxima of an image.

Parameters:

- ← *input* The input image.
- ← *nbh* The connexity of the regional maxima.
- *nlabels* The number of labeled regions.

Returns:

The label image.

9.94.2.19 `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) [inline]`

Connected component [labeling](#) of the regional minima of an image.

Parameters:

- ← *input* The input image.
- ← *nbh* The connexity of the regional minima.
- *nlabels* The number of labeled regions.

Returns:

The label image.

Referenced by `mln::morpho::meyer_wst()`.

9.94.2.20 `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:

- ← *label* the labeled image.
- ← *nlabels* the number of labels in `label`.
- ← *fv2v* function returning the new component id for each [pixel value](#).

Returns:

the relabeled image.

References `mln::data::transform()`.

9.94.2.21 `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:

- ← *label* the labeled image.
- ← *nlabels* the number of labels in `label`.
- *new_nlabels* the number of labels after relabeling.
- ← *fv2b* function returning whether a label must be replaced by the background.

Returns:

the relabeled image.

References `mln::make::relabelfun()`.

Referenced by `superpose()`.

9.94.2.22 `template<typename I, typename F> void mln::labeling::relabel_inplace (Image< I > &
label, typename I::value & nlabels, const Function_v2v< F > & fv2v) [inline]`

Remove components and relabel a labeled image inplace.

Parameters:

- ↔ *label* the labeled image.
- ↔ *nlabels* the number of labels in `label`.
- ← *fv2v* function returning the new component id for each [pixel value](#).

References `mln::data::transform_inplace()`.

9.94.2.23 `template<typename I, typename F> void mln::labeling::relabel_inplace (Image< I > & label, typename I::value & nlabels, const Function_v2b< F > & fv2b) [inline]`

Remove components and relabel a labeled image inplace.

Parameters:

- ↔ *label* the labeled image.
- ↔ *nlabels* the number of labels in *label*.
- ← *fv2b* function returning whether a label must be replaced by the background.

References `mln::make::relabelfun()`.

Referenced by `mln::labeled_image_base< I, E >::relabel()`.

9.94.2.24 `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) [inline]`

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:

- ← *lhs* A labeled image.
- ← *lhs_nlabels* The number of labels in *lhs*.
- ← *rhs* A labeled image.
- ← *rhs_nlabels* The number of labels in *rhs*.
- *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.

References `mln::duplicate()`, `mln::data::paste()`, `relabel()`, and `mln::literal::zero`.

9.94.2.25 `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) [inline]`

Connected component *labeling* of the image sites at a given *value*.

Parameters:

- ← *input* The input image.

- ← *val* The [value](#) to consider.
- ← *nbh* The connectivity of components.
- *nlabels* The number of labels.

Returns:

The label image.

Referenced by `background()`, and `foreground()`.

9.94.2.26 `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:

- ← *input* The label image.

Returns:

A new image with values wrapped with type `label_8`.

References `wrap()`.

9.94.2.27 `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:

- ← *value_type* The type used to wrap the label type.
- ← *input* The label image.

Returns:

A new image with values wrapped with type `V`.

References `mln::data::transform()`.

Referenced by `wrap()`.

9.95 mln::labeling::impl Namespace Reference

Implementation namespace of [labeling](#) namespace.

Namespaces

- namespace [generic](#)
Generic implementation namespace of [labeling](#) namespace.

9.95.1 Detailed Description

Implementation namespace of [labeling](#) namespace.

9.96 mln::labeling::impl::generic Namespace Reference

Generic implementation namespace of [labeling](#) namespace.

Functions

- `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](#).
- `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](#).

9.96.1 Detailed Description

Generic implementation namespace of [labeling](#) namespace.

9.96.2 Function Documentation

9.96.2.1 `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](#).

Compute an accumulator onto the [pixel](#) values of the image `input`.

Parameters:

- ← *accus* An array of accumulators.
- ← *input_* The input image.
- ← *label_* The labeled image.
- ← *nlabels* The number of labels in `label`.

Returns:

A `mln::p_array` of accumulator result (one result per label).

Referenced by `mln::labeling::compute()`, `mln::labeling::compute_image()`, `mln::labeling::fill_holes()`, `mln::make::p_edges_with_mass_centers()`, and `mln::make::p_vertices_with_mass_centers()`.

9.96.2.2 `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](#).

Compute an accumulator onto the [pixel](#) values of the image `input`.

Parameters:

- ← `a_` An accumulator.
- ← `input_` The input image.
- ← `label_` The labeled image.
- ← `nlabels` The number of labels in `label`.

Returns:

A [mln::p_array](#) of accumulator result (one result per label).

9.96.2.3 `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](#).

Compute an accumulator onto the [pixel](#) sites of each component domain of `label`.

Parameters:

- ← `a_` An accumulator.
- ← `label_` The labeled image.
- ← `nlabels` The number of labels in `label`.

Returns:

A [mln::p_array](#) of accumulator result (one result per label).

9.97 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, int dir)`
- `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_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>`
`mln_ch_convolve (I, W) convolve(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.

9.97.1 Detailed Description

Namespace of [linear](#) image processing routines.

9.97.2 Function Documentation

9.97.2.1 `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)
```

References `mln::initialize()`.

9.97.2.2 `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
```

References `mln::initialize()`.

Referenced by `mln::subsampling::gaussian_subsampling()`.

9.97.2.3 `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
```

References `mln::initialize()`.

9.97.2.4 `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)
```

References `mln::initialize()`.

9.97.2.5 `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
```

References `mln::initialize()`.

9.97.2.6 `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)
```

References `mln::initialize()`.

9.97.2.7 `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.

References `mln_ch_convolve()`, and `mln::make::w_window2d()`.

9.97.2.8 `template<typename I, typename W> mln::linear::mln_ch_convolve (I, W) const [inline]`

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

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

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

Referenced by `mln_ch_convolve()`, and `mln_ch_convolve_grad()`.

9.97.2.9 `template<typename I> mln::linear::mln_ch_convolve_grad (I, int) const` `[inline]`

Compute the vertical component of the 2D Sobel gradient.

References `mln_ch_convolve()`, and `mln::data::transform()`.

9.98 mln::linear::impl Namespace Reference

Namespace of [linear](#) image processing routines implementation details.

9.98.1 Detailed Description

Namespace of [linear](#) image processing routines implementation details.

9.99 mln::linear::local Namespace Reference

Specializations of [local linear](#) routines.

Namespaces

- namespace [impl](#)
Namespace of [local linear](#) routines implementation details.

Functions

- `template<typename P, typename W, typename R>`
`void convolve (const Generalized_Pixel< P > &p, const Weighted_Window< W > &w_win, R &result)`
- `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)`

9.99.1 Detailed Description

Specializations of [local linear](#) routines.

9.99.2 Function Documentation

9.99.2.1 `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.

References `convolve()`.

9.99.2.2 `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.

Referenced by `convolve()`.

9.100 mln::linear::local::impl Namespace Reference

Namespace of [local linear](#) routines implementation details.

9.100.1 Detailed Description

Namespace of [local linear](#) routines implementation details.

9.101 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.

9.101.1 Detailed Description

Namespace of literals.

9.101.2 Variable Documentation

9.101.2.1 `const black_t & mln::literal::black = black_t()`

Literal black.

Referenced by `mln::labeling::colorize()`, and `mln::registration::icp()`.

9.101.2.2 `const blue_t & mln::literal::blue = blue_t()`

Literal blue.

9.101.2.3 `const brown_t & mln::literal::brown = brown_t()`

Literal brown.

9.101.2.4 `const cyan_t & mln::literal::cyan = cyan_t()`

Literal cyan.

9.101.2.5 `const dark_gray_t & mln::literal::dark_gray = dark_gray_t()`

Literal dark gray.

9.101.2.6 `const green_t & mln::literal::green = green_t()`

Literal green.

Referenced by `mln::registration::icp()`, and `mln::make_debug_graph_image()`.

9.101.2.7 `const identity_t & mln::literal::identity = identity_t()`

Literal identity.

9.101.2.8 `const light_gray_t & mln::literal::light_gray = light_gray_t()`

Literal light gray.

9.101.2.9 `const lime_t & mln::literal::lime = lime_t()`

[Literal](#) lime.

9.101.2.10 `const magenta_t & mln::literal::magenta = magenta_t()`

[Literal](#) magenta.

9.101.2.11 `const max_t & mln::literal::max = max_t()`

[Literal](#) max.

9.101.2.12 `const medium_gray_t & mln::literal::medium_gray = medium_gray_t()`

[Literal](#) medium_gray.

9.101.2.13 `const min_t & mln::literal::min = min_t()`

[Literal](#) min.

9.101.2.14 `const olive_t & mln::literal::olive = olive_t()`

[Literal](#) olive.

9.101.2.15 `const one_t & mln::literal::one = one_t()`

[Literal](#) one.

Referenced by `mln::algebra::h_vec< d, C >::h_vec()`, `mln::operator++()`, and `mln::operator--()`.

9.101.2.16 `const orange_t & mln::literal::orange = orange_t()`

[Literal](#) orange.

9.101.2.17 `const origin_t & mln::literal::origin = origin_t()`

[Literal](#) origin.

Referenced by `mln::win::ball< G, C >::ball()`, `mln::geom::bbox()`, `mln::box< P >::box()`, `mln::geom::rotate()`, and `mln::make::w_window()`.

9.101.2.18 `const pink_t & mln::literal::pink = pink_t()`

[Literal](#) pink.

9.101.2.19 `const purple_t & mln::literal::purple = purple_t()`

[Literal](#) purple.

9.101.2.20 `const red_t & mln::literal::red = red_t()`

[Literal](#) red.

Referenced by `mln::morpho::watershed::superpose()`.

9.101.2.21 `const teal_t & mln::literal::teal = teal_t()`

[Literal](#) teal.

9.101.2.22 `const violet_t & mln::literal::violet = violet_t()`

[Literal](#) violet.

9.101.2.23 `const white_t & mln::literal::white = white_t()`

[Literal](#) white.

Referenced by `mln::registration::icp()`.

9.101.2.24 `const yellow_t & mln::literal::yellow = yellow_t()`

[Literal](#) yellow.

9.101.2.25 `const zero_t & mln::literal::zero = zero_t()`

[Literal](#) zero.

Referenced by `mln::morpho::impl::generic::hit_or_miss()`, `mln::accu::shape::volume< I >::init()`, `mln::morpho::attribute::sum< I, S >::init()`, `mln::accu::math::sum< T, S >::init()`, `mln::accu::rms< T, V >::init()`, `mln::accu::convolve< T1, T2, R >::init()`, `mln::accu::center< P, V >::init()`, `mln::window< D >::is_centered()`, `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::geom::impl::seeds2tiling_roundness()`, `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()`.

9.102 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)`

9.102.1 Detailed Description

Namespace of logic.

9.102.2 Function Documentation

9.102.2.1 `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:

- ↔ *lhs* First operand image.
- ← *rhs* Second operand image.

It performs:

for all `p` of `rhs.domain`

`lhs(p) = lhs(p) and rhs(p)`

Precondition:

`rhs.domain >= lhs.domain`

References `mln::data::transform_inplace()`.

9.102.2.2 `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:

- ← *lhs* First operand image.
- ← *rhs* Second operand image.

Returns:

The result image.

Precondition:

```
lhs.domain == rhs.domain
```

References `mln::data::transform()`.

9.102.2.3 `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:

- ↔ *lhs* First operand image.
- ← *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
```

References `mln::data::transform_inplace()`.

9.102.2.4 `template<typename I> void mln::logical::not_inplace (Image< I > & input) [inline]`

Point-wise in-place "logical not" of image *input*.

Parameters:

- ↔ *input* The target image.

It performs:

for all *p* of *input.domain*

input(*p*) = not *input*(*p*)

Precondition:

```
input.is_valid
```

References mln::data::transform_inplace().

9.102.2.5 `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:

- ↔ *lhs* First operand image.
- ← *rhs* Second operand image.

It performs:

for all p of rhs.domain

lhs(p) = lhs(p) or rhs(p)

Precondition:

```
rhs.domain >= lhs.domain
```

References mln::data::transform_inplace().

9.102.2.6 `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:

- ↔ *lhs* First operand image.
- ← *rhs* Second operand image.

It performs:

for all p of rhs.domain

lhs(p) = lhs(p) xor rhs(p)

Precondition:

```
rhs.domain >= lhs.domain
```

References mln::data::transform_inplace().

9.103 mln::logical::impl Namespace Reference

Implementation namespace of [logical](#) namespace.

Namespaces

- namespace [generic](#)
Generic implementation namespace of [logical](#) namespace.

9.103.1 Detailed Description

Implementation namespace of [logical](#) namespace.

9.104 mln::logical::impl::generic Namespace Reference

Generic implementation namespace of [logical](#) namespace.

9.104.1 Detailed Description

Generic implementation namespace of [logical](#) namespace.

9.105 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_psite< D, G > > attachment` (const `complex_psite< 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` (`def::coord` min_row, `def::coord` min_col, `def::coord` max_row, `def::coord` max_col)
Create an `mln::box2d`.
- `mln::box2d box2d` (unsigned nrows, unsigned ncols)
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` (`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`.
- `mln::box3d box3d` (unsigned nslis, unsigned nrows, unsigned ncols)
Create an `mln::box3d`.
- `template<unsigned D, typename G>`
`p_set< complex_psite< D, G > > cell` (const `complex_psite< 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_psite< D, G > > detachment` (const `complex_psite< 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>`
`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_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_vertices< G > dummy_p_vertices (const Graph< G > &g)`
Create a `p_vertices` 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 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 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 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 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 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 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 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 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 R, unsigned C>`
`mln::image2d< V > image (V(&values)[R][C])`
Create an `image2d` from an 2D array of values.
- `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 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 Image< 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 Image< 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_v2v< F > &fv2v, const V &nlabels, V &new_nlabels)`
Create a `i2v` function from a `v2v` function.
- `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 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 T>`
`algebra::vec< 1, T > vec (const T &v_0)`
Create an `mln::algebra::vec<n,T>`.

- `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 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 I, typename N>`
`p_vertices< util::graph, fun::i2v::array< typename I::site > > voronoi (Image< I > &ima_, Im-`
`age< I > &orig_, const Neighborhood< N > &nbh)`
Apply the Voronoi algorithm on ima_ with the original image orig_ for node computing with neighbor-
hood 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.

9.105.1 Detailed Description

Namespace of routines that help to [make](#) Milena's objects.

9.105.2 Function Documentation

9.105.2.1 `template<unsigned D, typename G, typename V> p_set< complex_psite< D, G > >
mln::make::attachment (const complex_psite< 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`.

References `cell()`, and `mln::topo::is_facet()`.

Referenced by `mln::topo::is_simple_cell< I >::operator()`.

9.105.2.2 `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:

← *min_ind* Minimum index.
← *max_ind* Maximum index.

Precondition:

`max_ind >= min_ind`.

Returns:

A 1D [box](#).

9.105.2.3 `mln::box1d mln::make::box1d (unsigned ninds) [inline]`

Create an [mln::box1d](#).

Parameters:

← *ninds* Number of indices.

Precondition:

`ninds != 0 and ncols != 0.`

Returns:

A 1D [box](#).

Referenced by `mln::image1d< T >::image1d()`.

9.105.2.4 `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:

- ← *min_row* Index of the top most row.
- ← *min_col* Index of the left most column.
- ← *max_row* Index of the botton most row.
- ← *max_col* Index of the right most column.

Precondition:

`max_row >= min_row and max_col >= min_col.`

Returns:

A 2D [box](#).

9.105.2.5 `mln::box2d mln::make::box2d (unsigned nrows, unsigned ncols)` `[inline]`

Create an [mln::box2d](#).

Parameters:

- ← *nrows* Number of rows.
- ← *ncols* Number of columns.

Precondition:

`nrows != 0 and ncols != 0.`

Returns:

A 2D [box](#).

Referenced by `mln::image2d< T >::image2d()`, and `mln::io::pnm::load()`.

9.105.2.6 `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:

- ← *min_row* Index of the top most row.
- ← *min_col* Index of the left most column.
- ← *max_row* Index of the bottom most row.
- ← *max_col* Index of the right most column.

Precondition:

`max_row >= min_row and max_col >= min_col.`

Returns:

A [2D_H box](#).

References `point2d_h()`.

9.105.2.7 `mln::box2d_h mln::make::box2d_h (unsigned nrows, unsigned ncols)` `[inline]`

Create an [mln::box2d_h](#).

Parameters:

- ← *nrows* Number of rows.
- ← *ncols* Number of columns.

Precondition:

`nrows != 0 and ncols != 0.`

Returns:

A [2D_H box](#).

References `point2d_h()`.

9.105.2.8 `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:

- ← *min_sli* Index of the lowest slice.

- ← *min_row* Index of the top most row.
- ← *min_col* Index of the left most column.
- ← *max_sli* Index of the highest slice.
- ← *max_row* Index of the bottom most row.
- ← *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](#).

9.105.2.9 `mln::box3d mln::make::box3d (unsigned nslis, unsigned nrows, unsigned ncols)` [inline]

Create an [mln::box3d](#).

Parameters:

- ← *nslis* Number of slices.
- ← *nrows* Number of rows.
- ← *ncols* Number of columns.

Precondition:

```
ninds != 0 and ncols != 0 and nslis != 0.
```

Returns:

A 3D [box](#).

Referenced by `image3d()`, and `mln::image3d< T >::image3d()`.

9.105.2.10 `template<unsigned D, typename G> p_set< complex_site< D, G > > mln::make::cell (const complex_site< 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.

References `mln::topo::is_facet()`, and `mln::complex_site< D, G >::n()`.

Referenced by `attachment()`, and `detachment()`.

9.105.2.11 `template<typename T, typename U> util::couple<T,U> mln::make::couple (const T & val1, const T & val2) [inline]`

Construct an `mln::util::couple` on-the-fly.

Referenced by `mln::labeling::blobs_and_compute()`, `mln::transform::distance_and_closest_point_geodesic()`, and `mln::transform::distance_and_influence_zone_geodesic()`.

9.105.2.12 `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 `couple.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.

References `cell()`, and `mln::topo::is_facet()`.

Referenced by `mln::topo::detach()`.

9.105.2.13 `mln::dpoint2d_h mln::make::dpoint2d_h (def::coord row, def::coord col) [inline]`

Create an `mln::dpoint2d_h`.

Parameters:

← *row* Row coordinate.
 ← *col* Column coordinate.

Returns:

A 2D `dpoint`.

9.105.2.14 `template<typename G> p_edges< G > mln::make::dummy_p_edges (const Graph< G > & g) [inline]`

Create a `p_edges` which associate a `graph` element to a constant site.

0 (int) is used as dummy site.

Parameters:

← *g* A `graph`.

Returns:

A [p_edges](#).

9.105.2.15 `template<typename G, typename P> p_edges< G, pw::cst_< P > >`
`mln::make::dummy_p_edges (const Graph< G > & g_, const P & dummy_site)`
`[inline]`

Create a [p_edges](#) which associate a [graph](#) element to a constant site.

Parameters:

← *g_* A [graph](#).

← *dummy_site* The dummy site mapped to [graph](#) edges.

Returns:

A [p_edges](#).

9.105.2.16 `template<typename G> p_vertices< G > mln::make::dummy_p_vertices (const`
`Graph< G > & g) [inline]`

Create a [p_vertices](#) which associate a [graph](#) element to a constant site.

0 (int) is used as dummy site.

Parameters:

← *g* A [graph](#).

Returns:

A [p_vertices](#).

9.105.2.17 `template<typename G, typename P> p_vertices< G, pw::cst_< P > >`
`mln::make::dummy_p_vertices (const Graph< G > & g_, const P & dummy_site)`
`[inline]`

Create a [p_vertices](#) which associate a [graph](#) element to a constant site.

Parameters:

← *g_* A [graph](#).

← *dummy_site* The dummy site mapped to [graph](#) vertices.

Returns:

A [p_vertices](#).

9.105.2.18 `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:

- ← *v_ima_* A vertex image.
- ← *fv_* A function mapping a vertex ids to a [value](#). The result is associated to the corresponding edge.

Returns:

an edge image without localization information mapped to [graph](#) elements.

References mln::data::fill().

9.105.2.19 `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:

- ← *v_ima_* A vertex image.
- ← *fv_* A function mapping two vertices ids to a [value](#). The result is associated to the corresponding edge.

Returns:

an edge image without localization information mapped to [graph](#) elements.

9.105.2.20 `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:

- ← *v_ima_* A vertex image.
- ← *pe* A [p_edges](#) mapping [graph](#) element to sites .
- ← *fv_* A function mapping two vertex ids to a [value](#). The result is associated to the corresponding edge.

Returns:

an edge image.

9.105.2.21 `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:

- ← *g_* A [graph](#)
- ← *fp* A function mapping edge ids to sites.
- ← *fv* A function mapping edge ids to values.

Returns:

an edge image.

9.105.2.22 `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) [inline]`

Construct an edge image.

Parameters:

- ← *g* A [graph](#)
- ← *fv* A function mapping edge ids to values.

Returns:

an edge image.

9.105.2.23 `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:

- ← *g* A [graph](#)
- ← *fv* A function mapping edge ids to values.

Returns:

an edge image.

9.105.2.24 `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>`.

Referenced by `mln::fun::x2x::rotation< n, C >::rotation()`.

9.105.2.25 `template<typename V, unsigned S, unsigned R, unsigned C> mln::image3d< V >
mln::make::image (V(&) values[S][R][C]) [inline]`

Create an [image3d](#) from an 3D array of values.

Parameters:

← *values* 3D array.

Returns:

A 3D image.

References `mln::opt::at()`.

9.105.2.26 `template<typename V, unsigned R, unsigned C> mln::image2d< V >
mln::make::image (V(&) values[R][C]) [inline]`

Create an [image2d](#) from an 2D array of values.

Parameters:

← *values* 2D array.

Returns:

A 2D image.

References `mln::opt::at()`.

9.105.2.27 `template<typename V, unsigned L> mln::image1d< V > mln::make::image (V(&)
values[L]) [inline]`

Create an [image1d](#) from an 1D array of values.

Parameters:

← *values* 1D array.

Returns:

A 1D image.

9.105.2.28 `template<typename V, unsigned S> mln::image2d< V > mln::make::image2d (V(&)
values[S]) [inline]`

Create an [image2d](#) from an 2D array of values.

Parameters:

← *values* 2D array.

Returns:

A 2D image.

9.105.2.29 `template<typename I> mln::image3d< typename I::value > mln::make::image3d(const Image< I > & ima) [inline]`

Create an [image3d](#) from a 2D image.

References [box3d\(\)](#), and [mln::data::paste\(\)](#).

9.105.2.30 `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.

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\(\)](#).

9.105.2.31 `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:

- ← *iz* influence zone image.
- ← *nbh* A neighborhood.
- ← *nlabels* number of influence zone in *iz*.

Returns:

[util::graph Graph](#) based on the adjacency of the influence zones.

Create a [graph](#) from an influence zone image.

Parameters:

- ← *iz_* influence zone image.
- ← *nbh_* A neighborhood.
- ← *nlabels* number of influence zone in *iz*.

Returns:

[util::graph Graph](#) based on the adjacency of the influence zones.

9.105.2.32 `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:

- ← *tab* Array of values.

Precondition:

The array dimension has to be $n * m$.

9.105.2.33 `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.

References `ord_pair()`.

Referenced by `ord_pair()`.

9.105.2.34 `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`

References `mln::labeling::compute()`.

9.105.2.35 `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`

References `mln::labeling::compute()`.

9.105.2.36 `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:

- ← *ima* The input image.
- ← *p* The `point` site.

Returns:

An `mln::util::pix`.

9.105.2.37 `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`.

9.105.2.38 `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`.

9.105.2.39 `mln::point2d_h mln::make::point2d_h (def::coord row, def::coord col)` `[inline]`

Create an `mln::point2d_h`.

Parameters:

- ← *row* Row coordinate.
- ← *col* Column coordinate.

Returns:

A 2D `point`.

Referenced by `box2d_h()`.

9.105.2.40 `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:

- ← *wshd_* Watershed image.
- ← *nbh_* `Neighborhood`
- ← *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)

9.105.2.41 `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:

- ← *wshd_* watershed image.
- ← *nbh* A neighborhood.
- ← *nbasins* number of influence zone in wshd.

Returns:

[util::graph](#) [Graph](#) based on the adjacency of the influence zones.

9.105.2.42 `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:

- ← *fv2v* A v2v function. This function maps an id to an already existing one.
- ← *nlabels* The number of labels.
- ← *new_nlabels* The number of labels after relabeling.

Returns:

a i2v function.

See also:

[mln::labeling::relabel](#)

References mln::literal::zero.

9.105.2.43 `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:

← *fv2b* A v2b function.

← *nlabels* The number of labels.

← *new_nlabels* The number of labels after relabeling.

Returns:

a i2v function.

See also:

[mln::labeling::relabel](#)

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().

9.105.2.44 `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:

← *v_0* First coordinate.

← *v_1* Second coordinate.

← *v_2* Third coordinate.

← *v_3* Fourth coordinate.

Returns:

A 4D vector.

9.105.2.45 `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:

- ← `v_0` First coordinate.
- ← `v_1` Second coordinate.
- ← `v_2` Third coordinate.

Returns:

A 3D vector.

9.105.2.46 `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:

- ← `v_0` First coordinate.
- ← `v_1` Second coordinate.

Returns:

A 2D vector.

9.105.2.47 `template<typename T> algebra::vec< 1, T > mln::make::vec (const T & v_0) [inline]`

Create an `mln::algebra::vec<n,T>`.

Parameters:

- ← `v_0` First coordinate.

Returns:

A 1D vector.

9.105.2.48 `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) [inline]`

Construct a vertex image.

Parameters:

- ← `g_` A [graph](#).

- ← *fp* A function mapping vertex ids to sites.
- ← *fv* A function mapping vertex ids to values.

Returns:

A vertex image.

9.105.2.49 `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) [inline]`

Construct a vertex image.

Parameters:

- ← *g* A [graph](#).
- ← *fv* A function mapping vertex ids to values.

Returns:

A vertex image.

9.105.2.50 `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:

- ← *ima_* The [labeling](#) image.
- ← *orig_* The original image.
- ← *nbh* The neighborhood for computing algorithm.

Returns:

The computed [graph](#).

References `mln::util::graph::add_edge()`, `mln::util::graph::add_vertex()`, and `mln::estim::min_max()`.

9.105.2.51 `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:

- ← *win* A simple [window](#).
- ← *wei* A weight function.

Returns:

A weighted [window](#).

References `mln::w_window< D, W >::insert()`, and `mln::literal::origin`.

9.105.2.52 `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:

← *weights* Array.

Precondition:

The array size, M, has to be a square of an odd integer.

Returns:

A 1D weighted [window](#).

References `mln::w_window< D, W >::insert()`.

Referenced by `w_window1d_int()`.

9.105.2.53 `template<unsigned M> mln::w_window1d_int mln::make::w_window1d_int (int(&)
weights[M])` `[inline]`

Create a [mln::w_window1d_int](#).

Parameters:

← *weights* Array of integers.

Precondition:

The array size, M, has to be a square of an odd integer.

Returns:

A 1D int-weighted [window](#).

References `w_window1d()`.

9.105.2.54 `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:

← *weights* Array.

Precondition:

The array size, S , has to be a square of an odd integer.

Returns:

A 2D weighted [window](#).

Referenced by `mln::linear::mln_ch_convolve()`, and `w_window2d_int()`.

9.105.2.55 `template<unsigned M> mln::w_window2d_int mln::make::w_window2d_int (int(&)
weights[M]) [inline]`

Create a [mln::w_window2d_int](#).

Parameters:

← *weights* Array of integers.

Precondition:

The array size, M , has to be a square of an odd integer.

Returns:

A 2D int-weighted [window](#).

References `w_window2d()`.

9.105.2.56 `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:

← *weights* Array.

Precondition:

The array size, M , has to be a cube of an odd integer.

Returns:

A 3D weighted [window](#).

References `mln::w_window< D, W >::insert()`.

Referenced by `w_window3d_int()`.

9.105.2.57 `template<unsigned M> mln::w_window3d_int mln::make::w_window3d_int (int(&)
weights[M]) [inline]`

Create a [mln::w_window3d_int](#).

Parameters:

← *weights* Array of integers.

Precondition:

The array size, *M*, has to be a cube of an odd integer.

Returns:

A 3D int-weighted [window](#).

References `w_window3d()`.

9.105.2.58 `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:

← *dp* A delta-point to [set](#) the orientation.

← *weights* An array of weights.

Returns:

A weighted [window](#).

The [window](#) length *L* has to be odd.

References `mln::w_window< D, W >::insert()`, and `mln::literal::zero`.

9.106 mln::math Namespace Reference

Namespace of mathematical routines.

Functions

- `template<unsigned n>`
`value::int_u< n > abs (const value::int_u< n > &v)`
Specialization for `mln::value::int_u`.
- `template<typename T>`
`T abs (const T &v)`
Generic version.
- `int abs (int v)`
Specializations for existing overloads of `std::abs`.

9.106.1 Detailed Description

Namespace of mathematical routines.

9.106.2 Function Documentation

9.106.2.1 `template<unsigned n> value::int_u< n > mln::math::abs (const value::int_u< n > &v)` `[inline]`

Specialization for `mln::value::int_u`.

9.106.2.2 `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]`).

9.106.2.3 `template<typename T> T mln::math::abs (const T &v)` `[inline]`

Generic version.

Referenced by `mln::morpho::line_gradient()`.

9.107 mln::metal Namespace Reference

Namespace of meta-programming tools.

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.

Namespaces

- namespace [impl](#)
Implementation namespace of [metal](#) namespace.
- namespace [math](#)
Namespace of static mathematical functions.

9.107.1 Detailed Description

Namespace of meta-programming tools.

9.108 mln::metal::impl Namespace Reference

Implementation namespace of [metal](#) namespace.

9.108.1 Detailed Description

Implementation namespace of [metal](#) namespace.

9.109 mln::metal::math Namespace Reference

Namespace of static mathematical functions.

Namespaces

- namespace [impl](#)
Implementation namespace of [metal::math](#) namespace.

9.109.1 Detailed Description

Namespace of static mathematical functions.

9.110 `mln::metal::math::impl` Namespace Reference

Implementation namespace of [metal::math](#) namespace.

9.110.1 Detailed Description

Implementation namespace of [metal::math](#) namespace.

9.111 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)`
Meyer's Watershed Transform (WST) algorithm, with no count of basins.

- 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 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 Window< 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).

9.111.1 Detailed Description

Namespace of mathematical morphology routines.

9.111.2 Function Documentation

9.111.2.1 `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).

Referenced by `hit_or_miss_background_closing()`, `hit_or_miss_background_opening()`, `hit_or_miss_closing()`, and `thinning()`.

9.111.2.2 `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).

9.111.2.3 `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 $Id + wth_B - bth_B$.

References `mln::arith::plus()`, `top_hat_black()`, and `top_hat_white()`.

9.111.2.4 `template<typename I, typename W> mln::trait::concrete< I >::ret mln::morpho::dilation (const Image< I > & input, const Window< W > & win) [inline]`

Morphological dilation.

References `general()`.

Referenced by `gradient()`, `gradient_external()`, `mln::morpho::impl::generic::hit_or_miss()`, `hit_or_miss_background_opening()`, `hit_or_miss_opening()`, `laplacian()`, `mln::morpho::opening::approx::structural()`, and `mln::morpho::closing::approx::structural()`.

9.111.2.5 `template<typename I, typename W> mln::trait::concrete< I >::ret mln::morpho::erosion (const Image< I > & input, const Window< W > & win) [inline]`

Morphological erosion.

References `general()`.

Referenced by `gradient()`, `gradient_internal()`, `mln::morpho::impl::generic::hit_or_miss()`, `laplacian()`, `mln::morpho::opening::approx::structural()`, and `mln::morpho::closing::approx::structural()`.

9.111.2.6 `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.

Referenced by dilation(), and erosion().

9.111.2.7 `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 $d_B - e_B$.

References dilation(), erosion(), minus(), and mln::test::positive().

9.111.2.8 `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 $d_B - Id$.

References dilation(), minus(), and mln::test::positive().

9.111.2.9 `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 $Id - e_B$.

References erosion(), minus(), and mln::test::positive().

9.111.2.10 `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 $HMT_(B_h, B_m) = e_{B_h} \wedge (e_{B_m} \circ C)$.

References dilation(), erosion(), mln::data::fill(), mln::initialize(), and mln::literal::zero.

Referenced by thickening(), and thinning().

9.111.2.11 `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 $C \circ \text{HMTopeBG} \circ C$.

References `complementation()`, `hit_or_miss_background_opening()`, and `hit_or_miss_closing()`.

9.111.2.12 `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 $\text{HMTopeBG} = \text{HMTope}_{(Bm, Bh)} \circ C = d_{(-Bm)} \circ \text{HMT}_{(Bh, Bm)}$.

References `complementation()`, `dilation()`, `hit_or_miss_opening()`, and `mln::win::sym()`.

Referenced by `hit_or_miss_background_closing()`, and `thick_miss()`.

9.111.2.13 `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 $C \circ \text{HMTope} \circ C$.

References `complementation()`, and `hit_or_miss_opening()`.

Referenced by `hit_or_miss_background_closing()`.

9.111.2.14 `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 $\text{HMTope}_{(Bh, Bm)} = d_{(-Bh)} \circ \text{HMT}_{(Bh, Bm)}$.

References `dilation()`, and `mln::win::sym()`.

Referenced by `hit_or_miss_background_opening()`, `hit_or_miss_closing()`, and `thin_fit()`.

9.111.2.15 `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 $(d_B - Id) - (Id - e_B)$.

References `dilation()`, `erosion()`, `mln::data::fill()`, and `minus()`.

9.111.2.16 `template<typename V> edge_image<util::site_pair<point2d>, V, util::graph> mln::morpho::line_gradient (const mln::image2d<V> & ima) [inline]`

Create a line [graph](#) image representing the gradient [norm](#) of a [mln::image2d](#).

References `mln::math::abs()`, `mln::image2d<T>::domain()`, `mln::box<P>::has()`, `mln::window<D>::insert()`, and `mln::Box<E>::nsites()`.

9.111.2.17 `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) [inline]`

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

Parameters:

← *input* The input image.

← *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.

9.111.2.18 `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, L & nbasins) [inline]`

Meyer's Watershed Transform (WST) algorithm.

Parameters:

← *input* The input image.

← *nbh* The connexity of markers.

→ *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.

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`.

9.111.2.19 `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).

9.111.2.20 `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).

9.111.2.21 `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).

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()`.

9.111.2.22 `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).

Referenced by `thick_miss()`, and `thickening()`.

9.111.2.23 `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`.

References `mln::extension::adjust_fill()`, `mln::geom::delta()`, `mln::accu::stat::rank< T >::init()`, `mln::initialize()`, and `mln::accu::stat::rank< T >::take()`.

9.111.2.24 `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 $THICK_B = Id + HMT_{TopeBG_B}$, where $B = (Bfg, Bbg)$.

References `hit_or_miss_background_opening()`, and `plus()`.

9.111.2.25 `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 $B = (Bfg, Bbg)$.

References `hit_or_miss()`, and `plus()`.

Referenced by `thinning()`.

9.111.2.26 `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 $B = (Bfg, Bbg)$.

References `hit_or_miss_opening()`, and `minus()`.

9.111.2.27 `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 $B = (Bfg, Bbg)$.

References `complementation()`, `hit_or_miss()`, `minus()`, and `thickening()`.

9.111.2.28 `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 $clo_B - Id$.

References `minus()`, and `mln::test::positive()`.

Referenced by `contrast()`.

9.111.2.29 `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

$= top_hat_white + top_hat_black$

$= (input - opening) + (closing - input)$

$= closing - opening$.

References `minus()`, and `mln::test::positive()`.

9.111.2.30 `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 $Id - ope_B$.

References `minus()`, and `mln::test::positive()`.

Referenced by `contrast()`.

9.112 `mln::morpho::approx` Namespace Reference

Namespace of approximate mathematical morphology routines.

9.112.1 Detailed Description

Namespace of approximate mathematical morphology routines.

9.113 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.

9.113.1 Detailed Description

Namespace of attributes used in mathematical morphology.

9.114 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.

9.114.1 Detailed Description

Namespace of approximate mathematical morphology closing routines.

9.114.2 Function Documentation

- 9.114.2.1** `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 $e_{-B} \circ d_B$.

References `mln::morpho::dilation()`, `mln::morpho::erosion()`, and `mln::win::sym()`.

9.115 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).

9.115.1 Detailed Description

Namespace of image processing routines of [elementary](#) mathematical morphology.

9.115.2 Function Documentation

- 9.115.2.1** `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.

Referenced by `top_hat_black()`, and `top_hat_self_complementary()`.

9.115.2.2 `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 $(d - id) - (id - e)$.

9.115.2.3 `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 $d \circ e$.

Referenced by `top_hat_self_complementary()`, and `top_hat_white()`.

9.115.2.4 `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 $clo - Id$.

References `closing()`, `mln::morpho::minus()`, and `mln::test::positive()`.

9.115.2.5 `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

$= top_hat_white + top_hat_black$

$= (Id - opening) + (closing - Id)$

$= closing - opening$.

References `closing()`, `mln::morpho::minus()`, `opening()`, and `mln::test::positive()`.

9.115.2.6 `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 $Id - ope$.

References `mln::morpho::minus()`, `opening()`, and `mln::test::positive()`.

9.116 mln::morpho::impl Namespace Reference

Namespace of mathematical morphology routines implementations.

Namespaces

- namespace [generic](#)

Namespace of mathematical morphology routines [generic](#) implementations.

9.116.1 Detailed Description

Namespace of mathematical morphology routines implementations.

9.117 mln::morpho::impl::generic Namespace Reference

Namespace of mathematical morphology routines [generic](#) implementations.

Functions

- `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 W>`
`mln::trait::concrete< I >::ret rank_filter (const Image< I > &input_, const Window< W > &win_,`
`unsigned k)`
Morphological rank_filter.

9.117.1 Detailed Description

Namespace of mathematical morphology routines [generic](#) implementations.

9.117.2 Function Documentation

9.117.2.1 `template<typename I, typename Wh, typename Wm> mln::trait::concrete< I >::ret`
`mln::morpho::impl::generic::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 $HMT_{(B_h, B_m)} = e_{B_h} \wedge (e_{B_m} \circ C)$.

References `mln::morpho::dilation()`, `mln::morpho::erosion()`, `mln::data::fill()`, `mln::initialize()`, and `mln::literal::zero`.

Referenced by `mln::morpho::thickening()`, and `mln::morpho::thinning()`.

9.117.2.2 `template<typename I, typename W> mln::trait::concrete< I >::ret`
`mln::morpho::impl::generic::rank_filter (const Image< I > &input_, const Window<`
`W > &win_, unsigned k) [inline]`

Morphological rank_filter.

References `mln::extension::adjust_fill()`, `mln::geom::delta()`, `mln::accu::stat::rank< T >::init()`, `mln::initialize()`, and `mln::accu::stat::rank< T >::take()`.

9.118 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.

9.118.1 Detailed Description

Namespace of approximate mathematical morphology opening routines.

9.118.2 Function Documentation

- 9.118.2.1** `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 $d_{\{-B\}} \circ e_B$.

References `mln::morpho::dilation()`, `mln::morpho::erosion()`, and `mln::win::sym()`.

9.119 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.

9.119.1 Detailed Description

Namespace of morphological [reconstruction](#) routines.

9.120 mln::morpho::reconstruction::by_dilation Namespace Reference

Namespace of morphological [reconstruction](#) by dilation routines.

9.120.1 Detailed Description

Namespace of morphological [reconstruction](#) by dilation routines.

9.121 mln::morpho::reconstruction::by_erosion Namespace Reference

Namespace of morphological [reconstruction](#) by erosion routines.

9.121.1 Detailed Description

Namespace of morphological [reconstruction](#) by erosion routines.

9.122 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 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_)
- `template<typename T, typename A>`
void `propagate_node_to_ancestors` (typename A::psite n, const T &t, `Image< A >` &a_, const typename A::value &v)
- `template<typename T, typename A>`
void `propagate_node_to_descendants` (typename A::psite &n, const T &t, `Image< A >` &a_, unsigned *nb_leaves=0)
- `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 F>`
void `propagate_representative` (const T &t, `Image< F >` &f_)

Propagate the representative node's value to non-representative points of the component.

9.122.1 Detailed Description

Namespace of morphological tree-related routines.

9.122.2 Function Documentation

9.122.2.1 `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:

← `a` Attribute.

← `t` Component `tree`.

→ `accu_image` Optional argument used to store image of `attribute` accumulator.

Returns:

The `attribute` image.

9.122.2.2 `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:

- ← *a* Attribute.
- ← *t* Component [tree](#).
- ← *values* [Value](#) image.
- *accu_image* Optional argument used to store image.

Returns:

9.122.2.3 `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 < .
```

9.122.2.4 `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:

← *f* The original image.

← *m* The connectivity mask.

← *nbh* The neighborhood of the mask.

Returns:

The computed [tree](#).

9.122.2.5 `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:

← *f* The input image.

← *nbh* The neighborhood.

Returns:

The corresponding max-tree structure.

References mln::data::sort_psites_increasing().

9.122.2.6 `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:

← *f* The input image.

← *nbh* The neighborhood.

Returns:

The corresponding min-tree structure.

References mln::data::sort_psites_decreasing().

9.122.2.7 `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).

Referenced by mln::morpho::tree::filter::subtractive().

9.122.2.8 `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).

9.122.2.9 `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:

- ← *n* Node to propagate.
- ← *t* Component *tree* used for propagation.
- ↔ *a_* Attribute image where values are propagated.

References `propagate_node_to_ancestors()`.

9.122.2.10 `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) [inline]`

Propagate a *value* *v* from a node *n* to its ancestors.

Parameters:

- ← *n* Node to propagate.
- ← *t* Component *tree* used for propagation.
- ← *a_* Attribute image where values are propagated.
- ← *v* Value to propagate.

Referenced by `propagate_node_to_ancestors()`.

9.122.2.11 `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:

- ← *n* Node to propagate.
- ← *t* Component *tree* used for propagation.
- ← *a_* Attribute image where values are propagated.
- *nb_leaves* Optional. Store the number of leaves in the component.

9.122.2.12 `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:

- ← *n* Node to propagate.
- ← *t* Component [tree](#) used for propagation.
- ← *a_* Attribute image where values are propagated.
- ← *v* [Value](#) to propagate.
- *nb_leaves* Optional. Store the number of leaves in the component.

9.122.2.13 `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.

9.123 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.

9.123.1 Detailed Description

Namespace for [attribute](#) filtering.

9.123.2 Function Documentation

9.123.2.1 `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:

- ← [tree](#) Component [tree](#).
- [f_](#) [Image](#) to [filter](#).
- ← [pred_](#) Filtering criterion.

9.123.2.2 `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.

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

9.123.2.3 `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:

← `tree` Component [tree](#).

→ `f_` Image to filter.

← `pred_` Filtering criterion.

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

9.123.2.4 `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:

← `tree` Component [tree](#).

→ `f_` Image to filter.

← `pred_` Filtering criterion.

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

9.123.2.5 `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:

- ← [tree](#) Component [tree](#).
- [f_](#) Image to [filter](#).
- ← [pred_](#) Filtering criterion.

References `mln::morpho::tree::propagate_if()`.

9.124 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)`
Meyer's Watershed Transform (WST) algorithm, with no count of basins.
- `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 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 [topological watershed transform](#) from [tree](#).

9.124.1 Detailed Description

Namespace of morphological [watershed](#) routines.

9.124.2 Function Documentation

- 9.124.2.1** `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) [inline]`

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

Parameters:

- ← *input* The input image.
- ← *nbh* The connexity of markers.
- \mathbb{L} is the type of labels, used to number the [watershed](#) itself (with the minimal [value](#)), and the basins.
- \mathbb{I} is the exact type of the input image.
- \mathbb{N} is the exact type of the neighborhood used to express *input*'s connexity.

Note that the first parameter, \mathbb{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.

9.124.2.2 `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, L & n_basins) [inline]`

Meyer's Watershed Transform (WST) algorithm.

Parameters:

- ← *input* The input image.
- ← *nbh* The connexity of markers.
- *n_basins* The number of basins.
- \mathbb{L} is the type of labels, used to number the [watershed](#) itself (with the minimal [value](#)), and the basins.
- \mathbb{I} is the exact type of the input image.
- \mathbb{N} is the exact type of the neighborhood used to express *input*'s connexity.

9.124.2.3 `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.

References `mln::literal::red`, and `superpose()`.

9.124.2.4 `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.

References `mln::data::convert()`, `mln::data::fill()`, and `mln::literal::zero`.

Referenced by `superpose()`.

9.124.2.5 `template<class T> T::image_t mln::morpho::watershed::topological (T & tree)`
[inline]

Compute a toological [watershed transform](#) from *tree*.

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()`.

9.125 mln::morpho::watershed::watershed Namespace Reference

Namespace of morphological [watershed](#) routines implementations.

Namespaces

- namespace [generic](#)

Namespace of morphological [watershed](#) routines [generic](#) implementations.

9.125.1 Detailed Description

Namespace of morphological [watershed](#) routines implementations.

9.126 mln::morpho::watershed::watershed::generic Namespace Reference

Namespace of morphological [watershed](#) routines [generic](#) implementations.

9.126.1 Detailed Description

Namespace of morphological [watershed](#) routines [generic](#) implementations.

9.127 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 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 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 l2_distance (const C(&vec1)[n], const C(&vec2)[n])`
L2-norm distance between vectors vec1 and vec2.
- `template<unsigned n, typename C>`
`C linfty (const C(&vec)[n])`
L-infinity-norm of a vector vec.
- `template<unsigned n, typename C>`
`C linfty_distance (const C(&vec1)[n], const C(&vec2)[n])`
L-infinity-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.

9.127.1 Detailed Description

Namespace of norms.

9.127.2 Function Documentation

9.127.2.1 `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*.

9.127.2.2 `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*.

9.127.2.3 `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*.

9.127.2.4 `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*.

9.127.2.5 `template<unsigned n, typename C> C mln::norm::linfty (const C(&) vec[n])
[inline]`

L-infinity-norm of a vector *vec*.

9.127.2.6 `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*.

9.127.2.7 `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*.

Referenced by `mln::geom::mesh_corner_point_area()`, and `mln::geom::mesh_normal()`.

9.128 `mln::norm::impl` Namespace Reference

Implementation namespace of [norm](#) namespace.

9.128.1 Detailed Description

Implementation namespace of [norm](#) namespace.

9.129 mln::opt Namespace Reference

Namespace of optional routines.

Namespaces

- namespace [impl](#)
Implementation namespace of [opt](#) namespace.

Functions

- `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).
- `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::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 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 ind)`
Read-write access to the ima [value](#) located at (ind).
- `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).

9.129.1 Detailed Description

Namespace of optional routines.

9.129.2 Function Documentation

9.129.2.1 `template<typename I> I::lvalue mln::opt::at (Image< I > &ima, def::coord sli, def::coord row, def::coord col) [inline]`

Read-write access to the ima [value](#) located at (sli, row, col).

9.129.2.2 `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).

9.129.2.3 `template<typename I> I::lvalue mln::opt::at (Image< I > & ima, def::coord row, def::coord col) [inline]`

Read-write access to the ima [value](#) located at (row, col).

9.129.2.4 `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).

9.129.2.5 `template<typename I> I::lvalue mln::opt::at (Image< I > & ima, def::coord ind) [inline]`

Read-write access to the ima [value](#) located at (ind).

9.129.2.6 `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).

Referenced by mln::transform::hough(), and mln::make::image().

9.130 mln::opt::impl Namespace Reference

Implementation namespace of [opt](#) namespace.

9.130.1 Detailed Description

Implementation namespace of [opt](#) namespace.

Three dimensions.

Two dimensions.

One dimension.

9.131 mln::pw Namespace Reference

Namespace of "point-wise" expression tools.

Classes

- class [image](#)
A generic point-wise [image](#) implementation.

9.131.1 Detailed Description

Namespace of "point-wise" expression tools.

9.132 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>`
`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, 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>`
`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.

9.132.1 Detailed Description

Namespace of "point-wise" expression tools.

9.132.2 Function Documentation

9.132.2.1 `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) [inline]`

FIXME: work only for 3d images.

References `mln::p_array< P >::nsites()`.

9.132.2.2 `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) [inline]`

Register [point](#) in `c` using a function of closest points `closest_point`.

Parameters:

- ← *P_* The cloud of points.
- ← *X* the reference surface.
- ← *closest_point* The function of closest points.

Returns:

the rigid transformation which may be use later to create a registered image.

9.132.2.3 `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:

- ← *P_* The cloud of points.
- ← *X* the reference surface.
- ← *closest_point* The function of closest points.
- ← *initial_rot* An initial rotation.
- ← *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.

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.

9.132.2.4 `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.

9.132.2.5 `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.

9.132.2.6 `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.

9.133 mln::select Namespace Reference

Select namespace (FIXME [doc](#)).

Classes

- struct [p_of](#)
Structure [p_of](#).

9.133.1 Detailed Description

Select namespace (FIXME [doc](#)).

9.134 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, 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 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 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 I>`
`mln_meta_accu_result (A, typename I::site) compute_with_weights(const Meta_Accumulator< A > &a`
`& &a`
Compute an accumulator on a site [set](#) described by an image.
- `template<typename A, typename S>`
`mln_meta_accu_result (A, typename S::site) compute(const Meta_Accumulator< A > &a`
`& &a`
Compute an accumulator onto a site [set](#).

9.134.1 Detailed Description

Namespace of image processing routines related to [pixel](#) sets.

9.134.2 Function Documentation

9.134.2.1 `template<typename S> unsigned mln::set::card (const Site_Set< S > &s)` `[inline]`

Compute the cardinality of the site [set](#) s.

9.134.2.2 `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:

- ← *a* An accumulator.
- ← *s* A site [set](#).

Returns:

The accumulator result.

Referenced by `mln::registration::icp()`.

9.134.2.3 `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) [inline]`

Compute an accumulator on every labeled sub-site-sets.

Parameters:

- ← *a* An accumulator.
- ← *w* An image of weights (a site -> a weight).
- ← *label* A label image.
- ← *nlabels* The number of labels in `label`.

Returns:

An array of accumulator result. One per label.

Compute an accumulator on every labeled sub-site-sets.

Parameters:

- ← *a_* An accumulator.
- ← *w_* An image of weights (a site -> a weight).
- ← *label_* A label image.
- ← *nlabels* The number of labels in `label`.

Returns:

An array of accumulator result. One per label.

9.134.2.4 `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:

- ← *a* An accumulator.
- ← *w* An image of weights (a site -> a weight).

Returns:

The accumulator result.

Compute an accumulator on a site [set](#) described by an image.

Parameters:

- ← *a_* An accumulator.
- ← *w_* An image of weights (a site -> a weight).

Returns:

The accumulator result.

9.134.2.5 `template<typename S> S::site mln::set::get (const Site_Set< S > & s, size_t index)`
`[inline]`

FIXME.

9.134.2.6 `template<typename S> bool mln::set::has (const Site_Set< S > & s, const typename S::site & e)` `[inline]`

FIXME.

9.134.2.7 `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:

- ← *a* A meta-accumulator.
- ← *w* An image of weights (a site -> a weight).

Returns:

The accumulator result.

9.134.2.8 `template<typename A, typename S> mln::set::mln_meta_accu_result (A, typename S::site) const` `[inline]`

Compute an accumulator onto a site [set](#).

Parameters:

- ← *a* A meta-accumulator.
- ← *s* A site [set](#).

9.135 mln::subsampling Namespace Reference

Namespace of "point-wise" expression tools.

Functions

- `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.

9.135.1 Detailed Description

Namespace of "point-wise" expression tools.

9.135.2 Function Documentation

- 9.135.2.1** `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.

References `mln::linear::gaussian()`, `mln::geom::ncols()`, and `mln::geom::nrows()`.

- 9.135.2.2** `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.

References `mln::geom::ncols()`, and `mln::geom::nrows()`.

9.136 mln::tag Namespace Reference

Namespace of image processing routines related to tags.

9.136.1 Detailed Description

Namespace of image processing routines related to tags.

9.137 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`.

9.137.1 Detailed Description

Namespace of image processing routines related to [pixel](#) tests.

9.137.2 Function Documentation

9.137.2.1 `template<typename I> bool mln::test::positive (const Image< I > &input)` `[inline]`

Test if an image only contains positive values.

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()`.

9.137.2.2 `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:

- ← *pset* The [point set](#).
- ← *f* The predicate.

9.137.2.3 `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:

- ← *lhs* The image.
- ← *rhs* The image.
- ← *f* The predicate.

9.137.2.4 `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:

- ← *ima* The image.
- ← *f* The predicate.

Referenced by `mln::operator<()`, `mln::operator<=()`, `mln::operator==()`, and `positive()`.

9.138 `mln::test::impl` Namespace Reference

Implementation namespace of [test](#) namespace.

9.138.1 Detailed Description

Implementation namespace of [test](#) namespace.

9.139 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`.
- struct [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`.*
- struct `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.
- 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_site`< 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_site`< 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 `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 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 D>
bool `operator==` (const `complex`< D > &lhs, const `complex`< D > &rhs)
Compare two complexes for equality.
- template<unsigned D>
`algebraic_n_face`< 1, D > `edge` (const `n_face`< 0, D > &f1, const `n_face`< 0, D > &f2)
Helpers.
- 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>`
`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 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 D>`
`bool operator== (const face< D > &lhs, const face< D > &rhs)`
Comparison of two instances of `mln::topo::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 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 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 D>`
`bool operator== (const algebraic_face< D > &lhs, const algebraic_face< D > &rhs)`
Comparison of two instances of `mln::topo::algebraic_face`.
- `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.
- `template<unsigned N, unsigned D>`
`algebraic_n_face< N, D > operator-` (const `n_face< N, D >` &f)
Inversion operators.
- `template<unsigned D>`
`algebraic_face< D > operator-` (const `face< D >` &f)
Inversion operators.

9.139.1 Detailed Description

Namespace of "point-wise" expression tools.

9.139.2 Function Documentation

9.139.2.1 `template<unsigned D, typename G> void mln::topo::detach (const complex_psite< 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.

References `mln::make::detachment()`, `mln::data::fill()`, and `is_facet()`.

9.139.2.2 `template<unsigned D> algebraic_n_face< 1, D > mln::topo::edge (const n_face< 0, D > &f1, const n_face< 0, D > &f2) [inline]`

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 f_1 and f_2 (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).

References `mln::topo::n_face< N, D >::higher_dim_adj_faces()`.

9.139.2.3 `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?

Referenced by `mln::make::attachment()`, `mln::make::cell()`, `detach()`, and `mln::make::detachment()`.

9.139.2.4 `template<unsigned D> algebraic_face< D > mln::topo::make_algebraic_face (const face< D > &f, bool sign) [inline]`

Create an algebraic [face](#) handle of a D -[complex](#).

9.139.2.5 `template<unsigned N, unsigned D> algebraic_n_face< N, D > mln::topo::make_algebraic_n_face (const n_face< N, D > &f, bool sign) [inline]`

Create an algebraic N -face handle of a D -[complex](#).

9.139.2.6 `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](#).

References `mln::topo::n_face< N, D >::cplx()`.

9.139.2.7 `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](#).

References `mln::topo::face< D >::cplx()`.

9.139.2.8 `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](#).

References mln::topo::n_face< N, D >::cplx().

9.139.2.9 `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](#).

References mln::topo::face< D >::cplx().

9.139.2.10 `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.

References mln::topo::n_faces_set< N, D >::add().

9.139.2.11 `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.

References mln::topo::n_faces_set< N, D >::add().

9.139.2.12 `template<unsigned N, unsigned D> algebraic_n_face< N, D > mln::topo::operator-(const n_face< N, D > & f) [inline]`

Inversion operators.

9.139.2.13 `template<unsigned D> algebraic_face< D > mln::topo::operator-(const face< D > & f) [inline]`

Inversion operators.

9.139.2.14 `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](#).

9.139.2.15 `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.

9.139.2.16 `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](#).

9.139.2.17 `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.

9.139.2.18 `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](#).

9.139.2.19 `template<unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const face< D > & f) [inline]`

Print an [mln::topo::face](#).

9.139.2.20 `template<unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const complex< D > & c) [inline]`

Pretty print a [complex](#).

References [mln::topo::complex< D >::print\(\)](#).

9.139.2.21 `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](#).

9.139.2.22 `template<unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const algebraic_face< D > & f) [inline]`

Print an [mln::topo::algebraic_face](#).

9.139.2.23 `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](#).

References [mln::topo::n_face< N, D >::cplx\(\)](#), and [mln::topo::n_face< N, D >::face_id\(\)](#).

9.139.2.24 `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](#).

References [mln::topo::face< D >::cplx\(\)](#), [mln::topo::face< D >::face_id\(\)](#), and [mln::topo::face< D >::n\(\)](#).

9.139.2.25 `template<unsigned D> bool mln::topo::operator== (const complex< D > & lhs, const complex< D > & rhs) [inline]`

Compare two complexes for equality.

9.139.2.26 `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](#).

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\(\)](#).

9.139.2.27 `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`.

References `mln::topo::face< D >::cplx()`, `mln::topo::face< D >::face_id()`, `mln::topo::face< D >::n()`, and `mln::topo::algebraic_face< D >::sign()`.

9.140 mln::trace Namespace Reference

Namespace of routines related to the [trace](#) mechanism.

9.140.1 Detailed Description

Namespace of routines related to the [trace](#) mechanism.

9.141 `mln::trait` Namespace Reference

Namespace where traits are defined.

9.141.1 Detailed Description

Namespace where traits are defined.

Namespace for image traits.

9.142 mln::transform Namespace Reference

Namespace of transforms.

Functions

- 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, mln::trait::ch_value< I, typename I::site >::ret > [distance_and_closest_point_geodesic](#) (const [Image](#)< I > &input, 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 [Neighborhood](#)< 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](#).

9.142.1 Detailed Description

Namespace of transforms.

9.142.2 Function Documentation

- 9.142.2.1 `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:

- ← *pset* an array of sites.
- ← *closest_point_domain* domain of the returned image.
- ← *nbh* neighborhood
- ← *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`.

References `mln::geom::bbox()`, `mln::make::couple()`, `mln::canvas::distance_geodesic()`, `mln::data::fill()`, and `mln::box< P >::is_valid()`.

- 9.142.2.2 `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:

- ← *input* [Image](#) from which the geodesic distance is computed.
- ← *nbh* [Neighborhood](#)

← *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`.

References `mln::make::couple()`, and `mln::canvas::distance_geodesic()`.

9.142.2.3 `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:

← *input* [Image](#) from which the geodesic distance is computed.

← *nbh* [Neighborhood](#)

← *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`.

References `mln::make::couple()`, and `mln::canvas::distance_geodesic()`.

9.142.2.4 `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](#).

References `mln::canvas::distance_front()`.

9.142.2.5 `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](#).

References `mln::canvas::distance_geodesic()`.

9.142.2.6 `template<typename I> image2d< float > mln::transform::hough (const Image< I > & input_) [inline]`

Compute the hough [transform](#) from a binary image.

Objects used for computation must be [set](#) to 'true'.

Parameters:

← *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.

References `mln::opt::at()`, `mln::data::fill()`, `mln::geom::min_col()`, `mln::geom::min_row()`, `mln::geom::ncols()`, and `mln::geom::nrows()`.

9.142.2.7 `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) [inline]`

Influence zone [transform](#).

References `influence_zone_front()`.

9.142.2.8 `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) [inline]`

Influence zone [transform](#).

References `mln::canvas::distance_front()`.

Referenced by `influence_zone_front()`.

9.142.2.9 `template<typename I, typename N> mln::trait::concrete< I >::ret mln::transform::influence_zone_geodesic (const Image< I > & input, const Neighborhood< N > & nbh) [inline]`

Geodesic influence zone [transform](#).

Parameters:

← *input* An image.

← *nbh* A neighborhood.

Returns:

An image of influence zone.

9.142.2.10 `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) [inline]`

Geodesic influence zone [transform](#).

Parameters:

- ← *input* An image.
- ← *nbh* A neighborhood.
- ← *max* The maximum influence zone distance.
- ← *background_value* The [value](#) used as background (i.e. not propagated).

Returns:

An image of influence zone.

References `mln::canvas::distance_geodesic()`.

9.143 mln::util Namespace Reference

Namespace of tools using for more complex algorithm.

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".

Namespaces

- namespace `impl`
Implementation namespace of `util` namespace.

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.

9.143.1 Detailed Description

Namespace of tools using for more complex algorithm.

Forward declaration.

9.143.2 Typedef Documentation

9.143.2.1 `typedef object_id<vertex_tag, unsigned> mln::util::vertex_id_t`

[Vertex](#) id type.

9.143.3 Function Documentation

9.143.3.1 `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:

- ← *ima_* The domain of output image.
- ← *tree_node* The root [tree_node](#) to display.

References [mln::data::fill\(\)](#).

9.143.3.2 `template<typename I, typename J> void mln::util::display_tree (const Image< J > & ima_, tree< I > & tree) [inline]`

Display a [tree](#).

Parameters:

- ← *ima_* The domain of output image.
- ← *tree* The [tree](#) to [display](#).

References `mln::util::tree< T >::root()`.

9.143.3.3 `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) [inline]`

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`.

9.143.3.4 `template<typename I> greater_point< I > mln::util::make_greater_point (const Image< I > & ima) [inline]`

Helper to build a [mln::util::greater_point](#).

References `make_greater_point()`.

Referenced by `make_greater_point()`.

9.143.3.5 `template<typename I> greater_psite< I > mln::util::make_greater_psite (const Image< I > & ima) [inline]`

Helper to build a [mln::util::greater_psite](#).

References `make_greater_psite()`.

Referenced by `make_greater_psite()`.

9.143.3.6 `template<typename G> bool mln::util::operator< (const vertex< G > & lhs, const vertex< G > & rhs) [inline]`

Less operator. Test whether `lhs.id() < rhs.id()`.

9.143.3.7 `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`.

9.143.3.8 `template<typename T> std::ostream & mln::util::operator<< (std::ostream & ostr, const array< T > & a) [inline]`

Operator<<.

References `mln::util::array< T >::nelements()`.

9.143.3.9 `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.

References `mln::util::vertex< G >::graph()`, and `mln::util::vertex< G >::id()`.

9.143.3.10 `template<typename T> bool mln::util::operator==(const array< T > & lhs, const array< T > & rhs) [inline]`

Operator==.

References `mln::util::array< T >::std_vector()`.

9.143.3.11 `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`.

References `ord_strict()`.

Referenced by `mln::util::ord_pair< T >::change_both()`, `mln::util::ord_pair< T >::change_first()`, `mln::util::ord_pair< T >::change_second()`, and `ord_strict()`.

9.143.3.12 `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`.

References `ord_weak()`.

Referenced by `mln::util::ord_pair< T >::change_both()`, `mln::util::ord_pair< T >::change_first()`, `mln::util::ord_pair< T >::change_second()`, `mln::box< P >::is_valid()`, and `ord_weak()`.

9.143.3.13 `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:

- ← *tree* The *tree* to *convert*.
- *output_* The image containing *tree* informations.

References `mln::util::impl::tree_fast_to_image()`.

Referenced by `tree_fast_to_image()`.

9.143.3.14 `template<typename T> tree_fast< T > mln::util::tree_to_fast (tree< T > & input)`
[inline]

Facade.

Convert a *tree* into an *tree_fast*.

Parameters:

- ← *input* The *tree* to *convert*.

Returns:

The *tree_fast* containing *tree* informations.

References `mln::util::tree< T >::root()`.

9.143.3.15 `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:

- ← *tree* The *tree* to *convert*.
- *output_* The image containing *tree* information.

9.144 mln::util::impl Namespace Reference

Implementation namespace of [util](#) namespace.

Functions

- `template<typename T, typename I>`
`void tree_fast_to_image (tree_fast< T > &tree, Image< I > &output_)`

9.144.1 Detailed Description

Implementation namespace of [util](#) namespace.

9.144.2 Function Documentation

9.144.2.1 `template<typename T, typename I> void mln::util::impl::tree_fast_to_image`
`(tree_fast< T > & tree, Image< I > & output_) [inline]`

Convert a `tree_fast` into an image.

Parameters:

- ← `tree` The `tree` to convert.
- `output_` The image containing `tree` informations.

References `tree_fast_to_image()`.

Referenced by `mln::util::tree_fast_to_image()`.

9.145 mln::value Namespace Reference

Namespace of materials related to [pixel value](#) types.

Classes

- class [float01](#)
Class for floating values restricted to the interval [0.
- 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 \mathbb{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 \mathbb{T} .

Namespaces

- namespace [impl](#)
Implementation namespace of [value](#) namespace.

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.
- `std::ostream & operator<<` (std::ostream &ostr, const `sign` &i)
Print an signed integer `i` into the output stream `ostr`.
- template<typename T>
`std::ostream & operator<<` (std::ostream &ostr, const `scalar_< T >` &s)
Print a scalar `s` in an 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`.
- `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 int_u_sat<n> &i)`
Print a saturated unsigned 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<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`.
- `std::ostream & operator<<< (std::ostream &ostr, const graylevel_f &g)`
`Op<<<`.
- `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<<<`.
- `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<unsigned n, typename S>`
`rgb<n>::interop operator* (const rgb<n> &lhs, const mln::value::scalar<S> &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)
Product.`

- `template<unsigned n>`
`rgb< n >::interop operator-` (const `rgb< n >` &lhs, const `rgb< n >` &rhs)
Subtraction.
- `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<unsigned n, typename S>`
`rgb< n >::interop operator/` (const `rgb< n >` &lhs, const `mln::value::scalar_< S >` &s)
Division.
- `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<typename I>`
`stack_image< 2, const I > stack` (const `Image< I >` &ima1, const `Image< I >` &ima2)
Shortcut to build a stack with two images.

9.145.1 Detailed Description

Namespace of materials related to [pixel value](#) types.

9.145.2 Typedef Documentation

9.145.2.1 typedef `float01_<16>` `mln::value::float01_16`

Alias for 16 bit [float01](#).

9.145.2.2 typedef `float01_<8>` `mln::value::float01_8`

Alias for 8 bit [float01](#).

9.145.2.3 typedef `graylevel<16>` `mln::value::gl16`

Alias for 16 bit [graylevel](#).

9.145.2.4 typedef graylevel<8> mln::value::gl8

Alias for 8 bit [graylevel](#).

9.145.2.5 typedef graylevel_f mln::value::glf

Alias for graylevels encoded by float.

9.145.2.6 typedef int_s<16> mln::value::int_s16

Alias for signed 16-bit integers.

9.145.2.7 typedef int_s<32> mln::value::int_s32

Alias for signed 32-bit integers.

9.145.2.8 typedef int_s<8> mln::value::int_s8

Alias for signed 8-bit integers.

9.145.2.9 typedef int_u<12> mln::value::int_u12

Alias for unsigned 12-bit integers.

9.145.2.10 typedef int_u<16> mln::value::int_u16

Alias for unsigned 16-bit integers.

9.145.2.11 typedef mln::value::int_u<32> mln::value::int_u32

Alias for unsigned 32-bit integers.

9.145.2.12 typedef mln::value::int_u<8> mln::value::int_u8

Alias for unsigned 8-bit integers.

9.145.2.13 typedef label<16> mln::value::label_16

Alias for 16-bit integers.

9.145.2.14 typedef label<32> mln::value::label_32

Alias for 32-bit integers.

9.145.2.15 `typedef mln::value::label<8> mln::value::label_8`

Alias for 8-bit labels.

9.145.2.16 `typedef rgb<16> mln::value::rgb16`

Color class for red-green-blue where every component is 16-bit encoded.

9.145.2.17 `typedef rgb<8> mln::value::rgb8`

Color class for red-green-blue where every component is 8-bit encoded.

9.145.3 Function Documentation**9.145.3.1** `template<typename Dest, typename Src> Dest mln::value::cast (const Src & src)`
[inline]

Cast a [value](#) `src` from type `Src` to type `Dest`.

9.145.3.2 `template<typename V> internal::equiv_< V >::ret mln::value::equiv (const mln::Value< V > & v)` [inline]

Access to the equivalent [value](#).

9.145.3.3 `template<unsigned n, typename S> rgb< n >::interop mln::value::operator* (const rgb< n > & lhs, const mln::value::scalar_< S > & s)` [inline]

Product.

9.145.3.4 `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)` [inline]

Product.

9.145.3.5 `template<unsigned n> rgb< n >::interop mln::value::operator+ (const rgb< n > & lhs, const rgb< n > & rhs)` [inline]

Addition.

```
{
```

9.145.3.6 `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)`
[inline]

Addition.

```
{
```

9.145.3.7 `template<unsigned n> rgb< n >::interop mln::value::operator- (const rgb< n > & lhs, const rgb< n > & rhs) [inline]`

Subtraction.

9.145.3.8 `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) [inline]`

Subtraction.

9.145.3.9 `template<unsigned n, typename S> rgb< n >::interop mln::value::operator/ (const rgb< n > & lhs, const mln::value::scalar_< S > & s) [inline]`

Division.

9.145.3.10 `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) [inline]`

Division.

9.145.3.11 `std::ostream & mln::value::operator<< (std::ostream & ostr, const sign & i) [inline]`

Print an signed integer *i* into the output stream *ostr*.

Parameters:

- ↔ *ostr* An output stream.
- ← *i* An [sign value](#)

Returns:

The modified output stream *ostr*.

References `mln::debug::format()`.

9.145.3.12 `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*.

9.145.3.13 `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:

- ↔ *ostr* An output stream.

← *c* An [rgb](#).

Returns:

The modified output stream `ostr`.

References `mln::debug::format()`.

9.145.3.14 `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:

↔ *ostr* An output stream.

← *l* A [label](#).

Returns:

The modified output stream `ostr`.

References `mln::debug::format()`.

9.145.3.15 `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:

↔ *ostr* An output stream.

← *i* A saturated unsigned integer.

Returns:

The modified output stream `ostr`.

References `mln::debug::format()`.

9.145.3.16 `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:

↔ *ostr* An output stream.

← *i* An unsigned integer.

Returns:

The modified output stream `ostr`.

References `mln::debug::format()`.

9.145.3.17 `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:

- ↔ *ostr* An output stream.
- ← *i* An signed integer.

Returns:

The modified output stream *ostr*.

References `mln::debug::format()`.

9.145.3.18 `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:

- ↔ *ostr* An output stream.
- ← *c* An [rgb](#).

Returns:

The modified output stream *ostr*.

References `mln::debug::format()`.

9.145.3.19 `std::ostream & mln::value::operator<< (std::ostream & ostr, const graylevel_f & g) [inline]`

Op<<.

References `mln::value::graylevel_f::value()`.

9.145.3.20 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr, const graylevel<n> & g) [inline]`

Op<<.

9.145.3.21 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr, const float01<n> & f) [inline]`

Op<<.

9.145.3.22 `bool mln::value::operator== (const sign & lhs, const sign & rhs) [inline]`

Comparison operator.

9.145.3.23 `template<typename H, typename S, typename L> bool mln::value::operator==(const hsl_< H, S, L > & lhs, const hsl_< H, S, L > & rhs) [inline]`

Comparison.

9.145.3.24 `template<typename V> V mln::value::other (const V & val) [inline]`

Give an other [value](#) than `val`.

9.145.3.25 `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.

9.146 mln::value::impl Namespace Reference

Implementation namespace of [value](#) namespace.

9.146.1 Detailed Description

Implementation namespace of [value](#) namespace.

9.147 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 [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 W>
[mln_regular](#) (W) [shift](#)(const [Window](#)< W > &win
Shift a [window](#) win with a delta-point dp.
- 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>
W [sym](#) (const [Weighted_Window](#)< W > &w_win)
Give the symmetrical weighted [window](#) of w_win.
- template<typename W>
W [sym](#) (const [Window](#)< W > &win)
Give the symmetrical [window](#) of win.

9.147.1 Detailed Description

Namespace of image processing routines related to [win](#).

9.147.2 Function Documentation

9.147.2.1 template<typename N1, typename N2> N2 [neighb](#)< typename N1::window::regular >
[mln::win::diff](#) (const [Neighborhood](#)< N1 > &nbh1, const [Neighborhood](#)< N2 > &
nbh2) [inline]

Set difference between a couple of neighborhoods nbh1 and nbh2.

Referenced by mln::operator-().

9.147.2.2 template<typename W> [mln::win::mln_regular](#) (W) const [inline]

Shift a [window](#) win with a delta-point dp.

9.147.2.3 `template<typename W1, typename W2> mln::win::mln_regular (W1) const`
[inline]

Set difference between a couple of windows `win1` and `win2`.

9.147.2.4 `template<typename W> W mln::win::sym (const Weighted_Window< W > & w_win)`
[inline]

Give the symmetrical weighted [window](#) of `w_win`.

9.147.2.5 `template<typename W> W mln::win::sym (const Window< W > & win)` [inline]

Give the symmetrical [window](#) of `win`.

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()`.

Chapter 10

Class Documentation

10.1 `mln::accu::center< P, V >` Struct Template Reference

Mass `center` accumulator.

```
#include <center.hh>
```

Inherits `mln::accu::internal::base< V, mln::accu::center< P, V > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `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`.
- `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*.

10.1.2 Member Function Documentation**10.1.2.1 `template<typename P, typename V> void mln::accu::center< P, V >::init ()` [inline]**

Manipulators.

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.

Referenced by `mln::accu::center< P, V >::to_result()`.

10.1.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t)` [inline, inherited]

Take as initialization the `value` `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.1.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t)` [inline, inherited]

Take `n` times the `value` `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.1.2.5 `template<typename P, typename V> V mln::accu::center< P, V >::to_result () const` [inline]

Get the `value` of the accumulator.

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 mln::accu::internal::base< R, mln::accu::convolve< T1, T2, R > >.

Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `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`.
- `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.

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.

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.

10.2.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References [mln::mln_exact\(\)](#).

10.2.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References [mln::mln_exact\(\)](#).

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.

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 mln::accu::internal::base< unsigned, mln::accu::count_adjacent_vertices< F, S > >.

Public Member Functions

- bool [is_valid](#) () const
Return whether this [accu](#) can return a result.
- 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.
- 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.

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.

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.

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`.

10.3.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the `value` `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.3.2.5 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the `value` `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

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.

10.4 mln::accu::count_labels< L > Struct Template Reference

Count the number of different labels in an [image](#).

```
#include <count_labels.hh>
```

Inherits mln::accu::internal::base< unsigned, mln::accu::count_labels< L > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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.
- 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.

10.4.2 Member Function Documentation

10.4.2.1 template<typename L> void mln::accu::count_labels< L >::init () [inline]

Manipulators.

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.

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*.

10.4.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t)` [inline, inherited]

Take as initialization the [value](#) *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.4.2.5 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t)` [inline, inherited]

Take *n* times the [value](#) *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.4.2.6 `template<typename L> unsigned mln::accu::count_labels< L >::to_result () const`
[inline]

Get the [value](#) of the accumulator.

10.5 mln::accu::count_value< V > Struct Template Reference

Count a given [value](#).

```
#include <count_value.hh>
```

Inherits mln::accu::internal::base< unsigned, mln::accu::count_value< V > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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.
- 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 >
```

Count a given [value](#).

10.5.2 Member Function Documentation

10.5.2.1 template<typename V> void mln::accu::count_value< V >::init () [inline]

Manipulators.

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.

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*.

10.5.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t)` [inline, inherited]

Take as initialization the [value](#) *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.5.2.5 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t)` [inline, inherited]

Take *n* times the [value](#) *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.5.2.6 `template<typename V> unsigned mln::accu::count_value< V >::to_result () const`
[inline]

Get the [value](#) of the accumulator.

10.6 mln::accu::histo< V > Struct Template Reference

Generic histogram class over a [value set](#) with type V.

```
#include <histo.hh>
```

Inherits mln::accu::internal::base< const std::vector< unsigned > &, mln::accu::histo< V > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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 [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.

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.

10.6.2.2 template<typename V> void mln::accu::histo< V >::take (const argument & t)
`[inline]`

Manipulators.

10.6.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References [mln::mln_exact\(\)](#).

10.6.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References [mln::mln_exact\(\)](#).

10.6.2.5 `template<typename V> const std::vector< unsigned > & mln::accu::histo< V >::vect () const [inline]`

Get the [value](#) of the accumulator.

10.7 mln::accu::label_used< L > Struct Template Reference

References all the labels used.

```
#include <label_used.hh>
```

Inherits mln::accu::internal::base< const mln::fun::i2v::array< bool > &, mln::accu::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.
- 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.
- 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.

10.7.2 Member Function Documentation

10.7.2.1 template<typename L> void mln::accu::label_used< L >::init () [inline]

Initialize accumulator attributes.

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.

10.7.2.3 `template<typename L> void mln::accu::label_used< L >::take (const argument & l)`
[inline]

Manipulators.

10.7.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t)` [inline, inherited]

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.7.2.5 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t)` [inline, inherited]

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

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.

10.8 mln::accu::logic::land Struct Reference

"Logical-and" accumulator.

```
#include <land.hh>
```

Inherits mln::accu::internal::base< bool, mln::accu::logic::land >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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.
- bool [to_result](#) () const
Get the [value](#) of the accumulator.
- void [init](#) ()
Manipulators.

10.8.1 Detailed Description

"Logical-and" accumulator.

10.8.2 Member Function Documentation

10.8.2.1 void mln::accu::logic::land::init () [inline]

Manipulators.

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.

10.8.2.3 template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T &t) [inline, inherited]

Take as initialization the [value](#) t.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References `mln::mln_exact()`.

10.8.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.8.2.5 `bool mln::accu::logic::land::to_result () const [inline]`

Get the [value](#) of the accumulator.

10.9 mln::accu::logic::land_basic Struct Reference

"Logical-and" accumulator.

```
#include <land_basic.hh>
```

Inherits mln::accu::internal::base< bool, mln::accu::logic::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.
- 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.
- 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.

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.

10.9.2.2 void mln::accu::logic::land_basic::init () [inline]

Manipulators.

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.

10.9.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t)` `[inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.9.2.5 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t)` `[inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.9.2.6 `bool mln::accu::logic::land_basic::to_result () const` `[inline]`

Get the [value](#) of the accumulator.

10.10 mln::accu::logic::lor Struct Reference

"Logical-or" accumulator.

```
#include <lor.hh>
```

Inherits mln::accu::internal::base< bool, mln::accu::logic::lor >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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.
- bool [to_result](#) () const
Get the [value](#) of the accumulator.
- void [init](#) ()
Manipulators.

10.10.1 Detailed Description

"Logical-or" accumulator.

10.10.2 Member Function Documentation

10.10.2.1 void mln::accu::logic::lor::init () [inline]

Manipulators.

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.

10.10.2.3 template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T &t) [inline, inherited]

Take as initialization the [value](#) t.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References `mln::mln_exact()`.

10.10.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take *n* times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.10.2.5 `bool mln::accu::logic::lor::to_result () const [inline]`

Get the [value](#) of the accumulator.

10.11 mln::accu::logic::lor_basic Struct Reference

"Logical-or" accumulator class.

```
#include <lor_basic.hh>
```

Inherits mln::accu::internal::base< bool, mln::accu::logic::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.
- 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.
- 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.

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.

10.11.2.2 void mln::accu::logic::lor_basic::init () [inline]

Manipulators.

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.

10.11.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t)` `[inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.11.2.5 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t)` `[inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.11.2.6 `bool mln::accu::logic::lor_basic::to_result () const` `[inline]`

Get the [value](#) of the accumulator.

10.12 mln::accu::maj_h< T > Struct Template Reference

Compute the majority [value](#).

```
#include <maj_h.hh>
```

Inherits mln::accu::internal::base< const T &, mln::accu::maj_h< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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.
- 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.

10.12.2 Member Function Documentation

10.12.2.1 template<typename T> void mln::accu::maj_h< T >::init () [inline]

Manipulators.

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.

10.12.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.12.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.12.2.5 `template<typename T> const T & mln::accu::maj_h< T >::to_result () const [inline]`

Get the [value](#) of the accumulator.

10.13 mln::accu::math::count< T > Struct Template Reference

Generic counter accumulator.

```
#include <count.hh>
```

Inherits mln::accu::internal::base< unsigned, mln::accu::math::count< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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.
- 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](#).

10.13.2 Member Function Documentation

10.13.2.1 template<typename T> void mln::accu::math::count< T >::init () [inline]

Manipulators.

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.

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*.

10.13.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t)` [inline, inherited]

Take as initialization the [value](#) *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.13.2.5 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t)` [inline, inherited]

Take *n* times the [value](#) *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.13.2.6 `template<typename T> unsigned mln::accu::math::count< T >::to_result () const`
[inline]

Get the [value](#) of the accumulator.

10.14 mln::accu::math::inf< T > Struct Template Reference

Generic [inf](#) accumulator class.

```
#include <inf.hh>
```

Inherits mln::accu::internal::base< const T &, mln::accu::math::inf< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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.
- 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.

10.14.2 Member Function Documentation

10.14.2.1 template<typename T> void mln::accu::math::inf< T >::init () [inline]

Manipulators.

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.

10.14.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.14.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.14.2.5 `template<typename T> const T & mln::accu::math::inf< T >::to_result () const [inline]`

Get the [value](#) of the accumulator.

10.15 mln::accu::math::sum< T, S > Struct Template Reference

Generic [sum](#) accumulator class.

```
#include <sum.hh>
```

Inherits mln::accu::internal::base< const S &, mln::accu::math::sum< T, S > >.

Public Member Functions

- `bool is_valid () const`
Check whether this [accu](#) is able to return a result.
- `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.
- `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.

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.

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.

10.15.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.15.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

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.

10.16 mln::accu::math::sup< T > Struct Template Reference

Generic [sup](#) accumulator class.

```
#include <sup.hh>
```

Inherits mln::accu::internal::base< const T &, mln::accu::math::sup< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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.
- 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.

10.16.2 Member Function Documentation

10.16.2.1 `template<typename T> void mln::accu::math::sup< T >::init () [inline]`

Manipulators.

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.

10.16.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.16.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.16.2.5 `template<typename T> const T & mln::accu::math::sup< T >::to_result () const [inline]`

Get the [value](#) of the accumulator.

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 mln::accu::internal::base< I::psite, mln::accu::max_site< I > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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.
- 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](#).

10.17.2 Member Function Documentation

10.17.2.1 template<typename I> void mln::accu::max_site< I >::init () [inline]

Manipulators.

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.

Referenced by mln::accu::max_site< I >::to_result().

10.17.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.17.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.17.2.5 `template<typename I> I::psite mln::accu::max_site< I >::to_result () const [inline]`

Get the [value](#) of the accumulator.

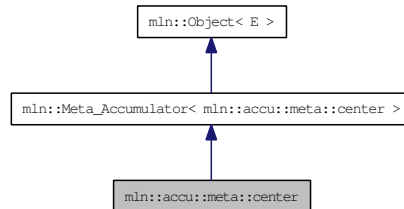
References `mln::accu::max_site< I >::is_valid()`.

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

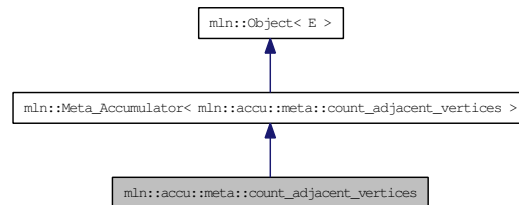
Meta accumulator for [center](#).

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

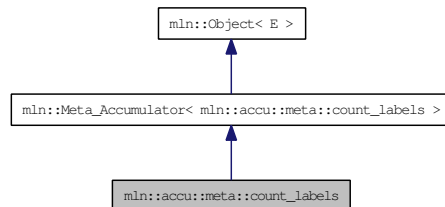
Meta accumulator for [count_adjacent_vertices](#).

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

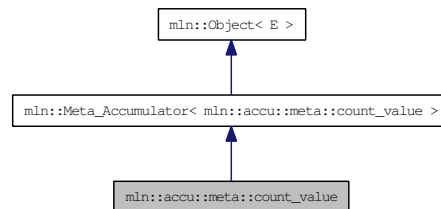
Meta accumulator for [count_labels](#).

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

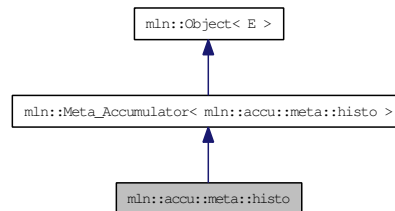
FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for [count_value](#).

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

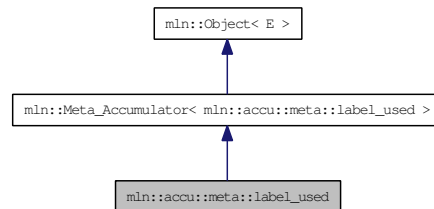
Meta accumulator for [histo](#).

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

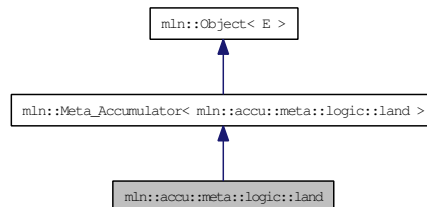
Meta accumulator for [label_used](#).

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

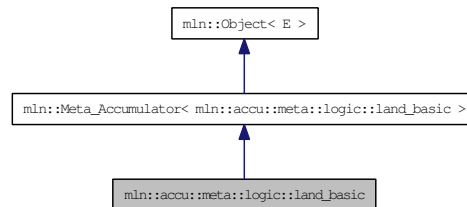
Meta accumulator for [land](#).

10.25 mln::accu::meta::logic::land_basic Struct Reference

Meta accumulator for [land_basic](#).

```
#include <land_basic.hh>
```

Inheritance diagram for mln::accu::meta::logic::land_basic:



10.25.1 Detailed Description

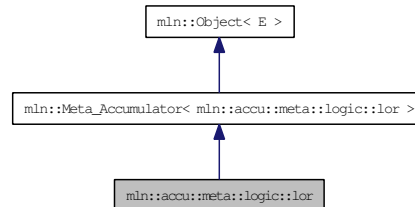
Meta accumulator for [land_basic](#).

10.26 mln::accu::meta::logic::lor Struct Reference

Meta accumulator for [lor](#).

```
#include <lor.hh>
```

Inheritance diagram for mln::accu::meta::logic::lor:



10.26.1 Detailed Description

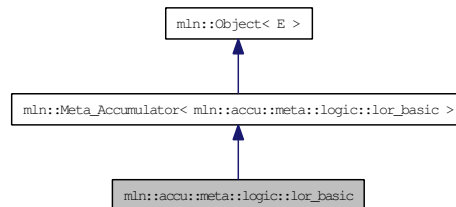
Meta accumulator for [lor](#).

10.27 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:



10.27.1 Detailed Description

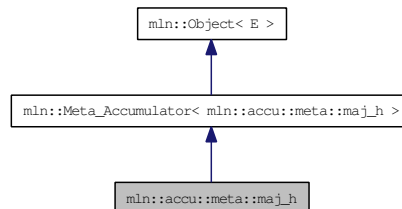
Meta accumulator for [lor_basic](#).

10.28 mln::accu::meta::maj_h Struct Reference

Meta accumulator for [maj_h](#).

```
#include <maj_h.hh>
```

Inheritance diagram for mln::accu::meta::maj_h:



10.28.1 Detailed Description

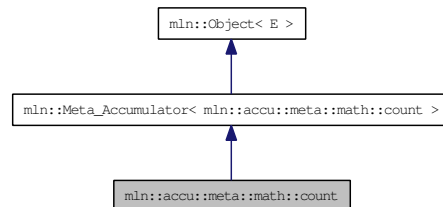
Meta accumulator for [maj_h](#).

10.29 mln::accu::meta::math::count Struct Reference

Meta accumulator for [count](#).

```
#include <count.hh>
```

Inheritance diagram for mln::accu::meta::math::count:



10.29.1 Detailed Description

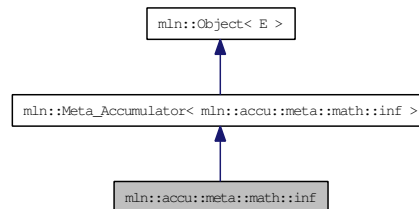
Meta accumulator for [count](#).

10.30 mln::accu::meta::math::inf Struct Reference

Meta accumulator for [inf](#).

```
#include <inf.hh>
```

Inheritance diagram for mln::accu::meta::math::inf:



10.30.1 Detailed Description

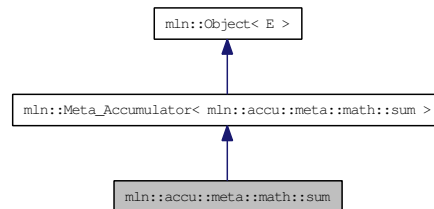
Meta accumulator for [inf](#).

10.31 mln::accu::meta::math::sum Struct Reference

Meta accumulator for [sum](#).

```
#include <sum.hh>
```

Inheritance diagram for mln::accu::meta::math::sum:



10.31.1 Detailed Description

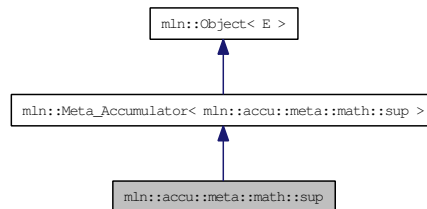
Meta accumulator for [sum](#).

10.32 mln::accu::meta::math::sup Struct Reference

Meta accumulator for [sup](#).

```
#include <sup.hh>
```

Inheritance diagram for mln::accu::meta::math::sup:



10.32.1 Detailed Description

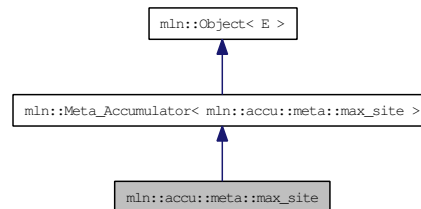
Meta accumulator for [sup](#).

10.33 mln::accu::meta::max_site Struct Reference

Meta accumulator for [max_site](#).

```
#include <max_site.hh>
```

Inheritance diagram for mln::accu::meta::max_site:



10.33.1 Detailed Description

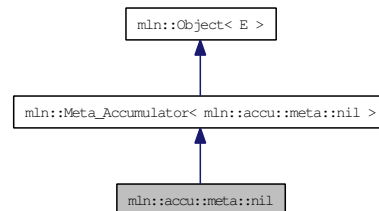
Meta accumulator for [max_site](#).

10.34 mln::accu::meta::nil Struct Reference

Meta accumulator for [nil](#).

```
#include <nil.hh>
```

Inheritance diagram for mln::accu::meta::nil:



10.34.1 Detailed Description

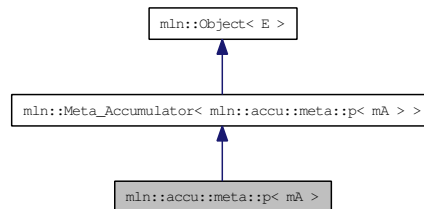
Meta accumulator for [nil](#).

10.35 mln::accu::meta::p< mA > Struct Template Reference

Meta accumulator for [p](#).

```
#include <p.hh>
```

Inheritance diagram for `mln::accu::meta::p< mA >`:



10.35.1 Detailed Description

```
template<typename mA> struct mln::accu::meta::p< mA >
```

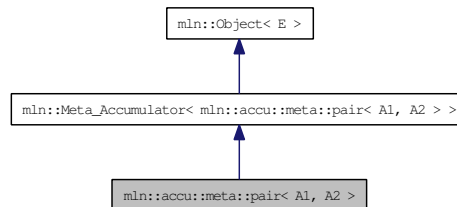
Meta accumulator for [p](#).

10.36 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 >:



10.36.1 Detailed Description

```
template<typename A1, typename A2> struct mln::accu::meta::pair< A1, A2 >
```

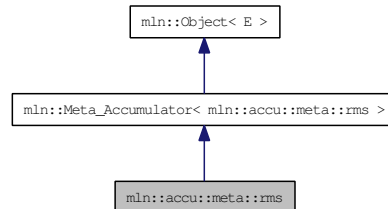
Meta accumulator for [pair](#).

10.37 mln::accu::meta::rms Struct Reference

Meta accumulator for [rms](#).

```
#include <rms.hh>
```

Inheritance diagram for mln::accu::meta::rms:



10.37.1 Detailed Description

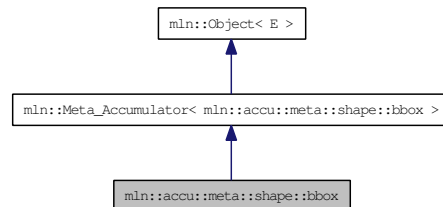
Meta accumulator for [rms](#).

10.38 mln::accu::meta::shape::bbox Struct Reference

Meta accumulator for [bbox](#).

```
#include <bbox.hh>
```

Inheritance diagram for mln::accu::meta::shape::bbox:



10.38.1 Detailed Description

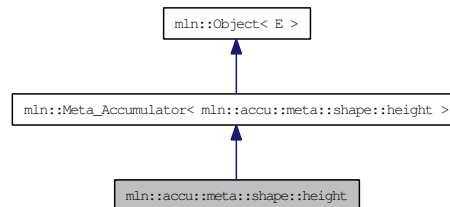
Meta accumulator for [bbox](#).

10.39 mln::accu::meta::shape::height Struct Reference

Meta accumulator for [height](#).

```
#include <height.hh>
```

Inheritance diagram for mln::accu::meta::shape::height:



10.39.1 Detailed Description

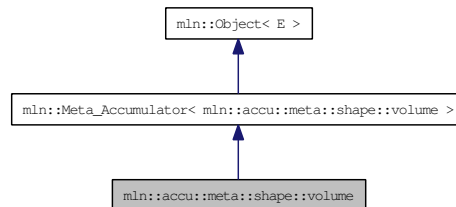
Meta accumulator for [height](#).

10.40 mln::accu::meta::shape::volume Struct Reference

Meta accumulator for [volume](#).

```
#include <volume.hh>
```

Inheritance diagram for mln::accu::meta::shape::volume:



10.40.1 Detailed Description

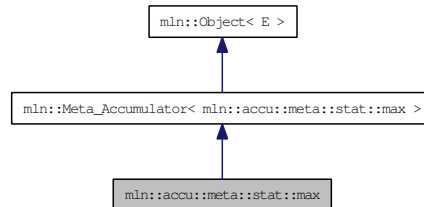
Meta accumulator for [volume](#).

10.41 mln::accu::meta::stat::max Struct Reference

Meta accumulator for [max](#).

```
#include <max.hh>
```

Inheritance diagram for mln::accu::meta::stat::max:



10.41.1 Detailed Description

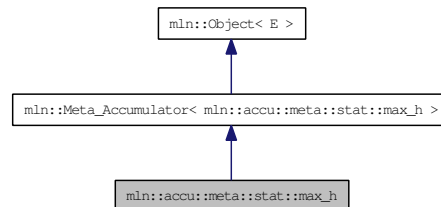
Meta accumulator for [max](#).

10.42 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:



10.42.1 Detailed Description

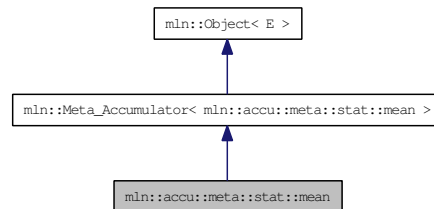
Meta accumulator for [max](#).

10.43 mln::accu::meta::stat::mean Struct Reference

Meta accumulator for [mean](#).

