

Milena (Olena)
User documentation 2.0a Id

Generated by Doxygen 1.8.2-20120930

Thu Nov 8 2012 10:58:46

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	3
2.1.1	Requirements	3
2.1.1.1	To compile the user examples	3
2.1.1.2	To compile the documentation (Optional)	3
2.1.1.3	To develop in Olena	3
2.1.2	Getting Olena	3
2.1.3	Building Olena	3
2.2	Foreword	4
2.2.1	Generality	4
2.2.2	Directory hierarchy	4
2.2.3	Writing and compiling a program with Olena	4
2.3	Site	4
2.4	Site set	4
2.4.1	Basic interface	4
2.4.2	Optional interface	4
2.5	Image	4
2.5.1	Definition	4
2.5.2	Possible image types	4
2.5.3	Possible value types	4
2.5.4	Domain	4
2.5.5	Border and extension	5
2.5.5.1	Image border	5
2.5.5.2	Generality on image extension	5
2.5.5.3	Different extensions	5
2.5.6	Interface	7

2.5.7	Load and save images	7
2.5.8	Create an image	7
2.5.9	Access and modify values	8
2.5.10	Image size	8
2.6	Structural elements: Window and neighborhood	8
2.6.1	Define an element	8
2.6.1.1	Window	8
2.6.1.2	Neighborhood	8
2.6.1.3	Custom structural elements	8
2.6.1.4	Conversion between Neighborhoods and Windows	9
2.7	Sites, psites and dpoints	9
2.7.1	Need for site	9
2.7.2	Need for psite	9
2.7.3	From psite to site	9
2.7.4	Dpoint	9
2.8	Iterators	10
2.9	Memory management	10
2.10	Basic routines	10
2.10.1	Fill	10
2.10.2	Paste	11
2.10.3	Blobs	11
2.10.4	Logical not	11
2.10.5	Compute	11
2.10.5.1	Accumulators	11
2.10.5.2	Example with labeling::compute()	12
2.10.6	Working with parts of an image	12
2.10.6.1	Restrict an image with a site set	12
2.10.6.2	Restrict an image with a predicate	13
2.10.6.3	Restrict an image with a C function	13
2.11	Input / Output	14
2.11.1	ImageMagick	14
2.11.2	GDCM	14
2.12	Graphs and images	14
2.12.1	Description	14
2.12.2	Example	14
2.13	Useful global variables	16
2.14	Useful macros	16
2.14.1	Variable declaration macros	16
2.14.2	Iterator type macros	16
2.14.2.1	Default iterator types	16

2.14.2.2	Forward iterator types	16
2.14.2.3	Backward iterators	16
2.14.2.4	Graph iterators	16
2.15	Common Compilation Errors	16
3	Tutorial	17
3.1	Welcome	17
3.1.1	How to learn Milena	17
3.1.2	Obtaining the library	17
3.1.3	Downloading the library	17
3.1.3.1	Downloading from SVN	17
3.1.3.2	Downloading packaged releases	17
3.1.4	Join the mailing lists	17
3.1.5	Directory structure	17
3.1.6	Documentation	17
3.1.7	Community and Support	17
3.1.8	Project status	18
3.1.9	A brief history of Milena	18
3.1.10	Contacts	18
3.2	Installation	18
3.2.1	Bootstrap (SVN Sources)	18
3.2.2	Configure	18
3.2.3	Install	18
3.2.4	Optional compilation	18
3.2.4.1	Examples	18
3.2.4.2	Tools	18
3.2.4.3	Tests	18
3.2.5	Installation content	18
3.3	Getting started with Milena	18
3.3.1	Getting familiar with genericity	18
3.3.2	First generic algorithm	19
3.3.3	Compilation	20
3.3.3.1	Include path	20
3.3.3.2	Library linking	20
3.3.3.3	Disable Debug	20
3.3.3.4	Compiler optimization flags	20
3.3.4	Debug hints	20
3.3.4.1	Using assertions and GDB	20
3.3.4.2	Traces	20
3.3.4.3	Debug routines	20

3.4	Data representation	21
3.4.1	Sites	21
3.4.2	Site sets	21
3.4.2.1	Creating a site set	21
3.4.2.2	Getting access to sites	22
3.4.3	Images	22
3.4.3.1	Creating an image	22
3.4.3.2	Reading an image from a file	22
3.4.3.3	Accessing data	22
3.5	Load and save images	22
3.6	Create your first image	22
3.7	Read and write images	23
3.8	Regions of interest	23
3.8.1	Image domain restricted by a site set	23
3.8.2	Image domain restricted by a function	23
3.8.3	Image domain restricted by a mask	24
3.8.4	Image domain restricted by a predicate	24
4	Module Index	25
4.1	Modules	25
5	Namespace Index	27
5.1	Namespace List	27
6	Hierarchical Index	33
6.1	Class Hierarchy	33
7	Class Index	73
7.1	Class List	73
8	Module Documentation	91
8.1	On site sets	91
8.1.1	Detailed Description	91
8.2	On images	92
8.2.1	Detailed Description	92
8.3	On values	93
8.3.1	Detailed Description	94
8.4	Multiple accumulators	95
8.4.1	Detailed Description	95
8.5	Graphes	96
8.5.1	Detailed Description	96
8.6	Images	97

8.6.1 Detailed Description	97
8.7 Basic types	98
8.7.1 Detailed Description	98
8.8 Image morphers	99
8.9 Values morphers	100
8.9.1 Detailed Description	100
8.10 Domain morphers	101
8.10.1 Detailed Description	101
8.11 Identity morphers	102
8.11.1 Detailed Description	102
8.12 Types	103
8.12.1 Detailed Description	103
8.13 Accumulators	104
8.13.1 Detailed Description	104
8.14 Routines	105
8.15 Canvas	106
8.16 Functions	107
8.16.1 Detailed Description	107
8.17 Neighborhoods	108
8.17.1 Detailed Description	108
8.18 1D neighborhoods	109
8.18.1 Detailed Description	109
8.18.2 Typedef Documentation	109
8.18.2.1 neighb1d	109
8.18.3 Function Documentation	109
8.18.3.1 c2	109
8.19 2D neighborhoods	110
8.19.1 Detailed Description	110
8.19.2 Typedef Documentation	110
8.19.2.1 neighb2d	110
8.19.3 Function Documentation	110
8.19.3.1 c2_col	110
8.19.3.2 c2_row	111
8.19.3.3 c4	111
8.19.3.4 c8	111
8.20 3D neighborhoods	112
8.20.1 Detailed Description	112
8.20.2 Typedef Documentation	112
8.20.2.1 neighb3d	112
8.20.3 Function Documentation	112

8.20.3.1	c18	112
8.20.3.2	c26	113
8.20.3.3	c2_3d_sli	113
8.20.3.4	c4_3d	113
8.20.3.5	c6	114
8.20.3.6	c8_3d	114
8.21	Site sets	115
8.21.1	Detailed Description	115
8.22	Basic types	116
8.22.1	Detailed Description	116
8.23	Graph based	117
8.23.1	Detailed Description	117
8.24	Complex based	118
8.24.1	Detailed Description	118
8.25	Sparse types	119
8.25.1	Detailed Description	119
8.26	Queue based	120
8.26.1	Detailed Description	120
8.27	Utilities	121
8.27.1	Detailed Description	121
8.28	Windows	122
8.28.1	Detailed Description	122
8.29	1D windows	123
8.29.1	Detailed Description	123
8.29.2	Typedef Documentation	123
8.29.2.1	segment1d	123
8.29.2.2	window1d	123
8.30	2D windows	124
8.30.1	Detailed Description	124
8.30.2	Typedef Documentation	124
8.30.2.1	disk2d	124
8.30.2.2	hline2d	125
8.30.2.3	vline2d	125
8.30.2.4	window2d	125
8.30.3	Function Documentation	125
8.30.3.1	win_c4p	125
8.30.3.2	win_c8p	125
8.31	3D windows	127
8.31.1	Detailed Description	127
8.31.2	Typedef Documentation	127

8.31.2.1	sline3d	127
8.31.2.2	sphere3d	128
8.31.2.3	window3d	128
8.31.3	Function Documentation	128
8.31.3.1	win_c4p_3d	128
8.31.3.2	win_c8p_3d	128
8.32	N-D windows	130
8.32.1	Detailed Description	130
8.33	Multiple windows	131
8.33.1	Detailed Description	131
8.34	v2w2v functions	132
8.35	v2w_w2v functions	133
8.36	vv2b functions	134
9	Namespace Documentation	135
9.1	mln Namespace Reference	135
9.1.1	Detailed Description	149
9.1.2	Typedef Documentation	151
9.1.2.1	bin_1complex_image2d	152
9.1.2.2	bin_2complex_image3df	152
9.1.2.3	box1d	152
9.1.2.4	box2d	152
9.1.2.5	box2d_h	152
9.1.2.6	box3d	152
9.1.2.7	discrete_plane_1complex_geometry	152
9.1.2.8	discrete_plane_2complex_geometry	153
9.1.2.9	dpoint1d	153
9.1.2.10	dpoint2d	153
9.1.2.11	dpoint2d_h	153
9.1.2.12	dpoint3d	153
9.1.2.13	float_2complex_image3df	153
9.1.2.14	int_u8_1complex_image2d	153
9.1.2.15	int_u8_2complex_image2d	153
9.1.2.16	int_u8_2complex_image3df	154
9.1.2.17	p_run2d	154
9.1.2.18	p_runs2d	154
9.1.2.19	point1d	154
9.1.2.20	point1df	154
9.1.2.21	point2d	154
9.1.2.22	point2d_h	154

9.1.2.23	point2df	154
9.1.2.24	point3d	154
9.1.2.25	point3df	155
9.1.2.26	rgb8_2complex_image3df	155
9.1.2.27	space_2complex_geometry	155
9.1.2.28	unsigned_2complex_image3df	155
9.1.2.29	vec2d_d	155
9.1.2.30	vec2d_f	155
9.1.2.31	vec3d_d	155
9.1.2.32	vec3d_f	155
9.1.3	Function Documentation	155
9.1.3.1	a_point_of	156
9.1.3.2	apply_p2p	156
9.1.3.3	apply_p2p	156
9.1.3.4	compose	156
9.1.3.5	duplicate	156
9.1.3.6	extend	157
9.1.3.7	extend	157
9.1.3.8	extend	157
9.1.3.9	implies	157
9.1.3.10	initialize	157
9.1.3.11	larger_than	158
9.1.3.12	make_debug_graph_image	158
9.1.3.13	mln_exact	158
9.1.3.14	mln_gen_complex_neighborhood	158
9.1.3.15	mln_gen_complex_neighborhood	158
9.1.3.16	mln_gen_complex_neighborhood	158
9.1.3.17	mln_gen_complex_neighborhood	158
9.1.3.18	mln_gen_complex_neighborhood	158
9.1.3.19	mln_gen_complex_neighborhood	159
9.1.3.20	mln_gen_complex_window	159
9.1.3.21	mln_gen_complex_window	159
9.1.3.22	mln_gen_complex_window	159
9.1.3.23	mln_gen_complex_window	159
9.1.3.24	mln_gen_complex_window	159
9.1.3.25	mln_gen_complex_window	159
9.1.3.26	mln_gen_complex_window_p	159
9.1.3.27	mln_gen_complex_window_p	159
9.1.3.28	mln_gen_complex_window_p	159
9.1.3.29	mln_gen_complex_window_p	160

9.1.3.30	min_gen_complex_window_p	160
9.1.3.31	min_gen_complex_window_p	160
9.1.3.32	min_regular	160
9.1.3.33	min_trait_op_neq	160
9.1.3.34	operator!=	160
9.1.3.35	operator!=	161
9.1.3.36	operator*	161
9.1.3.37	operator++	161
9.1.3.38	operator-	161
9.1.3.39	operator-	161
9.1.3.40	operator--	162
9.1.3.41	operator<	162
9.1.3.42	operator<	162
9.1.3.43	operator<	162
9.1.3.44	operator<<	163
9.1.3.45	operator<<	163
9.1.3.46	operator<<	163
9.1.3.47	operator<<	163
9.1.3.48	operator<=	163
9.1.3.49	operator<=	163
9.1.3.50	operator<=	164
9.1.3.51	operator<=	164
9.1.3.52	operator<=	164
9.1.3.53	operator==	164
9.1.3.54	operator==	164
9.1.3.55	operator==	165
9.1.3.56	operator==	165
9.1.3.57	operator==	165
9.1.3.58	operator==	165
9.1.3.59	operator==	165
9.1.3.60	operator 	166
9.1.3.61	operator 	166
9.1.3.62	operator 	166
9.1.3.63	operator 	166
9.1.3.64	operator 	166
9.1.3.65	operator 	166
9.1.3.66	primary	166
9.1.3.67	ptransform	167
9.1.4	Variable Documentation	167
9.1.4.1	before	167

9.1.4.2	sagittal_dec	167
9.1.4.3	up	167
9.2	mln::accu Namespace Reference	167
9.2.1	Detailed Description	168
9.2.2	Function Documentation	169
9.2.2.1	compute	169
9.2.2.2	line	169
9.2.2.3	mln_meta_accu_result	169
9.2.2.4	take	170
9.3	mln::accu::image Namespace Reference	170
9.3.1	Detailed Description	170
9.4	mln::accu::impl Namespace Reference	170
9.4.1	Detailed Description	170
9.5	mln::accu::logic Namespace Reference	170
9.5.1	Detailed Description	171
9.6	mln::accu::math Namespace Reference	171
9.6.1	Detailed Description	171
9.7	mln::accu::meta::logic Namespace Reference	171
9.7.1	Detailed Description	171
9.8	mln::accu::meta::math Namespace Reference	172
9.8.1	Detailed Description	172
9.9	mln::accu::meta::shape Namespace Reference	172
9.9.1	Detailed Description	172
9.10	mln::accu::meta::stat Namespace Reference	172
9.10.1	Detailed Description	173
9.11	mln::accu::shape Namespace Reference	173
9.11.1	Detailed Description	173
9.12	mln::accu::stat Namespace Reference	173
9.12.1	Detailed Description	174
9.12.2	Function Documentation	174
9.12.2.1	operator==	174
9.13	mln::algebra Namespace Reference	175
9.13.1	Detailed Description	175
9.13.2	Function Documentation	176
9.13.2.1	det	176
9.13.2.2	det	176
9.13.2.3	ldlt_decomp	176
9.13.2.4	ldlt_solve	176
9.13.2.5	operator*	176
9.13.2.6	tr	176

9.13.2.7	vprod	176
9.14	mln::arith Namespace Reference	177
9.14.1	Detailed Description	178
9.14.2	Function Documentation	179
9.14.2.1	diff_abs	179
9.14.2.2	div	179
9.14.2.3	div_cst	179
9.14.2.4	div_inplace	180
9.14.2.5	min	180
9.14.2.6	min_inplace	180
9.14.2.7	minus	181
9.14.2.8	minus	181
9.14.2.9	minus	181
9.14.2.10	minus_cst	182
9.14.2.11	minus_cst	182
9.14.2.12	minus_cst_inplace	183
9.14.2.13	minus_inplace	183
9.14.2.14	plus	183
9.14.2.15	plus	184
9.14.2.16	plus	184
9.14.2.17	plus_cst	184
9.14.2.18	plus_cst	185
9.14.2.19	plus_cst	185
9.14.2.20	plus_cst_inplace	186
9.14.2.21	plus_inplace	186
9.14.2.22	revert	186
9.14.2.23	revert_inplace	187
9.14.2.24	times	187
9.14.2.25	times_cst	187
9.14.2.26	times_inplace	188
9.15	mln::arith::impl Namespace Reference	188
9.15.1	Detailed Description	188
9.16	mln::arith::impl::generic Namespace Reference	189
9.16.1	Detailed Description	189
9.17	mln::binarization Namespace Reference	189
9.17.1	Detailed Description	189
9.17.2	Function Documentation	189
9.17.2.1	binarization	189
9.17.2.2	threshold	189
9.18	mln::border Namespace Reference	190

9.18.1 Detailed Description	190
9.18.2 Function Documentation	190
9.18.2.1 adjust	190
9.18.2.2 duplicate	191
9.18.2.3 equalize	191
9.18.2.4 fill	191
9.18.2.5 find	192
9.18.2.6 get	192
9.18.2.7 mirror	192
9.18.2.8 resize	192
9.19 mln::border::impl Namespace Reference	193
9.19.1 Detailed Description	193
9.20 mln::border::impl::generic Namespace Reference	193
9.20.1 Detailed Description	193
9.21 mln::canvas Namespace Reference	193
9.21.1 Detailed Description	194
9.21.2 Function Documentation	194
9.21.2.1 distance_front	194
9.21.2.2 distance_geodesic	194
9.22 mln::canvas::browsing Namespace Reference	194
9.22.1 Detailed Description	195
9.23 mln::canvas::impl Namespace Reference	195
9.23.1 Detailed Description	195
9.24 mln::canvas::labeling Namespace Reference	195
9.24.1 Detailed Description	196
9.24.2 Function Documentation	196
9.24.2.1 blobs	196
9.25 mln::canvas::labeling::impl Namespace Reference	196
9.25.1 Detailed Description	196
9.26 mln::canvas::morpho Namespace Reference	196
9.26.1 Detailed Description	196
9.27 mln::convert Namespace Reference	196
9.27.1 Detailed Description	198
9.27.2 Function Documentation	198
9.27.2.1 from_to	198
9.27.2.2 from_to	198
9.27.2.3 from_to	199
9.27.2.4 from_to	199
9.27.2.5 mln_image_from_grid	199
9.27.2.6 mln_image_from_grid	199

9.27.2.7	mln_image_from_grid	199
9.27.2.8	mln_image_from_grid	199
9.27.2.9	mln_window	199
9.27.2.10	to	199
9.27.2.11	to_dpoint	200
9.27.2.12	to_fun	200
9.27.2.13	to_image	200
9.27.2.14	to_p_array	200
9.27.2.15	to_p_array	200
9.27.2.16	to_p_array	200
9.27.2.17	to_p_set	200
9.27.2.18	to_p_set	201
9.27.2.19	to_p_set	201
9.27.2.20	to_p_set	201
9.27.2.21	to_p_set	201
9.27.2.22	to_qimage	201
9.27.2.23	to_upper_window	201
9.27.2.24	to_upper_window	201
9.27.2.25	to_window	202
9.27.2.26	to_window	202
9.27.2.27	to_window	202
9.27.3	Variable Documentation	202
9.27.3.1	to_fun	202
9.28	mln::data Namespace Reference	202
9.28.1	Detailed Description	204
9.28.2	Function Documentation	204
9.28.2.1	abs	204
9.28.2.2	abs_inplace	204
9.28.2.3	apply	204
9.28.2.4	compute	205
9.28.2.5	compute	205
9.28.2.6	convert	205
9.28.2.7	fast_median	205
9.28.2.8	fill	206
9.28.2.9	fill_with_image	206
9.28.2.10	fill_with_value	206
9.28.2.11	median	207
9.28.2.12	mln_meta_accu_result	207
9.28.2.13	paste	207
9.28.2.14	paste_without_localization	208

9.28.2.15	replace	208
9.28.2.16	saturate	208
9.28.2.17	saturate	208
9.28.2.18	saturate_inplace	209
9.28.2.19	sort_offsets_increasing	209
9.28.2.20	sort_psites_decreasing	209
9.28.2.21	sort_psites_increasing	209
9.28.2.22	stretch	210
9.28.2.23	to_enc	210
9.28.2.24	transform	210
9.28.2.25	transform	210
9.28.2.26	transform_inplace	211
9.28.2.27	transform_inplace	211
9.28.2.28	update	211
9.28.2.29	wrap	211
9.29	mln::data::approx Namespace Reference	212
9.29.1	Detailed Description	212
9.29.2	Function Documentation	212
9.29.2.1	median	212
9.29.2.2	median	213
9.29.2.3	median	213
9.30	mln::data::approx::impl Namespace Reference	213
9.30.1	Detailed Description	213
9.31	mln::data::impl Namespace Reference	213
9.31.1	Detailed Description	214
9.31.2	Function Documentation	214
9.31.2.1	paste_without_localization_fast	214
9.31.2.2	paste_without_localization_fastest	215
9.31.2.3	paste_without_localization_lines	215
9.31.2.4	stretch	215
9.31.2.5	transform_inplace_lowq	216
9.31.2.6	update_fastest	216
9.32	mln::data::impl::generic Namespace Reference	216
9.32.1	Detailed Description	217
9.32.2	Function Documentation	217
9.32.2.1	fill_with_image	217
9.32.2.2	fill_with_value	217
9.32.2.3	paste	217
9.32.2.4	transform	218
9.32.2.5	transform	218

9.32.2.6	transform_inplace	218
9.32.2.7	transform_inplace	218
9.32.2.8	update	219
9.33	mln::data::naive Namespace Reference	219
9.33.1	Detailed Description	219
9.33.2	Function Documentation	219
9.33.2.1	median	219
9.34	mln::data::naive::impl Namespace Reference	220
9.34.1	Detailed Description	220
9.35	mln::debug Namespace Reference	220
9.35.1	Detailed Description	221
9.35.2	Function Documentation	221
9.35.2.1	draw_graph	221
9.35.2.2	draw_graph	222
9.35.2.3	draw_graph	222
9.35.2.4	filename	222
9.35.2.5	format	222
9.35.2.6	format	222
9.35.2.7	format	222
9.35.2.8	format	223
9.35.2.9	iota	223
9.35.2.10	mosaic	223
9.35.2.11	println	223
9.35.2.12	println	223
9.35.2.13	println_with_border	223
9.35.2.14	put_word	224
9.35.2.15	slices_2d	224
9.35.2.16	slices_2d	224
9.35.2.17	superpose	224
9.35.2.18	superpose	224
9.35.2.19	z_order	225
9.36	mln::debug::impl Namespace Reference	225
9.36.1	Detailed Description	225
9.37	mln::def Namespace Reference	225
9.37.1	Detailed Description	225
9.37.2	Typedef Documentation	225
9.37.2.1	coord	225
9.37.2.2	coordf	226
9.37.3	Enumeration Type Documentation	226
9.37.3.1	anonymous enum	226

9.38	mln::display Namespace Reference	226
9.38.1	Detailed Description	226
9.39	mln::display::impl Namespace Reference	226
9.39.1	Detailed Description	226
9.40	mln::display::impl::generic Namespace Reference	226
9.40.1	Detailed Description	226
9.41	mln::doc Namespace Reference	227
9.41.1	Detailed Description	227
9.42	mln::draw Namespace Reference	228
9.42.1	Detailed Description	228
9.42.2	Function Documentation	228
9.42.2.1	box	228
9.42.2.2	box_plain	228
9.42.2.3	dashed_line	229
9.42.2.4	line	229
9.42.2.5	plot	229
9.42.2.6	polygon	230
9.42.2.7	site_set	230
9.43	mln::estim Namespace Reference	230
9.43.1	Detailed Description	231
9.43.2	Function Documentation	231
9.43.2.1	mean	231
9.43.2.2	mean	231
9.43.2.3	min_max	231
9.43.2.4	sum	232
9.43.2.5	sum	232
9.44	mln::extension Namespace Reference	232
9.44.1	Detailed Description	233
9.44.2	Function Documentation	233
9.44.2.1	adjust	233
9.44.2.2	adjust	233
9.44.2.3	adjust	233
9.44.2.4	adjust	233
9.44.2.5	adjust_duplicate	233
9.44.2.6	adjust_fill	234
9.44.2.7	duplicate	234
9.44.2.8	fill	234
9.45	mln::fun Namespace Reference	234
9.45.1	Detailed Description	235
9.46	mln::fun::access Namespace Reference	235

9.46.1 Detailed Description	235
9.47 mln::fun::i2v Namespace Reference	235
9.47.1 Detailed Description	236
9.47.2 Function Documentation	236
9.47.2.1 operator<<	236
9.48 mln::fun::n2v Namespace Reference	236
9.48.1 Detailed Description	236
9.49 mln::fun::p2b Namespace Reference	236
9.49.1 Detailed Description	236
9.50 mln::fun::p2p Namespace Reference	237
9.50.1 Detailed Description	237
9.51 mln::fun::p2v Namespace Reference	237
9.51.1 Detailed Description	237
9.52 mln::fun::stat Namespace Reference	237
9.52.1 Detailed Description	237
9.53 mln::fun::v2b Namespace Reference	237
9.53.1 Detailed Description	237
9.54 mln::fun::v2i Namespace Reference	237
9.54.1 Detailed Description	237
9.55 mln::fun::v2v Namespace Reference	238
9.55.1 Detailed Description	238
9.55.2 Variable Documentation	238
9.55.2.1 f_hsi_to_rgb_3x8	238
9.55.2.2 f_hsl_to_rgb_3x8	238
9.55.2.3 f_rgb_to_hsi_f	239
9.55.2.4 f_rgb_to_hsl_f	239
9.56 mln::fun::v2w2v Namespace Reference	239
9.56.1 Detailed Description	239
9.57 mln::fun::v2w_w2v Namespace Reference	239
9.57.1 Detailed Description	239
9.58 mln::fun::vv2b Namespace Reference	239
9.58.1 Detailed Description	240
9.59 mln::fun::vv2v Namespace Reference	240
9.59.1 Detailed Description	240
9.60 mln::fun::x2p Namespace Reference	240
9.60.1 Detailed Description	241
9.61 mln::fun::x2v Namespace Reference	241
9.61.1 Detailed Description	241
9.62 mln::fun::x2x Namespace Reference	241
9.62.1 Detailed Description	241

9.63	mln::geom Namespace Reference	241
9.63.1	Detailed Description	245
9.63.2	Function Documentation	245
9.63.2.1	bbox	245
9.63.2.2	bbox	245
9.63.2.3	bbox	245
9.63.2.4	bbox	245
9.63.2.5	chamfer	245
9.63.2.6	delta	245
9.63.2.7	delta	245
9.63.2.8	delta	246
9.63.2.9	max_col	246
9.63.2.10	max_col	246
9.63.2.11	max_ind	246
9.63.2.12	max_row	246
9.63.2.13	max_row	246
9.63.2.14	max_sli	246
9.63.2.15	mesh_corner_point_area	247
9.63.2.16	mesh_curvature	247
9.63.2.17	mesh_normal	247
9.63.2.18	min_col	248
9.63.2.19	min_col	248
9.63.2.20	min_ind	248
9.63.2.21	min_row	248
9.63.2.22	min_row	248
9.63.2.23	min_sli	248
9.63.2.24	ncols	248
9.63.2.25	ncols	249
9.63.2.26	ninds	249
9.63.2.27	nrows	249
9.63.2.28	nrows	249
9.63.2.29	nsites	249
9.63.2.30	nslis	249
9.63.2.31	pmin_pmax	249
9.63.2.32	pmin_pmax	250
9.63.2.33	pmin_pmax	250
9.63.2.34	pmin_pmax	250
9.63.2.35	rotate	250
9.63.2.36	rotate	250
9.63.2.37	rotate	251

9.63.2.38 rotate	251
9.63.2.39 rotate	251
9.63.2.40 seeds2tiling	251
9.63.2.41 translate	251
9.63.2.42 translate	252
9.63.2.43 translate	252
9.63.2.44 vertical_symmetry	252
9.64 mln::geom::impl Namespace Reference	252
9.64.1 Detailed Description	252
9.64.2 Function Documentation	253
9.64.2.1 seeds2tiling	253
9.65 mln::graph Namespace Reference	253
9.65.1 Detailed Description	253
9.65.2 Function Documentation	253
9.65.2.1 compute	253
9.65.2.2 labeling	254
9.65.2.3 to_neighb	254
9.65.2.4 to_win	255
9.66 mln::grid Namespace Reference	255
9.66.1 Detailed Description	255
9.67 mln::histo Namespace Reference	255
9.67.1 Detailed Description	256
9.67.2 Function Documentation	256
9.67.2.1 compute	256
9.67.2.2 equalize	256
9.68 mln::histo::impl Namespace Reference	256
9.68.1 Detailed Description	256
9.69 mln::histo::impl::generic Namespace Reference	257
9.69.1 Detailed Description	257
9.70 mln::impl Namespace Reference	257
9.70.1 Detailed Description	257
9.71 mln::io Namespace Reference	257
9.71.1 Detailed Description	258
9.72 mln::io::cloud Namespace Reference	258
9.72.1 Detailed Description	258
9.72.2 Function Documentation	258
9.72.2.1 load	258
9.72.2.2 save	259
9.73 mln::io::dicom Namespace Reference	259
9.73.1 Detailed Description	259

9.73.2	Function Documentation	259
9.73.2.1	get_header	259
9.73.2.2	load	259
9.74	mln::io::dump Namespace Reference	260
9.74.1	Detailed Description	260
9.74.2	Function Documentation	260
9.74.2.1	get_header	260
9.74.2.2	load	260
9.74.2.3	save	260
9.75	mln::io::fits Namespace Reference	261
9.75.1	Detailed Description	261
9.75.2	Function Documentation	261
9.75.2.1	load	261
9.75.2.2	load	261
9.76	mln::io::fld Namespace Reference	261
9.76.1	Detailed Description	262
9.76.2	Function Documentation	262
9.76.2.1	load	262
9.76.2.2	read_header	262
9.76.2.3	write_header	262
9.77	mln::io::magick Namespace Reference	263
9.77.1	Detailed Description	263
9.77.2	Function Documentation	263
9.77.2.1	load	263
9.77.2.2	save	263
9.77.2.3	save	264
9.78	mln::io::off Namespace Reference	264
9.78.1	Detailed Description	264
9.78.2	Function Documentation	264
9.78.2.1	load	264
9.78.2.2	save	265
9.78.2.3	save_bin_alt	265
9.79	mln::io::pbm Namespace Reference	265
9.79.1	Detailed Description	266
9.79.2	Function Documentation	266
9.79.2.1	load	266
9.79.2.2	load	266
9.79.2.3	save	266
9.80	mln::io::pbm::impl Namespace Reference	266
9.80.1	Detailed Description	266

9.81	mln::io::pbms Namespace Reference	267
9.81.1	Detailed Description	267
9.81.2	Function Documentation	267
9.81.2.1	load	267
9.82	mln::io::pbms::impl Namespace Reference	267
9.82.1	Detailed Description	267
9.83	mln::io::pfm Namespace Reference	267
9.83.1	Detailed Description	268
9.83.2	Function Documentation	268
9.83.2.1	load	268
9.83.2.2	load	268
9.83.2.3	save	268
9.84	mln::io::pfm::impl Namespace Reference	269
9.84.1	Detailed Description	269
9.85	mln::io::pgm Namespace Reference	269
9.85.1	Detailed Description	269
9.85.2	Function Documentation	269
9.85.2.1	load	269
9.85.2.2	load	269
9.85.2.3	save	270
9.86	mln::io::pgms Namespace Reference	270
9.86.1	Detailed Description	270
9.86.2	Function Documentation	270
9.86.2.1	load	270
9.87	mln::io::plot Namespace Reference	270
9.87.1	Detailed Description	271
9.87.2	Function Documentation	271
9.87.2.1	load	271
9.87.2.2	save	271
9.87.2.3	save	271
9.87.2.4	save	272
9.88	mln::io::pnm Namespace Reference	272
9.88.1	Detailed Description	272
9.88.2	Function Documentation	272
9.88.2.1	load	272
9.88.2.2	load	273
9.88.2.3	load_raw_2d	273
9.88.2.4	max_component	273
9.88.2.5	save	273
9.89	mln::io::pnm::impl Namespace Reference	273

9.89.1 Detailed Description	273
9.90 mln::io::pnms Namespace Reference	273
9.90.1 Detailed Description	274
9.90.2 Function Documentation	274
9.90.2.1 load	274
9.90.2.2 load	274
9.91 mln::io::ppm Namespace Reference	274
9.91.1 Detailed Description	275
9.91.2 Function Documentation	275
9.91.2.1 load	275
9.91.2.2 load	275
9.91.2.3 save	275
9.92 mln::io::ppms Namespace Reference	275
9.92.1 Detailed Description	276
9.92.2 Function Documentation	276
9.92.2.1 load	276
9.93 mln::io::raw Namespace Reference	276
9.93.1 Detailed Description	276
9.93.2 Function Documentation	276
9.93.2.1 get_header	276
9.93.2.2 load	277
9.93.2.3 save	277
9.94 mln::io::tiff Namespace Reference	277
9.94.1 Detailed Description	277
9.94.2 Function Documentation	277
9.94.2.1 load	277
9.95 mln::io::txt Namespace Reference	277
9.95.1 Detailed Description	278
9.95.2 Function Documentation	278
9.95.2.1 save	278
9.96 mln::labeling Namespace Reference	278
9.96.1 Detailed Description	281
9.96.2 Function Documentation	281
9.96.2.1 background	281
9.96.2.2 blobs	281
9.96.2.3 blobs_and_compute	282
9.96.2.4 colorize	282
9.96.2.5 colorize	283
9.96.2.6 colorize	283
9.96.2.7 compute	283

9.96.2.8	compute	283
9.96.2.9	compute	284
9.96.2.10	compute	284
9.96.2.11	compute	284
9.96.2.12	compute_image	285
9.96.2.13	compute_image	285
9.96.2.14	compute_image	285
9.96.2.15	compute_in_window	286
9.96.2.16	fill_holes	286
9.96.2.17	flat_zones	286
9.96.2.18	foreground	287
9.96.2.19	pack	287
9.96.2.20	pack	288
9.96.2.21	pack_inplace	288
9.96.2.22	pack_inplace	288
9.96.2.23	regional_maxima	288
9.96.2.24	regional_minima	289
9.96.2.25	relabel	289
9.96.2.26	relabel	289
9.96.2.27	relabel_inplace	290
9.96.2.28	relabel_inplace	290
9.96.2.29	superpose	290
9.96.2.30	value	291
9.96.2.31	value_and_compute	291
9.96.2.32	wrap	291
9.96.2.33	wrap	292
9.97	mln::labeling::impl Namespace Reference	292
9.97.1	Detailed Description	293
9.97.2	Function Documentation	293
9.97.2.1	compute_fastest	293
9.97.2.2	compute_fastest	293
9.98	mln::labeling::impl::generic Namespace Reference	293
9.98.1	Detailed Description	294
9.98.2	Function Documentation	294
9.98.2.1	compute	294
9.98.2.2	compute	294
9.98.2.3	compute	295
9.98.2.4	compute	295
9.99	mln::linear Namespace Reference	295
9.99.1	Detailed Description	296

9.99.2	Function Documentation	296
9.99.2.1	gaussian	296
9.99.2.2	gaussian	297
9.99.2.3	gaussian_1st_derivative	297
9.99.2.4	gaussian_2nd_derivative	297
9.99.2.5	gaussian_2nd_derivative	297
9.99.2.6	mln_ch_convolve	298
9.99.2.7	mln_ch_convolve	298
9.99.2.8	mln_ch_convolve	298
9.99.2.9	mln_ch_convolve	298
9.99.2.10	mln_ch_convolve_grad	298
9.100	mln::linear::impl Namespace Reference	299
9.100.1	Detailed Description	299
9.101	mln::linear::local Namespace Reference	299
9.101.1	Detailed Description	299
9.101.2	Function Documentation	299
9.101.2.1	convolve	299
9.101.2.2	convolve	299
9.102	mln::linear::local::impl Namespace Reference	300
9.102.1	Detailed Description	300
9.103	mln::literal Namespace Reference	300
9.103.1	Detailed Description	302
9.103.2	Variable Documentation	302
9.103.2.1	black	302
9.103.2.2	blue	302
9.103.2.3	brown	302
9.103.2.4	cyan	302
9.103.2.5	dark_gray	303
9.103.2.6	green	303
9.103.2.7	identity	303
9.103.2.8	light_gray	303
9.103.2.9	lime	303
9.103.2.10	magenta	303
9.103.2.11	max	303
9.103.2.12	medium_gray	303
9.103.2.13	min	303
9.103.2.14	olive	304
9.103.2.15	one	304
9.103.2.16	orange	304
9.103.2.17	origin	304

9.103.2.18pink	304
9.103.2.19purple	304
9.103.2.20red	304
9.103.2.21teal	304
9.103.2.22violet	305
9.103.2.23white	305
9.103.2.24yellow	305
9.103.2.25zero	305
9.104mln::logical Namespace Reference	305
9.104.1 Detailed Description	306
9.104.2 Function Documentation	306
9.104.2.1 and_inplace	306
9.104.2.2 and_not	306
9.104.2.3 and_not_inplace	306
9.104.2.4 not_inplace	307
9.104.2.5 or_inplace	307
9.104.2.6 xor_inplace	307
9.105mln::logical::impl Namespace Reference	307
9.105.1 Detailed Description	308
9.106mln::logical::impl::generic Namespace Reference	308
9.106.1 Detailed Description	308
9.107mln::make Namespace Reference	308
9.107.1 Detailed Description	312
9.107.2 Function Documentation	312
9.107.2.1 attachment	312
9.107.2.2 box1d	312
9.107.2.3 box1d	312
9.107.2.4 box2d	313
9.107.2.5 box2d	313
9.107.2.6 box2d_h	314
9.107.2.7 box2d_h	314
9.107.2.8 box3d	314
9.107.2.9 box3d	315
9.107.2.10cell	315
9.107.2.11couple	316
9.107.2.12detachment	316
9.107.2.13dpoint2d_h	316
9.107.2.14dummy_p_edges	317
9.107.2.15dummy_p_edges	317
9.107.2.16dummy_p_vertices	317

9.107.2.17dummy_p_vertices	317
9.107.2.18edge_image	318
9.107.2.19edge_image	318
9.107.2.20edge_image	318
9.107.2.21edge_image	319
9.107.2.22edge_image	319
9.107.2.23edge_image	319
9.107.2.24h_mat	320
9.107.2.25image	320
9.107.2.26image	320
9.107.2.27image	320
9.107.2.28image2d	321
9.107.2.29image3d	321
9.107.2.30image3d	321
9.107.2.31influence_zone_adjacency_graph	321
9.107.2.32mat	322
9.107.2.33ord_pair	322
9.107.2.34p_edges_with_mass_centers	322
9.107.2.35p_vertices_with_mass_centers	322
9.107.2.36pix	323
9.107.2.37pixel	323
9.107.2.38pixel	323
9.107.2.39point2d_h	323
9.107.2.40rag_and_labeled_wsl	324
9.107.2.41region_adjacency_graph	324
9.107.2.42relabelfun	324
9.107.2.43relabelfun	325
9.107.2.44vec	325
9.107.2.45vec	326
9.107.2.46vec	326
9.107.2.47vec	326
9.107.2.48vertex_image	327
9.107.2.49vertex_image	327
9.107.2.50voronoi	327
9.107.2.51w_window	327
9.107.2.52w_window1d	328
9.107.2.53w_window2d	328
9.107.2.54w_window3d	328
9.107.2.55w_window_directional	329
9.108mln::math Namespace Reference	329

9.108.1 Detailed Description	329
9.108.2 Function Documentation	330
9.108.2.1 abs	330
9.108.2.2 abs	330
9.108.2.3 abs	330
9.109mln::metal Namespace Reference	330
9.109.1 Detailed Description	331
9.110mln::metal::impl Namespace Reference	331
9.110.1 Detailed Description	331
9.111mln::metal::math Namespace Reference	331
9.111.1 Detailed Description	331
9.112mln::metal::math::impl Namespace Reference	331
9.112.1 Detailed Description	331
9.113mln::morpho Namespace Reference	331
9.113.1 Detailed Description	334
9.113.2 Function Documentation	334
9.113.2.1 complementation	334
9.113.2.2 complementation_inplace	334
9.113.2.3 contrast	334
9.113.2.4 dilation	334
9.113.2.5 erosion	334
9.113.2.6 erosion_tolerant	335
9.113.2.7 general	335
9.113.2.8 gradient	335
9.113.2.9 gradient_external	335
9.113.2.10gradient_internal	335
9.113.2.11hit_or_miss	335
9.113.2.12hit_or_miss_background_closing	336
9.113.2.13hit_or_miss_background_opening	336
9.113.2.14hit_or_miss_closing	336
9.113.2.15hit_or_miss_opening	336
9.113.2.16laplacian	336
9.113.2.17line_gradient	337
9.113.2.18meyer_wst	337
9.113.2.19meyer_wst	337
9.113.2.20min	337
9.113.2.21min_inplace	338
9.113.2.22minus	338
9.113.2.23plus	338
9.113.2.24rank_filter	338

9.113.2.25	thick_miss	338
9.113.2.26	thickening	338
9.113.2.27	thin_fit	339
9.113.2.28	thinning	339
9.113.2.29	top_hat_black	339
9.113.2.30	top_hat_self_complementary	339
9.113.2.31	top_hat_white	339
9.114	mln::morpho::approx Namespace Reference	340
9.114.1	Detailed Description	340
9.115	mln::morpho::attribute Namespace Reference	340
9.115.1	Detailed Description	340
9.116	mln::morpho::closing::approx Namespace Reference	340
9.116.1	Detailed Description	340
9.116.2	Function Documentation	341
9.116.2.1	structural	341
9.117	mln::morpho::elementary Namespace Reference	341
9.117.1	Detailed Description	341
9.117.2	Function Documentation	341
9.117.2.1	closing	341
9.117.2.2	mln_trait_op_minus_twice	342
9.117.2.3	opening	342
9.117.2.4	top_hat_black	342
9.117.2.5	top_hat_self_complementary	342
9.117.2.6	top_hat_white	342
9.118	mln::morpho::impl Namespace Reference	342
9.118.1	Detailed Description	343
9.119	mln::morpho::impl::generic Namespace Reference	343
9.119.1	Detailed Description	343
9.120	mln::morpho::opening::approx Namespace Reference	343
9.120.1	Detailed Description	343
9.120.2	Function Documentation	343
9.120.2.1	structural	343
9.121	mln::morpho::reconstruction Namespace Reference	343
9.121.1	Detailed Description	344
9.122	mln::morpho::reconstruction::by_dilation Namespace Reference	344
9.122.1	Detailed Description	344
9.123	mln::morpho::reconstruction::by_erosion Namespace Reference	344
9.123.1	Detailed Description	344
9.124	mln::morpho::tree Namespace Reference	344
9.124.1	Detailed Description	345

9.124.2 Function Documentation	346
9.124.2.1 compute_attribute_image	346
9.124.2.2 compute_attribute_image_from	346
9.124.2.3 compute_parent	347
9.124.2.4 dual_input_max_tree	348
9.124.2.5 max_tree	348
9.124.2.6 min_tree	348
9.124.2.7 propagate_if	349
9.124.2.8 propagate_if	349
9.124.2.9 propagate_if_value	349
9.124.2.10 propagate_node_to_ancestors	349
9.124.2.11 propagate_node_to_ancestors	350
9.124.2.12 propagate_node_to_descendants	350
9.124.2.13 propagate_node_to_descendants	350
9.124.2.14 propagate_representative	350
9.125mln::morpho::tree::filter Namespace Reference	351
9.125.1 Detailed Description	351
9.125.2 Function Documentation	351
9.125.2.1 direct	351
9.125.2.2 filter	351
9.125.2.3 max	352
9.125.2.4 min	352
9.125.2.5 subtractive	352
9.126mln::morpho::watershed Namespace Reference	353
9.126.1 Detailed Description	353
9.126.2 Function Documentation	353
9.126.2.1 flooding	353
9.126.2.2 flooding	354
9.126.2.3 superpose	354
9.126.2.4 superpose	354
9.126.2.5 topological	354
9.127mln::morpho::watershed::watershed Namespace Reference	354
9.127.1 Detailed Description	355
9.128mln::morpho::watershed::watershed::generic Namespace Reference	355
9.128.1 Detailed Description	355
9.129mln::norm Namespace Reference	355
9.129.1 Detailed Description	356
9.129.2 Function Documentation	356
9.129.2.1 l1	356
9.129.2.2 l1_distance	356

9.129.2.3 l2	356
9.129.2.4 l2_distance	356
9.129.2.5 linfty	356
9.129.2.6 linfty_distance	357
9.129.2.7 sqr_l2	357
9.130mln::norm::impl Namespace Reference	357
9.130.1 Detailed Description	357
9.131mln::opt Namespace Reference	357
9.131.1 Detailed Description	358
9.131.2 Function Documentation	358
9.131.2.1 at	358
9.131.2.2 at	358
9.131.2.3 at	358
9.131.2.4 at	358
9.131.2.5 at	358
9.131.2.6 at	358
9.132mln::opt::impl Namespace Reference	358
9.132.1 Detailed Description	359
9.133mln::pw Namespace Reference	359
9.133.1 Detailed Description	359
9.134mln::registration Namespace Reference	359
9.134.1 Detailed Description	360
9.134.2 Function Documentation	360
9.134.2.1 get_rot	360
9.134.2.2 icp	360
9.134.2.3 icp	361
9.134.2.4 registration1	361
9.134.2.5 registration2	361
9.134.2.6 registration3	361
9.135mln::select Namespace Reference	361
9.135.1 Detailed Description	362
9.136mln::set Namespace Reference	362
9.136.1 Detailed Description	362
9.136.2 Function Documentation	362
9.136.2.1 card	362
9.136.2.2 compute	363
9.136.2.3 compute_with_weights	363
9.136.2.4 compute_with_weights	363
9.136.2.5 get	363
9.136.2.6 has	364

9.136.2.7 mln_meta_accu_result	364
9.136.2.8 mln_meta_accu_result	364
9.137mln::subsampling Namespace Reference	364
9.137.1 Detailed Description	365
9.137.2 Function Documentation	365
9.137.2.1 antialiased	365
9.137.2.2 antialiased	365
9.137.2.3 gaussian_subsampling	365
9.137.2.4 subsampling	365
9.138mln::tag Namespace Reference	365
9.138.1 Detailed Description	366
9.139mln::test Namespace Reference	366
9.139.1 Detailed Description	366
9.139.2 Function Documentation	366
9.139.2.1 positive	366
9.139.2.2 predicate	366
9.139.2.3 predicate	367
9.139.2.4 predicate	367
9.140mln::test::impl Namespace Reference	367
9.140.1 Detailed Description	367
9.141mln::topo Namespace Reference	367
9.141.1 Detailed Description	370
9.141.2 Function Documentation	371
9.141.2.1 detach	371
9.141.2.2 edge	371
9.141.2.3 is_facet	371
9.141.2.4 make_algebraic_face	371
9.141.2.5 make_algebraic_n_face	371
9.141.2.6 operator!=	372
9.141.2.7 operator!=	372
9.141.2.8 operator!=	372
9.141.2.9 operator!=	372
9.141.2.10operator+	372
9.141.2.11operator-	373
9.141.2.12operator-	373
9.141.2.13operator-	373
9.141.2.14operator<	373
9.141.2.15operator<	373
9.141.2.16operator<	373
9.141.2.17operator<	374

9.141.2.18operator<<	374
9.141.2.19operator<<	374
9.141.2.20operator<<	374
9.141.2.21operator<<	374
9.141.2.22operator<<	374
9.141.2.23operator==	374
9.141.2.24operator==	375
9.141.2.25operator==	375
9.141.2.26operator==	375
9.141.2.27operator==	375
9.142mln::trace Namespace Reference	376
9.142.1 Detailed Description	376
9.143mln::trait Namespace Reference	376
9.143.1 Detailed Description	376
9.144mln::transform Namespace Reference	376
9.144.1 Detailed Description	377
9.144.2 Function Documentation	377
9.144.2.1 distance_and_closest_point_geodesic	377
9.144.2.2 distance_and_closest_point_geodesic	378
9.144.2.3 distance_and_influence_zone_geodesic	378
9.144.2.4 distance_front	379
9.144.2.5 distance_geodesic	379
9.144.2.6 hough	379
9.144.2.7 influence_zone_front	379
9.144.2.8 influence_zone_front	379
9.144.2.9 influence_zone_geodesic	380
9.144.2.10influence_zone_geodesic_saturated	380
9.144.2.11influence_zone_geodesic_saturated	380
9.145mln::util Namespace Reference	380
9.145.1 Detailed Description	383
9.145.2 Typedef Documentation	383
9.145.2.1 vertex_id_t	383
9.145.3 Function Documentation	383
9.145.3.1 display_branch	383
9.145.3.2 display_tree	383
9.145.3.3 lemmings	384
9.145.3.4 make_greater_point	384
9.145.3.5 make_greater_psite	384
9.145.3.6 operator<	384
9.145.3.7 operator<<	384

9.145.3.8 operator<<	384
9.145.3.9 operator==	385
9.145.3.10operator==	385
9.145.3.11ord_strict	385
9.145.3.12ord_weak	385
9.145.3.13ree_fast_to_image	385
9.145.3.14ree_to_fast	385
9.145.3.15ree_to_image	386
9.146mln::util::impl Namespace Reference	386
9.146.1 Detailed Description	386
9.147mln::value Namespace Reference	386
9.147.1 Detailed Description	389
9.147.2 Typedef Documentation	389
9.147.2.1 float01_16	389
9.147.2.2 float01_8	389
9.147.2.3 gl16	390
9.147.2.4 gl8	390
9.147.2.5 glf	390
9.147.2.6 int_s16	390
9.147.2.7 int_s32	390
9.147.2.8 int_s8	390
9.147.2.9 int_u12	390
9.147.2.10nt_u16	390
9.147.2.11int_u32	390
9.147.2.12nt_u8	391
9.147.2.13abel_16	391
9.147.2.14abel_32	391
9.147.2.15abel_8	391
9.147.2.16gb16	391
9.147.2.17rgb8	391
9.147.3 Function Documentation	391
9.147.3.1 cast	391
9.147.3.2 equiv	391
9.147.3.3 operator*	391
9.147.3.4 operator*	392
9.147.3.5 operator+	392
9.147.3.6 operator+	392
9.147.3.7 operator-	392
9.147.3.8 operator-	392
9.147.3.9 operator/	392

9.147.3.10operator/	392
9.147.3.11operator<<	393
9.147.3.12operator<<	393
9.147.3.13operator<<	393
9.147.3.14operator<<	393
9.147.3.15operator<<	393
9.147.3.16operator<<	394
9.147.3.17operator<<	394
9.147.3.18operator<<	394
9.147.3.19operator<<	395
9.147.3.20operator<<	395
9.147.3.21operator<<	395
9.147.3.22operator==	395
9.147.3.23operator==	395
9.147.3.24other	395
9.147.3.25stack	396
9.148mln::value::impl Namespace Reference	396
9.148.1 Detailed Description	396
9.149mln::win Namespace Reference	396
9.149.1 Detailed Description	397
9.149.2 Function Documentation	397
9.149.2.1 diff	397
9.149.2.2 mln_regular	397
9.149.2.3 mln_regular	398
9.149.2.4 sym	398
9.149.2.5 sym	398
10 Class Documentation	399
10.1 array< T > Class Template Reference	399
10.1.1 Detailed Description	399
10.2 C_Function< E > Struct Template Reference	399
10.2.1 Detailed Description	399
10.3 data< I > Struct Template Reference	399
10.3.1 Detailed Description	400
10.4 depth1st_piter< T > Class Template Reference	400
10.4.1 Detailed Description	400
10.5 dn_leaf_piter< T > Struct Template Reference	400
10.5.1 Detailed Description	400
10.6 dn_node_piter< T > Struct Template Reference	401
10.6.1 Detailed Description	401

10.7	dn_site_piter< T > Struct Template Reference	401
10.7.1	Detailed Description	401
10.8	face_data< N, D > Class Template Reference	401
10.8.1	Detailed Description	401
10.9	faces_set_mixin< N, D > Struct Template Reference	401
10.9.1	Detailed Description	402
10.10	graph_elt_mixed_window< G, S, S2 > Class Template Reference	402
10.10.1	Detailed Description	402
10.11	graph_elt_window< G, S > Class Template Reference	402
10.11.1	Detailed Description	402
10.12	graph_elt_window_if< G, S, I > Class Template Reference	402
10.12.1	Detailed Description	402
10.13	graph_mixed_window_iter_dispatch< G, S, S2 > Struct Template Reference	403
10.13.1	Detailed Description	403
10.14	graph_window_if_iter_dispatch< G, S > Struct Template Reference	403
10.14.1	Detailed Description	403
10.15	graph_window_iter_dispatch< G, S > Struct Template Reference	403
10.15.1	Detailed Description	403
10.16	int_s< n > Struct Template Reference	403
10.16.1	Detailed Description	404
10.17	line_graph< G > Class Template Reference	404
10.17.1	Detailed Description	404
10.18	mln::accu::center< P, V > Struct Template Reference	404
10.18.1	Detailed Description	404
10.18.2	Member Function Documentation	405
10.18.2.1	init	405
10.18.2.2	is_valid	405
10.18.2.3	nsites	405
10.18.2.4	take_as_init	405
10.18.2.5	take_n_times	405
10.18.2.6	to_result	405
10.19	mln::accu::convolve< T1, T2, R > Struct Template Reference	405
10.19.1	Detailed Description	406
10.19.2	Member Function Documentation	406
10.19.2.1	init	406
10.19.2.2	is_valid	406
10.19.2.3	take_as_init	406
10.19.2.4	take_n_times	406
10.19.2.5	to_result	407
10.20	mln::accu::count_adjacent_vertices< F, S > Struct Template Reference	407

10.20.1 Detailed Description	407
10.20.2 Member Function Documentation	407
10.20.2.1 init	407
10.20.2.2 is_valid	407
10.20.2.3 set_value	408
10.20.2.4 take_as_init	408
10.20.2.5 take_n_times	408
10.20.2.6 to_result	408
10.21 mln::accu::count_labels< L > Struct Template Reference	408
10.21.1 Detailed Description	409
10.21.2 Member Function Documentation	409
10.21.2.1 init	409
10.21.2.2 is_valid	409
10.21.2.3 set_value	409
10.21.2.4 take_as_init	409
10.21.2.5 take_n_times	409
10.21.2.6 to_result	409
10.22 mln::accu::count_value< V > Struct Template Reference	409
10.22.1 Detailed Description	410
10.22.2 Member Function Documentation	410
10.22.2.1 init	410
10.22.2.2 is_valid	410
10.22.2.3 set_value	410
10.22.2.4 take_as_init	410
10.22.2.5 take_n_times	411
10.22.2.6 to_result	411
10.23 mln::accu::histo< V > Struct Template Reference	411
10.23.1 Detailed Description	411
10.23.2 Member Function Documentation	411
10.23.2.1 is_valid	411
10.23.2.2 take	412
10.23.2.3 take_as_init	412
10.23.2.4 take_n_times	412
10.23.2.5 vect	412
10.24 mln::accu::label_used< L > Struct Template Reference	412
10.24.1 Detailed Description	412
10.24.2 Member Function Documentation	413
10.24.2.1 init	413
10.24.2.2 is_valid	413
10.24.2.3 take	413

10.24.2.4 take_as_init	413
10.24.2.5 take_n_times	413
10.24.2.6 to_result	413
10.25mln::accu::logic::land Struct Reference	413
10.25.1 Detailed Description	414
10.25.2 Member Function Documentation	414
10.25.2.1 init	414
10.25.2.2 is_valid	414
10.25.2.3 take_as_init	414
10.25.2.4 take_n_times	414
10.25.2.5 to_result	414
10.26mln::accu::logic::land_basic Struct Reference	415
10.26.1 Detailed Description	415
10.26.2 Member Function Documentation	415
10.26.2.1 can_stop	415
10.26.2.2 init	415
10.26.2.3 is_valid	415
10.26.2.4 take_as_init	416
10.26.2.5 take_n_times	416
10.26.2.6 to_result	416
10.27mln::accu::logic::lor Struct Reference	416
10.27.1 Detailed Description	416
10.27.2 Member Function Documentation	416
10.27.2.1 init	416
10.27.2.2 is_valid	417
10.27.2.3 take_as_init	417
10.27.2.4 take_n_times	417
10.27.2.5 to_result	417
10.28mln::accu::logic::lor_basic Struct Reference	417
10.28.1 Detailed Description	417
10.28.2 Member Function Documentation	418
10.28.2.1 can_stop	418
10.28.2.2 init	418
10.28.2.3 is_valid	418
10.28.2.4 take_as_init	418
10.28.2.5 take_n_times	418
10.28.2.6 to_result	418
10.29mln::accu::maj_h< T > Struct Template Reference	418
10.29.1 Detailed Description	419
10.29.2 Member Function Documentation	419

10.29.2.1 init	419
10.29.2.2 is_valid	419
10.29.2.3 take_as_init	419
10.29.2.4 take_n_times	419
10.29.2.5 to_result	419
10.30mln::accu::math::count< T > Struct Template Reference	420
10.30.1 Detailed Description	420
10.30.2 Member Function Documentation	420
10.30.2.1 init	420
10.30.2.2 is_valid	420
10.30.2.3 set_value	420
10.30.2.4 take_as_init	421
10.30.2.5 take_n_times	421
10.30.2.6 to_result	421
10.31mln::accu::math::inf< T > Struct Template Reference	421
10.31.1 Detailed Description	421
10.31.2 Member Function Documentation	421
10.31.2.1 init	421
10.31.2.2 is_valid	422
10.31.2.3 take_as_init	422
10.31.2.4 take_n_times	422
10.31.2.5 to_result	422
10.32mln::accu::math::sum< T, S > Struct Template Reference	422
10.32.1 Detailed Description	423
10.32.2 Member Function Documentation	423
10.32.2.1 init	423
10.32.2.2 is_valid	423
10.32.2.3 take_as_init	423
10.32.2.4 take_n_times	423
10.32.2.5 to_result	423
10.33mln::accu::math::sup< T > Struct Template Reference	423
10.33.1 Detailed Description	424
10.33.2 Member Function Documentation	424
10.33.2.1 init	424
10.33.2.2 is_valid	424
10.33.2.3 take_as_init	424
10.33.2.4 take_n_times	424
10.33.2.5 to_result	424
10.34mln::accu::max_site< I > Struct Template Reference	425
10.34.1 Detailed Description	425

10.34.2 Member Function Documentation	425
10.34.2.1 init	425
10.34.2.2 is_valid	425
10.34.2.3 take_as_init	425
10.34.2.4 take_n_times	425
10.34.2.5 to_result	426
10.35mln::accu::meta::center Struct Reference	426
10.35.1 Detailed Description	426
10.36mln::accu::meta::count_adjacent_vertices Struct Reference	426
10.36.1 Detailed Description	427
10.37mln::accu::meta::count_labels Struct Reference	427
10.37.1 Detailed Description	428
10.38mln::accu::meta::count_value Struct Reference	428
10.38.1 Detailed Description	429
10.39mln::accu::meta::histo Struct Reference	429
10.39.1 Detailed Description	430
10.40mln::accu::meta::label_used Struct Reference	430
10.40.1 Detailed Description	431
10.41mln::accu::meta::logic::land Struct Reference	431
10.41.1 Detailed Description	432
10.42mln::accu::meta::logic::land_basic Struct Reference	432
10.42.1 Detailed Description	433
10.43mln::accu::meta::logic::lor Struct Reference	433
10.43.1 Detailed Description	434
10.44mln::accu::meta::logic::lor_basic Struct Reference	434
10.44.1 Detailed Description	435
10.45mln::accu::meta::maj_h Struct Reference	435
10.45.1 Detailed Description	436
10.46mln::accu::meta::math::count Struct Reference	436
10.46.1 Detailed Description	437
10.47mln::accu::meta::math::inf Struct Reference	437
10.47.1 Detailed Description	438
10.48mln::accu::meta::math::sum Struct Reference	438
10.48.1 Detailed Description	439
10.49mln::accu::meta::math::sup Struct Reference	439
10.49.1 Detailed Description	440
10.50mln::accu::meta::max_site Struct Reference	440
10.50.1 Detailed Description	441
10.51mln::accu::meta::nil Struct Reference	441
10.51.1 Detailed Description	442

10.52mln::accu::meta::p< mA > Struct Template Reference	442
10.52.1 Detailed Description	443
10.53mln::accu::meta::pair< A1, A2 > Struct Template Reference	443
10.53.1 Detailed Description	444
10.54mln::accu::meta::rms Struct Reference	444
10.54.1 Detailed Description	445
10.55mln::accu::meta::shape::bbox Struct Reference	445
10.55.1 Detailed Description	446
10.56mln::accu::meta::shape::height Struct Reference	446
10.56.1 Detailed Description	447
10.57mln::accu::meta::shape::volume Struct Reference	447
10.57.1 Detailed Description	448
10.58mln::accu::meta::stat::max Struct Reference	448
10.58.1 Detailed Description	449
10.59mln::accu::meta::stat::max_h Struct Reference	449
10.59.1 Detailed Description	450
10.60mln::accu::meta::stat::mean Struct Reference	450
10.60.1 Detailed Description	451
10.61mln::accu::meta::stat::median_alt< T > Struct Template Reference	451
10.61.1 Detailed Description	452
10.62mln::accu::meta::stat::median_h Struct Reference	452
10.62.1 Detailed Description	453
10.63mln::accu::meta::stat::min Struct Reference	453
10.63.1 Detailed Description	454
10.64mln::accu::meta::stat::min_h Struct Reference	454
10.64.1 Detailed Description	455
10.65mln::accu::meta::stat::rank Struct Reference	455
10.65.1 Detailed Description	456
10.66mln::accu::meta::stat::rank_high_quant Struct Reference	456
10.66.1 Detailed Description	457
10.67mln::accu::meta::tuple< n, > Struct Template Reference	457
10.67.1 Detailed Description	458
10.68mln::accu::meta::val< mA > Struct Template Reference	458
10.68.1 Detailed Description	459
10.69mln::accu::nil< T > Struct Template Reference	459
10.69.1 Detailed Description	460
10.69.2 Member Function Documentation	460
10.69.2.1 init	460
10.69.2.2 is_valid	460
10.69.2.3 take_as_init	460

10.69.2.4 take_n_times	460
10.69.2.5 to_result	460
10.70mln::accu::p< A > Struct Template Reference	460
10.70.1 Detailed Description	461
10.70.2 Member Function Documentation	461
10.70.2.1 init	461
10.70.2.2 is_valid	461
10.70.2.3 take_as_init	461
10.70.2.4 take_n_times	461
10.70.2.5 to_result	461
10.71mln::accu::pair< A1, A2, T > Struct Template Reference	462
10.71.1 Detailed Description	463
10.71.2 Member Function Documentation	463
10.71.2.1 first	463
10.71.2.2 first_accu	463
10.71.2.3 init	463
10.71.2.4 is_valid	463
10.71.2.5 second	463
10.71.2.6 second_accu	463
10.71.2.7 take_as_init	464
10.71.2.8 take_n_times	464
10.71.2.9 to_result	464
10.72mln::accu::rms< T, V > Struct Template Reference	464
10.72.1 Detailed Description	464
10.72.2 Member Function Documentation	465
10.72.2.1 init	465
10.72.2.2 is_valid	465
10.72.2.3 take_as_init	465
10.72.2.4 take_n_times	465
10.72.2.5 to_result	465
10.73mln::accu::shape::bbox< P > Struct Template Reference	465
10.73.1 Detailed Description	466
10.73.2 Member Function Documentation	466
10.73.2.1 init	466
10.73.2.2 is_valid	466
10.73.2.3 take_as_init	466
10.73.2.4 take_n_times	466
10.73.2.5 to_result	466
10.74mln::accu::shape::height< I > Struct Template Reference	466
10.74.1 Detailed Description	467

10.74.2 Member Typedef Documentation	467
10.74.2.1 argument	467
10.74.2.2 value	467
10.74.3 Member Function Documentation	467
10.74.3.1 init	467
10.74.3.2 is_valid	468
10.74.3.3 set_value	468
10.74.3.4 take_as_init	468
10.74.3.5 take_n_times	468
10.74.3.6 to_result	468
10.75mln::accu::shape::volume< I > Struct Template Reference	468
10.75.1 Detailed Description	469
10.75.2 Member Typedef Documentation	469
10.75.2.1 argument	469
10.75.2.2 value	469
10.75.3 Member Function Documentation	469
10.75.3.1 init	469
10.75.3.2 is_valid	469
10.75.3.3 set_value	469
10.75.3.4 take_as_init	470
10.75.3.5 take_n_times	470
10.75.3.6 to_result	470
10.76mln::accu::site_set::rectangularity< P > Class Template Reference	470
10.76.1 Detailed Description	470
10.76.2 Constructor & Destructor Documentation	471
10.76.2.1 rectangularity	471
10.76.3 Member Function Documentation	471
10.76.3.1 area	471
10.76.3.2 bbox	471
10.76.3.3 take_as_init	471
10.76.3.4 take_n_times	471
10.76.3.5 to_result	471
10.77mln::accu::stat::deviation< T, S, M > Struct Template Reference	471
10.77.1 Detailed Description	472
10.77.2 Member Function Documentation	472
10.77.2.1 init	472
10.77.2.2 is_valid	472
10.77.2.3 take_as_init	472
10.77.2.4 take_n_times	473
10.77.2.5 to_result	473

10.78mln::accu::stat::histo3d_rgb< V > Struct Template Reference	473
10.78.1 Detailed Description	473
10.78.2 Constructor & Destructor Documentation	474
10.78.2.1 histo3d_rgb	474
10.78.3 Member Function Documentation	474
10.78.3.1 init	474
10.78.3.2 is_valid	474
10.78.3.3 take	474
10.78.3.4 take	474
10.78.3.5 take_as_init	475
10.78.3.6 take_n_times	475
10.78.3.7 to_result	475
10.79mln::accu::stat::max< T > Struct Template Reference	475
10.79.1 Detailed Description	475
10.79.2 Member Function Documentation	476
10.79.2.1 init	476
10.79.2.2 is_valid	476
10.79.2.3 set_value	476
10.79.2.4 take_as_init	476
10.79.2.5 take_n_times	476
10.79.2.6 to_result	476
10.80mln::accu::stat::max_h< V > Struct Template Reference	476
10.80.1 Detailed Description	477
10.80.2 Member Function Documentation	477
10.80.2.1 init	477
10.80.2.2 is_valid	477
10.80.2.3 take_as_init	477
10.80.2.4 take_n_times	477
10.80.2.5 to_result	477
10.81mln::accu::stat::mean< T, S, M > Struct Template Reference	477
10.81.1 Detailed Description	478
10.81.2 Member Function Documentation	478
10.81.2.1 count	478
10.81.2.2 init	478
10.81.2.3 is_valid	478
10.81.2.4 sum	479
10.81.2.5 take_as_init	479
10.81.2.6 take_n_times	479
10.81.2.7 to_result	479
10.82mln::accu::stat::median_alt< S > Struct Template Reference	479

10.82.1 Detailed Description	480
10.82.2 Member Function Documentation	480
10.82.2.1 is_valid	480
10.82.2.2 take	480
10.82.2.3 take_as_init	480
10.82.2.4 take_n_times	480
10.82.2.5 to_result	481
10.83mln::accu::stat::median_h< V > Struct Template Reference	481
10.83.1 Detailed Description	481
10.83.2 Member Function Documentation	482
10.83.2.1 init	482
10.83.2.2 is_valid	482
10.83.2.3 take_as_init	482
10.83.2.4 take_n_times	482
10.83.2.5 to_result	482
10.84mln::accu::stat::meta::deviation Struct Reference	482
10.84.1 Detailed Description	483
10.85mln::accu::stat::min< T > Struct Template Reference	483
10.85.1 Detailed Description	484
10.85.2 Member Function Documentation	484
10.85.2.1 init	484
10.85.2.2 is_valid	484
10.85.2.3 set_value	484
10.85.2.4 take_as_init	484
10.85.2.5 take_n_times	484
10.85.2.6 to_result	484
10.86mln::accu::stat::min_h< V > Struct Template Reference	485
10.86.1 Detailed Description	485
10.86.2 Member Function Documentation	485
10.86.2.1 init	485
10.86.2.2 is_valid	485
10.86.2.3 take_as_init	485
10.86.2.4 take_n_times	485
10.86.2.5 to_result	486
10.87mln::accu::stat::min_max< V > Struct Template Reference	486
10.87.1 Detailed Description	487
10.87.2 Member Function Documentation	487
10.87.2.1 first	487
10.87.2.2 first_accu	487
10.87.2.3 init	487

10.87.2.4 <code>is_valid</code>	487
10.87.2.5 <code>second</code>	487
10.87.2.6 <code>second_accu</code>	487
10.87.2.7 <code>take_as_init</code>	488
10.87.2.8 <code>take_n_times</code>	488
10.87.2.9 <code>to_result</code>	488
10.88mln::accu::stat::rank< T > Struct Template Reference	488
10.88.1 Detailed Description	488
10.88.2 Member Function Documentation	489
10.88.2.1 <code>init</code>	489
10.88.2.2 <code>is_valid</code>	489
10.88.2.3 <code>k</code>	489
10.88.2.4 <code>take_as_init</code>	489
10.88.2.5 <code>take_n_times</code>	489
10.88.2.6 <code>to_result</code>	489
10.89mln::accu::stat::rank< bool > Struct Template Reference	489
10.89.1 Detailed Description	490
10.89.2 Member Function Documentation	490
10.89.2.1 <code>init</code>	490
10.89.2.2 <code>is_valid</code>	490
10.89.2.3 <code>take_as_init</code>	490
10.89.2.4 <code>take_n_times</code>	490
10.89.2.5 <code>to_result</code>	490
10.90mln::accu::stat::rank_high_quant< T > Struct Template Reference	491
10.90.1 Detailed Description	491
10.90.2 Member Function Documentation	491
10.90.2.1 <code>init</code>	491
10.90.2.2 <code>is_valid</code>	491
10.90.2.3 <code>take_as_init</code>	491
10.90.2.4 <code>take_n_times</code>	492
10.90.2.5 <code>to_result</code>	492
10.91mln::accu::stat::var< T > Struct Template Reference	492
10.91.1 Detailed Description	492
10.91.2 Member Typedef Documentation	493
10.91.2.1 <code>mean_t</code>	493
10.91.3 Member Function Documentation	493
10.91.3.1 <code>init</code>	493
10.91.3.2 <code>is_valid</code>	493
10.91.3.3 <code>mean</code>	493
10.91.3.4 <code>n_items</code>	493

10.91.3.5 take_as_init	493
10.91.3.6 take_n_times	493
10.91.3.7 to_result	493
10.91.3.8 variance	494
10.92mln::accu::stat::variance< T, S, R > Struct Template Reference	494
10.92.1 Detailed Description	494
10.92.2 Member Function Documentation	495
10.92.2.1 init	495
10.92.2.2 is_valid	495
10.92.2.3 mean	495
10.92.2.4 n_items	495
10.92.2.5 standard_deviation	495
10.92.2.6 sum	495
10.92.2.7 take_n_times	495
10.92.2.8 to_result	496
10.92.2.9 var	496
10.93mln::accu::tuple< A, n, > Struct Template Reference	496
10.93.1 Detailed Description	496
10.93.2 Member Function Documentation	496
10.93.2.1 init	496
10.93.2.2 is_valid	497
10.93.2.3 take_as_init	497
10.93.2.4 take_n_times	497
10.93.2.5 to_result	497
10.94mln::accu::val< A > Struct Template Reference	497
10.94.1 Detailed Description	498
10.94.2 Member Function Documentation	498
10.94.2.1 init	498
10.94.2.2 is_valid	498
10.94.2.3 take_as_init	498
10.94.2.4 take_n_times	498
10.94.2.5 to_result	498
10.95mln::Accumulator< E > Struct Template Reference	498
10.95.1 Detailed Description	500
10.95.2 Member Function Documentation	500
10.95.2.1 take_as_init	500
10.95.2.2 take_n_times	500
10.96mln::algebra::h_mat< d, T > Struct Template Reference	500
10.96.1 Detailed Description	501
10.96.2 Member Enumeration Documentation	501

10.96.2.1 anonymous enum	501
10.96.3 Constructor & Destructor Documentation	501
10.96.3.1 h_mat	501
10.96.3.2 h_mat	501
10.96.4 Member Function Documentation	501
10.96.4.1 _1	501
10.96.4.2 t	501
10.97mln::algebra::h_vec< d, C > Class Template Reference	502
10.97.1 Detailed Description	502
10.97.2 Member Enumeration Documentation	502
10.97.2.1 anonymous enum	502
10.97.3 Constructor & Destructor Documentation	503
10.97.3.1 h_vec	503
10.97.3.2 h_vec	503
10.97.4 Member Function Documentation	503
10.97.4.1 operator mat< n, 1, U >	503
10.97.4.2 t	503
10.97.4.3 to_vec	503
10.97.5 Member Data Documentation	503
10.97.5.1 origin	503
10.97.5.2 zero	503
10.98mln::bkd_pixter1d< I > Class Template Reference	503
10.98.1 Detailed Description	504
10.98.2 Member Typedef Documentation	504
10.98.2.1 image	504
10.98.3 Constructor & Destructor Documentation	504
10.98.3.1 bkd_pixter1d	504
10.98.4 Member Function Documentation	504
10.98.4.1 next	504
10.99mln::bkd_pixter2d< I > Class Template Reference	505
10.99.1 Detailed Description	505
10.99.2 Member Typedef Documentation	505
10.99.2.1 image	505
10.99.3 Constructor & Destructor Documentation	505
10.99.3.1 bkd_pixter2d	505
10.99.4 Member Function Documentation	505
10.99.4.1 next	506
10.100mln::bkd_pixter3d< I > Class Template Reference	506
10.100.1 Detailed Description	506
10.100.2 Member Typedef Documentation	506

10.100.2.1image	506
10.100.3Constructor & Destructor Documentation	506
10.100.3.1bkd_pixter3d	506
10.100.4Member Function Documentation	507
10.100.4.1next	507
10.101In::box< P > Class Template Reference	507
10.101.1Detailed Description	510
10.101.2Member Typedef Documentation	510
10.101.2.1bkd_piter	510
10.101.2.2element	510
10.101.2.3wd_piter	510
10.101.2.4piter	510
10.101.2.5psite	510
10.101.2.6site	510
10.101.3Member Enumeration Documentation	510
10.101.3.1anonymous enum	510
10.101.4Constructor & Destructor Documentation	511
10.101.4.1box	511
10.101.4.2box	511
10.101.4.3box	511
10.101.5Member Function Documentation	511
10.101.5.1bbox	511
10.101.5.2crop_wrt	511
10.101.5.3enlarge	511
10.101.5.4enlarge	511
10.101.5.5has	512
10.101.5.6is_empty	512
10.101.5.7is_valid	512
10.101.5.8len	512
10.101.5.9memory_size	512
10.101.5.10merge	512
10.101.5.11sites	512
10.101.5.12center	513
10.101.5.13max	513
10.101.5.14max	513
10.101.5.15min	513
10.101.5.16min	513
10.101.5.17_larger	513
10.101.6Friends And Related Function Documentation	513
10.101.6.1operator<<	513

10.102.1	In::Box< E > Struct Template Reference	514
10.102.1	Detailed Description	515
10.102.2	Member Function Documentation	516
10.102.2.1	bbox	516
10.102.2.2	is_empty	516
10.102.2.3	len	516
10.102.2.4	nsites	516
10.102.3	Friends And Related Function Documentation	516
10.102.3.1	diff	516
10.102.3.2	inter	517
10.102.3.3	operator<	517
10.102.3.4	operator<	517
10.102.3.5	operator<<	517
10.102.3.6	operator<=	517
10.102.3.7	operator<=	518
10.102.3.8	operator==	518
10.102.3.9	sym_diff	518
10.102.3.10	uni	518
10.102.3.11	unique	518
10.103.1	In::box_runend_piter< P > Class Template Reference	518
10.103.1	Detailed Description	519
10.103.2	Member Function Documentation	519
10.103.2.1	next	519
10.103.2.2	run_length	519
10.104.1	In::box_runstart_piter< P > Class Template Reference	519
10.104.1	Detailed Description	520
10.104.2	Constructor & Destructor Documentation	520
10.104.2.1	box_runstart_piter	520
10.104.3	Member Function Documentation	520
10.104.3.1	next	520
10.104.3.2	run_length	520
10.105.1	In::Browsing< E > Struct Template Reference	520
10.105.1	Detailed Description	521
10.106.1	In::canvas::browsing::backdiagonal2d_t Struct Reference	521
10.106.1	Detailed Description	522
10.107.1	In::canvas::browsing::breadth_first_search_t Struct Reference	523
10.107.1	Detailed Description	523
10.108.1	In::canvas::browsing::depth_first_search_t Struct Reference	523
10.108.1	Detailed Description	523
10.109.1	In::canvas::browsing::diagonal2d_t Struct Reference	523

10.109.1	Detailed Description	524
10.110.0	Inn::canvas::browsing::dir_struct_elt_incr_update_t Struct Reference	525
10.110.1	Detailed Description	525
10.111.0	Inn::canvas::browsing::directional_t Struct Reference	526
10.111.1	Detailed Description	527
10.112.0	Inn::canvas::browsing::fwd_t Struct Reference	528
10.112.1	Detailed Description	528
10.113.0	Inn::canvas::browsing::hyper_directional_t Struct Reference	529
10.113.1	Detailed Description	530
10.114.0	Inn::canvas::browsing::snake_fwd_t Struct Reference	531
10.114.1	Detailed Description	531
10.115.0	Inn::canvas::browsing::snake_generic_t Struct Reference	532
10.115.1	Detailed Description	533
10.116.0	Inn::canvas::browsing::snake_vert_t Struct Reference	534
10.116.1	Detailed Description	534
10.117.0	Inn::canvas::chamfer< F > Struct Template Reference	535
10.117.1	Detailed Description	535
10.118.0	Inn::category< R(*) (A) > Struct Template Reference	535
10.118.1	Detailed Description	535
10.119.0	Inn::complex_image< D, G, V > Class Template Reference	536
10.119.1	Detailed Description	536
10.119.2	Member Typedef Documentation	537
10.119.2.1	geom	537
10.119.2.2	value	537
10.119.2.3	value	537
10.119.2.4	skeleton	537
10.119.2.5	value	537
10.119.3	Constructor & Destructor Documentation	537
10.119.3.1	complex_image	537
10.119.4	Member Function Documentation	537
10.119.4.1	domain	537
10.119.4.2	operator()	538
10.119.4.3	operator()	538
10.119.4.4	values	538
10.119.5	Member Data Documentation	538
10.119.5.1	dim	538
10.120.0	Inn::complex_neighborhood_bkd_piter< I, G, N > Class Template Reference	538
10.120.1	Detailed Description	539
10.120.2	Member Typedef Documentation	539
10.120.2.1	iter_type	539

10.120.2.2psite	539
10.120.3Constructor & Destructor Documentation	539
10.120.3.1complex_neighborhood_bkd_piter	539
10.120.4Member Function Documentation	539
10.120.4.1iter	539
10.120.4.2next	540
10.121In::complex_neighborhood_fwd_piter< I, G, N > Class Template Reference	540
10.121.1Detailed Description	540
10.121.2Member Typedef Documentation	540
10.121.2.1iter_type	540
10.121.2.2psite	541
10.121.3Constructor & Destructor Documentation	541
10.121.3.1complex_neighborhood_fwd_piter	541
10.121.4Member Function Documentation	541
10.121.4.1iter	541
10.121.4.2next	541
10.122In::complex_psite< D, G > Class Template Reference	541
10.122.1Detailed Description	542
10.122.2Constructor & Destructor Documentation	542
10.122.2.1complex_psite	542
10.122.2.2complex_psite	542
10.122.3Member Function Documentation	542
10.122.3.1change_target	542
10.122.3.2face	543
10.122.3.3face_id	543
10.122.3.4invalidate	543
10.122.3.5s_valid	543
10.122.3.6n	543
10.122.3.7site_set	543
10.123In::complex_window_bkd_piter< I, G, W > Class Template Reference	544
10.123.1Detailed Description	544
10.123.2Member Typedef Documentation	544
10.123.2.1iter_type	544
10.123.2.2psite	544
10.123.3Constructor & Destructor Documentation	545
10.123.3.1complex_window_bkd_piter	545
10.123.4Member Function Documentation	545
10.123.4.1iter	545
10.123.4.2next	545
10.124In::complex_window_fwd_piter< I, G, W > Class Template Reference	545

10.124.1Detailed Description	546
10.124.2Member Typedef Documentation	546
10.124.2.1iter_type	546
10.124.2.2psite	546
10.124.3Constructor & Destructor Documentation	546
10.124.3.1complex_window_fwd_piter	546
10.124.4Member Function Documentation	546
10.124.4.1iter	546
10.124.4.2next	546
10.125In::decorated_image< I, D > Struct Template Reference	547
10.125.1Detailed Description	547
10.125.2Member Typedef Documentation	547
10.125.2.1lvalue	547
10.125.2.2psite	548
10.125.2.3rvalue	548
10.125.2.4skeleton	548
10.125.3Constructor & Destructor Documentation	548
10.125.3.1decorated_image	548
10.125.4Member Function Documentation	548
10.125.4.1decoration	548
10.125.4.2decoration	548
10.125.4.3operator decorated_image< const I, D >	548
10.125.4.4operator()	548
10.125.4.5operator()	549
10.126In::Delta_Point_Site< E > Struct Template Reference	549
10.126.1Detailed Description	549
10.127In::Delta_Point_Site< void > Struct Template Reference	549
10.127.1Detailed Description	549
10.128In::doc::Accumulator< E > Struct Template Reference	550
10.128.1Detailed Description	550
10.128.2Member Typedef Documentation	550
10.128.2.1argument	550
10.128.3Member Function Documentation	550
10.128.3.1init	550
10.128.3.2ake	550
10.128.3.3ake	550
10.129In::doc::Box< E > Struct Template Reference	551
10.129.1Detailed Description	552
10.129.2Member Typedef Documentation	552
10.129.2.1bkd_piter	552

10.129.2.2	fwd_piter	552
10.129.2.3	psite	552
10.129.2.4	site	552
10.129.3	Member Function Documentation	552
10.129.3.1	bbox	552
10.129.3.2	has	552
10.129.3.3	nsites	553
10.129.3.4	pmax	553
10.129.3.5	pmin	553
10.130.0	In::doc::Dpoint< E > Struct Template Reference	553
10.130.1	Detailed Description	554
10.130.2	Member Typedef Documentation	554
10.130.2.1	coord	554
10.130.2.2	dpoint	554
10.130.2.3	point	554
10.130.3	Member Enumeration Documentation	555
10.130.3.1	anonymous enum	555
10.130.4	Member Function Documentation	555
10.130.4.1	operator[]	555
10.131.0	In::doc::Fastest_Image< E > Struct Template Reference	555
10.131.1	Detailed Description	557
10.131.2	Member Typedef Documentation	557
10.131.2.1	bkd_piter	557
10.131.2.2	coord	557
10.131.2.3	dpoint	557
10.131.2.4	fwd_piter	557
10.131.2.5	value	558
10.131.2.6	point	558
10.131.2.7	pset	558
10.131.2.8	psite	558
10.131.2.9	value	558
10.131.2.10	skeleton	558
10.131.2.11	value	558
10.131.2.12	set	559
10.131.3	Member Function Documentation	559
10.131.3.1	bbox	559
10.131.3.2	border	559
10.131.3.3	buffer	559
10.131.3.4	delta_index	559
10.131.3.5	domain	559

10.131.3.6has	560
10.131.3.7has	560
10.131.3.8s_valid	560
10.131.3.9elements	560
10.131.3.10sites	560
10.131.3.11operator()	560
10.131.3.12operator()	561
10.131.3.13operator[]	561
10.131.3.14operator[]	561
10.131.3.15point_at_index	562
10.131.3.16values	562
10.132.1In::doc::Generalized_Pixel< E > Struct Template Reference	562
10.132.1Detailed Description	563
10.132.2Member Typedef Documentation	563
10.132.2.1image	563
10.132.2.2value	563
10.132.2.3value	563
10.132.3Member Function Documentation	563
10.132.3.1ima	563
10.132.3.2val	564
10.133.1In::doc::Image< E > Struct Template Reference	564
10.133.1Detailed Description	565
10.133.2Member Typedef Documentation	565
10.133.2.1bkd_piter	565
10.133.2.2coord	566
10.133.2.3dpoint	566
10.133.2.4fwd_piter	566
10.133.2.5value	566
10.133.2.6point	566
10.133.2.7pset	566
10.133.2.8psite	567
10.133.2.9value	567
10.133.2.10skeleton	567
10.133.2.11value	567
10.133.2.12set	567
10.133.3Member Function Documentation	567
10.133.3.1bbox	567
10.133.3.2domain	568
10.133.3.3has	568
10.133.3.4has	568

10.133.3.5s_valid	568
10.133.3.6nsites	568
10.133.3.7operator()	568
10.133.3.8operator()	569
10.133.3.9values	569
10.134In::doc::Iterator< E > Struct Template Reference	569
10.134.1Detailed Description	570
10.134.2Member Function Documentation	570
10.134.2.1invalidate	570
10.134.2.2s_valid	570
10.134.2.3start	570
10.135In::doc::Neighborhood< E > Struct Template Reference	570
10.135.1Detailed Description	571
10.135.2Member Typedef Documentation	571
10.135.2.1bkd_niter	571
10.135.2.2dpoint	571
10.135.2.3fwd_niter	571
10.135.2.4niter	571
10.135.2.5point	571
10.136In::doc::Object< E > Struct Template Reference	572
10.136.1Detailed Description	572
10.137In::doc::Pixel_Iterator< E > Struct Template Reference	572
10.137.1Detailed Description	573
10.137.2Member Typedef Documentation	574
10.137.2.1image	574
10.137.2.2value	574
10.137.2.3value	574
10.137.2.4value	574
10.137.3Member Function Documentation	574
10.137.3.1ima	574
10.137.3.2invalidate	574
10.137.3.3s_valid	574
10.137.3.4start	574
10.137.3.5val	575
10.138In::doc::Point_Site< E > Struct Template Reference	575
10.138.1Detailed Description	575
10.138.2Member Typedef Documentation	575
10.138.2.1coord	575
10.138.2.2dpoint	576
10.138.2.3mesh	576

10.138.2.4point	576
10.138.3Member Enumeration Documentation	576
10.138.3.1anonymous enum	576
10.138.4Member Function Documentation	576
10.138.4.1operator[]	576
10.138.4.2o_point	577
10.139In::doc::Site_Iterator< E > Struct Template Reference	577
10.139.1Detailed Description	578
10.139.2Member Typedef Documentation	578
10.139.2.1psite	578
10.139.3Member Function Documentation	578
10.139.3.1invalidate	578
10.139.3.2s_valid	578
10.139.3.3operator psite	578
10.139.3.4start	578
10.140In::doc::Site_Set< E > Struct Template Reference	579
10.140.1Detailed Description	579
10.140.2Member Typedef Documentation	580
10.140.2.1bkd_piter	580
10.140.2.2fwd_piter	580
10.140.2.3psite	580
10.140.2.4site	580
10.140.3Member Function Documentation	580
10.140.3.1has	580
10.141In::doc::Value_Iterator< E > Struct Template Reference	580
10.141.1Detailed Description	581
10.141.2Member Typedef Documentation	582
10.141.2.1value	582
10.141.3Member Function Documentation	582
10.141.3.1invalidate	582
10.141.3.2s_valid	582
10.141.3.3operator value	582
10.141.3.4start	582
10.142In::doc::Value_Set< E > Struct Template Reference	582
10.142.1Detailed Description	583
10.142.2Member Typedef Documentation	583
10.142.2.1bkd_viter	583
10.142.2.2fwd_viter	583
10.142.2.3value	583
10.142.3Member Function Documentation	584

10.142.3.1has	584
10.142.3.2index_of	584
10.142.3.3nvalues	584
10.142.3.4operator[]	584
10.143In::doc::Weighted_Window< E > Struct Template Reference	584
10.143.1Detailed Description	585
10.143.2Member Typedef Documentation	585
10.143.2.1bkd_qiter	585
10.143.2.2dpoint	585
10.143.2.3wd_qiter	586
10.143.2.4point	586
10.143.2.5weight	586
10.143.2.6window	586
10.143.3Member Function Documentation	586
10.143.3.1delta	586
10.143.3.2s_centered	586
10.143.3.3s_empty	586
10.143.3.4sym	586
10.143.3.5win	586
10.144In::doc::Window< E > Struct Template Reference	586
10.144.1Detailed Description	587
10.144.2Member Typedef Documentation	587
10.144.2.1bkd_qiter	587
10.144.2.2wd_qiter	587
10.144.2.3qiter	588
10.145In::dpoint< G, C > Struct Template Reference	588
10.145.1Detailed Description	589
10.145.2Member Typedef Documentation	589
10.145.2.1coord	589
10.145.2.2grid	589
10.145.2.3psite	589
10.145.2.4site	590
10.145.2.5vec	590
10.145.3Member Enumeration Documentation	590
10.145.3.1anonymous enum	590
10.145.4Constructor & Destructor Documentation	590
10.145.4.1dpoint	590
10.145.4.2dpoint	590
10.145.4.3dpoint	590
10.145.4.4dpoint	590

10.145.4.5dpoint	590
10.145.5 Member Function Documentation	591
10.145.5.1operator mln::algebra::vec< dpoint< G, C >::dim, Q >	591
10.145.5.2operator[]	591
10.145.5.3operator[]	591
10.145.5.4set_all	591
10.145.5.5to_vec	591
10.146 mln::Dpoint< E > Struct Template Reference	591
10.146.1 Detailed Description	592
10.146.2 Member Function Documentation	592
10.146.2.1to_dpoint	592
10.147 mln::dpoints_bkd_pixter< I > Class Template Reference	592
10.147.1 Detailed Description	593
10.147.2 Constructor & Destructor Documentation	593
10.147.2.1dpoints_bkd_pixter	593
10.147.2.2dpoints_bkd_pixter	594
10.147.3 Member Function Documentation	594
10.147.3.1center_val	594
10.147.3.2invalidate	594
10.147.3.3s_valid	594
10.147.3.4next	594
10.147.3.5start	594
10.147.3.6update	594
10.148 mln::dpoints_fwd_pixter< I > Class Template Reference	595
10.148.1 Detailed Description	595
10.148.2 Constructor & Destructor Documentation	595
10.148.2.1dpoints_fwd_pixter	595
10.148.2.2dpoints_fwd_pixter	596
10.148.3 Member Function Documentation	596
10.148.3.1center_val	596
10.148.3.2invalidate	596
10.148.3.3s_valid	596
10.148.3.4next	596
10.148.3.5start	596
10.148.3.6update	596
10.149 mln::dpsites_bkd_piter< V > Class Template Reference	597
10.149.1 Detailed Description	597
10.149.2 Constructor & Destructor Documentation	597
10.149.2.1dpsites_bkd_piter	597
10.149.2.2dpsites_bkd_piter	597

10.149.3Member Function Documentation	597
10.149.3.1next	597
10.150In::dpsites_fwd_piter< V > Class Template Reference	598
10.150.1Detailed Description	598
10.150.2Constructor & Destructor Documentation	598
10.150.2.1dpsites_fwd_piter	598
10.150.2.2dpsites_fwd_piter	598
10.150.3Member Function Documentation	599
10.150.3.1next	599
10.151In::Edge< E > Struct Template Reference	599
10.151.1Detailed Description	599
10.152In::edge_image< P, V, G > Class Template Reference	599
10.152.1Detailed Description	600
10.152.2Member Typedef Documentation	600
10.152.2.1edge_nbh_t	600
10.152.2.2edge_win_t	600
10.152.2.3graph_t	600
10.152.2.4nbh_t	601
10.152.2.5site_function_t	601
10.152.2.6skeleton	601
10.152.2.7win_t	601
10.152.3Constructor & Destructor Documentation	601
10.152.3.1edge_image	601
10.152.4Member Function Documentation	601
10.152.4.1operator()	601
10.153In::extended< I > Struct Template Reference	601
10.153.1Detailed Description	602
10.153.2Member Typedef Documentation	602
10.153.2.1skeleton	602
10.153.2.2value	602
10.153.3Constructor & Destructor Documentation	602
10.153.3.1extended	602
10.153.3.2extended	602
10.153.4Member Function Documentation	603
10.153.4.1domain	603
10.154In::extension_fun< I, F > Class Template Reference	603
10.154.1Detailed Description	603
10.154.2Member Typedef Documentation	604
10.154.2.1rvalue	604
10.154.2.2skeleton	604

10.154.2.3value	604
10.154.3Constructor & Destructor Documentation	604
10.154.3.1extension_fun	604
10.154.3.2extension_fun	604
10.154.4Member Function Documentation	604
10.154.4.1extension	604
10.154.4.2has	604
10.154.4.3operator()	604
10.154.4.4operator()	605
10.155In::extension_ima< I, J > Class Template Reference	605
10.155.1Detailed Description	605
10.155.2Member Typedef Documentation	606
10.155.2.1value	606
10.155.2.2skeleton	606
10.155.2.3value	606
10.155.3Constructor & Destructor Documentation	606
10.155.3.1extension_ima	606
10.155.3.2extension_ima	606
10.155.4Member Function Documentation	606
10.155.4.1extension	606
10.155.4.2has	606
10.155.4.3operator()	606
10.155.4.4operator()	607
10.156In::extension_val< I > Class Template Reference	607
10.156.1Detailed Description	607
10.156.2Member Typedef Documentation	608
10.156.2.1value	608
10.156.2.2skeleton	608
10.156.2.3value	608
10.156.3Constructor & Destructor Documentation	608
10.156.3.1extension_val	608
10.156.3.2extension_val	608
10.156.4Member Function Documentation	608
10.156.4.1change_extension	608
10.156.4.2extension	608
10.156.4.3has	608
10.156.4.4operator()	609
10.156.4.5operator()	609
10.157In::flat_image< T, S > Struct Template Reference	609
10.157.1Detailed Description	610