```
#include <mean.hh>
```

Inheritance diagram for mln::accu::meta::stat::mean:



10.43.1 Detailed Description

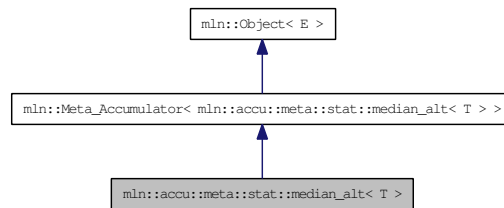
Meta accumulator for [mean](#).

10.44 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 >:



10.44.1 Detailed Description

```
template<typename T> struct mln::accu::meta::stat::median_alt< T >
```

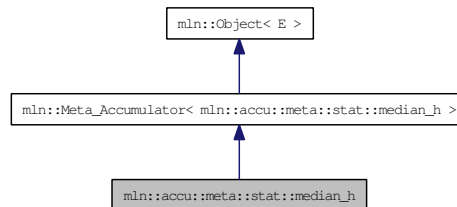
Meta accumulator for [median_alt](#).

10.45 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:



10.45.1 Detailed Description

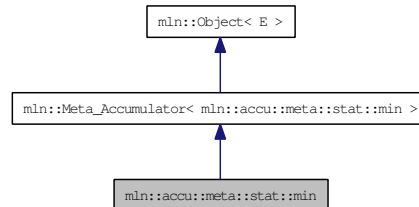
Meta accumulator for [median_h](#).

10.46 mln::accu::meta::stat::min Struct Reference

Meta accumulator for [min](#).

```
#include <min.hh>
```

Inheritance diagram for mln::accu::meta::stat::min:



10.46.1 Detailed Description

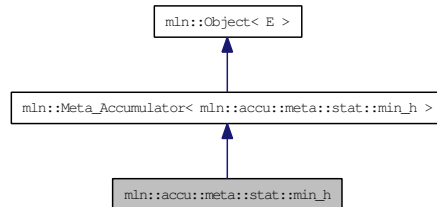
Meta accumulator for [min](#).

10.47 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:



10.47.1 Detailed Description

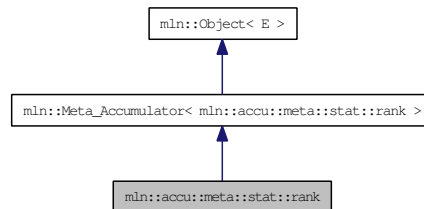
Meta accumulator for [min](#).

10.48 mln::accu::meta::stat::rank Struct Reference

Meta accumulator for [rank](#).

```
#include <rank.hh>
```

Inheritance diagram for mln::accu::meta::stat::rank:



10.48.1 Detailed Description

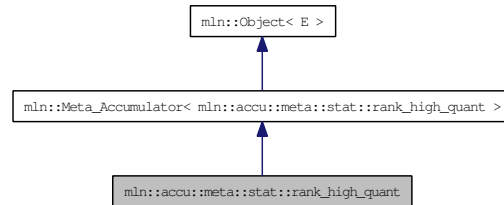
Meta accumulator for [rank](#).

10.49 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:



10.49.1 Detailed Description

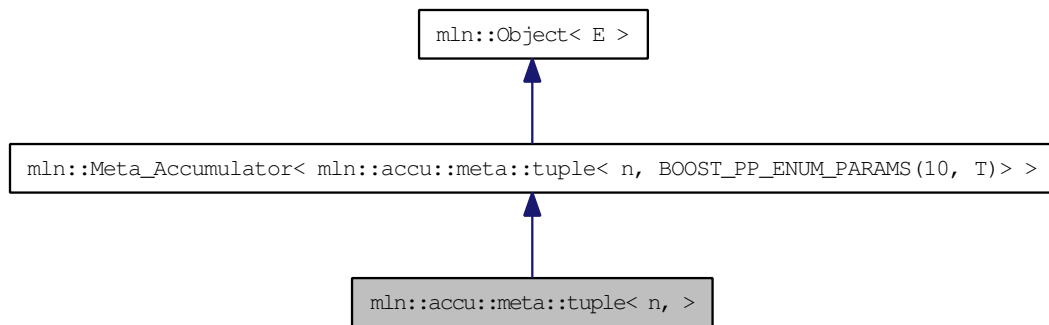
Meta accumulator for [rank_high_quant](#).

10.50 mln::accu::meta::tuple< n, > Struct Template Reference

Meta accumulator for [tuple](#).

```
#include <tuple.hh>
```

Inheritance diagram for mln::accu::meta::tuple< n, >:



10.50.1 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, >
```

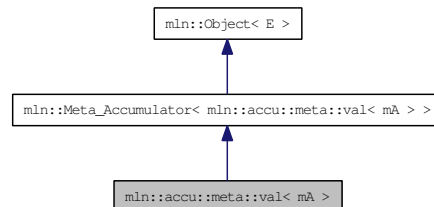
Meta accumulator for [tuple](#).

10.51 mln::accu::meta::val< mA > Struct Template Reference

Meta accumulator for [val](#).

```
#include <v.hh>
```

Inheritance diagram for mln::accu::meta::val< mA >:



10.51.1 Detailed Description

```
template<typename mA> struct mln::accu::meta::val< mA >
```

Meta accumulator for [val](#).

10.52 mln::accu::nil< T > Struct Template Reference

Define an accumulator that does nothing.

```
#include <nil.hh>
```

Inherits mln::accu::internal::base< mln::util::ignore, mln::accu::nil< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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.
- [util::ignore to_result](#) () const
Get the [value](#) of the accumulator.
- void [init](#) ()
Manipulators.

10.52.1 Detailed Description

```
template<typename T> struct mln::accu::nil< T >
```

Define an accumulator that does nothing.

10.52.2 Member Function Documentation

10.52.2.1 `template<typename T> void mln::accu::nil< T >::init () [inline]`

Manipulators.

10.52.2.2 `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.

10.52.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.52.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.52.2.5 `template<typename T> util::ignore mln::accu::nil< T >::to_result () const [inline]`

Get the [value](#) of the accumulator.

10.53 mln::accu::p< A > Struct Template Reference

Generic `p` of accumulators.

```
#include <p.hh>
```

Inherits `mln::accu::internal::base< const A::result &, mln::accu::p< A > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `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`.
- `const A::result & to_result () const`
Get the `value` of the accumulator.
- `void init ()`
Manipulators.

10.53.1 Detailed Description

```
template<typename A> struct mln::accu::p< A >
```

Generic `p` of accumulators.

The parameter `V` is the type of values.

10.53.2 Member Function Documentation

10.53.2.1 `template<typename A> void mln::accu::p< A >::init () [inline]`

Manipulators.

10.53.2.2 `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.

10.53.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.53.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.53.2.5 `template<typename A> const A::result & mln::accu::p< A >::to_result () const [inline]`

Get the [value](#) of the accumulator.

10.54 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

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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.
- std::pair< typename A1::result, typename A2::result > [to_result](#) () const
Get the [value](#) of the accumulator.
- void [init](#) ()
Manipulators.

10.54.1 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.

10.54.2 Member Function Documentation

10.54.2.1 template<typename A1, typename A2, typename T> void mln::accu::pair< A1, A2, T >::init () [inline]

Manipulators.

10.54.2.2 `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.

10.54.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References `mln::mln_exact()`.

10.54.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.54.2.5 `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.

10.55 mln::accu::rms< T, V > Struct Template Reference

Generic root mean square accumulator class.

```
#include <rms.hh>
```

Inherits mln::accu::internal::base< V, mln::accu::rms< T, V > >.

Public Member Functions

- bool `is_valid ()` const
Check whether this `accu` is able to return a result.
- 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`.
- V `to_result ()` const
Get the `value` of the accumulator.
- void `init ()`
Manipulators.

10.55.1 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`.

10.55.2 Member Function Documentation

10.55.2.1 `template<typename T, typename V> void mln::accu::rms< T, V >::init ()` [inline]

Manipulators.

References mln::literal::zero.

10.55.2.2 `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.

10.55.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.55.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.55.2.5 `template<typename T, typename V> V mln::accu::rms< T, V >::to_result () const [inline]`

Get the [value](#) of the accumulator.

10.56 mln::accu::shape::bbox< P > Struct Template Reference

Generic bounding [box](#) accumulator class.

```
#include <bbox.hh>
```

Inherits mln::accu::internal::base< const mln::box< P > &, mln::accu::shape::bbox< P > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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.
- const [box](#)< P > & [to_result](#) () const
Get the [value](#) of the accumulator.
- void [init](#) ()
Manipulators.

10.56.1 Detailed Description

```
template<typename P> struct mln::accu::shape::bbox< P >
```

Generic bounding [box](#) accumulator class.

The parameter P is the type of points.

10.56.2 Member Function Documentation

10.56.2.1 `template<typename P> void mln::accu::shape::bbox< P >::init () [inline]`

Manipulators.

10.56.2.2 `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.

10.56.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.56.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.56.2.5 `template<typename P> const box< P > & mln::accu::shape::bbox< P >::to_result () const [inline]`

Get the [value](#) of the accumulator.

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

10.57 mln::accu::shape::height< I > Struct Template Reference

Height accumulator.

```
#include <height.hh>
```

Inherits mln::accu::internal::base< unsigned, mln::accu::shape::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.
- 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.
- 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.

10.57.1 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.

10.57.2 Member Typedef Documentation

10.57.2.1 `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`

10.57.2.2 `template<typename I> typedef argument::value mln::accu::shape::height< I >::value`

The [value](#) type associated to the [pixel](#) type.

10.57.3 Member Function Documentation

10.57.3.1 `template<typename I> void mln::accu::shape::height< I >::init () [inline]`

Manipulators.

10.57.3.2 `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.

10.57.3.3 `template<typename I> void mln::accu::shape::height< I >::set_value (unsigned h) [inline]`

Force the [value](#) of the counter to *h*.

10.57.3.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.57.3.5 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take *n* times the [value](#) *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.57.3.6 `template<typename I> unsigned mln::accu::shape::height< I >::to_result () const`
`[inline]`

Get the [value](#) of the accumulator.

10.58 mln::accu::shape::volume< I > Struct Template Reference

Volume accumulator class.

```
#include <volume.hh>
```

Inherits mln::accu::internal::base< unsigned, mln::accu::shape::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.
- 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.
- 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.

10.58.1 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.

10.58.2 Member Typedef Documentation

10.58.2.1 `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](#)

10.58.2.2 `template<typename I> typedef argument::value mln::accu::shape::volume< I >::value`

The [value](#) type associated to the [pixel](#) type.

10.58.3 Member Function Documentation

10.58.3.1 `template<typename I> void mln::accu::shape::volume< I >::init () [inline]`

Manipulators.

References [mln::literal::zero](#).

10.58.3.2 `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.

10.58.3.3 `template<typename I> void mln::accu::shape::volume< I >::set_value (unsigned v) [inline]`

Force the [value](#) of the counter to *v*.

References [mln::literal::zero](#).

10.58.3.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References [mln::mln_exact\(\)](#).

10.58.3.5 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take *n* times the [value](#) *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.58.3.6 `template<typename I> unsigned mln::accu::shape::volume< I >::to_result () const`
[inline]

Get the [value](#) of the accumulator.

10.59 mln::accu::site_set::rectangularity< P > Class Template Reference

Compute the [rectangularity](#) of a site [set](#).

```
#include <rectangularity.hh>
```

Inherits mln::accu::internal::couple< mln::accu::shape::bbox< P >, mln::accu::math::count< P >, float, mln::accu::site_set::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.

10.59.1 Detailed Description

```
template<typename P> class mln::accu::site_set::rectangularity< P >
```

Compute the [rectangularity](#) of a site [set](#).

10.59.2 Constructor & Destructor Documentation

10.59.2.1 template<typename P> mln::accu::site_set::rectangularity< P >::rectangularity ()
[inline]

Constructor.

10.59.3 Member Function Documentation

10.59.3.1 `template<typename P> rectangularity< P >::A2::result
mln::accu::site_set::rectangularity< P >::area () const` `[inline]`

Return the site [set](#) area.

10.59.3.2 `template<typename P> rectangularity< P >::A1::result
mln::accu::site_set::rectangularity< P >::bbox () const` `[inline]`

Return the site [set](#) bounding [box](#).

10.59.3.3 `template<typename E> template<typename T> void mln::Accumulator< E
>::take_as_init (const T & t)` `[inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.59.3.4 `template<typename E> template<typename T> void mln::Accumulator< E
>::take_n_times (unsigned n, const T & t)` `[inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.59.3.5 `template<typename P> rectangularity< P >::result mln::accu::site_-
set::rectangularity< P >::to_result () const` `[inline]`

Return the [rectangularity value](#).

10.60 `mln::accu::stat::deviation< T, S, M >` Struct Template Reference

Generic standard `deviation` accumulator class.

```
#include <deviation.hh>
```

Inherits `mln::accu::internal::base< M, mln::accu::stat::deviation< T, S, M > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `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`.
- `M to_result () const`
Get the `value` of the accumulator.
- `void init ()`
Manipulators.

10.60.1 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`.

10.60.2 Member Function Documentation

10.60.2.1 `template<typename T, typename S, typename M> void mln::accu::stat::deviation< T, S, M >::init () [inline]`

Manipulators.

10.60.2.2 `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.

10.60.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References `mln::mln_exact()`.

10.60.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.60.2.5 `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.

10.61 mln::accu::stat::max< T > Struct Template Reference

Generic [max](#) accumulator class.

```
#include <max.hh>
```

Inherits mln::accu::internal::base< const T &, mln::accu::stat::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.
- 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.
- 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.

10.61.2 Member Function Documentation

10.61.2.1 `template<typename T> void mln::accu::stat::max< T >::init ()` `[inline]`

Manipulators.

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.

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*.

10.61.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t)` [inline, inherited]

Take as initialization the [value](#) *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.61.2.5 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t)` [inline, inherited]

Take *n* times the [value](#) *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.61.2.6 `template<typename T> const T & mln::accu::stat::max< T >::to_result () const`
[inline]

Get the [value](#) of the accumulator.

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 mln::accu::internal::base< const V &, mln::accu::stat::max_h< V > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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.
- 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.

10.62.2 Member Function Documentation

10.62.2.1 template<typename V> void mln::accu::stat::max_h< V >::init () [inline]

Manipulators.

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.

10.62.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References [mln::mln_exact\(\)](#).

10.62.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References [mln::mln_exact\(\)](#).

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.

10.63 mln::accu::stat::mean< T, S, M > Struct Template Reference

Generic [mean](#) accumulator class.

```
#include <mean.hh>
```

Inherits mln::accu::internal::base< M, mln::accu::stat::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.
- 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.
- 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.

10.63.2 Member Function Documentation

```
10.63.2.1 template<typename T, typename S, typename M> accu::math::count< T >::result
mln::accu::stat::mean< T, S, M >::count () const [inline]
```

Get the cardinality.

10.63.2.2 `template<typename T, typename S, typename M> void mln::accu::stat::mean< T, S, M >::init () [inline]`

Manipulators.

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.

10.63.2.4 `template<typename T, typename S, typename M> accu::math::sum< T >::result mln::accu::stat::mean< T, S, M >::sum () const [inline]`

Get the sum of values.

10.63.2.5 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References `mln::mln_exact()`.

10.63.2.6 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.63.2.7 `template<typename T, typename S, typename M> M mln::accu::stat::mean< T, S, M >::to_result () const [inline]`

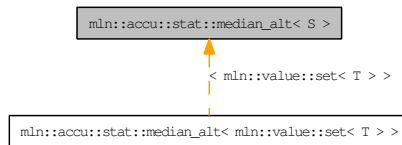
Get the [value](#) of the accumulator.

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.
- `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`.
- `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`.

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.

10.64.2.2 `template<typename S> void mln::accu::stat::median_alt< S >::take (const argument & t) [inline]`

Manipulators.

10.64.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.64.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.64.2.5 `template<typename S> const median_alt< S >::argument & mln::accu::stat::median_alt< S >::to_result () const [inline]`

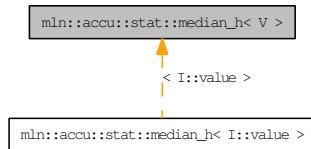
Get the [value](#) of the accumulator.

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.
- 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.
- 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.

10.65.2 Member Function Documentation

10.65.2.1 template<typename V> void mln::accu::stat::median_h< V >::init () [inline]

Manipulators.

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.

10.65.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.65.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.65.2.5 `template<typename V> const median_h< V >::argument & mln::accu::stat::median_h< V >::to_result () const [inline]`

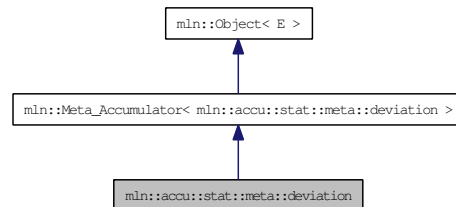
Get the [value](#) of the accumulator.

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](#).

10.67 mln::accu::stat::min< T > Struct Template Reference

Generic [min](#) accumulator class.

```
#include <min.hh>
```

Inherits mln::accu::internal::base< const T &, mln::accu::stat::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.
- 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.
- 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.

10.67.2 Member Function Documentation

10.67.2.1 template<typename T> void mln::accu::stat::min< T >::init () [inline]

Manipulators.

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.

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*.

10.67.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t)` [inline, inherited]

Take as initialization the [value](#) *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.67.2.5 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t)` [inline, inherited]

Take *n* times the [value](#) *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

References `mln::mln_exact()`.

10.67.2.6 `template<typename T> const T & mln::accu::stat::min< T >::to_result () const`
[inline]

Get the [value](#) of the accumulator.

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 `mln::accu::internal::base< const V &, mln::accu::stat::min_h< V > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this [accu](#) is able to return a result.
- `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`.
- `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`.

10.68.2 Member Function Documentation

10.68.2.1 `template<typename V> void mln::accu::stat::min_h< V >::init ()` `[inline]`

Manipulators.

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.

10.68.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References `mln::mln_exact()`.

10.68.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.68.2.5 `template<typename V> const min_h< V >::argument & mln::accu::stat::min_h< V >::to_result () const [inline]`

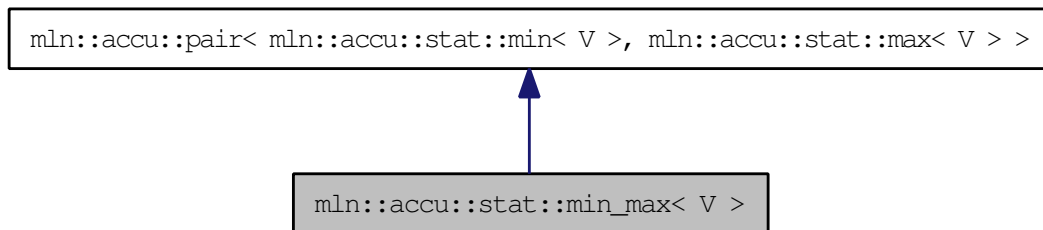
Get the [value](#) of the accumulator.

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

- bool [is_valid](#) () const

Check whether this [accu](#) is able to return a result.

- 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.

- std::pair< typename A1::result, typename A2::result > [to_result](#) () const

Get the [value](#) of the accumulator.

- void [init](#) ()

Manipulators.

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.

10.69.2 Member Function Documentation

10.69.2.1 `template<typename A1, typename A2, typename T> void mln::accu::pair< A1, A2, T >::init () [inline, inherited]`

Manipulators.

10.69.2.2 `template<typename A1, typename A2, typename T> bool mln::accu::pair< A1, A2, T >::is_valid () const [inline, inherited]`

Check whether this [accu](#) is able to return a result.

Always true here.

10.69.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References `mln::mln_exact()`.

10.69.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.69.2.5 `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, 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 mln::accu::internal::base< const T &, mln::accu::stat::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](#).
- 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.
- 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.

10.70.2 Member Function Documentation

10.70.2.1 template<typename T> void mln::accu::stat::rank< T >::init () [inline]

Manipulators.

Referenced by mln::morpho::impl::generic::rank_filter().

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.

10.70.2.3 `template<typename T> unsigned mln::accu::stat::rank< T >::k () const` `[inline]`

Give the [rank](#).

10.70.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t)` `[inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.70.2.5 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t)` `[inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.70.2.6 `template<typename T> const T & mln::accu::stat::rank< T >::to_result () const` `[inline]`

Get the [value](#) of the accumulator.

10.71 mln::accu::stat::rank< bool > Struct Template Reference

[rank](#) accumulator class for Boolean.

```
#include <rank_bool.hh>
```

Inherits mln::accu::internal::base< bool, mln::accu::stat::rank< bool > >.

Public Member Functions

- `bool is_valid () const`
Check whether this [accu](#) is able to return a result.
- `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`.
- `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.

10.71.2 Member Function Documentation

10.71.2.1 `void mln::accu::stat::rank< bool >::init () [inline]`

Manipulators.

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.

10.71.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References `mln::mln_exact()`.

10.71.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.71.2.5 `bool mln::accu::stat::rank< bool >::to_result () const [inline]`

Get the [value](#) of the accumulator.

10.72 mln::accu::stat::rank_high_quant< T > Struct Template Reference

Generic [rank](#) accumulator class.

```
#include <rank_high_quant.hh>
```

Inherits mln::accu::internal::base< const T &, mln::accu::stat::rank_high_quant< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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.
- 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.

10.72.2 Member Function Documentation

10.72.2.1 template<typename T> void mln::accu::stat::rank_high_quant< T >::init ()
[inline]

Manipulators.

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.

10.72.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References `mln::mln_exact()`.

10.72.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

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.

10.73 mln::accu::stat::var< T > Struct Template Reference

Var accumulator class.

```
#include <var.hh>
```

Inherits mln::accu::internal::base< mln::algebra::mat< T::dim, T::dim, float >, mln::accu::stat::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.
- 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
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

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.

10.73.3 Member Function Documentation

10.73.3.1 `template<typename T> void mln::accu::stat::var< T >::init () [inline]`

Manipulators.

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.

10.73.3.3 `template<typename T> var< T >::mean_t mln::accu::stat::var< T >::mean () const [inline]`

Get the [mean](#) vector.

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.

10.73.3.5 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.73.3.6 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

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](#)).

References `mln::literal::zero`.

Referenced by `mln::accu::stat::var< T >::variance()`.

10.73.3.8 `template<typename T> var< T >::result mln::accu::stat::var< T >::variance () const`
[inline]

Get the [variance](#) matrix.

References `mln::accu::stat::var< T >::to_result()`.

10.74 `mln::accu::stat::variance< T, S, R >` Struct Template Reference

Variance accumulator class.

```
#include <variance.hh>
```

Inherits `mln::accu::internal::base< R, mln::accu::stat::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`.
- `template<typename 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.
- `void take_as_init (const argument &t)`
Take as initialization the `value t`.

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`.

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.

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.

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](#).

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.

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](#).

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](#).

10.74.2.7 `template<typename T, typename S, typename R> void mln::accu::stat::variance< T, S, R >::take_as_init (const argument & t) [inline]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented from `mln::Accumulator< E >`.

10.74.2.8 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take *n* times the [value](#) *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

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](#)).

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](#).

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 mln::accu::internal::base< boost::tuple< BOOST_PP_REPEAT(10, RESULT_ACCU, Le Ricard ya que ca de vrai!) >, mln::accu::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.
- `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.
- `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, type-  
name T, boost::tuples::null_type)> struct mln::accu::tuple< A, n, >
```

Generic [tuple](#) of accumulators.

The parameter T is the type of values.

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.

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.

10.75.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.75.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

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.

10.76 mln::accu::val< A > Struct Template Reference

Generic [val](#) of accumulators.

```
#include <v.hh>
```

Inherits `mln::accu::internal::base< const A::result &, mln::accu::val< A > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this [accu](#) is able to return a result.
- `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`.
- `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.

10.76.2 Member Function Documentation

10.76.2.1 `template<typename A> void mln::accu::val< A >::init () [inline]`

Manipulators.

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.

10.76.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.76.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.76.2.5 `template<typename A> const A::result & mln::accu::val< A >::to_result () const [inline]`

Get the [value](#) of the accumulator.

10.77 mln::Accumulator< E > Struct Template Reference

Base class for implementation of accumulators.

```
#include <accumulator.hh>
```

Inherits [mln::Proxy< E >](#).

Inherited by [mln::accu::internal::base< R, 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.

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) [inline]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

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) [inline]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

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 (const mat< d+1, d+1, T > &x)`
Constructor with the underlying matrix.
- `h_mat ()`
Constructor without argument.
- `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.

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).

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.

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.

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.

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.

10.79 `mln::algebra::h_vec< d, C >` Struct 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` (`const vec< d+1, C > &other`)

Constructor with the underlying vector.

- `h_vec` ()

Constructor without argument.

- `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> struct mln::algebra::h_vec< d, C >
```

N-Dimensional vector with homogeneous coordinates.

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 ($\text{dim} + 1$).

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.

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.

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.

10.79.4.2 `template<unsigned n, typename T> mat< 1, n, T > mln::algebra::vec< n, T >::t () const` [inline, inherited]

Transposition.

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.

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)` [inline, static, inherited]

Origin [value](#).

10.79.5.2 `template<unsigned n, typename T> const vec< n, T > mln::algebra::vec< n, T >::zero = all_to(0)` [inline, static, inherited]

Zero [value](#).

10.80 mln::bkd_pixter1d< I > Class Template Reference

Backward [pixel](#) iterator on a 1-D image with [border](#).

```
#include <pixter1d.hh>
```

Inherits mln::internal::backward_pixel_iterator_base_< I, mln::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](#).

10.80.2 Member Typedef Documentation

10.80.2.1 template<typename I> typedef I mln::bkd_pixter1d< I >::image

[Image](#) type.

10.80.3 Constructor & Destructor Documentation

10.80.3.1 template<typename I> mln::bkd_pixter1d< I >::bkd_pixter1d (I & *image*)
[inline]

Constructor.

Parameters:

← *image* The image this [pixel](#) iterator is bound to.

10.80.4 Member Function Documentation

10.80.4.1 `template<typename E> void mln::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.

10.81 mln::bkd_pixter2d< I > Class Template Reference

Backward [pixel](#) iterator on a 2-D image with [border](#).

```
#include <pixter2d.hh>
```

Inherits mln::internal::backward_pixel_iterator_base_< I, mln::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](#).

10.81.2 Member Typedef Documentation

10.81.2.1 template<typename I> typedef I mln::bkd_pixter2d< I >::image

[Image](#) type.

10.81.3 Constructor & Destructor Documentation

10.81.3.1 template<typename I> mln::bkd_pixter2d< I >::bkd_pixter2d (I & *image*)
[inline]

Constructor.

Parameters:

← *image* The image this [pixel](#) iterator is bound to.

10.81.4 Member Function Documentation

10.81.4.1 `template<typename E> void mln::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.

10.82 mln::bkd_pixter3d< I > Class Template Reference

Backward [pixel](#) iterator on a 3-D image with [border](#).

```
#include <pixter3d.hh>
```

Inherits mln::internal::backward_pixel_iterator_base_< I, mln::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](#).

10.82.2 Member Typedef Documentation

10.82.2.1 template<typename I> typedef I mln::bkd_pixter3d< I >::image

[Image](#) type.

10.82.3 Constructor & Destructor Documentation

10.82.3.1 template<typename I> mln::bkd_pixter3d< I >::bkd_pixter3d (I & *image*)
[inline]

Constructor.

Parameters:

← *image* The image this [pixel](#) iterator is bound to.

10.82.4 Member Function Documentation

10.82.4.1 `template<typename E> void mln::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:

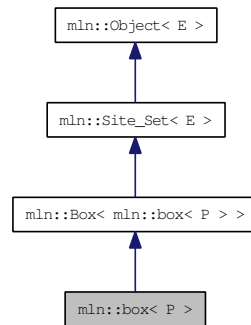
The iterator is valid.

10.83 mln::box< P > Struct 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 E & bbox () const`
Give the bounding `box` of this site `set`.
- `box (const site &pmin, const site &pmax)`

Constructor of a *box* going from *pmin* to *pmax*.

- `box ()`
Constructor without argument.
- `P center () const`
Return the approximated central site of this *box*.
- `void crop_wrt (const box< P > &b)`
Crop this *bbox* in order to fit in the reference *box* *b*.
- `void enlarge (unsigned dim, unsigned b)`
Enlarge the *box* with a *border* *b* for dimension *dim*.
- `void enlarge (unsigned b)`
Enlarge the *box* with a *border* *b*.
- `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.
- `unsigned nsites () const`
Give the number of sites of this *box*.
- `P & pmax ()`
Reference to the maximum *point*.
- `P pmax () const`
Maximum *point*.
- `P & pmin ()`
Reference to the minimum *point*.
- `P pmin () const`
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 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 P>`
`std::ostream & operator<< (std::ostream &ostr, const box< P > &b)`
Print a generic box b 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.

10.83.1 Detailed Description

`template<typename P> struct mln::box< P >`

Generic [box](#) class: site [set](#) containing points of a regular [grid](#).

Parameter `P` is the corresponding type of [point](#).

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.

10.83.2.2 `template<typename P> typedef P mln::box< P >::element`

Element associated type.

10.83.2.3 `template<typename P> typedef box_fwd_piter_<P> mln::box< P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

10.83.2.4 `template<typename P> typedef fwd_piter mln::box< P >::piter`

[Site_Iterator](#) associated type.

10.83.2.5 `template<typename P> typedef P mln::box< P >::psite`

Psite associated type.

10.83.2.6 `template<typename P> typedef P mln::box< P >::site`

[Site](#) associated type.

10.83.3 Member Enumeration Documentation

10.83.3.1 `template<typename P> anonymous enum`

Dimension.

10.83.4 Constructor & Destructor Documentation

10.83.4.1 `template<typename P> mln::box< P >::box () [inline]`

Constructor without argument.

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`.

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.

References `mln::literal::origin`.

10.83.5 Member Function Documentation

10.83.5.1 `template<typename E> const E & mln::Box< E >::bbox () const` `[inline, 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> P mln::box< P >::center () const` `[inline]`

Return the approximated central site of this [box](#).

References `mln::box< P >::is_valid()`.

10.83.5.3 `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`.

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

Referenced by `mln::make_debug_graph_image()`.

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`.

References `mln::box< P >::is_valid()`.

10.83.5.5 `template<typename P> void mln::box< P >::enlarge (unsigned b) [inline]`

Enlarge the `box` with a `border` `b`.

References `mln::box< P >::is_valid()`.

Referenced by `mln::registration::icp()`.

10.83.5.6 `template<typename P> bool mln::box< P >::has (const P & p) const [inline]`

Test if `p` belongs to the `box`.

Parameters:

← `p` A `point` site.

References `mln::box< P >::is_valid()`.

Referenced by `mln::morpho::line_gradient()`.

10.83.5.7 `template<typename E> bool mln::Box< E >::is_empty () const [inline, inherited]`

Test if this `box` is empty.

10.83.5.8 `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`.

References `mln::util::ord_weak()`.

Referenced by `mln::box< P >::box()`, `mln::box< P >::center()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::box< P >::enlarge()`, `mln::box< P >::has()`, `mln::box< P >::pmax()`, `mln::box< P >::pmin()`, and `mln::box< P >::to_larger()`.

10.83.5.9 `template<typename E> unsigned mln::Box< E >::len (unsigned i) const [inline, 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.

10.83.5.10 `template<typename P> std::size_t mln::box< P >::memory_size () const [inline]`

Return the size of this site `set` in memory.

10.83.5.11 `template<typename E> unsigned mln::Box< E >::nsites () const` [inline, 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.

Referenced by `mln::morpho::line_gradient()`.

10.83.5.12 `template<typename P> P & mln::box< P >::pmax ()` [inline]

Reference to the maximum [point](#).

10.83.5.13 `template<typename P> P mln::box< P >::pmax () const` [inline]

Maximum [point](#).

References `mln::box< P >::is_valid()`.

Referenced by `mln::box< P >::crop_wrt()`, `mln::make::image3d()`, `mln::larger_than()`, and `mln::io::fld::load()`.

10.83.5.14 `template<typename P> P & mln::box< P >::pmin ()` [inline]

Reference to the minimum [point](#).

10.83.5.15 `template<typename P> P mln::box< P >::pmin () const` [inline]

Minimum [point](#).

References `mln::box< P >::is_valid()`.

Referenced by `mln::box< P >::crop_wrt()`, `mln::make::image3d()`, `mln::larger_than()`, and `mln::io::fld::load()`.

10.83.5.16 `template<typename P> box< P > mln::box< P >::to_larger (unsigned b) const` [inline]

Give a larger [box](#).

References `mln::box< P >::is_valid()`.

10.83.6 Friends And Related Function Documentation

10.83.6.1 `template<typename Sl, typename Sr> p_set< typename Sl::site > diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Set theoretic difference of lhs and rhs.

10.83.6.2 `template<typename Sl, typename Sr> p_set< typename Sl::site > inter (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Intersection between a couple of [point](#) sets.

10.83.6.3 `template<typename Sl, typename Sr> bool operator< (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Strict inclusion [test](#) between site sets `lhs` and `rhs`.

Parameters:

- ← *lhs* A site [set](#) (strictly included?).
- ← *rhs* Another site [set](#) (includer?).

10.83.6.4 `template<typename Bl, typename Br> bool operator< (const Box< Bl > & lhs, const Box< Br > & rhs)` [related, inherited]

Strict inclusion [test](#) between boxes `lhs` and `rhs`.

Parameters:

- ← *lhs* A [box](#) (strictly included?).
- ← *rhs* Another [box](#) (includor?).

10.83.6.5 `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:

- ↔ *ostr* An output stream.
- ← *set* A site [set](#).

Returns:

The modified output stream `ostr`.

10.83.6.6 `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:

- ↔ *ostr* An output stream.
- ← *b* A generic [box](#).

Returns:

The modified output stream `ostr`.

10.83.6.7 `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:

- ← *lhs* A site [set](#) (included?).
- ← *rhs* Another site [set](#) (includer?).

10.83.6.8 `template<typename Bl, typename Br> bool operator<= (const Box< Bl > & lhs, const Box< Br > & rhs)` [related, inherited]

Inclusion [test](#) between boxes `lhs` and `rhs`.

Parameters:

- ← *lhs* A [box](#) (included?).
- ← *rhs* Another [box](#) (includer?).

10.83.6.9 `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:

- ← *lhs* A site [set](#).
- ← *rhs* Another site [set](#).

10.83.6.10 `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`.

10.83.6.11 `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.

10.83.6.12 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [related, inherited]

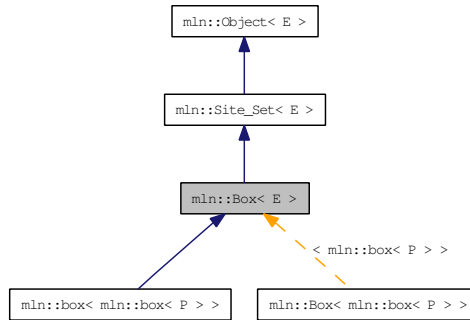
Give the unique [set](#) of `s`.

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 Bl, typename Br>`
`bool operator< (const Box< Bl > &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 into the output stream ostr.
- `template<typename Sl, typename Sr>`
`bool operator<= (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Inclusion test between site sets lhs and rhs.
- `template<typename Bl, typename Br>`
`bool operator<= (const Box< Bl > &lhs, const Box< Br > &rhs)`
Inclusion test between boxes lhs and rhs.
- `template<typename Sl, typename Sr>`
`bool operator== (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Equality test between site sets lhs and rhs.
- `template<typename Sl, typename Sr>`
`p_set< typename Sl::site > sym_diff (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic symmetrical difference of lhs and rhs.
- `template<typename Sl, typename Sr>`
`p_set< typename Sl::site > uni (const Site_Set< Sl > &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.

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.

10.84.2.2 `template<typename E> bool mln::Box< E >::is_empty () const` `[inline]`

Test if this [box](#) is empty.

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 < site::dim`

Warning:

This method is final for all [box](#) classes.

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.

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`.

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.

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:

- ← *lhs* A site [set](#) (strictly included?).
- ← *rhs* Another site [set](#) (includer?).

10.84.3.4 `template<typename Bl, typename Br> bool operator<< (const Box< Bl > & lhs, const Box< Br > & rhs)` [related]

Strict inclusion [test](#) between boxes `lhs` and `rhs`.

Parameters:

- ← *lhs* A [box](#) (strictly included?).
- ← *rhs* Another [box](#) (includor?).

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:

- ↔ *ostr* An output stream.
- ← *set* A site [set](#).

Returns:

The modified output stream `ostr`.

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:

- ← *lhs* A site [set](#) (included?).
- ← *rhs* Another site [set](#) (includer?).

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:

- ← *lhs* A [box](#) (included?).
- ← *rhs* Another [box](#) (includor?).

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:

- ← *lhs* A site [set](#).
- ← *rhs* Another site [set](#).

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`.

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.

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`.

10.85 mln::box_runend_piter< P > Class Template Reference

A generic backward iterator on points by lines.

```
#include <box_runend_piter.hh>
```

Inherits mln::internal::site_set_iterator_base< mln::box< P >, mln::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 lenght 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.

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:

← *b* A `box`.

10.85.3 Member Function Documentation

10.85.3.1 `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.

10.85.3.2 `template<typename P> unsigned mln::box_runend_piter< P >::run_length () const`
[inline]

Give the length of the run.

10.86 mln::box_runstart_piter< P > Class Template Reference

A generic forward iterator on points by lines.

```
#include <box_runstart_piter.hh>
```

Inherits mln::internal::site_set_iterator_base< mln::box< P >, mln::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 lenght 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.

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:

← *b* A [box](#).

10.86.3 Member Function Documentation

10.86.3.1 `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.

10.86.3.2 `template<typename P> unsigned mln::box_runstart_piter< P >::run_length () const`
[inline]

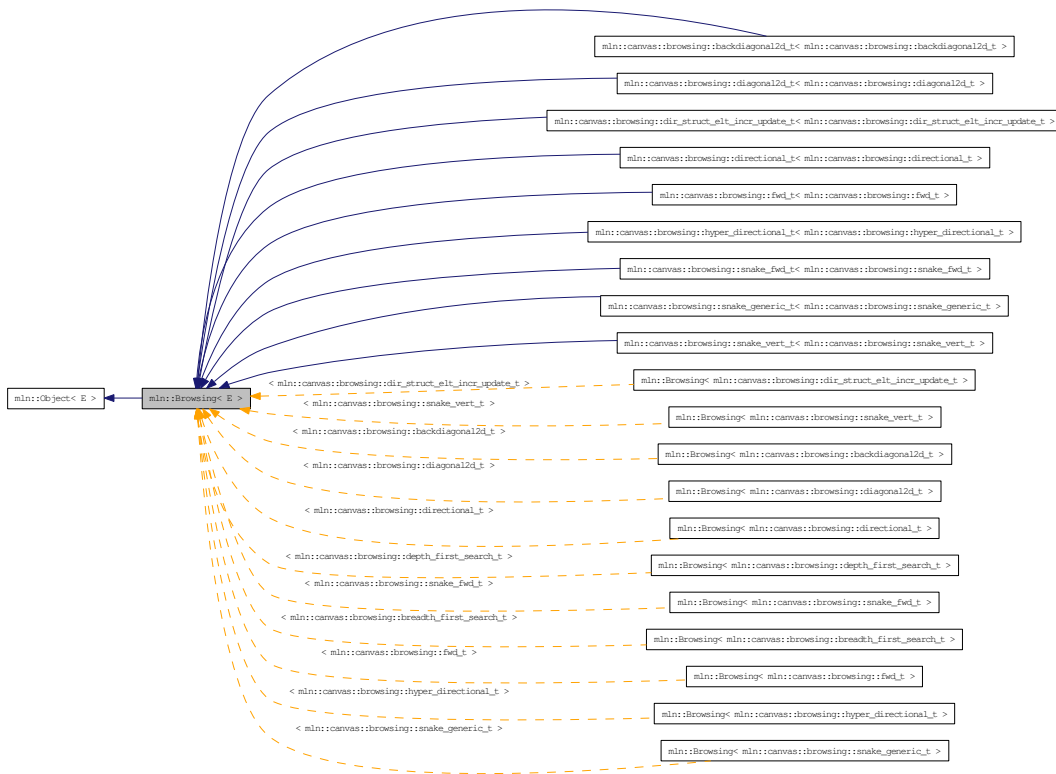
Give the length of the run.

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:

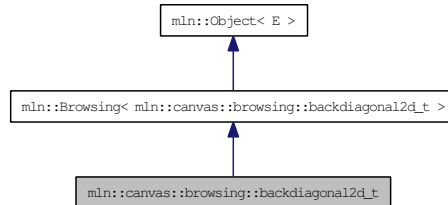
[mln::doc::Browsing](#) for a complete documentation of this class contents.

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
```

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 mln::canvas::browsing::internal::graph_first_search_t< mln::canvas::browsing::breadth_first_search_t, std::queue< T > >.

10.89.1 Detailed Description

Breadth-first search algorithm for [graph](#), on vertices.

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 mln::canvas::browsing::internal::graph_first_search_t< mln::canvas::browsing::depth_first_search_t, std::stack< T > >.

10.90.1 Detailed Description

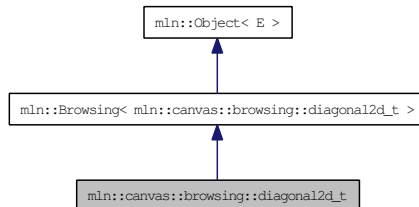
Breadth-first search algorithm for [graph](#), on vertices.

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 :

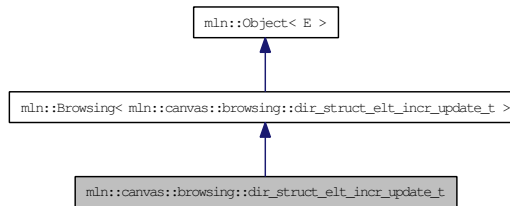
```
| 1 3 6 | 2 5 8 | 4 7 9 L——>
```

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();
```

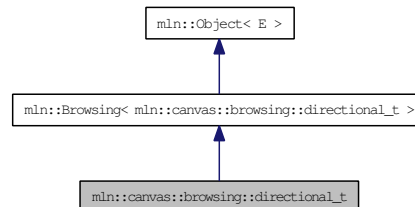
```
void init_line();
void add_point(q)
void remove_point(q)
void next();
void final();
}
```

10.93 mln::canvas::browsing::directional_t Struct Reference

Browsing in a certain direction.

```
#include <directional.hh>
```

Inheritance diagram for mln::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
```

400500600

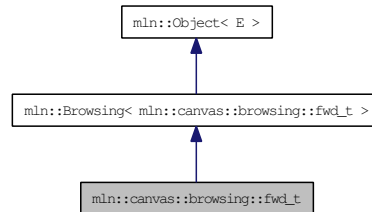
700800900

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:

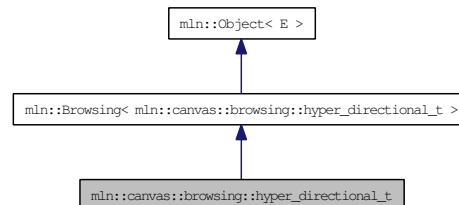
```
{
— as typedef:
I;
—as attributes:
input;
p;
— as method:
void init();
void next();
void final();
}
```

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 :

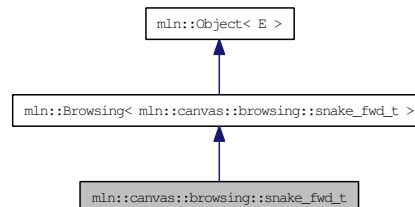
```
{
— as types:
I;
— as attributes:
dim;
dir; // and test dir < dim
input;
p;
— as methods:
void init();
void next();
void final();
}
```

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:

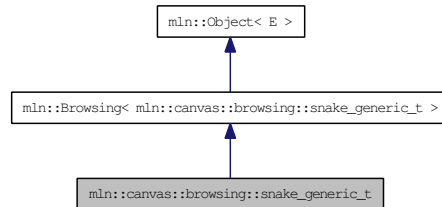
```
{
— as attributes:
input;
p;
— as methods:
void init();
void down();
void fwd();
void bkd();
}
```


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 :

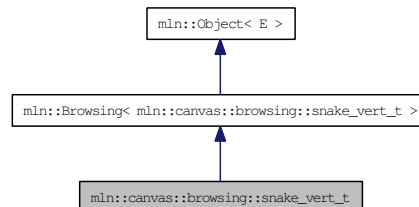
moves[i] is called at each move following the delta psite dps[i]

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 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 up();
void fwd();
}
```

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.

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.

10.101 mln::complex_image< D, G, V > Class Template Reference

[Image](#) based on a complex.

```
#include <complex_image.hh>
```

Inherits mln::internal::image_primary< V, mln::p_complex< D, G >, mln::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

- [lvalue operator\(\)](#) (const [complex_psite](#)< D, G > &p)
Read-write access of face [value](#) at [point](#) site p.
- [rvalue operator\(\)](#) (const [complex_psite](#)< D, G > &p) const
Read-only 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.

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.

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.

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.

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.

10.101.2.5 `template<unsigned D, typename G, typename V> typedef V mln::complex_image< D, G, V >::value`

[Value](#) associated type.

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.

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 od the image.

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.

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.

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.

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.

10.102 mln::complex_neighborhood_bkd_piter< I, G, N > Class Template Reference

Backward iterator on complex neighborhood.

```
#include <complex_neighborhood_piter.hh>
```

Inherits mln::internal::site_relative_iterator_base< N, mln::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.

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.

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.

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.

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.

10.102.4.2 `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.

10.103 mln::complex_neighborhood_fwd_piter< I, G, N > Class Template Reference

Forward iterator on complex neighborhood.

```
#include <complex_neighborhood_piter.hh>
```

Inherits mln::internal::site_relative_iterator_base< N, mln::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.

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.

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.

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.

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.

10.103.4.2 `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.

10.104 mln::complex_psite< D, G > Class Template Reference

Point site associated to a [mln::p_complex](#).

```
#include <complex_psite.hh>
```

Inherits [mln::internal::pseudo_site_base_< const G::site &, mln::complex_psite< D, G > >](#).

Public Member Functions

- void [change_target](#) (const [target](#) &new_target)
Set the target site_set.
- const [target](#) & [site_set](#) () const
Site set manipulators.
- [complex_psite](#) (const [p_complex](#)< D, G > &pc, const [topo::face](#)< D > &face)
- [complex_psite](#) ()
Construction and assignment.
- const [topo::face](#)< D > & [face](#) () const
Face handle manipulators.
- unsigned [face_id](#) () const
Return the id of the face of this psite.
- unsigned [n](#) () const
Return the dimension of the face of this psite.
- void [invalidate](#) ()
Invalidate this psite.
- bool [is_valid](#) () const
Psite manipulators.

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.

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.

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()`.

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.

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.

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.

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.

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?

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.

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.

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 mln::internal::site_relative_iterator_base< W, mln::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](#).

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.

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.

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.

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.

10.105.4.2 `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.

10.106 mln::complex_window_fwd_piter< I, G, W > Class Template Reference

Forward iterator on complex [window](#).

```
#include <complex_window_piter.hh>
```

Inherits mln::internal::site_relative_iterator_base< W, mln::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](#).

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.

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.

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.

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.

10.106.4.2 `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.

10.107 mln::decorated_image< I, D > Struct Template Reference

[Image](#) that can have additional features.

```
#include <decorated_image.hh>
```

Inherits mln::internal::decorated_image_impl_< I, mln::decorated_image< I, D > >, and mln::internal::image_identity< I, I::domain_t, mln::decorated_image< I, D > >.

Package 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.

Package 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.
- [lvalue operator](#)() (const [psite](#) &p)
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.
- [~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.

10.107.2 Member Typedef Documentation

10.107.2.1 `template<typename I, typename D> typedef impl_::lvalue mln::decorated_image< I, D >::lvalue` [package]

Return type of read-write access.

10.107.2.2 `template<typename I, typename D> typedef I ::psite mln::decorated_image< I, D >::psite` [package]

Type of the psite.

10.107.2.3 `template<typename I, typename D> typedef I ::rvalue mln::decorated_image< I, D >::rvalue` [package]

Return type of read-only access.

10.107.2.4 `template<typename I, typename D> typedef decorated_image< tag::image_<I>, tag::data_<D> > mln::decorated_image< I, D >::skeleton` [package]

Skeleton.

10.107.3 Constructor & Destructor Documentation

10.107.3.1 `template<typename I, typename D> mln::decorated_image< I, D >::decorated_image()` [inline, package]

Ctors.

10.107.3.2 `template<typename I, typename D> mln::decorated_image< I, D >::~~decorated_image()` [inline, package]

Dtor.

10.107.4 Member Function Documentation

10.107.4.1 `template<typename I, typename D> D & mln::decorated_image< I, D >::decoration()` [inline, package]

Give the decoration.

10.107.4.2 `template<typename I, typename D> const D & mln::decorated_image< I, D >::decoration () const` [inline, package]

Give the decoration.

10.107.4.3 `template<typename I, typename D> mln::decorated_image< I, D >::operator decorated_image< const I, D > () const` [inline, package]

Const promotion via conversion.

10.107.4.4 `template<typename I, typename D> decorated_image< I, D >::lvalue mln::decorated_image< I, D >::operator() (const psite & p)` [inline, package]

Read-write access of [pixel value](#) at [point](#) site p.

10.107.4.5 `template<typename I, typename D> decorated_image< I, D >::rvalue mln::decorated_image< I, D >::operator() (const psite & p) const` [inline, package]

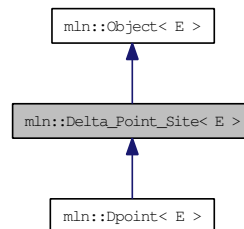
Read-only access of [pixel value](#) at [point](#) site p.

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!

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.

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](#)

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.

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)`

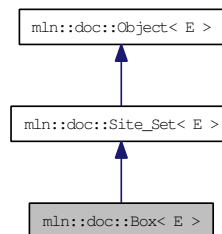
Take into account a argument t (an element).

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](#)

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.

10.111.2.2 `template<typename E> typedef void mln::doc::Site_Set< E >::fwd_piter`
[inherited]

Forward [Site_Iterator](#) associated type.

10.111.2.3 `template<typename E> typedef void mln::doc::Site_Set< E >::psite` [inherited]

PSite associated type.

10.111.2.4 `template<typename E> typedef void mln::doc::Site_Set< E >::site` [inherited]

[Site](#) associated type.

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:

← *p* A [psite](#).

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](#).

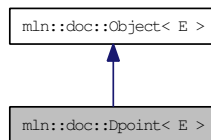
Return the "minimum" [point](#) w.r.t. the ordering between points. For instance, with [mln::box2d](#), this minimum is the top left [point](#) of the [box](#).

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](#)

10.112.2 Member Typedef Documentation

10.112.2.1 template<typename E> typedef void mln::doc::Dpoint< E >::coord

Coordinate associated type.

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](#).

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](#).

10.112.3 Member Enumeration Documentation

10.112.3.1 `template<typename E> anonymous enum`

Enumerator:

dim Dimension of the space.

Invariant:

$dim > 0$

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:

$\leftarrow i$ The coordinate index.

Precondition:

$i < dim$

Returns:

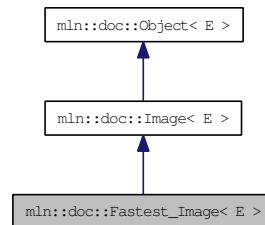
The [value](#) of the *i*-th coordinate.

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 p belongs to the image domain.
- bool **has** (const **psite** &p) const
*Test if the image owns the **point** site p.*
- 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.*
- **lvalue operator**[] (unsigned o)
*Read-write access to the image **value** at offset o.*
- **rvalue operator**[] (unsigned o) const
*Read-only 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".

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`.

10.113.2.2 `template<typename E> typedef void mln::doc::Image< E >::coord` [inherited]

Coordinate associated type.

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`.

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`.

10.113.2.5 `template<typename E> typedef void mln::doc::Image< E >::lvalue` [inherited]

Type returned by the read-write `pixel value` operator.

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](#).

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](#).

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](#).

10.113.2.9 `template<typename E> typedef void mln::doc::Image< E >::rvalue` [inherited]

Type returned by the read [pixel value](#) operator.

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.

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.

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](#).

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:

← *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 *p* belongs to the image domain.

Parameters:

← *p* A [point](#) site.

Returns:

True if *p* belongs to the image domain.

Invariant:

has(*p*) is true => has(*p*) is also true.

10.113.3.7 `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.

10.113.3.8 `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:

← *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:

$\leftarrow p$ 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> lvalue mln::doc::Fastest_Image< E >::operator[] (unsigned o)`

Read-write access to the image [value](#) at offset o .

Parameters:

$\leftarrow o$ An offset.

Precondition:

$o < \text{nelements}()$

Returns:

The [value](#) at o (assignable).

10.113.3.14 `]`

`template<typename E> rvalue mln::doc::Fastest_Image< E >::operator[] (unsigned o) const`

Read-only access to the image [value](#) at offset o .

Parameters:

$\leftarrow o$ An offset.

Precondition:

$o < \text{nelements}()$

Returns:

The [value](#) at o (not 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:

← `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:

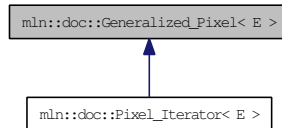
A reference to the [value set](#).

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](#)

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).

10.114.2.2 `template<typename E> typedef void mln::doc::Generalized_Pixel< E >::rvalue`

Read-only [value](#) associated type.

10.114.2.3 `template<typename E> typedef void mln::doc::Generalized_Pixel< E >::value`

[Value](#) associated type.

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](#).

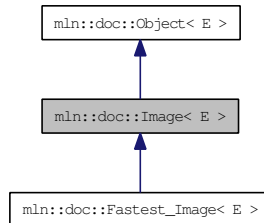
Reimplemented in [mln::doc::Pixel_Iterator< E >](#).

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 [p](#) belongs to the image domain.
- bool [has](#) (const [psite](#) &p) const
Test if the image owns the [point](#) site [p](#).
- bool [is_valid](#) () const
Test if the image have been initialized.
- 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](#).
- 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](#)

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](#).

10.115.2.2 `template<typename E> typedef void mln::doc::Image< E >::coord`

Coordinate associated type.

10.115.2.3 `template<typename E> typedef void mln::doc::Image< E >::dpoint`

Dpsite associated type.

Invariant:

This type has to derive from [mln::Dpoint](#).

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](#).

10.115.2.5 `template<typename E> typedef void mln::doc::Image< E >::lvalue`

Type returned by the read-write [pixel value](#) operator.

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](#).

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](#).

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](#).

10.115.2.9 `template<typename E> typedef void mln::doc::Image< E >::rvalue`

Type returned by the read [pixel value](#) operator.

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.

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.

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](#).

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 p belongs to the image domain.

Parameters:

$\leftarrow p$ A [point](#) site.

Returns:

True if p belongs to the image domain.

Invariant:

$\text{has}(p)$ is true \Rightarrow $\text{has}(p)$ is also true.

10.115.3.4 `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.

10.115.3.5 `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> lvalue mln::doc::Image< E >::operator() (const psite & p)`

Read-write access to the image [value](#) located at p .

Parameters:

$\leftarrow p$ A [point](#) site.

Precondition:

The image has to own the site p .

Returns:

The [value](#) at p (assignable).

10.115.3.8 `template<typename E> rvalue mln::doc::Image< E >::operator() (const psite & p) const`

Read-only access to the image [value](#) located at `p`.

Parameters:

← `p` A [point](#) site.

Precondition:

The image has to own the site `p`.

Returns:

The [value](#) at `p` (not 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:

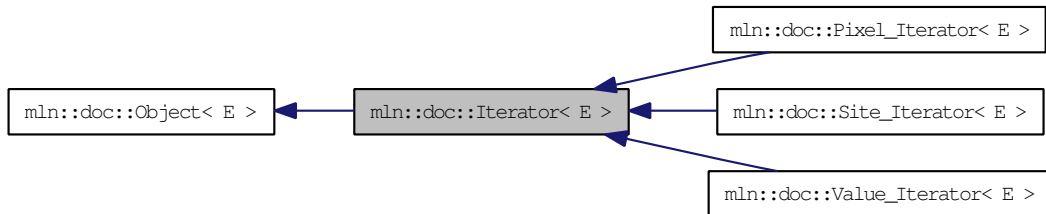
A reference to the [value set](#).

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](#)

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.

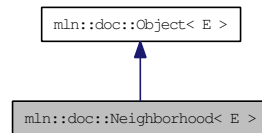
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

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](#)

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.

10.117.2.2 `template<typename E> typedef void mln::doc::Neighborhood< E >::dpoint`

Dpsite associated type.

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.

10.117.2.4 `template<typename E> typedef void mln::doc::Neighborhood< E >::niter`

[Site_Iterator](#) type associated to this neighborhood to browse neighbors.

10.117.2.5 `template<typename E> typedef void mln::doc::Neighborhood< E >::point`

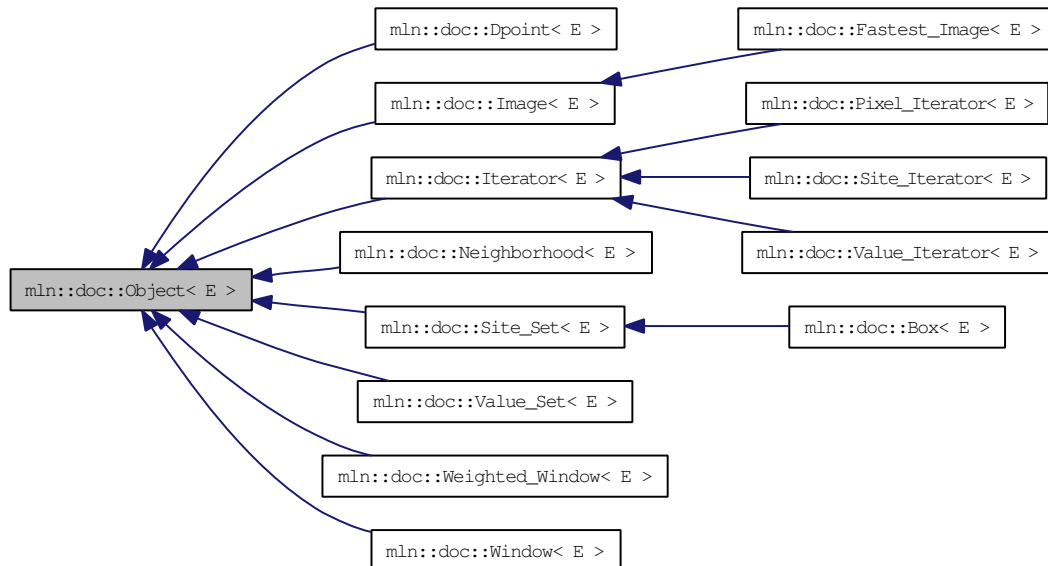
[Site](#) associated type.

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:

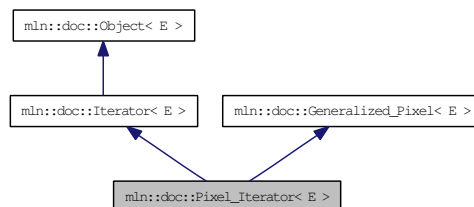
[mln::Object](#)

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](#)

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).

10.119.2.2 template<typename E> typedef void mln::doc::Pixel_Iterator< E >::lvalue

Type returned by the read-write dereference operator.

10.119.2.3 template<typename E> typedef void mln::doc::Generalized_Pixel< E >::rvalue
[inherited]

Read-only [value](#) associated type.

10.119.2.4 template<typename E> typedef void mln::doc::Generalized_Pixel< E >::value
[inherited]

[Value](#) associated type.

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](#)

10.120.2 Member Typedef Documentation

10.120.2.1 `template<typename E> typedef void mln::doc::Point_Site< E >::coord`

Coordinate associated type.

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](#).

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](#).

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](#).

10.120.3 Member Enumeration Documentation**10.120.3.1** `template<typename E> anonymous enum`**Enumerator:**

dim Dimension of the space.

Invariant:

$dim > 0$

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:

$\leftarrow i$ 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:

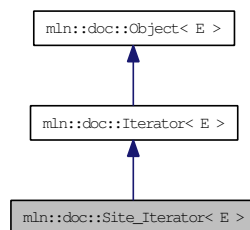
A [point](#) constant reference.

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](#)

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](#).

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.

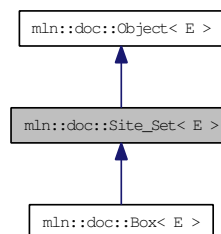
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

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](#)

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.

10.122.2.2 `template<typename E> typedef void mln::doc::Site_Set< E >::fwd_piter`

Forward [Site_Iterator](#) associated type.

10.122.2.3 `template<typename E> typedef void mln::doc::Site_Set< E >::psite`

PSite associated type.

10.122.2.4 `template<typename E> typedef void mln::doc::Site_Set< E >::site`

[Site](#) associated type.

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:

← *p* A psite.

Returns:

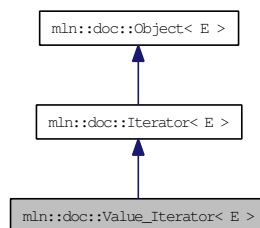
True if *p* is an element of the site [set](#).

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](#)

10.123.2 Member Typedef Documentation

10.123.2.1 `template<typename E> typedef void mln::doc::Value_Iterator< E >::value`

[Value](#) associated type.

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.

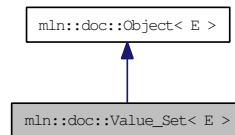
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

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](#)

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.

10.124.2.2 `template<typename E> typedef void mln::doc::Value_Set< E >::fwd_viter`

Forward [Value_Iterator](#) associated type.

10.124.2.3 `template<typename E> typedef void mln::doc::Value_Set< E >::value`

[Value](#) associated type.

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:

$\leftarrow v$ 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`

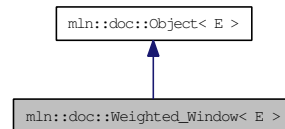
Give the i -th [value](#) of this [set](#).

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](#)

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.

10.125.2.2 `template<typename E> typedef void mln::doc::Weighted_Window< E >::dpoint`

Dpsite associated type.

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.

10.125.2.4 `template<typename E> typedef void mln::doc::Weighted_Window< E >::point`

[Site](#) associated type.

10.125.2.5 `template<typename E> typedef void mln::doc::Weighted_Window< E >::weight`

Weight associated type.

10.125.2.6 `template<typename E> typedef void mln::doc::Weighted_Window< E >::window`

[Window](#) associated type.

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`

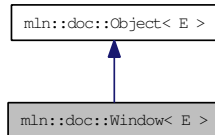
Give the corresponding [window](#).

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](#)

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.

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.

10.126.2.3 `template<typename E> typedef void mln::doc::Window< E >::qiter`

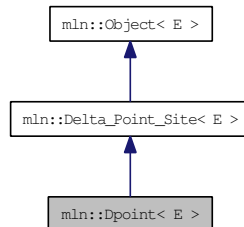
[Site_Iterator](#) type associated to this [window](#) to browse its points.

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.

10.127.2 Member Function Documentation

10.127.2.1 **template<typename E> const E & mln::Dpoint< E >::to_dpoint () const** [inline]

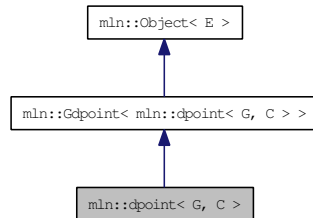
It is a [Dpoint](#) so it returns itself.

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

- `template<typename F>`
`dpoint` (const `Function_v2v< F >` &f)
Constructor; coordinates are set by function f.
- `template<typename C2>`
`dpoint` (const `algebra::vec< dim, C2 >` &v)
Constructor from an algebra vector.
- `dpoint` ()
Constructor without argument.
- `template<typename Q>`
`operator mln::algebra::vec< dpoint< G, C >::dim, Q > ()` const

Conversion towards a algebra::vec.

- C & [operator\[\]](#) (unsigned i)
Read-write access to the i-th coordinate value.
- C [operator\[\]](#) (unsigned i) const
Read-only 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](#) (const [literal::zero_t](#) &)
Constructors/assignments with literals.
- [dpoint](#) (C ind)

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.

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.

10.128.2.2 template<typename G, typename C> typedef G mln::dpoint< G, C >::grid

Grid associated type.

10.128.2.3 template<typename G, typename C> typedef point<G,C> mln::dpoint< G, C >::psite

Psite associated type.

10.128.2.4 template<typename G, typename C> typedef point<G,C> mln::dpoint< G, C >::site

[Site](#) associated type.

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.

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`

10.128.4 Constructor & Destructor Documentation

10.128.4.1 `template<typename G, typename C> mln::dpoint< G, C >::dpoint () [inline]`

Constructor without argument.

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.

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.

10.128.4.4 `template<typename G, typename C> mln::dpoint< G, C >::dpoint (const literal::zero_t &) [inline]`

Constructors/assignments with literals.

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`.

References `mln::dpoint< G, C >::dim`.

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`.

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:

← `i` The coordinate index.

Precondition:

`i < dim`

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:

← `i` The coordinate index.

Precondition:

`i < dim`

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`.

References `mln::dpoint< G, C >::dim`.

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.

References mln::dpoint< G, C >::dim.

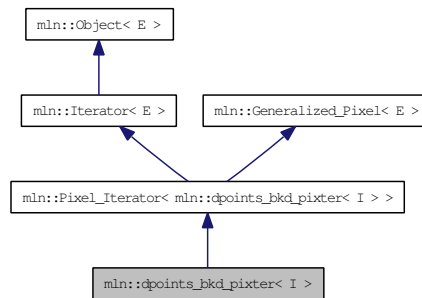
Referenced by mln::dpoint< G, C >::operator mln::algebra::vec< dpoint< G, C >::dim, Q >().

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 invalidate ()`
Invalidate the iterator.
- `bool is_valid () const`
Test the iterator validity.
- `void start ()`
Manipulation.
- `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.

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:

- ← *image* The image to iterate over.
- ← *dps* An object (neighborhood or [window](#)) that can provide a [set](#) of delta-points.
- ← *p_ref* Center (resp. reference) [point](#) of the neighborhood (resp. [window](#)).

**10.129.2.2 template<typename I> template<typename Dps, typename Pref>
mln::dpoints_bkd_pixter< I >::dpoints_bkd_pixter (const Generalized_Pixel< Pref >
& *p_xl_ref*, const Dps & *dps*) [inline]**

Constructor (using a generalized [pixel](#)).

Parameters:

- ← *p_xl_ref* Center (generalized) [pixel](#) to iterate around.
- ← *dps* An object (neighborhood or [window](#)) that can provide a [set](#) of delta-points.

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.

10.129.3.2 template<typename I> void mln::dpoints_bkd_pixter< I >::invalidate () [inline]

Invalidate the iterator.

**10.129.3.3 template<typename I> bool mln::dpoints_bkd_pixter< I >::is_valid () const
[inline]**

Test the iterator validity.

Referenced by `mln::dpoints_bkd_pixter< I >::update()`.

10.129.3.4 `template<typename E> void mln::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.

10.129.3.5 `template<typename I> void mln::dpoints_bkd_pixter< I >::start ()` [inline]

Manipulation.

Start an iteration.

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.

References `mln::dpoints_bkd_pixter< I >::is_valid()`.

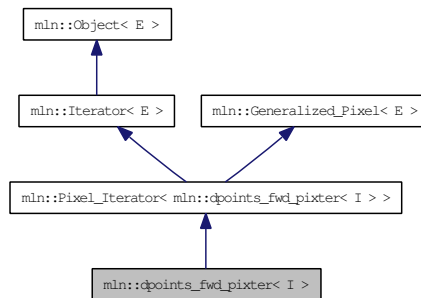
Referenced by `mln::dpoints_bkd_pixter< I >::start()`.

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 [invalidate](#) ()
Invalidate the iterator.
- bool [is_valid](#) () const
Test the iterator validity.
- void [start](#) ()
Manipulation.
- 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.

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:

- ← *image* The image to iterate over.
- ← *dps* An object (neighborhood or [window](#)) that can provide a [set](#) of delta-points.
- ← *p_ref* Center (resp. reference) [point](#) of the neighborhood (resp. [window](#)).

**10.130.2.2 template<typename I> template<typename Dps, typename Pref>
mln::dpoints_fwd_pixter< I >::dpoints_fwd_pixter (const Generalized_Pixel< Pref >
& *p_xl_ref*, const Dps & *dps*) [inline]**

Constructor (using a generalized [pixel](#)).

Parameters:

- ← *p_xl_ref* Center (generalized) [pixel](#) to iterate around.
- ← *dps* An object (neighborhood or [window](#)) that can provide a [set](#) of delta-points.

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.

10.130.3.2 template<typename I> void mln::dpoints_fwd_pixter< I >::invalidate () [inline]

Invalidate the iterator.

**10.130.3.3 template<typename I> bool mln::dpoints_fwd_pixter< I >::is_valid () const
[inline]**

Test the iterator validity.

Referenced by `mln::dpoints_fwd_pixter< I >::update()`.

10.130.3.4 `template<typename E> void mln::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.

10.130.3.5 `template<typename I> void mln::dpoints_fwd_pixter< I >::start ()` [inline]

Manipulation.

Start an iteration.

References mln::dpoints_fwd_pixter< I >::update().

10.130.3.6 `template<typename I> void mln::dpoints_fwd_pixter< I >::update ()` [inline]

Force this iterator to update its location to take into account that its center [point](#) may have moved.

References mln::dpoints_fwd_pixter< I >::is_valid().

Referenced by mln::dpoints_fwd_pixter< 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 mln::internal::site_relative_iterator_base< V, mln::dpsites_bkd_piter< V > >.

Public Member Functions

- [dpsites_bkd_piter \(\)](#)
Constructor without argument.
- [template<typename P> dpsites_bkd_piter \(const V &v, const P &c\)](#)
Constructor.
- [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.

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:

- ← **v** [Object](#) that can provide an array of delta-points.
- ← **c** Center [point](#) to iterate around.

10.131.2.2 [template<typename V> mln::dpsites_bkd_piter< V >::dpsites_bkd_piter \(\)](#) [inline]

Constructor without argument.

10.131.3 Member Function Documentation

10.131.3.1 `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.

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 mln::internal::site_relative_iterator_base< V, mln::dpsites_fwd_piter< V > >.

Public Member Functions

- [dpsites_fwd_piter](#) ()
Constructor without argument.
- `template<typename P>`
[dpsites_fwd_piter](#) (const V &v, const P &c)
Constructor.
- 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.

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:

- ← **v** **Object** that can provide an array of delta-points.
- ← **c** Center **point** to iterate around.

10.132.2.2 `template<typename V> mln::dpsites_fwd_piter< V >::dpsites_fwd_piter () [inline]`

Constructor without argument.

10.132.3 Member Function Documentation

10.132.3.1 `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.

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.

10.134 mln::edge_image< P, V, G > Class Template Reference

Image based on [graph](#) edges.

```
#include <edge_image.hh>
```

Inherits mln::pw::internal::image_base< mln::fun::i2v::array< V >, mln::p_edges< G, mln::internal::efsite_selector< P, G >::mln::fun::i2v::array >, mln::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.

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.

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.

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](#).

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.

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.

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.

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.

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.

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 [inline]`

[Value](#) accessors/operators overloads.

10.135 mln::extended< I > Struct Template Reference

Makes an image become restricted by a [point set](#).

```
#include <extended.hh>
```

Inherits mln::internal::image_domain_morpher< I, mln::box< I::site >, mln::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](#) (I &ima, const [box](#)< typename I::site > &b)

Constructor.

- [extended](#) ()

Constructor without argument.

10.135.1 Detailed Description

```
template<typename I> struct mln::extended< I >
```

Makes an image become restricted by a [point set](#).

10.135.2 Member Typedef Documentation

10.135.2.1 template<typename I> typedef tag::image_<I> mln::extended< I >::skeleton

Skeleton.

10.135.2.2 template<typename I> typedef I::value mln::extended< I >::value

[Value](#) type.

10.135.3 Constructor & Destructor Documentation

10.135.3.1 `template<typename I> mln::extended< I >::extended ()` `[inline]`

Constructor without argument.

10.135.3.2 `template<typename I> mln::extended< I >::extended (I & ima, const box< typename I::site > & b)` `[inline]`

Constructor.

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.

10.136 mln::extension_fun< I, F > Class Template Reference

Extends the domain of an image with a function.

```
#include <extension_fun.hh>
```

Inherits mln::internal::image_identity< I, I::domain_t, mln::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.

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.

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.

10.136.2.3 `template<typename I, typename F> typedef I ::value mln::extension_fun< I, F >::value`

Image value type.

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.

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`.

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.

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`.

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`.

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`;

10.137 mln::extension_ima< I, J > Class Template Reference

Extends the domain of an image with an image.

```
#include <extension_ima.hh>
```

Inherits mln::internal::image_identity< I, I::domain_t, mln::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_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`.

10.137.1 Detailed Description

```
template<typename I, typename J> class mln::extension_ima< I, J >
```

Extends the domain of an image with an image.

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.

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.

10.137.2.3 `template<typename I, typename J> typedef I ::value mln::extension_ima< I, J >::value`

[Image value](#) type.

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.

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`.

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).

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.

Referenced by `mln::extension_ima< I, J >::operator()()`.

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`.

References `mln::extension_ima< I, J >::has()`.

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`;

References `mln::extension_ima< I, J >::has()`.

10.138 mln::extension_val< I > Class Template Reference

Extends the domain of an image with a [value](#).

```
#include <extension_val.hh>
```

Inherits mln::internal::image_identity< I, I::domain_t, mln::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::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`.

10.138.1 Detailed Description

```
template<typename I> class mln::extension_val< I >
```

Extends the domain of an image with a [value](#).

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.

10.138.2.2 `template<typename I> typedef extension_val< tag::image_<I> > mln::extension_val< I >::skeleton`

Skeleton.

10.138.2.3 `template<typename I> typedef I::value mln::extension_val< I >::value`

[Image value](#) type.

10.138.3 Constructor & Destructor Documentation

10.138.3.1 `template<typename I> mln::extension_val< I >::extension_val () [inline]`

Constructor without argument.

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`.

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.

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.

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.

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`.

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`;

10.139 mln::faces_psite< N, D, P > Class Template Reference

[Point](#) site associated to a [mln::p_faces](#).

```
#include <faces_psite.hh>
```

Inherits mln::internal::pseudo_site_base_< const P &, mln::faces_psite< N, D, P > >.

Public Member Functions

- void [change_target](#) (const [target](#) &new_target)
Set the target site_set.
- const [target](#) & [site_set](#) () const
Site set manipulators.
- [topo::n_face](#)< N, D > [face](#) () const
Face handle manipulators.
- unsigned [face_id](#) () const
Return the id of the face of this psite.
- unsigned [n](#) () const
Return the dimension of the face of this psite.
- [faces_psite](#) (const [p_faces](#)< N, D, P > &pf, const [topo::n_face](#)< N, D > &face)
- [faces_psite](#) ()
Construction and assignment.
- void [invalidate](#) ()
Invalidate this psite.
- bool [is_valid](#) () const
Psite manipulators.

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.

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.

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()`.

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`.

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.

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.

10.139.3.4 `template<unsigned N, unsigned D, typename P> void mln::faces_psite< N, D, P >::invalidate () [inline]`

Invalidate this psite.

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?

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.

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.

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 mln::internal::image_primary< T, S, mln::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.
- T & [operator\(\)](#) (const typename S::psite &p)
Read-write access to the image [value](#) located at [point](#) p.
- const T & [operator\(\)](#) (const typename S::psite &p) const
Read-only 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](#).

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.

10.140.2.2 `template<typename T, typename S> typedef const T& mln::flat_image< T, S >::rvalue`

Return type of read-only access.

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.

10.140.2.4 `template<typename T, typename S> typedef T mln::flat_image< T, S >::value`

[Value](#) associated type.

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.

10.140.3.2 `template<typename T, typename S> mln::flat_image< T, S >::flat_image (const T & val, const S & pset)` [inline]

Constructor.

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.

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.

10.140.4.3 `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*.

10.140.4.4 `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`.

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< mln::fun::from_accu< A >, A * >`.

10.141.1 Detailed Description

```
template<typename A> struct mln::fun::from_accu< A >
```

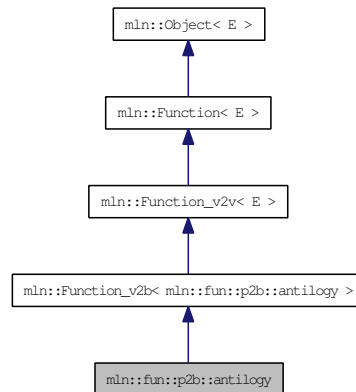
Wrap an accumulator into a function.

10.142 mln::fun::p2b::antilogy Struct Reference

A [p2b](#) function always returning `false`.

```
#include <antilogy.hh>
```

Inheritance diagram for `mln::fun::p2b::antilogy`:



10.142.1 Detailed Description

A [p2b](#) function always returning `false`.

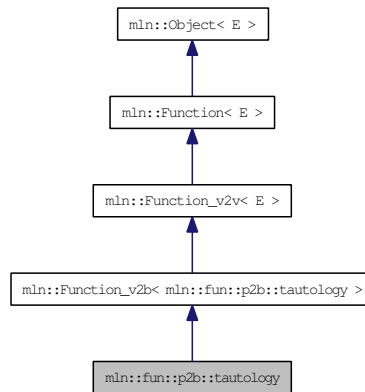
A simpler name would be `'false'`, but this is not a valid C++ identifier, as `false` is a keyword of the language.

10.143 mln::fun::p2b::tautology Struct Reference

A `p2b` function always returning `true`.

```
#include <tautology.hh>
```

Inheritance diagram for `mln::fun::p2b::tautology`:



10.143.1 Detailed Description

A `p2b` function always returning `true`.

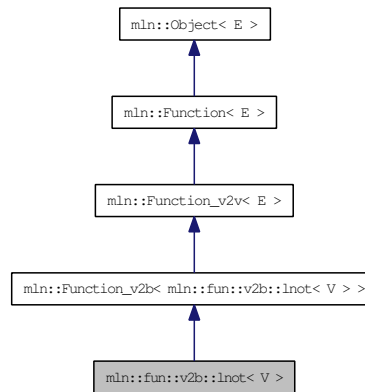
A simpler name would be ‘`true`’, but this is not a valid C++ identifier, as `true` is a keyword of the language.

10.144 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 >:



10.144.1 Detailed Description

```
template<typename V> struct mln::fun::v2b::lnot< V >
```

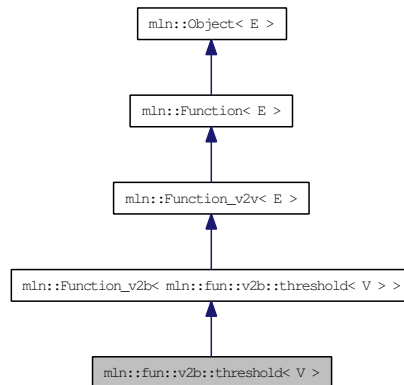
Functor computing logical-not on a [value](#).

10.145 mln::fun::v2b::threshold< V > Struct Template Reference

Threshold function.

```
#include <threshold.hh>
```

Inheritance diagram for mln::fun::v2b::threshold< V >:



10.145.1 Detailed Description

```
template<typename V> struct mln::fun::v2b::threshold< V >
```

Threshold function.

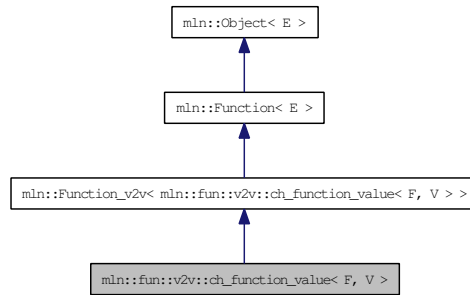
`f(v) = (v >= threshold).`

10.146 `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 >`:



10.146.1 Detailed Description

```
template<typename F, typename V> class mln::fun::v2v::ch_function_value< F, V >
```

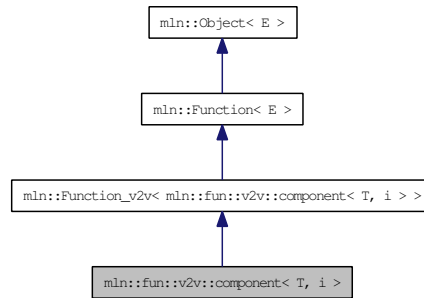
Wrap a function `v2v` and `convert` its result to another type.

10.147 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 >:



10.147.1 Detailed Description

```
template<typename T, unsigned i> struct mln::fun::v2v::component< T, i >
```

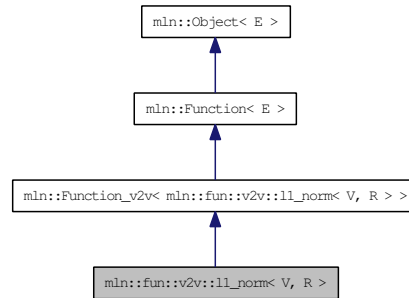
Functor that accesses the *i*-th [component](#) of a [value](#).

10.148 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 >:



10.148.1 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:

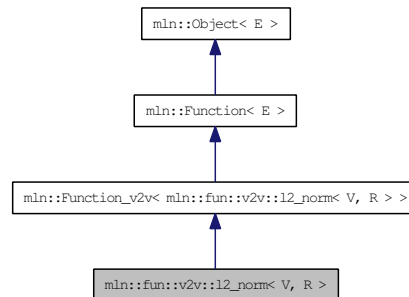
[mln::norm::l1](#).

10.149 `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 >`:



10.149.1 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:

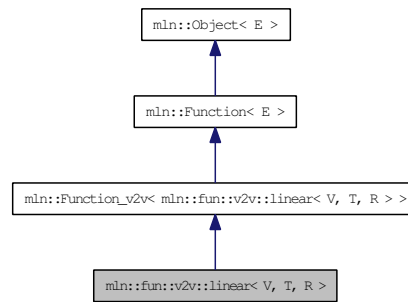
`mln::norm::l2`.

10.150 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 >:



10.150.1 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.

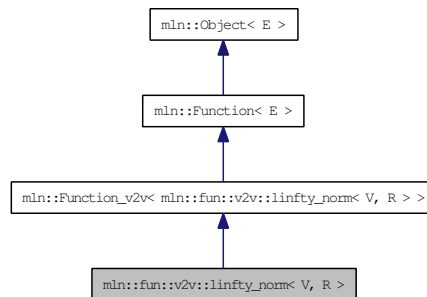
By default, T is V and R is T.

10.151 `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 >`:



10.151.1 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:

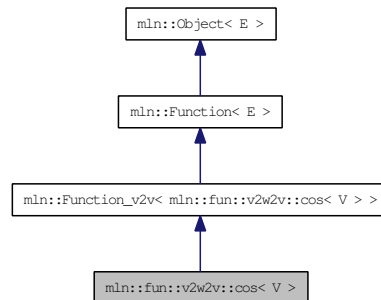
[mln::norm::linfty](#).

10.152 mln::fun::v2w2v::cos< V > Struct Template Reference

Cosinus bijective functor.

```
#include <cos.hh>
```

Inheritance diagram for mln::fun::v2w2v::cos< V >:



10.152.1 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:

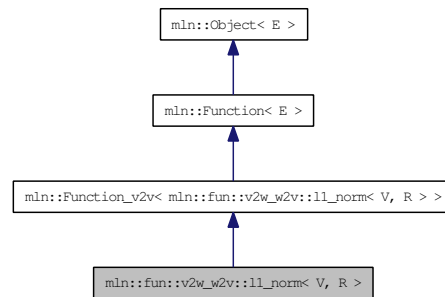
`mln::math::cos`.

10.153 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 >:



10.153.1 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:

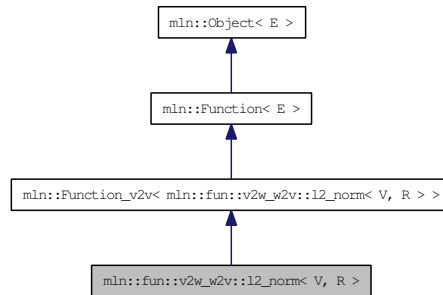
[mln::norm::l1](#).

10.154 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 >:



10.154.1 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:

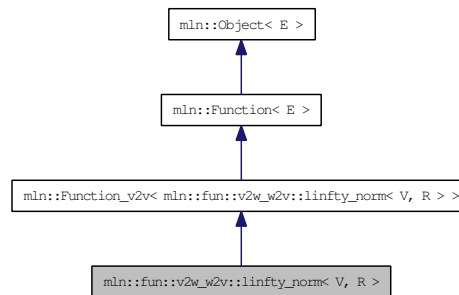
`mln::norm::l2`.

10.155 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 >:



10.155.1 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:

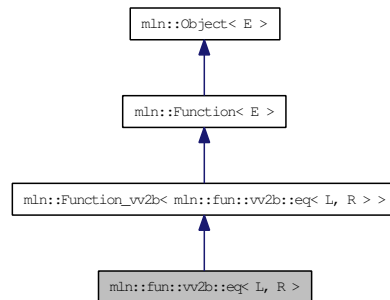
[mln::norm::linfty](#).

10.156 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 >:



10.156.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::eq< L, R >
```

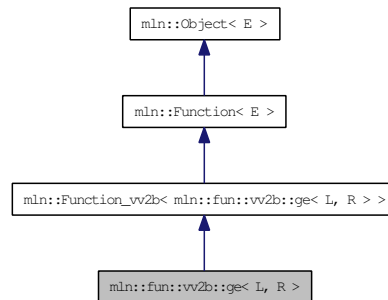
Functor computing equal between two values.

10.157 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 >:



10.157.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::ge< L, R >
```

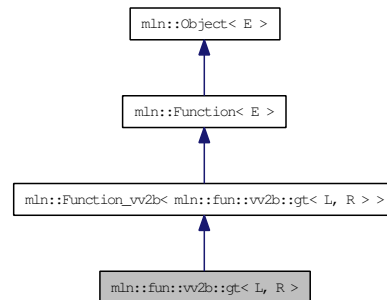
Functor computing "greater or equal than" between two values.

10.158 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 >:



10.158.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::gt< L, R >
```

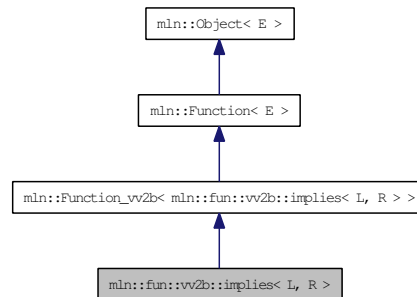
Functor computing "greater than" between two values.

10.159 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 >:



10.159.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::implies< L, R >
```

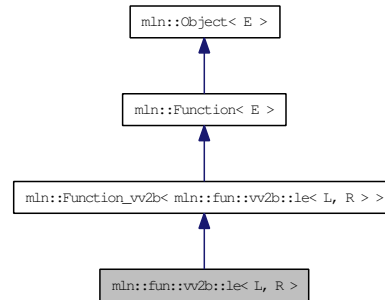
Functor computing logical-implies between two values.

10.160 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 >:



10.160.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::le< L, R >
```

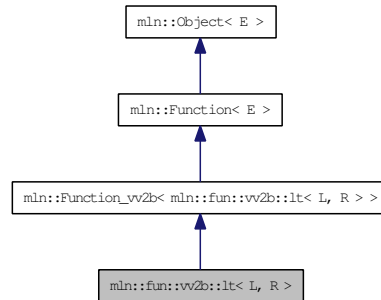
Functor computing "lower or equal than" between two values.

10.161 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 >:



10.161.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2b::lt< L, R >
```

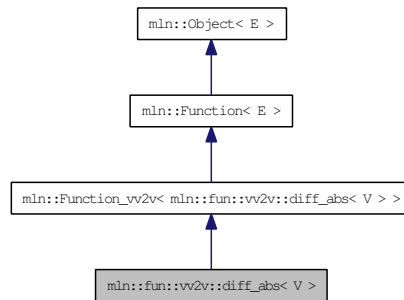
Functor computing "lower than" between two values.

10.162 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 >:



10.162.1 Detailed Description

```
template<typename V> struct mln::fun::vv2v::diff_abs< V >
```

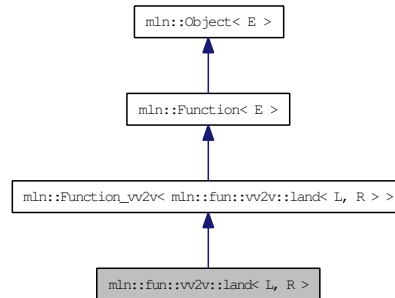
A functor computing the diff_absimum of two values.

10.163 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 >:



10.163.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::land< L, R >
```

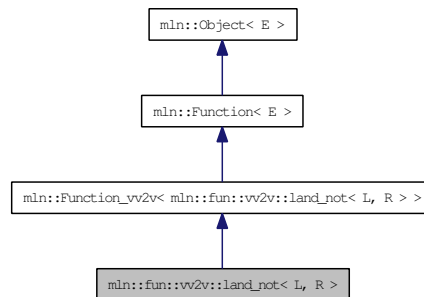
Functor computing logical-and between two values.

10.164 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 >:



10.164.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::land_not< L, R >
```

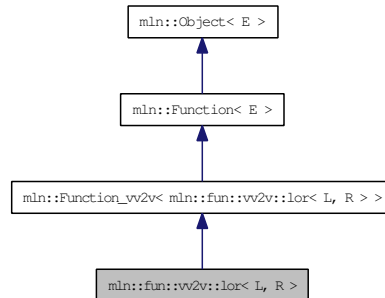
Functor computing [logical](#) and-not between two values.

10.165 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 >:



10.165.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::lor< L, R >
```

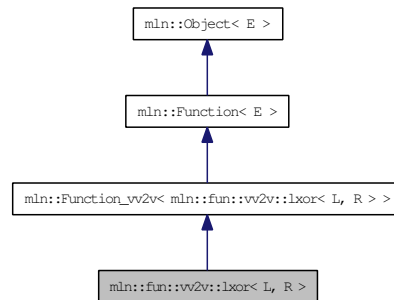
Functor computing logical-or between two values.

10.166 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 >:



10.166.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::lxor< L, R >
```

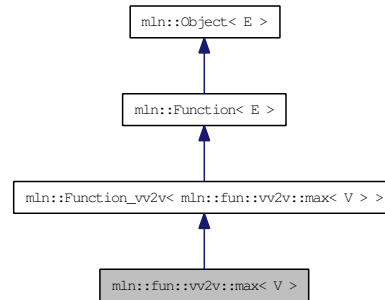
Functor computing logical-xor between two values.

10.167 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 >:



10.167.1 Detailed Description

```
template<typename V> struct mln::fun::vv2v::max< V >
```

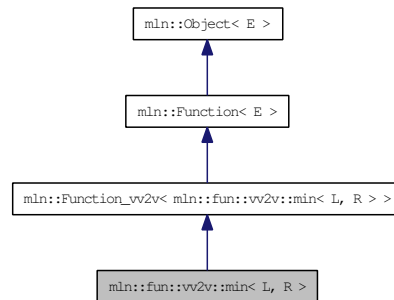
A functor computing the maximum of two values.

10.168 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 >:



10.168.1 Detailed Description

```
template<typename L, typename R = L> struct mln::fun::vv2v::min< L, R >
```

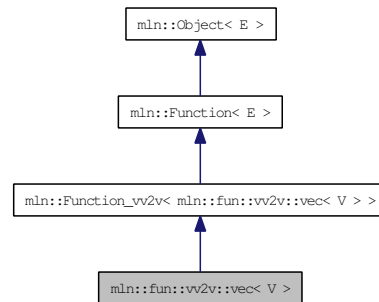
A functor computing the minimum of two values.

10.169 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 >:



10.169.1 Detailed Description

```
template<typename V> struct mln::fun::vv2v::vec< V >
```

A functor computing the vecimum of two values.

10.170 mln::fun::x2p::closest_point< P > Struct Template Reference

FIXME: doxygen + concept checking.

```
#include <closest_point.hh>
```

10.170.1 Detailed Description

```
template<typename P> struct mln::fun::x2p::closest_point< P >
```

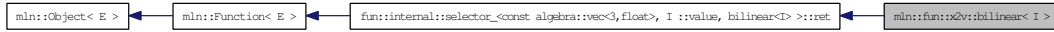
FIXME: doxygen + concept checking.

10.171 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.

10.171.1 Detailed Description

```
template<typename I> struct mln::fun::x2v::bilinear< I >
```

Represent a [bilinear](#) interolation of values from an underlying image.

10.171.2 Member Function Documentation

10.171.2.1 `template<typename I> template<typename T> I::value mln::fun::x2v::bilinear< I >::operator() (const algebra::vec< 3, T > & v) const` `[inline]`

Bilinear filtering on 3d images. Work on slices.

10.171.2.2 `template<typename I> template<typename T> I::value mln::fun::x2v::bilinear< I >::operator() (const algebra::vec< 2, T > & v) const` `[inline]`

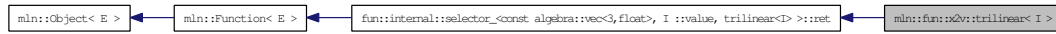
Bilinear filtering on 2d images.

10.172 mln::fun::x2v::trilinear< I > Struct Template Reference

Represent a [trilinear](#) interpolation of values from an underlying image.

```
#include <trilinear.hh>
```

Inheritance diagram for mln::fun::x2v::trilinear< I >:



10.172.1 Detailed Description

```
template<typename I> struct mln::fun::x2v::trilinear< I >
```

Represent a [trilinear](#) interpolation of values from an underlying image.

10.173 mln::fun::x2x::composed< T2, T1 > Struct Template Reference

Represent a composition of two transformations.

```
#include <composed.hh>
```

Public Member Functions

- [composed](#) (const T2 &f, const T1 &g)
Constructor with the two transformation to be [composed](#).
- [composed](#) ()
Constructor without argument.

10.173.1 Detailed Description

```
template<typename T2, typename T1> struct mln::fun::x2x::composed< T2, T1 >
```

Represent a composition of two transformations.

10.173.2 Constructor & Destructor Documentation

10.173.2.1 `template<typename T2, typename T1> mln::fun::x2x::composed< T2, T1 >::composed () [inline]`

Constructor without argument.

10.173.2.2 `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](#).

10.174 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.

10.174.1 Detailed Description

```
template<typename I> struct mln::fun::x2x::linear< I >
```

Represent a [linear](#) interolation of values from an underlying image.

10.174.2 Constructor & Destructor Documentation

10.174.2.1 template<typename I> mln::fun::x2x::linear< I >::linear (const I & *ima*) [inline]

Constructor with the underlying image.

10.174.3 Member Function Documentation

10.174.3.1 template<typename I> template<typename C> I::value mln::fun::x2x::linear< I >::operator() (const algebra::vec< 1, C > & v) const [inline]

Return the [interpolated value](#) in the underlying image at the given 'point' v.

10.174.4 Member Data Documentation

10.174.4.1 `template<typename I> const I& mln::fun::x2x::linear< I >::ima`

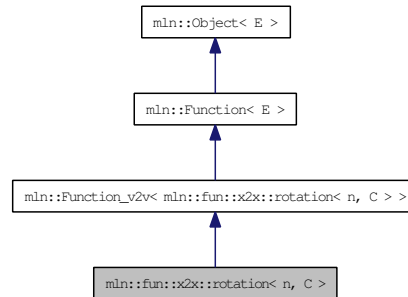
Underlying image.

10.175 `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 `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 (const algebra::h_mat<n, C> &m)`
Constructor with `h_mat`.
- `rotation (const algebra::quat &q)`
Constructor with quaternion.
- `rotation (C alpha, const algebra::vec<n, C> &axis)`
Constructor with radian `alpha` and a facultative direction ([rotation axis](#)).
- `rotation ()`
Constructor without argument.
- `void set_alpha (C alpha)`
Set a new grade `alpha`.
- `void set_axis (const algebra::vec<n, C> &axis)`
Set a new [rotation axis](#).

10.175.1 Detailed Description

`template<unsigned n, typename C> struct mln::fun::x2x::rotation< n, C >`

Represent a [rotation](#) function.

10.175.2 Member Typedef Documentation

10.175.2.1 `template<unsigned n, typename C> typedef rotation<n,C> mln::fun::x2x::rotation<n, C >::invert`

Type of the inverse function.

10.175.3 Constructor & Destructor Documentation

10.175.3.1 `template<unsigned n, typename C> mln::fun::x2x::rotation< n, C >::rotation ()`
[inline]

Constructor without argument.

10.175.3.2 `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).

10.175.3.3 `template<unsigned n, typename C> mln::fun::x2x::rotation< n, C >::rotation (const`
`algebra::quat & q) [inline]`

Constructor with quaternion.

References `mln::make::h_mat()`.

10.175.3.4 `template<unsigned n, typename C> mln::fun::x2x::rotation< n, C >::rotation (const`
`algebra::h_mat< n, C > & m) [inline]`

Constructor with `h_mat`.

10.175.4 Member Function Documentation

10.175.4.1 `template<unsigned n, typename C> rotation< n, C > mln::fun::x2x::rotation< n, C`
`>::inv () const [inline]`

Return the invere function.

10.175.4.2 `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.

10.175.4.3 `template<unsigned n, typename C> void mln::fun::x2x::rotation<n, C>::set_alpha(C alpha) [inline]`

Set a new grade alpha.

10.175.4.4 `template<unsigned n, typename C> void mln::fun::x2x::rotation<n, C>::set_axis(const algebra::vec<n, C> & axis) [inline]`

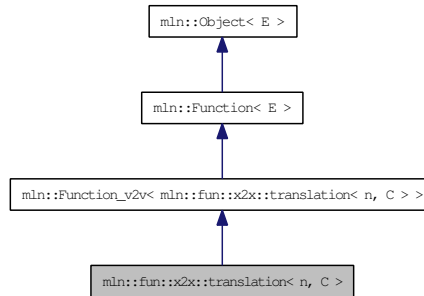
Set a new [rotation](#) axis.

10.176 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 [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.

10.176.1 Detailed Description

```
template<unsigned n, typename C> struct mln::fun::x2x::translation< n, C >
```

Translation function-object.

10.176.2 Member Typedef Documentation

```
10.176.2.1 template<unsigned n, typename C> typedef translation<n,C>
mln::fun::x2x::translation< n, C >::invert
```

Type of the inverse function.

10.176.3 Constructor & Destructor Documentation

```
10.176.3.1 template<unsigned n, typename C> mln::fun::x2x::translation< n, C >::translation ()
[inline]
```

Constructor without argument.

```
10.176.3.2 template<unsigned n, typename C> mln::fun::x2x::translation< n, C >::translation
(const algebra::vec< n, C > & t) [inline]
```

Constructor with the [translation](#) vector.

10.176.4 Member Function Documentation

```
10.176.4.1 template<unsigned n, typename C> translation< n, C > mln::fun::x2x::translation<
n, C >::inv () const [inline]
```

Return the inverse function.

```
10.176.4.2 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.

```
10.176.4.3 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.

```
10.176.4.4 template<unsigned n, typename C> const algebra::vec< n, C > &
mln::fun::x2x::translation< n, C >::t () const [inline]
```

Return the [translation](#) vector.

10.177 mln::fun_image< F, I > Struct Template Reference

[Image](#) read through a function.

```
#include <fun_image.hh>
```

Inherits mln::internal::image_value_morpher< I, F::result, mln::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](#) (const [Image](#)< I > &ima)
Constructor.
- [fun_image](#) (const [Function_v2v](#)< F > &f, const [Image](#)< I > &ima)
Constructor.
- [fun_image](#) ()
Constructor.
- F::result [operator](#)() (const typename I::psite &p)
Mutable access is for reading only.
- F::result [operator](#)() (const typename I::psite &p) const
Read-only access of [pixel value](#) at [point](#) site p.

10.177.1 Detailed Description

```
template<typename F, typename I> struct mln::fun_image< F, I >
```

[Image](#) read through a function.

10.177.2 Member Typedef Documentation

10.177.2.1 `template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::lvalue`

Return type of read-write access.

10.177.2.2 `template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::rvalue`

Return type of read-only access.

10.177.2.3 `template<typename F, typename I> typedef fun_image< tag::value_<typename F ::result>, tag::image_<I> > mln::fun_image< F, I >::skeleton`

Skeleton.

10.177.2.4 `template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::value`

[Value](#) associated type.

10.177.3 Constructor & Destructor Documentation

10.177.3.1 `template<typename F, typename I> mln::fun_image< F, I >::fun_image ()`
[inline]

Constructor.

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

Constructor.

10.177.3.3 `template<typename F, typename I> mln::fun_image< F, I >::fun_image (const Image< I > &ima)` [inline]

Constructor.

10.177.4 Member Function Documentation

10.177.4.1 `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.

10.177.4.2 `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.

10.178 mln::Function< E > Struct Template Reference

Base class for implementation of function-objects.

```
#include <function.hh>
```

Inherits [mln::Object< E >](#).

Inherited by [mln::Function_v2v< function< meta::blue< mln::value::rgb::mln::value::rgb< n > > >](#), [mln::Function_v2v< function< meta::green< mln::value::rgb::mln::value::rgb< n > > >](#), [mln::Function_v2v< function< meta::hue< mln::value::hsi::mln::value::hsi< H, S, I > > >](#), [mln::Function_v2v< function< meta::hue< mln::value::hsl::mln::value::hsl< H, S, L > > >](#), [mln::Function_v2v< function< meta::inty< mln::value::hsi::mln::value::hsi< H, S, I > > >](#), [mln::Function_v2v< function< meta::lum< mln::value::hsl::mln::value::hsl< H, S, I > > >](#), [mln::Function_v2v< function< meta::red< mln::value::rgb::mln::value::rgb< n > > >](#), [mln::Function_v2v< function< meta::sat< mln::value::hsi::mln::value::hsi< H, S, I > > >](#), [mln::Function_v2v< function< meta::sat< mln::value::hsl::mln::value::hsl< H, S, L > > >](#), [mln::Function_v2v< E >](#), [mln::Function_vv2b< E >](#), and [mln::Function_vv2v< E >](#).

Protected Member Functions

- [Function \(\)](#)

An operator() has to be provided.

10.178.1 Detailed Description

```
template<typename E> struct mln::Function< E >
```

Base class for implementation of function-objects.

The parameter *E* is the exact type.

10.178.2 Constructor & Destructor Documentation

10.178.2.1 `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.

10.179 mln::Function< void > Struct Template Reference

[Function](#) category flag type.

```
#include <function.hh>
```

10.179.1 Detailed Description

template<> struct mln::Function< void >

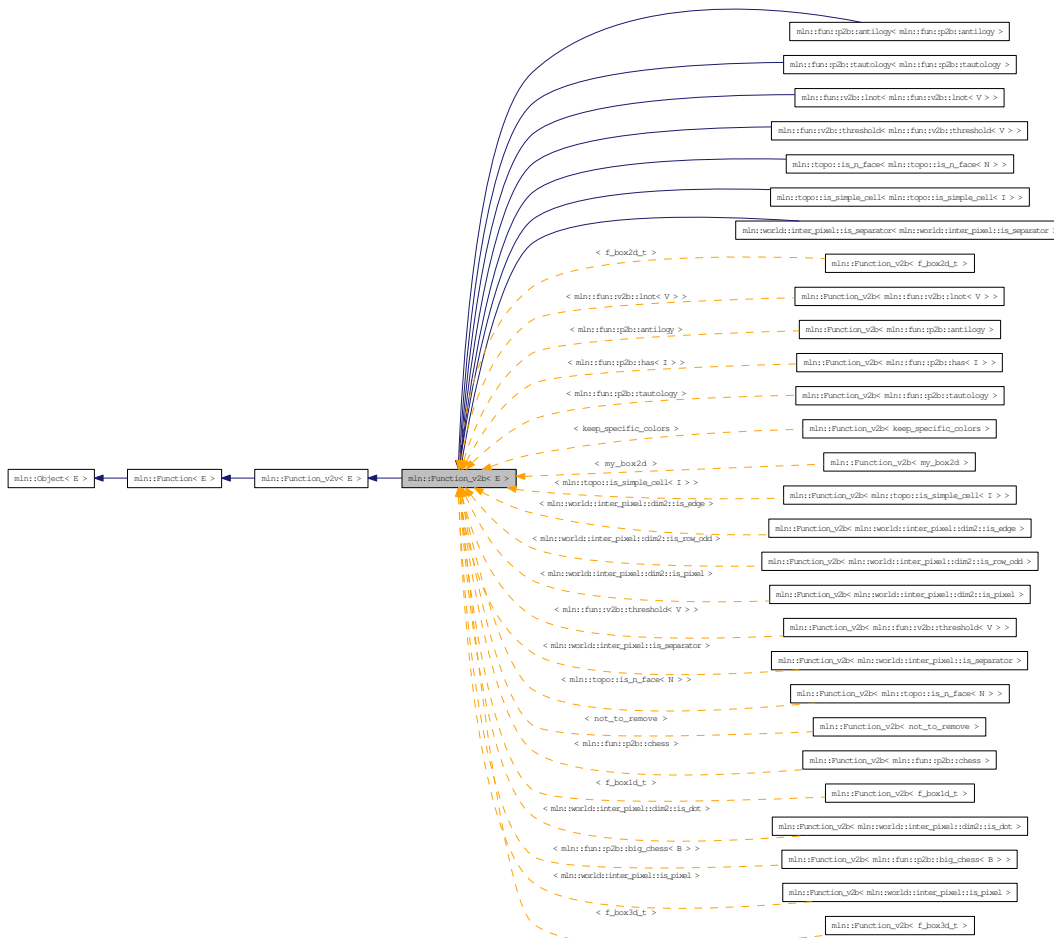
[Function](#) category flag type.

10.180 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 >:



10.180.1 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.

10.181 `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::edge_to_color< I, V >](#), [mln::fun::C< R\(*\)\(&A\) >](#), [mln::fun::cast_p2v_expr_-< V, F >](#), [mln::fun::i2v::all_to< T >](#), [mln::fun::i2v::value_at_index< T >](#), [mln::fun::i2v::value_at_index< bool >](#), [mln::fun::p2p::fold< P, dir_0, dir_1, dir_2 >](#), [mln::fun::p2p::mirror< B >](#), [mln::fun::p2p::translation_t< P >](#), [mln::fun::p2v::iota](#), [mln::fun::spe::impl::binary_impl< false, Fun, T1, T2 >](#), [mln::fun::spe::impl::binary_impl< true, Fun, T1, T2 >](#), [mln::fun::spe::impl::unary_impl< false, false, Fun, T >](#), [mln::fun::spe::impl::unary_impl< true, false, Fun, T >](#), [mln::fun::stat::mahalanobis< V >](#), [mln::fun::v2i::index_of_value< T >](#), [mln::fun::v2i::index_of_value< bool >](#), [mln::fun::v2v::abs< V >](#), [mln::fun::v2v::cast< V >](#), [mln::fun::v2v::ch_function_value< F, V >](#), [mln::fun::v2v::component< T, i >](#), [mln::fun::v2v::convert< V >](#), [mln::fun::v2v::dec< T >](#), [mln::fun::v2v::enc< V >](#), [mln::fun::v2v::f_hsi_to_rgb< T_rgb >](#), [mln::fun::v2v::f_hsl_to_rgb< T_rgb >](#), [mln::fun::v2v::f_rgb_to_hsi< T_hsi >](#), [mln::fun::v2v::f_rgb_to_hsl< T_hsl >](#), [mln::fun::v2v::id< T >](#), [mln::fun::v2v::inc< T >](#), [mln::fun::v2v::l1_norm< V, R >](#), [mln::fun::v2v::l2_norm< V, R >](#), [mln::fun::v2v::linear< V, T, R >](#), [mln::fun::v2v::linear_sat< V, T, R >](#), [mln::fun::v2v::linfty_norm< V, R >](#), [mln::fun::v2v::projection< P, dir >](#), [mln::fun::v2v::saturate< V >](#), [mln::fun::v2v::wrap< L >](#), [mln::fun::v2w2v::cos< V >](#), [mln::fun::v2w_w2v::l1_norm< V, R >](#), [mln::fun::v2w_w2v::l2_norm< V, R >](#), [mln::fun::v2w_w2v::linfty_norm< V, R >](#), [mln::fun::x2v::bilinear< I >](#), [mln::fun::x2v::l1_norm< V >](#), [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::nneighbor< I >](#), [mln::fun::x2x::rotation< n, C >](#), [mln::fun::x2x::translation< n, C >](#), [mln::function< meta::blue< value::rgb< n > >>](#), [mln::function< meta::green< value::rgb< n > >>](#), [mln::function< meta::hue< value::hsi< H, S, I > >>](#), [mln::function< meta::hue< value::hsl< H, S, L > >>](#), [mln::function< meta::inty< value::hsi< H, S, I > >>](#), [mln::function< meta::lum< value::hsl< H, S, I > >>](#), [mln::function< meta::red< value::rgb< n > >>](#), [mln::function< meta::sat< value::hsi< H, S, I > >>](#), [mln::function< meta::sat< value::hsl< H, S, L > >>](#), [mln::Function_v2b< E >](#)`[virtual]`, [mln::histo::point_from_value< T >](#), [mln::math::round< R >](#), [mln::math::round_sat< R >](#), [mln::my_ext](#), [mln::pw::var_< V >](#), [mln::ref_data](#), [mln::saturate_rgb8](#), [mln::to8bits](#), [mln::tofloat01](#), [mln::util::internal::id2element< G, Elt >](#), [my::sqrt](#), [test< T >](#), [to8bits](#), [to8bits](#), [to8bits](#), [to8bits](#), [to8bits](#), and [viota_t< S >](#).

10.181.1 Detailed Description

```
template<typename E> struct mln::Function_v2v< E >
```

Base class for implementation of function-objects from [value](#) to [value](#).

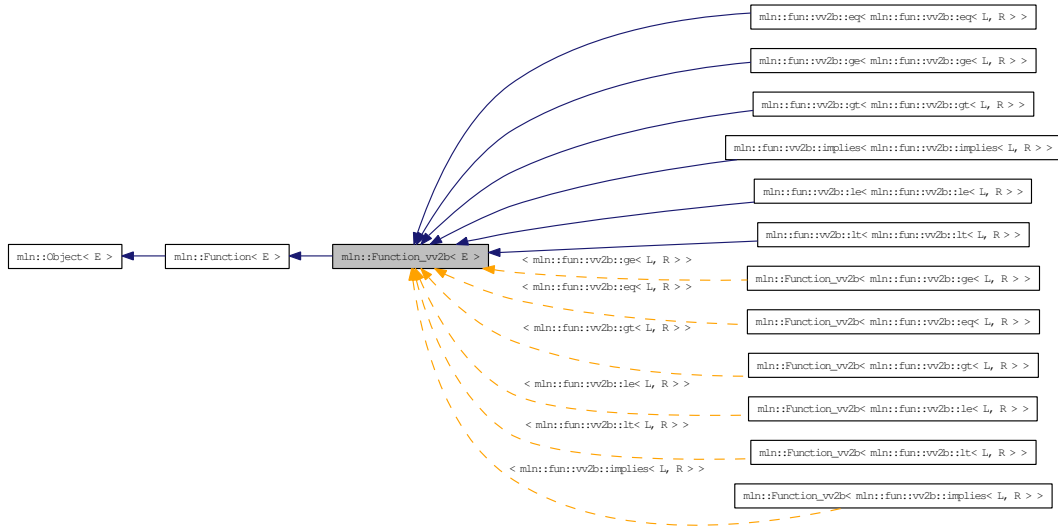
The parameter *E* is the exact type.

10.182 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 >:



10.182.1 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.

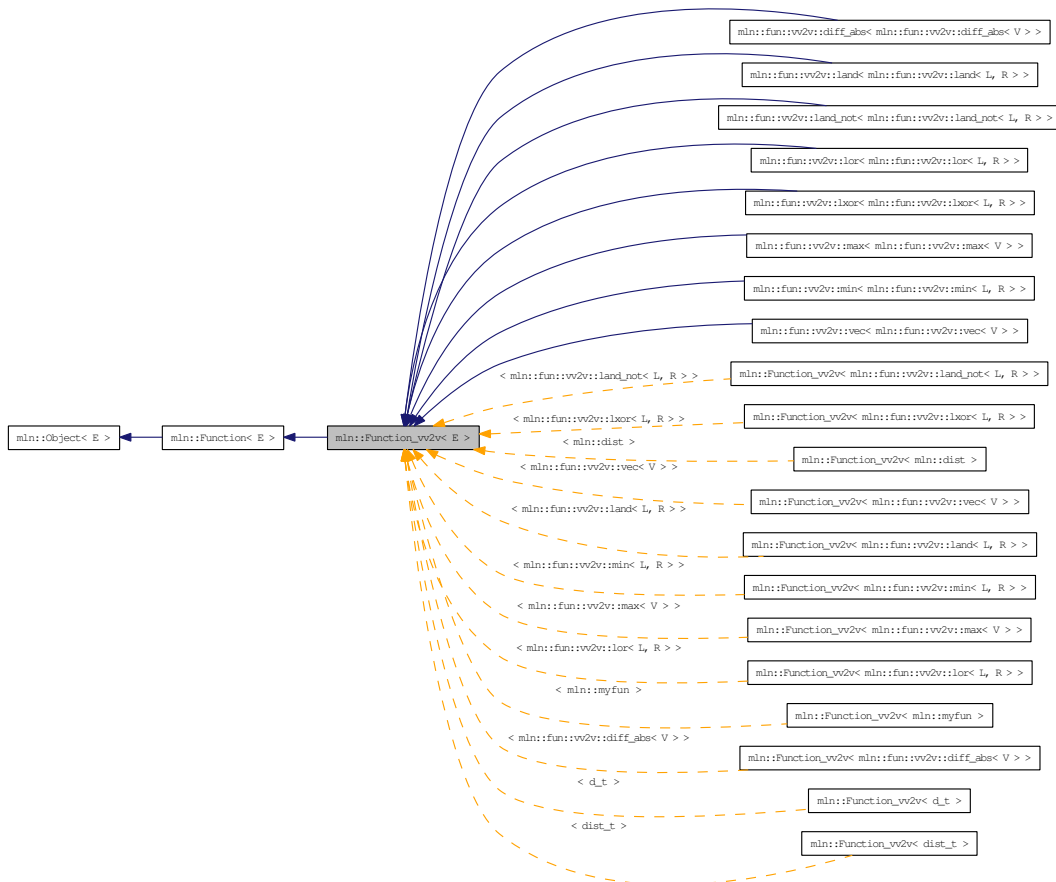
The parameter *E* is the exact type.

10.183 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 >:



10.183.1 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.

10.184 mln::fwd_pixter1d< I > Class Template Reference

Forward [pixel](#) iterator on a 1-D image with [border](#).

```
#include <pixter1d.hh>
```

Inherits mln::internal::forward_pixel_iterator_base_< I, mln::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.

10.184.1 Detailed Description

```
template<typename I> class mln::fwd_pixter1d< I >
```

Forward [pixel](#) iterator on a 1-D image with [border](#).

10.184.2 Member Typedef Documentation

10.184.2.1 template<typename I> typedef I mln::fwd_pixter1d< I >::image

[Image](#) type.

10.184.3 Constructor & Destructor Documentation

10.184.3.1 template<typename I> mln::fwd_pixter1d< I >::fwd_pixter1d (I & *image*)
[inline]

Constructor.

Parameters:

← *image* The image this [pixel](#) iterator is bound to.

10.184.4 Member Function Documentation

10.184.4.1 `template<typename E> void mln::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.

10.185 mln::fwd_pixter2d< I > Class Template Reference

Forward [pixel](#) iterator on a 2-D image with [border](#).

```
#include <pixter2d.hh>
```

Inherits mln::internal::forward_pixel_iterator_base_< I, mln::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.

10.185.1 Detailed Description

```
template<typename I> class mln::fwd_pixter2d< I >
```

Forward [pixel](#) iterator on a 2-D image with [border](#).

10.185.2 Member Typedef Documentation

10.185.2.1 template<typename I> typedef I mln::fwd_pixter2d< I >::image

[Image](#) type.

10.185.3 Constructor & Destructor Documentation

10.185.3.1 template<typename I> mln::fwd_pixter2d< I >::fwd_pixter2d (I & *image*)
[inline]

Constructor.

Parameters:

← *image* The image this [pixel](#) iterator is bound to.

10.185.4 Member Function Documentation

10.185.4.1 `template<typename E> void mln::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.

10.186 mln::fwd_pixter3d< I > Class Template Reference

Forward [pixel](#) iterator on a 3-D image with [border](#).

```
#include <pixter3d.hh>
```

Inherits mln::internal::forward_pixel_iterator_base_< I, mln::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.

10.186.1 Detailed Description

```
template<typename I> class mln::fwd_pixter3d< I >
```

Forward [pixel](#) iterator on a 3-D image with [border](#).

10.186.2 Member Typedef Documentation

10.186.2.1 template<typename I> typedef I mln::fwd_pixter3d< I >::image

[Image](#) type.

10.186.3 Constructor & Destructor Documentation

10.186.3.1 template<typename I> mln::fwd_pixter3d< I >::fwd_pixter3d (I & *image*)
[inline]

Constructor.

Parameters:

← *image* The image this [pixel](#) iterator is bound to.

10.186.4 Member Function Documentation

10.186.4.1 `template<typename E> void mln::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:

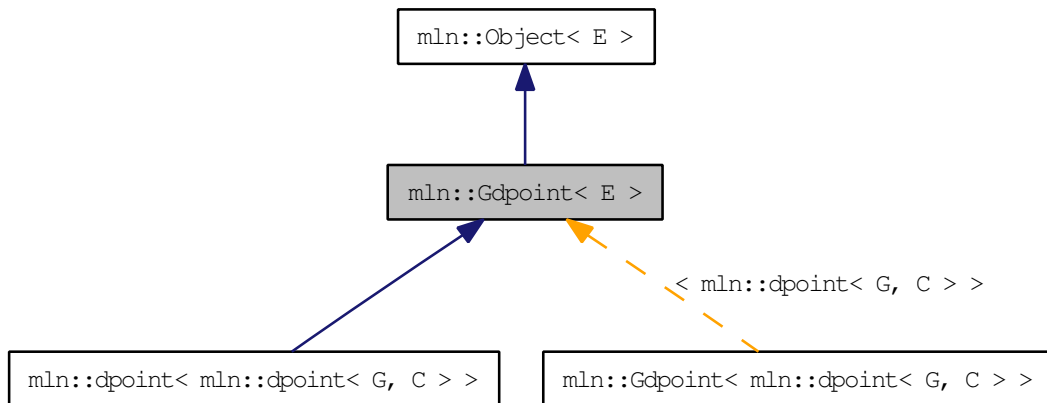
The iterator is valid.

10.187 mln::Gdpoint< E > Struct Template Reference

FIXME: Doc!

```
#include <gdpoint.hh>
```

Inheritance diagram for mln::Gdpoint< E >:



10.187.1 Detailed Description

```
template<typename E> struct mln::Gdpoint< E >
```

FIXME: Doc!

10.188 mln::Gdpoint< void > Struct Template Reference

Delta [point](#) site category flag type.

```
#include <gdpoint.hh>
```

10.188.1 Detailed Description

template<> struct mln::Gdpoint< void >

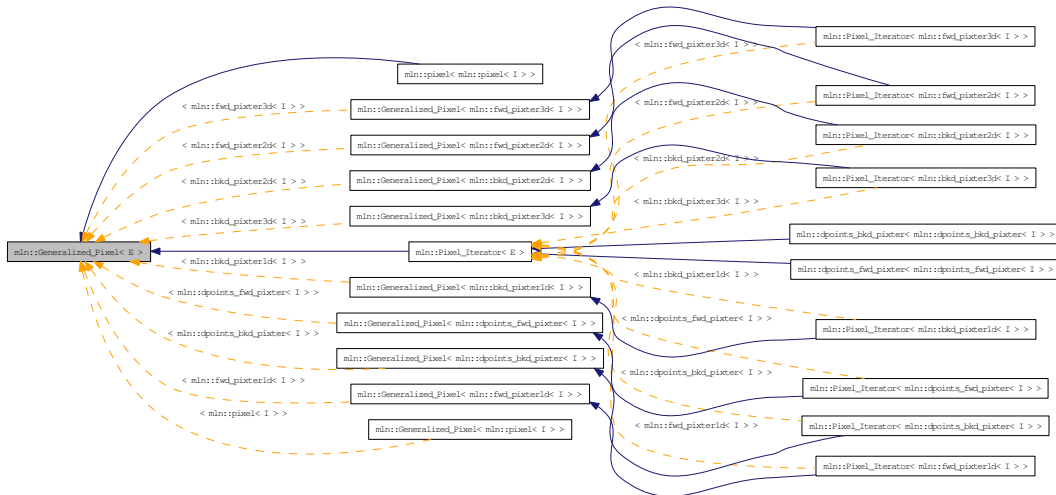
Delta [point](#) site category flag type.

10.189 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 >:



10.189.1 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.

10.190 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.

10.190.1 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.

10.190.2 Constructor & Destructor Documentation

```
10.190.2.1 template<unsigned D, typename P> mln::geom::complex_geometry< D, P  
>::complex_geometry () [inline]
```

Build a complex geometry object.

10.190.3 Member Function Documentation

10.190.3.1 `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).

10.190.3.2 `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 .

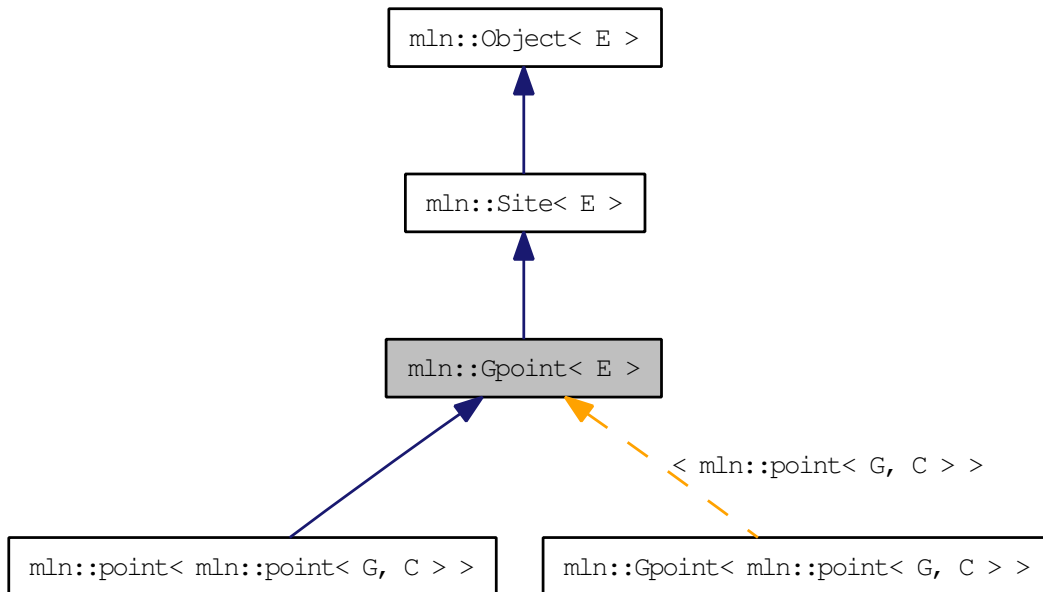
References `mln::topo::face< D >::face_id()`, and `mln::topo::face< D >::n()`.

10.191 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`.

10.191.1 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.

10.191.2 Friends And Related Function Documentation

10.191.2.1 `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:

- ← `p` A [grid point](#).
- ← `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](#)

10.191.2.2 `template<typename P, typename D> P & operator+= (Gpoint< P > &p, const Gdpoint< D > &dp)` [[related](#)]

Shift a [point](#) by a delta-point `dp`.

Parameters:

- ↔ `p` The targeted [point](#).
- ← `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 .

10.191.2.3 `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:

← *lhs* A first [grid point](#).

← *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](#)

10.191.2.4 `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:

↔ *p* The targeted [point](#).

← *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 .

10.191.2.5 `template<typename P, typename D> P operator/ (const Gpoint< P > & p, const value::scalar_< D > & dp)` [related]

Divide a [point](#) by a scalar *s*.

Parameters:

- ↔ *p* The targeted [point](#).
- ← *dp* A scalar.

Returns:

A reference to the [point](#) *p* once divided by *s*.

10.191.2.6 `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:

- ↔ *ostr* An output stream.
- ← *p* A [grid point](#).

Returns:

The modified output stream *ostr*.

References `mln::debug::format()`.

10.191.2.7 `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:

- ← *lhs* A first [grid point](#).
- ← *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:

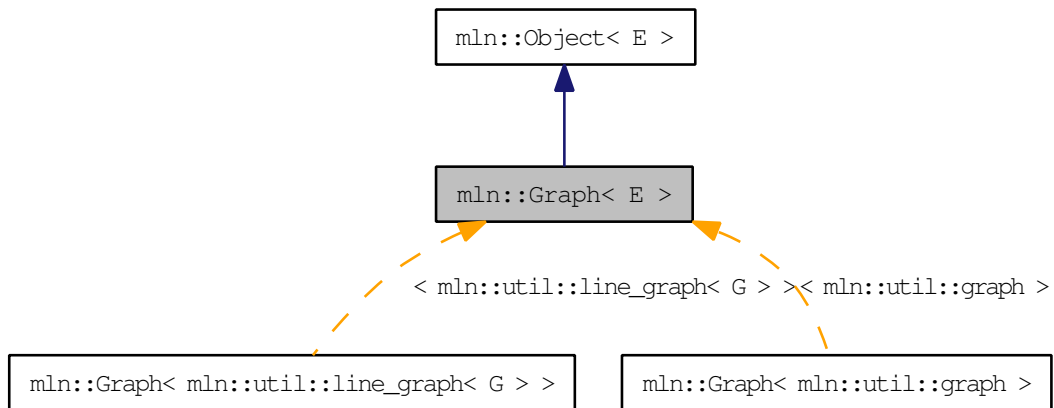
True if both [grid](#) points have the same coordinates, otherwise false.

10.192 mln::Graph< E > Struct Template Reference

Base class for implementation of [graph](#) classes.

```
#include <graph.hh>
```

Inheritance diagram for mln::Graph< E >:



10.192.1 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.

10.193 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](#).

10.193.1 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.

10.193.2 Member Typedef Documentation

10.193.2.1 typedef [util::array](#)<unsigned> mln::graph::attribute::card_t::result

Type of the computed [value](#).

10.194 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](#).

10.194.1 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.

10.194.2 Member Typedef Documentation

10.194.2.1 typedef [util::array](#)<unsigned> mln::graph::attribute::representative_t::result

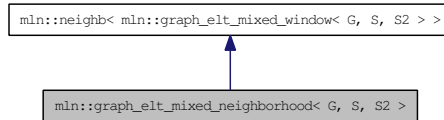
Type of the computed [value](#).

10.195 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< 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.

10.195.1 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.

10.195.2 Member Typedef Documentation

10.195.2.1 `template<typename W> typedef neighb_bkd_niter<W> mln::neighb< W >::bkd_niter` [inherited]

Backward site iterator associated type.

10.195.2.2 `template<typename W> typedef neighb_fwd_niter<W> mln::neighb< W >::fwd_niter` [inherited]

Forward site iterator associated type.

10.195.2.3 `template<typename W> typedef fwd_niter mln::neighb< W >::niter` [inherited]

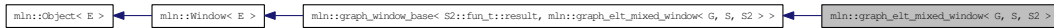
[Site](#) iterator associated type.

10.196 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 [graph_window_piter](#)< [target](#), [self_](#), [nbh_bkd_iter_](#) > [bkd_qiter](#)
Site_iterator type to browse the psites of the window w.r.t.
- typedef S::psite [center_t](#)
Type of the window center element.
- 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 target::graph_element [graph_element](#)
Type of the graph element pointed by this iterator.
- typedef target::psite [psite](#)
The type of psite corresponding to the window.
- typedef [fwd_qiter](#) [qiter](#)
The default qiter type.
- typedef super_::target [target](#)
Associated types.
- typedef P [site](#)
Associated types.

Public Member Functions

- bool [is_valid](#) () const
Return true by default.
- unsigned [delta](#) () const
Return the maximum coordinate gap between the window center and a window point.
- bool [is_centered](#) () const
Is the window centered?
- bool [is_empty](#) () const

Interface of the concept [Window](#).

- `bool is_symmetric () const`
Is the [window](#) symmetric?
- `self_ & sym ()`
Apply a central symmetry to the target [window](#).

10.196.1 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.

10.196.2 Member Typedef Documentation

10.196.2.1 `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.

10.196.2.2 `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.

10.196.2.3 `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.

10.196.2.4 `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.

10.196.2.5 `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](#).

10.196.2.6 `template<typename G, typename S, typename S2> typedef fwd_qiter
mln::graph_elt_mixed_window< G, S, S2 >::qiter`

The default qiter type.

10.196.2.7 `template<typename P, typename E> typedef P mln::graph_window_base< P, E >::site
[inherited]`

Associated types.

The type of site corresponding to the [window](#).

10.196.2.8 `template<typename G, typename S, typename S2> typedef super_::target
mln::graph_elt_mixed_window< G, S, S2 >::target`

Associated types.

10.196.3 Member Function Documentation

10.196.3.1 `template<typename P, typename E> unsigned mln::graph_window_base< P, E
>::delta () const [inline, inherited]`

Return the maximum coordinate gap between the [window](#) center and a [window point](#).

10.196.3.2 `template<typename P, typename E> bool mln::graph_window_base< P, E
>::is_centered () const [inline, inherited]`

Is the [window](#) centered?

10.196.3.3 `template<typename P, typename E> bool mln::graph_window_base< P, E
>::is_empty () const [inline, inherited]`

Interface of the concept [Window](#).

Is the [window](#) is empty?

10.196.3.4 `template<typename P, typename E> bool mln::graph_window_base< P, E
>::is_symmetric () const [inline, inherited]`

Is the [window](#) symmetric?

10.196.3.5 `template<typename P, typename E> bool mln::graph_window_base< P, E >::is_valid
() const [inline, inherited]`

Return true by default.

Reimplemented in [mln::graph_elt_window_if< G, S, I >](#).

10.196.3.6 `template<typename P, typename E> graph_window_base< P, E > & mln::graph_window_base< P, E >::sym ()` [inline, inherited]

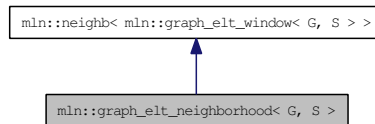
Apply a central symmetry to the target [window](#).

10.197 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< 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.

10.197.1 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.

10.197.2 Member Typedef Documentation

10.197.2.1 `template<typename W> typedef neighb_bkd_niter<W> mln::neighb< W >::bkd_niter` [inherited]

Backward site iterator associated type.

10.197.2.2 `template<typename W> typedef neighb_fwd_niter<W> mln::neighb< W >::fwd_niter` [inherited]

Forward site iterator associated type.

10.197.2.3 `template<typename W> typedef fwd_niter mln::neighb< W >::niter` [inherited]

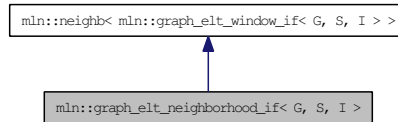
[Site](#) iterator associated type.

10.198 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](#)< 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

- [graph_elt_neighborhood_if](#) (const [Image](#)< I > &mask)
- [graph_elt_neighborhood_if](#) ()
Constructors @ { Construct an invalid neighborhood.
- const I & [mask](#) () const
@ }

10.198.1 Detailed Description

```
template<typename G, typename S, typename I> struct mln::graph_elt_neighborhood_if< G, S, I >
```

Elementary neighborhood_if on [graph](#) class.

10.198.2 Member Typedef Documentation

10.198.2.1 `template<typename W> typedef neighb_bkd_niter<W> mln::neighb< W >::bkd_niter` *[inherited]*

Backward site iterator associated type.

10.198.2.2 `template<typename W> typedef neighb_fwd_niter<W> mln::neighb< W >::fwd_niter` [inherited]

Forward site iterator associated type.

10.198.2.3 `template<typename W> typedef fwd_niter mln::neighb< W >::niter` [inherited]

[Site](#) iterator associated type.

10.198.3 Constructor & Destructor Documentation

10.198.3.1 `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.

10.198.3.2 `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:

← *mask* A [graph](#) image of Boolean.

10.198.4 Member Function Documentation

10.198.4.1 `template<typename G, typename S, typename I> const I & mln::graph_elt_neighborhood_if< G, S, I >::mask () const` [inline]

@ }

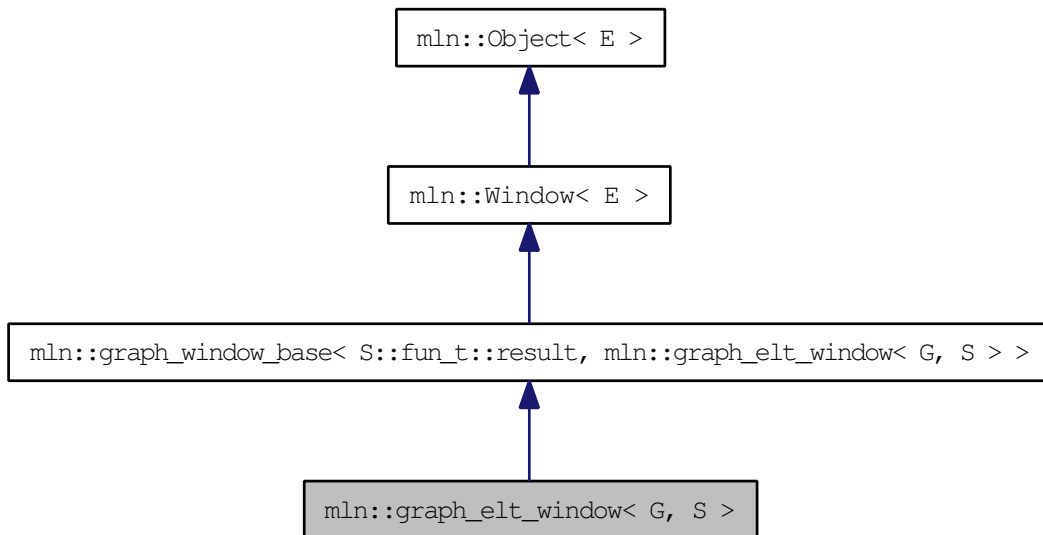
Return the [graph](#) image used as mask.

10.199 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 `graph_window_piter< S, self_, nbh_bkd_iter_ >` `bkd_qiter`
Site_iterator type to browse the psites of the *window* w.r.t.
- typedef `S::psite` `center_t`
Type of the window center element.
- 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 `S::graph_element` `graph_element`
Type of the graph element pointed by this iterator.
- typedef `S::psite` `psite`
The type of psite corresponding to the window.
- typedef `fwd_qiter` `qiter`
The default qiter type.
- typedef `S` `target`
Associated types.
- typedef `P` `site`
Associated types.

Public Member Functions

- `bool is_valid () const`
Return true by default.
- `unsigned delta () const`
Return the maximum coordinate gap between the [window](#) center and a [window point](#).
- `bool is_centered () const`
Is the [window](#) centered?
- `bool is_empty () const`
Interface of the concept [Window](#).
- `bool is_symmetric () const`
Is the [window](#) symmetric?
- `self_ & sym ()`
Apply a central symmetry to the target [window](#).

10.199.1 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.

10.199.2 Member Typedef Documentation

10.199.2.1 `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.

10.199.2.2 `template<typename G, typename S> typedef S ::psite mln::graph_elt_window< G, S >::center_t`

Type of the [window](#) center element.

10.199.2.3 `template<typename G, typename S> typedef graph_window_piter<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.

10.199.2.4 `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.

10.199.2.5 `template<typename G, typename S> typedef S ::psite mln::graph_elt_window< G, S
>::psite`

The type of psite corresponding to the [window](#).

10.199.2.6 `template<typename G, typename S> typedef fwd_qiter mln::graph_elt_window< G, S
>::qiter`

The default qiter type.

10.199.2.7 `template<typename P, typename E> typedef P mln::graph_window_base< P, E >::site
[inherited]`

Associated types.

The type of site corresponding to the [window](#).

10.199.2.8 `template<typename G, typename S> typedef S mln::graph_elt_window< G, S
>::target`

Associated types.

10.199.3 Member Function Documentation

10.199.3.1 `template<typename P, typename E> unsigned mln::graph_window_base< P, E
>::delta () const [inline, inherited]`

Return the maximum coordinate gap between the [window](#) center and a [window point](#).

10.199.3.2 `template<typename P, typename E> bool mln::graph_window_base< P, E
>::is_centered () const [inline, inherited]`

Is the [window](#) centered?

10.199.3.3 `template<typename P, typename E> bool mln::graph_window_base< P, E
>::is_empty () const [inline, inherited]`

Interface of the concept [Window](#).

Is the [window](#) is empty?

10.199.3.4 `template<typename P, typename E> bool mln::graph_window_base< P, E >::is_symmetric () const` [inline, inherited]

Is the [window](#) symmetric?

10.199.3.5 `template<typename P, typename E> bool mln::graph_window_base< P, E >::is_valid () const` [inline, inherited]

Return true by default.

Reimplemented in [mln::graph_elt_window_if< G, S, I >](#).

10.199.3.6 `template<typename P, typename E> graph_window_base< P, E > & mln::graph_window_base< P, E >::sym ()` [inline, inherited]

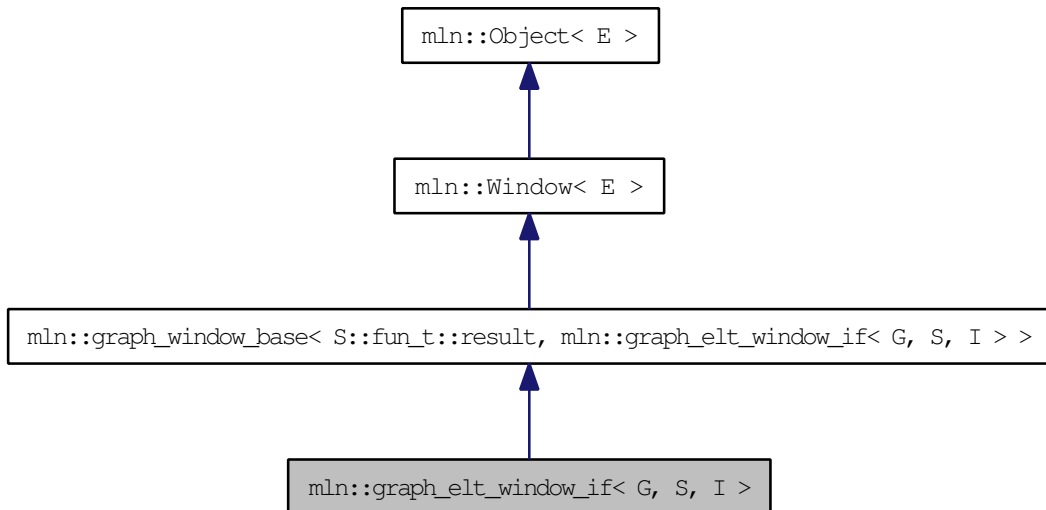
Apply a central symmetry to the target [window](#).

10.200 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 [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 [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 target::psite [psite](#)
The type of psite corresponding to the window.
- typedef [fwd_qiter](#) [qiter](#)
The default qiter type.
- typedef S [target](#)
@/
- typedef P [site](#)
Associated types.

Public Member Functions

- void [change_mask](#) (const [Image](#)< I > &mask)
Change mask image.
- [graph_elt_window_if](#) (const [Image](#)< I > &mask)
- [graph_elt_window_if](#) ()
Constructor.
- bool [is_valid](#) () const
Return true by default.
- const I & [mask](#) () const
Return the [graph](#) image used as mask.
- unsigned [delta](#) () const
Return the maximum coordinate gap between the [window](#) center and a [window point](#).
- bool [is_centered](#) () const
Is the [window](#) centered?
- bool [is_empty](#) () const
Interface of the concept [Window](#).
- bool [is_symmetric](#) () const
Is the [window](#) symmetric?
- [self_](#) & [sym](#) ()
Apply a central symmetry to the target [window](#).

10.200.1 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.

10.200.2 Member Typedef Documentation

10.200.2.1 `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.

10.200.2.2 `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.

10.200.2.3 `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.

10.200.2.4 `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](#).

10.200.2.5 `template<typename G, typename S, typename I> typedef fwd_qiter mln::graph_elt_window_if< G, S, I >::qiter`

The default qiter type.

10.200.2.6 `template<typename P, typename E> typedef P mln::graph_window_base< P, E >::site [inherited]`

Associated types.

The type of site corresponding to the [window](#).

10.200.2.7 `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.

10.200.3 Constructor & Destructor Documentation

10.200.3.1 `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](#).

10.200.3.2 `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:

← *mask* A [graph](#) image of bool.

See also:

[vertex_image](#), [edge_image](#).

10.200.4 Member Function Documentation

10.200.4.1 `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.

References `mln::graph_elt_window_if< G, S, I >::is_valid()`.

10.200.4.2 `template<typename P, typename E> unsigned mln::graph_window_base< P, E >::delta () const [inline, inherited]`

Return the maximum coordinate gap between the [window](#) center and a [window point](#).

10.200.4.3 `template<typename P, typename E> bool mln::graph_window_base< P, E >::is_centered () const [inline, inherited]`

Is the [window](#) centered?

10.200.4.4 `template<typename P, typename E> bool mln::graph_window_base< P, E >::is_empty () const [inline, inherited]`

Interface of the concept [Window](#).

Is the [window](#) is empty?

10.200.4.5 `template<typename P, typename E> bool mln::graph_window_base< P, E >::is_symmetric () const [inline, inherited]`

Is the [window](#) symmetric?

10.200.4.6 `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< P, E >`.

Referenced by `mln::graph_elt_window_if< G, S, I >::change_mask()`.

10.200.4.7 `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.

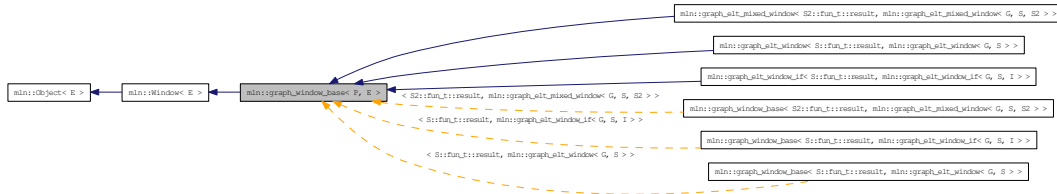
10.200.4.8 `template<typename P, typename E> graph_window_base< P, E > &
mln::graph_window_base< P, E >::sym () [inline, inherited]`

Apply a central symmetry to the target [window](#).

10.201 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.
- unsigned [delta](#) () const
Return the maximum coordinate gap between the [window](#) center and a [window point](#).
- bool [is_centered](#) () const
Is the [window](#) centered?
- bool [is_empty](#) () const
Interface of the concept [Window](#).
- bool [is_symmetric](#) () const
Is the [window](#) symmetric?
- [self_ & sym](#) ()
Apply a central symmetry to the target [window](#).

10.201.1 Detailed Description

```
template<typename P, typename E> class mln::graph_window_base< P, E >
```

Template Parameters:

P [Site](#) type.

10.201.2 Member Typedef Documentation

10.201.2.1 `template<typename P, typename E> typedef P mln::graph_window_base< P, E >::site`

Associated types.

The type of site corresponding to the [window](#).

10.201.3 Member Function Documentation

10.201.3.1 `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](#).

10.201.3.2 `template<typename P, typename E> bool mln::graph_window_base< P, E >::is_centered () const [inline]`

Is the [window](#) centered?

10.201.3.3 `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?

10.201.3.4 `template<typename P, typename E> bool mln::graph_window_base< P, E >::is_symmetric () const [inline]`

Is the [window](#) symmetric?

10.201.3.5 `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 >](#).

10.201.3.6 `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](#).

10.202 mln::graph_window_if_piter< S, W, I > Class Template Reference

Forward iterator on line [graph window](#).

```
#include <graph_window_if_piter.hh>
```

Inherits mln::internal::site_relative_iterator_base< W, mln::graph_window_if_piter< S, W, I > >, and mln::internal::is_masked_impl_selector< S, W::mask_t::domain_t, mln::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.
- const S::graph_element & [element](#) () const
Return the [graph](#) element pointed by this iterator.
- unsigned [id](#) () const
Return the [graph](#) element id.
- [graph_window_if_piter](#) ()
Construction.

10.202.1 Detailed Description

```
template<typename S, typename W, typename I> class mln::graph_window_if_piter< S, W, I >
```

Forward iterator on line [graph window](#).

10.202.2 Member Typedef Documentation

10.202.2.1 `template<typename S, typename W, typename I> typedef S::fun_t ::result mln::graph_window_if_piter< S, W, I >::P`

Associated types.

10.202.3 Constructor & Destructor Documentation

10.202.3.1 `template<typename S, typename W, typename I> mln::graph_window_if_piter< S, W, I >::graph_window_if_piter () [inline]`

Construction.

10.202.4 Member Function Documentation

10.202.4.1 `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.

10.202.4.2 `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.

10.202.4.3 `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-define this method. The actual "next" operation has to be defined through the *next_* method.

Precondition:

The iterator is valid.

10.203 mln::graph_window_piter< S, W, I > Class Template Reference

Forward iterator on line [graph window](#).

```
#include <graph_window_piter.hh>
```

Inherits mln::internal::site_relative_iterator_base< W, mln::graph_window_piter< S, W, I >, W::center_t >, and mln::internal::impl_selector< W::center_t, W::psite, mln::graph_window_piter< S, W, I > >.

Public Types

- 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.
- typedef S::fun_t::result P
Associated types
Type of the [window](#) elements.

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](#).
- const [graph_element](#) & [element](#) () const
Return the [graph](#) element pointed by this iterator.
- unsigned [id](#) () const
Return the [graph](#) element id.
- 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](#).
- 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](#).

- [graph_window_piter](#) ()
Construction.

10.203.1 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.

10.203.2 Member Typedef Documentation

10.203.2.1 `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.

10.203.2.2 `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.

10.203.2.3 `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.

10.203.3 Constructor & Destructor Documentation

10.203.3.1 `template<typename S, typename W, typename I> mln::graph_window_piter< S, W, I
>::graph_window_piter () [inline]`

Construction.

10.203.3.2 `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.

10.203.3.3 `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.

10.203.4 Member Function Documentation

10.203.4.1 `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](#).

10.203.4.2 `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.

10.203.4.3 `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.

10.203.4.4 `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.

10.203.4.5 `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](#).

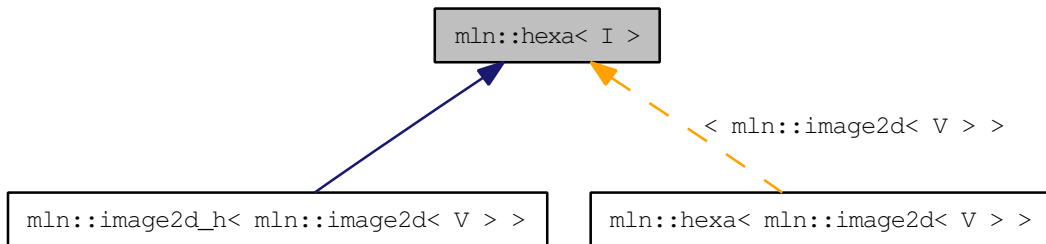
[Window](#) elements different from the center come from the target site [set](#).

10.204 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.

10.204.1 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 —————

10.204.2 Member Typedef Documentation

10.204.2.1 `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.

10.204.2.2 `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.

10.204.2.3 `template<typename I> typedef I ::lvalue mln::hexa< I >::lvalue`

Lvalue associated type.

10.204.2.4 `template<typename I> typedef point2d_h mln::hexa< I >::psite`

[Point](#) site type.

Reimplemented in [mln::image2d_h< V >](#).

10.204.2.5 `template<typename I> typedef I ::rvalue mln::hexa< I >::rvalue`

Return type of read-only access.

10.204.2.6 `template<typename I> typedef hexa< tag::image_<I> > mln::hexa< I >::skeleton`

Skeleton.

10.204.2.7 `template<typename I> typedef I ::value mln::hexa< I >::value`

Value associated type.

10.204.3 Constructor & Destructor Documentation**10.204.3.1** `template<typename I> mln::hexa< I >::hexa () [inline]`

Constructor without argument.

10.204.3.2 `template<typename I> mln::hexa< I >::hexa (I & ima) [inline]`

Constructor with an base image.

10.204.4 Member Function Documentation**10.204.4.1** `template<typename I> const box2d_h & mln::hexa< I >::domain () const [inline]`

Give the definition domain.

10.204.4.2 `template<typename I> bool mln::hexa< I >::has (const psite & p) const [inline]`

Test if *p* belongs to the image domain.

Referenced by `mln::hexa< I >::operator()`.

10.204.4.3 `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*.

References `mln::hexa< I >::has()`.

10.204.4.4 `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*.

References `mln::hexa< I >::has()`.

10.205 mln::histo::array< T > Struct Template Reference

Generic histogram class over a [value set](#) with type T.

```
#include <array.hh>
```

10.205.1 Detailed Description

```
template<typename T> struct mln::histo::array< T >
```

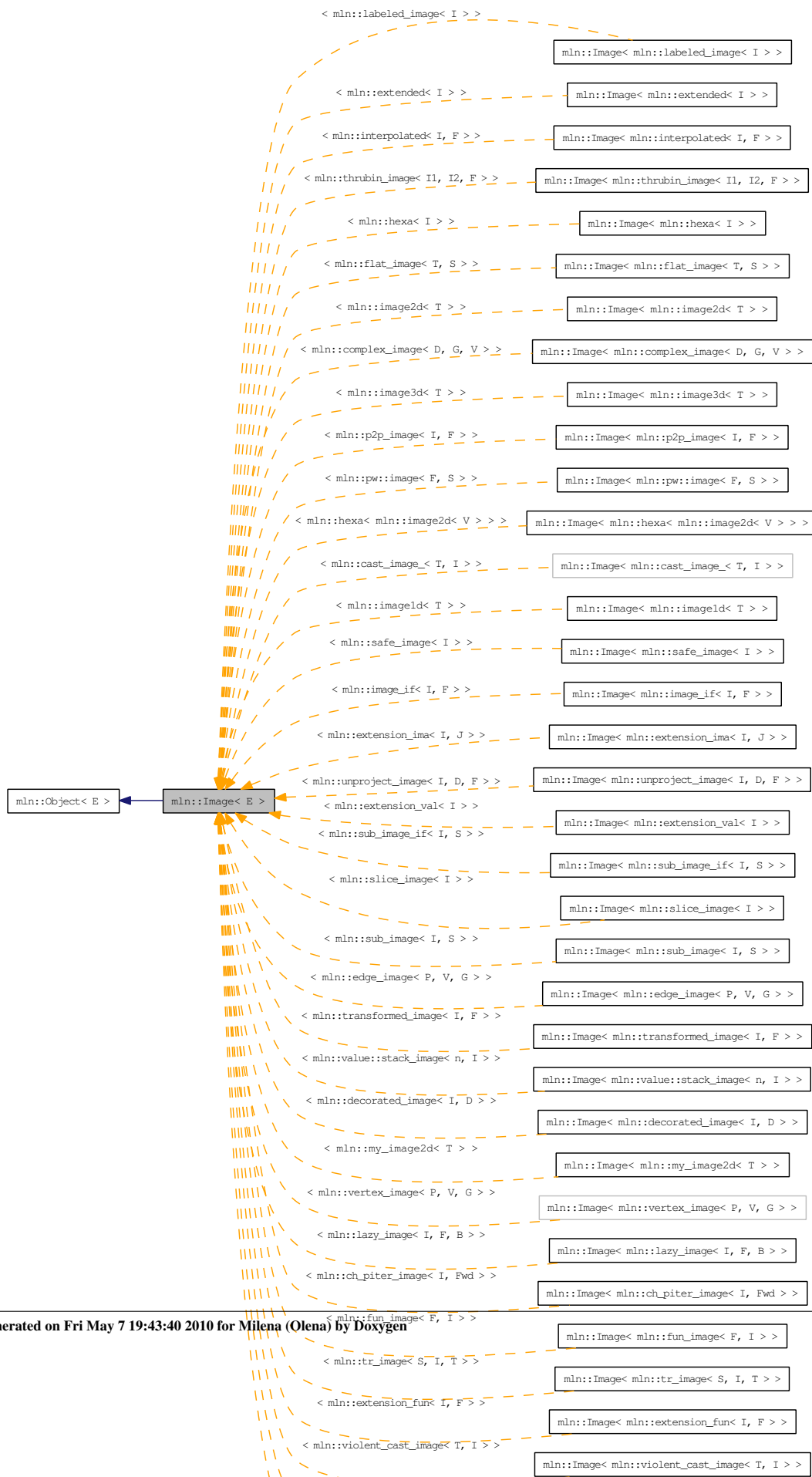
Generic histogram class over a [value set](#) with type T.

10.206 mln::Image< E > Struct Template Reference

Base class for implementation of image classes.

```
#include <image.hh>
```

Inheritance diagram for mln::Image< E >:



10.206.1 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.

10.207 mln::image1d< T > Struct Template Reference

Basic 1D image class.

```
#include <image1d.hh>
```

Inherits mln::internal::image_primary< T, mln::box, mln::image1d< T > >.

Package 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.

Package Functions

- const [box1d](#) & [bbox](#) () const
Give the bounding [box](#) domain.
- unsigned [border](#) () const
Give the [border](#) thickness.
- T * [buffer](#) ()
Give a hook to the [value](#) buffer.
- const T * [buffer](#) () const
Give a hook to the [value](#) buffer.
- int [delta_index](#) (const [dpoint1d](#) &dp) const
Give the offset corresponding to the delta-point dp.
- const [box1d](#) & [domain](#) () const
Give the definition domain.
- T & [element](#) (unsigned i)
Read-write access to the i-th image [value](#) (including the [border](#)).
- const T & [element](#) (unsigned i) const
Read-only access to the i-th image [value](#) (including the [border](#)).
- bool [has](#) (const [point1d](#) &p) const

Test if `p` is valid.

- `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.
- `image1d` ()
Constructor without argument.
- 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`.

10.207.1 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`.

10.207.2 Member Typedef Documentation

10.207.2.1 `template<typename T> typedef T& mln::image1d< T >::lvalue` [package]

Return type of read-write access.

10.207.2.2 `template<typename T> typedef const T& mln::image1d< T >::rvalue` [package]

Return type of read-only access.

10.207.2.3 `template<typename T> typedef image1d< tag::value_<T> > mln::image1d< T >::skeleton` [package]

Skeleton.

10.207.2.4 `template<typename T> typedef T mln::image1d< T >::value` [package]

Value associated type.

10.207.3 Constructor & Destructor Documentation

10.207.3.1 `template<typename T> mln::image1d< T >::image1d ()` [inline, package]

Constructor without argument.

10.207.3.2 `template<typename T> mln::image1d< T >::image1d (unsigned ninds, unsigned bdr = border::thickness)` [inline, package]

Constructor with the number of indices and the `border` thickness.

References `mln::make::box1d()`.

10.207.3.3 `template<typename T> mln::image1d< T >::image1d (const box1d & b, unsigned bdr = border::thickness)` [inline, package]

Constructor with a `box` and the `border` thickness.

10.207.4 Member Function Documentation

10.207.4.1 `template<typename T> const box1d & mln::image1d< T >::bbox () const` [inline, package]

Give the bounding `box` domain.

10.207.4.2 `template<typename T> unsigned mln::image1d< T >::border () const` [inline, package]

Give the `border` thickness.

10.207.4.3 `template<typename T> T * mln::image1d< T >::buffer ()` [inline, package]

Give a hook to the `value` buffer.

10.207.4.4 `template<typename T> const T * mln::image1d< T >::buffer () const` [inline, package]

Give a hook to the `value` buffer.

10.207.4.5 `template<typename T> int mln::image1d< T >::delta_index (const dpoint1d & dp) const` [inline, package]

Give the offset corresponding to the delta-point `dp`.

10.207.4.6 `template<typename T> const box1d & mln::image1d< T >::domain () const`
 [inline, package]

Give the definition domain.

10.207.4.7 `template<typename T> T & mln::image1d< T >::element (unsigned i)` [inline, package]

Read-write access to the i -th image [value](#) (including the [border](#)).

References `mln::image1d< T >::nelements()`.

10.207.4.8 `template<typename T> const T & mln::image1d< T >::element (unsigned i) const`
 [inline, package]

Read-only access to the i -th image [value](#) (including the [border](#)).

References `mln::image1d< T >::nelements()`.

10.207.4.9 `template<typename T> bool mln::image1d< T >::has (const point1d & p) const`
 [inline, package]

Test if p is valid.

Referenced by `mln::image1d< T >::operator()()`.

10.207.4.10 `template<typename T> unsigned mln::image1d< T >::nelements () const`
 [inline, package]

Give the number of cells (points including [border](#) ones).

Referenced by `mln::image1d< T >::element()`, and `mln::image1d< T >::point_at_index()`.

10.207.4.11 `template<typename T> unsigned mln::image1d< T >::ninds () const` [inline, package]

Give the number of indexes.

10.207.4.12 `template<typename T> T & mln::image1d< T >::operator() (const point1d & p)`
 [inline, package]

Read-write access to the image [value](#) located at [point](#) p .

References `mln::image1d< T >::has()`.

10.207.4.13 `template<typename T> const T & mln::image1d< T >::operator() (const point1d & p) const` [inline, package]

Read-only access to the image [value](#) located at [point](#) p .

References `mln::image1d< T >::has()`.

10.207.4.14 `template<typename T> point1d mln::image1d< T >::point_at_index (unsigned i)`
`const` [inline, package]

Give the [point](#) corresponding to the offset *o*.

References `mln::image1d< T >::nelements()`.

10.208 mln::image2d< T > Class Template Reference

Basic 2D image class.

```
#include <image2d.hh>
```

Inherits mln::internal::image_primary< T, mln::box, mln::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.
- T * [buffer](#) ()
Give a hook to the [value](#) buffer.
- const T * [buffer](#) () const
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` (const `box2d` &b, unsigned `bdr`=border::thickness)
Constructor with a `box` and the `border` thickness (default is 3).
- `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.
- 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.
- `T` & `operator()` (const `point2d` &p)
Read-write access to the image `value` located at `point` p .
- const `T` & `operator()` (const `point2d` &p) const
Read-only access to the image `value` located at `point` p .
- `point2d` `point_at_index` (unsigned `i`) const
Give the `point` corresponding to the index i .

10.208.1 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`.

10.208.2 Member Typedef Documentation

10.208.2.1 `template<typename T> typedef T& mln::image2d< T >::lvalue`

Return type of read-write access.

10.208.2.2 `template<typename T> typedef const T& mln::image2d< T >::rvalue`

Return type of read-only access.

10.208.2.3 `template<typename T> typedef image2d< tag::value_<T> > mln::image2d< T >::skeleton`

Skeleton.

10.208.2.4 `template<typename T> typedef T mln::image2d< T >::value`

Value associated type.

10.208.3 Constructor & Destructor Documentation

10.208.3.1 `template<typename T> mln::image2d< T >::image2d () [inline]`

Constructor without argument.

10.208.3.2 `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.

References `mln::make::box2d()`.

10.208.3.3 `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).

10.208.4 Member Function Documentation

10.208.4.1 `template<typename T> const box2d & mln::image2d< T >::bbox () const [inline]`

Give the bounding `box` domain.

10.208.4.2 `template<typename T> unsigned mln::image2d< T >::border () const [inline]`

Give the `border` thickness.

10.208.4.3 `template<typename T> T * mln::image2d< T >::buffer () [inline]`

Give a hook to the `value` buffer.

10.208.4.4 `template<typename T> const T * mln::image2d< T >::buffer () const [inline]`

Give a hook to the `value` buffer.

10.208.4.5 `template<typename T> int mln::image2d< T >::delta_index (const dpoint2d & dp) const [inline]`

Give the delta-index corresponding to the delta-point dp.

10.208.4.6 `template<typename T> const box2d & mln::image2d< T >::domain () const [inline]`

Give the definition domain.

Referenced by mln::morpho::line_gradient(), mln::make_debug_graph_image(), and mln::io::txt::save().

10.208.4.7 `template<typename T> T & mln::image2d< T >::element (unsigned i) [inline]`

Read-write access to the image [value](#) located at index i.

References mln::image2d< T >::nelements().

10.208.4.8 `template<typename T> const T & mln::image2d< T >::element (unsigned i) const [inline]`

Read-only access to the image [value](#) located at index i.

References mln::image2d< T >::nelements().

10.208.4.9 `template<typename T> bool mln::image2d< T >::has (const point2d & p) const [inline]`

Test if p is valid.

Referenced by mln::image2d< T >::operator>(), and mln::debug::put_word().

10.208.4.10 `template<typename T> unsigned mln::image2d< T >::ncols () const [inline]`

Give the number of columns.

10.208.4.11 `template<typename T> unsigned mln::image2d< T >::nelements () const [inline]`

Give the number of elements (points including [border](#) ones).

Referenced by mln::image2d< T >::element(), and mln::image2d< T >::point_at_index().

10.208.4.12 `template<typename T> unsigned mln::image2d< T >::nrows () const [inline]`

Give the number of rows.

10.208.4.13 `template<typename T> T & mln::image2d< T >::operator() (const point2d & p) [inline]`

Read-write access to the image [value](#) located at [point](#) p.

References `mln::image2d< T >::has()`.

10.208.4.14 `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`.

References `mln::image2d< T >::has()`.

10.208.4.15 `template<typename T> point2d mln::image2d< T >::point_at_index (unsigned i) const` `[inline]`

Give the `point` corresponding to the index `i`.

References `mln::image2d< T >::nelements()`.

10.209 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 `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.
- `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`.

10.209.1 Detailed Description

`template<typename V> struct mln::image2d_h< V >`

2d image based on an hexagonal mesh.

10.209.2 Member Typedef Documentation

10.209.2.1 `template<typename I> typedef hexa_bkd_piter_<box2d> mln::hexa< I >::bkd_piter`
[inherited]

FIXME : should it be in `box2d_h`? Backward [Site_Iterator](#) associated type.

10.209.2.2 `template<typename I> typedef hexa_fwd_piter_<box2d> mln::hexa< I >::fwd_piter`
[inherited]

FIXME : should it be in `box2d_h`? Forward [Site_Iterator](#) associated type.

10.209.2.3 `template<typename I> typedef I ::lvalue mln::hexa< I >::lvalue` [inherited]

Lvalue associated type.

10.209.2.4 `template<typename V> typedef point2d_h mln::image2d_h< V >::psite`

[Point](#) site type.

Reimplemented from `mln::hexa< I >`.

10.209.2.5 `template<typename I> typedef I ::rvalue mln::hexa< I >::rvalue` [inherited]

Return type of read-only access.

10.209.2.6 `template<typename I> typedef hexa< tag::image_<I> > mln::hexa< I >::skeleton`
[inherited]

Skeleton.

10.209.2.7 `template<typename I> typedef I ::value mln::hexa< I >::value` [inherited]

[Value](#) associated type.

10.209.3 Constructor & Destructor Documentation

10.209.3.1 `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
```

10.209.4 Member Function Documentation

10.209.4.1 `template<typename I> const box2d_h & mln::hexa< I >::domain () const [inline, inherited]`

Give the definition domain.

10.209.4.2 `template<typename I> bool mln::hexa< I >::has (const psite & p) const [inline, inherited]`

Test if `p` belongs to the image domain.

Referenced by `mln::hexa< I >::operator()`.

10.209.4.3 `template<typename I> hexa< I >::lvalue mln::hexa< I >::operator() (const point2d_h & p) [inline, inherited]`

Read-write access of `pixel value` at `hexa point` site `p`.

References `mln::hexa< I >::has()`.

10.209.4.4 `template<typename I> hexa< I >::rvalue mln::hexa< I >::operator() (const point2d_h & p) const [inline, inherited]`

Read-only access of `pixel value` at `hexa point` site `p`.

References `mln::hexa< I >::has()`.

10.210 mln::image3d< T > Struct Template Reference

Basic 3D image class.

```
#include <image3d.hh>
```

Inherits mln::internal::image_primary< T, mln::box, mln::image3d< T > >.

Package 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.

Package Functions

- const [box3d](#) & [bbox](#) () const
Give the bounding [box](#) domain.
- unsigned [border](#) () const
Give the [border](#) thickness.
- T * [buffer](#) ()
Give a hook to the [value](#) buffer.
- const T * [buffer](#) () const
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.
- 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 [point3d](#) &p) const

Test if *p* is valid.

- `image3d` (int nslis, int nrows, int ncols, unsigned bdr=border::thickness)
 Constructor with the numbers of indexes and the *border* thickness.
- `image3d` (const `box3d` &b, unsigned bdr=border::thickness)
 Constructor with a *box* and the *border* thickness (default is 3).
- `image3d` ()
 Constructor without argument.
- 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 `nslices` () const
 Give the number of slices.
- `T & operator()` (const `point3d` &p)
 Read-write access to the image *value* located at *point* *p*.
- const `T & operator()` (const `point3d` &p) const
 Read-only access to the image *value* located at *point* *p*.
- `point3d point_at_index` (unsigned o) const
 Give the *point* corresponding to the offset *o*.

10.210.1 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*.

10.210.2 Member Typedef Documentation

10.210.2.1 `template<typename T> typedef T& mln::image3d< T >::lvalue` [package]

Return type of read-write access.

10.210.2.2 `template<typename T> typedef const T& mln::image3d< T >::rvalue` [package]

Return type of read-only access.

10.210.2.3 `template<typename T> typedef image3d< tag::value_<T> > mln::image3d< T >::skeleton` [package]

Skeleton.

10.210.2.4 `template<typename T> typedef T mln::image3d< T >::value` [package]

[Value](#) associated type.

10.210.3 Constructor & Destructor Documentation

10.210.3.1 `template<typename T> mln::image3d< T >::image3d ()` [inline, package]

Constructor without argument.

10.210.3.2 `template<typename T> mln::image3d< T >::image3d (const box3d & b, unsigned bdr = border::thickness)` [inline, package]

Constructor with a [box](#) and the [border](#) thickness (default is 3).

10.210.3.3 `template<typename T> mln::image3d< T >::image3d (int nslis, int nrows, int ncols, unsigned bdr = border::thickness)` [inline, package]

Constructor with the numbers of indexes and the [border](#) thickness.

References `mln::make::box3d()`.

10.210.4 Member Function Documentation

10.210.4.1 `template<typename T> const box3d & mln::image3d< T >::bbox () const` [inline, package]

Give the bounding [box](#) domain.

10.210.4.2 `template<typename T> unsigned mln::image3d< T >::border () const` [inline, package]

Give the [border](#) thickness.

10.210.4.3 `template<typename T> T * mln::image3d< T >::buffer ()` [inline, package]

Give a hook to the [value](#) buffer.

10.210.4.4 `template<typename T> const T * mln::image3d< T >::buffer () const` [inline, package]

Give a hook to the [value](#) buffer.

10.210.4.5 `template<typename T> int mln::image3d< T >::delta_index (const dpoint3d & dp) const` [inline, package]

Fast [Image](#) method.

Give the offset corresponding to the delta-point `dp`.

10.210.4.6 `template<typename T> const box3d & mln::image3d< T >::domain () const` [inline, package]

Give the definition domain.

10.210.4.7 `template<typename T> T & mln::image3d< T >::element (unsigned i)` [inline, package]

Read-write access to the image [value](#) located at index `i`.

References `mln::image3d< T >::nelements()`.

10.210.4.8 `template<typename T> const T & mln::image3d< T >::element (unsigned i) const` [inline, package]

Read-only access to the image [value](#) located at index `i`.

References `mln::image3d< T >::nelements()`.

10.210.4.9 `template<typename T> bool mln::image3d< T >::has (const point3d & p) const` [inline, package]

Test if `p` is valid.

Referenced by `mln::image3d< T >::operator()()`.

10.210.4.10 `template<typename T> unsigned mln::image3d< T >::ncols () const` [inline, package]

Give the number of columns.

10.210.4.11 `template<typename T> unsigned mln::image3d< T >::nelements () const` [inline, package]

Give the number of cells (points including [border](#) ones).

Referenced by `mln::image3d< T >::element()`, and `mln::image3d< T >::point_at_index()`.

10.210.4.12 `template<typename T> unsigned mln::image3d< T >::nrows () const` [inline, package]

Give the number of rows.

10.210.4.13 `template<typename T> unsigned mln::image3d< T >::nslices () const` [inline, package]

Give the number of slices.

10.210.4.14 `template<typename T> T & mln::image3d< T >::operator() (const point3d & p)`
[inline, package]

Read-write access to the image [value](#) located at [point](#) p.

References `mln::image3d< T >::has()`.

10.210.4.15 `template<typename T> const T & mln::image3d< T >::operator() (const point3d & p) const` [inline, package]

Read-only access to the image [value](#) located at [point](#) p.

References `mln::image3d< T >::has()`.

10.210.4.16 `template<typename T> point3d mln::image3d< T >::point_at_index (unsigned o) const` [inline, package]

Give the [point](#) corresponding to the offset o.

References `mln::image3d< T >::nelements()`.

10.211 mln::image_if< I, F > Struct Template Reference

[Image](#) which domain is restricted by a function 'site -> Boolean'.

```
#include <image_if.hh>
```

Inherits mln::internal::image_domain_morpher< I, mln::p_if< I::domain_t, F >, mln::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.

10.211.1 Detailed Description

```
template<typename I, typename F> struct mln::image_if< I, F >
```

[Image](#) which domain is restricted by a function 'site -> Boolean'.

10.211.2 Member Typedef Documentation

10.211.2.1 `template<typename I, typename F> typedef image_if< tag::image_<I>, tag::function_<F> > mln::image_if< I, F >::skeleton`

Skeleton.

10.211.3 Constructor & Destructor Documentation

10.211.3.1 `template<typename I, typename F> mln::image_if< I, F >::image_if () [inline]`

Constructor without argument.

10.211.3.2 `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*.

10.211.4 Member Function Documentation

10.211.4.1 `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.

10.211.4.2 `template<typename I, typename F> mln::image_if< I, F >::operator image_if< const I, F > () const` `[inline]`

Const promotion via conversion.

10.212 mln::interpolated< I, F > Struct Template Reference

Makes the underlying image being accessed with floating coordinates.

```
#include <interpolated.hh>
```

Inherits mln::internal::image_identity< I, I::domain_t, mln::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

- bool [has](#) (const mln::algebra::vec< I::psite::dim, float > &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.

10.212.1 Detailed Description

```
template<typename I, template< class > class F> struct mln::interpolated< I, F >
```

Makes the underlying image being accessed with floating coordinates.

10.212.2 Member Typedef Documentation

10.212.2.1 `template<typename I, template< class > class F> typedef I ::lvalue mln::interpolated< I, F >::lvalue`

Return type of read-write access.

10.212.2.2 `template<typename I, template< class > class F> typedef I ::psite mln::interpolated< I, F >::psite`

[Point_Site](#) associated type.

10.212.2.3 `template<typename I, template< class > class F> typedef I ::rvalue mln::interpolated< I, F >::rvalue`

Return type of read-only access.

10.212.2.4 `template<typename I, template< class > class F> typedef interpolated< tag::image_<I>, F > mln::interpolated< I, F >::skeleton`

Skeleton.

10.212.2.5 `template<typename I, template< class > class F> typedef I ::value mln::interpolated< I, F >::value`

[Value](#) associated type.

10.212.3 Constructor & Destructor Documentation

10.212.3.1 `template<typename I, template< class > class F> mln::interpolated< I, F >::interpolated (I & ima) [inline]`

Constructors.

FIXME: don't we want a 'const' here?

10.212.4 Member Function Documentation

10.212.4.1 `template<typename I, template< class > class F> bool mln::interpolated< I, F >::has (const mln::algebra::vec< I::psite::dim, float > & v) const [inline]`

Test if a [pixel value](#) is accessible at v.

10.212.4.2 `template<typename I, template< class > class F> bool mln::interpolated< I, F >::is_valid () const [inline]`

Test if this image has been initialized.

10.213 mln::io::fld::fld_header Struct Reference

Define the header structure of an AVS field [data](#) file.

```
#include <header.hh>
```

10.213.1 Detailed Description

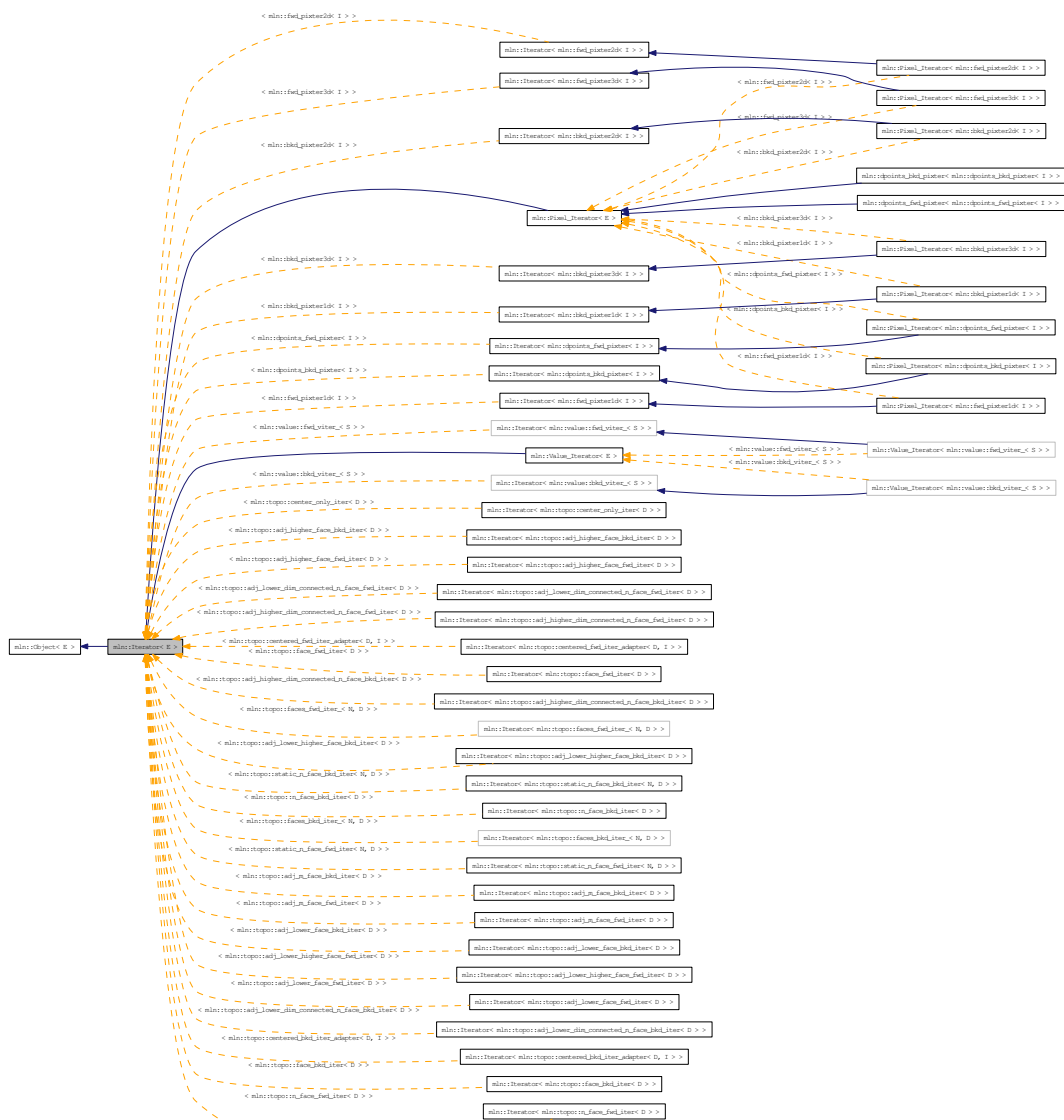
Define the header structure of an AVS field [data](#) file.

10.214 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.

10.214.1 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.

10.214.2 Member Function Documentation

10.214.2.1 `template<typename E> void mln::Iterator< E >::next ()` `[inline]`

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.

10.215 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](#) (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.
- [labeled_image](#) (const I &ima, const typename I::value &nlabels)
Constructor from an image ima and the number of labels nlabels.
- [labeled_image](#) ()
*Constructors
Constructor without argument.*

- `template<typename F>`
`void relabel (const Function_v2b< F > &f)`
Labels may be removed.
- `template<typename F>`
`void relabel (const Function_v2v< F > &f)`
Relabel according to a function.

Protected Member Functions

- `void update_data (const fun::i2v::array< typename I::value > &relabel_fun)`
Update bounding boxes information.

10.215.1 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).

10.215.2 Member Typedef Documentation

10.215.2.1 `template<typename I, typename E> typedef accu::shape::bbox<typename I ::psite>::result mln::labeled_image_base< I, E >::bbox_t` [inherited]

Type of the bounding component bounding boxes.

10.215.2.2 `template<typename I> typedef labeled_image< tag::image_<I> > mln::labeled_image< I >::skeleton`

Skeleton.

10.215.3 Constructor & Destructor Documentation

10.215.3.1 `template<typename I> mln::labeled_image< I >::labeled_image ()` [inline]

Constructors

Constructor without argument.

10.215.3.2 `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*.

10.215.3.3 `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.

References `mln::labeled_image_base< I, E >::bboxes()`, and `mln::data::compute()`.

10.215.4 Member Function Documentation

10.215.4.1 `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` [inline, inherited]

Return the bounding [box](#) of the component *label*.

Referenced by `mln::labeled_image_base< I, E >::subdomain()`.

10.215.4.2 `template<typename I, typename E> const util::array< typename labeled_image_base< I, E >::bbox_t > & mln::labeled_image_base< I, E >::bboxes () const` [inline, inherited]

Return the component bounding boxes.

Referenced by `mln::labeled_image< I >::labeled_image()`.

10.215.4.3 `template<typename I, typename E> I::value mln::labeled_image_base< I, E >::nlabels () const` [inline, inherited]

Return the number of labels;

10.215.4.4 `template<typename I, typename E> template<typename F> void mln::labeled_image_base< I, E >::relabel (const Function_v2b< F > & f)` [inline, inherited]

Labels may be removed.

This overload [make](#) sure the [labeling](#) is still contiguous.

References `mln::labeling::relabel_inplace()`, `mln::make::relabelfun()`, and `mln::labeled_image_base< I, E >::update_data()`.

10.215.4.5 `template<typename I, typename E> template<typename F> void mln::labeled_image_base< I, E >::relabel (const Function_v2v< F > & f)` [inline, inherited]

Relabel according to a function.

Merge or delete labels according to the given function. This method ensures that the [labeling](#) remains contiguous.

References mln::labeling::relabel_inplace(), mln::make::relabelfun(), and mln::labeled_image_base< I, E >::update_data().

10.215.4.6 `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` [inline, inherited]

Return the domain of the component with label `label`.

References mln::labeled_image_base< I, E >::bbox().

10.215.4.7 `template<typename I, typename E> void mln::labeled_image_base< I, E >::update_data (const fun::i2v::array< typename I::value > & relabel_fun)` [inline, protected, inherited]

Update bounding boxes information.

References mln::util::array< T >::size().

Referenced by mln::labeled_image_base< I, E >::relabel().

10.216 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_v2b`< F > &f)
Labels may be removed.
- template<typename F>
void `relabel` (const `Function_v2v`< F > &f)
Relabel according to a function.

Protected Member Functions

- void `update_data` (const fun::i2v::array< typename I::value > &relabel_fun)
Update bounding boxes information.

10.216.1 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).

10.216.2 Member Typedef Documentation

10.216.2.1 `template<typename I, typename E> typedef accu::shape::bbox<typename I
::site>::result mln::labeled_image_base< I, E >::bbox_t`

Type of the bounding component bounding boxes.

10.216.3 Constructor & Destructor Documentation

10.216.3.1 `template<typename I, typename E> mln::labeled_image_base< I, E
>::labeled_image_base () [inline]`

Constructors

Constructor without argument.

10.216.4 Member Function Documentation

10.216.4.1 `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
[inline]`

Return the bounding `box` of the component `label`.

Referenced by `mln::labeled_image_base< I, E >::subdomain()`.

10.216.4.2 `template<typename I, typename E> const util::array< typename
labeled_image_base< I, E >::bbox_t > & mln::labeled_image_base< I, E >::bboxes ()
const [inline]`

Return the component bounding boxes.

Referenced by `mln::labeled_image< I >::labeled_image()`.

10.216.4.3 `template<typename I, typename E> I::value mln::labeled_image_base< I, E >::nlabels () const [inline]`

Return the number of labels;.

10.216.4.4 `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.

References `mln::labeling::relabel_inplace()`, `mln::make::relabelfun()`, and `mln::labeled_image_base< I, E >::update_data()`.

10.216.4.5 `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.

References `mln::labeling::relabel_inplace()`, `mln::make::relabelfun()`, and `mln::labeled_image_base< I, E >::update_data()`.

10.216.4.6 `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 [inline]`

Return the domain of the component with label `label`.

References `mln::labeled_image_base< I, E >::bbox()`.

10.216.4.7 `template<typename I, typename E> void mln::labeled_image_base< I, E >::update_data (const fun::i2v::array< typename I::value > &relabel_fun) [inline, protected]`

Update bounding boxes information.

References `mln::util::array< T >::size()`.

Referenced by `mln::labeled_image_base< I, E >::relabel()`.

10.217 mln::lazy_image< I, F, B > Struct Template Reference

Image values are computed on the fly.

```
#include <lazy_image.hh>
```

Inherits mln::internal::image_identity< mln::trait::ch_value< I, F::result >::ret, I::domain_t, mln::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.
- **lvalue operator()** (const typename I::psite &p)
*Read and "write if possible" access of **pixel value** at **point** site p.*
- **rvalue operator()** (const typename I::psite &p) const
*Read-only 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.*
- F::result **operator()** (const typename F::input &x) const
*Read-only access of **pixel value** at F::input x.*

10.217.1 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.

10.217.2 Member Typedef Documentation

10.217.2.1 `template<typename I, typename F, typename B> typedef F ::result mln::lazy_image< I, F, B >::lvalue`

Return type of read-write access.

10.217.2.2 `template<typename I, typename F, typename B> typedef F ::result mln::lazy_image< I, F, B >::rvalue`

Return type of read access.

10.217.2.3 `template<typename I, typename F, typename B> typedef lazy_image< tag::image_<I>, F, B > mln::lazy_image< I, F, B >::skeleton`

Skeleton.

10.217.3 Constructor & Destructor Documentation

10.217.3.1 `template<typename I, typename F, typename B> mln::lazy_image< I, F, B >::lazy_image ()`

Constructors.

10.217.3.2 `template<typename I, typename F, typename B> mln::lazy_image< I, F, B >::lazy_image (const F & fun, const B & box) [inline]`

Constructors.

10.217.4 Member Function Documentation

10.217.4.1 `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`.

10.217.4.2 `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.

10.217.4.3 `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.

10.217.4.4 `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.

10.217.4.5 `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.

10.217.4.6 `template<typename I, typename F, typename B> F::result mln::lazy_image< I, F, B >::operator() (const typename F::input & x) const` [inline]

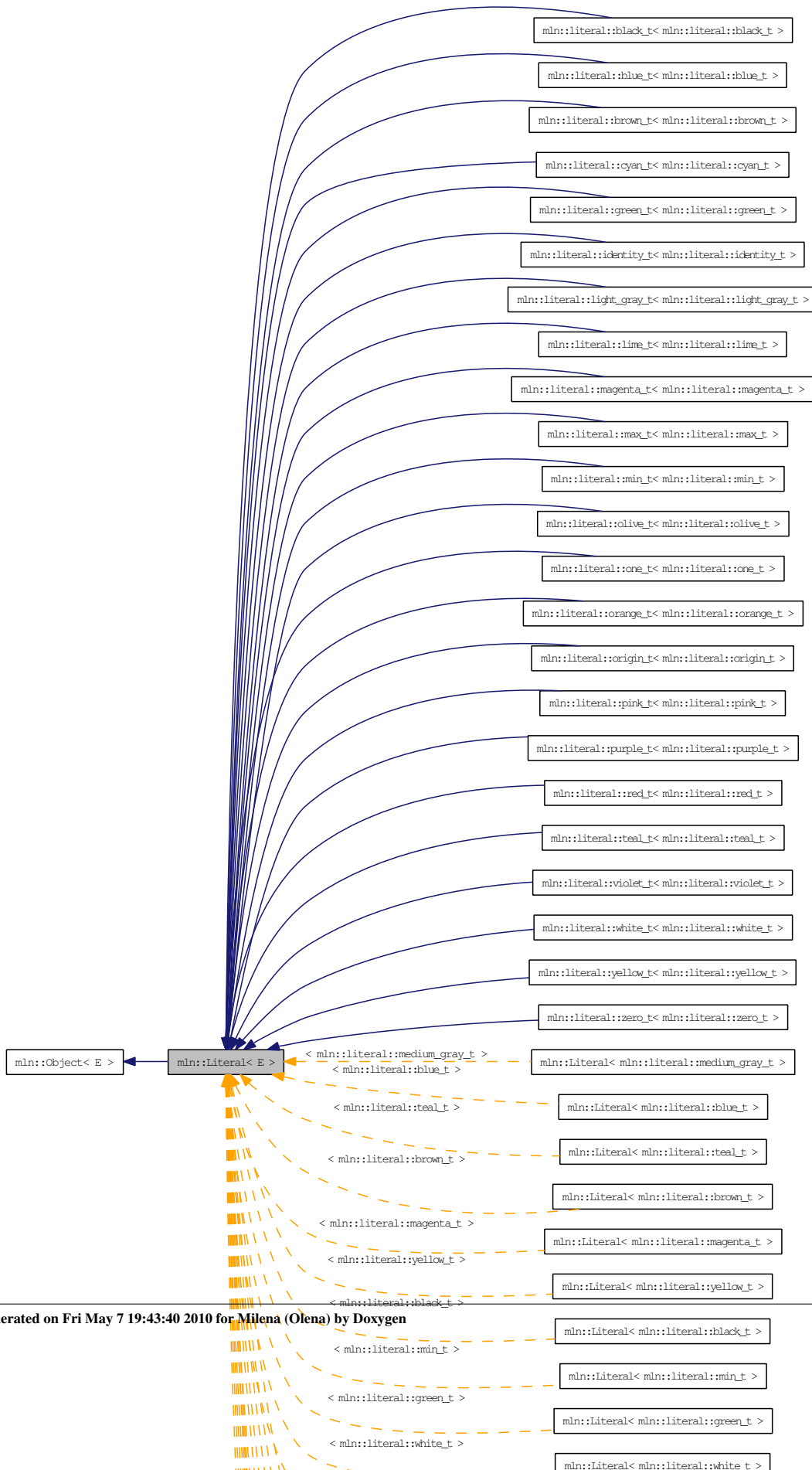
Read-only access of [pixel value](#) at F::input x.

10.218 mln::Literal< E > Struct Template Reference

Base class for implementation classes of literals.

```
#include <literal.hh>
```

Inheritance diagram for mln::Literal< E >:



10.218.1 Detailed Description

```
template<typename E> struct mln::Literal< E >
```

Base class for implementation classes of literals.

See also:

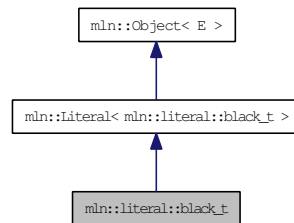
[mln::doc::Literal](#) for a complete documentation of this class contents.

10.219 mln::literal::black_t Struct Reference

Type of [literal](#) black.

```
#include <black.hh>
```

Inheritance diagram for mln::literal::black_t:



10.219.1 Detailed Description

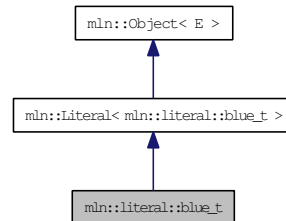
Type of [literal](#) black.

10.220 mln::literal::blue_t Struct Reference

Type of [literal](#) blue.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::blue_t:



10.220.1 Detailed Description

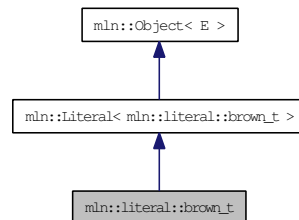
Type of [literal](#) blue.

10.221 mln::literal::brown_t Struct Reference

Type of [literal](#) brown.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::brown_t:



10.221.1 Detailed Description

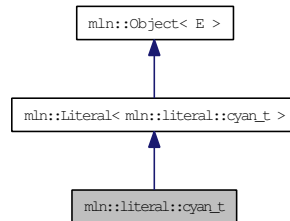
Type of [literal](#) brown.

10.222 mln::literal::cyan_t Struct Reference

Type of [literal](#) cyan.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::cyan_t:



10.222.1 Detailed Description

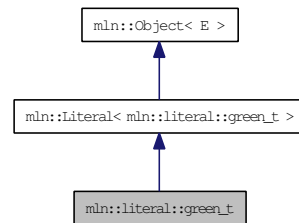
Type of [literal](#) cyan.

10.223 mln::literal::green_t Struct Reference

Type of [literal](#) green.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::green_t:



10.223.1 Detailed Description

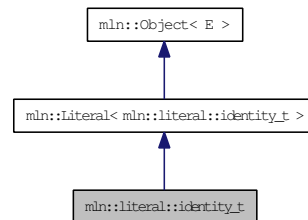
Type of [literal](#) green.

10.224 mln::literal::identity_t Struct Reference

Type of [literal](#) identity.

```
#include <identity.hh>
```

Inheritance diagram for mln::literal::identity_t:



10.224.1 Detailed Description

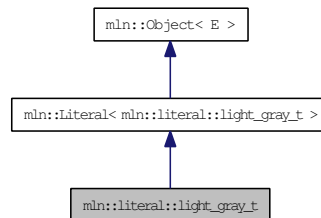
Type of [literal](#) identity.

10.225 mln::literal::light_gray_t Struct Reference

Type of [literal](#) grays.

```
#include <grays.hh>
```

Inheritance diagram for mln::literal::light_gray_t:



10.225.1 Detailed Description

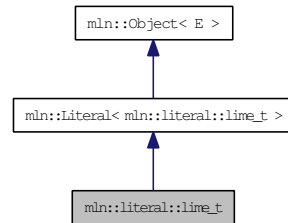
Type of [literal](#) grays.

10.226 mln::literal::lime_t Struct Reference

Type of [literal](#) lime.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::lime_t:



10.226.1 Detailed Description

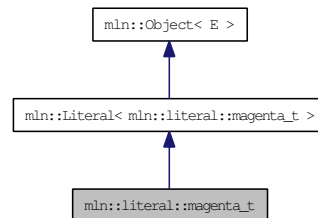
Type of [literal](#) lime.

10.227 mln::literal::magenta_t Struct Reference

Type of [literal](#) magenta.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::magenta_t:



10.227.1 Detailed Description

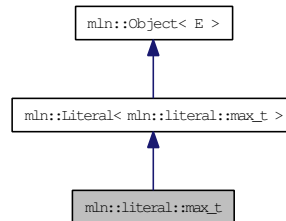
Type of [literal](#) magenta.

10.228 mln::literal::max_t Struct Reference

Type of [literal](#) max.

```
#include <max.hh>
```

Inheritance diagram for mln::literal::max_t:



10.228.1 Detailed Description

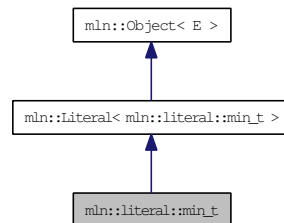
Type of [literal](#) max.

10.229 mln::literal::min_t Struct Reference

Type of [literal](#) min.

```
#include <min.hh>
```

Inheritance diagram for mln::literal::min_t:



10.229.1 Detailed Description

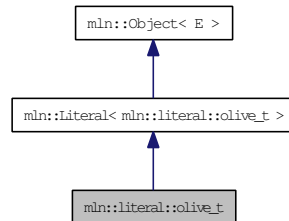
Type of [literal](#) min.

10.230 mln::literal::olive_t Struct Reference

Type of [literal](#) olive.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::olive_t:



10.230.1 Detailed Description

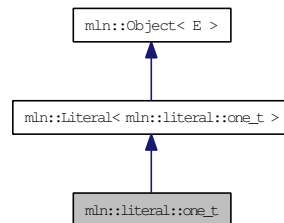
Type of [literal](#) olive.

10.231 mln::literal::one_t Struct Reference

Type of [literal](#) one.