10.157.2	Member Typedef Documentation	610
10.157.2.1	value	610
10.157.2.2	value	610
10.157.2.3	skeleton	610
10.157.2.4	value	610
10.157.3	Constructor & Destructor Documentation	610
10.157.3.1	flat_image	610
10.157.3.2	flat_image	610
10.157.4	Member Function Documentation	610
10.157.4.1	domain	610
10.157.4.2	has	611
10.157.4.3	operator()	611
10.157.4.4	operator()	611
10.158	fun::from_accu< A > Struct Template Reference	611
10.158.1	Detailed Description	611
10.159	fun::n2v::white_gaussian< V > Struct Template Reference	611
10.159.1	Detailed Description	612
10.160	fun::p2b::antilogy Struct Reference	612
10.160.1	Detailed Description	613
10.161	fun::p2b::tautology Struct Reference	613
10.161.1	Detailed Description	614
10.162	fun::v2b::Inot< V > Struct Template Reference	614
10.162.1	Detailed Description	615
10.163	fun::v2b::threshold< V > Struct Template Reference	616
10.163.1	Detailed Description	616
10.164	fun::v2v::ch_function_value< F, V > Class Template Reference	617
10.164.1	Detailed Description	617
10.165	fun::v2v::component< T, i > Struct Template Reference	618
10.165.1	Detailed Description	618
10.166	fun::v2v::l1_norm< V, R > Struct Template Reference	619
10.166.1	Detailed Description	619
10.167	fun::v2v::l2_norm< V, R > Struct Template Reference	620
10.167.1	Detailed Description	620
10.168	fun::v2v::linear< V, T, R > Struct Template Reference	621
10.168.1	Detailed Description	621
10.169	fun::v2v::linfty_norm< V, R > Struct Template Reference	622
10.169.1	Detailed Description	622
10.170	fun::v2v::rgb8_to_rgbn< n > Struct Template Reference	623
10.170.1	Detailed Description	624
10.170.2	Member Function Documentation	624

10.170.2.operator()	624
10.171.inn::fun::v2w2v::cos< V > Struct Template Reference	624
10.171.1.Detailed Description	625
10.172.inn::fun::v2w_w2v::l1_norm< V, R > Struct Template Reference	625
10.172.1.Detailed Description	626
10.173.inn::fun::v2w_w2v::l2_norm< V, R > Struct Template Reference	626
10.173.1.Detailed Description	627
10.174.inn::fun::v2w_w2v::linfty_norm< V, R > Struct Template Reference	628
10.174.1.Detailed Description	628
10.175.inn::fun::vv2b::eq< L, R > Struct Template Reference	629
10.175.1.Detailed Description	629
10.176.inn::fun::vv2b::ge< L, R > Struct Template Reference	629
10.176.1.Detailed Description	630
10.177.inn::fun::vv2b::gt< L, R > Struct Template Reference	630
10.177.1.Detailed Description	631
10.178.inn::fun::vv2b::implies< L, R > Struct Template Reference	631
10.178.1.Detailed Description	632
10.179.inn::fun::vv2b::le< L, R > Struct Template Reference	632
10.179.1.Detailed Description	633
10.180.inn::fun::vv2b::lt< L, R > Struct Template Reference	633
10.180.1.Detailed Description	634
10.181.inn::fun::vv2v::diff_abs< V > Struct Template Reference	634
10.181.1.Detailed Description	635
10.182.inn::fun::vv2v::land< L, R > Struct Template Reference	635
10.182.1.Detailed Description	636
10.183.inn::fun::vv2v::land_not< L, R > Struct Template Reference	636
10.183.1.Detailed Description	637
10.184.inn::fun::vv2v::lor< L, R > Struct Template Reference	637
10.184.1.Detailed Description	638
10.185.inn::fun::vv2v::lxor< L, R > Struct Template Reference	638
10.185.1.Detailed Description	639
10.186.inn::fun::vv2v::max< V > Struct Template Reference	639
10.186.1.Detailed Description	640
10.187.inn::fun::vv2v::min< L, R > Struct Template Reference	640
10.187.1.Detailed Description	641
10.188.inn::fun::vv2v::vec< V > Struct Template Reference	641
10.188.1.Detailed Description	642
10.189.inn::fun::x2p::closest_point< P > Struct Template Reference	642
10.189.1.Detailed Description	642
10.190.inn::fun::x2v::bilinear< I > Struct Template Reference	643

10.190.1 Detailed Description	643
10.190.2 Member Function Documentation	643
10.190.2.1 operator()	643
10.190.2.2 operator()	643
10.191 In::fun::x2v::trilinear< I > Struct Template Reference	644
10.191.1 Detailed Description	644
10.192 In::fun::x2x::composed< T2, T1 > Struct Template Reference	644
10.192.1 Detailed Description	644
10.192.2 Constructor & Destructor Documentation	644
10.192.2.1 composed	644
10.192.2.2 composed	645
10.193 In::fun::x2x::linear< I > Struct Template Reference	645
10.193.1 Detailed Description	645
10.193.2 Constructor & Destructor Documentation	645
10.193.2.1 linear	645
10.193.3 Member Function Documentation	646
10.193.3.1 operator()	646
10.193.4 Member Data Documentation	646
10.193.4.1 ma	646
10.194 In::fun::x2x::rotation< n, C > Struct Template Reference	646
10.194.1 Detailed Description	648
10.194.2 Member Typedef Documentation	648
10.194.2.1 data_t	648
10.194.2.2 invert	648
10.194.3 Constructor & Destructor Documentation	648
10.194.3.1 rotation	648
10.194.3.2 rotation	648
10.194.3.3 rotation	648
10.194.3.4 rotation	649
10.194.4 Member Function Documentation	649
10.194.4.1 inv	649
10.194.4.2 operator()	649
10.194.4.3 set_alpha	649
10.194.4.4 set_axis	649
10.195 In::fun::x2x::translation< n, C > Struct Template Reference	649
10.195.1 Detailed Description	651
10.195.2 Member Typedef Documentation	651
10.195.2.1 data_t	651
10.195.2.2 invert	651
10.195.3 Constructor & Destructor Documentation	651

10.195.3.1translation	651
10.195.3.2translation	651
10.195.4Member Function Documentation	651
10.195.4.1inv	651
10.195.4.2operator()	652
10.195.4.3set_t	652
10.195.4.4	652
10.196In::fun_image< F, I > Struct Template Reference	652
10.196.1Detailed Description	653
10.196.2Member Typedef Documentation	653
10.196.2.1value	653
10.196.2.2value	653
10.196.2.3skeleton	653
10.196.2.4value	653
10.196.3Constructor & Destructor Documentation	653
10.196.3.1fun_image	653
10.196.3.2fun_image	653
10.196.3.3fun_image	653
10.196.4Member Function Documentation	654
10.196.4.1operator()	654
10.196.4.2operator()	654
10.197In::Function< E > Struct Template Reference	654
10.197.1Detailed Description	654
10.197.2Constructor & Destructor Documentation	654
10.197.2.1Function	654
10.198In::Function< void > Struct Template Reference	654
10.198.1Detailed Description	655
10.199In::Function_n2v< E > Struct Template Reference	655
10.199.1Detailed Description	655
10.200In::Function_v2b< E > Struct Template Reference	656
10.200.1Detailed Description	656
10.201In::Function_v2v< E > Struct Template Reference	656
10.201.1Detailed Description	657
10.202In::Function_vv2b< E > Struct Template Reference	657
10.202.1Detailed Description	657
10.203In::Function_vv2v< E > Struct Template Reference	657
10.203.1Detailed Description	658
10.204In::fwd_pixter1d< I > Class Template Reference	658
10.204.1Detailed Description	659
10.204.2Member Typedef Documentation	659

10.204.2.1image	659
10.204.3Constructor & Destructor Documentation	659
10.204.3.1fwd_pixter1d	659
10.204.4Member Function Documentation	659
10.204.4.1next	659
10.205In::fwd_pixter2d< I > Class Template Reference	659
10.205.1Detailed Description	660
10.205.2Member Typedef Documentation	660
10.205.2.1image	660
10.205.3Constructor & Destructor Documentation	660
10.205.3.1fwd_pixter2d	660
10.205.4Member Function Documentation	660
10.205.4.1next	660
10.206In::fwd_pixter3d< I > Class Template Reference	661
10.206.1Detailed Description	661
10.206.2Member Typedef Documentation	661
10.206.2.1image	661
10.206.3Constructor & Destructor Documentation	661
10.206.3.1fwd_pixter3d	661
10.206.4Member Function Documentation	661
10.206.4.1next	661
10.207In::Gdpoint< E > Struct Template Reference	662
10.207.1Detailed Description	662
10.208In::Gdpoint< void > Struct Template Reference	662
10.208.1Detailed Description	663
10.209In::Generalized_Pixel< E > Struct Template Reference	663
10.209.1Detailed Description	663
10.210In::geom::complex_geometry< D, P > Class Template Reference	664
10.210.1Detailed Description	664
10.210.2Constructor & Destructor Documentation	664
10.210.2.1complex_geometry	664
10.210.3Member Function Documentation	664
10.210.3.1add_location	664
10.210.3.2operator()	665
10.211In::Gpoint< E > Struct Template Reference	665
10.211.1Detailed Description	666
10.211.2Friends And Related Function Documentation	666
10.211.2.1operator+	666
10.211.2.2operator+=	666
10.211.2.3operator-	667

10.211.2.4operator-=	667
10.211.2.5operator/	668
10.211.2.6operator<<	668
10.211.2.7operator==	668
10.212In::Graph< E > Struct Template Reference	669
10.212.1Detailed Description	669
10.213In::graph::attribute::card_t Struct Reference	669
10.213.1Detailed Description	670
10.213.2Member Typedef Documentation	670
10.213.2.1result	670
10.214In::graph::attribute::representative_t Struct Reference	670
10.214.1Detailed Description	670
10.214.2Member Typedef Documentation	670
10.214.2.1result	670
10.215In::graph_elt_mixed_neighborhood< G, S, S2 > Struct Template Reference	671
10.215.1Detailed Description	671
10.215.2Member Typedef Documentation	672
10.215.2.1bkd_niter	672
10.215.2.2fwd_niter	672
10.215.2.3niter	672
10.216In::graph_elt_mixed_window< G, S, S2 > Class Template Reference	672
10.216.1Detailed Description	674
10.216.2Member Typedef Documentation	674
10.216.2.1bkd_qiter	674
10.216.2.2center_t	674
10.216.2.3fwd_qiter	675
10.216.2.4graph_element	675
10.216.2.5psite	675
10.216.2.6qiter	675
10.216.2.7site	675
10.216.2.8target	675
10.216.3Member Function Documentation	675
10.216.3.1delta	675
10.216.3.2s_centered	675
10.216.3.3s_empty	676
10.216.3.4s_symmetric	676
10.216.3.5s_valid	676
10.216.3.6sym	676
10.217In::graph_elt_neighborhood< G, S > Struct Template Reference	676
10.217.1Detailed Description	677

10.217.2Member Typedef Documentation	677
10.217.2.1bkd_niter	677
10.217.2.2fwd_niter	677
10.217.2.3niter	677
10.218In::graph_elt_neighborhood_if< G, S, I > Struct Template Reference	677
10.218.1Detailed Description	678
10.218.2Member Typedef Documentation	679
10.218.2.1bkd_niter	679
10.218.2.2fwd_niter	679
10.218.2.3niter	679
10.218.3Constructor & Destructor Documentation	679
10.218.3.1graph_elt_neighborhood_if	679
10.218.3.2graph_elt_neighborhood_if	679
10.218.4Member Function Documentation	679
10.218.4.1mask	679
10.219In::graph_elt_window< G, S > Class Template Reference	679
10.219.1Detailed Description	681
10.219.2Member Typedef Documentation	681
10.219.2.1bkd_qiter	681
10.219.2.2center_t	681
10.219.2.3fwd_qiter	682
10.219.2.4graph_element	682
10.219.2.5psite	682
10.219.2.6qiter	682
10.219.2.7site	682
10.219.2.8target	682
10.219.3Member Function Documentation	682
10.219.3.1delta	682
10.219.3.2s_centered	682
10.219.3.3s_empty	683
10.219.3.4s_symmetric	683
10.219.3.5s_valid	683
10.219.3.6sym	683
10.220In::graph_elt_window_if< G, S, I > Class Template Reference	683
10.220.1Detailed Description	685
10.220.2Member Typedef Documentation	686
10.220.2.1bkd_qiter	686
10.220.2.2fwd_qiter	686
10.220.2.3mask_t	686
10.220.2.4psite	686

10.220.2.5qiter	686
10.220.2.6site	686
10.220.2.7target	686
10.220.3Constructor & Destructor Documentation	687
10.220.3.1graph_elt_window_if	687
10.220.3.2graph_elt_window_if	687
10.220.4Member Function Documentation	687
10.220.4.1change_mask	687
10.220.4.2delta	687
10.220.4.3s_centered	687
10.220.4.4s_empty	687
10.220.4.5s_symmetric	687
10.220.4.6s_valid	688
10.220.4.7mask	688
10.220.4.8sym	688
10.221In::graph_window_base< P, E > Class Template Reference	688
10.221.1Detailed Description	689
10.221.2Member Typedef Documentation	689
10.221.2.1site	689
10.221.3Member Function Documentation	689
10.221.3.1delta	689
10.221.3.2s_centered	689
10.221.3.3s_empty	689
10.221.3.4s_symmetric	689
10.221.3.5s_valid	690
10.221.3.6sym	690
10.222In::graph_window_if_piter< S, W, I > Class Template Reference	690
10.222.1Detailed Description	690
10.222.2Member Typedef Documentation	691
10.222.2.1P	691
10.222.3Constructor & Destructor Documentation	691
10.222.3.1graph_window_if_piter	691
10.222.4Member Function Documentation	691
10.222.4.1element	691
10.222.4.2d	691
10.222.4.3next	691
10.223In::graph_window_piter< S, W, I > Class Template Reference	691
10.223.1Detailed Description	692
10.223.2Member Typedef Documentation	692
10.223.2.1center_t	693

10.223.2.2graph_element	693
10.223.2.3P	693
10.223.3Constructor & Destructor Documentation	693
10.223.3.1graph_window_piter	693
10.223.3.2graph_window_piter	693
10.223.3.3graph_window_piter	693
10.223.4Member Function Documentation	694
10.223.4.1change_target_site_set	694
10.223.4.2element	694
10.223.4.3d	694
10.223.4.4next	694
10.223.4.5target_site_set	694
10.224In::hexa< I > Struct Template Reference	694
10.224.1Detailed Description	696
10.224.2Member Typedef Documentation	696
10.224.2.1bkd_piter	696
10.224.2.2fwd_piter	696
10.224.2.3value	696
10.224.2.4psite	696
10.224.2.5value	697
10.224.2.6skeleton	697
10.224.2.7value	697
10.224.3Constructor & Destructor Documentation	697
10.224.3.1hexa	697
10.224.3.2hexa	697
10.224.4Member Function Documentation	697
10.224.4.1domain	697
10.224.4.2has	697
10.224.4.3operator()	697
10.224.4.4operator()	698
10.225In::histo::array< T > Struct Template Reference	698
10.225.1Detailed Description	698
10.226In::Image< E > Struct Template Reference	698
10.226.1Detailed Description	699
10.227In::image1d< T > Struct Template Reference	700
10.227.1Detailed Description	701
10.227.2Member Typedef Documentation	701
10.227.2.1lvalue	701
10.227.2.2rvalue	701
10.227.2.3skeleton	701

10.227.2.4value	701
10.227.3Constructor & Destructor Documentation	701
10.227.3.1image1d	701
10.227.3.2image1d	702
10.227.3.3image1d	702
10.227.4Member Function Documentation	702
10.227.4.1bbox	702
10.227.4.2border	702
10.227.4.3buffer	702
10.227.4.4buffer	702
10.227.4.5delta_index	702
10.227.4.6domain	702
10.227.4.7element	702
10.227.4.8element	703
10.227.4.9has	703
10.227.4.10elements	703
10.227.4.11inds	703
10.227.4.12operator()	703
10.227.4.13operator()	703
10.227.4.14point_at_index	703
10.227.4.15bbox	703
10.228In::image2d< T > Class Template Reference	703
10.228.1Detailed Description	705
10.228.2Member Typedef Documentation	705
10.228.2.1lvalue	705
10.228.2.2rvalue	705
10.228.2.3skeleton	705
10.228.2.4value	705
10.228.3Constructor & Destructor Documentation	705
10.228.3.1image2d	705
10.228.3.2image2d	705
10.228.3.3image2d	705
10.228.4Member Function Documentation	706
10.228.4.1bbox	706
10.228.4.2border	706
10.228.4.3buffer	706
10.228.4.4buffer	706
10.228.4.5delta_index	706
10.228.4.6domain	706
10.228.4.7element	706

10.228.4.8element	706
10.228.4.9has	707
10.228.4.10cols	707
10.228.4.11elements	707
10.228.4.12rows	707
10.228.4.13operator()	707
10.228.4.14operator()	707
10.228.4.15point_at_index	707
10.229In::image2d_h< V > Struct Template Reference	707
10.229.1Detailed Description	709
10.229.2Member Typedef Documentation	709
10.229.2.1bkd_piter	709
10.229.2.2fwd_piter	709
10.229.2.3value	709
10.229.2.4psite	709
10.229.2.5rvalue	709
10.229.2.6skeleton	709
10.229.2.7value	709
10.229.3Constructor & Destructor Documentation	710
10.229.3.1image2d_h	710
10.229.4Member Function Documentation	710
10.229.4.1domain	710
10.229.4.2has	710
10.229.4.3operator()	710
10.229.4.4operator()	710
10.230In::image3d< T > Struct Template Reference	710
10.230.1Detailed Description	712
10.230.2Member Typedef Documentation	712
10.230.2.1lvalue	712
10.230.2.2rvalue	712
10.230.2.3skeleton	712
10.230.2.4value	712
10.230.3Constructor & Destructor Documentation	713
10.230.3.1image3d	713
10.230.3.2image3d	713
10.230.3.3image3d	713
10.230.4Member Function Documentation	713
10.230.4.1bbox	713
10.230.4.2border	713
10.230.4.3buffer	713

10.230.4.4	buffer	713
10.230.4.5	delta_index	713
10.230.4.6	domain	714
10.230.4.7	element	714
10.230.4.8	element	714
10.230.4.9	has	714
10.230.4.10	cols	714
10.230.4.11	elements	714
10.230.4.12	rows	714
10.230.4.13	slis	714
10.230.4.14	operator()	714
10.230.4.15	operator()	715
10.230.4.16	point_at_index	715
10.230.4.17	box	715
10.231	In::interpolated< I, F > Struct Template Reference	715
10.231.1	Detailed Description	716
10.231.2	Member Typedef Documentation	716
10.231.2.1	lvalue	716
10.231.2.2	psite	716
10.231.2.3	rvalue	716
10.231.2.4	skeleton	716
10.231.2.5	value	716
10.231.3	Constructor & Destructor Documentation	716
10.231.3.1	interpolated	716
10.231.4	Member Function Documentation	716
10.231.4.1	has	717
10.231.4.2	is_valid	717
10.232	In::io::dicom::dicom_header Struct Reference	717
10.232.1	Detailed Description	717
10.233	In::io::dump::dump_header Struct Reference	717
10.233.1	Detailed Description	717
10.234	In::io::fld::fld_header Struct Reference	717
10.234.1	Detailed Description	717
10.235	In::io::raw::raw_header Struct Reference	717
10.235.1	Detailed Description	718
10.236	In::Iterator< E > Struct Template Reference	718
10.236.1	Detailed Description	719
10.236.2	Member Function Documentation	720
10.236.2.1	next	720
10.237	In::labeled_image< I > Class Template Reference	720

10.237.1Detailed Description	722
10.237.2Member Typedef Documentation	722
10.237.2.1bbox_t	722
10.237.2.2skeleton	722
10.237.3Constructor & Destructor Documentation	722
10.237.3.1labeled_image	722
10.237.3.2labeled_image	723
10.237.3.3labeled_image	723
10.237.4Member Function Documentation	723
10.237.4.1bbox	723
10.237.4.2bboxes	723
10.237.4.3nlabels	723
10.237.4.4relabel	723
10.237.4.5relabel	723
10.237.4.6subdomain	723
10.237.4.7update_data	723
10.238In::labeled_image_base< I, E > Class Template Reference	724
10.238.1Detailed Description	725
10.238.2Member Typedef Documentation	725
10.238.2.1bbox_t	725
10.238.3Constructor & Destructor Documentation	725
10.238.3.1labeled_image_base	725
10.238.4Member Function Documentation	725
10.238.4.1bbox	726
10.238.4.2bboxes	726
10.238.4.3nlabels	726
10.238.4.4relabel	726
10.238.4.5relabel	726
10.238.4.6subdomain	726
10.238.4.7update_data	726
10.239In::lazy_image< I, F, B > Struct Template Reference	727
10.239.1Detailed Description	727
10.239.2Member Typedef Documentation	728
10.239.2.1lvalue	728
10.239.2.2rvalue	728
10.239.2.3skeleton	728
10.239.3Constructor & Destructor Documentation	728
10.239.3.1lazy_image	728
10.239.3.2azy_image	728
10.239.4Member Function Documentation	728

10.239.4.1domain	728
10.239.4.2has	728
10.239.4.3operator()	728
10.239.4.4operator()	729
10.239.4.5operator()	729
10.239.4.6operator()	729
10.240In::Literal< E > Struct Template Reference	729
10.240.1Detailed Description	730
10.241In::literal::black_t Struct Reference	731
10.241.1Detailed Description	731
10.242In::literal::blue_t Struct Reference	731
10.242.1Detailed Description	732
10.243In::literal::brown_t Struct Reference	732
10.243.1Detailed Description	733
10.244In::literal::cyan_t Struct Reference	733
10.244.1Detailed Description	734
10.245In::literal::green_t Struct Reference	734
10.245.1Detailed Description	735
10.246In::literal::identity_t Struct Reference	735
10.246.1Detailed Description	736
10.247In::literal::light_gray_t Struct Reference	736
10.247.1Detailed Description	737
10.248In::literal::lime_t Struct Reference	737
10.248.1Detailed Description	738
10.249In::literal::magenta_t Struct Reference	738
10.249.1Detailed Description	739
10.250In::literal::max_t Struct Reference	739
10.250.1Detailed Description	740
10.251In::literal::min_t Struct Reference	740
10.251.1Detailed Description	741
10.252In::literal::olive_t Struct Reference	741
10.252.1Detailed Description	742
10.253In::literal::one_t Struct Reference	742
10.253.1Detailed Description	743
10.254In::literal::orange_t Struct Reference	743
10.254.1Detailed Description	744
10.255In::literal::origin_t Struct Reference	744
10.255.1Detailed Description	745
10.256In::literal::pink_t Struct Reference	745
10.256.1Detailed Description	746

10.257	In::literal::purple_t Struct Reference	746
10.257.1	Detailed Description	747
10.258	In::literal::red_t Struct Reference	747
10.258.1	Detailed Description	748
10.259	In::literal::teal_t Struct Reference	748
10.259.1	Detailed Description	749
10.260	In::literal::violet_t Struct Reference	749
10.260.1	Detailed Description	750
10.261	In::literal::white_t Struct Reference	750
10.261.1	Detailed Description	751
10.262	In::literal::yellow_t Struct Reference	751
10.262.1	Detailed Description	752
10.263	In::literal::zero_t Struct Reference	752
10.263.1	Detailed Description	753
10.264	In::Mesh< E > Struct Template Reference	753
10.264.1	Detailed Description	754
10.265	In::Meta_Accumulator< E > Struct Template Reference	754
10.265.1	Detailed Description	755
10.266	In::Meta_Function< E > Struct Template Reference	756
10.266.1	Detailed Description	757
10.267	In::Meta_Function_v2v< E > Struct Template Reference	758
10.267.1	Detailed Description	758
10.268	In::Meta_Function_vv2v< E > Struct Template Reference	759
10.268.1	Detailed Description	759
10.269	In::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 > Struct Template Reference	759
10.269.1	Detailed Description	759
10.270	In::metal::converts_to< T, U > Struct Template Reference	760
10.270.1	Detailed Description	760
10.271	In::metal::equal< T1, T2 > Struct Template Reference	760
10.271.1	Detailed Description	760
10.272	In::metal::goes_to< T, U > Struct Template Reference	760
10.272.1	Detailed Description	761
10.273	In::metal::is< T, U > Struct Template Reference	761
10.273.1	Detailed Description	761
10.274	In::metal::is_a< T, M > Struct Template Reference	761
10.274.1	Detailed Description	761
10.275	In::metal::is_not< T, U > Struct Template Reference	761
10.275.1	Detailed Description	762
10.276	In::metal::is_not_a< T, M > Struct Template Reference	762
10.276.1	Detailed Description	762

10.277	<code>ln::morpho::attribute::card< I > Class Template Reference</code>	762
10.277.1	Detailed Description	763
10.277.2	Member Function Documentation	763
10.277.2.1	<code>init</code>	763
10.277.2.2	<code>is_valid</code>	763
10.277.2.3	<code>take_as_init</code>	763
10.277.2.4	<code>take_n_times</code>	763
10.277.2.5	<code>to_result</code>	763
10.278	<code>ln::morpho::attribute::count_adjacent_vertices< I > Struct Template Reference</code>	763
10.278.1	Detailed Description	764
10.278.2	Member Function Documentation	764
10.278.2.1	<code>init</code>	764
10.278.2.2	<code>is_valid</code>	764
10.278.2.3	<code>take_as_init</code>	764
10.278.2.4	<code>take_n_times</code>	764
10.278.2.5	<code>to_result</code>	764
10.279	<code>ln::morpho::attribute::height< I > Struct Template Reference</code>	764
10.279.1	Detailed Description	765
10.279.2	Member Function Documentation	765
10.279.2.1	<code>base_level</code>	765
10.279.2.2	<code>init</code>	765
10.279.2.3	<code>is_valid</code>	765
10.279.2.4	<code>take_as_init</code>	765
10.279.2.5	<code>take_n_times</code>	766
10.279.2.6	<code>to_result</code>	766
10.280	<code>ln::morpho::attribute::sharpness< I > Struct Template Reference</code>	766
10.280.1	Detailed Description	766
10.280.2	Member Function Documentation	767
10.280.2.1	<code>area</code>	767
10.280.2.2	<code>height</code>	767
10.280.2.3	<code>init</code>	767
10.280.2.4	<code>is_valid</code>	767
10.280.2.5	<code>take_as_init</code>	767
10.280.2.6	<code>take_n_times</code>	767
10.280.2.7	<code>to_result</code>	767
10.280.2.8	<code>volume</code>	767
10.281	<code>ln::morpho::attribute::sum< I, S > Class Template Reference</code>	767
10.281.1	Detailed Description	768
10.281.2	Member Function Documentation	768
10.281.2.1	<code>init</code>	768

10.281.2.2s_valid	768
10.281.2.3set_value	768
10.281.2.4take_as_init	769
10.281.2.5take_n_times	769
10.281.2.6o_result	769
10.281.2.7untake	769
10.282In::morpho::attribute::volume< I > Struct Template Reference	769
10.282.1Detailed Description	769
10.282.2Member Function Documentation	770
10.282.2.1area	770
10.282.2.2nit	770
10.282.2.3s_valid	770
10.282.2.4take_as_init	770
10.282.2.5take_n_times	770
10.282.2.6o_result	770
10.283In::neighb< W > Class Template Reference	770
10.283.1Detailed Description	771
10.283.2Member Typedef Documentation	771
10.283.2.1bkd_niter	771
10.283.2.2fwd_niter	771
10.283.2.3niter	772
10.283.3Constructor & Destructor Documentation	772
10.283.3.1neighb	772
10.283.3.2neighb	772
10.284In::Neighborhood< E > Struct Template Reference	772
10.284.1Detailed Description	772
10.285In::Neighborhood< void > Struct Template Reference	773
10.285.1Detailed Description	773
10.286In::Object< E > Struct Template Reference	773
10.286.1Detailed Description	773
10.287In::p2p_image< I, F > Struct Template Reference	773
10.287.1Detailed Description	774
10.287.2Member Typedef Documentation	774
10.287.2.1skeleton	774
10.287.3Constructor & Destructor Documentation	774
10.287.3.1p2p_image	774
10.287.3.2p2p_image	774
10.287.4Member Function Documentation	774
10.287.4.1domain	774
10.287.4.2fun	775

10.287.4.3operator()	775
10.287.4.4operator()	775
10.288In::p_array< P > Class Template Reference	775
10.288.1Detailed Description	776
10.288.2Member Typedef Documentation	776
10.288.2.1bkd_piter	776
10.288.2.2element	776
10.288.2.3fwd_piter	777
10.288.2.4_element	777
10.288.2.5piter	777
10.288.2.6psite	777
10.288.3Constructor & Destructor Documentation	777
10.288.3.1p_array	777
10.288.3.2p_array	777
10.288.4Member Function Documentation	777
10.288.4.1append	777
10.288.4.2append	777
10.288.4.3change	778
10.288.4.4clear	778
10.288.4.5has	778
10.288.4.6has	778
10.288.4.7insert	778
10.288.4.8s_valid	778
10.288.4.9memory_size	778
10.288.4.10sites	778
10.288.4.11operator[]	778
10.288.4.12operator[]	779
10.288.4.13operator[]	779
10.288.4.14reserve	779
10.288.4.15size	779
10.288.4.16std_vector	779
10.289In::p_centered< W > Class Template Reference	779
10.289.1Detailed Description	780
10.289.2Member Typedef Documentation	780
10.289.2.1bkd_piter	780
10.289.2.2element	780
10.289.2.3fwd_piter	780
10.289.2.4piter	780
10.289.2.5psite	781
10.289.2.6site	781

10.289.3	Constructor & Destructor Documentation	781
10.289.3.1	p_centered	781
10.289.3.2	p_centered	781
10.289.4	Member Function Documentation	781
10.289.4.1	center	781
10.289.4.2	has	781
10.289.4.3	s_valid	781
10.289.4.4	memory_size	781
10.289.4.5	window	782
10.290	nl::p_complex< D, G > Class Template Reference	782
10.290.1	Detailed Description	783
10.290.2	Member Typedef Documentation	783
10.290.2.1	bkd_piter	783
10.290.2.2	element	783
10.290.2.3	wd_piter	783
10.290.2.4	piter	783
10.290.2.5	site	783
10.290.3	Constructor & Destructor Documentation	784
10.290.3.1	p_complex	784
10.290.4	Member Function Documentation	784
10.290.4.1	cplx	784
10.290.4.2	cplx	784
10.290.4.3	geom	784
10.290.4.4	has	784
10.290.4.5	s_valid	784
10.290.4.6	faces	784
10.290.4.7	faces_of_dim	785
10.290.4.8	sites	785
10.291	nl::p_edges< G, F > Class Template Reference	785
10.291.1	Detailed Description	786
10.291.2	Member Typedef Documentation	786
10.291.2.1	bkd_piter	786
10.291.2.2	edge	786
10.291.2.3	element	787
10.291.2.4	fun_t	787
10.291.2.5	wd_piter	787
10.291.2.6	graph_element	787
10.291.2.7	graph_t	787
10.291.2.8	piter	787
10.291.2.9	site	787

10.291.3	Constructor & Destructor Documentation	787
10.291.3.1	p_edges	787
10.291.3.2	p_edges	788
10.291.3.3	p_edges	788
10.291.3.4	p_edges	788
10.291.4	Member Function Documentation	788
10.291.4.1	function	788
10.291.4.2	graph	788
10.291.4.3	has	789
10.291.4.4	has	789
10.291.4.5	invalidate	789
10.291.4.6	s_valid	789
10.291.4.7	memory_size	789
10.291.4.8	nedges	789
10.291.4.9	nsites	789
10.292	In::p_faces< N, D, P > Struct Template Reference	790
10.292.1	Detailed Description	790
10.292.2	Member Typedef Documentation	791
10.292.2.1	bkd_piter	791
10.292.2.2	element	791
10.292.2.3	wd_piter	791
10.292.2.4	piter	791
10.292.2.5	psite	791
10.292.3	Constructor & Destructor Documentation	791
10.292.3.1	p_faces	791
10.292.3.2	p_faces	791
10.292.4	Member Function Documentation	792
10.292.4.1	cplx	792
10.292.4.2	cplx	792
10.292.4.3	s_valid	792
10.292.4.4	nfaces	792
10.292.4.5	nsites	792
10.293	In::p_graph_piter< S, I > Class Template Reference	792
10.293.1	Detailed Description	793
10.293.2	Constructor & Destructor Documentation	793
10.293.2.1	p_graph_piter	793
10.293.3	Member Function Documentation	793
10.293.3.1	graph	793
10.293.3.2	d	793
10.293.3.3	ln_q_subject	793

10.293.3.4next	793
10.294Inn::p_if< S, F > Class Template Reference	794
10.294.1Detailed Description	795
10.294.2Member Typedef Documentation	795
10.294.2.1bkd_piter	795
10.294.2.2element	795
10.294.2.3fwd_piter	795
10.294.2.4piter	795
10.294.2.5psite	795
10.294.3Constructor & Destructor Documentation	795
10.294.3.1p_if	795
10.294.3.2p_if	795
10.294.4Member Function Documentation	796
10.294.4.1has	796
10.294.4.2s_valid	796
10.294.4.3memory_size	796
10.294.4.4overset	796
10.294.4.5pred	796
10.294.4.6predicate	796
10.295Inn::p_image< I > Class Template Reference	796
10.295.1Detailed Description	797
10.295.2Member Typedef Documentation	798
10.295.2.1bkd_piter	798
10.295.2.2element	798
10.295.2.3fwd_piter	798
10.295.2.4element	798
10.295.2.5piter	798
10.295.2.6psite	798
10.295.2.7element	798
10.295.2.8S	798
10.295.3Constructor & Destructor Documentation	798
10.295.3.1p_image	798
10.295.3.2p_image	799
10.295.4Member Function Documentation	799
10.295.4.1clear	799
10.295.4.2has	799
10.295.4.3insert	799
10.295.4.4s_valid	799
10.295.4.5memory_size	799
10.295.4.6nsites	799

10.295.4.7operator typename internal::p_image_site_set< I >::ret	799
10.295.4.8remove	799
10.295.4.9toggle	800
10.296In::p_indexed_bkd_piter< S > Class Template Reference	800
10.296.1Detailed Description	800
10.296.2Constructor & Destructor Documentation	800
10.296.2.1p_indexed_bkd_piter	800
10.296.2.2p_indexed_bkd_piter	800
10.296.3Member Function Documentation	800
10.296.3.1index	800
10.296.3.2next	801
10.297In::p_indexed_fwd_piter< S > Class Template Reference	801
10.297.1Detailed Description	801
10.297.2Constructor & Destructor Documentation	801
10.297.2.1p_indexed_fwd_piter	801
10.297.2.2p_indexed_fwd_piter	801
10.297.3Member Function Documentation	802
10.297.3.1index	802
10.297.3.2next	802
10.298In::p_indexed_psite< S > Class Template Reference	802
10.298.1Detailed Description	802
10.299In::p_key< K, P > Class Template Reference	802
10.299.1Detailed Description	804
10.299.2Member Typedef Documentation	804
10.299.2.1bkd_piter	804
10.299.2.2element	804
10.299.2.3fwd_piter	804
10.299.2.4element	804
10.299.2.5piter	804
10.299.2.6psite	804
10.299.2.7element	804
10.299.3Constructor & Destructor Documentation	805
10.299.3.1p_key	805
10.299.4Member Function Documentation	805
10.299.4.1change_key	805
10.299.4.2change_keys	805
10.299.4.3clear	805
10.299.4.4exists_key	805
10.299.4.5has	805
10.299.4.6has	805

10.299.4.7insert	805
10.299.4.8insert	806
10.299.4.9s_valid	806
10.299.4.10key	806
10.299.4.11keys	806
10.299.4.12memory_size	806
10.299.4.13sites	806
10.299.4.14operator()	806
10.299.4.15remove	806
10.299.4.16remove_key	806
10.300.1In::p_line2d Class Reference	807
10.300.1.1Detailed Description	808
10.300.2Member Typedef Documentation	808
10.300.2.1bkd_piter	808
10.300.2.2element	808
10.300.2.3fwd_piter	808
10.300.2.4piter	808
10.300.2.5psite	808
10.300.2.6q_box	808
10.300.3Constructor & Destructor Documentation	808
10.300.3.1p_line2d	808
10.300.3.2p_line2d	809
10.300.4Member Function Documentation	809
10.300.4.1bbox	809
10.300.4.2begin	809
10.300.4.3end	809
10.300.4.4has	809
10.300.4.5has	809
10.300.4.6s_valid	809
10.300.4.7memory_size	810
10.300.4.8nsites	810
10.300.4.9operator[]	810
10.300.4.10d_vector	810
10.301.1In::p_mutable_array_of< S > Class Template Reference	810
10.301.1.1Detailed Description	811
10.301.2Member Typedef Documentation	811
10.301.2.1bkd_piter	811
10.301.2.2element	811
10.301.2.3fwd_piter	811
10.301.2.4_element	812

10.301.2.5	iter	812
10.301.2.6	psite	812
10.301.3	Constructor & Destructor Documentation	812
10.301.3.1	p_mutable_array_of	812
10.301.4	Member Function Documentation	812
10.301.4.1	clear	812
10.301.4.2	has	812
10.301.4.3	insert	812
10.301.4.4	is_valid	812
10.301.4.5	memory_size	813
10.301.4.6	elements	813
10.301.4.7	operator[]	813
10.301.4.8	operator[]	813
10.301.4.9	reserve	813
10.302	In::p_n_faces_bkd_iter< D, G > Class Template Reference	813
10.302.1	Detailed Description	813
10.302.2	Constructor & Destructor Documentation	814
10.302.2.1	p_n_faces_bkd_iter	814
10.302.3	Member Function Documentation	814
10.302.3.1	in	814
10.302.3.2	next	814
10.303	In::p_n_faces_fwd_iter< D, G > Class Template Reference	814
10.303.1	Detailed Description	815
10.303.2	Constructor & Destructor Documentation	815
10.303.2.1	p_n_faces_fwd_iter	815
10.303.3	Member Function Documentation	815
10.303.3.1	in	815
10.303.3.2	next	815
10.304	In::p_priority< P, Q > Class Template Reference	815
10.304.1	Detailed Description	817
10.304.2	Member Typedef Documentation	817
10.304.2.1	bkd_iter	817
10.304.2.2	element	817
10.304.2.3	fwd_iter	817
10.304.2.4	_element	817
10.304.2.5	iter	817
10.304.2.6	psite	817
10.304.3	Constructor & Destructor Documentation	817
10.304.3.1	p_priority	817
10.304.4	Member Function Documentation	818

10.304.4.1clear	818
10.304.4.2exists_priority	818
10.304.4.3front	818
10.304.4.4has	818
10.304.4.5highest_priority	818
10.304.4.6insert	818
10.304.4.7insert	819
10.304.4.8s_valid	819
10.304.4.9lowest_priority	819
10.304.4.10memory_size	819
10.304.4.11sites	819
10.304.4.12operator()	819
10.304.4.13pop	819
10.304.4.14pop_front	820
10.304.4.15priorities	820
10.304.4.16push	820
10.305.1In::p_queue< P > Class Template Reference	820
10.305.1.1Detailed Description	821
10.305.2Member Typedef Documentation	822
10.305.2.1bkd_piter	822
10.305.2.2element	822
10.305.2.3fwd_piter	822
10.305.2.4_element	822
10.305.2.5piter	822
10.305.2.6psite	822
10.305.3Constructor & Destructor Documentation	822
10.305.3.1p_queue	822
10.305.4Member Function Documentation	822
10.305.4.1clear	822
10.305.4.2front	823
10.305.4.3has	823
10.305.4.4has	823
10.305.4.5insert	823
10.305.4.6s_valid	823
10.305.4.7memory_size	823
10.305.4.8sites	823
10.305.4.9operator[]	823
10.305.4.10pop	823
10.305.4.11pop_front	824
10.305.4.12push	824

10.305.4.1std_deque	824
10.306.1In::p_queue_fast< P > Class Template Reference	824
10.306.1Detailed Description	825
10.306.2Member Typedef Documentation	825
10.306.2.1bkd_piter	825
10.306.2.2element	826
10.306.2.3fwd_piter	826
10.306.2.4element	826
10.306.2.5piter	826
10.306.2.6psite	826
10.306.3Constructor & Destructor Documentation	826
10.306.3.1p_queue_fast	826
10.306.4Member Function Documentation	826
10.306.4.1clear	826
10.306.4.2compute_has	826
10.306.4.3empty	826
10.306.4.4front	827
10.306.4.5has	827
10.306.4.6has	827
10.306.4.7insert	827
10.306.4.8s_valid	827
10.306.4.9memory_size	827
10.306.4.10sites	827
10.306.4.11operator[]	827
10.306.4.12pop	827
10.306.4.13pop_front	828
10.306.4.14urge	828
10.306.4.15push	828
10.306.4.16reserve	828
10.306.4.17std_vector	828
10.307.1In::p_run< P > Class Template Reference	828
10.307.1Detailed Description	829
10.307.2Member Typedef Documentation	829
10.307.2.1bkd_piter	829
10.307.2.2element	830
10.307.2.3fwd_piter	830
10.307.2.4piter	830
10.307.2.5psite	830
10.307.2.6q_box	830
10.307.3Constructor & Destructor Documentation	830

10.307.3.1p_run	830
10.307.3.2p_run	830
10.307.3.3p_run	830
10.307.4Member Function Documentation	830
10.307.4.1bbox	830
10.307.4.2end	831
10.307.4.3has	831
10.307.4.4has	831
10.307.4.5has_index	831
10.307.4.6init	831
10.307.4.7is_valid	831
10.307.4.8length	831
10.307.4.9memory_size	831
10.307.4.10sites	831
10.307.4.11operator[]	832
10.307.4.12start	832
10.308.1In::p_set< P > Class Template Reference	832
10.308.1Detailed Description	833
10.308.2Member Typedef Documentation	833
10.308.2.1bkd_piter	833
10.308.2.2element	833
10.308.2.3fwd_piter	833
10.308.2.4_element	833
10.308.2.5piter	834
10.308.2.6psite	834
10.308.2.7_element	834
10.308.3Constructor & Destructor Documentation	834
10.308.3.1p_set	834
10.308.4Member Function Documentation	834
10.308.4.1clear	834
10.308.4.2has	834
10.308.4.3has	834
10.308.4.4has	834
10.308.4.5insert	834
10.308.4.6is_valid	835
10.308.4.7memory_size	835
10.308.4.8nsites	835
10.308.4.9operator[]	835
10.308.4.10move	835
10.308.4.11std_vector	835

10.308.4.12	til_set	835
10.309	ln::p_transformed< S, F > Class Template Reference	835
10.309.1	Detailed Description	836
10.309.2	Member Typedef Documentation	836
10.309.2.1	bkd_piter	836
10.309.2.2	element	837
10.309.2.3	fwd_piter	837
10.309.2.4	piter	837
10.309.2.5	psite	837
10.309.3	Constructor & Destructor Documentation	837
10.309.3.1	p_transformed	837
10.309.3.2	p_transformed	837
10.309.4	Member Function Documentation	837
10.309.4.1	function	837
10.309.4.2	has	837
10.309.4.3	s_valid	838
10.309.4.4	memory_size	838
10.309.4.5	primary_set	838
10.310	ln::p_transformed_piter< Pi, S, F > Struct Template Reference	838
10.310.1	Detailed Description	838
10.310.2	Constructor & Destructor Documentation	839
10.310.2.1	p_transformed_piter	839
10.310.2.2	p_transformed_piter	839
10.310.3	Member Function Documentation	839
10.310.3.1	change_target	839
10.310.3.2	next	839
10.311	ln::p_vaccess< V, S > Class Template Reference	839
10.311.1	Detailed Description	840
10.311.2	Member Typedef Documentation	841
10.311.2.1	bkd_piter	841
10.311.2.2	element	841
10.311.2.3	fwd_piter	841
10.311.2.4	element	841
10.311.2.5	piter	841
10.311.2.6	pset	841
10.311.2.7	psite	841
10.311.2.8	value	841
10.311.2.9	vset	841
10.311.3	Constructor & Destructor Documentation	842
10.311.3.1	p_vaccess	842

10.311.4	Member Function Documentation	842
10.311.4.1	has	842
10.311.4.2	has	842
10.311.4.3	insert	842
10.311.4.4	insert	842
10.311.4.5	is_valid	842
10.311.4.6	memory_size	842
10.311.4.7	operator()	842
10.311.4.8	values	843
10.312	nl::p_vertices< G, F > Class Template Reference	843
10.312.1	Detailed Description	844
10.312.2	Member Typedef Documentation	844
10.312.2.1	bkd_piter	844
10.312.2.2	element	844
10.312.2.3	fun_t	845
10.312.2.4	fwd_piter	845
10.312.2.5	graph_element	845
10.312.2.6	graph_t	845
10.312.2.7	piter	845
10.312.2.8	psite	845
10.312.2.9	vertex	845
10.312.3	Constructor & Destructor Documentation	845
10.312.3.1	p_vertices	845
10.312.3.2	p_vertices	846
10.312.3.3	p_vertices	846
10.312.3.4	p_vertices	846
10.312.3.5	p_vertices	846
10.312.4	Member Function Documentation	846
10.312.4.1	function	846
10.312.4.2	graph	847
10.312.4.3	has	847
10.312.4.4	has	847
10.312.4.5	invalidate	847
10.312.4.6	is_valid	847
10.312.4.7	memory_size	847
10.312.4.8	nsites	847
10.312.4.9	nvertices	848
10.312.4.10	operator()	848
10.313	nl::pixel< I > Struct Template Reference	848
10.313.1	Detailed Description	849

10.313.2	Constructor & Destructor Documentation	849
10.313.2.1	pixel	849
10.313.2.2	pixel	849
10.313.3	Member Function Documentation	849
10.313.3.1	change_to	849
10.313.3.2	s_valid	849
10.314	In::plain< I > Class Template Reference	849
10.314.1	Detailed Description	850
10.314.2	Member Typedef Documentation	850
10.314.2.1	skeleton	850
10.314.3	Constructor & Destructor Documentation	850
10.314.3.1	plain	850
10.314.3.2	plain	850
10.314.3.3	plain	851
10.314.4	Member Function Documentation	851
10.314.4.1	operator I	851
10.314.4.2	operator=	851
10.314.4.3	operator=	851
10.315	In::Point< P > Struct Template Reference	851
10.315.1	Detailed Description	852
10.315.2	Member Typedef Documentation	852
10.315.2.1	point	852
10.315.3	Member Function Documentation	852
10.315.3.1	to_point	852
10.315.4	Friends And Related Function Documentation	852
10.315.4.1	operator+=	852
10.315.4.2	operator-=	853
10.315.4.3	operator/	853
10.316	In::point< G, C > Struct Template Reference	853
10.316.1	Detailed Description	855
10.316.2	Member Typedef Documentation	856
10.316.2.1	coord	856
10.316.2.2	delta	856
10.316.2.3	dpsite	856
10.316.2.4	grid	856
10.316.2.5	h_vec	856
10.316.2.6	vec	856
10.316.3	Member Enumeration Documentation	856
10.316.3.1	anonymous enum	856
10.316.4	Constructor & Destructor Documentation	856

10.316.4.1point	856
10.316.4.2point	857
10.316.4.3point	857
10.316.4.4point	857
10.316.4.5point	857
10.316.5Member Function Documentation	857
10.316.5.1last_coord	857
10.316.5.2ast_coord	857
10.316.5.3minus_infty	857
10.316.5.4operator+=	857
10.316.5.5operator-=	858
10.316.5.6operator[]	858
10.316.5.7operator[]	858
10.316.5.8plus_infty	858
10.316.5.9set_all	858
10.316.5.10_h_vec	858
10.316.5.11b_vec	859
10.316.6Member Data Documentation	859
10.316.6.1origin	859
10.317Inn::Proxy< E > Struct Template Reference	859
10.317.1Detailed Description	859
10.318Inn::Proxy< void > Struct Template Reference	859
10.318.1Detailed Description	859
10.319Inn::Pseudo_Site< E > Struct Template Reference	859
10.319.1Detailed Description	860
10.320Inn::Pseudo_Site< void > Struct Template Reference	860
10.320.1Detailed Description	860
10.321Inn::pw::image< F, S > Class Template Reference	860
10.321.1Detailed Description	861
10.321.2Member Typedef Documentation	861
10.321.2.1skeleton	861
10.321.3Constructor & Destructor Documentation	861
10.321.3.1image	861
10.321.3.2image	861
10.322Inn::registration::closest_point_basic< P > Class Template Reference	861
10.322.1Detailed Description	861
10.323Inn::registration::closest_point_with_map< P > Class Template Reference	862
10.323.1Detailed Description	862
10.324Inn::Regular_Grid< E > Struct Template Reference	862
10.324.1Detailed Description	862

10.325	ln::safe_image< I > Class Template Reference	862
10.325.1	Detailed Description	863
10.325.2	Member Typedef Documentation	863
10.325.2.1	skeleton	863
10.325.3	Member Function Documentation	863
10.325.3.1	operator safe_image< const I >	863
10.326	ln::select::p_of< P > Struct Template Reference	863
10.326.1	Detailed Description	863
10.327	ln::Site< E > Struct Template Reference	863
10.327.1	Detailed Description	864
10.328	ln::Site< void > Struct Template Reference	864
10.328.1	Detailed Description	864
10.329	ln::Site_iterator< E > Struct Template Reference	865
10.329.1	Detailed Description	866
10.329.2	Member Function Documentation	866
10.329.2.1	next	866
10.330	ln::Site_Proxy< E > Struct Template Reference	866
10.330.1	Detailed Description	866
10.331	ln::Site_Proxy< void > Struct Template Reference	867
10.331.1	Detailed Description	867
10.332	ln::Site_Set< E > Struct Template Reference	867
10.332.1	Detailed Description	869
10.332.2	Friends And Related Function Documentation	869
10.332.2.1	diff	869
10.332.2.2	inter	869
10.332.2.3	operator<	869
10.332.2.4	operator<<	870
10.332.2.5	operator<=	870
10.332.2.6	operator==	870
10.332.2.7	sym_diff	870
10.332.2.8	uni	870
10.332.2.9	unique	871
10.333	ln::Site_Set< void > Struct Template Reference	871
10.333.1	Detailed Description	871
10.334	ln::sub_image< I, S > Class Template Reference	871
10.334.1	Detailed Description	871
10.334.2	Member Typedef Documentation	872
10.334.2.1	skeleton	872
10.334.3	Constructor & Destructor Documentation	872
10.334.3.1	sub_image	872

10.334.3.2sub_image	872
10.334.4Member Function Documentation	872
10.334.4.1domain	872
10.334.4.2operator sub_image< const I, S >	872
10.335In::sub_image_if< I, S > Struct Template Reference	872
10.335.1Detailed Description	873
10.335.2Member Typedef Documentation	873
10.335.2.1skeleton	873
10.335.3Constructor & Destructor Documentation	873
10.335.3.1sub_image_if	873
10.335.3.2sub_image_if	873
10.335.4Member Function Documentation	873
10.335.4.1domain	873
10.336In::thru_image< I, F > Class Template Reference	873
10.336.1Detailed Description	874
10.336.2Member Function Documentation	874
10.336.2.1operator thru_image< const I, F >	874
10.337In::thruin_image< I1, I2, F > Class Template Reference	874
10.337.1Detailed Description	875
10.337.2Member Typedef Documentation	875
10.337.2.1psite	875
10.337.2.2value	875
10.337.2.3skeleton	875
10.337.2.4value	875
10.337.3Member Function Documentation	875
10.337.3.1operator thruin_image< const I1, const I2, F >	875
10.338In::topo::adj_higher_dim_connected_n_face_bkd_iter< D > Class Template Reference	875
10.338.1Detailed Description	876
10.338.2Constructor & Destructor Documentation	876
10.338.2.1adj_higher_dim_connected_n_face_bkd_iter	876
10.338.3Member Function Documentation	876
10.338.3.1next	876
10.339In::topo::adj_higher_dim_connected_n_face_fwd_iter< D > Class Template Reference	876
10.339.1Detailed Description	877
10.339.2Constructor & Destructor Documentation	877
10.339.2.1adj_higher_dim_connected_n_face_fwd_iter	877
10.339.3Member Function Documentation	877
10.339.3.1next	877
10.340In::topo::adj_higher_face_bkd_iter< D > Class Template Reference	877
10.340.1Detailed Description	878

10.340.2	Constructor & Destructor Documentation	878
10.340.2.1	adj_higher_face_bkd_iter	878
10.340.3	Member Function Documentation	878
10.340.3.1	next	878
10.341	ln::topo::adj_higher_face_fwd_iter< D > Class Template Reference	878
10.341.1	Detailed Description	879
10.341.2	Constructor & Destructor Documentation	879
10.341.2.1	adj_higher_face_fwd_iter	879
10.341.3	Member Function Documentation	879
10.341.3.1	next	879
10.342	ln::topo::adj_lower_dim_connected_n_face_bkd_iter< D > Class Template Reference	879
10.342.1	Detailed Description	880
10.342.2	Constructor & Destructor Documentation	880
10.342.2.1	adj_lower_dim_connected_n_face_bkd_iter	880
10.342.3	Member Function Documentation	880
10.342.3.1	next	880
10.343	ln::topo::adj_lower_dim_connected_n_face_fwd_iter< D > Class Template Reference	880
10.343.1	Detailed Description	881
10.343.2	Constructor & Destructor Documentation	881
10.343.2.1	adj_lower_dim_connected_n_face_fwd_iter	881
10.343.3	Member Function Documentation	881
10.343.3.1	next	881
10.344	ln::topo::adj_lower_face_bkd_iter< D > Class Template Reference	881
10.344.1	Detailed Description	882
10.344.2	Constructor & Destructor Documentation	882
10.344.2.1	adj_lower_face_bkd_iter	882
10.344.3	Member Function Documentation	882
10.344.3.1	next	882
10.345	ln::topo::adj_lower_face_fwd_iter< D > Class Template Reference	882
10.345.1	Detailed Description	883
10.345.2	Constructor & Destructor Documentation	883
10.345.2.1	adj_lower_face_fwd_iter	883
10.345.3	Member Function Documentation	883
10.345.3.1	next	883
10.346	ln::topo::adj_lower_higher_face_bkd_iter< D > Class Template Reference	883
10.346.1	Detailed Description	884
10.346.2	Constructor & Destructor Documentation	884
10.346.2.1	adj_lower_higher_face_bkd_iter	884
10.346.3	Member Function Documentation	884
10.346.3.1	next	884

10.347.1	In::topo::adj_lower_higher_face_fwd_iter< D > Class Template Reference	884
10.347.1	Detailed Description	885
10.347.2	Constructor & Destructor Documentation	885
10.347.2.1	adj_lower_higher_face_fwd_iter	885
10.347.3	Member Function Documentation	885
10.347.3.1	next	885
10.348.1	In::topo::adj_m_face_bkd_iter< D > Class Template Reference	885
10.348.1	Detailed Description	886
10.348.2	Constructor & Destructor Documentation	886
10.348.2.1	adj_m_face_bkd_iter	886
10.348.2.2	adj_m_face_bkd_iter	886
10.348.3	Member Function Documentation	886
10.348.3.1	next	886
10.349.1	In::topo::adj_m_face_fwd_iter< D > Class Template Reference	887
10.349.1	Detailed Description	887
10.349.2	Constructor & Destructor Documentation	887
10.349.2.1	adj_m_face_fwd_iter	887
10.349.2.2	adj_m_face_fwd_iter	887
10.349.3	Member Function Documentation	888
10.349.3.1	next	888
10.350.1	In::topo::algebraic_face< D > Class Template Reference	888
10.350.1	Detailed Description	889
10.350.2	Constructor & Destructor Documentation	889
10.350.2.1	algebraic_face	889
10.350.2.2	algebraic_face	890
10.350.2.3	algebraic_face	890
10.350.2.4	algebraic_face	890
10.350.3	Member Function Documentation	890
10.350.3.1	cplx	890
10.350.3.2	data	890
10.350.3.3	dec_face_id	890
10.350.3.4	dec_n	890
10.350.3.5	face_id	890
10.350.3.6	higher_dim_adj_faces	891
10.350.3.7	nc_face_id	891
10.350.3.8	nc_n	891
10.350.3.9	invalidate	891
10.350.3.10	is_valid	891
10.350.3.11	lower_dim_adj_faces	891
10.350.3.12		891

10.350.3.1	set_cplx	891
10.350.3.1	set_face_id	892
10.350.3.1	set_n	892
10.350.3.1	set_sign	892
10.350.3.1	sign	892
10.351	In::topo::algebraic_n_face< N, D > Class Template Reference	892
10.351.1	Detailed Description	893
10.351.2	Constructor & Destructor Documentation	894
10.351.2.1	algebraic_n_face	894
10.351.2.2	algebraic_n_face	894
10.351.2.3	algebraic_n_face	894
10.351.3	Member Function Documentation	894
10.351.3.1	cplx	894
10.351.3.2	data	894
10.351.3.3	dec_face_id	894
10.351.3.4	face_id	894
10.351.3.5	higher_dim_adj_faces	895
10.351.3.6	inc_face_id	895
10.351.3.7	invalidate	895
10.351.3.8	is_valid	895
10.351.3.9	lower_dim_adj_faces	895
10.351.3.10	 	895
10.351.3.11	set_cplx	895
10.351.3.12	set_face_id	895
10.351.3.13	set_sign	896
10.351.3.14	sign	896
10.352	In::topo::center_only_iter< D > Class Template Reference	896
10.352.1	Detailed Description	896
10.352.2	Constructor & Destructor Documentation	897
10.352.2.1	center_only_iter	897
10.352.3	Member Function Documentation	897
10.352.3.1	next	897
10.353	In::topo::centered_bkd_iter_adapter< D, I > Class Template Reference	897
10.353.1	Detailed Description	897
10.353.2	Constructor & Destructor Documentation	898
10.353.2.1	centered_bkd_iter_adapter	898
10.353.3	Member Function Documentation	898
10.353.3.1	next	898
10.354	In::topo::centered_fwd_iter_adapter< D, I > Class Template Reference	898
10.354.1	Detailed Description	898

10.354.2	Constructor & Destructor Documentation	899
10.354.2.1	centered_fwd_iter_adapter	899
10.354.3	Member Function Documentation	899
10.354.3.1	next	899
10.355	ln::topo::complex< D > Class Template Reference	899
10.355.1	Detailed Description	900
10.355.2	Member Typedef Documentation	900
10.355.2.1	bkd_citer	900
10.355.2.2	fwd_citer	900
10.355.3	Constructor & Destructor Documentation	900
10.355.3.1	complex	900
10.355.4	Member Function Documentation	900
10.355.4.1	add_face	900
10.355.4.2	add_face	901
10.355.4.3	addr	901
10.355.4.4	faces	901
10.355.4.5	faces_of_dim	901
10.355.4.6	faces_of_static_dim	901
10.355.4.7	print	901
10.355.4.8	print_faces	902
10.356	ln::topo::face< D > Class Template Reference	902
10.356.1	Detailed Description	903
10.356.2	Constructor & Destructor Documentation	903
10.356.2.1	face	903
10.356.2.2	face	903
10.356.2.3	face	903
10.356.3	Member Function Documentation	904
10.356.3.1	cplx	904
10.356.3.2	data	904
10.356.3.3	dec_face_id	904
10.356.3.4	dec_n	904
10.356.3.5	face_id	904
10.356.3.6	higher_dim_adj_faces	904
10.356.3.7	nc_face_id	904
10.356.3.8	nc_n	904
10.356.3.9	invalidate	905
10.356.3.10	is_valid	905
10.356.3.11	lower_dim_adj_faces	905
10.356.3.12		905
10.356.3.13	set_cplx	905

10.356.3.1	set_face_id	905
10.356.3.1	set_n	905
10.357	ln::topo::face_bkd_iter< D > Class Template Reference	905
10.357.1	Detailed Description	906
10.357.2	Constructor & Destructor Documentation	906
10.357.2.1	face_bkd_iter	906
10.357.3	Member Function Documentation	906
10.357.3.1	next	906
10.357.3.2	start	906
10.358	ln::topo::face_fwd_iter< D > Class Template Reference	907
10.358.1	Detailed Description	907
10.358.2	Constructor & Destructor Documentation	907
10.358.2.1	face_fwd_iter	907
10.358.3	Member Function Documentation	907
10.358.3.1	next	907
10.358.3.2	start	908
10.359	ln::topo::is_n_face< N > Struct Template Reference	908
10.359.1	Detailed Description	909
10.360	ln::topo::is_simple_2d_t< N > Struct Template Reference	909
10.360.1	Detailed Description	909
10.361	ln::topo::is_simple_cell< I > Class Template Reference	909
10.361.1	Detailed Description	911
10.361.2	Member Typedef Documentation	911
10.361.2.1	psite	911
10.361.2.2	result	911
10.361.3	Member Function Documentation	911
10.361.3.1	ln_geom	911
10.361.3.2	operator()	911
10.361.3.3	set_image	911
10.361.4	Member Data Documentation	912
10.361.4.1	ID	912
10.362	ln::topo::n_face< N, D > Class Template Reference	912
10.362.1	Detailed Description	913
10.362.2	Constructor & Destructor Documentation	913
10.362.2.1	n_face	913
10.362.2.2	n_face	913
10.362.3	Member Function Documentation	913
10.362.3.1	cplx	913
10.362.3.2	data	913
10.362.3.3	dec_face_id	914

10.362.3.4	face_id	914
10.362.3.5	higher_dim_adj_faces	914
10.362.3.6	inc_face_id	914
10.362.3.7	invalidate	914
10.362.3.8	s_valid	914
10.362.3.9	lower_dim_adj_faces	914
10.362.3.10	914
10.362.3.11	set_cplx	915
10.362.3.12	set_face_id	915
10.363	ln::topo::n_face_bkd_iter< D > Class Template Reference	915
10.363.1	Detailed Description	915
10.363.2	Member Function Documentation	915
10.363.2.1	ln	915
10.363.2.2	next	916
10.363.2.3	start	916
10.364	ln::topo::n_face_fwd_iter< D > Class Template Reference	916
10.364.1	Detailed Description	916
10.364.2	Member Function Documentation	916
10.364.2.1	ln	917
10.364.2.2	next	917
10.364.2.3	start	917
10.365	ln::topo::n_faces_set< N, D > Class Template Reference	917
10.365.1	Detailed Description	918
10.365.2	Member Typedef Documentation	918
10.365.2.1	faces_type	918
10.365.3	Member Function Documentation	918
10.365.3.1	add	918
10.365.3.2	faces	918
10.365.3.3	reserve	918
10.366	ln::topo::skeleton::is_simple_point< N > Struct Template Reference	918
10.366.1	Detailed Description	918
10.367	ln::topo::static_n_face_bkd_iter< N, D > Class Template Reference	919
10.367.1	Detailed Description	919
10.367.2	Constructor & Destructor Documentation	919
10.367.2.1	static_n_face_bkd_iter	919
10.367.3	Member Function Documentation	920
10.367.3.1	next	920
10.367.3.2	start	920
10.368	ln::topo::static_n_face_fwd_iter< N, D > Class Template Reference	920
10.368.1	Detailed Description	920

10.368.2	Constructor & Destructor Documentation	921
10.368.2.1	static_n_face_fwd_iter	921
10.368.3	Member Function Documentation	921
10.368.3.1	next	921
10.368.3.2	start	921
10.369.0	In::tr_image< S, I, T > Struct Template Reference	921
10.369.1	Detailed Description	922
10.369.2	Member Typedef Documentation	922
10.369.2.1	lvalue	922
10.369.2.2	psite	922
10.369.2.3	rvalue	922
10.369.2.4	site	923
10.369.2.5	skeleton	923
10.369.2.6	value	923
10.369.3	Constructor & Destructor Documentation	923
10.369.3.1	tr_image	923
10.369.4	Member Function Documentation	923
10.369.4.1	domain	923
10.369.4.2	has	923
10.369.4.3	s_valid	923
10.369.4.4	operator()	923
10.369.4.5	set_tr	924
10.369.4.6	tr	924
10.370.0	In::transformed_image< I, F > Struct Template Reference	924
10.370.1	Detailed Description	924
10.370.2	Member Typedef Documentation	925
10.370.2.1	skeleton	925
10.370.3	Constructor & Destructor Documentation	925
10.370.3.1	transformed_image	925
10.370.3.2	transformed_image	925
10.370.4	Member Function Documentation	925
10.370.4.1	domain	925
10.370.4.2	operator transformed_image< const I, F >	925
10.370.4.3	operator()	925
10.370.4.4	operator()	925
10.371.0	In::unproject_image< I, D, F > Struct Template Reference	926
10.371.1	Detailed Description	926
10.371.2	Constructor & Destructor Documentation	926
10.371.2.1	unproject_image	926
10.371.2.2	unproject_image	926

10.371.3	Member Function Documentation	926
10.371.3.1	domain	926
10.371.3.2	operator()	927
10.371.3.3	operator()	927
10.372	std::in::adjacency_matrix< V > Class Template Reference	927
10.372.1	Detailed Description	927
10.372.2	Constructor & Destructor Documentation	927
10.372.2.1	adjacency_matrix	927
10.372.2.2	adjacency_matrix	927
10.373	std::in::array< T > Class Template Reference	928
10.373.1	Detailed Description	929
10.373.2	Member Typedef Documentation	930
10.373.2.1	bkd_eiter	930
10.373.2.2	eiter	930
10.373.2.3	element	930
10.373.2.4	fwd_eiter	930
10.373.2.5	result	930
10.373.3	Constructor & Destructor Documentation	930
10.373.3.1	array	930
10.373.3.2	array	930
10.373.3.3	array	930
10.373.4	Member Function Documentation	931
10.373.4.1	append	931
10.373.4.2	append	931
10.373.4.3	clear	931
10.373.4.4	fill	931
10.373.4.5	is_empty	931
10.373.4.6	last	931
10.373.4.7	last	931
10.373.4.8	memory_size	932
10.373.4.9	elements	932
10.373.4.10	operator()	932
10.373.4.11	operator()	932
10.373.4.12	operator[]	932
10.373.4.13	operator[]	932
10.373.4.14	reserve	933
10.373.4.15	size	933
10.373.4.16	size	933
10.373.4.17	size	933
10.373.4.18	std_vector	933

10.374	In::util::branch< T > Class Template Reference	933
10.374.1	Detailed Description	934
10.374.2	Constructor & Destructor Documentation	934
10.374.2.1	branch	934
10.374.3	Member Function Documentation	934
10.374.3.1	apex	934
10.374.3.2	util_tree	934
10.375	In::util::branch_iter< T > Class Template Reference	934
10.375.1	Detailed Description	935
10.375.2	Member Function Documentation	935
10.375.2.1	deepness	935
10.375.2.2	invalidate	935
10.375.2.3	is_valid	935
10.375.2.4	next	935
10.375.2.5	operator util::tree_node< T > &	935
10.375.2.6	start	936
10.376	In::util::branch_iter_ind< T > Class Template Reference	936
10.376.1	Detailed Description	936
10.376.2	Member Function Documentation	936
10.376.2.1	deepness	936
10.376.2.2	invalidate	936
10.376.2.3	is_valid	937
10.376.2.4	next	937
10.376.2.5	operator util::tree_node< T > &	937
10.376.2.6	start	937
10.377	In::util::couple< T, U > Class Template Reference	937
10.377.1	Detailed Description	938
10.377.2	Member Function Documentation	938
10.377.2.1	change_both	938
10.377.2.2	change_first	938
10.377.2.3	change_second	938
10.377.2.4	first	938
10.377.2.5	second	939
10.378	In::util::eat Struct Reference	939
10.378.1	Detailed Description	939
10.379	In::util::edge< G > Class Template Reference	939
10.379.1	Detailed Description	940
10.379.2	Member Typedef Documentation	940
10.379.2.1	category	940
10.379.2.2	graph_t	941

10.379.2.3d_t	941
10.379.2.4d_value_t	941
10.379.3Constructor & Destructor Documentation	941
10.379.3.1edge	941
10.379.4Member Function Documentation	941
10.379.4.1change_graph	941
10.379.4.2graph	941
10.379.4.3d	941
10.379.4.4invalidate	941
10.379.4.5s_valid	942
10.379.4.6th_nbh_edge	942
10.379.4.7max_nbh_edges	942
10.379.4.8operator edge_id_t	942
10.379.4.9update_id	942
10.379.4.10	942
10.379.4.11	942
10.379.4.12_other	942
10.380In::util::fibonacci_heap< P, T > Class Template Reference	943
10.380.1Detailed Description	944
10.380.2Constructor & Destructor Documentation	944
10.380.2.1fibonacci_heap	944
10.380.2.2fibonacci_heap	944
10.380.3Member Function Documentation	944
10.380.3.1clear	944
10.380.3.2front	944
10.380.3.3s_empty	944
10.380.3.4s_valid	945
10.380.3.5elements	945
10.380.3.6operator=	945
10.380.3.7pop_front	945
10.380.3.8push	945
10.380.3.9push	945
10.381In::util::graph Class Reference	945
10.381.1Detailed Description	947
10.381.2Member Typedef Documentation	947
10.381.2.1edge_fwd_iter	947
10.381.2.2edge_nbh_edge_fwd_iter	947
10.381.2.3edges_set_t	947
10.381.2.4edges_t	947
10.381.2.5vertex_nbh_edge_fwd_iter	948

10.381.2.6	vertex_nbh_vertex_fwd_iter	948
10.381.2.7	vertices_t	948
10.381.3	Constructor & Destructor Documentation	948
10.381.3.1	graph	948
10.381.3.2	graph	948
10.381.4	Member Function Documentation	948
10.381.4.1	add_edge	948
10.381.4.2	add_vertex	948
10.381.4.3	add_vertices	949
10.381.4.4	e_ith_nbh_edge	949
10.381.4.5	e_nmax	949
10.381.4.6	edge	949
10.381.4.7	edge	949
10.381.4.8	edges	949
10.381.4.9	has_e	950
10.381.4.10	has_v	950
10.381.4.11	is_subgraph_of	950
10.381.4.12	is	950
10.381.4.13	is	950
10.381.4.14	ith_nbh_edge	950
10.381.4.15	ith_nbh_vertex	950
10.381.4.16	nmax	951
10.381.4.17	vertex	951
10.382	Inn::util::greater_point< I > Class Template Reference	951
10.382.1	Detailed Description	951
10.382.2	Member Function Documentation	951
10.382.2.1	operator()	951
10.383	Inn::util::greater_psite< I > Class Template Reference	951
10.383.1	Detailed Description	952
10.383.2	Member Function Documentation	952
10.383.2.1	operator()	952
10.384	Inn::util::head< T, R > Class Template Reference	952
10.384.1	Detailed Description	952
10.385	Inn::util::ignore Struct Reference	952
10.385.1	Detailed Description	953
10.386	Inn::util::ilcell< T > Struct Template Reference	953
10.386.1	Detailed Description	953
10.387	Inn::util::line_graph< G > Class Template Reference	953
10.387.1	Detailed Description	955
10.387.2	Member Typedef Documentation	955

10.387.2.1edge_fwd_iter	955
10.387.2.2edge_nbh_edge_fwd_iter	955
10.387.2.3edges_t	955
10.387.2.4vertex_nbh_edge_fwd_iter	955
10.387.2.5vertex_nbh_vertex_fwd_iter	955
10.387.2.6vertices_t	956
10.387.3Member Function Documentation	956
10.387.3.1e_ith_nbh_edge	956
10.387.3.2e_nmax	956
10.387.3.3edge	956
10.387.3.4graph	956
10.387.3.5has	956
10.387.3.6has	956
10.387.3.7has_e	956
10.387.3.8has_v	957
10.387.3.9is_subgraph_of	957
10.387.3.10	957
10.387.3.11	957
10.387.3.12_ith_nbh_edge	957
10.387.3.13_ith_nbh_vertex	957
10.387.3.14_nmax	957
10.387.3.15vertex	957
10.388Inn::util::nil Struct Reference	958
10.388.1Detailed Description	958
10.389Inn::util::node< T, R > Class Template Reference	958
10.389.1Detailed Description	958
10.390Inn::util::object_id< Tag, V > Class Template Reference	958
10.390.1Detailed Description	959
10.390.2Member Typedef Documentation	959
10.390.2.1value_t	959
10.390.3Constructor & Destructor Documentation	960
10.390.3.1object_id	960
10.391Inn::util::ord< T > Struct Template Reference	960
10.391.1Detailed Description	960
10.392Inn::util::ord_pair< T > Struct Template Reference	960
10.392.1Detailed Description	961
10.392.2Member Function Documentation	961
10.392.2.1change_both	961
10.392.2.2change_first	961
10.392.2.3change_second	962

10.392.2.4first	962
10.392.2.5second	962
10.393.1In::util::pix< I > Struct Template Reference	962
10.393.1Detailed Description	962
10.393.2Member Typedef Documentation	963
10.393.2.1psite	963
10.393.2.2value	963
10.393.3Constructor & Destructor Documentation	963
10.393.3.1pix	963
10.393.4Member Function Documentation	963
10.393.4.1ima	963
10.393.4.2p	963
10.393.4.3v	963
10.394.1In::util::set< T > Class Template Reference	964
10.394.1Detailed Description	965
10.394.2Member Typedef Documentation	965
10.394.2.1bkd_eiter	965
10.394.2.2eiter	966
10.394.2.3element	966
10.394.2.4fwd_eiter	966
10.394.3Constructor & Destructor Documentation	966
10.394.3.1set	966
10.394.4Member Function Documentation	966
10.394.4.1clear	966
10.394.4.2first_element	966
10.394.4.3has	966
10.394.4.4insert	967
10.394.4.5insert	967
10.394.4.6is_empty	967
10.394.4.7last_element	967
10.394.4.8memory_size	968
10.394.4.9elements	968
10.394.4.10operator[]	968
10.394.4.11remove	968
10.394.4.12rd_vector	968
10.395.1In::util::site_pair< P > Class Template Reference	969
10.395.1Detailed Description	969
10.395.2Member Function Documentation	969
10.395.2.1first	969
10.395.2.2pair	970

10.395.2.3second	970
10.396In::util::soft_heap< T, R > Class Template Reference	970
10.396.1Detailed Description	971
10.396.2Member Typedef Documentation	971
10.396.2.1element	971
10.396.3Constructor & Destructor Documentation	971
10.396.3.1soft_heap	971
10.396.3.2~soft_heap	971
10.396.4Member Function Documentation	972
10.396.4.1clear	972
10.396.4.2s_empty	972
10.396.4.3s_valid	972
10.396.4.4elements	972
10.396.4.5pop_front	972
10.396.4.6push	972
10.396.4.7push	972
10.397In::util::timer Class Reference	972
10.397.1Detailed Description	973
10.398In::util::tracked_ptr< T > Struct Template Reference	973
10.398.1Detailed Description	974
10.398.2Constructor & Destructor Documentation	974
10.398.2.1tracked_ptr	974
10.398.2.2tracked_ptr	974
10.398.2.3~tracked_ptr	974
10.398.3Member Function Documentation	974
10.398.3.1operator bool	974
10.398.3.2operator!	974
10.398.3.3operator->	975
10.398.3.4operator->	975
10.398.3.5operator=	975
10.398.3.6operator=	975
10.399In::util::tree< T > Class Template Reference	975
10.399.1Detailed Description	976
10.399.2Constructor & Destructor Documentation	976
10.399.2.1tree	976
10.399.2.2tree	976
10.399.3Member Function Documentation	976
10.399.3.1add_tree_down	976
10.399.3.2add_tree_up	976
10.399.3.3check_consistency	976

10.399.3.4	main_branch	977
10.399.3.5	root	977
10.400	util::tree_node< T > Class Template Reference	977
10.400.1	Detailed Description	978
10.400.2	Constructor & Destructor Documentation	978
10.400.2.1	tree_node	978
10.400.2.2	tree_node	978
10.400.3	Member Function Documentation	978
10.400.3.1	add_child	978
10.400.3.2	add_child	979
10.400.3.3	check_consistency	979
10.400.3.4	children	979
10.400.3.5	children	979
10.400.3.6	delete_tree_node	979
10.400.3.7	elt	980
10.400.3.8	elt	980
10.400.3.9	parent	980
10.400.3.10	print	980
10.400.3.11	search	980
10.400.3.12	search_rec	981
10.400.3.13	set_parent	981
10.401	util::vertex< G > Class Template Reference	981
10.401.1	Detailed Description	982
10.401.2	Member Typedef Documentation	983
10.401.2.1	Category	983
10.401.2.2	graph_t	983
10.401.2.3	d_t	983
10.401.2.4	d_value_t	983
10.401.3	Constructor & Destructor Documentation	983
10.401.3.1	vertex	983
10.401.4	Member Function Documentation	983
10.401.4.1	change_graph	983
10.401.4.2	edge_with	983
10.401.4.3	graph	983
10.401.4.4	d	984
10.401.4.5	invalidate	984
10.401.4.6	is_valid	984
10.401.4.7	th_nbh_edge	984
10.401.4.8	th_nbh_vertex	984
10.401.4.9	max_nbh_edges	984

10.401.4.10	max_nbh_vertices	984
10.401.4.11	operator vertex_id_t	984
10.401.4.12	other	985
10.401.4.13	update_id	985
10.402	In::util::yes Struct Reference	985
10.402.1	Detailed Description	985
10.403	In::Value< E > Struct Template Reference	985
10.403.1	Detailed Description	986
10.404	In::value::float01 Class Reference	987
10.404.1	Detailed Description	987
10.404.2	Member Typedef Documentation	988
10.404.2.1	enc	988
10.404.2.2	equiv	988
10.404.3	Constructor & Destructor Documentation	988
10.404.3.1	float01	988
10.404.3.2	float01	988
10.404.3.3	float01	988
10.404.4	Member Function Documentation	988
10.404.4.1	nbits	988
10.404.4.2	operator float	988
10.404.4.3	set_nbits	988
10.404.4.4	to_nbits	989
10.404.4.5	value	989
10.404.4.6	value_ind	989
10.405	In::value::float01_f Struct Reference	989
10.405.1	Detailed Description	989
10.405.2	Constructor & Destructor Documentation	989
10.405.2.1	float01_f	989
10.405.2.2	float01_f	990
10.405.3	Member Function Documentation	990
10.405.3.1	operator float	990
10.405.3.2	operator=	990
10.405.3.3	value	990
10.406	In::value::graylevel< n > Struct Template Reference	990
10.406.1	Detailed Description	992
10.406.2	Constructor & Destructor Documentation	992
10.406.2.1	graylevel	992
10.406.2.2	graylevel	992
10.406.2.3	graylevel	992
10.406.2.4	graylevel	992

10.406.2.5graylevel	992
10.406.3Member Function Documentation	992
10.406.3.1operator=	992
10.406.3.2operator=	992
10.406.3.3operator=	993
10.406.3.4operator=	993
10.406.3.5to_float	993
10.406.3.6value	993
10.407In::value::graylevel_f Struct Reference	993
10.407.1Detailed Description	994
10.407.2Constructor & Destructor Documentation	994
10.407.2.1graylevel_f	994
10.407.2.2graylevel_f	994
10.407.2.3graylevel_f	994
10.407.2.4graylevel_f	994
10.407.2.5graylevel_f	994
10.407.3Member Function Documentation	995
10.407.3.1operator graylevel< n >	995
10.407.3.2operator=	995
10.407.3.3operator=	995
10.407.3.4operator=	995
10.407.3.5operator=	995
10.407.3.6value	995
10.408In::value::int_s< n > Struct Template Reference	995
10.408.1Detailed Description	996
10.408.2Constructor & Destructor Documentation	997
10.408.2.1int_s	997
10.408.2.2int_s	997
10.408.2.3int_s	997
10.408.3Member Function Documentation	997
10.408.3.1operator int	997
10.408.3.2operator=	997
10.408.4Member Data Documentation	997
10.408.4.1one	997
10.408.4.2zero	997
10.409In::value::int_u< n > Struct Template Reference	997
10.409.1Detailed Description	998
10.409.2Constructor & Destructor Documentation	999
10.409.2.1int_u	999
10.409.2.2int_u	999

10.409.2.3int_u	999
10.409.3Member Function Documentation	999
10.409.3.1next	999
10.409.3.2operator unsigned	999
10.409.3.3operator-	999
10.409.3.4operator=	999
10.410In::value::int_u_sat< n > Struct Template Reference	999
10.410.1Detailed Description	1000
10.410.2Constructor & Destructor Documentation	1001
10.410.2.1int_u_sat	1001
10.410.2.2nt_u_sat	1001
10.410.3Member Function Documentation	1001
10.410.3.1operator int	1001
10.410.3.2operator+=	1001
10.410.3.3operator-=	1001
10.410.3.4operator=	1001
10.410.4Member Data Documentation	1001
10.410.4.1one	1001
10.410.4.2zero	1001
10.411In::value::Integer< E > Struct Template Reference	1002
10.411.1Detailed Description	1002
10.412In::value::Integer< void > Struct Template Reference	1002
10.412.1Detailed Description	1002
10.413In::value::label< n > Struct Template Reference	1002
10.413.1Detailed Description	1003
10.413.2Member Typedef Documentation	1003
10.413.2.1enc	1003
10.413.3Constructor & Destructor Documentation	1003
10.413.3.1label	1003
10.413.3.2abel	1004
10.413.3.3abel	1004
10.413.4Member Function Documentation	1004
10.413.4.1next	1004
10.413.4.2operator unsigned	1004
10.413.4.3operator++	1004
10.413.4.4operator--	1004
10.413.4.5operator=	1004
10.413.4.6operator=	1004
10.413.4.7prev	1004
10.414In::value::lut_vec< S, T > Struct Template Reference	1005

10.414.1Detailed Description	1005
10.414.2Member Typedef Documentation	1006
10.414.2.1bkd_viter	1006
10.414.2.2fwd_viter	1006
10.414.2.3value	1006
10.414.3Constructor & Destructor Documentation	1006
10.414.3.1lut_vec	1006
10.414.3.2ut_vec	1006
10.414.3.3ut_vec	1006
10.414.4Member Function Documentation	1006
10.414.4.1has	1006
10.414.4.2index_of	1007
10.414.4.3nvalues	1007
10.414.4.4operator[]	1007
10.415In::value::proxy< I > Class Template Reference	1007
10.415.1Detailed Description	1008
10.415.2Member Typedef Documentation	1008
10.415.2.1enc	1008
10.415.2.2equiv	1008
10.415.3Constructor & Destructor Documentation	1008
10.415.3.1proxy	1008
10.415.3.2proxy	1009
10.415.3.3~proxy	1009
10.415.4Member Function Documentation	1009
10.415.4.1operator=	1009
10.415.4.2operator=	1009
10.415.4.3to_value	1009
10.416In::value::qt::rgb32 Struct Reference	1009
10.416.1Detailed Description	1010
10.416.2Constructor & Destructor Documentation	1010
10.416.2.1rgb32	1010
10.416.2.2rgb32	1010
10.416.2.3rgb32	1010
10.416.2.4rgb32	1010
10.416.3Member Function Documentation	1010
10.416.3.1operator=	1010
10.416.3.2ed	1011
10.416.4Member Data Documentation	1011
10.416.4.1zero	1011
10.417In::value::rgb< n > Struct Template Reference	1011

10.417.1Detailed Description	1011
10.417.2Constructor & Destructor Documentation	1012
10.417.2.1rgb	1012
10.417.2.2gb	1012
10.417.2.3gb	1012
10.417.2.4gb	1012
10.417.3Member Function Documentation	1012
10.417.3.1operator=	1012
10.417.3.2ed	1012
10.417.4Member Data Documentation	1012
10.417.4.1zero	1012
10.418In::value::set< T > Struct Template Reference	1012
10.418.1Detailed Description	1013
10.418.2Member Function Documentation	1013
10.418.2.1the	1013
10.419In::value::sign Class Reference	1013
10.419.1Detailed Description	1014
10.419.2Member Typedef Documentation	1014
10.419.2.1enc	1014
10.419.2.2equiv	1014
10.419.3Constructor & Destructor Documentation	1014
10.419.3.1sign	1014
10.419.3.2sign	1014
10.419.3.3sign	1014
10.419.4Member Function Documentation	1015
10.419.4.1operator int	1015
10.419.4.2operator=	1015
10.419.5Member Data Documentation	1015
10.419.5.1one	1015
10.419.5.2zero	1015
10.420In::value::stack_image< n, I > Struct Template Reference	1015
10.420.1Detailed Description	1016
10.420.2Member Typedef Documentation	1016
10.420.2.1domain_t	1016
10.420.2.2value	1016
10.420.2.3psite	1016
10.420.2.4value	1016
10.420.2.5skeleton	1017
10.420.2.6value	1017
10.420.3Constructor & Destructor Documentation	1017

10.420.3.1stack_image	1017
10.420.4Member Function Documentation	1017
10.420.4.1is_valid	1017
10.420.4.2operator()	1017
10.420.4.3operator()	1017
10.421In::value::super_value< sign > Struct Template Reference	1017
10.421.1Detailed Description	1017
10.422In::value::value_array< T, V > Struct Template Reference	1018
10.422.1Detailed Description	1018
10.422.2Constructor & Destructor Documentation	1018
10.422.2.1value_array	1018
10.422.3Member Function Documentation	1018
10.422.3.1operator()	1018
10.422.3.2operator[]	1019
10.422.3.3set	1019
10.423In::Vertex< E > Struct Template Reference	1019
10.423.1Detailed Description	1019
10.424In::vertex_image< P, V, G > Class Template Reference	1019
10.424.1Detailed Description	1020
10.424.2Member Typedef Documentation	1020
10.424.2.1graph_t	1020
10.424.2.2nbh_t	1020
10.424.2.3site_function_t	1020
10.424.2.4skeleton	1020
10.424.2.5win_t	1021
10.424.3Constructor & Destructor Documentation	1021
10.424.3.1vertex_image	1021
10.424.4Member Function Documentation	1021
10.424.4.1operator()	1021
10.425In::violent_cast_image< T, I > Struct Template Reference	1021
10.425.1Detailed Description	1022
10.425.2Member Typedef Documentation	1022
10.425.2.1value	1022
10.425.2.2value	1022
10.425.2.3skeleton	1022
10.425.2.4value	1022
10.425.3Constructor & Destructor Documentation	1022
10.425.3.1violent_cast_image	1022
10.425.4Member Function Documentation	1022
10.425.4.1operator()	1022

10.425.4.2operator()	1023
10.426Inn::w_window< D, W > Struct Template Reference	1023
10.426.1Detailed Description	1024
10.426.2Member Typedef Documentation	1024
10.426.2.1bkd_qiter	1024
10.426.2.2dpsite	1024
10.426.2.3wd_qiter	1024
10.426.2.4weight	1024
10.426.3Constructor & Destructor Documentation	1024
10.426.3.1w_window	1024
10.426.4Member Function Documentation	1025
10.426.4.1clear	1025
10.426.4.2insert	1025
10.426.4.3s_symmetric	1025
10.426.4.4std_vector	1025
10.426.4.5sym	1025
10.426.4.6w	1025
10.426.4.7weights	1025
10.426.4.8win	1026
10.426.5Friends And Related Function Documentation	1026
10.426.5.1operator<<	1026
10.426.5.2operator==	1026
10.427Inn::Weighted_Window< E > Struct Template Reference	1026
10.427.1Detailed Description	1027
10.427.2Friends And Related Function Documentation	1027
10.427.2.1operator-	1027
10.428Inn::win::backdiag2d Struct Reference	1027
10.428.1Detailed Description	1027
10.428.2Constructor & Destructor Documentation	1028
10.428.2.1backdiag2d	1028
10.428.3Member Function Documentation	1028
10.428.3.1length	1028
10.429Inn::win::ball< G, C > Struct Template Reference	1028
10.429.1Detailed Description	1028
10.429.2Constructor & Destructor Documentation	1029
10.429.2.1ball	1029
10.429.3Member Function Documentation	1029
10.429.3.1diameter	1029
10.430Inn::win::cube3d Struct Reference	1029
10.430.1Detailed Description	1029

10.430.2	Constructor & Destructor Documentation	1030
10.430.2.1	cube3d	1030
10.430.3	Member Function Documentation	1030
10.430.3.1	length	1030
10.431	In::win::cuboid3d Struct Reference	1030
10.431.1	Detailed Description	1031
10.431.2	Constructor & Destructor Documentation	1031
10.431.2.1	cuboid3d	1031
10.431.3	Member Function Documentation	1031
10.431.3.1	depth	1031
10.431.3.2	height	1032
10.431.3.3	volume	1032
10.431.3.4	width	1032
10.432	In::win::diag2d Struct Reference	1032
10.432.1	Detailed Description	1032
10.432.2	Constructor & Destructor Documentation	1032
10.432.2.1	diag2d	1032
10.432.3	Member Function Documentation	1033
10.432.3.1	length	1033
10.433	In::win::line< M, i, C > Struct Template Reference	1033
10.433.1	Detailed Description	1033
10.433.2	Member Enumeration Documentation	1034
10.433.2.1	anonymous enum	1034
10.433.3	Constructor & Destructor Documentation	1034
10.433.3.1	line	1034
10.433.4	Member Function Documentation	1034
10.433.4.1	length	1034
10.433.4.2	size	1034
10.434	In::win::multiple< W, F > Class Template Reference	1034
10.434.1	Detailed Description	1034
10.435	In::win::multiple_size< n, W, F > Class Template Reference	1035
10.435.1	Detailed Description	1035
10.436	In::win::octagon2d Struct Reference	1035
10.436.1	Detailed Description	1035
10.436.2	Constructor & Destructor Documentation	1036
10.436.2.1	octagon2d	1036
10.436.3	Member Function Documentation	1036
10.436.3.1	area	1036
10.436.3.2	length	1036
10.437	In::win::rectangle2d Struct Reference	1036

10.437.1Detailed Description	1037
10.437.2Constructor & Destructor Documentation	1037
10.437.2.1rectangle2d	1037
10.437.3Member Function Documentation	1037
10.437.3.1area	1037
10.437.3.2height	1037
10.437.3.3std_vector	1037
10.437.3.4width	1037
10.438In::Window< E > Struct Template Reference	1038
10.438.1Detailed Description	1038
10.439In::window< D > Class Template Reference	1038
10.439.1Detailed Description	1040
10.439.2Member Typedef Documentation	1040
10.439.2.1bkd_qiter	1040
10.439.2.2fwd_qiter	1040
10.439.2.3qiter	1040
10.439.2.4regular	1040
10.439.3Constructor & Destructor Documentation	1040
10.439.3.1window	1040
10.439.4Member Function Documentation	1040
10.439.4.1clear	1040
10.439.4.2delta	1040
10.439.4.3dp	1041
10.439.4.4has	1041
10.439.4.5insert	1041
10.439.4.6insert	1041
10.439.4.7insert	1041
10.439.4.8s_centered	1041
10.439.4.9s_empty	1041
10.439.4.10s_symmetric	1042
10.439.4.11print	1042
10.439.4.12size	1042
10.439.4.13std_vector	1042
10.439.4.14sym	1042
10.439.5Friends And Related Function Documentation	1042
10.439.5.1operator==	1042
10.440In::world::inter_pixel::is_separator Struct Reference	1042
10.440.1Detailed Description	1043
10.441point< G, C > Struct Template Reference	1043
10.441.1Detailed Description	1044

10.442	point< G, C > Struct Template Reference	1044
10.442.1	Detailed Description	1044
10.443	site_set_impl< Sc > Struct Template Reference	1044
10.443.1	Detailed Description	1044
10.444	subject_point_impl< P, E > Struct Template Reference	1044
10.444.1	Detailed Description	1045
10.445	trait::graph< I > Struct Template Reference	1045
10.445.1	Detailed Description	1045
10.446	trait::graph< mln::complex_image< 1, G, V > > Struct Template Reference	1045
10.446.1	Detailed Description	1045
10.447	trait::graph< mln::image2d< T > > Struct Template Reference	1046
10.447.1	Detailed Description	1046
10.448	tree_node< T > Class Template Reference	1046
10.448.1	Detailed Description	1046
10.449	p_leaf_piter< T > Struct Template Reference	1046
10.449.1	Detailed Description	1046
10.450	p_node_piter< T > Struct Template Reference	1046
10.450.1	Detailed Description	1047
10.451	p_site_piter< T > Struct Template Reference	1047
10.451.1	Detailed Description	1047

Index

1047

Chapter 1

Documentation of milena

1.1 Introduction

This is the documentation of Milena.

1.2 Overview of Milena.

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

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

1.3 Copyright and License.

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

This documentation is part of Olena.