```
#include <one.hh>
```

Inheritance diagram for mln::literal::one_t:



10.231.1 Detailed Description

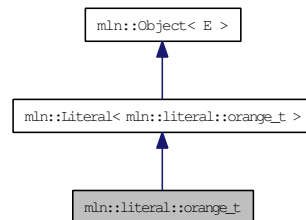
Type of [literal](#) one.

10.232 mln::literal::orange_t Struct Reference

Type of [literal](#) orange.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::orange_t:



10.232.1 Detailed Description

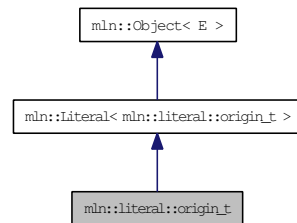
Type of [literal](#) orange.

10.233 mln::literal::origin_t Struct Reference

Type of [literal](#) origin.

```
#include <origin.hh>
```

Inheritance diagram for mln::literal::origin_t:



10.233.1 Detailed Description

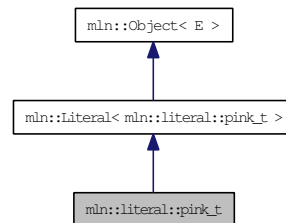
Type of [literal](#) origin.

10.234 mln::literal::pink_t Struct Reference

Type of [literal](#) pink.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::pink_t:



10.234.1 Detailed Description

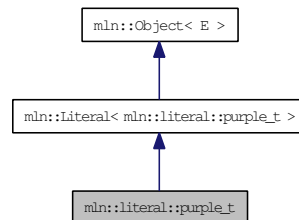
Type of [literal](#) pink.

10.235 mln::literal::purple_t Struct Reference

Type of [literal](#) purple.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::purple_t:



10.235.1 Detailed Description

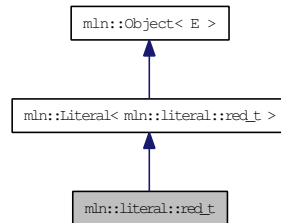
Type of [literal](#) purple.

10.236 mln::literal::red_t Struct Reference

Type of [literal](#) red.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::red_t:



10.236.1 Detailed Description

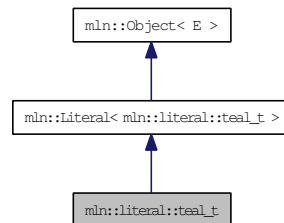
Type of [literal](#) red.

10.237 mln::literal::teal_t Struct Reference

Type of [literal](#) teal.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::teal_t:



10.237.1 Detailed Description

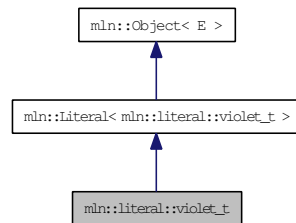
Type of [literal](#) teal.

10.238 mln::literal::violet_t Struct Reference

Type of [literal](#) violet.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::violet_t:



10.238.1 Detailed Description

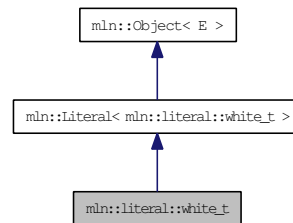
Type of [literal](#) violet.

10.239 mln::literal::white_t Struct Reference

Type of [literal](#) white.

```
#include <white.hh>
```

Inheritance diagram for mln::literal::white_t:



10.239.1 Detailed Description

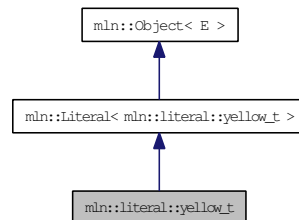
Type of [literal](#) white.

10.240 mln::literal::yellow_t Struct Reference

Type of [literal](#) yellow.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::yellow_t:



10.240.1 Detailed Description

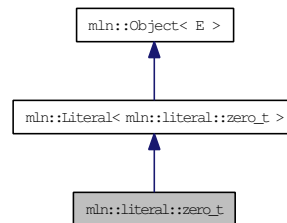
Type of [literal](#) yellow.

10.241 mln::literal::zero_t Struct Reference

Type of [literal](#) zero.

```
#include <zero.hh>
```

Inheritance diagram for mln::literal::zero_t:



10.241.1 Detailed Description

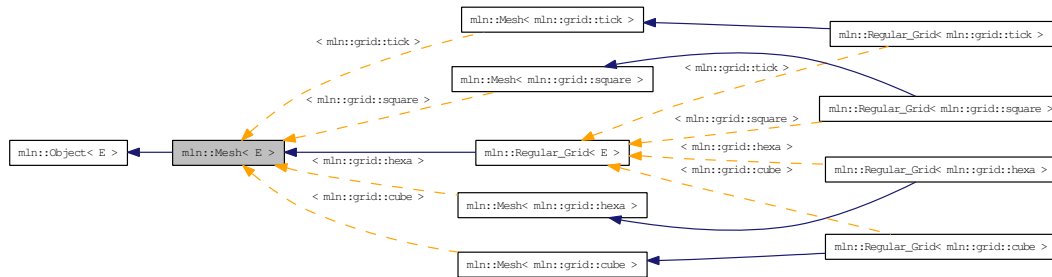
Type of [literal](#) zero.

10.242 mln::Mesh< E > Struct Template Reference

Base class for implementation classes of meshes.

```
#include <mesh.hh>
```

Inheritance diagram for mln::Mesh< E >:



10.242.1 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.

10.243 `mln::Meta_Accumulator< E >` Struct Template Reference

Base class for implementation of meta accumulators.

```
#include <meta_accumulator.hh>
```

Inherits `mln::Object< E >`.

Inherited by `mln::accu::meta::center`, `mln::accu::meta::count_adjacent_vertices`, `mln::accu::meta::count_labels`, `mln::accu::meta::count_value`, `mln::accu::meta::histo`, `mln::accu::meta::label_used`, `mln::accu::meta::logic::land`, `mln::accu::meta::logic::land_basic`, `mln::accu::meta::logic::lor`, `mln::accu::meta::logic::lor_basic`, `mln::accu::meta::maj_h`, `mln::accu::meta::math::count`, `mln::accu::meta::math::inf`, `mln::accu::meta::math::sum`, `mln::accu::meta::math::sup`, `mln::accu::meta::max_site`, `mln::accu::meta::nil`, `mln::accu::meta::p< mA >`, `mln::accu::meta::pair< A1, A2 >`, `mln::accu::meta::rms`, `mln::accu::meta::shape::bbox`, `mln::accu::meta::shape::height`, `mln::accu::meta::shape::volume`, `mln::accu::meta::stat::max`, `mln::accu::meta::stat::max_h`, `mln::accu::meta::stat::mean`, `mln::accu::meta::stat::median_alt< T >`, `mln::accu::meta::stat::median_h`, `mln::accu::meta::stat::min`, `mln::accu::meta::stat::min_h`, `mln::accu::meta::stat::rank`, `mln::accu::meta::stat::rank_high_quant`, `mln::accu::meta::tuple< n, >`, `mln::accu::meta::val< mA >`, and `mln::accu::stat::meta::deviation`.

10.243.1 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:

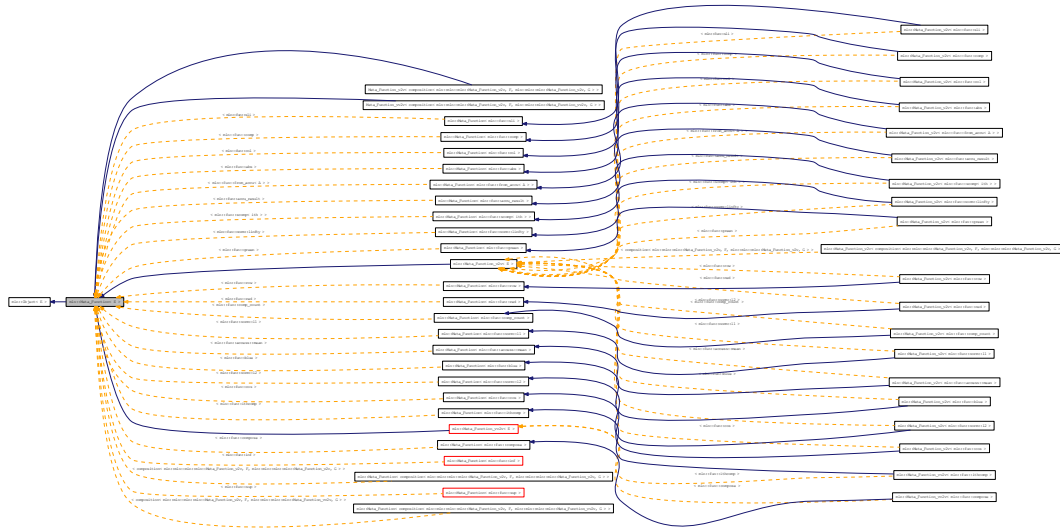
`mln::doc::Meta_Accumulator` for a complete documentation of this class contents.

10.244 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 >:



10.244.1 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:

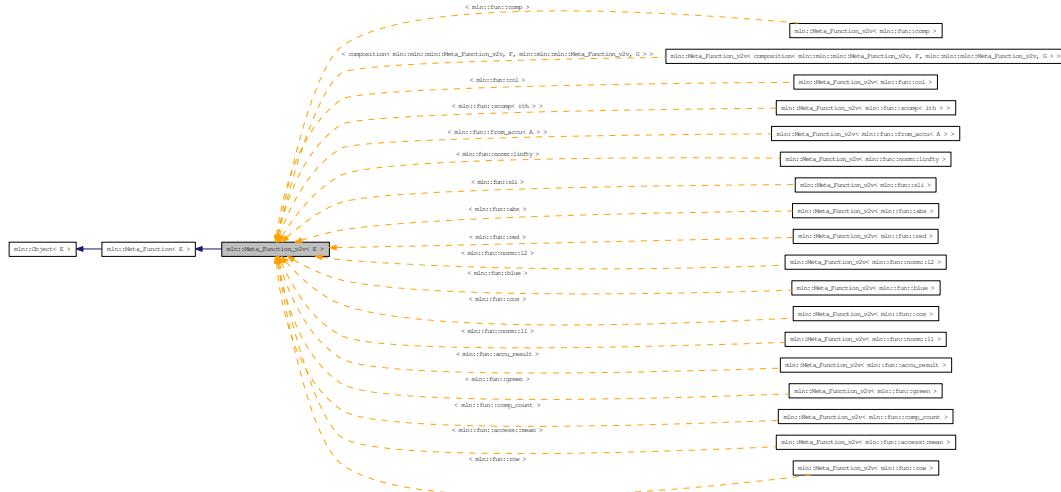
`mln::doc::Meta_Function` for a complete documentation of this class contents.

10.245 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 >:



10.245.1 Detailed Description

```
template<typename E> struct mln::Meta_Function_v2v< E >
```

Base class for implementation of function-objects from [value](#) to [value](#).

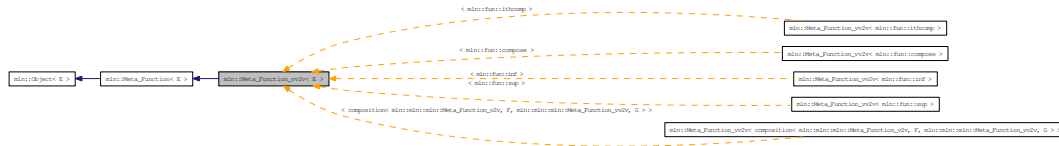
The parameter *E* is the exact type.

10.246 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 >:



10.246.1 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.

10.247 mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 > Struct Template Reference

Ands type.

```
#include <ands.hh>
```

10.247.1 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.

10.248 mln::metal::converts_to< T, U > Struct Template Reference

"converts-to" check.

```
#include <converts_to.hh>
```

Inherited by mln::metal::converts_to< T *, U * >.

10.248.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::converts_to< T, U >
```

"converts-to" check.

10.249 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 >:



10.249.1 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.

10.250 mln::metal::goes_to< T, U > Struct Template Reference

"goes-to" check.

```
#include <goes_to.hh>
```

10.250.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::goes_to< T, U >
```

"goes-to" check.

FIXME: Doc!

10.251 mln::metal::is< T, U > Struct Template Reference

"is" check.

```
#include <is.hh>
```

10.251.1 Detailed Description

template<typename T, typename U> struct mln::metal::is< T, U >

"is" check.

Check whether T inherits from U.

10.252 mln::metal::is_a< T, M > Struct Template Reference

"is_a" check.

```
#include <is_a.hh>
```

10.252.1 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.

10.253 mln::metal::is_not< T, U > Struct Template Reference

"is_not" check.

```
#include <is_not.hh>
```

10.253.1 Detailed Description

```
template<typename T, typename U> struct mln::metal::is_not< T, U >
```

"is_not" check.

FIXME: Doc!

10.254 mln::metal::is_not_a< T, M > Struct Template Reference

"is_not_a" static Boolean expression.

```
#include <is_not_a.hh>
```

10.254.1 Detailed Description

```
template<typename T, template< class > class M> struct mln::metal::is_not_a< T, M >
```

"is_not_a" static Boolean expression.

10.255 mln::mixed_neighb< W > Class Template Reference

Adapter class from [window](#) to neighborhood.

```
#include <mixed_neighb.hh>
```

Inherits mln::internal::neighb_base< W, mln::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](#) (const W &win)
Constructor from a [window](#) win.
- [mixed_neighb](#) ()
Constructor without argument.

10.255.1 Detailed Description

```
template<typename W> class mln::mixed_neighb< W >
```

Adapter class from [window](#) to neighborhood.

10.255.2 Member Typedef Documentation

10.255.2.1 `template<typename W> typedef mixed_neighb_bkd_niter<W> mln::mixed_neighb< W >::bkd_niter`

Backward site iterator associated type.

10.255.2.2 `template<typename W> typedef mixed_neighb_fwd_niter<W> mln::mixed_neighb< W >::fwd_niter`

Forward site iterator associated type.

10.255.2.3 `template<typename W> typedef fwd_niter mln::mixed_neighb< W >::niter`

[Site](#) iterator associated type.

10.255.3 Constructor & Destructor Documentation

10.255.3.1 `template<typename W> mln::mixed_neighb< W >::mixed_neighb () [inline]`

Constructor without argument.

10.255.3.2 `template<typename W> mln::mixed_neighb< W >::mixed_neighb (const W & win) [inline]`

Constructor from a [window win](#).

10.256 mln::morpho::attribute::card< I > Class Template Reference

Cardinality accumulator class.

```
#include <card.hh>
```

Inherits mln::accu::internal::base< unsigned, mln::morpho::attribute::card< I > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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.
- unsigned [to_result](#) () const
Get the [value](#) of the accumulator.
- void [init](#) ()
Manipulators.

10.256.1 Detailed Description

```
template<typename I> class mln::morpho::attribute::card< I >
```

Cardinality accumulator class.

10.256.2 Member Function Documentation

10.256.2.1 `template<typename I> void mln::morpho::attribute::card< I >::init ()` `[inline]`

Manipulators.

10.256.2.2 `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.

10.256.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.256.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.256.2.5 `template<typename I> unsigned mln::morpho::attribute::card< I >::to_result () const [inline]`

Get the [value](#) of the accumulator.

10.257 mln::morpho::attribute::count_adjacent_vertices< I > Struct Template Reference

Count_Adjacent_Vertices accumulator class.

```
#include <count_adjacent_vertices.hh>
```

Inherits mln::accu::internal::base< unsigned, mln::morpho::attribute::count_adjacent_vertices< I > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this [accu](#) is able to return a result.
- 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.
- unsigned [to_result](#) () const
Get the [value](#) of the accumulator.
- void [init](#) ()
Manipulators.

10.257.1 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.

10.257.2 Member Function Documentation

10.257.2.1 `template<typename I> void mln::morpho::attribute::count_adjacent_vertices< I >::init () [inline]`

Manipulators.

10.257.2.2 `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.

10.257.2.3 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References [mln::mln_exact\(\)](#).

10.257.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References [mln::mln_exact\(\)](#).

10.257.2.5 `template<typename I> unsigned mln::morpho::attribute::count_adjacent_vertices< I >::to_result () const [inline]`

Get the [value](#) of the accumulator.

10.258 mln::morpho::attribute::height< I > Struct Template Reference

Height accumulator class.

```
#include <height.hh>
```

Inherits mln::accu::internal::base< unsigned, mln::morpho::attribute::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.
- 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.
- unsigned [to_result](#) () const
Get the [value](#) of the accumulator.
- void [init](#) ()
Manipulators.

10.258.1 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.

10.258.2 Member Function Documentation

10.258.2.1 `template<typename I> unsigned mln::morpho::attribute::height< I >::base_level () const` `[inline]`

Get base & current level of the accumulator.

10.258.2.2 `template<typename I> void mln::morpho::attribute::height< I >::init ()` `[inline]`

Manipulators.

10.258.2.3 `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.

Referenced by `mln::morpho::attribute::height< I >::to_result()`.

10.258.2.4 `template<typename E> template<typename T> void mln::Accumulator< E`
`>::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in `mln::accu::stat::variance< T, S, R >`.

References `mln::mln_exact()`.

10.258.2.5 `template<typename E> template<typename T> void mln::Accumulator< E`
`>::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.258.2.6 `template<typename I> unsigned mln::morpho::attribute::height< I >::to_result ()`
`const [inline]`

Get the [value](#) of the accumulator.

References `mln::morpho::attribute::height< I >::is_valid()`.

10.259 mln::morpho::attribute::sharpness< I > Struct Template Reference

Sharpness accumulator class.

```
#include <sharpness.hh>
```

Inherits mln::accu::internal::base< double, mln::morpho::attribute::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.
- 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.
- double [to_result](#) () const
Get the [value](#) of the accumulator.
- unsigned [volume](#) () const
Give the [volume](#) of the component.
- void [init](#) ()
Manipulators.

10.259.1 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.

10.259.2 Member Function Documentation

10.259.2.1 `template<typename I> unsigned mln::morpho::attribute::sharpness< I >::area () const [inline]`

Give the area of the component.

10.259.2.2 `template<typename I> unsigned mln::morpho::attribute::sharpness< I >::height () const [inline]`

Give the [height](#).

10.259.2.3 `template<typename I> void mln::morpho::attribute::sharpness< I >::init () [inline]`

Manipulators.

10.259.2.4 `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.

10.259.2.5 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References `mln::mln_exact()`.

10.259.2.6 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.259.2.7 `template<typename I> double mln::morpho::attribute::sharpness< I >::to_result () const [inline]`

Get the [value](#) of the accumulator.

10.259.2.8 `template<typename I> unsigned mln::morpho::attribute::sharpness< I >::volume ()`
`const [inline]`

Give the [volume](#) of the component.

10.260 mln::morpho::attribute::sum< I, S > Class Template Reference

Suminality accumulator class.

```
#include <sum.hh>
```

Inherits mln::accu::internal::base< S, mln::morpho::attribute::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.
- 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.
- S [to_result](#) () const
Get the [value](#) of the accumulator.
- void [untake](#) (const argument &v)
Untake a [value](#) from the accumulator.
- void [init](#) ()
Manipulators.

10.260.1 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.

10.260.2 Member Function Documentation

10.260.2.1 `template<typename I, typename S> void mln::morpho::attribute::sum< I, S >::init ()`
[inline]

Manipulators.

References mln::literal::zero.

10.260.2.2 `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.

10.260.2.3 `template<typename I, typename S> void mln::morpho::attribute::sum< I, S >::set_value (const argument & v) [inline]`

Set the return [value](#) of the accumulator.

10.260.2.4 `template<typename E> template<typename T> void mln::Accumulator< E >::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References [mln::mln_exact\(\)](#).

10.260.2.5 `template<typename E> template<typename T> void mln::Accumulator< E >::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References [mln::mln_exact\(\)](#).

10.260.2.6 `template<typename I, typename S> S mln::morpho::attribute::sum< I, S >::to_result () const [inline]`

Get the [value](#) of the accumulator.

10.260.2.7 `template<typename I, typename S> void mln::morpho::attribute::sum< I, S >::untake (const argument & v) [inline]`

Untake a [value](#) from the accumulator.

10.261 mln::morpho::attribute::volume< I > Struct Template Reference

Volume accumulator class.

```
#include <volume.hh>
```

Inherits mln::accu::internal::base< unsigned, mln::morpho::attribute::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.
- 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.
- unsigned [to_result](#) () const
Get the [value](#) of the accumulator.
- void [init](#) ()
Manipulators.

10.261.1 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.

10.261.2 Member Function Documentation

10.261.2.1 template<typename I> unsigned mln::morpho::attribute::volume< I >::area () const
[inline]

Give the area.

10.261.2.2 `template<typename I> void mln::morpho::attribute::volume< I >::init ()`
[inline]

Manipulators.

10.261.2.3 `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.

10.261.2.4 `template<typename E> template<typename T> void mln::Accumulator< E`
`>::take_as_init (const T & t) [inline, inherited]`

Take as initialization the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Reimplemented in [mln::accu::stat::variance< T, S, R >](#).

References `mln::mln_exact()`.

10.261.2.5 `template<typename E> template<typename T> void mln::Accumulator< E`
`>::take_n_times (unsigned n, const T & t) [inline, inherited]`

Take `n` times the [value](#) `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

References `mln::mln_exact()`.

10.261.2.6 `template<typename I> unsigned mln::morpho::attribute::volume< I >::to_result ()`
`const [inline]`

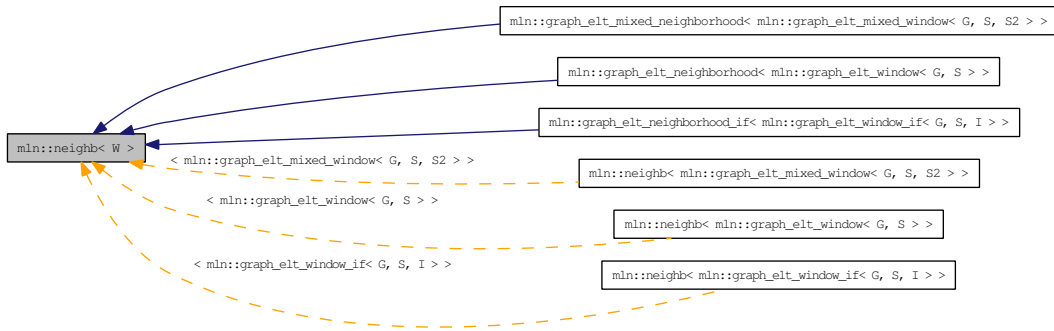
Get the [value](#) of the accumulator.

10.262 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` (const W &win)
Constructor from a [window](#) win.
- `neighb` ()
Constructor without argument.

10.262.1 Detailed Description

```
template<typename W> class mln::neighb< W >
```

Adapter class from [window](#) to neighborhood.

10.262.2 Member Typedef Documentation

10.262.2.1 `template<typename W> typedef neighb_bkd_niter<W> mln::neighb< W >::bkd_niter`

Backward site iterator associated type.

10.262.2.2 `template<typename W> typedef neighb_fwd_niter<W> mln::neighb< W >::fwd_niter`

Forward site iterator associated type.

10.262.2.3 `template<typename W> typedef fwd_niter mln::neighb< W >::niter`

[Site](#) iterator associated type.

10.262.3 Constructor & Destructor Documentation

10.262.3.1 `template<typename W> mln::neighb< W >::neighb () [inline]`

Constructor without argument.

10.262.3.2 `template<typename W> mln::neighb< W >::neighb (const W & win) [inline]`

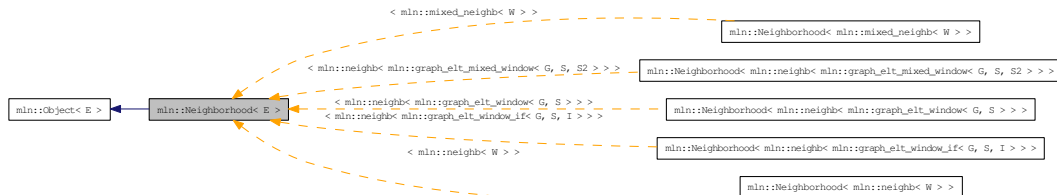
Constructor from a [window win](#).

10.263 mln::Neighborhood< E > Struct Template Reference

Base class for implementation classes that are neighborhoods.

```
#include <neighborhood.hh>
```

Inheritance diagram for mln::Neighborhood< E >:



10.263.1 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.

10.264 mln::Neighborhood< void > Struct Template Reference

[Neighborhood](#) category flag type.

```
#include <neighborhood.hh>
```

10.264.1 Detailed Description

```
template<> struct mln::Neighborhood< void >
```

[Neighborhood](#) category flag type.

10.265 mln::Object< E > Struct Template Reference

Base class for almost every class defined in Milena.

```
#include <object.hh>
```

Inherited by [mln::Function< function< meta::blue< mln::value::mln::value::rgb::mln::value::mln::value::rgb< n > > > >](#), [mln::Function< function< meta::green< mln::value::mln::value::rgb::mln::value::mln::value::rgb< n > > > >](#), [mln::Function< function< meta::hue< mln::value::mln::value::hsi_::mln::value::mln::value::hsi_< H, S, I > > > >](#), [mln::Function< function< meta::hue< mln::value::mln::value::hsl_::mln::value::mln::value::hsl_< H, S, L > > > >](#), [mln::Function< function< meta::inty< mln::value::mln::value::hsi_::mln::value::mln::value::hsi_< H, S, I > > > >](#), [mln::Function< function< meta::lum< mln::value::mln::value::hsl_::mln::value::mln::value::hsl_< H, S, I > > > >](#), [mln::Function< function< meta::red< mln::value::mln::value::rgb::mln::value::mln::value::rgb< n > > > >](#), [mln::Function< function< meta::sat< mln::value::mln::value::hsi_::mln::value::mln::value::hsi_< H, S, I > > > >](#), [mln::Function< function< meta::sat< mln::value::mln::value::hsl_::mln::value::mln::value::hsl_< H, S, L > > > >](#), [mln::algebra::mat< d+1, d+1, T >](#), [mln::Meta_Function< composition< mln::mln::mln::mln::Meta_Function_v2v, F, mln::mln::mln::mln::Meta_Function_v2v, G > >](#), [mln::Meta_Function< composition< mln::mln::mln::mln::Meta_Function_vv2v, F, mln::mln::mln::mln::Meta_Function_vv2v, G > >](#), [mln::algebra::internal::vec_base_< n, T >](#), [mln::algebra::internal::vec_base_< 1, T >](#), [mln::algebra::internal::vec_base_< 2, T >](#), [mln::algebra::internal::vec_base_< 3, T >](#), [mln::algebra::internal::vec_base_< 4, T >](#), [mln::algebra::mat< n, m, T >](#), [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::metal::array1d< T, Size >](#), [mln::metal::array2d< T, r, c >](#), [mln::metal::array3d< T, s, r, c >](#), [mln::metal::internal::vec_base_< n, T >](#), [mln::metal::internal::vec_base_< 1, T >](#), [mln::metal::internal::vec_base_< 2, T >](#), [mln::metal::internal::vec_base_< 3, T >](#), [mln::metal::internal::vec_base_< 4, T >](#), [mln::metal::mat< n, m, T >](#), [mln::Neighborhood< E >](#), [mln::pixel< I >](#), [mln::Point_Site< E >](#), [mln::Proxy< E >](#), [mln::Site< E >](#), [mln::Site_Set< E >](#), [mln::util::couple< T, U >](#), [mln::util::eat](#), [mln::util::fibonacci_heap< P, T >](#), [mln::util::ignore](#), [mln::util::lemmings_< I >](#), [mln::util::multi_site< P >](#), [mln::util::nil](#), [mln::util::ord_pair< T >](#), [mln::util::site_pair< P >](#), [mln::util::soft_heap< T, R >](#), [mln::util::yes](#), [mln::Value< E >](#), [mln::value::HSL< E >](#), [mln::value::interval_< T >](#), [mln::Value_Set< E >](#), [mln::Weighted_Window< E >](#), [mln::Window< E >](#), [test< T >](#), and [mln::algebra::internal::vec_base_< n, C >](#).

10.265.1 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.

10.266 mln::p2p_image< I, F > Struct Template Reference

FIXME: Doc!

```
#include <p2p_image.hh>
```

Inherits mln::internal::image_domain_morpher< I, I::domain_t, mln::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.
- internal::morpher_lvalue_< I >::ret [operator](#)() (const typename I::psite &p)
Read-write access to the image [value](#) located at [point](#) p.
- I::rvalue [operator](#)() (const typename I::psite &p) const
Read-only 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.

10.266.1 Detailed Description

```
template<typename I, typename F> struct mln::p2p_image< I, F >
```

FIXME: Doc!

10.266.2 Member Typedef Documentation

10.266.2.1 `template<typename I, typename F> typedef p2p_image< tag::image_<I>, tag::function_<F> > mln::p2p_image< I, F >::skeleton`

Skeleton.

10.266.3 Constructor & Destructor Documentation

10.266.3.1 `template<typename I, typename F> mln::p2p_image< I, F >::p2p_image ()`
`[inline]`

Constructor without argument.

10.266.3.2 `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*.

10.266.4 Member Function Documentation

10.266.4.1 `template<typename I, typename F> const I::domain_t & mln::p2p_image< I, F
>::domain () const` `[inline]`

Give the definition domain.

10.266.4.2 `template<typename I, typename F> const F & mln::p2p_image< I, F >::fun () const`
`[inline]`

Give the p2p function.

10.266.4.3 `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*.

10.266.4.4 `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*.

10.267 mln::p_array< P > Class Template Reference

Multi-set of sites.

```
#include <p_array.hh>
```

Inherits mln::internal::site_set_base_< P, mln::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_array](#)< P > &other)
Append an array other of points.
- [p_array](#)< P > & [append](#) (const P &p)
Append a point p.
- 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 (const std::vector< P > &vect)`
Constructor from a vector `vect`.
- `p_array ()`
Constructor.
- `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.

Related Functions

(Note that these are not member functions.)

- `template<typename Sl, typename Sr>`
`p_set< typename Sl::site > diff (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic difference of `lhs` and `rhs`.
- `template<typename Sl, typename Sr>`
`p_set< typename Sl::site > inter (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Intersection between a couple of [point sets](#).
- `template<typename Sl, typename Sr>`
`bool operator< (const Site_Set< Sl > &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 Sl, typename Sr>`
`bool operator<= (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Inclusion test between site sets lhs and rhs.
- `template<typename Sl, typename Sr>`
`bool operator== (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Equality test between site sets lhs and rhs.
- `template<typename Sl, typename Sr>`
`p_set< typename Sl::site > sym_diff (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic symmetrical difference of lhs and rhs.
- `template<typename Sl, typename Sr>`
`p_set< typename Sl::site > uni (const Site_Set< Sl > &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.

10.267.1 Detailed Description

`template<typename P> class mln::p_array< P >`

Multi-set of sites.

[Site set](#) class based on `std::vector`.

10.267.2 Member Typedef Documentation

10.267.2.1 `template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_array< P >::bkd_piter`

Backward [Site_Iterator](#) associated type.

10.267.2.2 `template<typename P> typedef P mln::p_array< P >::element`

Element associated type.

10.267.2.3 `template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_array< P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

10.267.2.4 `template<typename P> typedef P mln::p_array< P >::i_element`

Insertion element associated type.

10.267.2.5 `template<typename P> typedef fwd_piter mln::p_array< P >::piter`

[Site_Iterator](#) associated type.

10.267.2.6 `template<typename P> typedef p_indexed_psite<self_> mln::p_array< P >::psite`

Psite associated type.

10.267.3 **Constructor & Destructor Documentation****10.267.3.1** `template<typename P> mln::p_array< P >::p_array () [inline]`

Constructor.

10.267.3.2 `template<typename P> mln::p_array< P >::p_array (const std::vector< P > & vect) [inline]`

Constructor from a vector `vect`.

10.267.4 **Member Function Documentation****10.267.4.1** `template<typename P> p_array< P > & mln::p_array< P >::append (const p_array< P > & other) [inline]`

Append an array `other` of points.

References `mln::p_array< P >::std_vector()`.

10.267.4.2 `template<typename P> p_array< P > & mln::p_array< P >::append (const P & p) [inline]`

Append a [point](#) `p`.

Referenced by `mln::convert::to_p_array()`.

10.267.4.3 `template<typename P> void mln::p_array< P >::change (const psite & p, const P & new_p) [inline]`

Change site `p` into `new_p`.

References `mln::p_array< P >::has()`, and `mln::p_indexed_psite< S >::index()`.

10.267.4.4 `template<typename P> void mln::p_array< P >::clear () [inline]`

Clear this [set](#).

10.267.4.5 `template<typename P> bool mln::p_array< P >::has (const util::index & i) const`
`[inline]`

Test is index *i* belongs to this site [set](#).

References `mln::p_array< P >::nsites()`.

10.267.4.6 `template<typename P> bool mln::p_array< P >::has (const psite & p) const`
`[inline]`

Test is *p* belongs to this site [set](#).

References `mln::p_indexed_psite< S >::index()`.

Referenced by `mln::p_array< P >::change()`, and `mln::p_array< P >::operator[]()`.

10.267.4.7 `template<typename P> void mln::p_array< P >::insert (const P & p) [inline]`

Insert a [point](#) *p* (equivalent as 'append').

10.267.4.8 `template<typename P> bool mln::p_array< P >::is_valid () const [inline]`

Test this [set](#) validity so returns always true.

10.267.4.9 `template<typename P> std::size_t mln::p_array< P >::memory_size () const`
`[inline]`

Return the size of this site [set](#) in memory.

References `mln::p_array< P >::nsites()`.

10.267.4.10 `template<typename P> unsigned mln::p_array< P >::nsites () const [inline]`

Give the number of sites.

Referenced by `mln::registration::get_rot()`, `mln::p_array< P >::has()`, `mln::p_array< P >::memory_size()`, and `mln::p_array< P >::operator[]()`.

10.267.4.11 `]`

`template<typename P> const P & mln::p_array< P >::operator[] (const util::index & i) const`
`[inline]`

Return the *i*-th element.

References `mln::p_array< P >::has()`.

10.267.4.12 `]`

`template<typename P> P & mln::p_array< P >::operator[] (unsigned i) [inline]`

Return the *i*-th site (mutable).

References `mln::p_array< P >::nsites()`.

10.267.4.13]

```
template<typename P> const P & mln::p_array< P >::operator[] (unsigned i) const [inline]
```

Return the *i*-th site (constant).

References `mln::p_array< P >::nsites()`.

10.267.4.14 `template<typename P> void mln::p_array< P >::reserve (size_type n) [inline]`

Reserve *n* cells.

Referenced by `mln::convert::to_p_array()`.

10.267.4.15 `template<typename P> void mln::p_array< P >::resize (size_t size) [inline]`

Update the size of this array.

10.267.4.16 `template<typename P> const std::vector< P > & mln::p_array< P >::std_vector () const [inline]`

Return the corresponding `std::vector` of points.

Referenced by `mln::p_array< P >::append()`.

10.267.5 Friends And Related Function Documentation**10.267.5.1** `template<typename Sl, typename Sr> p_set< typename Sl::site > diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Set theoretic difference of *lhs* and *rhs*.

10.267.5.2 `template<typename Sl, typename Sr> p_set< typename Sl::site > inter (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Intersection between a couple of [point](#) sets.

10.267.5.3 `template<typename Sl, typename Sr> bool operator< (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Strict inclusion [test](#) between site sets *lhs* and *rhs*.

Parameters:

← *lhs* A site [set](#) (strictly included?).

← *rhs* Another site [set](#) (includer?).

10.267.5.4 `template<typename S> std::ostream & operator<< (std::ostream & ostr, const Site_Set< S > & set)` [related, inherited]

Print a site `set` into the output stream `ostr`.

Parameters:

↔ `ostr` An output stream.

← `set` A site `set`.

Returns:

The modified output stream `ostr`.

10.267.5.5 `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:

← `lhs` A site `set` (included?).

← `rhs` Another site `set` (includer?).

10.267.5.6 `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:

← `lhs` A site `set`.

← `rhs` Another site `set`.

10.267.5.7 `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`.

10.267.5.8 `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.

10.267.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [related, inherited]

Give the unique `set` of `s`.

10.268 mln::p_centered< W > Class Template Reference

[Site set](#) corresponding to a [window](#) centered on a site.

```
#include <p_centered.hh>
```

Inherits mln::internal::site_set_base_< W::psite, mln::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.

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.

10.268.1 Detailed Description

template<typename W> class mln::p_centered< W >

[Site set](#) corresponding to a [window](#) centered on a site.

10.268.2 Member Typedef Documentation

10.268.2.1 `template<typename W> typedef p_centered_piter<W> mln::p_centered< W >::bkd_piter`

Backward [Site_Iterator](#) associated type.

10.268.2.2 `template<typename W> typedef psite mln::p_centered< W >::element`

Element associated type.

10.268.2.3 `template<typename W> typedef p_centered_piter<W> mln::p_centered< W >::fwd_piter`

Forward [Site_Iterator](#) associated type.

10.268.2.4 `template<typename W> typedef fwd_piter mln::p_centered< W >::piter`

[Site_Iterator](#) associated type.

10.268.2.5 `template<typename W> typedef W ::psite mln::p_centered< W >::psite`

Psite associated type.

10.268.2.6 `template<typename W> typedef W ::site mln::p_centered< W >::site`

[Site](#) associated type.

10.268.3 Constructor & Destructor Documentation

10.268.3.1 `template<typename W> mln::p_centered< W >::p_centered () [inline]`

Constructor without argument.

10.268.3.2 `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*.

References `mln::p_centered< W >::is_valid()`.

10.268.4 Member Function Documentation

10.268.4.1 `template<typename W> const W::psite & mln::p_centered< W >::center () const [inline]`

Give the center of this site [set](#).

10.268.4.2 `template<typename W> template<typename P> bool mln::p_centered< W >::has(const P & p) const` [inline]

Test if `p` belongs to the `box`.

References `mln::p_centered< W >::is_valid()`.

10.268.4.3 `template<typename W> bool mln::p_centered< W >::is_valid () const` [inline]

Test if this site `set` is initialized.

Referenced by `mln::p_centered< W >::has()`, and `mln::p_centered< W >::p_centered()`.

10.268.4.4 `template<typename W> std::size_t mln::p_centered< W >::memory_size () const` [inline]

Return the size of this site `set` in memory.

10.268.4.5 `template<typename W> const W & mln::p_centered< W >::window () const` [inline]

Give the `window` this site `set` is defined upon.

10.268.5 Friends And Related Function Documentation

10.268.5.1 `template<typename Sl, typename Sr> p_set< typename Sl::site > diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Set theoretic difference of `lhs` and `rhs`.

10.268.5.2 `template<typename Sl, typename Sr> p_set< typename Sl::site > inter (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Intersection between a couple of `point` sets.

10.268.5.3 `template<typename Sl, typename Sr> bool operator< (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Strict inclusion `test` between site sets `lhs` and `rhs`.

Parameters:

← *lhs* A site `set` (strictly included?).

← *rhs* Another site `set` (includer?).

10.268.5.4 `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:

- ↔ *ostr* An output stream.
- ← *set* A site [set](#).

Returns:

The modified output stream `ostr`.

10.268.5.5 `template<typename SI, typename Sr> bool operator<= (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [[related](#), [inherited](#)]

Inclusion [test](#) between site sets `lhs` and `rhs`.

Parameters:

- ← *lhs* A site [set](#) (included?).
- ← *rhs* Another site [set](#) (includer?).

10.268.5.6 `template<typename SI, typename Sr> bool operator== (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [[related](#), [inherited](#)]

Equality [test](#) between site sets `lhs` and `rhs`.

Parameters:

- ← *lhs* A site [set](#).
- ← *rhs* Another site [set](#).

10.268.5.7 `template<typename SI, typename Sr> p_set< typename SI::site > sym_diff (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [[related](#), [inherited](#)]

Set theoretic symmetrical difference of `lhs` and `rhs`.

10.268.5.8 `template<typename SI, typename Sr> p_set< typename SI::site > uni (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [[related](#), [inherited](#)]

Union of a couple of [point](#) sets.

10.268.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [[related](#), [inherited](#)]

Give the unique [set](#) of `s`.

10.269 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 mln::internal::site_set_base_< mln::complex_psite< D, G >, mln::p_complex< D, G > >.

Public Types

- typedef p_complex_bkd_piter_< D, G > [bkd_piter](#)
Backward [Site_Iterator](#) associated type.
- typedef super_::site [element](#)
Associated types.
- typedef p_complex_fwd_piter_< D, G > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef [fwd_piter](#) [piter](#)
[Site_Iterator](#) associated type.
- typedef [complex_psite](#)< D, G > [psite](#)
[Point_Site](#) 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](#) ()
Return the complex associated to the [p_complex](#) domain (mutable version).
- [topo::complex](#)< D > & [cplx](#) () const
Accessors.

- `const G & geom () const`
Return the geometry of the complex.

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.

10.269.1 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.

10.269.2 Member Typedef Documentation

10.269.2.1 `template<unsigned D, typename G> typedef p_complex_bkd_piter_<D, G>
mln::p_complex< D, G >::bkd_piter`

Backward [Site_Iterator](#) associated type.

10.269.2.2 `template<unsigned D, typename G> typedef super_::site mln::p_complex< D, G
>::element`

Associated types.

Element associated type.

10.269.2.3 `template<unsigned D, typename G> typedef p_complex_fwd_piter_<D, G>
mln::p_complex< D, G >::fwd_piter`

Forward [Site_Iterator](#) associated type.

10.269.2.4 `template<unsigned D, typename G> typedef fwd_piter mln::p_complex< D, G
>::piter`

[Site_Iterator](#) associated type.

10.269.2.5 `template<unsigned D, typename G> typedef complex_psite<D, G> mln::p_complex<
D, G >::psite`

[Point_Site](#) associated type.

10.269.3 Constructor & Destructor Documentation

10.269.3.1 `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

10.269.4 Member Function Documentation

10.269.4.1 `template<unsigned D, typename G> topo::complex< D > & mln::p_complex< D, G >::cplx () [inline]`

Return the complex associated to the [p_complex](#) domain (mutable version).

References `mln::p_complex< D, G >::is_valid()`.

10.269.4.2 `template<unsigned D, typename G> topo::complex< D > & mln::p_complex< D, G >::cplx () const [inline]`

Accessors.

Return the complex associated to the [p_complex](#) domain (const version)

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==(())`.

10.269.4.3 `template<unsigned D, typename G> const G & mln::p_complex< D, G >::geom () const [inline]`

Return the geometry of the complex.

10.269.4.4 `template<unsigned D, typename G> bool mln::p_complex< D, G >::has (const psite & p) const [inline]`

Does this site [set](#) has *p*?

References `mln::complex_psite< D, G >::is_valid()`, `mln::p_complex< D, G >::is_valid()`, and `mln::complex_psite< D, G >::site_set()`.

10.269.4.5 `template<unsigned D, typename G> bool mln::p_complex< D, G >::is_valid () const [inline]`

Is this site [set](#) valid?

Referenced by `mln::p_complex< D, G >::cplx()`, and `mln::p_complex< D, G >::has()`.

10.269.4.6 `template<unsigned D, typename G> unsigned mln::p_complex< D, G >::nfaces () const [inline]`

Return the number of faces in the complex.

Referenced by `mln::p_complex< D, G >::nsites()`.

10.269.4.7 `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.

10.269.4.8 `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'.)

References `mln::p_complex< D, G >::nfaces()`.

10.269.5 Friends And Related Function Documentation

10.269.5.1 `template<typename Sl, typename Sr> p_set< typename Sl::site > diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Set theoretic difference of `lhs` and `rhs`.

10.269.5.2 `template<typename Sl, typename Sr> p_set< typename Sl::site > inter (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Intersection between a couple of `point` sets.

10.269.5.3 `template<typename Sl, typename Sr> bool operator< (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Strict inclusion `test` between site sets `lhs` and `rhs`.

Parameters:

← *lhs* A site `set` (strictly included?).

← *rhs* Another site `set` (includer?).

10.269.5.4 `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:

↔ *ostr* An output stream.

← *set* A site `set`.

Returns:

The modified output stream `ostr`.

10.269.5.5 `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:

- ← *lhs* A site [set](#) (included?).
- ← *rhs* Another site [set](#) (included?).

10.269.5.6 `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:

- ← *lhs* A site [set](#).
- ← *rhs* Another site [set](#).

10.269.5.7 `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`.

10.269.5.8 `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.

10.269.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [[related](#), [inherited](#)]

Give the unique [set](#) of `s`.

10.270 mln::p_edges< G, F > Class Template Reference

Site set mapping [graph](#) edges and image sites.

```
#include <p_edges.hh>
```

Inherits mln::internal::site_set_base_< F::result, mln::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 [p_graph_piter](#)< [self_](#), [mln_edge_bkd_iter](#)(G) > [bkd_piter](#)
Backward [Site_Iterator](#) associated type.
- typedef [super_::site](#) [element](#)
Associated types.
- typedef [p_graph_piter](#)< [self_](#), [mln_edge_fwd_iter](#)(G) > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef [fwd_piter](#) [piter](#)
[Site_Iterator](#) associated type.
- typedef [p_edges_psite](#)< G, F > [psite](#)
[Point_Site](#) associated type.

Public Member Functions

- template<typename G2>
bool [has](#) (const [util::edge](#)< G2 > &e) const
Does this site [set](#) has edge e?
- bool [has](#) (const [psite](#) &p) const
Does this site [set](#) has site p?
- 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.
- `const F & function () const`
Return the mapping function.
- `const G & graph () const`
Accessors.
- `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.
- `p_edges (const Graph< G > &gr, const Function< F > &f)`
Construct a [graph](#) edge psite [set](#) from a [graph](#) and a function.
- `p_edges (const Graph< G > &gr)`
Construct a [graph](#) edge psite [set](#) from a [graph](#).
- `p_edges ()`
Constructors
Default constructor.

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` into the output stream `ostr`.
- `template<typename Sl, typename Sr>`
`bool operator<= (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Inclusion test between site sets `lhs` and `rhs`.
- `template<typename Sl, typename Sr>`
`bool operator== (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Equality test between site sets `lhs` and `rhs`.
- `template<typename Sl, typename Sr>`
`p_set< typename Sl::site > sym_diff (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic symmetrical difference of `lhs` and `rhs`.
- `template<typename Sl, typename Sr>`
`p_set< typename Sl::site > uni (const Site_Set< Sl > &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`.

10.270.1 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.

10.270.2 Member Typedef Documentation

10.270.2.1 `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.

10.270.2.2 `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.

10.270.2.3 `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.

10.270.2.4 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>>`
`typedef F mln::p_edges< G, F >::fun_t`

Function associated type.

10.270.2.5 `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.

10.270.2.6 `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.

10.270.2.7 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>>`
`typedef G mln::p_edges< G, F >::graph_t`

[Graph](#) associated type.

10.270.2.8 `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.

10.270.2.9 `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.

10.270.3 Constructor & Destructor Documentation

10.270.3.1 `template<typename G, typename F> mln::p_edges< G, F >::p_edges () [inline]`

Constructors

Default constructor.

10.270.3.2 `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.

References `mln::p_edges< G, F >::is_valid()`.

10.270.3.3 `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.

References `mln::p_edges< G, F >::is_valid()`.

10.270.3.4 `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`.

References `mln::p_edges< G, F >::is_valid()`.

10.270.4 Member Function Documentation

10.270.4.1 `template<typename G, typename F> const F & mln::p_edges< G, F >::function () const [inline]`

Return the mapping function.

10.270.4.2 `template<typename G, typename F> const G & mln::p_edges< G, F >::graph () const [inline]`

Accessors.

Return the [graph](#) associated to this site [set](#)

References `mln::p_edges< G, F >::is_valid()`.

Referenced by `mln::operator==(())`.

10.270.4.3 `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*?

References `mln::util::edge< G >::graph()`, `mln::util::edge< G >::is_valid()`, and `mln::p_edges< G, F >::is_valid()`.

10.270.4.4 `template<typename G, typename F> bool mln::p_edges< G, F >::has (const psite & p) const [inline]`

Does this site [set](#) has site *p*?

References `mln::p_edges< G, F >::is_valid()`.

10.270.4.5 `template<typename G, typename F> void mln::p_edges< G, F >::invalidate () [inline]`

Invalidate this site [set](#).

10.270.4.6 `template<typename G, typename F> bool mln::p_edges< G, F >::is_valid () const [inline]`

Is this site [set](#) valid?

Referenced by `mln::p_edges< G, F >::graph()`, `mln::p_edges< G, F >::has()`, and `mln::p_edges< G, F >::p_edges()`.

10.270.4.7 `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;`

10.270.4.8 `template<typename G, typename F> unsigned mln::p_edges< G, F >::nedges () const [inline]`

Return The number of edges in the [graph](#).

Referenced by `mln::p_edges< G, F >::nsites()`.

10.270.4.9 `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*.

References `mln::p_edges< G, F >::nedges()`.

10.270.5 Friends And Related Function Documentation

10.270.5.1 `template<typename Sl, typename Sr> p_set< typename Sl::site > diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Set theoretic difference of `lhs` and `rhs`.

10.270.5.2 `template<typename Sl, typename Sr> p_set< typename Sl::site > inter (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Intersection between a couple of [point](#) sets.

10.270.5.3 `template<typename Sl, typename Sr> bool operator< (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Strict inclusion [test](#) between site sets `lhs` and `rhs`.

Parameters:

- ← *lhs* A site [set](#) (strictly included?).
- ← *rhs* Another site [set](#) (includer?).

10.270.5.4 `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:

- ↔ *ostr* An output stream.
- ← *set* A site [set](#).

Returns:

The modified output stream `ostr`.

10.270.5.5 `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:

- ← *lhs* A site [set](#) (included?).
- ← *rhs* Another site [set](#) (includer?).

10.270.5.6 `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:

- ← *lhs* A site [set](#).
- ← *rhs* Another site [set](#).

10.270.5.7 `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`.

10.270.5.8 `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.

10.270.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [related, inherited]

Give the unique [set](#) of s.

10.271 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 mln::internal::site_set_base_< mln::faces_psite< N, D, P >, mln::p_faces< N, D, P > >.

Package Types

- typedef p_faces_bkd_piter_< N, D, P > [bkd_piter](#)
Backward [Site_Iterator](#) associated type.
- typedef super_::site [element](#)
Associated types.
- typedef p_faces_fwd_piter_< N, D, P > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef [fwd_piter](#) [piter](#)
[Site_Iterator](#) associated type.
- typedef [faces_psite](#)< N, D, P > [psite](#)
[Point_Site](#) associated type.

Package 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 [p_complex](#)< D, P > &pc)
Construct a faces psite [set](#) from an [mln::p_complex](#).
- [p_faces](#) (const [topo::complex](#)< D > &cplx)
Construct a faces psite [set](#) from an [mln::complex](#).
- [topo::complex](#)< D > & [cplx](#) ()
Return the complex associated to the [p_faces](#) domain (mutable version).
- [topo::complex](#)< D > & [cplx](#) () const
Accessors.

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 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.

10.271.1 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).

10.271.2 Member Typedef Documentation

10.271.2.1 `template<unsigned N, unsigned D, typename P> typedef p_faces_bkd_piter_<N, D, P> mln::p_faces< N, D, P >::bkd_piter` [package]

Backward [Site_Iterator](#) associated type.

10.271.2.2 `template<unsigned N, unsigned D, typename P> typedef super_::site mln::p_faces< N, D, P >::element` [package]

Associated types.

Element associated type.

10.271.2.3 `template<unsigned N, unsigned D, typename P> typedef p_faces_fwd_piter_<N, D, P> mln::p_faces< N, D, P >::fwd_piter` [package]

Forward [Site_Iterator](#) associated type.

10.271.2.4 `template<unsigned N, unsigned D, typename P> typedef fwd_piter mln::p_faces< N, D, P >::piter` [package]

[Site_Iterator](#) associated type.

10.271.2.5 `template<unsigned N, unsigned D, typename P> typedef faces_psite<N, D, P> mln::p_faces< N, D, P >::psite` [package]

[Point_Site](#) associated type.

10.271.3 Constructor & Destructor Documentation

10.271.3.1 `template<unsigned N, unsigned D, typename P> mln::p_faces< N, D, P >::p_faces (const topo::complex< D > & cplx)` [inline, package]

Construct a faces psite [set](#) from an mln::complex.

Parameters:

cplx The complex upon which the complex psite [set](#) is built.

10.271.3.2 `template<unsigned N, unsigned D, typename P> mln::p_faces< N, D, P >::p_faces (const p_complex< D, P > & pc)` [inline, package]

Construct a faces psite [set](#) from an mln::p_complex.

Parameters:

pc The complex upon which the complex psite [set](#) is built.

10.271.4 Member Function Documentation

10.271.4.1 `template<unsigned N, unsigned D, typename P> topo::complex< D > & mln::p_faces< N, D, P >::cplx ()` [inline, package]

Return the complex associated to the [p_faces](#) domain (mutable version).

References `mln::p_faces< N, D, P >::is_valid()`.

10.271.4.2 `template<unsigned N, unsigned D, typename P> topo::complex< D > & mln::p_faces< N, D, P >::cplx () const` [inline, package]

Accessors.

Return the complex associated to the [p_faces](#) domain (const version).

References `mln::p_faces< N, D, P >::is_valid()`.

Referenced by `mln::faces_psite< N, D, P >::change_target()`, and `mln::operator==()`.

10.271.4.3 `template<unsigned N, unsigned D, typename P> bool mln::p_faces< N, D, P >::is_valid () const` [inline, package]

Is this site [set](#) valid?

Referenced by `mln::p_faces< N, D, P >::cplx()`.

10.271.4.4 `template<unsigned N, unsigned D, typename P> unsigned mln::p_faces< N, D, P >::nfaces () const` [inline, package]

Return The number of faces in the complex.

Referenced by `mln::p_faces< N, D, P >::nsites()`.

10.271.4.5 `template<unsigned N, unsigned D, typename P> unsigned mln::p_faces< N, D, P >::nsites () const` [inline, package]

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'.)

References `mln::p_faces< N, D, P >::nfaces()`.

10.271.5 Friends And Related Function Documentation

10.271.5.1 `template<typename Sl, typename Sr> p_set< typename Sl::site > diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Set theoretic difference of lhs and rhs.

10.271.5.2 `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.

10.271.5.3 `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:

← *lhs* A site [set](#) (strictly included?).

← *rhs* Another site [set](#) (includer?).

10.271.5.4 `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:

↔ *ostr* An output stream.

← *set* A site [set](#).

Returns:

The modified output stream `ostr`.

10.271.5.5 `template<typename SI, typename Sr> bool operator<= (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Inclusion [test](#) between site sets lhs and rhs.

Parameters:

← *lhs* A site [set](#) (included?).

← *rhs* Another site [set](#) (includer?).

10.271.5.6 `template<typename SI, typename Sr> bool operator== (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Equality [test](#) between site sets lhs and rhs.

Parameters:

← *lhs* A site [set](#).

← *rhs* Another site [set](#).

10.271.5.7 `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.

10.271.5.8 `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.

10.271.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [related, inherited]

Give the unique [set](#) of s.

10.272 mln::p_graph_piter< S, I > Class Template Reference

Generic iterator on [point](#) sites of a mln::S.

```
#include <p_graph_piter.hh>
```

Inherits mln::internal::site_set_iterator_base< S, mln::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.

10.272.1 Detailed Description

```
template<typename S, typename I> class mln::p_graph_piter< S, I >
```

Generic iterator on [point](#) sites of a mln::S.

10.272.2 Constructor & Destructor Documentation

10.272.2.1 `template<typename S, typename I> mln::p_graph_piter< S, I >::p_graph_piter ()`
[inline]

Constructors.

10.272.3 Member Function Documentation

10.272.3.1 `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.

10.272.3.2 `template<typename S, typename I> unsigned mln::p_graph_piter< S, I >::id () const`
[inline]

Return the [graph](#) element id.

10.272.3.3 `template<typename S, typename I> mln::p_graph_piter< S, I >::mln_q_subject (iter)`

Return the underlying [graph](#) element.

10.272.3.4 `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.

10.273 mln::p_if< S, F > Class Template Reference

[Site set](#) restricted w.r.t.

```
#include <p_if.hh>
```

Inherits mln::internal::site_set_base_< S::psite, mln::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.

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.*

10.273.1 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.

10.273.2 Member Typedef Documentation

10.273.2.1 `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.

10.273.2.2 `template<typename S, typename F> typedef S ::element mln::p_if< S, F >::element`

Element associated type.

10.273.2.3 `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.

10.273.2.4 `template<typename S, typename F> typedef fwd_piter mln::p_if< S, F >::piter`

[Site_Iterator](#) associated type.

10.273.2.5 `template<typename S, typename F> typedef S ::psite mln::p_if< S, F >::psite`

Psite associated type.

10.273.3 Constructor & Destructor Documentation

10.273.3.1 `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*.

10.273.3.2 `template<typename S, typename F> mln::p_if< S, F >::p_if () [inline]`

Constructor without argument.

10.273.4 Member Function Documentation

10.273.4.1 `template<typename S, typename F> bool mln::p_if< S, F >::has (const psite & p) const [inline]`

Test if *p* belongs to the subset.

References `mln::p_if< S, F >::has()`.

Referenced by `mln::p_if< S, F >::has()`.

10.273.4.2 `template<typename S, typename F> bool mln::p_if< S, F >::is_valid () const`
`[inline]`

Test if this site [set](#) is valid.

10.273.4.3 `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.

10.273.4.4 `template<typename S, typename F> const S & mln::p_if< S, F >::overset () const`
`[inline]`

Give the primary overset.

10.273.4.5 `template<typename S, typename F> bool mln::p_if< S, F >::pred (const psite & p)`
`const [inline]`

Test predicate on [point](#) site *p*.

10.273.4.6 `template<typename S, typename F> const F & mln::p_if< S, F >::predicate () const`
`[inline]`

Give the predicate function.

10.273.5 Friends And Related Function Documentation

10.273.5.1 `template<typename Sl, typename Sr> p_set< typename Sl::site > diff (const Site_Set<`
`Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Set theoretic difference of *lhs* and *rhs*.

10.273.5.2 `template<typename Sl, typename Sr> p_set< typename Sl::site > inter (const`
`Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Intersection between a couple of [point](#) sets.

10.273.5.3 `template<typename Sl, typename Sr> bool operator< (const Site_Set< Sl > & lhs,`
`const Site_Set< Sr > & rhs) [related, inherited]`

Strict inclusion [test](#) between site sets *lhs* and *rhs*.

Parameters:

← *lhs* A site [set](#) (strictly included?).

← *rhs* Another site [set](#) (includer?).

10.273.5.4 `template<typename S> std::ostream & operator<< (std::ostream & ostr, const Site_Set< S > & set)` [related, inherited]

Print a site `set` into the output stream `ostr`.

Parameters:

↔ *ostr* An output stream.

← *set* A site `set`.

Returns:

The modified output stream `ostr`.

10.273.5.5 `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:

← *lhs* A site `set` (included?).

← *rhs* Another site `set` (includer?).

10.273.5.6 `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:

← *lhs* A site `set`.

← *rhs* Another site `set`.

10.273.5.7 `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`.

10.273.5.8 `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.

10.273.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [related, inherited]

Give the unique `set` of `s`.

10.274 mln::p_image< I > Class Template Reference

[Site set](#) based on an image of Booleans.

```
#include <p_image.hh>
```

Inherits mln::internal::site_set_base_< I::psite, mln::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](#) (const I &ima)
Constructor.
- [p_image](#) ()
Constructor without argument.
- void [remove](#) (const [psite](#) &p)
Remove a site p.
- void [toggle](#) (const [psite](#) &p)
Change the status in/out of a site p.

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 Sl, typename Sr>`
`p_set< typename Sl::site > sym_diff (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic symmetrical difference of lhs and rhs.
- `template<typename Sl, typename Sr>`
`p_set< typename Sl::site > uni (const Site_Set< Sl > &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.

10.274.1 Detailed Description

`template<typename I> class mln::p_image< I >`

Site set based on an image of Booleans.

10.274.2 Member Typedef Documentation

10.274.2.1 `template<typename I> typedef S ::bkd_piter mln::p_image< I >::bkd_piter`

Backward [Site_Iterator](#) associated type.

10.274.2.2 `template<typename I> typedef I ::psite mln::p_image< I >::element`

Element associated type.

10.274.2.3 `template<typename I> typedef S ::fwd_piter mln::p_image< I >::fwd_piter`

Forward [Site_Iterator](#) associated type.

10.274.2.4 `template<typename I> typedef psite mln::p_image< I >::i_element`

Insertion element associated type.

10.274.2.5 `template<typename I> typedef S ::piter mln::p_image< I >::piter`

[Site_Iterator](#) associated type.

10.274.2.6 `template<typename I> typedef I ::psite mln::p_image< I >::psite`

Psite associated type.

10.274.2.7 `template<typename I> typedef psite mln::p_image< I >::r_element`

Removal element associated type.

10.274.2.8 `template<typename I> typedef internal::p_image_site_set<I>::ret mln::p_image< I >::S`

Equivalent site_set type.

10.274.3 Constructor & Destructor Documentation**10.274.3.1** `template<typename I> mln::p_image< I >::p_image () [inline]`

Constructor without argument.

10.274.3.2 `template<typename I> mln::p_image< I >::p_image (const I & ima) [inline]`

Constructor.

References mln::p_image< I >::clear().

10.274.4 Member Function Documentation**10.274.4.1** `template<typename I> void mln::p_image< I >::clear () [inline]`

Clear this [set](#).

References mln::data::fill_with_value(), and mln::p_image< I >::is_valid().

Referenced by mln::p_image< I >::p_image().

10.274.4.2 `template<typename I> bool mln::p_image< I >::has (const psite & p) const [inline]`

Test is the psite *p* belongs to this site [set](#).

References mln::p_image< I >::is_valid().

10.274.4.3 `template<typename I> void mln::p_image< I >::insert (const psite & p) [inline]`

Insert a site *p*.

References mln::p_image< I >::is_valid().

10.274.4.4 `template<typename I> bool mln::p_image< I >::is_valid () const [inline]`

Test if this site [set](#) is valid, i.e., initialized.

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().

10.274.4.5 `template<typename I> std::size_t mln::p_image< I >::memory_size () const [inline]`

Return the size of this site [set](#) in memory.

References `mln::p_image< I >::is_valid()`.

10.274.4.6 `template<typename I> unsigned mln::p_image< I >::nsites () const` `[inline]`

Give the number of sites.

10.274.4.7 `template<typename I> mln::p_image< I >::operator typename internal::p_image_site_set< I >::ret () const` `[inline]`

Conversion towards the equivalent site [set](#).

10.274.4.8 `template<typename I> void mln::p_image< I >::remove (const psite & p)` `[inline]`

Remove a site `p`.

References `mln::p_image< I >::is_valid()`.

10.274.4.9 `template<typename I> void mln::p_image< I >::toggle (const psite & p)` `[inline]`

Change the status in/out of a site `p`.

References `mln::p_image< I >::is_valid()`.

10.274.5 Friends And Related Function Documentation

10.274.5.1 `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`.

10.274.5.2 `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.

10.274.5.3 `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:

← *lhs* A site [set](#) (strictly included?).

← *rhs* Another site [set](#) (includer?).

10.274.5.4 `template<typename S> std::ostream & operator<< (std::ostream & ostr, const Site_Set< S > & set)` [related, inherited]

Print a site `set` into the output stream `ostr`.

Parameters:

↔ `ostr` An output stream.

← `set` A site `set`.

Returns:

The modified output stream `ostr`.

10.274.5.5 `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:

← `lhs` A site `set` (included?).

← `rhs` Another site `set` (includer?).

10.274.5.6 `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:

← `lhs` A site `set`.

← `rhs` Another site `set`.

10.274.5.7 `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`.

10.274.5.8 `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.

10.274.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [related, inherited]

Give the unique `set` of `s`.

10.275 mln::p_indexed_bkd_piter< S > Class Template Reference

Backward iterator on sites of an indexed site [set](#).

```
#include <p_array.hh>
```

Inherits mln::internal::site_set_iterator_base< S, mln::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.

10.275.1 Detailed Description

```
template<typename S> class mln::p_indexed_bkd_piter< S >
```

Backward iterator on sites of an indexed site [set](#).

10.275.2 Constructor & Destructor Documentation

10.275.2.1 `template<typename S> mln::p_indexed_bkd_piter< S >::p_indexed_bkd_piter ()`
[inline]

Constructor with no argument.

10.275.2.2 `template<typename S> mln::p_indexed_bkd_piter< S >::p_indexed_bkd_piter (const S &s)` [inline]

Constructor.

10.275.3 Member Function Documentation

10.275.3.1 `template<typename S> int mln::p_indexed_bkd_piter< S >::index () const`
[inline]

Return the current index.

10.275.3.2 `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.

10.276 mln::p_indexed_fwd_piter< S > Class Template Reference

Forward iterator on sites of an indexed site [set](#).

```
#include <p_array.hh>
```

Inherits mln::internal::site_set_iterator_base< S, mln::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.

10.276.1 Detailed Description

```
template<typename S> class mln::p_indexed_fwd_piter< S >
```

Forward iterator on sites of an indexed site [set](#).

10.276.2 Constructor & Destructor Documentation

10.276.2.1 `template<typename S> mln::p_indexed_fwd_piter< S >::p_indexed_fwd_piter ()`
[inline]

Constructor with no argument.

10.276.2.2 `template<typename S> mln::p_indexed_fwd_piter< S >::p_indexed_fwd_piter (const S &s)` [inline]

Constructor.

10.276.3 Member Function Documentation

10.276.3.1 `template<typename S> int mln::p_indexed_fwd_piter< S >::index () const`
[inline]

Return the current index.

10.276.3.2 `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.

10.277 mln::p_indexed_psite< S > Class Template Reference

Psite class for indexed site sets such as [p_array](#).

```
#include <p_array.hh>
```

Inherits mln::internal::pseudo_site_base_< const S::element &, mln::p_indexed_psite< S > >.

10.277.1 Detailed Description

```
template<typename S> class mln::p_indexed_psite< S >
```

Psite class for indexed site sets such as [p_array](#).

.

10.278 mln::p_key< K, P > Class Template Reference

Priority queue class.

```
#include <p_key.hh>
```

Inherits mln::internal::site_set_base_< P, mln::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 P &p) const

*Test is the psite p belongs to this site *set*.*

- `bool has (const psite &) const`
*Test is the psite p belongs to this site *set*.*
- `void insert (const K &k, const P &p)`
Insert a pair (key k , site p).
- `void insert (const i_element &k_p)`
Insert a pair k_p (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 .

Related Functions

(Note that these are not member functions.)

- `template<typename Sl, typename Sr>`
`p_set< typename Sl::site > diff (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic difference of lhs and rhs.
- `template<typename Sl, typename Sr>`
`p_set< typename Sl::site > inter (const Site_Set< Sl > &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 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.

10.278.1 Detailed Description

`template<typename K, typename P> class mln::p_key< K, P >`

Priority queue class.

10.278.2 Member Typedef Documentation

10.278.2.1 `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.

10.278.2.2 `template<typename K, typename P> typedef P mln::p_key< K, P >::element`

Element associated type.

10.278.2.3 `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.

10.278.2.4 `template<typename K, typename P> typedef std::pair<K,P> mln::p_key< K, P >::i_element`

Insertion element associated type.

10.278.2.5 `template<typename K, typename P> typedef fwd_piter mln::p_key< K, P >::piter`

[Site_Iterator](#) associated type.

10.278.2.6 `template<typename K, typename P> typedef p_double_psite< self_, p_set<P> > mln::p_key< K, P >::psite`

Psite associated type.

10.278.2.7 `template<typename K, typename P> typedef P mln::p_key< K, P >::r_element`

Removal element associated type.

10.278.3 Constructor & Destructor Documentation

10.278.3.1 `template<typename K, typename P> mln::p_key< K, P >::p_key () [inline]`

Constructor.

10.278.4 Member Function Documentation

10.278.4.1 `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`.

References `mln::p_set< P >::nsites()`.

10.278.4.2 `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`.

References `mln::util::set< T >::insert()`.

10.278.4.3 `template<typename K, typename P> void mln::p_key< K, P >::clear () [inline]`

Clear this site [set](#).

10.278.4.4 `template<typename K, typename P> bool mln::p_key< K, P >::exists_key (const K & key) const [inline]`

Test if the `priority` exists.

Referenced by `mln::p_key< K, P >::operator()()`.

10.278.4.5 `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](#).

10.278.4.6 `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](#).

Referenced by `mln::p_key< K, P >::insert()`.

10.278.4.7 `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`).

References `mln::p_key< K, P >::has()`.

10.278.4.8 `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`).

10.278.4.9 `template<typename K, typename P> bool mln::p_key< K, P >::is_valid () const [inline]`

Test this [set](#) validity so returns always true.

10.278.4.10 `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`.

10.278.4.11 `template<typename K, typename P> const util::set< K > & mln::p_key< K, P >::keys () const [inline]`

Give the [set](#) of keys.

10.278.4.12 `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.

10.278.4.13 `template<typename K, typename P> unsigned mln::p_key< K, P >::nsites () const`
`[inline]`

Give the number of sites.

10.278.4.14 `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.

References `mln::p_key< K, P >::exists_key()`.

10.278.4.15 `template<typename K, typename P> void mln::p_key< K, P >::remove (const P & p)`
`[inline]`

Remove a site `p`.

10.278.4.16 `template<typename K, typename P> void mln::p_key< K, P >::remove_key (const K & k)` `[inline]`

Remove all sites with key `k`.

References `mln::p_set< P >::nsites()`.

10.278.5 Friends And Related Function Documentation

10.278.5.1 `template<typename Sl, typename Sr> p_set< typename Sl::site > diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` `[related, inherited]`

Set theoretic difference of `lhs` and `rhs`.

10.278.5.2 `template<typename Sl, typename Sr> p_set< typename Sl::site > inter (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` `[related, inherited]`

Intersection between a couple of [point](#) sets.

10.278.5.3 `template<typename Sl, typename Sr> bool operator< (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` `[related, inherited]`

Strict inclusion [test](#) between site sets `lhs` and `rhs`.

Parameters:

← *lhs* A site [set](#) (strictly included?).

← *rhs* Another site [set](#) (includer?).

10.278.5.4 `template<typename S> std::ostream & operator<< (std::ostream & ostr, const Site_Set< S > & set)` [related, inherited]

Print a site `set` into the output stream `ostr`.

Parameters:

↔ *ostr* An output stream.

← *set* A site `set`.

Returns:

The modified output stream `ostr`.

10.278.5.5 `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:

← *lhs* A site `set` (included?).

← *rhs* Another site `set` (includer?).

10.278.5.6 `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:

← *lhs* A site `set`.

← *rhs* Another site `set`.

10.278.5.7 `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`.

10.278.5.8 `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.

10.278.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [related, inherited]

Give the unique `set` of `s`.

10.279 mln::p_line2d Class Reference

2D discrete line of points.

```
#include <p_line2d.hh>
```

Inherits mln::internal::site_set_base_< mln::point, mln::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.

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 Sl, typename Sr>`
`p_set< typename Sl::site > uni` (const `Site_Set< Sl >` &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.

10.279.1 Detailed Description

2D discrete line of points.

It is based on [p_array](#).

10.279.2 Member Typedef Documentation

10.279.2.1 `typedef p_indexed_bkd_piter<self_> mln::p_line2d::bkd_piter`

Backward [Site_Iterator](#) associated type.

10.279.2.2 `typedef point2d mln::p_line2d::element`

Element associated type.

10.279.2.3 `typedef p_indexed_fwd_piter<self_> mln::p_line2d::fwd_piter`

Forward [Site_Iterator](#) associated type.

10.279.2.4 `typedef p_indexed_fwd_piter<self_> mln::p_line2d::piter`

[Site_Iterator](#) associated type.

10.279.2.5 `typedef p_indexed_psite<self_> mln::p_line2d::psite`

Psite associated type.

10.279.2.6 `typedef const box2d& mln::p_line2d::q_box`

[Box](#) (qualified) associated type.

10.279.3 Constructor & Destructor Documentation

10.279.3.1 `mln::p_line2d::p_line2d () [inline]`

Constructor without argument.

References `is_valid()`.

10.279.3.2 `mln::p_line2d::p_line2d (const point2d & beg, const point2d & end, bool is_end_excluded = false) [inline]`

Constructor from [point](#) beg to [point](#) end.

References `is_valid()`.

10.279.4 Member Function Documentation**10.279.4.1** `const box2d & mln::p_line2d::bbox () const [inline]`

Give the exact bounding [box](#).

References `is_valid()`.

10.279.4.2 `const point2d & mln::p_line2d::begin () const [inline]`

Give the [point](#) that begins the line.

References `is_valid()`.

Referenced by `mln::debug::draw_graph()`.

10.279.4.3 `const point2d & mln::p_line2d::end () const [inline]`

Give the [point](#) that ends the line.

References `is_valid()`, and `nsites()`.

Referenced by `mln::debug::draw_graph()`.

10.279.4.4 `bool mln::p_line2d::has (const util::index & i) const [inline]`

Test if index `i` belongs to this [point set](#).

References `nsites()`.

10.279.4.5 `bool mln::p_line2d::has (const psite & p) const [inline]`

Test if `p` belongs to this [point set](#).

References `mln::p_indexed_psite< S >::index()`.

10.279.4.6 `bool mln::p_line2d::is_valid () const [inline]`

Test if this line is valid, i.e., initialized.

References `mln::implies()`.

Referenced by `bbox()`, `begin()`, `end()`, and `p_line2d()`.

10.279.4.7 `std::size_t mln::p_line2d::memory_size () const [inline]`

Return the size of this [site set](#) in memory.

10.279.4.8 `unsigned mln::p_line2d::nsites () const` [inline]

Give the number of points.

Referenced by `end()`, `has()`, and `operator[]()`.

10.279.4.9 `]`

`const point2d & mln::p_line2d::operator[] (unsigned i) const` [inline]

Return the *i*-th `point` of the line.

References `nsites()`.

10.279.4.10 `const std::vector< point2d > & mln::p_line2d::std_vector () const` [inline]

Return the corresponding `std::vector` of points.

10.279.5 Friends And Related Function Documentation**10.279.5.1** `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`.

10.279.5.2 `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.

10.279.5.3 `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:

← *lhs* A site `set` (strictly included?).

← *rhs* Another site `set` (includer?).

10.279.5.4 `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:

↔ *ostr* An output stream.

← *set* A site `set`.

Returns:

The modified output stream `ostr`.

10.279.5.5 `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:

- ← *lhs* A site [set](#) (included?).
- ← *rhs* Another site [set](#) (includer?).

10.279.5.6 `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:

- ← *lhs* A site [set](#).
- ← *rhs* Another site [set](#).

10.279.5.7 `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`.

10.279.5.8 `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.

10.279.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [related, inherited]

Give the unique [set](#) of `s`.

10.280 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 `mln::internal::site_set_base_< S::site, mln::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.

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.

10.280.1 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.

10.280.2 Member Typedef Documentation

10.280.2.1 `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.

10.280.2.2 `template<typename S> typedef S mln::p_mutable_array_of< S >::element`

Element associated type.

10.280.2.3 `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.

10.280.2.4 `template<typename S> typedef S mln::p_mutable_array_of< S >::i_element`

Insertion element associated type.

10.280.2.5 `template<typename S> typedef fwd_piter mln::p_mutable_array_of< S >::piter`

[Site_Iterator](#) associated type.

10.280.2.6 `template<typename S> typedef p_double_psite<self_, element>
mln::p_mutable_array_of< S >::psite`

Psite associated type.

10.280.3 Constructor & Destructor Documentation

10.280.3.1 `template<typename S> mln::p_mutable_array_of< S >::p_mutable_array_of ()
[inline]`

Constructor without arguments.

10.280.4 Member Function Documentation

10.280.4.1 `template<typename S> void mln::p_mutable_array_of< S >::clear () [inline]`

Clear this [set](#).

10.280.4.2 `template<typename S> bool mln::p_mutable_array_of< S >::has (const psite & p)
const [inline]`

Test if *p* belongs to this [point set](#).

10.280.4.3 `template<typename S> void mln::p_mutable_array_of< S >::insert (const S & s)
[inline]`

Insert a site [set](#) *s*.

Precondition:

s is valid.

10.280.4.4 `template<typename S> bool mln::p_mutable_array_of< S >::is_valid () const
[inline]`

Test this [set](#) validity so returns always true.

10.280.4.5 `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.

10.280.4.6 `template<typename S> unsigned mln::p_mutable_array_of< S >::nelements () const
[inline]`

Give the number of elements (site sets) of this composite.

10.280.4.7]

`template<typename S> S & mln::p_mutable_array_of< S >::operator[] (unsigned i) [inline]`

Return the *i*-th site [set](#) (mutable version).

10.280.4.8]

`template<typename S> const S & mln::p_mutable_array_of< S >::operator[] (unsigned i) const
[inline]`

Return the *i*-th site [set](#) (const version).

10.280.4.9 `template<typename S> void mln::p_mutable_array_of< S >::reserve (unsigned n)`
`[inline]`

Reserve memory for `n` elements.

10.280.5 Friends And Related Function Documentation

10.280.5.1 `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`.

10.280.5.2 `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.

10.280.5.3 `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:

- ← *lhs* A site [set](#) (strictly included?).
- ← *rhs* Another site [set](#) (includer?).

10.280.5.4 `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:

- ↔ *ostr* An output stream.
- ← *set* A site [set](#).

Returns:

The modified output stream `ostr`.

10.280.5.5 `template<typename SI, typename Sr> bool operator<= (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` `[related, inherited]`

Inclusion [test](#) between site sets `lhs` and `rhs`.

Parameters:

- ← *lhs* A site [set](#) (included?).
- ← *rhs* Another site [set](#) (includer?).

10.280.5.6 `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:

← *lhs* A site [set](#).

← *rhs* Another site [set](#).

10.280.5.7 `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`.

10.280.5.8 `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.

10.280.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [related, inherited]

Give the unique [set](#) of `s`.

10.281 mln::p_n_faces_bkd_piter< D, P > Class Template Reference

Backward iterator on the n-faces sites of an mln::p_complex<D, P>.

```
#include <p_n_faces_piter.hh>
```

Inherits mln::internal::p_complex_piter_base_< mln::topo::n_face_bkd_iter< D >, mln::p_complex< D, P >, P, mln::p_n_faces_bkd_piter< D, P > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- unsigned [n](#) () const
Accessors.
- [p_n_faces_bkd_piter](#) ()
Construction and assignment.

10.281.1 Detailed Description

```
template<unsigned D, typename P> class mln::p_n_faces_bkd_piter< D, P >
```

Backward iterator on the n-faces sites of an mln::p_complex<D, P>.

10.281.2 Constructor & Destructor Documentation

10.281.2.1 `template<unsigned D, typename P> mln::p_n_faces_bkd_piter< D, P >::p_n_faces_bkd_piter () [inline]`

Construction and assignment.

10.281.3 Member Function Documentation

10.281.3.1 `template<unsigned D, typename P> unsigned mln::p_n_faces_bkd_piter< D, P >::n () const [inline]`

Accessors.

Shortcuts to face_'s accessors.

10.281.3.2 `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-define this method. The actual "next" operation has to be defined through the *next_* method.

Precondition:

The iterator is valid.

10.282 mln::p_n_faces_fwd_piter< D, P > Class Template Reference

Forward iterator on the n-faces sites of an mln::p_complex<D, P>.

```
#include <p_n_faces_piter.hh>
```

Inherits mln::internal::p_complex_piter_base_< mln::topo::n_face_fwd_iter< D >, mln::p_complex< D, P >, P, mln::p_n_faces_fwd_piter< D, P > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- unsigned [n](#) () const
Accessors.
- [p_n_faces_fwd_piter](#) ()
Construction and assignment.

10.282.1 Detailed Description

```
template<unsigned D, typename P> class mln::p_n_faces_fwd_piter< D, P >
```

Forward iterator on the n-faces sites of an mln::p_complex<D, P>.

10.282.2 Constructor & Destructor Documentation

10.282.2.1 `template<unsigned D, typename P> mln::p_n_faces_fwd_piter< D, P >::p_n_faces_fwd_piter () [inline]`

Construction and assignment.

10.282.3 Member Function Documentation

10.282.3.1 `template<unsigned D, typename P> unsigned mln::p_n_faces_fwd_piter< D, P >::n () const [inline]`

Accessors.

Shortcuts to face_'s accessors.

10.282.3.2 `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-define this method. The actual "next" operation has to be defined through the *next_* method.

Precondition:

The iterator is valid.

10.283 mln::p_priority< P, Q > Class Template Reference

Priority queue.

```
#include <p_priority.hh>
```

Inherits mln::internal::site_set_base_< Q::site, mln::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 [p_priority](#)< P, Q > &other)
Insert elements from another priority queue.

- void `insert` (const `i_element` &p_e)
Insert a pair p_e (priority p, element e).
- 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.

Related Functions

(Note that these are not member functions.)

- template<typename S1, typename Sr>
`p_set`< typename S1::site > `diff` (const `Site_Set`< S1 > &lhs, const `Site_Set`< Sr > &rhs)
Set theoretic difference of lhs and rhs.
- template<typename S1, typename Sr>
`p_set`< typename S1::site > `inter` (const `Site_Set`< S1 > &lhs, const `Site_Set`< Sr > &rhs)
Intersection between a couple of point sets.
- template<typename S1, typename Sr>
bool `operator`< (const `Site_Set`< S1 > &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`.*

10.283.1 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>`).

10.283.2 Member Typedef Documentation

10.283.2.1 `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.

10.283.2.2 `template<typename P, typename Q> typedef Q ::element mln::p_priority< P, Q >::element`

Element associated type.

10.283.2.3 `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.

10.283.2.4 `template<typename P, typename Q> typedef std::pair<P, element> mln::p_priority< P, Q >::i_element`

Insertion element associated type.

10.283.2.5 `template<typename P, typename Q> typedef fwd_piter mln::p_priority< P, Q >::piter`

[Site_Iterator](#) associated type.

10.283.2.6 `template<typename P, typename Q> typedef p_double_psite<self_, Q> mln::p_priority< P, Q >::psite`

Psite associated type.

10.283.3 Constructor & Destructor Documentation

10.283.3.1 `template<typename P, typename Q> mln::p_priority< P, Q >::p_priority () [inline]`

Constructor.

10.283.4 Member Function Documentation

10.283.4.1 `template<typename P, typename Q> void mln::p_priority< P, Q >::clear () [inline]`

Clear the queue.

10.283.4.2 `template<typename P, typename Q> bool mln::p_priority< P, Q >::exists_priority (const P & priority) const [inline]`

Test if the `priority` exists.

Referenced by `mln::p_priority< P, Q >::operator()`.

10.283.4.3 `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()

References mln::p_priority< P, Q >::highest_priority().

Referenced by mln::morpho::meyer_wst(), mln::p_priority< P, Q >::pop_front(), and mln::morpho::watershed::topological().

10.283.4.4 **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](#).

10.283.4.5 **template<typename P, typename Q> const P mln::p_priority< P, Q >::highest_priority () const** [inline]

Give the highest priority.

Precondition:

! is_empty()

Referenced by mln::p_priority< P, Q >::front(), and mln::p_priority< P, Q >::pop().

10.283.4.6 **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.

10.283.4.7 **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).

References mln::p_priority< P, Q >::push().

10.283.4.8 **template<typename P, typename Q> bool mln::p_priority< P, Q >::is_valid () const** [inline]

Test this [set](#) validity so returns always true.

10.283.4.9 **template<typename P, typename Q> const P mln::p_priority< P, Q >::lowest_priority () const** [inline]

Give the lowest priority.

Precondition:

! is_empty()

10.283.4.10 `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.

10.283.4.11 `template<typename P, typename Q> unsigned mln::p_priority< P, Q >::nsites () const` [inline]

Give the number of sites.

Referenced by `mln::p_priority< P, Q >::operator()`.

10.283.4.12 `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.

References `mln::p_priority< P, Q >::exists_priority()`, and `mln::p_priority< P, Q >::nsites()`.

10.283.4.13 `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()`

References `mln::p_priority< P, Q >::highest_priority()`.

Referenced by `mln::morpho::meyer_wst()`, `mln::p_priority< P, Q >::pop_front()`, and `mln::morpho::watershed::topological()`.

10.283.4.14 `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()`

References `mln::p_priority< P, Q >::front()`, and `mln::p_priority< P, Q >::pop()`.

Referenced by `mln::geom::impl::seeds2tiling_roundness()`.

10.283.4.15 `template<typename P, typename Q> const util::set< P > & mln::p_priority< P, Q >::priorities () const` [inline]

Give the [set](#) of priorities.

10.283.4.16 `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`.

Referenced by `mln::p_priority< P, Q >::insert()`, `mln::morpho::meyer_wst()`, `mln::geom::impl::seeds2tiling_roundness()`, and `mln::morpho::watershed::topological()`.

10.283.5 Friends And Related Function Documentation

10.283.5.1 `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`.

10.283.5.2 `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.

10.283.5.3 `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:

- ← *lhs* A site [set](#) (strictly included?).
- ← *rhs* Another site [set](#) (includer?).

10.283.5.4 `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:

- ↔ *ostr* An output stream.
- ← *set* A site [set](#).

Returns:

The modified output stream `ostr`.

10.283.5.5 `template<typename SI, typename Sr> bool operator<= (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Inclusion [test](#) between site sets `lhs` and `rhs`.

Parameters:

- ← *lhs* A site [set](#) (included?).
- ← *rhs* Another site [set](#) (includer?).

10.283.5.6 `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:

- ← *lhs* A site [set](#).
- ← *rhs* Another site [set](#).

10.283.5.7 `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`.

10.283.5.8 `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.

10.283.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [[related](#), [inherited](#)]

Give the unique [set](#) of `s`.

10.284 mln::p_queue< P > Class Template Reference

Queue of sites (based on std::deque).

```
#include <p_queue.hh>
```

Inherits mln::internal::site_set_base_< P, mln::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 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` ()
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.

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.

10.284.1 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.

10.284.2 Member Typedef Documentation

10.284.2.1 `template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_queue< P >::bkd_piter`

Backward [Site_Iterator](#) associated type.

10.284.2.2 `template<typename P> typedef P mln::p_queue< P >::element`

Element associated type.

10.284.2.3 `template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_queue< P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

10.284.2.4 `template<typename P> typedef P mln::p_queue< P >::i_element`

Insertion element associated type.

10.284.2.5 `template<typename P> typedef fwd_piter mln::p_queue< P >::piter`

[Site_Iterator](#) associated type.

10.284.2.6 `template<typename P> typedef p_indexed_psite<self_> mln::p_queue< P >::psite`

Psite associated type.

10.284.3 Constructor & Destructor Documentation**10.284.3.1** `template<typename P> mln::p_queue< P >::p_queue () [inline]`

Constructor without argument.

10.284.4 Member Function Documentation**10.284.4.1** `template<typename P> void mln::p_queue< P >::clear () [inline]`

Clear the queue.

10.284.4.2 `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.

Referenced by `mln::p_queue< P >::pop_front()`, and `mln::geom::impl::seeds2tiling()`.

10.284.4.3 `template<typename P> bool mln::p_queue< P >::has (const util::index & i) const [inline]`

Test if index *i* belongs to this site [set](#).

References `mln::p_queue< P >::nsites()`.

10.284.4.4 `template<typename P> bool mln::p_queue< P >::has (const psite & p) const [inline]`

Test if *p* belongs to this site [set](#).

References `mln::p_indexed_psite< S >::index()`, and `mln::p_queue< P >::nsites()`.

10.284.4.5 `template<typename P> void mln::p_queue< P >::insert (const P & p) [inline]`

Insert a site *p* (equivalent as 'push').

References `mln::p_queue< P >::push()`.

10.284.4.6 `template<typename P> bool mln::p_queue< P >::is_valid () const [inline]`

This [set](#) is always valid so it returns true.

10.284.4.7 `template<typename P> std::size_t mln::p_queue< P >::memory_size () const [inline]`

Return the size of this site [set](#) in memory.

References `mln::p_queue< P >::nsites()`.

10.284.4.8 `template<typename P> unsigned mln::p_queue< P >::nsites () const` [inline]

Give the number of sites.

Referenced by `mln::p_queue< P >::has()`, `mln::p_queue< P >::memory_size()`, and `mln::p_queue< P >::operator[]()`.

10.284.4.9]

`template<typename P> const P & mln::p_queue< P >::operator[] (unsigned i) const` [inline]

Return the *i*-th site.

References `mln::p_queue< P >::nsites()`.

10.284.4.10 `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.

Referenced by `mln::p_queue< P >::pop_front()`, and `mln::geom::impl::seeds2tiling()`.

10.284.4.11 `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.

References `mln::p_queue< P >::front()`, and `mln::p_queue< P >::pop()`.

10.284.4.12 `template<typename P> void mln::p_queue< P >::push (const P & p)` [inline]

Push a site *p* in the queue.

Referenced by `mln::p_queue< P >::insert()`, and `mln::geom::impl::seeds2tiling()`.

10.284.4.13 `template<typename P> const std::deque< P > & mln::p_queue< P >::std_deque () const` [inline]

Return the corresponding `std::deque` of sites.

10.284.5 Friends And Related Function Documentation

10.284.5.1 `template<typename Sl, typename Sr> p_set< typename Sl::site > diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Set theoretic difference of *lhs* and *rhs*.

10.284.5.2 `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.

10.284.5.3 `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:

- ← *lhs* A site [set](#) (strictly included?).
- ← *rhs* Another site [set](#) (includer?).

10.284.5.4 `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:

- ↔ *ostr* An output stream.
- ← *set* A site [set](#).

Returns:

The modified output stream `ostr`.

10.284.5.5 `template<typename SI, typename Sr> bool operator<= (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Inclusion [test](#) between site sets `lhs` and `rhs`.

Parameters:

- ← *lhs* A site [set](#) (included?).
- ← *rhs* Another site [set](#) (includer?).

10.284.5.6 `template<typename SI, typename Sr> bool operator== (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Equality [test](#) between site sets `lhs` and `rhs`.

Parameters:

- ← *lhs* A site [set](#).
- ← *rhs* Another site [set](#).

10.284.5.7 `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.

10.284.5.8 `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.

10.284.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [related, inherited]

Give the unique [set](#) of s.

10.285 mln::p_queue_fast< P > Class Template Reference

Queue of sites class (based on [p_array](#)).

```
#include <p_queue_fast.hh>
```

Inherits mln::internal::site_set_base_< P, mln::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.

Related Functions

(Note that these are not member functions.)

- `template<typename Sl, typename Sr>`
`p_set< typename Sl::site > diff (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic difference of `lhs` and `rhs`.
- `template<typename Sl, typename Sr>`
`p_set< typename Sl::site > inter (const Site_Set< Sl > &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.

10.285.1 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.

10.285.2 Member Typedef Documentation

10.285.2.1 `template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_queue_fast< P >::bkd_piter`

Backward `Site_Iterator` associated type.

10.285.2.2 `template<typename P> typedef P mln::p_queue_fast< P >::element`

Element associated type.

10.285.2.3 `template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_queue_fast< P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

10.285.2.4 `template<typename P> typedef P mln::p_queue_fast< P >::i_element`

Insertion element associated type.

10.285.2.5 `template<typename P> typedef fwd_piter mln::p_queue_fast< P >::piter`

[Site_Iterator](#) associated type.

10.285.2.6 `template<typename P> typedef p_indexed_psite<self_> mln::p_queue_fast< P >::psite`

Psite associated type.

10.285.3 Constructor & Destructor Documentation

10.285.3.1 `template<typename P> mln::p_queue_fast< P >::p_queue_fast () [inline]`

Constructor without argument.

10.285.4 Member Function Documentation

10.285.4.1 `template<typename P> void mln::p_queue_fast< P >::clear () [inline]`

Clear the queue.

10.285.4.2 `template<typename P> bool mln::p_queue_fast< P >::compute_has (const P & p) const [inline]`

Test if `p` belongs to this site [set](#).

10.285.4.3 `template<typename P> bool mln::p_queue_fast< P >::empty () const [inline]`

Test if the queue is empty.

10.285.4.4 `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.

Referenced by `mln::p_queue_fast< P >::pop_front()`.

10.285.4.5 `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](#).

References `mln::p_queue_fast< P >::nsites()`.

10.285.4.6 `template<typename P> bool mln::p_queue_fast< P >::has (const psite & p) const [inline]`

Test if *p* belongs to this site [set](#).

References `mln::p_indexed_psite< S >::index()`, and `mln::p_queue_fast< P >::nsites()`.

10.285.4.7 `template<typename P> void mln::p_queue_fast< P >::insert (const P & p) [inline]`

Insert a site *p* (equivalent as 'push').

References `mln::p_queue_fast< P >::push()`.

10.285.4.8 `template<typename P> bool mln::p_queue_fast< P >::is_valid () const [inline]`

This [set](#) is always valid so it returns true.

10.285.4.9 `template<typename P> std::size_t mln::p_queue_fast< P >::memory_size () const [inline]`

Return the size of this site [set](#) in memory.

10.285.4.10 `template<typename P> unsigned mln::p_queue_fast< P >::nsites () const [inline]`

Give the number of sites.

Referenced by `mln::p_queue_fast< P >::has()`, and `mln::p_queue_fast< P >::operator[]()`.

10.285.4.11 `]`

`template<typename P> const P & mln::p_queue_fast< P >::operator[] (unsigned i) const [inline]`

Return the *i*-th site.

References `mln::p_queue_fast< P >::nsites()`.

10.285.4.12 `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.

Referenced by `mln::p_queue_fast< P >::pop_front()`.

10.285.4.13 `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.

References `mln::p_queue_fast< P >::front()`, and `mln::p_queue_fast< P >::pop()`.

10.285.4.14 `template<typename P> void mln::p_queue_fast< P >::purge ()` `[inline]`

Purge the queue to save (free) some memory.

10.285.4.15 `template<typename P> void mln::p_queue_fast< P >::push (const P & p)`
`[inline]`

Push a site `p` in the queue.

Referenced by `mln::p_queue_fast< P >::insert()`.

10.285.4.16 `template<typename P> void mln::p_queue_fast< P >::reserve (typename p_array< P >::size_type n)` `[inline]`

Reserve `n` cells.

10.285.4.17 `template<typename P> const std::vector< P > & mln::p_queue_fast< P >::std_vector () const` `[inline]`

Return the corresponding `std::vector` of sites.

10.285.5 Friends And Related Function Documentation

10.285.5.1 `template<typename Sl, typename Sr> p_set< typename Sl::site > diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` `[related, inherited]`

Set theoretic difference of `lhs` and `rhs`.

10.285.5.2 `template<typename Sl, typename Sr> p_set< typename Sl::site > inter (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` `[related, inherited]`

Intersection between a couple of [point](#) sets.

10.285.5.3 `template<typename Sl, typename Sr> bool operator< (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` `[related, inherited]`

Strict inclusion [test](#) between site sets `lhs` and `rhs`.

Parameters:

← *lhs* A site [set](#) (strictly included?).

← *rhs* Another site [set](#) (includer?).

10.285.5.4 `template<typename S> std::ostream & operator<< (std::ostream & ostr, const Site_Set< S > & set)` [related, inherited]

Print a site `set` into the output stream `ostr`.

Parameters:

↔ *ostr* An output stream.

← *set* A site `set`.

Returns:

The modified output stream `ostr`.

10.285.5.5 `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:

← *lhs* A site `set` (included?).

← *rhs* Another site `set` (includer?).

10.285.5.6 `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:

← *lhs* A site `set`.

← *rhs* Another site `set`.

10.285.5.7 `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`.

10.285.5.8 `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.

10.285.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [related, inherited]

Give the unique `set` of `s`.

10.286 mln::p_run< P > Class Template Reference

[Point set](#) class in [run](#).

```
#include <p_run.hh>
```

Inherits [mln::internal::site_set_base_< P, mln::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` (const `P &start`, const `P &end`)
Constructor.
- `p_run` (const `P &start`, unsigned short `len`)
Constructor.
- `p_run ()`
Constructor without argument.
- const `P & start ()` const
Return the starting `point`.

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.

10.286.1 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.

10.286.2 Member Typedef Documentation

10.286.2.1 `template<typename P> typedef p_run_bkd_piter_<P> mln::p_run< P >::bkd_piter`

Backward Site_Iterator associated type.

10.286.2.2 `template<typename P> typedef P mln::p_run< P >::element`

Element associated type.

10.286.2.3 `template<typename P> typedef p_run_fwd_piter_<P> mln::p_run< P >::fwd_piter`

Forward Site_Iterator associated type.

10.286.2.4 `template<typename P> typedef fwd_piter mln::p_run< P >::piter`

Site_Iterator associated type.

10.286.2.5 `template<typename P> typedef p_run_psite<P> mln::p_run< P >::psite`

Psite associated type.

10.286.2.6 `template<typename P> typedef mln::box<P> mln::p_run< P >::q_box`

[Box](#) associated type.

10.286.3 **Constructor & Destructor Documentation****10.286.3.1** `template<typename P> mln::p_run< P >::p_run () [inline]`

Constructor without argument.

10.286.3.2 `template<typename P> mln::p_run< P >::p_run (const P & start, unsigned short len) [inline]`

Constructor.

References `mln::p_run< P >::init()`.

10.286.3.3 `template<typename P> mln::p_run< P >::p_run (const P & start, const P & end) [inline]`

Constructor.

10.286.4 **Member Function Documentation****10.286.4.1** `template<typename P> mln::box< P > mln::p_run< P >::bbox () const [inline]`

Give the exact bounding [box](#).

References `mln::p_run< P >::end()`.

10.286.4.2 `template<typename P> P mln::p_run< P >::end () const [inline]`

Return (compute) the ending [point](#).

References `mln::point< G, C >::last_coord()`.

Referenced by `mln::p_run< P >::bbox()`.

10.286.4.3 `template<typename P> bool mln::p_run< P >::has (const P & p) const [inline]`

Test if `p` belongs to this [point set](#).

References `mln::p_run< P >::is_valid()`.

10.286.4.4 `template<typename P> bool mln::p_run< P >::has (const psite & p) const [inline]`

Test if `p` belongs to this [point set](#).

10.286.4.5 `template<typename P> bool mln::p_run< P >::has_index (unsigned short i) const` `[inline]`

Test if index *i* belongs to this [point set](#).

10.286.4.6 `template<typename P> void mln::p_run< P >::init (const P & start, unsigned short len) [inline]`

Set the starting [point](#).

Referenced by `mln::p_run< P >::p_run()`.

10.286.4.7 `template<typename P> bool mln::p_run< P >::is_valid () const [inline]`

Test if this run is valid, i.e., with length > 0.

Referenced by `mln::p_run< P >::has()`, `mln::p_run< P >::length()`, `mln::p_run< P >::nsites()`, and `mln::p_run< P >::operator[]()`.

10.286.4.8 `template<typename P> unsigned short mln::p_run< P >::length () const [inline]`

Give the length of the run.

References `mln::p_run< P >::is_valid()`.

10.286.4.9 `template<typename P> std::size_t mln::p_run< P >::memory_size () const` `[inline]`

Return the size of this site [set](#) in memory.

10.286.4.10 `template<typename P> unsigned mln::p_run< P >::nsites () const [inline]`

Give the number of sites.

References `mln::p_run< P >::is_valid()`.

10.286.4.11 `]`

`template<typename P> P mln::p_run< P >::operator[] (unsigned short i) const [inline]`

Return the *i*-th [point](#).

References `mln::p_run< P >::is_valid()`, and `mln::point< G, C >::last_coord()`.

10.286.4.12 `template<typename P> const P & mln::p_run< P >::start () const [inline]`

Return the starting [point](#).

10.286.5 Friends And Related Function Documentation

10.286.5.1 `template<typename Sl, typename Sr> p_set< typename Sl::site > diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Set theoretic difference of lhs and rhs.

10.286.5.2 `template<typename Sl, typename Sr> p_set< typename Sl::site > inter (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Intersection between a couple of [point](#) sets.

10.286.5.3 `template<typename Sl, typename Sr> bool operator< (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Strict inclusion [test](#) between site sets lhs and rhs.

Parameters:

← *lhs* A site [set](#) (strictly included?).

← *rhs* Another site [set](#) (includer?).

10.286.5.4 `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:

↔ *ostr* An output stream.

← *set* A site [set](#).

Returns:

The modified output stream `ostr`.

10.286.5.5 `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:

← *lhs* A site [set](#) (included?).

← *rhs* Another site [set](#) (includer?).

10.286.5.6 `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:

- ← *lhs* A site [set](#).
- ← *rhs* Another site [set](#).

10.286.5.7 `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`.

10.286.5.8 `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.

10.286.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [related, inherited]

Give the unique [set](#) of `s`.

10.287 mln::p_set< P > Class Template Reference

Mathematical [set](#) of sites (based on [util::set](#)).

```
#include <p_set.hh>
```

Inherits [mln::internal::site_set_base_< P, mln::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 [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](#).
- bool [has](#) (const [psite](#) &p) const
Test if [psite](#) [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.

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 Sl, typename Sr>`
`p_set< typename Sl::site > sym_diff (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`

Set theoretic symmetrical difference of lhs and rhs.

- `template<typename Sl, typename Sr>`
`p_set< typename Sl::site > uni (const Site_Set< Sl > &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.

10.287.1 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.

10.287.2 Member Typedef Documentation

10.287.2.1 `template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_set< P >::bkd_piter`

Backward `Site_Iterator` associated type.

10.287.2.2 `template<typename P> typedef P mln::p_set< P >::element`

Element associated type.

10.287.2.3 `template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_set< P >::fwd_piter`

Forward `Site_Iterator` associated type.

10.287.2.4 `template<typename P> typedef P mln::p_set< P >::i_element`

Insertion element associated type.

10.287.2.5 `template<typename P> typedef fwd_piter mln::p_set< P >::piter`

`Site_Iterator` associated type.

10.287.2.6 `template<typename P> typedef p_indexed_psite<self_> mln::p_set< P >::psite`

Psite associated type.

10.287.2.7 `template<typename P> typedef P mln::p_set< P >::r_element`

Removal element associated type.

10.287.3 Constructor & Destructor Documentation**10.287.3.1** `template<typename P> mln::p_set< P >::p_set () [inline]`

Constructor.

10.287.4 Member Function Documentation**10.287.4.1** `template<typename P> void mln::p_set< P >::clear () [inline]`

Clear this [set](#).

10.287.4.2 `template<typename P> bool mln::p_set< P >::has (const util::index & i) const [inline]`

Test if index *i* belongs to this [point set](#).

References `mln::p_set< P >::nsites()`.

10.287.4.3 `template<typename P> bool mln::p_set< P >::has (const P & p) const [inline]`

Test if *p* belongs to this [point set](#).

10.287.4.4 `template<typename P> bool mln::p_set< P >::has (const psite & p) const [inline]`

Test if psite *p* belongs to this [point set](#).

References `mln::p_indexed_psite< S >::index()`.

10.287.4.5 `template<typename P> void mln::p_set< P >::insert (const P & p) [inline]`

Insert a site *p*.

Referenced by `mln::convert::to_p_set()`.

10.287.4.6 `template<typename P> bool mln::p_set< P >::is_valid () const [inline]`

Test this [set](#) validity so returns always true.

10.287.4.7 `template<typename P> std::size_t mln::p_set< P >::memory_size () const`
`[inline]`

Return the size of this site [set](#) in memory.

10.287.4.8 `template<typename P> unsigned mln::p_set< P >::nsites () const` `[inline]`

Give the number of sites.

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()`.

10.287.4.9 `]`

`template<typename P> const P & mln::p_set< P >::operator[] (unsigned i) const` `[inline]`

Return the *i*-th site.

References `mln::p_set< P >::nsites()`.

10.287.4.10 `template<typename P> void mln::p_set< P >::remove (const P & p)` `[inline]`

Remove a site *p*.

10.287.4.11 `template<typename P> const std::vector< P > & mln::p_set< P >::std_vector ()`
`const` `[inline]`

Return the corresponding `std::vector` of sites.

10.287.4.12 `template<typename P> const util::set< P > & mln::p_set< P >::util_set () const`
`[inline]`

Return the corresponding [util::set](#) of sites.

10.287.5 Friends And Related Function Documentation

10.287.5.1 `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*.

10.287.5.2 `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.

10.287.5.3 `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:

- ← *lhs* A site [set](#) (strictly included?).
- ← *rhs* Another site [set](#) (includer?).

10.287.5.4 `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:

- ↔ *ostr* An output stream.
- ← *set* A site [set](#).

Returns:

The modified output stream `ostr`.

10.287.5.5 `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:

- ← *lhs* A site [set](#) (included?).
- ← *rhs* Another site [set](#) (includer?).

10.287.5.6 `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:

- ← *lhs* A site [set](#).
- ← *rhs* Another site [set](#).

10.287.5.7 `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`.

10.287.5.8 `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.

10.287.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)`
[related, inherited]

Give the unique [set](#) of `s`.

10.288 mln::p_set_of< S > Class Template Reference

`p_set_of` is a [set](#) of site sets.

```
#include <p_set_of.hh>
```

Inherits `mln::internal::site_set_base_< S::site, mln::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.

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.*

10.288.1 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.

10.288.2 Member Typedef Documentation

10.288.2.1 `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.

10.288.2.2 `template<typename S> typedef S mln::p_set_of< S >::element`

Element associated type.

10.288.2.3 `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.

10.288.2.4 `template<typename S> typedef S mln::p_set_of< S >::i_element`

Insertion element associated type.

10.288.2.5 `template<typename S> typedef fwd_piter mln::p_set_of< S >::piter`

[Site_Iterator](#) associated type.

10.288.2.6 `template<typename S> typedef p_double_psite<self_, element> mln::p_set_of< S >::psite`

Psite associated type.

10.288.3 Constructor & Destructor Documentation

10.288.3.1 `template<typename S> mln::p_set_of< S >::p_set_of () [inline]`

Constructor without arguments.

10.288.4 Member Function Documentation

10.288.4.1 `template<typename S> void mln::p_set_of< S >::clear () [inline]`

Clear this [set](#).

10.288.4.2 `template<typename S> bool mln::p_set_of< S >::has (const psite & p) const`
`[inline]`

Test if *p* belongs to this [point set](#).

10.288.4.3 `template<typename S> void mln::p_set_of< S >::insert (const S & s) [inline]`

Insert a site [set](#) *s*.

10.288.4.4 `template<typename S> bool mln::p_set_of< S >::is_valid () const [inline]`

Test if this [set](#) of runs is valid.

10.288.4.5 `template<typename S> std::size_t mln::p_set_of< S >::memory_size () const`
`[inline]`

Return the size of this site [set](#) in memory.

10.288.4.6 `template<typename S> unsigned mln::p_set_of< S >::nelements () const [inline]`

Give the number of elements (site sets) of this composite.

10.288.4.7]

`template<typename S> const S & mln::p_set_of< S >::operator[] (unsigned i) const [inline]`

Return the *i*-th site [set](#).

10.288.5 Friends And Related Function Documentation

10.288.5.1 `template<typename Sl, typename Sr> p_set< typename Sl::site > diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Set theoretic difference of *lhs* and *rhs*.

10.288.5.2 `template<typename Sl, typename Sr> p_set< typename Sl::site > inter (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Intersection between a couple of [point](#) sets.

10.288.5.3 `template<typename Sl, typename Sr> bool operator< (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Strict inclusion [test](#) between site sets *lhs* and *rhs*.

Parameters:

← *lhs* A site [set](#) (strictly included?).

← *rhs* Another site [set](#) (includer?).

10.288.5.4 `template<typename S> std::ostream & operator<< (std::ostream & ostr, const Site_Set< S > & set)` [related, inherited]

Print a site `set` into the output stream `ostr`.

Parameters:

- ↔ *ostr* An output stream.
- ← *set* A site `set`.

Returns:

The modified output stream `ostr`.

10.288.5.5 `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:

- ← *lhs* A site `set` (included?).
- ← *rhs* Another site `set` (includer?).

10.288.5.6 `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:

- ← *lhs* A site `set`.
- ← *rhs* Another site `set`.

10.288.5.7 `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`.

10.288.5.8 `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.

10.288.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [related, inherited]

Give the unique `set` of `s`.

10.289 mln::p_transformed< S, F > Class Template Reference

[Site set](#) transformed through a function.

```
#include <p_transformed.hh>
```

Inherits mln::internal::site_set_base_< S::psite, mln::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](#).

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 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.

10.289.1 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.

10.289.2 Member Typedef Documentation

10.289.2.1 `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.

10.289.2.2 `template<typename S, typename F> typedef S ::element mln::p_transformed< S, F
>::element`

Element associated type.

10.289.2.3 `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.

10.289.2.4 `template<typename S, typename F> typedef fwd_piter mln::p_transformed< S, F
>::piter`

[Site_Iterator](#) associated type.

10.289.2.5 `template<typename S, typename F> typedef S ::psite mln::p_transformed< S, F
>::psite`

Psite associated type.

10.289.3 Constructor & Destructor Documentation

10.289.3.1 `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`.

10.289.3.2 `template<typename S, typename F> mln::p_transformed< S, F >::p_transformed ()
[inline]`

Constructor without argument.

10.289.4 Member Function Documentation

10.289.4.1 `template<typename S, typename F> const F & mln::p_transformed< S, F >::function
() const [inline]`

Return the transformation function.

10.289.4.2 `template<typename S, typename F> bool mln::p_transformed< S, F >::has (const psite & p) const` [inline]

Test if `p` belongs to the subset.

10.289.4.3 `template<typename S, typename F> bool mln::p_transformed< S, F >::is_valid () const` [inline]

Test if this site `set` is valid.

10.289.4.4 `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.

10.289.4.5 `template<typename S, typename F> const S & mln::p_transformed< S, F >::primary_set () const` [inline]

Return the primary `set`.

Referenced by `mln::p_transformed_piter< Pi, S, F >::change_target()`.

10.289.5 Friends And Related Function Documentation

10.289.5.1 `template<typename Sl, typename Sr> p_set< typename Sl::site > diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Set theoretic difference of `lhs` and `rhs`.

10.289.5.2 `template<typename Sl, typename Sr> p_set< typename Sl::site > inter (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Intersection between a couple of `point` sets.

10.289.5.3 `template<typename Sl, typename Sr> bool operator< (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related, inherited]

Strict inclusion `test` between site sets `lhs` and `rhs`.

Parameters:

← *lhs* A site `set` (strictly included?).

← *rhs* Another site `set` (includer?).

10.289.5.4 `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:

- ↔ *ostr* An output stream.
- ← *set* A site [set](#).

Returns:

The modified output stream `ostr`.

10.289.5.5 `template<typename SI, typename Sr> bool operator<= (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [[related](#), [inherited](#)]

Inclusion [test](#) between site sets `lhs` and `rhs`.

Parameters:

- ← *lhs* A site [set](#) (included?).
- ← *rhs* Another site [set](#) (includer?).

10.289.5.6 `template<typename SI, typename Sr> bool operator== (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [[related](#), [inherited](#)]

Equality [test](#) between site sets `lhs` and `rhs`.

Parameters:

- ← *lhs* A site [set](#).
- ← *rhs* Another site [set](#).

10.289.5.7 `template<typename SI, typename Sr> p_set< typename SI::site > sym_diff (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [[related](#), [inherited](#)]

Set theoretic symmetrical difference of `lhs` and `rhs`.

10.289.5.8 `template<typename SI, typename Sr> p_set< typename SI::site > uni (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [[related](#), [inherited](#)]

Union of a couple of [point](#) sets.

10.289.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [[related](#), [inherited](#)]

Give the unique [set](#) of `s`.

10.290 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< mln::p_transformed< S, F >, mln::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.

10.290.1 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](#)

10.290.2 Constructor & Destructor Documentation

10.290.2.1 `template<typename Pi, typename S, typename F> mln::p_transformed_piter< Pi, S, F >::p_transformed_piter () [inline]`

Constructor without argument.

10.290.2.2 `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](#).

References [mln::p_transformed_piter< Pi, S, F >::change_target\(\)](#).

10.290.3 Member Function Documentation

10.290.3.1 `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.

References `mln::p_transformed< S, F >::primary_set()`.

Referenced by `mln::p_transformed_piter< Pi, S, F >::p_transformed_piter()`.

10.290.3.2 `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.

10.291 mln::p_vaccess< V, S > Class Template Reference

[Site set](#) in which sites are grouped by their associated [value](#).

```
#include <p_vaccess.hh>
```

Inherits mln::internal::site_set_base_< S::site, mln::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 V &v, const typename S::psite &p) const
Test if the couple (value v, psite p) belongs to this site [set](#).
- bool [has](#) (const [psite](#) &p) const
Test if p belongs to this site [set](#).
- void [insert](#) (const V &v, const [element](#) &e)
Insert e at value v.
- void [insert](#) (const [i_element](#) &v_e)

Insert a pair v_e (value v , element e).

- 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.

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 Sl, typename Sr>`
`p_set< typename Sl::site > uni` (const [Site_Set](#)< Sl > &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.

10.291.1 Detailed Description

`template<typename V, typename S> class mln::p_vaccess< V, S >`

[Site set](#) in which sites are grouped by their associated [value](#).

10.291.2 Member Typedef Documentation

10.291.2.1 `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.

10.291.2.2 `template<typename V, typename S> typedef S ::element mln::p_vaccess< V, S
>::element`

Element associated type.

10.291.2.3 `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.

10.291.2.4 `template<typename V, typename S> typedef std::pair<V, element> mln::p_vaccess<
V, S >::i_element`

Insertion element associated type.

10.291.2.5 `template<typename V, typename S> typedef fwd_piter mln::p_vaccess< V, S >::piter`

[Site_Iterator](#) associated type.

10.291.2.6 `template<typename V, typename S> typedef S mln::p_vaccess< V, S >::pset`

Inner [site set](#) associated type.

10.291.2.7 `template<typename V, typename S> typedef p_double_psite<self_, S>
mln::p_vaccess< V, S >::psite`

Psite associated type.

10.291.2.8 `template<typename V, typename S> typedef V mln::p_vaccess< V, S >::value`

Value associated type.

10.291.2.9 `template<typename V, typename S> typedef mln::value::set<V> mln::p_vaccess< V,
S >::vset`

Value_Set associated type.

10.291.3 Constructor & Destructor Documentation

10.291.3.1 `template<typename V, typename S> mln::p_vaccess< V, S >::p_vaccess ()
[inline]`

Constructor.

10.291.4 Member Function Documentation

10.291.4.1 `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.

10.291.4.2 `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.

10.291.4.3 `template<typename V, typename S> void mln::p_vaccess< V, S >::insert (const V & v,
const element & e) [inline]`

Insert e at value v.

10.291.4.4 `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).

10.291.4.5 `template<typename V, typename S> bool mln::p_vaccess< V, S >::is_valid () const
[inline]`

Test if this site set is valid.

10.291.4.6 `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.

10.291.4.7 `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`.

10.291.4.8 `template<typename V, typename S> const mln::value::set< V > & mln::p_vaccess< V, S >::values() const` `[inline]`

Give the [set](#) of values.

10.291.5 Friends And Related Function Documentation

10.291.5.1 `template<typename Sl, typename Sr> p_set< typename Sl::site > diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` `[related, inherited]`

Set theoretic difference of `lhs` and `rhs`.

10.291.5.2 `template<typename Sl, typename Sr> p_set< typename Sl::site > inter (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` `[related, inherited]`

Intersection between a couple of [point](#) sets.

10.291.5.3 `template<typename Sl, typename Sr> bool operator< (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` `[related, inherited]`

Strict inclusion [test](#) between site sets `lhs` and `rhs`.

Parameters:

- ← *lhs* A site [set](#) (strictly included?).
- ← *rhs* Another site [set](#) (includer?).

10.291.5.4 `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:

- ↔ *ostr* An output stream.
- ← *set* A site [set](#).

Returns:

The modified output stream `ostr`.

10.291.5.5 `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:

- ← *lhs* A site [set](#) (included?).
- ← *rhs* Another site [set](#) (includer?).

10.291.5.6 `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:

- ← *lhs* A site [set](#).
- ← *rhs* Another site [set](#).

10.291.5.7 `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`.

10.291.5.8 `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.

10.291.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [related, inherited]

Give the unique [set](#) of `s`.

10.292 mln::p_vertices< G, F > Class Template Reference

Site set based mapping [graph](#) vertices to sites.

```
#include <p_vertices.hh>
```

Inherits [mln::internal::site_set_base_< F::result, mln::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 [p_graph_piter< self_, mln_vertex_bkd_iter\(G\) >](#) [bkd_piter](#)
Backward [Site_Iterator](#) associated type.
- typedef [super_::site](#) [element](#)
Associated types.
- typedef [p_graph_piter< self_, mln_vertex_fwd_iter\(G\) >](#) [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef [fwd_piter](#) [piter](#)
[Site_Iterator](#) associated type.
- typedef [p_vertices_psite< G, F >](#) [psite](#)
[Point_Site](#) associated type.

Public Member Functions

- [template<typename G2>](#)
[bool has](#) (const [util::vertex< G2 >](#) &v) const
Does this [site set](#) has v?
- [bool has](#) (const [psite](#) &p) const
Does this [site set](#) has p?
- 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](#).
- `template<typename F2>`
`p_vertices (const p_vertices< G, F2 > &other)`
Copy constructor.
- `template<typename F2>`
`p_vertices (const Graph< G > &gr, const Function< F2 > &f)`
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.
- `p_vertices (const Graph< G > &gr)`
Construct a [graph](#) psite [set](#) from a [graph](#) of points.
- `p_vertices ()`
Constructor without argument.
- `const F & function () const`
Return the association function.
- `const G & graph () const`
Accessors.
- `F::result operator() (const psite &p) const`
Return the [value](#) associated to an element of this site [set](#).

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.

10.292.1 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.

10.292.2 Member Typedef Documentation

10.292.2.1 `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.

10.292.2.2 `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.

10.292.2.3 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef F mln::p_vertices< G, F >::fun_t`

Function associated type.

10.292.2.4 `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.

10.292.2.5 `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.

10.292.2.6 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef G mln::p_vertices< G, F >::graph_t`

[Graph](#) associated type.

10.292.2.7 `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.

10.292.2.8 `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.

10.292.2.9 `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.

10.292.3 Constructor & Destructor Documentation

10.292.3.1 `template<typename G, typename F> mln::p_vertices< G, F >::p_vertices ()
[inline]`

Constructor without argument.

10.292.3.2 `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.

References `mln::p_vertices< G, F >::is_valid()`.

10.292.3.3 `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.

References `mln::p_vertices< G, F >::is_valid()`.

10.292.3.4 `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`.

References `mln::p_vertices< G, F >::is_valid()`.

10.292.3.5 `template<typename G, typename F> template<typename F2> mln::p_vertices< G, F >::p_vertices (const p_vertices< G, F2 > & other) [inline]`

Copy constructor.

References `mln::p_vertices< G, F >::function()`, `mln::p_vertices< G, F >::graph()`, and `mln::p_vertices< G, F >::is_valid()`.

10.292.4 Member Function Documentation

10.292.4.1 `template<typename G, typename F> const F & mln::p_vertices< G, F >::function () const [inline]`

Return the association function.

Referenced by `mln::p_vertices< G, F >::p_vertices()`.

10.292.4.2 `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)

References `mln::p_vertices< G, F >::is_valid()`.

Referenced by `mln::debug::draw_graph()`, `mln::operator==()`, and `mln::p_vertices< G, F >::p_vertices()`.

10.292.4.3 `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*?

References `mln::util::vertex< G >::graph()`, `mln::util::vertex< G >::is_valid()`, and `mln::p_vertices< G, F >::is_valid()`.

10.292.4.4 `template<typename G, typename F> bool mln::p_vertices< G, F >::has (const psite & p) const [inline]`

Does this site [set](#) has *p*?

References `mln::p_vertices< G, F >::is_valid()`.

10.292.4.5 `template<typename G, typename F> void mln::p_vertices< G, F >::invalidate () [inline]`

Invalidate this site [set](#).

10.292.4.6 `template<typename G, typename F> bool mln::p_vertices< G, F >::is_valid () const [inline]`

Test this site [set](#) validity.

Referenced by `mln::p_vertices< G, F >::graph()`, `mln::p_vertices< G, F >::has()`, and `mln::p_vertices< G, F >::p_vertices()`.

10.292.4.7 `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;`

10.292.4.8 `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.

References `mln::p_vertices< G, F >::nvertices()`.

10.292.4.9 `template<typename G, typename F> unsigned mln::p_vertices< G, F >::nvertices () const [inline]`

Return The number of vertices in the [graph](#).

Referenced by `mln::p_vertices< G, F >::nsites()`.

10.292.4.10 `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](#).

10.292.5 Friends And Related Function Documentation

10.292.5.1 `template<typename Sl, typename Sr> p_set< typename Sl::site > diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Set theoretic difference of `lhs` and `rhs`.

10.292.5.2 `template<typename Sl, typename Sr> p_set< typename Sl::site > inter (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Intersection between a couple of [point](#) sets.

10.292.5.3 `template<typename Sl, typename Sr> bool operator< (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs) [related, inherited]`

Strict inclusion [test](#) between site sets `lhs` and `rhs`.

Parameters:

← *lhs* A site [set](#) (strictly included?).

← *rhs* Another site [set](#) (includer?).

10.292.5.4 `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:

↔ *ostr* An output stream.

← *set* A site [set](#).

Returns:

The modified output stream `ostr`.

10.292.5.5 `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:

- ← *lhs* A site [set](#) (included?).
- ← *rhs* Another site [set](#) (includer?).

10.292.5.6 `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:

- ← *lhs* A site [set](#).
- ← *rhs* Another site [set](#).

10.292.5.7 `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`.

10.292.5.8 `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.

10.292.5.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [related, inherited]

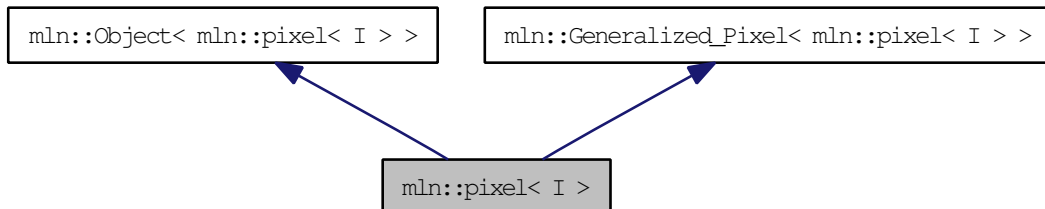
Give the unique [set](#) of `s`.

10.293 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, const typename I::psite &p)
Constructor.
- [pixel](#) (I &image)
Constructor.

10.293.1 Detailed Description

```
template<typename I> struct mln::pixel< I >
```

Generic [pixel](#) class.

The parameter is I the type of the image it belongs to.

10.293.2 Constructor & Destructor Documentation

10.293.2.1 `template<typename I> mln::pixel< I >::pixel (I & image)` `[inline]`

Constructor.

10.293.2.2 `template<typename I> mln::pixel< I >::pixel (I & image, const typename I::psite & p)` `[inline]`

Constructor.

References `mln::pixel< I >::change_to()`.

10.293.3 Member Function Documentation

10.293.3.1 `template<typename I> void mln::pixel< I >::change_to (const typename I::psite & p)`
[inline]

Change the [pixel](#) to the one at [point](#) p.

Referenced by mln::pixel< I >::pixel().

10.293.3.2 `template<typename I> bool mln::pixel< I >::is_valid () const` [inline]

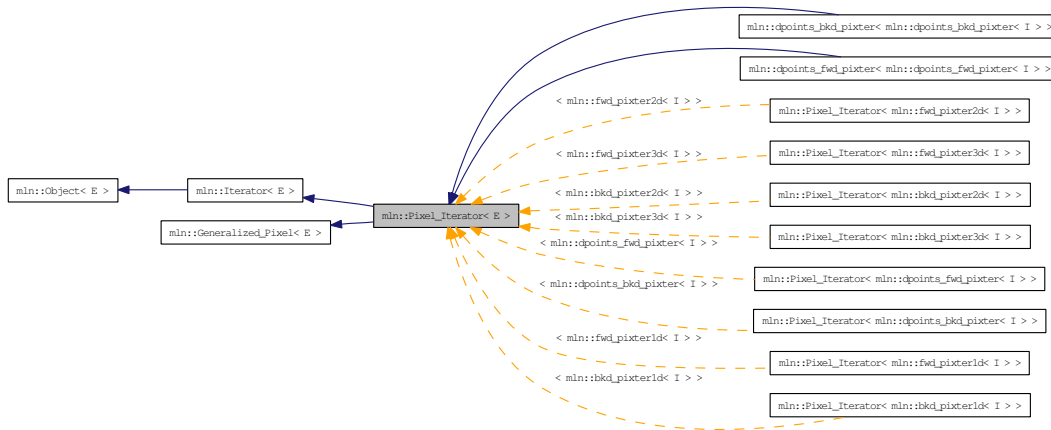
Test if this [pixel](#) is valid.

10.294 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.

10.294.1 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.

10.294.2 Member Function Documentation

10.294.2.1 `template<typename E> void mln::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.

10.295 mln::plain< I > Class Template Reference

Prevents an image from sharing its [data](#).

```
#include <plain.hh>
```

Inherits mln::internal::image_identity< I, I::domain_t, mln::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 I &ima)
Copy constructor from an image ima.
- [plain](#) (const [plain](#)< I > &rhs)
Copy constructor.
- [plain](#) ()
Constructor without argument.

10.295.1 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.

10.295.2 Member Typedef Documentation

10.295.2.1 `template<typename I> typedef plain< tag::image_<I> > mln::plain< I >::skeleton`

Skeleton.

10.295.3 Constructor & Destructor Documentation

10.295.3.1 `template<typename I> mln::plain< I >::plain ()` [inline]

Constructor without argument.

10.295.3.2 `template<typename I> mln::plain< I >::plain (const plain< I > & rhs)` [inline]

Copy constructor.

10.295.3.3 `template<typename I> mln::plain< I >::plain (const I & ima)` [inline]

Copy constructor from an image *ima*.

10.295.4 Member Function Documentation

10.295.4.1 `template<typename I> mln::plain< I >::operator I () const` [inline]

Conversion into an image with type *I*.

References `mln::duplicate()`.

10.295.4.2 `template<typename I> plain< I > & mln::plain< I >::operator= (const I & ima)`
[inline]

Assignment operator from an image *ima*.

10.295.4.3 `template<typename I> plain< I > & mln::plain< I >::operator= (const plain< I > & rhs)` [inline]

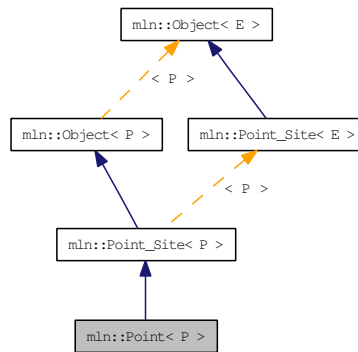
Assignment operator.

10.296 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.

10.296.1 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.

10.296.2 Member Typedef Documentation

10.296.2.1 `template<typename P> typedef P mln::Point< P >::point`

The associated [point](#) type is itself.

10.296.3 Member Function Documentation

10.296.3.1 `template<typename P> const P & mln::Point< P >::to_point () const [inline]`

It is a [Point](#) so it returns itself.

10.296.4 Friends And Related Function Documentation

10.296.4.1 `template<typename P, typename D> P & operator+= (Point< P > & p, const Dpoint< D > & dp) [related]`

Shift a [point](#) by a delta-point `dp`.

Parameters:

↔ *p* The targeted [point](#).

← *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`.

10.296.4.2 `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:

↔ *p* The targeted [point](#).

← *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 .

10.296.4.3 `template<typename P, typename D> P & operator/ (Point< P > & p, const value::Scalar< D > & dp)` [related]

Divide a [point](#) by a scalar s .

Parameters:

↔ p The targeted [point](#).

← dp A scalar.

Returns:

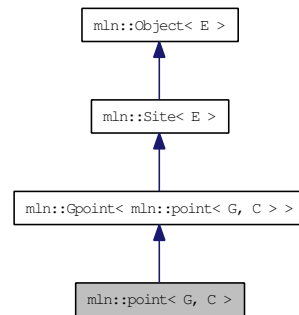
A reference to the [point](#) p once divided by s .

10.297 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 `algebra::h_vec< G::dim, float > h_vec`
Algebra hexagonal vector (hvec) associated type.
- typedef `algebra::vec< G::dim, float > vec`
Algebra vector (vec) associated type.

Public Member Functions

- `C & last_coord ()`
Read-write access to the last coordinate.
- `const C & last_coord () const`
Read-only 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.

- `template<typename C2>`
`point (const algebra::vec< dim, C2 > &v)`

Constructor from an algebra vector.

- `point ()`

Constructor without argument.

- `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 (const literal::origin_t &)`

Constructors/assignments with literals.

- `point (C ind)`

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).

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.

10.297.1 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.

10.297.2 Member Typedef Documentation

10.297.2.1 `template<typename G, typename C> typedef C mln::point< G, C >::coord`

Coordinate associated type.

10.297.2.2 `template<typename G, typename C> typedef dpoint<G,C> mln::point< G, C >::delta`

Delta associated type.

10.297.2.3 `template<typename G, typename C> typedef dpoint<G,C> mln::point< G, C >::dpsite`

DPsite associated type.

10.297.2.4 `template<typename G, typename C> typedef G mln::point< G, C >::grid`

Grid associated type.

10.297.2.5 `template<typename G, typename C> typedef algebra::h_vec<G::dim, float> mln::point< G, C >::h_vec`

Algebra hexagonal vector (hvec) associated type.

10.297.2.6 `template<typename G, typename C> typedef algebra::vec<G::dim, float> mln::point< G, C >::vec`

Algebra vector (vec) associated type.

10.297.3 Member Enumeration Documentation

10.297.3.1 `template<typename G, typename C> anonymous enum`

Enumerator:

dim Dimension of the space.

Invariant:

`dim > 0`

10.297.4 Constructor & Destructor Documentation

10.297.4.1 `template<typename G, typename C> mln::point< G, C >::point () [inline]`

Constructor without argument.

10.297.4.2 `template<typename G, typename C> template<typename C2> mln::point< G, C >::point (const algebra::vec< dim, C2 > & v) [inline]`

Constructor from an [algebra](#) vector.

References mln::point< G, C >::dim.

10.297.4.3 `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.

10.297.4.4 `template<typename G, typename C> mln::point< G, C >::point (const literal::origin_t &) [inline]`

Constructors/assignments with literals.

10.297.4.5 `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*.

References mln::point< G, C >::dim.

10.297.5 Member Function Documentation

10.297.5.1 `template<typename G, typename C> C & mln::point< G, C >::last_coord () [inline]`

Read-write access to the last coordinate.

References mln::point< G, C >::dim.

10.297.5.2 `template<typename G, typename C> const C & mln::point< G, C >::last_coord () const [inline]`

Read-only access to the last coordinate.

References mln::point< G, C >::dim.

Referenced by mln::p_run< P >::end(), mln::p_run< P >::operator[](), and mln::debug::put_word().

10.297.5.3 `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](#).

10.297.5.4 `template<typename G, typename C> point< G, C > & mln::point< G, C >::operator+= (const delta & dp) [inline]`

Shifting by *dp*.

References `mln::point< G, C >::dim`.

10.297.5.5 `template<typename G, typename C> point< G, C > & mln::point< G, C >::operator=(const delta & dp) [inline]`

Shifting by `the` inverse of `dp`.

References `mln::point< G, C >::dim`.

10.297.5.6]

`template<typename G, typename C> C & mln::point< G, C >::operator[] (unsigned i) [inline]`

Read-write access to the `i`-th coordinate [value](#).

Parameters:

← `i` The coordinate index.

Precondition:

`i < dim`

References `mln::point< G, C >::dim`.

10.297.5.7]

`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:

← `i` The coordinate index.

Precondition:

`i < dim`

References `mln::point< G, C >::dim`.

10.297.5.8 `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](#).

10.297.5.9 `template<typename G, typename C> void mln::point< G, C >::set_all (C c) [inline]`

Set all coordinates to the [value](#) `c`.

10.297.5.10 `template<typename G, typename C> point< G, C >::h_vec mln::point< G, C >::to_h_vec () const` [inline]

Transform to [point](#) in homogene coordinate system.

References mln::point< G, C >::dim.

10.297.5.11 `template<typename G, typename C> point< G, C >::vec mln::point< G, C >::to_vec () const` [inline]

Explicit conversion towards mln::algebra::vec.

References mln::point< G, C >::dim.

Referenced by mln::io::magick::load(), mln::io::dicom::load(), and mln::io::magick::save().

10.297.6 Friends And Related Function Documentation

10.297.6.1 `template<typename P, typename D> P operator+ (const Gpoint< P > & p, const Gdpoint< D > & dp)` [related, inherited]

Add a delta-point rhs to a [grid point](#) lhs.

Parameters:

← *p* A [grid point](#).

← *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](#)

10.297.6.2 `template<typename P, typename D> P & operator+= (Gpoint< P > & p, const Gdpoint< D > & dp)` [related, inherited]

Shift a [point](#) by a delta-point *dp*.

Parameters:

↔ *p* The targeted [point](#).

← *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*.

10.297.6.3 `template<typename L, typename R> L::delta operator- (const Gpoint< L > & lhs, const Gpoint< R > & rhs)` [related, inherited]

Difference between a couple of [grid point](#) lhs and rhs.

Parameters:

← *lhs* A first [grid point](#).

← *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](#)

10.297.6.4 `template<typename P, typename D> P & operator-= (Gpoint< P > & p, const Gdpoint< D > & dp)` [related, inherited]

Shift a [point](#) by the negate of a delta-point dp.

Parameters:

↔ *p* The targeted [point](#).

← *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.

10.297.6.5 `template<typename P, typename D> P operator/ (const Gpoint< P > & p, const value::scalar_< D > & dp)` [related, inherited]

Divide a [point](#) by a scalar *s*.

Parameters:

- ↔ *p* The targeted [point](#).
- ← *dp* A scalar.

Returns:

A reference to the [point](#) *p* once divided by *s*.

10.297.6.6 `template<typename P> std::ostream & operator<< (std::ostream & ostr, const Gpoint< P > & p)` [related, inherited]

Print a [grid point](#) *p* into the output stream *ostr*.

Parameters:

- ↔ *ostr* An output stream.
- ← *p* A [grid point](#).

Returns:

The modified output stream *ostr*.

References `mln::debug::format()`.

10.297.6.7 `template<typename L, typename R> bool operator== (const Gpoint< L > & lhs, const Gpoint< R > & rhs)` [related, inherited]

Equality comparison between a couple of [grid point](#) *lhs* and *rhs*.

Parameters:

- ← *lhs* A first [grid point](#).
- ← *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.

10.297.7 Member Data Documentation

10.297.7.1 `template<typename G, typename C> const point< G, C > mln::point< G, C >::origin = all_to(0)` [inline, static]

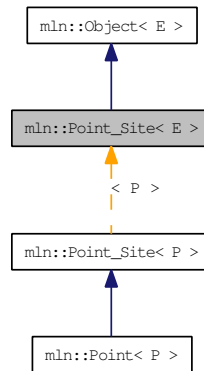
Origin [point](#) (all coordinates are 0).

10.298 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)
`}`

10.298.1 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).

10.298.2 Friends And Related Function Documentation

10.298.2.1 `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:

- ← *p* A [point](#) site.
- ← *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](#)

10.298.2.2 `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:

- ← *p* A [point](#) site.
- ← *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::Dpoint](#)
[mln::Delta_Point_Site](#)

10.298.2.3 `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:

- ← *lhs* A first [point](#) site.
- ← *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](#)

10.298.2.4 `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:

- ↔ *ostr* An output stream.
- ← *p* A [point](#) site.

Returns:

The modified output stream `ostr`.

10.298.2.5 `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:

- ← *lhs* A first [point](#) site.

← *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.

10.299 mln::Point_Site< void > Struct Template Reference

[Point](#) site category flag type.

```
#include <point_site.hh>
```

10.299.1 Detailed Description

`template<> struct mln::Point_Site< void >`

[Point](#) site category flag type.

10.300 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 >](#), [mln::Site_Proxy< E >](#), [mln::util::array_bkd_iter< T >](#), [mln::util::array_fwd_iter< T >](#), [mln::util::set_bkd_iter< T >](#), [mln::util::set_fwd_iter< T >](#), [mln::util::timer](#), [mln::value::proxy< I >](#), and [mln::value::shell< F, I >](#).

10.300.1 Detailed Description

```
template<typename E> struct mln::Proxy< E >
```

Base class for implementation classes of the notion of "proxy".

10.301 mln::Proxy< void > Struct Template Reference

[Proxy](#) category flag type.

```
#include <proxy.hh>
```

10.301.1 Detailed Description

template<> struct mln::Proxy< void >

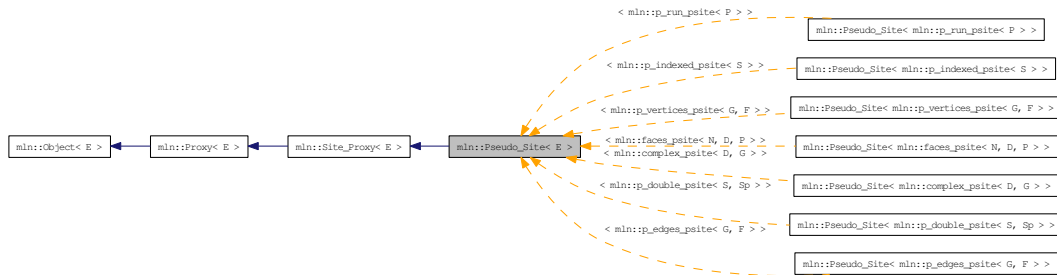
[Proxy](#) category flag type.

10.302 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 >:



10.302.1 Detailed Description

```
template<typename E> struct mln::Pseudo_Site< E >
```

Base class for implementation classes of the notion of "pseudo site".

FIXME: Explain...

10.303 mln::Pseudo_Site< void > Struct Template Reference

[Pseudo_Site](#) category flag type.

```
#include <pseudo_site.hh>
```

10.303.1 Detailed Description

`template<> struct mln::Pseudo_Site< void >`

[Pseudo_Site](#) category flag type.

10.304 mln::pw::image< F, S > Class Template Reference

A generic point-wise [image](#) implementation.

```
#include <image.hh>
```

Inherits mln::pw::internal::image_base< F, S, mln::pw::image< F, S > >.

Public Types

- typedef [image](#)< tag::function_< F >, tag::domain_< S > > [skeleton](#)
Skeleton.

Public Member Functions

- [image](#) (const [Function_v2v](#)< F > &f, const [Site_Set](#)< S > &ps)
Constructor.
- [image](#) ()
Constructor without argument.

10.304.1 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.

10.304.2 Member Typedef Documentation

10.304.2.1 `template<typename F, typename S> typedef image< tag::function_<F>, tag::domain_<S> > mln::pw::image< F, S >::skeleton`

Skeleton.

10.304.3 Constructor & Destructor Documentation

10.304.3.1 `template<typename F, typename S> mln::pw::image< F, S >::image () [inline]`

Constructor without argument.

10.304.3.2 `template<typename F, typename S> mln::pw::image< F, S >::image (const Function_v2v< F > &f, const Site_Set< S > &ps) [inline]`

Constructor.

10.305 mln::registration::closest_point_basic< P > Class Template Reference

Closest [point](#) functor based on map distance.

```
#include <icp.hh>
```

10.305.1 Detailed Description

```
template<typename P> class mln::registration::closest_point_basic< P >
```

Closest [point](#) functor based on map distance.

10.306 mln::registration::closest_point_with_map< P > Class Template Reference

Closest [point](#) functor based on map distance.

```
#include <icp.hh>
```

10.306.1 Detailed Description

```
template<typename P> class mln::registration::closest_point_with_map< P >
```

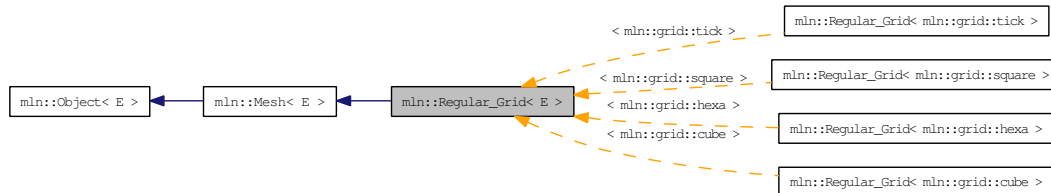
Closest [point](#) functor based on map distance.

10.307 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 >:



10.307.1 Detailed Description

```
template<typename E> struct mln::Regular_Grid< E >
```

Base class for implementation classes of regular grids.

10.308 mln::safe_image< I > Class Template Reference

Makes an image accessible at undefined location.

```
#include <safe.hh>
```

Inherits mln::internal::image_identity< I, I::domain_t, mln::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.

10.308.1 Detailed Description

```
template<typename I> class mln::safe_image< I >
```

Makes an image accessible at undefined location.

10.308.2 Member Typedef Documentation

10.308.2.1 `template<typename I> typedef safe_image< tag::image_<I> > mln::safe_image< I >::skeleton`

Skeleton.

10.308.3 Member Function Documentation

10.308.3.1 `template<typename I> mln::safe_image< I >::operator safe_image< const I > () const [inline]`

Const promotion via conversion.

10.309 mln::select::p_of< P > Struct Template Reference

Structure [p_of](#).

```
#include <pix.hh>
```

10.309.1 Detailed Description

```
template<typename P> struct mln::select::p_of< P >
```

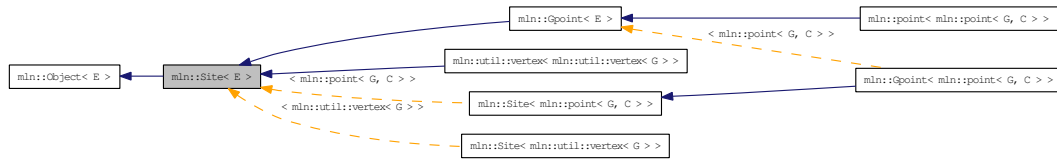
Structure [p_of](#).

10.310 mln::Site< E > Struct Template Reference

Base class for classes that are explicitly sites.

```
#include <site.hh>
```

Inheritance diagram for mln::Site< E >:



10.310.1 Detailed Description

```
template<typename E> struct mln::Site< E >
```

Base class for classes that are explicitly sites.

10.311 mln::Site< void > Struct Template Reference

[Site](#) category flag type.

```
#include <site.hh>
```

10.311.1 Detailed Description

template<> struct mln::Site< void >

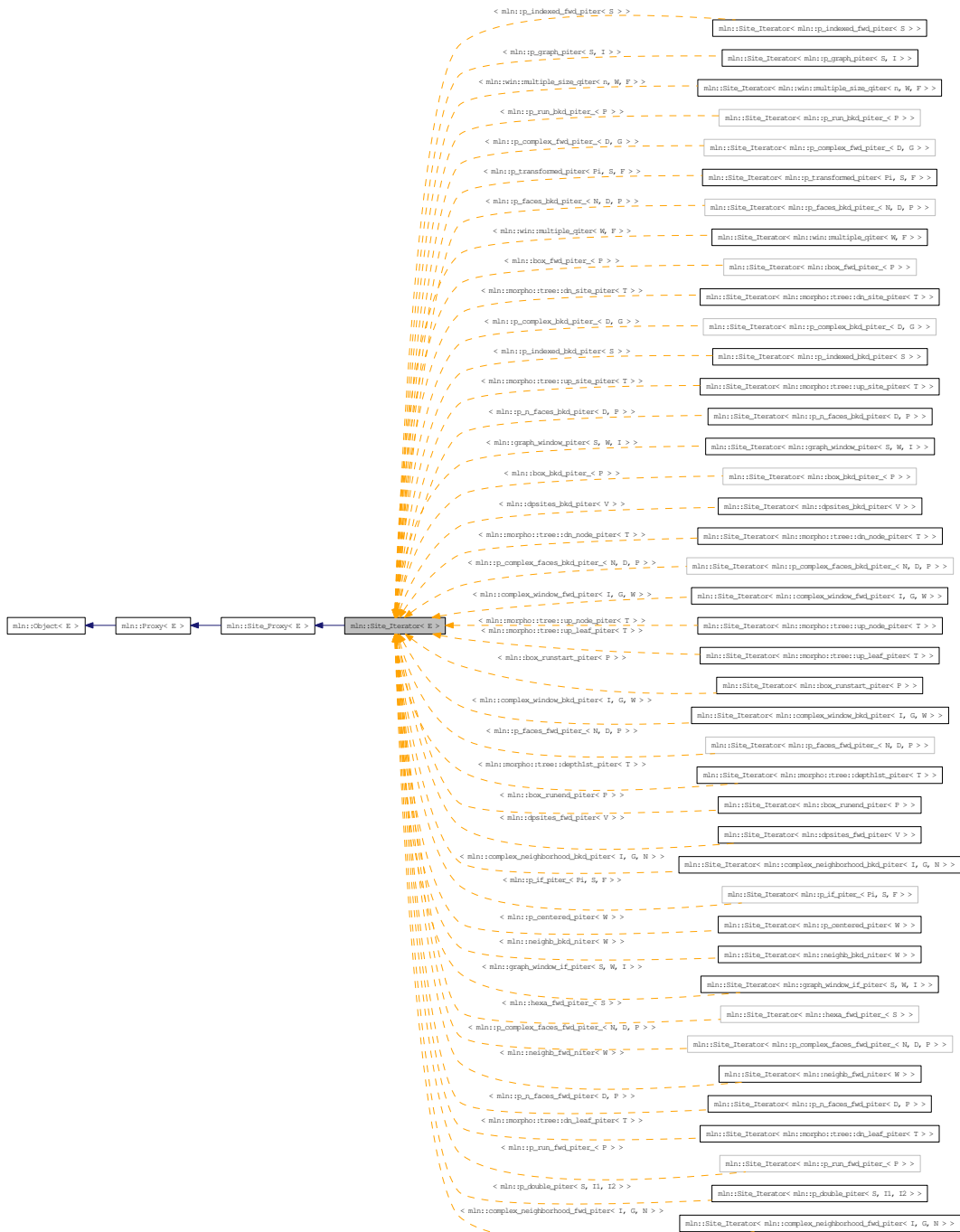
[Site](#) category flag type.

10.312 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.

10.312.1 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.

10.312.2 Member Function Documentation

10.312.2.1 `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.

10.313 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 >](#).

10.313.1 Detailed Description

```
template<typename E> struct mln::Site_Proxy< E >
```

Base class for implementation classes of the notion of "site proxy".

FIXME: Explain...

10.314 mln::Site_Proxy< void > Struct Template Reference

[Site_Proxy](#) category flag type.

```
#include <site_proxy.hh>
```

10.314.1 Detailed Description

`template<> struct mln::Site_Proxy< void >`

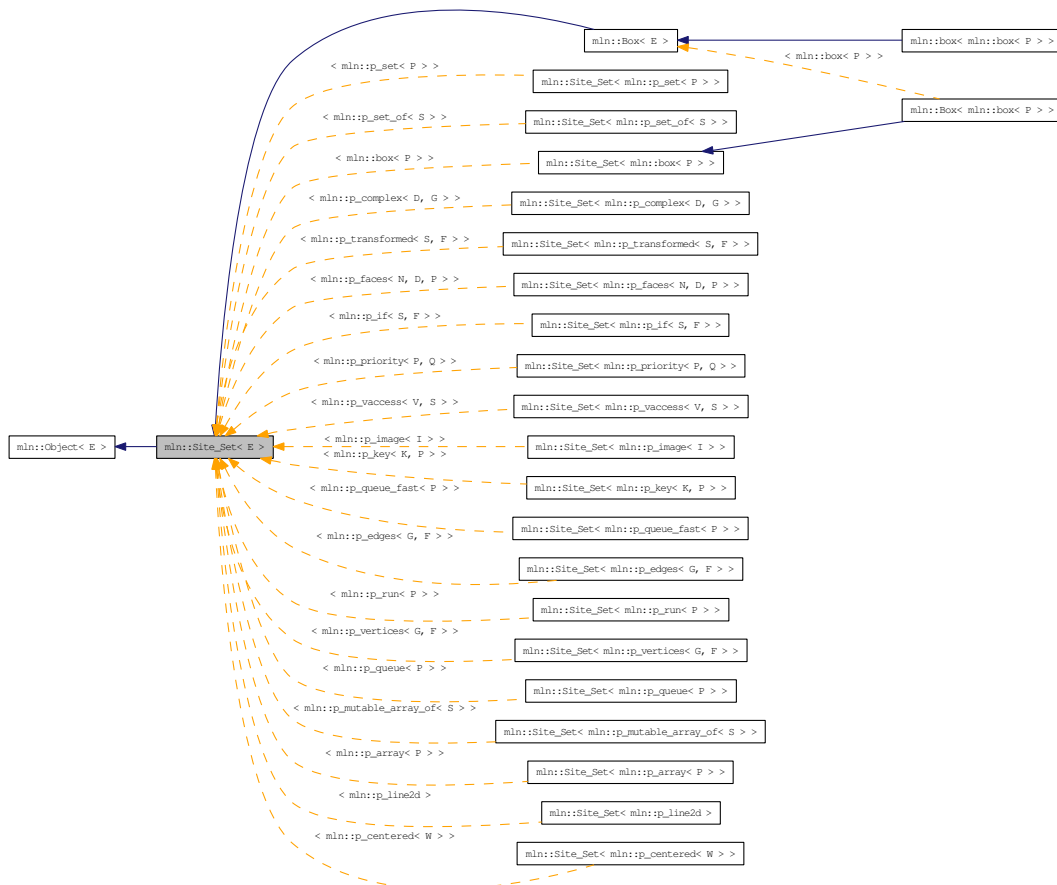
[Site_Proxy](#) category flag type.

10.315 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 Sl, typename Sr>`
`bool operator<= (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Inclusion test between site sets lhs and rhs.
- `template<typename Sl, typename Sr>`
`bool operator== (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Equality test between site sets lhs and rhs.
- `template<typename Sl, typename Sr>`
`p_set< typename Sl::site > sym_diff (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic symmetrical difference of lhs and rhs.
- `template<typename Sl, typename Sr>`
`p_set< typename Sl::site > uni (const Site_Set< Sl > &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.

10.315.1 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.

10.315.2 Friends And Related Function Documentation

10.315.2.1 `template<typename Sl, typename Sr> p_set< typename Sl::site > diff (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)` [related]

Set theoretic difference of lhs and rhs.

10.315.2.2 `template<typename Sl, typename Sr> p_set< typename Sl::site > inter (const Site_Set< Sl > &lhs, const Site_Set< Sr > &rhs)` [related]

Intersection between a couple of point sets.

10.315.2.3 `template<typename Sl, typename Sr> bool operator< (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related]

Strict inclusion [test](#) between site sets `lhs` and `rhs`.

Parameters:

- ← *lhs* A site [set](#) (strictly included?).
- ← *rhs* Another site [set](#) (includer?).

10.315.2.4 `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:

- ↔ *ostr* An output stream.
- ← *set* A site [set](#).

Returns:

The modified output stream `ostr`.

10.315.2.5 `template<typename Sl, typename Sr> bool operator<= (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related]

Inclusion [test](#) between site sets `lhs` and `rhs`.

Parameters:

- ← *lhs* A site [set](#) (included?).
- ← *rhs* Another site [set](#) (includer?).

10.315.2.6 `template<typename Sl, typename Sr> bool operator== (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related]

Equality [test](#) between site sets `lhs` and `rhs`.

Parameters:

- ← *lhs* A site [set](#).
- ← *rhs* Another site [set](#).

10.315.2.7 `template<typename Sl, typename Sr> p_set< typename Sl::site > sym_diff (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related]

Set theoretic symmetrical difference of `lhs` and `rhs`.

10.315.2.8 `template<typename Sl, typename Sr> p_set< typename Sl::site > uni (const Site_Set< Sl > & lhs, const Site_Set< Sr > & rhs)` [related]

Union of a couple of [point](#) sets.

10.315.2.9 `template<typename S> p_set< typename S::site > unique (const Site_Set< S > & s)` [related]

Give the unique [set](#) of s.

10.316 mln::Site_Set< void > Struct Template Reference

[Site_Set](#) category flag type.

```
#include <site_set.hh>
```

10.316.1 Detailed Description

`template<> struct mln::Site_Set< void >`

[Site_Set](#) category flag type.

10.317 mln::slice_image< I > Struct Template Reference

2D image extracted from a slice of a 3D image.

```
#include <slice_image.hh>
```

Inherits mln::internal::image_domain_morpher< I, mln::box, mln::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.
- internal::morpher_lvalue_< I >::ret [operator](#)() (const [point2d](#) &p)
Read-write access to the image [value](#) located at [point](#) p.
- I::rvalue [operator](#)() (const [point2d](#) &p) const
Read-only access to the image [value](#) located at [point](#) p.
- [def::coord sli](#) () const
Give the slice number.
- [slice_image](#) (I &ima, [def::coord sli](#))
Constructor from an image [ima](#) and a predicate [f](#).
- [slice_image](#) ()
Constructor without argument.

10.317.1 Detailed Description

```
template<typename I> struct mln::slice_image< I >
```

2D image extracted from a slice of a 3D image.

10.317.2 Member Typedef Documentation

10.317.2.1 `template<typename I> typedef slice_image< tag::image_<I> > mln::slice_image< I >::skeleton`

Skeleton.

10.317.3 Constructor & Destructor Documentation

10.317.3.1 `template<typename I> mln::slice_image< I >::slice_image () [inline]`

Constructor without argument.

10.317.3.2 `template<typename I> mln::slice_image< I >::slice_image (I & ima, def::coord sli) [inline]`

Constructor from an image *ima* and a predicate *f*.

10.317.4 Member Function Documentation

10.317.4.1 `template<typename I> const box2d & mln::slice_image< I >::domain () const [inline]`

Give the definition domain.

10.317.4.2 `template<typename I> mln::slice_image< I >::operator slice_image< const I > () const [inline]`

Const promotion via conversion.

10.317.4.3 `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*.

10.317.4.4 `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*.

10.317.4.5 `template<typename I> def::coord mln::slice_image< I >::sli () const [inline]`

Give the slice number.

10.318 mln::sub_image< I, S > Struct Template Reference

[Image](#) having its domain restricted by a site [set](#).

```
#include <sub_image.hh>
```

Inherits `mln::internal::image_domain_morpher< I, S, mln::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 (const I &ima, const S &pset)`
Constructor.
- `sub_image ()`
Constructor without argument.

10.318.1 Detailed Description

```
template<typename I, typename S> struct mln::sub_image< I, S >
```

[Image](#) having its domain restricted by a site [set](#).

10.318.2 Member Typedef Documentation

10.318.2.1 `template<typename I, typename S> typedef sub_image< tag::image_<I>, tag::domain_<S> > mln::sub_image< I, S >::skeleton`

Skeleton.

10.318.3 Constructor & Destructor Documentation

10.318.3.1 `template<typename I, typename S> mln::sub_image< I, S >::sub_image ()`
[inline]

Constructor without argument.

10.318.3.2 `template<typename I, typename S> mln::sub_image< I, S >::sub_image (const I & ima, const S & pset) [inline]`

Constructor.

10.318.4 Member Function Documentation

10.318.4.1 `template<typename I, typename S> const S & mln::sub_image< I, S >::domain () const [inline]`

Give the definition domain.

10.318.4.2 `template<typename I, typename S> mln::sub_image< I, S >::operator sub_image< const I, S > () const [inline]`

Const promotion via conversion.

10.319 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 mln::internal::image_domain_morpher< I, mln::p_if< S, mln::fun::p2b::has< I > >, mln::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](#) (I &ima, const S &s)

Constructor.

- [sub_image_if](#) ()

Constructor without argument.

10.319.1 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.

10.319.2 Member Typedef Documentation

10.319.2.1 `template<typename I, typename S> typedef sub_image_if< tag::image_<I>, tag::domain_<S> > mln::sub_image_if< I, S >::skeleton`

Skeleton.

10.319.3 Constructor & Destructor Documentation

10.319.3.1 `template<typename I, typename S> mln::sub_image_if< I, S >::sub_image_if () [inline]`

Constructor without argument.

10.319.3.2 `template<typename I, typename S> mln::sub_image_if< I, S >::sub_image_if (I & ima, const S & s) [inline]`

Constructor.

10.319.4 Member Function Documentation

10.319.4.1 `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.

10.320 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.

10.320.1 Detailed Description

```
template<typename I, typename F> class mln::thru_image< I, F >
```

Morph image values through a function.

10.320.2 Member Function Documentation

10.320.2.1 `template<typename I, typename F> mln::thru_image< I, F >::operator thru_image< const I, F > () const` `[inline]`

Const promotion via conversion.

10.321 mln::thrubin_image< I1, I2, F > Class Template Reference

Morphes values from two images through a binary function.

```
#include <thrubin_image.hh>
```

Inherits mln::internal::image_value_morpher< I1, F::result, mln::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.

Public Member Functions

- [operator thrubin_image](#)< const I1, const I2, F > () const
Const promotion via conversion.

10.321.1 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.

10.321.2 Member Typedef Documentation

10.321.2.1 `template<typename I1, typename I2, typename F> typedef I1 ::psite
mln::thrubin_image< I1, I2, F >::psite`

[Point_Site](#) associated type.

10.321.2.2 `template<typename I1, typename I2, typename F> typedef value
mln::thrubin_image< I1, I2, F >::rvalue`

Return type of read-only access.

10.321.2.3 `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.

10.321.2.4 `template<typename I1, typename I2, typename F> typedef F ::result mln::thrubin_image< I1, I2, F >::value`

[Value](#) associated type.

10.321.3 Member Function Documentation

10.321.3.1 `template<typename I1, typename I2, typename F> mln::thrubin_image< I1, I2, F >::operator thrubin_image< const I1, const I2, F > () const [inline]`

Const promotion via conversion.

10.322 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 mln::topo::internal::backward_complex_relative_iterator_base< mln::topo::face< D >, mln::topo::algebraic_face< D >, mln::topo::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.

10.322.1 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.

10.322.2 Constructor & Destructor Documentation

10.322.2.1 `template<unsigned D> mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D >::adj_higher_dim_connected_n_face_bkd_iter () [inline]`

Construction.

10.322.3 Member Function Documentation

10.322.3.1 `template<typename E> void mln::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.

10.323 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 mln::topo::internal::forward_complex_relative_iterator_base< mln::topo::face< D >, mln::topo::algebraic_face< D >, mln::topo::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.

10.323.1 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.

10.323.2 Constructor & Destructor Documentation

10.323.2.1 `template<unsigned D> mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D >::adj_higher_dim_connected_n_face_fwd_iter () [inline]`

Construction.

10.323.3 Member Function Documentation

10.323.3.1 `template<typename E> void mln::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.

10.324 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 mln::topo::internal::backward_complex_relative_iterator_base< mln::topo::face< D >, mln::topo::algebraic_face< D >, mln::topo::adj_higher_face_bkd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_higher_face_bkd_iter](#) ()
Construction.

10.324.1 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.

10.324.2 Constructor & Destructor Documentation

10.324.2.1 `template<unsigned D> mln::topo::adj_higher_face_bkd_iter< D >::adj_higher_face_bkd_iter () [inline]`

Construction.

10.324.3 Member Function Documentation

10.324.3.1 `template<typename E> void mln::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.

10.325 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 mln::topo::internal::forward_complex_relative_iterator_base< mln::topo::face< D >, mln::topo::algebraic_face< D >, mln::topo::adj_higher_face_fwd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_higher_face_fwd_iter](#) ()
Construction.

10.325.1 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.

10.325.2 Constructor & Destructor Documentation

10.325.2.1 `template<unsigned D> mln::topo::adj_higher_face_fwd_iter< D >::adj_higher_face_fwd_iter () [inline]`

Construction.

10.325.3 Member Function Documentation

10.325.3.1 `template<typename E> void mln::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.

10.326 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 mln::topo::internal::backward_complex_relative_iterator_base< mln::topo::face< D >, mln::topo::algebraic_face< D >, mln::topo::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.

10.326.1 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.

10.326.2 Constructor & Destructor Documentation

10.326.2.1 `template<unsigned D> mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D >::adj_lower_dim_connected_n_face_bkd_iter () [inline]`

Construction.

10.326.3 Member Function Documentation

10.326.3.1 `template<typename E> void mln::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.

10.327 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 mln::topo::internal::forward_complex_relative_iterator_base< mln::topo::face< D >, mln::topo::algebraic_face< D >, mln::topo::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.

10.327.1 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.

10.327.2 Constructor & Destructor Documentation

10.327.2.1 `template<unsigned D> mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D >::adj_lower_dim_connected_n_face_fwd_iter () [inline]`

Construction.

10.327.3 Member Function Documentation

10.327.3.1 `template<typename E> void mln::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.

10.328 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 mln::topo::internal::backward_complex_relative_iterator_base< mln::topo::face< D >, mln::topo::algebraic_face< D >, mln::topo::adj_lower_face_bkd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_lower_face_bkd_iter](#) ()
Construction.

10.328.1 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.

10.328.2 Constructor & Destructor Documentation

10.328.2.1 `template<unsigned D> mln::topo::adj_lower_face_bkd_iter< D >::adj_lower_face_bkd_iter () [inline]`

Construction.

10.328.3 Member Function Documentation

10.328.3.1 `template<typename E> void mln::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.

10.329 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 mln::topo::internal::forward_complex_relative_iterator_base< mln::topo::face< D >, mln::topo::algebraic_face< D >, mln::topo::adj_lower_face_fwd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_lower_face_fwd_iter](#) ()
Construction.

10.329.1 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.

10.329.2 Constructor & Destructor Documentation

10.329.2.1 `template<unsigned D> mln::topo::adj_lower_face_fwd_iter< D >::adj_lower_face_fwd_iter () [inline]`

Construction.

10.329.3 Member Function Documentation

10.329.3.1 `template<typename E> void mln::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.

10.330 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 mln::topo::internal::complex_relative_iterator_sequence< mln::topo::adj_higher_face_bkd_iter< D >, mln::topo::adj_lower_face_bkd_iter< D >, mln::topo::adj_lower_higher_face_bkd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_lower_higher_face_bkd_iter](#) ()
Construction.

10.330.1 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.

10.330.2 Constructor & Destructor Documentation

10.330.2.1 `template<unsigned D> mln::topo::adj_lower_higher_face_bkd_iter< D >::adj_lower_higher_face_bkd_iter () [inline]`

Construction.

10.330.3 Member Function Documentation

10.330.3.1 `template<typename E> void mln::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.

10.331 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 mln::topo::internal::complex_relative_iterator_sequence< mln::topo::adj_lower_face_fwd_iter< D >, mln::topo::adj_higher_face_fwd_iter< D >, mln::topo::adj_lower_higher_face_fwd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_lower_higher_face_fwd_iter](#) ()
Construction.

10.331.1 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.

10.331.2 Constructor & Destructor Documentation

10.331.2.1 `template<unsigned D> mln::topo::adj_lower_higher_face_fwd_iter< D >::adj_lower_higher_face_fwd_iter () [inline]`

Construction.

10.331.3 Member Function Documentation

10.331.3.1 `template<typename E> void mln::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.

10.332 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 mln::topo::internal::backward_complex_relative_iterator_base< mln::topo::face< D >, mln::topo::algebraic_face< D >, mln::topo::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.
- 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.
- [adj_m_face_bkd_iter](#) ()
Construction.

10.332.1 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.

10.332.2 Constructor & Destructor Documentation

10.332.2.1 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.

10.332.2.2 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*.

10.332.3 Member Function Documentation

10.332.3.1 `template<typename E> void mln::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.

10.333 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 mln::topo::internal::forward_complex_relative_iterator_base< mln::topo::face< D >, mln::topo::algebraic_face< D >, mln::topo::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.
- 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.
- [adj_m_face_fwd_iter](#) ()
Construction.

10.333.1 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.

10.333.2 Constructor & Destructor Documentation

10.333.2.1 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.

10.333.2.2 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*.

10.333.3 Member Function Documentation

10.333.3.1 `template<typename E> void mln::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:

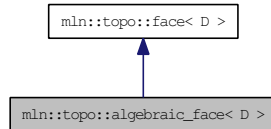
The iterator is valid.

10.334 mln::topo::algebraic_face< D > Struct 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

- `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`.
- `algebraic_face` (`complex`< D > &`complex`, unsigned `n`, unsigned `face_id`, bool `sign`)
Build an algebraic [face](#) handle from `complex` and `face_id`.
- `algebraic_face` ()
Build a non-initialized algebraic [face](#) handle.
- void `invalidate` ()
Invalidate this handle.
- bool `is_valid` () const
Is this handle valid?
- `complex`< D > `cplx` () const
Accessors.
- `template<unsigned N>`
`face_data`< N, D > & `data` () const
Return the `mln::topo::face_data` pointed by this handle.
- void `dec_face_id` ()
Decrement the id of the [face](#).
- void `dec_n` ()
Decrement the dimension of the [face](#).
- unsigned `face_id` () const
Return the id of the [face](#).

- `std::vector< algebraic_face< D > > higher_dim_adj_faces ()` const
Return an array of *face* handles pointing to adjacent (n+1)-faces.
- `void inc_face_id ()`
Increment the id of the *face*.
- `void inc_n ()`
Increment the dimension of the *face*.
- `std::vector< algebraic_face< D > > lower_dim_adj_faces ()` const
Return an array of *face* handles pointing to adjacent (n-1)-faces.
- `unsigned n ()` const
Return the dimension of the *face*.
- `void set_cplx (const complex< D > &cplx)`
Set the *complex* the *face* belongs to.
- `void set_face_id (unsigned face_id)`
Set the id of the *face*.
- `void set_n (unsigned n)`
Set the dimension of the *face*.
- `void set_sign (bool sign)`
Set the sign of this *face*.
- `bool sign ()` const
Accessors.

10.334.1 Detailed Description

`template<unsigned D> struct 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.

10.334.2 Constructor & Destructor Documentation

10.334.2.1 `template<unsigned D> mln::topo::algebraic_face< D >::algebraic_face ()`
[inline]

Build a non-initialized algebraic *face* handle.

10.334.2.2 `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*.

10.334.2.3 `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`.

References `mln::topo::face< D >::n()`.

10.334.2.4 `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`.

10.334.3 Member Function Documentation

10.334.3.1 `template<unsigned D> complex< D > mln::topo::face< D >::cplx () const [inline, inherited]`

Accessors.

Return the [complex](#) the [face](#) belongs to.

Referenced by `mln::complex_psite< D, G >::complex_psite()`, `mln::topo::operator!=()`, and `mln::topo::operator==()`.

10.334.3.2 `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.

References `mln::topo::face< D >::is_valid()`.

10.334.3.3 `template<unsigned D> void mln::topo::face< D >::dec_face_id () [inline, inherited]`

Decrement the id of the [face](#).

10.334.3.4 `template<unsigned D> void mln::topo::face< D >::dec_n () [inline, inherited]`

Decrement the dimension of the [face](#).

10.334.3.5 `template<unsigned D> unsigned mln::topo::face< D >::face_id () const [inline, inherited]`

Return the id of the [face](#).

Referenced by `mln::geom::complex_geometry< D, P >::operator()()`, and `mln::topo::operator==()`.

10.334.3.6 `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.

10.334.3.7 `template<unsigned D> void mln::topo::face< D >::inc_face_id ()` [inline, inherited]

Increment the id of the [face](#).

10.334.3.8 `template<unsigned D> void mln::topo::face< D >::inc_n ()` [inline, inherited]

Increment the dimension of the [face](#).

10.334.3.9 `template<unsigned D> void mln::topo::face< D >::invalidate ()` [inline, inherited]

Invalidate this handle.

References `mln::topo::face< D >::set_face_id()`, and `mln::topo::face< D >::set_n()`.

10.334.3.10 `template<unsigned D> bool mln::topo::face< D >::is_valid () const` [inline, inherited]

Is this handle valid?

Referenced by `mln::topo::face< D >::data()`.

10.334.3.11 `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.

10.334.3.12 `template<unsigned D> unsigned mln::topo::face< D >::n () const` [inline, inherited]

Return the dimension of the [face](#).

Referenced by `mln::topo::algebraic_face< D >::algebraic_face()`, `mln::geom::complex_geometry< D, P >::operator()`, and `mln::topo::operator==(())`.

10.334.3.13 `template<unsigned D> void mln::topo::face< D >::set_cplx (const complex< D > & cplx)` [inline, inherited]

Set the [complex](#) the [face](#) belongs to.

10.334.3.14 `template<unsigned D> void mln::topo::face< D >::set_face_id (unsigned face_id)`
[inline, inherited]

Set the id of the [face](#).

Referenced by mln::topo::face< D >::invalidate().

10.334.3.15 `template<unsigned D> void mln::topo::face< D >::set_n (unsigned n)` [inline, inherited]

Set the dimension of the [face](#).

Referenced by mln::topo::face< D >::invalidate().

10.334.3.16 `template<unsigned D> void mln::topo::algebraic_face< D >::set_sign (bool sign)`
[inline]

Set the sign of this [face](#).

10.334.3.17 `template<unsigned D> bool mln::topo::algebraic_face< D >::sign () const`
[inline]

Accessors.

Return the sign of this [face](#).

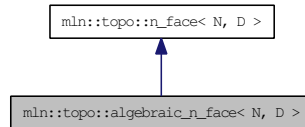
Referenced by mln::topo::operator==().

10.335 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](#) (const [n_face](#)< N, D > &f, bool [sign](#))
Build an algebraic [face](#) handle from an [mln::n_face](#).
- [algebraic_n_face](#) ([complex](#)< D > &[complex](#), unsigned [face_id](#), bool [sign](#))
Build an algebraic [face](#) handle from [complex](#) and [face_id](#).
- [algebraic_n_face](#) ()
Build a non-initialized algebraic [face](#) handle.
- void [invalidate](#) ()
Invalidate this handle.
- bool [is_valid](#) () const
Is this handle valid?
- [complex](#)< D > [cplx](#) () const
Accessors.
- [face_data](#)< N, D > & [data](#) () const
Return the [mln::topo::face_data](#) pointed by this handle.
- void [dec_face_id](#) ()
Decrement the id of the [face](#).
- unsigned [face_id](#) () const
Return the id of the [face](#).
- 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.
- void [inc_face_id](#) ()
Increment the id of the [face](#).
- 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.

- unsigned `n` () const
Return the dimension of the *face*.
- void `set_cplx` (const `complex< D >` &cplx)
Set the *complex* the *face* belongs to.
- void `set_face_id` (unsigned `face_id`)
Set the id of the *face*.
- void `set_sign` (bool `sign`)
Set the sign of this *face*.
- bool `sign` () const
Accessors.

10.335.1 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.

10.335.2 Constructor & Destructor Documentation

```
10.335.2.1 template<unsigned N, unsigned D> mln::topo::algebraic_n_face< N, D
>::algebraic_n_face () [inline]
```

Build a non-initialized algebraic `face` handle.

References `mln::topo::n_face< N, D >::is_valid()`.

```
10.335.2.2 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`.

```
10.335.2.3 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`.

10.335.3 Member Function Documentation

10.335.3.1 `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.

Referenced by `mln::topo::n_faces_set< N, D >::add()`, `mln::topo::operator!=()`, and `mln::topo::operator==()`.

10.335.3.2 `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.

References `mln::topo::n_face< N, D >::is_valid()`.

10.335.3.3 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::dec_face_id ()` [inline, inherited]

Decrement the id of the [face](#).

10.335.3.4 `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::face_id () const` [inline, inherited]

Return the id of the [face](#).

Referenced by `mln::topo::operator==()`.

10.335.3.5 `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.

References `mln::topo::n_face< N, D >::is_valid()`.

Referenced by `mln::topo::edge()`.

10.335.3.6 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::inc_face_id ()` [inline, inherited]

Increment the id of the [face](#).

10.335.3.7 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::invalidate ()` [inline, inherited]

Invalidate this handle.

References `mln::topo::n_face< N, D >::set_face_id()`.

10.335.3.8 `template<unsigned N, unsigned D> bool mln::topo::n_face< N, D >::is_valid () const`
`[inline, inherited]`

Is this handle valid?

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()`.

10.335.3.9 `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.

References `mln::topo::n_face< N, D >::is_valid()`.

10.335.3.10 `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::n () const`
`[inline, inherited]`

Return the dimension of the [face](#).

10.335.3.11 `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.

10.335.3.12 `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](#).

Referenced by `mln::topo::n_face< N, D >::invalidate()`.

10.335.3.13 `template<unsigned N, unsigned D> void mln::topo::algebraic_n_face< N, D`
`>::set_sign (bool sign)` `[inline]`

Set the sign of this [face](#).

10.335.3.14 `template<unsigned N, unsigned D> bool mln::topo::algebraic_n_face< N, D >::sign`
`() const` `[inline]`

Accessors.

Return the sign of this [face](#).

Referenced by `mln::topo::operator==()`.

10.336 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 `mln::topo::internal::forward_complex_relative_iterator_base< mln::topo::face< D >, mln::topo::algebraic_face< D >, mln::topo::center_only_iter< D > >`.

Public Member Functions

- void [next](#) ()

Go to the next element.

- [center_only_iter](#) ()

Construction.

10.336.1 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
```

10.336.2 Constructor & Destructor Documentation

10.336.2.1 `template<unsigned D> mln::topo::center_only_iter< D >::center_only_iter ()`
[inline]

Construction.

10.336.3 Member Function Documentation

10.336.3.1 `template<typename E> void mln::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.

10.337 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 mln::topo::internal::complex_relative_iterator_sequence< I, mln::topo::center_only_iter< D >, mln::topo::centered_bkd_iter_adapter< D, I > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [centered_bkd_iter_adapter](#) ()
Construction.

10.337.1 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.

10.337.2 Constructor & Destructor Documentation

10.337.2.1 `template<unsigned D, typename I> mln::topo::centered_bkd_iter_adapter< D, I >::centered_bkd_iter_adapter () [inline]`

Construction.

10.337.3 Member Function Documentation

10.337.3.1 `template<typename E> void mln::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.

10.338 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 mln::topo::internal::complex_relative_iterator_sequence< mln::topo::center_only_iter< D >, I, mln::topo::centered_fwd_iter_adapter< D, I > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [centered_fwd_iter_adapter](#) ()
Construction.

10.338.1 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.

10.338.2 Constructor & Destructor Documentation

10.338.2.1 `template<unsigned D, typename I> mln::topo::centered_fwd_iter_adapter< D, I >::centered_fwd_iter_adapter () [inline]`

Construction.

10.338.3 Member Function Documentation

10.338.3.1 `template<typename E> void mln::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.

10.339 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`.
- `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 $N \geq 0$).
- `n_face< 0u, D > add_face ()`
Add a 0-face to the `complex`.
- `complex ()`
Complex construction.
- `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 .

10.339.1 Detailed Description

`template<unsigned D> class mln::topo::complex< D >`

General [complex](#) of dimension D.

10.339.2 Member Typedef Documentation

10.339.2.1 `template<unsigned D> typedef face_bkd_iter<D> mln::topo::complex< D >::bkd_citer`

Backward [mln::Iterator](#) type iterating on all faces.

10.339.2.2 `template<unsigned D> typedef face_fwd_iter<D> mln::topo::complex< D >::fwd_citer`

Forward [mln::Iterator](#) type iterating on all faces.

10.339.3 Constructor & Destructor Documentation

10.339.3.1 `template<unsigned D> mln::topo::complex< D >::complex () [inline]`

Complex construction.

Create a new D-complex.

10.339.4 Member Function Documentation

10.339.4.1 `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.

References `mln::topo::n_faces_set< N, D >::faces()`.

10.339.4.2 `template<unsigned D> n_face< 0u, D > mln::topo::complex< D >::add_face () [inline]`

Add a 0-face to the [complex](#).

10.339.4.3 `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](#).

10.339.4.4 `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.

10.339.4.5 `template<unsigned D> unsigned mln::topo::complex< D >::nfaces_of_dim (unsigned n) const` [inline]

Dynamic manipulators.

These methods use input known 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).

10.339.4.6 `template<unsigned D> template<unsigned N> unsigned mln::topo::complex< D >::nfaces_of_static_dim () const` [inline]

Return the number of N-faces.

10.339.4.7 `template<unsigned D> void mln::topo::complex< D >::print (std::ostream & ostr) const` [inline]

Pretty-printing.

Print the [complex](#).

Referenced by `mln::topo::operator<<()`.

10.339.4.8 `template<unsigned D> template<unsigned N> void mln::topo::complex< D >::print_faces (std::ostream & ostr) const` [inline]

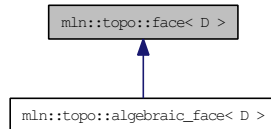
Print the faces of dimension N.

10.340 mln::topo::face< D > Struct 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

- `template<unsigned N>`
`face` (const `n_face`< N, D > &f)
Build a [face](#) handle from an `mln::topo::n_face`.
- `face` (`complex`< D > &`complex`, unsigned n, unsigned face_id)
Build a [face](#) handle from `complex` and face_id.
- `face` ()
Build a non-initialized [face](#) handle.
- void `invalidate` ()
Invalidate this handle.
- bool `is_valid` () const
Is this handle valid?
- `complex`< D > `cplx` () const
Accessors.
- `template<unsigned N>`
`face_data`< N, D > & `data` () const
Return the `mln::topo::face_data` pointed by this handle.
- void `dec_face_id` ()
Decrement the id of the [face](#).
- void `dec_n` ()
Decrement the dimension of the [face](#).
- unsigned `face_id` () const
Return the id of the [face](#).
- `std::vector`< `algebraic_face`< D > > `higher_dim_adj_faces` () const
Return an array of [face](#) handles pointing to adjacent (n+1)-faces.

- void `inc_face_id ()`
Increment the id of the `face`.
- void `inc_n ()`
Increment the dimension of the `face`.
- `std::vector< algebraic_face< D > > lower_dim_adj_faces () const`
Return an array of `face` handles pointing to adjacent (n-1)-faces.
- unsigned `n () const`
Return the dimension of the `face`.
- void `set_cplx (const complex< D > &cplx)`
Set the `complex` the `face` belongs to.
- void `set_face_id (unsigned face_id)`
Set the id of the `face`.
- void `set_n (unsigned n)`
Set the dimension of the `face`.

10.340.1 Detailed Description

`template<unsigned D> struct 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.

10.340.2 Constructor & Destructor Documentation

10.340.2.1 `template<unsigned D> mln::topo::face< D >::face () [inline]`

Build a non-initialized `face` handle.

10.340.2.2 `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`.

10.340.2.3 `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`.

10.340.3 Member Function Documentation

10.340.3.1 `template<unsigned D> complex< D > mln::topo::face< D >::cplx () const [inline]`

Accessors.

Return the [complex](#) the [face](#) belongs to.

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]`

Return the mln::topo::face_data pointed by this handle.

References mln::topo::face< D >::is_valid().

10.340.3.3 `template<unsigned D> void mln::topo::face< D >::dec_face_id () [inline]`

Decrement the id of the [face](#).

10.340.3.4 `template<unsigned D> void mln::topo::face< D >::dec_n () [inline]`

Decrement the dimension of the [face](#).

10.340.3.5 `template<unsigned D> unsigned mln::topo::face< D >::face_id () const [inline]`

Return the id of the [face](#).

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]`

Return an array of [face](#) handles pointing to adjacent (n+1)-faces.

10.340.3.7 `template<unsigned D> void mln::topo::face< D >::inc_face_id () [inline]`

Increment the id of the [face](#).

10.340.3.8 `template<unsigned D> void mln::topo::face< D >::inc_n () [inline]`

Increment the dimension of the [face](#).

10.340.3.9 `template<unsigned D> void mln::topo::face< D >::invalidate () [inline]`

Invalidate this handle.

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]`

Is this handle valid?

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]

Return an array of [face](#) handles pointing to adjacent (n-1)-faces.

10.340.3.12 `template<unsigned D> unsigned mln::topo::face< D >::n () const` [inline]

Return the dimension of the [face](#).

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]

Set the [complex](#) the [face](#) belongs to.

10.340.3.14 `template<unsigned D> void mln::topo::face< D >::set_face_id (unsigned face_id)` [inline]

Set the id of the [face](#).

Referenced by `mln::topo::face< D >::invalidate()`.

10.340.3.15 `template<unsigned D> void mln::topo::face< D >::set_n (unsigned n)` [inline]

Set the dimension of the [face](#).

Referenced by `mln::topo::face< D >::invalidate()`.

10.341 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 mln::topo::internal::complex_set_iterator_base< mln::topo::face< D >, mln::topo::face_bkd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [face_bkd_iter](#) ()
Construction and assignment.
- void [start](#) ()
Manipulation.

10.341.1 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.

10.341.2 Constructor & Destructor Documentation

10.341.2.1 `template<unsigned D> mln::topo::face_bkd_iter< D >::face_bkd_iter ()` [inline]

Construction and assignment.

10.341.3 Member Function Documentation

10.341.3.1 `template<typename E> void mln::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.

10.341.3.2 `template<unsigned D> void mln::topo::face_bkd_iter<D>::start () [inline]`

Manipulation.

Start an iteration.

10.342 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 mln::topo::internal::complex_set_iterator_base< mln::topo::face< D >, mln::topo::face_fwd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [face_fwd_iter](#) ()
Construction and assignment.
- void [start](#) ()
Manipulation.

10.342.1 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.

10.342.2 Constructor & Destructor Documentation

10.342.2.1 `template<unsigned D> mln::topo::face_fwd_iter< D >::face_fwd_iter ()` [[inline](#)]

Construction and assignment.

10.342.3 Member Function Documentation

10.342.3.1 `template<typename E> void mln::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.

10.342.3.2 `template<unsigned D> void mln::topo::face_fwd_iter<D>::start()` `[inline]`

Manipulation.

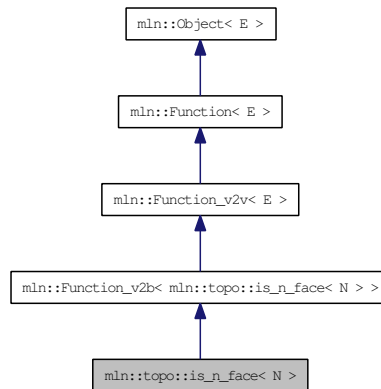
Test if the iterator is valid.

10.343 mln::topo::is_n_face< N > Struct Template Reference

A functor testing whether a [mln::complex_site](#) is an N -face.

```
#include <is_n_face.hh>
```

Inheritance diagram for mln::topo::is_n_face< N >:



10.343.1 Detailed Description

```
template<unsigned N> struct mln::topo::is_n_face< N >
```

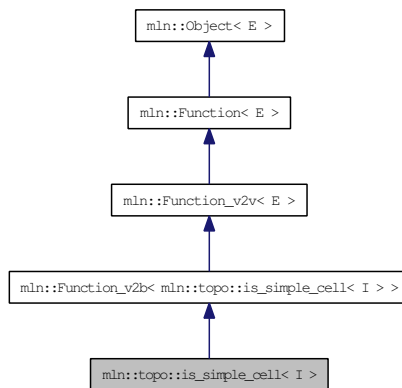
A functor testing whether a [mln::complex_site](#) is an N -face.

10.344 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](#)).

10.344.1 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.

10.344.2 Member Typedef Documentation

**10.344.2.1 template<typename I> typedef mln::complex_psite<D, G>
mln::topo::is_simple_cell< I >::psite**

Psite type.

10.344.2.2 template<typename I> typedef bool mln::topo::is_simple_cell< I >::result

Result type of the functor.

Reimplemented from [mln::Function_v2b< E >](#).

10.344.3 Member Function Documentation

10.344.3.1 template<typename I> typedef mln::topo::is_simple_cell< I >::mln_geom (I)

Geometry of the image.

**10.344.3.2 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](#).

References [mln::make::attachment\(\)](#).

**10.344.3.3 template<typename I> void mln::topo::is_simple_cell< I >::set_image (const
mln::Image< I > & ima) [inline]**

Set the underlying image.

10.344.4 Member Data Documentation

**10.344.4.1 template<typename I> const unsigned mln::topo::is_simple_cell< I >::D = I::dim
[static]**

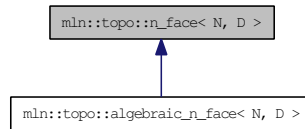
Dimension of the image (and therefore of the [complex](#)).

10.345 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](#) (complex< D > &complex, unsigned face_id)
Build a [face](#) handle from [complex](#) and face_id.
- [n_face](#) ()
Build a non-initialized [face](#) handle.
- complex< D > [cplx](#) () const
Accessors.
- face_data< N, D > & [data](#) () const
Return the mln::topo::face_data pointed by this handle.
- void [dec_face_id](#) ()
Decrement the id of the [face](#).
- unsigned [face_id](#) () const
Return the id of the [face](#).
- 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.
- void [inc_face_id](#) ()
Increment the id of the [face](#).
- 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.
- unsigned [n](#) () const
Return the dimension of the [face](#).

- void `set_cplx` (const `complex< D >` &cplx)
Set the *complex* the *face* belongs to.
- void `set_face_id` (unsigned `face_id`)
Set the *id* of the *face*.

10.345.1 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.

10.345.2 Constructor & Destructor Documentation

10.345.2.1 `template<unsigned N, unsigned D> mln::topo::n_face< N, D >::n_face ()` `[inline]`

Build a non-initialized `face` handle.

References `mln::topo::n_face< N, D >::is_valid()`.

10.345.2.2 `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`.

10.345.3 Member Function Documentation

10.345.3.1 `template<unsigned N, unsigned D> complex< D > mln::topo::n_face< N, D >::cplx ()` `const` `[inline]`

Accessors.

Return the `complex` the `face` belongs to.

Referenced by `mln::topo::n_faces_set< N, D >::add()`, `mln::topo::operator!=()`, and `mln::topo::operator==()`.

10.345.3.2 `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.

References `mln::topo::n_face< N, D >::is_valid()`.

10.345.3.3 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::dec_face_id ()` `[inline]`

Decrement the *id* of the `face`.

10.345.3.4 `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::face_id () const [inline]`

Return the id of the [face](#).

Referenced by `mln::topo::operator==(())`.

10.345.3.5 `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.

References `mln::topo::n_face< N, D >::is_valid()`.

Referenced by `mln::topo::edge()`.

10.345.3.6 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::inc_face_id () [inline]`

Increment the id of the [face](#).

10.345.3.7 `template<unsigned N, unsigned D> void mln::topo::n_face< N, D >::invalidate () [inline]`

Invalidate this handle.

References `mln::topo::n_face< N, D >::set_face_id()`.

10.345.3.8 `template<unsigned N, unsigned D> bool mln::topo::n_face< N, D >::is_valid () const [inline]`

Is this handle valid?

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()`.

10.345.3.9 `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.

References `mln::topo::n_face< N, D >::is_valid()`.

10.345.3.10 `template<unsigned N, unsigned D> unsigned mln::topo::n_face< N, D >::n () const [inline]`

Return the dimension of the [face](#).

10.345.3.11 `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.

10.345.3.12 `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](#).

Referenced by `mln::topo::n_face< N, D >::invalidate()`.

10.346 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 mln::topo::internal::complex_set_iterator_base< mln::topo::face< D >, mln::topo::n_face_bkd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- unsigned [n](#) () const
Accessors.
- [n_face_bkd_iter](#) ()
Construction and assignment.
- void [start](#) ()
Manipulation.

10.346.1 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.

10.346.2 Constructor & Destructor Documentation

10.346.2.1 `template<unsigned D> mln::topo::n_face_bkd_iter< D >::n_face_bkd_iter ()`
[inline]

Construction and assignment.

10.346.3 Member Function Documentation

10.346.3.1 `template<unsigned D> unsigned mln::topo::n_face_bkd_iter< D >::n () const`
[inline]

Accessors.

Shortcuts to face_'s accessors.

Referenced by mln::topo::n_face_bkd_iter< D >::start().

10.346.3.2 `template<typename E> void mln::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.

10.346.3.3 `template<unsigned D> void mln::topo::n_face_bkd_iter< D >::start ()` [inline]

Manipulation.

Start an iteration.

References mln::topo::n_face_bkd_iter< D >::n().

10.347 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 mln::topo::internal::complex_set_iterator_base< mln::topo::face< D >, mln::topo::n_face_fwd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- unsigned [n](#) () const
Accessors.
- [n_face_fwd_iter](#) ()
Construction and assignment.
- void [start](#) ()
Manipulation.

10.347.1 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.

10.347.2 Constructor & Destructor Documentation

10.347.2.1 `template<unsigned D> mln::topo::n_face_fwd_iter< D >::n_face_fwd_iter ()`
[inline]

Construction and assignment.

10.347.3 Member Function Documentation

10.347.3.1 `template<unsigned D> unsigned mln::topo::n_face_fwd_iter< D >::n () const`
[inline]

Accessors.

Shortcuts to face_'s accessors.

10.347.3.2 `template<typename E> void mln::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.

10.347.3.3 `template<unsigned D> void mln::topo::n_face_fwd_iter< D >::start ()` [inline]

Manipulation.

Test if the iterator is valid.

10.348 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.

10.348.1 Detailed Description

```
template<unsigned N, unsigned D> class mln::topo::n_faces_set< N, D >
```

Set of [face](#) handles of dimension N.

10.348.2 Member Typedef Documentation

10.348.2.1 `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.

10.348.3 Member Function Documentation

10.348.3.1 `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.

References `mln::topo::n_face< N, D >::cplx()`.

Referenced by `mln::topo::operator+()`, and `mln::topo::operator-()`.

10.348.3.2 `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.

Referenced by mln::topo::complex< D >::add_face().

10.348.3.3 `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.

10.349 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 `mln::topo::internal::complex_set_iterator_base< mln::topo::face< D >, mln::topo::static_n_face_bkd_iter< N, D > >`.

Public Member Functions

- void [next](#) ()
Go to the next element.
- void [start](#) ()
Manipulation.
- [static_n_face_bkd_iter](#) ()
Construction and assignment.

10.349.1 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.

10.349.2 Constructor & Destructor Documentation

10.349.2.1 `template<unsigned N, unsigned D> mln::topo::static_n_face_bkd_iter< N, D >::static_n_face_bkd_iter () [inline]`

Construction and assignment.

10.349.3 Member Function Documentation

10.349.3.1 `template<typename E> void mln::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.

10.349.3.2 `template<unsigned N, unsigned D> void mln::topo::static_n_face_bkd_iter< N, D >::start () [inline]`

Manipulation.

Start an iteration.

10.350 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 `mln::topo::internal::complex_set_iterator_base< mln::topo::face< D >, mln::topo::static_n_face_fwd_iter< N, D >>`.

Public Member Functions

- void [next](#) ()
Go to the next element.
- void [start](#) ()
Manipulation.
- [static_n_face_fwd_iter](#) ()
Construction and assignment.

10.350.1 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.

10.350.2 Constructor & Destructor Documentation

10.350.2.1 `template<unsigned N, unsigned D> mln::topo::static_n_face_fwd_iter< N, D >::static_n_face_fwd_iter () [inline]`

Construction and assignment.

10.350.3 Member Function Documentation

10.350.3.1 `template<typename E> void mln::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.

10.350.3.2 `template<unsigned N, unsigned D> void mln::topo::static_n_face_fwd_iter< N, D >::start () [inline]`

Manipulation.

Test if the iterator is valid.

10.351 mln::tr_image< S, I, T > Struct Template Reference

Transform an image by a given transformation.

```
#include <tr_image.hh>
```

Inherits mln::internal::image_identity< I, S, mln::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.

10.351.1 Detailed Description

`template<typename S, typename I, typename T> struct mln::tr_image< S, I, T >`

Transform an image by a given transformation.

10.351.2 Member Typedef Documentation

10.351.2.1 `template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::lvalue`

Return type of read-write access.

10.351.2.2 `template<typename S, typename I, typename T> typedef I ::psite mln::tr_image< S, I, T >::psite`

[Point_Site](#) associated type.

10.351.2.3 `template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::rvalue`

Return type of read-only access.

10.351.2.4 `template<typename S, typename I, typename T> typedef I ::site mln::tr_image< S, I, T >::site`

[Site](#) associated type.

10.351.2.5 `template<typename S, typename I, typename T> typedef tr_image< S, tag::image_<I>, T> mln::tr_image< S, I, T >::skeleton`

Skeleton.