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

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

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

Chapter 2

Quick Reference Guide

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

2.1 Installation

2.1.1 Requirements

2.1.1.1 To compile the user examples

2.1.1.2 To compile the documentation (Optional)

2.1.1.3 To develop in Olena

2.1.2 Getting Olena

2.1.3 Building Olena

2.2 Foreword

2.2.1 Generality

2.2.2 Directory hierarchy

2.2.3 Writing and compiling a program with Olena

2.3 Site

2.4 Site set

Iterators

2.4.1 Basic interface

2.4.2 Optional interface

```

box2d b(2,3);

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

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

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

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

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

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

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

```

2.5 Image

2.5.1 Definition

2.5.2 Possible image types

2.5.3 Possible value types

2.5.4 Domain

```

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

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

```

```

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

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

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

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

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

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

point2d p(9,9);

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

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

False

```

2.5.5 Border and extension

2.5.5.1 Image border

```

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

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

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

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

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

=====

- | |
| - -
| | -

```

2.5.5.2 Generality on image extension

imamorphed

2.5.5.3 Different extensions

```
image2d<rgb8> lena;
```

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

2.5.5.3.1 Extension with a value

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

2.5.5.3.2 Extension with a function

```
namespace mln
{
    struct my_ext : public Function_v2v<my_ext>
    {
        typedef value::rgb8 result;

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

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

2.5.5.3.3 Extension with an image

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

// Default border size is set to 0.

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

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

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

std::cout << "imal.has(2, 5) : "
    << imal.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;

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

```

border::thickness = 30;

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

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

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

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

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

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

my_routine(ima | ima.domain());

```

2.5.6 Interface

2.5.7 Load and save images

```

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

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

```

2.5.8 Create an image

```

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

2.5.9 Access and modify values

```

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

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

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

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

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

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

```

Iterators

2.5.10 Image size

```

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

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

nrows = 11 - ncols = 13

```

2.6 Structural elements: Window and neighborhood

2.6.1 Define an element

2.6.1.1 Window

2.6.1.2 Neighborhood

```

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

```

2.6.1.3 Custom structural elements

```

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

o -
o X
o -

bool b[9] = { 1, 0, 0,

```



```

        1, 0, 0,
        1, 0, 0 };

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

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

```

2.6.1.4 Conversion between Neighborhoods and Windows

2.7 Sites, psites and dpoints

2.7.1 Need for site

```

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

```

2.7.2 Need for psite

```

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

```

```

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

```

```

mln_VAR(ima, my_values | arr);

```

```

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

```

```

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

```

2.7.3 From psite to site

2.7.4 Dpoint

```

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

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

```

```

(0,1)

```

2.8 Iterators

```

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

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

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

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

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

```

Useful macros

2.9 Memory management

```

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

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

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

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

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

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

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

3 - 0
0

```

2.10 Basic routines

2.10.1 Fill

```

image2d<char> imga(5, 5);

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

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

```

2.10.2 Paste

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

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

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

debug::println(imga);

98 98 98 98
98 98 98 98
98 98 98 98
```

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

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

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

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

2.10.3 Blobs

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

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

2.10.4 Logical not

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

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

logical::not_inplace(ima);
```

2.10.5 Compute

2.10.5.1 Accumulators

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

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

2.10.5.2 Example with labeling::compute()

```

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

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

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

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

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

unsigned nsites = geom::nsites(ima);

```

2.10.6 Working with parts of an image

```

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

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

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

```

2.10.6.1 Restrict an image with a site set

```

p_array<point2d> arr;

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

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

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

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

```

-

-

2.10.6.2 Restrict an image with a predicate

```

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

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

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

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

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

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

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

```

2.10.6.3 Restrict an image with a C function

```

bool row_oddity(mIn::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

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

1
1 1 1

box2d b1(1,0, 1, 2);
mIn_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.
mIn_VAR(imad, imab1 | b2);
debug::println(imad);

ima / sub_D

```

2.11 Input / Output

2.11.1 ImageMagick

2.11.2 GDCM

2.12 Graphs and images

2.12.1 Description

2.12.2 Example

```

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

util::graph g;

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

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

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

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

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

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

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

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

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

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

```

```

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

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

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

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

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

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

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

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

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

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

// Iterator on edges adjacent to e.
mln_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.
mln_vertex_iter_(util::graph) v(g);

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

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

```

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

2.13 Useful global variables

2.14 Useful macros

2.14.1 Variable declaration macros

2.14.2 Iterator type macros

2.14.2.1 Default iterator types

2.14.2.2 Forward iterator types

2.14.2.3 Backward iterators

2.14.2.4 Graph iterators

2.15 Common Compilation Errors

Chapter 3

Tutorial

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

3.1 Welcome

3.1.1 How to learn Milena

3.1.2 Obtaining the library

3.1.3 Downloading the library

3.1.3.1 Downloading from SVN

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

3.1.3.2 Downloading packaged releases

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

3.1.4 Join the mailing lists

3.1.5 Directory structure

3.1.6 Documentation

3.1.7 Community and Support

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

3.1.8 Project status

[Join the mailing lists](#)

3.1.9 A brief history of Milena

3.1.10 Contacts

[Installation](#)

3.2 Installation

3.2.1 Bootstrap (SVN Sources)

[Configure](#)

3.2.2 Configure

3.2.3 Install

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

3.2.4 Optional compilation

3.2.4.1 Examples

3.2.4.2 Tools

3.2.4.3 Tests

3.2.5 Installation content

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

3.3 Getting started with Milena

3.3.1 Getting familiar with genericity

```
// Java or C -like code

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

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

fill(ima, literal::green);
```

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

3.3.2 First generic algorithm

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

#include <mln/data/fill.hh>

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

#include <mln/data/compare.hh>

#include <mln/util/array.hh>

#include <mln/value/label_8.hh>

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

#include <mln/pw/all.hh>

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

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

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

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

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

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

        trace::exiting("my_algorithm");
        return output;
    }
} // end of namespace mln

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

    trace::entering("my_algorithm");
```

Debug hints

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

```

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

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

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

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

```

3.3.3 Compilation

3.3.3.1 Include path

3.3.3.2 Library linking

Input / Output

3.3.3.3 Disable Debug

3.3.3.4 Compiler optimization flags

3.3.3.4.1 GCC

3.3.3.4.2 Other compilers

3.3.4 Debug hints

3.3.4.1 Using assertions and GDB

3.3.4.2 Traces

```

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

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

trace::quiet = true;

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

```

3.3.4.3 Debug routines

```

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

```

```

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

```

```

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

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

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

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

```

InstallationData representation

3.4 Data representation

3.4.1 Sites

```

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

(3,3)

```

3.4.2 Site sets

Site set

3.4.2.1 Creating a site set

```

box2d b(4,4);

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

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

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

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

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

```

3.4.2.2 Getting access to sites

3.4.3 Images

3.4.3.1 Creating an image

3.4.3.2 Reading an image from a file

3.4.3.3 Accessing data

[Getting started with MilenaLoad and save images](#)

3.5 Load and save images

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

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

[Load and save imagesData representationCreate your first image](#)

3.6 Create your first image

See Also

[tuto2_first_image.cc](#)

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

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

[Create an image](#)

```
debug::println(ima);

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

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

[Possible value typesLoad and save imagesRead and write images](#)

3.7 Read and write images

See Also

[tuto3_rw_image.cc](#)

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

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

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

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

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

data::fill(ima, lena);

data::paste(ima, lena);
```

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

3.8 Regions of interest

See Also

[tuto4_genericity_and_algorithms.cc](#)

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

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

3.8.1 Image domain restricted by a site set

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

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

3.8.2 Image domain restricted by a function

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

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

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

3.8.3 Image domain restricted by a mask

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

3.8.4 Image domain restricted by a predicate

```
image2d<bool> lena_bw = binarization::binarization(lena, keep_specific_colors  
());  
value::label_8 nlabels;  
image2d<value::label_8> label = labeling::blobs(lena_bw, c8(), nlabels);  
  
data::fill((lena | (pw::value(label) == pw::cst(0u))).rw(), literal::blue);
```

[Read and write images](#)

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

Types	103
Graphes	96
Images	97
Basic types	98
Image morphers	99
Values morphers	100
Domain morphers	101
Identity morphers	102
Neighborhoods	108
1D neighborhoods	109
2D neighborhoods	110
3D neighborhoods	112
Site sets	115
Basic types	116
Graph based	117
Complex based	118
Sparse types	119
Queue based	120
Utilities	121
Windows	122
1D windows	123
2D windows	124
3D windows	127
N-D windows	130
Multiple windows	131
Accumulators	104
On site sets	91
On images	92
On values	93
Multiple accumulators	95
Routines	105
Canvas	106
Functions	107
v2w2v functions	132
v2w_w2v functions	133
vv2b functions	134

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	135
mln::accu	Namespace of accumulators	167
mln::accu::image	Namespace of accumulator image routines	170
mln::accu::impl	Implementation namespace of accumulator namespace	170
mln::accu::logic	Namespace of logical accumulators	170
mln::accu::math	Namespace of mathematic accumulators	171
mln::accu::meta::logic	Namespace of logical meta-accumulators	171
mln::accu::meta::math	Namespace of mathematic meta-accumulators	172
mln::accu::meta::shape	Namespace of shape meta-accumulators	172
mln::accu::meta::stat	Namespace of statistical meta-accumulators	172
mln::accu::shape	Namespace of shape accumulators	173
mln::accu::stat	Namespace of statistical accumulators	173
mln::algebra	Namespace of algebraic structure	175
mln::arith	Namespace of arithmetic	177
mln::arith::impl	Implementation namespace of arith namespace	188
mln::arith::impl::generic	Generic implementation namespace of arith namespace	189
mln::binarization	Namespace of "point-wise" expression tools	189
mln::border	Namespace of routines related to image virtual (outer) border	190
mln::border::impl	Implementation namespace of border namespace	193

mln::border::impl::generic	Generic implementation namespace of border namespace	193
mln::canvas	Namespace of canvas	193
mln::canvas::browsing	Namespace of browsing canvas	194
mln::canvas::impl	Implementation namespace of canvas namespace	195
mln::canvas::labeling	Namespace of labeling canvas	195
mln::canvas::labeling::impl	Implementation namespace of labeling canvas namespace	196
mln::canvas::morpho	Namespace of morphological canvas	196
mln::convert	Namespace of conversion routines	196
mln::data	Namespace of image processing routines related to pixel data	202
mln::data::approx	Namespace of image processing routines related to pixel levels with approximation	212
mln::data::approx::impl	Implementation namespace of data::approx namespace	213
mln::data::impl	Implementation namespace of data namespace	213
mln::data::impl::generic	Generic implementation namespace of data namespace	216
mln::data::naive	Namespace of image processing routines related to pixel levels with naive approach	219
mln::data::naive::impl	Implementation namespace of data::naive namespace	220
mln::debug	Namespace of routines that help to debug	220
mln::debug::impl	Implementation namespace of debug namespace	225
mln::def	Namespace for core definitions	225
mln::display	Namespace of routines that help to display images	226
mln::display::impl	Implementation namespace of display namespace	226
mln::display::impl::generic	Generic implementation namespace of display namespace	226
mln::doc	The namespace mln::doc is only for documentation purpose	227
mln::draw	Namespace of drawing routines	228
mln::estim	Namespace of estimation materials	230
mln::extension	Namespace of extension tools	232
mln::fun	Namespace of functions	234
mln::fun::access	Namespace for access functions	235
mln::fun::i2v	Namespace of integer-to-value functions	235
mln::fun::n2v	Namespace of functions from nil to value	236

mln::fun::p2b	Namespace of functions from point to boolean	236
mln::fun::p2p	Namespace of functions from grid point to grid point	237
mln::fun::p2v	Namespace of functions from point to value	237
mln::fun::stat	Namespace of statistical functions	237
mln::fun::v2b	Namespace of functions from value to logic value	237
mln::fun::v2i	Namespace of value-to-integer functions	237
mln::fun::v2v	Namespace of functions from value to value	238
mln::fun::v2w2v	Namespace of bijective functions	239
mln::fun::v2w_w2v	Namespace of functions from value to value	239
mln::fun::vv2b	Namespace of functions from value to value	239
mln::fun::vv2v	Namespace of functions from a couple of values to a value	240
mln::fun::x2p	Namespace of functions from point to value	240
mln::fun::x2v	Namespace of functions from vector to value	241
mln::fun::x2x	Namespace of functions from vector to vector	241
mln::geom	Namespace of all things related to geometry	241
mln::geom::impl	Implementation namespace of geom namespace	252
mln::graph	Namespace of graph related routines	253
mln::grid	Namespace of grids definitions	255
mln::histo	Namespace of histograms	255
mln::histo::impl	Implementation namespace of histo namespace	256
mln::histo::impl::generic	Generic implementation namespace of histo namespace	257
mln::impl	Implementation namespace of mln namespace	257
mln::io	Namespace of input/output handling	257
mln::io::cloud	Namespace of cloud input/output handling	258
mln::io::dicom	Namespace of DICOM input/output handling	259
mln::io::dump	Namespace of dump input/output handling	260
mln::io::fits	Namespace of fits input/output handling	261
mln::io::fld	Namespace of pgm input/output handling	261
mln::io::magick	Namespace of magick input/output handling	263

mln::io::off	Namespace of off input/output handling	264
mln::io::pbm	Namespace of pbm input/output handling	265
mln::io::pbm::impl	Namespace of pbm implementation details	266
mln::io::pbms	Namespace of pbms input/output handling	267
mln::io::pbms::impl	Namespace of pbms implementation details	267
mln::io::pfm	Namespace of pfm input/output handling	267
mln::io::pfm::impl	Implementation namespace of pfm namespace	269
mln::io::pgm	Namespace of pgm input/output handling	269
mln::io::pgms	Namespace of pgms input/output handling	270
mln::io::plot	Namespace of plot input/output handling	270
mln::io::pnm	Namespace of pnm input/output handling	272
mln::io::pnm::impl	Namespace of pnm's implementation details	273
mln::io::pnms	Namespace of pnms input/output handling	273
mln::io::ppm	Namespace of ppm input/output handling	274
mln::io::ppms	Namespace of ppms input/output handling	275
mln::io::raw	Namespace of raw input/output handling	276
mln::io::tiff	Namespace of tiff input/output handling	277
mln::io::txt	Namespace of txt input/output handling	277
mln::labeling	Namespace of labeling routines	278
mln::labeling::impl	Implementation namespace of labeling namespace	292
mln::labeling::impl::generic	Generic implementation namespace of labeling namespace	293
mln::linear	Namespace of linear image processing routines	295
mln::linear::impl	Namespace of linear image processing routines implementation details	299
mln::linear::local	Specializations of local linear routines	299
mln::linear::local::impl	Namespace of local linear routines implementation details	300
mln::literal	Namespace of literals	300
mln::logical	Namespace of logic	305
mln::logical::impl	Implementation namespace of logical namespace	307
mln::logical::impl::generic	Generic implementation namespace of logical namespace	308

mln::make	Namespace of routines that help to make Milena's objects	308
mln::math	Namespace of mathematical routines	329
mln::metal	Namespace of meta-programming tools	330
mln::metal::impl	Implementation namespace of metal namespace	331
mln::metal::math	Namespace of static mathematical functions	331
mln::metal::math::impl	Implementation namespace of metal::math namespace	331
mln::morpho	Namespace of mathematical morphology routines	331
mln::morpho::approx	Namespace of approximate mathematical morphology routines	340
mln::morpho::attribute	Namespace of attributes used in mathematical morphology	340
mln::morpho::closing::approx	Namespace of approximate mathematical morphology closing routines	340
mln::morpho::elementary	Namespace of image processing routines of elementary mathematical morphology	341
mln::morpho::impl	Namespace of mathematical morphology routines implementations	342
mln::morpho::impl::generic	Namespace of mathematical morphology routines generic implementations	343
mln::morpho::opening::approx	Namespace of approximate mathematical morphology opening routines	343
mln::morpho::reconstruction	Namespace of morphological reconstruction routines	343
mln::morpho::reconstruction::by_dilation	Namespace of morphological reconstruction by dilation routines	344
mln::morpho::reconstruction::by_erosion	Namespace of morphological reconstruction by erosion routines	344
mln::morpho::tree	Namespace of morphological tree-related routines	344
mln::morpho::tree::filter	Namespace for attribute filtering	351
mln::morpho::watershed	Namespace of morphological watershed routines	353
mln::morpho::watershed::watershed	Namespace of morphological watershed routines implementations	354
mln::morpho::watershed::watershed::generic	Namespace of morphological watershed routines generic implementations	355
mln::norm	Namespace of norms	355
mln::norm::impl	Implementation namespace of norm namespace	357
mln::opt	Namespace of optional routines	357
mln::opt::impl	Implementation namespace of opt namespace	358
mln::pw	Namespace of "point-wise" expression tools	359
mln::registration	Namespace of "point-wise" expression tools	359
mln::select	Select namespace (FIXME doc)	361

mln::set	Namespace of image processing routines related to pixel sets	362
mln::subsampling	Namespace of "point-wise" expression tools	364
mln::tag	Namespace of image processing routines related to tags	365
mln::test	Namespace of image processing routines related to pixel tests	366
mln::test::impl	Implementation namespace of test namespace	367
mln::topo	Namespace of "point-wise" expression tools	367
mln::trace	Namespace of routines related to the trace mechanism	376
mln::trait	Namespace where traits are defined	376
mln::transform	Namespace of transforms	376
mln::util	Namespace of tools using for more complex algorithm	380
mln::util::impl	Implementation namespace of util namespace	386
mln::value	Namespace of materials related to pixel value types	386
mln::value::impl	Implementation namespace of value namespace	396
mln::win	Namespace of image processing routines related to win	396

Chapter 6

Hierarchical Index

6.1 Class Hierarchy

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

array< T >	399
mln::util::array< unsigned >	928
C_Function< E >	399
data< I >	399
data< labeled_image_base< I, labeled_image< I > > >	399
depth1st_piter< T >	400
dn_leaf_piter< T >	400
dn_node_piter< T >	401
dn_site_piter< T >	401
face_data< N, D >	401
faces_set_mixin< N, D >	401
mln::Generalized_Pixel< bkd_pixter1d< I > >	663
Pixel_Iterator< bkd_pixter1d< I > >	
pixel_iterator_base_< I, bkd_pixter1d< I > >	
backward_pixel_iterator_base_< I, bkd_pixter1d< I > >	
mln::bkd_pixter1d< I >	503
mln::Generalized_Pixel< bkd_pixter2d< I > >	663
Pixel_Iterator< bkd_pixter2d< I > >	
pixel_iterator_base_< I, bkd_pixter2d< I > >	
backward_pixel_iterator_base_< I, bkd_pixter2d< I > >	
mln::bkd_pixter2d< I >	505
mln::Generalized_Pixel< bkd_pixter3d< I > >	663
Pixel_Iterator< bkd_pixter3d< I > >	
pixel_iterator_base_< I, bkd_pixter3d< I > >	
backward_pixel_iterator_base_< I, bkd_pixter3d< I > >	
mln::bkd_pixter3d< I >	506
mln::Generalized_Pixel< dpoints_bkd_pixter< I > >	663
Pixel_Iterator< dpoints_bkd_pixter< I > >	
mln::dpoints_bkd_pixter< I >	592
mln::Generalized_Pixel< dpoints_fwd_pixter< I > >	663
Pixel_Iterator< dpoints_fwd_pixter< I > >	
mln::dpoints_fwd_pixter< I >	595
mln::Generalized_Pixel< fwd_pixter1d< I > >	663
Pixel_Iterator< fwd_pixter1d< I > >	
pixel_iterator_base_< I, fwd_pixter1d< I > >	
forward_pixel_iterator_base_< I, fwd_pixter1d< I > >	

mln::fwd_pixter1d< I >	658
mln::Generalized_Pixel< fwd_pixter2d< I > >	663
Pixel_iterator< fwd_pixter2d< I > >	
pixel_iterator_base< I, fwd_pixter2d< I > >	
forward_pixel_iterator_base< I, fwd_pixter2d< I > >	
mln::fwd_pixter2d< I >	659
mln::Generalized_Pixel< fwd_pixter3d< I > >	663
Pixel_iterator< fwd_pixter3d< I > >	
pixel_iterator_base< I, fwd_pixter3d< I > >	
forward_pixel_iterator_base< I, fwd_pixter3d< I > >	
mln::fwd_pixter3d< I >	661
mln::Generalized_Pixel< pixel< I > >	663
mln::pixel< I >	848
graph_elt_mixed_window< G, S, S2 >	402
graph_elt_window< G, S >	402
graph_elt_window_if< G, S, I >	402
graph_mixed_window_iter_dispatch< G, S, S2 >	403
graph_window_if_iter_dispatch< G, S >	403
graph_window_iter_dispatch< G, S >	403
mln::internal::image_base< algebra::vec< n, l::value >, l::domain_t, stack_image< n, I > >	
image_morpher< I, algebra::vec< n, l::value >, l::domain_t, stack_image< n, I > >	
image_value_morpher< I, algebra::vec< n, l::value >, stack_image< n, I > >	
mln::value::stack_image< n, I >	1015
mln::internal::image_base< const l::value, l::domain_t, E >	
image_morpher< const I, const l::value, l::domain_t, E >	
image_identity< const I, l::domain_t, E >	
mln::labeled_image_base< I, E >	724
mln::internal::image_base< const l::value, l::domain_t, labeled_image< I > >	
image_morpher< const I, const l::value, l::domain_t, labeled_image< I > >	
image_identity< const I, l::domain_t, labeled_image< I > >	
mln::labeled_image_base< I, labeled_image< I > >	724
mln::labeled_image< I >	720
mln::internal::image_base< F::result, l1::domain_t, thrubin_image< l1, l2, F > >	
image_morpher< l1, F::result, l1::domain_t, thrubin_image< l1, l2, F > >	
image_value_morpher< l1, F::result, thrubin_image< l1, l2, F > >	
mln::thrubin_image< l1, l2, F >	874
mln::internal::image_base< F::result, l::domain_t, fun_image< F, I > >	
image_morpher< I, F::result, l::domain_t, fun_image< F, I > >	
image_value_morpher< I, F::result, fun_image< F, I > >	
mln::fun_image< F, I >	652
mln::internal::image_base< F::result, S, E >	
image_primary< F::result, S, E >	
mln::internal::image_base< F::result, S, image< F, S > >	
image_primary< F::result, S, image< F, S > >	
image_base< F, S, image< F, S > >	
mln::pw::image< F, S >	860
mln::internal::image_base< fun::i2v::array< V >::result, p_edges< G, internal::efsite_selector< P, G >::site_ -	
function_t >, edge_image< P, V, G > >	
image_primary< fun::i2v::array< V >::result, p_edges< G, internal::efsite_selector< P, G >::site_function_ -	
t >, edge_image< P, V, G > >	
image_base< fun::i2v::array< V >, p_edges< G, internal::efsite_selector< P, G >::site_function_t >,	
edge_image< P, V, G > >	
mln::edge_image< P, V, G >	599
mln::internal::image_base< fun::i2v::array< V >::result, p_vertices< G, internal::vfsite_selector< P, G >::site_ -	
_function_t >, vertex_image< P, V, G > >	
image_primary< fun::i2v::array< V >::result, p_vertices< G, internal::vfsite_selector< P, G >::site_ -	
function_t >, vertex_image< P, V, G > >	

image_base< fun::i2v::array< V >, p_vertices< G, internal::vfsite_selector< P, G >::site_function_t >, vertex_image< P, V, G > >	
mln::vertex_image< P, V, G >	1019
mln::internal::image_base< I::value, box2d_h, hexa< I > >	
image_morpher< I, I::value, box2d_h, hexa< I > >	
image_domain_morpher< I, box2d_h, hexa< I > >	
mln::hexa< I >	694
mln::internal::image_base< I::value, box< I::site >, extended< I > >	
image_morpher< I, I::value, box< I::site >, extended< I > >	
image_domain_morpher< I, box< I::site >, extended< I > >	
mln::extended< I >	601
mln::internal::image_base< I::value, D, unproject_image< I, D, F > >	
image_morpher< I, I::value, D, unproject_image< I, D, F > >	
image_domain_morpher< I, D, unproject_image< I, D, F > >	
mln::unproject_image< I, D, F >	926
mln::internal::image_base< I::value, I::domain_t, decorated_image< I, D > >	
image_morpher< I, I::value, I::domain_t, decorated_image< I, D > >	
image_identity< I, I::domain_t, decorated_image< I, D > >	
mln::decorated_image< I, D >	547
mln::internal::image_base< I::value, I::domain_t, extension_fun< I, F > >	
image_morpher< I, I::value, I::domain_t, extension_fun< I, F > >	
image_identity< I, I::domain_t, extension_fun< I, F > >	
mln::extension_fun< I, F >	603
mln::internal::image_base< I::value, I::domain_t, extension_ima< I, J > >	
image_morpher< I, I::value, I::domain_t, extension_ima< I, J > >	
image_identity< I, I::domain_t, extension_ima< I, J > >	
mln::extension_ima< I, J >	605
mln::internal::image_base< I::value, I::domain_t, extension_val< I > >	
image_morpher< I, I::value, I::domain_t, extension_val< I > >	
image_identity< I, I::domain_t, extension_val< I > >	
mln::extension_val< I >	607
mln::internal::image_base< I::value, I::domain_t, interpolated< I, F > >	
image_morpher< I, I::value, I::domain_t, interpolated< I, F > >	
image_identity< I, I::domain_t, interpolated< I, F > >	
mln::interpolated< I, F >	715
mln::internal::image_base< I::value, I::domain_t, p2p_image< I, F > >	
image_morpher< I, I::value, I::domain_t, p2p_image< I, F > >	
image_domain_morpher< I, I::domain_t, p2p_image< I, F > >	
mln::p2p_image< I, F >	773
mln::internal::image_base< I::value, I::domain_t, plain< I > >	
image_morpher< I, I::value, I::domain_t, plain< I > >	
image_identity< I, I::domain_t, plain< I > >	
mln::plain< I >	849
mln::internal::image_base< I::value, I::domain_t, safe_image< I > >	
image_morpher< I, I::value, I::domain_t, safe_image< I > >	
image_identity< I, I::domain_t, safe_image< I > >	
mln::safe_image< I >	862
mln::internal::image_base< I::value, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > >	
image_morpher< I, I::value, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > >	
image_domain_morpher< I, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > >	
mln::sub_image_if< I, S >	872
mln::internal::image_base< I::value, p_transformed< I::domain_t, F >, transformed_image< I, F > >	
image_morpher< I, I::value, p_transformed< I::domain_t, F >, transformed_image< I, F > >	
image_domain_morpher< I, p_transformed< I::domain_t, F >, transformed_image< I, F > >	
mln::transformed_image< I, F >	924
mln::internal::image_base< I::value, S, E >	
image_morpher< I, I::value, S, E >	
mln::internal::image_base< I::value, S, sub_image< I, S > >	

```

    image_morpher< I, I::value, S, sub_image< I, S > >
    image_domain_morpher< I, S, sub_image< I, S > >
    mln::sub_image< I, S > . . . . . 871
mln::internal::image_base< I::value, S, tr_image< S, I, T > >
    image_morpher< I, I::value, S, tr_image< S, I, T > >
    image_identity< I, S, tr_image< S, I, T > >
    mln::tr_image< S, I, T > . . . . . 921
mln::internal::image_base< image2d< V >::value, box2d_h, hexa< image2d< V > > >
    image_morpher< image2d< V >, image2d< V >::value, box2d_h, hexa< image2d< V > > >
    image_domain_morpher< image2d< V >, box2d_h, hexa< image2d< V > > >
    mln::hexa< image2d< V > > . . . . . 694
    mln::image2d_h< V > . . . . . 707
mln::internal::image_base< mln::trait::ch_value< I, F::result >::ret::value, I::domain_t, lazy_image< I, F, B >
>
    image_morpher< mln::trait::ch_value< I, F::result >::ret, mln::trait::ch_value< I, F::result >::ret::value, I::
domain_t, lazy_image< I, F, B > >
    image_identity< mln::trait::ch_value< I, F::result >::ret, I::domain_t, lazy_image< I, F, B > >
    mln::lazy_image< I, F, B > . . . . . 727
mln::internal::image_base< T, box1d, image1d< T > >
    image_primary< T, box1d, image1d< T > >
    mln::image1d< T > . . . . . 700
mln::internal::image_base< T, box3d, image3d< T > >
    image_primary< T, box3d, image3d< T > >
    mln::image3d< T > . . . . . 710
mln::internal::image_base< T, I::domain_t, E >
    image_morpher< I, T, I::domain_t, E >
mln::internal::image_base< T, I::domain_t, violent_cast_image< T, I > >
    image_morpher< I, T, I::domain_t, violent_cast_image< T, I > >
    image_value_morpher< I, T, violent_cast_image< T, I > >
    mln::violent_cast_image< T, I > . . . . . 1021
mln::internal::image_base< T, mln::box2d, image2d< T > >
    image_primary< T, mln::box2d, image2d< T > >
    mln::image2d< T > . . . . . 703
mln::internal::image_base< T, S, flat_image< T, S > >
    image_primary< T, S, flat_image< T, S > >
    mln::flat_image< T, S > . . . . . 609
mln::internal::image_base< unsigned, box3d, image3d< unsigned > >
    image_primary< unsigned, box3d, image3d< unsigned > >
    mln::image3d< unsigned > . . . . . 710
mln::internal::image_base< V, p_complex< D, G >, complex_image< D, G, V > >
    image_primary< V, p_complex< D, G >, complex_image< D, G, V > >
    mln::complex_image< D, G, V > . . . . . 536
mln::internal::check::image_fastest_< complex_image< D, G, V >, mln::metal::equal< mln_trait_image_
speed(complex_image< D, G, V >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< decorated_image< I, D >, mln::metal::equal< mln_trait_image_
speed(decorated_image< I, D >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< E, mln::metal::equal< mln_trait_image_speed(E), trait::image::speed::
fastest >::eval >
mln::internal::check::image_fastest_< edge_image< P, V, G >, mln::metal::equal< mln_trait_image_
speed(edge_image< P, V, G >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< extended< I >, mln::metal::equal< mln_trait_image_speed(extended<
I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< extension_fun< I, F >, mln::metal::equal< mln_trait_image_
speed(extension_fun< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< extension_ima< I, J >, mln::metal::equal< mln_trait_image_
speed(extension_ima< I, J >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< extension_val< I >, mln::metal::equal< mln_trait_image_speed(extension_
_val< I >), trait::image::speed::fastest >::eval >

```