10.351.2.6 `template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::value`

[Value](#) associated type.

10.351.3 Constructor & Destructor Documentation

10.351.3.1 `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.

10.351.4 Member Function Documentation

10.351.4.1 `template<typename S, typename I, typename T> const S & mln::tr_image< S, I, T >::domain () const` [inline]

Return the domain morpher.

10.351.4.2 `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*.

10.351.4.3 `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.

10.351.4.4 `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).

10.351.4.5 `template<typename S, typename I, typename T> void mln::tr_image< S, I, T >::set_tr (T & tr)` [inline]

Set the transformation.

10.351.4.6 `template<typename S, typename I, typename T> const T & mln::tr_image< S, I, T >::tr () const` [inline]

Return the underlying transformation.

10.352 mln::transformed_image< I, F > Struct Template Reference

[Image](#) having its domain restricted by a site [set](#).

```
#include <transformed_image.hh>
```

Inherits mln::internal::image_domain_morpher< I, mln::p_transformed< I::domain_t, F >, mln::transformed_image< I, F > >.

Public Types

- typedef [transformed_image](#)< tag::image_< I >, tag::function_< F > > [skeleton](#)
Skeleton.

Public Member Functions

- const [p_transformed](#)< typename I::domain_t, F > & [domain](#) () const
Give the definition domain.
- [operator transformed_image](#)< const I, F > () const
Const promotion via conversion.
- internal::morpher_lvalue_< I >::ret [operator](#)() (const typename I::psite &p)
Read and "write if possible" access of [pixel value](#) at [point](#) site p.
- I::rvalue [operator](#)() (const typename I::psite &p) const
Read-only access of [pixel value](#) at [point](#) site p.
- [transformed_image](#) (I &ima, const F &f)
Constructor.
- [transformed_image](#) ()
Constructor without argument.

10.352.1 Detailed Description

```
template<typename I, typename F> struct mln::transformed_image< I, F >
```

[Image](#) having its domain restricted by a site [set](#).

10.352.2 Member Typedef Documentation

10.352.2.1 `template<typename I, typename F> typedef transformed_image< tag::image_<I>, tag::function_<F> > mln::transformed_image< I, F >::skeleton`

Skeleton.

10.352.3 Constructor & Destructor Documentation

10.352.3.1 `template<typename I, typename F> mln::transformed_image< I, F >::transformed_image () [inline]`

Constructor without argument.

10.352.3.2 `template<typename I, typename F> mln::transformed_image< I, F >::transformed_image (I & ima, const F & f) [inline]`

Constructor.

10.352.4 Member Function Documentation

10.352.4.1 `template<typename I, typename F> const p_transformed< typename I::domain_t, F > & mln::transformed_image< I, F >::domain () const [inline]`

Give the definition domain.

10.352.4.2 `template<typename I, typename F> mln::transformed_image< I, F >::operator transformed_image< const I, F > () const [inline]`

Const promotion via conversion.

10.352.4.3 `template<typename I, typename F> internal::morpher_lvalue_< I >::ret mln::transformed_image< I, F >::operator() (const typename I::psite & p) [inline]`

Read and "write if possible" access of [pixel value](#) at [point](#) site *p*.

10.352.4.4 `template<typename I, typename F> I::rvalue mln::transformed_image< I, F >::operator() (const typename I::psite & p) const [inline]`

Read-only access of [pixel value](#) at [point](#) site *p*.

10.353 mln::unproject_image< I, D, F > Struct Template Reference

Un-projects an image.

```
#include <unproject_image.hh>
```

Inherits mln::internal::image_domain_morpher< I, D, mln::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](#) (I &ima, const D &dom, const F &f)
Constructor from an image [ima](#), a domain [dom](#), and a function [f](#).
- [unproject_image](#) ()
Constructor without argument.

10.353.1 Detailed Description

```
template<typename I, typename D, typename F> struct mln::unproject_image< I, D, F >
```

Un-projects an image.

10.353.2 Constructor & Destructor Documentation

10.353.2.1 `template<typename I, typename D, typename F> mln::unproject_image< I, D, F >::unproject_image () [inline]`

Constructor without argument.

10.353.2.2 `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](#).

10.353.3 Member Function Documentation

10.353.3.1 `template<typename I, typename D, typename F> const D & mln::unproject_image< I, D, F >::domain () const [inline]`

Give the definition domain.

10.353.3.2 `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.

10.353.3.3 `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.

10.354 mln::util::adjacency_matrix< V > Class Template Reference

A class of adjacency matrix.

```
#include <adjacency_matrix.hh>
```

Inherits mln::util::internal::adjacency_matrix_impl_selector< V, mln::metal::equal< mln_trait_value_-quant(V), mln::trait::value::quant::low >::eval >.

Public Member Functions

- [adjacency_matrix](#) (const V &nelements)
Construct an adjacency matrix with nelements elements maximum.
- [adjacency_matrix](#) ()
Constructors.

10.354.1 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.

10.354.2 Constructor & Destructor Documentation

10.354.2.1 `template<typename V> mln::util::adjacency_matrix< V >::adjacency_matrix ()`
[inline]

Constructors.

```
@{
```

Default

10.354.2.2 `template<typename V> mln::util::adjacency_matrix< V >::adjacency_matrix (const V & nelements)` [inline]

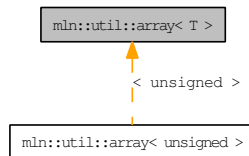
Construct an adjacency matrix with nelements elements maximum.

10.355 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 `array_bkd_iter< T >` [bkd_eiter](#)
Backward iterator associated type.
- typedef `fwd_eiter` [eiter](#)
Iterator associated type.
- typedef `array_fwd_iter< T >` [fwd_eiter](#)
Iterator types
Forward iterator associated type.
- typedef `T` [result](#)
Returned value types.

Public Member Functions

- `template<typename U>`
`array< T >` & [append](#) (const `array< U >` &`other`)
Add the elements of `other` at the end of this `array`.
- `array< T >` & [append](#) (const `T` &`elt`)
Add the element `elt` 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.
- `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`.
- 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`.
- ro_result `operator[] (unsigned i) const`
Return the `i`-th element of the `array`.
- void `reserve (unsigned n)`
Reserve memory for `n` elements.
- void `resize (unsigned n, const T &value)`
Resize this `array` to `n` elements with `value` as `value`.
- void `resize (unsigned n)`
Resize this `array` to `n` elements.
- 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 (unsigned n, const T &value)`
Construct a new `array`, resize it to `n` elements and fill it with `default_value`.
- `array (unsigned n)`
Construct a new `array` and resize it to `n` elements.
- `array ()`
*Constructors
Constructor without arguments.*

10.355.1 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.

10.355.2 Member Typedef Documentation

10.355.2.1 template<typename T> typedef array_bkd_iter<T> mln::util::array< T >::bkd_eiter

Backward iterator associated type.

10.355.2.2 template<typename T> typedef fwd_eiter mln::util::array< T >::eiter

[Iterator](#) associated type.

10.355.2.3 template<typename T> typedef T mln::util::array< T >::element

Element associated type.

10.355.2.4 template<typename T> typedef array_fwd_iter<T> mln::util::array< T >::fwd_eiter

[Iterator](#) types

Forward iterator associated type.

10.355.2.5 template<typename T> typedef T mln::util::array< T >::result

Returned [value](#) types.

Related to the [Function_v2v](#) concept.

10.355.3 Constructor & Destructor Documentation

10.355.3.1 template<typename T> mln::util::array< T >::array () [inline]

Constructors

Constructor without arguments.

10.355.3.2 template<typename T> mln::util::array< T >::array (unsigned n) [inline]

Construct a new [array](#) and resize it to elements.

10.355.3.3 `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`.

10.355.4 Member Function Documentation

10.355.4.1 `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](#).
 References `mln::util::array< T >::is_empty()`, and `mln::util::array< T >::std_vector()`.

10.355.4.2 `template<typename T> array< T > & mln::util::array< T >::append (const T & elt)` `[inline]`

Add the element `elt` at the end of this [array](#).
 Referenced by `mln::io::plot::load()`, and `mln::data::impl::generic::sort_offsets_increasing()`.

10.355.4.3 `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`

References `mln::util::array< T >::is_empty()`.
 Referenced by `mln::io::plot::load()`.

10.355.4.4 `template<typename T> void mln::util::array< T >::fill (const T & value)` `[inline]`

Fill the whole [array](#) with `value`.

10.355.4.5 `template<typename T> bool mln::util::array< T >::is_empty () const` `[inline]`

Test if the [array](#) is empty.
 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()`.

10.355.4.6 `template<typename T> std::size_t mln::util::array< T >::memory_size () const` `[inline]`

Return the size of this [array](#) in memory.

References `mln::util::array< T >::nelements()`.

10.355.4.7 `template<typename T> unsigned mln::util::array< T >::nelements () const` [inline]

Return the number of elements of the [array](#).

Referenced by `mln::labeling::fill_holes()`, `mln::make::image3d()`, `mln::util::array< T >::is_empty()`, `mln::io::pnms::load()`, `mln::util::array< T >::memory_size()`, `mln::util::operator<<()`, `mln::util::array< T >::operator[]()`, and `mln::util::array< T >::size()`.

10.355.4.8 `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()`

10.355.4.9 `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()`

10.355.4.10]

`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()`

References `mln::util::array< T >::nelements()`.

10.355.4.11]

`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()`

References `mln::util::array< T >::nelements()`.

10.355.4.12 `template<typename T> void mln::util::array< T >::reserve (unsigned n)`
[inline]

Reserve memory for *n* elements.

Referenced by `mln::data::impl::generic::sort_offsets_increasing()`.

10.355.4.13 `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*.

10.355.4.14 `template<typename T> void mln::util::array< T >::resize (unsigned n)` [inline]

Resize this [array](#) to *n* elements.

10.355.4.15 `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](#)

References `mln::util::array< T >::nelements()`.

Referenced by `mln::value::lut_vec< S, T >::lut_vec()`, and `mln::labeled_image_base< I, E >::update_data()`.

10.355.4.16 `template<typename T> const std::vector< T > & mln::util::array< T >::std_vector () const` [inline]

Return the corresponding `std::vector` of elements.

Referenced by `mln::util::array< T >::append()`, `mln::value::lut_vec< S, T >::lut_vec()`, and `mln::util::operator==(())`.

10.356 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 apex.
- [branch](#) ([tree](#)< T > &[tree](#), [tree_node](#)< T > &[apex](#))
Constructor.
- [tree](#)< T > & [util_tree](#) ()
The getter of the tree.

10.356.1 Detailed Description

```
template<typename T> class mln::util::branch< T >
```

Class of generic [branch](#).

10.356.2 Constructor & Destructor Documentation

10.356.2.1 `template<typename T> mln::util::branch< T >::branch (util::tree< T > & tree, util::tree_node< T > & apex)` `[inline]`

Constructor.

Parameters:

- ← *tree* The [tree](#) of the [branch](#).
- ← *apex* The apex of the [branch](#).

10.356.3 Member Function Documentation

10.356.3.1 `template<typename T> util::tree_node< T > & mln::util::branch< T >::apex ()` `[inline]`

The getter of the apex.

Returns:

- The [tree_node](#) apex of the current [branch](#).

10.356.3.2 `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](#).

10.357 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.

10.357.1 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](#).

10.357.2 Member Function Documentation

10.357.2.1 `template<typename T> unsigned mln::util::branch_iter< T >::deepness () const` `[inline]`

Give how deep is the iterator in the [branch](#).

References `mln::util::branch_iter< T >::is_valid()`, and `mln::util::tree_node< T >::parent()`.

10.357.2.2 `template<typename T> void mln::util::branch_iter< T >::invalidate ()` `[inline]`

Invalidate the iterator.

Referenced by `mln::util::branch_iter< T >::next()`.

10.357.2.3 `template<typename T> bool mln::util::branch_iter< T >::is_valid () const`
[inline]

Test the iterator validity.

Referenced by mln::util::branch_iter< T >::deepness().

10.357.2.4 `template<typename T> void mln::util::branch_iter< T >::next ()` [inline]

Go to the next [point](#).

References mln::util::branch_iter< T >::invalidate().

10.357.2.5 `template<typename T> mln::util::branch_iter< T >::operator util::tree_node< T >`
`& () const` [inline]

Conversion to [node](#).

10.357.2.6 `template<typename T> void mln::util::branch_iter< T >::start ()` [inline]

Start an iteration.

10.358 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.

10.358.1 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](#).

10.358.2 Member Function Documentation

10.358.2.1 `template<typename T> unsigned mln::util::branch_iter_ind< T >::deepness () const`
[inline]

Give how deep is the iterator in the [branch](#).

References `mln::util::branch_iter_ind< T >::is_valid()`, and `mln::util::tree_node< T >::parent()`.

10.358.2.2 `template<typename T> void mln::util::branch_iter_ind< T >::invalidate ()`
[inline]

Invalidate the iterator.

Referenced by `mln::util::branch_iter_ind< T >::next()`.

10.358.2.3 `template<typename T> bool mln::util::branch_iter_ind< T >::is_valid () const`
[inline]

Test the iterator validity.

Referenced by mln::util::branch_iter_ind< T >::deepness().

10.358.2.4 `template<typename T> void mln::util::branch_iter_ind< T >::next ()` [inline]

Go to the next [point](#).

References mln::util::branch_iter_ind< T >::invalidate().

10.358.2.5 `template<typename T> mln::util::branch_iter_ind< T >::operator util::tree_node< T > & () const` [inline]

Conversion to [node](#).

10.358.2.6 `template<typename T> void mln::util::branch_iter_ind< T >::start ()` [inline]

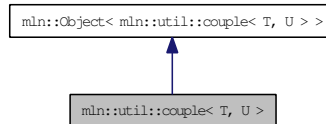
Start an iteration.

10.359 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](#).

10.359.1 Detailed Description

```
template<typename T, typename U> class mln::util::couple< T, U >
```

Definition of a [couple](#).

10.359.2 Member Function Documentation

10.359.2.1 `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*.

10.359.2.2 `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*.

10.359.2.3 `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*.

10.359.2.4 `template<typename T, typename U> const T & mln::util::couple< T, U >::first () const [inline]`

Get the first member of the [couple](#).

10.359.2.5 `template<typename T, typename U> const U & mln::util::couple< T, U >::second () const [inline]`

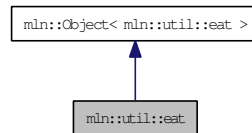
Get the second member of the [couple](#).

10.360 mln::util::eat Struct Reference

Eat structure.

```
#include <eat.hh>
```

Inheritance diagram for mln::util::eat:



10.360.1 Detailed Description

Eat structure.

10.361 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

- void [change_graph](#) (const G &g)
Set g_ with g;.
- const G & [graph](#) () const
Return a reference to the graph holding this edge.
- [edge_id_t](#) [id](#) () const
Return the edge id.
- void [invalidate](#) ()
Invalidate that vertex.
- bool [is_valid](#) () const
Misc.
- operator [edge_id_t](#) () const
Conversion to the edge id.
- void [update_id](#) (const [edge_id_t](#) &id)
Set id_ with id;.
- [edge](#) ()
Constructors.
- [edge_id_t](#) [ith_nbh_edge](#) (unsigned i) const

Return the i th adjacent *edge*.

- `size_t nmax_nbh_edges () const`
Return the number max of adjacent edges.
- `vertex_id_t v1 () const`
Edge oriented.
- `vertex_id_t v2 () const`
Return the highest *vertex* id adjacent to this *edge*.
- `vertex_id_t v_other (const vertex_id_t &id_v) const`
Vertex and edges oriented.

10.361.1 Detailed Description

```
template<typename G> class mln::util::edge< G >
```

Edge of a [graph](#) G.

10.361.2 Member Typedef Documentation

10.361.2.1 `template<typename G> typedef Edge<void> mln::util::edge< G >::category`

Object category.

10.361.2.2 `template<typename G> typedef G mln::util::edge< G >::graph_t`

Graph associated type.

10.361.2.3 `template<typename G> typedef edge_id_t mln::util::edge< G >::id_t`

The [edge](#) type id.

10.361.2.4 `template<typename G> typedef edge_id_t::value_t mln::util::edge< G >::id_value_t`

The underlying type used to store [edge](#) ids.

10.361.3 Constructor & Destructor Documentation

10.361.3.1 `template<typename G> mln::util::edge< G >::edge () [inline]`

Constructors.

References `mln::util::edge< G >::invalidate()`.

10.361.4 Member Function Documentation

10.361.4.1 `template<typename G> void mln::util::edge< G >::change_graph (const G & g)`
[inline]

Set `g_` with `g`;

10.361.4.2 `template<typename G> const G & mln::util::edge< G >::graph () const` [inline]

Return a reference to the [graph](#) holding this [edge](#).

Referenced by `mln::p_edges< G, F >::has()`, and `mln::util::line_graph< G >::has()`.

10.361.4.3 `template<typename G> edge_id_t mln::util::edge< G >::id () const` [inline]

Return the [edge](#) id.

Referenced by `mln::util::line_graph< G >::has()`.

10.361.4.4 `template<typename G> void mln::util::edge< G >::invalidate ()` [inline]

Invalidate that [vertex](#).

Referenced by `mln::util::edge< G >::edge()`.

10.361.4.5 `template<typename G> bool mln::util::edge< G >::is_valid () const` [inline]

Misc.

Return whether is points to a known [edge](#).

References `mln::util::object_id< Tag, V >::is_valid()`.

Referenced by `mln::p_edges< G, F >::has()`.

10.361.4.6 `template<typename G> edge_id_t mln::util::edge< G >::ith_nbh_edge (unsigned i)`
`const` [inline]

Return the `i` th adjacent [edge](#).

10.361.4.7 `template<typename G> size_t mln::util::edge< G >::nmax_nbh_edges () const`
[inline]

Return the number max of adjacent edges.

10.361.4.8 `template<typename G> mln::util::edge< G >::operator edge_id_t () const`
[inline]

Conversion to the [edge](#) id.

10.361.4.9 `template<typename G> void mln::util::edge< G >::update_id (const edge_id_t & id)`
[inline]

Set `id_` with `id`;

10.361.4.10 `template<typename G> vertex_id_t mln::util::edge< G >::v1 () const` [inline]

[Edge](#) oriented.

Return the lowest [vertex](#) id adjacent to this [edge](#).

Referenced by `mln::util::edge< G >::v_other()`.

10.361.4.11 `template<typename G> vertex_id_t mln::util::edge< G >::v2 () const` [inline]

Return the highest [vertex](#) id adjacent to this [edge](#).

Referenced by `mln::util::edge< G >::v_other()`.

10.361.4.12 `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`.

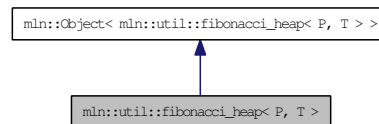
References `mln::util::edge< G >::v1()`, and `mln::util::edge< G >::v2()`.

10.362 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](#) (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.
- [fibonacci_heap](#) ()
Default constructor.
- 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](#) ([fibonacci_heap](#)< P, T > &other_heap)
Take other_heap' s elements and insert them in this heap.
- void [push](#) (const P &priority, const T &value)
Push a new element in the heap.

10.362.1 Detailed Description

`template<typename P, typename T> class mln::util::fibonacci_heap< P, T >`

Fibonacci heap.

10.362.2 Constructor & Destructor Documentation

10.362.2.1 `template<typename P, typename T> mln::util::fibonacci_heap< P, T >::fibonacci_heap () [inline]`

Default constructor.

10.362.2.2 `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.

References `mln::util::fibonacci_heap< P, T >::min_root`, `mln::util::fibonacci_heap< P, T >::num_marked_nodes`, `mln::util::fibonacci_heap< P, T >::num_nodes`, and `mln::util::fibonacci_heap< P, T >::num_trees`.

10.362.3 Member Function Documentation

10.362.3.1 `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.

References `mln::util::fibonacci_heap< P, T >::pop_front()`.

10.362.3.2 `template<typename P, typename T> const T & mln::util::fibonacci_heap< P, T >::front () const [inline]`

Return the minimum `value` in the heap.

10.362.3.3 `template<typename P, typename T> bool mln::util::fibonacci_heap< P, T >::is_empty () const [inline]`

Is it empty?

Referenced by `mln::util::fibonacci_heap< P, T >::pop_front()`, and `mln::util::fibonacci_heap< P, T >::push()`.

10.362.3.4 `template<typename P, typename T> bool mln::util::fibonacci_heap< P, T >::is_valid () const [inline]`

return false if it is empty.

Referenced by `mln::util::fibonacci_heap< P, T >::pop_front()`.

10.362.3.5 `template<typename P, typename T> unsigned mln::util::fibonacci_heap< P, T >::nelements () const [inline]`

Return the number of elements.

10.362.3.6 `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.

References `mln::util::fibonacci_heap< P, T >::min_root`, `mln::util::fibonacci_heap< P, T >::num_marked_nodes`, `mln::util::fibonacci_heap< P, T >::num_nodes`, and `mln::util::fibonacci_heap< P, T >::num_trees`.

10.362.3.7 `template<typename P, typename T> T mln::util::fibonacci_heap< P, T >::pop_front () [inline]`

Return and remove the minimum [value](#) in the heap.

References `mln::util::fibonacci_heap< P, T >::is_empty()`, `mln::util::fibonacci_heap< P, T >::is_valid()`, `mln::util::fibonacci_heap< P, T >::min_root`, and `mln::util::fibonacci_heap< P, T >::push()`.

Referenced by `mln::util::fibonacci_heap< P, T >::clear()`.

10.362.3.8 `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.

References `mln::util::fibonacci_heap< P, T >::is_empty()`, `mln::util::fibonacci_heap< P, T >::min_root`, `mln::util::fibonacci_heap< P, T >::num_marked_nodes`, `mln::util::fibonacci_heap< P, T >::num_nodes`, and `mln::util::fibonacci_heap< P, T >::num_trees`.

10.362.3.9 `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`

Referenced by `mln::util::fibonacci_heap< P, T >::pop_front()`.

10.363 mln::util::graph Class Reference

Undirected [graph](#).

```
#include <graph.hh>
```

Inherits [mln::util::internal::graph_base< mln::util::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::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.
- 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.

Public Member Functions

- [graph](#) (unsigned nvertices)
Construct a graph with nvertices vertices.
- [graph](#) ()
- 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 .

- `edge_id_t add_edge` (const `vertex_id_t` & id_v1 , const `vertex_id_t` & id_v2)
Edge oriented.
- `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 .
- `size_t e_nmax` () const
Return the number of edges in the *graph*.
- `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_t edge` (const `vertex_t` & $v1$, const `vertex_t` & $v2$) const
@}
- `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.
- `bool has_e` (const `edge_id_t` & id_e) const
Return whether id_e is in the *graph*.
- `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*.
- `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 .

- `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.

10.363.1 Detailed Description

Undirected [graph](#).

10.363.2 Member Typedef Documentation

10.363.2.1 `typedef mln::internal::edge_fwd_iterator<graph> mln::util::graph::edge_fwd_iter`

[Edge](#) iterators.

10.363.2.2 `typedef mln::internal::edge_nbh_edge_fwd_iterator<graph>
mln::util::graph::edge_nbh_edge_fwd_iter`

[Edge](#) centered [edge](#) iterators.

10.363.2.3 `typedef std::set<edge_data_t> mln::util::graph::edges_set_t`

A [set](#) to [test](#) the presence of a given [edge](#).

10.363.2.4 `typedef std::vector<edge_data_t> mln::util::graph::edges_t`

The type of the [set](#) of edges.

10.363.2.5 `typedef mln::internal::vertex_fwd_iterator<graph> mln::util::graph::vertex_fwd_iter`

[Iterator](#) types

[Vertex](#) iterators.

10.363.2.6 `typedef mln::internal::vertex_nbh_edge_fwd_iterator<graph>
mln::util::graph::vertex_nbh_edge_fwd_iter`

[Vertex](#) centered [edge](#) iterators.

10.363.2.7 `typedef mln::internal::vertex_nbh_vertex_fwd_iterator<graph>
mln::util::graph::vertex_nbh_vertex_fwd_iter`

[Vertex](#) centered [vertex](#) iterators.

10.363.2.8 typedef std::vector<vertex_data_t> mln::util::graph::vertices_t

The type of the [set](#) of vertices.

10.363.3 Constructor & Destructor Documentation

10.363.3.1 mln::util::graph::graph () [inline]

Constructor.

10.363.3.2 mln::util::graph::graph (unsigned *nvertices*) [inline]

Construct a [graph](#) with `nvertices` vertices.

10.363.4 Member Function Documentation

10.363.4.1 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)`.

References `edge()`, and `has_v()`.

Referenced by `mln::make::voronoi()`.

10.363.4.2 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](#).

References `v_nmax()`.

Referenced by `mln::make::voronoi()`.

10.363.4.3 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.

References `v_nmax()`.

10.363.4.4 `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`.

References `e_nmax()`, `e_nmax_nbh_edges()`, `has_e()`, `v1()`, `v2()`, `v_ith_nbh_edge()`, and `v_nmax_nbh_edges()`.

10.363.4.5 `size_t mln::util::graph::e_nmax () const` [inline]

Return the number of edges in the `graph`.

Referenced by `e_ith_nbh_edge()`, and `edge()`.

10.363.4.6 `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`.

References `has_e()`, `v1()`, `v2()`, and `v_nmax_nbh_edges()`.

Referenced by `e_ith_nbh_edge()`.

10.363.4.7 `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`.

References `has_v()`, and `mln::util::vertex< G >::id()`.

10.363.4.8 `graph::edge_t mln::util::graph::edge (const edge_id_t & e) const` [inline]

Return the `edge` whose id is *e*.

References `e_nmax()`.

Referenced by `add_edge()`.

10.363.4.9 `const std::vector< util::ord_pair< vertex_id_t > > & mln::util::graph::edges () const` [inline]

Return the list of all edges.

10.363.4.10 `bool mln::util::graph::has_e (const edge_id_t & id_e) const` [inline]

Return whether `id_e` is in the `graph`.

@{

Referenced by `e_ith_nbh_edge()`, `e_nmax_nbh_edges()`, `v1()`, and `v2()`.

10.363.4.11 `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](#).

Referenced by `add_edge()`, `edge()`, `v_ith_nbh_edge()`, `v_ith_nbh_vertex()`, `v_nmax_nbh_edges()`, `v_nmax_nbh_vertices()`, and `vertex()`.

10.363.4.12 `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`.

10.363.4.13 `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`.

References `has_e()`.

Referenced by `e_ith_nbh_edge()`, and `e_nmax_nbh_edges()`.

10.363.4.14 `vertex_id_t mln::util::graph::v2 (const edge_id_t & id_e) const` [inline]

Return the second [vertex](#) associated to [edge](#) `id_e`.

References `has_e()`.

Referenced by `e_ith_nbh_edge()`, and `e_nmax_nbh_edges()`.

10.363.4.15 `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`.

References `has_v()`, and `v_nmax_nbh_edges()`.

Referenced by `e_ith_nbh_edge()`, and `v_ith_nbh_vertex()`.

10.363.4.16 `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`.

References `has_v()`, and `v_ith_nbh_edge()`.

10.363.4.17 `size_t mln::util::graph::v_nmax () const` [inline]

Return the number of vertices in the [graph](#).

Referenced by `add_vertex()`, and `add_vertices()`.

10.363.4.18 `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`.

References `has_v()`.

Referenced by `e_ith_nbh_edge()`, `e_nmax_nbh_edges()`, `v_ith_nbh_edge()`, and `v_nmax_nbh_vertices()`.

10.363.4.19 `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`.

References `has_v()`, and `v_nmax_nbh_edges()`.

10.363.4.20 `graph::vertex_t mln::util::graph::vertex (vertex_id_t id_v) const` [inline]

Return the `vertex` whose id is `v`.

References `has_v()`.

10.364 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?

10.364.1 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.

10.364.2 Member Function Documentation

10.364.2.1 `template<typename I> bool mln::util::greater_point< I >::operator() (const point &x, const point &y) [inline]`

Is x greater than y?

10.365 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?

10.365.1 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.

10.365.2 Member Function Documentation

10.365.2.1 `template<typename I> bool mln::util::greater_psite< I >::operator() (const psite & x, const psite & y) [inline]`

Is x greater than y?

10.366 mln::util::head< T, R > Class Template Reference

Top structure of the soft heap.

```
#include <soft_heap.hh>
```

10.366.1 Detailed Description

```
template<typename T, typename R> class mln::util::head< T, R >
```

Top structure of the soft heap.

10.367 mln::util::ignore Struct Reference

Ignore structure.

```
#include <ignore.hh>
```

Inheritance diagram for mln::util::ignore:



10.367.1 Detailed Description

Ignore structure.

10.368 mln::util::icell< T > Struct Template Reference

Element of an item list. Store the [data](#) (key) used in [soft_heap](#).

```
#include <soft_heap.hh>
```

10.368.1 Detailed Description

```
template<typename T> struct mln::util::icell< T >
```

Element of an item list. Store the [data](#) (key) used in [soft_heap](#).

10.369 mln::util::line_graph< G > Class Template Reference

Undirected line [graph](#) of a [graph](#) of type G.

```
#include <line_graph.hh>
```

Inherits mln::util::internal::graph_base< mln::util::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::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_fwd_iterator< [line_graph](#)< G > > [vertex_fwd_iter](#)
Iterator types
Vertex iterators.
- typedef mln::internal::vertex_nbh_edge_fwd_iterator< [line_graph](#)< G > > [vertex_nbh_edge_fwd_iter](#)
Vertex nbh edge iterators.
- typedef mln::internal::vertex_nbh_vertex_fwd_iterator< [line_graph](#)< G > > [vertex_nbh_vertex_fwd_iter](#)
Vertex nbh vertex iterators.

Public Member Functions

- template<typename G2>
bool [has](#) (const [util::vertex](#)< G2 > &v) const
Check whether an [edge](#) v exists in the [graph](#).
- 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 .

- `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 .

- `size_t e_nmax` () const

Return the number of edges in the *graph*.

- `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_t edge` (const `edge_id_t` & e) const

Edge oriented.

- const `G` & `graph` () const

Return the underlying *graph*.

- template<typename G2>

`bool has` (const `util::edge`< G2 > & e) const

Return whether e is in the *graph*.

- `bool has_e` (const `util::edge_id_t` & id_e) const

Return whether id_e is in the *graph*.

- 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*.

- `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 .

- `vertex_t vertex` (const `vertex_id_t` & id_v) const

Vertex oriented.

10.369.1 Detailed Description

```
template<typename G> class mln::util::line_graph< G >
```

Undirected line [graph](#) of a [graph](#) of type G.

10.369.2 Member Typedef Documentation

10.369.2.1 `template<typename G> typedef mln::internal::edge_fwd_iterator< line_graph<G>
> mln::util::line_graph< G >::edge_fwd_iter`

[Edge](#) iterators.

10.369.2.2 `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.

10.369.2.3 `template<typename G> typedef std::vector<edge_data_t> mln::util::line_graph< G
>::edges_t`

The type of the [set](#) of edges.

10.369.2.4 `template<typename G> typedef mln::internal::vertex_fwd_iterator< line_graph<G>
> mln::util::line_graph< G >::vertex_fwd_iter`

[Iterator](#) types

[Vertex](#) iterators.

10.369.2.5 `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.

10.369.2.6 `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.

10.369.2.7 `template<typename G> typedef std::vector<vertex_data_t> mln::util::line_graph<
G >::vertices_t`

The type of the [set](#) of vertices.

10.369.3 Member Function Documentation

10.369.3.1 `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*.

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()`.

10.369.3.2 `template<typename G> size_t mln::util::line_graph< G >::e_nmax () const` [inline]

Return the number of edges in the [graph](#).

Referenced by `mln::util::line_graph< G >::e_ith_nbh_edge()`, and `mln::util::line_graph< G >::edge()`.

10.369.3.3 `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*.

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()`.

10.369.3.4 `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*.

References `mln::util::line_graph< G >::e_nmax()`.

10.369.3.5 `template<typename G> const G & mln::util::line_graph< G >::graph () const` [inline]

Return the underlying [graph](#).

10.369.3.6 `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 [graph](#).

References `mln::util::edge< G >::graph()`, `mln::util::line_graph< G >::has_e()`, and `mln::util::edge< G >::id()`.

10.369.3.7 `template<typename G> template<typename G2> bool mln::util::line_graph< G >::has (const util::vertex< G2 > & v) const [inline]`

Check whether an [edge](#) `v` exists in the [graph](#).

References `mln::util::vertex< G >::graph()`, `mln::util::line_graph< G >::has_v()`, and `mln::util::vertex< G >::id()`.

10.369.3.8 `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 [graph](#).

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()`.

10.369.3.9 `template<typename G> bool mln::util::line_graph< G >::has_v (const vertex_id_t & id_v) const [inline]`

Check whether a [vertex](#) `id_v` exists in the [graph](#).

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()`.

10.369.3.10 `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`.

10.369.3.11 `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`.

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()`.

10.369.3.12 `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`.

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()`.

10.369.3.13 `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*.

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()`.

10.369.3.14 `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*.

References `mln::util::line_graph< G >::has_v()`, and `mln::util::line_graph< G >::v_ith_nbh_edge()`.

10.369.3.15 `template<typename G> size_t mln::util::line_graph< G >::v_nmax () const` [inline]

Return the number of vertices in the [graph](#).

Referenced by `mln::util::line_graph< G >::v_ith_nbh_edge()`.

10.369.3.16 `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*.

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()`.

10.369.3.17 `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*.

References `mln::util::line_graph< G >::has_v()`, and `mln::util::line_graph< G >::v_nmax_nbh_edges()`.

10.369.3.18 `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*.

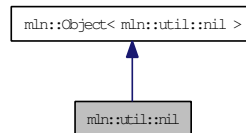
References `mln::util::line_graph< G >::has_v()`.

10.370 mln::util::nil Struct Reference

Nil structure.

```
#include <nil.hh>
```

Inheritance diagram for mln::util::nil:



10.370.1 Detailed Description

Nil structure.

10.371 mln::util::node< T, R > Class Template Reference

Meta-data of an element in the heap.

```
#include <soft_heap.hh>
```

10.371.1 Detailed Description

```
template<typename T, typename R> class mln::util::node< T, R >
```

Meta-data of an element in the heap.

10.372 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.

10.372.1 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](#).

10.372.2 Member Typedef Documentation

10.372.2.1 `template<typename Tag, typename V> typedef V mln::util::object_id< Tag, V >::value_t`

The underlying type id.

10.372.3 Constructor & Destructor Documentation

10.372.3.1 `template<typename Tag, typename V> mln::util::object_id< Tag, V >::object_id()`
[inline]

Constructors.

10.373 `mln::util::ord< T >` Struct Template Reference

Function-object that defines an ordering between objects with type `T`: *lhs R rhs*.

```
#include <ord.hh>
```

10.373.1 Detailed Description

```
template<typename T> struct mln::util::ord< T >
```

Function-object that defines an ordering between objects with type `T`: *lhs R rhs*.

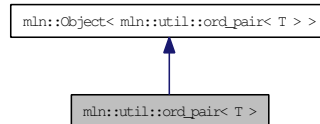
Its meaning is "lhs less-than rhs."

10.374 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.

10.374.1 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.

10.374.2 Member Function Documentation

10.374.2.1 `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).

References mln::util::ord_strict(), and mln::util::ord_weak().

10.374.2.2 `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).

References mln::util::ord_strict(), and mln::util::ord_weak().

10.374.2.3 `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).

References mln::util::ord_strict(), and mln::util::ord_weak().

10.374.2.4 `template<typename T> const T & mln::util::ord_pair< T >::first () const`
[inline]

Get the first (lowest) member of the pair.

10.374.2.5 `template<typename T> const T & mln::util::ord_pair< T >::second () const`
[inline]

Get the second (highest) member of the pair.

10.375 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.

10.375.1 Detailed Description

```
template<typename I> struct mln::util::pix< I >
```

Structure [pix](#).

10.375.2 Member Typedef Documentation

10.375.2.1 template<typename I> typedef I::psite mln::util::pix< I >::psite

[Point_Site](#) associated type.

10.375.2.2 template<typename I> typedef I::value mln::util::pix< I >::value

[Value](#) associated type.

10.375.3 Constructor & Destructor Documentation

10.375.3.1 `template<typename I> mln::util::pix< I >::pix (const Image< I > & ima, const typename I::psite & p)` [inline]

Constructor.

Parameters:

← *ima* The image.

← *p* The p_site.

10.375.4 Member Function Documentation

10.375.4.1 `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_.

10.375.4.2 `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_.

10.375.4.3 `template<typename I> I::rvalue mln::util::pix< I >::v () const` [inline]

The getter of [value](#) associate to [pix](#) structure.

Returns:

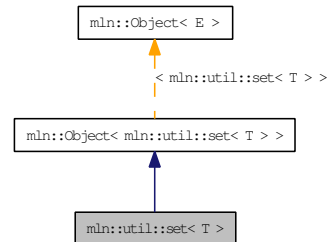
The [value](#) of [pix](#).

10.376 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](#).
- template<typename U>
`set< T >` & [insert](#) (const `set< U >` &other)
Insert the elements of `other` into the [set](#).
- `set< T >` & [insert](#) (const `T` &elt)
Insert an element `elt` 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.

10.376.1 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](#)

10.376.2 Member Typedef Documentation

10.376.2.1 template<typename T> typedef set_bkd_iter<T> mln::util::set< T >::bkd_eiter

Backward iterator associated type.

10.376.2.2 `template<typename T> typedef fwd_eiter mln::util::set< T >::eiter`

Iterator associated type.

10.376.2.3 `template<typename T> typedef T mln::util::set< T >::element`

Element associated type.

10.376.2.4 `template<typename T> typedef set_fwd_iter<T> mln::util::set< T >::fwd_eiter`

Forward iterator associated type.

10.376.3 Constructor & Destructor Documentation**10.376.3.1** `template<typename T> mln::util::set< T >::set () [inline]`

Constructor without arguments.

10.376.4 Member Function Documentation**10.376.4.1** `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`

References `mln::util::set< T >::is_empty()`.

10.376.4.2 `template<typename T> const T mln::util::set< T >::first_element () const [inline]`

Return the first element of the [set](#).

Precondition:

`not is_empty()`

References `mln::util::set< T >::is_empty()`.

10.376.4.3 `template<typename T> bool mln::util::set< T >::has (const T & elt) const [inline]`

Test if the object `elt` belongs to the [set](#).

Parameters:

← *elt* A possible element of the [set](#).

Returns:

True if `elt` is in the [set](#).

10.376.4.4 `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:

← *other* The [set](#) containing the elements to be inserted.

Returns:

The [set](#) itself after insertion.

References `mln::util::set< T >::is_empty()`, and `mln::util::set< T >::std_vector()`.

10.376.4.5 `template<typename T> set< T > & mln::util::set< T >::insert (const T & elt) [inline]`

Insert an element `elt` into the [set](#).

Parameters:

← *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.

Referenced by `mln::p_key< K, P >::change_keys()`.

10.376.4.6 `template<typename T> bool mln::util::set< T >::is_empty () const [inline]`

Test if the [set](#) is empty.

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()`.

10.376.4.7 `template<typename T> const T mln::util::set< T >::last_element () const [inline]`

Return the last element of the [set](#).

Precondition:

not `is_empty()`

References `mln::util::set< T >::is_empty()`.

10.376.4.8 `template<typename T> std::size_t mln::util::set< T >::memory_size () const`
`[inline]`

Return the size of this [set](#) in memory.

References `mln::util::set< T >::nelements()`.

10.376.4.9 `template<typename T> unsigned mln::util::set< T >::nelements () const` `[inline]`

Return the number of elements of the [set](#).

Referenced by `mln::util::set< T >::is_empty()`, `mln::util::set< T >::memory_size()`, and `mln::util::set< T >::operator[]()`.

10.376.4.10 `]`

`template<typename T> const T & mln::util::set< T >::operator[] (unsigned i) const` `[inline]`

Return the *i*-th element of the [set](#).

Parameters:

← *i* Index of the element to retrieve.

Precondition:

i < `nelements()`

The element is returned by reference and is constant.

References `mln::util::set< T >::nelements()`.

10.376.4.11 `template<typename T> set< T > & mln::util::set< T >::remove (const T & elt)`
`[inline]`

Remove an element `elt` into the [set](#).

Parameters:

← *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.

10.376.4.12 `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.

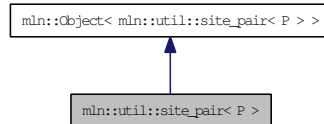
Referenced by mln::util::set< T >::insert().

10.377 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.

10.377.1 Detailed Description

```
template<typename P> class mln::util::site_pair< P >
```

A pair of sites.

It can be used as site.

10.377.2 Member Function Documentation

10.377.2.1 `template<typename P> const P & mln::util::site_pair< P >::first () const` `[inline]`

Return the first site.

10.377.2.2 `template<typename P> const util::ord_pair< P > & mln::util::site_pair< P >::pair () const` `[inline]`

Return the underlying pair.

10.377.2.3 `template<typename P> const P & mln::util::site_pair< P >::second () const` `[inline]`

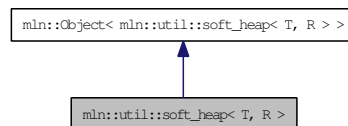
Return the second site.

10.378 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.

10.378.1 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

10.378.2 Member Typedef Documentation

10.378.2.1 template<typename T, typename R> typedef T mln::util::soft_heap< T, R >::element

Element associated type.

10.378.3 Constructor & Destructor Documentation

10.378.3.1 template<typename T, typename R> mln::util::soft_heap< T, R >::soft_heap (unsigned r = 20) [inline]

Default constructor.

A corruption threshold r 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.

10.378.3.2 template<typename T, typename R> mln::util::soft_heap< T, R >::~~soft_heap () [inline]

Destructor.

References `mln::util::head< T, R >::next()`, and `mln::util::head< T, R >::queue()`.

10.378.4 Member Function Documentation

10.378.4.1 template<typename T, typename R> void mln::util::soft_heap< T, R >::clear () [inline]

Clear the heap.

References `mln::util::head< T, R >::next()`, `mln::util::head< T, R >::queue()`, `mln::util::head< T, R >::set_next()`, and `mln::util::head< T, R >::set_prev()`.

10.378.4.2 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.

10.378.4.3 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.

Referenced by `mln::util::soft_heap< T, R >::pop_front()`.

10.378.4.4 `template<typename T, typename R> int mln::util::soft_heap< T, R >::nelements () const` [inline]

Return the number of element in the heap.

Referenced by mln::util::soft_heap< T, R >::push().

10.378.4.5 `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.

References mln::util::soft_heap< T, R >::is_valid(), mln::util::head< T, R >::next(), mln::util::node< T, R >::next(), mln::util::head< T, R >::prev(), mln::util::head< T, R >::queue(), and mln::util::head< T, R >::set_queue().

10.378.4.6 `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.

References mln::util::soft_heap< T, R >::nelements(), mln::util::head< T, R >::next(), and mln::util::head< T, R >::queue().

10.378.4.7 `template<typename T, typename R> void mln::util::soft_heap< T, R >::push (const T & element)` [inline]

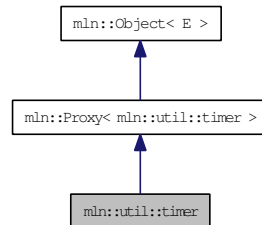
Add a new element element.

10.379 mln::util::timer Class Reference

Timer structure.

```
#include <timer.hh>
```

Inheritance diagram for mln::util::timer:



10.379.1 Detailed Description

Timer structure.

10.380 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 (const tracked_ptr< T > &rhs)`
Copy constructor.
- `tracked_ptr ()`
Constructors.

10.380.1 Detailed Description

```
template<typename T> struct mln::util::tracked_ptr< T >
```

Smart pointer for shared [data](#) with tracking.

10.380.2 Constructor & Destructor Documentation

10.380.2.1 `template<typename T> mln::util::tracked_ptr< T >::tracked_ptr () [inline]`

Constructors.

10.380.2.2 `template<typename T> mln::util::tracked_ptr< T >::tracked_ptr (const tracked_ptr< T > & rhs) [inline]`

Copy constructor.

10.380.2.3 `template<typename T> mln::util::tracked_ptr< T >::~~tracked_ptr () [inline]`

Destructor.

10.380.3 Member Function Documentation

10.380.3.1 `template<typename T> mln::util::tracked_ptr< T >::operator bool () const [inline]`

Coercion towards Boolean (for arithmetical tests).

10.380.3.2 `template<typename T> bool mln::util::tracked_ptr< T >::operator! () const [inline]`

Negation (for arithmetical tests).

10.380.3.3 `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.

10.380.3.4 `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.

10.380.3.5 `template<typename T> tracked_ptr< T > & mln::util::tracked_ptr< T >::operator= (T * ptr) [inline]`

Assignment.

10.380.3.6 `template<typename T> tracked_ptr< T > & mln::util::tracked_ptr< T >::operator= (const tracked_ptr< T > & rhs) [inline]`

Assignment.

10.381 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](#) ([tree_node](#)< T > *root)
Constructor.
- [tree](#) ()
Constructor.

10.381.1 Detailed Description

```
template<typename T> class mln::util::tree< T >
```

Class of generic [tree](#).

10.381.2 Constructor & Destructor Documentation

10.381.2.1 `template<typename T> mln::util::tree< T >::tree ()` [`inline`]

Constructor.

10.381.2.2 `template<typename T> mln::util::tree< T >::tree (tree_node< T > * root)`
[`inline`]

Constructor.

Parameters:

← *root* The root of the [tree](#).

10.381.3 Member Function Documentation

10.381.3.1 `template<typename T> void mln::util::tree< T >::add_tree_down (T & elt)`
[inline]

Bind a new [tree](#) downer the current.

Parameters:

← *elt* The new [value](#) of the new [tree_node](#) of the new [tree](#) add downer the current.

10.381.3.2 `template<typename T> void mln::util::tree< T >::add_tree_up (T & elt)` [inline]

Bind a new [tree](#) upper the current.

Parameters:

← *elt* The new [value](#) of the new [tree_node](#) of the new [tree](#) add upper the current.

References `mln::util::tree_node< T >::children()`.

10.381.3.3 `template<typename T> bool mln::util::tree< T >::check_consistency ()` [inline]

Check the consistency of the [tree](#).

Returns:

true if no error, else false.

References `mln::util::tree< T >::root()`.

10.381.3.4 `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](#).

References `mln::util::tree< T >::root()`.

10.381.3.5 `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](#).

Referenced by `mln::util::tree< T >::check_consistency()`, `mln::util::display_tree()`, `mln::util::tree< T >::main_branch()`, and `mln::util::tree_to_fast()`.

10.382 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](#) ([tree_node](#)< T > *[tree_node](#))
Bind [tree_node](#) to the current [tree_node](#) and become its child.
- [tree_node](#)< T > * [add_child](#) (T elt)
Create a [tree_node](#) with elt which become the child of the current [tree_node](#).
- bool [check_consistency](#) ()
Check the consistency of the [tree_node](#).
- const children_t & [children](#) () const
The getter of the children.
- children_t & [children](#) ()
The getter of the children.
- [tree_node](#)< T > * [delete_tree_node](#) ()
Delete the current [tree_node](#).
- const T & [elt](#) () const
The const getter of the element.
- T & [elt](#) ()
The 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](#) (T elt)
Constructor.

- [tree_node](#) ()

Constructor.

10.382.1 Detailed Description

`template<typename T> class mln::util::tree_node< T >`

Class of generic [tree_node](#) for [tree](#).

10.382.2 Constructor & Destructor Documentation

10.382.2.1 `template<typename T> mln::util::tree_node< T >::tree_node ()` [inline]

Constructor.

10.382.2.2 `template<typename T> mln::util::tree_node< T >::tree_node (T elt)` [inline]

Constructor.

Parameters:

← *elt* The element of [tree_node](#).

10.382.3 Member Function Documentation

10.382.3.1 `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:

← *tree_node* The new child [tree_node](#).

Returns:

The child [tree_node](#).

References `mln::util::tree_node< T >::children()`, and `mln::util::tree_node< T >::parent()`.

10.382.3.2 `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:

← *elt* The element of the new child to add.

Returns:

The new [tree_node](#) created.

10.382.3.3 `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.

10.382.3.4 `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.

10.382.3.5 `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](#).

Referenced by `mln::util::tree_node< T >::add_child()`, and `mln::util::tree< T >::add_tree_up()`.

10.382.3.6 `template<typename T> tree_node< T > * mln::util::tree_node< T`
`>::delete_tree_node ()` [inline]

Delete the current [tree_node](#).

10.382.3.7 `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.

10.382.3.8 `template<typename T> T & mln::util::tree_node< T >::elt ()` [inline]

The getter of the element.

Returns:

The element of the [tree_node](#).

Referenced by `mln::util::tree_node< T >::print()`.

10.382.3.9 `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](#).

Referenced by `mln::util::tree_node< T >::add_child()`, `mln::util::branch_iter_ind< T >::deepness()`, and `mln::util::branch_iter< T >::deepness()`.

10.382.3.10 `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:

← *ostr* The output stream.

← *level* The deep level

References `mln::util::tree_node< T >::elt()`.

10.382.3.11 `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:

← *elt* The `value` of the searched [tree_node](#).

Returns:

If not found 0 else the [tree_node](#) with `elt` `value`.

References `mln::util::tree_node< T >::search_rec()`.

10.382.3.12 `template<typename T> int mln::util::tree_node< T >::search_rec (tree_node< T >`
`** res, T & elt)` [inline]

The using method for method search.

Referenced by `mln::util::tree_node< T >::search()`.

10.382.3.13 `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:

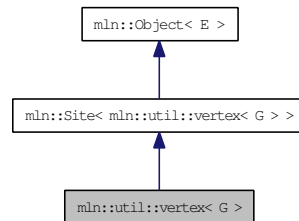
← *parent* The new parent [tree_node](#).

10.383 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.

10.383.1 Detailed Description

```
template<typename G> class mln::util::vertex< G >
```

Vertex of a *graph* *G*.

10.383.2 Member Typedef Documentation

10.383.2.1 `template<typename G> typedef Vertex<void> mln::util::vertex< G >::Category`

Object category.

10.383.2.2 `template<typename G> typedef G mln::util::vertex< G >::graph_t`

Graph associated type.

10.383.2.3 `template<typename G> typedef vertex_id_t mln::util::vertex< G >::id_t`

The *vertex* type id.

10.383.2.4 `template<typename G> typedef vertex_id_t::value_t mln::util::vertex< G >::id_value_t`

The underlying type used to store [vertex](#) ids.

10.383.3 Constructor & Destructor Documentation

10.383.3.1 `template<typename G> mln::util::vertex< G >::vertex () [inline]`

Constructors.

References `mln::util::vertex< G >::invalidate()`.

10.383.4 Member Function Documentation

10.383.4.1 `template<typename G> void mln::util::vertex< G >::change_graph (const G & g) [inline]`

Change the parent [graph](#) of that [vertex](#).

10.383.4.2 `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](#).

10.383.4.3 `template<typename G> const G & mln::util::vertex< G >::graph () const [inline]`

Returns the [graph](#) pointer this [vertex](#) belongs to.

Referenced by `mln::p_vertices< G, F >::has()`, `mln::util::line_graph< G >::has()`, and `mln::util::operator==(())`.

10.383.4.4 `template<typename G> const vertex_id_t & mln::util::vertex< G >::id () const [inline]`

Returns the [vertex](#) id.

Referenced by `mln::util::graph::edge()`, `mln::util::line_graph< G >::has()`, and `mln::util::operator==(())`.

10.383.4.5 `template<typename G> void mln::util::vertex< G >::invalidate () [inline]`

Invalidate that [vertex](#).

Referenced by `mln::util::vertex< G >::vertex()`.

10.383.4.6 `template<typename G> bool mln::util::vertex< G >::is_valid () const [inline]`

Check whether the [vertex](#) is still part of the [graph](#).

Referenced by `mln::p_vertices< G, F >::has()`.

10.383.4.7 `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](#).

10.383.4.8 `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](#).

10.383.4.9 `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](#).

10.383.4.10 `template<typename G> unsigned mln::util::vertex< G >::nmax_nbh_vertices () const [inline]`

Returns the number max of vertices adjacent to this [vertex](#).

10.383.4.11 `template<typename G> mln::util::vertex< G >::operator vertex_id_t () const [inline]`

Conversion to the [vertex](#) id.

FIXME: May cause ambiguities... :(

10.383.4.12 `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*.

10.383.4.13 `template<typename G> void mln::util::vertex< G >::update_id (const vertex_id_t & id) [inline]`

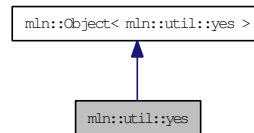
Update the [vertex](#) id.

10.384 mln::util::yes Struct Reference

[Object](#) that always says "yes".

```
#include <yes.hh>
```

Inheritance diagram for mln::util::yes:



10.384.1 Detailed Description

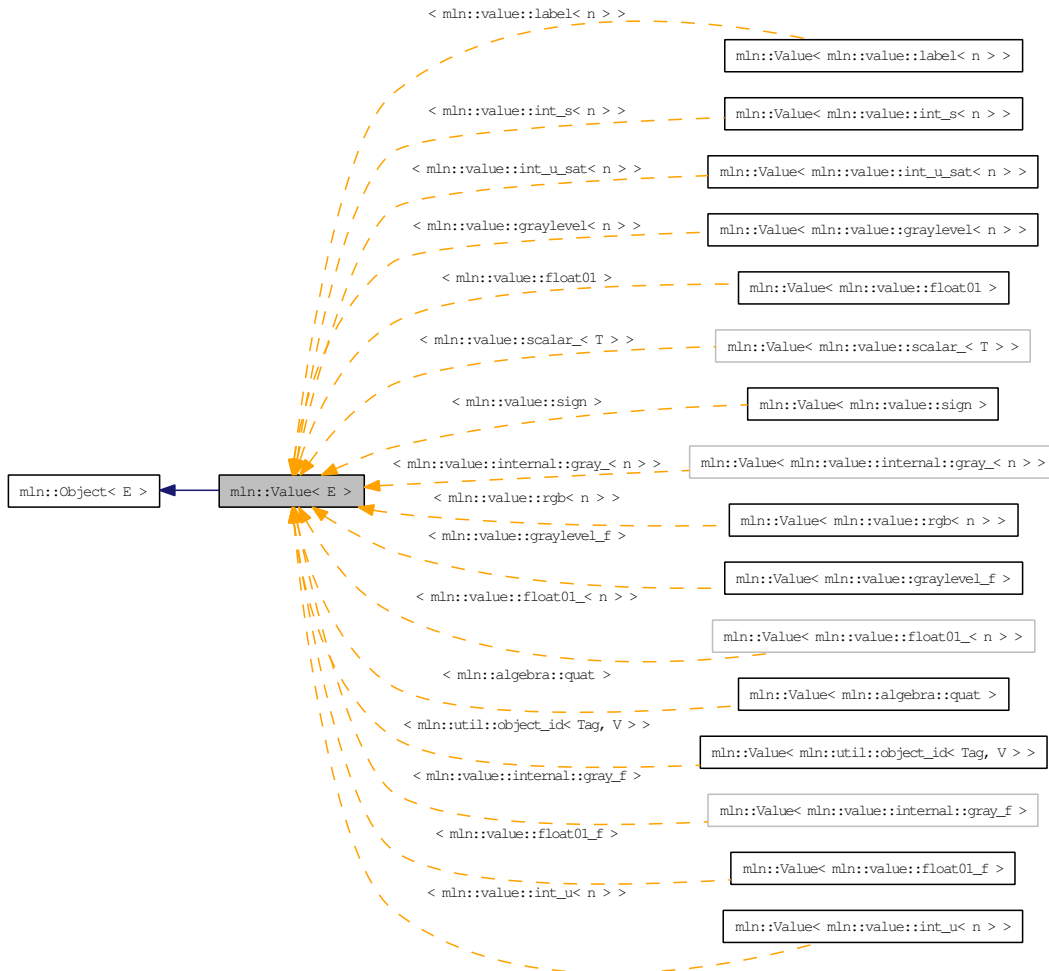
[Object](#) that always says "yes".

10.385 mln::Value< E > Struct Template Reference

Base class for implementation classes of values.

```
#include <value.hh>
```

Inheritance diagram for mln::Value< E >:



10.385.1 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.

10.386 mln::value::float01 Class Reference

Class for floating values restricted to the interval [0.

```
#include <float01.hh>
```

Inherits mln::value::Floating< mln::value::float01 >.

Public Types

- typedef std::pair< unsigned, unsigned long > [enc](#)
Encoding associated type.
- typedef float [equiv](#)
Equivalent associated type.

Public Member Functions

- [float01](#) (unsigned nbits, float val)
Ctor.
- template<unsigned n>
[float01](#) (const float01_< n > &val)
Ctor.
- [float01](#) ()
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.

10.386.1 Detailed Description

Class for floating values restricted to the interval [0.
.1] and discretized with n bits.

10.386.2 Member Typedef Documentation

10.386.2.1 `typedef std::pair<unsigned, unsigned long> mln::value::float01::enc`

Encoding associated type.

10.386.2.2 `typedef float mln::value::float01::equiv`

Equivalent associated type.

10.386.3 Constructor & Destructor Documentation

10.386.3.1 `mln::value::float01::float01 () [inline]`

Ctor.

10.386.3.2 `template<unsigned n> mln::value::float01::float01 (const float01_<n> & val) [inline]`

Ctor.

10.386.3.3 `mln::value::float01::float01 (unsigned nbits, float val) [inline]`

Ctor.

10.386.4 Member Function Documentation

10.386.4.1 `unsigned mln::value::float01::nbits () const [inline]`

Access to the encoding size.

10.386.4.2 `mln::value::float01::operator float () const [inline]`

Conversion to float.

10.386.4.3 `float01 & mln::value::float01::set_nbits (unsigned nbits) [inline]`

Set the encoding size to nbits.

Referenced by `to_nbits()`.

10.386.4.4 `const float01 mln::value::float01::to_nbits (unsigned nbits) const` [inline]

Return an equivalent gray encoded on `nbits` bits.

References `set_nbits()`.

10.386.4.5 `float mln::value::float01::value () const` [inline]

Access to std type.

10.386.4.6 `unsigned long mln::value::float01::value_ind () const` [inline]

Access to the position in the quantized interval.

10.387 mln::value::float01_f Struct Reference

Class for floating values restricted to the interval [0..1].

```
#include <float01_f.hh>
```

Inherits mln::value::Floating< mln::value::float01_f >, and mln::value::internal::value_like_< float, float, float, mln::value::float01_f >.

Public Member Functions

- [float01_f](#) (float val)
Constructor from a float.
- [float01_f](#) ()
Constructor without argument.
- [operator float](#) () const
Conversion to a float.
- [float01_f & operator=](#) (const float val)
Assignment from a float.
- float [value](#) () const
Access to float value.

10.387.1 Detailed Description

Class for floating values restricted to the interval [0..1].

10.387.2 Constructor & Destructor Documentation

10.387.2.1 mln::value::float01_f::float01_f () [inline]

Constructor without argument.

10.387.2.2 mln::value::float01_f::float01_f (float val) [inline]

Constructor from a float.

10.387.3 Member Function Documentation

10.387.3.1 mln::value::float01_f::operator float () const [inline]

Conversion to a float.

10.387.3.2 float01_f & mln::value::float01_f::operator= (const float *val*) [inline]

Assignment from a float.

10.387.3.3 float mln::value::float01_f::value () const [inline]

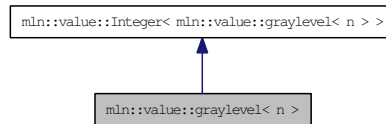
Access to float [value](#).

10.388 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

- `template<unsigned m>`
`graylevel` (const `graylevel< m >` &rhs)
Constructor from any graylevel.
- `graylevel` (int val)
Constructor from int.
- `graylevel` (const `graylevel< n >` &rhs)
Copy constructor.
- `graylevel` ()
Constructor without argument.
- `template<unsigned m>`
`graylevel< n >` & `operator=` (const `graylevel< m >` &rhs)
Assignment with any graylevel.
- `graylevel< n >` & `operator=` (int val)
Assignment with int.
- `graylevel< n >` & `operator=` (const `graylevel< n >` &rhs)
Assignment.
- `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` &)
Assignment with literals.

10.388.1 Detailed Description

`template<unsigned n> struct mln::value::graylevel< n >`

General gray-level class on n bits.

10.388.2 Constructor & Destructor Documentation

10.388.2.1 `template<unsigned n> mln::value::graylevel< n >::graylevel ()` [inline]

Constructor without argument.

10.388.2.2 `template<unsigned n> mln::value::graylevel< n >::graylevel (const graylevel< n > & rhs)` [inline]

Copy constructor.

10.388.2.3 `template<unsigned n> mln::value::graylevel< n >::graylevel (int val)` [inline]

Constructor from int.

10.388.2.4 `template<unsigned n> template<unsigned m> mln::value::graylevel< n >::graylevel (const graylevel< m > & rhs)` [inline]

Constructor from any [graylevel](#).

References `mln::value::graylevel< n >::value()`.

10.388.2.5 `template<unsigned n> mln::value::graylevel< n >::graylevel (const mln::literal::black_t &)` [inline]

Ctors with literals.

10.388.3 Member Function Documentation

10.388.3.1 `template<unsigned n> graylevel< n > & mln::value::graylevel< n >::operator= (const mln::literal::black_t &)` [inline]

Assignment with literals.

10.388.3.2 `template<unsigned n> template<unsigned m> graylevel< n > & mln::value::graylevel< n >::operator= (const graylevel< m > & rhs)` [inline]

Assignment with any [graylevel](#).

References `mln::value::graylevel< n >::value()`.

10.388.3.3 `template<unsigned n> graylevel< n > & mln::value::graylevel< n >::operator= (int val) [inline]`

Assignment with int.

10.388.3.4 `template<unsigned n> graylevel< n > & mln::value::graylevel< n >::operator= (const graylevel< n > & rhs) [inline]`

Assignment.

10.388.3.5 `template<unsigned n> float mln::value::graylevel< n >::to_float () const [inline]`

Conversion to float between 0 and 1.

Referenced by `mln::value::graylevel_f::graylevel_f()`, and `mln::value::graylevel_f::operator=()`.

10.388.3.6 `template<unsigned n> unsigned mln::value::graylevel< n >::value () const [inline]`

Access to std type.

Referenced by `mln::value::graylevel< n >::graylevel()`, and `mln::value::graylevel< n >::operator=()`.

10.389 mln::value::graylevel_f Struct Reference

General gray-level class on n bits.

```
#include <graylevel_f.hh>
```

Inherits mln::value::Floating< mln::value::graylevel_f >, and mln::value::internal::value_like_< mln::value::float01_f, float01_f::enc, mln::value::internal::gray_f, mln::value::graylevel_f >.

Public Member Functions

- template<unsigned n>
graylevel_f (const graylevel< n > &rhs)
Constructor from graylevel.
- graylevel_f (float val)
Constructor from float.
- graylevel_f (const graylevel_f &rhs)
Copy constructor.
- graylevel_f ()
Constructor without argument.
- template<unsigned n>
operator graylevel< n > () const
Conversion to graylevel<n>.
- template<unsigned n>
graylevel_f & operator= (const graylevel< n > &rhs)
Assignment with graylevel.
- graylevel_f & operator= (float val)
Assignment with float.
- graylevel_f & operator= (const graylevel_f &rhs)
Assignment.
- 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.

10.389.1 Detailed Description

General gray-level class on n bits.

10.389.2 Constructor & Destructor Documentation

10.389.2.1 `mln::value::graylevel_f::graylevel_f()` [inline]

Constructor without argument.

10.389.2.2 `mln::value::graylevel_f::graylevel_f(const graylevel_f & rhs)` [inline]

Copy constructor.

10.389.2.3 `mln::value::graylevel_f::graylevel_f(float val)` [inline]

Constructor from float.

10.389.2.4 `template<unsigned n> mln::value::graylevel_f::graylevel_f(const graylevel<n> & rhs)` [inline]

Constructor from [graylevel](#).

References `mln::value::graylevel<n>::to_float()`.

10.389.2.5 `mln::value::graylevel_f::graylevel_f(const mln::literal::black_t &)` [inline]

Ctors with literals.

10.389.3 Member Function Documentation

10.389.3.1 `template<unsigned n> mln::value::graylevel_f::operator graylevel<n>() const` [inline]

Conversion to `graylevel<n>`.

10.389.3.2 `graylevel_f & mln::value::graylevel_f::operator=(const mln::literal::black_t &)` [inline]

Assignment with literals.

10.389.3.3 `template<unsigned n> graylevel_f & mln::value::graylevel_f::operator=(const graylevel<n> & rhs)` [inline]

Assignment with [graylevel](#).

References `mln::value::graylevel<n>::to_float()`.

10.389.3.4 graylevel_f & mln::value::graylevel_f::operator= (float *val*) [inline]

Assignment with float.

10.389.3.5 graylevel_f & mln::value::graylevel_f::operator= (const graylevel_f & *rhs*) [inline]

Assignment.

10.389.3.6 float mln::value::graylevel_f::value () const [inline]

Access to std type.

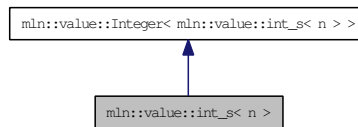
Referenced by mln::value::operator<<().

10.390 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](#) (int i)
Constructor from an integer.
- [int_s](#) ()
Constructor without argument.
- [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.

10.390.1 Detailed Description

```
template<unsigned n> struct mln::value::int_s< n >
```

Signed integer [value](#) class.

The parameter is n the number of encoding bits.

10.390.2 Constructor & Destructor Documentation

10.390.2.1 `template<unsigned n> mln::value::int_s< n >::int_s ()` [inline]

Constructor without argument.

10.390.2.2 `template<unsigned n> mln::value::int_s< n >::int_s (int i)` [inline]

Constructor from an integer.

10.390.2.3 `template<unsigned n> mln::value::int_s< n >::int_s (const mln::literal::zero_t &)`
[inline]

Constructors/assignments with literals.

10.390.3 Member Function Documentation

10.390.3.1 `template<unsigned n> mln::value::int_s< n >::operator int () const` [inline]

Conversion to an integer.

10.390.3.2 `template<unsigned n> int_s< n > & mln::value::int_s< n >::operator= (int i)`
[inline]

Assignment from an integer.

10.390.4 Member Data Documentation

10.390.4.1 `template<unsigned n> const int_s< n > mln::value::int_s< n >::one = 1` [inline,
static]

Unit [value](#).

10.390.4.2 `template<unsigned n> const int_s< n > mln::value::int_s< n >::zero = 0` [inline,
static]

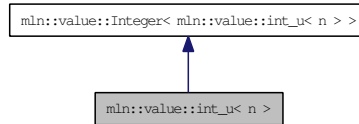
Zero [value](#).

10.391 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` (int i)
Constructor from an integer.
- `int_u` ()
Constructor without argument.
- `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.

10.391.1 Detailed Description

```
template<unsigned n> struct mln::value::int_u< n >
```

Unsigned integer `value` class.

The parameter is `n` the number of encoding bits.

10.391.2 Constructor & Destructor Documentation

10.391.2.1 `template<unsigned n> mln::value::int_u< n >::int_u ()` `[inline]`

Constructor without argument.

10.391.2.2 `template<unsigned n> mln::value::int_u< n >::int_u (int i) [inline]`

Constructor from an integer.

10.391.2.3 `template<unsigned n> mln::value::int_u< n >::int_u (const mln::literal::zero_t &) [inline]`

Constructors/assignments with literals.

10.391.3 Member Function Documentation

10.391.3.1 `template<unsigned n> int_u< n > mln::value::int_u< n >::next () const [inline]`

Give the next [value](#) (i.e., $i + 1$).

10.391.3.2 `template<unsigned n> mln::value::int_u< n >::operator unsigned () const [inline]`

Conversion to an unsigned integer.

10.391.3.3 `template<unsigned n> int mln::value::int_u< n >::operator- () const [inline]`

Unary operator minus.

10.391.3.4 `template<unsigned n> int_u< n > & mln::value::int_u< n >::operator= (int i) [inline]`

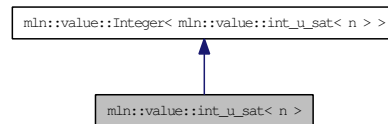
Assignment from an integer.

10.392 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](#) (int i)
Constructor from an integer.
- [int_u_sat](#) ()
Constructor without argument.
- [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.

10.392.1 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.

10.392.2 Constructor & Destructor Documentation

10.392.2.1 `template<unsigned n> mln::value::int_u_sat< n >::int_u_sat () [inline]`

Constructor without argument.

10.392.2.2 `template<unsigned n> mln::value::int_u_sat< n >::int_u_sat (int i) [inline]`

Constructor from an integer.

10.392.3 Member Function Documentation

10.392.3.1 `template<unsigned n> mln::value::int_u_sat< n >::operator int () const [inline]`

Conversion to an integer.

10.392.3.2 `template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator+= (int i) [inline]`

Self addition.

10.392.3.3 `template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator-= (int i) [inline]`

Self subtraction.

10.392.3.4 `template<unsigned n> int_u_sat< n > & mln::value::int_u_sat< n >::operator= (int i) [inline]`

Assignment from an integer.

10.392.4 Member Data Documentation

10.392.4.1 `template<unsigned n> const int_u_sat< n > mln::value::int_u_sat< n >::one = 1 [inline, static]`

Unit [value](#).

10.392.4.2 `template<unsigned n> const int_u_sat< n > mln::value::int_u_sat< n >::zero = 0 [inline, static]`

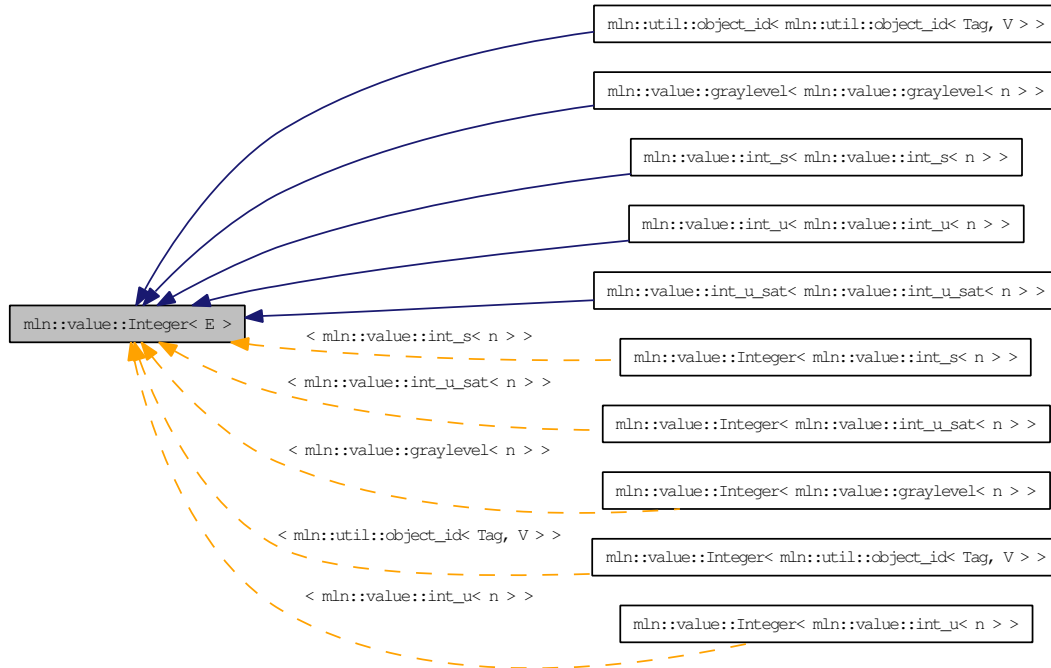
Zero [value](#).

10.393 mln::value::Integer< E > Struct Template Reference

Concept of integer.

```
#include <integer.hh>
```

Inheritance diagram for mln::value::Integer< E >:



10.393.1 Detailed Description

template<typename E> struct mln::value::Integer< E >

Concept of integer.

10.394 mln::value::Integer< void > Struct Template Reference

Category flag type.

```
#include <integer.hh>
```

10.394.1 Detailed Description

template<> struct mln::value::Integer< void >

Category flag type.

10.395 mln::value::label< n > Struct Template Reference

Label [value](#) class.

```
#include <label.hh>
```

Inherits [mln::value::Symbolic< mln::value::label< n > >](#), and [mln::value::internal::value_like_< unsigned, mln::value::internal::encoding_unsigned_< n >::ret, int, mln::value::label< n > >](#).

Public Types

- `typedef internal::encoding_unsigned_< n >::ret` [enc](#)

Encoding associated type.

Public Member Functions

- [label](#) (const [literal::zero_t](#) &v)
Constructor from [literal::zero](#).
- [label](#) (unsigned i)
Constructor from an (unsigned) integer.
- [label](#) ()
Constructor without argument.
- [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=](#) (const [literal::zero_t](#) &v)
Assignment from [literal::zero](#).
- [label< n > & operator=](#) (unsigned i)
Assignment from an (unsigned) integer.
- [label< n > prev](#) () const
Return the previous [value](#).

10.395.1 Detailed Description

`template<unsigned n> struct mln::value::label< n >`

Label [value](#) class.

The parameter `n` is the number of encoding bits.

10.395.2 Member Typedef Documentation

10.395.2.1 `template<unsigned n> typedef internal::encoding_unsigned_<n>::ret mln::value::label< n >::enc`

Encoding associated type.

10.395.3 Constructor & Destructor Documentation

10.395.3.1 `template<unsigned n> mln::value::label< n >::label () [inline]`

Constructor without argument.

10.395.3.2 `template<unsigned n> mln::value::label< n >::label (unsigned i) [inline]`

Constructor from an (unsigned) integer.

10.395.3.3 `template<unsigned n> mln::value::label< n >::label (const literal::zero_t & v) [inline]`

Constructor from [literal::zero](#).

10.395.4 Member Function Documentation

10.395.4.1 `template<unsigned n> label< n > mln::value::label< n >::next () const [inline]`

Return the next [value](#).

10.395.4.2 `template<unsigned n> mln::value::label< n >::operator unsigned () const [inline]`

Conversion to an unsigned integer.

10.395.4.3 `template<unsigned n> label< n > & mln::value::label< n >::operator++ () [inline]`

Self increment.

10.395.4.4 `template<unsigned n> label< n > & mln::value::label< n >::operator- ()`
[inline]

Self decrement.

10.395.4.5 `template<unsigned n> label< n > & mln::value::label< n >::operator= (const literal::zero_t & v)` [inline]

Assignment from [literal::zero](#).

10.395.4.6 `template<unsigned n> label< n > & mln::value::label< n >::operator= (unsigned i)`
[inline]

Assignment from an (unsigned) integer.

10.395.4.7 `template<unsigned n> label< n > mln::value::label< n >::prev () const` [inline]

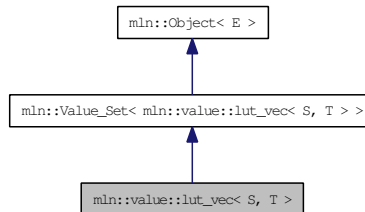
Return the previous [value](#).

10.396 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 V>
lut_vec (const S &vset, const Function_v2v< util::array< V > > &f)
Constructor from a value set and any util::array.
- 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 F>`
`lut_vec` (`const S &vset, const Function_v2v< F > &f`)
Constructors
Constructor from a [value set](#) and any [Function_v2v](#).

10.396.1 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

10.396.2 Member Typedef Documentation

10.396.2.1 `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.

10.396.2.2 `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.

10.396.2.3 `template<typename S, typename T> typedef T mln::value::lut_vec< S, T >::value`

[Value](#) associated type.

10.396.3 Constructor & Destructor Documentation

10.396.3.1 `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](#).

10.396.3.2 `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`.

10.396.3.3 `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](#).

References `mln::util::array< T >::size()`, and `mln::util::array< T >::std_vector()`.

10.396.4 Member Function Documentation

10.396.4.1 `template<typename S, typename T> bool mln::value::lut_vec< S, T >::has (const value & v) const [inline]`

Test if `v` belongs to this [set](#).

10.396.4.2 `template<typename S, typename T> unsigned mln::value::lut_vec< S, T >::index_of (const value & v) const [inline]`

Give the index of [value](#) `v` in this [set](#).

10.396.4.3 `template<typename S, typename T> unsigned mln::value::lut_vec< S, T >::nvalues () const [inline]`

Give the number of values.

Referenced by `mln::value::lut_vec< S, T >::operator[]()`.

10.396.4.4]

`template<typename S, typename T> T mln::value::lut_vec< S, T >::operator[] (unsigned i) const [inline]`

Give the `i`-th [value](#).

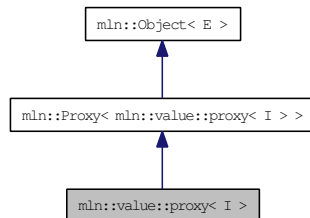
References `mln::value::lut_vec< S, T >::nvalues()`.

10.397 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

- template<typename J>
[proxy](#)< I > & [operator=](#) (const [proxy](#)< J > &rhs)
Assignment (write access); with other [proxy](#).
- [proxy](#)< I > & [operator=](#) (const [proxy](#)< I > &rhs)
Assignment (write access); replacement for default op.
- [proxy](#) (I &ima, const typename I::psite &p)
Constructor.
- [proxy](#) ()
Constructor.
- I::value [to_value](#) () const
Explicit read access.
- [~proxy](#) ()
Destructor.

10.397.1 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.

10.397.2 Member Typedef Documentation

10.397.2.1 `template<typename I> typedef void mln::value::proxy< I >::enc`

Encoding associated type.

10.397.2.2 `template<typename I> typedef I::value mln::value::proxy< I >::equiv`

Equivalent associated type.

10.397.3 Constructor & Destructor Documentation

10.397.3.1 `template<typename I> mln::value::proxy< I >::proxy ()` [inline]

Constructor.

10.397.3.2 `template<typename I> mln::value::proxy< I >::proxy (I & ima, const typename I::psite & p)` [inline]

Constructor.

10.397.3.3 `template<typename I> mln::value::proxy< I >::~~proxy ()` [inline]

Destructor.

10.397.4 Member Function Documentation

10.397.4.1 `template<typename I> template<typename J> proxy< I > & mln::value::proxy< I >::operator= (const proxy< J > & rhs)` [inline]

Assignment (write access); with other [proxy](#).

References `mln::value::proxy< I >::to_value()`.

10.397.4.2 `template<typename I> proxy< I > & mln::value::proxy< I >::operator= (const proxy< I > & rhs)` [inline]

Assignment (write access); replacement for default op.

References `mln::value::proxy< I >::to_value()`.

10.397.4.3 `template<typename I> I::value mln::value::proxy< I >::to_value () const`
[inline]

Explicit read access.

Referenced by `mln::value::proxy< I >::operator=()`.

10.398 `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< mln::value::rgb< n > >`, and `mln::value::internal::value_like_< mln::algebra::vec< 3, mln::value::int_u< n > >, mln::algebra::vec< 3, mln::value::int_u< n > >, mln::algebra::vec< 3, int >, mln::value::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.

10.398.1 Detailed Description

```
template<unsigned n> struct mln::value::rgb< n >
```

Color class for red-green-blue where every component is n-bit encoded.

10.398.2 Constructor & Destructor Documentation

10.398.2.1 `template<unsigned n> mln::value::rgb< n >::rgb () [inline]`

Constructor without argument.

10.398.2.2 `template<unsigned n> mln::value::rgb<n>::rgb (int r, int g, int b) [inline]`

Constructor from component values.

10.398.2.3 `template<unsigned n> mln::value::rgb<n>::rgb (const algebra::vec<3, int> & rhs) [inline]`

Constructor from a algebra::vec.

10.398.2.4 `template<unsigned n> mln::value::rgb<n>::rgb (const mln::literal::white_t &) [inline]`

Constructors with literals.

10.398.3 Member Function Documentation

10.398.3.1 `template<unsigned n> rgb<n> & mln::value::rgb<n>::operator= (const rgb<n> & rhs) [inline]`

Assignment.

10.398.3.2 `template<unsigned n> int_u<n> mln::value::rgb<n>::red () const [inline]`

Acces to red/green/blue component.

Referenced by mln::io::magick::do_it().

10.398.4 Member Data Documentation

10.398.4.1 `template<unsigned n> const rgb<n> mln::value::rgb<n>::zero [inline, static]`

Zero [value](#).

10.399 mln::value::set< T > Struct Template Reference

Class that defines the [set](#) of values of type T.

```
#include <set.hh>
```

Inherits mln::value::internal::set_selector_< T, mln::value::set< T >, mln::metal::equal< mln_trait_value_quant(T), mln::trait::value::quant::low >::value >.

Static Public Member Functions

- static const [set](#)< T > & [the](#) ()

Return a singleton.

10.399.1 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.

10.399.2 Member Function Documentation

10.399.2.1 `template<typename T> const set< T > & mln::value::set< T >::the ()` [inline, static]

Return a singleton.

10.400 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 `mln::value::internal::Integer< mln::value::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 (int i)`
Constructor from an integer.
- `sign ()`
Constructor without argument.
- `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.

10.400.1 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.

10.400.2 Member Typedef Documentation

10.400.2.1 typedef int mln::value::sign::enc

FIXME Are these typedefs correct?

Define the encoding type

10.400.2.2 typedef int mln::value::sign::equiv

Define the equivalent type.

10.400.3 Constructor & Destructor Documentation

10.400.3.1 mln::value::sign::sign () [inline]

Constructor without argument.

10.400.3.2 mln::value::sign::sign (int *i*) [inline]

Constructor from an integer.

10.400.3.3 mln::value::sign::sign (const mln::literal::zero_t &) [inline]

Constructors/assignments with literals.

10.400.4 Member Function Documentation

10.400.4.1 mln::value::sign::operator int () const [inline]

Conversion to an integer.

10.400.4.2 sign & mln::value::sign::operator= (int *i*) [inline]

Assignment from an integer.

10.400.5 Member Data Documentation

10.400.5.1 const sign mln::value::sign::one = 1 [static]

Unit [value](#).

10.400.5.2 const sign mln::value::sign::zero = 0 [static]

Zero [value](#).

10.401 mln::value::stack_image< n, I > Struct Template Reference

Stack image class.

```
#include <stack.hh>
```

Inherits mln::internal::image_value_morpher< I, mln::algebra::vec< n, I::value >, mln::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.

10.401.1 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. Access a `value` will compute a vector which contains `n` coordinates : `[stack[0](p), stack[1](p), ... , stack[n](p)]`

10.401.2 Member Typedef Documentation

10.401.2.1 `template<unsigned n, typename I> typedef I ::domain_t mln::value::stack_image< n, I >::domain_t`

`Site_Set` associated type.

10.401.2.2 `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.

10.401.2.3 `template<unsigned n, typename I> typedef I ::psite mln::value::stack_image< n, I >::psite`

`Point_Site` associated type.

10.401.2.4 `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).

10.401.2.5 `template<unsigned n, typename I> typedef stack_image< n, tag::image_<I> > mln::value::stack_image< n, I >::skeleton`

Skeleton.

10.401.2.6 `template<unsigned n, typename I> typedef algebra::vec<n, typename I ::value> mln::value::stack_image< n, I >::value`

`Value` associated type.

10.401.3 Constructor & Destructor Documentation

10.401.3.1 `template<unsigned n, typename I> mln::value::stack_image< n, I >::stack_image (const algebra::vec< n, I > & imas) [inline]`

Constructors.

10.401.4 Member Function Documentation

10.401.4.1 `template<unsigned n, typename I> bool mln::value::stack_image< n, I >::is_valid () const [inline]`

Test if this image has been initialized.

10.401.4.2 `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*.

10.401.4.3 `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*.

10.402 mln::value::super_value< sign > Struct Template Reference

Specializations:

```
#include <super_value.hh>
```

10.402.1 Detailed Description

template<> struct mln::value::super_value< sign >

Specializations:

Sign type is a subset of the short [value](#) type.

10.403 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
}

10.403.1 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.

10.403.2 Constructor & Destructor Documentation

10.403.2.1 `template<typename T, typename V> mln::value::value_array< T, V >::value_array ()`
[inline]

Constructors.

```
{
```

10.403.3 Member Function Documentation

10.403.3.1 `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. {

10.403.3.2 `]`

`template<typename T, typename V> const V & mln::value::value_array< T, V >::operator[] (unsigned i) const` [inline]

```
}
```

Access elements through array indexes. {

```
10.403.3.3 template<typename T, typename V> const mln::value::set< T > &  
mln::value::value_array< T, V >::vset () const [inline]
```

```
}
```

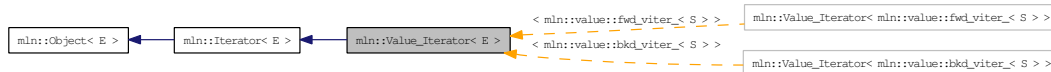
Reference to the [set](#) of T.

10.404 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`.

10.404.1 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.

10.404.2 Member Function Documentation

10.404.2.1 `template<typename E> void mln::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.

10.404.3 Friends And Related Function Documentation

10.404.3.1 `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:

- ↔ *ostr* An output stream.
- ← *v* An iterator on [value set](#).

Precondition:

v is a valid.

Returns:

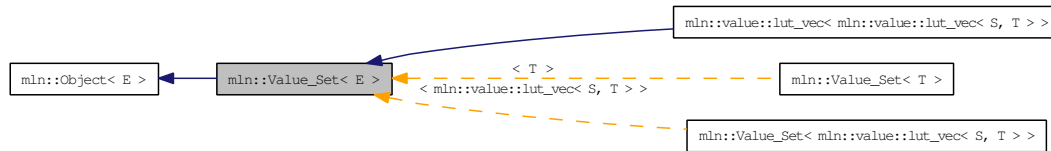
The modified output stream *ostr*.

10.405 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 >:



10.405.1 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.

10.406 mln::Vertex< E > Struct Template Reference

[Vertex](#) category flag type.

```
#include <vertex.hh>
```

10.406.1 Detailed Description

```
template<typename E> struct mln::Vertex< E >
```

[Vertex](#) category flag type.

10.407 mln::vertex_image< P, V, G > Class Template Reference

Image based on [graph](#) vertices.

```
#include <vertex_image.hh>
```

Inherits [mln::pw::internal::image_base< mln::fun::i2v::array< V >, mln::p_vertices< G, mln::internal::vfsite_selector< P, G >::mln::fun::i2v::array >, mln::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::vfsite_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

- rvalue [operator\(\)](#) (unsigned [v_id](#)) const
Value accessors/operators overloads.
- [vertex_image](#) ()
Constructors.

10.407.1 Detailed Description

```
template<typename P, typename V, typename G = util::graph> class mln::vertex_image< P, V, G >
```

Image based on [graph](#) vertices.

10.407.2 Member Typedef Documentation

10.407.2.1 `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](#).

10.407.2.2 `template<typename P, typename V, typename G = util::graph> typedef vertex_nbh_t mln::vertex_image< P, V, G >::nbh_t`

[Neighborhood](#) type.

10.407.2.3 `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.

10.407.2.4 `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.

10.407.2.5 `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.

10.407.2.6 `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.

10.407.2.7 `template<typename P, typename V, typename G = util::graph> typedef vertex_win_t mln::vertex_image< P, V, G >::win_t`

[Window](#) type.

10.407.3 Constructor & Destructor Documentation

10.407.3.1 `template<typename P, typename V, typename G> mln::vertex_image< P, V, G >::vertex_image () [inline]`

Constructors.

10.407.4 Member Function Documentation

10.407.4.1 `template<typename P, typename V, typename G> vertex_image< P, V, G >::rvalue
mln::vertex_image< P, V, G >::operator() (unsigned v_id) const [inline]`

[Value](#) accessors/operators overloads.

10.408 mln::violent_cast_image< T, I > Struct Template Reference

Violently cast image values to a given type.

```
#include <violent_cast_image.hh>
```

Inherits mln::internal::image_value_morpher< I, T, mln::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)
Mutable access is only OK for reading (not writing).
- T [operator\(\)](#) (const typename I::psite &p) const
Read-only access of [pixel value](#) at [point](#) site p.
- [violent_cast_image](#) (const [Image](#)< I > &ima)
Constructor.

10.408.1 Detailed Description

```
template<typename T, typename I> struct mln::violent_cast_image< T, I >
```

Violently cast image values to a given type.

10.408.2 Member Typedef Documentation

10.408.2.1 template<typename T, typename I> typedef T mln::violent_cast_image< T, I >::lvalue

Return type of read-write access.

10.408.2.2 `template<typename T, typename I> typedef T mln::violent_cast_image< T, I >::rvalue`

Return type of read-only access.

10.408.2.3 `template<typename T, typename I> typedef violent_cast_image< tag::value_<T>, tag::image_<I> > mln::violent_cast_image< T, I >::skeleton`

Skeleton.

10.408.2.4 `template<typename T, typename I> typedef T mln::violent_cast_image< T, I >::value`

[Value](#) associated type.

10.408.3 Constructor & Destructor Documentation

10.408.3.1 `template<typename T, typename I> mln::violent_cast_image< T, I >::violent_cast_image (const Image< I > & ima) [inline]`

Constructor.

10.408.4 Member Function Documentation

10.408.4.1 `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).

10.408.4.2 `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*.

10.409 mln::w_window< D, W > Struct Template Reference

Generic `w_window` class.

```
#include <w_window.hh>
```

Inherits `mln::internal::weighted_window_base< mln::window< D >, mln::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 W>`
`W operator-` (const `Weighted_Window< W >` &rhs)
Compute the symmetrical weighted [window](#) of rhs.
- `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.

10.409.1 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.

10.409.2 Member Typedef Documentation

10.409.2.1 `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`.

10.409.2.2 `template<typename D, typename W> typedef D mln::w_window< D, W >::dpsite`

Dpsite associated type.

10.409.2.3 `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`.

10.409.2.4 `template<typename D, typename W> typedef W mln::w_window< D, W >::weight`

Weight associated type.

10.409.3 Constructor & Destructor Documentation

10.409.3.1 `template<typename D, typename W> mln::w_window< D, W >::w_window ()`
`[inline]`

Constructor without argument.

10.409.4 Member Function Documentation

10.409.4.1 `template<typename D, typename W> void mln::w_window< D, W >::clear ()`
`[inline]`

Clear this [window](#).

References `mln::w_window< D, W >::clear()`.

Referenced by `mln::w_window< D, W >::clear()`.

10.409.4.2 `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`.

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()`.

10.409.4.3 `template<typename D, typename W> bool mln::w_window< D, W >::is_symmetric ()`
`const [inline]`

Test if the [window](#) is symmetric.

References `mln::w_window< D, W >::sym()`.

10.409.4.4 `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.

10.409.4.5 `template<typename D, typename W> void mln::w_window< D, W >::sym ()`
`[inline]`

Apply a central symmetry to the [window](#).

References `mln::w_window< D, W >::insert()`.

Referenced by `mln::w_window< D, W >::is_symmetric()`.

10.409.4.6 `template<typename D, typename W> W mln::w_window< D, W >::w (unsigned i)`
`const [inline]`

Give the `i`-th weight.

10.409.4.7 `template<typename D, typename W> const std::vector< W > & mln::w_window< D, W >::weights () const` [inline]

Give access to the vector of weights.

Referenced by `mln::w_window< D, W >::operator==(())`.

10.409.4.8 `template<typename D, typename W> const mln::window< D > & mln::w_window< D, W >::win () const` [inline]

Give the corresponding [window](#).

Referenced by `mln::w_window< D, W >::operator==(())`.

10.409.5 Friends And Related Function Documentation

10.409.5.1 `template<typename W> W operator- (const Weighted_Window< W > & rhs)`
[related, inherited]

Compute the symmetrical weighted [window](#) of `rhs`.

10.409.5.2 `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`.

10.409.5.3 `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`.

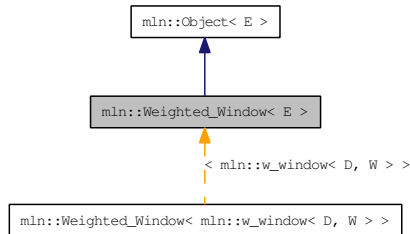
References `mln::w_window< D, W >::weights()`, and `mln::w_window< D, W >::win()`.

10.410 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.

10.410.1 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.

10.410.2 Friends And Related Function Documentation

10.410.2.1 `template<typename W> W operator- (const Weighted_Window< W > & rhs)`
 [related]

Compute the symmetrical weighted window of rhs.

10.411 mln::win::backdiag2d Struct Reference

Diagonal [line window](#) defined on the 2D square [grid](#).

```
#include <backdiag2d.hh>
```

Inherits mln::internal::classical_window_base< mln::dpoint, mln::win::backdiag2d >.

Public Member Functions

- [backdiag2d](#) (unsigned length)
Constructor.
- unsigned [length](#) () const
Give the diagonal length, that is, its width.

10.411.1 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
*       o
*
```

is defined with length = 5.

10.411.2 Constructor & Destructor Documentation

10.411.2.1 mln::win::backdiag2d::backdiag2d (unsigned length) [inline]

Constructor.

Parameters:

← *length* Length, thus width, of the diagonal [line](#).

Precondition:

length is odd.

10.411.3 Member Function Documentation

10.411.3.1 unsigned mln::win::backdiag2d::length () const [inline]

Give the diagonal length, that is, its width.

10.412 mln::win::ball< G, C > Struct Template Reference

Generic [ball window](#) defined on a given [grid](#).

```
#include <ball.hh>
```

Inherits mln::internal::classical_window_base< mln::dpoint< G, C >, mln::win::ball< G, C > >.

Public Member Functions

- [ball](#) (unsigned diameter)
Constructor.
- unsigned [diameter](#) () const
Give the [ball](#) diameter.

10.412.1 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.

10.412.2 Constructor & Destructor Documentation

10.412.2.1 `template<typename G, typename C> mln::win::ball< G, C >::ball (unsigned diameter) [inline]`

Constructor.

Parameters:

← *diameter* Diameter of the [ball](#).

Precondition:

`diameter` is odd.

References mln::literal::origin.

10.412.3 Member Function Documentation

10.412.3.1 `template<typename G, typename C> unsigned mln::win::ball< G, C >::diameter () const [inline]`

Give the [ball](#) diameter.

10.413 mln::win::cube3d Struct Reference

Cube [window](#) defined on the 3D [grid](#).

```
#include <cube3d.hh>
```

Inherits `mln::internal::classical_window_base< mln::dpoint, mln::win::cube3d >`.

Public Member Functions

- [cube3d](#) (unsigned length)

Constructor.

- unsigned [length](#) () const

Give the cube length, that is, its height.

10.413.1 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.

10.413.2 Constructor & Destructor Documentation

10.413.2.1 mln::win::cube3d::cube3d (unsigned length) [inline]

Constructor.

Parameters:

← *length* Length, thus height, of the [cube3d](#).

Precondition:

`length` is odd.

10.413.3 Member Function Documentation

10.413.3.1 unsigned mln::win::cube3d::length () const [inline]

Give the cube length, that is, its height.

10.414 mln::win::cuboid3d Struct Reference

Cuboid defined on the 3-D square [grid](#).

```
#include <cuboid3d.hh>
```

Inherits mln::internal::classical_window_base< mln::dpoint, mln::win::cuboid3d >.

Public Member Functions

- [cuboid3d](#) (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.

10.414.1 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>

10.414.2 Constructor & Destructor Documentation

10.414.2.1 mln::win::cuboid3d::cuboid3d (unsigned *depth*, unsigned *height*, unsigned *width*) [inline]

Constructor.

Parameters:

- ← *depth* The depth of the [cuboid3d](#).
- ← *height* The height of the [cuboid3d](#).
- ← *width* The width of the [cuboid3d](#).

Precondition:

Argument *depth*, *height* and *width* must be odd.

10.414.3 Member Function Documentation

10.414.3.1 unsigned mln::win::cuboid3d::depth () const [inline]

Accessors.

Return the depth of the cuboid.

10.414.3.2 unsigned mln::win::cuboid3d::height () const [inline]

Return the height of the cuboid.

10.414.3.3 unsigned mln::win::cuboid3d::volume () const [inline]

Return the volume of the cuboid.

10.414.3.4 unsigned mln::win::cuboid3d::width () const [inline]

Return the width of the cuboid.

10.415 mln::win::diag2d Struct Reference

Diagonal [line window](#) defined on the 2D square [grid](#).

```
#include <diag2d.hh>
```

Inherits mln::internal::classical_window_base< mln::dpoint, mln::win::diag2d >.

Public Member Functions

- [diag2d](#) (unsigned length)
Constructor.
- unsigned [length](#) () const
Give the diagonal length, that is, its width.

10.415.1 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.

10.415.2 Constructor & Destructor Documentation

10.415.2.1 mln::win::diag2d::diag2d (unsigned length) [inline]

Constructor.

Parameters:

← *length* Length, thus width, of the diagonal [line](#).

Precondition:

length is odd.

10.415.3 Member Function Documentation

10.415.3.1 unsigned mln::win::diag2d::length () const [inline]

Give the diagonal length, that is, its width.

10.416 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 mln::internal::classical_window_base< mln::dpoint< M, C >, mln::win::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.

10.416.1 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 example of his use.

10.416.2 Member Enumeration Documentation

10.416.2.1 template<typename M, unsigned i, typename C> anonymous enum

Direction.

10.416.3 Constructor & Destructor Documentation

10.416.3.1 `template<typename M, unsigned i, typename C> mln::win::line< M, i, C >::line (unsigned length) [inline]`

Constructor.

Parameters:

← *length* Length of the [line](#).

Precondition:

`length` is odd.

References `mln::dpoint< G, C >::set_all()`.

10.416.4 Member Function Documentation

10.416.4.1 `template<typename M, unsigned i, typename C> unsigned mln::win::line< M, i, C >::length () const [inline]`

Give the [line](#) length.

10.416.4.2 `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.

10.417 mln::win::multiple< W, F > Class Template Reference

Multiple [window](#).

```
#include <multiple.hh>
```

Inherits mln::internal::window_base< W::dpsite, mln::win::multiple< W, F > >.

10.417.1 Detailed Description

```
template<typename W, typename F> class mln::win::multiple< W, F >
```

Multiple [window](#).

10.418 mln::win::multiple_size< n, W, F > Class Template Reference

Definition of a multiple-size [window](#).

```
#include <multiple_size.hh>
```

Inherits mln::internal::window_base< W::dpsite, mln::win::multiple_size< n, W, F > >.

10.418.1 Detailed Description

```
template<unsigned n, typename W, typename F> class mln::win::multiple_size< n, W, F >
```

Definition of a multiple-size [window](#).

10.419 mln::win::octagon2d Struct Reference

Octagon [window](#) defined on the 2D square [grid](#).

```
#include <octagon2d.hh>
```

Inherits mln::internal::classical_window_base< mln::dpoint, mln::win::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.

10.419.1 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 $L = 7$ ($l = 1$).

10.419.2 Constructor & Destructor Documentation

10.419.2.1 mln::win::octagon2d::octagon2d (unsigned *length*) [inline]

Constructor.

Parameters:

← *length* Length, of the octagon.

Precondition:

length is such as $length = 6 * x + 1$ where $x \geq 0$.

10.419.3 Member Function Documentation

10.419.3.1 `unsigned mln::win::octagon2d::area () const` [inline]

Give the area.

10.419.3.2 `unsigned mln::win::octagon2d::length () const` [inline]

Give the octagon length, that is, its width.

10.420 mln::win::rectangle2d Struct Reference

Rectangular [window](#) defined on the 2D square [grid](#).

```
#include <rectangle2d.hh>
```

Inherits mln::internal::classical_window_base< mln::dpoint, mln::win::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.

10.420.1 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.

10.420.2 Constructor & Destructor Documentation

10.420.2.1 mln::win::rectangle2d::rectangle2d (unsigned *height*, unsigned *width*) [inline]

Constructor.

Parameters:

- ← *height* Height of the [rectangle2d](#).
- ← *width* Width of the [rectangle2d](#).

Precondition:

Height and width are odd.

10.420.3 Member Function Documentation

10.420.3.1 `unsigned mln::win::rectangle2d::area () const` [inline]

Give the rectangle area.

10.420.3.2 `unsigned mln::win::rectangle2d::height () const` [inline]

Give the rectangle height.

10.420.3.3 `const std::vector< dpoint2d > & mln::win::rectangle2d::std_vector () const` [inline]

Give the std vector of delta-points.

10.420.3.4 `unsigned mln::win::rectangle2d::width () const` [inline]

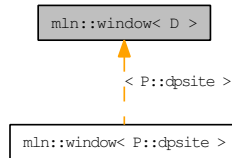
Give the rectangle width.

10.422 mln::window< D > Class Template Reference

Generic [window](#) class.

```
#include <window.hh>
```

Inheritance diagram for mln::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.
- template<typename W>
[window](#)< D > & [insert](#) (const [Window](#)< W > &win)
Insert another window win.
- [window](#)< D > & [insert](#) (const D &dp)
Insert a delta-point dp.

- 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.

10.422.1 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.

10.422.2 Member Typedef Documentation

10.422.2.1 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.

10.422.2.2 `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.

10.422.2.3 `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.

10.422.2.4 `template<typename D> typedef window<D> mln::window< D >::regular`

Regular [window](#) associated type.

10.422.3 Constructor & Destructor Documentation

10.422.3.1 `template<typename D> mln::window< D >::window () [inline]`

Constructor without argument.

The constructed [window](#) is empty.

10.422.4 Member Function Documentation

10.422.4.1 `template<typename D> void mln::window< D >::clear () [inline]`

Clear the [window](#).

10.422.4.2 `template<typename D> unsigned mln::window< D >::delta () const [inline]`

Give the maximum coordinate gap between the [window](#) center and a [window point](#).

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

10.422.4.3 `template<typename D> const D & mln::window< D >::dp (unsigned i) const [inline]`

Give the *i*-th delta-point.

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

Referenced by `mln::window< D >::delta()`, and `mln::window< D >::insert()`.

10.422.4.4 `template<typename D> bool mln::window< D >::has (const D & dp) const [inline]`

Test if `dp` is in this [window](#) definition.

Referenced by `mln::window< D >::is_centered()`.

10.422.4.5 `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.

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

10.422.4.6 `template<typename D> template<typename W> window< D > & mln::window< D >::insert (const Window< W > & win)` [inline]

Insert another [window](#) `win`.

10.422.4.7 `template<typename D> window< D > & mln::window< D >::insert (const D & dp)` [inline]

Insert a delta-point `dp`.

Referenced by `mln::c18()`, `mln::c26()`, `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()`.

10.422.4.8 `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](#).

References `mln::window< D >::has()`, and `mln::literal::zero`.

10.422.4.9 `template<typename D> bool mln::window< D >::is_empty () const` [inline]

Test if the [window](#) is empty (null size; no delta-point).

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

Referenced by `mln::window< D >::is_empty()`.

10.422.4.10 `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](#).

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

10.422.4.11 `template<typename D> void mln::window< D >::print (std::ostream & ostr) const` [inline]

Print the [window](#) definition into `ostr`.

10.422.4.12 `template<typename D> unsigned mln::window< D >::size () const` [inline]

Give the [window](#) size, i.e., the number of delta-sites.

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()`.

10.422.4.13 `template<typename D> const std::vector< D > & mln::window< D >::std_vector () const` [inline]

Give the std vector of delta-points.

10.422.4.14 `template<typename D> void mln::window< D >::sym ()` [inline]

Apply a central symmetry to the target [window](#).

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

Referenced by `mln::window< D >::is_symmetric()`.

10.422.5 Friends And Related Function Documentation**10.422.5.1** `template<typename D> bool operator==(const window< D > & lhs, const window< D > & rhs)` [related]

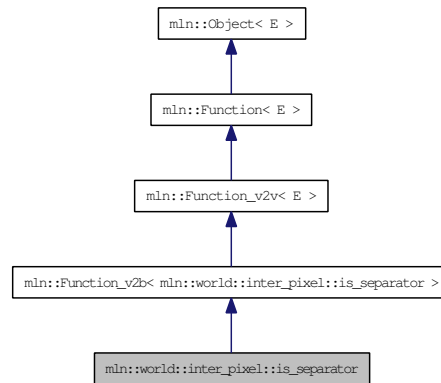
Equality comparison between windows `lhs` and `rhs`.

10.423 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:



10.423.1 Detailed Description

Functor returning whether a site is a separator in an inter-pixel image.

10.424 `trait::graph< I >` Struct Template Reference

Graph traits.

```
#include <morpho.hh>
```

10.424.1 Detailed Description

```
template<typename I> struct trait::graph< I >
```

Graph traits.

10.425 `trait::graph< mln::complex_image< 1, G, V > >` Struct Template Reference

Graph traits for 1-complexes images.

```
#include <morpho.hh>
```

10.425.1 Detailed Description

```
template<typename G, typename V> struct trait::graph< mln::complex_image< 1, G, V > >
```

Graph traits for 1-complexes images.

10.426 `trait::graph< mln::image2d< T > >` Struct Template Reference

Graph traits for [mln::image2d](#).