```

mln::internal::check::image_fastest_< flat_image< T, S >, mln::metal::equal< mln_trait_image_speed(flat_
image< T, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< fun_image< F, I >, mln::metal::equal< mln_trait_image_speed(fun_
image< F, I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< hexa< I >, mln::metal::equal< mln_trait_image_speed(hexa< I >), trait-
::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< hexa< image2d< V > >, mln::metal::equal< mln_trait_image_
speed(hexa< image2d< V > >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image1d< T >, mln::metal::equal< mln_trait_image_speed(image1d<
T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image2d< T >, mln::metal::equal< mln_trait_image_speed(image2d<
T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image3d< T >, mln::metal::equal< mln_trait_image_speed(image3d<
T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image3d< unsigned >, mln::metal::equal< mln_trait_image_
speed(image3d< unsigned >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< image< F, S >, mln::metal::equal< mln_trait_image_speed(image< F,
S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< interpolated< I, F >, mln::metal::equal< mln_trait_image_speed(interpolated<
I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< labeled_image< I >, mln::metal::equal< mln_trait_image_speed(labeled-
_image< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< lazy_image< I, F, B >, mln::metal::equal< mln_trait_image_speed(lazy-
_image< I, F, B >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< p2p_image< I, F >, mln::metal::equal< mln_trait_image_speed(p2p_
image< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< plain< I >, mln::metal::equal< mln_trait_image_speed(plain< I >), trait-
::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< safe_image< I >, mln::metal::equal< mln_trait_image_speed(safe_
image< I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< stack_image< n, I >, mln::metal::equal< mln_trait_image_speed(stack-
_image< n, I >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< sub_image< I, S >, mln::metal::equal< mln_trait_image_speed(sub_
image< I, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< sub_image_if< I, S >, mln::metal::equal< mln_trait_image_speed(sub-
_image_if< I, S >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< thrubin_image< I1, I2, F >, mln::metal::equal< mln_trait_image_
speed(thrubin_image< I1, I2, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< tr_image< S, I, T >, mln::metal::equal< mln_trait_image_speed(tr_
image< S, I, T >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< transformed_image< I, F >, mln::metal::equal< mln_trait_image_
speed(transformed_image< I, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< unproject_image< I, D, F >, mln::metal::equal< mln_trait_image_
speed(unproject_image< I, D, F >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< vertex_image< P, V, G >, mln::metal::equal< mln_trait_image_
speed(vertex_image< P, V, G >), trait::image::speed::fastest >::eval >
mln::internal::check::image_fastest_< violent_cast_image< T, I >, mln::metal::equal< mln_trait_image_
speed(violent_cast_image< T, I >), trait::image::speed::fastest >::eval >
mln::internal::impl_selector< W::center_t, W::psite, graph_window_piter< S, W, I > >
    mln::graph_window_piter< S, W, I > . . . . . 691
int_s< n > . . . . . 403
mln::value::Integer< graylevel< n > > . . . . . 1002
    mln::value::graylevel< n > . . . . . 990
mln::value::Integer< int_s< n > > . . . . . 1002
    mln::value::int_s< n > . . . . . 995
mln::value::Integer< int_u< n > > . . . . . 1002
    mln::value::int_u< n > . . . . . 997
mln::value::Integer< int_u_sat< n > > . . . . . 1002

```

mln::value::int_u_sat< n >	999
mln::value::Integer< object_id< Tag, V > >	1002
mln::util::object_id< Tag, V >	958
mln::internal::is_masked_impl_selector< S, W::mask_t::domain_t, graph_window_if_piter< S, W, I > > mln::graph_window_if_piter< S, W, I >	690
line_graph< G >	404
mln::algebra::h_mat< d, T >	500
mln::algebra::h_vec< d, C >	502
mln::canvas::chamfer< F >	535
mln::category< R(*) (A) >	535
mln::Delta_Point_Site< void >	549
mln::doc::Accumulator< E >	550
mln::doc::Generalized_Pixel< E >	562
mln::doc::Pixel_Iterator< E >	572
mln::doc::Object< E >	572
mln::doc::Dpoint< E >	553
mln::doc::Image< E >	564
mln::doc::Fastest_Image< E >	555
mln::doc::Iterator< E >	569
mln::doc::Pixel_Iterator< E >	572
mln::doc::Site_Iterator< E >	577
mln::doc::Value_Iterator< E >	580
mln::doc::Neighborhood< E >	570
mln::doc::Site_Set< E >	579
mln::doc::Box< E >	551
mln::doc::Value_Set< E >	582
mln::doc::Weighted_Window< E >	584
mln::doc::Window< E >	586
mln::doc::Point_Site< E >	575
mln::Edge< E >	599
mln::fun::from_accu< A >	611
mln::fun::internal::ch_function_value_impl< F, V > mln::fun::v2v::ch_function_value< F, V >	617
mln::fun::x2p::closest_point< P >	642
mln::fun::x2x::composed< T2, T1 >	644
mln::Function< void >	654
mln::Gdpoint< void >	662
mln::Generalized_Pixel< E >	663
mln::Pixel_Iterator mln::internal::pixel_iterator_base_	
mln::geom::complex_geometry< D, P >	664
mln::graph::attribute::card_t	669
mln::graph::attribute::representative_t	670
mln::histo::array< T >	698
mln::internal::image_base< T, S, E > mln::internal::neighborhood_base< W, E > mln::internal::neighb_base	
mln::neighb< graph_elt_mixed_window< G, S, S2 > >	770
mln::graph_elt_mixed_neighborhood< G, S, S2 >	671
mln::neighb< graph_elt_window< G, S > >	770
mln::graph_elt_neighborhood< G, S >	676
mln::neighb< graph_elt_window_if< G, S, I > >	770
mln::graph_elt_neighborhood_if< G, S, I >	677
mln::internal::pixel_impl< I, E > mln::internal::pixel_iterator_base_	
mln::io::dicom::dicom_header	717

mln::io::dump::dump_header	717
mln::io::fld::fld_header	717
mln::io::raw::raw_header	717
mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 >	759
mln::metal::bool_< false >	
mln::metal::equal< T1::coord, T2::coord >	760
mln::metal::equal< T1::point, T2::point >	760
mln::metal::equal< T1, T2 >	760
mln::metal::converts_to< T, U >	760
mln::metal::goes_to< T, U >	760
mln::metal::is< T, U >	761
mln::metal::is_a< T, M >	761
mln::metal::is_not< T, U >	761
mln::metal::is_not_a< T, M >	762
mln::Neighborhood< void >	773
mln::Object< E >	773
mln::Browsing< E >	520
mln::Delta_Point_Site< E >	549
mln::Dpoint< E >	591
mln::Function< E >	654
mln::Function_n2v< E >	655
mln::Function_v2v< E >	656
mln::fun::x2v::bilinear< I >	643
mln::fun::x2v::trilinear< I >	644
mln::fun::x2x::linear< I >	645
mln::Function_v2b< E >	656
mln::Function_vv2b< E >	657
mln::Function_vv2v< E >	657
mln::Gdpoint< E >	662
mln::Graph< E >	669
mln::Image< E >	698
mln::Iterator< E >	718
mln::Pixel_iterator	
mln::topo::internal::complex_iterator_base	
mln::topo::internal::complex_relative_iterator_base	
mln::Literal< E >	729
mln::Mesh< E >	753
mln::Regular_Grid< E >	862
mln::Meta_Accumulator< E >	754
mln::Meta_Function< E >	756
mln::Meta_Function_v2v< E >	758
mln::Meta_Function_vv2v< E >	759
mln::Neighborhood< E >	772
mln::Proxy< E >	859
mln::Accumulator< E >	498
mln::accu::internal::base	
couple< accu::shape::bbox< P >, accu::math::count< P >, float, rectangularity< P > >	
mln::accu::site_set::rectangularity< P >	470
mln::accu::pair< min< V >, max< V > >	462
mln::accu::stat::min_max< V >	486
mln::Site_Proxy< E >	866
mln::Pseudo_Site< E >	859
mln::Site_iterator< E >	865
mln::internal::site_iterator_base	
mln::internal::site_set_iterator_base	
mln::p_transformed_piter< Pi, S, F >	838
mln::Site< E >	863

mln::Gpoint< E >	665
mln::Site_Set< E >	867
mln::Box< E >	514
mln::Value< E >	985
mln::Weighted_Window< E >	1026
mln::Window< E >	1038
mln::graph_window_base< P, E >	688
mln::internal::window_base	
mln::Proxy< void >	859
mln::Pseudo_Site< void >	860
mln::registration::closest_point_basic< P >	861
mln::registration::closest_point_with_map< P >	862
mln::select::p_of< P >	863
mln::Site< void >	864
mln::Site_Proxy< void >	867
mln::Site_Set< void >	871
mln::thru_image< I, F >	873
mln::topo::complex< D >	899
mln::topo::face< D >	902
mln::topo::algebraic_face< D >	888
mln::topo::is_simple_2d_t< N >	909
mln::topo::n_face< N, D >	912
mln::topo::algebraic_n_face< N, D >	892
mln::topo::n_faces_set< N, D >	917
mln::topo::skeleton::is_simple_point< N >	918
mln::util::adjacency_matrix< V >	927
mln::util::array< T >	928
mln::util::branch< T >	933
mln::util::branch_iter< T >	934
mln::util::branch_iter_ind< T >	936
mln::util::greater_point< I >	951
mln::util::greater_psite< I >	951
mln::util::head< T, R >	952
mln::util::ilcell< T >	953
mln::util::internal::edge_impl< G >	
mln::util::edge< G >	939
mln::util::internal::vertex_impl< G >	
mln::util::vertex< G >	981
mln::util::node< T, R >	958
mln::util::ord< T >	960
mln::util::pix< I >	962
mln::util::tracked_ptr< T >	973
mln::util::tree< T >	975
mln::util::tree_node< T >	977
mln::value::float01	987
mln::value::Integer< E >	1002
mln::value::Integer< void >	1002
mln::value::internal::value_like< V, C, N, E >	
mln::value::float01_f	989
mln::value::graylevel< n >	990
mln::value::graylevel_f	993
mln::value::int_s< n >	995
mln::value::int_u< n >	997
mln::value::int_u_sat< n >	999
mln::value::label< n >	1002
mln::value::qt::rgb32	1009
mln::value::rgb< n >	1011

mln::value::set< T >	1012
mln::value::sign	1013
mln::value::super_value< sign >	1017
mln::value::value_array< T, V >	1018
mln::Vertex< E >	1019
mln::internal::neighborhood_base< W, neighb< W > > neighb_base< W, neighb< W > > mln::neighb< W >	770
mln::Object< abs >	773
mln::Meta_Function< abs >	756
mln::Meta_Function_v2v< abs >	758
mln::Object< abs< V > >	773
mln::Function< abs< V > >	654
mln::Function_v2v< abs< V > >	656
mln::Object< accu_result >	773
mln::Meta_Function< accu_result >	756
mln::Meta_Function_v2v< accu_result >	758
mln::Object< adj_higher_dim_connected_n_face_bkd_iter< D > >	773
mln::Iterator< adj_higher_dim_connected_n_face_bkd_iter< D > >	718
complex_iterator_base< algebraic_face< D >, adj_higher_dim_connected_n_face_bkd_iter< D > > complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_ - connected_n_face_bkd_iter< D > > backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_ - dim_connected_n_face_bkd_iter< D > > mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D >	875
mln::Object< adj_higher_dim_connected_n_face_fwd_iter< D > >	773
mln::Iterator< adj_higher_dim_connected_n_face_fwd_iter< D > >	718
complex_iterator_base< algebraic_face< D >, adj_higher_dim_connected_n_face_fwd_iter< D > > complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_dim_ - connected_n_face_fwd_iter< D > > forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_ - dim_connected_n_face_fwd_iter< D > > mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D >	876
mln::Object< adj_higher_face_bkd_iter< D > >	773
mln::Iterator< adj_higher_face_bkd_iter< D > >	718
complex_iterator_base< algebraic_face< D >, adj_higher_face_bkd_iter< D > > complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_bkd_ - iter< D > > backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_ - face_bkd_iter< D > > mln::topo::adj_higher_face_bkd_iter< D >	877
mln::Object< adj_higher_face_fwd_iter< D > >	773
mln::Iterator< adj_higher_face_fwd_iter< D > >	718
complex_iterator_base< algebraic_face< D >, adj_higher_face_fwd_iter< D > > complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_face_fwd_ - iter< D > > forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_higher_ - face_fwd_iter< D > > mln::topo::adj_higher_face_fwd_iter< D >	878
mln::Object< adj_lower_dim_connected_n_face_bkd_iter< D > >	773
mln::Iterator< adj_lower_dim_connected_n_face_bkd_iter< D > >	718
complex_iterator_base< algebraic_face< D >, adj_lower_dim_connected_n_face_bkd_iter< D > > complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_ - connected_n_face_bkd_iter< D > > backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_ - dim_connected_n_face_bkd_iter< D > >	

mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D >	879
mln::Object< adj_lower_dim_connected_n_face_fwd_iter< D > >	773
mln::iterator< adj_lower_dim_connected_n_face_fwd_iter< D > >	718
complex_iterator_base< algebraic_face< D >, adj_lower_dim_connected_n_face_fwd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_dim_	
connected_n_face_fwd_iter< D > >	
forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_	
dim_connected_n_face_fwd_iter< D > >	
mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D >	880
mln::Object< adj_lower_face_bkd_iter< D > >	773
mln::iterator< adj_lower_face_bkd_iter< D > >	718
complex_iterator_base< algebraic_face< D >, adj_lower_face_bkd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_face_bkd_iter<	
D > >	
backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_	
face_bkd_iter< D > >	
mln::topo::adj_lower_face_bkd_iter< D >	881
mln::Object< adj_lower_face_fwd_iter< D > >	773
mln::iterator< adj_lower_face_fwd_iter< D > >	718
complex_iterator_base< algebraic_face< D >, adj_lower_face_fwd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_face_fwd_iter<	
D > >	
forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_lower_	
face_fwd_iter< D > >	
mln::topo::adj_lower_face_fwd_iter< D >	882
mln::Object< adj_lower_higher_face_bkd_iter< D > >	773
mln::iterator< adj_lower_higher_face_bkd_iter< D > >	718
complex_relative_iterator_sequence< adj_higher_face_bkd_iter< D >, adj_lower_face_bkd_iter< D > ,	
adj_lower_higher_face_bkd_iter< D > >	
mln::topo::adj_lower_higher_face_bkd_iter< D >	883
mln::Object< adj_lower_higher_face_fwd_iter< D > >	773
mln::iterator< adj_lower_higher_face_fwd_iter< D > >	718
complex_relative_iterator_sequence< adj_lower_face_fwd_iter< D >, adj_higher_face_fwd_iter< D > ,	
adj_lower_higher_face_fwd_iter< D > >	
mln::topo::adj_lower_higher_face_fwd_iter< D >	884
mln::Object< adj_m_face_bkd_iter< D > >	773
mln::iterator< adj_m_face_bkd_iter< D > >	718
complex_iterator_base< algebraic_face< D >, adj_m_face_bkd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_bkd_iter< D	
> >	
backward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_	
face_bkd_iter< D > >	
mln::topo::adj_m_face_bkd_iter< D >	885
mln::Object< adj_m_face_fwd_iter< D > >	773
mln::iterator< adj_m_face_fwd_iter< D > >	718
complex_iterator_base< algebraic_face< D >, adj_m_face_fwd_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_fwd_iter< D	
> >	
forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, adj_m_face_	
fwd_iter< D > >	
mln::topo::adj_m_face_fwd_iter< D >	887
mln::Object< all_to< T > >	773
mln::Function< all_to< T > >	654
mln::Function_v2v< all_to< T > >	656
mln::Object< antilogy >	773

mln::Function< antilogy >	654
mln::Function_v2v< antilogy >	656
mln::Function_v2b< antilogy >	656
mln::fun::p2b::antilogy	612
mln::Object< array1d< T, Size > >	773
mln::Object< array2d< T, r, c > >	773
mln::Object< array3d< T, s, r, c > >	773
mln::Object< array_bkd_iter< T > >	773
mln::Proxy< array_bkd_iter< T > >	859
mln::Object< array_fwd_iter< T > >	773
mln::Proxy< array_fwd_iter< T > >	859
mln::Object< asc_propagation >	773
mln::Object< backdiag2d >	773
mln::Window< backdiag2d >	1038
window_base< dpoint2d, backdiag2d >	
classical_window_base< dpoint2d, backdiag2d >	
mln::win::backdiag2d	1027
mln::Object< backdiagonal2d_t >	773
mln::Browsing< backdiagonal2d_t >	520
mln::canvas::browsing::backdiagonal2d_t	521
mln::Object< ball< G, C > >	773
mln::Window< ball< G, C > >	1038
window_base< dpoint< G, C >, ball< G, C > >	
classical_window_base< dpoint< G, C >, ball< G, C > >	
mln::win::ball< G, C >	1028
mln::Object< bbox >	773
mln::Meta_Accumulator< bbox >	754
mln::accu::meta::shape::bbox	445
mln::Object< bbox< P > >	773
mln::Proxy< bbox< P > >	859
mln::Accumulator< bbox< P > >	498
base< const box< P > &, bbox< P > >	
mln::accu::shape::bbox< P >	465
mln::Object< big_chess< B > >	773
mln::Function< big_chess< B > >	654
mln::Function_v2v< big_chess< B > >	656
mln::Function_v2b< big_chess< B > >	656
mln::Object< bin_off_loader >	773
mln::Object< bin_off_saver >	773
mln::Object< binary< Fun, T1, T2 > >	773
mln::Function< binary< Fun, T1, T2 > >	654
mln::Function_v2v< binary< Fun, T1, T2 > >	656
mln::Object< bkd_pixter1d< I > >	773
mln::Iterator< bkd_pixter1d< I > >	718
Pixel_Iterator< bkd_pixter1d< I > >	
mln::Object< bkd_pixter2d< I > >	773
mln::Iterator< bkd_pixter2d< I > >	718
Pixel_Iterator< bkd_pixter2d< I > >	
mln::Object< bkd_pixter3d< I > >	773
mln::Iterator< bkd_pixter3d< I > >	718
Pixel_Iterator< bkd_pixter3d< I > >	
mln::Object< black_t >	773
mln::Literal< black_t >	729

mln::literal::black_t	731
mln::Object< blue >	773
mln::Meta_Function< blue >	756
mln::Meta_Function_v2v< blue >	758
mln::Object< blue_t >	773
mln::Literal< blue_t >	729
mln::literal::blue_t	731
mln::Object< box< P > >	773
mln::Site_Set< box< P > >	867
mln::Box< box< P > >	514
mln::box< P >	507
mln::Object< box_runend_piter< P > >	773
mln::Proxy< box_runend_piter< P > >	859
mln::Site_Proxy< box_runend_piter< P > >	866
mln::Site_Iterator< box_runend_piter< P > >	865
site_iterator_base< box< P >, box_runend_piter< P > >	
site_set_iterator_base< box< P >, box_runend_piter< P > >	
mln::box_runend_piter< P >	518
mln::Object< box_runstart_piter< P > >	773
mln::Proxy< box_runstart_piter< P > >	859
mln::Site_Proxy< box_runstart_piter< P > >	866
mln::Site_Iterator< box_runstart_piter< P > >	865
site_iterator_base< box< P >, box_runstart_piter< P > >	
site_set_iterator_base< box< P >, box_runstart_piter< P > >	
mln::box_runstart_piter< P >	519
mln::Object< breadth_first_search_t >	773
mln::Browsing< breadth_first_search_t >	520
graph_first_search_t< breadth_first_search_t, std::queue >	
mln::canvas::browsing::breadth_first_search_t	523
mln::Object< brown_t >	773
mln::Literal< brown_t >	729
mln::literal::brown_t	732
mln::Object< card< I > >	773
mln::Proxy< card< I > >	859
mln::Accumulator< card< I > >	498
base< unsigned, card< I > >	
mln::morpho::attribute::card< I >	762
mln::Object< cast< V > >	773
mln::Function< cast< V > >	654
mln::Function_v2v< cast< V > >	656
mln::Object< center >	773
mln::Meta_Accumulator< center >	754
mln::accu::meta::center	426
mln::Object< center< P, V > >	773
mln::Proxy< center< P, V > >	859
mln::Accumulator< center< P, V > >	498
base< V, center< P, V > >	
mln::accu::center< P, V >	404
mln::Object< center_only_iter< D > >	773
mln::Iterator< center_only_iter< D > >	718
complex_iterator_base< algebraic_face< D >, center_only_iter< D > >	
complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, center_only_iter< D >	
>	

forward_complex_relative_iterator_base< topo::face< D >, algebraic_face< D >, center_only_	
iter< D > >	
mln::topo::center_only_iter< D >	896
mln::Object< centered_bkd_iter_adapter< D, I > >	773
mln::Iterator< centered_bkd_iter_adapter< D, I > >	718
complex_relative_iterator_sequence< I, center_only_iter< D >, centered_bkd_iter_adapter< D, I > >	
mln::topo::centered_bkd_iter_adapter< D, I >	897
mln::Object< centered_fwd_iter_adapter< D, I > >	773
mln::Iterator< centered_fwd_iter_adapter< D, I > >	718
complex_relative_iterator_sequence< center_only_iter< D >, I, centered_fwd_iter_adapter< D, I > >	
mln::topo::centered_fwd_iter_adapter< D, I >	898
mln::Object< ch_function_value< F, V > >	773
mln::Function< ch_function_value< F, V > >	654
mln::Function_v2v< ch_function_value< F, V > >	656
mln::fun::v2v::ch_function_value< F, V >	617
mln::Object< ch_piter_image< I, Fwd > >	773
mln::Image< ch_piter_image< I, Fwd > >	698
mln::Object< chess >	773
mln::Function< chess >	654
mln::Function_v2v< chess >	656
mln::Function_v2b< chess >	656
mln::Object< col >	773
mln::Meta_Function< col >	756
mln::Meta_Function_v2v< col >	758
mln::Object< colorize >	773
mln::Function< colorize >	654
mln::Function_v2v< colorize >	656
mln::Object< comp >	773
mln::Meta_Function< comp >	756
mln::Meta_Function_v2v< comp >	758
mln::Object< comp_count >	773
mln::Meta_Function< comp_count >	756
mln::Meta_Function_v2v< comp_count >	758
mln::Object< complex_image< D, G, V > >	773
mln::Image< complex_image< D, G, V > >	698
mln::Object< complex_neighborhood_bkd_piter< I, G, N > >	773
mln::Proxy< complex_neighborhood_bkd_piter< I, G, N > >	859
mln::Site_Proxy< complex_neighborhood_bkd_piter< I, G, N > >	866
mln::Site_Iterator< complex_neighborhood_bkd_piter< I, G, N > >	865
site_iterator_base< N, complex_neighborhood_bkd_piter< I, G, N > >	
site_relative_iterator_base< N, complex_neighborhood_bkd_piter< I, G, N > >	
mln::complex_neighborhood_bkd_piter< I, G, N >	538
mln::Object< complex_neighborhood_fwd_piter< I, G, N > >	773
mln::Proxy< complex_neighborhood_fwd_piter< I, G, N > >	859
mln::Site_Proxy< complex_neighborhood_fwd_piter< I, G, N > >	866
mln::Site_Iterator< complex_neighborhood_fwd_piter< I, G, N > >	865
site_iterator_base< N, complex_neighborhood_fwd_piter< I, G, N > >	
site_relative_iterator_base< N, complex_neighborhood_fwd_piter< I, G, N > >	
mln::complex_neighborhood_fwd_piter< I, G, N >	540
mln::Object< complex_psite< D, G > >	773
mln::Proxy< complex_psite< D, G > >	859
mln::Site_Proxy< complex_psite< D, G > >	866
mln::Pseudo_Site< complex_psite< D, G > >	859

pseudo_site_base< const G::site &, complex_psite< D, G > >	
mln::complex_psite< D, G >	541
mln::Object< complex_window_bkd_piter< I, G, W > >	773
mln::Proxy< complex_window_bkd_piter< I, G, W > >	859
mln::Site_Proxy< complex_window_bkd_piter< I, G, W > >	866
mln::Site_Iterator< complex_window_bkd_piter< I, G, W > >	865
site_iterator_base< W, complex_window_bkd_piter< I, G, W > >	
site_relative_iterator_base< W, complex_window_bkd_piter< I, G, W > >	
mln::complex_window_bkd_piter< I, G, W >	544
mln::Object< complex_window_fwd_piter< I, G, W > >	773
mln::Proxy< complex_window_fwd_piter< I, G, W > >	859
mln::Site_Proxy< complex_window_fwd_piter< I, G, W > >	866
mln::Site_Iterator< complex_window_fwd_piter< I, G, W > >	865
site_iterator_base< W, complex_window_fwd_piter< I, G, W > >	
site_relative_iterator_base< W, complex_window_fwd_piter< I, G, W > >	
mln::complex_window_fwd_piter< I, G, W >	545
mln::Object< component< T, i > >	773
mln::Function< component< T, i > >	654
mln::Function_v2v< component< T, i > >	656
mln::fun::v2v::component< T, i >	618
mln::Object< compose >	773
mln::Meta_Function< compose >	756
mln::Meta_Function_vv2v< compose >	759
mln::Object< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v, G > >	773
mln::Meta_Function< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v, G > >	756
mln::Meta_Function_v2v< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_v2v, G	
> >	758
mln::Object< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_vv2v, G > >	773
mln::Meta_Function< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_vv2v, G > >	756
mln::Meta_Function_vv2v< composition< mln::Meta_Function_v2v, F, mln::Meta_Function_vv2v,	
G > >	759
mln::Object< concrete >	773
mln::Object< convert< V > >	773
mln::Function< convert< V > >	654
mln::Function_v2v< convert< V > >	656
mln::Object< convolve< T1, T2, R > >	773
mln::Proxy< convolve< T1, T2, R > >	859
mln::Accumulator< convolve< T1, T2, R > >	498
base< R, convolve< T1, T2, R > >	
mln::accu::convolve< T1, T2, R >	405
mln::Object< cos >	773
mln::Meta_Function< cos >	756
mln::Meta_Function_v2v< cos >	758
mln::Object< cos< V > >	773
mln::Function< cos< V > >	654
mln::Function_v2v< cos< V > >	656
mln::fun::v2w2v::cos< V >	624
mln::Object< count >	773
mln::Meta_Accumulator< count >	754
mln::accu::meta::math::count	436
mln::Object< count< T > >	773
mln::Proxy< count< T > >	859
mln::Accumulator< count< T > >	498

base< unsigned, count< T > >	
mln::accu::math::count< T >	420
mln::Object< count_adjacent_vertices >	773
mln::Meta_Accumulator< count_adjacent_vertices >	754
mln::accu::meta::count_adjacent_vertices	426
mln::Object< count_adjacent_vertices< F, S > >	773
mln::Proxy< count_adjacent_vertices< F, S > >	859
mln::Accumulator< count_adjacent_vertices< F, S > >	498
base< unsigned, count_adjacent_vertices< F, S > >	
mln::accu::count_adjacent_vertices< F, S >	407
mln::Object< count_adjacent_vertices< I > >	773
mln::Proxy< count_adjacent_vertices< I > >	859
mln::Accumulator< count_adjacent_vertices< I > >	498
base< unsigned, count_adjacent_vertices< I > >	
mln::morpho::attribute::count_adjacent_vertices< I >	763
mln::Object< count_labels >	773
mln::Meta_Accumulator< count_labels >	754
mln::accu::meta::count_labels	427
mln::Object< count_labels< L > >	773
mln::Proxy< count_labels< L > >	859
mln::Accumulator< count_labels< L > >	498
base< unsigned, count_labels< L > >	
mln::accu::count_labels< L >	408
mln::Object< count_value >	773
mln::Meta_Accumulator< count_value >	754
mln::accu::meta::count_value	428
mln::Object< count_value< V > >	773
mln::Proxy< count_value< V > >	859
mln::Accumulator< count_value< V > >	498
base< unsigned, count_value< V > >	
mln::accu::count_value< V >	409
mln::Object< couple< T, U > >	773
mln::util::couple< T, U >	937
mln::Object< cube >	773
mln::Mesh< cube >	753
mln::Regular_Grid< cube >	862
mln::Object< cube3d >	773
mln::Window< cube3d >	1038
window_base< dpoint3d, cube3d >	
classical_window_base< dpoint3d, cube3d >	
mln::win::cube3d	1029
mln::Object< cuboid3d >	773
mln::Window< cuboid3d >	1038
window_base< dpoint3d, cuboid3d >	
classical_window_base< dpoint3d, cuboid3d >	
mln::win::cuboid3d	1030
mln::Object< cyan_t >	773
mln::Literal< cyan_t >	729
mln::literal::cyan_t	733
mln::Object< d_t >	773
mln::Function< d_t >	654
mln::Function_vv2v< d_t >	657
mln::Object< dark_gray_t >	773

mln::Literal< dark_gray_t >	729
mln::Object< dashed_line_f< I, dim > >	773
mln::Function< dashed_line_f< I, dim > >	654
mln::Function_v2v< dashed_line_f< I, dim > >	656
mln::Function_v2b< dashed_line_f< I, dim > >	656
mln::Object< decorated_image< I, D > >	773
mln::Image< decorated_image< I, D > >	698
mln::Object< depth1st_piter< T > >	773
mln::Proxy< depth1st_piter< T > >	859
mln::Site_Proxy< depth1st_piter< T > >	866
mln::Site_Iterator< depth1st_piter< T > >	865
mln::Object< depth_first_search_t >	773
mln::Browsing< depth_first_search_t >	520
graph_first_search_t< depth_first_search_t, std::stack > mln::canvas::browsing::depth_first_search_t	523
mln::Object< desc_propagation >	773
mln::Object< deviation >	773
mln::Meta_Accumulator< deviation >	754
mln::accu::stat::meta::deviation	482
mln::Object< deviation< T, S, M > >	773
mln::Proxy< deviation< T, S, M > >	859
mln::Accumulator< deviation< T, S, M > >	498
base< M, deviation< T, S, M > > mln::accu::stat::deviation< T, S, M >	471
mln::Object< diag2d >	773
mln::Window< diag2d >	1038
window_base< dpoint2d, diag2d > classical_window_base< dpoint2d, diag2d > mln::win::diag2d	1032
mln::Object< diagonal2d_t >	773
mln::Browsing< diagonal2d_t >	520
mln::canvas::browsing::diagonal2d_t	523
mln::Object< diff_abs< V > >	773
mln::Function< diff_abs< V > >	654
mln::Function_vv2v< diff_abs< V > >	657
mln::fun::vv2v::diff_abs< V >	634
mln::Object< dir_struct_elt_incr_update_t >	773
mln::Browsing< dir_struct_elt_incr_update_t >	520
mln::canvas::browsing::dir_struct_elt_incr_update_t	525
mln::Object< directional_t >	773
mln::Browsing< directional_t >	520
mln::canvas::browsing::directional_t	526
mln::Object< dist >	773
mln::Function< dist >	654
mln::Function_vv2v< dist >	657
mln::Object< dist_t >	773
mln::Function< dist_t >	654
mln::Function_vv2v< dist_t >	657
mln::Object< dn_leaf_piter< T > >	773
mln::Proxy< dn_leaf_piter< T > >	859
mln::Site_Proxy< dn_leaf_piter< T > >	866
mln::Site_Iterator< dn_leaf_piter< T > >	865
mln::Object< dn_node_piter< T > >	773

mln::Proxy< dn_node_piter< T > >	859
mln::Site_Proxy< dn_node_piter< T > >	866
mln::Site_Iterator< dn_node_piter< T > >	865
mln::Object< dn_site_piter< T > >	773
mln::Proxy< dn_site_piter< T > >	859
mln::Site_Proxy< dn_site_piter< T > >	866
mln::Site_Iterator< dn_site_piter< T > >	865
mln::Object< dpoint< G, C > >	773
mln::Gdpoint< dpoint< G, C > >	662
mln::dpoint< G, C >	588
mln::Object< dpoints_bkd_pixter< I > >	773
mln::Iterator< dpoints_bkd_pixter< I > >	718
Pixel_Iterator< dpoints_bkd_pixter< I > >	
mln::Object< dpoints_fwd_pixter< I > >	773
mln::Iterator< dpoints_fwd_pixter< I > >	718
Pixel_Iterator< dpoints_fwd_pixter< I > >	
mln::Object< dpsites_bkd_piter< V > >	773
mln::Proxy< dpsites_bkd_piter< V > >	859
mln::Site_Proxy< dpsites_bkd_piter< V > >	866
mln::Site_Iterator< dpsites_bkd_piter< V > >	865
site_iterator_base< V, dpsites_bkd_piter< V > >	
site_relative_iterator_base< V, dpsites_bkd_piter< V > >	
mln::dpsites_bkd_piter< V >	597
mln::Object< dpsites_fwd_piter< V > >	773
mln::Proxy< dpsites_fwd_piter< V > >	859
mln::Site_Proxy< dpsites_fwd_piter< V > >	866
mln::Site_Iterator< dpsites_fwd_piter< V > >	865
site_iterator_base< V, dpsites_fwd_piter< V > >	
site_relative_iterator_base< V, dpsites_fwd_piter< V > >	
mln::dpsites_fwd_piter< V >	598
mln::Object< eat >	773
mln::util::eat	939
mln::Object< edge_bkd_iterator< G > >	773
mln::Proxy< edge_bkd_iterator< G > >	859
mln::Object< edge_fwd_iterator< G > >	773
mln::Proxy< edge_fwd_iterator< G > >	859
mln::Object< edge_image< P, V, G > >	773
mln::Image< edge_image< P, V, G > >	698
mln::Object< edge_nbh_edge_bkd_iterator< G > >	773
mln::Proxy< edge_nbh_edge_bkd_iterator< G > >	859
mln::Object< edge_nbh_edge_fwd_iterator< G > >	773
mln::Proxy< edge_nbh_edge_fwd_iterator< G > >	859
mln::Object< edge_to_color< I, V > >	773
mln::Function< edge_to_color< I, V > >	654
mln::Function_v2v< edge_to_color< I, V > >	656
mln::Object< enc< V > >	773
mln::Function< enc< V > >	654
mln::Function_v2v< enc< V > >	656
mln::Object< eq< L, R > >	773
mln::Function< eq< L, R > >	654
mln::Function_vv2b< eq< L, R > >	657
mln::fun::vv2b::eq< L, R >	629

mln::Object< extended< I > >	773
mln::Image< extended< I > >	698
mln::Object< extension_fun< I, F > >	773
mln::Image< extension_fun< I, F > >	698
mln::Object< extension_ima< I, J > >	773
mln::Image< extension_ima< I, J > >	698
mln::Object< extension_val< I > >	773
mln::Image< extension_val< I > >	698
mln::Object< f_16_to_8 >	773
mln::Function< f_16_to_8 >	654
mln::Function_v2v< f_16_to_8 >	656
mln::Object< f_box1d_t >	773
mln::Function< f_box1d_t >	654
mln::Function_v2v< f_box1d_t >	656
mln::Function_v2b< f_box1d_t >	656
mln::Object< f_box2d_t >	773
mln::Function< f_box2d_t >	654
mln::Function_v2v< f_box2d_t >	656
mln::Function_v2b< f_box2d_t >	656
mln::Object< f_box3d_t >	773
mln::Function< f_box3d_t >	654
mln::Function_v2v< f_box3d_t >	656
mln::Function_v2b< f_box3d_t >	656
mln::Object< f_hsi_to_rgb< T_rgb > >	773
mln::Function< f_hsi_to_rgb< T_rgb > >	654
mln::Function_v2v< f_hsi_to_rgb< T_rgb > >	656
mln::Object< f_hsl_to_rgb< T_rgb > >	773
mln::Function< f_hsl_to_rgb< T_rgb > >	654
mln::Function_v2v< f_hsl_to_rgb< T_rgb > >	656
mln::Object< f_rgb_to_hsi< T_hsi > >	773
mln::Function< f_rgb_to_hsi< T_hsi > >	654
mln::Function_v2v< f_rgb_to_hsi< T_hsi > >	656
mln::Object< f_rgb_to_hsl< T_hsl > >	773
mln::Function< f_rgb_to_hsl< T_hsl > >	654
mln::Function_v2v< f_rgb_to_hsl< T_hsl > >	656
mln::Object< f_zero_t >	773
mln::Function< f_zero_t >	654
mln::Function_vv2v< f_zero_t >	657
mln::Object< face_bkd_iter< D > >	773
mln::Iterator< face_bkd_iter< D > >	718
complex_iterator_base< topo::face< D >, face_bkd_iter< D > >	
complex_set_iterator_base< topo::face< D >, face_bkd_iter< D > >	
mln::topo::face_bkd_iter< D >	905
mln::Object< face_fwd_iter< D > >	773
mln::Iterator< face_fwd_iter< D > >	718
complex_iterator_base< topo::face< D >, face_fwd_iter< D > >	
complex_set_iterator_base< topo::face< D >, face_fwd_iter< D > >	
mln::topo::face_fwd_iter< D >	907
mln::Object< faces_psite< N, D, P > >	773
mln::Proxy< faces_psite< N, D, P > >	859
mln::Site_Proxy< faces_psite< N, D, P > >	866
mln::Pseudo_Site< faces_psite< N, D, P > >	859

mln::Object< fibonacci_heap< P, T > >	773
mln::util::fibonacci_heap< P, T >	943
mln::Object< flat_image< T, S > >	773
mln::Image< flat_image< T, S > >	698
mln::Object< float01 >	773
mln::Value< float01 >	985
mln::Object< float01_f >	773
mln::Value< float01_f >	985
mln::Object< float_off_loader >	773
mln::Object< float_off_saver >	773
mln::Object< fold< P, dir_0, dir_1, dir_2 > >	773
mln::Function< fold< P, dir_0, dir_1, dir_2 > >	654
mln::Function_v2v< fold< P, dir_0, dir_1, dir_2 > >	656
mln::Object< from_accu< A > >	773
mln::Meta_Function< from_accu< A > >	756
mln::Meta_Function_v2v< from_accu< A > >	758
mln::Object< fun_image< F, I > >	773
mln::Image< fun_image< F, I > >	698
mln::Object< function< meta::blue< value::rgb< n > > > >	773
mln::Function< function< meta::blue< value::rgb< n > > > >	654
mln::Function_v2v< function< meta::blue< value::rgb< n > > > >	656
mln::Object< function< meta::first< util::couple< T, U > > > >	773
mln::Function< function< meta::first< util::couple< T, U > > > >	654
mln::Function_v2v< function< meta::first< util::couple< T, U > > > >	656
mln::Object< function< meta::green< value::rgb< n > > > >	773
mln::Function< function< meta::green< value::rgb< n > > > >	654
mln::Function_v2v< function< meta::green< value::rgb< n > > > >	656
mln::Object< function< meta::red< value::rgb< n > > > >	773
mln::Function< function< meta::red< value::rgb< n > > > >	654
mln::Function_v2v< function< meta::red< value::rgb< n > > > >	656
mln::Object< function< meta::second< util::couple< T, U > > > >	773
mln::Function< function< meta::second< util::couple< T, U > > > >	654
mln::Function_v2v< function< meta::second< util::couple< T, U > > > >	656
mln::Object< function< meta::to_enc< T > > >	773
mln::Function< function< meta::to_enc< T > > >	654
mln::Function_v2v< function< meta::to_enc< T > > >	656
mln::Object< fwd_pixter1d< I > >	773
mln::Iterator< fwd_pixter1d< I > >	718
Pixel_Iterator< fwd_pixter1d< I > >	
mln::Object< fwd_pixter2d< I > >	773
mln::Iterator< fwd_pixter2d< I > >	718
Pixel_Iterator< fwd_pixter2d< I > >	
mln::Object< fwd_pixter3d< I > >	773
mln::Iterator< fwd_pixter3d< I > >	718
Pixel_Iterator< fwd_pixter3d< I > >	
mln::Object< fwd_t >	773
mln::Browsing< fwd_t >	520
mln::canvas::browsing::fwd_t	528
mln::Object< ge< L, R > >	773
mln::Function< ge< L, R > >	654
mln::Function_vv2b< ge< L, R > >	657
mln::fun::vv2b::ge< L, R >	629

mln::Object< graph >	773
mln::Graph< graph >	669
graph_base< graph >	
mln::util::graph	945
mln::Object< graph_elt_mixed_window< G, S, S2 > >	773
mln::Window< graph_elt_mixed_window< G, S, S2 > >	1038
mln::graph_window_base< S2::fun_t::result, graph_elt_mixed_window< G, S, S2 > >	688
mln::graph_elt_mixed_window< G, S, S2 >	672
mln::Object< graph_elt_window< G, S > >	773
mln::Window< graph_elt_window< G, S > >	1038
mln::graph_window_base< S::fun_t::result, graph_elt_window< G, S > >	688
mln::graph_elt_window< G, S >	679
mln::Object< graph_elt_window_if< G, S, I > >	773
mln::Window< graph_elt_window_if< G, S, I > >	1038
mln::graph_window_base< S::fun_t::result, graph_elt_window_if< G, S, I > >	688
mln::graph_elt_window_if< G, S, I >	683
mln::Object< graph_window_if_piter< S, W, I > >	773
mln::Proxy< graph_window_if_piter< S, W, I > >	859
mln::Site_Proxy< graph_window_if_piter< S, W, I > >	866
mln::Site_Iterator< graph_window_if_piter< S, W, I > >	865
site_iterator_base< W, graph_window_if_piter< S, W, I > >	
site_relative_iterator_base< W, graph_window_if_piter< S, W, I > >	
mln::graph_window_if_piter< S, W, I >	690
mln::Object< graph_window_piter< S, W, I > >	773
mln::Proxy< graph_window_piter< S, W, I > >	859
mln::Site_Proxy< graph_window_piter< S, W, I > >	866
mln::Site_Iterator< graph_window_piter< S, W, I > >	865
site_iterator_base< W, graph_window_piter< S, W, I > >	
site_relative_iterator_base< W, graph_window_piter< S, W, I >, W::center_t >	
mln::graph_window_piter< S, W, I >	691
mln::Object< gray_f >	773
mln::Value< gray_f >	985
mln::Object< graylevel< n > >	773
mln::Value< graylevel< n > >	985
mln::Object< graylevel_f >	773
mln::Value< graylevel_f >	985
mln::Object< green >	773
mln::Meta_Function< green >	756
mln::Meta_Function_v2v< green >	758
mln::Object< green_t >	773
mln::Literal< green_t >	729
mln::literal::green_t	734
mln::Object< gt< L, R > >	773
mln::Function< gt< L, R > >	654
mln::Function_vv2b< gt< L, R > >	657
mln::fun::vv2b::gt< L, R >	630
mln::Object< has< I > >	773
mln::Function< has< I > >	654
mln::Function_v2v< has< I > >	656
mln::Function_v2b< has< I > >	656
mln::Object< height >	773
mln::Meta_Accumulator< height >	754
mln::accu::meta::shape::height	446

mln::Object< height< I > >	773
mln::Proxy< height< I > >	859
mln::Accumulator< height< I > >	498
base< unsigned, height< I > >	
mln::accu::shape::height< I >	466
mln::morpho::attribute::height< I >	764
mln::Object< hexa >	773
mln::Mesh< hexa >	753
mln::Regular_Grid< hexa >	862
mln::Object< hexa< I > >	773
mln::Image< hexa< I > >	698
mln::Object< hexa< image2d< V > > >	773
mln::Image< hexa< image2d< V > > >	698
mln::Object< histo >	773
mln::Meta_Accumulator< histo >	754
mln::accu::meta::histo	429
mln::Object< histo3d_rgb >	773
mln::Meta_Accumulator< histo3d_rgb >	754
mln::Object< histo3d_rgb< V > >	773
mln::Proxy< histo3d_rgb< V > >	859
mln::Accumulator< histo3d_rgb< V > >	498
base< image3d< unsigned >, histo3d_rgb< V > >	
mln::accu::stat::histo3d_rgb< V >	473
mln::Object< histo< V > >	773
mln::Proxy< histo< V > >	859
mln::Accumulator< histo< V > >	498
base< const std::vector< unsigned > &, histo< V > >	
mln::accu::histo< V >	411
mln::Object< hyper_directional_t >	773
mln::Browsing< hyper_directional_t >	520
mln::canvas::browsing::hyper_directional_t	529
mln::Object< id2element< G, Elt > >	773
mln::Function< id2element< G, Elt > >	654
mln::Function_v2v< id2element< G, Elt > >	656
mln::Object< identity_t >	773
mln::Literal< identity_t >	729
mln::literal::identity_t	735
mln::Object< ignore >	773
mln::util::ignore	952
mln::Object< image1d< T > >	773
mln::Image< image1d< T > >	698
mln::Object< image2d< T > >	773
mln::Image< image2d< T > >	698
mln::Object< image3d< T > >	773
mln::Image< image3d< T > >	698
mln::Object< image3d< t_label > >	773
mln::Image< image3d< t_label > >	698
mln::Object< image3d< unsigned > >	773
mln::Image< image3d< unsigned > >	698
mln::Object< image< F, S > >	773
mln::Image< image< F, S > >	698
mln::Object< image_if< I, F > >	773

mln::Image< image_if< I, F > >	698
mln::Object< implies< L, R > >	773
mln::Function< implies< L, R > >	654
mln::Function_vv2b< implies< L, R > >	657
mln::fun::vv2b::implies< L, R >	631
mln::Object< index_of_value< bool > >	773
mln::Function< index_of_value< bool > >	654
mln::Function_v2v< index_of_value< bool > >	656
mln::Object< index_of_value< T > >	773
mln::Function< index_of_value< T > >	654
mln::Function_v2v< index_of_value< T > >	656
mln::Object< inf >	773
mln::Meta_Accumulator< inf >	754
mln::accu::meta::math::inf	437
mln::Meta_Function< inf >	756
mln::Meta_Function_vv2v< inf >	759
mln::Object< inf< T > >	773
mln::Proxy< inf< T > >	859
mln::Accumulator< inf< T > >	498
base< const T &, inf< T > >	
mln::accu::math::inf< T >	421
mln::Object< int_s< n > >	773
mln::Object< int_u8_off_saver >	773
mln::Object< int_u< n > >	773
mln::Object< int_u_sat< n > >	773
mln::Value< int_u_sat< n > >	985
mln::Object< interpolated< I, F > >	773
mln::Image< interpolated< I, F > >	698
mln::Object< iota >	773
mln::Function< iota >	654
mln::Function_v2v< iota >	656
mln::Object< is_dot >	773
mln::Function< is_dot >	654
mln::Function_v2v< is_dot >	656
mln::Function_v2b< is_dot >	656
mln::Object< is_edge >	773
mln::Function< is_edge >	654
mln::Function_v2v< is_edge >	656
mln::Function_v2b< is_edge >	656
mln::Object< is_n_face< N > >	773
mln::Function< is_n_face< N > >	654
mln::Function_v2v< is_n_face< N > >	656
mln::Function_v2b< is_n_face< N > >	656
mln::topo::is_n_face< N >	908
mln::Object< is_pixel >	773
mln::Function< is_pixel >	654
mln::Function_v2v< is_pixel >	656
mln::Function_v2b< is_pixel >	656
mln::Object< is_row_odd >	773
mln::Function< is_row_odd >	654
mln::Function_v2v< is_row_odd >	656
mln::Function_v2b< is_row_odd >	656
mln::Object< is_separator >	773

mln::Function< is_separator >	654
mln::Function_v2v< is_separator >	656
mln::Function_v2b< is_separator >	656
mln::world::inter_pixel::is_separator	1042
mln::Object< is_simple_cell< I > >	773
mln::Function< is_simple_cell< I > >	654
mln::Function_v2v< is_simple_cell< I > >	656
mln::Function_v2b< is_simple_cell< I > >	656
mln::topo::is_simple_cell< I >	909
mln::Object< ithcomp >	773
mln::Meta_Function< ithcomp >	756
mln::Meta_Function_vv2v< ithcomp >	759
mln::Object< keep_specific_colors >	773
mln::Function< keep_specific_colors >	654
mln::Function_v2v< keep_specific_colors >	656
mln::Function_v2b< keep_specific_colors >	656
mln::Object< l1 >	773
mln::Meta_Function< l1 >	756
mln::Meta_Function_v2v< l1 >	758
mln::Object< l1_norm< V > >	773
mln::Function< l1_norm< V > >	654
mln::Function_v2v< l1_norm< V > >	656
mln::Object< l1_norm< V, R > >	773
mln::Function< l1_norm< V, R > >	654
mln::Function_v2v< l1_norm< V, R > >	656
mln::fun::v2v::l1_norm< V, R >	619
mln::fun::v2w_w2v::l1_norm< V, R >	625
mln::Object< l2 >	773
mln::Meta_Function< l2 >	756
mln::Meta_Function_v2v< l2 >	758
mln::Object< l2_norm< V, R > >	773
mln::Function< l2_norm< V, R > >	654
mln::Function_v2v< l2_norm< V, R > >	656
mln::fun::v2v::l2_norm< V, R >	620
mln::fun::v2w_w2v::l2_norm< V, R >	626
mln::Object< label< n > >	773
mln::Value< label< n > >	985
mln::Object< label_used >	773
mln::Meta_Accumulator< label_used >	754
mln::accu::meta::label_used	430
mln::Object< label_used< L > >	773
mln::Proxy< label_used< L > >	859
mln::Accumulator< label_used< L > >	498
base< const fun::i2v::array< bool > &, label_used< L > > mln::accu::label_used< L >	412
mln::Object< labeled_image< I > >	773
mln::Image< labeled_image< I > >	698
mln::Object< land >	773
mln::Meta_Accumulator< land >	754
mln::accu::meta::logic::land	431
mln::Proxy< land >	859
mln::Accumulator< land >	498
base< bool, land >	

mln::accu::logic::land	413
mln::Object< land< L, R > >	773
mln::Function< land< L, R > >	654
mln::Function_vv2v< land< L, R > >	657
mln::fun::vv2v::land< L, R >	635
mln::Object< land_basic >	773
mln::Meta_Accumulator< land_basic >	754
mln::accu::meta::logic::land_basic	432
mln::Proxy< land_basic >	859
mln::Accumulator< land_basic >	498
base< bool, land_basic >	
mln::accu::logic::land_basic	415
mln::Object< land_not< L, R > >	773
mln::Function< land_not< L, R > >	654
mln::Function_vv2v< land_not< L, R > >	657
mln::fun::vv2v::land_not< L, R >	636
mln::Object< lap_fill2_k2_t< V > >	773
mln::Function< lap_fill2_k2_t< V > >	654
mln::Function_vv2v< lap_fill2_k2_t< V > >	657
mln::Object< lazy_image< I, F, B > >	773
mln::Image< lazy_image< I, F, B > >	698
mln::Object< le< L, R > >	773
mln::Function< le< L, R > >	654
mln::Function_vv2b< le< L, R > >	657
mln::fun::vv2b::le< L, R >	632
mln::Object< light_gray_t >	773
mln::Literal< light_gray_t >	729
mln::literal::light_gray_t	736
mln::Object< lime_t >	773
mln::Literal< lime_t >	729
mln::literal::lime_t	737
mln::Object< line< M, i, C > >	773
mln::Window< line< M, i, C > >	1038
window_base< dpoint< M, C >, line< M, i, C > >	
classical_window_base< dpoint< M, C >, line< M, i, C > >	
mln::win::line< M, i, C >	1033
mln::Object< line_graph< G > >	773
mln::Graph< line_graph< G > >	669
graph_base< line_graph< G > >	
mln::util::line_graph< G >	953
mln::Object< linear< V, T, R > >	773
mln::Function< linear< V, T, R > >	654
mln::Function_v2v< linear< V, T, R > >	656
mln::fun::v2v::linear< V, T, R >	621
mln::Object< linear_sat< V, T, R > >	773
mln::Function< linear_sat< V, T, R > >	654
mln::Function_v2v< linear_sat< V, T, R > >	656
mln::Object< linfty >	773
mln::Meta_Function< linfty >	756
mln::Meta_Function_v2v< linfty >	758
mln::Object< linfty_norm< V, R > >	773
mln::Function< linfty_norm< V, R > >	654
mln::Function_v2v< linfty_norm< V, R > >	656

mln::fun::v2v::linfty_norm< V, R >	622
mln::fun::v2w_w2v::linfty_norm< V, R >	628
mln::Object< lnot< V > >	773
mln::Function< lnot< V > >	654
mln::Function_v2v< lnot< V > >	656
mln::Function_v2b< lnot< V > >	656
mln::fun::v2b::lnot< V >	614
mln::Object< lor >	773
mln::Meta_Accumulator< lor >	754
mln::accu::meta::logic::lor	433
mln::Proxy< lor >	859
mln::Accumulator< lor >	498
base< bool, lor >	
mln::accu::logic::lor	416
mln::Object< lor< L, R > >	773
mln::Function< lor< L, R > >	654
mln::Function_vv2v< lor< L, R > >	657
mln::fun::vv2v::lor< L, R >	637
mln::Object< lor_basic >	773
mln::Meta_Accumulator< lor_basic >	754
mln::accu::meta::logic::lor_basic	434
mln::Proxy< lor_basic >	859
mln::Accumulator< lor_basic >	498
base< bool, lor_basic >	
mln::accu::logic::lor_basic	417
mln::Object< lt< L, R > >	773
mln::Function< lt< L, R > >	654
mln::Function_vv2b< lt< L, R > >	657
mln::fun::vv2b::lt< L, R >	633
mln::Object< lut_vec< S, T > >	773
Value_Set< lut_vec< S, T > >	
mln::value::lut_vec< S, T >	1005
mln::Object< lxor< L, R > >	773
mln::Function< lxor< L, R > >	654
mln::Function_vv2v< lxor< L, R > >	657
mln::fun::vv2v::lxor< L, R >	638
mln::Object< magenta_t >	773
mln::Literal< magenta_t >	729
mln::literal::magenta_t	738
mln::Object< mahalanobis< V > >	773
mln::Function< mahalanobis< V > >	654
mln::Function_v2v< mahalanobis< V > >	656
mln::Object< maj_h >	773
mln::Meta_Accumulator< maj_h >	754
mln::accu::meta::maj_h	435
mln::Object< maj_h< T > >	773
mln::Proxy< maj_h< T > >	859
mln::Accumulator< maj_h< T > >	498
base< const T &, maj_h< T > >	
mln::accu::maj_h< T >	418
mln::Object< mat< n, m, T > >	773
mln::Object< max >	773
mln::Meta_Accumulator< max >	754

mln::accu::meta::stat::max	448
mln::Object< max< T > >	773
mln::Proxy< max< T > >	859
mln::Accumulator< max< T > >	498
base< const T &, max< T > >	
mln::accu::stat::max< T >	475
mln::Object< max< V > >	773
mln::Function< max< V > >	654
mln::Function_vv2v< max< V > >	657
mln::fun::vv2v::max< V >	639
mln::Object< max_h >	773
mln::Meta_Accumulator< max_h >	754
mln::accu::meta::stat::max_h	449
mln::Object< max_h< V > >	773
mln::Proxy< max_h< V > >	859
mln::Accumulator< max_h< V > >	498
base< const V &, max_h< V > >	
mln::accu::stat::max_h< V >	476
mln::Object< max_site >	773
mln::Meta_Accumulator< max_site >	754
mln::accu::meta::max_site	440
mln::Object< max_site< I > >	773
mln::Proxy< max_site< I > >	859
mln::Accumulator< max_site< I > >	498
base< I::psite, max_site< I > >	
mln::accu::max_site< I >	425
mln::Object< max_t >	773
mln::Literal< max_t >	729
mln::literal::max_t	739
mln::Object< mean >	773
mln::Meta_Accumulator< mean >	754
mln::accu::meta::stat::mean	450
mln::Meta_Function< mean >	756
mln::Meta_Function_v2v< mean >	758
mln::Object< mean< T, S, M > >	773
mln::Proxy< mean< T, S, M > >	859
mln::Accumulator< mean< T, S, M > >	498
base< M, mean< T, S, M > >	
mln::accu::stat::mean< T, S, M >	477
mln::Object< median_alt< S > >	773
mln::Proxy< median_alt< S > >	859
mln::Accumulator< median_alt< S > >	498
base< const S::value &, median_alt< S > >	
mln::accu::stat::median_alt< S >	479
mln::Object< median_alt< T > >	773
mln::Meta_Accumulator< median_alt< T > >	754
mln::accu::meta::stat::median_alt< T >	451
mln::Object< median_alt< value::set< T > > >	773
mln::Proxy< median_alt< value::set< T > > >	859
mln::Accumulator< median_alt< value::set< T > > >	498
base< const value::set< T >::value &, median_alt< value::set< T > > >	
mln::accu::stat::median_alt< value::set< T > >	479
mln::Object< median_h >	773

mln::Meta_Accumulator< median_h >	754
mln::accu::meta::stat::median_h	452
mln::Object< median_h< typename I::value > >	773
mln::Proxy< median_h< typename I::value > >	859
mln::Accumulator< median_h< typename I::value > >	498
base< const typename I::value &, median_h< typename I::value > >	481
mln::accu::stat::median_h< typename I::value >	481
mln::Object< median_h< V > >	773
mln::Proxy< median_h< V > >	859
mln::Accumulator< median_h< V > >	498
base< const V &, median_h< V > >	481
mln::accu::stat::median_h< V >	481
mln::Object< medium_gray_t >	773
mln::Literal< medium_gray_t >	729
mln::Object< min >	773
mln::Meta_Accumulator< min >	754
mln::accu::meta::stat::min	453
mln::Object< min< L, R > >	773
mln::Function< min< L, R > >	654
mln::Function_vv2v< min< L, R > >	657
mln::fun::vv2v::min< L, R >	640
mln::Object< min< T > >	773
mln::Proxy< min< T > >	859
mln::Accumulator< min< T > >	498
base< const T &, min< T > >	483
mln::accu::stat::min< T >	483
mln::Object< min_h >	773
mln::Meta_Accumulator< min_h >	754
mln::accu::meta::stat::min_h	454
mln::Object< min_h< V > >	773
mln::Proxy< min_h< V > >	859
mln::Accumulator< min_h< V > >	498
base< const V &, min_h< V > >	485
mln::accu::stat::min_h< V >	485
mln::Object< min_t >	773
mln::Literal< min_t >	729
mln::literal::min_t	740
mln::Object< mirror< B > >	773
mln::Function< mirror< B > >	654
mln::Function_v2v< mirror< B > >	656
mln::Object< mixed_neighb< W > >	773
mln::Neighborhood< mixed_neighb< W > >	772
mln::Object< mln::util::set< T > >	773
mln::util::set< T >	964
mln::Object< multi_site< P > >	773
mln::Object< multiple< W, F > >	773
mln::Window< multiple< W, F > >	1038
window_base< W::dpsite, multiple< W, F > >	1034
mln::win::multiple< W, F >	1034
mln::Object< multiple_qiter< W, F > >	773
mln::Proxy< multiple_qiter< W, F > >	859
mln::Site_Proxy< multiple_qiter< W, F > >	866
mln::Site_Iterator< multiple_qiter< W, F > >	865

mln::Object< multiple_size< n, W, F > >	773
mln::Window< multiple_size< n, W, F > >	1038
window_base< W::dpsite, multiple_size< n, W, F > >	
mln::win::multiple_size< n, W, F >	1035
mln::Object< multiple_size_qiter< n, W, F > >	773
mln::Proxy< multiple_size_qiter< n, W, F > >	859
mln::Site_Proxy< multiple_size_qiter< n, W, F > >	866
mln::Site_Iterator< multiple_size_qiter< n, W, F > >	865
mln::Object< my_box2d >	773
mln::Function< my_box2d >	654
mln::Function_v2v< my_box2d >	656
mln::Function_v2b< my_box2d >	656
mln::Object< my_ext >	773
mln::Function< my_ext >	654
mln::Function_v2v< my_ext >	656
mln::Object< my_fun< G > >	773
mln::Function< my_fun< G > >	654
mln::Object< my_image2d< T > >	773
mln::Image< my_image2d< T > >	698
mln::Object< my_values_t >	773
mln::Function< my_values_t >	654
mln::Function_v2v< my_values_t >	656
mln::Object< myfun >	773
mln::Function< myfun >	654
mln::Function_vv2v< myfun >	657
mln::Object< mysqrt >	773
mln::Function< mysqrt >	654
mln::Function_v2v< mysqrt >	656
mln::Object< n_face_bkd_iter< D > >	773
mln::Iterator< n_face_bkd_iter< D > >	718
complex_iterator_base< topo::face< D >, n_face_bkd_iter< D > >	
complex_set_iterator_base< topo::face< D >, n_face_bkd_iter< D > >	
mln::topo::n_face_bkd_iter< D >	915
mln::Object< n_face_fwd_iter< D > >	773
mln::Iterator< n_face_fwd_iter< D > >	718
complex_iterator_base< topo::face< D >, n_face_fwd_iter< D > >	
complex_set_iterator_base< topo::face< D >, n_face_fwd_iter< D > >	
mln::topo::n_face_fwd_iter< D >	916
mln::Object< neighb< graph_elt_mixed_window< G, S, S2 > > >	773
mln::Neighborhood< neighb< graph_elt_mixed_window< G, S, S2 > > >	772
mln::Object< neighb< graph_elt_window< G, S > > >	773
mln::Neighborhood< neighb< graph_elt_window< G, S > > >	772
mln::Object< neighb< graph_elt_window_if< G, S, I > > >	773
mln::Neighborhood< neighb< graph_elt_window_if< G, S, I > > >	772
mln::Object< neighb< W > >	773
mln::Neighborhood< neighb< W > >	772
mln::Object< neighb_bkd_niter< W > >	773
mln::Proxy< neighb_bkd_niter< W > >	859
mln::Site_Proxy< neighb_bkd_niter< W > >	866
mln::Site_Iterator< neighb_bkd_niter< W > >	865
mln::Object< neighb_fwd_niter< W > >	773
mln::Proxy< neighb_fwd_niter< W > >	859

mln::Site_Proxy< neighb_fwd_niter< W > >	866
mln::Site_Iterator< neighb_fwd_niter< W > >	865
mln::Object< nil >	773
mln::Meta_Accumulator< nil >	754
mln::accu::meta::nil	441
mln::util::nil	958
mln::Object< nil< T > >	773
mln::Proxy< nil< T > >	859
mln::Accumulator< nil< T > >	498
base< util::ignore, nil< T > >	
mln::accu::nil< T >	459
mln::Object< not_to_remove >	773
mln::Function< not_to_remove >	654
mln::Function_v2v< not_to_remove >	656
mln::Function_v2b< not_to_remove >	656
mln::Object< object_id< Tag, V > >	773
mln::Value< object_id< Tag, V > >	985
mln::Object< octagon2d >	773
mln::Window< octagon2d >	1038
window_base< dpoint2d, octagon2d >	
classical_window_base< dpoint2d, octagon2d >	
mln::win::octagon2d	1035
mln::Object< olive_t >	773
mln::Literal< olive_t >	729
mln::literal::olive_t	741
mln::Object< one_t >	773
mln::Literal< one_t >	729
mln::literal::one_t	742
mln::Object< orange_t >	773
mln::Literal< orange_t >	729
mln::literal::orange_t	743
mln::Object< ord_pair< T > >	773
mln::util::ord_pair< T >	960
mln::Object< origin_t >	773
mln::Literal< origin_t >	729
mln::literal::origin_t	744
mln::Object< P >	773
Point_Site< P >	
mln::Point< P >	851
mln::Object< p2p_image< I, F > >	773
mln::Image< p2p_image< I, F > >	698
mln::Object< p< A > >	773
mln::Proxy< p< A > >	859
mln::Accumulator< p< A > >	498
base< const A::result &, p< A > >	
mln::accu::p< A >	460
mln::Object< p< mA > >	773
mln::Meta_Accumulator< p< mA > >	754
mln::accu::meta::p< mA >	442
mln::Object< p_array< P > >	773
mln::Site_Set< p_array< P > >	867
site_set_base< P, p_array< P > >	
mln::p_array< P >	775

mln::Object< p_centered< W > >	773
mln::Site_Set< p_centered< W > >	867
site_set_base_< W::psite, p_centered< W > >	
mln::p_centered< W >	779
mln::Object< p_centered_piter< W > >	773
mln::Proxy< p_centered_piter< W > >	859
mln::Site_Proxy< p_centered_piter< W > >	866
mln::Site_Iterator< p_centered_piter< W > >	865
mln::Object< p_complex< D, G > >	773
mln::Site_Set< p_complex< D, G > >	867
site_set_base_< complex_psite< D, G >, p_complex< D, G > >	
mln::p_complex< D, G >	782
mln::Object< p_double_piter< S, I1, I2 > >	773
mln::Proxy< p_double_piter< S, I1, I2 > >	859
mln::Site_Proxy< p_double_piter< S, I1, I2 > >	866
mln::Site_Iterator< p_double_piter< S, I1, I2 > >	865
mln::Object< p_double_psite< S, Sp > >	773
mln::Proxy< p_double_psite< S, Sp > >	859
mln::Site_Proxy< p_double_psite< S, Sp > >	866
mln::Pseudo_Site< p_double_psite< S, Sp > >	859
mln::Object< p_edges< G, F > >	773
mln::Site_Set< p_edges< G, F > >	867
site_set_base_< F::result, p_edges< G, F > >	
mln::p_edges< G, F >	785
mln::Object< p_edges_psite< G, F > >	773
mln::Proxy< p_edges_psite< G, F > >	859
mln::Site_Proxy< p_edges_psite< G, F > >	866
mln::Pseudo_Site< p_edges_psite< G, F > >	859
mln::Object< p_faces< N, D, P > >	773
mln::Site_Set< p_faces< N, D, P > >	867
site_set_base_< faces_psite< N, D, P >, p_faces< N, D, P > >	
mln::p_faces< N, D, P >	790
mln::Object< p_graph_piter< S, I > >	773
mln::Proxy< p_graph_piter< S, I > >	859
mln::Site_Proxy< p_graph_piter< S, I > >	866
mln::Site_Iterator< p_graph_piter< S, I > >	865
site_iterator_base< S, p_graph_piter< S, I > >	
site_set_iterator_base< S, p_graph_piter< S, I > >	
mln::p_graph_piter< S, I >	792
mln::Object< p_if< S, F > >	773
mln::Site_Set< p_if< S, F > >	867
site_set_base_< S::psite, p_if< S, F > >	
mln::p_if< S, F >	794
mln::Object< p_image< I > >	773
mln::Site_Set< p_image< I > >	867
site_set_base_< I::psite, p_image< I > >	
mln::p_image< I >	796
mln::Object< p_indexed_bkd_piter< S > >	773
mln::Proxy< p_indexed_bkd_piter< S > >	859
mln::Site_Proxy< p_indexed_bkd_piter< S > >	866
mln::Site_Iterator< p_indexed_bkd_piter< S > >	865
site_iterator_base< S, p_indexed_bkd_piter< S > >	
site_set_iterator_base< S, p_indexed_bkd_piter< S > >	

mln::p_indexed_bkd_piter< S >	800
mln::Object< p_indexed_fwd_piter< S > >	773
mln::Proxy< p_indexed_fwd_piter< S > >	859
mln::Site_Proxy< p_indexed_fwd_piter< S > >	866
mln::Site_Iterator< p_indexed_fwd_piter< S > >	865
site_iterator_base< S, p_indexed_fwd_piter< S > >	
site_set_iterator_base< S, p_indexed_fwd_piter< S > >	
mln::p_indexed_fwd_piter< S >	801
mln::Object< p_indexed_psite< S > >	773
mln::Proxy< p_indexed_psite< S > >	859
mln::Site_Proxy< p_indexed_psite< S > >	866
mln::Pseudo_Site< p_indexed_psite< S > >	859
pseudo_site_base< const S::element &, p_indexed_psite< S > >	
mln::p_indexed_psite< S >	802
mln::Object< p_key< K, P > >	773
mln::Site_Set< p_key< K, P > >	867
site_set_base< P, p_key< K, P > >	
mln::p_key< K, P >	802
mln::Object< p_line2d >	773
mln::Site_Set< p_line2d >	867
site_set_base< point2d, p_line2d >	
mln::p_line2d	807
mln::Object< p_mutable_array_of< S > >	773
mln::Site_Set< p_mutable_array_of< S > >	867
site_set_base< S::site, p_mutable_array_of< S > >	
mln::p_mutable_array_of< S >	810
mln::Object< p_n_faces_bkd_piter< D, G > >	773
mln::Proxy< p_n_faces_bkd_piter< D, G > >	859
mln::Site_Proxy< p_n_faces_bkd_piter< D, G > >	866
mln::Site_Iterator< p_n_faces_bkd_piter< D, G > >	865
site_iterator_base< p_complex< D, G >, p_n_faces_bkd_piter< D, G > >	
site_set_iterator_base< p_complex< D, G >, p_n_faces_bkd_piter< D, G > >	
p_complex_piter_base< topo::n_face_bkd_iter< D >, p_complex< D, G >, G::site, p_n_faces_bkd_piter< D, G > >	
mln::p_n_faces_bkd_piter< D, G >	813
mln::Object< p_n_faces_fwd_piter< D, G > >	773
mln::Proxy< p_n_faces_fwd_piter< D, G > >	859
mln::Site_Proxy< p_n_faces_fwd_piter< D, G > >	866
mln::Site_Iterator< p_n_faces_fwd_piter< D, G > >	865
site_iterator_base< p_complex< D, G >, p_n_faces_fwd_piter< D, G > >	
site_set_iterator_base< p_complex< D, G >, p_n_faces_fwd_piter< D, G > >	
p_complex_piter_base< topo::n_face_fwd_iter< D >, p_complex< D, G >, G::site, p_n_faces_fwd_piter< D, G > >	
mln::p_n_faces_fwd_piter< D, G >	814
mln::Object< p_priority< P, Q > >	773
mln::Site_Set< p_priority< P, Q > >	867
site_set_base< Q::site, p_priority< P, Q > >	
mln::p_priority< P, Q >	815
mln::Object< p_queue< P > >	773
mln::Site_Set< p_queue< P > >	867
site_set_base< P, p_queue< P > >	
mln::p_queue< P >	820
mln::Object< p_queue_fast< P > >	773
mln::Site_Set< p_queue_fast< P > >	867

site_set_base_< P, p_queue_fast< P > >	
mln::p_queue_fast< P >	824
mln::Object< p_run< P > >	773
mln::Site_Set< p_run< P > >	867
site_set_base_< P, p_run< P > >	
mln::p_run< P >	828
mln::Object< p_run_psite< P > >	773
mln::Proxy< p_run_psite< P > >	859
mln::Site_Proxy< p_run_psite< P > >	866
mln::Pseudo_Site< p_run_psite< P > >	859
mln::Object< p_set< P > >	773
mln::Site_Set< p_set< P > >	867
site_set_base_< P, p_set< P > >	
mln::p_set< P >	832
mln::Object< p_set_of< S > >	773
mln::Site_Set< p_set_of< S > >	867
mln::Object< p_transformed< S, F > >	773
mln::Site_Set< p_transformed< S, F > >	867
site_set_base_< S::psite, p_transformed< S, F > >	
mln::p_transformed< S, F >	835
mln::Object< p_transformed_piter< Pi, S, F > >	773
mln::Proxy< p_transformed_piter< Pi, S, F > >	859
mln::Site_Proxy< p_transformed_piter< Pi, S, F > >	866
mln::Site_Iterator< p_transformed_piter< Pi, S, F > >	865
mln::Object< p_vaccess< V, S > >	773
mln::Site_Set< p_vaccess< V, S > >	867
site_set_base_< S::site, p_vaccess< V, S > >	
mln::p_vaccess< V, S >	839
mln::Object< p_vertices< G, F > >	773
mln::Site_Set< p_vertices< G, F > >	867
site_set_base_< F::result, p_vertices< G, F > >	
mln::p_vertices< G, F >	843
mln::Object< p_vertices_psite< G, F > >	773
mln::Proxy< p_vertices_psite< G, F > >	859
mln::Site_Proxy< p_vertices_psite< G, F > >	866
mln::Pseudo_Site< p_vertices_psite< G, F > >	859
mln::Object< pair< A1, A2 > >	773
mln::Meta_Accumulator< pair< A1, A2 > >	754
mln::accu::meta::pair< A1, A2 >	443
mln::Object< pair< A1, A2, T > >	773
mln::Proxy< pair< A1, A2, T > >	859
mln::Accumulator< pair< A1, A2, T > >	498
base< std::pair< A1::result, A2::result >, pair< A1, A2, T > >	
mln::accu::pair< A1, A2, T >	462
mln::Object< pair< min< V >, max< V >, mln_argument(min< V >) > >	773
mln::Proxy< pair< min< V >, max< V >, mln_argument(min< V >) > >	859
mln::Accumulator< pair< min< V >, max< V >, mln_argument(min< V >) > >	498
mln::Object< pink_t >	773
mln::Literal< pink_t >	729
mln::literal::pink_t	745
mln::Object< pixel< I > >	773
mln::pixel< I >	848
mln::Object< plain< I > >	773

mln::Image< plain< I > >	698
mln::Object< point< G, C > >	773
mln::Site< point< G, C > >	863
mln::Gpoint< point< G, C > >	665
mln::point< G, C >	853
mln::Object< point_from_value< T > >	773
mln::Function< point_from_value< T > >	654
mln::Function_v2v< point_from_value< T > >	656
mln::Object< projection< P, dir > >	773
mln::Function< projection< P, dir > >	654
mln::Function_v2v< projection< P, dir > >	656
mln::Object< proxy< I > >	773
mln::Proxy< proxy< I > >	859
mln::value::proxy< I >	1007
mln::Object< purple_t >	773
mln::Literal< purple_t >	729
mln::literal::purple_t	746
mln::Object< qrde >	773
mln::Function< qrde >	654
mln::Function_v2v< qrde >	656
mln::Object< qt_rgb_to_int_u< n > >	773
mln::Function< qt_rgb_to_int_u< n > >	654
mln::Function_v2v< qt_rgb_to_int_u< n > >	656
mln::Object< quat >	773
mln::Value< quat >	985
mln::Object< rank >	773
mln::Meta_Accumulator< rank >	754
mln::accu::meta::stat::rank	455
mln::Object< rank< bool > >	773
mln::Proxy< rank< bool > >	859
mln::Accumulator< rank< bool > >	498
base< bool, rank< bool > >	
mln::accu::stat::rank< bool >	489
mln::Object< rank< T > >	773
mln::Proxy< rank< T > >	859
mln::Accumulator< rank< T > >	498
base< const T &, rank< T > >	
mln::accu::stat::rank< T >	488
mln::Object< rank_high_quant >	773
mln::Meta_Accumulator< rank_high_quant >	754
mln::accu::meta::stat::rank_high_quant	456
mln::Object< rank_high_quant< T > >	773
mln::Proxy< rank_high_quant< T > >	859
mln::Accumulator< rank_high_quant< T > >	498
base< const T &, rank_high_quant< T > >	
mln::accu::stat::rank_high_quant< T >	491
mln::Object< rectangle2d >	773
mln::Window< rectangle2d >	1038
window_base< dpoint2d, rectangle2d >	
classical_window_base< dpoint2d, rectangle2d >	
mln::win::rectangle2d	1036
mln::Object< rectangularity< P > >	773

mln::Proxy< rectangularity< P > >	859
mln::Accumulator< rectangularity< P > >	498
mln::Object< red >	773
mln::Meta_Function< red >	756
mln::Meta_Function_v2v< red >	758
mln::Object< red_t >	773
mln::Literal< red_t >	729
mln::literal:red_t	747
mln::Object< ref_data >	773
mln::Function< ref_data >	654
mln::Function_v2v< ref_data >	656
mln::Object< rgb32 >	773
mln::Value< rgb32 >	985
mln::Object< rgb8_off_loader >	773
mln::Object< rgb8_off_saver >	773
mln::Object< rgb8_to_rgbn< n > >	773
mln::Function< rgb8_to_rgbn< n > >	654
mln::Function_v2v< rgb8_to_rgbn< n > >	656
mln::fun::v2v::rgb8_to_rgbn< n >	623
mln::Object< rgb< n > >	773
mln::Value< rgb< n > >	985
mln::Object< rgb_to_dist< T, n > >	773
mln::Function< rgb_to_dist< T, n > >	654
mln::Function_v2v< rgb_to_dist< T, n > >	656
mln::Object< rgb_to_int_u< n > >	773
mln::Function< rgb_to_int_u< n > >	654
mln::Function_v2v< rgb_to_int_u< n > >	656
mln::Object< rgb_to_luma< T_luma > >	773
mln::Function< rgb_to_luma< T_luma > >	654
mln::Function_v2v< rgb_to_luma< T_luma > >	656
mln::Object< rgbn_to_lbl8< n > >	773
mln::Function< rgbn_to_lbl8< n > >	654
mln::Function_v2v< rgbn_to_lbl8< n > >	656
mln::Object< rms >	773
mln::Meta_Accumulator< rms >	754
mln::accu::meta::rms	444
mln::Object< rms< T, V > >	773
mln::Proxy< rms< T, V > >	859
mln::Accumulator< rms< T, V > >	498
base< V, rms< T, V > >	
mln::accu::rms< T, V >	464
mln::Object< rotation< n, C > >	773
mln::Function< rotation< n, C > >	654
mln::Function_v2v< rotation< n, C > >	656
mln::fun::x2x::rotation< n, C >	646
mln::Object< round< R > >	773
mln::Function< round< R > >	654
mln::Function_v2v< round< R > >	656
mln::Object< row >	773
mln::Meta_Function< row >	756
mln::Meta_Function_v2v< row >	758
mln::Object< safe_image< I > >	773

mln::Image< safe_image< I > >	698
mln::Object< saturate< V > >	773
mln::Function< saturate< V > >	654
mln::Function_v2v< saturate< V > >	656
mln::Object< saturate_rgb8 >	773
mln::Function< saturate_rgb8 >	654
mln::Function_v2v< saturate_rgb8 >	656
mln::Object< scomp< ith > >	773
mln::Meta_Function< scomp< ith > >	756
mln::Meta_Function_v2v< scomp< ith > >	758
mln::Object< set_bkd_iter< T > >	773
mln::Proxy< set_bkd_iter< T > >	859
mln::Object< set_fwd_iter< T > >	773
mln::Proxy< set_fwd_iter< T > >	859
mln::Object< sharpness< I > >	773
mln::Proxy< sharpness< I > >	859
mln::Accumulator< sharpness< I > >	498
base< double, sharpness< I > >	
mln::morpho::attribute::sharpness< I >	766
mln::Object< shell< F, I > >	773
mln::Proxy< shell< F, I > >	859
mln::Object< sign >	773
mln::Value< sign >	985
mln::Object< site_pair< P > >	773
mln::util::site_pair< P >	969
mln::Object< sli >	773
mln::Meta_Function< sli >	756
mln::Meta_Function_v2v< sli >	758
mln::Object< slice_image< I > >	773
mln::Image< slice_image< I > >	698
mln::Object< snake_fwd_t >	773
mln::Browsing< snake_fwd_t >	520
mln::canvas::browsing::snake_fwd_t	531
mln::Object< snake_generic_t >	773
mln::Browsing< snake_generic_t >	520
mln::canvas::browsing::snake_generic_t	532
mln::Object< snake_vert_t >	773
mln::Browsing< snake_vert_t >	520
mln::canvas::browsing::snake_vert_t	534
mln::Object< soft_heap< T, R > >	773
mln::util::soft_heap< T, R >	970
mln::Object< sqrt >	773
mln::Function< sqrt >	654
mln::Function_v2v< sqrt >	656
mln::Object< square >	773
mln::Mesh< square >	753
mln::Regular_Grid< square >	862
mln::Object< stack_image< n, I > >	773
mln::Image< stack_image< n, I > >	698
mln::Object< static_n_face_bkd_iter< N, D > >	773
mln::Iterator< static_n_face_bkd_iter< N, D > >	718
complex_iterator_base< topo::face< D >, static_n_face_bkd_iter< N, D > >	

complex_set_iterator_base< topo::face< D >, static_n_face_bkd_iter< N, D > >	
mln::topo::static_n_face_bkd_iter< N, D >	919
mln::Object< static_n_face_fwd_iter< N, D > >	773
mln::Iterator< static_n_face_fwd_iter< N, D > >	718
complex_iterator_base< topo::face< D >, static_n_face_fwd_iter< N, D > >	
complex_set_iterator_base< topo::face< D >, static_n_face_fwd_iter< N, D > >	
mln::topo::static_n_face_fwd_iter< N, D >	920
mln::Object< sub_image< I, S > >	773
mln::Image< sub_image< I, S > >	698
mln::Object< sub_image_if< I, S > >	773
mln::Image< sub_image_if< I, S > >	698
mln::Object< sum >	773
mln::Meta_Accumulator< sum >	754
mln::accu::meta::math::sum	438
mln::Object< sum< I, S > >	773
mln::Proxy< sum< I, S > >	859
mln::Accumulator< sum< I, S > >	498
base< S, sum< I, S > >	
mln::morpho::attribute::sum< I, S >	767
mln::Object< sum< T, S > >	773
mln::Proxy< sum< T, S > >	859
mln::Accumulator< sum< T, S > >	498
base< const S &, sum< T, S > >	
mln::accu::math::sum< T, S >	422
mln::Object< sup >	773
mln::Meta_Accumulator< sup >	754
mln::accu::meta::math::sup	439
mln::Meta_Function< sup >	756
mln::Meta_Function_vv2v< sup >	759
mln::Object< sup< T > >	773
mln::Proxy< sup< T > >	859
mln::Accumulator< sup< T > >	498
base< const T &, sup< T > >	
mln::accu::math::sup< T >	423
mln::Object< T >	773
mln::Object< tautology >	773
mln::Function< tautology >	654
mln::Function_v2v< tautology >	656
mln::Function_v2b< tautology >	656
mln::fun::p2b::tautology	613
mln::Object< teal_t >	773
mln::Literal< teal_t >	729
mln::literal::teal_t	748
mln::Object< test >	773
mln::Function< test >	654
mln::Function_v2v< test >	656
mln::Object< threshold< V > >	773
mln::Function< threshold< V > >	654
mln::Function_v2v< threshold< V > >	656
mln::Function_v2b< threshold< V > >	656
mln::fun::v2b::threshold< V >	616
mln::Object< thru_image< I, F > >	773
mln::Image< thru_image< I, F > >	698

mln::Object< thrubin_image< I1, I2, F > >	773
mln::Image< thrubin_image< I1, I2, F > >	698
mln::Object< tick >	773
mln::Mesh< tick >	753
mln::Regular_Grid< tick >	862
mln::Object< timer >	773
mln::Proxy< timer >	859
mln::util::timer	972
mln::Object< to16bits >	773
mln::Function< to16bits >	654
mln::Function_v2v< to16bits >	656
mln::Object< to19bits >	773
mln::Function< to19bits >	654
mln::Function_v2v< to19bits >	656
mln::Object< to23bits >	773
mln::Function< to23bits >	654
mln::Function_v2v< to23bits >	656
mln::Object< to27bits >	773
mln::Function< to27bits >	654
mln::Function_v2v< to27bits >	656
mln::Object< to8bits >	773
mln::Function< to8bits >	654
mln::Function_v2v< to8bits >	656
mln::Object< tofloat01 >	773
mln::Function< tofloat01 >	654
mln::Function_v2v< tofloat01 >	656
mln::Object< tr_image< S, I, T > >	773
mln::Image< tr_image< S, I, T > >	698
mln::Object< transformed_image< I, F > >	773
mln::Image< transformed_image< I, F > >	698
mln::Object< translation< n, C > >	773
mln::Function< translation< n, C > >	654
mln::Function_v2v< translation< n, C > >	656
mln::fun::x2x::translation< n, C >	649
mln::Object< translation_t< P > >	773
mln::Function< translation_t< P > >	654
mln::Function_v2v< translation_t< P > >	656
mln::Object< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >	773
mln::Proxy< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >	859
mln::Accumulator< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >	498
base< boost::tuple< BOOST_PP_REPEAT(10, RESULT_ACCU, Le Ricard ya que ca de vrai) >, tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >	
mln::accu::tuple< A, n, >	496
mln::Object< tuple< n, BOOST_PP_ENUM_PARAMS(10, T)> >	773
mln::Meta_Accumulator< tuple< n, BOOST_PP_ENUM_PARAMS(10, T)> >	754
mln::accu::meta::tuple< n, >	457
mln::Object< unary< Fun, T > >	773
mln::Function< unary< Fun, T > >	654
mln::Function_v2v< unary< Fun, T > >	656
mln::Object< unproject_image< I, D, F > >	773
mln::Image< unproject_image< I, D, F > >	698
mln::Object< up_leaf_piter< T > >	773

mln::Proxy< up_leaf_piter< T > >	859
mln::Site_Proxy< up_leaf_piter< T > >	866
mln::Site_Iterator< up_leaf_piter< T > >	865
mln::Object< up_node_piter< T > >	773
mln::Proxy< up_node_piter< T > >	859
mln::Site_Proxy< up_node_piter< T > >	866
mln::Site_Iterator< up_node_piter< T > >	865
mln::Object< up_site_piter< T > >	773
mln::Proxy< up_site_piter< T > >	859
mln::Site_Proxy< up_site_piter< T > >	866
mln::Site_Iterator< up_site_piter< T > >	865
mln::Object< val< A > >	773
mln::Proxy< val< A > >	859
mln::Accumulator< val< A > >	498
base< const A::result &, val< A > >	
mln::accu::val< A >	497
mln::Object< val< mA > >	773
mln::Meta_Accumulator< val< mA > >	754
mln::accu::meta::val< mA >	458
mln::Object< value_at_index< bool > >	773
mln::Function< value_at_index< bool > >	654
mln::Function_v2v< value_at_index< bool > >	656
mln::Object< value_at_index< T > >	773
mln::Function< value_at_index< T > >	654
mln::Function_v2v< value_at_index< T > >	656
mln::Object< var< T > >	773
mln::Proxy< var< T > >	859
mln::Accumulator< var< T > >	498
base< algebra::mat< T::dim, T::dim, float >, var< T > >	
mln::accu::stat::var< T >	492
mln::Object< variance< T, S, R > >	773
mln::Proxy< variance< T, S, R > >	859
mln::Accumulator< variance< T, S, R > >	498
base< R, variance< T, S, R > >	
mln::accu::stat::variance< T, S, R >	494
mln::Object< vec< 1, T > >	773
mln::Object< vec< 2, T > >	773
mln::Object< vec< 3, T > >	773
mln::Object< vec< 4, T > >	773
mln::Object< vec< n, C > >	773
mln::Object< vec< n, T > >	773
mln::Object< vec< V > >	773
mln::Function< vec< V > >	654
mln::Function_vv2v< vec< V > >	657
mln::fun::vv2v::vec< V >	641
mln::Object< vertex< G > >	773
mln::Site< vertex< G > >	863
mln::util::vertex< G >	981
mln::Object< vertex_bkd_iterator< G > >	773
mln::Proxy< vertex_bkd_iterator< G > >	859
mln::Object< vertex_fwd_iterator< G > >	773
mln::Proxy< vertex_fwd_iterator< G > >	859
mln::Object< vertex_image< P, V, G > >	773

mln::Image< vertex_image< P, V, G > >	698
mln::Object< vertex_nbh_edge_bkd_iterator< G > >	773
mln::Proxy< vertex_nbh_edge_bkd_iterator< G > >	859
mln::Object< vertex_nbh_edge_fwd_iterator< G > >	773
mln::Proxy< vertex_nbh_edge_fwd_iterator< G > >	859
mln::Object< vertex_nbh_vertex_bkd_iterator< G > >	773
mln::Proxy< vertex_nbh_vertex_bkd_iterator< G > >	859
mln::Object< vertex_nbh_vertex_fwd_iterator< G > >	773
mln::Proxy< vertex_nbh_vertex_fwd_iterator< G > >	859
mln::Object< violent_cast_image< T, I > >	773
mln::Image< violent_cast_image< T, I > >	698
mln::Object< violet_t >	773
mln::Literal< violet_t >	729
mln::literal::violet_t	749
mln::Object< viota_t >	773
mln::Function< viota_t >	654
mln::Function_v2v< viota_t >	656
mln::Object< viota_t< S > >	773
mln::Function< viota_t< S > >	654
mln::Function_v2v< viota_t< S > >	656
mln::Object< volume >	773
mln::Meta_Accumulator< volume >	754
mln::accu::meta::shape::volume	447
mln::Object< volume< I > >	773
mln::Proxy< volume< I > >	859
mln::Accumulator< volume< I > >	498
base< unsigned, volume< I > >	
mln::accu::shape::volume< I >	468
mln::morpho::attribute::volume< I >	769
mln::Object< W >	773
mln::Object< w_window< D, W > >	773
mln::Weighted_Window< w_window< D, W > >	1026
weighted_window_base< mln::window< D >, w_window< D, W > >	
mln::w_window< D, W >	1023
mln::Object< white_gaussian< V > >	773
mln::Function< white_gaussian< V > >	654
mln::Function_n2v< white_gaussian< V > >	655
mln::fun::n2v::white_gaussian< V >	611
mln::Object< white_t >	773
mln::Literal< white_t >	729
mln::literal::white_t	750
mln::Object< window< D > >	773
mln::Window< window< D > >	1038
window_base< D, window< D > >	
mln::window< D >	1038
mln::Object< wrap >	773
mln::Function< wrap >	654
mln::Function_v2v< wrap >	656
mln::Object< wrap< L > >	773
mln::Function< wrap< L > >	654
mln::Function_v2v< wrap< L > >	656
mln::Object< yellow_t >	773

mln::Literal< yellow_t >	729
mln::literal::yellow_t	751
mln::Object< yes >	773
mln::util::yes	985
mln::Object< zero_t >	773
mln::Literal< zero_t >	729
mln::literal::zero_t	752
mln::internal::pixel_impl_< I, bkd_pixter1d< I > >	
pixel_iterator_base_< I, bkd_pixter1d< I > >	
mln::internal::pixel_impl_< I, bkd_pixter2d< I > >	
pixel_iterator_base_< I, bkd_pixter2d< I > >	
mln::internal::pixel_impl_< I, bkd_pixter3d< I > >	
pixel_iterator_base_< I, bkd_pixter3d< I > >	
mln::internal::pixel_impl_< I, dpoints_bkd_pixter< I > >	
mln::dpoints_bkd_pixter< I >	592
mln::internal::pixel_impl_< I, dpoints_fwd_pixter< I > >	
mln::dpoints_fwd_pixter< I >	595
mln::internal::pixel_impl_< I, fwd_pixter1d< I > >	
pixel_iterator_base_< I, fwd_pixter1d< I > >	
mln::internal::pixel_impl_< I, fwd_pixter2d< I > >	
pixel_iterator_base_< I, fwd_pixter2d< I > >	
mln::internal::pixel_impl_< I, fwd_pixter3d< I > >	
pixel_iterator_base_< I, fwd_pixter3d< I > >	
mln::internal::pixel_impl_< I, pixel< I > >	
mln::pixel< I >	848
point< G, C >	1044
point< G, C >	1044
site_set_impl< Sc >	1044
subject_point_impl< P, E >	1044
subject_point_impl< point< grid::cube, C >, E >	1044
trait::graph< I >	1045
trait::graph< mln::complex_image< 1, G, V > >	1045
trait::graph< mln::image2d< T > >	1046
tree_node< T >	1046
up_leaf_piter< T >	1046
up_node_piter< T >	1046
up_site_piter< T >	1047

Chapter 7

Class Index

7.1 Class List

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

array< T >	Forward declaration	399
C_Function< E >	Concept-like	399
data< I >	Class of image internal data	399
depth1st_piter< T >	Depth1st tree traversal iterator	400
dn_leaf_piter< T >	Iterate on tree's leaves in the same way of dn_node_piter	400
dn_node_piter< T >	Iterate on tree's nodes (component representants) from leaves to roots	401
dn_site_piter< T >	Iterate on tree's sites from roots to leaves	401
face_data< N, D >	Data (adjacent faces) associated to a N-face of a D-complex	401
faces_set_mixin< N, D >	Recursive mixins of set of faces	401
graph_elt_mixed_window< G, S, S2 >	Forward declaration	402
graph_elt_window< G, S >	Forward declaration	402
graph_elt_window_if< G, S, I >	Forward declaration	402
graph_mixed_window_iter_dispatch< G, S, S2 >	Default The given site set parameter is not supported yet!	403
graph_window_if_iter_dispatch< G, S >	Default The given site set parameter is not supported yet!	403
graph_window_iter_dispatch< G, S >	Default The given site set parameter is not supported yet!	403
int_s< n >	Fwd decls	403
line_graph< G >	Fwd declaration	404
mln::accu::center< P, V >	Mass center accumulator	404
mln::accu::convolve< T1, T2, R >	Generic convolution accumulator class	405

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)	407
mln::accu::count_labels< L >	
Count the number of different labels in an image	408
mln::accu::count_value< V >	
Define an accumulator that counts the occurrence of a given value	409
mln::accu::histo< V >	
Generic histogram class over a value set with type \mathbb{V}	411
mln::accu::label_used< L >	
References all the labels used	412
mln::accu::logic::land	
"Logical-and" accumulator	413
mln::accu::logic::land_basic	
"Logical-and" accumulator	415
mln::accu::logic::lor	
"Logical-or" accumulator	416
mln::accu::logic::lor_basic	
"Logical-or" accumulator class	417
mln::accu::maj_h< T >	
Compute the majority value	418
mln::accu::math::count< T >	
Generic counter accumulator	420
mln::accu::math::inf< T >	
Generic inf accumulator class	421
mln::accu::math::sum< T, S >	
Generic sum accumulator class	422
mln::accu::math::sup< T >	
Generic sup accumulator class	423
mln::accu::max_site< I >	
Define an accumulator that computes the first site with the maximum value in an image	425
mln::accu::meta::center	
Meta accumulator for center	426
mln::accu::meta::count_adjacent_vertices	
Meta accumulator for count_adjacent_vertices	426
mln::accu::meta::count_labels	
Meta accumulator for count_labels	427
mln::accu::meta::count_value	
FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for count_value	428
mln::accu::meta::histo	
Meta accumulator for histo	429
mln::accu::meta::label_used	
Meta accumulator for label_used	430
mln::accu::meta::logic::land	
Meta accumulator for land	431
mln::accu::meta::logic::land_basic	
Meta accumulator for land_basic	432
mln::accu::meta::logic::lor	
Meta accumulator for lor	433
mln::accu::meta::logic::lor_basic	
Meta accumulator for lor_basic	434
mln::accu::meta::maj_h	
Meta accumulator for maj_h	435
mln::accu::meta::math::count	
Meta accumulator for count	436
mln::accu::meta::math::inf	
Meta accumulator for inf	437

mln::accu::meta::math::sum	
Meta accumulator for sum	438
mln::accu::meta::math::sup	
Meta accumulator for sup	439
mln::accu::meta::max_site	
Meta accumulator for max_site	440
mln::accu::meta::nil	
Meta accumulator for nil	441
mln::accu::meta::p< mA >	
Meta accumulator for p	442
mln::accu::meta::pair< A1, A2 >	
Meta accumulator for pair	443
mln::accu::meta::rms	
Meta accumulator for rms	444
mln::accu::meta::shape::bbox	
Meta accumulator for bbox	445
mln::accu::meta::shape::height	
Meta accumulator for height	446
mln::accu::meta::shape::volume	
Meta accumulator for volume	447
mln::accu::meta::stat::max	
Meta accumulator for max	448
mln::accu::meta::stat::max_h	
Meta accumulator for max	449
mln::accu::meta::stat::mean	
Meta accumulator for mean	450
mln::accu::meta::stat::median_alt< T >	
Meta accumulator for median_alt	451
mln::accu::meta::stat::median_h	
Meta accumulator for median_h	452
mln::accu::meta::stat::min	
Meta accumulator for min	453
mln::accu::meta::stat::min_h	
Meta accumulator for min	454
mln::accu::meta::stat::rank	
Meta accumulator for rank	455
mln::accu::meta::stat::rank_high_quant	
Meta accumulator for rank_high_quant	456
mln::accu::meta::tuple< n, >	
Meta accumulator for tuple	457
mln::accu::meta::val< mA >	
Meta accumulator for val	458
mln::accu::nil< T >	
Define an accumulator that does nothing	459
mln::accu::p< A >	
Generic p of accumulators	460
mln::accu::pair< A1, A2, T >	
Generic pair of accumulators	462
mln::accu::rms< T, V >	
Generic root mean square accumulator class	464
mln::accu::shape::bbox< P >	
Generic bounding box accumulator class	465
mln::accu::shape::height< I >	
Height accumulator	466
mln::accu::shape::volume< I >	
Volume accumulator class	468
mln::accu::site_set::rectangularity< P >	
Compute the rectangularity of a site set	470

mln::accu::stat::deviation< T, S, M >	
Generic standard deviation accumulator class	471
mln::accu::stat::histo3d_rgb< V >	
Define a histogram as accumulator which returns an image3d	473
mln::accu::stat::max< T >	
Generic max accumulator class	475
mln::accu::stat::max_h< V >	
Generic max function based on histogram over a value set with type \mathbb{V}	476
mln::accu::stat::mean< T, S, M >	
Generic mean accumulator class	477
mln::accu::stat::median_alt< S >	
Generic median_alt function based on histogram over a value set with type S	479
mln::accu::stat::median_h< V >	
Generic median function based on histogram over a value set with type \mathbb{V}	481
mln::accu::stat::meta::deviation	
Meta accumulator for deviation	482
mln::accu::stat::min< T >	
Generic min accumulator class	483
mln::accu::stat::min_h< V >	
Generic min function based on histogram over a value set with type \mathbb{V}	485
mln::accu::stat::min_max< V >	
Generic min and max accumulator class	486
mln::accu::stat::rank< T >	
Generic rank accumulator class	488
mln::accu::stat::rank< bool >	
Rank accumulator class for Boolean	489
mln::accu::stat::rank_high_quant< T >	
Generic rank accumulator class	491
mln::accu::stat::var< T >	
Var accumulator class	492
mln::accu::stat::variance< T, S, R >	
Variance accumulator class	494
mln::accu::tuple< A, n, >	
Generic tuple of accumulators	496
mln::accu::val< A >	
Generic val of accumulators	497
mln::Accumulator< E >	
Base class for implementation of accumulators	498
mln::algebra::h_mat< d, T >	
N-Dimensional matrix with homogeneous coordinates	500
mln::algebra::h_vec< d, C >	
N-Dimensional vector with homogeneous coordinates	502
mln::bkd_pixter1d< I >	
Backward pixel iterator on a 1-D image with border	503
mln::bkd_pixter2d< I >	
Backward pixel iterator on a 2-D image with border	505
mln::bkd_pixter3d< I >	
Backward pixel iterator on a 3-D image with border	506
mln::box< P >	
Generic box class: site set containing points of a regular grid	507
mln::Box< E >	
Base class for implementation classes of boxes	514
mln::box_runend_piter< P >	
A generic backward iterator on points by lines	518
mln::box_runstart_piter< P >	
A generic forward iterator on points by lines	519
mln::Browsing< E >	
Base class for implementation classes that are browsings	520

mln::canvas::browsing::backdiagonal2d_t	
Browsing in a certain direction	521
mln::canvas::browsing::breadth_first_search_t	
Breadth-first search algorithm for graph, on vertices	523
mln::canvas::browsing::depth_first_search_t	
Breadth-first search algorithm for graph, on vertices	523
mln::canvas::browsing::diagonal2d_t	
Browsing in a certain direction	523
mln::canvas::browsing::dir_struct_elt_incr_update_t	
Browsing in a certain direction with a segment	525
mln::canvas::browsing::directional_t	
Browsing in a certain direction	526
mln::canvas::browsing::fwd_t	
Canvas for forward browsing	528
mln::canvas::browsing::hyper_directional_t	
Browsing in a certain direction	529
mln::canvas::browsing::snake_fwd_t	
Browsing in a snake-way, forward	531
mln::canvas::browsing::snake_generic_t	
Multidimensional Browsing in a given-way	532
mln::canvas::browsing::snake_vert_t	
Browsing in a snake-way, forward	534
mln::canvas::chamfer< F >	
Compute chamfer distance	535
mln::category< R*(A) >	
Category declaration for a unary C function	535
mln::complex_image< D, G, V >	
Image based on a complex	536
mln::complex_neighborhood_bkd_piter< I, G, N >	
Backward iterator on complex neighborhood	538
mln::complex_neighborhood_fwd_piter< I, G, N >	
Forward iterator on complex neighborhood	540
mln::complex_psite< D, G >	
Point site associated to a mln::p_complex	541
mln::complex_window_bkd_piter< I, G, W >	
Backward iterator on complex window	544
mln::complex_window_fwd_piter< I, G, W >	
Forward iterator on complex window	545
mln::decorated_image< I, D >	
Image that can have additional features	547
mln::Delta_Point_Site< E >	
FIXME: Doc!	549
mln::Delta_Point_Site< void >	
Delta point site category flag type	549
mln::doc::Accumulator< E >	
Documentation class for mln::Accumulator	550
mln::doc::Box< E >	
Documentation class for mln::Box	551
mln::doc::Dpoint< E >	
Documentation class for mln::Dpoint	553
mln::doc::Fastest_Image< E >	
Documentation class for the concept of images that have the speed property set to "fastest"	555
mln::doc::Generalized_Pixel< E >	
Documentation class for mln::Generalized_Pixel	562
mln::doc::Image< E >	
Documentation class for mln::Image	564
mln::doc::Iterator< E >	
Documentation class for mln::Iterator	569

mln::doc::Neighborhood< E >	
Documentation class for mln::Neighborhood	570
mln::doc::Object< E >	
Documentation class for mln::Object	572
mln::doc::Pixel_Iterator< E >	
Documentation class for mln::Iterator	572
mln::doc::Point_Site< E >	
Documentation class for mln::Point_Site	575
mln::doc::Site_Iterator< E >	
Documentation class for mln::Site_Iterator	577
mln::doc::Site_Set< E >	
Documentation class for mln::Site_Set	579
mln::doc::Value_Iterator< E >	
Documentation class for mln::Value_Iterator	580
mln::doc::Value_Set< E >	
Documentation class for mln::Value_Set	582
mln::doc::Weighted_Window< E >	
Documentation class for mln::Weighted_Window	584
mln::doc::Window< E >	
Documentation class for mln::Window	586
mln::dpoint< G, C >	
Generic delta-point class	588
mln::Dpoint< E >	
Base class for implementation of delta-point classes	591
mln::dpoints_bkd_pixter< I >	
A generic backward iterator on the pixels of a dpoint-based window or neighborhood	592
mln::dpoints_fwd_pixter< I >	
A generic forward iterator on the pixels of a dpoint-based window or neighborhood	595
mln::dpsites_bkd_piter< V >	
A generic backward iterator on points of windows and of neighborhoods	597
mln::dpsites_fwd_piter< V >	
A generic forward iterator on points of windows and of neighborhoods	598
mln::Edge< E >	
Edge category flag type	599
mln::edge_image< P, V, G >	
Image based on graph edges	599
mln::extended< I >	
Makes an image become restricted by a point set	601
mln::extension_fun< I, F >	
Extends the domain of an image with a function	603
mln::extension_ima< I, J >	
Extends the domain of an image with an image	605
mln::extension_val< I >	
Extends the domain of an image with a value	607
mln::flat_image< T, S >	
Image with a single value	609
mln::fun::from_accu< A >	
Wrap an accumulator into a function	611
mln::fun::n2v::white_gaussian< V >	
Generate a White Gaussian Noise	611
mln::fun::p2b::antilogy	
A p2b function always returning <code>false</code>	612
mln::fun::p2b::tautology	
A p2b function always returning <code>true</code>	613
mln::fun::v2b::lnot< V >	
Functor computing logical-not on a value	614
mln::fun::v2b::threshold< V >	
Threshold function	616

mln::fun::v2v::ch_function_value< F, V >	
Wrap a function <code>v2v</code> and convert its result to another type	617
mln::fun::v2v::component< T, i >	
Functor that accesses the <code>i</code> -th component of a value	618
mln::fun::v2v::l1_norm< V, R >	
L1-norm	619
mln::fun::v2v::l2_norm< V, R >	
L2-norm	620
mln::fun::v2v::linear< V, T, R >	
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	621
mln::fun::v2v::linfty_norm< V, R >	
L-infty norm	622
mln::fun::v2v::rgb8_to_rgn< n >	
Convert a rgb8 value to a rgn, $n < 8$	623
mln::fun::v2w2v::cos< V >	
Cosinus bijective functor	624
mln::fun::v2w_w2v::l1_norm< V, R >	
L1-norm	625
mln::fun::v2w_w2v::l2_norm< V, R >	
L2-norm	626
mln::fun::v2w_w2v::linfty_norm< V, R >	
L-infty norm	628
mln::fun::vv2b::eq< L, R >	
Functor computing equal between two values	629
mln::fun::vv2b::ge< L, R >	
Functor computing "greater or equal than" between two values	629
mln::fun::vv2b::gt< L, R >	
Functor computing "greater than" between two values	630
mln::fun::vv2b::implies< L, R >	
Functor computing logical-implies between two values	631
mln::fun::vv2b::le< L, R >	
Functor computing "lower or equal than" between two values	632
mln::fun::vv2b::lt< L, R >	
Functor computing "lower than" between two values	633
mln::fun::vv2v::diff_abs< V >	
A functor computing the <code>diff_abs</code> imum of two values	634
mln::fun::vv2v::land< L, R >	
Functor computing logical-and between two values	635
mln::fun::vv2v::land_not< L, R >	
Functor computing logical and-not between two values	636
mln::fun::vv2v::lor< L, R >	
Functor computing logical-or between two values	637
mln::fun::vv2v::lxor< L, R >	
Functor computing logical-xor between two values	638
mln::fun::vv2v::max< V >	
A functor computing the maximum of two values	639
mln::fun::vv2v::min< L, R >	
A functor computing the minimum of two values	640
mln::fun::vv2v::vec< V >	
A functor computing the vecimum of two values	641
mln::fun::x2p::closest_point< P >	
FIXME: doxygen + concept checking	642
mln::fun::x2v::bilinear< I >	
Represent a bilinear interolation of values from an underlying image	643
mln::fun::x2v::trilinear< I >	
Represent a trilinear interolation of values from an underlying image	644

mln::fun::x2x::composed< T2, T1 >	644
Represent a composition of two transformations	
mln::fun::x2x::linear< I >	645
Represent a linear interolation of values from an underlying image	
mln::fun::x2x::rotation< n, C >	646
Represent a rotation function	
mln::fun::x2x::translation< n, C >	649
Translation function-object	
mln::fun_image< F, I >	652
Image read through a function	
mln::Function< E >	654
Base class for implementation of function-objects	
mln::Function< void >	654
Function category flag type	
mln::Function_n2v< E >	655
Base class for implementation of function-objects from Nil to value	
mln::Function_v2b< E >	656
Base class for implementation of function-objects from a value to a Boolean	
mln::Function_v2v< E >	656
Base class for implementation of function-objects from value to value	
mln::Function_vv2b< E >	657
Base class for implementation of function-objects from a couple of values to a Boolean	
mln::Function_vv2v< E >	657
Base class for implementation of function-objects from a couple of values to a value	
mln::fwd_pixter1d< I >	658
Forward pixel iterator on a 1-D image with border	
mln::fwd_pixter2d< I >	659
Forward pixel iterator on a 2-D image with border	
mln::fwd_pixter3d< I >	661
Forward pixel iterator on a 3-D image with border	
mln::Gdpoint< E >	662
FIXME: Doc!	
mln::Gdpoint< void >	662
Delta point site category flag type	
mln::Generalized_Pixel< E >	663
Base class for implementation classes that are pixels or that have the behavior of pixels	
mln::geom::complex_geometry< D, P >	664
A functor returning the sites of the faces of a complex where the locations of each 0-face is stored	
mln::Gpoint< E >	665
Base class for implementation of point classes	
mln::Graph< E >	669
Base class for implementation of graph classes	
mln::graph::attribute::card_t	669
Compute the cardinality of every component in a graph	
mln::graph::attribute::representative_t	670
Compute the representative vertex of every component in a graph	
mln::graph_elt_mixed_neighborhood< G, S, S2 >	671
Elementary neighborhood on graph class	
mln::graph_elt_mixed_window< G, S, S2 >	672
Elementary window on graph class	
mln::graph_elt_neighborhood< G, S >	676
Elementary neighborhood on graph class	
mln::graph_elt_neighborhood_if< G, S, I >	677
Elementary neighborhood_if on graph class	
mln::graph_elt_window< G, S >	679
Elementary window on graph class	
mln::graph_elt_window_if< G, S, I >	683
Custom window on graph class	

mln::graph_window_base< P, E >	688
mln::graph_window_if_piter< S, W, I >	
Forward iterator on line graph window	690
mln::graph_window_piter< S, W, I >	
Forward iterator on line graph window	691
mln::hexa< I >	
Hexagonal image class	694
mln::histo::array< T >	
Generic histogram class over a value set with type T	698
mln::Image< E >	
Base class for implementation of image classes	698
mln::image1d< T >	
Basic 1D image class	700
mln::image2d< T >	
Basic 2D image class	703
mln::image2d_h< V >	
2d image based on an hexagonal mesh	707
mln::image3d< T >	
Basic 3D image class	710
mln::interpolated< I, F >	
Makes the underlying image being accessed with floating coordinates	715
mln::io::dicom::dicom_header	
Store dicom file header	717
mln::io::dump::dump_header	
Store dump file header	717
mln::io::fld::fld_header	
Define the header structure of an AVS field data file	717
mln::io::raw::raw_header	
Store raw file header	717
mln::iterator< E >	
Base class for implementation classes that are iterators	718
mln::labeled_image< I >	
Morpher providing an improved interface for labeled image	720
mln::labeled_image_base< I, E >	
Base class Morpher providing an improved interface for labeled image	724
mln::lazy_image< I, F, B >	
Image values are computed on the fly	727
mln::Literal< E >	
Base class for implementation classes of literals	729
mln::literal::black_t	
Type of literal black	731
mln::literal::blue_t	
Type of literal blue	731
mln::literal::brown_t	
Type of literal brown	732
mln::literal::cyan_t	
Type of literal cyan	733
mln::literal::green_t	
Type of literal green	734
mln::literal::identity_t	
Type of literal identity	735
mln::literal::light_gray_t	
Type of literal grays	736
mln::literal::lime_t	
Type of literal lime	737
mln::literal::magenta_t	
Type of literal magenta	738

mln::literal::max_t	
Type of literal max	739
mln::literal::min_t	
Type of literal min	740
mln::literal::olive_t	
Type of literal olive	741
mln::literal::one_t	
Type of literal one	742
mln::literal::orange_t	
Type of literal orange	743
mln::literal::origin_t	
Type of literal origin	744
mln::literal::pink_t	
Type of literal pink	745
mln::literal::purple_t	
Type of literal purple	746
mln::literal::red_t	
Type of literal red	747
mln::literal::teal_t	
Type of literal teal	748
mln::literal::violet_t	
Type of literal violet	749
mln::literal::white_t	
Type of literal white	750
mln::literal::yellow_t	
Type of literal yellow	751
mln::literal::zero_t	
Type of literal zero	752
mln::Mesh< E >	
Base class for implementation classes of meshes	753
mln::Meta_Accumulator< E >	
Base class for implementation of meta accumulators	754
mln::Meta_Function< E >	
Base class for implementation of meta functions	756
mln::Meta_Function_v2v< E >	
Base class for implementation of function-objects from value to value	758
mln::Meta_Function_vv2v< E >	
Base class for implementation of function-objects from value to value	759
mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 >	
Ands type	759
mln::metal::converts_to< T, U >	
"converts-to" check	760
mln::metal::equal< T1, T2 >	
Definition of a static 'equal' test	760
mln::metal::goes_to< T, U >	
"goes-to" check	760
mln::metal::is< T, U >	
"is" check	761
mln::metal::is_a< T, M >	
"is_a" check	761
mln::metal::is_not< T, U >	
"is_not" check	761
mln::metal::is_not_a< T, M >	
"is_not_a" static Boolean expression	762
mln::morpho::attribute::card< I >	
Cardinality accumulator class	762
mln::morpho::attribute::count_adjacent_vertices< I >	
Count_Adjacent_Vertices accumulator class	763

mln::morpho::attribute::height< I >	
Height accumulator class	764
mln::morpho::attribute::sharpness< I >	
Sharpness accumulator class	766
mln::morpho::attribute::sum< I, S >	
Suminality accumulator class	767
mln::morpho::attribute::volume< I >	
Volume accumulator class	769
mln::neighb< W >	
Adapter class from window to neighborhood	770
mln::Neighborhood< E >	
Base class for implementation classes that are neighborhoods	772
mln::Neighborhood< void >	
Neighborhood category flag type	773
mln::Object< E >	
Base class for almost every class defined in Milena	773
mln::p2p_image< I, F >	
FIXME: Doc!	773
mln::p_array< P >	
Multi-set of sites	775
mln::p_centered< W >	
Site set corresponding to a window centered on a site	779
mln::p_complex< D, G >	
A complex psite set based on the N-faces of a complex of dimension D (a D-complex)	782
mln::p_edges< G, F >	
Site set mapping graph edges and image sites	785
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)	790
mln::p_graph_piter< S, I >	
Generic iterator on point sites of a mln::S	792
mln::p_if< S, F >	
Site set restricted w.r.t	794
mln::p_image< I >	
Site set based on an image of Booleans	796
mln::p_indexed_bkd_piter< S >	
Backward iterator on sites of an indexed site set	800
mln::p_indexed_fwd_piter< S >	
Forward iterator on sites of an indexed site set	801
mln::p_indexed_psite< S >	
Psite class for indexed site sets such as p_array	802
mln::p_key< K, P >	
Priority queue class	802
mln::p_line2d	
2D discrete line of points	807
mln::p_mutable_array_of< S >	
P_mutable_array_of is a mutable array of site sets	810
mln::p_n_faces_bkd_piter< D, G >	
Backward iterator on the n-faces sites of an mln::p_complex<D, G>	813
mln::p_n_faces_fwd_piter< D, G >	
Forward iterator on the n-faces sites of an mln::p_complex<D, G>	814
mln::p_priority< P, Q >	
Priority queue	815
mln::p_queue< P >	
Queue of sites (based on std::deque)	820
mln::p_queue_fast< P >	
Queue of sites class (based on p_array	824
mln::p_run< P >	
Point set class in run	828

mln::p_set< P >	Mathematical set of sites (based on util::set)	832
mln::p_transformed< S, F >	Site set transformed through a function	835
mln::p_transformed_piter< Pi, S, F >	Iterator on p_transformed<S,F>	838
mln::p_vaccess< V, S >	Site set in which sites are grouped by their associated value	839
mln::p_vertices< G, F >	Site set based mapping graph vertices to sites	843
mln::pixel< I >	Generic pixel class	848
mln::plain< I >	Prevents an image from sharing its data	849
mln::Point< P >	Base class for implementation of point classes	851
mln::point< G, C >	Generic point class	853
mln::Proxy< E >	Base class for implementation classes of the notion of "proxy"	859
mln::Proxy< void >	Proxy category flag type	859
mln::Pseudo_Site< E >	Base class for implementation classes of the notion of "pseudo site"	859
mln::Pseudo_Site< void >	Pseudo_Site category flag type	860
mln::pw::image< F, S >	A generic point-wise image implementation	860
mln::registration::closest_point_basic< P >	Closest point functor based on map distance	861
mln::registration::closest_point_with_map< P >	Closest point functor based on map distance	862
mln::Regular_Grid< E >	Base class for implementation classes of regular grids	862
mln::safe_image< I >	Makes an image accessible at undefined location	862
mln::select::p_of< P >	Structure p_of	863
mln::Site< E >	Base class for classes that are explicitly sites	863
mln::Site< void >	Site category flag type	864
mln::Site_Iterator< E >	Base class for implementation of classes of iterator on points	865
mln::Site_Proxy< E >	Base class for implementation classes of the notion of "site proxy"	866
mln::Site_Proxy< void >	Site_Proxy category flag type	867
mln::Site_Set< E >	Base class for implementation classes of site sets	867
mln::Site_Set< void >	Site_Set category flag type	871
mln::sub_image< I, S >	Image having its domain restricted by a site set	871
mln::sub_image_if< I, S >	Image having its domain restricted by a site set and a function	872
mln::thru_image< I, F >	Morph image values through a function	873

mln::thrubin_image< I1, I2, F >	
Morphes values from two images through a binary function	874
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>	875
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>	876
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>	877
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>	878
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>	879
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>	880
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>	881
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>	882
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>	883
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>	884
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	885
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	887
mln::topo::algebraic_face< D >	
Algebraic face handle in a complex; the face dimension is dynamic	888
mln::topo::algebraic_n_face< N, D >	
Algebraic N-face handle in a complex	892
mln::topo::center_only_iter< D >	
Iterator on all the adjacent (n-1)-faces of the n-face of an mln::complex<D>	896
mln::topo::centered_bkd_iter_adapter< D, I >	
Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces	897
mln::topo::centered_fwd_iter_adapter< D, I >	
Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces	898
mln::topo::complex< D >	
General complex of dimension D	899
mln::topo::face< D >	
Face handle in a complex; the face dimension is dynamic	902
mln::topo::face_bkd_iter< D >	
Backward iterator on all the faces of an mln::complex<D>	905
mln::topo::face_fwd_iter< D >	
Forward iterator on all the faces of an mln::complex<D>	907
mln::topo::is_n_face< N >	
A functor testing wheter a mln::complex_psite is an N -face	908
mln::topo::is_simple_2d_t< N >	
Test if a point is simple or not	909
mln::topo::is_simple_cell< I >	
A predicate for the simplicity of a point based on the collapse property of the attachment	909

mln::topo::n_face< N, D >	
N-face handle in a complex	912
mln::topo::n_face_bkd_iter< D >	
Backward iterator on all the faces of an mln::complex<D>	915
mln::topo::n_face_fwd_iter< D >	
Forward iterator on all the faces of an mln::complex<D>	916
mln::topo::n_faces_set< N, D >	
Set of face handles of dimension N	917
mln::topo::skeleton::is_simple_point< N >	918
mln::topo::static_n_face_bkd_iter< N, D >	
Backward iterator on all the N-faces of a mln::complex<D>	919
mln::topo::static_n_face_fwd_iter< N, D >	
Forward iterator on all the N-faces of a mln::complex<D>	920
mln::tr_image< S, I, T >	
Transform an image by a given transformation	921
mln::transformed_image< I, F >	
Image having its domain restricted by a site set	924
mln::unproject_image< I, D, F >	
Un-projects an image	926
mln::util::adjacency_matrix< V >	
A class of adjacency matrix	927
mln::util::array< T >	
A dynamic array class	928
mln::util::branch< T >	
Class of generic branch	933
mln::util::branch_iter< T >	
Basic 2D image class	934
mln::util::branch_iter_ind< T >	
Basic 2D image class	936
mln::util::couple< T, U >	
Definition of a couple	937
mln::util::eat	
Eat structure	939
mln::util::edge< G >	
Edge of a graph G	939
mln::util::fibonacci_heap< P, T >	
Fibonacci heap	943
mln::util::graph	
Undirected graph	945
mln::util::greater_point< I >	
A "greater than" functor comparing points w.r.t	951
mln::util::greater_psite< I >	
A "greater than" functor comparing psites w.r.t	951
mln::util::head< T, R >	
Top structure of the soft heap	952
mln::util::ignore	
Ignore structure	952
mln::util::ilcell< T >	
Element of an item list. Store the data (key) used in soft_heap	953
mln::util::line_graph< G >	
Undirected line graph of a graph of type G	953
mln::util::nil	
Nil structure	958
mln::util::node< T, R >	
Meta-data of an element in the heap	958
mln::util::object_id< Tag, V >	
Base class of an object id	958

mln::util::ord< T >	Function-object that defines an ordering between objects with type T : <i>lhs R rhs</i>	960
mln::util::ord_pair< T >	Ordered pair structure s.a	960
mln::util::pix< I >	Structure pix	962
mln::util::set< T >	An "efficient" mathematical set class	964
mln::util::site_pair< P >	A pair of sites	969
mln::util::soft_heap< T, R >	Soft heap	970
mln::util::timer	Timer structure	972
mln::util::tracked_ptr< T >	Smart pointer for shared data with tracking	973
mln::util::tree< T >	Class of generic tree	975
mln::util::tree_node< T >	Class of generic tree_node for tree	977
mln::util::vertex< G >	Vertex of a graph G	981
mln::util::yes	Object that always says "yes"	985
mln::Value< E >	Base class for implementation classes of values	985
mln::value::float01	Class for floating values restricted to the interval $[0..1]$ and discretized with n bits	987
mln::value::float01_f	Class for floating values restricted to the interval $[0..1]$	989
mln::value::graylevel< n >	General gray-level class on n bits	990
mln::value::graylevel_f	General gray-level class on n bits	993
mln::value::int_s< n >	Signed integer value class	995
mln::value::int_u< n >	Unsigned integer value class	997
mln::value::int_u_sat< n >	Unsigned integer value class with saturation behavior	999
mln::value::Integer< E >	Concept of integer	1002
mln::value::Integer< void >	Category flag type	1002
mln::value::label< n >	Label value class	1002
mln::value::lut_vec< S, T >	Class that defines FIXME	1005
mln::value::proxy< I >	Generic proxy class for an image pixel value	1007
mln::value::qt::rgb32	Color class for red-green-blue where every component is n -bit encoded	1009
mln::value::rgb< n >	Color class for red-green-blue where every component is n -bit encoded	1011
mln::value::set< T >	Class that defines the set of values of type T	1012
mln::value::sign	Value type composed by the set $\{-1, 0, 1\}$ sign value type is a subset of the int value type . . .	1013

mln::value::stack_image< n, I >	
Stack image class	1015
mln::value::super_value< sign >	
Specializations:	1017
mln::value::value_array< T, V >	
Generic array class over indexed by a value set with type T	1018
mln::Vertex< E >	
Vertex category flag type	1019
mln::vertex_image< P, V, G >	
Image based on graph vertices	1019
mln::violent_cast_image< T, I >	
Violently cast image values to a given type	1021
mln::w_window< D, W >	
Generic w_window class	1023
mln::Weighted_Window< E >	
Base class for implementation classes that are weighted_windows	1026
mln::win::backdiag2d	
Diagonal line window defined on the 2D square grid	1027
mln::win::ball< G, C >	
Generic ball window defined on a given grid	1028
mln::win::cube3d	
Cube window defined on the 3D grid	1029
mln::win::cuboid3d	
Cuboid defined on the 3-D square grid	1030
mln::win::diag2d	
Diagonal line window defined on the 2D square grid	1032
mln::win::line< M, i, C >	
Generic line window defined on a given grid in the given dimension	1033
mln::win::multiple< W, F >	
Multiple window	1034
mln::win::multiple_size< n, W, F >	
Definition of a multiple-size window	1035
mln::win::octagon2d	
Octagon window defined on the 2D square grid	1035
mln::win::rectangle2d	
Rectangular window defined on the 2D square grid	1036
mln::Window< E >	
Base class for implementation classes that are windows	1038
mln::window< D >	
Generic window class	1038
mln::world::inter_pixel::is_separator	
Functor returning whether a site is a separator in an inter-pixel image	1042
point< G, C >	
Forward declarations	1044
point< G, C >	
Forward declarations	1044
site_set_impl< Sc >	
The facade	1044
subject_point_impl< P, E >	
Subject_impl specialization (Proxy)	1044
trait::graph< I >	
Graph traits	1045
trait::graph< mln::complex_image< 1, G, V > >	
Graph traits for 1-complexes images	1045
trait::graph< mln::image2d< T > >	
Graph traits for mln::image2d	1046
tree_node< T >	
Fwd declarations	1046

up_leaf_piter< T >	
Iterate on tree's leaves in the same way of up_node_piter	1046
up_node_piter< T >	
Iterate on tree's nodes (component representants) from leaves to roots	1046
up_site_piter< T >	
Iterate on tree's sites from leaves to roots	1047

Chapter 8

Module Documentation

8.1 On site sets

Accumulators working on site sets.

Classes

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

8.1.1 Detailed Description

Accumulators working on site sets.

8.2 On images

Accumulators working on images.

Classes

- struct `mln::accu::count_adjacent_vertices< F, S >`
Accumulator class counting the number of vertices adjacent to a set of `mln::p_edges_psite` (i.e., a set of edges).
- struct `mln::accu::max_site< I >`
Define an accumulator that computes the first site with the maximum value in an image.
- struct `mln::accu::shape::height< I >`
Height accumulator.
- struct `mln::accu::shape::volume< I >`
Volume accumulator class.

8.2.1 Detailed Description

Accumulators working on images.

8.3 On values

Accumulators working on image values.

Classes

- struct `mln::accu::convolve< T1, T2, R >`
Generic convolution accumulator class.
- struct `mln::accu::count_labels< L >`
Count the number of different labels in an image.
- struct `mln::accu::count_value< V >`
Define an accumulator that counts the occurrence of a given value.
- struct `mln::accu::histo< V >`
Generic histogram class over a value set with type V .
- struct `mln::accu::label_used< L >`
References all the labels used.
- struct `mln::accu::logic::land`
"Logical-and" accumulator.
- struct `mln::accu::logic::land_basic`
"Logical-and" accumulator.
- struct `mln::accu::logic::lor`
"Logical-or" accumulator.
- struct `mln::accu::logic::lor_basic`
"Logical-or" accumulator class.
- struct `mln::accu::maj_h< T >`
Compute the majority value.
- struct `mln::accu::math::inf< T >`
Generic inf accumulator class.
- struct `mln::accu::math::sum< T, S >`
Generic sum accumulator class.
- struct `mln::accu::math::sup< T >`
Generic sup accumulator class.
- struct `mln::accu::rms< T, V >`
Generic root mean square accumulator class.
- struct `mln::accu::stat::deviation< T, S, M >`
Generic standard deviation accumulator class.
- struct `mln::accu::stat::histo3d_rgb< V >`
*Define a histogram as accumulator which returns an *image3d*.*
- struct `mln::accu::stat::max< T >`
Generic max accumulator class.
- struct `mln::accu::stat::max_h< V >`
Generic max function based on histogram over a value set with type V .
- struct `mln::accu::stat::mean< T, S, M >`
Generic mean accumulator class.
- struct `mln::accu::stat::median_alt< S >`
*Generic *median_alt* function based on histogram over a value set with type S .*
- struct `mln::accu::stat::median_h< V >`
Generic median function based on histogram over a value set with type V .
- struct `mln::accu::stat::min< T >`
Generic min accumulator class.

- struct `mln::accu::stat::min_h< V >`
Generic min function based on histogram over a value set with type V.
- struct `mln::accu::stat::min_max< V >`
Generic min and max accumulator class.
- struct `mln::accu::stat::rank< T >`
Generic rank accumulator class.
- struct `mln::accu::stat::rank< bool >`
rank accumulator class for Boolean.
- struct `mln::accu::stat::rank_high_quant< T >`
Generic rank accumulator class.
- struct `mln::accu::stat::var< T >`
Var accumulator class.
- struct `mln::accu::stat::variance< T, S, R >`
Variance accumulator class.

8.3.1 Detailed Description

Accumulators working on image values.

8.4 Multiple accumulators

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

Classes

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

8.4.1 Detailed Description

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

8.5 Graphes

All graphes implementations.

Classes

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

8.5.1 Detailed Description

All graphes implementations.

8.6 Images

All the generic image types provided in Olena.

Modules

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

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.

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::p2p_image< I, F >`
FIXME: Doc!
- class `mln::sub_image< I, S >`
Image having its domain restricted by a site set.
- struct `mln::sub_image_if< I, S >`
Image having its domain restricted by a site set and a function.
- struct `mln::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.

All algorithms/routines provided in Milena.

8.15 Canvas

All canvas.

All canvas.

8.16 Functions

All predefined functions.

Modules

- [v2w2v functions](#)
All bijective functions.
- [v2w_w2v functions](#)
All bijective function.
- [vv2b functions](#)
All functions mapping two values to a logical value.

Namespaces

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

Classes

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

8.16.1 Detailed Description

All predefined functions.

8.17 Neighborhoods

All the predefined generic neighborhoods.

Modules

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

8.17.1 Detailed Description

All the predefined generic neighborhoods.

8.18 1D neighborhoods

Predefined 1D neighborhoods.

Typedefs

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

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

Functions

- `const neighb1d & mln::c2 ()`

2-connectivity neighborhood on the 1D grid.

8.18.1 Detailed Description

Predefined 1D neighborhoods.

8.18.2 Typedef Documentation

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

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

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

8.18.3 Function Documentation

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

2-connectivity neighborhood on the 1D grid.

○ x ○

Returns

A `neighb1d`.

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

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.

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

8.19.3 Function Documentation

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

Vertical 2-connectivity neighborhood on the 2D grid.

```

- o -
- x -
- o -

```

Returns

A `neighb2d`.

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

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

Horizontal 2-connectivity neighborhood on the 2D grid.

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

Returns

A neighb2d.

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

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

4-connectivity neighborhood on the 2D grid.

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

Returns

A neighb2d.

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

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

8-connectivity neighborhood on the 2D grid.

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

Returns

A neighb2d.

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

8.20 3D neighborhoods

Predefined 3D neighborhoods.

Typedefs

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

Functions

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

8.20.1 Detailed Description

Predefined 3D neighborhoods.

8.20.2 Typedef Documentation

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

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

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

8.20.3 Function Documentation

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

18-connectivity neighborhood on the 3D grid.

```

      . o .
    o o o
  . o .

      o o o
    o x o
  o o o

      . o .
    o o o
  . o .

```


Returns

A neighb3d.

Definition at line 288 of file neighb3d.hh.

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

Referenced by mln::c26().

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

26-connectivity neighborhood on the 3D grid.

```

  o o o
  o o o
o o o

  o o o
  o x o
o o o

  o o o
  o o o
o o o

```

Returns

A neighb3d.

Definition at line 309 of file neighb3d.hh.

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

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

depth 2-connectivity neighborhood on the 3D grid.

```

  . . .
  . o .
. . .

  . . .
  . x .
. . .

  . . .
  . o .
. . .

```

Returns

A neighb3d.

Definition at line 226 of file neighb3d.hh.

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

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

4-connectivity neighborhood on the 3D grid.

```

. . .
. . .
. . .

```

```

. o .
o x o
. o .

```

```

. . .
. . .
. . .

```

Returns

A neighb3d.

Definition at line 241 of file neighb3d.hh.

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

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

6-connectivity neighborhood on the 3D grid.

```

. . .
. o .
. . .

```

```

. o .
o x o
. o .

```

```

. . .
. o .
. . .

```

Returns

A neighb3d.

Definition at line 271 of file neighb3d.hh.

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

Referenced by `mln::c18()`.

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

8-connectivity neighborhood on the 3D grid.

```

. . .
. . .
. . .

```

```

o o o
o x o
o o o

```

```

. . .
. . .
. . .

```

Returns

A neighb3d.

Definition at line 257 of file neighb3d.hh.

8.21 Site sets

All Site set types.

Modules

- [Basic types](#)

Basic site sets.

- [Graph based](#)

Site sets based on a graph.

- [Complex based](#)

Site sets based on a complexes.

- [Sparse types](#)

Sparse site sets.

- [Queue based](#)

Site sets based on a queue.

8.21.1 Detailed Description

All Site set types.

8.22 Basic types

Basic site sets.

Classes

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

8.22.1 Detailed Description

Basic site sets.

8.23 Graph based

Site sets based on a graph.

Classes

- class `mln::p_edges< G, F >`
Site set mapping graph edges and image sites.
- struct `mln::p_faces< N, D, P >`
A complex psite set based on 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`.

Definition at line 56 of file `segment1d.hh`.

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

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

Definition at line 47 of file `window1d.hh`.

8.30 2D windows

Predefined 2D windows.

Classes

- struct `mln::win::backdiag2d`
Diagonal line window defined on the 2D square grid.
- struct `mln::win::diag2d`
Diagonal line window defined on the 2D square grid.
- struct `mln::win::octagon2d`
Octagon window defined on the 2D square grid.
- struct `mln::win::rectangle2d`
Rectangular window defined on the 2D square grid.

Typedefs

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

Functions

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

8.30.1 Detailed Description

Predefined 2D windows.

8.30.2 Typedef Documentation

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

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

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

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

Horizontal line window defined on the 2D square grid.

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

For instance:

```
○ ○ x ○ ○
```

is defined with length = 5.

Definition at line 57 of file hline2d.hh.

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

Vertical line window defined on the 2D square grid.

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

For instance:

```
○
x
○
```

is defined with length = 3.

Definition at line 58 of file vline2d.hh.

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

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

Definition at line 49 of file window2d.hh.

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

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

```
- ○ -
○ x ○
- ○ -
```

Returns

A window2d.

Definition at line 105 of file window2d.hh.

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

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

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

```
○ ○ ○
○ x ○
○ ○ ○
```

Returns

A window2d.

Definition at line 121 of file window2d.hh.

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

8.31 3D windows

Predefined 3D windows.

Classes

- struct `mln::win::cube3d`
Cube window defined on the 3D grid.
- struct `mln::win::cuboid3d`
Cuboid defined on the 3-D square grid.

Typedefs

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

Functions

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

8.31.1 Detailed Description

Predefined 3D windows.

8.31.2 Typedef Documentation

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

Depth line window defined on the 3D cubic grid.

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

For instance:

```

. . .
. o .
. . .

. . .
. x .
. . .

. . .
. o .
. . .

```

is defined with length = 3.

Definition at line 68 of file sline3d.hh.

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

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

Definition at line 47 of file sphere3d.hh.

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

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

Definition at line 48 of file window3d.hh.

8.31.3 Function Documentation

8.31.3.1 `const window3d & mln::win_c4p_3d () [inline]`

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

```

  - - -
  - - -
  - - -

  - o -
  o x o
  - o -

  - - -
  - - -
  - - -

```

Returns

A window3d.

Definition at line 119 of file window3d.hh.

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

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

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

```

  - - -
  - - -
  - - -

  o o o
  o x o
  o o o

  - - -
  - - -
  - - -

```


Returns

A window3d.

Definition at line 135 of file window3d.hh.

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

8.32 N-D windows

Predefined N-D windows.

Classes

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

8.32.1 Detailed Description

Predefined N-D windows.

8.33 Multiple windows

Generic multiple windows.

Classes

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

8.33.1 Detailed Description

Generic multiple windows.

8.34 v2w2v functions

All bijective functions.

All bijective functions.

8.35 v2w_w2v functions

All bijective function.

All bijective function.

8.36 vv2b functions

All functions mapping two values to a logical value.

All functions mapping two values to a logical value.

Chapter 9

Namespace Documentation

9.1 mIn Namespace Reference

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

Namespaces

- namespace [accu](#)
Namespace of accumulators.
- namespace [algebra](#)
Namespace of algebraic structure.
- namespace [arith](#)
Namespace of arithmetic.
- namespace [binarization](#)
Namespace of "point-wise" expression tools.
- namespace [border](#)
Namespace of routines related to image virtual (outer) border.
- namespace [canvas](#)
Namespace of canvas.
- namespace [convert](#)
Namespace of conversion routines.
- namespace [data](#)
Namespace of image processing routines related to pixel data.
- namespace [debug](#)
Namespace of routines that help to debug.
- namespace [def](#)
Namespace for core definitions.
- namespace [display](#)
Namespace of routines that help to display images.
- namespace [doc](#)
The namespace [mIn::doc](#) is only for documentation purpose.
- namespace [draw](#)
Namespace of drawing routines.
- namespace [estim](#)
Namespace of estimation materials.
- namespace [extension](#)
Namespace of extension tools.

- namespace [fun](#)
Namespace of functions.
- namespace [geom](#)
Namespace of all things related to geometry.
- namespace [graph](#)
Namespace of graph related routines.
- namespace [grid](#)
Namespace of grids definitions.
- namespace [histo](#)
Namespace of histograms.
- namespace [impl](#)
Implementation namespace of mln namespace.
- namespace [io](#)
Namespace of input/output handling.
- namespace [labeling](#)
Namespace of labeling routines.
- namespace [linear](#)
Namespace of linear image processing routines.
- namespace [literal](#)
Namespace of literals.
- namespace [logical](#)
Namespace of logic.
- namespace [make](#)
Namespace of routines that help to make Milena's objects.
- namespace [math](#)
Namespace of mathematical routines.
- namespace [metal](#)
Namespace of meta-programming tools.
- namespace [morpho](#)
Namespace of mathematical morphology routines.
- namespace [norm](#)
Namespace of norms.
- namespace [opt](#)
Namespace of optional routines.
- namespace [pw](#)
Namespace of "point-wise" expression tools.
- namespace [registration](#)
Namespace of "point-wise" expression tools.
- namespace [select](#)
Select namespace (FIXME doc).
- namespace [set](#)
Namespace of image processing routines related to pixel sets.
- namespace [subsampling](#)
Namespace of "point-wise" expression tools.
- namespace [tag](#)
Namespace of image processing routines related to tags.
- namespace [test](#)
Namespace of image processing routines related to pixel tests.
- namespace [topo](#)
Namespace of "point-wise" expression tools.
- namespace [trace](#)

Namespace of routines related to the trace mechanism.

- namespace [trait](#)

Namespace where traits are defined.

- namespace [transform](#)

Namespace of transforms.

- namespace [util](#)

Namespace of tools using for more complex algorithm.

- namespace [value](#)

Namespace of materials related to pixel value types.

- namespace [win](#)

Namespace of image processing routines related to win.

Classes

- struct [Accumulator](#)

Base class for implementation of accumulators.

- class [bkd_pixter1d](#)

Backward pixel iterator on a 1-D image with border.

- class [bkd_pixter2d](#)

Backward pixel iterator on a 2-D image with border.

- class [bkd_pixter3d](#)

Backward pixel iterator on a 3-D image with border.

- class [box](#)

Generic box class: site set containing points of a regular grid.

- struct [Box](#)

Base class for implementation classes of boxes.

- class [box_runend_piter](#)

A generic backward iterator on points by lines.

- class [box_runstart_piter](#)

A generic forward iterator on points by lines.

- struct [Browsing](#)

Base class for implementation classes that are browsings.

- struct [category< R\(*\) \(A\) >](#)

Category declaration for a unary C function.

- class [complex_image](#)

Image based on a complex.

- class [complex_neighborhood_bkd_piter](#)

Backward iterator on complex neighborhood.

- class [complex_neighborhood_fwd_piter](#)

Forward iterator on complex neighborhood.

- class [complex_psite](#)

Point site associated to a `mln::p_complex`.

- class [complex_window_bkd_piter](#)

Backward iterator on complex window.

- class [complex_window_fwd_piter](#)

Forward iterator on complex window.

- struct [decorated_image](#)

Image that can have additional features.

- struct [Delta_Point_Site](#)

FIXME: Doc!

- struct [Delta_Point_Site< void >](#)
Delta point site category flag type.
- struct [dpoint](#)
Generic delta-point class.
- struct [Dpoint](#)
Base class for implementation of delta-point classes.
- class [dpoints_bkd_pixter](#)
A generic backward iterator on the pixels of a dpoint-based window or neighborhood.
- class [dpoints_fwd_pixter](#)
A generic forward iterator on the pixels of a dpoint-based window or neighborhood.
- class [dpsites_bkd_piter](#)
A generic backward iterator on points of windows and of neighborhoods.
- class [dpsites_fwd_piter](#)
A generic forward iterator on points of windows and of neighborhoods.
- struct [Edge](#)
edge category flag type.
- class [edge_image](#)
[Image](#) based on graph edges.
- struct [extended](#)
Makes an image become restricted by a point set.
- class [extension_fun](#)
Extends the domain of an image with a function.
- class [extension_ima](#)
Extends the domain of an image with an image.
- class [extension_val](#)
Extends the domain of an image with a value.
- struct [flat_image](#)
[Image](#) with a single value.
- struct [fun_image](#)
[Image](#) read through a function.
- struct [Function](#)
Base class for implementation of function-objects.
- struct [Function< void >](#)
[Function](#) category flag type.
- struct [Function_n2v](#)
Base class for implementation of function-objects from Nil to value.
- struct [Function_v2b](#)
Base class for implementation of function-objects from a value to a Boolean.
- struct [Function_v2v](#)
Base class for implementation of function-objects from value to value.
- struct [Function_vv2b](#)
Base class for implementation of function-objects from a couple of values to a Boolean.
- struct [Function_vv2v](#)
Base class for implementation of function-objects from a couple of values to a value.
- class [fwd_pixter1d](#)
Forward pixel iterator on a 1-D image with border.
- class [fwd_pixter2d](#)
Forward pixel iterator on a 2-D image with border.
- class [fwd_pixter3d](#)
Forward pixel iterator on a 3-D image with border.
- struct [Gdpoint](#)

- FIXME: Doc!*
- struct [Gdpoint< void >](#)
Delta point site category flag type.
 - struct [Generalized_Pixel](#)
Base class for implementation classes that are pixels or that have the behavior of pixels.
 - struct [Gpoint](#)
Base class for implementation of point classes.
 - struct [Graph](#)
Base class for implementation of graph classes.
 - struct [graph_elt_mixed_neighborhood](#)
Elementary neighborhood on graph class.
 - class [graph_elt_mixed_window](#)
Elementary window on graph class.
 - struct [graph_elt_neighborhood](#)
Elementary neighborhood on graph class.
 - struct [graph_elt_neighborhood_if](#)
Elementary neighborhood_if on graph class.
 - class [graph_elt_window](#)
Elementary window on graph class.
 - class [graph_elt_window_if](#)
Custom window on graph class.
 - class [graph_window_base](#)
 - class [graph_window_if_piter](#)
Forward iterator on line graph window.
 - class [graph_window_piter](#)
Forward iterator on line graph window.
 - struct [hexa](#)
hexagonal image class.
 - struct [Image](#)
Base class for implementation of image classes.
 - struct [image1d](#)
Basic 1D image class.
 - class [image2d](#)
Basic 2D image class.
 - struct [image2d_h](#)
2d image based on an hexagonal mesh.
 - struct [image3d](#)
Basic 3D image class.
 - struct [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 [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 the N-faces of a complex of dimension D (a D -complex).
 - class [p_graph_piter](#)

Generic iterator on point sites of a $mln::S$.
 - class [p_if](#)

[Site](#) set restricted w.r.t.
 - class [p_image](#)

[Site](#) set based on an image of Booleans.
 - class [p_indexed_bkd_piter](#)

Backward iterator on sites of an indexed site set.
 - class [p_indexed_fwd_piter](#)

Forward iterator on sites of an indexed site set.
 - class [p_indexed_psite](#)

Psite class for indexed site sets such as [p_array](#).
 - class [p_key](#)

Priority queue class.
 - class [p_line2d](#)

2D discrete line of points.
 - class [p_mutable_array_of](#)

[p_mutable_array_of](#) is a mutable array of site sets.
 - class [p_n_faces_bkd_piter](#)

Backward iterator on the n -faces sites of an $mln::p_complex<D, G>$.
 - class [p_n_faces_fwd_piter](#)

Forward iterator on the n -faces sites of an $mln::p_complex<D, G>$.

- class [p_priority](#)
Priority queue.
- class [p_queue](#)
Queue of sites (based on `std::deque`).
- class [p_queue_fast](#)
Queue of sites class (based on [p_array](#)).
- class [p_run](#)
Point set class in run.
- class [p_set](#)
Mathematical set of sites (based on `util::set`).
- class [p_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.
- class [plain](#)
Prevents an image from sharing its data.
- struct [point](#)
Generic point class.
- struct [Point](#)
Base class for implementation of point classes.
- 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.
- class `sub_image`
Image having its domain restricted by a site set.
- struct `sub_image_if`
Image having its domain restricted by a site set and a function.
- class `thru_image`
Morph image values through a function.
- class `thrubin_image`
Morphes values from two images through a binary function.
- struct `tr_image`
Transform an image by a given transformation.
- struct `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 `Vertex`
Vertex category flag type.
- class `vertex_image`
Image based on graph vertices.
- struct `violent_cast_image`
Violently cast image values to a given type.
- struct `w_window`
Generic w_window class.
- struct `Weighted_Window`
Base class for implementation classes that are *weighted_windows*.
- class `window`
Generic window class.
- struct `Window`
Base class for implementation classes that are *windows*.

Typedefs

- typedef `mln::complex_image`
`< 1, mln::discrete_plane_1complex_geometry,`
`bool > bin_1complex_image2d`
Type alias for a binary image based on a 1-complex, where 0-faces are located at discrete (integer) 2-dimensional points.
- typedef `mln::complex_image`
`< 2, mln::space_2complex_geometry,`
`bool > bin_2complex_image3df`
Type alias for a binary image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.
- typedef `box< mln::point1d > box1d`
Type alias for a box defined on the 1D square grid with integer coordinates.
- typedef `box< mln::point2d > box2d`
Type alias for a box defined on the 2D square grid with integer coordinates.
- typedef `box< point2d_h > box2d_h`
FIXME.
- typedef `box< point3d > box3d`
Type alias for a box defined on the 3D square grid with integer coordinates.

- typedef `mln::geom::complex_geometry`
`< 1, point2d > discrete_plane_1complex_geometry`
Type alias for the geometry of a 1-complex (e.g., a graph) located in a discrete 2-dimensional plane (with integer coordinates).
- typedef `mln::geom::complex_geometry`
`< 2, point2d > discrete_plane_2complex_geometry`
Type alias for the geometry of a 2-complex located in a discrete 2-dimensional plane (with integer coordinates).
- typedef `dpoint`
`< mln::grid::tick, def::coord > dpoint1d`
Type alias for a delta-point defined on the 1D square grid with integer coordinates.
- typedef `dpoint`
`< mln::grid::square, mln::def::coord > dpoint2d`
Type alias for a delta-point defined on the 2D square grid with integer coordinates.
- typedef `dpoint`
`< mln::grid::hexa, def::coord > dpoint2d_h`
Type alias for a delta-point defined on the 2D square grid with integer coordinates.
- typedef `dpoint`
`< mln::grid::cube, def::coord > dpoint3d`
Type alias for a delta-point defined on the 3D square grid with integer coordinates.
- typedef `mln::complex_image`
`< 2, mln::space_2complex_geometry, float > float_2complex_image3df`
Type alias for a floating-point image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.
- typedef `mln::complex_image`
`< 1, mln::discrete_plane_1complex_geometry, mln::value::int_u8 > int_u8_1complex_image2d`
Type alias for an 8-bit gray-level image based on a 1-complex, where 0-faces are located at discrete (integer) 2-dimensional points.
- typedef `mln::complex_image`
`< 2, mln::discrete_plane_2complex_geometry, mln::value::int_u8 > int_u8_2complex_image2d`
Type alias for an 8-bit gray-level image based on a 2-complex, where 0-faces are located at discrete (integer) 2-dimensional points.
- typedef `mln::complex_image`
`< 2, mln::space_2complex_geometry, mln::value::int_u8 > int_u8_2complex_image3df`
Type alias for an 8-bit gray-level image based on a 2-complex, where 0-faces are located at floating-point 3-dimensional points.
- typedef `neighb< window1d > neighb1d`
Type alias for a neighborhood defined on the 1D square grid with integer coordinates.
- typedef `neighb< window2d > neighb2d`
Type alias for a neighborhood defined on the 2D square grid with integer coordinates.
- typedef `neighb< window3d > neighb3d`
Type alias for a neighborhood defined on the 3D square grid with integer coordinates.
- typedef `p_run< point2d > p_run2d`
Type alias for a run of 2d points.
- typedef `p_set_of< p_run2d > p_runs2d`
Type alias for a set of runs of 2d points.
- typedef `point< grid::tick, def::coord > 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 window``< mln::dpoint1d > window1d`

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

- `typedef window``< mln::dpoint2d > window2d`

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

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

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

- `typedef point``< grid::tick,`
`def::coord > point1d`

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

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

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

- `typedef point``< grid::hexa,`
`def::coord > point2d_h`

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

- `typedef point``< grid::cube,`
`def::coord > point3d`

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

Functions

- template<typename I >
I::psite [a_point_of](#) (const [Image](#)< I > &ima)
Give a point of an image.
- template<typename I , typename F >
[p2p_image](#)< I, F > [apply_p2p](#) ([Image](#)< I > &ima, const [Function_v2v](#)< F > &f)
FIXME: Doc!
- template<typename I , typename F >
[p2p_image](#)< const I, F > [apply_p2p](#) (const [Image](#)< I > &ima, const [Function_v2v](#)< F > &f)
FIXME: Doc!
- const [neighb3d](#) & [c18](#) ()
18-connectivity neighborhood on the 3D grid.
- const [neighb1d](#) & [c2](#) ()
2-connectivity neighborhood on the 1D grid.
- const [neighb3d](#) & [c26](#) ()
26-connectivity neighborhood on the 3D grid.
- const [neighb3d](#) & [c2_3d_sli](#) ()
depth 2-connectivity neighborhood on the 3D grid.
- const [neighb2d](#) & [c2_col](#) ()
Vertical 2-connectivity neighborhood on the 2D grid.
- const [neighb2d](#) & [c2_row](#) ()
Horizontal 2-connectivity neighborhood on the 2D grid.
- const [neighb2d](#) & [c4](#) ()
4-connectivity neighborhood on the 2D grid.
- const [neighb3d](#) & [c4_3d](#) ()
4-connectivity neighborhood on the 3D grid.
- const [neighb3d](#) & [c6](#) ()
6-connectivity neighborhood on the 3D grid.
- const [neighb2d](#) & [c8](#) ()
8-connectivity neighborhood on the 2D grid.
- const [neighb3d](#) & [c8_3d](#) ()
8-connectivity neighborhood on the 3D grid.
- template<typename T2 , typename T1 >
[fun::x2x::composed](#)< T2, T1 > [compose](#) (T2 f, T1 g)
Do a composition of two transformations.
- template<typename I >
mln::trait::concrete< I >::ret [duplicate](#) (const [Image](#)< I > &model)
Duplicate the image `model` with the values of the image `data`.
- template<typename I , typename F >
[extension_fun](#)< const I, F > [extend](#) (const [Image](#)< I > &ima, const [Function_v2v](#)< F > &fun)
Routines for domain extension with a function.
- template<typename I , typename J >
[extension_ima](#)< const I, const J > [extend](#) (const [Image](#)< I > &ima, const [Image](#)< J > &ext)
Routines for domain extension with an image.
- template<typename I >
[extension_val](#)< const I > [extend](#) (const [Image](#)< I > &ima, const typename I::value &val)
Routines for domain extension with a value.
- bool [implies](#) (bool lexpr, bool rexpr)
Implication.
- template<typename I , typename J >
void [initialize](#) ([Image](#)< I > &target, const [Image](#)< J > &model)

- `template<typename P >`
`box< P > larger_than (const box< P > a, const box< P > b)`
Return the minimum box including box a and box b.
- `template<typename I , typename V , typename E >`
`image2d< typename I::value > make_debug_graph_image (const I &input, const V &ima_v, const E &ima_e, const value::rgb8 &bg)`
Draw a graph.
- `mln_gen_complex_neighborhood (complex_lower_neighborhood, complex_lower_window)`
Neighborhood centered on an n-face of complex returning its adjacent (n-1)-faces.
- `mln_gen_complex_neighborhood (complex_higher_neighborhood, complex_higher_window)`
Neighborhood centered on an n-face of complex returning its adjacent (n+1)-faces.
- `mln_gen_complex_neighborhood (complex_lower_higher_neighborhood, complex_lower_higher_window)`
Neighborhood centered on an n-face of complex returning its adjacent (n-1)-faces and (n+1)-faces.
- `mln_gen_complex_neighborhood (complex_lower_dim_connected_n_face_neighborhood, complex_lower_dim_connected_n_face_window)`
Neighborhood centered on an n-face of complex returning the n-faces sharing an (n-1)-face with the center n-face.
- `mln_gen_complex_neighborhood (complex_higher_dim_connected_n_face_neighborhood, complex_higher_dim_connected_n_face_window)`
Neighborhood centered on an n-face of complex returning the n-faces sharing an (n+1)-face with the center n-face.
- `mln_gen_complex_neighborhood (complex_m_face_neighborhood, complex_m_face_window)`
Neighborhood centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face.
- `mln_gen_complex_window (complex_lower_window, topo::adj_lower_face_fwd_iter, topo::adj_lower_face_bkd_iter)`
Window centered on an n-face of complex returning its adjacent (n-1)-faces.
- `mln_gen_complex_window (complex_higher_window, topo::adj_higher_face_fwd_iter, topo::adj_higher_face_bkd_iter)`
Window centered on an n-face of complex returning its adjacent (n+1)-faces.
- `mln_gen_complex_window (complex_lower_higher_window, topo::adj_lower_higher_face_fwd_iter, topo::adj_lower_higher_face_bkd_iter)`
Window centered on an n-face of complex returning its adjacent (n-1)-faces and (n+1)-faces.
- `mln_gen_complex_window (complex_lower_dim_connected_n_face_window, topo::adj_lower_dim_connected_n_face_fwd_iter, topo::adj_lower_dim_connected_n_face_bkd_iter)`
Window centered on an n-face of complex returning the n-faces sharing an (n-1)-face with the center n-face.
- `mln_gen_complex_window (complex_higher_dim_connected_n_face_window, topo::adj_higher_dim_connected_n_face_fwd_iter, topo::adj_higher_dim_connected_n_face_bkd_iter)`
Window centered on an n-face of complex returning the n-faces sharing an (n+1)-face with the center n-face.
- `mln_gen_complex_window (complex_m_face_window, topo::adj_m_face_fwd_iter, topo::adj_m_face_bkd_iter)`
Window centered on an n-face of complex returning the m-faces transitively adjacent to this center n-face.
- `mln_gen_complex_window_p (complex_lower_window_p, topo::adj_lower_face_fwd_iter, topo::adj_lower_face_bkd_iter)`
Window centered on an n-face of complex returning its adjacent (n-1)-faces as well as the center n-face.
- `mln_gen_complex_window_p (complex_higher_window_p, topo::adj_higher_face_fwd_iter, topo::adj_higher_face_bkd_iter)`
Window centered on an n-face of complex returning its adjacent (n+1)-faces as well as the center n-face.
- `mln_gen_complex_window_p (complex_lower_higher_window_p, topo::adj_lower_higher_face_fwd_iter, topo::adj_lower_higher_face_bkd_iter)`
Window centered on an n-face of complex returning its adjacent (n-1)-faces and (n+1)-faces as well as the center n-face.
- `mln_gen_complex_window_p (complex_lower_dim_connected_n_face_window_p, topo::adj_lower_dim_connected_n_face_fwd_iter, topo::adj_lower_dim_connected_n_face_bkd_iter)`
Window centered on an n-face of complex returning the n-faces sharing an (n-1)-face with the center n-face, as well as this center n-face.

- `mln_gen_complex_window_p` (`complex_higher_dim_connected_n_face_window_p`, `topo::adj_higher_dim_connected_n_face_fwd_iter`, `topo::adj_higher_dim_connected_n_face_bkd_iter`)
Window centered on an n -face of complex returning the n -faces sharing an $(n+1)$ -face with the center n -face, as well as this center n -face.
- `mln_gen_complex_window_p` (`complex_m_face_window_p`, `topo::adj_m_face_fwd_iter`, `topo::adj_m_face_bkd_iter`)
Window centered on an n -face of complex returning the m -faces transitively adjacent to this center n -face, as well as this center n -face.
- `template<typename W1, typename W2>`
`mln_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_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_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_fwd_piter< I, G, N > &p`)
Print an `mln::complex_neighborhood_fwd_piter`.
- `template<typename I, typename G, typename W>`
`std::ostream & operator<<` (`std::ostream &ostr`, `const complex_window_bkd_piter< I, G, W > &p`)
Print an `mln::complex_window_bkd_piter`.
- `template<typename I, typename G, typename N>`
`std::ostream & operator<<` (`std::ostream &ostr`, `const complex_neighborhood_bkd_piter< I, G, N > &p`)
Print an `mln::complex_neighborhood_bkd_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<unsigned N, unsigned D, typename P>`
`bool operator<=` (`const p_faces< N, D, P > &lhs`, `const p_faces< N, D, P > &rhs`)
Inclusion of a `mln::p_faces` in another one.
- `template<typename G, typename F>`
`bool operator<=` (`const p_edges< G, F > &lhs`, `const p_edges< G, F > &rhs`)

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

- `template<typename 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 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<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<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 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 S , typename F >`
`p_if< S, F > operator| (const Site_Set< S > &s, const Function_v2b< F > &f)`

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

- `template<typename F , typename S >`
`pw::image< F, S > operator| (const Function_v2v< F > &f, const Site_Set< S > &ps)`

Construct an image from a function and a site set.

- `template<typename V , typename G , typename P >`
`edge_image< P, V, G > operator| (const fun::i2v::array< V > &edge_values, const p_edges< G, fun::i2v::array< P > > &pe)`

Construct a edge image from a [fun::i2v::array](#) and a [p_edges](#).

- `template<typename V , typename G , typename P >`
`vertex_image< P, V, G > operator| (const fun::i2v::array< V > &vertex_values, const p_vertices< G, fun::i2v::array< P > > &pv)`

Construct a vertex image from a [fun::i2v::array](#) and a [p_vertices](#).

- `template<typename I , 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 , 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 >`
`const internal::primary_type`
`< I >::ret & primary (const Image< I > &input)`

FIXME: Doc!

- `template<typename S , typename F >`
`p_transformed< S, F > ptransform (const Site_Set< S > &s, const Function_v2v< F > &f)`

*Transform a site set *s* through the function *f*.*

- `const window2d & win_c4p ()`

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

- `const window3d & win_c4p_3d ()`

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

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

Variables

- const [dpoint1d before](#) = [dpoint1d](#)(-1)
Definition of a shortcut for delta point in 1d.
- const [dpoint2d up](#) = [dpoint2d](#)(-1, 0)
Definition of a shortcut for delta point in 2d.
- const [dpoint3d sagittal_dec](#) = [dpoint3d](#)(0, 0, -1)
Definition of a shortcut for delta point in 3d.

9.1.1 Detailed Description

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

Generic class for hierarchical queues.

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

The generic dual input tree algorithm for high quantized image.

The dual input tree algorithm specialized for low quantized image.

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

Read AVS header from a file.

Define a function which aborts a process in io module.

Forward declaration.

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

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

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

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

See Also

mln::morpho::closing::volume
mln::morpho::opening::volume

The functor should provide the following methods:

- `template <typename g>=""> void init(const Graph<G>& g)` Will be called at the beginning.
- `bool to_be_treated(unsigned id)` Return whether this vertex has already been marked or if it may be a 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 olenia project but in a simplified way. Some classes have the same name in both projects and roughly have the same behavior.

Warning

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

File that includes all core definitions.

The set of operators defined in this file is:

```
l += r : l = l + r, -> l&
l -= r : l = l - r, -> l&
l *= r : l = l * r, -> l&
```

```

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

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

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

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

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

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

```

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

```

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

l == r

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

```

Convolution by a line-shaped (directional) kernel.

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

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

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

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

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

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

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

Do we want it to be exact?

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

FIXME: consider hqueues as a site set ?

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

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

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

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

9.1.2 Typedef Documentation

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

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

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

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

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

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

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

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

See Also

`mln::win::rectangle1d`.

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

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

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

See Also

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

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

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

FIXME.

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

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

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

See Also

`mln::win::rectangle3d`.

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

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

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

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

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

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

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

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

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

Definition at line 44 of file dpoint1d.hh.

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

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

Definition at line 44 of file dpoint2d.hh.

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

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

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

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

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

Definition at line 43 of file dpoint3d.hh.

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

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

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

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

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

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

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

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

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

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

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

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

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

Type alias for a run of 2d points.

Definition at line 42 of file `p_run2d.hh`.

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

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

Definition at line 42 of file `p_runs2d.hh`.

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

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

Definition at line 45 of file `point1d.hh`.

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

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

Definition at line 49 of file `point1d.hh`.

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

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

Definition at line 45 of file `point2d.hh`.

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

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

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

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

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

Definition at line 49 of file `point2d.hh`.

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

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

Definition at line 44 of file `point3d.hh`.

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

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

Definition at line 48 of file point3d.hh.

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

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

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

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

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

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

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

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

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

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

2D vector with double coordinates.

Definition at line 43 of file vec2d.hh.

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

2D vector with float coordinates.

Definition at line 40 of file vec2d.hh.

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

3D vector with double coordinates.

Definition at line 43 of file vec3d.hh.

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

3D vector with float coordinates.

Definition at line 40 of file vec3d.hh.

9.1.3 Function Documentation

9.1.3.1 `template<typename I> mln::a_point_of(const Image< I> & ima) [inline]`

Give a point of an image.

Definition at line 51 of file `a_point_of.hh`.

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

FIXME: Doc!

Definition at line 242 of file `p2p_image.hh`.

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

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

FIXME: Doc!

Definition at line 256 of file `p2p_image.hh`.

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

Do a composition of two transformations.

Parameters

<code>in</code>	<code><i>f</i></code>	The second transformation.
<code>in</code>	<code><i>g</i></code>	The first transformation.

Returns

The composed transformation `fog`.

Definition at line 306 of file `composed.hh`.

References `compose()`.

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

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

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

Parameters

<code>in</code>	<code><i>model</i></code>	The image to be duplicated.
-----------------	---------------------------	-----------------------------

Returns

The duplicate.

Precondition

`model.is_valid`

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

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

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

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

Routines for domain extension with a function.

Definition at line 140 of file `extend.hh`.

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

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

Routines for domain extension with an image.

Definition at line 162 of file `extend.hh`.

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

Routines for domain extension with a value.

Definition at line 175 of file `extend.hh`.

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

Implication.

Definition at line 68 of file `contract.hh`.

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

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

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

`\param[in, out] target` The image to be initialized.

`\param[in] model` The image to provide data for the initialization.

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

Definition at line 56 of file `initialize.hh`.

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

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

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

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

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

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

Draw a graph.

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

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

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

Exact cast routine for mln objects.

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

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

Returns

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

Definition at line 85 of file `routine/exact.hh`.

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

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

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

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

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

9.1.3.16 `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.

9.1.3.17 `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_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.

9.1.3.19 `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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

9.1.3.29 `mln::mln_gen_complex_window_p (complex_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_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.

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

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

9.1.3.32 `template<typename W1 , typename W2 > mln::mln_regular (W1) const [inline]`

Set difference between a couple of windows `win1` and `win2`.

Inter a window `win` with a delta-point `dp`.

It just calls [mln::win::diff](#).

9.1.3.33 `template<typename O1 , typename O2 > mln::mln_trait_op_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](#).

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

Is *lhs* equal to *rhs*?

Precondition

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

Definition at line 337 of file `faces_psite.hh`.

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

Is *lhs* not equal to *rhs*?

Precondition

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

Definition at line 353 of file `complex_psite.hh`.

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

```
9.1.3.36  template<typename P , typename S > P mln::operator* ( const Gpoint< P > & p, const value::scalar_< S > & s )
        [inline]
```

Multiply a point *p* by a scalar *s*.

Definition at line 405 of file `gpoint.hh`.

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

Pre-incrementation for any scalar type.

Definition at line 77 of file `concept/scalar.hh`.

References `mln::literal::one`.

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

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

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

Definition at line 155 of file `win/diff.hh`.

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

```
9.1.3.39  template<typename P , typename D > P mln::operator- ( const Gpoint< P > & p, const Gdpoint< D > & dp )
        [inline]
```

Substract a delta-point *dp* to a grid point *p*.

Parameters

<i>in</i>	<i>p</i>	A grid point.
<i>in</i>	<i>dp</i>	A delta-point.

The type of *dp* has to compatible with the type of *p*.

Returns

A point (temporary object).

See Also

[mln::Gdpoint](#)
[mln::Gdpoint](#)

Definition at line 395 of file gpoint.hh.

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

Pre-decrementation for any scalar type.

Definition at line 85 of file concept/scalar.hh.

References mln::literal::one.

9.1.3.41 `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

<code>in</code>	<code>lhs</code>	A first image.
<code>in</code>	<code>rhs</code>	A second image.

Precondition

`lhs.domain == rhs.domain`

Definition at line 107 of file compare.hh.

References mln::test::predicate().

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

Is `lhs` "less" than `rhs`?

This comparison is required by algorithms sorting psites.

Precondition

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

Definition at line 346 of file faces_psite.hh.

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

Is `lhs` "less" than `rhs`?

This comparison is required by algorithms sorting psites.

Precondition

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

Definition at line 362 of file `complex_psite.hh`.

9.1.3.44 `template<typename I, typename G, typename W > std::ostream & mln::operator<< (std::ostream & ostr, const complex_window_fwd_piter< I, G, W > & p) [inline]`

Print an [mln::complex_window_fwd_piter](#).

Definition at line 292 of file `complex_window_piter.hh`.

9.1.3.45 `template<typename I, typename G, typename N > std::ostream & mln::operator<< (std::ostream & ostr, const complex_neighborhood_fwd_piter< I, G, N > & p) [inline]`

Print an [mln::complex_neighborhood_fwd_piter](#).

Definition at line 292 of file `complex_neighborhood_piter.hh`.

9.1.3.46 `template<typename I, typename G, typename W > std::ostream & mln::operator<< (std::ostream & ostr, const complex_window_bkd_piter< I, G, W > & p) [inline]`

Print an [mln::complex_window_bkd_piter](#).

Definition at line 402 of file `complex_window_piter.hh`.

9.1.3.47 `template<typename I, typename G, typename N > std::ostream & mln::operator<< (std::ostream & ostr, const complex_neighborhood_bkd_piter< I, G, N > & p) [inline]`

Print an [mln::complex_neighborhood_bkd_piter](#).

Definition at line 399 of file `complex_neighborhood_piter.hh`.

9.1.3.48 `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

<i>in</i>	<i>lhs</i>	A first image.
<i>in</i>	<i>rhs</i>	A second image.

Precondition

`lhs.domain == rhs.domain`

Definition at line 126 of file `compare.hh`.

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

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

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

This inclusion relation is very strict for the moment, since our infrastrure for complexes is simple: a [mln::p_faces](#) is included in another one if their are equal.

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

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

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

Definition at line 339 of file `p_edges.hh`.

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

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

This inclusion relation is very strict for the moment, since our infrastructure for graphs is simple: a [mln::p_vertices](#) is included in another one if their are equal.

Definition at line 399 of file `p_vertices.hh`.

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

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

This inclusion relation is very strict for the moment, since our infrastrure for complexes is simple: a [mln::p_complex](#) is included in another one if their are equal.

Definition at line 330 of file `p_complex.hh`.