```
#include <morpho.hh>
```

10.426.1 Detailed Description

```
template<typename T> struct trait::graph< mln::image2d< T > >
```

Graph traits for [mln::image2d](#).

Index

- ~decorated_image
 - mln::decorated_image, 688
- ~proxy
 - mln::value::proxy, 1325
- ~soft_heap
 - mln::util::soft_heap, 1282
- ~tracked_ptr
 - mln::util::tracked_ptr, 1286
- _1
 - mln::algebra::h_mat, 630
- 1D neighborhoods, 112
- 1D windows, 126
- 2D neighborhoods, 113
- 2D windows, 127
- 3D neighborhoods, 115
- 3D windows, 130
- a_point_of
 - mln, 165
- abs
 - mln::data, 232
 - mln::math, 408
- abs_inplace
 - mln::data, 232
- Accumulators, 106
- add
 - mln::topo::n_faces_set, 1212
- add_child
 - mln::util::tree_node, 1290
- add_edge
 - mln::util::graph, 1251
- add_face
 - mln::topo::complex, 1191
- add_location
 - mln::geom::complex_geometry, 833
- add_tree_down
 - mln::util::tree, 1288
- add_tree_up
 - mln::util::tree, 1288
- add_vertex
 - mln::util::graph, 1251
- add_vertices
 - mln::util::graph, 1251
- addr
 - mln::topo::complex, 1191
- adj_higher_dim_connected_n_face_bkd_iter
 - mln::topo::adj_higher_dim_connected_n_face_bkd_iter, 1159
- adj_higher_dim_connected_n_face_fwd_iter
 - mln::topo::adj_higher_dim_connected_n_face_fwd_iter, 1161
- adj_higher_face_bkd_iter
 - mln::topo::adj_higher_face_bkd_iter, 1163
- adj_higher_face_fwd_iter
 - mln::topo::adj_higher_face_fwd_iter, 1164
- adj_lower_dim_connected_n_face_bkd_iter
 - mln::topo::adj_lower_dim_connected_n_face_bkd_iter, 1165
- adj_lower_dim_connected_n_face_fwd_iter
 - mln::topo::adj_lower_dim_connected_n_face_fwd_iter, 1167
- adj_lower_face_bkd_iter
 - mln::topo::adj_lower_face_bkd_iter, 1169
- adj_lower_face_fwd_iter
 - mln::topo::adj_lower_face_fwd_iter, 1170
- adj_lower_higher_face_bkd_iter
 - mln::topo::adj_lower_higher_face_bkd_iter, 1171
- adj_lower_higher_face_fwd_iter
 - mln::topo::adj_lower_higher_face_fwd_iter, 1172
- adj_m_face_bkd_iter
 - mln::topo::adj_m_face_bkd_iter, 1173
- adj_m_face_fwd_iter
 - mln::topo::adj_m_face_fwd_iter, 1175
- adjacency_matrix
 - mln::util::adjacency_matrix, 1225
- adjust
 - mln::border, 211
 - mln::extension, 271, 272
- adjust_duplicate
 - mln::extension, 272
- adjust_fill
 - mln::extension, 272
- algebraic_face
 - mln::topo::algebraic_face, 1178, 1179
- algebraic_n_face
 - mln::topo::algebraic_n_face, 1183
- and_inplace
 - mln::logical, 379

- and_not
 - mln::logical, 379
- and_not_inplace
 - mln::logical, 380
- apex
 - mln::util::branch, 1232
- append
 - mln::p_array, 974
 - mln::util::array, 1229
- apply
 - mln::data, 232
- apply_p2p
 - mln, 165
- area
 - mln::accu::site_set::rectangularity, 592
 - mln::morpho::attribute::sharpness, 958
 - mln::morpho::attribute::volume, 962
 - mln::win::octagon2d, 1364
 - mln::win::rectangle2d, 1366
- argument
 - mln::accu::shape::height, 586
 - mln::accu::shape::volume, 589
 - mln::doc::Accumulator, 692
- array
 - mln::util::array, 1228
- at
 - mln::opt, 451, 452
- attachment
 - mln::make, 389
- backdiag2d
 - mln::win::backdiag2d, 1352
- background
 - mln::labeling, 351
- ball
 - mln::win::ball, 1353
- base_level
 - mln::morpho::attribute::height, 955
- Basic types, 100, 119
- bbox
 - mln::accu::site_set::rectangularity, 592
 - mln::Box, 649
 - mln::box, 643
 - mln::doc::Box, 695
 - mln::doc::Fastest_Image, 703
 - mln::doc::Image, 712
 - mln::geom, 296, 297
 - mln::image1d, 877
 - mln::image2d, 882
 - mln::image3d, 890
 - mln::labeled_image, 902
 - mln::labeled_image_base, 905
 - mln::p_line2d, 1031
 - mln::p_run, 1069
- bbox_t
 - mln::labeled_image, 901
 - mln::labeled_image_base, 905
- bboxes
 - mln::labeled_image, 902
 - mln::labeled_image_base, 905
- before
 - mln, 177
- begin
 - mln::p_line2d, 1031
- bin_1complex_image2d
 - mln, 161
- bin_2complex_image3df
 - mln, 161
- binarization
 - mln::binarization, 210
- bkd_citer
 - mln::topo::complex, 1191
- bkd_eiter
 - mln::util::array, 1228
 - mln::util::set, 1275
- bkd_niter
 - mln::doc::Neighborhood, 717
 - mln::graph_elt_mixed_neighborhood, 841
 - mln::graph_elt_neighborhood, 847
 - mln::graph_elt_neighborhood_if, 849
 - mln::mixed_neighb, 949
 - mln::neighb, 965
- bkd_piter
 - mln::box, 642
 - mln::doc::Box, 695
 - mln::doc::Fastest_Image, 701
 - mln::doc::Image, 711
 - mln::doc::Site_Set, 729
 - mln::hexa, 869
 - mln::image2d_h, 886
 - mln::p_array, 973
 - mln::p_centered, 980
 - mln::p_complex, 985
 - mln::p_edges, 991
 - mln::p_faces, 999
 - mln::p_if, 1007
 - mln::p_image, 1012
 - mln::p_key, 1023
 - mln::p_line2d, 1030
 - mln::p_mutable_array_of, 1036
 - mln::p_priority, 1046
 - mln::p_queue, 1054
 - mln::p_queue_fast, 1061
 - mln::p_run, 1068
 - mln::p_set, 1075
 - mln::p_set_of, 1082
 - mln::p_transformed, 1087
 - mln::p_vaccess, 1094

- mln::p_vertices, 1100
- bkd_pixter1d
 - mln::bkd_pixter1d, 633
- bkd_pixter2d
 - mln::bkd_pixter2d, 635
- bkd_pixter3d
 - mln::bkd_pixter3d, 637
- bkd_qiter
 - mln::doc::Weighted_Window, 735
 - mln::doc::Window, 737
 - mln::graph_elt_mixed_window, 844
 - mln::graph_elt_window, 852
 - mln::graph_elt_window_if, 856
 - mln::w_window, 1348
 - mln::window, 1369
- bkd_viter
 - mln::doc::Value_Set, 733
 - mln::value::lut_vec, 1322
- black
 - mln::literal, 376
- blobs
 - mln::canvas::labeling, 221
 - mln::labeling, 352
- blobs_and_compute
 - mln::labeling, 352
- blue
 - mln::literal, 376
- border
 - mln::doc::Fastest_Image, 703
 - mln::image1d, 877
 - mln::image2d, 882
 - mln::image3d, 890
- box
 - mln::box, 642, 643
 - mln::draw, 267
- box1d
 - mln, 161
 - mln::make, 389
- box2d
 - mln, 161
 - mln::make, 390
- box2d_h
 - mln, 162
 - mln::make, 390, 391
- box3d
 - mln, 162
 - mln::make, 391, 392
- box_runend_piter
 - mln::box_runend_piter, 653
- box_runstart_piter
 - mln::box_runstart_piter, 655
- branch
 - mln::util::branch, 1232
- brown
 - mln::literal, 376
- buffer
 - mln::doc::Fastest_Image, 703
 - mln::image1d, 877
 - mln::image2d, 882
 - mln::image3d, 890
- c18
 - modneighb3d, 115
- c2
 - modneighb1d, 112
- c26
 - modneighb3d, 116
- c2_col
 - modneighb2d, 113
- c2_row
 - modneighb2d, 113
- c4
 - modneighb2d, 114
- c4_3d
 - modneighb3d, 116
- c6
 - modneighb3d, 117
- c8
 - modneighb2d, 114
- c8_3d
 - modneighb3d, 117
- can_stop
 - mln::accu::logic::land_basic, 523
 - mln::accu::logic::lor_basic, 527
- Canvas, 108
- card
 - mln::set, 459
- cast
 - mln::value, 498
- Category
 - mln::util::vertex, 1294
- category
 - mln::util::edge, 1242
- cell
 - mln::make, 392
- center
 - mln::box, 643
 - mln::p_centered, 980
- center_only_iter
 - mln::topo::center_only_iter, 1186
- center_t
 - mln::graph_elt_mixed_window, 844
 - mln::graph_elt_window, 852
 - mln::graph_window_piter, 865
- center_val
 - mln::dpoints_bkd_pixter, 745
 - mln::dpoints_fwd_pixter, 748
- centered_bkd_iter_adapter

- mln::topo::centered_bkd_iter_adapter, 1188
- centered_fwd_iter_adapter
 - mln::topo::centered_fwd_iter_adapter, 1189
- chamfer
 - mln::geom, 297
- change
 - mln::p_array, 974
- change_both
 - mln::util::couple, 1238
 - mln::util::ord_pair, 1270
- change_extension
 - mln::extension_val, 767
- change_first
 - mln::util::couple, 1238
 - mln::util::ord_pair, 1271
- change_graph
 - mln::util::edge, 1243
 - mln::util::vertex, 1295
- change_key
 - mln::p_key, 1024
- change_keys
 - mln::p_key, 1024
- change_mask
 - mln::graph_elt_window_if, 858
- change_second
 - mln::util::couple, 1238
 - mln::util::ord_pair, 1271
- change_target
 - mln::complex_psite, 681
 - mln::faces_psite, 770
 - mln::p_transformed_piter, 1091
- change_target_site_set
 - mln::graph_window_piter, 866
- change_to
 - mln::pixel, 1107
- check_consistency
 - mln::util::tree, 1288
 - mln::util::tree_node, 1290
- children
 - mln::util::tree_node, 1291
- clear
 - mln::p_array, 974
 - mln::p_image, 1013
 - mln::p_key, 1024
 - mln::p_mutable_array_of, 1037
 - mln::p_priority, 1047
 - mln::p_queue, 1055
 - mln::p_queue_fast, 1062
 - mln::p_set, 1076
 - mln::p_set_of, 1082
 - mln::util::array, 1229
 - mln::util::fibonacci_heap, 1246
 - mln::util::set, 1276
 - mln::util::soft_heap, 1282
- mln::w_window, 1349
- mln::window, 1370
- closing
 - mln::morpho::elementary, 425
- colorize
 - mln::labeling, 353
- complementation
 - mln::morpho, 416
- complementation_inplace
 - mln::morpho, 416
- complex
 - mln::topo::complex, 1191
- Complex based, 121
- complex_geometry
 - mln::geom::complex_geometry, 832
- complex_image
 - mln::complex_image, 674
- complex_neighborhood_bkd_piter
 - mln::complex_neighborhood_bkd_piter, 677
- complex_neighborhood_fwd_piter
 - mln::complex_neighborhood_fwd_piter, 679
- complex_psite
 - mln::complex_psite, 681
- complex_window_bkd_piter
 - mln::complex_window_bkd_piter, 684
- complex_window_fwd_piter
 - mln::complex_window_fwd_piter, 686
- compose
 - mln, 165
- composed
 - mln::fun::x2x::composed, 807
- compute
 - mln::accu, 181
 - mln::data, 233
 - mln::graph, 307
 - mln::histo, 311
 - mln::labeling, 353–355
 - mln::labeling::impl::generic, 364, 365
 - mln::set, 459
- compute_attribute_image
 - mln::morpho::tree, 434
- compute_attribute_image_from
 - mln::morpho::tree, 434
- compute_has
 - mln::p_queue_fast, 1062
- compute_image
 - mln::labeling, 356
- compute_parent
 - mln::morpho::tree, 435
- compute_with_weights
 - mln::set, 460
- contrast
 - mln::morpho, 416
- convert

- mln::data, 234
- mln::data::impl::generic, 249
- convolve
 - mln::linear::local, 371
- coord
 - mln::def, 261
 - mln::doc::Dpoint, 697
 - mln::doc::Fastest_Image, 701
 - mln::doc::Image, 711
 - mln::doc::Point_Site, 723
 - mln::dpoint, 740
 - mln::point, 1118
- coordf
 - mln::def, 261
- count
 - mln::accu::stat::mean, 599
- couple
 - mln::make, 392
- cplx
 - mln::p_complex, 986
 - mln::p_faces, 1000
 - mln::topo::algebraic_face, 1179
 - mln::topo::algebraic_n_face, 1184
 - mln::topo::face, 1194
 - mln::topo::n_face, 1205
- crop_wrt
 - mln::box, 643
- cube3d
 - mln::win::cube3d, 1354
- cuboid3d
 - mln::win::cuboid3d, 1357
- cyan
 - mln::literal, 376
- D
 - mln::topo::is_simple_cell, 1203
- dark_gray
 - mln::literal, 376
- data
 - mln::topo::algebraic_face, 1179
 - mln::topo::algebraic_n_face, 1184
 - mln::topo::face, 1195
 - mln::topo::n_face, 1205
- dec_face_id
 - mln::topo::algebraic_face, 1179
 - mln::topo::algebraic_n_face, 1184
 - mln::topo::face, 1195
 - mln::topo::n_face, 1205
- dec_n
 - mln::topo::algebraic_face, 1179
 - mln::topo::face, 1195
- decorated_image
 - mln::decorated_image, 688
- decoration
 - mln::decorated_image, 688
- deepness
 - mln::util::branch_iter, 1234
 - mln::util::branch_iter_ind, 1236
- delete_tree_node
 - mln::util::tree_node, 1291
- delta
 - mln::doc::Weighted_Window, 735
 - mln::geom, 297
 - mln::graph_elt_mixed_window, 845
 - mln::graph_elt_window, 853
 - mln::graph_elt_window_if, 858
 - mln::graph_window_base, 861
 - mln::point, 1118
 - mln::window, 1370
- delta_index
 - mln::doc::Fastest_Image, 703
 - mln::image1d, 877
 - mln::image2d, 882
 - mln::image3d, 891
- depth
 - mln::win::cuboid3d, 1357
- detach
 - mln::topo, 471
- detachment
 - mln::make, 393
- diag2d
 - mln::win::diag2d, 1358
- diameter
 - mln::win::ball, 1353
- diff
 - mln::Box, 650
 - mln::box, 645
 - mln::p_array, 976
 - mln::p_centered, 981
 - mln::p_complex, 987
 - mln::p_edges, 994
 - mln::p_faces, 1000
 - mln::p_if, 1008
 - mln::p_image, 1014
 - mln::p_key, 1026
 - mln::p_line2d, 1032
 - mln::p_mutable_array_of, 1038
 - mln::p_priority, 1050
 - mln::p_queue, 1056
 - mln::p_queue_fast, 1064
 - mln::p_run, 1071
 - mln::p_set, 1077
 - mln::p_set_of, 1083
 - mln::p_transformed, 1088
 - mln::p_vaccess, 1096
 - mln::p_vertices, 1104
 - mln::Site_Set, 1146
 - mln::win, 505

- diff_abs
 - mln::arith, 198
- dilation
 - mln::morpho, 416
- dim
 - mln::complex_image, 675
 - mln::doc::Dpoint, 698
 - mln::doc::Point_Site, 724
 - mln::dpoint, 741
 - mln::point, 1118
- direct
 - mln::morpho::tree::filter, 440
- discrete_plane_1complex_geometry
 - mln, 162
- discrete_plane_2complex_geometry
 - mln, 162
- disk2d
 - modwin2d, 128
- display_branch
 - mln::util, 487
- display_tree
 - mln::util, 487
- distance_and_closest_point_geodesic
 - mln::transform, 480
- distance_and_influence_zone_geodesic
 - mln::transform, 481
- distance_front
 - mln::canvas, 218
 - mln::transform, 481
- distance_geodesic
 - mln::canvas, 218
 - mln::transform, 481
- div
 - mln::arith, 198
- div_cst
 - mln::arith, 198
- div_inplace
 - mln::arith, 199
- do_it
 - mln::io::magick, 323
- domain
 - mln::complex_image, 675
 - mln::doc::Fastest_Image, 703
 - mln::doc::Image, 712
 - mln::extended, 759
 - mln::flat_image, 773
 - mln::hexa, 870
 - mln::image1d, 877
 - mln::image2d, 883
 - mln::image2d_h, 887
 - mln::image3d, 891
 - mln::image_if, 894
 - mln::lazy_image, 908
 - mln::p2p_image, 970
 - mln::slice_image, 1151
 - mln::sub_image, 1153
 - mln::sub_image_if, 1155
 - mln::tr_image, 1220
 - mln::transformed_image, 1222
 - mln::unproject_image, 1223
- Domain morphers, 103
- domain_t
 - mln::value::stack_image, 1333
- dp
 - mln::window, 1370
- dpoint
 - mln::doc::Dpoint, 697
 - mln::doc::Fastest_Image, 701
 - mln::doc::Image, 711
 - mln::doc::Neighborhood, 717
 - mln::doc::Point_Site, 723
 - mln::doc::Weighted_Window, 735
 - mln::dpoint, 741
- dpoint1d
 - mln, 162
- dpoint2d
 - mln, 162
- dpoint2d_h
 - mln, 162
 - mln::make, 393
- dpoint3d
 - mln, 162
- dpoints_bkd_pixter
 - mln::dpoints_bkd_pixter, 745
- dpoints_fwd_pixter
 - mln::dpoints_fwd_pixter, 748
- dpsite
 - mln::point, 1118
 - mln::w_window, 1348
- dpsites_bkd_piter
 - mln::dpsites_bkd_piter, 750
- dpsites_fwd_piter
 - mln::dpsites_fwd_piter, 752
- draw_graph
 - mln::debug, 256, 257
- dual_input_max_tree
 - mln::morpho::tree, 436
- dummy_p_edges
 - mln::make, 393, 394
- dummy_p_vertices
 - mln::make, 394
- duplicate
 - mln, 166
 - mln::border, 212
 - mln::extension, 272
- e_ith_nbh_edge
 - mln::util::graph, 1252

- mln::util::line_graph, 1263
- e_nmax
 - mln::util::graph, 1252
 - mln::util::line_graph, 1263
- e_nmax_nbh_edges
 - mln::util::graph, 1252
 - mln::util::line_graph, 1263
- edge
 - mln::p_edges, 991
 - mln::topo, 471
 - mln::util::edge, 1242
 - mln::util::graph, 1252
 - mln::util::line_graph, 1263
- edge_fwd_iter
 - mln::util::graph, 1250
 - mln::util::line_graph, 1262
- edge_image
 - mln::edge_image, 756
 - mln::make, 394–396
- edge_nbh_edge_fwd_iter
 - mln::util::graph, 1250
 - mln::util::line_graph, 1262
- edge_nbh_t
 - mln::edge_image, 756
- edge_win_t
 - mln::edge_image, 756
- edge_with
 - mln::util::vertex, 1295
- edges
 - mln::util::graph, 1252
- edges_set_t
 - mln::util::graph, 1250
- edges_t
 - mln::util::graph, 1250
 - mln::util::line_graph, 1262
- eiter
 - mln::util::array, 1228
 - mln::util::set, 1275
- element
 - mln::box, 642
 - mln::graph_window_if_piter, 863
 - mln::graph_window_piter, 866
 - mln::image1d, 878
 - mln::image2d, 883
 - mln::image3d, 891
 - mln::p_array, 973
 - mln::p_centered, 980
 - mln::p_complex, 985
 - mln::p_edges, 991
 - mln::p_faces, 999
 - mln::p_if, 1007
 - mln::p_image, 1012
 - mln::p_key, 1023
 - mln::p_line2d, 1030
 - mln::p_mutable_array_of, 1036
 - mln::p_priority, 1046
 - mln::p_queue, 1054
 - mln::p_queue_fast, 1061
 - mln::p_run, 1068
 - mln::p_set, 1075
 - mln::p_set_of, 1082
 - mln::p_transformed, 1087
 - mln::p_vaccess, 1094
 - mln::p_vertices, 1100
 - mln::util::array, 1228
 - mln::util::set, 1276
 - mln::util::soft_heap, 1282
- elt
 - mln::util::tree_node, 1291
- empty
 - mln::p_queue_fast, 1062
- enc
 - mln::value::float01, 1300
 - mln::value::label, 1319
 - mln::value::proxy, 1325
 - mln::value::sign, 1331
- end
 - mln::p_line2d, 1031
 - mln::p_run, 1069
- enlarge
 - mln::box, 643
- equalize
 - mln::border, 212
- equiv
 - mln::value, 498
 - mln::value::float01, 1300
 - mln::value::proxy, 1325
 - mln::value::sign, 1331
- erosion
 - mln::morpho, 416
- exists_key
 - mln::p_key, 1024
- exists_priority
 - mln::p_priority, 1047
- extend
 - mln, 166
- extended
 - mln::extended, 759
- extension
 - mln::extension_fun, 761
 - mln::extension_ima, 764
 - mln::extension_val, 767
- extension_fun
 - mln::extension_fun, 761
- extension_ima
 - mln::extension_ima, 764
- extension_val
 - mln::extension_val, 767

- f_hsi_to_rgb_3x8
 - mln::fun::v2v, 285
- f_hsl_to_rgb_3x8
 - mln::fun::v2v, 285
- f_rgb_to_hsi_f
 - mln::fun::v2v, 285
- f_rgb_to_hsl_f
 - mln::fun::v2v, 285
- face
 - mln::complex_psite, 681
 - mln::faces_psite, 770
 - mln::topo::face, 1194
- face_bkd_iter
 - mln::topo::face_bkd_iter, 1197
- face_fwd_iter
 - mln::topo::face_fwd_iter, 1199
- face_id
 - mln::complex_psite, 681
 - mln::faces_psite, 770
 - mln::topo::algebraic_face, 1179
 - mln::topo::algebraic_n_face, 1184
 - mln::topo::face, 1195
 - mln::topo::n_face, 1205
- faces
 - mln::topo::n_faces_set, 1212
- faces_psite
 - mln::faces_psite, 770
- faces_type
 - mln::topo::n_faces_set, 1212
- fast_median
 - mln::data, 234
- fibonacci_heap
 - mln::util::fibonacci_heap, 1246
- filename
 - mln::debug, 257
- fill
 - mln::border, 212
 - mln::data, 234
 - mln::extension, 272
 - mln::util::array, 1229
- fill_holes
 - mln::labeling, 357
- fill_with_image
 - mln::data, 235
 - mln::data::impl::generic, 249
- fill_with_value
 - mln::data, 235
 - mln::data::impl::generic, 249
- filter
 - mln::morpho::tree::filter, 440
- find
 - mln::border, 213
- first
 - mln::util::couple, 1239
 - mln::util::ord_pair, 1271
 - mln::util::site_pair, 1280
- first_element
 - mln::util::set, 1276
- flat_image
 - mln::flat_image, 773
- flat_zones
 - mln::labeling, 357
- float01
 - mln::value::float01, 1300
- float01_16
 - mln::value, 496
- float01_8
 - mln::value, 496
- float01_f
 - mln::value::float01_f, 1302
- float_2complex_image3df
 - mln, 162
- flooding
 - mln::morpho::watershed, 443, 444
- foreground
 - mln::labeling, 358
- format
 - mln::debug, 257
- from_to
 - mln::convert, 226
- front
 - mln::p_priority, 1047
 - mln::p_queue, 1055
 - mln::p_queue_fast, 1062
 - mln::util::fibonacci_heap, 1246
- fun
 - mln::p2p_image, 970
- fun_image
 - mln::fun_image, 816
- fun_t
 - mln::p_edges, 991
 - mln::p_vertices, 1101
- Function
 - mln::Function, 817
- function
 - mln::p_edges, 993
 - mln::p_transformed, 1087
 - mln::p_vertices, 1102
- Functions, 109
- fwd_citer
 - mln::topo::complex, 1191
- fwd_eiter
 - mln::util::array, 1228
 - mln::util::set, 1276
- fwd_niter
 - mln::doc::Neighborhood, 717
 - mln::graph_elt_mixed_neighborhood, 841
 - mln::graph_elt_neighborhood, 847

- mln::graph_elt_neighborhood_if, 849
- mln::mixed_neighb, 949
- mln::neighb, 965
- fwd_piter
 - mln::box, 642
 - mln::doc::Box, 695
 - mln::doc::Fastest_Image, 701
 - mln::doc::Image, 711
 - mln::doc::Site_Set, 729
 - mln::hexa, 869
 - mln::image2d_h, 886
 - mln::p_array, 973
 - mln::p_centered, 980
 - mln::p_complex, 985
 - mln::p_edges, 992
 - mln::p_faces, 999
 - mln::p_if, 1007
 - mln::p_image, 1012
 - mln::p_key, 1023
 - mln::p_line2d, 1030
 - mln::p_mutable_array_of, 1036
 - mln::p_priority, 1046
 - mln::p_queue, 1054
 - mln::p_queue_fast, 1061
 - mln::p_run, 1068
 - mln::p_set, 1075
 - mln::p_set_of, 1082
 - mln::p_transformed, 1087
 - mln::p_vaccess, 1094
 - mln::p_vertices, 1101
- fwd_pixter1d
 - mln::fwd_pixter1d, 823
- fwd_pixter2d
 - mln::fwd_pixter2d, 825
- fwd_pixter3d
 - mln::fwd_pixter3d, 827
- fwd_qiter
 - mln::doc::Weighted_Window, 735
 - mln::doc::Window, 737
 - mln::graph_elt_mixed_window, 844
 - mln::graph_elt_window, 852
 - mln::graph_elt_window_if, 856
 - mln::w_window, 1348
 - mln::window, 1369
- fwd_viter
 - mln::doc::Value_Set, 733
 - mln::value::lut_vec, 1322
- gaussian
 - mln::linear, 367
- gaussian_1st_derivative
 - mln::linear, 367
- gaussian_2nd_derivative
 - mln::linear, 368
- gaussian_subsampling
 - mln::subsampling, 462
- general
 - mln::morpho, 416
- geom
 - mln::complex_image, 674
 - mln::p_complex, 986
- get
 - mln::border, 213
 - mln::set, 461
- get_color
 - mln::io::magick, 323
- get_rot
 - mln::registration, 456
- gl16
 - mln::value, 496
- gl8
 - mln::value, 496
- glf
 - mln::value, 497
- gradient
 - mln::morpho, 417
- gradient_external
 - mln::morpho, 417
- gradient_internal
 - mln::morpho, 417
- graph
 - mln::p_edges, 993
 - mln::p_graph_piter, 1003
 - mln::p_vertices, 1102
 - mln::util::edge, 1243
 - mln::util::graph, 1251
 - mln::util::line_graph, 1263
 - mln::util::vertex, 1295
- Graph based, 120
- graph_element
 - mln::graph_elt_mixed_window, 844
 - mln::graph_elt_window, 852
 - mln::graph_window_piter, 865
 - mln::p_edges, 992
 - mln::p_vertices, 1101
- graph_elt_neighborhood_if
 - mln::graph_elt_neighborhood_if, 850
- graph_elt_window_if
 - mln::graph_elt_window_if, 857
- graph_t
 - mln::edge_image, 756
 - mln::p_edges, 992
 - mln::p_vertices, 1101
 - mln::util::edge, 1242
 - mln::util::vertex, 1294
 - mln::vertex_image, 1343
- graph_window_if_piter
 - mln::graph_window_if_piter, 863

- graph_window_piter
 - mln::graph_window_piter, 865, 866
- Graphes, 98
- graylevel
 - mln::value::graylevel, 1305
- graylevel_f
 - mln::value::graylevel_f, 1308
- green
 - mln::literal, 376
- grid
 - mln::dpoint, 740
 - mln::point, 1118
- h_mat
 - mln::algebra::h_mat, 629
 - mln::make, 396
- h_vec
 - mln::algebra::h_vec, 632
 - mln::point, 1118
- has
 - mln::box, 644
 - mln::doc::Box, 695
 - mln::doc::Fastest_Image, 703, 704
 - mln::doc::Image, 712, 713
 - mln::doc::Site_Set, 729
 - mln::doc::Value_Set, 733
 - mln::extension_fun, 761
 - mln::extension_ima, 764
 - mln::extension_val, 767
 - mln::flat_image, 773
 - mln::hexa, 870
 - mln::image1d, 878
 - mln::image2d, 883
 - mln::image2d_h, 887
 - mln::image3d, 891
 - mln::interpolated, 896
 - mln::lazy_image, 908
 - mln::p_array, 974, 975
 - mln::p_centered, 980
 - mln::p_complex, 986
 - mln::p_edges, 993
 - mln::p_if, 1007
 - mln::p_image, 1013
 - mln::p_key, 1025
 - mln::p_line2d, 1031
 - mln::p_mutable_array_of, 1037
 - mln::p_priority, 1048
 - mln::p_queue, 1055
 - mln::p_queue_fast, 1062, 1063
 - mln::p_run, 1069
 - mln::p_set, 1076
 - mln::p_set_of, 1082
 - mln::p_transformed, 1087
 - mln::p_vaccess, 1095
 - mln::p_vertices, 1103
 - mln::set, 461
 - mln::tr_image, 1220
 - mln::util::line_graph, 1263
 - mln::util::set, 1276
 - mln::value::lut_vec, 1323
 - mln::window, 1370
- has_e
 - mln::util::graph, 1252
 - mln::util::line_graph, 1264
- has_index
 - mln::p_run, 1069
- has_v
 - mln::util::graph, 1252
 - mln::util::line_graph, 1264
- height
 - mln::morpho::attribute::sharpness, 958
 - mln::win::cuboid3d, 1357
 - mln::win::rectangle2d, 1366
- hexa
 - mln::hexa, 870
- higher_dim_adj_faces
 - mln::topo::algebraic_face, 1179
 - mln::topo::algebraic_n_face, 1184
 - mln::topo::face, 1195
 - mln::topo::n_face, 1206
- highest_priority
 - mln::p_priority, 1048
- hit_or_miss
 - mln::morpho, 417
 - mln::morpho::impl::generic, 428
- hit_or_miss_background_closing
 - mln::morpho, 417
- hit_or_miss_background_opening
 - mln::morpho, 418
- hit_or_miss_closing
 - mln::morpho, 418
- hit_or_miss_opening
 - mln::morpho, 418
- hline2d
 - modwin2d, 128
- hough
 - mln::transform, 481
- i_element
 - mln::p_array, 973
 - mln::p_image, 1012
 - mln::p_key, 1024
 - mln::p_mutable_array_of, 1036
 - mln::p_priority, 1047
 - mln::p_queue, 1054
 - mln::p_queue_fast, 1062
 - mln::p_set, 1075
 - mln::p_set_of, 1082

- mln::p_vaccess, 1094
- icp
 - mln::registration, 456
- id
 - mln::graph_window_if_piter, 863
 - mln::graph_window_piter, 866
 - mln::p_graph_piter, 1003
 - mln::util::edge, 1243
 - mln::util::vertex, 1295
- id_t
 - mln::util::edge, 1242
 - mln::util::vertex, 1294
- id_value_t
 - mln::util::edge, 1242
 - mln::util::vertex, 1294
- identity
 - mln::literal, 376
- Identity morphers, 104
- ima
 - mln::doc::Generalized_Pixel, 708
 - mln::doc::Pixel_Iterator, 721
 - mln::fun::x2x::linear, 809
 - mln::util::pix, 1273
- image
 - mln::bkd_pixter1d, 633
 - mln::bkd_pixter2d, 635
 - mln::bkd_pixter3d, 637
 - mln::doc::Generalized_Pixel, 707
 - mln::doc::Pixel_Iterator, 721
 - mln::fwd_pixter1d, 823
 - mln::fwd_pixter2d, 825
 - mln::fwd_pixter3d, 827
 - mln::make, 396, 397
 - mln::pw::image, 1133
- Image morphers, 101
- image1d
 - mln::image1d, 877
- image2d
 - mln::image2d, 882
 - mln::make, 397
- image2d_h
 - mln::image2d_h, 887
- image3d
 - mln::image3d, 890
 - mln::make, 397, 398
- image_if
 - mln::image_if, 893
- Images, 99
- implies
 - mln, 166
- inc_face_id
 - mln::topo::algebraic_face, 1180
 - mln::topo::algebraic_n_face, 1184
 - mln::topo::face, 1195
- mln::topo::n_face, 1206
- inc_n
 - mln::topo::algebraic_face, 1180
 - mln::topo::face, 1195
- index
 - mln::p_indexed_bkd_piter, 1016
 - mln::p_indexed_fwd_piter, 1018
- index_of
 - mln::doc::Value_Set, 733
 - mln::value::lut_vec, 1323
- influence_zone_adjacency_graph
 - mln::make, 398
- influence_zone_front
 - mln::transform, 482
- influence_zone_geodesic
 - mln::transform, 482
- influence_zone_geodesic_saturated
 - mln::transform, 482
- init
 - mln::accu::center, 508
 - mln::accu::convolve, 509
 - mln::accu::count_adjacent_vertices, 511
 - mln::accu::count_labels, 513
 - mln::accu::count_value, 515
 - mln::accu::label_used, 519
 - mln::accu::logic::land, 521
 - mln::accu::logic::land_basic, 523
 - mln::accu::logic::lor, 525
 - mln::accu::logic::lor_basic, 527
 - mln::accu::maj_h, 529
 - mln::accu::math::count, 531
 - mln::accu::math::inf, 533
 - mln::accu::math::sum, 535
 - mln::accu::math::sup, 537
 - mln::accu::max_site, 539
 - mln::accu::nil, 575
 - mln::accu::p, 577
 - mln::accu::pair, 579
 - mln::accu::rms, 581
 - mln::accu::shape::bbox, 583
 - mln::accu::shape::height, 586
 - mln::accu::shape::volume, 589
 - mln::accu::stat::deviation, 593
 - mln::accu::stat::max, 595
 - mln::accu::stat::max_h, 597
 - mln::accu::stat::mean, 599
 - mln::accu::stat::median_h, 603
 - mln::accu::stat::min, 606
 - mln::accu::stat::min_h, 608
 - mln::accu::stat::min_max, 611
 - mln::accu::stat::rank, 612
 - mln::accu::stat::rank < bool >, 614
 - mln::accu::stat::rank_high_quant, 616
 - mln::accu::stat::var, 619

- mln::accu::stat::variance, 622
- mln::accu::tuple, 624
- mln::accu::val, 626
- mln::doc::Accumulator, 692
- mln::morpho::attribute::card, 951
- mln::morpho::attribute::count_adjacent_vertices, 953
- mln::morpho::attribute::height, 955
- mln::morpho::attribute::sharpness, 958
- mln::morpho::attribute::sum, 960
- mln::morpho::attribute::volume, 962
- mln::p_run, 1070
- initialize
 - mln, 167
- insert
 - mln::p_array, 975
 - mln::p_image, 1013
 - mln::p_key, 1025
 - mln::p_mutable_array_of, 1037
 - mln::p_priority, 1048
 - mln::p_queue, 1055
 - mln::p_queue_fast, 1063
 - mln::p_set, 1076
 - mln::p_set_of, 1083
 - mln::p_vaccess, 1095
 - mln::util::set, 1277
 - mln::w_window, 1349
 - mln::window, 1370, 1371
- int_s
 - mln::value::int_s, 1311
- int_s16
 - mln::value, 497
- int_s32
 - mln::value, 497
- int_s8
 - mln::value, 497
- int_u
 - mln::value::int_u, 1312, 1313
- int_u12
 - mln::value, 497
- int_u16
 - mln::value, 497
- int_u32
 - mln::value, 497
- int_u8
 - mln::value, 497
- int_u8_1complex_image2d
 - mln, 163
- int_u8_2complex_image2d
 - mln, 163
- int_u8_2complex_image3df
 - mln, 163
- int_u_sat
 - mln::value::int_u_sat, 1315
- inter
 - mln::Box, 650
 - mln::box, 645
 - mln::p_array, 976
 - mln::p_centered, 981
 - mln::p_complex, 987
 - mln::p_edges, 994
 - mln::p_faces, 1000
 - mln::p_if, 1008
 - mln::p_image, 1014
 - mln::p_key, 1026
 - mln::p_line2d, 1032
 - mln::p_mutable_array_of, 1038
 - mln::p_priority, 1050
 - mln::p_queue, 1056
 - mln::p_queue_fast, 1064
 - mln::p_run, 1071
 - mln::p_set, 1077
 - mln::p_set_of, 1083
 - mln::p_transformed, 1088
 - mln::p_vaccess, 1096
 - mln::p_vertices, 1104
 - mln::Site_Set, 1146
- interpolated
 - mln::interpolated, 896
- inv
 - mln::fun::x2x::rotation, 811
 - mln::fun::x2x::translation, 814
- invalidate
 - mln::complex_psite, 681
 - mln::doc::Iterator, 715
 - mln::doc::Pixel_Iterator, 721
 - mln::doc::Site_Iterator, 727
 - mln::doc::Value_Iterator, 731
 - mln::dpoints_bkd_pixter, 745
 - mln::dpoints_fwd_pixter, 748
 - mln::faces_psite, 770
 - mln::p_edges, 994
 - mln::p_vertices, 1103
 - mln::topo::algebraic_face, 1180
 - mln::topo::algebraic_n_face, 1184
 - mln::topo::face, 1195
 - mln::topo::n_face, 1206
 - mln::util::branch_iter, 1234
 - mln::util::branch_iter_ind, 1236
 - mln::util::edge, 1243
 - mln::util::vertex, 1295
- invert
 - mln::fun::x2x::rotation, 811
 - mln::fun::x2x::translation, 814
- iota
 - mln::debug, 257
- is_centered
 - mln::doc::Weighted_Window, 735

- mln::graph_elt_mixed_window, 845
- mln::graph_elt_window, 853
- mln::graph_elt_window_if, 858
- mln::graph_window_base, 861
- mln::window, 1371
- is_empty
 - mln::Box, 650
 - mln::box, 644
 - mln::doc::Weighted_Window, 736
 - mln::graph_elt_mixed_window, 845
 - mln::graph_elt_window, 853
 - mln::graph_elt_window_if, 858
 - mln::graph_window_base, 861
 - mln::util::array, 1229
 - mln::util::fibonacci_heap, 1246
 - mln::util::set, 1277
 - mln::util::soft_heap, 1282
 - mln::window, 1371
- is_facet
 - mln::topo, 472
- is_simple_2d
 - mln, 167
- is_subgraph_of
 - mln::util::graph, 1253
 - mln::util::line_graph, 1264
- is_symmetric
 - mln::graph_elt_mixed_window, 845
 - mln::graph_elt_window, 853
 - mln::graph_elt_window_if, 858
 - mln::graph_window_base, 861
 - mln::w_window, 1349
 - mln::window, 1371
- is_valid
 - mln::accu::center, 508
 - mln::accu::convolve, 509
 - mln::accu::count_adjacent_vertices, 511
 - mln::accu::count_labels, 513
 - mln::accu::count_value, 515
 - mln::accu::histo, 517
 - mln::accu::label_used, 519
 - mln::accu::logic::land, 521
 - mln::accu::logic::land_basic, 523
 - mln::accu::logic::lor, 525
 - mln::accu::logic::lor_basic, 527
 - mln::accu::maj_h, 529
 - mln::accu::math::count, 531
 - mln::accu::math::inf, 533
 - mln::accu::math::sum, 535
 - mln::accu::math::sup, 537
 - mln::accu::max_site, 539
 - mln::accu::nil, 575
 - mln::accu::p, 577
 - mln::accu::pair, 579
 - mln::accu::rms, 581
 - mln::accu::shape::bbox, 583
 - mln::accu::shape::height, 586
 - mln::accu::shape::volume, 589
 - mln::accu::stat::deviation, 593
 - mln::accu::stat::max, 595
 - mln::accu::stat::max_h, 597
 - mln::accu::stat::mean, 600
 - mln::accu::stat::median_alt, 601
 - mln::accu::stat::median_h, 603
 - mln::accu::stat::min, 606
 - mln::accu::stat::min_h, 608
 - mln::accu::stat::min_max, 611
 - mln::accu::stat::rank, 612
 - mln::accu::stat::rank< bool >, 614
 - mln::accu::stat::rank_high_quant, 616
 - mln::accu::stat::var, 619
 - mln::accu::stat::variance, 622
 - mln::accu::tuple, 624
 - mln::accu::val, 626
 - mln::box, 644
 - mln::complex_psite, 681
 - mln::doc::Fastest_Image, 704
 - mln::doc::Image, 713
 - mln::doc::Iterator, 715
 - mln::doc::Pixel_Iterator, 721
 - mln::doc::Site_Iterator, 727
 - mln::doc::Value_Iterator, 731
 - mln::dpoints_bkd_pixter, 745
 - mln::dpoints_fwd_pixter, 748
 - mln::faces_psite, 770
 - mln::graph_elt_mixed_window, 845
 - mln::graph_elt_window, 854
 - mln::graph_elt_window_if, 858
 - mln::graph_window_base, 861
 - mln::interpolated, 896
 - mln::morpho::attribute::card, 951
 - mln::morpho::attribute::count_adjacent_vertices, 953
 - mln::morpho::attribute::height, 955
 - mln::morpho::attribute::sharpness, 958
 - mln::morpho::attribute::sum, 960
 - mln::morpho::attribute::volume, 963
 - mln::p_array, 975
 - mln::p_centered, 981
 - mln::p_complex, 986
 - mln::p_edges, 994
 - mln::p_faces, 1000
 - mln::p_if, 1007
 - mln::p_image, 1013
 - mln::p_key, 1025
 - mln::p_line2d, 1031
 - mln::p_mutable_array_of, 1037
 - mln::p_priority, 1048
 - mln::p_queue, 1055

- mln::p_queue_fast, 1063
- mln::p_run, 1070
- mln::p_set, 1076
- mln::p_set_of, 1083
- mln::p_transformed, 1088
- mln::p_vaccess, 1095
- mln::p_vertices, 1103
- mln::pixel, 1107
- mln::topo::algebraic_face, 1180
- mln::topo::algebraic_n_face, 1184
- mln::topo::face, 1195
- mln::topo::n_face, 1206
- mln::tr_image, 1220
- mln::util::branch_iter, 1234
- mln::util::branch_iter_ind, 1236
- mln::util::edge, 1243
- mln::util::fibonacci_heap, 1246
- mln::util::soft_heap, 1282
- mln::util::vertex, 1295
- mln::value::stack_image, 1334
- iter
 - mln::complex_neighborhood_bkd_piter, 677
 - mln::complex_neighborhood_fwd_piter, 679
 - mln::complex_window_bkd_piter, 684
 - mln::complex_window_fwd_piter, 686
- iter_type
 - mln::complex_neighborhood_bkd_piter, 676
 - mln::complex_neighborhood_fwd_piter, 678
 - mln::complex_window_bkd_piter, 683
 - mln::complex_window_fwd_piter, 685
- ith_nbh_edge
 - mln::util::edge, 1243
 - mln::util::vertex, 1295
- ith_nbh_vertex
 - mln::util::vertex, 1296
- k
 - mln::accu::stat::rank, 612
- key
 - mln::p_key, 1025
- keys
 - mln::p_key, 1025
- l1
 - mln::norm, 449
- l1_distance
 - mln::norm, 449
- l2
 - mln::norm, 449
- l2_distance
 - mln::norm, 449
- label
 - mln::value::label, 1319
- label_16
 - mln::value, 497
- label_32
 - mln::value, 497
- label_8
 - mln::value, 497
- labeled_image
 - mln::labeled_image, 901, 902
- labeled_image_base
 - mln::labeled_image_base, 905
- labeling
 - mln::graph, 307
- laplacian
 - mln::morpho, 418
- larger_than
 - mln, 167
- last_coord
 - mln::point, 1119
- last_element
 - mln::util::set, 1277
- lazy_image
 - mln::lazy_image, 908
- ldlt_decomp
 - mln::algebra, 194
- ldlt_solve
 - mln::algebra, 194
- lemmings
 - mln::util, 488
- len
 - mln::Box, 650
 - mln::box, 644
- length
 - mln::p_run, 1070
 - mln::win::backdiag2d, 1352
 - mln::win::cube3d, 1355
 - mln::win::diag2d, 1358
 - mln::win::line, 1360
 - mln::win::octagon2d, 1364
- light_gray
 - mln::literal, 376
- lime
 - mln::literal, 376
- line
 - mln::accu, 181
 - mln::draw, 267
 - mln::win::line, 1360
- line_gradient
 - mln::morpho, 418
- linear
 - mln::fun::x2x::linear, 808
- linfty
 - mln::norm, 449
- linfty_distance
 - mln::norm, 449
- load

- mln::io::cloud, 317
- mln::io::dicom, 318
- mln::io::dump, 319
- mln::io::fits, 320
- mln::io::fld, 321
- mln::io::magick, 323
- mln::io::off, 325
- mln::io::pbm, 327
- mln::io::pbms, 330
- mln::io::pfm, 332
- mln::io::pgm, 335
- mln::io::pgms, 337
- mln::io::plot, 338
- mln::io::pnm, 340, 341
- mln::io::pnms, 343
- mln::io::ppm, 344
- mln::io::ppms, 346
- mln::io::tiff, 347
- load_ascii_builtin
 - mln::io::pnm, 341
- load_ascii_value
 - mln::io::pnm, 341
- load_raw_2d
 - mln::io::pnm, 341
- lower_dim_adj_faces
 - mln::topo::algebraic_face, 1180
 - mln::topo::algebraic_n_face, 1185
 - mln::topo::face, 1195
 - mln::topo::n_face, 1206
- lowest_priority
 - mln::p_priority, 1048
- lut_vec
 - mln::value::lut_vec, 1322
- lvalue
 - mln::complex_image, 674
 - mln::decorated_image, 688
 - mln::doc::Fastest_Image, 701
 - mln::doc::Image, 711
 - mln::doc::Pixel_Iterator, 721
 - mln::flat_image, 773
 - mln::fun_image, 816
 - mln::hexa, 869
 - mln::image1d, 876
 - mln::image2d, 881
 - mln::image2d_h, 886
 - mln::image3d, 889
 - mln::interpolated, 895
 - mln::lazy_image, 908
 - mln::tr_image, 1219
 - mln::value::stack_image, 1333
 - mln::violent_cast_image, 1345
- magenta
 - mln::literal, 377
- main_branch
 - mln::util::tree, 1288
- make_algebraic_face
 - mln::topo, 472
- make_algebraic_n_face
 - mln::topo, 472
- make_debug_graph_image
 - mln, 167
- make_greater_point
 - mln::util, 488
- make_greater_psite
 - mln::util, 488
- mask
 - mln::graph_elt_neighborhood_if, 850
 - mln::graph_elt_window_if, 858
- mask_t
 - mln::graph_elt_window_if, 857
- mat
 - mln::make, 398
- max
 - mln::literal, 377
 - mln::morpho::tree::filter, 441
- max_col
 - mln::geom, 297, 298
- max_component
 - mln::io::pnm, 341
- max_ind
 - mln::geom, 298
- max_row
 - mln::geom, 298
- max_sli
 - mln::geom, 298
- max_tree
 - mln::morpho::tree, 436
- mean
 - mln::accu::stat::var, 619
 - mln::accu::stat::variance, 622
 - mln::estim, 269
- mean_t
 - mln::accu::stat::var, 619
- median
 - mln::data, 235
 - mln::data::approx, 243, 244
 - mln::data::impl::generic, 249
 - mln::data::naive, 253
- medium_gray
 - mln::literal, 377
- memory_size
 - mln::box, 644
 - mln::p_array, 975
 - mln::p_centered, 981
 - mln::p_edges, 994
 - mln::p_if, 1008
 - mln::p_image, 1013

- mln::p_key, 1025
- mln::p_line2d, 1031
- mln::p_mutable_array_of, 1037
- mln::p_priority, 1048
- mln::p_queue, 1055
- mln::p_queue_fast, 1063
- mln::p_run, 1070
- mln::p_set, 1076
- mln::p_set_of, 1083
- mln::p_transformed, 1088
- mln::p_vaccess, 1095
- mln::p_vertices, 1103
- mln::util::array, 1229
- mln::util::set, 1277
- mesh
 - mln::doc::Point_Site, 724
- mesh_corner_point_area
 - mln::geom, 298
- mesh_curvature
 - mln::geom, 299
- mesh_normal
 - mln::geom, 299
- meyer_wst
 - mln::morpho, 418, 419
- min
 - mln::arith, 199
 - mln::literal, 377
 - mln::morpho, 419
 - mln::morpho::tree::filter, 441
- min_col
 - mln::geom, 299
- min_ind
 - mln::geom, 300
- min_inplace
 - mln::arith, 200
 - mln::morpho, 419
- min_max
 - mln::estim, 270
- min_row
 - mln::geom, 300
- min_sli
 - mln::geom, 300
- min_tree
 - mln::morpho::tree, 437
- minus
 - mln::arith, 200
 - mln::morpho, 419
- minus_cst
 - mln::arith, 201
- minus_cst_inplace
 - mln::arith, 202
- minus_infty
 - mln::point, 1119
- minus_inplace
 - mln::arith, 202
- mirror
 - mln::border, 213
- mixed_neighb
 - mln::mixed_neighb, 950
- mln, 137
 - a_point_of, 165
 - apply_p2p, 165
 - before, 177
 - bin_1complex_image2d, 161
 - bin_2complex_image3df, 161
 - box1d, 161
 - box2d, 161
 - box2d_h, 162
 - box3d, 162
 - compose, 165
 - discrete_plane_1complex_geometry, 162
 - discrete_plane_2complex_geometry, 162
 - dpoint1d, 162
 - dpoint2d, 162
 - dpoint2d_h, 162
 - dpoint3d, 162
 - duplicate, 166
 - extend, 166
 - float_2complex_image3df, 162
 - implies, 166
 - initialize, 167
 - int_u8_1complex_image2d, 163
 - int_u8_2complex_image2d, 163
 - int_u8_2complex_image3df, 163
 - is_simple_2d, 167
 - larger_than, 167
 - make_debug_graph_image, 167
 - mln_exact, 168
 - mln_gen_complex_neighborhood, 168
 - mln_gen_complex_window, 168, 169
 - mln_gen_complex_window_p, 169, 170
 - mln_regular, 170
 - mln_trait_op_geq, 170
 - mln_trait_op_greater, 170
 - mln_trait_op_leq, 171
 - mln_trait_op_neq, 171
 - operator!=, 171
 - operator<, 173
 - operator<<, 173, 174
 - operator<=, 174
 - operator*, 172
 - operator++, 172
 - operator-, 172
 - operator~, 172
 - operator==, 175, 176
 - operator|, 176, 177
 - p_run2d, 163
 - p_runs2d, 163

- point1d, 163
- point1df, 163
- point2d, 163
- point2d_h, 163
- point2df, 163
- point3d, 164
- point3df, 164
- primary, 177
- ptransform, 177
- rgb8_2complex_image3df, 164
- sagittal_dec, 177
- space_2complex_geometry, 164
- unsigned_2complex_image3df, 164
- up, 178
- vec2d_d, 164
- vec2d_f, 164
- vec3d_d, 164
- vec3d_f, 164
- w_window1d_float, 164
- w_window1d_int, 165
- w_window2d_float, 165
- w_window2d_int, 165
- w_window3d_float, 165
- w_window3d_int, 165
- mln::accu, 179
 - compute, 181
 - line, 181
 - mln_meta_accu_result, 181
 - take, 182
- mln::accu::center, 507
 - init, 508
 - is_valid, 508
 - take_as_init, 508
 - take_n_times, 508
 - to_result, 508
- mln::accu::convolve, 509
 - init, 509
 - is_valid, 509
 - take_as_init, 509
 - take_n_times, 510
 - to_result, 510
- mln::accu::count_adjacent_vertices, 511
 - init, 511
 - is_valid, 511
 - set_value, 512
 - take_as_init, 512
 - take_n_times, 512
 - to_result, 512
- mln::accu::count_labels, 513
 - init, 513
 - is_valid, 513
 - set_value, 513
 - take_as_init, 514
 - take_n_times, 514
- to_result, 514
- mln::accu::count_value, 515
 - init, 515
 - is_valid, 515
 - set_value, 515
 - take_as_init, 516
 - take_n_times, 516
 - to_result, 516
- mln::accu::histo, 517
 - is_valid, 517
 - take, 517
 - take_as_init, 517
 - take_n_times, 518
 - vect, 518
- mln::accu::image, 183
- mln::accu::impl, 184
- mln::accu::label_used, 519
 - init, 519
 - is_valid, 519
 - take, 519
 - take_as_init, 520
 - take_n_times, 520
 - to_result, 520
- mln::accu::logic, 185
- mln::accu::logic::land, 521
 - init, 521
 - is_valid, 521
 - take_as_init, 521
 - take_n_times, 522
 - to_result, 522
- mln::accu::logic::land_basic, 523
 - can_stop, 523
 - init, 523
 - is_valid, 523
 - take_as_init, 524
 - take_n_times, 524
 - to_result, 524
- mln::accu::logic::lor, 525
 - init, 525
 - is_valid, 525
 - take_as_init, 525
 - take_n_times, 526
 - to_result, 526
- mln::accu::logic::lor_basic, 527
 - can_stop, 527
 - init, 527
 - is_valid, 527
 - take_as_init, 528
 - take_n_times, 528
 - to_result, 528
- mln::accu::maj_h, 529
 - init, 529
 - is_valid, 529
 - take_as_init, 529

- take_n_times, 530
- to_result, 530
- mln::accu::math, 186
- mln::accu::math::count, 531
 - init, 531
 - is_valid, 531
 - set_value, 531
 - take_as_init, 532
 - take_n_times, 532
 - to_result, 532
- mln::accu::math::inf, 533
 - init, 533
 - is_valid, 533
 - take_as_init, 533
 - take_n_times, 534
 - to_result, 534
- mln::accu::math::sum, 535
 - init, 535
 - is_valid, 535
 - take_as_init, 536
 - take_n_times, 536
 - to_result, 536
- mln::accu::math::sup, 537
 - init, 537
 - is_valid, 537
 - take_as_init, 537
 - take_n_times, 538
 - to_result, 538
- mln::accu::max_site, 539
 - init, 539
 - is_valid, 539
 - take_as_init, 539
 - take_n_times, 540
 - to_result, 540
- mln::accu::meta::center, 541
- mln::accu::meta::count_adjacent_vertices, 542
- mln::accu::meta::count_labels, 543
- mln::accu::meta::count_value, 544
- mln::accu::meta::histo, 545
- mln::accu::meta::label_used, 546
- mln::accu::meta::logic, 187
- mln::accu::meta::logic::land, 547
- mln::accu::meta::logic::land_basic, 548
- mln::accu::meta::logic::lor, 549
- mln::accu::meta::logic::lor_basic, 550
- mln::accu::meta::maj_h, 551
- mln::accu::meta::math, 188
- mln::accu::meta::math::count, 552
- mln::accu::meta::math::inf, 553
- mln::accu::meta::math::sum, 554
- mln::accu::meta::math::sup, 555
- mln::accu::meta::max_site, 556
- mln::accu::meta::nil, 557
- mln::accu::meta::p, 558
- mln::accu::meta::pair, 559
- mln::accu::meta::rms, 560
- mln::accu::meta::shape, 189
- mln::accu::meta::shape::bbox, 561
- mln::accu::meta::shape::height, 562
- mln::accu::meta::shape::volume, 563
- mln::accu::meta::stat, 190
- mln::accu::meta::stat::max, 564
- mln::accu::meta::stat::max_h, 565
- mln::accu::meta::stat::mean, 566
- mln::accu::meta::stat::median_alt, 567
- mln::accu::meta::stat::median_h, 568
- mln::accu::meta::stat::min, 569
- mln::accu::meta::stat::min_h, 570
- mln::accu::meta::stat::rank, 571
- mln::accu::meta::stat::rank_high_quant, 572
- mln::accu::meta::tuple, 573
- mln::accu::meta::val, 574
- mln::accu::nil, 575
 - init, 575
 - is_valid, 575
 - take_as_init, 575
 - take_n_times, 576
 - to_result, 576
- mln::accu::p, 577
 - init, 577
 - is_valid, 577
 - take_as_init, 577
 - take_n_times, 578
 - to_result, 578
- mln::accu::pair, 579
 - init, 579
 - is_valid, 579
 - take_as_init, 580
 - take_n_times, 580
 - to_result, 580
- mln::accu::rms, 581
 - init, 581
 - is_valid, 581
 - take_as_init, 581
 - take_n_times, 582
 - to_result, 582
- mln::accu::shape, 191
- mln::accu::shape::bbox, 583
 - init, 583
 - is_valid, 583
 - take_as_init, 583
 - take_n_times, 584
 - to_result, 584
- mln::accu::shape::height, 585
 - argument, 586
 - init, 586
 - is_valid, 586
 - set_value, 586

- take_as_init, 586
- take_n_times, 586
- to_result, 586
- value, 586
- mln::accu::shape::volume, 588
 - argument, 589
 - init, 589
 - is_valid, 589
 - set_value, 589
 - take_as_init, 589
 - take_n_times, 589
 - to_result, 590
 - value, 589
- mln::accu::site_set::rectangularity, 591
 - area, 592
 - bbox, 592
 - rectangularity, 591
 - take_as_init, 592
 - take_n_times, 592
 - to_result, 592
- mln::accu::stat, 192
- mln::accu::stat::deviation, 593
 - init, 593
 - is_valid, 593
 - take_as_init, 594
 - take_n_times, 594
 - to_result, 594
- mln::accu::stat::max, 595
 - init, 595
 - is_valid, 595
 - set_value, 595
 - take_as_init, 596
 - take_n_times, 596
 - to_result, 596
- mln::accu::stat::max_h, 597
 - init, 597
 - is_valid, 597
 - take_as_init, 597
 - take_n_times, 598
 - to_result, 598
- mln::accu::stat::mean, 599
 - count, 599
 - init, 599
 - is_valid, 600
 - sum, 600
 - take_as_init, 600
 - take_n_times, 600
 - to_result, 600
- mln::accu::stat::median_alt, 601
 - is_valid, 601
 - take, 601
 - take_as_init, 602
 - take_n_times, 602
 - to_result, 602
- mln::accu::stat::median_h, 603
 - init, 603
 - is_valid, 603
 - take_as_init, 604
 - take_n_times, 604
 - to_result, 604
- mln::accu::stat::meta::deviation, 605
- mln::accu::stat::min, 606
 - init, 606
 - is_valid, 606
 - set_value, 606
 - take_as_init, 607
 - take_n_times, 607
 - to_result, 607
- mln::accu::stat::min_h, 608
 - init, 608
 - is_valid, 608
 - take_as_init, 608
 - take_n_times, 609
 - to_result, 609
- mln::accu::stat::min_max, 610
 - init, 611
 - is_valid, 611
 - take_as_init, 611
 - take_n_times, 611
 - to_result, 611
- mln::accu::stat::rank, 612
 - init, 612
 - is_valid, 612
 - k, 612
 - take_as_init, 613
 - take_n_times, 613
 - to_result, 613
- mln::accu::stat::rank< bool >, 614
 - init, 614
 - is_valid, 614
 - take_as_init, 614
 - take_n_times, 615
 - to_result, 615
- mln::accu::stat::rank_high_quant, 616
 - init, 616
 - is_valid, 616
 - take_as_init, 616
 - take_n_times, 617
 - to_result, 617
- mln::accu::stat::var, 618
 - init, 619
 - is_valid, 619
 - mean, 619
 - mean_t, 619
 - n_items, 619
 - take_as_init, 619
 - take_n_times, 619
 - to_result, 619

- variance, 620
- mln::accu::stat::variance, 621
 - init, 622
 - is_valid, 622
 - mean, 622
 - n_items, 622
 - standard_deviation, 622
 - sum, 622
 - take_as_init, 622
 - take_n_times, 622
 - to_result, 623
 - var, 623
- mln::accu::tuple, 624
 - init, 624
 - is_valid, 624
 - take_as_init, 624
 - take_n_times, 625
 - to_result, 625
- mln::accu::val, 626
 - init, 626
 - is_valid, 626
 - take_as_init, 626
 - take_n_times, 627
 - to_result, 627
- mln::Accumulator, 628
 - take_as_init, 628
 - take_n_times, 628
- mln::algebra, 194
 - ldlt_decomp, 194
 - ldlt_solve, 194
 - operator*, 195
 - vprod, 195
- mln::algebra::h_mat, 629
 - _1, 630
 - h_mat, 629
 - t, 630
- mln::algebra::h_vec, 631
 - h_vec, 632
 - operator mat< n, 1, U >, 632
 - origin, 632
 - t, 632
 - to_vec, 632
 - zero, 632
- mln::arith, 196
 - diff_abs, 198
 - div, 198
 - div_cst, 198
 - div_inplace, 199
 - min, 199
 - min_inplace, 200
 - minus, 200
 - minus_cst, 201
 - minus_cst_inplace, 202
 - minus_inplace, 202
 - plus, 202, 203
 - plus_cst, 203, 204
 - plus_cst_inplace, 204
 - plus_inplace, 204
 - revert, 205
 - revert_inplace, 205
 - times, 206
 - times_cst, 206
 - times_inplace, 206
- mln::arith::impl, 208
- mln::arith::impl::generic, 209
- mln::binarization, 210
 - binarization, 210
 - threshold, 210
- mln::bkd_pixter1d, 633
 - bkd_pixter1d, 633
 - image, 633
 - next, 634
- mln::bkd_pixter2d, 635
 - bkd_pixter2d, 635
 - image, 635
 - next, 636
- mln::bkd_pixter3d, 637
 - bkd_pixter3d, 637
 - image, 637
 - next, 638
- mln::border, 211
 - adjust, 211
 - duplicate, 212
 - equalize, 212
 - fill, 212
 - find, 213
 - get, 213
 - mirror, 213
 - resize, 213
- mln::border::impl, 215
- mln::border::impl::generic, 216
- mln::Box, 648
 - bbox, 649
 - diff, 650
 - inter, 650
 - is_empty, 650
 - len, 650
 - nsites, 650
 - operator<, 650, 651
 - operator<<, 651
 - operator<=, 651
 - operator==, 651
 - sym_diff, 652
 - uni, 652
 - unique, 652
- mln::box, 639
 - bbox, 643
 - bkd_piter, 642

- box, 642, 643
- center, 643
- crop_wrt, 643
- diff, 645
- element, 642
- enlarge, 643
- fwd_piter, 642
- has, 644
- inter, 645
- is_empty, 644
- is_valid, 644
- len, 644
- memory_size, 644
- nsites, 644
- operator<, 646
- operator<<, 646
- operator<=, 646, 647
- operator==, 647
- piter, 642
- pmax, 645
- pmin, 645
- psite, 642
- site, 642
- sym_diff, 647
- to_larger, 645
- uni, 647
- unique, 647
- mln::box_runend_piter, 653
 - box_runend_piter, 653
 - next, 653
 - run_length, 654
- mln::box_runstart_piter, 655
 - box_runstart_piter, 655
 - next, 655
 - run_length, 656
- mln::Browsing, 657
- mln::canvas, 217
 - distance_front, 218
 - distance_geodesic, 218
- mln::canvas::browsing, 219
- mln::canvas::browsing::backdiagonal2d_t, 658
- mln::canvas::browsing::breadth_first_search_t, 659
- mln::canvas::browsing::depth_first_search_t, 660
- mln::canvas::browsing::diagonal2d_t, 661
- mln::canvas::browsing::dir_struct_elt_incr_update_t, 662
- mln::canvas::browsing::directional_t, 664
- mln::canvas::browsing::fwd_t, 666
- mln::canvas::browsing::hyper_directional_t, 667
- mln::canvas::browsing::snake_fwd_t, 668
- mln::canvas::browsing::snake_generic_t, 669
- mln::canvas::browsing::snake_vert_t, 670
- mln::canvas::chamfer, 671
- mln::canvas::impl, 220
- mln::canvas::labeling, 221
 - blobs, 221
- mln::canvas::labeling::impl, 222
- mln::canvas::morpho, 223
- mln::category< R(*) (A) >, 672
- mln::complex_image, 673
 - complex_image, 674
 - dim, 675
 - domain, 675
 - geom, 674
 - lvalue, 674
 - operator(), 675
 - rvalue, 674
 - skeleton, 674
 - value, 674
 - values, 675
- mln::complex_neighborhood_bkd_piter, 676
 - complex_neighborhood_bkd_piter, 677
 - iter, 677
 - iter_type, 676
 - next, 677
 - psite, 676
- mln::complex_neighborhood_fwd_piter, 678
 - complex_neighborhood_fwd_piter, 679
 - iter, 679
 - iter_type, 678
 - next, 679
 - psite, 678
- mln::complex_psite, 680
 - change_target, 681
 - complex_psite, 681
 - face, 681
 - face_id, 681
 - invalidate, 681
 - is_valid, 681
 - n, 682
 - site_set, 682
- mln::complex_window_bkd_piter, 683
 - complex_window_bkd_piter, 684
 - iter, 684
 - iter_type, 683
 - next, 684
 - psite, 683
- mln::complex_window_fwd_piter, 685
 - complex_window_fwd_piter, 686
 - iter, 686
 - iter_type, 685
 - next, 686
 - psite, 685
- mln::convert, 224
 - from_to, 226
 - mln_image_from_grid, 226, 227
 - mln_window, 227
 - to, 227

- to_dpoint, 227
- to_fun, 227
- to_image, 227
- to_p_array, 227, 228
- to_p_set, 228
- to_upper_window, 229
- to_window, 229
- mln::data, 230
 - abs, 232
 - abs_inplace, 232
 - apply, 232
 - compute, 233
 - convert, 234
 - fast_median, 234
 - fill, 234
 - fill_with_image, 235
 - fill_with_value, 235
 - median, 235
 - mln_meta_accu_result, 236
 - paste, 236
 - paste_without_localization, 237
 - replace, 237
 - saturate, 237
 - saturate_inplace, 238
 - sort_offsets_increasing, 238
 - sort_psites_decreasing, 238
 - sort_psites_increasing, 238
 - stretch, 239
 - to_enc, 239
 - transform, 240
 - transform_inplace, 241
 - update, 241
 - wrap, 242
- mln::data::approx, 243
 - median, 243, 244
- mln::data::approx::impl, 245
- mln::data::impl, 246
 - stretch, 246
 - transform_inplace_lowq, 246
 - update_fastest, 247
- mln::data::impl::generic, 248
 - convert, 249
 - fill_with_image, 249
 - fill_with_value, 249
 - median, 249
 - paste, 250
 - sort_offsets_increasing, 250
 - transform, 250
 - transform_inplace, 251
 - update, 251
- mln::data::naive, 253
 - median, 253
- mln::data::naive::impl, 254
- mln::debug, 255
 - draw_graph, 256, 257
 - filename, 257
 - format, 257
 - iota, 257
 - println, 258
 - println_with_border, 258
 - put_word, 258
 - slices_2d, 258
 - superpose, 258
- mln::debug::impl, 260
- mln::decorated_image, 687
 - ~decorated_image, 688
 - decorated_image, 688
 - decoration, 688
 - lvalue, 688
 - operator decorated_image< const I, D >, 689
 - operator(), 689
 - psite, 688
 - rvalue, 688
 - skeleton, 688
- mln::def, 261
 - coord, 261
 - coordf, 261
- mln::Delta_Point_Site, 690
- mln::Delta_Point_Site< void >, 691
- mln::display, 262
- mln::display::impl, 263
- mln::display::impl::generic, 264
- mln::doc, 265
- mln::doc::Accumulator, 692
 - argument, 692
 - init, 692
 - take, 692
- mln::doc::Box, 694
 - bbox, 695
 - bkd_piter, 695
 - fwd_piter, 695
 - has, 695
 - nsites, 696
 - pmax, 696
 - pmin, 696
 - psite, 695
 - site, 695
- mln::doc::Dpoint, 697
 - coord, 697
 - dim, 698
 - dpoint, 697
 - point, 698
- mln::doc::Fastest_Image, 699
 - bbox, 703
 - bkd_piter, 701
 - border, 703
 - buffer, 703
 - coord, 701

- delta_index, 703
- domain, 703
- dpoint, 701
- fwd_piter, 701
- has, 703, 704
- is_valid, 704
- lvalue, 701
- nelements, 704
- nsites, 704
- operator(), 704, 705
- point, 701
- point_at_index, 705
- pset, 702
- psite, 702
- rvalue, 702
- skeleton, 702
- value, 702
- values, 706
- vset, 702
- mln::doc::Generalized_Pixel, 707
 - ima, 708
 - image, 707
 - rvalue, 707
 - val, 708
 - value, 708
- mln::doc::Image, 709
 - bbox, 712
 - bkd_piter, 711
 - coord, 711
 - domain, 712
 - dpoint, 711
 - fwd_piter, 711
 - has, 712, 713
 - is_valid, 713
 - lvalue, 711
 - nsites, 713
 - operator(), 713
 - point, 711
 - pset, 711
 - psite, 711
 - rvalue, 712
 - skeleton, 712
 - value, 712
 - values, 714
 - vset, 712
- mln::doc::Iterator, 715
 - invalidate, 715
 - is_valid, 715
 - start, 715
- mln::doc::Neighborhood, 717
 - bkd_niter, 717
 - dpoint, 717
 - fwd_niter, 717
 - niter, 718
 - point, 718
- mln::doc::Object, 719
- mln::doc::Pixel_Iterator, 720
 - ima, 721
 - image, 721
 - invalidate, 721
 - is_valid, 721
 - lvalue, 721
 - rvalue, 721
 - start, 721
 - val, 722
 - value, 721
- mln::doc::Point_Site
 - dim, 724
- mln::doc::Point_Site, 723
 - coord, 723
 - dpoint, 723
 - mesh, 724
 - point, 724
 - to_point, 724
- mln::doc::Site_Iterator, 726
 - invalidate, 727
 - is_valid, 727
 - operator psite, 727
 - psite, 727
 - start, 727
- mln::doc::Site_Set, 728
 - bkd_piter, 729
 - fwd_piter, 729
 - has, 729
 - psite, 729
 - site, 729
- mln::doc::Value_Iterator, 730
 - invalidate, 731
 - is_valid, 731
 - operator value, 731
 - start, 731
 - value, 731
- mln::doc::Value_Set, 732
 - bkd_viter, 733
 - fwd_viter, 733
 - has, 733
 - index_of, 733
 - nvalues, 733
 - value, 733
- mln::doc::Weighted_Window, 734
 - bkd_qiter, 735
 - delta, 735
 - dpoint, 735
 - fwd_qiter, 735
 - is_centered, 735
 - is_empty, 736
 - point, 735
 - sym, 736

- weight, 735
- win, 736
- window, 735
- mln::doc::Window, 737
 - bkd_qiter, 737
 - fwd_qiter, 737
 - qiter, 737
- mln::Dpoint, 738
 - to_dpoint, 738
- mln::dpoint, 739
 - coord, 740
 - dim, 741
 - dpoint, 741
 - grid, 740
 - operator mln::algebra::vec< dpoint< G, C
>::dim, Q >, 742
 - psite, 740
 - set_all, 742
 - site, 740
 - to_vec, 742
 - vec, 740
- mln::dpoints_bkd_pixter, 744
 - center_val, 745
 - dpoints_bkd_pixter, 745
 - invalidate, 745
 - is_valid, 745
 - next, 745
 - start, 746
 - update, 746
- mln::dpoints_fwd_pixter, 747
 - center_val, 748
 - dpoints_fwd_pixter, 748
 - invalidate, 748
 - is_valid, 748
 - next, 748
 - start, 749
 - update, 749
- mln::dpsites_bkd_piter, 750
 - dpsites_bkd_piter, 750
 - next, 751
- mln::dpsites_fwd_piter, 752
 - dpsites_fwd_piter, 752
 - next, 753
- mln::draw, 267
 - box, 267
 - line, 267
 - plot, 268
- mln::Edge, 754
- mln::edge_image, 755
 - edge_image, 756
 - edge_nbh_t, 756
 - edge_win_t, 756
 - graph_t, 756
 - nbh_t, 756
 - operator(), 757
 - site_function_t, 756
 - skeleton, 756
 - win_t, 756
- mln::estim, 269
 - mean, 269
 - min_max, 270
 - sum, 270
- mln::extended, 758
 - domain, 759
 - extended, 759
 - skeleton, 758
 - value, 758
- mln::extension, 271
 - adjust, 271, 272
 - adjust_duplicate, 272
 - adjust_fill, 272
 - duplicate, 272
 - fill, 272
- mln::extension_fun, 760
 - extension, 761
 - extension_fun, 761
 - has, 761
 - operator(), 761
 - rvalue, 761
 - skeleton, 761
 - value, 761
- mln::extension_ima, 763
 - extension, 764
 - extension_ima, 764
 - has, 764
 - operator(), 764
 - rvalue, 764
 - skeleton, 764
 - value, 764
- mln::extension_val, 766
 - change_extension, 767
 - extension, 767
 - extension_val, 767
 - has, 767
 - operator(), 767
 - rvalue, 767
 - skeleton, 767
 - value, 767
- mln::faces_psite, 769
 - change_target, 770
 - face, 770
 - face_id, 770
 - faces_psite, 770
 - invalidate, 770
 - is_valid, 770
 - n, 771
 - site_set, 771
- mln::flat_image, 772

- domain, 773
- flat_image, 773
- has, 773
- lvalue, 773
- operator(), 773
- rvalue, 773
- skeleton, 773
- value, 773
- mln::fun, 274
- mln::fun::access, 276
- mln::fun::from_accu, 775
- mln::fun::i2v, 277
 - operator<<, 277
- mln::fun::p2b, 278
- mln::fun::p2b::antilog, 776
- mln::fun::p2b::tautology, 777
- mln::fun::p2p, 279
- mln::fun::p2v, 280
- mln::fun::stat, 281
- mln::fun::v2b, 282
- mln::fun::v2b::lnot, 778
- mln::fun::v2b::threshold, 779
- mln::fun::v2i, 283
- mln::fun::v2v, 284
 - f_hsi_to_rgb_3x8, 285
 - f_hsl_to_rgb_3x8, 285
 - f_rgb_to_hsi_f, 285
 - f_rgb_to_hsl_f, 285
- mln::fun::v2v::ch_function_value, 780
- mln::fun::v2v::component, 781
- mln::fun::v2v::l1_norm, 782
- mln::fun::v2v::l2_norm, 783
- mln::fun::v2v::linear, 784
- mln::fun::v2v::linfty_norm, 785
- mln::fun::v2w2v, 286
- mln::fun::v2w2v::cos, 786
- mln::fun::v2w_w2v, 287
- mln::fun::v2w_w2v::l1_norm, 787
- mln::fun::v2w_w2v::l2_norm, 788
- mln::fun::v2w_w2v::linfty_norm, 789
- mln::fun::vv2b, 288
- mln::fun::vv2b::eq, 790
- mln::fun::vv2b::ge, 791
- mln::fun::vv2b::gt, 792
- mln::fun::vv2b::implies, 793
- mln::fun::vv2b::le, 794
- mln::fun::vv2b::lt, 795
- mln::fun::vv2v, 289
- mln::fun::vv2v::diff_abs, 796
- mln::fun::vv2v::land, 797
- mln::fun::vv2v::land_not, 798
- mln::fun::vv2v::lor, 799
- mln::fun::vv2v::lxor, 800
- mln::fun::vv2v::max, 801
- mln::fun::vv2v::min, 802
- mln::fun::vv2v::vec, 803
- mln::fun::x2p, 290
- mln::fun::x2p::closest_point, 804
- mln::fun::x2v, 291
- mln::fun::x2v::bilinear, 805
 - operator(), 805
- mln::fun::x2v::trilinear, 806
- mln::fun::x2x, 292
- mln::fun::x2x::composed, 807
 - composed, 807
- mln::fun::x2x::linear, 808
 - ima, 809
 - linear, 808
 - operator(), 808
- mln::fun::x2x::rotation, 810
 - inv, 811
 - invert, 811
 - operator(), 811
 - rotation, 811
 - set_alpha, 811
 - set_axis, 812
- mln::fun::x2x::translation, 813
 - inv, 814
 - invert, 814
 - operator(), 814
 - set_t, 814
 - t, 814
 - translation, 814
- mln::fun_image, 815
- fun_image, 816
- lvalue, 816
- operator(), 816
- rvalue, 816
- skeleton, 816
- value, 816
- mln::Function, 817
 - Function, 817
- mln::Function< void >, 818
- mln::Function_v2b, 819
- mln::Function_v2v, 820
- mln::Function_vv2b, 821
- mln::Function_vv2v, 822
- mln::fwd_pixter1d, 823
 - fwd_pixter1d, 823
 - image, 823
 - next, 824
- mln::fwd_pixter2d, 825
 - fwd_pixter2d, 825
 - image, 825
 - next, 826
- mln::fwd_pixter3d, 827
 - fwd_pixter3d, 827
 - image, 827

- next, 828
- mln::Gdpoint, 829
- mln::Gdpoint< void >, 830
- mln::Generalized_Pixel, 831
- mln::geom, 293
 - bbox, 296, 297
 - chamfer, 297
 - delta, 297
 - max_col, 297, 298
 - max_ind, 298
 - max_row, 298
 - max_sli, 298
 - mesh_corner_point_area, 298
 - mesh_curvature, 299
 - mesh_normal, 299
 - min_col, 299
 - min_ind, 300
 - min_row, 300
 - min_sli, 300
 - ncols, 300
 - ninds, 300
 - nrows, 301
 - nsites, 301
 - nslis, 301
 - pmin_pmax, 301, 302
 - rotate, 302
 - seeds2tiling, 302
 - seeds2tiling_roundness, 303
 - translate, 303
- mln::geom::complex_geometry, 832
 - add_location, 833
 - complex_geometry, 832
 - operator(), 833
- mln::geom::impl, 305
 - seeds2tiling, 305
 - seeds2tiling_roundness, 305
- mln::Gpoint, 834
 - operator<<, 837
 - operator+, 835
 - operator+=", 835
 - operator-, 836
 - operator=, 836
 - operator/, 836
 - operator==, 837
- mln::Graph, 838
- mln::graph, 307
 - compute, 307
 - labeling, 307
 - to_neighb, 308
 - to_win, 308
- mln::graph::attribute::card_t, 839
 - result, 839
- mln::graph::attribute::representative_t, 840
 - result, 840
- mln::graph_elt_mixed_neighborhood, 841
 - bkd_niter, 841
 - fwd_niter, 841
 - niter, 841
- mln::graph_elt_mixed_window, 843
 - bkd_qiter, 844
 - center_t, 844
 - delta, 845
 - fwd_qiter, 844
 - graph_element, 844
 - is_centered, 845
 - is_empty, 845
 - is_symmetric, 845
 - is_valid, 845
 - psite, 844
 - qiter, 844
 - site, 845
 - sym, 845
 - target, 845
- mln::graph_elt_neighborhood, 847
 - bkd_niter, 847
 - fwd_niter, 847
 - niter, 847
- mln::graph_elt_neighborhood_if, 849
 - bkd_niter, 849
 - fwd_niter, 849
 - graph_elt_neighborhood_if, 850
 - mask, 850
 - niter, 850
- mln::graph_elt_window, 851
 - bkd_qiter, 852
 - center_t, 852
 - delta, 853
 - fwd_qiter, 852
 - graph_element, 852
 - is_centered, 853
 - is_empty, 853
 - is_symmetric, 853
 - is_valid, 854
 - psite, 853
 - qiter, 853
 - site, 853
 - sym, 854
 - target, 853
- mln::graph_elt_window_if, 855
 - bkd_qiter, 856
 - change_mask, 858
 - delta, 858
 - fwd_qiter, 856
 - graph_elt_window_if, 857
 - is_centered, 858
 - is_empty, 858
 - is_symmetric, 858
 - is_valid, 858

- mask, 858
- mask_t, 857
- psite, 857
- qiter, 857
- site, 857
- sym, 859
- target, 857
- mln::graph_window_base, 860
 - delta, 861
 - is_centered, 861
 - is_empty, 861
 - is_symmetric, 861
 - is_valid, 861
 - site, 861
 - sym, 861
- mln::graph_window_if_piter, 862
 - element, 863
 - graph_window_if_piter, 863
 - id, 863
 - next, 863
 - P, 862
- mln::graph_window_piter, 864
 - center_t, 865
 - change_target_site_set, 866
 - element, 866
 - graph_element, 865
 - graph_window_piter, 865, 866
 - id, 866
 - next, 866
 - P, 865
 - target_site_set, 867
- mln::grid, 310
- mln::hexa, 868
 - bkd_piter, 869
 - domain, 870
 - fwd_piter, 869
 - has, 870
 - hexa, 870
 - lvalue, 869
 - operator(), 870
 - psite, 869
 - rvalue, 869
 - skeleton, 869
 - value, 870
- mln::histo, 311
 - compute, 311
- mln::histo::array, 871
- mln::histo::impl, 312
- mln::histo::impl::generic, 313
- mln::Image, 872
- mln::image1d, 875
 - bbox, 877
 - border, 877
 - buffer, 877
 - delta_index, 877
 - domain, 877
 - element, 878
 - has, 878
 - image1d, 877
 - lvalue, 876
 - nelements, 878
 - ninds, 878
 - operator(), 878
 - point_at_index, 878
 - rvalue, 876
 - skeleton, 876
 - value, 876
- mln::image2d, 880
 - bbox, 882
 - border, 882
 - buffer, 882
 - delta_index, 882
 - domain, 883
 - element, 883
 - has, 883
 - image2d, 882
 - lvalue, 881
 - ncols, 883
 - nelements, 883
 - nrows, 883
 - operator(), 883, 884
 - point_at_index, 884
 - rvalue, 881
 - skeleton, 881
 - value, 882
- mln::image2d_h, 885
 - bkd_piter, 886
 - domain, 887
 - fwd_piter, 886
 - has, 887
 - image2d_h, 887
 - lvalue, 886
 - operator(), 887
 - psite, 886
 - rvalue, 886
 - skeleton, 886
 - value, 886
- mln::image3d, 888
 - bbox, 890
 - border, 890
 - buffer, 890
 - delta_index, 891
 - domain, 891
 - element, 891
 - has, 891
 - image3d, 890
 - lvalue, 889
 - ncols, 891

- nelements, 891
- nrows, 891
- nslices, 892
- operator(), 892
- point_at_index, 892
- rvalue, 889
- skeleton, 890
- value, 890
- mln::image_if, 893
 - domain, 894
 - image_if, 893
 - operator image_if< const I, F >, 894
 - skeleton, 893
- mln::impl, 314
- mln::interpolated, 895
 - has, 896
 - interpolated, 896
 - is_valid, 896
 - lvalue, 895
 - psite, 895
 - rvalue, 896
 - skeleton, 896
 - value, 896
- mln::io, 315
- mln::io::cloud, 317
 - load, 317
 - save, 317
- mln::io::dicom, 318
 - load, 318
- mln::io::dump, 319
 - load, 319
 - save, 319
- mln::io::fits, 320
 - load, 320
- mln::io::fld, 321
 - load, 321
 - read_header, 321
 - write_header, 322
- mln::io::fld::fld_header, 897
- mln::io::magick, 323
 - do_it, 323
 - get_color, 323
 - load, 323
 - save, 324
- mln::io::off, 325
 - load, 325
 - save, 325
 - save_bin_alt, 326
- mln::io::pbm, 327
 - load, 327
 - save, 328
- mln::io::pbm::impl, 329
- mln::io::pbms, 330
 - load, 330
- mln::io::pbms::impl, 331
- mln::io::pfm, 332
 - load, 332
 - save, 333
- mln::io::pfm::impl, 334
- mln::io::pgm, 335
 - load, 335
 - save, 336
- mln::io::pgms, 337
 - load, 337
- mln::io::plot, 338
 - load, 338
 - save, 338, 339
- mln::io::pnm, 340
 - load, 340, 341
 - load_ascii_builtin, 341
 - load_ascii_value, 341
 - load_raw_2d, 341
 - max_component, 341
 - save, 341
- mln::io::pnm::impl, 342
- mln::io::pnms, 343
 - load, 343
- mln::io::ppm, 344
 - load, 344
 - save, 345
- mln::io::ppms, 346
 - load, 346
- mln::io::tiff, 347
 - load, 347
- mln::io::txt, 348
 - save, 348
- mln::Iterator, 898
 - next, 899
- mln::labeled_image, 900
 - bbox, 902
 - bbox_t, 901
 - bboxes, 902
 - labeled_image, 901, 902
 - nlabels, 902
 - relabel, 902
 - skeleton, 901
 - subdomain, 903
 - update_data, 903
- mln::labeled_image_base, 904
 - bbox, 905
 - bbox_t, 905
 - bboxes, 905
 - labeled_image_base, 905
 - nlabels, 906
 - relabel, 906
 - subdomain, 906
 - update_data, 906
- mln::labeling, 349

- background, 351
- blobs, 352
- blobs_and_compute, 352
- colorize, 353
- compute, 353–355
- compute_image, 356
- fill_holes, 357
- flat_zones, 357
- foreground, 358
- pack, 358
- pack_inplace, 358
- regional_maxima, 359
- regional_minima, 359
- relabel, 359, 360
- relabel_inplace, 360
- superpose, 361
- value, 361
- wrap, 362
- mln::labeling::impl, 363
- mln::labeling::impl::generic, 364
 - compute, 364, 365
- mln::lazy_image, 907
 - domain, 908
 - has, 908
 - lazy_image, 908
 - lvalue, 908
 - operator(), 909
 - rvalue, 908
 - skeleton, 908
- mln::linear, 366
 - gaussian, 367
 - gaussian_1st_derivative, 367
 - gaussian_2nd_derivative, 368
 - mln_ch_convolve, 368
 - mln_ch_convolve_grad, 369
- mln::linear::impl, 370
- mln::linear::local, 371
 - convolve, 371
- mln::linear::local::impl, 372
- mln::Literal, 910
- mln::literal, 373
 - black, 376
 - blue, 376
 - brown, 376
 - cyan, 376
 - dark_gray, 376
 - green, 376
 - identity, 376
 - light_gray, 376
 - lime, 376
 - magenta, 377
 - max, 377
 - medium_gray, 377
 - min, 377
 - olive, 377
 - one, 377
 - orange, 377
 - origin, 377
 - pink, 377
 - purple, 377
 - red, 377
 - teal, 378
 - violet, 378
 - white, 378
 - yellow, 378
 - zero, 378
- mln::literal::black_t, 913
- mln::literal::blue_t, 914
- mln::literal::brown_t, 915
- mln::literal::cyan_t, 916
- mln::literal::green_t, 917
- mln::literal::identity_t, 918
- mln::literal::light_gray_t, 919
- mln::literal::lime_t, 920
- mln::literal::magenta_t, 921
- mln::literal::max_t, 922
- mln::literal::min_t, 923
- mln::literal::olive_t, 924
- mln::literal::one_t, 925
- mln::literal::orange_t, 926
- mln::literal::origin_t, 927
- mln::literal::pink_t, 928
- mln::literal::purple_t, 929
- mln::literal::red_t, 930
- mln::literal::teal_t, 931
- mln::literal::violet_t, 932
- mln::literal::white_t, 933
- mln::literal::yellow_t, 934
- mln::literal::zero_t, 935
- mln::logical, 379
 - and_inplace, 379
 - and_not, 379
 - and_not_inplace, 380
 - not_inplace, 380
 - or_inplace, 381
 - xor_inplace, 381
- mln::logical::impl, 382
- mln::logical::impl::generic, 383
- mln::make, 384
 - attachment, 389
 - box1d, 389
 - box2d, 390
 - box2d_h, 390, 391
 - box3d, 391, 392
 - cell, 392
 - couple, 392
 - detachment, 393
 - dpoint2d_h, 393

- dummy_p_edges, 393, 394
- dummy_p_vertices, 394
- edge_image, 394–396
- h_mat, 396
- image, 396, 397
- image2d, 397
- image3d, 397, 398
- influence_zone_adjacency_graph, 398
- mat, 398
- ord_pair, 399
- p_edges_with_mass_centers, 399
- p_vertices_with_mass_centers, 399
- pix, 399
- pixel, 400
- point2d_h, 400
- rag_and_labeled_wsl, 400
- region_adjacency_graph, 401
- relabelfun, 401, 402
- vec, 402, 403
- vertex_image, 403, 404
- voronoi, 404
- w_window, 404
- w_window1d, 405
- w_window1d_int, 405
- w_window2d, 405
- w_window2d_int, 406
- w_window3d, 406
- w_window3d_int, 406
- w_window_directional, 407
- mln::math, 408
 - abs, 408
- mln::Mesh, 936
- mln::Meta_Accumulator, 937
- mln::Meta_Function, 938
- mln::Meta_Function_v2v, 939
- mln::Meta_Function_vv2v, 940
- mln::metal, 409
 - ands, 941
 - converts_to, 942
 - equal, 943
 - goes_to, 944
 - impl, 410
 - is, 945
 - is_a, 946
 - is_not, 947
 - is_not_a, 948
 - math, 411
 - math::impl, 412
 - mixed_neighb, 949
 - bkd_niter, 949
 - fwd_niter, 949
 - mixed_neighb, 950
 - niter, 949
- mln::morpho, 413
 - complementation, 416
 - complementation_inplace, 416
 - contrast, 416
 - dilation, 416
 - erosion, 416
 - general, 416
 - gradient, 417
 - gradient_external, 417
 - gradient_internal, 417
 - hit_or_miss, 417
 - hit_or_miss_background_closing, 417
 - hit_or_miss_background_opening, 418
 - hit_or_miss_closing, 418
 - hit_or_miss_opening, 418
 - laplacian, 418
 - line_gradient, 418
 - meyer_wst, 418, 419
 - min, 419
 - min_inplace, 419
 - minus, 419
 - plus, 420
 - rank_filter, 420
 - thick_miss, 420
 - thickening, 420
 - thin_fit, 420
 - thinning, 421
 - top_hat_black, 421
 - top_hat_self_complementary, 421
 - top_hat_white, 421
- mln::morpho::approx, 422
- mln::morpho::attribute, 423
 - mln::morpho::attribute::card, 951
 - init, 951
 - is_valid, 951
 - take_as_init, 951
 - take_n_times, 952
 - to_result, 952
 - mln::morpho::attribute::count_adjacent_vertices, 953
 - init, 953
 - is_valid, 953
 - take_as_init, 953
 - take_n_times, 954
 - to_result, 954
 - mln::morpho::attribute::height, 955
 - base_level, 955
 - init, 955
 - is_valid, 955
 - take_as_init, 956
 - take_n_times, 956
 - to_result, 956
 - mln::morpho::attribute::sharpness, 957
 - area, 958
 - height, 958

- init, 958
- is_valid, 958
- take_as_init, 958
- take_n_times, 958
- to_result, 958
- volume, 958
- mln::morpho::attribute::sum, 960
 - init, 960
 - is_valid, 960
 - set_value, 961
 - take_as_init, 961
 - take_n_times, 961
 - to_result, 961
 - untake, 961
- mln::morpho::attribute::volume, 962
 - area, 962
 - init, 962
 - is_valid, 963
 - take_as_init, 963
 - take_n_times, 963
 - to_result, 963
- mln::morpho::closing::approx, 424
 - structural, 424
- mln::morpho::elementary, 425
 - closing, 425
 - mln_trait_op_minus_twice, 426
 - opening, 426
 - top_hat_black, 426
 - top_hat_self_complementary, 426
 - top_hat_white, 426
- mln::morpho::impl, 427
- mln::morpho::impl::generic, 428
 - hit_or_miss, 428
 - rank_filter, 428
- mln::morpho::opening::approx, 429
 - structural, 429
- mln::morpho::reconstruction, 430
- mln::morpho::reconstruction::by_dilation, 431
- mln::morpho::reconstruction::by_erosion, 432
- mln::morpho::tree, 433
 - compute_attribute_image, 434
 - compute_attribute_image_from, 434
 - compute_parent, 435
 - dual_input_max_tree, 436
 - max_tree, 436
 - min_tree, 437
 - propagate_if, 437
 - propagate_if_value, 437
 - propagate_node_to_ancestors, 438
 - propagate_node_to_descendants, 438
 - propagate_representative, 439
- mln::morpho::tree::filter, 440
 - direct, 440
 - filter, 440
- max, 441
- min, 441
- subtractive, 441
- mln::morpho::watershed, 443
 - flooding, 443, 444
 - superpose, 444
 - topological, 444
- mln::morpho::watershed::watershed, 446
- mln::morpho::watershed::watershed::generic, 447
- mln::neighb, 964
 - bkd_niter, 965
 - fwd_niter, 965
 - neighb, 965
 - niter, 965
- mln::Neighborhood, 966
- mln::Neighborhood< void >, 967
- mln::norm, 448
 - l1, 449
 - l1_distance, 449
 - l2, 449
 - l2_distance, 449
 - linfty, 449
 - linfty_distance, 449
 - sqr_l2, 449
- mln::norm::impl, 450
- mln::Object, 968
- mln::opt, 451
 - at, 451, 452
- mln::opt::impl, 453
- mln::p2p_image, 969
 - domain, 970
 - fun, 970
 - operator(), 970
 - p2p_image, 970
 - skeleton, 969
- mln::p_array, 971
 - append, 974
 - bkd_piter, 973
 - change, 974
 - clear, 974
 - diff, 976
 - element, 973
 - fwd_piter, 973
 - has, 974, 975
 - i_element, 973
 - insert, 975
 - inter, 976
 - is_valid, 975
 - memory_size, 975
 - nsites, 975
 - operator<, 976
 - operator<<, 976
 - operator<=, 977
 - operator==, 977

- p_array, 974
- piter, 974
- psite, 974
- reserve, 976
- resize, 976
- std_vector, 976
- sym_diff, 977
- uni, 977
- unique, 977
- mln::p_centered, 978
 - bkd_piter, 980
 - center, 980
 - diff, 981
 - element, 980
 - fwd_piter, 980
 - has, 980
 - inter, 981
 - is_valid, 981
 - memory_size, 981
 - operator<, 981
 - operator<<, 981
 - operator<=, 982
 - operator==, 982
 - p_centered, 980
 - piter, 980
 - psite, 980
 - site, 980
 - sym_diff, 982
 - uni, 982
 - unique, 982
 - window, 981
- mln::p_complex, 983
 - bkd_piter, 985
 - cplx, 986
 - diff, 987
 - element, 985
 - fwd_piter, 985
 - geom, 986
 - has, 986
 - inter, 987
 - is_valid, 986
 - nfaces, 986
 - nfaces_of_dim, 986
 - nsites, 986
 - operator<, 987
 - operator<<, 987
 - operator<=, 987
 - operator==, 988
 - p_complex, 985
 - piter, 985
 - psite, 985
 - sym_diff, 988
 - uni, 988
 - unique, 988
- mln::p_edges, 989
 - bkd_piter, 991
 - diff, 994
 - edge, 991
 - element, 991
 - fun_t, 991
 - function, 993
 - fwd_piter, 992
 - graph, 993
 - graph_element, 992
 - graph_t, 992
 - has, 993
 - inter, 994
 - invalidate, 994
 - is_valid, 994
 - memory_size, 994
 - nedges, 994
 - nsites, 994
 - operator<, 994
 - operator<<, 995
 - operator<=, 995
 - operator==, 995
 - p_edges, 992, 993
 - piter, 992
 - psite, 992
 - sym_diff, 995
 - uni, 995
 - unique, 996
- mln::p_faces, 997
 - bkd_piter, 999
 - cplx, 1000
 - diff, 1000
 - element, 999
 - fwd_piter, 999
 - inter, 1000
 - is_valid, 1000
 - nfaces, 1000
 - nsites, 1000
 - operator<, 1001
 - operator<<, 1001
 - operator<=, 1001
 - operator==, 1001
 - p_faces, 999
 - piter, 999
 - psite, 999
 - sym_diff, 1001
 - uni, 1002
 - unique, 1002
- mln::p_graph_piter, 1003
 - graph, 1003
 - id, 1003
 - mln_q_subject, 1004
 - next, 1004
 - p_graph_piter, 1003

- mln::p_if, 1005
 - bkd_piter, 1007
 - diff, 1008
 - element, 1007
 - fwd_piter, 1007
 - has, 1007
 - inter, 1008
 - is_valid, 1007
 - memory_size, 1008
 - operator<, 1008
 - operator<<, 1008
 - operator<=, 1009
 - operator==, 1009
 - overset, 1008
 - p_if, 1007
 - piter, 1007
 - pred, 1008
 - predicate, 1008
 - psite, 1007
 - sym_diff, 1009
 - uni, 1009
 - unique, 1009
- mln::p_image, 1010
 - bkd_piter, 1012
 - clear, 1013
 - diff, 1014
 - element, 1012
 - fwd_piter, 1012
 - has, 1013
 - i_element, 1012
 - insert, 1013
 - inter, 1014
 - is_valid, 1013
 - memory_size, 1013
 - nsites, 1014
 - operator typename internal::p_image_site_-
set< I >::ret, 1014
 - operator<, 1014
 - operator<<, 1014
 - operator<=, 1015
 - operator==, 1015
 - p_image, 1013
 - piter, 1012
 - psite, 1012
 - r_element, 1012
 - remove, 1014
 - S, 1012
 - sym_diff, 1015
 - toggle, 1014
 - uni, 1015
 - unique, 1015
- mln::p_indexed_bkd_piter, 1016
 - index, 1016
 - next, 1016
 - p_indexed_bkd_piter, 1016
- mln::p_indexed_fwd_piter, 1018
 - index, 1018
 - next, 1018
 - p_indexed_fwd_piter, 1018
- mln::p_indexed_psite, 1020
- mln::p_key, 1021
 - bkd_piter, 1023
 - change_key, 1024
 - change_keys, 1024
 - clear, 1024
 - diff, 1026
 - element, 1023
 - exists_key, 1024
 - fwd_piter, 1023
 - has, 1025
 - i_element, 1024
 - insert, 1025
 - inter, 1026
 - is_valid, 1025
 - key, 1025
 - keys, 1025
 - memory_size, 1025
 - nsites, 1025
 - operator<, 1026
 - operator<<, 1026
 - operator<=, 1027
 - operator(), 1026
 - operator==, 1027
 - p_key, 1024
 - piter, 1024
 - psite, 1024
 - r_element, 1024
 - remove, 1026
 - remove_key, 1026
 - sym_diff, 1027
 - uni, 1027
 - unique, 1027
- mln::p_line2d, 1028
 - bbox, 1031
 - begin, 1031
 - bkd_piter, 1030
 - diff, 1032
 - element, 1030
 - end, 1031
 - fwd_piter, 1030
 - has, 1031
 - inter, 1032
 - is_valid, 1031
 - memory_size, 1031
 - nsites, 1031
 - operator<, 1032
 - operator<<, 1032
 - operator<=, 1033

- operator==, 1033
- p_line2d, 1030
- piter, 1030
- psite, 1030
- q_box, 1030
- std_vector, 1032
- sym_diff, 1033
- uni, 1033
- unique, 1033
- mln::p_mutable_array_of, 1034
 - bkd_piter, 1036
 - clear, 1037
 - diff, 1038
 - element, 1036
 - fwd_piter, 1036
 - has, 1037
 - i_element, 1036
 - insert, 1037
 - inter, 1038
 - is_valid, 1037
 - memory_size, 1037
 - nelements, 1037
 - operator<, 1038
 - operator<<, 1038
 - operator<=, 1038
 - operator==, 1038
 - p_mutable_array_of, 1036
 - piter, 1036
 - psite, 1036
 - reserve, 1037
 - sym_diff, 1039
 - uni, 1039
 - unique, 1039
- mln::p_n_faces_bkd_piter, 1040
 - n, 1040
 - next, 1040
 - p_n_faces_bkd_piter, 1040
- mln::p_n_faces_fwd_piter, 1042
 - n, 1042
 - next, 1042
 - p_n_faces_fwd_piter, 1042
- mln::p_priority, 1044
 - bkd_piter, 1046
 - clear, 1047
 - diff, 1050
 - element, 1046
 - exists_priority, 1047
 - front, 1047
 - fwd_piter, 1046
 - has, 1048
 - highest_priority, 1048
 - i_element, 1047
 - insert, 1048
 - inter, 1050
 - is_valid, 1048
 - lowest_priority, 1048
 - memory_size, 1048
 - nsites, 1049
 - operator<, 1050
 - operator<<, 1050
 - operator<=, 1050
 - operator(), 1049
 - operator==, 1051
 - p_priority, 1047
 - piter, 1047
 - pop, 1049
 - pop_front, 1049
 - priorities, 1049
 - psite, 1047
 - push, 1049
 - sym_diff, 1051
 - uni, 1051
 - unique, 1051
- mln::p_queue, 1052
 - bkd_piter, 1054
 - clear, 1055
 - diff, 1056
 - element, 1054
 - front, 1055
 - fwd_piter, 1054
 - has, 1055
 - i_element, 1054
 - insert, 1055
 - inter, 1056
 - is_valid, 1055
 - memory_size, 1055
 - nsites, 1056
 - operator<, 1057
 - operator<<, 1057
 - operator<=, 1057
 - operator==, 1057
 - p_queue, 1055
 - piter, 1054
 - pop, 1056
 - pop_front, 1056
 - psite, 1054
 - push, 1056
 - std_deque, 1056
 - sym_diff, 1057
 - uni, 1058
 - unique, 1058
- mln::p_queue_fast, 1059
 - bkd_piter, 1061
 - clear, 1062
 - compute_has, 1062
 - diff, 1064
 - element, 1061
 - empty, 1062

- front, 1062
- fwd_piter, 1061
- has, 1062, 1063
- i_element, 1062
- insert, 1063
- inter, 1064
- is_valid, 1063
- memory_size, 1063
- nsites, 1063
- operator<, 1064
- operator<<, 1064
- operator<=, 1065
- operator==, 1065
- p_queue_fast, 1062
- piter, 1062
- pop, 1063
- pop_front, 1063
- psite, 1062
- purge, 1064
- push, 1064
- reserve, 1064
- std_vector, 1064
- sym_diff, 1065
- uni, 1065
- unique, 1065
- mln::p_run, 1066
 - bbox, 1069
 - bkd_piter, 1068
 - diff, 1071
 - element, 1068
 - end, 1069
 - fwd_piter, 1068
 - has, 1069
 - has_index, 1069
 - init, 1070
 - inter, 1071
 - is_valid, 1070
 - length, 1070
 - memory_size, 1070
 - nsites, 1070
 - operator<, 1071
 - operator<<, 1071
 - operator<=, 1071
 - operator==, 1071
 - p_run, 1069
 - piter, 1068
 - psite, 1068
 - q_box, 1068
 - start, 1070
 - sym_diff, 1072
 - uni, 1072
 - unique, 1072
- mln::p_set, 1073
 - bkd_piter, 1075
 - clear, 1076
 - diff, 1077
 - element, 1075
 - fwd_piter, 1075
 - has, 1076
 - i_element, 1075
 - insert, 1076
 - inter, 1077
 - is_valid, 1076
 - memory_size, 1076
 - nsites, 1077
 - operator<, 1077
 - operator<<, 1078
 - operator<=, 1078
 - operator==, 1078
 - p_set, 1076
 - piter, 1075
 - psite, 1075
 - r_element, 1076
 - remove, 1077
 - std_vector, 1077
 - sym_diff, 1078
 - uni, 1078
 - unique, 1078
 - util_set, 1077
- mln::p_set_of, 1080
 - bkd_piter, 1082
 - clear, 1082
 - diff, 1083
 - element, 1082
 - fwd_piter, 1082
 - has, 1082
 - i_element, 1082
 - insert, 1083
 - inter, 1083
 - is_valid, 1083
 - memory_size, 1083
 - nelements, 1083
 - operator<, 1083
 - operator<<, 1083
 - operator<=, 1084
 - operator==, 1084
 - p_set_of, 1082
 - piter, 1082
 - psite, 1082
 - sym_diff, 1084
 - uni, 1084
 - unique, 1084
- mln::p_transformed, 1085
 - bkd_piter, 1087
 - diff, 1088
 - element, 1087
 - function, 1087
 - fwd_piter, 1087

- has, 1087
- inter, 1088
- is_valid, 1088
- memory_size, 1088
- operator<, 1088
- operator<<, 1088
- operator<=, 1089
- operator==, 1089
- p_transformed, 1087
- piter, 1087
- primary_set, 1088
- psite, 1087
- sym_diff, 1089
- uni, 1089
- unique, 1089
- mln::p_transformed_piter, 1090
 - change_target, 1091
 - next, 1091
 - p_transformed_piter, 1090
- mln::p_vaccess, 1092
 - bkd_piter, 1094
 - diff, 1096
 - element, 1094
 - fwd_piter, 1094
 - has, 1095
 - i_element, 1094
 - insert, 1095
 - inter, 1096
 - is_valid, 1095
 - memory_size, 1095
 - operator<, 1096
 - operator<<, 1096
 - operator<=, 1096
 - operator(), 1096
 - operator==, 1097
 - p_vaccess, 1095
 - piter, 1094
 - pset, 1094
 - psite, 1094
 - sym_diff, 1097
 - uni, 1097
 - unique, 1097
 - value, 1095
 - values, 1096
 - vset, 1095
- mln::p_vertices, 1098
 - bkd_piter, 1100
 - diff, 1104
 - element, 1100
 - fun_t, 1101
 - function, 1102
 - fwd_piter, 1101
 - graph, 1102
 - graph_element, 1101
 - graph_t, 1101
 - has, 1103
 - inter, 1104
 - invalidate, 1103
 - is_valid, 1103
 - memory_size, 1103
 - nsites, 1103
 - nvertices, 1103
 - operator<, 1104
 - operator<<, 1104
 - operator<=, 1104
 - operator(), 1104
 - operator==, 1105
 - p_vertices, 1101, 1102
 - piter, 1101
 - psite, 1101
 - sym_diff, 1105
 - uni, 1105
 - unique, 1105
 - vertex, 1101
- mln::pixel, 1106
 - change_to, 1107
 - is_valid, 1107
 - pixel, 1106
- mln::Pixel_Iterator, 1108
 - next, 1108
- mln::plain, 1110
 - operator I, 1111
 - operator=, 1111
 - plain, 1111
 - skeleton, 1110
- mln::Point, 1112
 - operator+=", 1113
 - operator=, 1113
 - operator/, 1114
 - point, 1113
 - to_point, 1113
- mln::point, 1115
 - coord, 1118
 - delta, 1118
 - dim, 1118
 - dpsite, 1118
 - grid, 1118
 - h_vec, 1118
 - last_coord, 1119
 - minus_infty, 1119
 - operator<<, 1123
 - operator+, 1121
 - operator+=", 1119, 1121
 - operator-, 1121
 - operator=, 1120, 1122
 - operator/, 1122
 - operator==, 1123
 - origin, 1123

- plus_infty, 1120
- point, 1118, 1119
- set_all, 1120
- to_h_vec, 1120
- to_vec, 1121
- vec, 1118
- mln::Point_Site, 1124
 - operator<<, 1126
 - operator+, 1125
 - operator-, 1125
 - operator==, 1126
- mln::Point_Site< void >, 1128
- mln::Proxy, 1129
- mln::Proxy< void >, 1130
- mln::Pseudo_Site, 1131
- mln::Pseudo_Site< void >, 1132
- mln::pw, 454
- mln::pw::image, 1133
 - image, 1133
 - skeleton, 1133
- mln::registration, 455
 - get_rot, 456
 - icp, 456
 - registration1, 457
 - registration2, 457
 - registration3, 457
- mln::registration::closest_point_basic, 1134
- mln::registration::closest_point_with_map, 1135
- mln::Regular_Grid, 1136
- mln::safe_image, 1137
 - operator safe_image< const I >, 1137
 - skeleton, 1137
- mln::select, 458
- mln::select::p_of, 1138
- mln::set, 459
 - card, 459
 - compute, 459
 - compute_with_weights, 460
 - get, 461
 - has, 461
 - mln_meta_accu_result, 461
- mln::Site, 1139
- mln::Site< void >, 1140
- mln::Site_Iterator, 1141
 - next, 1142
- mln::Site_Proxy, 1143
- mln::Site_Proxy< void >, 1144
- mln::Site_Set, 1145
 - diff, 1146
 - inter, 1146
 - operator<, 1146
 - operator<<, 1147
 - operator<=, 1147
 - operator==, 1147
 - sym_diff, 1147
 - uni, 1147
 - unique, 1148
- mln::Site_Set< void >, 1149
- mln::slice_image, 1150
 - domain, 1151
 - operator slice_image< const I >, 1151
 - operator(), 1151
 - skeleton, 1150
 - sli, 1151
 - slice_image, 1151
- mln::sub_image, 1152
 - domain, 1153
 - operator sub_image< const I, S >, 1153
 - skeleton, 1152
 - sub_image, 1152
- mln::sub_image_if, 1154
 - domain, 1155
 - skeleton, 1154
 - sub_image_if, 1154
- mln::subsampling, 462
 - gaussian_subsampling, 462
 - subsampling, 462
- mln::tag, 463
- mln::test, 464
 - positive, 464
 - predicate, 464, 465
- mln::test::impl, 466
- mln::thru_image, 1156
 - operator thru_image< const I, F >, 1156
- mln::thrubin_image, 1157
 - operator thrubin_image< const I1, const I2, F >, 1158
 - psite, 1157
 - rvalue, 1157
 - skeleton, 1157
 - value, 1158
- mln::topo, 467
 - detach, 471
 - edge, 471
 - is_facet, 472
 - make_algebraic_face, 472
 - make_algebraic_n_face, 472
 - operator!=, 472, 473
 - operator<, 473, 474
 - operator<<, 474, 475
 - operator+, 473
 - operator-, 473
 - operator==, 475
- mln::topo::adj_higher_dim_connected_n_face_-
 - bkd_iter, 1159
 - adj_higher_dim_connected_n_face_bkd_iter, 1159
 - next, 1159

- mln::topo::adj_higher_dim_connected_n_face_-
fwd_iter, 1161
- adj_higher_dim_connected_n_face_fwd_iter,
1161
- next, 1161
- mln::topo::adj_higher_face_bkd_iter, 1163
- adj_higher_face_bkd_iter, 1163
- next, 1163
- mln::topo::adj_higher_face_fwd_iter, 1164
- adj_higher_face_fwd_iter, 1164
- next, 1164
- mln::topo::adj_lower_dim_connected_n_face_-
bkd_iter, 1165
- adj_lower_dim_connected_n_face_bkd_iter,
1165
- next, 1165
- mln::topo::adj_lower_dim_connected_n_face_-
fwd_iter, 1167
- adj_lower_dim_connected_n_face_fwd_iter,
1167
- next, 1167
- mln::topo::adj_lower_face_bkd_iter, 1169
- adj_lower_face_bkd_iter, 1169
- next, 1169
- mln::topo::adj_lower_face_fwd_iter, 1170
- adj_lower_face_fwd_iter, 1170
- next, 1170
- mln::topo::adj_lower_higher_face_bkd_iter, 1171
- adj_lower_higher_face_bkd_iter, 1171
- next, 1171
- mln::topo::adj_lower_higher_face_fwd_iter, 1172
- adj_lower_higher_face_fwd_iter, 1172
- next, 1172
- mln::topo::adj_m_face_bkd_iter, 1173
- adj_m_face_bkd_iter, 1173
- next, 1174
- mln::topo::adj_m_face_fwd_iter, 1175
- adj_m_face_fwd_iter, 1175
- next, 1176
- mln::topo::algebraic_face, 1177
- algebraic_face, 1178, 1179
- cplx, 1179
- data, 1179
- dec_face_id, 1179
- dec_n, 1179
- face_id, 1179
- higher_dim_adj_faces, 1179
- inc_face_id, 1180
- inc_n, 1180
- invalidate, 1180
- is_valid, 1180
- lower_dim_adj_faces, 1180
- n, 1180
- set_cplx, 1180
- set_face_id, 1180
- set_n, 1181
- set_sign, 1181
- sign, 1181
- mln::topo::algebraic_n_face, 1182
- algebraic_n_face, 1183
- cplx, 1184
- data, 1184
- dec_face_id, 1184
- face_id, 1184
- higher_dim_adj_faces, 1184
- inc_face_id, 1184
- invalidate, 1184
- is_valid, 1184
- lower_dim_adj_faces, 1185
- n, 1185
- set_cplx, 1185
- set_face_id, 1185
- set_sign, 1185
- sign, 1185
- mln::topo::center_only_iter, 1186
- center_only_iter, 1186
- next, 1187
- mln::topo::centered_bkd_iter_adapter, 1188
- centered_bkd_iter_adapter, 1188
- next, 1188
- mln::topo::centered_fwd_iter_adapter, 1189
- centered_fwd_iter_adapter, 1189
- next, 1189
- mln::topo::complex, 1190
- add_face, 1191
- addr, 1191
- bkd_citer, 1191
- complex, 1191
- fwd_citer, 1191
- nfaces, 1191
- nfaces_of_dim, 1192
- nfaces_of_static_dim, 1192
- print, 1192
- print_faces, 1192
- mln::topo::face, 1193
- cplx, 1194
- data, 1195
- dec_face_id, 1195
- dec_n, 1195
- face, 1194
- face_id, 1195
- higher_dim_adj_faces, 1195
- inc_face_id, 1195
- inc_n, 1195
- invalidate, 1195
- is_valid, 1195
- lower_dim_adj_faces, 1195
- n, 1196

- set_cplx, 1196
 - set_face_id, 1196
 - set_n, 1196
- mln::topo::face_bkd_iter, 1197
 - face_bkd_iter, 1197
 - next, 1197
 - start, 1197
- mln::topo::face_fwd_iter, 1199
 - face_fwd_iter, 1199
 - next, 1199
 - start, 1199
- mln::topo::is_n_face, 1201
- mln::topo::is_simple_cell, 1202
 - D, 1203
 - mln_geom, 1203
 - operator(), 1203
 - psite, 1203
 - result, 1203
 - set_image, 1203
- mln::topo::n_face, 1204
 - cplx, 1205
 - data, 1205
 - dec_face_id, 1205
 - face_id, 1205
 - higher_dim_adj_faces, 1206
 - inc_face_id, 1206
 - invalidate, 1206
 - is_valid, 1206
 - lower_dim_adj_faces, 1206
 - n, 1206
 - n_face, 1205
 - set_cplx, 1206
 - set_face_id, 1207
- mln::topo::n_face_bkd_iter, 1208
 - n, 1208
 - n_face_bkd_iter, 1208
 - next, 1209
 - start, 1209
- mln::topo::n_face_fwd_iter, 1210
 - n, 1210
 - n_face_fwd_iter, 1210
 - next, 1210
 - start, 1211
- mln::topo::n_faces_set, 1212
 - add, 1212
 - faces, 1212
 - faces_type, 1212
 - reserve, 1213
- mln::topo::static_n_face_bkd_iter, 1214
 - next, 1214
 - start, 1215
 - static_n_face_bkd_iter, 1214
- mln::topo::static_n_face_fwd_iter, 1216
 - next, 1216
- start, 1217
 - static_n_face_fwd_iter, 1216
- mln::tr_image, 1218
 - domain, 1220
 - has, 1220
 - is_valid, 1220
 - lvalue, 1219
 - operator(), 1220
 - psite, 1219
 - rvalue, 1219
 - set_tr, 1220
 - site, 1219
 - skeleton, 1219
 - tr, 1220
 - tr_image, 1219
 - value, 1219
- mln::trace, 477
- mln::trait, 478
- mln::transform, 479
 - distance_and_closest_point_geodesic, 480
 - distance_and_influence_zone_geodesic, 481
 - distance_front, 481
 - distance_geodesic, 481
 - hough, 481
 - influence_zone_front, 482
 - influence_zone_geodesic, 482
 - influence_zone_geodesic_saturated, 482
- mln::transformed_image, 1221
 - domain, 1222
 - operator transformed_image< const I, F >, 1222
 - operator(), 1222
 - skeleton, 1221
 - transformed_image, 1222
- mln::unproject_image, 1223
 - domain, 1223
 - operator(), 1223, 1224
 - unproject_image, 1223
- mln::util, 484
 - display_branch, 487
 - display_tree, 487
 - lemmings, 488
 - make_greater_point, 488
 - make_greater_psite, 488
 - operator<, 488
 - operator<<, 488, 489
 - operator==, 489
 - ord_strict, 489
 - ord_weak, 489
 - tree_fast_to_image, 489
 - tree_to_fast, 490
 - tree_to_image, 490
 - vertex_id_t, 487
- mln::util::adjacency_matrix, 1225

- adjacency_matrix, 1225
- mln::util::array, 1226
 - append, 1229
 - array, 1228
 - bkd_eiter, 1228
 - clear, 1229
 - eiter, 1228
 - element, 1228
 - fill, 1229
 - fwd_eiter, 1228
 - is_empty, 1229
 - memory_size, 1229
 - nelements, 1230
 - operator(), 1230
 - reserve, 1230
 - resize, 1231
 - result, 1228
 - size, 1231
 - std_vector, 1231
- mln::util::branch, 1232
 - apex, 1232
 - branch, 1232
 - util_tree, 1232
- mln::util::branch_iter, 1234
 - deepness, 1234
 - invalidate, 1234
 - is_valid, 1234
 - next, 1235
 - operator util::tree_node< T > &, 1235
 - start, 1235
- mln::util::branch_iter_ind, 1236
 - deepness, 1236
 - invalidate, 1236
 - is_valid, 1236
 - next, 1237
 - operator util::tree_node< T > &, 1237
 - start, 1237
- mln::util::couple, 1238
 - change_both, 1238
 - change_first, 1238
 - change_second, 1238
 - first, 1239
 - second, 1239
- mln::util::eat, 1240
- mln::util::edge, 1241
 - category, 1242
 - change_graph, 1243
 - edge, 1242
 - graph, 1243
 - graph_t, 1242
 - id, 1243
 - id_t, 1242
 - id_value_t, 1242
 - invalidate, 1243
 - is_valid, 1243
 - ith_nbh_edge, 1243
 - nmax_nbh_edges, 1243
 - operator edge_id_t, 1243
 - update_id, 1243
 - v1, 1244
 - v2, 1244
 - v_other, 1244
- mln::util::fibonacci_heap, 1245
 - clear, 1246
 - fibonacci_heap, 1246
 - front, 1246
 - is_empty, 1246
 - is_valid, 1246
 - nelements, 1246
 - operator=, 1247
 - pop_front, 1247
 - push, 1247
- mln::util::graph, 1248
 - add_edge, 1251
 - add_vertex, 1251
 - add_vertices, 1251
 - e_ith_nbh_edge, 1252
 - e_nmax, 1252
 - e_nmax_nbh_edges, 1252
 - edge, 1252
 - edge_fwd_iter, 1250
 - edge_nbh_edge_fwd_iter, 1250
 - edges, 1252
 - edges_set_t, 1250
 - edges_t, 1250
 - graph, 1251
 - has_e, 1252
 - has_v, 1252
 - is_subgraph_of, 1253
 - v1, 1253
 - v2, 1253
 - v_ith_nbh_edge, 1253
 - v_ith_nbh_vertex, 1253
 - v_nmax, 1253
 - v_nmax_nbh_edges, 1253
 - v_nmax_nbh_vertices, 1254
 - vertex, 1254
 - vertex_fwd_iter, 1250
 - vertex_nbh_edge_fwd_iter, 1250
 - vertex_nbh_vertex_fwd_iter, 1250
 - vertices_t, 1250
- mln::util::greater_point, 1255
 - operator(), 1255
- mln::util::greater_psite, 1256
 - operator(), 1256
- mln::util::head, 1257
- mln::util::ignore, 1258
- mln::util::ilcell, 1259

- mln::util::impl, 491
 - tree_fast_to_image, 491
- mln::util::line_graph, 1260
 - e_ith_nbh_edge, 1263
 - e_nmax, 1263
 - e_nmax_nbh_edges, 1263
 - edge, 1263
 - edge_fwd_iter, 1262
 - edge_nbh_edge_fwd_iter, 1262
 - edges_t, 1262
 - graph, 1263
 - has, 1263
 - has_e, 1264
 - has_v, 1264
 - is_subgraph_of, 1264
 - v1, 1264
 - v2, 1264
 - v_ith_nbh_edge, 1264
 - v_ith_nbh_vertex, 1265
 - v_nmax, 1265
 - v_nmax_nbh_edges, 1265
 - v_nmax_nbh_vertices, 1265
 - vertex, 1265
 - vertex_fwd_iter, 1262
 - vertex_nbh_edge_fwd_iter, 1262
 - vertex_nbh_vertex_fwd_iter, 1262
 - vertices_t, 1262
- mln::util::nil, 1266
- mln::util::node, 1267
- mln::util::object_id, 1268
 - object_id, 1268
 - value_t, 1268
- mln::util::ord, 1269
- mln::util::ord_pair, 1270
 - change_both, 1270
 - change_first, 1271
 - change_second, 1271
 - first, 1271
 - second, 1271
- mln::util::pix, 1272
 - ima, 1273
 - p, 1273
 - pix, 1273
 - psite, 1272
 - v, 1273
 - value, 1272
- mln::util::set, 1274
 - bkd_eiter, 1275
 - clear, 1276
 - eiter, 1275
 - element, 1276
 - first_element, 1276
 - fwd_eiter, 1276
 - has, 1276
 - insert, 1277
 - is_empty, 1277
 - last_element, 1277
 - memory_size, 1277
 - nelements, 1278
 - remove, 1278
 - set, 1276
 - std_vector, 1278
- mln::util::site_pair, 1280
 - first, 1280
 - pair, 1280
 - second, 1280
- mln::util::soft_heap, 1281
 - ~soft_heap, 1282
 - clear, 1282
 - element, 1282
 - is_empty, 1282
 - is_valid, 1282
 - nelements, 1282
 - pop_front, 1283
 - push, 1283
 - soft_heap, 1282
- mln::util::timer, 1284
- mln::util::tracked_ptr, 1285
 - ~tracked_ptr, 1286
 - operator bool, 1286
 - operator!, 1286
 - operator->, 1286
 - operator=, 1286
 - tracked_ptr, 1285
- mln::util::tree, 1287
 - add_tree_down, 1288
 - add_tree_up, 1288
 - check_consistency, 1288
 - main_branch, 1288
 - root, 1288
 - tree, 1287
- mln::util::tree_node, 1289
 - add_child, 1290
 - check_consistency, 1290
 - children, 1291
 - delete_tree_node, 1291
 - elt, 1291
 - parent, 1291
 - print, 1292
 - search, 1292
 - search_rec, 1292
 - set_parent, 1292
 - tree_node, 1290
- mln::util::vertex, 1293
 - Category, 1294
 - change_graph, 1295
 - edge_with, 1295
 - graph, 1295

- graph_t, 1294
- id, 1295
- id_t, 1294
- id_value_t, 1294
- invalidate, 1295
- is_valid, 1295
- ith_nbh_edge, 1295
- ith_nbh_vertex, 1296
- nmax_nbh_edges, 1296
- nmax_nbh_vertices, 1296
- operator vertex_id_t, 1296
- other, 1296
- update_id, 1296
- vertex, 1295
- mln::util::yes, 1297
- mln::Value, 1298
- mln::value, 492
 - cast, 498
 - equiv, 498
 - float01_16, 496
 - float01_8, 496
 - gl16, 496
 - gl8, 496
 - glf, 497
 - int_s16, 497
 - int_s32, 497
 - int_s8, 497
 - int_u12, 497
 - int_u16, 497
 - int_u32, 497
 - int_u8, 497
 - label_16, 497
 - label_32, 497
 - label_8, 497
 - operator<<, 499–501
 - operator*, 498
 - operator+, 498
 - operator-, 498, 499
 - operator/, 499
 - operator==, 501
 - other, 502
 - rgb16, 498
 - rgb8, 498
 - stack, 502
- mln::value::float01, 1299
 - enc, 1300
 - equiv, 1300
 - float01, 1300
 - nbits, 1300
 - operator float, 1300
 - set_nbits, 1300
 - to_nbits, 1300
 - value, 1301
 - value_ind, 1301
- mln::value::float01_f, 1302
 - float01_f, 1302
 - operator float, 1302
 - operator=, 1302
 - value, 1303
- mln::value::graylevel, 1304
 - graylevel, 1305
 - operator=, 1305, 1306
 - to_float, 1306
 - value, 1306
- mln::value::graylevel_f, 1307
 - graylevel_f, 1308
 - operator graylevel< n >, 1308
 - operator=, 1308, 1309
 - value, 1309
- mln::value::impl, 503
- mln::value::int_s, 1310
 - int_s, 1311
 - one, 1311
 - operator int, 1311
 - operator=, 1311
 - zero, 1311
- mln::value::int_u, 1312
 - int_u, 1312, 1313
 - next, 1313
 - operator unsigned, 1313
 - operator-, 1313
 - operator=, 1313
- mln::value::int_u_sat, 1314
 - int_u_sat, 1315
 - one, 1315
 - operator int, 1315
 - operator+=", 1315
 - operator-=, 1315
 - operator=, 1315
 - zero, 1315
- mln::value::Integer, 1316
- mln::value::Integer< void >, 1317
- mln::value::label, 1318
 - enc, 1319
 - label, 1319
 - next, 1319
 - operator unsigned, 1319
 - operator++, 1319
 - operator-, 1319
 - operator=, 1320
 - prev, 1320
- mln::value::lut_vec, 1321
 - bkd_viter, 1322
 - fwd_viter, 1322
 - has, 1323
 - index_of, 1323
 - lut_vec, 1322
 - nvalues, 1323

- value, 1322
- mln::value::proxy, 1324
 - ~proxy, 1325
 - enc, 1325
 - equiv, 1325
 - operator=, 1325
 - proxy, 1325
 - to_value, 1325
- mln::value::rgb, 1327
 - operator=, 1328
 - red, 1328
 - rgb, 1327, 1328
 - zero, 1328
- mln::value::set, 1329
 - the, 1329
- mln::value::sign, 1330
 - enc, 1331
 - equiv, 1331
 - one, 1331
 - operator int, 1331
 - operator=, 1331
 - sign, 1331
 - zero, 1331
- mln::value::stack_image, 1332
 - domain_t, 1333
 - is_valid, 1334
 - lvalue, 1333
 - operator(), 1334
 - psite, 1333
 - rvalue, 1333
 - skeleton, 1333
 - stack_image, 1334
 - value, 1333
- mln::value::super_value< sign >, 1335
- mln::value::value_array, 1336
 - operator(), 1336
 - value_array, 1336
 - vset, 1337
- mln::Value_Iterator, 1338
 - next, 1338
 - operator<<, 1339
- mln::Value_Set, 1340
- mln::Vertex, 1341
- mln::vertex_image, 1342
 - graph_t, 1343
 - nbh_t, 1343
 - operator(), 1344
 - site_function_t, 1343
 - skeleton, 1343
 - vertex_image, 1343
 - vertex_nbh_t, 1343
 - vertex_win_t, 1343
 - win_t, 1343
- mln::violent_cast_image, 1345
 - lvalue, 1345
 - operator(), 1346
 - rvalue, 1345
 - skeleton, 1346
 - value, 1346
 - violent_cast_image, 1346
- mln::w_window, 1347
 - bkd_qiter, 1348
 - clear, 1349
 - dpsite, 1348
 - fwd_qiter, 1348
 - insert, 1349
 - is_symmetric, 1349
 - operator<<, 1350
 - operator-, 1350
 - operator==, 1350
 - std_vector, 1349
 - sym, 1349
 - w, 1349
 - w_window, 1349
 - weight, 1348
 - weights, 1349
 - win, 1350
- mln::Weighted_Window, 1351
 - operator-, 1351
- mln::win, 504
 - diff, 505
 - mln_regular, 505
 - sym, 506
- mln::win::backdiag2d, 1352
 - backdiag2d, 1352
 - length, 1352
- mln::win::ball, 1353
 - ball, 1353
 - diameter, 1353
- mln::win::cube3d, 1354
 - cube3d, 1354
 - length, 1355
- mln::win::cuboid3d, 1356
 - cuboid3d, 1357
 - depth, 1357
 - height, 1357
 - volume, 1357
 - width, 1357
- mln::win::diag2d, 1358
 - diag2d, 1358
 - length, 1358
- mln::win::line, 1359
 - length, 1360
 - line, 1360
 - size, 1360
- mln::win::multiple, 1361
- mln::win::multiple_size, 1362
- mln::win::octagon2d, 1363

- area, 1364
- length, 1364
- octagon2d, 1363
- mln::win::rectangle2d, 1365
 - area, 1366
 - height, 1366
 - rectangle2d, 1365
 - std_vector, 1366
 - width, 1366
- mln::Window, 1367
- mln::window, 1368
 - bkd_qiter, 1369
 - clear, 1370
 - delta, 1370
 - dp, 1370
 - fwd_qiter, 1369
 - has, 1370
 - insert, 1370, 1371
 - is_centered, 1371
 - is_empty, 1371
 - is_symmetric, 1371
 - operator==, 1372
 - print, 1371
 - qiter, 1370
 - regular, 1370
 - size, 1371
 - std_vector, 1372
 - sym, 1372
 - window, 1370
- mln::world::inter_pixel::is_separator, 1373
- mln_ch_convolve
 - mln::linear, 368
- mln_ch_convolve_grad
 - mln::linear, 369
- mln_exact
 - mln, 168
- mln_gen_complex_neighborhood
 - mln, 168
- mln_gen_complex_window
 - mln, 168, 169
- mln_gen_complex_window_p
 - mln, 169, 170
- mln_geom
 - mln::topo::is_simple_cell, 1203
- mln_image_from_grid
 - mln::convert, 226, 227
- mln_meta_accu_result
 - mln::accu, 181
 - mln::data, 236
 - mln::set, 461
- mln_q_subject
 - mln::p_graph_piter, 1004
- mln_regular
 - mln, 170
- mln::win, 505
- mln_trait_op_geq
 - mln, 170
- mln_trait_op_greater
 - mln, 170
- mln_trait_op_leq
 - mln, 171
- mln_trait_op_minus_twice
 - mln::morpho::elementary, 426
- mln_trait_op_neq
 - mln, 171
- mln_window
 - mln::convert, 227
- modneighb1d
 - c2, 112
 - neighb1d, 112
- modneighb2d
 - c2_col, 113
 - c2_row, 113
 - c4, 114
 - c8, 114
 - neighb2d, 113
- modneighb3d
 - c18, 115
 - c26, 116
 - c4_3d, 116
 - c6, 117
 - c8_3d, 117
 - neighb3d, 115
- modwin1d
 - segment1d, 126
 - window1d, 126
- modwin2d
 - disk2d, 128
 - hline2d, 128
 - vline2d, 128
 - win_c4p, 128
 - win_c8p, 128
 - window2d, 128
- modwin3d
 - sphere3d, 130
 - win_c4p_3d, 131
 - win_c8p_3d, 131
 - window3d, 130
- Multiple accumulators, 97
- Multiple windows, 133
- n
 - mln::complex_psite, 682
 - mln::faces_psite, 771
 - mln::p_n_faces_bkd_piter, 1040
 - mln::p_n_faces_fwd_piter, 1042
 - mln::topo::algebraic_face, 1180
 - mln::topo::algebraic_n_face, 1185

- mln::topo::face, 1196
- mln::topo::n_face, 1206
- mln::topo::n_face_bkd_iter, 1208
- mln::topo::n_face_fwd_iter, 1210
- N-D windows, 132
- n_face
 - mln::topo::n_face, 1205
- n_face_bkd_iter
 - mln::topo::n_face_bkd_iter, 1208
- n_face_fwd_iter
 - mln::topo::n_face_fwd_iter, 1210
- n_items
 - mln::accu::stat::var, 619
 - mln::accu::stat::variance, 622
- nbh_t
 - mln::edge_image, 756
 - mln::vertex_image, 1343
- nbits
 - mln::value::float01, 1300
- ncols
 - mln::geom, 300
 - mln::image2d, 883
 - mln::image3d, 891
- nedges
 - mln::p_edges, 994
- neighb
 - mln::neighb, 965
- neighb1d
 - modneighb1d, 112
- neighb2d
 - modneighb2d, 113
- neighb3d
 - modneighb3d, 115
- Neighborhoods, 111
- nelements
 - mln::doc::Fastest_Image, 704
 - mln::image1d, 878
 - mln::image2d, 883
 - mln::image3d, 891
 - mln::p_mutable_array_of, 1037
 - mln::p_set_of, 1083
 - mln::util::array, 1230
 - mln::util::fibonacci_heap, 1246
 - mln::util::set, 1278
 - mln::util::soft_heap, 1282
- next
 - mln::bkd_pixter1d, 634
 - mln::bkd_pixter2d, 636
 - mln::bkd_pixter3d, 638
 - mln::box_runend_piter, 653
 - mln::box_runstart_piter, 655
 - mln::complex_neighborhood_bkd_piter, 677
 - mln::complex_neighborhood_fwd_piter, 679
 - mln::complex_window_bkd_piter, 684
 - mln::complex_window_fwd_piter, 686
 - mln::dpoints_bkd_pixter, 745
 - mln::dpoints_fwd_pixter, 748
 - mln::dpsites_bkd_piter, 751
 - mln::dpsites_fwd_piter, 753
 - mln::fwd_pixter1d, 824
 - mln::fwd_pixter2d, 826
 - mln::fwd_pixter3d, 828
 - mln::graph_window_if_piter, 863
 - mln::graph_window_piter, 866
 - mln::Iterator, 899
 - mln::p_graph_piter, 1004
 - mln::p_indexed_bkd_piter, 1016
 - mln::p_indexed_fwd_piter, 1018
 - mln::p_n_faces_bkd_piter, 1040
 - mln::p_n_faces_fwd_piter, 1042
 - mln::p_transformed_piter, 1091
 - mln::Pixel_Iterator, 1108
 - mln::Site_Iterator, 1142
 - mln::topo::adj_higher_dim_connected_n_-
face_bkd_iter, 1159
 - mln::topo::adj_higher_dim_connected_n_-
face_fwd_iter, 1161
 - mln::topo::adj_higher_face_bkd_iter, 1163
 - mln::topo::adj_higher_face_fwd_iter, 1164
 - mln::topo::adj_lower_dim_connected_n_-
face_bkd_iter, 1165
 - mln::topo::adj_lower_dim_connected_n_-
face_fwd_iter, 1167
 - mln::topo::adj_lower_face_bkd_iter, 1169
 - mln::topo::adj_lower_face_fwd_iter, 1170
 - mln::topo::adj_lower_higher_face_bkd_iter,
1171
 - mln::topo::adj_lower_higher_face_fwd_iter,
1172
 - mln::topo::adj_m_face_bkd_iter, 1174
 - mln::topo::adj_m_face_fwd_iter, 1176
 - mln::topo::center_only_iter, 1187
 - mln::topo::centered_bkd_iter_adapter, 1188
 - mln::topo::centered_fwd_iter_adapter, 1189
 - mln::topo::face_bkd_iter, 1197
 - mln::topo::face_fwd_iter, 1199
 - mln::topo::n_face_bkd_iter, 1209
 - mln::topo::n_face_fwd_iter, 1210
 - mln::topo::static_n_face_bkd_iter, 1214
 - mln::topo::static_n_face_fwd_iter, 1216
 - mln::util::branch_iter, 1235
 - mln::util::branch_iter_ind, 1237
 - mln::value::int_u, 1313
 - mln::value::label, 1319
 - mln::Value_Iterator, 1338
- nfaces
 - mln::p_complex, 986
 - mln::p_faces, 1000

- mln::topo::complex, 1191
- nfaces_of_dim
 - mln::p_complex, 986
 - mln::topo::complex, 1192
- nfaces_of_static_dim
 - mln::topo::complex, 1192
- ninds
 - mln::geom, 300
 - mln::image1d, 878
- niter
 - mln::doc::Neighborhood, 718
 - mln::graph_elt_mixed_neighborhood, 841
 - mln::graph_elt_neighborhood, 847
 - mln::graph_elt_neighborhood_if, 850
 - mln::mixed_neighb, 949
 - mln::neighb, 965
- nlabels
 - mln::labeled_image, 902
 - mln::labeled_image_base, 906
- nmax_nbh_edges
 - mln::util::edge, 1243
 - mln::util::vertex, 1296
- nmax_nbh_vertices
 - mln::util::vertex, 1296
- not_inplace
 - mln::logical, 380
- nrows
 - mln::geom, 301
 - mln::image2d, 883
 - mln::image3d, 891
- nsites
 - mln::Box, 650
 - mln::box, 644
 - mln::doc::Box, 696
 - mln::doc::Fastest_Image, 704
 - mln::doc::Image, 713
 - mln::geom, 301
 - mln::p_array, 975
 - mln::p_complex, 986
 - mln::p_edges, 994
 - mln::p_faces, 1000
 - mln::p_image, 1014
 - mln::p_key, 1025
 - mln::p_line2d, 1031
 - mln::p_priority, 1049
 - mln::p_queue, 1056
 - mln::p_queue_fast, 1063
 - mln::p_run, 1070
 - mln::p_set, 1077
 - mln::p_vertices, 1103
- nslices
 - mln::image3d, 892
- nslis
 - mln::geom, 301
- nvalues
 - mln::doc::Value_Set, 733
 - mln::value::lut_vec, 1323
- nvertices
 - mln::p_vertices, 1103
- object_id
 - mln::util::object_id, 1268
- octagon2d
 - mln::win::octagon2d, 1363
- olive
 - mln::literal, 377
- On images, 94
- On site sets, 93
- On values, 95
- one
 - mln::literal, 377
 - mln::value::int_s, 1311
 - mln::value::int_u_sat, 1315
 - mln::value::sign, 1331
- opening
 - mln::morpho::elementary, 426
- operator bool
 - mln::util::tracked_ptr, 1286
- operator decorated_image< const I, D >
 - mln::decorated_image, 689
- operator edge_id_t
 - mln::util::edge, 1243
- operator float
 - mln::value::float01, 1300
 - mln::value::float01_f, 1302
- operator graylevel< n >
 - mln::value::graylevel_f, 1308
- operator I
 - mln::plain, 1111
- operator image_if< const I, F >
 - mln::image_if, 894
- operator int
 - mln::value::int_s, 1311
 - mln::value::int_u_sat, 1315
 - mln::value::sign, 1331
- operator mat< n, 1, U >
 - mln::algebra::h_vec, 632
- operator mln::algebra::vec< dpoint< G, C >::dim, Q >
 - mln::dpoint, 742
- operator psite
 - mln::doc::Site_Iterator, 727
- operator safe_image< const I >
 - mln::safe_image, 1137
- operator slice_image< const I >
 - mln::slice_image, 1151
- operator sub_image< const I, S >
 - mln::sub_image, 1153

- operator thru_image< const I, F >
 - mln::thru_image, 1156
- operator thrubin_image< const I1, const I2, F >
 - mln::thrubin_image, 1158
- operator transformed_image< const I, F >
 - mln::transformed_image, 1222
- operator typename internal::p_image_site_set< I >::ret
 - mln::p_image, 1014
- operator unsigned
 - mln::value::int_u, 1313
 - mln::value::label, 1319
- operator util::tree_node< T > &
 - mln::util::branch_iter, 1235
 - mln::util::branch_iter_ind, 1237
- operator value
 - mln::doc::Value_Iterator, 731
- operator vertex_id_t
 - mln::util::vertex, 1296
- operator!
 - mln::util::tracked_ptr, 1286
- operator!=
 - mln, 171
 - mln::topo, 472, 473
- operator<
 - mln, 173
 - mln::Box, 650, 651
 - mln::box, 646
 - mln::p_array, 976
 - mln::p_centered, 981
 - mln::p_complex, 987
 - mln::p_edges, 994
 - mln::p_faces, 1001
 - mln::p_if, 1008
 - mln::p_image, 1014
 - mln::p_key, 1026
 - mln::p_line2d, 1032
 - mln::p_mutable_array_of, 1038
 - mln::p_priority, 1050
 - mln::p_queue, 1057
 - mln::p_queue_fast, 1064
 - mln::p_run, 1071
 - mln::p_set, 1077
 - mln::p_set_of, 1083
 - mln::p_transformed, 1088
 - mln::p_vaccess, 1096
 - mln::p_vertices, 1104
 - mln::Site_Set, 1146
 - mln::topo, 473, 474
 - mln::util, 488
- operator<<
 - mln, 173, 174
 - mln::Box, 651
 - mln::box, 646
- mln::fun::i2v, 277
- mln::Gpoint, 837
- mln::p_array, 976
- mln::p_centered, 981
- mln::p_complex, 987
- mln::p_edges, 995
- mln::p_faces, 1001
- mln::p_if, 1008
- mln::p_image, 1014
- mln::p_key, 1026
- mln::p_line2d, 1032
- mln::p_mutable_array_of, 1038
- mln::p_priority, 1050
- mln::p_queue, 1057
- mln::p_queue_fast, 1064
- mln::p_run, 1071
- mln::p_set, 1078
- mln::p_set_of, 1083
- mln::p_transformed, 1088
- mln::p_vaccess, 1096
- mln::p_vertices, 1104
- mln::point, 1123
- mln::Point_Site, 1126
- mln::Site_Set, 1147
- mln::topo, 474, 475
- mln::util, 488, 489
- mln::value, 499–501
- mln::Value_Iterator, 1339
- mln::w_window, 1350
- operator<=
 - mln, 174
 - mln::Box, 651
 - mln::box, 646, 647
 - mln::p_array, 977
 - mln::p_centered, 982
 - mln::p_complex, 987
 - mln::p_edges, 995
 - mln::p_faces, 1001
 - mln::p_if, 1009
 - mln::p_image, 1015
 - mln::p_key, 1027
 - mln::p_line2d, 1033
 - mln::p_mutable_array_of, 1038
 - mln::p_priority, 1050
 - mln::p_queue, 1057
 - mln::p_queue_fast, 1065
 - mln::p_run, 1071
 - mln::p_set, 1078
 - mln::p_set_of, 1084
 - mln::p_transformed, 1089
 - mln::p_vaccess, 1096
 - mln::p_vertices, 1104
 - mln::Site_Set, 1147
- operator*

- mln, 172
- mln::algebra, 195
- mln::value, 498
- operator()
 - mln::complex_image, 675
 - mln::decorated_image, 689
 - mln::doc::Fastest_Image, 704, 705
 - mln::doc::Image, 713
 - mln::edge_image, 757
 - mln::extension_fun, 761
 - mln::extension_ima, 764
 - mln::extension_val, 767
 - mln::flat_image, 773
 - mln::fun::x2v::bilinear, 805
 - mln::fun::x2x::linear, 808
 - mln::fun::x2x::rotation, 811
 - mln::fun::x2x::translation, 814
 - mln::fun_image, 816
 - mln::geom::complex_geometry, 833
 - mln::hexa, 870
 - mln::image1d, 878
 - mln::image2d, 883, 884
 - mln::image2d_h, 887
 - mln::image3d, 892
 - mln::lazy_image, 909
 - mln::p2p_image, 970
 - mln::p_key, 1026
 - mln::p_priority, 1049
 - mln::p_vaccess, 1096
 - mln::p_vertices, 1104
 - mln::slice_image, 1151
 - mln::topo::is_simple_cell, 1203
 - mln::tr_image, 1220
 - mln::transformed_image, 1222
 - mln::unproject_image, 1223, 1224
 - mln::util::array, 1230
 - mln::util::greater_point, 1255
 - mln::util::greater_psite, 1256
 - mln::value::stack_image, 1334
 - mln::value::value_array, 1336
 - mln::vertex_image, 1344
 - mln::violent_cast_image, 1346
- operator+
 - mln::Gpoint, 835
 - mln::point, 1121
 - mln::Point_Site, 1125
 - mln::topo, 473
 - mln::value, 498
- operator++
 - mln, 172
 - mln::value::label, 1319
- operator+=
 - mln::Gpoint, 835
 - mln::Point, 1113
 - mln::point, 1119, 1121
 - mln::value::int_u_sat, 1315
- operator-
 - mln, 172
 - mln::Gpoint, 836
 - mln::point, 1121
 - mln::Point_Site, 1125
 - mln::topo, 473
 - mln::value, 498, 499
 - mln::value::int_u, 1313
 - mln::w_window, 1350
 - mln::Weighted_Window, 1351
- operator->
 - mln::util::tracked_ptr, 1286
- operator-
 - mln, 172
 - mln::value::label, 1319
- operator=
 - mln::Gpoint, 836
 - mln::Point, 1113
 - mln::point, 1120, 1122
 - mln::value::int_u_sat, 1315
- operator/
 - mln::Gpoint, 836
 - mln::Point, 1114
 - mln::point, 1122
 - mln::value, 499
- operator=
 - mln::plain, 1111
 - mln::util::fibonacci_heap, 1247
 - mln::util::tracked_ptr, 1286
 - mln::value::float01_f, 1302
 - mln::value::graylevel, 1305, 1306
 - mln::value::graylevel_f, 1308, 1309
 - mln::value::int_s, 1311
 - mln::value::int_u, 1313
 - mln::value::int_u_sat, 1315
 - mln::value::label, 1320
 - mln::value::proxy, 1325
 - mln::value::rgb, 1328
 - mln::value::sign, 1331
- operator==
 - mln, 175, 176
 - mln::Box, 651
 - mln::box, 647
 - mln::Gpoint, 837
 - mln::p_array, 977
 - mln::p_centered, 982
 - mln::p_complex, 988
 - mln::p_edges, 995
 - mln::p_faces, 1001
 - mln::p_if, 1009
 - mln::p_image, 1015
 - mln::p_key, 1027

- mln::p_line2d, 1033
- mln::p_mutable_array_of, 1038
- mln::p_priority, 1051
- mln::p_queue, 1057
- mln::p_queue_fast, 1065
- mln::p_run, 1071
- mln::p_set, 1078
- mln::p_set_of, 1084
- mln::p_transformed, 1089
- mln::p_vaccess, 1097
- mln::p_vertices, 1105
- mln::point, 1123
- mln::Point_Site, 1126
- mln::Site_Set, 1147
- mln::topo, 475
- mln::util, 489
- mln::value, 501
- mln::w_window, 1350
- mln::window, 1372
- operator |
 - mln, 176, 177
- or_inplace
 - mln::logical, 381
- orange
 - mln::literal, 377
- ord_pair
 - mln::make, 399
- ord_strict
 - mln::util, 489
- ord_weak
 - mln::util, 489
- origin
 - mln::algebra::h_vec, 632
 - mln::literal, 377
 - mln::point, 1123
- other
 - mln::util::vertex, 1296
 - mln::value, 502
- overset
 - mln::p_if, 1008
- P
 - mln::graph_window_if_piter, 862
 - mln::graph_window_piter, 865
- p
 - mln::util::pix, 1273
- p2p_image
 - mln::p2p_image, 970
- p_array
 - mln::p_array, 974
- p_centered
 - mln::p_centered, 980
- p_complex
 - mln::p_complex, 985
- p_edges
 - mln::p_edges, 992, 993
- p_edges_with_mass_centers
 - mln::make, 399
- p_faces
 - mln::p_faces, 999
- p_graph_piter
 - mln::p_graph_piter, 1003
- p_if
 - mln::p_if, 1007
- p_image
 - mln::p_image, 1013
- p_indexed_bkd_piter
 - mln::p_indexed_bkd_piter, 1016
- p_indexed_fwd_piter
 - mln::p_indexed_fwd_piter, 1018
- p_key
 - mln::p_key, 1024
- p_line2d
 - mln::p_line2d, 1030
- p_mutable_array_of
 - mln::p_mutable_array_of, 1036
- p_n_faces_bkd_piter
 - mln::p_n_faces_bkd_piter, 1040
- p_n_faces_fwd_piter
 - mln::p_n_faces_fwd_piter, 1042
- p_priority
 - mln::p_priority, 1047
- p_queue
 - mln::p_queue, 1055
- p_queue_fast
 - mln::p_queue_fast, 1062
- p_run
 - mln::p_run, 1069
- p_run2d
 - mln, 163
- p_runs2d
 - mln, 163
- p_set
 - mln::p_set, 1076
- p_set_of
 - mln::p_set_of, 1082
- p_transformed
 - mln::p_transformed, 1087
- p_transformed_piter
 - mln::p_transformed_piter, 1090
- p_vaccess
 - mln::p_vaccess, 1095
- p_vertices
 - mln::p_vertices, 1101, 1102
- p_vertices_with_mass_centers
 - mln::make, 399
- pack
 - mln::labeling, 358

- pack_inplace
 - mln::labeling, 358
- pair
 - mln::util::site_pair, 1280
- parent
 - mln::util::tree_node, 1291
- paste
 - mln::data, 236
 - mln::data::impl::generic, 250
- paste_without_localization
 - mln::data, 237
- pink
 - mln::literal, 377
- piter
 - mln::box, 642
 - mln::p_array, 974
 - mln::p_centered, 980
 - mln::p_complex, 985
 - mln::p_edges, 992
 - mln::p_faces, 999
 - mln::p_if, 1007
 - mln::p_image, 1012
 - mln::p_key, 1024
 - mln::p_line2d, 1030
 - mln::p_mutable_array_of, 1036
 - mln::p_priority, 1047
 - mln::p_queue, 1054
 - mln::p_queue_fast, 1062
 - mln::p_run, 1068
 - mln::p_set, 1075
 - mln::p_set_of, 1082
 - mln::p_transformed, 1087
 - mln::p_vaccess, 1094
 - mln::p_vertices, 1101
- pix
 - mln::make, 399
 - mln::util::pix, 1273
- pixel
 - mln::make, 400
 - mln::pixel, 1106
- plain
 - mln::plain, 1111
- plot
 - mln::draw, 268
- plus
 - mln::arith, 202, 203
 - mln::morpho, 420
- plus_est
 - mln::arith, 203, 204
- plus_est_inplace
 - mln::arith, 204
- plus_infty
 - mln::point, 1120
- plus_inplace
 - mln::arith, 204
- pmax
 - mln::box, 645
 - mln::doc::Box, 696
- pmin
 - mln::box, 645
 - mln::doc::Box, 696
- pmin_pmax
 - mln::geom, 301, 302
- point
 - mln::doc::Dpoint, 698
 - mln::doc::Fastest_Image, 701
 - mln::doc::Image, 711
 - mln::doc::Neighborhood, 718
 - mln::doc::Point_Site, 724
 - mln::doc::Weighted_Window, 735
 - mln::Point, 1113
 - mln::point, 1118, 1119
- point1d
 - mln, 163
- point1df
 - mln, 163
- point2d
 - mln, 163
- point2d_h
 - mln, 163
 - mln::make, 400
- point2df
 - mln, 163
- point3d
 - mln, 164
- point3df
 - mln, 164
- point_at_index
 - mln::doc::Fastest_Image, 705
 - mln::image1d, 878
 - mln::image2d, 884
 - mln::image3d, 892
- pop
 - mln::p_priority, 1049
 - mln::p_queue, 1056
 - mln::p_queue_fast, 1063
- pop_front
 - mln::p_priority, 1049
 - mln::p_queue, 1056
 - mln::p_queue_fast, 1063
 - mln::util::fibonacci_heap, 1247
 - mln::util::soft_heap, 1283
- positive
 - mln::test, 464
- pred
 - mln::p_if, 1008
- predicate
 - mln::p_if, 1008

- mln::test, 464, 465
- prev
 - mln::value::label, 1320
- primary
 - mln, 177
- primary_set
 - mln::p_transformed, 1088
- print
 - mln::topo::complex, 1192
 - mln::util::tree_node, 1292
 - mln::window, 1371
- print_faces
 - mln::topo::complex, 1192
- println
 - mln::debug, 258
- println_with_border
 - mln::debug, 258
- priorities
 - mln::p_priority, 1049
- propagate_if
 - mln::morpho::tree, 437
- propagate_if_value
 - mln::morpho::tree, 437
- propagate_node_to_ancestors
 - mln::morpho::tree, 438
- propagate_node_to_descendants
 - mln::morpho::tree, 438
- propagate_representative
 - mln::morpho::tree, 439
- proxy
 - mln::value::proxy, 1325
- pset
 - mln::doc::Fastest_Image, 702
 - mln::doc::Image, 711
 - mln::p_vaccess, 1094
- psite
 - mln::box, 642
 - mln::complex_neighborhood_bkd_piter, 676
 - mln::complex_neighborhood_fwd_piter, 678
 - mln::complex_window_bkd_piter, 683
 - mln::complex_window_fwd_piter, 685
 - mln::decorated_image, 688
 - mln::doc::Box, 695
 - mln::doc::Fastest_Image, 702
 - mln::doc::Image, 711
 - mln::doc::Site_Iterator, 727
 - mln::doc::Site_Set, 729
 - mln::dpoint, 740
 - mln::graph_elt_mixed_window, 844
 - mln::graph_elt_window, 853
 - mln::graph_elt_window_if, 857
 - mln::hexa, 869
 - mln::image2d_h, 886
 - mln::interpolated, 895
 - mln::p_array, 974
 - mln::p_centered, 980
 - mln::p_complex, 985
 - mln::p_edges, 992
 - mln::p_faces, 999
 - mln::p_if, 1007
 - mln::p_image, 1012
 - mln::p_key, 1024
 - mln::p_line2d, 1030
 - mln::p_mutable_array_of, 1036
 - mln::p_priority, 1047
 - mln::p_queue, 1054
 - mln::p_queue_fast, 1062
 - mln::p_run, 1068
 - mln::p_set, 1075
 - mln::p_set_of, 1082
 - mln::p_transformed, 1087
 - mln::p_vaccess, 1094
 - mln::p_vertices, 1101
 - mln::thrubin_image, 1157
 - mln::topo::is_simple_cell, 1203
 - mln::tr_image, 1219
 - mln::util::pix, 1272
 - mln::value::stack_image, 1333
- ptransform
 - mln, 177
- purge
 - mln::p_queue_fast, 1064
- purple
 - mln::literal, 377
- push
 - mln::p_priority, 1049
 - mln::p_queue, 1056
 - mln::p_queue_fast, 1064
 - mln::util::fibonacci_heap, 1247
 - mln::util::soft_heap, 1283
- put_word
 - mln::debug, 258
- q_box
 - mln::p_line2d, 1030
 - mln::p_run, 1068
- qiter
 - mln::doc::Window, 737
 - mln::graph_elt_mixed_window, 844
 - mln::graph_elt_window, 853
 - mln::graph_elt_window_if, 857
 - mln::window, 1370
- Queue based, 123
- r_element
 - mln::p_image, 1012
 - mln::p_key, 1024
 - mln::p_set, 1076

- rag_and_labeled_wsl
 - mln::make, 400
 - rank_filter
 - mln::morpho, 420
 - mln::morpho::impl::generic, 428
 - read_header
 - mln::io::fld, 321
 - rectangle2d
 - mln::win::rectangle2d, 1365
 - rectangularity
 - mln::accu::site_set::rectangularity, 591
 - red
 - mln::literal, 377
 - mln::value::rgb, 1328
 - region_adjacency_graph
 - mln::make, 401
 - regional_maxima
 - mln::labeling, 359
 - regional_minima
 - mln::labeling, 359
 - registration1
 - mln::registration, 457
 - registration2
 - mln::registration, 457
 - registration3
 - mln::registration, 457
 - regular
 - mln::window, 1370
 - relabel
 - mln::labeled_image, 902
 - mln::labeled_image_base, 906
 - mln::labeling, 359, 360
 - relabel_inplace
 - mln::labeling, 360
 - relabelfun
 - mln::make, 401, 402
 - remove
 - mln::p_image, 1014
 - mln::p_key, 1026
 - mln::p_set, 1077
 - mln::util::set, 1278
 - remove_key
 - mln::p_key, 1026
 - replace
 - mln::data, 237
 - reserve
 - mln::p_array, 976
 - mln::p_mutable_array_of, 1037
 - mln::p_queue_fast, 1064
 - mln::topo::n_faces_set, 1213
 - mln::util::array, 1230
 - resize
 - mln::border, 213
 - mln::p_array, 976
 - mln::util::array, 1231
 - result
 - mln::graph::attribute::card_t, 839
 - mln::graph::attribute::representative_t, 840
 - mln::topo::is_simple_cell, 1203
 - mln::util::array, 1228
 - revert
 - mln::arith, 205
 - revert_inplace
 - mln::arith, 205
 - rgb
 - mln::value::rgb, 1327, 1328
 - rgb16
 - mln::value, 498
 - rgb8
 - mln::value, 498
 - rgb8_2complex_image3df
 - mln, 164
 - root
 - mln::util::tree, 1288
 - rotate
 - mln::geom, 302
 - rotation
 - mln::fun::x2x::rotation, 811
 - Routines, 107
 - run_length
 - mln::box_runend_piter, 654
 - mln::box_runstart_piter, 656
 - rvalue
 - mln::complex_image, 674
 - mln::decorated_image, 688
 - mln::doc::Fastest_Image, 702
 - mln::doc::Generalized_Pixel, 707
 - mln::doc::Image, 712
 - mln::doc::Pixel_Iterator, 721
 - mln::extension_fun, 761
 - mln::extension_ima, 764
 - mln::extension_val, 767
 - mln::flat_image, 773
 - mln::fun_image, 816
 - mln::hexa, 869
 - mln::image1d, 876
 - mln::image2d, 881
 - mln::image2d_h, 886
 - mln::image3d, 889
 - mln::interpolated, 896
 - mln::lazy_image, 908
 - mln::thrubin_image, 1157
 - mln::tr_image, 1219
 - mln::value::stack_image, 1333
 - mln::violent_cast_image, 1345
- S
- mln::p_image, 1012

- sagittal_dec
 - mln, 177
- saturate
 - mln::data, 237
- saturate_inplace
 - mln::data, 238
- save
 - mln::io::cloud, 317
 - mln::io::dump, 319
 - mln::io::magick, 324
 - mln::io::off, 325
 - mln::io::pbm, 328
 - mln::io::pfm, 333
 - mln::io::pgm, 336
 - mln::io::plot, 338, 339
 - mln::io::pnm, 341
 - mln::io::ppm, 345
 - mln::io::txt, 348
- save_bin_alt
 - mln::io::off, 326
- search
 - mln::util::tree_node, 1292
- search_rec
 - mln::util::tree_node, 1292
- second
 - mln::util::couple, 1239
 - mln::util::ord_pair, 1271
 - mln::util::site_pair, 1280
- seeds2tiling
 - mln::geom, 302
 - mln::geom::impl, 305
- seeds2tiling_roundness
 - mln::geom, 303
 - mln::geom::impl, 305
- segment1d
 - modwin1d, 126
- set
 - mln::util::set, 1276
- set_all
 - mln::dpoint, 742
 - mln::point, 1120
- set_alpha
 - mln::fun::x2x::rotation, 811
- set_axis
 - mln::fun::x2x::rotation, 812
- set_cplx
 - mln::topo::algebraic_face, 1180
 - mln::topo::algebraic_n_face, 1185
 - mln::topo::face, 1196
 - mln::topo::n_face, 1206
- set_face_id
 - mln::topo::algebraic_face, 1180
 - mln::topo::algebraic_n_face, 1185
 - mln::topo::face, 1196
 - mln::topo::n_face, 1207
- set_image
 - mln::topo::is_simple_cell, 1203
- set_n
 - mln::topo::algebraic_face, 1181
 - mln::topo::face, 1196
- set_nbits
 - mln::value::float01, 1300
- set_parent
 - mln::util::tree_node, 1292
- set_sign
 - mln::topo::algebraic_face, 1181
 - mln::topo::algebraic_n_face, 1185
- set_t
 - mln::fun::x2x::translation, 814
- set_tr
 - mln::tr_image, 1220
- set_value
 - mln::accu::count_adjacent_vertices, 512
 - mln::accu::count_labels, 513
 - mln::accu::count_value, 515
 - mln::accu::math::count, 531
 - mln::accu::shape::height, 586
 - mln::accu::shape::volume, 589
 - mln::accu::stat::max, 595
 - mln::accu::stat::min, 606
 - mln::morpho::attribute::sum, 961
- sign
 - mln::topo::algebraic_face, 1181
 - mln::topo::algebraic_n_face, 1185
 - mln::value::sign, 1331
- site
 - mln::box, 642
 - mln::doc::Box, 695
 - mln::doc::Site_Set, 729
 - mln::dpoint, 740
 - mln::graph_elt_mixed_window, 845
 - mln::graph_elt_window, 853
 - mln::graph_elt_window_if, 857
 - mln::graph_window_base, 861
 - mln::p_centered, 980
 - mln::tr_image, 1219
- Site sets, 118
- site_function_t
 - mln::edge_image, 756
 - mln::vertex_image, 1343
- site_set
 - mln::complex_psite, 682
 - mln::faces_psite, 771
- size
 - mln::util::array, 1231
 - mln::win::line, 1360
 - mln::window, 1371
- skeleton

- mln::complex_image, 674
- mln::decorated_image, 688
- mln::doc::Fastest_Image, 702
- mln::doc::Image, 712
- mln::edge_image, 756
- mln::extended, 758
- mln::extension_fun, 761
- mln::extension_ima, 764
- mln::extension_val, 767
- mln::flat_image, 773
- mln::fun_image, 816
- mln::hexa, 869
- mln::image1d, 876
- mln::image2d, 881
- mln::image2d_h, 886
- mln::image3d, 890
- mln::image_if, 893
- mln::interpolated, 896
- mln::labeled_image, 901
- mln::lazy_image, 908
- mln::p2p_image, 969
- mln::plain, 1110
- mln::pw::image, 1133
- mln::safe_image, 1137
- mln::slice_image, 1150
- mln::sub_image, 1152
- mln::sub_image_if, 1154
- mln::thruin_image, 1157
- mln::tr_image, 1219
- mln::transformed_image, 1221
- mln::value::stack_image, 1333
- mln::vertex_image, 1343
- mln::violent_cast_image, 1346
- sli
 - mln::slice_image, 1151
- slice_image
 - mln::slice_image, 1151
- slices_2d
 - mln::debug, 258
- soft_heap
 - mln::util::soft_heap, 1282
- sort_offsets_increasing
 - mln::data, 238
 - mln::data::impl::generic, 250
- sort_psites_decreasing
 - mln::data, 238
- sort_psites_increasing
 - mln::data, 238
- space_2complex_geometry
 - mln, 164
- Sparse types, 122
- sphere3d
 - modwin3d, 130
- sqr_l2
 - mln::norm, 449
- stack
 - mln::value, 502
- stack_image
 - mln::value::stack_image, 1334
- standard_deviation
 - mln::accu::stat::variance, 622
- start
 - mln::doc::Iterator, 715
 - mln::doc::Pixel_Iterator, 721
 - mln::doc::Site_Iterator, 727
 - mln::doc::Value_Iterator, 731
 - mln::dpoints_bkd_pixter, 746
 - mln::dpoints_fwd_pixter, 749
 - mln::p_run, 1070
 - mln::topo::face_bkd_iter, 1197
 - mln::topo::face_fwd_iter, 1199
 - mln::topo::n_face_bkd_iter, 1209
 - mln::topo::n_face_fwd_iter, 1211
 - mln::topo::static_n_face_bkd_iter, 1215
 - mln::topo::static_n_face_fwd_iter, 1217
 - mln::util::branch_iter, 1235
 - mln::util::branch_iter_ind, 1237
- static_n_face_bkd_iter
 - mln::topo::static_n_face_bkd_iter, 1214
- static_n_face_fwd_iter
 - mln::topo::static_n_face_fwd_iter, 1216
- std_deque
 - mln::p_queue, 1056
- std_vector
 - mln::p_array, 976
 - mln::p_line2d, 1032
 - mln::p_queue_fast, 1064
 - mln::p_set, 1077
 - mln::util::array, 1231
 - mln::util::set, 1278
 - mln::w_window, 1349
 - mln::win::rectangle2d, 1366
 - mln::window, 1372
- stretch
 - mln::data, 239
 - mln::data::impl, 246
- structural
 - mln::morpho::closing::approx, 424
 - mln::morpho::opening::approx, 429
- sub_image
 - mln::sub_image, 1152
- sub_image_if
 - mln::sub_image_if, 1154
- subdomain
 - mln::labeled_image, 903
 - mln::labeled_image_base, 906
- subsampling
 - mln::subsampling, 462

- subtractive
 - mln::morpho::tree::filter, 441
- sum
 - mln::accu::stat::mean, 600
 - mln::accu::stat::variance, 622
 - mln::estim, 270
- superpose
 - mln::debug, 258
 - mln::labeling, 361
 - mln::morpho::watershed, 444
- sym
 - mln::doc::Weighted_Window, 736
 - mln::graph_elt_mixed_window, 845
 - mln::graph_elt_window, 854
 - mln::graph_elt_window_if, 859
 - mln::graph_window_base, 861
 - mln::w_window, 1349
 - mln::win, 506
 - mln::window, 1372
- sym_diff
 - mln::Box, 652
 - mln::box, 647
 - mln::p_array, 977
 - mln::p_centered, 982
 - mln::p_complex, 988
 - mln::p_edges, 995
 - mln::p_faces, 1001
 - mln::p_if, 1009
 - mln::p_image, 1015
 - mln::p_key, 1027
 - mln::p_line2d, 1033
 - mln::p_mutable_array_of, 1039
 - mln::p_priority, 1051
 - mln::p_queue, 1057
 - mln::p_queue_fast, 1065
 - mln::p_run, 1072
 - mln::p_set, 1078
 - mln::p_set_of, 1084
 - mln::p_transformed, 1089
 - mln::p_vaccess, 1097
 - mln::p_vertices, 1105
 - mln::Site_Set, 1147
- t
 - mln::algebra::h_mat, 630
 - mln::algebra::h_vec, 632
 - mln::fun::x2x::translation, 814
- take
 - mln::accu, 182
 - mln::accu::histo, 517
 - mln::accu::label_used, 519
 - mln::accu::stat::median_alt, 601
 - mln::doc::Accumulator, 692
- take_as_init
 - mln::accu::center, 508
 - mln::accu::convolve, 509
 - mln::accu::count_adjacent_vertices, 512
 - mln::accu::count_labels, 514
 - mln::accu::count_value, 516
 - mln::accu::histo, 517
 - mln::accu::label_used, 520
 - mln::accu::logic::land, 521
 - mln::accu::logic::land_basic, 524
 - mln::accu::logic::lor, 525
 - mln::accu::logic::lor_basic, 528
 - mln::accu::maj_h, 529
 - mln::accu::math::count, 532
 - mln::accu::math::inf, 533
 - mln::accu::math::sum, 536
 - mln::accu::math::sup, 537
 - mln::accu::max_site, 539
 - mln::accu::nil, 575
 - mln::accu::p, 577
 - mln::accu::pair, 580
 - mln::accu::rms, 581
 - mln::accu::shape::bbox, 583
 - mln::accu::shape::height, 586
 - mln::accu::shape::volume, 589
 - mln::accu::site_set::rectangularity, 592
 - mln::accu::stat::deviation, 594
 - mln::accu::stat::max, 596
 - mln::accu::stat::max_h, 597
 - mln::accu::stat::mean, 600
 - mln::accu::stat::median_alt, 602
 - mln::accu::stat::median_h, 604
 - mln::accu::stat::min, 607
 - mln::accu::stat::min_h, 608
 - mln::accu::stat::min_max, 611
 - mln::accu::stat::rank, 613
 - mln::accu::stat::rank < bool >, 614
 - mln::accu::stat::rank_high_quant, 616
 - mln::accu::stat::var, 619
 - mln::accu::stat::variance, 622
 - mln::accu::tuple, 624
 - mln::accu::val, 626
 - mln::Accumulator, 628
 - mln::morpho::attribute::card, 951
 - mln::morpho::attribute::count_adjacent_vertices, 953
 - mln::morpho::attribute::height, 956
 - mln::morpho::attribute::sharpness, 958
 - mln::morpho::attribute::sum, 961
 - mln::morpho::attribute::volume, 963
- take_n_times
 - mln::accu::center, 508
 - mln::accu::convolve, 510
 - mln::accu::count_adjacent_vertices, 512
 - mln::accu::count_labels, 514

- mln::accu::count_value, 516
- mln::accu::histo, 518
- mln::accu::label_used, 520
- mln::accu::logic::land, 522
- mln::accu::logic::land_basic, 524
- mln::accu::logic::lor, 526
- mln::accu::logic::lor_basic, 528
- mln::accu::maj_h, 530
- mln::accu::math::count, 532
- mln::accu::math::inf, 534
- mln::accu::math::sum, 536
- mln::accu::math::sup, 538
- mln::accu::max_site, 540
- mln::accu::nil, 576
- mln::accu::p, 578
- mln::accu::pair, 580
- mln::accu::rms, 582
- mln::accu::shape::bbox, 584
- mln::accu::shape::height, 586
- mln::accu::shape::volume, 589
- mln::accu::site_set::rectangularity, 592
- mln::accu::stat::deviation, 594
- mln::accu::stat::max, 596
- mln::accu::stat::max_h, 598
- mln::accu::stat::mean, 600
- mln::accu::stat::median_alt, 602
- mln::accu::stat::median_h, 604
- mln::accu::stat::min, 607
- mln::accu::stat::min_h, 609
- mln::accu::stat::min_max, 611
- mln::accu::stat::rank, 613
- mln::accu::stat::rank < bool >, 615
- mln::accu::stat::rank_high_quant, 617
- mln::accu::stat::var, 619
- mln::accu::stat::variance, 622
- mln::accu::tuple, 625
- mln::accu::val, 627
- mln::Accumulator, 628
- mln::morpho::attribute::card, 952
- mln::morpho::attribute::count_adjacent_vertices, 954
- mln::morpho::attribute::height, 956
- mln::morpho::attribute::sharpness, 958
- mln::morpho::attribute::sum, 961
- mln::morpho::attribute::volume, 963
- target
 - mln::graph_elt_mixed_window, 845
 - mln::graph_elt_window, 853
 - mln::graph_elt_window_if, 857
- target_site_set
 - mln::graph_window_piter, 867
- teal
 - mln::literal, 378
- the
 - mln::value::set, 1329
- thick_miss
 - mln::morpho, 420
- thickening
 - mln::morpho, 420
- thin_fit
 - mln::morpho, 420
- thinning
 - mln::morpho, 421
- threshold
 - mln::binarization, 210
- times
 - mln::arith, 206
- times_cst
 - mln::arith, 206
- times_inplace
 - mln::arith, 206
- to
 - mln::convert, 227
- to_dpoint
 - mln::convert, 227
 - mln::Dpoint, 738
- to_enc
 - mln::data, 239
- to_float
 - mln::value::graylevel, 1306
- to_fun
 - mln::convert, 227
- to_h_vec
 - mln::point, 1120
- to_image
 - mln::convert, 227
- to_larger
 - mln::box, 645
- to_nbits
 - mln::value::float01, 1300
- to_neighb
 - mln::graph, 308
- to_p_array
 - mln::convert, 227, 228
- to_p_set
 - mln::convert, 228
- to_point
 - mln::doc::Point_Site, 724
 - mln::Point, 1113
- to_result
 - mln::accu::center, 508
 - mln::accu::convolve, 510
 - mln::accu::count_adjacent_vertices, 512
 - mln::accu::count_labels, 514
 - mln::accu::count_value, 516
 - mln::accu::label_used, 520
 - mln::accu::logic::land, 522
 - mln::accu::logic::land_basic, 524

- mln::accu::logic::lor, 526
- mln::accu::logic::lor_basic, 528
- mln::accu::maj_h, 530
- mln::accu::math::count, 532
- mln::accu::math::inf, 534
- mln::accu::math::sum, 536
- mln::accu::math::sup, 538
- mln::accu::max_site, 540
- mln::accu::nil, 576
- mln::accu::p, 578
- mln::accu::pair, 580
- mln::accu::rms, 582
- mln::accu::shape::bbox, 584
- mln::accu::shape::height, 586
- mln::accu::shape::volume, 590
- mln::accu::site_set::rectangularity, 592
- mln::accu::stat::deviation, 594
- mln::accu::stat::max, 596
- mln::accu::stat::max_h, 598
- mln::accu::stat::mean, 600
- mln::accu::stat::median_alt, 602
- mln::accu::stat::median_h, 604
- mln::accu::stat::min, 607
- mln::accu::stat::min_h, 609
- mln::accu::stat::min_max, 611
- mln::accu::stat::rank, 613
- mln::accu::stat::rank< bool >, 615
- mln::accu::stat::rank_high_quant, 617
- mln::accu::stat::var, 619
- mln::accu::stat::variance, 623
- mln::accu::tuple, 625
- mln::accu::val, 627
- mln::morpho::attribute::card, 952
- mln::morpho::attribute::count_adjacent_vertices, 954
- mln::morpho::attribute::height, 956
- mln::morpho::attribute::sharpness, 958
- mln::morpho::attribute::sum, 961
- mln::morpho::attribute::volume, 963
- to_upper_window
 - mln::convert, 229
- to_value
 - mln::value::proxy, 1325
- to_vec
 - mln::algebra::h_vec, 632
 - mln::dpoint, 742
 - mln::point, 1121
- to_win
 - mln::graph, 308
- to_window
 - mln::convert, 229
- toggle
 - mln::p_image, 1014
- top_hat_black
 - mln::morpho, 421
 - mln::morpho::elementary, 426
- top_hat_self_complementary
 - mln::morpho, 421
 - mln::morpho::elementary, 426
- top_hat_white
 - mln::morpho, 421
 - mln::morpho::elementary, 426
- topological
 - mln::morpho::watershed, 444
- tr
 - mln::tr_image, 1220
- tr_image
 - mln::tr_image, 1219
- tracked_ptr
 - mln::util::tracked_ptr, 1285
- trait::graph, 1374
- trait::graph< mln::complex_image< 1, G, V > >, 1375
- trait::graph< mln::image2d< T > >, 1376
- transform
 - mln::data, 240
 - mln::data::impl::generic, 250
- transform_inplace
 - mln::data, 241
 - mln::data::impl::generic, 251
- transform_inplace_lowq
 - mln::data::impl, 246
- transformed_image
 - mln::transformed_image, 1222
- translate
 - mln::geom, 303
- translation
 - mln::fun::x2x::translation, 814
- tree
 - mln::util::tree, 1287
- tree_fast_to_image
 - mln::util, 489
 - mln::util::impl, 491
- tree_node
 - mln::util::tree_node, 1290
- tree_to_fast
 - mln::util, 490
- tree_to_image
 - mln::util, 490
- Types, 105
- uni
 - mln::Box, 652
 - mln::box, 647
 - mln::p_array, 977
 - mln::p_centered, 982
 - mln::p_complex, 988
 - mln::p_edges, 995

- mln::p_faces, [1002](#)
- mln::p_if, [1009](#)
- mln::p_image, [1015](#)
- mln::p_key, [1027](#)
- mln::p_line2d, [1033](#)
- mln::p_mutable_array_of, [1039](#)
- mln::p_priority, [1051](#)
- mln::p_queue, [1058](#)
- mln::p_queue_fast, [1065](#)
- mln::p_run, [1072](#)
- mln::p_set, [1078](#)
- mln::p_set_of, [1084](#)
- mln::p_transformed, [1089](#)
- mln::p_vaccess, [1097](#)
- mln::p_vertices, [1105](#)
- mln::Site_Set, [1147](#)
- unique
 - mln::Box, [652](#)
 - mln::box, [647](#)
 - mln::p_array, [977](#)
 - mln::p_centered, [982](#)
 - mln::p_complex, [988](#)
 - mln::p_edges, [996](#)
 - mln::p_faces, [1002](#)
 - mln::p_if, [1009](#)
 - mln::p_image, [1015](#)
 - mln::p_key, [1027](#)
 - mln::p_line2d, [1033](#)
 - mln::p_mutable_array_of, [1039](#)
 - mln::p_priority, [1051](#)
 - mln::p_queue, [1058](#)
 - mln::p_queue_fast, [1065](#)
 - mln::p_run, [1072](#)
 - mln::p_set, [1078](#)
 - mln::p_set_of, [1084](#)
 - mln::p_transformed, [1089](#)
 - mln::p_vaccess, [1097](#)
 - mln::p_vertices, [1105](#)
 - mln::Site_Set, [1148](#)
- unproject_image
 - mln::unproject_image, [1223](#)
- unsigned_2complex_image3df
 - mln, [164](#)
- untake
 - mln::morpho::attribute::sum, [961](#)
- up
 - mln, [178](#)
- update
 - mln::data, [241](#)
 - mln::data::impl::generic, [251](#)
 - mln::dpoints_bkd_pixter, [746](#)
 - mln::dpoints_fwd_pixter, [749](#)
- update_data
 - mln::labeled_image, [903](#)
 - mln::labeled_image_base, [906](#)
- update_fastest
 - mln::data::impl, [247](#)
- update_id
 - mln::util::edge, [1243](#)
 - mln::util::vertex, [1296](#)
- util_set
 - mln::p_set, [1077](#)
- util_tree
 - mln::util::branch, [1232](#)
- Utilities, [124](#)
- v
 - mln::util::pix, [1273](#)
- v1
 - mln::util::edge, [1244](#)
 - mln::util::graph, [1253](#)
 - mln::util::line_graph, [1264](#)
- v2
 - mln::util::edge, [1244](#)
 - mln::util::graph, [1253](#)
 - mln::util::line_graph, [1264](#)
- v2w2v functions, [134](#)
- v2w_w2v functions, [135](#)
- v_ith_nbh_edge
 - mln::util::graph, [1253](#)
 - mln::util::line_graph, [1264](#)
- v_ith_nbh_vertex
 - mln::util::graph, [1253](#)
 - mln::util::line_graph, [1265](#)
- v_nmax
 - mln::util::graph, [1253](#)
 - mln::util::line_graph, [1265](#)
- v_nmax_nbh_edges
 - mln::util::graph, [1253](#)
 - mln::util::line_graph, [1265](#)
- v_nmax_nbh_vertices
 - mln::util::graph, [1254](#)
 - mln::util::line_graph, [1265](#)
- v_other
 - mln::util::edge, [1244](#)
- val
 - mln::doc::Generalized_Pixel, [708](#)
 - mln::doc::Pixel_Iterator, [722](#)
- value
 - mln::accu::shape::height, [586](#)
 - mln::accu::shape::volume, [589](#)
 - mln::complex_image, [674](#)
 - mln::doc::Fastest_Image, [702](#)
 - mln::doc::Generalized_Pixel, [708](#)
 - mln::doc::Image, [712](#)
 - mln::doc::Pixel_Iterator, [721](#)
 - mln::doc::Value_Iterator, [731](#)
 - mln::doc::Value_Set, [733](#)

- mln::extended, 758
- mln::extension_fun, 761
- mln::extension_ima, 764
- mln::extension_val, 767
- mln::flat_image, 773
- mln::fun_image, 816
- mln::hexa, 870
- mln::image1d, 876
- mln::image2d, 882
- mln::image2d_h, 886
- mln::image3d, 890
- mln::interpolated, 896
- mln::labeling, 361
- mln::p_vaccess, 1095
- mln::thrubin_image, 1158
- mln::tr_image, 1219
- mln::util::pix, 1272
- mln::value::float01, 1301
- mln::value::float01_f, 1303
- mln::value::graylevel, 1306
- mln::value::graylevel_f, 1309
- mln::value::lut_vec, 1322
- mln::value::stack_image, 1333
- mln::violent_cast_image, 1346
- value_array
 - mln::value::value_array, 1336
- value_ind
 - mln::value::float01, 1301
- value_t
 - mln::util::object_id, 1268
- values
 - mln::complex_image, 675
 - mln::doc::Fastest_Image, 706
 - mln::doc::Image, 714
 - mln::p_vaccess, 1096
- Values morphers, 102
- var
 - mln::accu::stat::variance, 623
- variance
 - mln::accu::stat::var, 620
- vec
 - mln::dpoint, 740
 - mln::make, 402, 403
 - mln::point, 1118
- vec2d_d
 - mln, 164
- vec2d_f
 - mln, 164
- vec3d_d
 - mln, 164
- vec3d_f
 - mln, 164
- vect
 - mln::accu::histo, 518
- vertex
 - mln::p_vertices, 1101
 - mln::util::graph, 1254
 - mln::util::line_graph, 1265
 - mln::util::vertex, 1295
- vertex_fwd_iter
 - mln::util::graph, 1250
 - mln::util::line_graph, 1262
- vertex_id_t
 - mln::util, 487
- vertex_image
 - mln::make, 403, 404
 - mln::vertex_image, 1343
- vertex_nbh_edge_fwd_iter
 - mln::util::graph, 1250
 - mln::util::line_graph, 1262
- vertex_nbh_t
 - mln::vertex_image, 1343
- vertex_nbh_vertex_fwd_iter
 - mln::util::graph, 1250
 - mln::util::line_graph, 1262
- vertex_win_t
 - mln::vertex_image, 1343
- vertices_t
 - mln::util::graph, 1250
 - mln::util::line_graph, 1262
- violent_cast_image
 - mln::violent_cast_image, 1346
- violet
 - mln::literal, 378
- vline2d
 - modwin2d, 128
- volume
 - mln::morpho::attribute::sharpness, 958
 - mln::win::cuboid3d, 1357
- voronoi
 - mln::make, 404
- vprod
 - mln::algebra, 195
- vset
 - mln::doc::Fastest_Image, 702
 - mln::doc::Image, 712
 - mln::p_vaccess, 1095
 - mln::value::value_array, 1337
- vv2b functions, 136
- w
 - mln::w_window, 1349
- w_window
 - mln::make, 404
 - mln::w_window, 1349
- w_window1d
 - mln::make, 405
- w_window1d_float

- mln, 164
- w_window1d_int
 - mln, 165
 - mln::make, 405
- w_window2d
 - mln::make, 405
- w_window2d_float
 - mln, 165
- w_window2d_int
 - mln, 165
 - mln::make, 406
- w_window3d
 - mln::make, 406
- w_window3d_float
 - mln, 165
- w_window3d_int
 - mln, 165
 - mln::make, 406
- w_window_directional
 - mln::make, 407
- weight
 - mln::doc::Weighted_Window, 735
 - mln::w_window, 1348
- weights
 - mln::w_window, 1349
- white
 - mln::literal, 378
- width
 - mln::win::cuboid3d, 1357
 - mln::win::rectangle2d, 1366
- win
 - mln::doc::Weighted_Window, 736
 - mln::w_window, 1350
- win_c4p
 - modwin2d, 128
- win_c4p_3d
 - modwin3d, 131
- win_c8p
 - modwin2d, 128
- win_c8p_3d
 - modwin3d, 131
- win_t
 - mln::edge_image, 756
 - mln::vertex_image, 1343
- window
 - mln::doc::Weighted_Window, 735
 - mln::p_centered, 981
 - mln::window, 1370
- window1d
 - modwin1d, 126
- window2d
 - modwin2d, 128
- window3d
 - modwin3d, 130
- Windows, 125
- wrap
 - mln::data, 242
 - mln::labeling, 362
- write_header
 - mln::io::fld, 322
- xor_inplace
 - mln::logical, 381
- yellow
 - mln::literal, 378
- zero
 - mln::algebra::h_vec, 632
 - mln::literal, 378
 - mln::value::int_s, 1311
 - mln::value::int_u_sat, 1315
 - mln::value::rgb, 1328
 - mln::value::sign, 1331