```
9.1.3.53  template<typename L , typename R > bool mln::operator== ( const Image< L > & lhs, const Image< R > & rhs )
        [inline]
```

Point-wise test if the pixel values of `lhs` are equal to the pixel values of `rhs`.

Parameters

<code>in</code>	<code>lhs</code>	A first image.
<code>in</code>	<code>rhs</code>	A second image.

Precondition

`lhs.domain == rhs.domain`

Definition at line 86 of file `compare.hh`.

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

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

Comparison of two instances of `mln::faces_psite`.

Is `lhs` equal to `rhs`?

Precondition

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

Definition at line 328 of file `faces_psite.hh`.

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

Comparison of two instances of `mln::complex_psite`.

Is *lhs* equal to *rhs*?

Precondition

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

Definition at line 344 of file `complex_psite.hh`.

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

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

Comparison between two `mln::p_faces`'s.

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

Definition at line 279 of file `p_faces.hh`.

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

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

Comparison between two `mln::p_edges`'s.

Two `mln::p_edges`'s are considered equal if they share the same graph.

Definition at line 332 of file `p_edges.hh`.

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

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

Comparison between two `mln::p_vertices`'s.

Two `mln::p_vertices`'s are considered equal if they share the same graph.

Definition at line 392 of file `p_vertices.hh`.

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

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

Comparison between two `mln::p_complex`'s.

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

Definition at line 321 of file `p_complex.hh`.

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

9.1.3.60 `template<typename S , typename F > p_if<S, F> mln::operator| (const Site_Set< S > & s, const Function_v2b< F > & f)`

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

Parameters

<i>in</i>	<i>s</i>	A site set.
<i>in</i>	<i>f</i>	A function from point to Boolean.

Returns

A subset of points.

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

Construct an image from a function and a site set.

`image = function | site_set.`

9.1.3.62 `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.`

Definition at line 217 of file `core/image/edge_image.hh`.

9.1.3.63 `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.`

Definition at line 215 of file `core/image/vertex_image.hh`.

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

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

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

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

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

FIXME: Doc!

Definition at line 129 of file `primary.hh`.

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

9.1.3.67 `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

<i>in</i>	<i>s</i>	A site set.
<i>in</i>	<i>f</i>	A function from site to site.

Returns

The transformed site set.

Definition at line 146 of file `p_transformed.hh`.

9.1.4 Variable Documentation

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

Definition of a shortcut for delta point in 1d.

Definition at line 73 of file `dpoint1d.hh`.

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

Definition of a shortcut for delta point in 3d.

Definition at line 72 of file `dpoint3d.hh`.

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

Definition of a shortcut for delta point in 2d.

Definition at line 76 of file `dpoint2d.hh`.

9.2 mln::accu Namespace Reference

Namespace of accumulators.

Namespaces

- namespace [image](#)
Namespace of accumulator image routines.
- namespace [impl](#)
Implementation namespace of accumulator namespace.
- namespace [logic](#)
Namespace of logical accumulators.
- namespace [math](#)
Namespace of mathematic accumulators.
- namespace [shape](#)
Namespace of shape accumulators.
- namespace [stat](#)
Namespace of statistical accumulators.

Classes

- struct [center](#)
Mass center accumulator.
- struct [convolve](#)
Generic convolution accumulator class.
- struct [count_adjacent_vertices](#)
Accumulator class counting the number of vertices adjacent to a set of `mln::p_edges_psite` (i.e., a set of edges).
- struct [count_labels](#)
Count the number of different labels in an image.
- struct [count_value](#)
Define an accumulator that counts the occurrence of a given value.
- struct [histo](#)
Generic histogram class over a value set with type `V`.
- struct [label_used](#)
References all the labels used.
- struct [maj_h](#)
Compute the majority value.
- struct [max_site](#)
Define an accumulator that computes the first site with the maximum value in an image.
- struct [nil](#)
Define an accumulator that does nothing.
- struct [p](#)
Generic `p` of accumulators.
- struct [pair](#)
Generic pair of accumulators.
- struct [rms](#)
Generic root mean square accumulator class.
- struct [tuple](#)
Generic tuple of accumulators.
- struct [val](#)
Generic `val` of accumulators.

Functions

- `template<typename A , typename I >`
`A::result compute (const Accumulator< A > &a, const Image< I > &input)`
Make an accumulator compute the pixels of the image `input`.
- `template<typename Meta_Accu , unsigned Dir, typename I , typename O >`
`void line (const Image< I > &input, const typename I::site &p_start, def::coord len, def::coord 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

<code>in</code>	<code>input</code>	The input image.
<code>in</code>	<code>a</code>	An accumulator.

This routine runs:

`a.take(make::pix(input, p));` on all pixels on the images.

Warning

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

Definition at line 130 of file `accu/compute.hh`.

9.2.2.2 `template<typename Meta_Accu , unsigned Dir, typename I , typename O > void mln::accu::line (const Image< I > & input, const typename I::site & p_start, def::coord len, def::coord half_length, Image< O > & output)`

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

```
\param[in] input The input image.
\param[in] p_start The starting site of the line.
\param[in] len The line length.
\param[in] half_length The half length of the line.
\param[in,out] output The resulting image.
```

```
This routine runs: \n
tmp = \p a \n
tmp.init() \n
accu::take(\p input, tmp) \n
return tmp.to_result() \n
```

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

<code>in</code>	<code>input</code>	The input image.
<code>in</code>	<code>a</code>	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

<code>in</code>	<code><i>input</i></code>	The input image.
<code>in, out</code>	<code><i>a</i></code>	The accumulator.

This routine runs:

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

Warning

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

Definition at line 87 of file `take.hh`.

9.3 mln::accu::image Namespace Reference

Namespace of accumulator image routines.

9.3.1 Detailed Description

Namespace of accumulator image routines.

9.4 mln::accu::impl Namespace Reference

Implementation namespace of accumulator namespace.

9.4.1 Detailed Description

Implementation namespace of accumulator namespace.

9.5 mln::accu::logic Namespace Reference

Namespace of logical accumulators.

Classes

- struct [land](#)
"Logical-and" accumulator.
- struct [land_basic](#)
"Logical-and" accumulator.

- struct [lor](#)
"Logical-or" accumulator.
- struct [lor_basic](#)
"Logical-or" accumulator class.

9.5.1 Detailed Description

Namespace of logical accumulators.

9.6 mln::accu::math Namespace Reference

Namespace of mathematic accumulators.

Classes

- struct [count](#)
Generic counter accumulator.
- struct [inf](#)
Generic inf accumulator class.
- struct [sum](#)
Generic sum accumulator class.
- struct [sup](#)
Generic sup accumulator class.

9.6.1 Detailed Description

Namespace of mathematic accumulators.

9.7 mln::accu::meta::logic Namespace Reference

Namespace of logical meta-accumulators.

Classes

- struct [land](#)
Meta accumulator for [land](#).
- struct [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 [histo3d_rgb](#)
 - *Define a histogram as accumulator which returns an [image3d](#).*
- struct [max](#)
 - *Generic max accumulator class.*
- struct [max_h](#)

- Generic max function based on histogram over a value set with type V .*

 - struct [mean](#)

Generic mean accumulator class.
- struct [median_alt](#)

Generic [median_alt](#) function based on histogram over a value set with type S .
- struct [median_h](#)

Generic median function based on histogram over a value set with type V .
- struct [min](#)

Generic min accumulator class.
- struct [min_h](#)

Generic min function based on histogram over a value set with type V .
- struct [min_max](#)

Generic min and max accumulator class.
- struct [rank](#)

Generic rank accumulator class.
- struct [rank< bool >](#)

rank accumulator class for Boolean.
- struct [rank_high_quant](#)

Generic rank accumulator class.
- struct [var](#)

Var accumulator class.
- struct [variance](#)

Variance accumulator class.

Functions

- `template<typename V >`
`bool operator== (const histo3d_rgb< V > &histo1, const histo3d_rgb< V > &histo2)`
Check whether an histogram is equal to an other one.

9.12.1 Detailed Description

Namespace of statistical accumulators.

9.12.2 Function Documentation

- 9.12.2.1 `template<typename V > bool mIn::accu::stat::operator== (const histo3d_rgb< V > & histo1, const histo3d_rgb< V > & histo2)`

Check whether an histogram is equal to an other one.

Parameters

<code>in</code>	<i>histo1</i>	the first histogram to compare with.
<code>in</code>	<i>histo2</i>	the second histogram.

The operator compares all the bins from the two histograms. Nobody uses this method unless unitary tests.

Definition at line 315 of file [histo3d_rgb.hh](#).

References [mIn::image3d< T >::domain\(\)](#).

9.13 mln::algebra Namespace Reference

Namespace of algebraic structure.

Classes

- struct [h_mat](#)
N-Dimensional matrix with homogeneous coordinates.
- class [h_vec](#)
N-Dimensional vector with homogeneous coordinates.

Functions

- template<typename T , typename E >
mln::trait::value_< typename
mln::trait::op::times< T, T >
::ret >::sum [det](#) (const internal::mat_base< 2, 2, T > &m)

Compute the determinant of a matrix 2x2.
- template<typename T , typename E >
mln::trait::value_< typename
mln::trait::op::times< T, T >
::ret >::sum [det](#) (const internal::mat_base< 3, 3, T > &m)

Compute the determinant of a matrix 3x3.
- 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<unsigned n, typename T , typename E >
mln::value::props< T >::sum [tr](#) (const internal::mat_base< n, n, T > &m)

Compute the trace of a matrix.
- 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<typename T, typename E> mln::trait::value_< typename mln::trait::op::times< T, T >::ret >::sum
mln::algebra::det (const internal::mat_base< 2, 2, T> & m) [inline]`

Compute the determinant of a matrix 2x2.

Definition at line 770 of file mat_base.hh.

9.13.2.2 `template<typename T, typename E> mln::trait::value_< typename mln::trait::op::times< T, T >::ret >::sum
mln::algebra::det (const internal::mat_base< 3, 3, T> & m) [inline]`

Compute the determinant of a matrix 3x3.

Definition at line 778 of file mat_base.hh.

9.13.2.3 `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 ldltc routine.

Definition at line 79 of file misc.hh.

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

9.13.2.4 `template<unsigned N, typename T> void mln::algebra::ldlt_solve (const mat< N, N, T> & A, const vec< N, T> &
rdiag, const vec< N, T> & B, vec< N, T> & x) [inline]`

Solve $Ax = B$ after `mln::algebra::ldlt_decomp`.

Definition at line 112 of file misc.hh.

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

9.13.2.5 `template<unsigned n, typename T, typename U> mln::trait::value_< typename mln::trait::op::times< T, U >::ret
>::sum mln::algebra::operator* (const vec< n, T> & lhs, const vec< n, U> & rhs) [inline]`

Scalar product (dot product).

Definition at line 633 of file algebra/vec.hh.

References `mln::literal::zero`.

9.13.2.6 `template<unsigned n, typename T, typename E> mln::value::props< T>::sum mln::algebra::tr (const
internal::mat_base< n, n, T> & m) [inline]`

Compute the trace of a matrix.

Definition at line 756 of file mat_base.hh.

References `mln::literal::zero`.

9.13.2.7 `template<typename T, typename U> vec< 3, typename mln::trait::op::times< T, U>::ret > mln::algebra::vprod (const vec< 3, T> & lhs, const vec< 3, U> & rhs) [inline]`

Vectorial product (cross product).

Definition at line 714 of file algebra/vec.hh.

References `vprod()`.

Referenced by `mln::geom::mesh_corner_point_area()`, `mln::geom::mesh_curvature()`, `mln::geom::mesh_normal()`, and `vprod()`.

9.14 mln::arith Namespace Reference

Namespace of arithmetic.

Namespaces

- namespace `impl`
Implementation namespace of arith namespace.

Functions

- template<typename I >
`mln::trait::concrete< I >::ret diff_abs` (const `Image< I >` &lhs, const `Image< I >` &rhs)
Point-wise absolute difference of images lhs and rhs.
- template<typename L, typename R, typename O >
`void div` (const `Image< L >` &lhs, const `Image< R >` &rhs, `Image< O >` &output)
Point-wise division of images lhs and rhs.
- template<typename I, typename V, typename O >
`void div_cst` (const `Image< I >` &input, const V &val, `Image< O >` &output)
Point-wise division of the value val to image input.
- template<typename L, typename R >
`void div_inplace` (`Image< L >` &lhs, const `Image< R >` &rhs)
Point-wise division of image rhs in image lhs.
- template<typename L, typename R >
`mln::trait::concrete< L >::ret min` (const `Image< L >` &lhs, const `Image< R >` &rhs)
Point-wise min of images lhs and rhs.
- template<typename L, typename R >
`void min_inplace` (`Image< L >` &lhs, const `Image< R >` &rhs)
Point-wise min of image lhs in image rhs.
- template<typename L, typename R >
`mln::trait::op::minus< L, R >::ret minus` (const `Image< L >` &lhs, const `Image< R >` &rhs)
Point-wise addition of images lhs and rhs.
- template<typename L, typename R, typename F >
`mln::trait::ch_value< L,`
typename F::result >::ret `minus` (const `Image< L >` &lhs, const `Image< R >` &rhs, const `Function_v2v< F`
> &f)
Point-wise addition of images lhs and rhs.
- template<typename V, typename L, typename R >
`mln::trait::ch_value< L, V >::ret minus` (const `Image< L >` &lhs, const `Image< R >` &rhs)
Point-wise addition of images lhs and rhs.
- template<typename I, typename V >
`mln::trait::op::minus< I, V >::ret minus_cst` (const `Image< I >` &input, const V &val)
Point-wise addition of the value val to image input.
- template<typename I, typename V, typename F >
`mln::trait::ch_value< I,`
typename F::result >::ret `minus_cst` (const `Image< I >` &input, const V &val, const `Function_v2v< F >` &f)

- Point-wise addition of the value `val` to image `input`.*

 - `template<typename I , typename V >`
`I & minus_cst_inplace (Image< I > &input, const V &val)`

Point-wise addition of the value `val` to image `input`.
- `template<typename L , typename R >`
`void minus_inplace (Image< L > &lhs, const Image< R > &rhs)`

Point-wise addition of image `rhs` in image `lhs`.
- `template<typename L , typename R >`
`mIn::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 L , typename R , typename F >`
`mIn::trait::ch_value< L,`
`typename F::result >::ret plus (const Image< L > &lhs, const Image< R > &rhs, const Function_v2v< F >`
`&f)`

Point-wise addition of images `lhs` and `rhs`.
- `template<typename V , typename L , typename R >`
`mIn::trait::ch_value< L, V >::ret plus (const Image< L > &lhs, const Image< R > &rhs)`

Point-wise addition of images `lhs` and `rhs`.
- `template<typename I , typename V >`
`mIn::trait::op::plus< I, V >::ret plus_cst (const Image< I > &input, const V &val)`

Point-wise addition of the value `val` to image `input`.
- `template<typename I , typename V , typename F >`
`mIn::trait::ch_value< I,`
`typename F::result >::ret plus_cst (const Image< I > &input, const V &val, const Function_v2v< F > &f)`

Point-wise addition of the value `val` to image `input`.
- `template<typename W , typename I , typename V >`
`mIn::trait::ch_value< I, W >::ret plus_cst (const Image< I > &input, const V &val)`

Point-wise addition of the value `val` to image `input`.
- `template<typename I , typename V >`
`I & plus_cst_inplace (Image< I > &input, const V &val)`

Point-wise addition of the value `val` to image `input`.
- `template<typename L , typename R >`
`void plus_inplace (Image< L > &lhs, const Image< R > &rhs)`

Point-wise addition of image `rhs` in image `lhs`.
- `template<typename I >`
`mIn::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

<code>in</code>	<code>lhs</code>	First operand image.
<code>in</code>	<code>rhs</code>	Second operand image.

Returns

The result image.

Precondition

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

Definition at line 63 of file `arith/diff_abs.hh`.

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

9.14.2.2 `template<typename L, typename R, typename O > void mln::arith::div (const Image< L > & lhs, const Image< R > & rhs, Image< O > & output) [inline]`

Point-wise division of images `lhs` and `rhs`.

Parameters

<code>in</code>	<code>lhs</code>	First operand image.
<code>in</code>	<code>rhs</code>	Second operand image.
<code>out</code>	<code>output</code>	The result image.

Precondition

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

Definition at line 227 of file `arith/div.hh`.

9.14.2.3 `template<typename I, typename V, typename O > void mln::arith::div_cst (const Image< I > & input, const V & val, Image< O > & output) [inline]`

Point-wise division of the value `val` to image `input`.

Parameters

<code>in</code>	<code>input</code>	The image.
<code>in</code>	<code>val</code>	The value.
<code>out</code>	<code>output</code>	The result image.

Precondition

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

Definition at line 242 of file `arith/div.hh`.

References `div_cst()`.

Referenced by `div_cst()`.

9.14.2.4 `template<typename L , typename R > void mln::arith::div_inplace (Image< L > & lhs, const Image< R > & rhs)`
`[inline]`

Point-wise division of image `rhs` in image `lhs`.

Parameters

<code>in</code>	<code>lhs</code>	First operand image (subject to division).
<code>in, out</code>	<code>rhs</code>	Second operand image (to div <code>lhs</code>).

This addition performs:

for all `p` of `rhs.domain`

`lhs(p) /= rhs(p)`

Precondition

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

Definition at line 255 of file `arith/div.hh`.

References `div_inplace()`.

Referenced by `div_inplace()`.

9.14.2.5 `template<typename L , typename R > mln::trait::concrete< L >::ret mln::arith::min (const Image< L > & lhs, const Image< R > & rhs)`
`[inline]`

Point-wise min of images `lhs` and `rhs`.

Parameters

<code>in</code>	<code>lhs</code>	First operand image.
<code>in</code>	<code>rhs</code>	Second operand image.

Returns

The result image.

Precondition

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

Definition at line 112 of file `arith/min.hh`.

References `mln::initialize()`.

9.14.2.6 `template<typename L , typename R > void mln::arith::min_inplace (Image< L > & lhs, const Image< R > & rhs)`
`[inline]`

Point-wise min of image `lhs` in image `rhs`.

Parameters

<code>in, out</code>	<code>lhs</code>	First operand image.
<code>in</code>	<code>rhs</code>	Second operand image.

Precondition

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

Definition at line 128 of file arith/min.hh.

9.14.2.7 `template<typename L , typename R > mln::trait::op::minus< L, R >::ret mln::arith::minus (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

<code>in</code>	<code>lhs</code>	First operand image.
<code>in</code>	<code>rhs</code>	Second operand image.

Returns

The result image.

Precondition

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

Definition at line 331 of file arith/minus.hh.

References `mln::initialize()`.

9.14.2.8 `template<typename L , typename R , typename F > mln::trait::ch_value< L, typename F::result >::ret mln::arith::minus (const Image< L > & lhs, const Image< R > & rhs, const Function_v2v< F > & f) [inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

<code>in</code>	<code>lhs</code>	First operand image.
<code>in</code>	<code>rhs</code>	Second operand image.
<code>in</code>	<code>f</code>	Function .

Returns

The result image.

Precondition

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

Definition at line 350 of file arith/minus.hh.

References `mln::initialize()`.

9.14.2.9 `template<typename V , typename L , typename R > mln::trait::ch_value< L, V >::ret mln::arith::minus (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

<i>in</i>	<i>lhs</i>	First operand image.
<i>in</i>	<i>rhs</i>	Second operand image.

Returns

The result image.

The free parameter *V* sets the destination value type.

Precondition

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

Definition at line 369 of file arith/minus.hh.

9.14.2.10 `template<typename I, typename V> mln::trait::op::minus<I, V>::ret mln::arith::minus_cst (const Image<I> & input, const V & val) [inline]`

Point-wise addition of the value *val* to image *input*.

Parameters

<i>in</i>	<i>input</i>	The image.
<i>in</i>	<i>val</i>	The value.

Returns

The result image.

Precondition

```
input.is_valid
```

Definition at line 387 of file arith/minus.hh.

9.14.2.11 `template<typename I, typename V, typename F> mln::trait::ch_value<I, typename F::result>::ret mln::arith::minus_cst (const Image<I> & input, const V & val, const Function_v2v<F> & f) [inline]`

Point-wise addition of the value *val* to image *input*.

Parameters

<i>in</i>	<i>input</i>	The image.
<i>in</i>	<i>val</i>	The value.
<i>in</i>	<i>f</i>	Function .

Returns

The result image.

Precondition

```
input.is_valid
```

Definition at line 405 of file arith/minus.hh.

9.14.2.12 `template<typename I , typename V > I & mln::arith::minus_cst_inplace (Image< I > & input, const V & val)`
`[inline]`

Point-wise addition of the value `val` to image `input`.

Parameters

<code>in, out</code>	<code><i>input</i></code>	The image.
<code>in</code>	<code><i>val</i></code>	The value.

Precondition

`input.is_valid`

Definition at line 440 of file `arith/minus.hh`.

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

Referenced by `minus_cst_inplace()`.

9.14.2.13 `template<typename L , typename R > void mln::arith::minus_inplace (Image< L > & lhs, const Image< R > & rhs)`
`[inline]`

Point-wise addition of image `rhs` in image `lhs`.

Parameters

<code>in, out</code>	<code><i>lhs</i></code>	First operand image (subject to addition).
<code>in</code>	<code><i>rhs</i></code>	Second operand image (to be added to <code>lhs</code>).

This addition performs:

for all `p` of `rhs.domain`

`lhs(p) -= rhs(p)`

Precondition

`rhs.domain == lhs.domain`

Definition at line 424 of file `arith/minus.hh`.

References `minus_inplace()`.

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

9.14.2.14 `template<typename L , typename R > mln::trait::op::plus< L, R >::ret mln::arith::plus (const Image< L > & lhs, const Image< R > & rhs)`
`[inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

<code>in</code>	<code><i>lhs</i></code>	First operand image.
<code>in</code>	<code><i>rhs</i></code>	Second operand image.

Returns

The result image.

Precondition

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

Definition at line 348 of file arith/plus.hh.

References `mln::initialize()`.

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

9.14.2.15 `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

<code>in</code>	<code>lhs</code>	First operand image.
<code>in</code>	<code>rhs</code>	Second operand image.
<code>in</code>	<code>f</code>	Function .

Returns

The result image.

Precondition

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

Definition at line 367 of file arith/plus.hh.

References `mln::initialize()`.

9.14.2.16 `template<typename V , typename L , typename R > mln::trait::ch_value< L, V >::ret mln::arith::plus (const Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

<code>in</code>	<code>lhs</code>	First operand image.
<code>in</code>	<code>rhs</code>	Second operand image.

Returns

The result image.

The free parameter `V` sets the destination value type.

Precondition

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

Definition at line 386 of file arith/plus.hh.

9.14.2.17 `template<typename I , typename V > mln::trait::op::plus< I, V >::ret mln::arith::plus_cst (const Image< I > & input, const V & val) [inline]`

Point-wise addition of the value `val` to image `input`.

Parameters

<code>in</code>	<code>input</code>	The image.
<code>in</code>	<code>val</code>	The value.

Returns

The result image.

Precondition

`input.is_valid`

Definition at line 404 of file arith/plus.hh.

Referenced by `plus_cst()`.

9.14.2.18 `template<typename I , typename V , typename F > mln::trait::ch_value< I, typename F::result >::ret
mln::arith::plus_cst (const Image< I > & input, const V & val, const Function_v2v< F > & f) [inline]`

Point-wise addition of the value `val` to image `input`.

Parameters

<code>in</code>	<code>input</code>	The image.
<code>in</code>	<code>val</code>	The value.
<code>in</code>	<code>f</code>	Function .

Returns

The result image.

Precondition

`input.is_valid`

Definition at line 422 of file arith/plus.hh.

9.14.2.19 `template<typename W , typename I , typename V > mln::trait::ch_value< I, W >::ret mln::arith::plus_cst (const
Image< I > & input, const V & val) [inline]`

Point-wise addition of the value `val` to image `input`.

Parameters

<code>in</code>	<code>input</code>	The image.
<code>in</code>	<code>val</code>	The value.

Returns

The result image.

Precondition

```
input.is_valid
```

Definition at line 441 of file arith/plus.hh.

References `plus_cst()`.

9.14.2.20 `template<typename I, typename V > I & mln::arith::plus_cst.inplace (Image< I > & input, const V & val)`
`[inline]`

Point-wise addition of the value `val` to image `input`.

Parameters

<code>in, out</code>	<code>input</code>	The image.
<code>in</code>	<code>val</code>	The value.

Precondition

```
input.is_valid
```

Definition at line 475 of file arith/plus.hh.

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

Referenced by `plus_cst_inplace()`.

9.14.2.21 `template<typename L, typename R > void mln::arith::plus_inplace (Image< L > & lhs, const Image< R > & rhs)`
`[inline]`

Point-wise addition of image `rhs` in image `lhs`.

Parameters

<code>in, out</code>	<code>lhs</code>	First operand image (subject to addition).
<code>in</code>	<code>rhs</code>	Second operand image (to be added to <code>lhs</code>).

This addition performs:

for all `p` of `rhs.domain`

`lhs(p) += rhs(p)`

Precondition

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

Definition at line 459 of file arith/plus.hh.

References `plus_inplace()`.

Referenced by `plus_cst_inplace()`, and `plus_inplace()`.

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

Point-wise reversion of image `input`.

Parameters

<code>in</code>	<code>input</code>	the input image.
-----------------	--------------------	------------------

Returns

The result image.

Precondition

`input.is_valid`

It performs:

for all `p` of `input.domain`

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

Definition at line 158 of file `revert.hh`.

References `mln::initialize()`.

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

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

Parameters

<code>in, out</code>	<code>input</code>	The target image.
----------------------	--------------------	-------------------

Precondition

`input.is_valid`

It performs:

for all `p` of `input.domain`

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

Definition at line 174 of file `revert.hh`.

9.14.2.24 `template<typename L , typename R , typename O > void mln::arith::times (const Image< L > & lhs, const Image< R > & rhs, Image< O > & output) [inline]`

Point-wise addition of images `lhs` and `rhs`.

Parameters

<code>in</code>	<code>lhs</code>	First operand image.
<code>in</code>	<code>rhs</code>	Second operand image.
<code>out</code>	<code>output</code>	The result image.

Precondition

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

Definition at line 226 of file `arith/times.hh`.

9.14.2.25 `template<typename I , typename V , typename O > void mln::arith::times_cst (const Image< I > & input, const V & val, Image< O > & output) [inline]`

Point-wise addition of the value `val` to image `input`.

Parameters

<code>in</code>	<i>input</i>	The image.
<code>in</code>	<i>val</i>	The value.
<code>out</code>	<i>output</i>	The result image.

Precondition

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

Definition at line 241 of file arith/times.hh.

References `times_cst()`.

Referenced by `times_cst()`.

9.14.2.26 `template<typename L, typename R> void mln::arith::times_inplace (Image< L > & lhs, const Image< R > & rhs) [inline]`

Point-wise addition of image `rhs` in image `lhs`.

Parameters

<code>in</code>	<i>lhs</i>	First operand image (subject to addition).
<code>in, out</code>	<i>rhs</i>	Second operand image (to be added to <code>lhs</code>).

This addition performs:

for all `p` of `rhs.domain`

```
lhs(p) += rhs(p)
```

Precondition

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

Definition at line 254 of file arith/times.hh.

References `times_inplace()`.

Referenced by `times_inplace()`.

9.15 mln::arith::impl Namespace Reference

Implementation namespace of arith namespace.

Namespaces

- namespace [generic](#)

Generic implementation namespace of arith namespace.

9.15.1 Detailed Description

Implementation namespace of arith namespace.

9.16 mln::arith::impl::generic Namespace Reference

Generic implementation namespace of arith namespace.

9.16.1 Detailed Description

Generic implementation namespace of arith namespace.

9.17 mln::binarization Namespace Reference

Namespace of "point-wise" expression tools.

Functions

- `template<typename I, typename F >`
`mln::trait::ch_value< I, bool >`
`::ret binarization (const Image< I > &input, const Function_v2b< F > &fun)`
Thresholds the values of `input` so that they can be stored in the `output` binary image.
- `template<typename I >`
`mln::trait::ch_value< I, bool >`
`::ret threshold (const Image< I > &input, const typename I::value threshold)`
Thresholds the values of `input` so that they can be stored in the `output` binary image.

9.17.1 Detailed Description

Namespace of "point-wise" expression tools.

9.17.2 Function Documentation

9.17.2.1 `template<typename I, typename F > mln::trait::ch_value< I, bool >::ret mln::binarization::binarization (const Image< I > & input, const Function_v2b< F > & fun) [inline]`

Thresholds the values of `input` so that they can be stored in the `output` binary image.

Parameters

<code>in</code>	<code>input</code>	The input image.
<code>in</code>	<code>fun</code>	The thresholding function, from value(I) to bool.

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

Definition at line 86 of file `binarization.hh`.

Referenced by `threshold()`.

9.17.2.2 `template<typename I > mln::trait::ch_value< I, bool >::ret mln::binarization::threshold (const Image< I > & input, const typename I::value threshold) [inline]`

Thresholds the values of `input` so that they can be stored in the `output` binary image.

Parameters

<code>in</code>	<code>input</code>	The input image.
<code>in</code>	<code>threshold</code>	The threshold.

If `input(p)` is greater or equal than the threshold, the value in the output image in the same point will be `TRUE`, else `FALSE`.

Definition at line 61 of file `binarization/threshold.hh`.

References `binarization()`.

9.18 `mln::border` Namespace Reference

Namespace of routines related to image virtual (outer) border.

Namespaces

- namespace `impl`
Implementation namespace of border namespace.

Functions

- template<typename I >
void `adjust` (const `Image`< I > &`ima`, unsigned `min_thickness`)
 - template<typename I >
void `duplicate` (const `Image`< I > &`ima`)
 - template<typename I , typename J >
void `equalize` (const `Image`< I > &`ima1`, const `Image`< J > &`ima2`, unsigned `min_thickness`)
 - template<typename I >
void `fill` (const `Image`< I > &`ima`, const typename I::value &`v`)
 - template<typename I >
unsigned `find` (const `Image`< I > &`ima`)
 - template<typename I >
unsigned `get` (const `Image`< I > &`ima`)
 - template<typename I >
void `mirror` (const `Image`< I > &`ima`)
 - template<typename I >
void `resize` (const `Image`< I > &`ima`, unsigned `thickness`)
- Facade.*

9.18.1 Detailed Description

Namespace of routines related to image virtual (outer) border.

9.18.2 Function Documentation

9.18.2.1 template<typename I > void `mln::border::adjust` (const `Image`< I > &`ima`, unsigned `min_thickness`)
[inline]

Adjust the virtual (outer) border of image `ima` so that its size is at least `min_thickness`.

Parameters

<code>in, out</code>	<code>ima</code>	The image whose border is to be adjusted.
<code>in</code>	<code>min_thickness</code>	The expected border minimum thickness.

Precondition

ima has to be initialized.

Warning

If the image border is already larger than `min_thickness`, this routine is a no-op.

Definition at line 62 of file `border/adjust.hh`.

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

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

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

Parameters

<i>in, out</i>	<i>ima</i>	The image whose border is to be duplicated.
----------------	------------	---

Precondition

ima has to be initialized.

Definition at line 254 of file `border/duplicate.hh`.

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

<i>in, out</i>	<i>ima1</i>	The first image whose border is to be equalizeed.
<i>in, out</i>	<i>ima2</i>	The second image whose border is to be equalizeed.
<i>in</i>	<i>min_thickness</i>	The expected border minimum thickness of both images.

Precondition

ima1 has to be initialized.

ima2 has to be initialized.

Warning

If both image borders already have the same thickness and if this thickness is larger than `min_thickness`, this routine is a no-op.

Definition at line 112 of file `border/equalize.hh`.

References `get()`.

9.18.2.4 `template<typename I> void mln::border::fill (const Image< I > & ima, const typename I::value & v) [inline]`

Fill the virtual (outer) border of image *ima* with the single value *v*.

Parameters

<i>in, out</i>	<i>ima</i>	The image whose border is to be filled.
<i>in</i>	<i>v</i>	The value to assign to all border pixels.

Precondition

ima has to be initialized.

Definition at line 204 of file border/fill.hh.

9.18.2.5 `template<typename I> unsigned mln::border::find (const Image<I> & ima)` `[inline]`

Find the virtual (outer) border thickness of image *ima*.

```
\param[in] ima The image.
\result The border thickness (0 if there is no border).

\pre \p ima has to be initialized.
```

Definition at line 95 of file find.hh.

9.18.2.6 `template<typename I> unsigned mln::border::get (const Image<I> & ima)` `[inline]`

Get the virtual (outer) border thickness of image *ima*.

```
\param[in] ima The image.
\result The border thickness (0 if there is no border).

\pre \a ima has to be initialized.
```

Definition at line 90 of file border/get.hh.

Referenced by `adjust()`, `duplicate()`, and `equalize()`.

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

Mirror the virtual (outer) border of image *ima* with the (inner) level contents of this image.

Parameters

<i>in, out</i>	<i>ima</i>	The image whose border is to be mirrored.
----------------	------------	---

Precondition

ima has to be initialized.

Definition at line 212 of file border/mirror.hh.

9.18.2.8 `template<typename I> void mln::border::resize (const Image<I> & ima, unsigned thickness)` `[inline]`

Facade.

Resize the virtual (outer) border of image *ima* to exactly *thickness*.

Parameters

<i>in, out</i>	<i>ima</i>	The image whose border is to be resized.
<i>in</i>	<i>thickness</i>	The expected border thickness.

Precondition

`ima` has to be initialized.

Warning

If the image border already has the expected thickness, this routine is a no-op.

Definition at line 126 of file `resize.hh`.

References `mln::primary()`, and `resize()`.

Referenced by `adjust()`, and `resize()`.

9.19 mln::border::impl Namespace Reference

Implementation namespace of border namespace.

Namespaces

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

9.19.1 Detailed Description

Implementation namespace of border namespace.

9.20 mln::border::impl::generic Namespace Reference

Generic implementation namespace of border namespace.

9.20.1 Detailed Description

Generic implementation namespace of border namespace.

9.21 mln::canvas Namespace Reference

Namespace of canvas.

Namespaces

- namespace [browsing](#)
Namespace of browsing canvas.
- namespace [impl](#)
Implementation namespace of canvas namespace.
- namespace [labeling](#)
Namespace of labeling canvas.
- namespace [morpho](#)
Namespace of morphological canvas.

Classes

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

Functions

- `template<typename I, typename N, typename W, typename D, typename F > mln::trait::ch_value< I, D >::ret distance_front (const Image< I > &input, const Neighborhood< N > &nbh, const Weighted_Window< W > &w_win, D max, F &functor)`
Canvas of discrete distance computation by thick front propagation.
- `template<typename I, typename N, typename D, typename F > mln::trait::ch_value< I, D >::ret distance_geodesic (const Image< I > &input, const Neighborhood< N > &nbh, D max, F &functor)`
Discrete geodesic distance canvas.

9.21.1 Detailed Description

Namespace of canvas.

9.21.2 Function Documentation

- 9.21.2.1 `template<typename I, typename N, typename W, typename D, typename F > mln::trait::ch_value< I, D >::ret mln::canvas::distance_front (const Image< I > & input, const Neighborhood< N > & nbh, const Weighted_Window< W > & w_win, D max, F & functor)` `[inline]`

Canvas of discrete distance computation by thick front propagation.

Definition at line 397 of file `canvas/distance_front.hh`.

Referenced by `mln::transform::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.

Definition at line 321 of file `canvas/distance_geodesic.hh`.

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

<i>in</i>	<i>input</i>	The input image.
<i>in</i>	<i>nbh</i>	The connexity of the objects.
<i>out</i>	<i>nlabels</i>	The Number of labels. Its value is set in the algorithms.
<i>in, out</i>	<i>functor</i>	A functor computing data while labeling.

Returns

The label image.

Precondition

The input image has to be binary (checked at compile-time).

A fast queue is used so that the algorithm is not recursive and can handle large binary objects (blobs).

Definition at line 167 of file canvas/labeling/blobs.hh.

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 int &from, [Value](#)< V > &to)
Conversion of a int `from` towards a value `to`.
- template<typename V >
void [from_to](#) (const double &from, [Value](#)< V > &to)
Conversion of a double `from` towards a value `to`.
- template<typename V >
void [from_to](#) (const float &from, [Value](#)< V > &to)
Conversion of a float `from` towards a value `to`.
- template<typename V >
void [from_to](#) (const unsigned &from, [Value](#)< V > &to)
Conversion of an unsigned `from` towards a value `to`.
- template<typename S >
[mln_image_from_grid](#) (typename S::site::grid, bool) [to_image](#)(const [Site_Set](#)< S > &pset)
Convert a point set `pset` into a binary image.
- template<typename W >
[mln_image_from_grid](#) (typename W::site::grid, bool) [to_image](#)(const [Window](#)< W > &win)
Convert a window `win` into a binary image.
- template<typename W >
[mln_image_from_grid](#) (typename W::site::grid, mln_weight(W)) [to_image](#)(const [Weighted_Window](#)< W > &w_win)
Convert a weighted window `w_win` into an image.
- template<typename N >
[mln_image_from_grid](#) (typename N::site::grid, bool) [to_image](#)(const [Neighborhood](#)< N > &nbh)
Convert a neighborhood `nbh` into a binary image.
- template<typename N >
[mln_window](#) (N) [to_window](#)(const [Neighborhood](#)< N > &nbh)
Convert a neighborhood `nbh` into a window.
- template<typename T, typename O >
[T](#) [to](#) (const O &from)
Conversion of the object `from` towards an object with type `T`.
- template<typename P >
P::dpoint [to_dpoint](#) (const [Point_Site](#)< P > &p)
Convert a point site `p` into a delta-point.
- template<typename I >
pw::value< I > [to_fun](#) (const [Image](#)< I > &ima)
Convert an image into a function.
- template<typename T >
[image1d](#)< unsigned > [to_image](#) (const [histo::array](#)< T > &h)
Convert an histo `h` into an [image1d](#)<unsigned>.
- template<typename S >
[p_array](#)< typename S::psite > [to_p_array](#) (const [Site_Set](#)< S > &pset)
Convert a point set `pset` into a [p_array](#) (point set vector).
- template<typename W >
[p_array](#)< typename W::psite > [to_p_array](#) (const [Window](#)< W > &win, const typename W::psite &p)
Convert a window `win` centered at point `p` into a [p_array](#) (point set vector).
- template<typename I >
[p_array](#)< typename I::psite > [to_p_array](#) (const [Image](#)< I > &img)
Convert an image `img` into a [p_array](#).
- template<typename N >
[p_set](#)< typename N::psite > [to_p_set](#) (const [Neighborhood](#)< N > &nbh)

- Convert a neighborhood `nbh` into a site set.*

 - `template<typename I >`
`p_set< typename I::psite > to_p_set (const Image< I > &ima)`
Convert a binary image `ima` into a site set.
 - `template<typename W >`
`p_set< typename W::psite > to_p_set (const Window< W > &win)`
Convert a `Window` `win` into a site set.
 - `template<typename P , typename C >`
`p_set< P > to_p_set (const std::set< P, C > &s)`
Convert an `std::set` `s` of sites into a site set.
 - `template<typename S >`
`p_set< typename S::psite > to_p_set (const Site_Set< S > &ps)`
Convert any site set `ps` into a '`mln::p_set`' site set.
 - `template<typename I >`
`QImage to_qimage (const Image< I > &ima)`
Convert a Milena image to a QImage.
 - `template<typename W >`
`window< typename W::dpsite > to_upper_window (const Window< W > &win)`
Convert a window `nbh` into an upper window.
 - `template<typename N >`
`window< typename N::dpoint > to_upper_window (const Neighborhood< N > &nbh)`
Convert a neighborhood `nbh` into an upper window.
 - `template<typename I >`
`window< typename I::site::dpsite > to_window (const Image< I > &ima)`
Convert a binary image `ima` into a window.
 - `template<typename S >`
`window< typename S::site::dpsite > to_window (const Site_Set< S > &pset)`
Convert a site set `pset` into a window.
 - `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.

Variables

- `template<typename R , typename A >`
`fun::C< R(*) (A)> to_fun (R(*) (A))`
Convert a C unary function into an `mln::fun::C`.

9.27.1 Detailed Description

Namespace of conversion routines.

9.27.2 Function Documentation

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

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

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

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

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

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

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

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

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

Convert a point set *pset* into a binary image.

Width of the converted image will be *pset.bbox* + 2 * *border*.

Referenced by `mln_image_from_grid()`.

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

Convert a window *win* into a binary image.

Definition at line 104 of file `to_image.hh`.

References `mln_image_from_grid()`.

9.27.2.7 `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.

Definition at line 116 of file `to_image.hh`.

References `mln_image_from_grid()`.

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

Convert a neighborhood *nbh* into a binary image.

Definition at line 128 of file `to_image.hh`.

References `to_image()`.

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

Convert a neighborhood *nbh* into a window.

Definition at line 74 of file `to_window.hh`.

9.27.2.10 `template<typename T, typename O> T mln::convert::to (const O & from) [inline]`

Conversion of the object *from* towards an object with type *T*.

Definition at line 63 of file `to.hh`.

References `mln::mln_exact()`.

Referenced by `mln::make_debug_graph_image()`.

9.27.2.11 `template<typename P> P::dpoint mln::convert::to_dpoint (const Point_Site< P> & p)` `[inline]`

Convert a point site `p` into a delta-point.

Definition at line 52 of file `to_dpoint.hh`.

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

Convert an image into a function.

Definition at line 64 of file `to_fun.hh`.

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

Convert an histo `h` into an `image1d<unsigned>`.

Definition at line 138 of file `to_image.hh`.

References `mln::make::box1d()`.

Referenced by `mln_image_from_grid()`.

9.27.2.14 `template<typename S> p_array< typename S::psite> mln::convert::to_p_array (const Site_Set< S> & pset)`
`[inline]`

Convert a point set `pset` into a `p_array` (point set vector).

Definition at line 68 of file `to_p_array.hh`.

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

9.27.2.15 `template<typename W> p_array< typename W::psite> mln::convert::to_p_array (const Window< W> & win, const typename W::psite & p)` `[inline]`

Convert a window `win` centered at point `p` into a `p_array` (point set vector).

Definition at line 82 of file `to_p_array.hh`.

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

9.27.2.16 `template<typename I> p_array< typename I::psite> mln::convert::to_p_array (const Image< I> & img)`
`[inline]`

Convert an image `img` into a `p_array`.

Definition at line 98 of file `to_p_array.hh`.

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

9.27.2.17 `template<typename N> p_set< typename N::psite> mln::convert::to_p_set (const Neighborhood< N> & nbh)`
`[inline]`

Convert a neighborhood `nbh` into a site set.

Definition at line 83 of file `to_p_set.hh`.

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

9.27.2.18 `template<typename I > p_set< typename I::psite > mln::convert::to_p_set (const Image< I > & ima)`
`[inline]`

Convert a binary image `ima` into a site set.

Definition at line 97 of file `to_p_set.hh`.

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

9.27.2.19 `template<typename W > p_set< typename W::psite > mln::convert::to_p_set (const Window< W > & win)`
`[inline]`

Convert a [Window](#) `win` into a site set.

Definition at line 117 of file `to_p_set.hh`.

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

9.27.2.20 `template<typename P , typename C > p_set< P > mln::convert::to_p_set (const std::set< P, C > & s)`
`[inline]`

Convert an `std::set s` of sites into a site set.

`C` is the comparison functor.

Definition at line 130 of file `to_p_set.hh`.

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

9.27.2.21 `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 '`mln::p_set`' site set.

Definition at line 144 of file `to_p_set.hh`.

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

9.27.2.22 `template<typename I > QImage mln::convert::to_qimage (const Image< I > & ima)` `[inline]`

Convert a Milena image to a QImage.

Definition at line 281 of file `to_qimage.hh`.

9.27.2.23 `template<typename W > window< typename W::dpsite > mln::convert::to_upper_window (const Window< W > & win)` `[inline]`

Convert a window `nbh` into an upper window.

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

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

9.27.2.24 `template<typename N > window< typename N::dpoint > mln::convert::to_upper_window (const Neighborhood< N > & nbh)` `[inline]`

Convert a neighborhood `nbh` into an upper window.

Definition at line 82 of file `to_upper_window.hh`.

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

9.27.2.25 `template<typename I > window< typename I::site::dpsite > mln::convert::to_window (const Image< I > & ima)`
`[inline]`

Convert a binary image `ima` into a window.

Definition at line 94 of file `to_window.hh`.

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

Referenced by `to_window()`.

9.27.2.26 `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.

Definition at line 117 of file `to_window.hh`.

References `to_window()`.

9.27.2.27 `template<typename D , typename C > window< D > mln::convert::to_window (const std::set< D, C > & s)`
`[inline]`

Convert an `std::set s` of delta-sites into a window.

Definition at line 128 of file `to_window.hh`.

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

9.27.3 Variable Documentation

9.27.3.1 `template<typename R , typename A > fun::C<R(*) (A)> mln::convert::to_fun(R(*) (A))`

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

Definition at line 45 of file `to_fun.hh`.

9.28 mln::data Namespace Reference

Namespace of image processing routines related to pixel data.

Namespaces

- namespace [approx](#)
Namespace of image processing routines related to pixel levels with approximation.
- namespace [impl](#)
Implementation namespace of data namespace.
- namespace [naive](#)
Namespace of image processing routines related to pixel levels with naive approach.

Functions

- `template<typename I , typename O >`
`void abs (const Image< I > &input, Image< O > &output)`
- `template<typename I >`
`void abs_inplace (Image< I > &input)`

- template<typename I , typename F >
void [apply](#) ([Image](#)< I > &input, const [Function_v2v](#)< F > &f)
- template<typename A , typename I >
A::result [compute](#) (const [Accumulator](#)< A > &a, const [Image](#)< I > &input)
Compute an accumulator onto the pixel values of the image input.
- template<typename A , typename I >
A::result [compute](#) ([Accumulator](#)< A > &a, const [Image](#)< I > &input)
Compute an accumulator onto the pixel values of the image input.
- template<typename V , typename I >
mln::trait::ch_value< I, V >::ret [convert](#) (const V &v, const [Image](#)< I > &input)
Convert the image input by changing the value type.
- template<typename I , typename W , typename O >
void [fast_median](#) (const [Image](#)< I > &input, const [Window](#)< W > &win, [Image](#)< O > &output)
- template<typename I , typename D >
void [fill](#) ([Image](#)< I > &ima, const D &data)
- template<typename I , typename J >
void [fill_with_image](#) ([Image](#)< I > &ima, const [Image](#)< J > &data)
Fill the image ima with the values of the image data.
- template<typename I , typename W >
mln::trait::concrete< I >::ret [median](#) (const [Image](#)< I > &input, const [Window](#)< W > &win)
- template<typename A , typename I >
[mln_meta_accu_result](#) (A, typename I::value) [compute](#)(const [Meta_Accumulator](#)< A > &a
Compute an accumulator onto the pixel values of the image input.
- template<typename I , typename J >
void [paste](#) (const [Image](#)< I > &input, [Image](#)< J > &output)
Paste the contents of image input into the image output.
- template<typename I , typename J >
void [paste_without_localization](#) (const [Image](#)< I > &input, [Image](#)< J > &output)
Paste the contents of image input into the image output without taking into account the localization of sites.
- template<typename I >
void [replace](#) ([Image](#)< I > &input, const typename I::value &old_value, const typename I::value &new_value)
- template<typename V , typename I >
mln::trait::ch_value< I, V >::ret [saturate](#) (V v, const [Image](#)< I > &input)
- 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 I >
void [saturate_inplace](#) ([Image](#)< I > &input, const typename I::value &min, const typename I::value &max)
- template<typename I >
[util::array](#)< unsigned > [sort_offsets_increasing](#) (const [Image](#)< I > &input)
Sort pixel offsets of the image input wrt increasing pixel values.
- template<typename I >
[p_array](#)< typename I::psite > [sort_psites_decreasing](#) (const [Image](#)< I > &input)
- template<typename I >
[p_array](#)< typename I::psite > [sort_psites_increasing](#) (const [Image](#)< I > &input)
- template<typename V , typename I >
mln::trait::ch_value< I, V >::ret [stretch](#) (const V &v, const [Image](#)< I > &input)
- template<typename I , typename O >
void [to_enc](#) (const [Image](#)< I > &input, [Image](#)< O > &output)
- template<typename I , typename F >
mln::trait::ch_value< I,
typename F::result >::ret [transform](#) (const [Image](#)< I > &input, const [Function_v2v](#)< F > &f)
- template<typename I1 , typename I2 , typename F >
mln::trait::ch_value< I1,
typename F::result >::ret [transform](#) (const [Image](#)< I1 > &input1, const [Image](#)< I2 > &input2, const [Function-
_vv2v](#)< F > &f)

- `template<typename I, typename F >`
`void transform_inplace (Image< I > &ima, const Function_v2v< F > &f)`
- `template<typename I1, typename I2, typename F >`
`void transform_inplace (Image< I1 > &ima, const Image< I2 > &aux, const Function_vv2v< F > &f)`
- `template<typename A, typename I >`
`A::result update (Accumulator< A > &a, const Image< I > &input)`
- `template<typename V, typename I >`
`mln::trait::ch_value< I, V >::ret wrap (const V &v, const Image< I > &input)`
Routine to wrap values such as 0 -> 0 and [1, Lmax] maps to [1, Lmax] (using modulus).
- `template<typename I, typename V >`
`void fill_with_value (Image< I > &ima, const V &val)`
Fill the whole image ima with the single value v.

9.28.1 Detailed Description

Namespace of image processing routines related to pixel data.

9.28.2 Function Documentation

9.28.2.1 `template<typename I, typename O > void mln::data::abs (const Image< I > &input, Image< O > &output)`
`[inline]`

Apply the absolute value (abs) function to image pixel values.

```
\param[in] input The input image.
\param[out] output The output image.
```

Definition at line 68 of file data/abs.hh.

References `transform()`.

9.28.2.2 `template<typename I > void mln::data::abs_inplace (Image< I > &input)` `[inline]`

Apply the absolute value (abs) function to image pixel values.

```
\param[in,out] input The input image.
```

Definition at line 80 of file data/abs.hh.

References `apply()`.

9.28.2.3 `template<typename I, typename F > void mln::data::apply (Image< I > &input, const Function_v2v< F > &f)`
`[inline]`

Apply a function-object to the image input.

```
\param[in,out] input The input image.
\param[in] f The function-object.
```

```
This routine runs: \n
for all p of \p input, \p input(p) = \p f( \p input(p) ) \n
```

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

```
\xrefitem todo 145.
```

Definition at line 95 of file apply.hh.

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

9.28.2.4 `template<typename A , typename I > A::result mln::data::compute (const Accumulator< A > & a, const Image< I > & input) [inline]`

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

Be ware that the given accumulator won't be modified and won't store any result.

Parameters

<code>in</code>	<code>a</code>	An accumulator.
<code>in</code>	<code>input</code>	The input image.

Returns

The accumulator result.

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

Definition at line 93 of file `data/compute.hh`.

Referenced by `mln::labeling::colorize()`, `mln::estim::mean()`, `mln::estim::min_max()`, `mln::labeling::pack()`, `mln::labeling::pack_inplace()`, and `mln::estim::sum()`.

9.28.2.5 `template<typename A , typename I > A::result mln::data::compute (Accumulator< A > & a, const Image< I > & input) [inline]`

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

Parameters

<code>in, out</code>	<code>a</code>	An accumulator.
<code>in</code>	<code>input</code>	The input image.

Returns

The accumulator result.

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

Definition at line 104 of file `data/compute.hh`.

9.28.2.6 `template<typename V , typename I > mln::trait::ch_value< I, V >::ret mln::data::convert (const V & v, const Image< I > & input) [inline]`

Convert the image `input` by changing the value type.

Parameters

<code>in</code>	<code>v</code>	A value of the destination type.
<code>in</code>	<code>input</code>	The input image.

Definition at line 154 of file `mln/data/convert.hh`.

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

9.28.2.7 `template<typename I , typename W , typename O > void mln::data::fast_median (const Image< I > & input, const Window< W > & win, Image< O > & output) [inline]`

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

Parameters

<code>in</code>	<i>input</i>	The image to be filtered.
<code>in</code>	<i>win</i>	The window.
<code>in, out</code>	<i>output</i>	The output image.

Precondition

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

Definition at line 167 of file `fast_median.hh`.

9.28.2.8 `template<typename I, typename D > void mln::data::fill (Image< I > & ima, const D & data) [inline]`

Fill the whole image `ima` with the data provided by `aux`.

```
\param[in,out] ima The image to be filled.
\param[in] data The auxiliary data to fill the image \p ima.

\pre \p ima has to be initialized.
```

Definition at line 138 of file `data/fill.hh`.

Referenced by `mln::draw::box_plain()`, `mln::draw::dashed_line()`, `mln::topo::detach()`, `mln::util::display_branch()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::duplicate()`, `mln::make::edge_image()`, `mln::morpho::tree::filter::filter()`, `mln::transform::hough()`, `mln::registration::icp()`, `mln::accu::stat::histo3d_rgb< V >::init()`, `mln::graph::labeling()`, `mln::morpho::laplacian()`, `mln::make_debug_graph_image()`, `mln::morpho::tree::filter::max()`, `mln::geom::mesh_corner_point_area()`, `mln::geom::mesh_normal()`, `mln::morpho::meyer_wst()`, `mln::morpho::tree::filter::min()`, `mln::debug::mosaic()`, `mln::debug::slices_2d()`, `mln::morpho::watershed::superpose()`, `mln::debug::superpose()`, `mln::morpho::watershed::topological()`, and `mln::geom::translate()`.

9.28.2.9 `template<typename I, typename J > void mln::data::fill_with_image (Image< I > & ima, const Image< J > & data) [inline]`

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

Parameters

<code>in, out</code>	<i>ima</i>	The image to be filled.
<code>in</code>	<i>data</i>	The image.

Warning

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

Precondition

`ima.domain <= data.domain`.

Definition at line 124 of file `fill_with_image.hh`.

9.28.2.10 `template<typename I, typename V > void mln::data::fill_with_value (Image< I > & ima, const V & val) [inline]`

Fill the whole image `ima` with the single value `v`.

Parameters

<i>in, out</i>	<i>ima</i>	The image to be filled.
<i>in</i>	<i>val</i>	The value to assign to all sites.

Precondition

ima has to be initialized.

Definition at line 130 of file fill_with_value.hh.

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

9.28.2.11 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::data::median (const Image< I > & input, const Window< W > & win)`

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

Parameters

<i>in</i>	<i>input</i>	The image to be filtered.
<i>in</i>	<i>win</i>	The window.

Precondition

input have to be initialized.

Definition at line 270 of file median.hh.

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

<i>in</i>	<i>a</i>	A meta-accumulator.
<i>in</i>	<i>input</i>	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

<i>in</i>	<i>input</i>	The input image providing pixels values.
<i>in, out</i>	<i>output</i>	The image in which values are assigned.

This routine runs:

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

Warning

The definition domain of `input` has to be included in the one of `output`; so using `mln::safe_image` does not make pasting outside the output domain work.

Precondition

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

Definition at line 137 of file `paste.hh`.

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

9.28.2.14 `template<typename I, typename J> void mln::data::paste_without_localization (const Image< I > & input, Image< J > & output) [inline]`

Paste the contents of image `input` into the image `output` without taking into account the localization of sites.

Parameters

<code>in</code>	<code>input</code>	The input image providing pixels values.
<code>in, out</code>	<code>output</code>	The image in which values are assigned.

Definition at line 349 of file `paste_without_localization.hh`.

9.28.2.15 `template<typename I> void mln::data::replace (Image< I > & input, const typename I::value & old_value, const typename I::value & new_value)`

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

```
\param[in] input The input image.
\param[in] old_value The value to be replaced...
\param[in] new_value ...by this one.
```

Definition at line 88 of file `replace.hh`.

9.28.2.16 `template<typename V, typename I> mln::trait::ch_value< I, V >::ret mln::data::saturate (V v, const Image< I > & input) [inline]`

Apply the `saturate` function to image pixel values.

```
\param[in] v      A value of the output type.
\param[in] input The input image.
```

The saturation is based on the min and max values of the output value type. This assumes that the range of values in the input image is larger than the one of the output image.

Definition at line 87 of file `data/saturate.hh`.

References `transform()`.

9.28.2.17 `template<typename I, typename V> mln::trait::ch_value< I, V >::ret mln::data::saturate (const Image< I > & input, const V & min, const V & max) [inline]`

Apply the `saturate` function to image pixel values.


```
\param[in] input The input image.
\param[in] min The minimum output value.
\param[in] max The maximum output value.
```

Definition at line 103 of file data/saturate.hh.

References `transform()`.

9.28.2.18 `template<typename I > void mln::data::saturate_inplace (Image< I > & input, const typename I::value & min, const typename I::value & max) [inline]`

Apply the saturate function to image pixel values.

```
\param[in,out] input The input image.
\param[in] min The minimum output value.
\param[in] max The maximum output value
```

Definition at line 119 of file data/saturate.hh.

References `apply()`.

9.28.2.19 `template<typename I > util::array< unsigned > mln::data::sort_offsets_increasing (const Image< I > & input) [inline]`

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

Definition at line 298 of file sort_offsets.hh.

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

Sort psites the image `input` through a function `f` to set the `output` image in decreasing way.

Parameters

<code>in</code>	<code>input</code>	The input image.
-----------------	--------------------	------------------

Precondition

```
input.is_valid
```

Definition at line 229 of file sort_psites.hh.

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

9.28.2.21 `template<typename I > p_array< typename I::psite > mln::data::sort_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

<code>in</code>	<code>input</code>	The input image.
-----------------	--------------------	------------------

Precondition

```
input.is_valid
```

Definition at line 219 of file sort_psites.hh.

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

9.28.2.22 `template<typename V , typename I > mln::trait::ch_value< I, V >::ret mln::data::stretch (const V & v, const Image< I > & input) [inline]`

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

Parameters

<code>in</code>	<code>v</code>	A value to set the output value type.
<code>in</code>	<code>input</code>	The input image.

Returns

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

Precondition

`input.is_valid`

Definition at line 128 of file `stretch.hh`.

References `mln::data::impl::stretch()`.

9.28.2.23 `template<typename I , typename O > void mln::data::to_enc (const Image< I > & input, Image< O > & output) [inline]`

Set the output image with the encoding values of the image `input` pixels.

```
\param[in] input The input image.
\param[out] output The result image.

\pre \p output.domain >= \p input.domain
```

Definition at line 60 of file `data/to_enc.hh`.

References `transform()`.

9.28.2.24 `template<typename I , typename F > mln::trait::ch_value< I, typename F::result >::ret mln::data::transform (const Image< I > & input, const Function_v2v< F > & f) [inline]`

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

```
\param[in] input The input image.
\param[in] f The function.

This routine runs: \n
for all p of \p input, \p output(p) = \p f( \p input(p) ).
```

Definition at line 202 of file `data/transform.hh`.

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

9.28.2.25 `template<typename I1 , typename I2 , typename F > mln::trait::ch_value< I1, typename F::result >::ret mln::data::transform (const Image< I1 > & input1, const Image< I2 > & input2, const Function_vv2v< F > & f) [inline]`

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

```
\param[in] input1 The 1st input image.
\param[in] input2 The 2nd input image.
\param[in] f The function.
```

This routine runs: \n
for all p of \p input, \p output(p) = \p f(\p input1(p), \p input2(p)).

Definition at line 219 of file data/transform.hh.

9.28.2.26 `template<typename I, typename F> void mln::data::transform_inplace (Image< I > & ima, const Function_v2v< F > & f)`

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

```
\param[in,out] ima The image to be transformed.
\param[in] f The function.
```

This routine runs: \n
for all p of \p ima, \p ima(p) = \p f(\p ima(p)).

Definition at line 490 of file transform_inplace.hh.

Referenced by `mln::logical::and_inplace()`, `mln::logical::and_not_inplace()`, `mln::logical::not_inplace()`, `mln::logical::or_inplace()`, `mln::labeling::relabel_inplace()`, and `mln::logical::xor_inplace()`.

9.28.2.27 `template<typename I1, typename I2, typename F> void mln::data::transform_inplace (Image< I1 > & ima, const Image< I2 > & aux, const Function_vv2v< F > & f)`

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

Parameters

in	<i>ima</i>	The image to be transformed.
in	<i>aux</i>	The auxiliary image.
in	<i>f</i>	The function.

This routine runs:

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

Definition at line 502 of file transform_inplace.hh.

9.28.2.28 `template<typename A, typename I> A::result mln::data::update (Accumulator< A > & a, const Image< I > & input) [inline]`

Update an accumulator with the pixel values of the image *input*.

```
\param[in] a The accumulator.
\param[in] input The input image.
\return The accumulator result.
```

Definition at line 191 of file update.hh.

9.28.2.29 `template<typename V, typename I> mln::trait::ch_value< I, V >::ret mln::data::wrap (const V & v, const Image< I > & input)`

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

Parameters

<code>in</code>	<code>v</code>	The target value type.
<code>in</code>	<code>input</code>	Input image.

Returns

An image with wrapped values.

Definition at line 65 of file `data/wrap.hh`.

References `transform()`.

9.29 `mln::data::approx` Namespace Reference

Namespace of image processing routines related to pixel levels with approximation.

Namespaces

- namespace [impl](#)
Implementation namespace of [data::approx](#) namespace.

Functions

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

9.29.1 Detailed Description

Namespace of image processing routines related to pixel levels with approximation.

9.29.2 Function Documentation

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

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

Parameters

<code>in</code>	<code>input</code>	The image to be filtered.
<code>in</code>	<code>win</code>	The rectangle.

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

Precondition

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

Definition at line 112 of file `approx/median.hh`.

References mln::data::median().

Referenced by median().

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

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

Parameters

<i>in</i>	<i>input</i>	The image to be filtered.
<i>in</i>	<i>win</i>	The disk.

The approximation is based on a vertical median and an horizontal median an two diagonal median.

Precondition

input and *output* have to be initialized.

Definition at line 132 of file approx/median.hh.

References 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::octagon2d & win) [inline]`

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

Parameters

<i>in</i>	<i>input</i>	The image to be filtered.
<i>in</i>	<i>win</i>	The octagon.

The approximation is based on a vertical median and an horizontal median an two diagonal median.

Precondition

input and *output* have to be initialized.

Definition at line 159 of file approx/median.hh.

References median().

9.30 mln::data::approx::impl Namespace Reference

Implementation namespace of [data::approx](#) namespace.

9.30.1 Detailed Description

Implementation namespace of [data::approx](#) namespace.

9.31 mln::data::impl Namespace Reference

Implementation namespace of data namespace.

Namespaces

- namespace [generic](#)

Generic implementation namespace of data namespace.

Functions

- `template<typename I , typename J >`
`void paste_without_localization_fast (const Image< I > &input_, Image< J > &output_)`
Paste data to an image without using localization. Performs a point-wise copy.
- `template<typename I , typename J >`
`void paste_without_localization_fastest (const Image< I > &input_, Image< J > &output_)`
Paste data to an image without using localization. Performs a one-block memory copy.
- `template<typename I , typename J >`
`void paste_without_localization_lines (const Image< I > &input_, Image< J > &output_)`
Paste data to an image without using localization. Performs a line-per-line memory copy.
- `template<typename V , typename I >`
`mln::trait::ch_value< I, V >::ret stretch (const V &v, const Image< I > &input)`
Generic implementation of [data::stretch](#).
- `template<typename I , typename F >`
`void transform_inplace_lowq (Image< I > &input_, const Function_v2v< F > &f_)`
Specialized implementation.
- `template<typename A , typename I >`
`A::result update_fastest (Accumulator< A > &a_, const Image< I > &input_)`
Fastest implementation of [data::update](#).

9.31.1 Detailed Description

Implementation namespace of data namespace.

9.31.2 Function Documentation

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

Paste data to an image without using localization. Performs a point-wise copy.

`input` and `output` must have both the following properties:

- `mln::trait::image::value_alignment::with_grid`
- `mln::trait::image::value_storage::one_block`
- `mln::trait::image::value_access::direct`
- `mln::trait::image::ext_domain::some`

They must also fulfill the following conditions:

- Same domain size.

Definition at line 220 of file `paste_without_localization.hh`.

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

Paste data to an image without using localization. Performs a one-block memory copy.

`input` and `output` must have both the following properties:

- `mln::trait::image::value_alignment::with_grid`
- `mln::trait::image::value_storage::one_block`
- `mln::trait::image::value_access::direct`
- `mln::trait::image::ext_domain::some`

They must also fulfill the following conditions:

- Same border size.
- Same domain size.
- Same value type.

Definition at line 142 of file `paste_without_localization.hh`.

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

Paste data to an image without using localization. Performs a line-per-line memory copy.

`input` and `output` must have both the following properties:

- `mln::trait::image::value_alignment::with_grid`
- `mln::trait::image::value_storage::one_block`
- `mln::trait::image::value_access::direct`
- `mln::trait::image::ext_domain::some`

They must also fulfill the following conditions:

- Same domain size.
- Same value type.

Definition at line 179 of file `paste_without_localization.hh`.

9.31.2.4 `template<typename V, typename I> mln::trait::ch_value< I, V >::ret mln::data::impl::stretch (const V & v, const Image< I > & input) [inline]`

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

Parameters

<code>in</code>	<code>v</code>	A value to set the output value type.
<code>in</code>	<code>input</code>	The input image.

Returns

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

Definition at line 83 of file `stretch.hh`.

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

Referenced by `mln::data::stretch()`.

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

Specialized implementation.

Definition at line 203 of file `transform_inplace.hh`.

9.31.2.6 `template<typename A, typename I> A ::result mln::data::impl::update_fastest (Accumulator< A > & a_, const Image< I > & input_) [inline]`

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

Parameters

<code>in</code>	<code>a_</code>	The accumulator.
<code>in</code>	<code>input_</code>	The input image.

Returns

The accumulator result.

Definition at line 129 of file `update.hh`.

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

Generic implementation namespace of data namespace.

Functions

- `template<typename I, typename J>`
`void fill_with_image (Image< I > &ima_, const Image< J > &data_)`
Generic implementation.
- `template<typename I, typename V>`
`void fill_with_value (Image< I > &ima_, const V &val)`
Fill the whole image `ima` with the single value `v`.
- `template<typename I, typename J>`
`void paste (const Image< I > &input_, Image< J > &output_)`
Generic implementation of [data::paste](#).
- `template<typename I, typename F>`
`mln::trait::ch_value< I,`
`typename F::result >::ret transform (const Image< I > &input_, const Function_v2v< F > &f_)`
Generic implementation of [data::transform](#).
- `template<typename I1, typename I2, typename F>`
`mln::trait::ch_value< I1,`
`typename F::result >::ret transform (const Image< I1 > &input1_, const Image< I2 > &input2_, const`
`Function_vv2v< F > &f_)`

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

- `template<typename I , typename F >`
`void transform_inplace (Image< I > &ima_, const Function_v2v< F > &f_)`

Generic implementation of `transform_inplace`.

- `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 A , typename I >`
`A::result update (Accumulator< A > &a_, const Image< I > &input_)`

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

9.32.1 Detailed Description

Generic implementation namespace of data namespace.

9.32.2 Function Documentation

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

Generic implementation.

Parameters

<code>in, out</code>	<code>ima_</code>	The image to be filled.
<code>in</code>	<code>data_</code>	The image.

Definition at line 100 of file `fill_with_image.hh`.

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

Fill the whole image `ima` with the single value `v`.

Parameters

<code>in, out</code>	<code>ima_</code>	The image to be filled.
<code>in</code>	<code>val</code>	The value to assign to all sites.

Precondition

`ima` has to be initialized.

Definition at line 103 of file `fill_with_value.hh`.

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

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

Parameters

<code>in</code>	<code>input_</code>	The input image providing pixels values.
<code>in, out</code>	<code>output_</code>	The image in which values are assigned.

Definition at line 111 of file paste.hh.

```
9.32.2.4  template<typename I , typename F > mln::trait::ch_value< I , typename F ::result >::ret
          mln::data::impl::generic::transform ( const Image< I > & input_, const Function_v2v< F > & f_ )
```

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

Parameters

<i>in</i>	<i>input_</i>	The input image.
<i>in</i>	<i>f_</i>	The function.

Definition at line 137 of file data/transform.hh.

References `mln::initialize()`.

```
9.32.2.5  template<typename I1 , typename I2 , typename F > mln::trait::ch_value< I1 , typename F ::result >::ret
          mln::data::impl::generic::transform ( const Image< I1 > & input1_, const Image< I2 > & input2_, const
          Function_vv2v< F > & f_ )
```

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

Parameters

<i>in</i>	<i>input1_</i>	The 1st input image.
<i>in</i>	<i>input2_</i>	The 2nd input image.
<i>in</i>	<i>f_</i>	The function.

Definition at line 166 of file data/transform.hh.

References `mln::initialize()`.

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

Generic implementation of `transform_inplace`.

Parameters

<i>in, out</i>	<i>ima_</i>	The image to be transformed.
<i>in</i>	<i>f_</i>	The function.

Definition at line 149 of file transform_inplace.hh.

```
9.32.2.7  template<typename I1 , typename I2 , typename F > void mln::data::impl::generic::transform_inplace ( Image< I1 >
          & ima_, const Image< I2 > & aux_, const Function_vv2v< F > & f_ )
```

Generic implementation of `transform_inplace`.

Parameters

<i>in</i>	<i>ima_</i>	The image to be transformed.
<i>in</i>	<i>aux_</i>	The auxiliary image.
<i>in</i>	<i>f_</i>	The function.

Definition at line 176 of file transform_inplace.hh.

9.32.2.8 `template<typename A , typename I > A ::result mln::data::impl::generic::update (Accumulator< A > & a_, const Image< I > & input_) [inline]`

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

Parameters

<code>in</code>	<code>a_</code>	The accumulator.
<code>in</code>	<code>input_</code>	The input image.

Returns

The accumulator result.

Definition at line 100 of file `update.hh`.

9.33 mln::data::naive Namespace Reference

Namespace of image processing routines related to pixel levels with naive approach.

Namespaces

- namespace [impl](#)
Implementation namespace of [data::naive](#) namespace.

Functions

- `template<typename I , typename W , typename O > void median (const Image< I > &input, const Window< W > &win, Image< O > &output)`
Compute in output the median filter of image input by the window win.

9.33.1 Detailed Description

Namespace of image processing routines related to pixel levels with naive approach.

9.33.2 Function Documentation

9.33.2.1 `template<typename I , typename W , typename O > void mln::data::naive::median (const Image< I > & input, const Window< W > & win, Image< O > & output) [inline]`

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

Parameters

<code>in</code>	<code>input</code>	The image to be filtered.
<code>in</code>	<code>win</code>	The window.
<code>in, out</code>	<code>output</code>	The output image.

This is a NAIVE version for test / comparison purpose so do NOT use it.

Precondition

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

See Also

[mln::data::median](#)

Definition at line 99 of file `naive/median.hh`.

9.34 mln::data::naive::impl Namespace Reference

Implementation namespace of [data::naive](#) namespace.

9.34.1 Detailed Description

Implementation namespace of [data::naive](#) namespace.

9.35 mln::debug Namespace Reference

Namespace of routines that help to debug.

Namespaces

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

Functions

- `template<typename I , typename G , typename F >`
`void draw_graph (Image< I > &ima, const p_vertices< G, F > &pv, typename I::value vcolor, typename I::value ecolord)`
Draw an image `ima` from a [mln::p_vertices](#) `pv`, with value `vcolor` for vertices, value `ecolor` for edges and 0 for the background.
- `template<typename I , typename G , typename F , typename V , typename E >`
`void draw_graph (Image< I > &ima, const p_vertices< G, F > &pv, const Function< V > &vcolor_f_, const Function< E > &ecolor_f_)`
Draw an image `ima` from a [mln::p_vertices](#) `pv`.
- `template<typename I , typename G , typename F , typename V , typename E >`
`void draw_graph (Image< I > &ima, const p_vertices< util::line_graph< G >, F > &pv, const Function< V > &vcolor_f_, const Function< E > &ecolor_f_)`
Draw an image `ima` from a [mln::p_vertices](#) `pv`.
- `std::string filename (const std::string &filename, int id)`
Constructs and returns a formatted output file name.
- `template<typename T >`
`const T & format (const T &v)`
Default version for formatting a value is a no-op.
- `char format (bool v)`
Format a Boolean to print it nicely: `"|"` for true and `"-"` for false.
- `signed short format (signed char v)`

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

 - unsigned short `format` (unsigned char `v`)

Format an unsigned char to print it properly, i.e., like an integer value.
- template<typename I >
void `iota` (Image< I > &input, unsigned base_index)
- template<typename I >
mln::trait::concrete< I >::ret `mosaic` (const util::array< I > &input, unsigned n_horizontal, const typename I::value &bg)

Create a single image from an array of image.
- template<typename I >
void `println` (const Image< I > &input)

Print the image `input` on the standard output.
- 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_with_border` (const Image< I > &input)

Print the image `input` on the standard output.
- void `put_word` (image2d< char > &inout, const point2d &word_start, const std::string &word)

Put the word starting at location `word_start` in the image `inout`.
- template<typename I >
image2d< typename I::value > `slices_2d` (const Image< I > &input, unsigned n_horizontal, unsigned n_vertical, const typename I::value &bg)

Create a 2D image of the slices of the 3D image `input`.
- template<typename I >
image2d< typename I::value > `slices_2d` (const Image< I > &input, float ratio_hv, const typename I::value &bg)

Create a 2D image of the slices of the 3D image `input`.
- template<typename I, typename J >
mln::trait::ch_value< I,
value::rgb8 >::ret `superpose` (const Image< I > &input_, const Image< J > &object_, const value::rgb8 &object_color)

Superpose two images.
- template<typename I, typename J >
mln::trait::ch_value< I,
value::rgb8 >::ret `superpose` (const Image< I > &input, const Image< J > &object)
- template<typename I >
void `z_order` (Image< I > &input)

9.35.1 Detailed Description

Namespace of routines that help to debug.

9.35.2 Function Documentation

- 9.35.2.1 template<typename I, typename G, typename F > void mln::debug::draw_graph (Image< I > & *ima*, const p_vertices< G, F > & *p_v*, typename I::value *vcolor*, typename I::value *ecolor*) [inline]

Draw an image *ima* from a mln::p_vertices *p_v*, with value *vcolor* for vertices, value *ecolor* for edges and 0 for the background.

Definition at line 142 of file draw_graph.hh.

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

Referenced by mln::make_debug_graph_image().

9.35.2.2 `template<typename I, typename G, typename F, typename V, typename E > void mln::debug::draw_graph (Image< I > & ima, const p_vertices< G, F > & pv, const Function< V > & vcolor_f_, const Function< E > & ecolor_f_) [inline]`

Draw an image `ima` from a [mln::p_vertices](#) `pv`.

Colors for vertices are defined through `vcolor_f_`. Colors for edges are defined through `ecolor_f_`.

Definition at line 173 of file `draw_graph.hh`.

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

9.35.2.3 `template<typename I, typename G, typename F, typename V, typename E > void mln::debug::draw_graph (Image< I > & ima, const p_vertices< util::line_graph< G >, F > & pv, const Function< V > & vcolor_f_, const Function< E > & ecolor_f_) [inline]`

Draw an image `ima` from a [mln::p_vertices](#) `pv`.

Colors for vertices are defined through `vcolor_f_`. Colors for edges are defined through `ecolor_f_`.

Definition at line 211 of file `draw_graph.hh`.

References `mln::p_line2d::begin()`, `mln::p_line2d::end()`, `mln::p_vertices< G, F >::graph()`, and `mln::draw::line()`.

9.35.2.4 `std::string mln::debug::filename (const std::string & filename, int id = -1) [inline]`

Constructs and returns a formatted output file name.

The file name is formatted as follow:

`filename_prefix_id_filename`

Where:

- `filename_prefix` can be set through the global variable `debug::internal::filename_prefix`.

`postfix_id` is autoincremented by default. Its value can be forced.

- `filename` is the given filename

Definition at line 86 of file `filename.hh`.

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

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

Definition at line 64 of file `format.hh`.

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

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

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

Definition at line 71 of file `format.hh`.

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

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

Definition at line 78 of file `format.hh`.

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

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

Definition at line 85 of file format.hh.

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

Fill the image `input` with successive values.

```
\param[in,out] input The image in which values are
assigned.
```

Definition at line 88 of file debug/iota.hh.

References `iota()`.

Referenced by `iota()`.

9.35.2.10 template<typename I> mln::trait::concrete<I>::ret mln::debug::mosaic (const util::array<I> & input, unsigned n_horizontal, const typename I::value & bg) [inline]

Create a single image from an array of image.

The size of the output image is defined by:

`width = n_horizontal * max(input[i].ncols())` `height = (input.size() / n_horizontal) * max(input[i].nrows())`

Returns

a single image where all the input images are displayed as a mosaic.

Definition at line 77 of file mosaic.hh.

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

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

Print the image `input` on the standard output.

Definition at line 87 of file println.hh.

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

Referenced by `println()`.

9.35.2.12 template<typename I> void mln::debug::println (const std::string & msg, const Image<I> & input)

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

Definition at line 98 of file println.hh.

References `println()`.

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

Print the image `input` on the standard output.

Definition at line 79 of file println_with_border.hh.

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

9.35.2.14 `void mln::debug::put_word (image2d< char > & inout, const point2d & word_start, const std::string & word)`
`[inline]`

Put the `word` starting at location `word_start` in the image `inout`.

Definition at line 55 of file `put_word.hh`.

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

9.35.2.15 `template<typename I> image2d< typename I::value> mln::debug::slices_2d (const Image< I> & input,`
`unsigned n_horizontal, unsigned n_vertical, const typename I::value & bg) [inline]`

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

Definition at line 79 of file `slices_2d.hh`.

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

Referenced by `slices_2d()`.

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

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

Definition at line 164 of file `slices_2d.hh`.

References `slices_2d()`.

9.35.2.17 `template<typename I, typename J> mln::trait::ch_value< I, value::rgb8>::ret mln::debug::superpose (const`
`Image< I> & input_, const Image< J> & object_, const value::rgb8 & object_color)`

Superpose two images.

Parameters

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

Precondition

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

Returns

A color image.

Definition at line 79 of file `debug/superpose.hh`.

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

Referenced by `superpose()`.

9.35.2.18 `template<typename I, typename J> mln::trait::ch_value< I, value::rgb8>::ret mln::debug::superpose (const`
`Image< I> & input, const Image< J> & object)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 106 of file debug/superpose.hh.

References `mln::literal::red`, and `superpose()`.

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

Fill the image `input` with Z-order (curve) values.

`\param[in,out]` `input` The image in which values are assigned.

Reference: [http://en.wikipedia.org/wiki/Z-order_\(curve\)](http://en.wikipedia.org/wiki/Z-order_(curve))

Definition at line 142 of file `z_order.hh`.

9.36 mln::debug::impl Namespace Reference

Implementation namespace of debug namespace.

9.36.1 Detailed Description

Implementation namespace of debug namespace.

9.37 mln::def Namespace Reference

Namespace for core definitions.

Typedefs

- typedef short `coord`
Definition of the default coordinate type: 'short'.
- typedef float `coordf`
Definition of the floating coordinate type.

Enumerations

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

9.37.1 Detailed Description

Namespace for core definitions.

9.37.2 Typedef Documentation

9.37.2.1 `typedef short mln::def::coord`

Definition of the default coordinate type: 'short'.

Definition at line 43 of file `coord.hh`.

9.37.2.2 `typedef float mln::def::coordf`

Definition of the floating coordinate type.

Definition at line 41 of file coordf.hh.

9.37.3 Enumeration Type Documentation

9.37.3.1 `anonymous enum`

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

Definition at line 43 of file low_quant_nbits.hh.

9.38 `mln::display` Namespace Reference

Namespace of routines that help to display images.

Namespaces

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

9.38.1 Detailed Description

Namespace of routines that help to display images.

9.39 `mln::display::impl` Namespace Reference

Implementation namespace of display namespace.

Namespaces

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

9.39.1 Detailed Description

Implementation namespace of display namespace.

9.40 `mln::display::impl::generic` Namespace Reference

Generic implementation namespace of display namespace.

9.40.1 Detailed Description

Generic implementation namespace of display namespace.

9.41 mln::doc Namespace Reference

The namespace `mln::doc` is only for documentation purpose.

Classes

- struct `Accumulator`
Documentation class for `mln::Accumulator`.
- struct `Box`
Documentation class for `mln::Box`.
- struct `Dpoint`
Documentation class for `mln::Dpoint`.
- struct `Fastest_Image`
Documentation class for the concept of images that have the speed property set to "fastest".
- struct `Generalized_Pixel`
Documentation class for `mln::Generalized_Pixel`.
- struct `Image`
Documentation class for `mln::Image`.
- struct `Iterator`
Documentation class for `mln::Iterator`.
- struct `Neighborhood`
Documentation class for `mln::Neighborhood`.
- struct `Object`
Documentation class for `mln::Object`.
- struct `Pixel_Iterator`
Documentation class for `mln::Iterator`.
- struct `Point_Site`
Documentation class for `mln::Point_Site`.
- struct `Site_Iterator`
Documentation class for `mln::Site_Iterator`.
- struct `Site_Set`
Documentation class for `mln::Site_Set`.
- struct `Value_Iterator`
Documentation class for `mln::Value_Iterator`.
- struct `Value_Set`
Documentation class for `mln::Value_Set`.
- struct `Weighted_Window`
Documentation class for `mln::Weighted_Window`.
- struct `Window`
Documentation class for `mln::Window`.

9.41.1 Detailed Description

The namespace `mln::doc` is only for documentation purpose. Since concepts are not yet part of the C++ Standard, they are not explicitly expressed in code. Their documentation is handled by their respective ghost class, located in this namespace.

Warning

The ghost classes located in `mln::doc` should not be used by the client.

9.42 mln::draw Namespace Reference

Namespace of drawing routines.

Functions

- `template<typename I, typename B >
void box (Image< I > &ima, const Box< B > &b, const typename I::value &v)`
- `template<typename I, typename B >
void box_plain (Image< I > &ima, const Box< B > &b, const typename I::value &v)`
- `template<typename I >
void dashed_line (Image< I > &ima, const typename I::psite &beg, const typename I::psite &end, const
typename I::value &v)`
- `template<typename I >
void line (Image< I > &ima, const typename I::psite &beg, const typename I::psite &end, const typename
I::value &v)`
- `template<typename I >
void plot (Image< I > &ima, const typename I::point &p, const typename I::value &v)`
- `template<typename I >
void polygon (Image< I > &ima, const p_array< typename I::site > &par, const typename I::value &v, un-
signed output_ratio)`
- `template<typename I, typename S >
void site_set (Image< I > &ima, const Site_Set< S > &s, const typename I::value &v, unsigned output_
ratio=1)`

9.42.1 Detailed Description

Namespace of drawing routines.

9.42.2 Function Documentation

9.42.2.1 `template<typename I, typename B > void mln::draw::box (Image< I > & ima, const Box< B > & b, const
typename I::value & v)` `[inline]`

Draw a box at value *v* in image *ima*

```
\param[in,out] ima The image to be drawn.
\param[in] b the box to draw.
\param[in] v The value to assign to all drawn pixels.

\pre \p ima has to be initialized.
\pre \p ima has \p beg.
\pre \p ima has \p end.
```

Definition at line 72 of file draw/box.hh.

References [line\(\)](#).

9.42.2.2 `template<typename I, typename B > void mln::draw::box_plain (Image< I > & ima, const Box< B > & b, const
typename I::value & v)` `[inline]`

Draw a plain box at value *v* in image *ima*

```
\param[in,out] ima The image to be drawn.
\param[in] b the box to draw.
\param[in] v The value to assign to all drawn pixels.
```

```
\pre \p ima has to be initialized.
\pre \p ima has \p beg.
\pre \p ima has \p end.
```

Definition at line 71 of file box_plain.hh.

References mln::data::fill().

Referenced by mln::debug::draw_graph().

9.42.2.3 `template<typename I> void mln::draw::dashed_line (Image< I > & ima, const typename I::psite & beg, const typename I::psite & end, const typename I::value & v) [inline]`

Draw a dashed line at level *v* in image *ima* between the points *beg* and *end*.

Parameters

<i>in, out</i>	<i>ima</i>	The image to be drawn.
<i>in</i>	<i>beg</i>	The start point to drawn dashed_line.
<i>in</i>	<i>end</i>	The end point to drawn dashed_line.
<i>in</i>	<i>v</i>	The value to assign to all drawn pixels.

Precondition

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

Definition at line 91 of file dashed_line.hh.

References mln::data::fill().

9.42.2.4 `template<typename I> void mln::draw::line (Image< I > & ima, const typename I::psite & beg, const typename I::psite & end, const typename I::value & v) [inline]`

Draw a line at level *v* in image *ima* between the points *beg* and *end*.

Parameters

<i>in, out</i>	<i>ima</i>	The image to be drawn.
<i>in</i>	<i>beg</i>	The start point to drawn line.
<i>in</i>	<i>end</i>	The end point to drawn line.
<i>in</i>	<i>v</i>	The value to assign to all drawn pixels.

Precondition

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

Definition at line 72 of file draw/line.hh.

References mln::data::paste().

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

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

Plot a point at level *v* in image *ima*

```

\param[in,out] ima The image to be drawn.
\param[in] p The point to be plotted.
\param[in] v The value to assign to all drawn pixels.

\pre \p ima has to be initialized.
\pre \p ima has \p p.

```

9.42.2.6 `template<typename I> void mln::draw::polygon (Image< I> & ima, const p_array< typename I::site> & par, const typename I::value & v, unsigned output_ratio)`

Draw a polygon at level `v` in image `ima`.

```

\param[in,out] ima The image to be drawn.
\param[in] par The polygon site set.
\param[in] v The value to assign to all drawn pixels.

\pre \p ima has to be initialized.

```

Definition at line 70 of file `polygon.hh`.

References `line()`.

9.42.2.7 `template<typename I, typename S> void mln::draw::site_set (Image< I> & ima, const Site_Set< S> & s, const typename I::value & v, unsigned output_ratio = 1)`

Draw a sites with value `v` in image `ima`

```

\param[in,out] ima The image to be drawn.
\param[in] b the site set to draw.
\param[in] v The value to assign to all drawn pixels.
\param[in] output_ratio size ratio between output image and the
image from which the bboxes were calculated.

\pre \p s is included in \p ima domain.

```

Definition at line 65 of file `mln/draw/site_set.hh`.

9.43 `mln::estim` Namespace Reference

Namespace of estimation materials.

Functions

- `template<typename I>`
`mln::value::props< typename`
`I::value>::sum mean (const Image< I> &input)`
Compute the mean value of the pixels of image `input`.
- `template<typename S, typename I, typename M>`
`void mean (const Image< I> &input, M &result)`
Compute the mean value of the pixels of image `input`.
- `template<typename I>`
`void min_max (const Image< I> &input, typename I::value &min, typename I::value &max)`
Compute the min and max values of the pixels of image `input`.
- `template<typename I>`
`mln::value::props< typename`
`I::value>::sum sum (const Image< I> &input)`
Compute the sum value of the pixels of image `input`.

- `template<typename I, typename S >`
`void sum (const Image< I > &input, S &result)`
Compute the sum value of the pixels of image `input`.

9.43.1 Detailed Description

Namespace of estimation materials.

9.43.2 Function Documentation

9.43.2.1 `template<typename I > mln::value::props< typename I::value >::sum mln::estim::mean (const Image< I > & input) [inline]`

Compute the mean value of the pixels of image `input`.

Parameters

<code>in</code>	<code>input</code>	The image.
-----------------	--------------------	------------

Returns

The mean value.

Definition at line 68 of file `estim/mean.hh`.

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

9.43.2.2 `template<typename S, typename I, typename M > void mln::estim::mean (const Image< I > & input, M & result) [inline]`

Compute the mean value of the pixels of image `input`.

Parameters

<code>in</code>	<code>input</code>	The image.
<code>out</code>	<code>result</code>	The mean value.

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

Definition at line 76 of file `estim/mean.hh`.

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

9.43.2.3 `template<typename I > void mln::estim::min_max (const Image< I > & input, typename I::value & min, typename I::value & max) [inline]`

Compute the min and max values of the pixels of image `input`.

Parameters

<code>in</code>	<code>input</code>	The image.
<code>out</code>	<code>min</code>	The minimum pixel value of <code>input</code> .
<code>out</code>	<code>max</code>	The maximum pixel value of <code>input</code> .

Definition at line 61 of file `estim/min_max.hh`.

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

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

9.43.2.4 `template<typename I> mln::value::props< typename I::value >::sum mln::estim::sum (const Image< I> & input)`
`[inline]`

Compute the sum value of the pixels of image `input`.

Parameters

<code>in</code>	<code>input</code>	The image.
-----------------	--------------------	------------

Returns

The sum value.

Definition at line 67 of file `estim/sum.hh`.

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

9.43.2.5 `template<typename I, typename S> void mln::estim::sum (const Image< I> & input, S & result)` `[inline]`

Compute the sum value of the pixels of image `input`.

Parameters

<code>in</code>	<code>input</code>	The image.
<code>out</code>	<code>result</code>	The sum value.

Definition at line 75 of file `estim/sum.hh`.

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

9.44 mln::extension Namespace Reference

Namespace of extension tools.

Functions

- `template<typename I, typename W>`
`void adjust (const Image< I> &ima, const Window< W> &win)`
Adjust the domain extension of image `ima` with the size of the window `win`.
- `template<typename I, typename W>`
`void adjust (const Image< I> &ima, const Weighted_Window< W> &wwin)`
Adjust the domain extension of image `ima` with the size of the weighted window `wwin`.
- `template<typename I, 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>`
`void adjust (const Image< I> &ima, unsigned delta)`
Adjust the domain extension of image `ima` with the size `delta`.
- `template<typename I, typename W>`
`void adjust_duplicate (const Image< I> &ima, const Window< W> &win)`
Adjust then duplicate.

- `template<typename I , typename W >`
`void adjust_fill (const Image< I > &ima, const Window< W > &win, const typename I::value &val)`
Adjust then fill.
- `template<typename I >`
`void duplicate (const Image< I > &ima)`
*Assign the contents of the domain extension by duplicating the values of the inner boundary of image *ima*.*
- `template<typename I >`
`void fill (const Image< I > &ima, const typename I::value &val)`

9.44.1 Detailed Description

Namespace of extension tools.

9.44.2 Function Documentation

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

Adjust the domain extension of image *ima* with the size of the window *win*.

Definition at line 89 of file `extension/adjust.hh`.

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

Referenced by `adjust_duplicate()`, and `adjust_fill()`.

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

Adjust the domain extension of image *ima* with the size of the weighted window *wwin*.

Definition at line 97 of file `extension/adjust.hh`.

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

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

Adjust the domain extension of image *ima* with the size of the neighborhood *nbh*.

Definition at line 105 of file `extension/adjust.hh`.

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

9.44.2.4 `template<typename I > void mln::extension::adjust (const Image< I > & ima, unsigned delta)`

Adjust the domain extension of image *ima* with the size *delta*.

Definition at line 113 of file `extension/adjust.hh`.

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

Adjust then duplicate.

Definition at line 70 of file `adjust_duplicate.hh`.

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

9.44.2.6 `template<typename I, typename W> void mln::extension::adjust_fill (const Image< I > & ima, const Window< W > & win, const typename I::value & val)`

Adjust then fill.

Definition at line 72 of file `adjust_fill.hh`.

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

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

Assign the contents of the domain extension by duplicating the values of the inner boundary of image `ima`.

Definition at line 58 of file `extension/duplicate.hh`.

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

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

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

Fill the domain extension of image `ima` with the single value `v`.

Parameters

<code>in, out</code>	<code>ima</code>	The image whose domain extension is to be filled.
<code>in</code>	<code>val</code>	The value to assign.

Precondition

`ima` has to be initialized.

Definition at line 175 of file `extension/fill.hh`.

Referenced by `adjust_fill()`.

9.45 mln::fun Namespace Reference

Namespace of functions.

Namespaces

- namespace [access](#)
Namespace for access functions.
- namespace [i2v](#)
Namespace of integer-to-value functions.
- namespace [n2v](#)
Namespace of functions from nil to value.
- namespace [p2b](#)
Namespace of functions from point to boolean.
- namespace [p2p](#)
Namespace of functions from grid point to grid point.
- namespace [p2v](#)
Namespace of functions from point to value.
- namespace [stat](#)

Namespace of statistical functions.

- namespace [v2b](#)

Namespace of functions from value to logic value.

- namespace [v2i](#)

Namespace of value-to-integer functions.

- namespace [v2v](#)

Namespace of functions from value to value.

- namespace [v2w2v](#)

Namespace of bijective functions.

- namespace [v2w_w2v](#)

Namespace of functions from value to value.

- namespace [vv2b](#)

Namespace of functions from value to value.

- namespace [vv2v](#)

Namespace of functions from a couple of values to a value.

- namespace [x2p](#)

Namespace of functions from point to value.

- namespace [x2v](#)

Namespace of functions from vector to value.

- namespace [x2x](#)

Namespace of functions from vector to vector.

Classes

- struct [from_accu](#)

Wrap an accumulator into a function.

9.45.1 Detailed Description

Namespace of functions. Forward declarations.

`fun::i2v::array`

Forward declaration.

9.46 mln::fun::access Namespace Reference

Namespace for access functions.

9.46.1 Detailed Description

Namespace for access functions.

9.47 mln::fun::i2v Namespace Reference

Namespace of integer-to-value functions.

Functions

- `template<typename T>`
`std::ostream & operator<< (std::ostream &ostr, const array< T > &a)`
Operator<<.

9.47.1 Detailed Description

Namespace of integer-to-value functions.

9.47.2 Function Documentation

9.47.2.1 `template<typename T> std::ostream & mln::fun::i2v::operator<< (std::ostream & ostr, const array< T > &a)`

Operator<<.

Definition at line 353 of file fun/i2v/array.hh.

9.48 mln::fun::n2v Namespace Reference

Namespace of functions from nil to value.

Classes

- struct `white_gaussian`
Generate a White Gaussian Noise.

9.48.1 Detailed Description

Namespace of functions from nil to value.

9.49 mln::fun::p2b Namespace Reference

Namespace of functions from point to boolean.

Classes

- struct `antilogy`
A `p2b` function always returning `false`.
- struct `tautology`
A `p2b` function always returning `true`.

9.49.1 Detailed Description

Namespace of functions from point to boolean.

9.50 mln::fun::p2p Namespace Reference

Namespace of functions from grid point to grid point.

9.50.1 Detailed Description

Namespace of functions from grid point to grid point.

9.51 mln::fun::p2v Namespace Reference

Namespace of functions from point to value.

9.51.1 Detailed Description

Namespace of functions from point to value.

9.52 mln::fun::stat Namespace Reference

Namespace of statistical functions.

9.52.1 Detailed Description

Namespace of statistical functions.

9.53 mln::fun::v2b Namespace Reference

Namespace of functions from value to logic value.

Classes

- struct [lnot](#)
Functor computing logical-not on a value.
- struct [threshold](#)
Threshold function.

9.53.1 Detailed Description

Namespace of functions from value to logic value.

9.54 mln::fun::v2i Namespace Reference

Namespace of value-to-integer functions.

9.54.1 Detailed Description

Namespace of value-to-integer functions.

9.55 mln::fun::v2v Namespace Reference

Namespace of functions from value to value.

Classes

- class [ch_function_value](#)
Wrap a function [v2v](#) and convert its result to another type.
- struct [component](#)
Functor that accesses the i -th component of a value.
- struct [l1_norm](#)
 $L1$ -norm.
- struct [l2_norm](#)
 $L2$ -norm.
- struct [linear](#)
*Linear function. $f(v) = a * v + b$. V is the type of input values; T is the type used to compute the result; R is the result type.*
- struct [linfty_norm](#)
 L -infty norm.
- struct [rgb8_to_rgn](#)
Convert a $rgb8$ value to a rgn , $n < 8$.

Variables

- [f_hsi_to_rgb_3x8_t](#) [f_hsi_to_rgb_3x8](#)
Global variable.
- [f_hsl_to_rgb_3x8_t](#) [f_hsl_to_rgb_3x8](#)
Global variables.
- [f_rgb_to_hsi_f_t](#) [f_rgb_to_hsi_f](#)
Global variables.
- [f_rgb_to_hsl_f_t](#) [f_rgb_to_hsl_f](#)
Global variables.

9.55.1 Detailed Description

Namespace of functions from value to value.

9.55.2 Variable Documentation

9.55.2.1 [f_hsi_to_rgb_3x8_t](#) [mln::fun::v2v::f_hsi_to_rgb_3x8](#)

Global variable.

Definition at line 79 of file [hsi_to_rgb.hh](#).

9.55.2.2 [f_hsl_to_rgb_3x8_t](#) [mln::fun::v2v::f_hsl_to_rgb_3x8](#)

Global variables.

Definition at line 95 of file [hsl_to_rgb.hh](#).

9.55.2.3 f_rgb_to_hsi_f_t mln::fun::v2v::f_rgb_to_hsi_f

Global variables.

Definition at line 71 of file rgb_to_hsi.hh.

9.55.2.4 f_rgb_to_hsl_f_t mln::fun::v2v::f_rgb_to_hsl_f

Global variables.

Definition at line 79 of file rgb_to_hsl.hh.

9.56 mln::fun::v2w2v Namespace Reference

Namespace of bijective functions.

Classes

- struct [cos](#)
Cosinus bijective functor.

9.56.1 Detailed Description

Namespace of bijective functions.

9.57 mln::fun::v2w_w2v Namespace Reference

Namespace of functions from value to value.

Classes

- struct [l1_norm](#)
L1-norm.
- struct [l2_norm](#)
L2-norm.
- struct [linfty_norm](#)
L-infty norm.

9.57.1 Detailed Description

Namespace of functions from value to value.

9.58 mln::fun::vv2b Namespace Reference

Namespace of functions from value to value.

Classes

- struct [eq](#)
Functor computing equal between two values.
- struct [ge](#)
Functor computing "greater or equal than" between two values.
- struct [gt](#)
Functor computing "greater than" between two values.
- struct [implies](#)
Functor computing logical-implies between two values.
- struct [le](#)
Functor computing "lower or equal than" between two values.
- struct [lt](#)
Functor computing "lower than" between two values.

9.58.1 Detailed Description

Namespace of functions from value to value.

9.59 `mln::fun::vv2v` Namespace Reference

Namespace of functions from a couple of values to a value.

Classes

- struct [diff_abs](#)
A functor computing the diff_absimum of two values.
- struct [land](#)
Functor computing logical-and between two values.
- struct [land_not](#)
Functor computing logical and-not between two values.
- struct [lor](#)
Functor computing logical-or between two values.
- struct [lxor](#)
Functor computing logical-xor between two values.
- struct [max](#)
A functor computing the maximum of two values.
- struct [min](#)
A functor computing the minimum of two values.
- struct [vec](#)
A functor computing the vecimum of two values.

9.59.1 Detailed Description

Namespace of functions from a couple of values to a value.

9.60 `mln::fun::x2p` Namespace Reference

Namespace of functions from point to value.

Classes

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

9.60.1 Detailed Description

Namespace of functions from point to value.

9.61 mln::fun::x2v Namespace Reference

Namespace of functions from vector to value.

Classes

- struct [bilinear](#)
Represent a bilinear interolation of values from an underlying image.
- struct [trilinear](#)
Represent a trilinear interolation of values from an underlying image.

9.61.1 Detailed Description

Namespace of functions from vector to value.

9.62 mln::fun::x2x Namespace Reference

Namespace of functions from vector to vector.

Classes

- struct [composed](#)
Represent a composition of two transformations.
- struct [linear](#)
Represent a linear interolation of values from an underlying image.
- struct [rotation](#)
Represent a rotation function.
- struct [translation](#)
Translation function-object.

9.62.1 Detailed Description

Namespace of functions from vector to vector.

9.63 mln::geom Namespace Reference

Namespace of all things related to geometry.

Namespaces

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

Classes

- class [complex_geometry](#)
A functor returning the sites of the faces of a complex where the locations of each 0-face is stored.

Functions

- `template<typename S >`
`box< typename S::site > bbox (const Site_Set< S > &pset)`
Compute the precise bounding box of a point set `pset`.
- `template<typename I >`
`box< typename I::site > bbox (const Image< I > &ima)`
Compute the precise bounding box of a point set `pset`.
- `template<typename W >`
`box< typename W::psite > bbox (const Window< W > &win)`
Compute the precise bounding box of a window `win`.
- `template<typename W >`
`box< typename W::psite > bbox (const Weighted_Window< W > &win)`
Compute the precise bounding box of a weighted window `win`.
- `template<typename I, typename W >`
`mln::trait::ch_value< I,`
`unsigned >::ret chamfer (const Image< I > &input_, const W &w_win_, unsigned max=mln_-`
`max(unsigned))`
Apply chamfer algorithm to a binary image.
- `template<typename W >`
`unsigned delta (const Window< W > &win)`
Compute the delta of a window `win`.
- `template<typename W >`
`unsigned delta (const Weighted_Window< W > &wwin)`
Compute the delta of a weighted window `wwin`.
- `template<typename N >`
`unsigned delta (const Neighborhood< N > &nbh)`
Compute the delta of a neighborhood `nbh`.
- `template<typename I >`
`I::site::coord max_col (const Image< I > &ima)`
Give the maximum column of an image.
- `template<typename B >`
`B::site::coord max_col (const Box< B > &b)`
Give the maximum col of an box 2d or 3d.
- `template<typename I >`
`I::site::coord max_ind (const Image< I > &ima)`
Give the maximum ind of an image.
- `template<typename I >`
`I::site::coord max_row (const Image< I > &ima)`
Give the maximum row of an image.
- `template<typename B >`
`B::site::coord max_row (const Box< B > &b)`

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

 - template<typename I >
I::site::coord [max_sli](#) (const [Image](#)< I > &ima)

Give the maximum sli of an image.
- std::pair< [complex_image](#)
< 2, [mln::space_2complex_geometry](#),
[algebra::vec](#)< 3, float >
>, [complex_image](#)
< 2, [mln::space_2complex_geometry](#),
float > > [mesh_corner_point_area](#) (const [p_complex](#)< 2, [space_2complex_geometry](#) > &mesh)

Compute the area "belonging" to normals at vertices.
- std::pair< [complex_image](#)
< 2, [mln::space_2complex_geometry](#),
float >, [complex_image](#)
< 2, [mln::space_2complex_geometry](#),
float > > [mesh_curvature](#) (const [p_complex](#)< 2, [space_2complex_geometry](#) > &mesh)

Compute the principal curvatures of a surface at vertices.
- [complex_image](#)
< 2, [mln::space_2complex_geometry](#),
[algebra::vec](#)< 3, float > > [mesh_normal](#) (const [p_complex](#)< 2, [space_2complex_geometry](#) > &mesh)

Compute normals at vertices.
- template<typename I >
I::site::coord [min_col](#) (const [Image](#)< I > &ima)

Give the minimum column of an image.
- template<typename B >
B::site::coord [min_col](#) (const [Box](#)< B > &b)

Give the minimum column of an box 2d or 3d.
- template<typename I >
I::site::coord [min_ind](#) (const [Image](#)< I > &ima)

Give the minimum ind of an image.
- template<typename I >
I::site::coord [min_row](#) (const [Image](#)< I > &ima)

Give the minimum row of an image.
- template<typename B >
B::site::coord [min_row](#) (const [Box](#)< B > &b)

Give the minimum row of an box 2d or 3d.
- template<typename I >
I::site::coord [min_sli](#) (const [Image](#)< I > &ima)

Give the minimum sli of an image.
- template<typename I >
unsigned [ncols](#) (const [Image](#)< I > &ima)

Give the number of columns of an image.
- template<typename B >
unsigned [ncols](#) (const [Box](#)< B > &b)

Give the number of cols of a box 2d or 3d.
- template<typename I >
unsigned [ninds](#) (const [Image](#)< I > &ima)

Give the number of inds of an image.
- template<typename I >
unsigned [nrows](#) (const [Image](#)< I > &ima)

Give the number of rows of an image.
- template<typename B >
unsigned [nrows](#) (const [Box](#)< B > &b)

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

- `template<typename I >`
`unsigned nsites (const Image< I > &input)`
Compute the number of sites of the image `input`.
- `template<typename I >`
`unsigned nslis (const Image< I > &ima)`
Give the number of slices of an image.
- `template<typename S >`
`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 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 I >`
`std::pair< typename I::site,`
`typename I::site > pmin_pmax (const Site_Iterator< I > &p)`
Compute the minimum and maximum points when browsing with iterator `p`.
- `template<typename I >`
`void pmin_pmax (const Site_Iterator< I > &p, typename I::site &pmin, typename I::site &pmax)`
Compute the minimum and maximum points, `pmin` and `max`, when browsing with iterator `p`.
- `template<typename I, typename Ext, 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 Ext >`
`mln::trait::concrete< I >::ret rotate (const Image< I > &input, double angle, const Ext &extension)`
- `template<typename I >`
`mln::trait::concrete< I >::ret rotate (const Image< I > &input, double angle)`
This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use `literal::zero` as default value for the extension.
- `template<typename B >`
`B rotate (const Box< B > &box_, double angle, const typename B::site &ref)`
Rotate a box.
- `template<typename B >`
`B rotate (const Box< B > &box, double angle)`
This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. The rotation center `ref` is set to `box.pcenter()`.
- `template<typename I, typename N >`
`mln::trait::concrete< I >::ret seeds2tiling (const Image< I > &ima_, const Neighborhood< N > &nbh)`
Take a labeled image `ima_` with seeds and extend them until creating tiles.
- `template<typename I, typename V, typename Ext, typename S >`
`mln::trait::concrete< I >::ret translate (const Image< I > &input, const algebra::vec< I::site::dim, V > &ref,`
`const Ext &extension, const Site_Set< S > &output_domain)`
Perform a translation from the center of an image.
- `template<typename I, typename V, typename Ext >`
`mln::trait::concrete< I >::ret translate (const Image< I > &input, const algebra::vec< I::site::dim, V > &ref,`
`const Ext &extension)`
- `template<typename I, 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 >`
`mln::trait::concrete< I >::ret vertical_symmetry (const Image< I > &input)`
Performs a vertical symmetry.

9.63.1 Detailed Description

Namespace of all things related to geometry. Namespace of essential things related to geometry.

9.63.2 Function Documentation

9.63.2.1 `template<typename S> box< typename S::site > mln::geom::bbox (const Site_Set< S > & pset) [inline]`

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

Definition at line 122 of file `geom/bbox.hh`.

Referenced by `bbox()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::registration::icp()`, `max_col()`, `max_row()`, `max_sli()`, `min_col()`, `min_row()`, `min_sli()`, `mln::debug::println()`, `mln::debug::println_with_border()`, and `rotate()`.

9.63.2.2 `template<typename I> box< typename I::site > mln::geom::bbox (const Image< I > & ima)`

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

Definition at line 133 of file `geom/bbox.hh`.

References `bbox()`.

9.63.2.3 `template<typename W> box< typename W::psite > mln::geom::bbox (const Window< W > & win)`

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

Definition at line 143 of file `geom/bbox.hh`.

References `mln::literal::origin`.

9.63.2.4 `template<typename W> box< typename W::psite > mln::geom::bbox (const Weighted_Window< W > & win)`

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

Definition at line 156 of file `geom/bbox.hh`.

References `bbox()`.

9.63.2.5 `template<typename I, typename W> mln::trait::ch_value< I, unsigned >::ret mln::geom::chamfer (const Image< I > & input_, const W & w_win_, unsigned max = mln_max(unsigned))`

Apply chamfer algorithm to a binary image.

Definition at line 114 of file `geom/chamfer.hh`.

9.63.2.6 `template<typename W> unsigned mln::geom::delta (const Window< W > & win)`

Compute the delta of a window `win`.

Definition at line 96 of file `delta.hh`.

Referenced by `mln::extension::adjust()`, and `delta()`.

9.63.2.7 `template<typename W> unsigned mln::geom::delta (const Weighted_Window< W > & wwin)`

Compute the delta of a weighted window `wwin`.

Definition at line 105 of file delta.hh.

References `delta()`.

9.63.2.8 `template<typename N > unsigned mln::geom::delta (const Neighborhood< N > & nbh)`

Compute the delta of a neighborhood `nbh`.

Definition at line 112 of file delta.hh.

References `delta()`.

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

Give the maximum column of an image.

Definition at line 56 of file max_col.hh.

References `bbox()`.

Referenced by `mln::io::magick::load()`, and `ncols()`.

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

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

Definition at line 67 of file max_col.hh.

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

Give the maximum ind of an image.

Definition at line 51 of file max_ind.hh.

Referenced by `ninds()`.

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

Give the maximum row of an image.

Definition at line 57 of file max_row.hh.

References `bbox()`.

Referenced by `mln::io::magick::load()`, and `nrows()`.

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

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

Definition at line 68 of file max_row.hh.

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

Give the maximum sli of an image.

Definition at line 53 of file max_sli.hh.

References `bbox()`.

Referenced by `nslis()`.

9.63.2.15 `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).”

Definition at line 249 of file `misc.hh`.

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

Referenced by `mesh_curvature()`.

9.63.2.16 `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

<code>in</code>	<code>mesh</code>	The surface (triangle mesh) on which the curvature is to be computed.
-----------------	-------------------	---

Definition at line 486 of file `misc.hh`.

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

9.63.2.17 `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.

Definition at line 161 of file `misc.hh`.

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

Referenced by `mesh_curvature()`.

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

Give the minimum column of an image.

Definition at line 57 of file min_col.hh.

References `bbox()`.

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

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

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

Definition at line 68 of file min_col.hh.

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

Give the minimum ind of an image.

Definition at line 51 of file min_ind.hh.

Referenced by `ninds()`.

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

Give the minimum row of an image.

Definition at line 60 of file min_row.hh.

References `bbox()`.

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

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

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

Definition at line 71 of file min_row.hh.

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

Give the minimum sli of an image.

Definition at line 53 of file min_sli.hh.

References `bbox()`.

Referenced by `nslis()`.

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

Give the number of columns of an image.

Definition at line 57 of file ncols.hh.

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

Referenced by `mln::labeling::impl::compute_fastest()`, `mln::subsampling::gaussian_subsampling()`, `mln::transform::hough()`, `ncols()`, `mln::io::magick::save()`, and `mln::subsampling::subsampling()`.

9.63.2.25 `template<typename B> unsigned mln::geom::ncols (const Box< B> & b)`

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

Definition at line 67 of file ncols.hh.

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

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

Give the number of inds of an image.

Definition at line 52 of file ninds.hh.

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

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

Give the number of rows of an image.

Definition at line 57 of file nrows.hh.

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

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

9.63.2.28 `template<typename B> unsigned mln::geom::nrows (const Box< B> & b)`

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

Definition at line 66 of file nrows.hh.

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

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

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

Definition at line 52 of file nsites.hh.

Referenced by `pmin_pmax()`.

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

Give the number of slices of an image.

Definition at line 53 of file nslis.hh.

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

9.63.2.31 `template<typename S> std::pair< typename S::site, typename S::site> mln::geom::pmin_pmax (const Site_Set< S> & s) [inline]`

Compute the minimum and maximum points of point set `s`.

Definition at line 157 of file pmin_pmax.hh.

References `nsites()`.

Referenced by `pmin_pmax()`.

9.63.2.32 `template<typename S > void mln::geom::pmin_pmax (const Site_Set< S > & s, typename S::site & pmin, typename S::site & pmax) [inline]`

Compute the minimum and maximum points, `pmin` and `max`, of point set `s`.

Definition at line 148 of file `pmin_pmax.hh`.

References `nsites()`.

9.63.2.33 `template<typename I > std::pair< typename I::site, typename I::site > mln::geom::pmin_pmax (const Site_Iterator< I > & p) [inline]`

Compute the minimum and maximum points when browsing with iterator `p`.

Definition at line 106 of file `pmin_pmax.hh`.

References `pmin_pmax()`.

9.63.2.34 `template<typename I > void mln::geom::pmin_pmax (const Site_Iterator< I > & p, typename I::site & pmin, typename I::site & pmax) [inline]`

Compute the minimum and maximum points, `pmin` and `max`, when browsing with iterator `p`.

Definition at line 84 of file `pmin_pmax.hh`.

9.63.2.35 `template<typename I , typename Ext , typename S > mln::trait::concrete< I >::ret mln::geom::rotate (const Image< I > & input, double angle, const Ext & extension, const Site_Set< S > & output_domain)`

Perform a rotation from the center of an image.

Parameters

<code>in</code>	<code>input</code>	An image.
<code>in</code>	<code>angle</code>	An angle in degrees.
<code>in</code>	<code>extension</code>	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.
<code>in</code>	<code>output_domain</code>	The domain of the output image. An invalid domain, causes the routine to use a domain large enough to display the whole original image.

Returns

An image with the same domain as `input`.

Definition at line 123 of file `rotate.hh`.

References `bbox()`, `mln::compose()`, `mln::extension::duplicate()`, `mln::initialize()`, `mln::mln_exact()`, `mln::literal::origin`, and `mln::data::paste()`.

Referenced by `rotate()`.

9.63.2.36 `template<typename I , typename Ext > mln::trait::concrete< I >::ret mln::geom::rotate (const Image< I > & input, double angle, const Ext & extension)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 185 of file `rotate.hh`.

References `rotate()`.

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

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use [literal::zero](#) as default value for the extension.

Definition at line 197 of file rotate.hh.

References [rotate\(\)](#), and [mln::literal::zero](#).

9.63.2.38 `template<typename B > B mln::geom::rotate (const Box< B > & box_, double angle, const typename B::site & ref)`

Rotate a box.

FIXME: the return type may be too generic and may lead to invalid covariance.

Definition at line 205 of file rotate.hh.

References [mln::compose\(\)](#), [mln::literal::origin](#), and [mln::accu::shape::bbox< P >::to_result\(\)](#).

9.63.2.39 `template<typename B > B mln::geom::rotate (const Box< B > & box, double angle)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. The rotation center *ref* is set to [box.pcenter\(\)](#).

Definition at line 252 of file rotate.hh.

References [rotate\(\)](#).

9.63.2.40 `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

<i>in</i> , <i>out</i>	<i>ima_</i>	The labeled image with seed.
<i>in</i>	<i>nbh</i>	The neighborhood to use on this algorithm.

Returns

A tiled image.

Precondition

ima_ has to be initialized.

Definition at line 136 of file seeds2tiling.hh.

References [mln::geom::impl::seeds2tiling\(\)](#).

9.63.2.41 `template<typename I , typename V , typename Ext , typename S > mln::trait::concrete< I >::ret mln::geom::translate (const Image< I > & input, const algebra::vec< I::site::dim, V > & ref, const Ext & extension, const Site_Set< S > & output_domain)`

Perform a translation from the center of an image.

Parameters

<code>in</code>	<i>input</i>	An image.
<code>in</code>	<i>ref</i>	The translation vector.
<code>in</code>	<i>extension</i>	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.
<code>in</code>	<i>output_domain</i>	The domain of the output image. An invalid domain, causes the routine to use the translated <code>input_domain</code> .

Returns

An image with the same domain as `input`.

Definition at line 99 of file `translate.hh`.

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

Referenced by `translate()`.

9.63.2.42 `template<typename I, typename V, typename Ext > mln::trait::concrete< I >::ret mln::geom::translate (const Image< I > & input, const algebra::vec< I::site::dim, V > & ref, const Ext & extension)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 132 of file `translate.hh`.

References `translate()`.

9.63.2.43 `template<typename I, typename V > mln::trait::concrete< I >::ret mln::geom::translate (const Image< I > & input, const algebra::vec< I::site::dim, V > & ref)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use [literal::zero](#) as default value for the extension.

Definition at line 146 of file `translate.hh`.

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

9.63.2.44 `template<typename I > mln::trait::concrete< I >::ret mln::geom::vertical_symmetry (const Image< I > & input)`

Performs a vertical symmetry.

Definition at line 174 of file `vertical_symmetry.hh`.

9.64 mln::geom::impl Namespace Reference

Implementation namespace of `geom` namespace.

Functions

- `template<typename I, typename N > mln::trait::concrete< I >::ret seeds2tiling (const Image< I > & ima_, const Neighborhood< N > & nbh_)`
Generic implementation of `geom::seed2tiling`.

9.64.1 Detailed Description

Implementation namespace of `geom` namespace.

9.64.2 Function Documentation

9.64.2.1 `template<typename I, typename N> mln::trait::concrete< I>::ret mln::geom::impl::seeds2tiling (const Image< I> & ima_, const Neighborhood< N> & nbh_) [inline]`

Generic implementation of geom::seed2tiling.

Parameters

in, out	<i>ima_</i>	The labeled image with seed.
in	<i>nbh_</i>	The neighborhood to use on this algorithm.

Definition at line 77 of file seeds2tiling.hh.

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

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

9.65 mln::graph Namespace Reference

Namespace of graph related routines.

Functions

- `template<typename G, typename F> F::result compute (const Graph< G> &g_, F &functor)`
Base routine to compute attributes on a graph.
- `template<typename I, typename N, typename L> mln::trait::ch_value< I, L>::ret labeling (const Image< I> &graph_image_, const Neighborhood< N> &nbh_, L &nlabels)`
Label graph components.
- `template<typename I, typename M> graph_elt_neighborhood_if< mln_graph(I), typename I::domain_t, M> to_neighb (const Image< I> &graph_image_, const Image< M> &graph_mask_image_)`
Make a custom graph neighborhood from a mask image.
- `template<typename I, typename M> graph_elt_window_if< mln_graph(I), typename I::domain_t, M> to_win (const Image< I> &graph_image_, const Image< M> &graph_mask_image_)`
Make a custom graph window from a mask image.

9.65.1 Detailed Description

Namespace of graph related routines.

9.65.2 Function Documentation

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

Base routine to compute attributes on a graph.

Parameters

<code>in</code>	<code>g_</code>	A graph.
<code>in</code>	<code>functor</code>	A functor implementing the right interface.

Returns

The computed data.

See Also

`canvas::browsing::depth_first_search`

Definition at line 63 of file `graph/compute.hh`.

9.65.2.2 `template<typename I, typename N, typename L > mln::trait::ch_value< I, L >::ret mln::graph::labeling (const Image< I > & graph_image_, const Neighborhood< N > & nbh_, L & nlabels)`

Label graph components.

[Vertex](#) with id 0, usually used to represent the background component, will be labeled with an id different from 0. Therefore, the labeling starts from 1.

Parameters

<code>in</code>	<code>graph_image_</code>	A graph image (
-----------------	---------------------------	-----------------

See Also

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

Parameters

<code>in</code>	<code>nbh_</code>	A graph neighborhood.
<code>in, out</code>	<code>nlabels</code>	The number of labels found.

Returns

a [Graph](#) image of labels.

Definition at line 72 of file `labeling.hh`.

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

9.65.2.3 `template<typename I, typename M > graph_elt_neighborhood_if< mln_graph(I), typename I::domain_t, M > mln::graph::to_neighb (const Image< I > & graph_image_, const Image< M > & graph_mask_image_)`

Make a custom graph neighborhood from a mask image.

Parameters

<code>in</code>	<code>graph_image_</code>	A graph image (
-----------------	---------------------------	-----------------

See Also

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

Parameters

in	<i>graph_mask_image_</i>	A graph image of bool used as a mask.
----	--------------------------	---------------------------------------

Returns

A masked neighborhood on graph.

Definition at line 57 of file to_neighb.hh.

9.65.2.4 `template<typename I, typename M> graph_elt_window_if< mln_graph(I), typename I::domain_t, M> mln::graph::to_win (const Image< I> & graph_image_, const Image< M> & graph_mask_image_)`

Make a custom graph window from a mask image.

Parameters

in	<i>graph_image_</i>	A graph image (
----	---------------------	-----------------

See Also

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

Parameters

in	<i>graph_mask_image_</i>	A graph image of bool used as a mask.
----	--------------------------	---------------------------------------

Returns

A masked window on graph.

Definition at line 57 of file to_win.hh.

9.66 mln::grid Namespace Reference

Namespace of grids definitions.

9.66.1 Detailed Description

Namespace of grids definitions. Compute the image::space trait from a point type.

9.67 mln::histo Namespace Reference

Namespace of histograms.

Namespaces

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

Classes

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

Functions

- `template<typename I >`
`histo::array< typename I::value > compute (const Image< I > &input)`
Compute the histogram of image `input`.
- `template<typename I >`
`mln::trait::concrete< I >::ret equalize (const Image< I > &input)`
Equalizes the histogram of image `input`.

9.67.1 Detailed Description

Namespace of histograms.

9.67.2 Function Documentation

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

Compute the histogram of image `input`.

Definition at line 79 of file `histo/compute.hh`.

Referenced by `equalize()`.

9.67.2.2 `template<typename I > mln::trait::concrete< I >::ret mln::histo::equalize (const Image< I > &input)`

Equalizes the histogram of image `input`.

Author

J. Fabrizio, R. Levillain

Definition at line 55 of file `histo/equalize.hh`.

References `compute()`, and `mln::initialize()`.

9.68 mln::histo::impl Namespace Reference

Implementation namespace of `histo` namespace.

Namespaces

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

9.68.1 Detailed Description

Implementation namespace of `histo` namespace.

9.69 mln::histo::impl::generic Namespace Reference

Generic implementation namespace of histo namespace.

9.69.1 Detailed Description

Generic implementation namespace of histo namespace.

9.70 mln::impl Namespace Reference

Implementation namespace of mln namespace.

9.70.1 Detailed Description

Implementation namespace of mln namespace.

9.71 mln::io Namespace Reference

Namespace of input/output handling.

Namespaces

- namespace [cloud](#)
Namespace of cloud input/output handling.
- namespace [dicom](#)
Namespace of DICOM input/output handling.
- namespace [dump](#)
Namespace of dump input/output handling.
- namespace [fits](#)
Namespace of fits input/output handling.
- namespace [fld](#)
Namespace of pgm input/output handling.
- namespace [magick](#)
Namespace of magick input/output handling.
- namespace [off](#)
Namespace of off input/output handling.
- namespace [pbm](#)
Namespace of pbm input/output handling.
- namespace [pbms](#)
Namespace of pbms input/output handling.
- namespace [pfm](#)
Namespace of pfm input/output handling.
- namespace [pgm](#)
Namespace of pgm input/output handling.
- namespace [pgms](#)
Namespace of pgms input/output handling.
- namespace [plot](#)
Namespace of plot input/output handling.

- namespace [pnm](#)
Namespace of pnm input/output handling.
- namespace [pnms](#)
Namespace of pnms input/output handling.
- namespace [ppm](#)
Namespace of ppm input/output handling.
- namespace [ppms](#)
Namespace of ppms input/output handling.
- namespace [raw](#)
Namespace of raw input/output handling.
- namespace [tiff](#)
Namespace of tiff input/output handling.
- namespace [txt](#)
Namespace of txt input/output handling.

9.71.1 Detailed Description

Namespace of input/output handling.

9.72 mln::io::cloud Namespace Reference

Namespace of cloud input/output handling.

Functions

- `template<typename P >
void load (p_array< P > &arr, const std::string &filename)`
Load a cloud of points.
- `template<typename P >
void save (const p_array< P > &arr, const std::string &filename)`
Load a cloud of points.

9.72.1 Detailed Description

Namespace of cloud input/output handling.

9.72.2 Function Documentation

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

Load a cloud of points.

Parameters

<code>in, out</code>	<code>arr</code>	the site set where to load the data.
<code>in</code>	<code>filename</code>	file to load.

Definition at line 88 of file cloud/load.hh.

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

Load a cloud of points.

Parameters

<code>in</code>	<code>arr</code>	the cloud of points to save.
<code>in</code>	<code>filename</code>	the destination.

Definition at line 83 of file cloud/save.hh.

9.73 mln::io::dicom Namespace Reference

Namespace of DICOM input/output handling.

Classes

- struct [dicom_header](#)
Store dicom file header.

Functions

- [dicom_header get_header](#) (const std::string &filename)
Retrieve header in a dicom file.
- `template<typename I >`
`void load (Image< I > &ima, const std::string &filename)`

9.73.1 Detailed Description

Namespace of DICOM input/output handling.

9.73.2 Function Documentation

9.73.2.1 `dicom_header mln::io::dicom::get_header (const std::string & filename)`

Retrieve header in a dicom file.

Definition at line 76 of file dicom/get_header.hh.

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

9.73.2.2 `template<typename I > void mln::io::dicom::load (Image< I > & ima, const std::string & filename)` `[inline]`

Load a DICOM file in a Milena image.

```
\param[out] ima A reference to the image which will receive
data.
\param[in] filename The source.
```

Common compilation flags to link to gdcm if this file is used:

```
-lgdcmCommon -lgdcmDICT -lgdcmDSED -lgdcmIOD -lgdcmMSFF -lgdcmexpat -lgdcmjpeg12 -lgdcmjpeg16 -lgdcmjpeg8
```

Definition at line 96 of file dicom/load.hh.

References `mln::initialize()`.

9.74 mln::io::dump Namespace Reference

Namespace of dump input/output handling.

Classes

- struct [dump_header](#)
Store dump file header.

Functions

- [dump_header get_header](#) (const std::string &filename)
Retrieve header in a dump file.
- template<typename I >
void [load](#) ([Image](#)< I > &ima_, const std::string &filename)
Load a Milena image by dumped into a file.
- template<typename I >
void [save](#) (const [Image](#)< I > &ima_, const std::string &filename)
Save a Milena image by dumping its data to a file.

9.74.1 Detailed Description

Namespace of dump input/output handling.

9.74.2 Function Documentation

9.74.2.1 [dump_header mln::io::dump::get_header](#) (const std::string & *filename*) [inline]

Retrieve header in a dump file.

Definition at line 70 of file dump/get_header.hh.

References [mln::util::array< T >::resize\(\)](#).

9.74.2.2 [template<typename I > void mln::io::dump::load](#) ([Image](#)< I > & *ima_*, const std::string & *filename*)

Load a Milena image by dumped into a file.

Parameters

<i>in, out</i>	<i>ima_</i>	The image to load.
<i>in</i>	<i>filename</i>	the destination.

Definition at line 171 of file dump/load.hh.

9.74.2.3 [template<typename I > void mln::io::dump::save](#) (const [Image](#)< I > & *ima_*, const std::string & *filename*)

Save a Milena image by dumping its data to a file.

Parameters

<i>in</i>	<i>ima_</i>	The image to save.
<i>in</i>	<i>filename</i>	the destination.

Definition at line 131 of file dump/save.hh.

9.75 mln::io::fits Namespace Reference

Namespace of fits input/output handling.

Functions

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

9.75.1 Detailed Description

Namespace of fits input/output handling.

9.75.2 Function Documentation

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

Load a fits image in a Milena image.

Parameters

out	<i>ima</i>	A reference to the image2d<float> which will receive data.
in	<i>filename</i>	The source.

Definition at line 132 of file fits/load.hh.

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

Load a fits image in a image2d<float>.

Parameters

in	<i>filename</i>	The image source.
----	-----------------	-------------------

Returns

An image2d<float> which contains loaded data.

Definition at line 85 of file fits/load.hh.

9.76 mln::io::fld Namespace Reference

Namespace of pgm input/output handling.

Classes

- struct [fld_header](#)

Define the header structure of an AVS field data file.

Functions

- `template<typename I >`
`void load (Image< I > &ima_, const char *filename)`
Load an image from an AVS field file.
- `fld_header read_header (std::istream &ins)`
Read the header form an AVS field file.
- `void write_header (std::ostream &file, const fld_header &h)`
Write the AVS header in a file.

9.76.1 Detailed Description

Namespace of pgm input/output handling.

9.76.2 Function Documentation

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

Load an image from an AVS field file.

Parameters

<code>in, out</code>	<code>ima_</code>	The image to load.
<code>in</code>	<code>filename</code>	The path to the AVS file.

Definition at line 199 of file fld/load.hh.

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

9.76.2.2 `fld_header mln::io::fld::read_header (std::istream & ins) [inline]`

Read the header form an AVS field file.

Parameters

<code>ins</code>	The file to read.
------------------	-------------------

Returns

The header.

Definition at line 76 of file fld/load_header.hh.

Referenced by `load()`.

9.76.2.3 `void mln::io::fld::write_header (std::ostream & file, const fld_header & h) [inline]`

Write the AVS header in a file.

Parameters

<code>file</code>	The file to write.
<code>h</code>	The AVS header.

Definition at line 58 of file write_header.hh.

9.77 mln::io::magick Namespace Reference

Namespace of magick input/output handling.

Functions

- template<typename I >
void [load](#) (Image< I > &ima, const std::string &filename)
Load data from a file into a Milena image using Magick++.
- template<typename I >
void [save](#) (const Image< I > &ima, const std::string &filename)
Save a Milena image into a file using Magick++.
- template<typename I, typename J >
void [save](#) (const Image< I > &ima, const Image< J > &opacity_mask, const std::string &filename)
Save a Milena image into a file using Magick++.

9.77.1 Detailed Description

Namespace of magick input/output handling.

9.77.2 Function Documentation

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

Load data from a file into a Milena image using Magick++.

Parameters

out	<i>ima</i>	The image data are loaded into.
in	<i>filename</i>	The name of the input file.

Definition at line 139 of file magick/load.hh.

References mln::initialize(), mln::geom::max_col(), mln::geom::max_row(), mln::geom::min_col(), and mln::geom::min_row().

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

Save a Milena image into a file using Magick++.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

in	<i>ima</i>	The image to save.
in	<i>filename</i>	The name of the output file.

Definition at line 288 of file magick/save.hh.

9.77.2.3 `template<typename I , typename J > void mln::io::magick::save (const Image< I > & ima, const Image< J > & opacity_mask, const std::string & filename)`

Save a Milena image into a file using Magick++.

Parameters

in	<i>ima</i>	The image to save.
in	<i>opacity_mask</i>	Mask used to set pixel opacity_mask in output image. Output format must support this feature to be taken into account.
in	<i>filename</i>	The name of the output file.

Definition at line 228 of file magick/save.hh.

References `mln::geom::ncols()`, and `mln::geom::nrows()`.

9.78 mln::io::off Namespace Reference

Namespace of off input/output handling.

Functions

- void `load (bin_2complex_image3df &ima, const std::string &filename)`
Load a (binary) OFF image into a complex image.
- void `save (const bin_2complex_image3df &ima, const std::string &filename)`
Save a (binary) OFF image into a complex image.
- `template<typename I >`
void `save_bin_alt (const I &ima, const std::string &filename)`
FIXME: Similar to `mln::io::off::save(const bin_2complex_image3df&, const std::string&)`, but does not save faces whose value is 'false'.

9.78.1 Detailed Description

Namespace of off input/output handling.

9.78.2 Function Documentation

9.78.2.1 `void mln::io::off::load (bin_2complex_image3df & ima, const std::string & filename)` `[inline]`

Load a (binary) OFF image into a complex image.

Load a 3x8-bit RGB (color) OFF image into a complex image.

Load a floating-point OFF image into a complex image.

Parameters

out	<i>ima</i>	A reference to the image to construct.
in	<i>filename</i>	The name of the file to load.

The image is said binary since data only represent the existence of faces.

Parameters

out	<i>ima</i>	A reference to the image to construct.
in	<i>filename</i>	The name of the file to load.

Read floating-point data is attached to 2-faces only; 1-faces and 0-faces are set to 0.0f.

Definition at line 181 of file off/load.hh.

9.78.2.2 `void mln::io::off::save (const bin_2complex_image3df & ima, const std::string & filename)` `[inline]`

Save a (binary) OFF image into a complex image.

Save a 3x8-bit RGB (color) OFF image into a complex image.

Save a floating-point value grey-level OFF image into a complex image.

Save an 8-bit grey-level OFF image into a complex image.

Parameters

in	<i>ima</i>	The image to save.
in	<i>filename</i>	The name of the file where to save the image.

The image is said binary since data represent only the existence of faces.

Parameters

in	<i>ima</i>	The image to save.
in	<i>filename</i>	The name of the file where to save the image.

Only data is attached to 2-faces is saved; the OFF file cannot store data attached to faces of other dimensions.

Definition at line 167 of file off/save.hh.

9.78.2.3 `template<typename I> void mln::io::off::save_bin_alt (const I & ima, const std::string & filename)`

FIXME: Similar to `mln::io::off::save(const bin_2complex_image3df&, const std::string&)`, but does not save faces whose value is 'false'.

Definition at line 59 of file save_bin_alt.hh.

9.79 mln::io::pbm Namespace Reference

Namespace of pbm input/output handling.

Namespaces

- namespace [impl](#)
Namespace of pbm implementation details.

Functions

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

- `template<typename I >`
`void save (const Image< I > &ima, const std::string &filename)`

9.79.1 Detailed Description

Namespace of pbm input/output handling.

9.79.2 Function Documentation

9.79.2.1 `void mln::io::pbm::load (image2d< bool > &ima, const std::string &filename)` `[inline]`

Load a pbm image in a Milena image.

Parameters

out	<i>ima</i>	A reference to the image2d<bool> which will receive data.
in	<i>filename</i>	The source.

Definition at line 156 of file pbm/load.hh.

Referenced by `mln::io::pnms::load()`.

9.79.2.2 `image2d< bool > mln::io::pbm::load (const std::string &filename)` `[inline]`

Load a pbm image in a image2d<float>.

Parameters

in	<i>filename</i>	The image source.
----	-----------------	-------------------

Returns

An image2d<float> which contains loaded data.

Definition at line 128 of file pbm/load.hh.

9.79.2.3 `template<typename I > void mln::io::pbm::save (const Image< I > &ima, const std::string &filename)`
`[inline]`

Save a Milena image as a pbm image.

```
\param[in] ima The image to save.
\param[in,out] filename the destination.
```

Definition at line 119 of file pbm/save.hh.

9.80 mln::io::pbm::impl Namespace Reference

Namespace of pbm implementation details.

9.80.1 Detailed Description

Namespace of pbm implementation details.

9.81 mln::io::pbms Namespace Reference

Namespace of pbms input/output handling.

Namespaces

- namespace [impl](#)
Namespace of pbms implementation details.

Functions

- void [load](#) (image3d< bool > &ima, const util::array< std::string > &filenames)
Load pbms images as slices of a 3D Milena image.

9.81.1 Detailed Description

Namespace of pbms input/output handling.

9.81.2 Function Documentation

9.81.2.1 void mln::io::pbms::load (image3d< bool > & *ima*, const util::array< std::string > & *filenames*) `[inline]`

Load pbms images as slices of a 3D Milena image.

Parameters

out	<i>ima</i>	A reference to the 3D image which will receive data.
in	<i>filenames</i>	The list of 2D images to load..

Definition at line 65 of file pbms/load.hh.

References `mln::io::pnms::load()`.

9.82 mln::io::pbms::impl Namespace Reference

Namespace of pbms implementation details.

9.82.1 Detailed Description

Namespace of pbms implementation details.

9.83 mln::io::pfm Namespace Reference

Namespace of pfm input/output handling.

Namespaces

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

Functions

- void [load](#) ([image2d](#)< float > &ima, const std::string &filename)
Load a pfm image in a Milena image.
- [image2d](#)< float > [load](#) (const std::string &filename)
Load a pfm image in a [image2d](#)<float>.
- template<typename I >
void [save](#) (const [Image](#)< I > &ima, const std::string &filename)
Save a Milena image as a pfm image.

9.83.1 Detailed Description

Namespace of pfm input/output handling.

9.83.2 Function Documentation

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

Load a pfm image in a Milena image.

Parameters

out	<i>ima</i>	A reference to the image2d <float> which will receive data.
in	<i>filename</i>	The source.

Definition at line 162 of file pfm/load.hh.

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

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

Parameters

in	<i>filename</i>	The image source.
----	-----------------	-------------------

Returns

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

Definition at line 138 of file pfm/load.hh.

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

Save a Milena image as a pfm image.

Parameters

in	<i>ima</i>	The image to save.
in, out	<i>filename</i>	the destination.

Definition at line 101 of file pfm/save.hh.

9.84 mln::io::pfm::impl Namespace Reference

Implementation namespace of pfm namespace.

9.84.1 Detailed Description

Implementation namespace of pfm namespace.

9.85 mln::io::pgm Namespace Reference

Namespace of pgm input/output handling.

Functions

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

9.85.1 Detailed Description

Namespace of pgm input/output handling.

9.85.2 Function Documentation

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

Load a pgm image in a Milena image.

Parameters

out	<i>ima</i>	A reference to the image which will receive data.
in	<i>filename</i>	The source.

Definition at line 87 of file pgm/load.hh.

9.85.2.2 template<typename V > [image2d](#)< V > mln::io::pgm::load (const std::string & *filename*) [inline]

Load a pgm image in a Milena image.

To use this routine, you should specialize the template with the value type of the image loaded. (ex : [load](#)<value::int_u8>("..."))

Parameters

in	<i>filename</i>	The image source.
----	-----------------	-------------------

Returns

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

Definition at line 77 of file `pgm/load.hh`.

9.85.2.3 `template<typename I > void mln::io::pgm::save (const Image< I > & ima, const std::string & filename)`
`[inline]`

Save a Milena image as a pgm image.

```
\param[in] ima The image to save.
\param[in,out] filename the destination.
```

Definition at line 78 of file `pgm/save.hh`.

References `mln::io::pnm::save()`.

9.86 mln::io::pgms Namespace Reference

Namespace of pgms input/output handling.

Functions

- `template<typename V >`
`void load (image3d< V > &ima, const util::array< std::string > &filenames)`
Load pgm images as slices of a 3D Milena image.

9.86.1 Detailed Description

Namespace of pgms input/output handling.

9.86.2 Function Documentation

9.86.2.1 `template<typename V > void mln::io::pgms::load (image3d< V > & ima, const util::array< std::string > & filenames)` `[inline]`

Load pgm images as slices of a 3D Milena image.

Parameters

out	<i>ima</i>	A reference to the 3D image which will receive data.
in	<i>filenames</i>	The list of 2D images to load..

Definition at line 69 of file `pgms/load.hh`.

9.87 mln::io::plot Namespace Reference

Namespace of plot input/output handling.

Functions

- template<typename I >
void [load](#) (util::array< I > &arr, const std::string &filename)
- template<typename I >
void [save](#) (const [image1d](#)< I > &ima, const std::string &filename)
Save a Milena 1D image in a plot file.
- template<typename T >
void [save](#) (const util::array< T > &arr, const std::string &filename, int start_value=0)
Save a Milena array in a plot file.
- template<typename T >
void [save](#) (const [histo::array](#)< T > &arr, const std::string &filename)

9.87.1 Detailed Description

Namespace of plot input/output handling.

9.87.2 Function Documentation

9.87.2.1 template<typename I > void mln::io::plot::load (util::array<I > & arr, const std::string & filename) [inline]

Load a Milena 1D image from a plot file.

```
\param[in] ima A reference to the image to load.
\param[out] filename The output file.
\param[in] start_value The start index value of the plot
(optional).
```

Load a Milena array from a plot file.

```
\param[in] arr A reference to the array to load.
\param[out] filename The output file.
```

Definition at line 93 of file plot/load.hh.

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

9.87.2.2 template<typename I > void mln::io::plot::save (const [image1d](#)< I > & ima, const std::string & filename)

Save a Milena 1D image in a plot file.

Parameters

in	<i>ima</i>	A reference to the image to save.
out	<i>filename</i>	The output file.

9.87.2.3 template<typename T > void mln::io::plot::save (const util::array< T > & arr, const std::string & filename, int start_value = 0) [inline]

Save a Milena array in a plot file.

Parameters

in	<i>arr</i>	A reference to the array to save.
out	<i>filename</i>	The output file.
in	<i>start_value</i>	The start index value of the plot (optional).

Definition at line 89 of file plot/save.hh.

References `mln::util::array< T >::nelements()`.

9.87.2.4 `template<typename T > void mln::io::plot::save (const histo::array< T > & arr, const std::string & filename)`
`[inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 103 of file plot/save.hh.

9.88 mln::io::pnm Namespace Reference

Namespace of pnm input/output handling.

Namespaces

- namespace [impl](#)
Namespace of pnm's implementation details.

Functions

- `template<typename V > image2d< V > load (char type_, const std::string &filename)`
main function : load pnm format
- `template<typename I > void load (char type_, Image< I > &ima_, const std::string &filename)`
An other way to load pnm files : the destination is an argument to check if the type match the file to load.
- `template<typename I > void load_raw_2d (std::ifstream &file, I &ima)`
load_raw_2d.
- `template<typename V > unsigned int max_component (const V &)`
Give the maximum value which can be stored as a component value type V.
- `template<typename I > void save (char type, const Image< I > &ima_, const std::string &filename)`

9.88.1 Detailed Description

Namespace of pnm input/output handling.

9.88.2 Function Documentation

9.88.2.1 `template<typename V > image2d<V> mln::io::pnm::load (char type_, const std::string & filename)`
`[inline]`

main function : load pnm format

Definition at line 210 of file pnm/load.hh.

References `load_raw_2d()`, and `max_component()`.

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

An other way to load pnm files : the destination is an argument to check if the type match the file to load.

Definition at line 257 of file pnm/load.hh.

References `mln::make::box2d()`, `load_raw_2d()`, and `max_component()`.

9.88.2.3 `template<typename I> void mln::io::pnm::load_raw_2d (std::ifstream & file, I & ima) [inline]`

`load_raw_2d`.

for all pnm 8/16 bits formats

Definition at line 198 of file pnm/load.hh.

Referenced by `load()`.

9.88.2.4 `template<typename V> unsigned int mln::io::pnm::max_component (const V &) [inline]`

Give the maximum value which can be stored as a component value type V.

Definition at line 56 of file max_component.hh.

Referenced by `load()`.

9.88.2.5 `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.

```
\param[in] type The type of the image to save (can be PPM,  
PGM, PBM) .  
\param[in] ima_ The image to save.  
\param[in,out] filename the destination.
```

Definition at line 185 of file pnm/save.hh.

Referenced by `mln::io::ppm::save()`, and `mln::io::pgm::save()`.

9.89 mln::io::pnm::impl Namespace Reference

Namespace of pnm's implementation details.

9.89.1 Detailed Description

Namespace of pnm's implementation details.

9.90 mln::io::pnms Namespace Reference

Namespace of pnms input/output handling.

Functions

- `template<typename V>
void load (char type, image3d< V> &ima, const util::array< std::string> &filenames)`

Load pnm images as slices of a 3D Milena image.

- void `load` (char type, `image3d`< bool > &ima, const `util::array`< std::string > &filenames)

9.90.1 Detailed Description

Namespace of pnms input/output handling.

9.90.2 Function Documentation

- 9.90.2.1 `template<typename V> void mln::io::pnms::load (char type, image3d< V> & ima, const util::array< std::string> & filenames) [inline]`

Load pnm images as slices of a 3D Milena image.

Parameters

in	<i>type</i>	The type of the pnm files.
out	<i>ima</i>	A reference to the 3D image which will receive data.
in	<i>filenames</i>	The list of 2D images to load..

Definition at line 79 of file pnms/load.hh.

References `mln::make::image3d()`, `mln::util::array< T>::is_empty()`, and `mln::util::array< T>::nelements()`.

Referenced by `mln::io::pbms::load()`.

- 9.90.2.2 `void mln::io::pnms::load (char type, image3d< bool> & ima, const util::array< std::string> & filenames) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 102 of file pnms/load.hh.

References `mln::make::image3d()`, `mln::util::array< T>::is_empty()`, `mln::io::pbm::load()`, and `mln::util::array< T>::nelements()`.

9.91 mln::io::ppm Namespace Reference

Namespace of ppm input/output handling.

Functions

- `template<typename I> void load (Image< I> &ima, const std::string &filename)`
Load a ppm image in a Milena image.
- `template<typename V> image2d< V> load (const std::string &filename)`
Load a ppm image in a Milena image.
- `template<typename I> void save (const Image< I> &ima, const std::string &filename)`

9.91.1 Detailed Description

Namespace of ppm input/output handling.

9.91.2 Function Documentation

9.91.2.1 `template<typename I > void mln::io::ppm::load (Image< I > & ima, const std::string & filename)` `[inline]`

Load a ppm image in a Milena image.

Parameters

<code>out</code>	<code><i>ima</i></code>	A reference to the image which will receive data.
<code>in</code>	<code><i>filename</i></code>	The source.

Definition at line 89 of file ppm/load.hh.

9.91.2.2 `template<typename V > image2d< V > mln::io::ppm::load (const std::string & filename)` `[inline]`

Load a ppm image in a Milena image.

To use this routine, you should specialize the template with the value type of the image loaded. (ex : `load<value::int_u8>("...")`)

Parameters

<code>in</code>	<code><i>filename</i></code>	The image source.
-----------------	------------------------------	-------------------

Returns

An `image2d` which contains loaded data.

Definition at line 79 of file ppm/load.hh.

9.91.2.3 `template<typename I > void mln::io::ppm::save (const Image< I > & ima, const std::string & filename)`
`[inline]`

Save a Milena image as a ppm image.

```
\param[in] ima The image to save.
\param[in,out] filename the destination.
```

Definition at line 65 of file ppm/save.hh.

References `mln::io::pnm::save()`.

Referenced by `mln::registration::icp()`.

9.92 mln::io::ppms Namespace Reference

Namespace of ppms input/output handling.

Functions

- `template<typename V >`
void `load (image3d< V > &ima, const util::array< std::string > &filenames)`
Load ppm images as slices of a 3D Milena image.

9.92.1 Detailed Description

Namespace of ppms input/output handling.

9.92.2 Function Documentation

9.92.2.1 `template<typename V > void mln::io::ppms::load (image3d< V > & ima, const util::array< std::string > & filenames) [inline]`

Load ppm images as slices of a 3D Milena image.

Parameters

out	ima	A reference to the 3D image which will receive data.
in	filenames	The list of 2D images to load..

Definition at line 67 of file ppms/load.hh.

9.93 mln::io::raw Namespace Reference

Namespace of raw input/output handling.

Classes

- struct [raw_header](#)
Store raw file header.

Functions

- [raw_header get_header](#) (const std::string &filename)
Retrieve header in a raw file.
- `template<typename I >`
void [load](#) ([Image](#)< I > &ima_, const std::string &filename)
Load an image saved as a raw data file.
- `template<typename I >`
void [save](#) (const [Image](#)< I > &ima_, const std::string &filename)
Save a Milena image as a raw data file.

9.93.1 Detailed Description

Namespace of raw input/output handling.

9.93.2 Function Documentation

9.93.2.1 `raw_header mln::io::raw::get_header (const std::string & filename)`

Retrieve header in a raw file.

Definition at line 68 of file raw/get_header.hh.

References `mln::util::array< T >::resize()`.

9.93.2.2 `template<typename I> void mln::io::raw::load (Image< I> & ima_, const std::string & filename)`

Load an image saved as a raw data file.

Parameters

<code>in, out</code>	<code>ima_</code>	The image to load.
<code>in</code>	<code>filename</code>	the destination.

This routine try to read two input files: 'filename' and 'filename.info'. 'filename' is the raw data. 'filename.info' store various information about the image.

Definition at line 184 of file raw/load.hh.

9.93.2.3 `template<typename I> void mln::io::raw::save (const Image< I> & ima_, const std::string & filename)`

Save a Milena image as a raw data file.

Parameters

<code>in</code>	<code>ima_</code>	The image to save.
<code>in</code>	<code>filename</code>	the destination.

This routine produce two output files: 'filename' and 'filename.info'. 'filename' is the raw data. 'filename.info' store various information about the image.

Definition at line 135 of file raw/save.hh.

9.94 mln::io::tiff Namespace Reference

Namespace of tiff input/output handling.

Functions

- `template<typename I>`
`void load (Image< I> &ima_, const std::string &filename)`
Load a TIFF image to a Milena image.

9.94.1 Detailed Description

Namespace of tiff input/output handling.

9.94.2 Function Documentation

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

Load a TIFF image to a Milena image.

Definition at line 323 of file tiff/load.hh.

9.95 mln::io::txt Namespace Reference

Namespace of txt input/output handling.

Functions

- void [save](#) (const [image2d](#)< char > &ima, const std::string &filename)
Save an image as txt file.

9.95.1 Detailed Description

Namespace of txt input/output handling.

9.95.2 Function Documentation

9.95.2.1 void `mln::io::txt::save (const image2d< char > & ima, const std::string & filename)` `[inline]`

Save an image as txt file.

Parameters

<code>in</code>	<code>ima</code>	The image to save. Must be an image of char.
<code>in</code>	<code>filename</code>	the destination.

Definition at line 63 of file txt/save.hh.

References `mln::image2d< T >::domain()`.

9.96 mln::labeling Namespace Reference

Namespace of labeling routines.

Namespaces

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

Functions

- template<typename I , typename N , typename L >
`mln::trait::ch_value< I, L >::ret background (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`
- template<typename I , typename N , typename L >
`mln::trait::ch_value< I, L >::ret blobs (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)`
Connected component labeling of the binary objects of a binary image.
- template<typename I , typename N , typename L , typename A >
`util::couple`
`< mln::trait::ch_value< I, L >`
`::ret, util::couple`
`< util::array< typename`
`A::result >, util::array< A > > > blobs_and_compute (const Image< I > &input, const Neighborhood< N`
`> &nbh, L &nlabels, const Accumulator< A > &accu)`
- template<typename V , typename L >
`mln::trait::ch_value< L, V >::ret colorize (const V &value, const Image< L > &labeled_image, const type-`
`name L::value &nlabels)`
Create a new color image from a labeled image and fill each component with a random color.

- `template<typename V , typename L >`
`mln::trait::ch_value< L, V >::ret colorize (const V &value, const Image< L > &labeled_image)`
- `template<typename L >`
`mln::trait::ch_value< L,`
`mln::value::rgb8 >::ret colorize (const Image< L > &input, const typename L::value &nlabels)`
- `template<typename A , typename I , typename L >`
`util::array< 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`
`< mln_meta_accu_result(A,`
`typename I::value)> compute (const Meta_Accumulator< A > &a, const Image< I > &input, const Image<`
`L > &label, const typename L::value &nlabels)`
Compute an accumulator onto the pixel values of the image `input`.
- `template<typename A , typename L >`
`util::array< typename A::result > compute (const Accumulator< A > &a, const Image< L > &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`
`< mln_meta_accu_result(A,`
`typename L::psite)> compute (const Meta_Accumulator< A > &a, const Image< L > &label, const type-`
`name L::value &nlabels)`
Compute an accumulator onto the pixel sites of each component domain of `label`.
- `template<typename A , typename I , typename L >`
`util::array< typename A::result > compute (util::array< A > &a, const Image< I > &input, const Image< L`
`> &label, const typename L::value &nlabels)`
Compute an accumulator onto the pixel values of the image `input`.
- `template<typename A , typename I , typename L >`
`mln::trait::ch_value< L,`
`typename A::result >::ret compute_image (const util::array< typename A::result > &a, const Image< I >`
`&input, const Image< L > &labels, const typename L::value &nlabels)`
Compute an accumulator onto the pixel values of the image `input`.
- `template<typename A , typename I , typename L >`
`mln::trait::ch_value< L,`
`typename A::result >::ret compute_image (const Accumulator< A > &accu, const Image< I > &input, const`
`Image< L > &labels, const typename L::value &nlabels)`
Compute an accumulator onto the pixel values of the image `input`.
- `template<typename A , typename I , typename L >`
`mln::trait::ch_value< L,`
`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 , typename W >`
`mln::util::array< typename`
`A::result > compute_in_window (const Accumulator< A > &a, const Image< I > &input, const Image< L >`
`&label, const typename L::value &nlabels, const Window< W > &win)`
Compute an accumulator for each image pixel values using neighbor pixel values.
- `template<typename I , typename N , typename L >`
`mln::trait::concrete< I >::ret fill_holes (const Image< I > &input, const Neighborhood< N > &nbh, L &nla-`
`bels)`
Filling holes of a single object in a binary image.

- template<typename I, typename N, typename L >
 mln::trait::ch_value< I, L >::ret flat_zones (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)
Connected component labeling of the flat zones of an image.
- template<typename I, typename N, typename L >
 mln::trait::ch_value< I, L >::ret foreground (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)
- template<typename I >
 mln::trait::concrete< I >::ret pack (const Image< I > &label, typename I::value &new_nlabels, fun::i2v::array< typename I::value > &repack_fun)
Relabel a labeled image in order to have a contiguous labeling.
- template<typename I >
 mln::trait::concrete< I >::ret pack (const Image< I > &label, typename I::value &new_nlabels)
- template<typename I >
 void pack_inplace (Image< I > &label, typename I::value &new_nlabels, fun::i2v::array< typename I::value > &repack_fun)
Relabel inplace a labeled image in order to have a contiguous labeling.
- template<typename I >
 void pack_inplace (Image< I > &label, typename I::value &new_nlabels)
- template<typename I, typename N, typename L >
 mln::trait::ch_value< I, L >::ret regional_maxima (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)
- template<typename I, typename N, typename L >
 mln::trait::ch_value< I, L >::ret regional_minima (const Image< I > &input, const Neighborhood< N > &nbh, L &nlabels)
- template<typename I, typename F >
 mln::trait::concrete< I >::ret relabel (const Image< I > &label, const typename I::value &nlabels, typename I::value &new_nlabels, const Function_v2b< F > &fv2b)
Remove components and relabel a labeled image.
- template<typename I, typename F >
 mln::trait::concrete< I >::ret relabel (const Image< I > &label, const typename I::value &nlabels, const Function_v2v< F > &fv2v)
Remove components and relabel a labeled image.
- template<typename I, typename F >
 void relabel_inplace (Image< I > &label, const typename I::value &nlabels, const Function_v2b< F > &fv2b)
Remove components and relabel a labeled image inplace.
- template<typename I, typename F >
 void relabel_inplace (Image< I > &label, const typename I::value &nlabels, const Function_v2v< F > &fv2v)
Remove components and relabel a labeled image inplace.
- template<typename I, typename J >
 mln::trait::concrete< I >::ret superpose (const Image< I > &lhs, const typename I::value &lhs_nlabels, const Image< J > &rhs, const typename J::value &rhs_nlabels, typename I::value &new_nlabels)
Superpose two labeled image.
- template<typename I, typename N, typename L >
 mln::trait::ch_value< I, L >::ret value (const Image< I > &input, const typename I::value &val, const Neighborhood< N > &nbh, L &nlabels)
Connected component labeling of the image sites at a given value.
- template<typename I, typename N, typename L, typename A >
 util::couple
 < mln::trait::ch_value< I, L >
 ::ret, util::couple
 < util::array< typename
 A::result >, util::array< A > > > value_and_compute (const Image< I > &input, const typename I::value &val, const Neighborhood< N > &nbh, L &nlabels, const Accumulator< A > &accu)

Connected component labeling of the image sites at a given value.

- `template<typename V , typename I >`
`mln::trait::ch_value< I, V >::ret wrap (const V &value_type, const Image< I > &input)`

Wrap labels such as 0 -> 0 and [1, Lmax] maps to [1, Lmax] (using modulus).

- `template<typename I >`
`mln::trait::ch_value< I,`
`mln::value::label_8 >::ret wrap (const Image< I > &input)`

Wrap labels such as 0 -> 0 and [1, Lmax] maps to [1, Lmax] (using modulus).

9.96.1 Detailed Description

Namespace of labeling routines.

9.96.2 Function Documentation

- 9.96.2.1 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret mln::labeling::background (const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels) [inline]`

Connected component labeling of the background part in a binary image.

Parameters

<code>in</code>	<code>input</code>	The input image.
<code>in</code>	<code>nbh</code>	The connexity of the background.
<code>out</code>	<code>nlabels</code>	The number of labels.

Returns

The label image.

Precondition

The input image has to be binary (checked at compile-time).

This routine actually calls `mln::labeling::value` with the value set to `false`.

See Also

[mln::labeling::value](#)

Definition at line 69 of file background.hh.

References `value()`.

Referenced by `fill_holes()`.

- 9.96.2.2 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret mln::labeling::blobs (const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels) [inline]`

Connected component labeling of the binary objects of a binary image.

Parameters

<code>in</code>	<code>input</code>	The input image.
<code>in</code>	<code>nbh</code>	The connexity of the objects.
<code>out</code>	<code>nlabels</code>	The Number of labels. Its value is set in the algorithms.

Returns

The label image.

Precondition

The input image has to be binary (checked at compile-time).

A fast queue is used so that the algorithm is not recursive and can handle large binary objects (blobs).

Definition at line 102 of file labeling/blobs.hh.

References `mln::canvas::labeling::blobs()`.

Referenced by `mln::graph::labeling()`.

9.96.2.3 `template<typename I, typename N, typename L, typename A> util::couple< mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result >, util::array< A > > > mln::labeling::blobs_and_compute (const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels, const Accumulator< A > & accu)`

Label an image and compute given accumulators.

```
\param[in]      input    A binary image.
\param[in]      nbh      A neighborhood used for labeling.
\param[in,out]  nlabels   The number of labels found.
\param[in]      accu      An accumulator to be computed while labeling.

\return The labeled image, computed attributes for each regions
        and an array of the accumulators used to compute the
        attributes.
```

Definition at line 160 of file blobs_and_compute.hh.

References `mln::canvas::labeling::blobs()`, and `mln::make::couple()`.

9.96.2.4 `template<typename V, typename L> mln::trait::ch_value< L, V >::ret mln::labeling::colorize (const V & value, const Image< L > & labeled_image, const typename L::value & nlabels) [inline]`

Create a new color image from a labeled image and fill each component with a random color.

`litera::black` is used for component 0, e.g. the background. Min and max values for RGB values can be set through the global variables `mln::labeling::colorize_::min_value` and `mln::labeling::colorize_::max_value`.

Parameters

<code>in</code>	<i>value</i>	value type used in the returned image.
<code>in</code>	<i>labeled_image</i>	A labeled image (

See Also

[labeling::blobs](#)).

Parameters

<code>in</code>	<i>nlabels</i>	Number of labels.
-----------------	----------------	-------------------

Definition at line 190 of file colorize.hh.

References `mln::literal::black`, and `mln::data::transform()`.

Referenced by `colorize()`.

9.96.2.5 `template<typename V , typename L > mln::trait::ch_value< L, V >::ret mln::labeling::colorize (const V & value, const Image< L > & labeled_image) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 228 of file `colorize.hh`.

References `colorize()`, and `mln::data::compute()`.

9.96.2.6 `template<typename L > mln::trait::ch_value< L, mln::value::rgb8 >::ret mln::labeling::colorize (const Image< L > & input, const typename L::value & nlabels) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 247 of file `colorize.hh`.

References `colorize()`.

9.96.2.7 `template<typename A , typename I , typename L > util::array< typename A::result > mln::labeling::compute (const Accumulator< A > & a, const Image< I > & input, const Image< L > & label, const typename L::value & nlabels) [inline]`

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

for each component of the image `label`.

Parameters

in	<i>a</i>	An accumulator.
in	<i>input</i>	The input image.
in	<i>label</i>	The labeled image.
in	<i>nlabels</i>	The number of labels in <code>label</code> .

Returns

A `util::array` of accumulator result (one result per label).

Definition at line 715 of file `labeling/compute.hh`.

Referenced by `compute()`, `compute_image()`, `fill_holes()`, `mln::make::p_edges_with_mass_centers()`, and `mln::make::p_vertices_with_mass_centers()`.

9.96.2.8 `template<typename A , typename I , typename L > util::array< mln::meta_accu_result(A, typename I::value)> mln::labeling::compute (const Meta_Accumulator< A > & a, const Image< I > & input, const Image< L > & label, const typename L::value & nlabels) [inline]`

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

for each component of the image `label`.

Parameters

in	<i>a</i>	A meta-accumulator.
in	<i>input</i>	The input image.
in	<i>label</i>	The labeled image.
in	<i>nlabels</i>	The number of labels in <code>label</code> .

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 734 of file labeling/compute.hh.

References [compute\(\)](#).

9.96.2.9 `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

<code>in</code>	<code>a</code>	An accumulator.
<code>in</code>	<code>label</code>	The labeled image.
<code>in</code>	<code>nlabels</code>	The number of labels in <code>label</code> .

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 771 of file labeling/compute.hh.

9.96.2.10 `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

<code>in</code>	<code>a</code>	A meta-accumulator.
<code>in</code>	<code>label</code>	The labeled image.
<code>in</code>	<code>nlabels</code>	The number of labels in <code>label</code> .

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 792 of file labeling/compute.hh.

References [compute\(\)](#).

9.96.2.11 `template<typename A , typename I , typename L > util::array< typename A::result > mln::labeling::compute (util::array< A > & a, const Image< I > & input, const Image< L > & label, const typename L::value & nlabels) [inline]`

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

for each component of the image `label`.

Parameters

<code>in</code>	<code>a</code>	An array of accumulator.
<code>in</code>	<code>input</code>	The input image.
<code>in</code>	<code>label</code>	The labeled image.
<code>in</code>	<code>nlabels</code>	The number of labels in <code>label</code> .

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 696 of file labeling/compute.hh.

9.96.2.12 `template<typename A , typename I , typename L > mln::trait::ch_value< L , typename A ::result >::ret
mln::labeling::compute_image (const util::array< typename A ::result > & a, const Image< I > & input, const
Image< L > & labels, const typename L::value & nlabels)`

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

for each component of the image `label`.

Parameters

<code>in</code>	<code>a</code>	The mln::p_array of accumulator result.
<code>in</code>	<code>input</code>	The input image (values).
<code>in</code>	<code>labels</code>	The label image.
<code>in</code>	<code>nlabels</code>	The count of labels.

Returns

The image where labels are replaced by the result of the accumulator.

9.96.2.13 `template<typename A , typename I , typename L > mln::trait::ch_value< L , typename A ::result >::ret
mln::labeling::compute_image (const Accumulator< A > & accu, const Image< I > & input, const Image< L > &
labels, const typename L::value & nlabels) [inline]`

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

for each component of the image `label`.

Parameters

<code>in</code>	<code>accu</code>	The accumulator.
<code>in</code>	<code>input</code>	The input image (values).
<code>in</code>	<code>labels</code>	The label image.
<code>in</code>	<code>nlabels</code>	The count of labels.

Returns

The image where labels are replaced by the result of the accumulator.

Definition at line 161 of file compute_image.hh.

References `compute()`.

9.96.2.14 `template<typename A , typename I , typename L > mln::trait::ch_value< L , mln_meta_accu_result(A, typename
L::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

<i>in</i>	<i>accu</i>	The meta-accumulator.
<i>in</i>	<i>input</i>	The input image (values).
<i>in</i>	<i>labels</i>	The label image.
<i>in</i>	<i>nlabels</i>	The count of labels.

Returns

The image where labels are replaced by the result of the accumulator.

Definition at line 181 of file `compute_image.hh`.

References `compute()`.

```
9.96.2.15  template<typename A , typename I , typename L , typename W > mln::util::array< typename A::result >
            mln::labeling::compute_in_window ( const Accumulator< A > & a, const Image< I > & input, const Image< L > &
            label, const typename L::value & nlabels, const Window< W > & win )
```

Compute an accumulator for each image pixel values using neighbor pixel values.

Definition at line 171 of file `compute_in_window.hh`.

```
9.96.2.16  template<typename I , typename N , typename L > mln::trait::concrete< I >::ret mln::labeling::fill_holes ( const
            Image< I > & input, const Neighborhood< N > & nbh, L & nlabels )  [inline]
```

Filling holes of a single object in a binary image.

Parameters

<i>in</i>	<i>input</i>	The input image.
<i>in</i>	<i>nbh</i>	The connexity of the background.
<i>out</i>	<i>nlabels</i>	The number of labels.

Returns

The binary image with a simple object without holes.

Precondition

The input image has to be binary (checked at compile-time).

This routine actually calls [mln::labeling::background](#)

See Also

[mln::labeling::background](#)

Definition at line 73 of file `fill_holes.hh`.

References `background()`, `compute()`, `mln::util::array< T >::nelements()`, and `mln::data::transform()`.

```
9.96.2.17  template<typename I , typename N , typename L > mln::trait::ch_value< I , L >::ret mln::labeling::flat_zones ( const
            Image< I > & input, const Neighborhood< N > & nbh, L & nlabels )
```

Connected component labeling of the flat zones of an image.

Parameters

in	<i>input</i>	The input image.
in	<i>nbh</i>	The connexity of the flat zones.
out	<i>nlabels</i>	The number of labels.

Returns

The label image.

Definition at line 124 of file flat_zones.hh.

9.96.2.18 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret mln::labeling::foreground (const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels) [inline]`

Connected component labeling of the object part in a binary image.

Parameters

in	<i>input</i>	The input image.
in	<i>nbh</i>	The connexity of the foreground.
out	<i>nlabels</i>	The number of labels.

Returns

The label image.

Precondition

The input image has to be binary (checked at compile-time).

This routine actually calls [mln::labeling::value](#) with the value set to `true`.

See Also

[mln::labeling::value](#)

Definition at line 69 of file foreground.hh.

References `value()`.

9.96.2.19 `template<typename I > mln::trait::concrete< I >::ret mln::labeling::pack (const Image< I > & label, typename I::value & new_nlabels, fun::i2v::array< typename I::value > & repack_fun)`

Relabel a labeled image in order to have a contiguous labeling.

Parameters

in	<i>label</i>	The labeled image.
out	<i>new_nlabels</i>	The number of labels after relabeling.
out	<i>repack_fun</i>	The function used to repack the labels.

Returns

The relabeled image.

Definition at line 124 of file pack.hh.

References `mln::data::compute()`, `mln::make::relabelfun()`, and `mln::data::transform()`.

Referenced by `pack()`.

9.96.2.20 `template<typename I> mln::trait::concrete<I>::ret mln::labeling::pack (const Image<I> & label, typename I::value & new_nlabels)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 115 of file `pack.hh`.

References `pack()`.

9.96.2.21 `template<typename I> void mln::labeling::pack_inplace (Image<I> & label, typename I::value & new_nlabels, fun::i2v::array< typename I::value> & repack_fun)`

Relabel inplace a labeled image in order to have a contiguous labeling.

Parameters

in	<i>label</i>	The labeled image.
out	<i>new_nlabels</i>	The number of labels after relabeling.
out	<i>repack_fun</i>	The function used to repack the labels.

Definition at line 157 of file `pack.hh`.

References `mln::data::compute()`, `mln::make::relabelfun()`, and `mln::data::transform()`.

Referenced by `pack_inplace()`.

9.96.2.22 `template<typename I> void mln::labeling::pack_inplace (Image<I> & label, typename I::value & new_nlabels)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 148 of file `pack.hh`.

References `pack_inplace()`.

9.96.2.23 `template<typename I, typename N, typename L> mln::trait::ch_value<I, L>::ret mln::labeling::regional_maxima (const Image<I> & input, const Neighborhood<N> & nbh, L & nlabels)`

Connected component labeling of the regional maxima of an image.

Parameters

in	<i>input</i>	The input image.
in	<i>nbh</i>	The connexity of the regional maxima.
out	<i>nlabels</i>	The number of labeled regions.

Returns

The label image.

Definition at line 130 of file `regional_maxima.hh`.

9.96.2.24 `template<typename I, typename N, typename L> mln::trait::ch_value< I, L >::ret mln::labeling::regional_minima (const Image< I > & input, const Neighborhood< N > & nbh, L & nlabels)`

Connected component labeling of the regional minima of an image.

Parameters

<i>in</i>	<i>input</i>	The input image.
<i>in</i>	<i>nbh</i>	The connexity of the regional minima.
<i>out</i>	<i>nlabels</i>	The number of labeled regions.

Returns

The label image.

Definition at line 140 of file `regional_minima.hh`.

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

9.96.2.25 `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

<i>in</i>	<i>label</i>	the labeled image.
<i>in</i>	<i>nlabels</i>	the number of labels in <i>label</i> .
<i>out</i>	<i>new_nlabels</i>	the number of labels after relabeling.
<i>in</i>	<i>fv2b</i>	function returning whether a label must be replaced by the background.

Returns

the relabeled image.

Definition at line 200 of file `relabel.hh`.

References `mln::make::relabelfun()`.

Referenced by `superpose()`.

9.96.2.26 `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

<i>in</i>	<i>label</i>	the labeled image.
<i>in</i>	<i>nlabels</i>	the number of labels in <i>label</i> .
<i>in</i>	<i>fv2v</i>	function returning the new component id for each pixel value.

Returns

the relabeled image.

Definition at line 179 of file `relabel.hh`.

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

9.96.2.27 `template<typename I, typename F> void mln::labeling::relabel.inplace (Image< I > & label, const typename I::value & nlabels, const Function_v2b< F > & fv2b) [inline]`

Remove components and relabel a labeled image inplace.

Parameters

<code>in, out</code>	<code>label</code>	the labeled image.
<code>in</code>	<code>nlabels</code>	the number of labels in <code>label</code> .
<code>in</code>	<code>fv2b</code>	function returning whether a label must be replaced by the background.

Definition at line 240 of file `relabel.hh`.

References `mln::make::relabelfun()`.

Referenced by `mln::labeled_image_base< I, E >::relabel()`.

9.96.2.28 `template<typename I, typename F> void mln::labeling::relabel.inplace (Image< I > & label, const typename I::value & nlabels, const Function_v2v< F > & fv2v) [inline]`

Remove components and relabel a labeled image inplace.

Parameters

<code>in, out</code>	<code>label</code>	the labeled image.
<code>in</code>	<code>nlabels</code>	the number of labels in <code>label</code> .
<code>in</code>	<code>fv2v</code>	function returning the new component id for each pixel value.

Definition at line 221 of file `relabel.hh`.

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

9.96.2.29 `template<typename I, typename J> mln::trait::concrete< I >::ret mln::labeling::superpose (const Image< I > & lhs, const typename I::value & lhs_nlabels, const Image< J > & rhs, const typename J::value & rhs_nlabels, typename I::value & new_nlabels)`

Superpose two labeled image.

Labels in `lhs` are preserved in the output. Labels of `rhs` are renumbered from the last label value of `lhs`. It avoids duplicate label values in several components.

Parameters

<code>in</code>	<code>lhs</code>	A labeled image.
<code>in</code>	<code>lhs_nlabels</code>	The number of labels in <code>lhs</code> .
<code>in</code>	<code>rhs</code>	A labeled image.
<code>in</code>	<code>rhs_nlabels</code>	The number of labels in <code>rhs</code> .
<code>out</code>	<code>new_nlabels</code>	The number of labels in the output image.

Returns

An image with all the components of `rhs` and `lhs`.

Precondition

`rhs` and `lhs` must have the same domain.
The value type of `rhs` must be convertible towards `lhs`'s.

Definition at line 83 of file `labeling/superpose.hh`.

References `mln::duplicate()`, `mln::value::equiv()`, `mln::data::paste()`, `relabel()`, and `mln::literal::zero`.

9.96.2.30 `template<typename I , typename N , typename L > mln::trait::ch_value< I, L >::ret mln::labeling::value (const Image< I > & input, const typename I::value & val, const Neighborhood< N > & nbh, L & nlabels)`

Connected component labeling of the image sites at a given value.

Parameters

<code>in</code>	<code><i>input</i></code>	The input image.
<code>in</code>	<code><i>val</i></code>	The value to consider.
<code>in</code>	<code><i>nbh</i></code>	The connectivity of components.
<code>out</code>	<code><i>nlabels</i></code>	The number of labels.

Returns

The label image.

Definition at line 149 of file `labeling/value.hh`.

Referenced by `background()`, and `foreground()`.

9.96.2.31 `template<typename I , typename N , typename L , typename A > util::couple< mln::trait::ch_value< I, L >::ret, util::couple< util::array< typename A::result >, util::array< A > > > mln::labeling::value_and_compute (const Image< I > & input, const typename I::value & val, const Neighborhood< N > & nbh, L & nlabels, const Accumulator< A > & accu)`

Connected component labeling of the image sites at a given value.

Parameters

<code>in</code>	<code><i>input</i></code>	The input image.
<code>in</code>	<code><i>val</i></code>	The value to consider.
<code>in</code>	<code><i>nbh</i></code>	The connectivity of components.
<code>out</code>	<code><i>nlabels</i></code>	The number of labels.

Returns

The label image.

Definition at line 212 of file `value_and_compute.hh`.

References `mln::make::couple()`.

9.96.2.32 `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

<code>in</code>	<code>value_type</code>	The type used to wrap the label type.
<code>in</code>	<code>input</code>	The label image.

Returns

A new image with values wrapped with type V.

Definition at line 75 of file labeling/wrap.hh.

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

Referenced by `wrap()`.

9.96.2.33 `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

<code>in</code>	<code>input</code>	The label image.
-----------------	--------------------	------------------

Returns

A new image with values wrapped with type `label_8`.

Definition at line 93 of file labeling/wrap.hh.

References `wrap()`.

9.97 mln::labeling::impl Namespace Reference

Implementation namespace of labeling namespace.

Namespaces

- namespace [generic](#)

Generic implementation namespace of labeling namespace.

Functions

- `template<typename A, typename I, typename L> util::array<typename A::result> compute_fastest (const Accumulator<A> &a_, const Image<I> &input_, const Image<L> &label_, const typename L::value &nlabels)`

Fastest implementation of [labeling::compute](#).

- `template<typename A, typename I, typename L> util::array<typename A::result> compute_fastest (util::array<A> &accus, const Image<I> &input_, const Image<L> &label_, const typename L::value &nlabels)`

Fastest implementation of [labeling::compute](#).

9.97.1 Detailed Description

Implementation namespace of labeling namespace.

9.97.2 Function Documentation

9.97.2.1 `template<typename A , typename I , typename L > util::array<typename A ::result>
mln::labeling::impl::compute_fastest (const Accumulator< A > & a_, const Image< I > & input_, const Image< L >
& label_, const typename L::value & nlabels) [inline]`

Fastest implementation of [labeling::compute](#).

Parameters

in	<i>a_</i>	An accumulator.
in	<i>input_</i>	The input image.
in	<i>label_</i>	The labeled image.
in	<i>nlabels</i>	The number of labels in <code>label</code> .

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 373 of file `labeling/compute.hh`.

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

9.97.2.2 `template<typename A , typename I , typename L > util::array<typename A ::result>
mln::labeling::impl::compute_fastest (util::array< A > & accus, const Image< I > & input_, const Image< L > &
label_, const typename L::value & nlabels) [inline]`

Fastest implementation of [labeling::compute](#).

Parameters

in	<i>accus</i>	An array of accumulators.
in	<i>input_</i>	The input image.
in	<i>label_</i>	The labeled image.
in	<i>nlabels</i>	The number of labels in <code>label</code> .

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 427 of file `labeling/compute.hh`.

References `mln::geom::ncols()`, `mln::util::array< T >::resize()`, and `mln::util::array< T >::size()`.

9.98 mln::labeling::impl::generic Namespace Reference

Generic implementation namespace of labeling namespace.

Functions

- `template<typename A , typename L >
util::array< typename A::result > compute (const Accumulator< A > &a_, const Image< L > &label_, const`

typename L::value &nlabels)

Generic implementation of [labeling::compute](#).

- template<typename A , typename L >
[util::array](#)< typename A::result > [compute](#) ([util::array](#)< A > &accus, const [Image](#)< L > &label_, const typename L::value &nlabels)

Generic implementation of [labeling::compute](#).

- template<typename A , typename I , typename L >
[util::array](#)< typename A::result > [compute](#) (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 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](#).

9.98.1 Detailed Description

Generic implementation namespace of labeling namespace.

9.98.2 Function Documentation

9.98.2.1 `template<typename A , typename L > util::array<typename A ::result> mln::labeling::impl::generic::compute (const Accumulator< A > &a_, const Image< L > &label_, const typename L::value & nlabels) [inline]`

Generic implementation of [labeling::compute](#).

Parameters

in	<i>a_</i>	An accumulator.
in	<i>label_</i>	The labeled image.
in	<i>nlabels</i>	The number of labels in <code>label</code> .

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 204 of file labeling/compute.hh.

9.98.2.2 `template<typename A , typename L > util::array<typename A ::result> mln::labeling::impl::generic::compute (util::array< A > &accus, const Image< L > &label_, const typename L::value & nlabels) [inline]`

Generic implementation of [labeling::compute](#).

Parameters

in	<i>accus_</i>	An array of accumulators. If the size is set to nlabels + 1, the accumulators are considered as initialized. Otherwise, the size is adjusted.
in	<i>label_</i>	The labeled image.
in	<i>nlabels</i>	The number of labels in <code>label</code> .

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 241 of file labeling/compute.hh.

References `mln::util::array< T >::resize()`, and `mln::util::array< T >::size()`.

9.98.2.3 `template<typename A , typename I , typename L > util::array<typename A ::result>
mln::labeling::impl::generic::compute (const Accumulator< A > & a_, const Image< I > & input_, const Image< L
> & label_, const typename L::value & nlabels) [inline]`

Generic implementation of [labeling::compute](#).

Parameters

in	<code>a_</code>	An accumulator.
in	<code>input_</code>	The input image.
in	<code>label_</code>	The labeled image.
in	<code>nlabels</code>	The number of labels in <code>label</code> .

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 283 of file `labeling/compute.hh`.

9.98.2.4 `template<typename A , typename I , typename L > util::array<typename A ::result>
mln::labeling::impl::generic::compute (util::array< A > & accus, const Image< I > & input_, const Image< L > &
label_, const typename L::value & nlabels) [inline]`

Generic implementation of [labeling::compute](#).

Parameters

in	<code>accus</code>	An array of accumulators.
in	<code>input_</code>	The input image.
in	<code>label_</code>	The labeled image.
in	<code>nlabels</code>	The number of labels in <code>label</code> .

Returns

A [util::array](#) of accumulator result (one result per label).

Definition at line 321 of file `labeling/compute.hh`.

References `mln::util::array< T >::resize()`, and `mln::util::array< T >::size()`.

9.99 mln::linear Namespace Reference

Namespace of linear image processing routines.

Namespaces

- namespace [impl](#)
Namespace of linear image processing routines implementation details.
- namespace [local](#)
Specializations of local linear routines.

Functions

- template<typename I >
 mln::trait::concrete< I >::ret [gaussian](#) (const [Image](#)< I > &input, float sigma)
Gaussian filter of an image `input`.
- template<typename I >
 mln::trait::concrete< I >::ret [gaussian](#) (const [Image](#)< I > &input, float sigma, int dir)
- template<typename I >
 mln::trait::concrete< I >::ret [gaussian_1st_derivative](#) (const [Image](#)< I > &input, float sigma, int dir)
- template<typename I >
 mln::trait::concrete< I >::ret [gaussian_2nd_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 , typename W , unsigned Sh, unsigned Sv>
[mln_ch_convolve](#) (I, W) convolve_2x1d(const [Image](#)< I > &input)
- template<typename I , typename W >
[mln_ch_convolve](#) (I, W) convolve(const [Image](#)< I > &input = convolve(input, w_win)
- template<typename I , typename W , unsigned S>
[mln_ch_convolve](#) (I, W) convolve_directional(const [Image](#)< I > &input)
- template<typename I >
[mln_ch_convolve_grad](#) (I, int) sobel_2d(const [Image](#)< I > &input)
Compute the vertical component of the 2D Sobel gradient.
- template<typename I >
[mln_ch_convolve](#) (I, int) sobel_2d_h(const [Image](#)< I > &input)
Sobel_2d gradient components.

9.99.1 Detailed Description

Namespace of linear image processing routines.

9.99.2 Function Documentation

9.99.2.1 template<typename I > mln::trait::concrete< I >::ret mln::linear::gaussian (const [Image](#)< I > &input, float *sigma*) [inline]

Gaussian filter of an image `input`.

Precondition

output.domain = input.domain

Apply an approximated gaussian filter of `sigma` on `input`. This filter is applied in all the input image direction.

Precondition

input.is_valid

Definition at line 750 of file gaussian.hh.

References mln::initialize().

Referenced by mln::subsampling::gaussian_subsampling().

9.99.2.2 `template<typename I> mln::trait::concrete<I>::ret mln::linear::gaussian (const Image<I> & input, float sigma, int dir) [inline]`

Apply an approximated gaussian filter of `sigma` on `input`. on a specific direction `dir` if `dir = 0`, the filter is applied on the first image dimension. if `dir = 1`, the filter is applied on the second image dimension. And so on...

Precondition

```
input.is_valid
dir < dimension(input)
```

Definition at line 653 of file gaussian.hh.

References `mln::initialize()`.

9.99.2.3 `template<typename I> mln::trait::concrete<I>::ret mln::linear::gaussian_1st_derivative (const Image<I> & input, float sigma, int dir) [inline]`

Apply an approximated first derivative gaussian filter of `sigma` on `input`. on a specific direction `dir` if `dir = 0`, the filter is applied on the first image dimension. if `dir = 1`, the filter is applied on the second image dimension. And so on...

Precondition

```
input.is_valid
dir < dimension(input)
```

Definition at line 685 of file gaussian.hh.

References `mln::initialize()`.

9.99.2.4 `template<typename I> mln::trait::concrete<I>::ret mln::linear::gaussian_2nd_derivative (const Image<I> & input, float sigma, int dir) [inline]`

Apply an approximated second derivative gaussian filter of `sigma` on `input`. on a specific direction `dir` if `dir = 0`, the filter is applied on the first image dimension. if `dir = 1`, the filter is applied on the second image dimension. And so on...

Precondition

```
input.is_valid
dir < dimension(input)
```

Definition at line 718 of file gaussian.hh.

References `mln::initialize()`.

9.99.2.5 `template<typename I> mln::trait::concrete<I>::ret mln::linear::gaussian_2nd_derivative (const Image<I> & input, float sigma) [inline]`

Apply an approximated second derivative gaussian filter of `sigma` on `input`. This filter is applied in all the input image direction.

Precondition

```
input.is_valid
```

Definition at line 807 of file gaussian.hh.

References `mln::initialize()`.

9.99.2.6 `template<typename I> mln::linear::mln_ch_convolve (I, int) const [inline]`

Sobel_2d gradient components.

Compute the L1 norm of the 2D Sobel gradient.

Compute the vertical component of the 2D Sobel gradient.

Compute the horizontal component of the 2D Sobel gradient.

Definition at line 70 of file lap.hh.

References `mln_ch_convolve()`, and `mln::make::w_window2d()`.

9.99.2.7 `template<typename I, typename W, unsigned Sh, unsigned Sv> mln::linear::mln_ch_convolve (I, W) const`

Convolution of an image `input` by two weighted line-shapes windows.

Warning

The weighted window is used as-is, considering that its symmetrization is handled by the client.

Precondition

`input.is_valid`

9.99.2.8 `template<typename I, typename W> mln::linear::mln_ch_convolve (I, W) const = convolve(input, w_win)`

Convolution of an image `input` by the weighted window `w_win`.

```
\warning Computation of \p output(p) is performed with the
value type of \p output.
```

```
\warning The weighted window is used as-is, considering that
its symmetrization is handled by the client.
```

```
\pre input.is_valid
```

Referenced by `mln_ch_convolve()`, and `mln_ch_convolve_grad()`.

9.99.2.9 `template<typename I, typename W, unsigned S> mln::linear::mln_ch_convolve (I, W) const [inline]`

Convolution of an image `input` by a line-shaped (directional) weighted window defined by the array of `weights`.

Warning

Computation of `output(p)` is performed with the value type of `output`.

The weighted window is used as-is, considering that its symmetrization is handled by the client.

Precondition

`input.is_valid`

9.99.2.10 `template<typename I> mln::linear::mln_ch_convolve_grad (I, int) const`

Compute the vertical component of the 2D Sobel gradient.

Definition at line 124 of file `sobel_2d.hh`.

References `mln_ch_convolve()`, and `mln::data::transform()`.

9.100 mln::linear::impl Namespace Reference

Namespace of linear image processing routines implementation details.

9.100.1 Detailed Description

Namespace of linear image processing routines implementation details.

9.101 mln::linear::local Namespace Reference

Specializations of local linear routines.

Namespaces

- namespace [impl](#)
Namespace of local linear routines implementation details.

Functions

- template<typename I , typename P , typename W , typename R >
void [convolve](#) (const [Image](#)< I > &input, const [Site](#)< P > &p, const [Weighted_Window](#)< W > &w_win, R &result)
- template<typename P , typename W , typename R >
void [convolve](#) (const [Generalized_Pixel](#)< P > &p, const [Weighted_Window](#)< W > &w_win, R &result)

9.101.1 Detailed Description

Specializations of local linear routines.

9.101.2 Function Documentation

9.101.2.1 template<typename I , typename P , typename W , typename R > void mln::linear::local::convolve (const [Image](#)< I > & *input*, const [Site](#)< P > & *p*, const [Weighted_Window](#)< W > & *w_win*, R & *result*) [\[inline\]](#)

Local convolution of image *input* at point *p* by the weighted window *w_win*.

Warning

Computation of the *result* is performed with the type *R*.
The weighted window is used as-is, considering that its symmetrization is handled by the client.

Definition at line 149 of file linear/local/convolve.hh.

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

9.101.2.2 template<typename P , typename W , typename R > void mln::linear::local::convolve (const [Generalized_Pixel](#)< P > & *p*, const [Weighted_Window](#)< W > & *w_win*, R & *result*) [\[inline\]](#)

Local convolution around (generalized) pixel *p* by the weighted window *w_win*.

Warning

Computation of the `result` is performed with the type `R`.
The weighted window is used as-is, considering that its symmetrization is handled by the client.

Definition at line 161 of file `linear/local/convolve.hh`.

References `convolve()`.

9.102 `mln::linear::local::impl` Namespace Reference

Namespace of local linear routines implementation details.

9.102.1 Detailed Description

Namespace of local linear routines implementation details.

9.103 `mln::literal` Namespace Reference

Namespace of literals.

Classes

- struct [black_t](#)
Type of literal black.
- struct [blue_t](#)
Type of literal blue.
- struct [brown_t](#)
Type of literal brown.
- struct [cyan_t](#)
Type of literal cyan.
- struct [green_t](#)
Type of literal green.
- struct [identity_t](#)
Type of literal identity.
- struct [light_gray_t](#)
Type of literal grays.
- struct [lime_t](#)
Type of literal lime.
- struct [magenta_t](#)
Type of literal magenta.
- struct [max_t](#)
Type of literal max.
- struct [min_t](#)
Type of literal min.
- struct [olive_t](#)
Type of literal olive.
- struct [one_t](#)
Type of literal one.
- struct [orange_t](#)

- Type of literal orange.*
- struct [origin_t](#)
Type of literal origin.
- struct [pink_t](#)
Type of literal pink.
- struct [purple_t](#)
Type of literal purple.
- struct [red_t](#)
Type of literal red.
- struct [teal_t](#)
Type of literal teal.
- struct [violet_t](#)
Type of literal violet.
- struct [white_t](#)
Type of literal white.
- struct [yellow_t](#)
Type of literal yellow.
- struct [zero_t](#)
Type of literal zero.

Variables

- const [black_t](#) [black](#)
Literal black.
- const [blue_t](#) [blue](#)
Literal blue.
- const [brown_t](#) [brown](#)
Literal brown.
- const [cyan_t](#) [cyan](#)
Literal cyan.
- const [dark_gray_t](#) [dark_gray](#)
Literal dark gray.
- const [green_t](#) [green](#)
Literal green.
- const [identity_t](#) [identity](#)
Literal identity.
- const [light_gray_t](#) [light_gray](#)
Literal light gray.
- const [lime_t](#) [lime](#)
Literal lime.
- const [magenta_t](#) [magenta](#)
Literal magenta.
- const [max_t](#) [max](#)
Literal max.
- const [medium_gray_t](#) [medium_gray](#)
Literal medium gray.
- const [min_t](#) [min](#)
Literal min.
- const [olive_t](#) [olive](#)
Literal olive.

- const [one_t](#) one
Literal one.
- const [orange_t](#) orange
Literal orange.
- const [origin_t](#) origin
Literal origin.
- const [pink_t](#) pink
Literal pink.
- const [purple_t](#) purple
Literal purple.
- const [red_t](#) red
Literal red.
- const [teal_t](#) teal
Literal teal.
- const [violet_t](#) violet
Literal violet.
- const [white_t](#) white
Literal white.
- const [yellow_t](#) yellow
Literal yellow.
- const [zero_t](#) zero
Literal zero.

9.103.1 Detailed Description

Namespace of literals.

9.103.2 Variable Documentation

9.103.2.1 const [black_t](#) mln::literal::black

Literal black.

Definition at line 64 of file black.hh.

Referenced by mln::labeling::colorize(), and mln::registration::icp().

9.103.2.2 const [blue_t](#) mln::literal::blue

Literal blue.

Definition at line 274 of file colors.hh.

9.103.2.3 const [brown_t](#) mln::literal::brown

Literal brown.

Definition at line 276 of file colors.hh.

9.103.2.4 const [cyan_t](#) mln::literal::cyan

Literal cyan.

Definition at line 290 of file colors.hh.

9.103.2.5 const dark_gray_t mln::literal::dark_gray

[Literal](#) dark gray.

Definition at line 82 of file grays.hh.

9.103.2.6 const green_t mln::literal::green

[Literal](#) green.

Definition at line 272 of file colors.hh.

Referenced by mln::registration::icp(), and mln::make_debug_graph_image().

9.103.2.7 const identity_t mln::literal::identity

[Literal](#) identity.

Definition at line 63 of file identity.hh.

9.103.2.8 const light_gray_t mln::literal::light_gray

[Literal](#) light gray.

Definition at line 78 of file grays.hh.

9.103.2.9 const lime_t mln::literal::lime

[Literal](#) lime.

Definition at line 278 of file colors.hh.

9.103.2.10 const magenta_t mln::literal::magenta

[Literal](#) magenta.

Definition at line 292 of file colors.hh.

9.103.2.11 const max_t mln::literal::max

[Literal](#) max.

Definition at line 77 of file literal/max.hh.

9.103.2.12 const medium_gray_t mln::literal::medium_gray

[Literal](#) medium_gray.

Definition at line 80 of file grays.hh.

9.103.2.13 const min_t mln::literal::min

[Literal](#) min.

Definition at line 75 of file literal/min.hh.

9.103.2.14 const olive_t mln::literal::olive

[Literal](#) olive.

Definition at line 296 of file colors.hh.

9.103.2.15 const one_t mln::literal::one

[Literal](#) one.

Definition at line 79 of file one.hh.

Referenced by `mln::algebra::h_vec< d, C >::h_vec()`, `mln::operator++()`, and `mln::operator--()`.

9.103.2.16 const orange_t mln::literal::orange

[Literal](#) orange.

Definition at line 280 of file colors.hh.

9.103.2.17 const origin_t mln::literal::origin

[Literal](#) origin.

Definition at line 64 of file origin.hh.

Referenced by `mln::win::ball< G, C >::ball()`, `mln::geom::bbox()`, `mln::box< P >::box()`, `mln::geom::rotate()`, and `mln::make::w_window()`.

9.103.2.18 const pink_t mln::literal::pink

[Literal](#) pink.

Definition at line 282 of file colors.hh.

9.103.2.19 const purple_t mln::literal::purple

[Literal](#) purple.

Definition at line 284 of file colors.hh.

9.103.2.20 const red_t mln::literal::red

[Literal](#) red.

Definition at line 270 of file colors.hh.

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

9.103.2.21 const teal_t mln::literal::teal

[Literal](#) teal.

Definition at line 286 of file colors.hh.

9.103.2.22 `const violet_t mln::literal::violet`

[Literal](#) violet.

Definition at line 288 of file colors.hh.

9.103.2.23 `const white_t mln::literal::white`

[Literal](#) white.

Definition at line 64 of file white.hh.

Referenced by `mln::registration::icp()`.

9.103.2.24 `const yellow_t mln::literal::yellow`

[Literal](#) yellow.

Definition at line 294 of file colors.hh.

9.103.2.25 `const zero_t mln::literal::zero`

[Literal](#) zero.

Definition at line 79 of file zero.hh.

Referenced by `mln::transform::influence_zone_geodesic_saturated()`, `mln::accu::rms< T, V >::init()`, `mln::accu::convolve< T1, T2, R >::init()`, `mln::accu::center< P, V >::init()`, `mln::accu::stat::variance< T, S, R >::init()`, `mln::accu::shape::volume< I >::init()`, `mln::morpho::attribute::sum< I, S >::init()`, `mln::accu::math::sum< T, S >::init()`, `mln::accu::stat::histo3d_rgb< V >::init()`, `mln::window< D >::is_centered()`, `mln::accu::stat::variance< T, S, R >::mean()`, `mln::accu::stat::var< T >::mean()`, `mln::geom::mesh_corner_point_area()`, `mln::geom::mesh_curvature()`, `mln::geom::mesh_normal()`, `mln::morpho::meyer_wst()`, `mln::algebra::operator*()`, `mln::test::positive()`, `mln::make::relabelfun()`, `mln::geom::rotate()`, `mln::accu::shape::volume< I >::set_value()`, `mln::morpho::watershed::superpose()`, `mln::debug::superpose()`, `mln::labeling::superpose()`, `mln::accu::stat::var< T >::to_result()`, `mln::algebra::tr()`, `mln::geom::translate()`, and `mln::make::w_window_directional()`.

9.104 mln::logical Namespace Reference

Namespace of logic.

Namespaces

- namespace [impl](#)

Implementation namespace of logical namespace.

Functions

- `template<typename L, typename R >`
`void and_inplace (Image< L > &lhs, const Image< R > &rhs)`
- `template<typename L, typename R >`
`mln::trait::ch_value< L,`
`typename`
`mln::fun::vv2v::land_not`
`< typename L::value, typename`
`R::value >::result >::ret and_not (const Image< L > &lhs, const Image< R > &rhs)`

- `template<typename L , typename R >`
`void and_not_inplace (Image< L > &lhs, const Image< R > &rhs)`
- `template<typename I >`
`void not_inplace (Image< I > &input)`
- `template<typename L , typename R >`
`void or_inplace (Image< L > &lhs, const Image< R > &rhs)`
- `template<typename L , typename R >`
`void xor_inplace (Image< L > &lhs, const Image< R > &rhs)`

9.104.1 Detailed Description

Namespace of logic.

9.104.2 Function Documentation

9.104.2.1 `template<typename L , typename R > void mln::logical::and_inplace (Image< L > & lhs, const Image< R > & rhs)` `[inline]`

Point-wise in-place "logical and" of image `rhs` in image `lhs`.

```
\param[in,out] lhs First operand image.
\param[in] rhs Second operand image.
```

```
It performs: \n
for all p of rhs.domain \n
lhs(p) = lhs(p) and rhs(p)
```

```
\pre \p rhs.domain >= \p lhs.domain
```

Definition at line 91 of file `logical/and.hh`.

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

9.104.2.2 `template<typename L , typename R > mln::trait::ch_value< L, typename mln::fun::vv2v::land_not< typename L::value, typename R::value >::result >::ret mln::logical::and_not (const Image< L > & lhs, const Image< R > & rhs)` `[inline]`

Point-wise "logical and-not" between images `lhs` and `rhs`.

```
\param[in] lhs First operand image.
\param[in] rhs Second operand image.
\result The result image.
```

```
\pre \p lhs.domain == \p rhs.domain
```

Definition at line 76 of file `and_not.hh`.

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

9.104.2.3 `template<typename L , typename R > void mln::logical::and_not_inplace (Image< L > & lhs, const Image< R > & rhs)` `[inline]`

Point-wise in-place "logical and-not" of image `rhs` in image `lhs`.

```
\param[in,out] lhs First operand image.
\param[in] rhs Second operand image.
```

```
It performs: \n
for all p of rhs.domain \n
lhs(p) = lhs(p) and not rhs(p)
```

```
\pre \p rhs.domain >= \p lhs.domain
```

Definition at line 91 of file and_not.hh.

References mln::data::transform_inplace().

9.104.2.4 `template<typename I> void mln::logical::not_inplace (Image<I> & input) [inline]`

Point-wise in-place "logical not" of image `input`.

```
\param[in,out] input The target image.
```

```
It performs: \n
  for all p of input.domain \n
    input(p) = not input(p)
```

```
\pre \p input.is_valid
```

Definition at line 88 of file logical/not.hh.

References mln::data::transform_inplace().

9.104.2.5 `template<typename L, typename R> void mln::logical::or_inplace (Image<L> & lhs, const Image<R> & rhs) [inline]`

Point-wise in-place "logical or" of image `rhs` in image `lhs`.

```
\param[in,out] lhs First operand image.
\param[in] rhs Second operand image.
```

```
It performs: \n
  for all p of rhs.domain \n
    lhs(p) = lhs(p) or rhs(p)
```

```
\pre \p rhs.domain >= \p lhs.domain
```

Definition at line 91 of file logical/or.hh.

References mln::data::transform_inplace().

9.104.2.6 `template<typename L, typename R> void mln::logical::xor_inplace (Image<L> & lhs, const Image<R> & rhs) [inline]`

Point-wise in-place "logical xor" of image `rhs` in image `lhs`.

```
\param[in,out] lhs First operand image.
\param[in] rhs Second operand image.
```

```
It performs: \n
  for all p of rhs.domain \n
    lhs(p) = lhs(p) xor rhs(p)
```

```
\pre \p rhs.domain >= \p lhs.domain
```

Definition at line 91 of file logical/xor.hh.

References mln::data::transform_inplace().

9.105 mln::logical::impl Namespace Reference

Implementation namespace of logical namespace.

Namespaces

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

9.105.1 Detailed Description

Implementation namespace of logical namespace.

9.106 mln::logical::impl::generic Namespace Reference

Generic implementation namespace of logical namespace.

9.106.1 Detailed Description

Generic implementation namespace of logical namespace.

9.107 mln::make Namespace Reference

Namespace of routines that help to make Milena's objects.

Functions

- `template<unsigned D, typename G , typename V >`
`p_set< complex_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 (unsigned ninds)`
Create an [mln::box1d](#).
- `mln::box1d box1d (def::coord min_ind, def::coord max_ind)`
Create an [mln::box1d](#).
- `mln::box2d box2d (unsigned nrows, unsigned ncols)`
Create an [mln::box2d](#).
- `mln::box2d box2d (def::coord min_row, def::coord min_col, def::coord max_row, def::coord max_col)`
Create an [mln::box2d](#).
- `mln::box2d_h box2d_h (unsigned nrows, unsigned ncols)`
Create an [mln::box2d_h](#).
- `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::box3d box3d (unsigned nslis, unsigned nrows, unsigned ncols)`
Create an [mln::box3d](#).
- `mln::box3d box3d (def::coord min_sli, def::coord min_row, def::coord min_col, def::coord max_sli, def::coord max_row, def::coord max_col)`
Create an [mln::box3d](#).
- `template<unsigned D, typename G >`
`p_set< complex_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, typename P >`
`p_edges< G, pw::cst_< P > > dummy_p_edges` (const `Graph< G >` &g_, const `P` &dummy_site)

Create a `p_edges` which associate a graph element to a constant site.

- `template<typename G >`
`p_edges< G > dummy_p_edges` (const `Graph< G >` &g)

Create a `p_edges` which associate a graph element to a constant site.

- `template<typename G, typename P >`
`p_vertices< G, pw::cst_< P > > dummy_p_vertices` (const `Graph< G >` &g_, const `P` &dummy_site)

Create a `p_vertices` which associate a graph element to a constant site.

- `template<typename G >`
`p_vertices< G > dummy_p_vertices` (const `Graph< G >` &g)

Create a `p_vertices` which associate a graph element to a constant site.

- `template<typename V, typename G >`
`mln::edge_image< void, V, G > edge_image` (const `Graph< G >` &g, const `fun::i2v::array< V >` &fv)

Construct an edge image.

- `template<typename FV, typename G >`
`mln::edge_image< void,`
`typename FV::result, G > edge_image` (const `Graph< G >` &g, const `Function_v2v< FV >` &fv)

Construct an edge image.

- `template<typename FP, typename FV, typename G >`
`mln::edge_image< typename`
`FP::result, typename`
`FV::result, G > edge_image` (const `Graph< G >` &g_, const `Function_v2v< FP >` &fp, const `Function_v2v< FV >` &fv)

Construct an edge image.

- `template<typename P, typename V, typename G, typename FP, typename FV >`
`mln::edge_image< typename`
`FP::result, typename`
`FV::result, G > edge_image` (const `mln::vertex_image< P, V, G >` &v_ima_, const `p_edges< G, FP >` pe, const `Function_vv2v< FV >` &fv_)

Construct an edge image.

- `template<typename P, typename V, typename G, typename FV >`
`mln::edge_image< void,`
`typename FV::result, G > edge_image` (const `mln::vertex_image< P, V, G >` &v_ima_, const `Function_vv2v< FV >` &fv_)

Construct an edge image.

- `template<typename P, typename V, typename G, typename F >`
`mln::edge_image< void, bool, G > edge_image` (const `mln::vertex_image< P, V, G >` &v_ima_, const `Function_v2b< F >` &fv_)

Construct an edge image.

- `template<typename T, unsigned N>`
`algebra::h_mat< mlc_sqrt_int(N),`
`T > h_mat` (const `T(&tab)[N]`)

Create an `mln::algebra::mat<n,n,T>`.

- `template<typename V, unsigned L>`
`mln::image1d< V > image` (V(&values)[L])

Create an `image1d` from an 1D array of values.

- `template<typename V , unsigned R, unsigned C>`
`mln::image2d< V > image (V(&values)[R][C])`
Create an `image2d` from an 2D array of values.
- `template<typename V , unsigned S, unsigned R, unsigned C>`
`mln::image3d< V > image (V(&values)[S][R][C])`
Create an `image3d` from an 3D array of values.
- `template<typename V , unsigned S>`
`mln::image2d< V > image2d (V(&values)[S])`
Create an `image2d` from an 2D array of values.
- `template<typename I >`
`mln::image3d< typename I::value > image3d (const util::array< I > &ima)`
Create an `image3d` from an array of 2D images.
- `template<typename I >`
`mln::image3d< typename I::value > image3d (const Image< I > &ima)`
Create an `image3d` from a 2D image.
- `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< 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`.
- `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`.
- `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_v2b< F >` &fv2b, const `V` &nlabels, `V` &new_nlabels)
Create a i2v function from a v2b function.
- `template<typename V , typename F >`
`fun::i2v::array< V >` `relabelfun` (const `Function_v2v< F >` &fv2v, const `V` &nlabels, `V` &new_nlabels)
Create a i2v function from a v2v function.
- `template<typename T >`
`algebra::vec< 1, T >` `vec` (const `T` &v_0)
Create an `mln::algebra::vec<n,T>`.
- `template<typename T >`
`algebra::vec< 2, T >` `vec` (const `T` &v_0, const `T` &v_1)
Create an `mln::algebra::vec<2,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< 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 G , typename FV >`
`mln::vertex_image< void,`
`typename FV::result, G >` `vertex_image` (const `Graph< G >` &g, const `Function_v2v< FV >` &fv)
Construct a vertex image.
- `template<typename FP , typename FV , typename G >`
`mln::vertex_image< typename`
`FP::result, typename`
`FV::result, G >` `vertex_image` (const `Graph< G >` &g_, const `Function_v2v< FP >` &fp, const `Function_v2v< FV >` &fv)
Construct a vertex image.
- `template<typename I , typename N >`
`p_vertices< util::graph,`
`fun::i2v::array< typename`
`I::site >` `voronoi` (`Image< I >` &ima_, `Image< I >` &orig_, const `Neighborhood< N >` &nbh)
Apply the Voronoi algorithm on `ima_` with the original image `orig_` for node computing with neighborhood `nbh`.
- `template<typename W , typename F >`
`mln::w_window< typename`
`W::dpsite, typename F::result >` `w_window` (const `Window< W >` &win, const `Function_v2v< F >` &wei)
Create a `mln::w_window` from a window and a weight function.
- `template<typename W , unsigned M>`
`mln::w_window< mln::dpoint1d, W >` `w_window1d` (`W(&weights)[M]`)
Create a 1D `mln::w_window` from an array of weights.
- `template<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<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<typename D , typename W , unsigned L>`
`mln::w_window< D, W >` `w_window_directional` (const `Gdpoint< D >` &dp, `W(&weights)[L]`)
Create a directional centered weighted window.

9.107.1 Detailed Description

Namespace of routines that help to make Milena's objects.

9.107.2 Function Documentation

9.107.2.1 `template<unsigned D, typename G, typename V> p_set< complex_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 `couprie.08.pami`). We use the following (equivalent) definition: an N-face *F* in *CELL* is in the attachment of *CELL* to *IMA* if it is adjacent to at least an (N-1)-face or an (N+1)-face that does not belong to *CELL*.

Definition at line 68 of file `attachment.hh`.

References `cell()`, and `mln::topo::is_facet()`.

Referenced by `mln::topo::is_simple_cell< I >::operator()()`.

9.107.2.2 `mln::box1d mln::make::box1d (unsigned ninds) [inline]`

Create an `mln::box1d`.

Parameters

<i>in</i>	<i>ninds</i>	Number of indices.
-----------	--------------	--------------------

Precondition

ninds != 0 and *ncols* != 0.

Returns

A 1D box.

Definition at line 70 of file `make/box1d.hh`.

Referenced by `mln::convert::to_image()`.

9.107.2.3 `mln::box1d mln::make::box1d (def::coord min_ind, def::coord max_ind) [inline]`

Create an `mln::box1d`.

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

in	<i>min_ind</i>	Minimum index.
in	<i>max_ind</i>	Maximum index.

Precondition

`max_ind >= min_ind.`

Returns

A 1D box.

Definition at line 79 of file make/box1d.hh.

9.107.2.4 mln::box2d mln::make::box2d (unsigned *nrows*, unsigned *ncols*) [inline]

Create an [mln::box2d](#).

Parameters

in	<i>nrows</i>	Number of rows.
in	<i>ncols</i>	Number of columns.

Precondition

`nrows != 0 and ncols != 0.`

Returns

A 2D box.

Definition at line 78 of file make/box2d.hh.

Referenced by `mln::io::pnm::load()`.

9.107.2.5 mln::box2d mln::make::box2d (def::coord *min_row*, def::coord *min_col*, def::coord *max_row*, def::coord *max_col*) [inline]

Create an [mln::box2d](#).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

in	<i>min_row</i>	Index of the top most row.
in	<i>min_col</i>	Index of the left most column.
in	<i>max_row</i>	Index of the bottom most row.
in	<i>max_col</i>	Index of the right most column.

Precondition

`max_row >= min_row and max_col >= min_col.`

Returns

A 2D box.

Definition at line 88 of file `make/box2d.hh`.

9.107.2.6 `mln::box2d_h mln::make::box2d_h (unsigned nrows, unsigned ncols)` `[inline]`

Create an `mln::box2d_h`.

Parameters

<code>in</code>	<code><i>nrows</i></code>	Number of rows.
<code>in</code>	<code><i>ncols</i></code>	Number of columns.

Precondition

`nrows != 0 and ncols != 0.`

Returns

A 2D_H box.

Definition at line 75 of file `make/box2d_h.hh`.

References `point2d_h()`.

9.107.2.7 `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

<code>in</code>	<code><i>min_row</i></code>	Index of the top most row.
<code>in</code>	<code><i>min_col</i></code>	Index of the left most column.
<code>in</code>	<code><i>max_row</i></code>	Index of the bottom most row.
<code>in</code>	<code><i>max_col</i></code>	Index of the right most column.

Precondition

`max_row >= min_row and max_col >= min_col.`

Returns

A 2D_H box.

Definition at line 85 of file `make/box2d_h.hh`.

References `point2d_h()`.

9.107.2.8 `mln::box3d mln::make::box3d (unsigned nslis, unsigned nrows, unsigned ncols)` `[inline]`

Create an `mln::box3d`.

Parameters

in	<i>nslis</i>	Number of slices.
in	<i>nrows</i>	Number of rows.
in	<i>ncols</i>	Number of columns.

Precondition

`ninds != 0 and ncols != 0 and nslis != 0.`

Returns

A 3D box.

Definition at line 80 of file `make/box3d.hh`.

Referenced by `image3d()`.

9.107.2.9 `mln::box3d mln::make::box3d (def::coord min_sli, def::coord min_row, def::coord min_col, def::coord max_sli, def::coord max_row, def::coord max_col) [inline]`

Create an [mln::box3d](#).

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

in	<i>min_sli</i>	Index of the lowest slice.
in	<i>min_row</i>	Index of the top most row.
in	<i>min_col</i>	Index of the left most column.
in	<i>max_sli</i>	Index of the highest slice.
in	<i>max_row</i>	Index of the bottom most row.
in	<i>max_col</i>	Index of the right most column.

Precondition

`max_sli >= min_sli.`
`max_row >= min_row.`
`max_col >= min_col.`

Returns

A 3D box.

Definition at line 92 of file `make/box3d.hh`.

9.107.2.10 `template<unsigned D, typename G > p_set< complex_psite< D, G > > mln::make::cell (const complex_psite< D, G > & f) [inline]`

Compute the set of faces of the cell corresponding to the facet *f*.

Precondition

f is a facet (it does not belong to any face of higher dimension).

Returns

An [mln::p_set](#) of sites (faces) containing the attachment.

Definition at line 63 of file cell.hh.

References [mln::topo::is_facet\(\)](#), and [mln::complex_psite< D, G >::n\(\)](#).

Referenced by [attachment\(\)](#), and [detachment\(\)](#).

9.107.2.11 `template<typename T , typename U > util::couple<T,U> mln::make::couple (const T & val1, const T & val2)`

Construct an [mln::util::couple](#) on-the-fly.

Referenced by [mln::labeling::blobs_and_compute\(\)](#), [mln::transform::distance_and_closest_point_geodesic\(\)](#), [mln::transform::distance_and_influence_zone_geodesic\(\)](#), and [mln::labeling::value_and_compute\(\)](#).

9.107.2.12 `template<unsigned D, typename G , typename V > p_set< complex_psite< D, G > > mln::make::detachment
(const complex_psite< D, G > & f, const complex_image< D, G, V > & ima) [inline]`

Compute the detachment of the cell corresponding to the facet *f* to the image *ima*.

Precondition

f is a facet (it does not belong to any face of higher dimension).

ima is an image of Boolean values.

Returns

a set of faces containing the detachment.

We do not use the formal definition of the detachment here (see [coupric.08.pami](#)). We use the following (equivalent) definition: an N-face F in CELL is not in the detachment of CELL from IMA if it is adjacent to at least an (N-1)-face or an (N+1)-face that does not belong to CELL.

Definition at line 68 of file detachment.hh.

References [cell\(\)](#), and [mln::topo::is_facet\(\)](#).

Referenced by [mln::topo::detach\(\)](#).

9.107.2.13 `mln::dpoint2d_h mln::make::dpoint2d_h (def::coord row, def::coord col) [inline]`

Create an [mln::dpoint2d_h](#).

Parameters

<i>in</i>	<i>row</i>	Row coordinate.
<i>in</i>	<i>col</i>	Column coordinate.

Returns

A 2D dpoint.

Definition at line 55 of file make/dpoint2d_h.hh.

9.107.2.14 `template<typename G , typename P > p_edges< G, pw::cst_< P > > mln::make::dummy_p_edges (const Graph< G > & g_, const P & dummy_site)`

Create a [p_edges](#) which associate a graph element to a constant site.

Parameters

in	<i>g_</i>	A graph.
in	<i>dummy_site</i>	The dummy site mapped to graph edges.

Returns

A [p_edges](#).

Definition at line 77 of file dummy_p_edges.hh.

9.107.2.15 `template<typename G > p_edges< G > mln::make::dummy_p_edges (const Graph< G > & g)`

Create a [p_edges](#) which associate a graph element to a constant site.

0 (int) is used as dummy site.

Parameters

in	<i>g</i>	A graph.
----	----------	----------

Returns

A [p_edges](#).

Definition at line 93 of file dummy_p_edges.hh.

9.107.2.16 `template<typename G , typename P > p_vertices< G, pw::cst_< P > > mln::make::dummy_p_vertices (const Graph< G > & g_, const P & dummy_site)`

Create a [p_vertices](#) which associate a graph element to a constant site.

Parameters

in	<i>g_</i>	A graph.
in	<i>dummy_site</i>	The dummy site mapped to graph vertices.

Returns

A [p_vertices](#).

Definition at line 77 of file dummy_p_vertices.hh.

9.107.2.17 `template<typename G > p_vertices< G > mln::make::dummy_p_vertices (const Graph< G > & g)`

Create a [p_vertices](#) which associate a graph element to a constant site.

0 (int) is used as dummy site.

Parameters

in	<i>g</i>	A graph.
----	----------	----------

Returns

A [p_vertices](#).

Definition at line 93 of file dummy_p_vertices.hh.

9.107.2.18 `template<typename V , typename G > mln::edge_image< void, V, G > mln::make::edge_image (const Graph< G > & g, const fun::i2v::array< V > & fv) [inline]`

Construct an edge image.

Parameters

in	<i>g</i>	A graph.
in	<i>fv</i>	A function mapping edge ids to values.

Returns

an edge image.

Definition at line 141 of file make/edge_image.hh.

9.107.2.19 `template<typename FV , typename G > mln::edge_image< void, typename FV::result, G > mln::make::edge_image (const Graph< G > & g, const Function_v2v< FV > & fv)`

Construct an edge image.

Parameters

in	<i>g</i>	A graph.
in	<i>fv</i>	A function mapping edge ids to values.

Returns

an edge image.

Definition at line 155 of file make/edge_image.hh.

9.107.2.20 `template<typename FP , typename FV , typename G > mln::edge_image< typename FP::result, typename FV::result, G > mln::make::edge_image (const Graph< G > & g_, const Function_v2v< FP > & fp, const Function_v2v< FV > & fv) [inline]`

Construct an edge image.

Parameters

in	<i>g_</i>	A graph.
in	<i>fp</i>	A function mapping edge ids to sites.
in	<i>fv</i>	A function mapping edge ids to values.

Returns

an edge image.

Definition at line 179 of file make/edge_image.hh.

```
9.107.2.21  template<typename P , typename V , typename G , typename FP , typename FV > mln::edge_image< typename
FP::result, typename FV::result, G > mln::make::edge_image ( const mln::vertex_image< P, V, G > & v_ima_,
const p_edges< G, FP > pe, const Function_vv2v< FV > & fv_ ) [inline]
```

Construct an edge image.

Parameters

in	<i>v_ima_</i>	A vertex image.
in	<i>pe</i>	A p_edges mapping graph elements to sites.
in	<i>fv_</i>	A function mapping two vertex ids to a value. The result is associated to the corresponding edge.

Returns

an edge image.

Definition at line 199 of file make/edge_image.hh.

```
9.107.2.22  template<typename P , typename V , typename G , typename FV > mln::edge_image< void, typename
FV::result, G > mln::make::edge_image ( const mln::vertex_image< P, V, G > & v_ima_, const Function_vv2v<
FV > & fv_ ) [inline]
```

Construct an edge image.

Parameters

in	<i>v_ima_</i>	A vertex image.
in	<i>fv_</i>	A function mapping two vertices' values to a value. The result is associated to the corresponding edge.

Returns

an edge image without localization information mapped to graph elements.

Definition at line 225 of file make/edge_image.hh.

```
9.107.2.23  template<typename P , typename V , typename G , typename F > mln::edge_image< void, bool, G >
mln::make::edge_image ( const mln::vertex_image< P, V, G > & v_ima_, const Function_v2b< F > & fv_ )
[inline]
```

Construct an edge image.

Parameters

in	<i>v_ima_</i>	A vertex image.
in	<i>fv_</i>	A predicate on a vertex's value. The (Boolean) result is associated to the edges adjacent to the vertex.

Returns

an edge image without localization information mapped to graph elements.

Definition at line 250 of file `make/edge_image.hh`.

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

9.107.2.24 `template<typename T, unsigned N> algebra::h_mat< mlc_sqrt_int(N), T > mln::make::h_mat (const T(&tab[N])) [inline]`

Create an `mln::algebra::mat<n,n,T>`.

Definition at line 60 of file `make/h_mat.hh`.

Referenced by `mln::fun::x2x::rotation< n, C >::rotation()`.

9.107.2.25 `template<typename V, unsigned L> mln::image1d< V > mln::make::image (V(& values[L]))`

Create an [image1d](#) from an 1D array of values.

Parameters

<code>in</code>	<code>values</code>	1D array.
-----------------	---------------------	-----------

Returns

A 1D image.

Definition at line 85 of file `make/image.hh`.

9.107.2.26 `template<typename V, unsigned R, unsigned C> mln::image2d< V > mln::make::image (V(& values[R][C]))`

Create an [image2d](#) from an 2D array of values.

Parameters

<code>in</code>	<code>values</code>	2D array.
-----------------	---------------------	-----------

Returns

A 2D image.

Definition at line 97 of file `make/image.hh`.

References `mln::opt::at()`.

9.107.2.27 `template<typename V, unsigned S, unsigned R, unsigned C> mln::image3d< V > mln::make::image (V(& values[S][R][C]))`

Create an [image3d](#) from an 3D array of values.

Parameters

<code>in</code>	<code>values</code>	3D array.
-----------------	---------------------	-----------

Returns

A 3D image.

Definition at line 112 of file make/image.hh.

References `mln::opt::at()`.

9.107.2.28 `template<typename V , unsigned S> mln::image2d< V > mln::make::image2d (V(&) values[S])`

Create an [image2d](#) from an 2D array of values.

Parameters

<code>in</code>	<code>values</code>	2D array.
-----------------	---------------------	-----------

Returns

A 2D image.

Definition at line 61 of file make/image2d.hh.

9.107.2.29 `template<typename I > mln::image3d< typename I::value > mln::make::image3d (const util::array< I > & ima) [inline]`

Create an [image3d](#) from an array of 2D images.

Definition at line 70 of file make/image3d.hh.

References `box3d()`, `mln::util::array< T >::is_empty()`, `mln::util::array< T >::nelements()`, `mln::data::paste()`, `mln::box< P >::pmax()`, and `mln::box< P >::pmin()`.

Referenced by `mln::io::pnms::load()`.

9.107.2.30 `template<typename I > mln::image3d< typename I::value > mln::make::image3d (const Image< I > & ima) [inline]`

Create an [image3d](#) from a 2D image.

Definition at line 92 of file make/image3d.hh.

References `box3d()`, and `mln::data::paste()`.

9.107.2.31 `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

<code>in</code>	<code>iz</code>	influence zone image.
<code>in</code>	<code>nbh</code>	A neighborhood.
<code>in</code>	<code>nlabels</code>	number of influence zone in <code>iz</code> .

Returns

[util::graph Graph](#) based on the adjacency of the influence zones.

Definition at line 175 of file influence_zone_adjacency_graph.hh.

9.107.2.32 `template<unsigned n, unsigned m, typename T > algebra::mat< n, m, T > mln::make::mat (const T(& tab[n*m])`
`[inline]`

Create an `mln::algebra::mat<n,m,T>`.

Parameters

<code>in</code>	<code>tab</code>	Array of values.
-----------------	------------------	------------------

Precondition

The array dimension has to be $n * m$.

Definition at line 61 of file `make/mat.hh`.

9.107.2.33 `template<typename T > util::ord_pair< T > mln::make::ord_pair (const T & val1, const T & val2)`
`[inline]`

Construct an `mln::util::ord_pair` on-the-fly.

Definition at line 277 of file `ord_pair.hh`.

9.107.2.34 `template<typename W , typename G > p_edges< G, fun::i2v::array< util::site_pair< typename W::site > >`
`> mln::make::p_edges_with_mass_centers (const Image< W > & wst_, const Graph< G > & g_) [inline]`

Construct a `p_edges` from a watershed image and a region adjacency graph (RAG).

Map each graph edge to a pair of mass centers of two adjacent regions.

Parameters

<code>wst_</code>	A watershed image.
<code>g_</code>	A region adjacency graph.

Returns

A `p_edges`.

See Also

[edge_image](#), [p_edges](#), [make::region_adjacency_graph](#)

Definition at line 81 of file `p_edges_with_mass_centers.hh`.

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

9.107.2.35 `template<typename W , typename G > p_vertices< G, fun::i2v::array< typename W::site > >`
`mln::make::p_vertices_with_mass_centers (const Image< W > & wst_, const Graph< G > & g_) [inline]`

Construct a `p_vertices` from a watershed image and a region adjacency graph (RAG).

Map each graph vertex to the mass center of its corresponding region.

Parameters

<code>wst_</code>	A watershed image.
<code>g_</code>	A region adjacency graph.

Returns

A [p_vertices](#).

See Also

[edge_image](#), [vertex_image](#), [p_vertices](#), [p_edges](#), [make::region_adjacency_graph](#)

Definition at line 77 of file `p_vertices_with_mass_centers.hh`.

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

9.107.2.36 `template<typename I> mln::util::pix< I> mln::make::pix (const Image< I> & ima, const typename I::psite & p) [inline]`

Create an [mln::util::pix](#) from an image `ima` and a psite `p`.

Parameters

<code>in</code>	<code>ima</code>	The input image.
<code>in</code>	<code>p</code>	The point site.

Returns

An [mln::util::pix](#).

Definition at line 58 of file `make/pix.hh`.

9.107.2.37 `template<typename I> mln::pixel< const I> mln::make::pixel (const Image< I> & ima, const typename I::psite & p) [inline]`

Create a [mln::pixel](#) from a constant image `ima` and a point `p`.

Definition at line 55 of file `make/pixel.hh`.

9.107.2.38 `template<typename I> mln::pixel< I> mln::make::pixel (Image< I> & ima, const typename I::psite & p) [inline]`

Create a [mln::pixel](#) from a mutable image `ima` and a point `p`.

Definition at line 63 of file `make/pixel.hh`.

9.107.2.39 `mln::point2d_h mln::make::point2d_h (def::coord row, def::coord col) [inline]`

Create an [mln::point2d_h](#).

Parameters

<code>in</code>	<code>row</code>	Row coordinate.
<code>in</code>	<code>col</code>	Column coordinate.

Returns

A 2D point.

Definition at line 55 of file `make/point2d_h.hh`.

Referenced by `box2d_h()`.

9.107.2.40 `template<typename I , typename N > util::couple< util::graph, typename mln::trait::concrete< I >::ret > mln::make::rag_and_labeled_wsl (const Image< I > & wshd_, const Neighborhood< N > & nbh_, const typename I::value & nbasins) [inline]`

Create a region adjacency graph and a label image of the watershed line from a watershed image.

Parameters

in	<i>wshd_</i>	Watershed image.
in	<i>nbh_</i>	Neighborhood
in	<i>nbasins</i>	Number of influence zone in wshd.

Returns

A couple. First element is the graph, second element is an image with a labeled watershed line.

<pre> ----- 1 1 1 0 2 2 0 3 1 1 0 2 2 2 0 3 1 0 4 0 2 0 3 3 0 4 4 4 0 5 0 3 ----- </pre> <p>Watershed image (watershed line labeled with 0)</p>	---->	<pre> ----- . . . 1 . . 2 . . . 1 . . . 2 . . 1 . 3 . 4 . . 1 . . . 5 . 6 . ----- </pre> <p>Labeled watershed line</p>
---	-------	--

```

      |
      |
      |
      v
1  -- 2  - 3
 \  /  /
  4 -- 5

Region Adjacency graph (RAG)

```

Definition at line 229 of file `rag_and_labeled_wsl.hh`.

9.107.2.41 `template<typename I , typename N > util::graph mln::make::region_adjacency_graph (const Image< I > & wshd_, const Neighborhood< N > & nbh, const typename I::value & nbasins) [inline]`

Create a region adjacency graph from a watershed image.

Parameters

in	<i>wshd_</i>	watershed image.
in	<i>nbh</i>	A neighborhood.
in	<i>nbasins</i>	number of influence zone in wshd.

Returns

[util::graph Graph](#) based on the adjacency of the influence zones.

Definition at line 179 of file `region_adjacency_graph.hh`.

9.107.2.42 `template<typename V , typename F > fun::i2v::array< V > mln::make::relabelfun (const Function_v2b< F > & fv2b, const V & nlabels, V & new_nlabels) [inline]`

Create a i2v function from a v2b function.

This function can be used to relabel a labeled image.

Parameters

in	<i>fv2b</i>	A v2b function.
in	<i>nlabels</i>	The number of labels.
in	<i>new_nlabels</i>	The number of labels after relabeling.

Returns

a i2v function.

See Also

[mln::labeling::relabel](#)

Definition at line 83 of file relabelfun.hh.

References mln::literal::zero.

Referenced by mln::labeling::pack(), mln::labeling::pack_inplace(), mln::labeling::relabel(), mln::labeled_image_base< I, E >::relabel(), and mln::labeling::relabel_inplace().

9.107.2.43 `template<typename V , typename F > fun::i2v::array< V > mln::make::relabelfun (const Function_v2v< F > & fv2v, const V & nlabels, V & new_nlabels) [inline]`

Create a i2v function from a v2v function.

This function can be used to relabel a labeled image.

Parameters

in	<i>fv2v</i>	A v2v function. This function maps an id to an already existing one.
in	<i>nlabels</i>	The number of labels.
in	<i>new_nlabels</i>	The number of labels after relabeling.

Returns

a i2v function.

See Also

[mln::labeling::relabel](#)

Definition at line 106 of file relabelfun.hh.

References mln::literal::zero.

9.107.2.44 `template<typename T > algebra::vec< 1, T > mln::make::vec (const T & v_0) [inline]`

Create an mln::algebra::vec<n,T>.

Parameters

in	<i>v_0</i>	First coordinate.
----	------------	-------------------

Returns

A 1D vector.

Definition at line 91 of file make/vec.hh.

```
9.107.2.45  template<typename T > algebra::vec< 2, T > mln::make::vec ( const T & v_0, const T & v_1 )  [inline]
```

Create an mln::algebra::vec<2,T>.

Parameters

in	v_0	First coordinate.
in	v_1	Second coordinate.

Returns

A 2D vector.

Definition at line 100 of file make/vec.hh.

```
9.107.2.46  template<typename T > algebra::vec< 3, T > mln::make::vec ( const T & v_0, const T & v_1, const T & v_2 )
           [inline]
```

Create an mln::algebra::vec<3,T>.

Parameters

in	v_0	First coordinate.
in	v_1	Second coordinate.
in	v_2	Third coordinate.

Returns

A 3D vector.

Definition at line 110 of file make/vec.hh.

```
9.107.2.47  template<typename T > algebra::vec< 4, T > mln::make::vec ( const T & v_0, const T & v_1, const T & v_2, const
           T & v_3 )  [inline]
```

Create an mln::algebra::vec<4,T>.

Parameters

in	v_0	First coordinate.
in	v_1	Second coordinate.
in	v_2	Third coordinate.
in	v_3	Fourth coordinate.

Returns

A 4D vector.

Definition at line 121 of file make/vec.hh.

9.107.2.48 `template<typename G , typename FV > mln::vertex_image< void, typename FV::result, G >`
`mln::make::vertex_image (const Graph< G > & g, const Function_v2v< FV > & fv)`

Construct a vertex image.

Parameters

<i>in</i>	<i>g</i>	A graph.
<i>in</i>	<i>fv</i>	A function mapping vertex ids to values.

Returns

A vertex image.

Definition at line 77 of file make/vertex_image.hh.

9.107.2.49 `template<typename FP , typename FV , typename G > mln::vertex_image< typename FP::result, typename FV::result, G > mln::make::vertex_image (const Graph< G > & g_, const Function_v2v< FP > & fp, const Function_v2v< FV > & fv)`

Construct a vertex image.

Parameters

<i>in</i>	<i>g_</i>	A graph.
<i>in</i>	<i>fp</i>	A function mapping vertex ids to sites.
<i>in</i>	<i>fv</i>	A function mapping vertex ids to values.

Returns

A vertex image.

Definition at line 92 of file make/vertex_image.hh.

9.107.2.50 `template<typename I , typename N > p_vertices< util::graph, fun::i2v::array< typename I::site > >`
`mln::make::voronoi (Image< I > & ima_, Image< I > & orig_, const Neighborhood< N > & nbh) [inline]`

Apply the Voronoi algorithm on *ima_* with the original image *orig_* for node computing with neighborhood *nbh*.

Parameters

<i>in</i>	<i>ima_</i>	The labeling image.
<i>in</i>	<i>orig_</i>	The original image.
<i>in</i>	<i>nbh</i>	The neighborhood for computing algorithm.

Returns

The computed graph.

Definition at line 68 of file voronoi.hh.

References `mln::util::graph::add_edge()`, `mln::util::graph::add_vertex()`, and `mln::estim::min_max()`.

9.107.2.51 `template<typename W , typename F > mln::w_window< typename W::dpsite, typename F::result >`
`mln::make::w_window (const Window< W > & win, const Function_v2v< F > & wei) [inline]`

Create a `mln::w_window` from a window and a weight function.

Parameters

in	<i>win</i>	A simple window.
in	<i>wei</i>	A weight function.

Returns

A weighted window.

Definition at line 63 of file `make/w_window.hh`.

References `mln::w_window< D, W >::insert()`, and `mln::literal::origin`.

9.107.2.52 `template<typename W , unsigned M> mln::w_window< mln::dpoint1d, W > mln::make::w_window1d (W(& weights[M])) [inline]`

Create a 1D `mln::w_window` from an array of weights.

Parameters

in	<i>weights</i>	Array.
----	----------------	--------

Precondition

The array size, *M*, has to be a square of an odd integer.

Returns

A 1D weighted window.

Definition at line 63 of file `w_window1d.hh`.

References `mln::w_window< D, W >::insert()`.

9.107.2.53 `template<typename W , unsigned S> mln::w_window< mln::dpoint2d, W > mln::make::w_window2d (W(& weights[S])) [inline]`

Create a 2D `mln::w_window` from an array of weights.

Parameters

in	<i>weights</i>	Array.
----	----------------	--------

Precondition

The array size, *S*, has to be a square of an odd integer.

Returns

A 2D weighted window.

Definition at line 62 of file `w_window2d.hh`.

Referenced by `mln::linear::mln_ch_convolve()`.

9.107.2.54 `template<typename W , unsigned M> mln::w_window< mln::dpoint3d, W > mln::make::w_window3d (W(& weights[M])) [inline]`

Create a 3D `mln::w_window` from an array of weights.

Parameters

in	<i>weights</i>	Array.
----	----------------	--------

Precondition

The array size, M , has to be a cube of an odd integer.

Returns

A 3D weighted window.

Definition at line 64 of file w_window3d.hh.

References `mln::w_window< D, W >::insert()`.

```
9.107.2.55  template<typename D , typename W , unsigned L> mln::w_window< D, W > mln::make::w_window_directional
            ( const Gdpoint< D > & dp, W(&) weights[L] )  [inline]
```

Create a directional centered weighted window.

Parameters

in	<i>dp</i>	A delta-point to set the orientation.
in	<i>weights</i>	An array of weights.

Returns

A weighted window.

The window length L has to be odd.

Definition at line 61 of file w_window_directional.hh.

References `mln::w_window< D, W >::insert()`, and `mln::literal::zero`.

9.108 mln::math Namespace Reference

Namespace of mathematical routines.

Functions

- `template<typename T >`
`T abs (const T &v)`
Generic version.
- `template<unsigned n>`
`value::int_u< n > abs (const value::int_u< n > &v)`
Specialization for [mln::value::int_u](#).
- `int abs (int v)`
Specializations for existing overloads of `std::abs`.

9.108.1 Detailed Description

Namespace of mathematical routines.

9.108.2 Function Documentation

9.108.2.1 `template<typename T> T mln::math::abs (const T & v) [inline]`

Generic version.

Definition at line 74 of file `math/abs.hh`.

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

9.108.2.2 `int mln::math::abs (int v) [inline]`

Specializations for existing overloads of `std::abs`.

Reference: ISO/IEC 14882:2003 C++ standard, section 26.5 (C Library, `[lib.c.math]`).

Definition at line 79 of file `math/abs.hh`.

9.108.2.3 `template<unsigned n> value::int_u<n> mln::math::abs (const value::int_u<n> & v) [inline]`

Specialization for `mln::value::int_u`.

Definition at line 88 of file `math/abs.hh`.

9.109 mln::metal Namespace Reference

Namespace of meta-programming tools.

Namespaces

- namespace [impl](#)
Implementation namespace of metal namespace.
- namespace [math](#)
Namespace of static mathematical functions.

Classes

- struct [ands](#)
Ands type.
- struct [converts_to](#)
"converts-to" check.
- struct [equal](#)
Definition of a static 'equal' test.
- struct [goes_to](#)
"goes-to" check.
- struct [is](#)
"is" check.
- struct [is_a](#)
"is_a" check.
- struct [is_not](#)
"is_not" check.
- struct [is_not_a](#)
"is_not_a" static Boolean expression.

9.109.1 Detailed Description

Namespace of meta-programming tools.

9.110 mIn::metal::impl Namespace Reference

Implementation namespace of metal namespace.

9.110.1 Detailed Description

Implementation namespace of metal namespace.

9.111 mIn::metal::math Namespace Reference

Namespace of static mathematical functions.

Namespaces

- namespace [impl](#)
Implementation namespace of [metal::math](#) namespace.

9.111.1 Detailed Description

Namespace of static mathematical functions.

9.112 mIn::metal::math::impl Namespace Reference

Implementation namespace of [metal::math](#) namespace.

9.112.1 Detailed Description

Implementation namespace of [metal::math](#) namespace.

9.113 mIn::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 I , typename W >`
`mln::trait::concrete< I >::ret erosion_tolerant (const Image< I > &input, const Window< W > &win, unsigned rank)`
Morphological tolerant erosion.
- `template<typename Op , typename I , typename W >`
`mln::trait::concrete< I >::ret general (const Op &op, const Image< I > &input, const Window< W > &win)`
Morphological general routine.
- `template<typename I , typename W >`
`mln::trait::concrete< I >::ret gradient (const Image< I > &input, const Window< W > &win)`
Morphological gradient.
- `template<typename I , typename W >`
`mln::trait::concrete< I >::ret gradient_external (const Image< I > &input, const Window< W > &win)`
Morphological external gradient.
- `template<typename I , typename W >`
`mln::trait::concrete< I >::ret gradient_internal (const Image< I > &input, const Window< W > &win)`
Morphological internal gradient.
- `template<typename I , typename Wh , typename Wm >`
`mln::trait::concrete< I >::ret hit_or_miss (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`
Morphological hit-or-miss.
- `template<typename I , typename Wh , typename Wm >`
`mln::trait::concrete< I >::ret hit_or_miss_background_closing (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`
Morphological hit-or-miss closing of the background.
- `template<typename I , typename Wh , typename Wm >`
`mln::trait::concrete< I >::ret hit_or_miss_background_opening (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`
Morphological hit-or-miss opening of the background.
- `template<typename I , typename Wh , typename Wm >`
`mln::trait::concrete< I >::ret hit_or_miss_closing (const Image< I > &input, const Window< Wh > &win_hit, const Window< Wm > &win_miss)`

Morphological hit-or-miss closing.

- template<typename I , typename Wh , typename Wm >
mln::trait::concrete< I >::ret [hit_or_miss_opening](#) (const [Image](#)< I > &input, const [Window](#)< Wh > &win_
hit, const [Window](#)< Wm > &win_miss)

Morphological hit-or-miss opening.

- template<typename I , typename W , typename O >
void [laplacian](#) (const [Image](#)< I > &input, const [Window](#)< W > &win, [Image](#)< O > &output)
- template<typename V >
[edge_image](#)< [util::site_pair](#)
< [point2d](#) >, V, [util::graph](#) > [line_gradient](#) (const [mln::image2d](#)< V > &ima)

Create a line graph image representing the gradient norm of a [mln::image2d](#).

- template<typename L , typename I , typename N >
mln::trait::ch_value< I, L >::ret [meyer_wst](#) (const [Image](#)< I > &input, const [Neighborhood](#)< N > &nbh, L
&nbasins)

Meyer's Watershed Transform (WST) algorithm.

- template<typename L , typename I , typename N >
mln::trait::ch_value< I, L >::ret [meyer_wst](#) (const [Image](#)< I > &input, const [Neighborhood](#)< N > &nbh)

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

- template<typename I , typename J >
mln::trait::concrete< I >::ret [min](#) (const [Image](#)< I > &lhs, const [Image](#)< J > &rhs)
- template<typename I , typename J >
void [min_inplace](#) ([Image](#)< I > &lhs, const [Image](#)< J > &rhs)
- template<typename I , typename J >
mln::trait::concrete< I >::ret [minus](#) (const [Image](#)< I > &lhs, const [Image](#)< J > &rhs)
- template<typename I , typename J >
mln::trait::concrete< I >::ret [plus](#) (const [Image](#)< I > &lhs, const [Image](#)< J > &rhs)
- template<typename I , typename W >
mln::trait::concrete< I >::ret [rank_filter](#) (const [Image](#)< I > &input, const [Window](#)< W > &win, unsigned k)

Morphological rank_filter.

- template<typename I , typename Wfg , typename Wbg >
mln::trait::concrete< I >::ret [thick_miss](#) (const [Image](#)< I > &input, const [Window](#)< Wfg > &win_fg, const
[Window](#)< Wbg > &win_bg)
- template<typename I , typename Wfg , typename Wbg >
mln::trait::concrete< I >::ret [thickening](#) (const [Image](#)< I > &input, const [Window](#)< Wfg > &win_fg, const
[Window](#)< Wbg > &win_bg)
- template<typename I , typename Wfg , typename Wbg >
mln::trait::concrete< I >::ret [thin_fit](#) (const [Image](#)< I > &input, const [Window](#)< Wfg > &win_fg, const [Win-](#)
[dow](#)< 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).

9.113.1 Detailed Description

Namespace of mathematical morphology routines.

9.113.2 Function Documentation

9.113.2.1 `template<typename I > mln::trait::concrete< I >::ret mln::morpho::complementation (const Image< I > & input)`
[inline]

Morphological complementation: either a logical "not" (if morpho on sets) or an arithmetical complementation (if morpho on functions).

Definition at line 116 of file complementation.hh.

Referenced by `hit_or_miss_background_closing()`, `hit_or_miss_background_opening()`, `hit_or_miss_closing()`, and `thinning()`.

9.113.2.2 `template<typename I > void mln::morpho::complementation.inplace (Image< I > & input)` [inline]

Morphological complementation, inplace version: either a logical "not" (if morpho on sets) or an arithmetical complementation (if morpho on functions).

Definition at line 130 of file complementation.hh.

9.113.2.3 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::contrast (const Image< I > & input, const Window< W > & win)` [inline]

Morphological contrast operator (based on top-hats).

This operator is $I_d + w_{th_B} - b_{th_B}$.

Definition at line 57 of file contrast.hh.

References `mln::arith::plus()`, `top_hat_black()`, and `top_hat_white()`.

9.113.2.4 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::dilation (const Image< I > & input, const Window< W > & win)` [inline]

Morphological dilation.

Definition at line 162 of file dilation.hh.

References `general()`.

Referenced by `gradient()`, `gradient_external()`, `hit_or_miss_background_opening()`, `hit_or_miss_opening()`, `laplacian()`, `mln::morpho::opening::approx::structural()`, and `mln::morpho::closing::approx::structural()`.

9.113.2.5 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::erosion (const Image< I > & input, const Window< W > & win)` [inline]

Morphological erosion.

Definition at line 163 of file erosion.hh.

References `general()`.

Referenced by `gradient()`, `gradient_internal()`, `laplacian()`, `mln::morpho::opening::approx::structural()`, and `mln::morpho::closing::approx::structural()`.

9.113.2.6 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::erosion_tolerant (const Image< I > & input, const Window< W > & win, unsigned rank) [inline]`

Morphological tolerant erosion.

Definition at line 200 of file erosion_tolerant.hh.

9.113.2.7 `template<typename Op , typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::general (const Op & op, const Image< I > & input, const Window< W > & win) [inline]`

Morphological general routine.

Definition at line 168 of file general.hh.

Referenced by dilation(), and erosion().

9.113.2.8 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::gradient (const Image< I > & input, const Window< W > & win) [inline]`

Morphological gradient.

This operator is $d_B - e_B$.

Definition at line 76 of file gradient.hh.

References dilation(), erosion(), minus(), and mln::test::positive().

9.113.2.9 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::gradient_external (const Image< I > & input, const Window< W > & win) [inline]`

Morphological external gradient.

This operator is $d_B - Id$.

Definition at line 110 of file gradient.hh.

References dilation(), minus(), and mln::test::positive().

9.113.2.10 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::gradient_internal (const Image< I > & input, const Window< W > & win) [inline]`

Morphological internal gradient.

This operator is $Id - e_B$.

Definition at line 93 of file gradient.hh.

References erosion(), minus(), and mln::test::positive().

9.113.2.11 `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 $HMT_(B_h, B_m) = e_{B_h} \wedge (e_{B_m} \circ C)$.

Definition at line 281 of file hit_or_miss.hh.

Referenced by thickening(), and thinning().

9.113.2.12 `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 $C \circ \text{HMTopeBG} \circ C$.

Definition at line 362 of file `hit_or_miss.hh`.

References `complementation()`, `hit_or_miss_background_opening()`, and `hit_or_miss_closing()`.

9.113.2.13 `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 $\text{HMTopeBG} = \text{HMTope_}(B_m, B_h) \circ C = d_(-B_m) \circ \text{HMT_}(B_h, B_m)$.

Definition at line 319 of file `hit_or_miss.hh`.

References `complementation()`, `dilation()`, `hit_or_miss_opening()`, and `mln::win::sym()`.

Referenced by `hit_or_miss_background_closing()`, and `thick_miss()`.

9.113.2.14 `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 $C \circ \text{HMTope} \circ C$.

Definition at line 342 of file `hit_or_miss.hh`.

References `complementation()`, and `hit_or_miss_opening()`.

Referenced by `hit_or_miss_background_closing()`.

9.113.2.15 `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 $\text{HMTope_}(B_h, B_m) = d_(-B_h) \circ \text{HMT_}(B_h, B_m)$.

Definition at line 299 of file `hit_or_miss.hh`.

References `dilation()`, and `mln::win::sym()`.

Referenced by `hit_or_miss_background_opening()`, `hit_or_miss_closing()`, and `thin_fit()`.

9.113.2.16 `template<typename I , typename W , typename O > void mln::morpho::laplacian (const Image< I > & input, const
Window< W > & win, Image< O > & output) [inline]`

Morphological laplacian.

This operator is $(d_B - Id) - (Id - e_B)$.

Definition at line 63 of file `laplacian.hh`.

References `dilation()`, `erosion()`, `mln::data::fill()`, and `minus()`.

9.113.2.17 `template<typename V > edge_image< util::site_pair< point2d >, V, util::graph >
mln::morpho::line_gradient (const mln::image2d< V > & ima)`

Create a line graph image representing the gradient norm of a [mln::image2d](#).

Definition at line 70 of file `line_gradient.hh`.

References `mln::math::abs()`, `mln::image2d< T >::domain()`, and `mln::window< D >::insert()`.

9.113.2.18 `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret mln::morpho::meyer_wst (
const Image< I > & input, const Neighborhood< N > & nbh, L & nbasins)`

Meyer's Watershed Transform (WST) algorithm.

Parameters

<code>in</code>	<code>input</code>	The input image.
<code>in</code>	<code>nbh</code>	The connexity of markers.
<code>out</code>	<code>nbasins</code>	The number of basins.

- `L` is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- `I` is the exact type of the input image.
- `N` is the exact type of the neighborhood used to express *input's* connexity.

Definition at line 108 of file `meyer_wst.hh`.

References `mln::data::fill()`, `mln::p_priority< P, Q >::front()`, `mln::initialize()`, `mln::p_priority< P, Q >::pop()`, `mln::p_priority< P, Q >::push()`, `mln::labeling::regional_minima()`, and `mln::literal::zero`.

9.113.2.19 `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret mln::morpho::meyer_wst (
const Image< I > & input, const Neighborhood< N > & nbh)`

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

Parameters

<code>in</code>	<code>input</code>	The input image.
<code>in</code>	<code>nbh</code>	The connexity of markers.

- `L` is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- `I` is the exact type of the input image.
- `N` is the exact type of the neighborhood used to express *input's* connexity.

Note that the first parameter, `L`, is not automatically valued from the type of the actual argument during implicit instantiation: you have to explicitly pass this parameter at call sites.

Definition at line 202 of file `meyer_wst.hh`.

9.113.2.20 `template<typename I , typename J > mln::trait::concrete< I >::ret mln::morpho::min (const Image< I > & lhs,
const Image< J > & rhs) [inline]`

Morphological min: either a logical "and" (if morpho on sets) or an arithmetical min (if morpho on functions).

Definition at line 112 of file `morpho/min.hh`.

9.113.2.21 `template<typename I , typename J > void mln::morpho::min_inplace (Image< I > & lhs, const Image< J > & rhs) [inline]`

Morphological min, inplace version: either a logical "and" (if morpho on sets) or an arithmetical min (if morpho on functions).

Definition at line 125 of file morpho/min.hh.

9.113.2.22 `template<typename I , typename J > mln::trait::concrete< I >::ret mln::morpho::minus (const Image< I > & lhs, const Image< J > & rhs) [inline]`

Morphological minus: either a logical "and not" (if morpho on sets) or an arithmetical minus (if morpho on functions).

Definition at line 87 of file morpho/minus.hh.

Referenced by `gradient()`, `gradient_external()`, `gradient_internal()`, `laplacian()`, `thin_fit()`, `thinning()`, `top_hat_black()`, `mln::morpho::elementary::top_hat_black()`, `top_hat_self_complementary()`, `mln::morpho::elementary::top_hat_self_complementary()`, `top_hat_white()`, and `mln::morpho::elementary::top_hat_white()`.

9.113.2.23 `template<typename I , typename J > mln::trait::concrete< I >::ret mln::morpho::plus (const Image< I > & lhs, const Image< J > & rhs) [inline]`

Morphological plus: either a "logical or" (if morpho on sets) or an "arithmetical plus" (if morpho on functions).

Definition at line 86 of file morpho/plus.hh.

Referenced by `thick_miss()`, and `thickening()`.

9.113.2.24 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::rank_filter (const Image< I > & input, const Window< W > & win, unsigned k) [inline]`

Morphological `rank_filter`.

Definition at line 213 of file rank_filter.hh.

9.113.2.25 `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I >::ret mln::morpho::thick_miss (const Image< I > & input, const Window< Wfg > & win_fg, const Window< Wbg > & win_bg) [inline]`

Morphological thick-miss.

This operator is `THICK_B = Id + HMTopeBG_B`, where `B = (Bfg, Bbg)`.

Definition at line 59 of file thick_miss.hh.

References `hit_or_miss_background_opening()`, and `plus()`.

9.113.2.26 `template<typename I , typename Wfg , typename Wbg > mln::trait::concrete< I >::ret mln::morpho::thickening (const Image< I > & input, const Window< Wfg > & win_fg, const Window< Wbg > & win_bg) [inline]`

Morphological thickening.

This operator is `THICK_B = Id + HMT_B`, where `B = (Bfg, Bbg)`.

Definition at line 88 of file thickening.hh.

References `hit_or_miss()`, and `plus()`.

Referenced by `thinning()`.

9.113.2.27 `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 $THIN_B = Id - HMT_{Tope_B}$ where $B = (Bfg, Bbg)$.

Definition at line 87 of file thin_fit.hh.

References hit_or_miss_opening(), and minus().

9.113.2.28 `template<typename I, typename Wfg, typename Wbg > mln::trait::concrete<I>::ret mln::morpho::thinning (const Image<I> & input, const Window<Wfg> & win_fg, const Window<Wbg> & win_bg) [inline]`

Morphological thinning.

This operator is $THIN_B = Id - HMT_B$, where $B = (Bfg, Bbg)$.

Definition at line 90 of file thinning.hh.

References complementation(), hit_or_miss(), minus(), and thickening().

9.113.2.29 `template<typename I, typename W > mln::trait::concrete<I>::ret mln::morpho::top_hat_black (const Image<I> & input, const Window<W> & win) [inline]`

Morphological black top-hat (for background / dark objects).

This operator is $clo_B - Id$.

Definition at line 102 of file top_hat.hh.

References minus(), and mln::test::positive().

Referenced by contrast().

9.113.2.30 `template<typename I, typename W > mln::trait::concrete<I>::ret mln::morpho::top_hat_self_complementary (const Image<I> & input, const Window<W> & win) [inline]`

Morphological self-complementary top-hat.

This operator is

$= top_hat_white + top_hat_black$

$= (input - opening) + (closing - input)$

$= closing - opening$.

Definition at line 121 of file top_hat.hh.

References minus(), and mln::test::positive().

9.113.2.31 `template<typename I, typename W > mln::trait::concrete<I>::ret mln::morpho::top_hat_white (const Image<I> & input, const Window<W> & win) [inline]`

Morphological white top-hat (for object / light objects).

This operator is $Id - ope_B$.

Definition at line 83 of file top_hat.hh.

References minus(), and mln::test::positive().

Referenced by contrast().

9.114 mln::morpho::approx Namespace Reference

Namespace of approximate mathematical morphology routines.

9.114.1 Detailed Description

Namespace of approximate mathematical morphology routines.

9.115 mln::morpho::attribute Namespace Reference

Namespace of attributes used in mathematical morphology.

Classes

- class [card](#)
Cardinality accumulator class.
- struct [count_adjacent_vertices](#)
Count_Adjacent_Vertices accumulator class.
- struct [height](#)
Height accumulator class.
- struct [sharpness](#)
Sharpness accumulator class.
- class [sum](#)
Suminality accumulator class.
- struct [volume](#)
Volume accumulator class.

9.115.1 Detailed Description

Namespace of attributes used in mathematical morphology.

9.116 mln::morpho::closing::approx Namespace Reference

Namespace of approximate mathematical morphology closing routines.

Functions

- template<typename I , typename W >
mln::trait::concrete< I >::ret [structural](#) (const [Image](#)< I > &input, const [Window](#)< W > &win)
Approximate of morphological structural closing.

9.116.1 Detailed Description

Namespace of approximate mathematical morphology closing routines.

9.116.2 Function Documentation

9.116.2.1 `template<typename I, typename W> mln::trait::concrete< I>::ret mln::morpho::closing::approx::structural (const Image< I> & input, const Window< W> & win) [inline]`

Approximate of morphological structural closing.

This operator is $e_{\{-B\}} \circ d_B$.

Definition at line 65 of file closing/approx/structural.hh.

References `mln::morpho::dilation()`, `mln::morpho::erosion()`, and `mln::win::sym()`.

9.117 mln::morpho::elementary Namespace Reference

Namespace of image processing routines of elementary mathematical morphology.

Functions

- `template<typename I, typename N> mln::trait::concrete< I>::ret closing (const Image< I> &input, const Neighborhood< N> &nbh)`
Morphological elementary closing.
- `template<typename I, typename N> mln_trait_op_minus_twice (typename mln::trait::concrete< I>::ret) laplacian(const Image< I> &input)`
Morphological elementary laplacian.
- `template<typename I, typename N> mln::trait::concrete< I>::ret opening (const Image< I> &input, const Neighborhood< N> &nbh)`
Morphological elementary opening.
- `template<typename I, typename N> mln::trait::concrete< I>::ret top_hat_black (const Image< I> &input, const Neighborhood< N> &nbh)`
Morphological elementary black top-hat (for background / dark objects).
- `template<typename I, typename N> mln::trait::concrete< I>::ret top_hat_self_complementary (const Image< I> &input, const Neighborhood< N> &nbh)`
Morphological elementary self-complementary top-hat.
- `template<typename I, typename N> mln::trait::concrete< I>::ret top_hat_white (const Image< I> &input, const Neighborhood< N> &nbh)`
Morphological elementary white top-hat (for object / light objects).

9.117.1 Detailed Description

Namespace of image processing routines of elementary mathematical morphology.

9.117.2 Function Documentation

9.117.2.1 `template<typename I, typename N> mln::trait::concrete< I>::ret mln::morpho::elementary::closing (const Image< I> & input, const Neighborhood< N> & nbh) [inline]`

Morphological elementary closing.

This operator is $e \circ d$.

Definition at line 58 of file closing.hh.

Referenced by `top_hat_black()`, and `top_hat_self_complementary()`.

9.117.2.2 `template<typename I , typename N > mln::morpho::elementary::mln_trait_op_minus_twice (typename
mln::trait::concrete< I >::ret) const [inline]`

Morphological elementary laplacian.

This operator is $(d - id) - (id - e)$.

9.117.2.3 `template<typename I , typename N > mln::trait::concrete< I >::ret mln::morpho::elementary::opening (const
Image< I > & input, const Neighborhood< N > & nbh) [inline]`

Morphological elementary opening.

This operator is $d \circ e$.

Definition at line 58 of file opening.hh.

Referenced by `top_hat_self_complementary()`, and `top_hat_white()`.

9.117.2.4 `template<typename I , typename N > mln::trait::concrete< I >::ret mln::morpho::elementary::top_hat_black (const
Image< I > & input, const Neighborhood< N > & nbh) [inline]`

Morphological elementary black top-hat (for background / dark objects).

This operator is $clo - Id$.

Definition at line 105 of file elementary/top_hat.hh.

References `closing()`, `mln::morpho::minus()`, and `mln::test::positive()`.

9.117.2.5 `template<typename I , typename N > mln::trait::concrete< I >::ret mln::morpho::elementary-
::top_hat_self_complementary (const Image< I > & input, const Neighborhood< N > & nbh)
[inline]`

Morphological elementary self-complementary top-hat.

This operator is

$= \text{top_hat_white} + \text{top_hat_black}$

$= (Id - \text{opening}) + (\text{closing} - Id)$

$= \text{closing} - \text{opening}$.

Definition at line 125 of file elementary/top_hat.hh.

References `closing()`, `mln::morpho::minus()`, `opening()`, and `mln::test::positive()`.

9.117.2.6 `template<typename I , typename N > mln::trait::concrete< I >::ret mln::morpho::elementary::top_hat_white (const
Image< I > & input, const Neighborhood< N > & nbh) [inline]`

Morphological elementary white top-hat (for object / light objects).

This operator is $Id - ope$.

Definition at line 85 of file elementary/top_hat.hh.

References `mln::morpho::minus()`, `opening()`, and `mln::test::positive()`.

9.118 mln::morpho::impl Namespace Reference

Namespace of mathematical morphology routines implementations.

Namespaces

- namespace [generic](#)

Namespace of mathematical morphology routines generic implementations.

9.118.1 Detailed Description

Namespace of mathematical morphology routines implementations.

9.119 mln::morpho::impl::generic Namespace Reference

Namespace of mathematical morphology routines generic implementations.

9.119.1 Detailed Description

Namespace of mathematical morphology routines generic implementations.

9.120 mln::morpho::opening::approx Namespace Reference

Namespace of approximate mathematical morphology opening routines.

Functions

- `template<typename I , typename W > mln::trait::concrete< I >::ret structural (const Image< I > &input, const Window< W > &win)`
Approximate of morphological structural opening.

9.120.1 Detailed Description

Namespace of approximate mathematical morphology opening routines.

9.120.2 Function Documentation

9.120.2.1 `template<typename I , typename W > mln::trait::concrete< I >::ret mln::morpho::opening::approx::structural (const Image< I > &input, const Window< W > &win) [inline]`

Approximate of morphological structural opening.

This operator is $d_{\{-B\}} \circ e_B$.

Definition at line 64 of file opening/approx/structural.hh.

References `mln::morpho::dilation()`, `mln::morpho::erosion()`, and `mln::win::sym()`.

9.121 mln::morpho::reconstruction Namespace Reference

Namespace of morphological reconstruction routines.

Namespaces

- namespace [by_dilation](#)
Namespace of morphological reconstruction by dilation routines.
- namespace [by_erosion](#)
Namespace of morphological reconstruction by erosion routines.

9.121.1 Detailed Description

Namespace of morphological reconstruction routines.

9.122 mln::morpho::reconstruction::by_dilation Namespace Reference

Namespace of morphological reconstruction by dilation routines.

9.122.1 Detailed Description

Namespace of morphological reconstruction by dilation routines.

9.123 mln::morpho::reconstruction::by_erosion Namespace Reference

Namespace of morphological reconstruction by erosion routines.

9.123.1 Detailed Description

Namespace of morphological reconstruction by erosion routines.

9.124 mln::morpho::tree Namespace Reference

Namespace of morphological tree-related routines.

Namespaces

- namespace [filter](#)
Namespace for attribute filtering.

Functions

- `template<typename A , typename T >
mln::trait::ch_value< typename
T::function, typename
A::result >::ret compute_attribute_image (const Accumulator< A > &a, const T &t, mln::trait::ch_value<
typename T::function, A >::ret *accu_image=0)`
Compute an attribute image using tree with a parent relationship between sites.

- template<typename A , typename T , typename V >
mln::trait::ch_value< typename
T::function, typename
A::result >::ret [compute_attribute_image_from](#) (const [Accumulator](#)< A > &a, const T &t, const [Image](#)< V >
&values, mln::trait::ch_value< typename T::function, A >::ret *accu_image=0)

The same as [compute_attribute_image](#) but uses the values stored by [values](#) image instead.
 - template<typename I , typename N , typename S >
mln::trait::ch_value< I,
typename I::psite >::ret [compute_parent](#) (const [Image](#)< I > &f, const [Neighborhood](#)< N > &nbh, const
[Site_Set](#)< S > &s)

Compute a tree with a parent relationship between sites.
 - template<typename I , typename N >
[data](#)< I, [p_array](#)< typename
I::psite > > [dual_input_max_tree](#) (const [Image](#)< I > &f, const [Image](#)< I > &m, const [Neighborhood](#)< N >
&nbh)

Compute the dual input max tree using mask-based connectivity.
 - template<typename I , typename N >
[data](#)< I, [p_array](#)< typename
I::psite > > [max_tree](#) (const [Image](#)< I > &f, const [Neighborhood](#)< N > &nbh)

Compute a canonized max-tree.
 - template<typename I , typename N >
[data](#)< I, [p_array](#)< typename
I::psite > > [min_tree](#) (const [Image](#)< I > &f, const [Neighborhood](#)< N > &nbh)

Compute a canonized min-tree.
 - template<typename T , typename A , typename P , typename W >
void [propagate_if](#) (const T &tree, [Image](#)< A > &a_, const way_of_propagation< W > &prop_, const
[Function_v2b](#)< P > &pred_, const typename A::value &v)
 - template<typename T , typename A , typename P >
void [propagate_if](#) (const T &tree, [Image](#)< A > &a_, const desc_propagation &prop_, const [Function_v2b](#)<
P > &pred_)
 - template<typename T , typename A , typename W >
void [propagate_if_value](#) (const T &tree, [Image](#)< A > &a_, const way_of_propagation< W > &prop, const
typename A::value &v)
 - template<typename T , typename A >
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_ancestors](#) (typename A::psite n, const T &t, [Image](#)< A > &a_)
 - template<typename T , typename A >
void [propagate_node_to_descendants](#) (typename A::psite n, const T &t, [Image](#)< A > &a_, const typename
A::value &v, unsigned *nb_leaves=0)
 - template<typename T , typename A >
void [propagate_node_to_descendants](#) (typename A::psite &n, const T &t, [Image](#)< A > &a_, 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.*

9.124.1 Detailed Description

Namespace of morphological tree-related routines.

9.124.2 Function Documentation

9.124.2.1 `template<typename A , typename T > mln::trait::ch_value< typename T::function, typename A::result >::ret
mln::morpho::tree::compute_attribute_image (const Accumulator< A > & a, const T & t, mln::trait::ch_value<
typename T::function, A >::ret * accu_image = 0) [inline]`

Compute an attribute image using tree with a parent relationship between sites.

In the attribute image, the resulting value at a node is the 'sum' of its sub-components value + the attribute value at this node.

Warning: *s* translates the ordering related to the "natural" childhood relationship. The parenthood is thus inverted w.r.t. to *s*.

It is very convenient since all processing upon the parent tree are performed following *s* (in the default "forward" way).

FIXME: Put it more clearly...

The parent result image verifies:

- *p* is root iff `parent(p) == p`
- *p* is a node iff either *p* is root or `f(parent(p)) != f(p)`.

Parameters

in	<i>a</i>	Attribute.
in	<i>t</i>	Component tree.
out	<i>accu_image</i>	Optional argument used to store image of attribute accumulator.

Returns

The attribute image.

Definition at line 217 of file `compute_attribute_image.hh`.

9.124.2.2 `template<typename A , typename T , typename V > mln::trait::ch_value< typename T::function, typename A::result
>::ret mln::morpho::tree::compute_attribute_image_from (const Accumulator< A > & a, const T & t, const Image<
V > & values, mln::trait::ch_value< typename T::function, A >::ret * accu_image = 0) [inline]`

The same as `compute_attribute_image` but uses the values stored by `values` image instead.

Parameters

in	<i>a</i>	Attribute.
in	<i>t</i>	Component tree.
in	<i>values</i>	Value image.
out	<i>accu_image</i>	Optional argument used to store image.

Returns

Definition at line 234 of file compute_attribute_image.hh.

```
9.124.2.3  template<typename I , typename N , typename S > mln::trait::ch_value< I, typename I::psite >::ret
           mln::morpho::tree::compute_parent ( const Image< I > & f, const Neighborhood< N > & nbh, const Site_Set< S
           > & s ) [inline]
```

Compute a tree with a parent relationship between sites.

Warning: *s* translates the ordering related to the "natural" childhood relationship. The parenthood is thus inverted w.r.t. to *s*.

It is very convenient since most processing routines upon the parent tree are performed following *s* (in the default "forward" way). Indeed that is the way to propagate information from parents to children.

The parent result image verifies:

- *p* is root iff parent(*p*) == *p*
- *p* is a node iff either *p* is root or *f*(parent(*p*)) != *f*(*p*).

The choice "s means childhood" is consistent with labeling in binary images. In that particular case, while browsing the image in forward scan (video), we expect to find first a tree root (a first point, representative of a component) and then the other component points. Please note that it leads to increasing values of labels in the "natural" video scan.

Since mathematical morphology on functions is related to morphology on sets, we clearly want to keep the equivalence between "component labeling" and "component filtering" using trees.

FIXME: Put it more clearly... Insert pictures!

A binary image:

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

where '|' means true and '-' means false.

Its labeling:

```
0 1 1 0 0
0 1 1 0 2
0 0 0 0 0
0 0 3 3 0
```

The corresponding forest:

```
x o . x x
x . . x o
x x x x x
x x o . x
```

where 'x' means "no data", 'o' is a tree root (representative point for a component), and '.' is a tree regular (non-root) point (in a component by not its representative point).

The forest, with the parent relationship looks like:

```
o < .
^ r
.. o
o < .
```

Definition at line 286 of file compute_parent.hh.

```
9.124.2.4  template<typename I , typename N > morpho::tree::data< I, p_array< typename I::psite > >
            mln::morpho::tree::dual_input_max_tree ( const Image< I > & f, const Image< I > & m, const Neighborhood< N >
            & nbh )  [inline]
```

Compute the dual input max tree using mask-based connectivity.

Parameters

in	<i>f</i>	The original image.
in	<i>m</i>	The connectivity mask.
in	<i>nbh</i>	The neighborhood of the mask.

Returns

The computed tree.

Definition at line 109 of file dual_input_tree.hh.

```
9.124.2.5  template<typename I , typename N > data< I, p_array< typename I::psite > > mln::morpho::tree::max_tree (
            const Image< I > & f, const Neighborhood< N > & nbh )  [inline]
```

Compute a canonized max-tree.

Parameters

in	<i>f</i>	The input image.
in	<i>nbh</i>	The neighborhood.

Returns

The corresponding max-tree structure.

Definition at line 100 of file component_tree.hh.

References mln::data::sort_psites_increasing().

```
9.124.2.6  template<typename I , typename N > data< I, p_array< typename I::psite > > mln::morpho::tree::min_tree (
            const Image< I > & f, const Neighborhood< N > & nbh )  [inline]
```

Compute a canonized min-tree.

Parameters

in	<i>f</i>	The input image.
in	<i>nbh</i>	The neighborhood.

Returns

The corresponding min-tree structure.

Definition at line 77 of file component_tree.hh.

References mln::data::sort_psites_decreasing().

9.124.2.7 `template<typename T , typename A , typename P , typename W > void mln::morpho::tree::propagate_if (const T & tree, Image< A > & a_, const way_of_propagation< W > & prop_, const Function_v2b< P > & pred_, const typename A::value & v) [inline]`

Propagate nodes checking the predicate `pred` in the way defined by `way_of_propagation`.

Parameters

<i>tree</i>	Component tree used for propagation.
<i>a_</i>	Attributed image where values are propagated.
<i>prop_</i>	Propagate node in ascendant or descendant way.
<i>pred_</i>	Predicate that node must check to be propagated.
<i>v</i>	Value to be propagated. (By default <code>v</code> is the value at the node being propagated).

Definition at line 293 of file propagate_if.hh.

Referenced by mln::morpho::tree::filter::subtractive().

9.124.2.8 `template<typename T , typename A , typename P > void mln::morpho::tree::propagate_if (const T & tree, Image< A > & a_, const desc_propagation & prop_, const Function_v2b< P > & pred_) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 310 of file propagate_if.hh.

9.124.2.9 `template<typename T , typename A , typename W > void mln::morpho::tree::propagate_if_value (const T & tree, Image< A > & a_, const way_of_propagation< W > & prop, const typename A::value & v) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 278 of file propagate_if.hh.

9.124.2.10 `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)`

Propagate a value `v` from a node `n` to its ancestors.

Parameters

in	<i>n</i>	Node to propagate.
in	<i>t</i>	Component tree used for propagation.
in	<i>a_</i>	Attribute image where values are propagated.
in	<i>v</i>	Value to propagate.

Definition at line 170 of file propagate_node.hh.

Referenced by propagate_node_to_ancestors().

9.124.2.11 `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_ancestors (typename A::psite n, const T & t, Image< A > & a_) [inline]`

Propagate the node's value to its ancestors.

Parameters

in	<i>n</i>	Node to propagate.
in	<i>t</i>	Component tree used for propagation.
in, out	<i>a_</i>	Attribute image where values are propagated.

Definition at line 197 of file `propagate_node.hh`.

References `propagate_node_to_ancestors()`.

9.124.2.12 `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

in	<i>n</i>	Node to propagate.
in	<i>t</i>	Component tree used for propagation.
in	<i>a_</i>	Attribute image where values are propagated.
in	<i>v</i>	Value to propagate.
out	<i>nb_leaves</i>	Optional. Store the number of leaves in the component.

Definition at line 120 of file `propagate_node.hh`.

9.124.2.13 `template<typename T , typename A > void mln::morpho::tree::propagate_node_to_descendants (typename A::psite & n, const T & t, Image< A > & a_, unsigned * nb_leaves = 0) [inline]`

Propagate the node's value to its descendants.

Parameters

in	<i>n</i>	Node to propagate.
in	<i>t</i>	Component tree used for propagation.
in	<i>a_</i>	Attribute image where values are propagated.
out	<i>nb_leaves</i>	Optional. Store the number of leaves in the component.

9.124.2.14 `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

<i>t</i>	Component tree.
<i>f_</i>	Value image.

Definition at line 65 of file `propagate_representative.hh`.

9.125 mln::morpho::tree::filter Namespace Reference

Namespace for attribute filtering.

Functions

- template<typename T , typename F , typename P >
void [direct](#) (const T &tree, [Image](#)< F > &f_, const [Function_v2b](#)< P > &pred_)
Direct non-pruning strategy.
- template<typename T , typename F , typename P >
void [filter](#) (const T &tree, [Image](#)< F > &f_, const [Function_v2b](#)< P > &pred_, const typename F::value &v)
Filter the image f_ with a given value.
- template<typename T , typename F , typename P >
void [max](#) (const T &tree, [Image](#)< F > &f_, const [Function_v2b](#)< P > &pred_)
Max pruning strategy.
- template<typename T , typename F , typename P >
void [min](#) (const T &tree, [Image](#)< F > &f_, const [Function_v2b](#)< P > &pred_)
Min pruning strategy.
- template<typename T , typename F , typename P >
void [subtractive](#) (const T &tree, [Image](#)< F > &f_, const [Function_v2b](#)< P > &pred_)
Subtractive pruning strategy.

9.125.1 Detailed Description

Namespace for attribute filtering.

9.125.2 Function Documentation

9.125.2.1 template<typename T , typename F , typename P > void mln::morpho::tree::filter::direct (const T & tree, [Image](#)< F > & f_, const [Function_v2b](#)< P > & pred_) [inline]

Direct non-pruning strategy.

A node is removed if it does not verify the predicate. The sub-components remain intact.

Parameters

in	<i>tree</i>	Component tree.
out	<i>f_</i>	Image to filter.
in	<i>pred_</i>	Filtering criterion.

Definition at line 73 of file direct.hh.

9.125.2.2 template<typename T , typename F , typename P > void mln::morpho::tree::filter::filter (const T & tree, [Image](#)< F > & f_, const [Function_v2b](#)< P > & pred_, const typename F::value & v) [inline]

Filter the image f_ with a given value.

The sub-components of nodes that does not match the predicate *pred_* are filled with the given value *v*.

Parameters

<i>tree</i>	Component tree.
<i>f_</i>	Image function.
<i>pred_</i>	Predicate.

<code>v</code>	Value to propagate.
----------------	---------------------

Definition at line 73 of file `morpho/tree/filter/filter.hh`.

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

9.125.2.3 `template<typename T, typename F, typename P> void mln::morpho::tree::filter::max (const T & tree, Image< F > & f_, const Function_v2b< P > & pred_) [inline]`

Max pruning strategy.

A node is removed iif all of its children are removed or if it does not verify the predicate `pred_`.

Parameters

in	<i>tree</i>	Component tree.
out	<i>f_</i>	Image to filter.
in	<i>pred_</i>	Filtering criterion.

Definition at line 74 of file `morpho/tree/filter/max.hh`.

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

9.125.2.4 `template<typename T, typename F, typename P> void mln::morpho::tree::filter::min (const T & tree, Image< F > & f_, const Function_v2b< P > & pred_) [inline]`

Min pruning strategy.

A node is removed iif its parent is removed or if it does not verify the predicate `pred_`.

Parameters

in	<i>tree</i>	Component tree.
out	<i>f_</i>	Image to filter.
in	<i>pred_</i>	Filtering criterion.

Definition at line 75 of file `morpho/tree/filter/min.hh`.

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

9.125.2.5 `template<typename T, typename F, typename P> void mln::morpho::tree::filter::subtractive (const T & tree, Image< F > & f_, const Function_v2b< P > & pred_) [inline]`

Subtractive pruning strategy.

The node is removed if it does not verify the predicate. The sub-components values are set to the value of the removed component.

Parameters

in	<i>tree</i>	Component tree.
out	<i>f_</i>	Image to filter.
in	<i>pred_</i>	Filtering criterion.

Definition at line 77 of file `subtractive.hh`.

References `mln::morpho::tree::propagate_if()`.

9.126 mln::morpho::watershed Namespace Reference

Namespace of morphological watershed routines.

Namespaces

- namespace [watershed](#)
Namespace of morphological watershed routines implementations.

Functions

- template<typename L , typename I , typename N >
mln::trait::ch_value< I, L >::ret [flooding](#) (const [Image](#)< I > &input, const [Neighborhood](#)< N > &nbh, L &n_basins)
Meyer's Watershed Transform (WST) algorithm.
- template<typename L , typename I , typename N >
mln::trait::ch_value< I, L >::ret [flooding](#) (const [Image](#)< I > &input, const [Neighborhood](#)< N > &nbh)
Meyer's Watershed Transform (WST) algorithm, with no count of basins.
- template<typename I , typename J >
mln::trait::ch_value< I, [value::rgb8](#) >::ret [superpose](#) (const [Image](#)< I > &input_, const [Image](#)< J > &ws_ima_, const [value::rgb8](#) &wsl_color)
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)
Convert an image to a rgb8 image and draw the watershed lines.
- template<class T >
T::image_t [topological](#) (T &tree)
Compute a toological watershed transform from tree.

9.126.1 Detailed Description

Namespace of morphological watershed routines.

9.126.2 Function Documentation

- 9.126.2.1 template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret mln::morpho::watershed::flooding (const [Image](#)< I > & input, const [Neighborhood](#)< N > & nbh, L & n_basins) [\[inline\]](#)

Meyer's Watershed Transform (WST) algorithm.

Parameters

in	<i>input</i>	The input image.
in	<i>nbh</i>	The connexity of markers.
out	<i>n_basins</i>	The number of basins.

- L is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- I is the exact type of the input image.
- N is the exact type of the neighborhood used to express *input's* connexity.

Definition at line 381 of file flooding.hh.

9.126.2.2 `template<typename L , typename I , typename N > mln::trait::ch_value< I, L >::ret mln::morpho::watershed::flooding (const Image< I > & input, const Neighborhood< N > & nbh)`

Meyer's Watershed Transform (WST) algorithm, with no count of basins.

Parameters

<code>in</code>	<code>input</code>	The input image.
<code>in</code>	<code>nbh</code>	The connexity of markers.

- `L` is the type of labels, used to number the watershed itself (with the minimal value), and the basins.
- `I` is the exact type of the input image.
- `N` is the exact type of the neighborhood used to express *input's* connexity.

Note that the first parameter, `L`, is not automatically valued from the type of the actual argument during implicit instantiation: you have to explicitly pass this parameter at call sites.

Definition at line 396 of file flooding.hh.

9.126.2.3 `template<typename I , typename J > mln::trait::ch_value< I, value::rgb8 >::ret mln::morpho::watershed::superpose (const Image< I > & input_, const Image< J > & ws_ima_, const value::rgb8 & wsl_color) [inline]`

Convert an image to a rgb8 image and draw the watershed lines.

Definition at line 85 of file morpho/watershed/superpose.hh.

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

Referenced by `superpose()`.

9.126.2.4 `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) [inline]`

Convert an image to a rgb8 image and draw the watershed lines.

Definition at line 109 of file morpho/watershed/superpose.hh.

References `mln::literal::red`, and `superpose()`.

9.126.2.5 `template<class T > T::image_t mln::morpho::watershed::topological (T & tree)`

Compute a toological watershed transform from *tree*.

Definition at line 675 of file topological.hh.

References `mln::data::fill()`, `mln::p_priority< P, Q >::front()`, `mln::initialize()`, `mln::p_priority< P, Q >::pop()`, `mln::p_priority< P, Q >::push()`, and `topological()`.

Referenced by `topological()`.

9.127 mln::morpho::watershed::watershed Namespace Reference

Namespace of morphological watershed routines implementations.

Namespaces

- namespace [generic](#)
Namespace of morphological watershed routines generic implementations.

9.127.1 Detailed Description

Namespace of morphological watershed routines implementations.

9.128 mln::morpho::watershed::watershed::generic Namespace Reference

Namespace of morphological watershed routines generic implementations.

9.128.1 Detailed Description

Namespace of morphological watershed routines generic implementations.

9.129 mln::norm Namespace Reference

Namespace of norms.

Namespaces

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

Functions

- template<unsigned n, typename C >
mln::trait::value_< typename
mln::trait::op::times< C, C >
::ret >::sum [l2](#) (const C(&vec)[n])
L2-norm of a vector vec.
- template<unsigned n, typename C >
mln::trait::value_< typename
mln::trait::op::times< C, C >
::ret >::sum [l1](#) (const C(&vec)[n])
L1-norm of a vector vec.
- template<unsigned n, typename C >
mln::trait::value_< typename
mln::trait::op::times< C, C >
::ret >::sum [l1_distance](#) (const C(&vec1)[n], const C(&vec2)[n])
L1-norm distance between vectors vec1 and vec2.
- template<unsigned n, typename C >
mln::trait::value_< typename
mln::trait::op::times< C, C >
::ret >::sum [sqr_l2](#) (const C(&vec)[n])

Squared L2-norm of a vector `vec`.

- `template<unsigned n, typename C >`
`mln::trait::value_< typename`
`mln::trait::op::times< C, C >`
`::ret >::sum l2_distance (const C(&vec1)[n], const C(&vec2)[n])`

L2-norm distance between vectors `vec1` and `vec2`.

- `template<unsigned n, typename C >`
`C 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`.

9.129.1 Detailed Description

Namespace of norms.

9.129.2 Function Documentation

- 9.129.2.1 `template<unsigned n, typename C > mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum`
`mln::norm::l1 (const C(&) vec[n]) [inline]`

L1-norm of a vector `vec`.

Definition at line 108 of file l1.hh.

- 9.129.2.2 `template<unsigned n, typename C > mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum`
`mln::norm::l1_distance (const C(&) vec1[n], const C(&) vec2[n]) [inline]`

L1-norm distance between vectors `vec1` and `vec2`.

Definition at line 124 of file l1.hh.

- 9.129.2.3 `template<unsigned n, typename C > mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum`
`mln::norm::l2 (const C(&) vec[n]) [inline]`

L2-norm of a vector `vec`.

Definition at line 139 of file l2.hh.

- 9.129.2.4 `template<unsigned n, typename C > mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum`
`mln::norm::l2_distance (const C(&) vec1[n], const C(&) vec2[n]) [inline]`

L2-norm distance between vectors `vec1` and `vec2`.

Definition at line 173 of file l2.hh.

- 9.129.2.5 `template<unsigned n, typename C > C mln::norm::linfty (const C(&) vec[n]) [inline]`

L-infinity-norm of a vector `vec`.

Definition at line 113 of file linfty.hh.

9.129.2.6 `template<unsigned n, typename C > C mln::norm::linfty_distance (const C(&) vec1[n], const C(&) vec2[n])`
`[inline]`

L-infinity-norm distance between vectors *vec1* and *vec2*.

Definition at line 127 of file *linfty.hh*.

9.129.2.7 `template<unsigned n, typename C > mln::trait::value_< typename mln::trait::op::times< C, C >::ret >::sum`
`mln::norm::sqr_l2 (const C(&) vec[n]) [inline]`

Squared L2-norm of a vector *vec*.

Definition at line 156 of file *l2.hh*.

Referenced by `mln::geom::mesh_corner_point_area()`, and `mln::geom::mesh_normal()`.

9.130 mln::norm::impl Namespace Reference

Implementation namespace of norm namespace.

9.130.1 Detailed Description

Implementation namespace of norm namespace.

9.131 mln::opt Namespace Reference

Namespace of optional routines.

Namespaces

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

Functions

- `template<typename I >`
`I::rvalue at (const Image< I > &ima, def::coord ind)`
*One dimension Read-only access to the *ima* value located at (*ind*).*
- `template<typename I >`
`I::lvalue at (Image< I > &ima, def::coord ind)`
*Read-write access to the *ima* value located at (*ind*).*
- `template<typename I >`
`I::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 row, def::coord col)`
*Read-write access to the *ima* value located at (*row*, *col*).*
- `template<typename I >`
`I::rvalue at (const Image< I > &ima, def::coord sli, def::coord row, def::coord col)`
*Three dimensions Read-only access to the *ima* value located at (*sli*, *row*, *col*).*

- `template<typename I >`
`I::lvalue at (Image< I > &ima, def::coord sli, def::coord row, def::coord col)`
Read-write access to the ima value located at (sli, row, col).

9.131.1 Detailed Description

Namespace of optional routines.

9.131.2 Function Documentation

9.131.2.1 `template<typename I > I::rvalue mln::opt::at (const Image< I > &ima, def::coord ind)` `[inline]`

One dimension Read-only access to the ima value located at (ind).

Definition at line 151 of file at.hh.

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

9.131.2.2 `template<typename I > I::lvalue mln::opt::at (Image< I > &ima, def::coord ind)`

Read-write access to the ima value located at (ind).

Definition at line 160 of file at.hh.

9.131.2.3 `template<typename I > I::rvalue mln::opt::at (const Image< I > &ima, def::coord row, def::coord col)`
`[inline]`

Two dimensions Read-only access to the ima value located at (row, col).

Definition at line 236 of file at.hh.

9.131.2.4 `template<typename I > I::lvalue mln::opt::at (Image< I > &ima, def::coord row, def::coord col)`

Read-write access to the ima value located at (row, col).

Definition at line 245 of file at.hh.

9.131.2.5 `template<typename I > I::rvalue mln::opt::at (const Image< I > &ima, def::coord sli, def::coord row, def::coord col)` `[inline]`

Three dimensions Read-only access to the ima value located at (sli, row, col).

Definition at line 320 of file at.hh.

9.131.2.6 `template<typename I > I::lvalue mln::opt::at (Image< I > &ima, def::coord sli, def::coord row, def::coord col)`

Read-write access to the ima value located at (sli, row, col).

Definition at line 330 of file at.hh.

9.132 mln::opt::impl Namespace Reference

Implementation namespace of opt namespace.

9.132.1 Detailed Description

Implementation namespace of opt namespace. Three dimensions.

Two dimensions.

One dimension.

9.133 mln::pw Namespace Reference

Namespace of "point-wise" expression tools.

Classes

- class [image](#)
A generic point-wise image implementation.

9.133.1 Detailed Description

Namespace of "point-wise" expression tools.

9.134 mln::registration Namespace Reference

Namespace of "point-wise" expression tools.

Classes

- class [closest_point_basic](#)
Closest point functor based on map distance.
- class [closest_point_with_map](#)
Closest point functor based on map distance.

Functions

- template<typename P , typename F >
algebra::quat [get_rot](#) (const [p_array](#)< P > &P_, const [vec3d_f](#) &mu_P, const [vec3d_f](#) &mu_Yk, const F &closest_point, const algebra::quat &qR, const [vec3d_f](#) &qT)
FIXME: work only for 3d images.
- template<typename P , typename F >
std::pair< algebra::quat, mln_vec(P)> [icp](#) (const [p_array](#)< P > &P_, const [p_array](#)< P > &X, const F &closest_point, const algebra::quat &initial_rot, const mln_vec(P)&initial_translation)
Base version of the ICP algorithm. It is called in other variants.
- template<typename P , typename F >
[composed](#)< [translation](#)< P::dim, float >, [rotation](#)< P::dim, float > > [icp](#) (const [p_array](#)< P > &P_, const [p_array](#)< P > &X, const F &closest_point)
- template<typename P >
[composed](#)< [translation](#)< P::dim, float >, [rotation](#)< P::dim, float > > [registration1](#) (const [box](#)< P > &domain, const [p_array](#)< P > &P_, const [p_array](#)< P > &X)

Call ICP once and return the resulting transformation.

- `template<typename P >`
`composed< translation< P::dim,`
`float >, rotation< P::dim,`
`float > > registration2 (const box< P > &domain, const p_array< P > &P_, const p_array< P > &X)`

Call ICP 10 times.

- `template<typename P >`
`composed< translation< P::dim,`
`float >, rotation< P::dim,`
`float > > registration3 (const box< P > &domain, const p_array< P > &P_, const p_array< P > &X)`

Call ICP 10 times.

9.134.1 Detailed Description

Namespace of "point-wise" expression tools.

9.134.2 Function Documentation

9.134.2.1 `template<typename P , typename F > algebra::quat mln::registration::get_rot (const p_array< P > &P_, const vec3d_f & mu_P, const vec3d_f & mu_Yk, const F & closest_point, const algebra::quat & qR, const vec3d_f & qT)`

FIXME: work only for 3d images.

Definition at line 527 of file icp.hh.

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

9.134.2.2 `template<typename P , typename F > std::pair< algebra::quat, mln_vec(P)> mln::registration::icp (const p_array< P > &P_, const p_array< P > &X, const F & closest_point, const algebra::quat & initial_rot, const mln_vec(P)& initial_translation) [inline]`

Base version of the ICP algorithm. It is called in other variants.

Register point in `c` using a function of closest points `closest_point`. This overload allows to specify initial transformations.

Parameters

in	<code>P_</code>	The cloud of points.
in	<code>X</code>	the reference surface.
in	<code>closest_point</code>	The function of closest points.
in	<code>initial_rot</code>	An initial rotation.
in	<code>initial_translation</code>	An initial translation.

Returns

the rigid transformation which may be use later to create a registered image.

WARNING: the function `closest_point` *MUST* take float/double vector as arguments. Otherwise the resulting transformation may be wrong due to the truncation of the vector coordinate values.

Precondition

`P_` and `X` must not be empty.

Reference article: "A Method for Registration of 3-D Shapes", Paul J. Besl and Neil D. McKay, IEEE, 2, February 1992.

Definition at line 612 of file icp.hh.

References `mln::geom::bbox()`, `mln::literal::black`, `mln::set::compute()`, `mln::duplicate()`, `mln::box< P >::enlarge()`, `mln::data::fill()`, `mln::literal::green`, `mln::io::ppm::save()`, and `mln::literal::white`.

9.134.2.3 `template<typename P , typename F > composed< translation<P::dim,float>,rotation<P::dim,float> >
mln::registration::icp (const p_array< P > & P_, const p_array< P > & X, const F & closest_point)`

Register point in `c` using a function of closest points `closest_point`.

Parameters

<code>in</code>	<code>P_</code>	The cloud of points.
<code>in</code>	<code>X</code>	the reference surface.
<code>in</code>	<code>closest_point</code>	The function of closest points.

Returns

the rigid transformation which may be use later to create a registered image.

9.134.2.4 `template<typename P > composed< translation< P::dim, float >, rotation< P::dim, float > >
mln::registration::registration1 (const box< P > & domain, const p_array< P > & P_, const p_array< P > & X)
[inline]`

Call ICP once and return the resulting transformation.

Definition at line 325 of file registration.hh.

9.134.2.5 `template<typename P > composed< translation< P::dim, float >, rotation< P::dim, float > >
mln::registration::registration2 (const box< P > & domain, const p_array< P > & P_, const p_array< P > & X)
[inline]`

Call ICP 10 times.

Do the first call to ICP with all sites then work on a subset of which size is decreasing. For each call, a distance criterion is computed on a subset. Sites part of the subset which are too far or too close are removed. Removed sites are *NOT* reused later in the subset.

Definition at line 345 of file registration.hh.

9.134.2.6 `template<typename P > composed< translation< P::dim, float >, rotation< P::dim, float > >
mln::registration::registration3 (const box< P > & domain, const p_array< P > & P_, const p_array< P > & X)
[inline]`

Call ICP 10 times.

Do the first call to ICP with all sites then work on a subset. For each call, a distance criterion is computed on a subset. A new subset is computed from the whole set of points according to this distance. It will be used in the next call. Removed Sites *MAY* be reintegrated.

Definition at line 365 of file registration.hh.

9.135 mln::select Namespace Reference

Select namespace (FIXME doc).

Classes

- struct [p_of](#)
Structure [p_of](#).

9.135.1 Detailed Description

Select namespace (FIXME doc).

9.136 mln::set Namespace Reference

Namespace of image processing routines related to pixel sets.

Functions

- template<typename S >
unsigned [card](#) (const [Site_Set](#)< S > &s)
Compute the cardinality of the site set s .
- template<typename A , typename S >
A::result [compute](#) (const [Accumulator](#)< A > &a, const [Site_Set](#)< S > &s)
Compute an accumulator onto a site set.
- template<typename A , typename I >
A::result [compute_with_weights](#) (const [Accumulator](#)< A > &a, const [Image](#)< I > &w)
Compute an accumulator on a site set described by an image.
- template<typename A , typename I , typename L >
[util::array](#)< typename A::result > [compute_with_weights](#) (const [Accumulator](#)< A > &a, const [Image](#)< I > &w, const [Image](#)< L > &label, const typename L::value &nlabels)
Compute an accumulator on every labeled sub-site-sets.
- template<typename S >
S::site [get](#) (const [Site_Set](#)< S > &s, size_t index)
FIXME.
- template<typename S >
bool [has](#) (const [Site_Set](#)< S > &s, const typename S::site &e)
FIXME.
- template<typename A , typename S >
[mln_meta_accu_result](#) (A, typename S::site) [compute](#)(const [Meta_Accumulator](#)< A > &a)
Compute an accumulator onto a site set.
- template<typename A , typename I >
[mln_meta_accu_result](#) (A, typename I::site) [compute_with_weights](#)(const [Meta_Accumulator](#)< A > &a)
Compute an accumulator on a site set described by an image.

9.136.1 Detailed Description

Namespace of image processing routines related to pixel sets.

9.136.2 Function Documentation

9.136.2.1 template<typename S > unsigned mln::set::card (const Site_Set< S > & s) [inline]

Compute the cardinality of the site set s .

Definition at line 134 of file set/card.hh.

9.136.2.2 `template<typename A , typename S > A::result mln::set::compute (const Accumulator< A > & a, const Site_Set< S > & s) [inline]`

Compute an accumulator onto a site set.

Parameters

<i>in</i>	<i>a</i>	An accumulator.
<i>in</i>	<i>s</i>	A site set.

Returns

The accumulator result.

Definition at line 112 of file set/compute.hh.

Referenced by mln::registration::icp().

9.136.2.3 `template<typename A , typename I > A::result mln::set::compute_with_weights (const Accumulator< A > & a, const Image< I > & w) [inline]`

Compute an accumulator on a site set described by an image.

Parameters

<i>in</i>	<i>a</i>	An accumulator.
<i>in</i>	<i>w</i>	An image of weights (a site -> a weight).

Returns

The accumulator result.

Definition at line 217 of file compute_with_weights.hh.

9.136.2.4 `template<typename A , typename I , typename L > util::array< typename A::result > mln::set::compute_with_weights (const Accumulator< A > & a, const Image< I > & w, const Image< L > & label, const typename L::value & nlabels)`

Compute an accumulator on every labeled sub-site-sets.

Parameters

<i>in</i>	<i>a</i>	An accumulator.
<i>in</i>	<i>w</i>	An image of weights (a site -> a weight).
<i>in</i>	<i>label</i>	A label image.
<i>in</i>	<i>nlabels</i>	The number of labels in <i>label</i> .

Returns

An array of accumulator result. One per label.

Definition at line 234 of file compute_with_weights.hh.

9.136.2.5 `template<typename S > S::site mln::set::get (const Site_Set< S > & s, size_t index)`

FIXME.

Definition at line 56 of file set/get.hh.

9.136.2.6 `template<typename S> bool mln::set::has (const Site_Set< S> & s, const typename S::site & e)`

FIXME.

Definition at line 56 of file set/has.hh.

9.136.2.7 `template<typename A , typename S> mln::set::mln_meta_accu_result (A , typename S::site) const`

Compute an accumulator onto a site set.

Parameters

<code>in</code>	<code>a</code>	A meta-accumulator.
<code>in</code>	<code>s</code>	A site set.

9.136.2.8 `template<typename A , typename I> mln::set::mln_meta_accu_result (A , typename I::site) const` `[inline]`

Compute an accumulator on a site set described by an image.

Parameters

<code>in</code>	<code>a</code>	A meta-accumulator.
<code>in</code>	<code>w</code>	An image of weights (a site -> a weight).

Returns

The accumulator result.

9.137 mln::subsampling Namespace Reference

Namespace of "point-wise" expression tools.

Functions

- `template<typename I>`
`mln::trait::concrete< I>::ret antialiased (const Image< I> &input, unsigned factor, const typename I::domain_t &output_domain, unsigned border_thickness)`
Antialiased subsampling.
- `template<typename I>`
`mln::trait::concrete< I>::ret antialiased (const Image< I> &input, unsigned factor)`
- `template<typename I>`
`mln::trait::concrete< I>::ret gaussian_subsampling (const Image< I> &input, float sigma, const typename I::dpsite &first_p, const typename I::site::coord &gap)`
Gaussian subsampling FIXME : doxy.
- `template<typename I>`
`mln::trait::concrete< I>::ret subsampling (const Image< I> &input, const typename I::site::delta &first_p, const typename I::site::coord &gap)`
Subsampling FIXME : doxy.

9.137.1 Detailed Description

Namespace of "point-wise" expression tools.

9.137.2 Function Documentation

9.137.2.1 `template<typename I> mln::trait::concrete<I>::ret mln::subsampling::antialiased (const Image<I> & input, unsigned factor, const typename I::domain_t & output_domain, unsigned border_thickness) [inline]`

Antialiased subsampling.

Parameters

in	<i>input</i>	A gray-level image.
in	<i>factor</i>	Subsampling ratio. Must be divisible by 2 or 3.
in	<i>output_domain</i>	Force output domain.
in	<i>border_thickness</i>	Force output border thickness.

Definition at line 428 of file antialiased.hh.

Referenced by antialiased().

9.137.2.2 `template<typename I> mln::trait::concrete<I>::ret mln::subsampling::antialiased (const Image<I> & input, unsigned factor) [inline]`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 453 of file antialiased.hh.

References antialiased().

9.137.2.3 `template<typename I> mln::trait::concrete<I>::ret mln::subsampling::gaussian_subsampling (const Image<I> & input, float sigma, const typename I::dpsite & first_p, const typename I::site::coord & gap) [inline]`

Gaussian subsampling FIXME : doxy.

Definition at line 63 of file gaussian_subsampling.hh.

References mln::linear::gaussian(), mln::geom::ncols(), and mln::geom::nrows().

9.137.2.4 `template<typename I> mln::trait::concrete<I>::ret mln::subsampling::subsampling (const Image<I> & input, const typename I::site::delta & first_p, const typename I::site::coord & gap) [inline]`

Subsampling FIXME : doxy.

Definition at line 91 of file subsampling.hh.

References mln::geom::ncols(), and mln::geom::nrows().

9.138 mln::tag Namespace Reference

Namespace of image processing routines related to tags.

9.138.1 Detailed Description

Namespace of image processing routines related to tags.

9.139 mln::test Namespace Reference

Namespace of image processing routines related to pixel tests.

Namespaces

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

Functions

- template<typename I >
bool [positive](#) (const [Image](#)< I > &input)
Test if an image only contains positive values.
- template<typename 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.
- 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 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.

9.139.1 Detailed Description

Namespace of image processing routines related to pixel tests.

9.139.2 Function Documentation

9.139.2.1 template<typename I > bool mln::test::positive (const Image< I > & input) [inline]

Test if an image only contains positive values.

Definition at line 54 of file positive.hh.

References [predicate\(\)](#), and [mln::literal::zero](#).

Referenced by [mln::morpho::gradient\(\)](#), [mln::morpho::gradient_external\(\)](#), [mln::morpho::gradient_internal\(\)](#), [mln::morpho::top_hat_black\(\)](#), [mln::morpho::elementary::top_hat_black\(\)](#), [mln::morpho::top_hat_self_complementary\(\)](#), [mln::morpho::elementary::top_hat_self_complementary\(\)](#), [mln::morpho::top_hat_white\(\)](#), and [mln::morpho::elementary::top_hat_white\(\)](#).

9.139.2.2 template<typename 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

<code>in</code>	<code>ima</code>	The image.
<code>in</code>	<code>f</code>	The predicate.

Definition at line 207 of file predicate.hh.

Referenced by `mln::operator<()`, `mln::operator<=()`, `mln::operator==()`, and `positive()`.

9.139.2.3 `template<typename I, typename J, typename F> bool mln::test::predicate (const Image< I > & lhs, const Image< J > & rhs, const Function_vv2b< F > & f) [inline]`

Test if all pixel values of `lhs` and `rhs` verify the predicate `f`.

Parameters

<code>in</code>	<code>lhs</code>	The image.
<code>in</code>	<code>rhs</code>	The image.
<code>in</code>	<code>f</code>	The predicate.

Definition at line 222 of file predicate.hh.

9.139.2.4 `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

<code>in</code>	<code>pset</code>	The point set.
<code>in</code>	<code>f</code>	The predicate.

Definition at line 242 of file predicate.hh.

9.140 mln::test::impl Namespace Reference

Implementation namespace of test namespace.

9.140.1 Detailed Description

Implementation namespace of test namespace.

9.141 mln::topo Namespace Reference

Namespace of "point-wise" expression tools.

Classes

- class [adj_higher_dim_connected_n_face_bkd_iter](#)
Backward iterator on all the n -faces sharing an adjacent $(n+1)$ -face with a (reference) n -face of an `mln::complex<D>`.
- class [adj_higher_dim_connected_n_face_fwd_iter](#)
Forward iterator on all the n -faces sharing an adjacent $(n+1)$ -face with a (reference) n -face of an `mln::complex<D>`.

- class [adj_higher_face_bkd_iter](#)
Backward iterator on all the adjacent $(n+1)$ -faces of the n -face of an `mln::complex<D>`.
- class [adj_higher_face_fwd_iter](#)
Forward iterator on all the adjacent $(n+1)$ -faces of the n -face of an `mln::complex<D>`.
- class [adj_lower_dim_connected_n_face_bkd_iter](#)
Backward iterator on all the n -faces sharing an adjacent $(n-1)$ -face with a (reference) n -face of an `mln::complex<D>`.
- class [adj_lower_dim_connected_n_face_fwd_iter](#)
Forward iterator on all the n -faces sharing an adjacent $(n-1)$ -face with a (reference) n -face of an `mln::complex<D>`.
- class [adj_lower_face_bkd_iter](#)
Backward iterator on all the adjacent $(n-1)$ -faces of the n -face of an `mln::complex<D>`.
- class [adj_lower_face_fwd_iter](#)
Forward iterator on all the adjacent $(n-1)$ -faces of the n -face of an `mln::complex<D>`.
- class [adj_lower_higher_face_bkd_iter](#)
Forward iterator on all the adjacent $(n-1)$ -faces and $(n+1)$ -faces of the n -face of an `mln::complex<D>`.
- class [adj_lower_higher_face_fwd_iter](#)
Forward iterator on all the adjacent $(n-1)$ -faces and $(n+1)$ -faces of the n -face of an `mln::complex<D>`.
- class [adj_m_face_bkd_iter](#)
Backward iterator on all the m -faces transitively adjacent to a (reference) n -face in a complex.
- class [adj_m_face_fwd_iter](#)
Forward iterator on all the m -faces transitively adjacent to a (reference) n -face in a complex.
- class [algebraic_face](#)
Algebraic face handle in a complex; the face dimension is dynamic.
- class [algebraic_n_face](#)
Algebraic N -face handle in a complex.
- class [center_only_iter](#)
Iterator on all the adjacent $(n-1)$ -faces of the n -face of an `mln::complex<D>`.
- class [centered_bkd_iter_adapter](#)
Forward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.
- class [centered_fwd_iter_adapter](#)
Backward complex relative iterator adapters adding the central (reference) point to the set of iterated faces.
- class [complex](#)
General complex of dimension D .
- class [face](#)
Face handle in a complex; the face dimension is dynamic.
- class [face_bkd_iter](#)
Backward iterator on all the faces of an `mln::complex<D>`.
- class [face_fwd_iter](#)
Forward iterator on all the faces of an `mln::complex<D>`.
- struct [is_n_face](#)
A functor testing wheter a `mln::complex_psit` is an N -face.
- struct [is_simple_2d_t](#)
Test if a point is simple or not.
- class [is_simple_cell](#)
A predicate for the simplicity of a point based on the collapse property of the attachment.
- class [n_face](#)
 N -face handle in a complex.
- class [n_face_bkd_iter](#)
Backward iterator on all the faces of an `mln::complex<D>`.
- class [n_face_fwd_iter](#)
Forward iterator on all the faces of an `mln::complex<D>`.
- class [n_faces_set](#)

Set of face handles of dimension N .

- class [static_n_face_bkd_iter](#)
Backward iterator on all the N -faces of a `mln::complex<D>`.
- class [static_n_face_fwd_iter](#)
Forward iterator on all the N -faces of a `mln::complex<D>`.

Functions

- `template<unsigned D, typename G >`
`void detach (const complex_psite< D, G > &f, complex_image< D, G, bool > &ima)`
Detach the cell corresponding to f from ima .
- `template<unsigned D, typename G >`
`bool is_facet (const complex_psite< D, G > &f)`
Is f a facet, i.e., a face not "included in" (adjacent to) a face of higher dimension?
- `template<unsigned D>`
`algebraic_face< D > make_algebraic_face (const face< D > &f, bool sign)`
Create an algebraic face handle of a D -complex.
- `template<unsigned N, unsigned D>`
`algebraic_n_face< N, D > make_algebraic_n_face (const n_face< N, D > &f, bool sign)`
Create an algebraic N -face handle of a D -complex.
- `template<unsigned N, unsigned D>`
`std::ostream & operator<< (std::ostream &ostr, const algebraic_n_face< N, D > &f)`
Print an `mln::topo::algebraic_n_face`.
- `template<unsigned D>`
`std::ostream & operator<< (std::ostream &ostr, const algebraic_face< D > &f)`
Print an `mln::topo::algebraic_face`.
- `template<unsigned N, unsigned D>`
`std::ostream & operator<< (std::ostream &ostr, const n_face< N, D > &f)`
Print an `mln::topo::n_face`.
- `template<unsigned D>`
`std::ostream & operator<< (std::ostream &ostr, const face< D > &f)`
Print an `mln::topo::face`.
- `template<unsigned D>`
`std::ostream & operator<< (std::ostream &ostr, const complex< D > &c)`
Pretty print a complex.
- `template<unsigned D>`
`bool operator== (const complex< D > &lhs, const complex< D > &rhs)`
Compare two complexes for equality.
- `template<unsigned D>`
`algebraic_face< D > operator- (const face< D > &f)`
Inversion operators.
- `template<unsigned D>`
`bool operator== (const algebraic_face< D > &lhs, const algebraic_face< D > &rhs)`
Comparison of two instances of `mln::topo::algebraic_face`.
- `template<unsigned D>`
`bool operator!= (const algebraic_face< D > &lhs, const algebraic_face< D > &rhs)`
Is lhs different from rhs?
- `template<unsigned D>`
`bool operator< (const algebraic_face< D > &lhs, const algebraic_face< D > &rhs)`
Is lhs "less" than rhs?

- `template<unsigned N, unsigned D>`
`algebraic_n_face< N, D > operator-` (const `n_face< N, D >` &f)
Inversion operators.
- `template<unsigned N, unsigned D>`
`bool operator==` (const `algebraic_n_face< N, D >` &lhs, const `algebraic_n_face< N, D >` &rhs)
Comparison of two instances of `mln::topo::algebraic_n_face`.
- `template<unsigned N, unsigned D>`
`bool operator!=` (const `algebraic_n_face< N, D >` &lhs, const `algebraic_n_face< N, D >` &rhs)
Is lhs different from rhs?
- `template<unsigned N, unsigned D>`
`bool operator<` (const `algebraic_n_face< N, D >` &lhs, const `algebraic_n_face< N, D >` &rhs)
Is lhs "less" than rhs?
- `template<unsigned D>`
`algebraic_n_face< 1, D > edge` (const `n_face< 0, D >` &f1, const `n_face< 0, D >` &f2)
Helpers.
- `template<unsigned D>`
`bool operator==` (const `face< D >` &lhs, const `face< D >` &rhs)
Comparison of two instances of `mln::topo::face`.
- `template<unsigned D>`
`bool operator!=` (const `face< D >` &lhs, const `face< D >` &rhs)
Is lhs different from rhs?
- `template<unsigned D>`
`bool operator<` (const `face< D >` &lhs, const `face< D >` &rhs)
Is lhs "less" than rhs?
- `template<unsigned N, unsigned D>`
`bool operator==` (const `n_face< N, D >` &lhs, const `n_face< N, D >` &rhs)
Comparison of two instances of `mln::topo::n_face`.
- `template<unsigned N, unsigned D>`
`bool operator!=` (const `n_face< N, D >` &lhs, const `n_face< N, D >` &rhs)
Is lhs different from rhs?
- `template<unsigned N, unsigned D>`
`bool operator<` (const `n_face< N, D >` &lhs, const `n_face< N, D >` &rhs)
Is lhs "less" than rhs?
- `template<unsigned N, unsigned D>`
`n_faces_set< N, D > operator+` (const `algebraic_n_face< N, D >` &f1, const `algebraic_n_face< N, D >` &f2)
Addition.
- `template<unsigned N, unsigned D>`
`n_faces_set< N, D > operator-` (const `algebraic_n_face< N, D >` &f1, const `algebraic_n_face< N, D >` &f2)
Subtraction.

9.141.1 Detailed Description

Namespace of "point-wise" expression tools.

9.141.2 Function Documentation

9.141.2.1 `template<unsigned D, typename G > void mln::topo::detach (const complex_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.

Definition at line 58 of file detach.hh.

References `mln::make::detachment()`, `mln::data::fill()`, and `is_facet()`.

9.141.2.2 `template<unsigned D> algebraic_n_face< 1, D > mln::topo::edge (const n_face< 0, D > & f1, const n_face< 0, D > & f2)`

Helpers.

Return the algebraic 1-face (edge) linking the 0-faces (vertices) *f1* and *f2*. If there is no 1-face between *f1* and *f2*, return an invalid 1-face.

Precondition

f1 and *f2* must belong to the same complex.

Note: this routine assumes the complex is not degenerated, i.e.,

- it does not check that *f1* and *f2* are the only 0-faces adjacent to an hypothetical 1-face; it just checks that *f1* and *f2* share a common 1-face;
- if there are several adjacent 1-faces shared by *f1* and *f2* (if the complex is ill-formed), there is no guarantee on the returned 1-face (the current implementation return the first 1-face found, but client code should not rely on this implementation-defined behavior).

Definition at line 286 of file algebraic_n_face.hh.

References `mln::topo::n_face< N, D >::higher_dim_adj_faces()`.

9.141.2.3 `template<unsigned D, typename G > bool mln::topo::is_facet (const complex_psite< D, G > & f) [inline]`

Is *f* a facet, i.e., a face not “included in” (adjacent to) a face of higher dimension?

Definition at line 58 of file is_facet.hh.

Referenced by `mln::make::attachment()`, `mln::make::cell()`, `detach()`, and `mln::make::detachment()`.

9.141.2.4 `template<unsigned D> algebraic_face< D > mln::topo::make_algebraic_face (const face< D > & f, bool sign)`

Create an algebraic face handle of a *D*-complex.

Definition at line 211 of file algebraic_face.hh.

9.141.2.5 `template<unsigned N, unsigned D> algebraic_n_face< N, D > mln::topo::make_algebraic_n_face (const n_face< N, D > & f, bool sign)`

Create an algebraic *N*-face handle of a *D*-complex.

Definition at line 213 of file algebraic_n_face.hh.

9.141.2.6 `template<unsigned N, unsigned D> bool mln::topo::operator!= (const algebraic_n_face< N, D > & lhs, const algebraic_n_face< N, D > & rhs) [inline]`

Is *lhs* different from *rhs*?

Precondition

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

Definition at line 250 of file `algebraic_n_face.hh`.

References `mln::topo::n_face< N, D >::cplx()`.

9.141.2.7 `template<unsigned D> bool mln::topo::operator!= (const algebraic_face< D > & lhs, const algebraic_face< D > & rhs) [inline]`

Is *lhs* different from *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 250 of file `algebraic_face.hh`.

References `mln::topo::face< D >::cplx()`.

9.141.2.8 `template<unsigned N, unsigned D> bool mln::topo::operator!= (const n_face< N, D > & lhs, const n_face< N, D > & rhs) [inline]`

Is *lhs* different from *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 291 of file `n_face.hh`.

References `mln::topo::n_face< N, D >::cplx()`.

9.141.2.9 `template<unsigned D> bool mln::topo::operator!= (const face< D > & lhs, const face< D > & rhs) [inline]`

Is *lhs* different from *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 394 of file `face.hh`.

References `mln::topo::face< D >::cplx()`.

9.141.2.10 `template<unsigned N, unsigned D> n_faces_set< N, D > mln::topo::operator+ (const algebraic_n_face< N, D > & f1, const algebraic_n_face< N, D > & f2) [inline]`

Addition.

Definition at line 206 of file `n_faces_set.hh`.

References `mln::topo::n_faces_set< N, D >::add()`.

9.141.2.11 `template<unsigned N, unsigned D> algebraic_n_face< N, D > mln::topo::operator- (const n_face< N, D > & f)`

Inversion operators.

Definition at line 221 of file algebraic_n_face.hh.

9.141.2.12 `template<unsigned D> algebraic_face< D > mln::topo::operator- (const face< D > & f)`

Inversion operators.

Definition at line 219 of file algebraic_face.hh.

9.141.2.13 `template<unsigned N, unsigned D> n_faces_set< N, D > mln::topo::operator- (const algebraic_n_face< N, D > & f1, const algebraic_n_face< N, D > & f2) [inline]`

Subtraction.

Definition at line 284 of file n_faces_set.hh.

References `mln::topo::n_faces_set< N, D >::add()`.

9.141.2.14 `template<unsigned N, unsigned D> bool mln::topo::operator< (const algebraic_n_face< N, D > & lhs, const algebraic_n_face< N, D > & rhs) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting algebraic face handles.

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 261 of file algebraic_n_face.hh.

9.141.2.15 `template<unsigned D> bool mln::topo::operator< (const algebraic_face< D > & lhs, const algebraic_face< D > & rhs) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting algebraic face handles.

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Arguments *lhs* and *rhs* must have the same dimension.

Definition at line 260 of file algebraic_face.hh.

9.141.2.16 `template<unsigned N, unsigned D> bool mln::topo::operator< (const n_face< N, D > & lhs, const n_face< N, D > & rhs) [inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting face handles.

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 301 of file n_face.hh.

9.141.2.17 `template<unsigned D> bool mln::topo::operator< (const face< D > & lhs, const face< D > & rhs)`
`[inline]`

Is *lhs* “less” than *rhs*?

This comparison is required by algorithms sorting face handles.

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Arguments *lhs* and *rhs* must have the same dimension.

Definition at line 404 of file face.hh.

9.141.2.18 `template<unsigned N, unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const algebraic_n_face< N, D > & f)`
`[inline]`

Print an [mln::topo::algebraic_n_face](#).

Definition at line 273 of file algebraic_n_face.hh.

9.141.2.19 `template<unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const algebraic_face< D > & f)`
`[inline]`

Print an [mln::topo::algebraic_face](#).

Definition at line 273 of file algebraic_face.hh.

9.141.2.20 `template<unsigned N, unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const n_face< N, D > & f)`
`[inline]`

Print an [mln::topo::n_face](#).

Definition at line 312 of file n_face.hh.

9.141.2.21 `template<unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const face< D > & f)`
`[inline]`

Print an [mln::topo::face](#).

Definition at line 416 of file face.hh.

9.141.2.22 `template<unsigned D> std::ostream & mln::topo::operator<< (std::ostream & ostr, const complex< D > & c)`
`[inline]`

Pretty print a complex.

Definition at line 670 of file complex.hh.

References [mln::topo::complex< D >::print\(\)](#).

9.141.2.23 `template<unsigned N, unsigned D> bool mln::topo::operator== (const algebraic_n_face< N, D > & lhs, const algebraic_n_face< N, D > & rhs)`
`[inline]`

Comparison of two instances of [mln::topo::algebraic_n_face](#).

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 239 of file algebraic_n_face.hh.

References [mln::topo::n_face< N, D >::cplx\(\)](#), [mln::topo::n_face< N, D >::face_id\(\)](#), and [mln::topo::algebraic_n_face< N, D >::sign\(\)](#).

9.141.2.24 `template<unsigned D> bool mln::topo::operator==(const algebraic_face< D > & lhs, const algebraic_face< D > & rhs) [inline]`

Comparison of two instances of [mln::topo::algebraic_face](#).

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 237 of file algebraic_face.hh.

References [mln::topo::face< D >::cplx\(\)](#), [mln::topo::face< D >::face_id\(\)](#), [mln::topo::face< D >::n\(\)](#), and [mln::topo::algebraic_face< D >::sign\(\)](#).

9.141.2.25 `template<unsigned N, unsigned D> bool mln::topo::operator==(const n_face< N, D > & lhs, const n_face< N, D > & rhs) [inline]`

Comparison of two instances of [mln::topo::n_face](#).

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 281 of file n_face.hh.

References [mln::topo::n_face< N, D >::cplx\(\)](#), and [mln::topo::n_face< N, D >::face_id\(\)](#).

9.141.2.26 `template<unsigned D> bool mln::topo::operator==(const face< D > & lhs, const face< D > & rhs) [inline]`

Comparison of two instances of [mln::topo::face](#).

Is *lhs* equal to *rhs*?

Precondition

Arguments *lhs* and *rhs* must belong to the same [mln::topo::complex](#).

Definition at line 384 of file face.hh.

References [mln::topo::face< D >::cplx\(\)](#), [mln::topo::face< D >::face_id\(\)](#), and [mln::topo::face< D >::n\(\)](#).

9.141.2.27 `template<unsigned D> bool mln::topo::operator==(const complex< D > & lhs, const complex< D > & rhs) [inline]`

Compare two complexes for equality.

Definition at line 657 of file complex.hh.

9.142 mln::trace Namespace Reference

Namespace of routines related to the trace mechanism.

9.142.1 Detailed Description

Namespace of routines related to the trace mechanism.

9.143 mln::trait Namespace Reference

Namespace where traits are defined.

9.143.1 Detailed Description

Namespace where traits are defined. Namespace for image traits.

9.144 mln::transform Namespace Reference

Namespace of transforms.

Functions

- `template<typename I , typename N , typename D >`
`util::couple`
`< mln::trait::ch_value< I, D >`
`::ret, mln::trait::ch_value< I,`
`typename I::psite >::ret > distance_and_closest_point_geodesic (const Image< I > &input, const Neighborhood< N > &nbh, D max)`
Discrete geodesic distance transform.
- `template<typename P , typename N , typename D >`
`util::couple`
`< mln_image_from_grid(mln_grid(P),`
`D), mln_image_from_grid(mln_grid(P),`
`unsigned)> distance_and_closest_point_geodesic (const p_array< P > &pset, const box< P > &closest_`
`point_domain, const Neighborhood< N > &nbh, D max)`
Discrete geodesic distance transform.
- `template<typename I , typename N , typename D >`
`util::couple`
`< mln::trait::ch_value< I, D >`
`::ret, I > distance_and_influence_zone_geodesic (const Image< I > &input, const Neighborhood< N >`
`&nbh, D max)`
Discrete geodesic distance transform.
- `template<typename I , typename N , typename W , typename D >`
`mln::trait::ch_value< I, D >::ret distance_front (const Image< I > &input, const Neighborhood< N > &nbh,`
`const Weighted_Window< W > &w_win, D max)`
Discrete front distance transform.
- `template<typename I , typename N , typename D >`
`mln::trait::ch_value< I, D >::ret distance_geodesic (const Image< I > &input, const 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 , 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 , 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 >`
`mln::trait::concrete< I >::ret influence_zone_geodesic (const Image< I > &input, const Neighborhood< N > &nbh)`
Geodesic influence zone transform.
- `template<typename I , typename N , typename D >`
`mln::trait::concrete< I >::ret influence_zone_geodesic_saturated (const Image< I > &input, const Neighborhood< N > &nbh, const D &max, const typename I::value &background_value)`
Geodesic influence zone transform.
- `template<typename I , typename N , typename D >`
`mln::trait::concrete< I >::ret influence_zone_geodesic_saturated (const Image< I > &input, const Neighborhood< N > &nbh, const D &max)`

9.144.1 Detailed Description

Namespace of transforms.

9.144.2 Function Documentation

- 9.144.2.1 `template<typename I , typename N , typename D > util::couple< mln::trait::ch_value< I, D >::ret, mln::trait::ch_value< I, typename I::psite >::ret > mln::transform::distance_and_closest_point_geodesic (const Image< I > &input, const Neighborhood< N > &nbh, D max) [inline]`

Discrete geodesic distance transform.

Parameters

<code>in</code>	<code>input</code>	Image from which the geodesic distance is computed.
<code>in</code>	<code>nbh</code>	Neighborhood
<code>in</code>	<code>max</code>	Max distance of propagation.

Returns

a couple of images. The first one is the distance map and the second one is the closest point image. The closest point image contains sites.

Postcondition

The returned images have the same domain as `input`.

Definition at line 90 of file `distance_and_closest_point_geodesic.hh`.

References `mln::make::couple()`, and `mln::canvas::distance_geodesic()`.

```
9.144.2.2  template<typename P , typename N , typename D > util::couple< mln_image_from_grid(mln_grid(P), D),
mln_image_from_grid(mln_grid(P), unsigned)> mln::transform::distance_and_closest_point_geodesic ( const
p_array< P > & pset, const box< P > & closest_point_domain, const Neighborhood< N > & nbh, D max )
[inline]
```

Discrete geodesic distance transform.

Parameters

in	<i>pset</i>	an array of sites.
in	<i>closest_point_domain</i>	domain of the returned image.
in	<i>nbh</i>	neighborhood
in	<i>max</i>	max distance of propagation.

Returns

A couple of images. The first one is the distance map and the second one is the closest point image. The closest point image contains site indexes.

Postcondition

The returned image domains are defined on `closest_point_domain`.

Definition at line 110 of file `distance_and_closest_point_geodesic.hh`.

References `mln::geom::bbox()`, `mln::make::couple()`, `mln::canvas::distance_geodesic()`, `mln::data::fill()`, and `mln::box< P >::is_valid()`.

```
9.144.2.3  template<typename I , typename N , typename D > util::couple< mln::trait::ch_value< I, D >::ret, I >
mln::transform::distance_and_influence_zone_geodesic ( const Image< I > & input, const Neighborhood< N > &
nbh, D max ) [inline]
```

Discrete geodesic distance transform.

Parameters

in	<i>input</i>	Image from which the geodesic distance is computed.
in	<i>nbh</i>	Neighborhood
in	<i>max</i>	Max distance of propagation.

Returns

a couple of images. The first one is the distance map and the second one is the closest point image. The closest point image contains sites.

Postcondition

The returned images have the same domain as `input`.

Definition at line 69 of file `distance_and_influence_zone_geodesic.hh`.

References `mln::make::couple()`, and `mln::canvas::distance_geodesic()`.

9.144.2.4 `template<typename I , typename N , typename W , typename D > mln::trait::ch_value< I, D >::ret
mln::transform::distance_front (const Image< I > & input, const Neighborhood< N > & nbh, const
Weighted_Window< W > & w_win, D max) [inline]`

Discrete front distance transform.

Definition at line 56 of file transform/distance_front.hh.

References mln::canvas::distance_front().

9.144.2.5 `template<typename I , typename N , typename D > mln::trait::ch_value< I, D >::ret
mln::transform::distance_geodesic (const Image< I > & input, const Neighborhood< N > & nbh, D max)
[inline]`

Discrete geodesic distance transform.

Definition at line 55 of file transform/distance_geodesic.hh.

References mln::canvas::distance_geodesic().

9.144.2.6 `template<typename I > image2d< float > mln::transform::hough (const Image< I > & input_)`

Compute the hough transform from a binary image.

Objects used for computation must be set to 'true'.

Parameters

<i>in</i>	<i>input_</i>	A binary image.
-----------	---------------	-----------------

Returns

A 2D image of float. Rows are used for the distance and columns are used for the angles. Angles go from 0 to 359. Distance goes from 0 to the maximum distance between the center and a corner. The site having the maximum value indicates through its column index the document inclination.

Definition at line 98 of file hough.hh.

References mln::opt::at(), mln::data::fill(), mln::geom::min_col(), mln::geom::min_row(), mln::geom::ncols(), and mln::geom::nrows().

9.144.2.7 `template<typename I , typename N , typename W , typename D > mln::trait::concrete< I >::ret
mln::transform::influence_zone_front (const Image< I > & input, const Neighborhood< N > & nbh, const
Weighted_Window< W > & w_win, D max)`

Influence zone transform.

Definition at line 60 of file influence_zone_front.hh.

References mln::canvas::distance_front().

Referenced by influence_zone_front().

9.144.2.8 `template<typename I , typename N , typename W > mln::trait::concrete< I >::ret mln::transform::influence_zone_
front (const Image< I > & input, const Neighborhood< N > & nbh, const Weighted_Window< W > & w_win
)`

Influence zone transform.

Definition at line 78 of file influence_zone_front.hh.

References influence_zone_front().

9.144.2.9 `template<typename I , typename N > mln::trait::concrete< I >::ret mln::transform::influence_zone_geodesic (const Image< I > & input, const Neighborhood< N > & nbh)`

Geodesic influence zone transform.

Parameters

<i>in</i>	<i>input</i>	An image.
<i>in</i>	<i>nbh</i>	A neighborhood.

Returns

An image of influence zone.

Definition at line 219 of file `influence_zone_geodesic.hh`.

9.144.2.10 `template<typename I , typename N , typename D > mln::trait::concrete< I >::ret mln::transform::influence_zone_geodesic_saturated (const Image< I > & input, const Neighborhood< N > & nbh, const D & max, const typename I::value & background_value)`

Geodesic influence zone transform.

Parameters

<i>in</i>	<i>input</i>	An image.
<i>in</i>	<i>nbh</i>	A neighborhood.
<i>in</i>	<i>max</i>	The maximum influence zone distance.
<i>in</i>	<i>background_value</i>	The value used as background (i.e. not propagated).

Returns

An image of influence zone.

Definition at line 73 of file `influence_zone_geodesic_saturated.hh`.

References `mln::canvas::distance_geodesic()`.

Referenced by `influence_zone_geodesic_saturated()`.

9.144.2.11 `template<typename I , typename N , typename D > mln::trait::concrete< I >::ret mln::transform::influence_zone_geodesic_saturated (const Image< I > & input, const Neighborhood< N > & nbh, const D & max)`

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 93 of file `influence_zone_geodesic_saturated.hh`.

References `influence_zone_geodesic_saturated()`, and `mln::literal::zero`.

9.145 mln::util Namespace Reference

Namespace of tools using for more complex algorithm.

Namespaces

- namespace [impl](#)

Implementation namespace of util namespace.

Classes

- class [adjacency_matrix](#)
A class of adjacency matrix.
- class [array](#)
A dynamic array class.
- class [branch](#)
Class of generic branch.
- class [branch_iter](#)
Basic 2D image class.
- class [branch_iter_ind](#)
Basic 2D image class.
- class [couple](#)
Definition of a couple.
- struct [eat](#)
Eat structure.
- class [edge](#)
Edge of a graph G .
- class [fibonacci_heap](#)
Fibonacci heap.
- class [graph](#)
Undirected graph.
- class [greater_point](#)
A "greater than" functor comparing points w.r.t.
- class [greater_psite](#)
A "greater than" functor comparing psites w.r.t.
- class [head](#)
Top structure of the soft heap.
- struct [ignore](#)
Ignore structure.
- struct [ilcell](#)
Element of an item list. Store the data (key) used in [soft_heap](#).
- class [line_graph](#)
Undirected line graph of a graph of type G .
- struct [nil](#)
Nil structure.
- class [node](#)
Meta-data of an element in the heap.
- class [object_id](#)
Base class of an object id.
- struct [ord](#)
Function-object that defines an ordering between objects with type T : lhs R rhs.
- struct [ord_pair](#)
Ordered pair structure s.a.
- struct [pix](#)
Structure pix.
- class [set](#)
An "efficient" mathematical set class.

- class [site_pair](#)
A pair of sites.
- class [soft_heap](#)
Soft heap.
- class [timer](#)
Timer structure.
- struct [tracked_ptr](#)
Smart pointer for shared data with tracking.
- class [tree](#)
Class of generic tree.
- class [tree_node](#)
Class of generic [tree_node](#) for tree.
- class [vertex](#)
[Vertex](#) of a graph G .
- struct [yes](#)
Object that always says "yes".

Typedefs

- typedef [object_id](#)< [vertex_tag](#),
unsigned > [vertex_id_t](#)
[Vertex](#) id type.

Functions

- template<typename I, typename J >
void [display_branch](#) (const [Image](#)< J > &ima_, [tree_node](#)< I > *tree_node)
Display an arborescence from [tree_node](#).
- template<typename I, typename J >
void [display_tree](#) (const [Image](#)< J > &ima_, [tree](#)< I > &tree)
Display a tree.
- template<typename I >
I::psite [lemmings](#) (const [Image](#)< I > &ima, const typename I::psite &pt, const typename I::psite::delta &dpt,
const typename I::value &val)
Launch a lemmings on an image.
- template<typename I >
[greater_point](#)< I > [make_greater_point](#) (const [Image](#)< I > &ima)
Helper to build a [mln::util::greater_point](#).
- template<typename I >
[greater_psite](#)< I > [make_greater_psite](#) (const [Image](#)< I > &ima)
Helper to build a [mln::util::greater_psite](#).
- template<typename G >
bool [operator](#)< (const [vertex](#)< G > &lhs, const [vertex](#)< G > &rhs)
Less operator. Test whether $lhs.id() < rhs.id()$.
- template<typename G >
std::ostream & [operator](#)<< (std::ostream &ostr, const [vertex](#)< G > &v)
Push the vertex v in the output stream $ostr$.
- template<typename T >
std::ostream & [operator](#)<< (std::ostream &ostr, const [array](#)< T > &a)
Operator<<.
- template<typename G >
bool [operator](#)== (const [vertex](#)< G > &v1, const [vertex](#)< G > &v2)

Equality operator.

- `template<typename T >`
`bool operator== (const array< T > &lhs, const array< T > &rhs)`

Operator==.

- `template<typename T >`
`bool ord_strict (const T &lhs, const T &rhs)`

Routine to test if lhs is strictly "less-than" rhs.

- `template<typename T >`
`bool ord_weak (const T &lhs, const T &rhs)`

Routine to test if lhs is "less-than or equal-to" rhs.

- `template<typename T , typename I >`
`void tree_fast_to_image (tree_fast< T > &tree, Image< I > &output_)`

- `template<typename T >`
`tree_fast< T > tree_to_fast (tree< T > &input)`

Facade.

- `template<typename T , typename I >`
`void tree_to_image (tree< T > &tree, Image< I > &output_)`

Convert a tree into an image.

9.145.1 Detailed Description

Namespace of tools using for more complex algorithm. Forward declaration.

9.145.2 Typedef Documentation

9.145.2.1 typedef object_id<vertex_tag, unsigned> mln::util::vertex_id_t

Vertex id type.

Definition at line 43 of file graph_ids.hh.

9.145.3 Function Documentation

9.145.3.1 template<typename I , typename J > void mln::util::display_branch (const Image< J > & ima_, tree_node< I > * tree_node) [inline]

Display an arborescence from `tree_node`.

Parameters

in	<code>ima_</code>	The domain of output image.
in	<code>tree_node</code>	The root <code>tree_node</code> to display.

Definition at line 210 of file tree_to_image.hh.

References `mln::data::fill()`.

9.145.3.2 template<typename I , typename J > void mln::util::display_tree (const Image< J > & ima_, tree< I > & tree) [inline]

Display a tree.

Parameters

<code>in</code>	<code>ima_</code>	The domain of output image.
<code>in</code>	<code>tree</code>	The tree to display.

Definition at line 192 of file `tree_to_image.hh`.

References `mln::util::tree< T >::root()`.

9.145.3.3 `template<typename I > I::psite mln::util::lemmings (const Image< I > & ima, const typename I::psite & pt, const typename I::psite::delta & dpt, const typename I::value & val)`

Launch a lemmings on an image.

A lemmings is the point `pt` that you put on an image `ima` . This point will move through the image using the delta-point `dpt` while consider his value on the given image.

Returns

The first point that is not in the domain `domain` or which value on the given image is different to the value `val`.

Precondition

The domain `domain` must be contained in the domain of `ima`.

Definition at line 104 of file `lemmings.hh`.

9.145.3.4 `template<typename I > greater_point< I > mln::util::make_greater_point (const Image< I > & ima)`

Helper to build a [mln::util::greater_point](#).

Definition at line 82 of file `greater_point.hh`.

9.145.3.5 `template<typename I > greater_psite< I > mln::util::make_greater_psite (const Image< I > & ima)`

Helper to build a [mln::util::greater_psite](#).

Definition at line 82 of file `greater_psite.hh`.

9.145.3.6 `template<typename G > bool mln::util::operator< (const vertex< G > & lhs, const vertex< G > & rhs)`
`[inline]`

Less operator. Test whether `lhs.id() < rhs.id()`.

Definition at line 390 of file `vertex.hh`.

9.145.3.7 `template<typename G > std::ostream & mln::util::operator<< (std::ostream & ostr, const vertex< G > & v)`
`[inline]`

Push the vertex `v` in the output stream `ostr`.

Definition at line 372 of file `vertex.hh`.

9.145.3.8 `template<typename T > std::ostream & mln::util::operator<< (std::ostream & ostr, const array< T > & a)`

Operator<<.

Definition at line 796 of file `util/array.hh`.

References `mln::util::array< T >::nelements()`.

9.145.3.9 `template<typename G> bool mln::util::operator==(const vertex< G> & v1, const vertex< G> & v2)`
`[inline]`

Equality operator.

Test whether two vertices have the same id.

Definition at line 380 of file vertex.hh.

References `mln::util::vertex< G>::graph()`, and `mln::util::vertex< G>::id()`.

9.145.3.10 `template<typename T> bool mln::util::operator==(const array< T> & lhs, const array< T> & rhs)`

Operator==.

Definition at line 815 of file util/array.hh.

References `mln::util::array< T>::std_vector()`.

9.145.3.11 `template<typename T> bool mln::util::ord_strict (const T & lhs, const T & rhs)` `[inline]`

Routine to test if *lhs* is strictly "less-than" *rhs*.

Definition at line 90 of file util/ord.hh.

Referenced by `mln::util::ord_pair< T>::change_both()`, `mln::util::ord_pair< T>::change_first()`, and `mln::util::ord_pair< T>::change_second()`.

9.145.3.12 `template<typename T> bool mln::util::ord_weak (const T & lhs, const T & rhs)` `[inline]`

Routine to test if *lhs* is "less-than or equal-to" *rhs*.

Definition at line 101 of file util/ord.hh.

Referenced by `mln::util::ord_pair< T>::change_both()`, `mln::util::ord_pair< T>::change_first()`, `mln::util::ord_pair< T>::change_second()`, and `mln::box< P>::is_valid()`.

9.145.3.13 `template<typename T, typename I> void mln::util::tree_fast_to_image (tree_fast< T> & tree, Image< I> & output_)` `[inline]`

Convert a `tree_fast` into an image.

```
\param[in] tree The tree to convert.
\param[out] output_ The image containing tree informations.
```

Definition at line 99 of file tree_fast_to_image.hh.

9.145.3.14 `template<typename T> tree_fast< T> mln::util::tree_to_fast (tree< T> & input)` `[inline]`

Facade.

Convert a tree into an `tree_fast`.

```
\param[in] input The tree to convert.

\return The tree_fast containing tree informations.
```

Definition at line 90 of file tree_to_fast.hh.

References `mln::util::tree< T>::root()`.

9.145.3.15 `template<typename T , typename I > void mln::util::tree_to_image (tree< T > & tree, Image< I > & output_)`
`[inline]`

Convert a tree into an image.

Parameters

<code>in</code>	<code>tree</code>	The tree to convert.
<code>out</code>	<code>output_</code>	The image containing tree information.

Definition at line 178 of file `tree_to_image.hh`.

9.146 mln::util::impl Namespace Reference

Implementation namespace of util namespace.

9.146.1 Detailed Description

Implementation namespace of util namespace.

9.147 mln::value Namespace Reference

Namespace of materials related to pixel value types.

Namespaces

- namespace [impl](#)
Implementation namespace of value namespace.

Classes

- class [float01](#)
Class for floating values restricted to the interval [0..1] and discretized with n bits.
- struct [float01_f](#)
Class for floating values restricted to the interval [0..1].
- struct [graylevel](#)
General gray-level class on n bits.
- struct [graylevel_f](#)
General gray-level class on n bits.
- struct [int_s](#)
Signed integer value class.
- struct [int_u](#)
Unsigned integer value class.
- struct [int_u_sat](#)
Unsigned integer value class with saturation behavior.
- struct [Integer](#)
Concept of integer.
- struct [Integer< void >](#)
Category flag type.

- struct [label](#)
Label value class.
- struct [lut_vec](#)
Class that defines FIXME.
- class [proxy](#)
Generic proxy class for an image pixel value.
- struct [rgb](#)
Color class for red-green-blue where every component is n-bit encoded.
- struct [set](#)
Class that defines the set of values of type T.
- class [sign](#)
The sign class represents the value type composed by the set (-1, 0, 1) sign value type is a subset of the int value type.
- struct [stack_image](#)
Stack image class.
- struct [super_value](#)< [sign](#) >
Specializations:
- struct [value_array](#)
Generic array class over indexed by a value set with type T.

Typedefs

- typedef [float01](#)< 16 > [float01_16](#)
Alias for 16 bit [float01](#).
- typedef [float01](#)< 8 > [float01_8](#)
Alias for 8 bit [float01](#).
- typedef [graylevel](#)< 16 > [gl16](#)
Alias for 16 bit graylevel.
- typedef [graylevel](#)< 8 > [gl8](#)
Alias for 8 bit graylevel.
- typedef [graylevel_f](#) [glf](#)
Alias for graylevels encoded by float.
- typedef [int_s](#)< 16 > [int_s16](#)
Alias for signed 16-bit integers.
- typedef [int_s](#)< 32 > [int_s32](#)
Alias for signed 32-bit integers.
- typedef [int_s](#)< 8 > [int_s8](#)
Alias for signed 8-bit integers.
- typedef [int_u](#)< 12 > [int_u12](#)
Alias for unsigned 12-bit integers.
- typedef [int_u](#)< 16 > [int_u16](#)
Alias for unsigned 16-bit integers.
- typedef [mln::value::int_u](#)< 32 > [int_u32](#)
Alias for unsigned 32-bit integers.
- typedef [mln::value::int_u](#)< 8 > [int_u8](#)
Alias for unsigned 8-bit integers.
- typedef [label](#)< 16 > [label_16](#)
Alias for 16-bit integers.
- typedef [label](#)< 32 > [label_32](#)
Alias for 32-bit integers.
- typedef [mln::value::label](#)< 8 > [label_8](#)

Alias for 8-bit labels.

- typedef `rgb< 16 > rgb16`

Color class for red-green-blue where every component is 16-bit encoded.

- typedef `rgb< 8 > rgb8`

Color class for red-green-blue where every component is 8-bit encoded.

Functions

- template<typename Dest , typename Src >
Dest `cast` (const Src &src)
Cast a value `src` from type `Src` to type `Dest`.
- template<typename V >
internal::equiv_< V >::ret `equiv` (const mln::Value< V > &v)
Access to the equivalent value.
- template<typename H , typename S , typename L >
hsl_< H, S, L > `operator+` (const hsl_< H, S, L > &lhs, const hsl_< H, S, L > &rhs)
Addition.
- template<unsigned n>
`rgb< n >::interop operator+` (const `rgb< n >` &lhs, const `rgb< n >` &rhs)
Addition.
- template<typename T >
std::ostream & `operator<<` (std::ostream &ostr, const scalar_< T > &s)
Print a scalar `s` in an output stream `ostr`.
- std::ostream & `operator<<` (std::ostream &ostr, const `sign` &i)
Print an signed integer `i` into the output stream `ostr`.
- template<unsigned n>
std::ostream & `operator<<` (std::ostream &ostr, const `int_u_sat`< n > &i)
Print a saturated unsigned integer `i` into the output stream `ostr`.
- template<unsigned n>
std::ostream & `operator<<` (std::ostream &ostr, const float01_< n > &f)
Op<<.
- template<unsigned n>
std::ostream & `operator<<` (std::ostream &ostr, const `int_s`< n > &i)
Print an signed integer `i` into the output stream `ostr`.
- template<unsigned n>
std::ostream & `operator<<` (std::ostream &ostr, const `label`< n > &l)
Print a label `l` 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<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 `rgb`< n > &c)
Print an rgb `c` into the output stream `ostr`.
- bool `operator==` (const `sign` &lhs, const `sign` &rhs)

Comparison operator.

- template<typename V >
V **other** (const V &val)
Give an other value than val.
- template<typename H , typename S , typename L >
hsl_< H, S, L > **operator-** (const hsl_< H, S, L > &lhs, const hsl_< H, S, L > &rhs)
Subtraction.
- template<typename H , typename S , typename L , typename S2 >
hsl_< H, S, L > **operator*** (const hsl_< H, S, L > &lhs, const mln::value::scalar_< S2 > &s)
Product.
- template<typename H , typename S , typename L , typename S2 >
hsl_< H, S, L > **operator/** (const hsl_< H, S, L > &lhs, const mln::value::scalar_< S2 > &s)
Division.
- template<typename H , typename S , typename L >
bool **operator==** (const hsl_< H, S, L > &lhs, const hsl_< H, S, L > &rhs)
Comparison.
- template<unsigned n>
rgb< n >::interop **operator-** (const rgb< n > &lhs, const rgb< n > &rhs)
Subtraction.
- template<unsigned n, typename S >
rgb< n >::interop **operator*** (const rgb< n > &lhs, const mln::value::scalar_< S > &s)
Product.
- template<unsigned n, typename S >
rgb< n >::interop **operator/** (const rgb< n > &lhs, const mln::value::scalar_< S > &s)
Division.
- template<typename I >
stack_image< 2, const I > **stack** (const **Image**< I > &ima1, const **Image**< I > &ima2)
Shortcut to build a stack with two images.

9.147.1 Detailed Description

Namespace of materials related to pixel value types.

9.147.2 Typedef Documentation

9.147.2.1 typedef float01_<16> mln::value::float01_16

Alias for 16 bit **float01**.

Definition at line 45 of file float01_16.hh.

9.147.2.2 typedef float01_<8> mln::value::float01_8

Alias for 8 bit **float01**.

Definition at line 45 of file float01_8.hh.

9.147.2.3 typedef graylevel<16> mln::value::gl16

Alias for 16 bit graylevel.

Definition at line 45 of file gl16.hh.

9.147.2.4 typedef graylevel<8> mln::value::gl8

Alias for 8 bit graylevel.

Definition at line 45 of file gl8.hh.

9.147.2.5 typedef graylevel_f mln::value::glf

Alias for graylevels encoded by float.

Definition at line 44 of file glf.hh.

9.147.2.6 typedef int_s<16> mln::value::int_s16

Alias for signed 16-bit integers.

Definition at line 45 of file int_s16.hh.

9.147.2.7 typedef int_s<32> mln::value::int_s32

Alias for signed 32-bit integers.

Definition at line 45 of file int_s32.hh.

9.147.2.8 typedef int_s<8> mln::value::int_s8

Alias for signed 8-bit integers.

Definition at line 45 of file int_s8.hh.

9.147.2.9 typedef int_u<12> mln::value::int_u12

Alias for unsigned 12-bit integers.

Definition at line 45 of file int_u12.hh.

9.147.2.10 typedef int_u<16> mln::value::int_u16

Alias for unsigned 16-bit integers.

Definition at line 45 of file int_u16.hh.

9.147.2.11 typedef mln::value::int_u<32> mln::value::int_u32

Alias for unsigned 32-bit integers.

Definition at line 45 of file int_u32.hh.

9.147.2.12 typedef mln::value::int_u<8> mln::value::int_u8

Alias for unsigned 8-bit integers.

Definition at line 44 of file int_u8.hh.

9.147.2.13 typedef label<16> mln::value::label_16

Alias for 16-bit integers.

Definition at line 44 of file label_16.hh.

9.147.2.14 typedef label<32> mln::value::label_32

Alias for 32-bit integers.

Definition at line 44 of file label_32.hh.

9.147.2.15 typedef mln::value::label<8> mln::value::label_8

Alias for 8-bit labels.

Definition at line 44 of file label_8.hh.

9.147.2.16 typedef rgb<16> mln::value::rgb16

Color class for red-green-blue where every component is 16-bit encoded.

Definition at line 45 of file rgb16.hh.

9.147.2.17 typedef rgb<8> mln::value::rgb8

Color class for red-green-blue where every component is 8-bit encoded.

Definition at line 45 of file rgb8.hh.

9.147.3 Function Documentation**9.147.3.1 template<typename Dest , typename Src > Dest mln::value::cast (const Src & *src*) [inline]**

Cast a value *src* from type *Src* to type *Dest*.

Definition at line 85 of file value/cast.hh.

9.147.3.2 template<typename V > internal::equiv_< V >::ret mln::value::equiv (const mln::Value< V > & *v*) [inline]

Access to the equivalent value.

Definition at line 153 of file equiv.hh.

Referenced by mln::labeling::superpose().

9.147.3.3 template<typename H , typename S , typename L , typename S2 > hsl_< H, S, L > mln::value::operator* (const hsl_< H, S, L > & *lhs*, const mln::value::scalar_< S2 > & *s*)

Product.

Definition at line 357 of file hsl.hh.

9.147.3.4 `template<unsigned n, typename S> rgb<n>::interop mln::value::operator* (const rgb<n> & lhs, const mln::value::scalar_<S> & s) [inline]`

Product.

Definition at line 717 of file value/rgb.hh.

9.147.3.5 `template<typename H, typename S, typename L> hsl_<H, S, L> mln::value::operator+ (const hsl_<H, S, L> & lhs, const hsl_<H, S, L> & rhs)`

Addition.

{

Definition at line 337 of file hsl.hh.

9.147.3.6 `template<unsigned n> rgb<n>::interop mln::value::operator+ (const rgb<n> & lhs, const rgb<n> & rhs) [inline]`

Addition.

{

Definition at line 663 of file value/rgb.hh.

9.147.3.7 `template<typename H, typename S, typename L> hsl_<H, S, L> mln::value::operator- (const hsl_<H, S, L> & lhs, const hsl_<H, S, L> & rhs)`

Subtraction.

Definition at line 347 of file hsl.hh.

9.147.3.8 `template<unsigned n> rgb<n>::interop mln::value::operator- (const rgb<n> & lhs, const rgb<n> & rhs) [inline]`

Subtraction.

Definition at line 690 of file value/rgb.hh.

9.147.3.9 `template<typename H, typename S, typename L, typename S2> hsl_<H, S, L> mln::value::operator/ (const hsl_<H, S, L> & lhs, const mln::value::scalar_<S2> & s)`

Division.

Definition at line 367 of file hsl.hh.

9.147.3.10 `template<unsigned n, typename S> rgb<n>::interop mln::value::operator/ (const rgb<n> & lhs, const mln::value::scalar_<S> & s) [inline]`

Division.

Definition at line 735 of file value/rgb.hh.

9.147.3.11 `template<typename T> std::ostream & mln::value::operator<< (std::ostream & ostr, const scalar_< T> & s)`
`[inline]`

Print a scalar *s* in an output stream *ostr*.

Definition at line 130 of file scalar.hh.

9.147.3.12 `std::ostream & mln::value::operator<< (std::ostream & ostr, const sign & i)` `[inline]`

Print an signed integer *i* into the output stream *ostr*.

Parameters

<i>in</i> , <i>out</i>	<i>ostr</i>	An output stream.
<i>in</i>	<i>i</i>	An sign value

Returns

The modified output stream *ostr*.

Definition at line 184 of file value/sign.hh.

References `mln::debug::format()`.

9.147.3.13 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr, const int_u_sat< n> & i)`
`[inline]`

Print a saturated unsigned integer *i* into the output stream *ostr*.

Parameters

<i>in</i> , <i>out</i>	<i>ostr</i>	An output stream.
<i>in</i>	<i>i</i>	A saturated unsigned integer.

Returns

The modified output stream *ostr*.

Definition at line 220 of file int_u_sat.hh.

References `mln::debug::format()`.

9.147.3.14 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr, const float01_< n> & f)`
`[inline]`

`Op<<`.

Definition at line 253 of file float01_.hh.

9.147.3.15 `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

<i>in</i> , <i>out</i>	<i>ostr</i>	An output stream.
<i>in</i>	<i>i</i>	An signed integer.

Returns

The modified output stream `ostr`.

Definition at line 260 of file `int_s.hh`.

References `mln::debug::format()`.

9.147.3.16 `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

<code>in, out</code>	<code>ostr</code>	An output stream.
<code>in</code>	<code>l</code>	A label.

Returns

The modified output stream `ostr`.

Definition at line 353 of file `label.hh`.

References `mln::debug::format()`.

9.147.3.17 `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

<code>in, out</code>	<code>ostr</code>	An output stream.
<code>in</code>	<code>i</code>	An unsigned integer.

Returns

The modified output stream `ostr`.

Definition at line 357 of file `int_u.hh`.

References `mln::debug::format()`.

9.147.3.18 `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

<code>in, out</code>	<code>ostr</code>	An output stream.
<code>in</code>	<code>c</code>	An rgb.

Returns

The modified output stream `ostr`.

Definition at line 326 of file `hsl.hh`.

References `mln::debug::format()`.

9.147.3.19 `std::ostream & mln::value::operator<< (std::ostream & ostr, const graylevel_f & g)` `[inline]`

`Op<<`.

Definition at line 459 of file `graylevel_f.hh`.

References `mln::value::graylevel_f::value()`.

9.147.3.20 `template<unsigned n> std::ostream & mln::value::operator<< (std::ostream & ostr, const graylevel<n> & g)`
`[inline]`

`Op<<`.

Definition at line 591 of file `graylevel.hh`.

9.147.3.21 `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

<code>in, out</code>	<code>ostr</code>	An output stream.
<code>in</code>	<code>c</code>	An <code>rgb</code> .

Returns

The modified output stream `ostr`.

Definition at line 743 of file `value/rgb.hh`.

References `mln::debug::format()`.

9.147.3.22 `bool mln::value::operator== (const sign & lhs, const sign & rhs)` `[inline]`

Comparaison operator.

Definition at line 190 of file `value/sign.hh`.

9.147.3.23 `template<typename H, typename S, typename L> bool mln::value::operator== (const hsl<H, S, L> & lhs,`
`const hsl<H, S, L> & rhs)`

Comparison.

Definition at line 376 of file `hsl.hh`.

9.147.3.24 `template<typename V> V mln::value::other (const V & val)` `[inline]`

Give an other value than `val`.

Definition at line 115 of file `other.hh`.

```
9.147.3.25  template<typename I > stack_image< 2, const I > mln::value::stack ( const Image< I > & ima1, const Image<
              I > & ima2 )  [inline]
```

Shortcut to build a stack with two images.

Definition at line 306 of file stack.hh.

9.148 mln::value::impl Namespace Reference

Implementation namespace of value namespace.

9.148.1 Detailed Description

Implementation namespace of value namespace.

9.149 mln::win Namespace Reference

Namespace of image processing routines related to win.

Classes

- struct [backdiag2d](#)
Diagonal line window defined on the 2D square grid.
- struct [ball](#)
Generic ball window defined on a given grid.
- struct [cube3d](#)
Cube window defined on the 3D grid.
- struct [cuboid3d](#)
Cuboid defined on the 3-D square grid.
- struct [diag2d](#)
Diagonal line window defined on the 2D square grid.
- struct [line](#)
Generic line window defined on a given grid in the given dimension.
- class [multiple](#)
Multiple window.
- class [multiple_size](#)
Definition of a multiple-size window.
- struct [octagon2d](#)
Octagon window defined on the 2D square grid.
- struct [rectangle2d](#)
Rectangular window defined on the 2D square grid.

Typedefs

- typedef [ball](#)< grid::square,
def::coord > [disk2d](#)
2D disk window; precisely, ball-shaped window defined on the 2D square grid.
- typedef [line](#)< grid::square,
1, def::coord > [hline2d](#)

Horizontal line window defined on the 2D square grid.

- typedef `line`< grid::tick,
0, `def::coord` > `segment1d`

Segment window defined on the 1D grid.

- typedef `line`< grid::cube,
0, `def::coord` > `sline3d`

Depth line window defined on the 3D cubic grid.

- typedef `ball`< grid::cube,
`def::coord` > `sphere3d`

3D sphere window; precisely, ball-shaped window defined on the 3D cubic grid.

- typedef `line`< grid::square,
0, `def::coord` > `vline2d`

Vertical line window defined on the 2D square grid.

Functions

- template<typename N1 , typename N2 >
`neighb`< typename
N1::window::regular > `diff` (const `Neighborhood`< N1 > &nbh1, const `Neighborhood`< N2 > &nbh2)

Set difference between a couple of neighborhoods nbh1 and nbh2.

- template<typename 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 `Window`< W > &win)

Give the symmetrical window of win.

- template<typename W >
W `sym` (const `Weighted_Window`< W > &w_win)

Give the symmetrical weighted window of w_win.

9.149.1 Detailed Description

Namespace of image processing routines related to win.

9.149.2 Function Documentation

- 9.149.2.1 template<typename N1 , typename N2 > `neighb`< typename N1::window::regular > mln::win::diff (const `Neighborhood`< N1 > & nbh1, const `Neighborhood`< N2 > & nbh2)

Set difference between a couple of neighborhoods nbh1 and nbh2.

Definition at line 132 of file win/diff.hh.

Referenced by mln::operator-().

- 9.149.2.2 template<typename W > mln::win::mln_regular (W) const [inline]

Shift a window win with a delta-point dp.

9.149.2.3 `template<typename W1 , typename W2 > mln::win::mln_regular (W1) const` `[inline]`

Set difference between a couple of windows `win1` and `win2`.

9.149.2.4 `template<typename W > W mln::win::sym (const Window< W > & win)` `[inline]`

Give the symmetrical window of `win`.

Definition at line 59 of file `sym.hh`.

Referenced by `mln::c18()`, `mln::c26()`, `mln::c4_3d()`, `mln::c6()`, `mln::morpho::hit_or_miss_background_opening()`, `mln::morpho::hit_or_miss_opening()`, `mln::morpho::opening::approx::structural()`, and `mln::morpho::closing::approx::structural()`.

9.149.2.5 `template<typename W > W mln::win::sym (const Weighted_Window< W > & w_win)` `[inline]`

Give the symmetrical weighted window of `w_win`.

Definition at line 71 of file `sym.hh`.

Chapter 10

Class Documentation

10.1 `array< T >` Class Template Reference

Forward declaration.

```
#include <array.hh>
```

10.1.1 Detailed Description

```
template<typename T>class array< T >
```

Forward declaration.

Definition at line 56 of file util/array.hh.

10.2 `C_Function< E >` Struct Template Reference

Concept-like.

```
#include <c.hh>
```

10.2.1 Detailed Description

```
template<typename E>struct C_Function< E >
```

Concept-like.

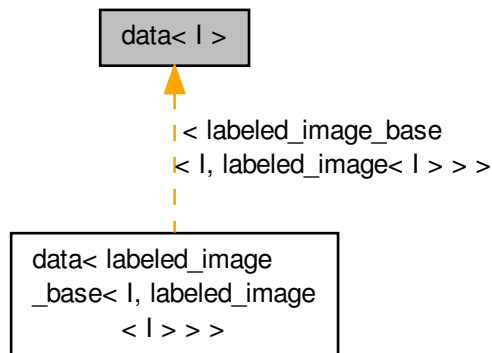
Definition at line 46 of file c.hh.

10.3 `data< I >` Struct Template Reference

Class of image internal data.

```
#include <data.hh>
```

Inheritance diagram for `data< I >`:



10.3.1 Detailed Description

```
template<typename I>struct data< I >
```

Class of image internal data.

It has to be specialized for every image type.

Definition at line 47 of file `core/internal/data.hh`.

10.4 `depth1st_piter< T >` Class Template Reference

Depth1st tree traversal iterator.

```
#include <data.hh>
```

10.4.1 Detailed Description

```
template<typename T>class depth1st_piter< T >
```

Depth1st tree traversal iterator.

Definition at line 96 of file `morpho/tree/data.hh`.

10.5 `dn_leaf_piter< T >` Struct Template Reference

Iterate on tree's leaves in the same way of [dn_node_piter](#).

```
#include <data.hh>
```

10.5.1 Detailed Description


```
template<typename T>struct dn_leaf_piter< T >
```

Iterate on tree's leaves in the same way of [dn_node_piter](#).

Definition at line 93 of file `morpho/tree/data.hh`.

10.6 `dn_node_piter< T >` Struct Template Reference

Iterate on tree's nodes (component representants) from leaves to roots.

```
#include <data.hh>
```

10.6.1 Detailed Description

```
template<typename T>struct dn_node_piter< T >
```

Iterate on tree's nodes (component representants) from leaves to roots.

Definition at line 87 of file `morpho/tree/data.hh`.

10.7 `dn_site_piter< T >` Struct Template Reference

Iterate on tree's sites from roots to leaves.

```
#include <data.hh>
```

10.7.1 Detailed Description

```
template<typename T>struct dn_site_piter< T >
```

Iterate on tree's sites from roots to leaves.

Definition at line 81 of file `morpho/tree/data.hh`.

10.8 `face_data< N, D >` Class Template Reference

Data (adjacent faces) associated to a N-face of a D-complex.

```
#include <face_data.hh>
```

10.8.1 Detailed Description

```
template<unsigned N, unsigned D>class face_data< N, D >
```

Data (adjacent faces) associated to a N-face of a D-complex.

Definition at line 75 of file `face_data.hh`.

10.9 `faces_set_mixin< N, D >` Struct Template Reference

Recursive mixins of set of faces.

```
#include <complex.hh>
```

10.9.1 Detailed Description

```
template<unsigned N, unsigned D>struct faces_set_mixin< N, D >
```

Recursive mixins of set of faces.

Definition at line 309 of file complex.hh.

10.10 `graph_elt_mixed_window< G, S, S2 >` Class Template Reference

Forward declaration.

```
#include <graph_elt_mixed_window.hh>
```

10.10.1 Detailed Description

```
template<typename G, typename S, typename S2>class graph_elt_mixed_window< G, S, S2 >
```

Forward declaration.

Definition at line 45 of file graph_elt_mixed_window.hh.

10.11 `graph_elt_window< G, S >` Class Template Reference

Forward declaration.

```
#include <graph_elt_window.hh>
```

10.11.1 Detailed Description

```
template<typename G, typename S>class graph_elt_window< G, S >
```

Forward declaration.

Definition at line 44 of file graph_elt_window.hh.

10.12 `graph_elt_window_if< G, S, I >` Class Template Reference

Forward declaration.

```
#include <graph_elt_window_if.hh>
```

10.12.1 Detailed Description

```
template<typename G, typename S, typename I>class graph_elt_window_if< G, S, I >
```

Forward declaration.

Definition at line 47 of file graph_elt_window_if.hh.

10.13 graph_mixed_window_iter_dispatch< G, S, S2 > Struct Template Reference

Default The given site set parameter is not supported yet!

```
#include <graph_elt_mixed_window.hh>
```

10.13.1 Detailed Description

```
template<typename G, typename S, typename S2>struct graph_mixed_window_iter_dispatch< G, S, S2 >
```

Default The given site set parameter is not supported yet!

G is the graph type. S is an image site set from where the center is extracted. S2 is an image site set from where the neighbors are extracted.

Definition at line 74 of file graph_elt_mixed_window.hh.

10.14 graph_window_if_iter_dispatch< G, S > Struct Template Reference

Default The given site set parameter is not supported yet!

```
#include <graph_elt_window_if.hh>
```

10.14.1 Detailed Description

```
template<typename G, typename S>struct graph_window_if_iter_dispatch< G, S >
```

Default The given site set parameter is not supported yet!

Definition at line 65 of file graph_elt_window_if.hh.

10.15 graph_window_iter_dispatch< G, S > Struct Template Reference

Default The given site set parameter is not supported yet!

```
#include <graph_elt_window.hh>
```

10.15.1 Detailed Description

```
template<typename G, typename S>struct graph_window_iter_dispatch< G, S >
```

Default The given site set parameter is not supported yet!

G is the graph type. S is an image site set from where the center is extracted.

Definition at line 71 of file graph_elt_window.hh.

10.16 int_s< n > Struct Template Reference

Fwd decls.

```
#include <int_s.hh>
```

10.16.1 Detailed Description

```
template<unsigned n>struct int_s< n >
```

Fwd decls.

Definition at line 52 of file int_s.hh.

10.17 line_graph< G > Class Template Reference

Fwd declaration.

```
#include <line_graph.hh>
```

10.17.1 Detailed Description

```
template<typename G>class line_graph< G >
```

Fwd declaration.

Definition at line 46 of file line_graph.hh.

10.18 mln::accu::center< P, V > Struct Template Reference

Mass center accumulator.

```
#include <center.hh>
```

Inherits mln::accu::internal::base< V, center< P, V > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- unsigned [nsites](#) () const
Return the number of sites taken in consideration.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- V [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.18.1 Detailed Description

```
template<typename P, typename V = typename P::vec>struct mln::accu::center< P, V >
```

Mass center accumulator.

Template Parameters

<i>P</i>	the type of site.
<i>V</i>	the type of vector to be used as result. The default vector type is the one provided by <i>P</i> .

Definition at line 55 of file center.hh.

10.18.2 Member Function Documentation

10.18.2.1 `template<typename P, typename V> void mln::accu::center<P, V>::init () [inline]`

Manipulators.

Definition at line 116 of file center.hh.

References `mln::literal::zero`.

10.18.2.2 `template<typename P, typename V> bool mln::accu::center<P, V>::is_valid () const [inline]`

Check whether this accu is able to return a result.

Definition at line 160 of file center.hh.

10.18.2.3 `template<typename P, typename V> unsigned mln::accu::center<P, V>::nsites () const [inline]`

Return the number of sites taken in consideration.

Definition at line 168 of file center.hh.

10.18.2.4 `void mln::Accumulator<center<P, V>>::take_as_init (const T & t) [inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.18.2.5 `void mln::Accumulator<center<P, V>>::take_n_times (unsigned n, const T & t) [inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.18.2.6 `template<typename P, typename V> V mln::accu::center<P, V>::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 142 of file center.hh.

10.19 mln::accu::convolve< T1, T2, R > Struct Template Reference

Generic convolution accumulator class.

```
#include <convolve.hh>
```

Inherits `mln::accu::internal::base< R, convolve< T1, T2, R > >`, and `check_t< mln::trait::value_< mln::trait::op-
::times< T1, T2 >::ret >::sum, R >`.

Public Member Functions

- bool `is_valid` () const
Check whether this `accu` is able to return a result.
- void `take_as_init` (const T &t)
Take as initialization the value `t`.
- void `take_n_times` (unsigned n, const T &t)
Take `n` times the value `t`.
- R `to_result` () const
Get the value of the accumulator.
- void `init` ()
Manipulators.

10.19.1 Detailed Description

```
template<typename T1, typename T2, typename R = typename mln::trait::value_< typename mln::trait::op::times< T1 , T2 >::ret
>::sum>struct mln::accu::convolve< T1, T2, R >
```

Generic convolution accumulator class.

Parameters `T1` and `T2` are the type of values to be convolved. Parameter `R` is the result type.

Definition at line 54 of file `accu/convolve.hh`.

10.19.2 Member Function Documentation

10.19.2.1 `template<typename T1, typename T2, typename R > void mln::accu::convolve< T1, T2, R >::init ()`
[inline]

Manipulators.

Definition at line 96 of file `accu/convolve.hh`.

References `mln::literal::zero`.

10.19.2.2 `template<typename T1, typename T2, typename R > bool mln::accu::convolve< T1, T2, R >::is_valid ()`
`const` [inline]

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 137 of file `accu/convolve.hh`.

10.19.2.3 `void mln::Accumulator< convolve< T1, T2, R > >::take_as_init (const T & t)` [inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.19.2.4 `void mln::Accumulator< convolve< T1, T2, R > >::take_n_times (unsigned n, const T & t)`
[inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.19.2.5 `template<typename T1 , typename T2 , typename R > R mln::accu::convolve< T1, T2, R >::to_result () const`
`[inline]`

Get the value of the accumulator.

Definition at line 129 of file `accu/convolve.hh`.

10.20 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, count_adjacent_vertices< F, S > >`.

Public Member Functions

- `bool is_valid () const`
Return whether this accu can return a result.
- `void take_as_init (const T &t)`
Take as initialization the value t.
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t.
- `unsigned to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.
- `void set_value (unsigned c)`
Force the value of the counter to c.

10.20.1 Detailed Description

`template<typename F, typename S> struct mln::accu::count_adjacent_vertices< F, S >`

[Accumulator](#) class counting the number of vertices adjacent to a set of `mln::p_edges_psite` (i.e., a set of edges).

The type to be count is `mln::util::pix< pw::image<F, S> >` where `F` and `S` are the parameters of this class.

This accumulator is used by `mln::closing_area_on_vertices` and `mln::opening_area_on_vertices`.

Definition at line 58 of file `accu/count_adjacent_vertices.hh`.

10.20.2 Member Function Documentation

10.20.2.1 `template<typename F , typename S > void count_adjacent_vertices< F, S >::init ()` `[inline]`

Manipulators.

Definition at line 123 of file `accu/count_adjacent_vertices.hh`.

10.20.2.2 `template<typename F , typename S > bool count_adjacent_vertices< F, S >::is_valid () const` `[inline]`

Return whether this accu can return a result.

Definition at line 177 of file `accu/count_adjacent_vertices.hh`.

10.20.2.3 `template<typename F , typename S > void count_adjacent_vertices< F, S >::set_value (unsigned c)`
`[inline]`

Force the value of the counter to *c*.

Definition at line 159 of file `accu/count_adjacent_vertices.hh`.

10.20.2.4 `void mln::Accumulator< count_adjacent_vertices< F, S > >::take_as_init (const T & t)`
`[inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.20.2.5 `void mln::Accumulator< count_adjacent_vertices< F, S > >::take_n_times (unsigned n, const T & t)`
`[inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.20.2.6 `template<typename F , typename S > unsigned count_adjacent_vertices< F, S >::to_result () const`
`[inline]`

Get the value of the accumulator.

Definition at line 151 of file `accu/count_adjacent_vertices.hh`.

10.21 `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, count_labels< L > >`.

Public Member Functions

- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
*Take as initialization the value *t*.*
- void `take_n_times` (unsigned n, const T &t)
*Take *n* times the value *t*.*
- unsigned `to_result` () const
Get the value of the accumulator.
- void `init` ()
Manipulators.
- void `set_value` (unsigned c)
*Force the value of the counter to *c*.*

10.21.1 Detailed Description

`template<typename L> struct mln::accu::count_labels< L >`

Count the number of different labels in an image.

The parameter *L* is the label type to be count.

Definition at line 52 of file count_labels.hh.

10.21.2 Member Function Documentation

10.21.2.1 `template<typename L > void mln::accu::count_labels< L >::init () [inline]`

Manipulators.

Definition at line 113 of file count_labels.hh.

10.21.2.2 `template<typename L > bool mln::accu::count_labels< L >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 163 of file count_labels.hh.

10.21.2.3 `template<typename L > void mln::accu::count_labels< L >::set_value (unsigned c) [inline]`

Force the value of the counter to *c*.

Definition at line 155 of file count_labels.hh.

10.21.2.4 `void mln::Accumulator< count_labels< L > >::take_as_init (const T & t) [inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.21.2.5 `void mln::Accumulator< count_labels< L > >::take_n.times (unsigned n, const T & t) [inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.21.2.6 `template<typename L > unsigned mln::accu::count_labels< L >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 146 of file count_labels.hh.

10.22 mln::accu::count_value< V > Struct Template Reference

Define an accumulator that counts the occurrence of a given value.

`#include <count_value.hh>`

Inherits `mln::accu::internal::base< unsigned, count_value< V > >`.

Public Member Functions

- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
Take as initialization the value t.
- void `take_n_times` (unsigned n, const T &t)
Take n times the value t.
- unsigned `to_result` () const
Get the value of the accumulator.

- void `init` ()
Manipulators.
- void `set_value` (unsigned c)
Force the value of the counter to c.

10.22.1 Detailed Description

`template<typename V>struct mln::accu::count_value< V >`

Define an accumulator that counts the occurrence of a given value.

Definition at line 72 of file `count_value.hh`.

10.22.2 Member Function Documentation

10.22.2.1 `template<typename V> void count_value< V >::init () [inline]`

Manipulators.

Definition at line 153 of file `count_value.hh`.

10.22.2.2 `template<typename V> bool count_value< V >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 216 of file `count_value.hh`.

10.22.2.3 `template<typename V> void count_value< V >::set_value (unsigned c) [inline]`

Force the value of the counter to c.

Definition at line 208 of file `count_value.hh`.

10.22.2.4 `void mln::Accumulator< count_value< V > >::take_as_init (const T &t) [inherited]`

Take as initialization the value t.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.22.2.5 void mln::Accumulator< count_value< V > >::take_n_times (unsigned *n*, const T & *t*) [inherited]

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.22.2.6 template<typename V> unsigned count_value< V >::to_result () const [inline]

Get the value of the accumulator.

Definition at line 199 of file count_value.hh.

10.23 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 > &, histo< V > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
*Take as initialization the value *t*.*
- void [take_n_times](#) (unsigned *n*, const T &t)
*Take *n* times the value *t*.*
- void [take](#) (const argument &t)
Manipulators.
- const std::vector< unsigned > & [vect](#) () const
Get the value of the accumulator.

10.23.1 Detailed Description

```
template<typename V>struct mln::accu::histo< V >
```

Generic histogram class over a value set with type *V*.

Definition at line 56 of file accu/histo.hh.

10.23.2 Member Function Documentation

10.23.2.1 template<typename V> bool mln::accu::histo< V >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 236 of file accu/histo.hh.

10.23.2.2 `template<typename V> void mln::accu::histo< V>::take (const argument & t) [inline]`

Manipulators.

Definition at line 129 of file `accu/histo.hh`.

10.23.2.3 `void mln::Accumulator< histo< V>>::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.23.2.4 `void mln::Accumulator< histo< V>>::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.23.2.5 `template<typename V> const std::vector< unsigned> & mln::accu::histo< V>::vect () const [inline]`

Get the value of the accumulator.

Definition at line 201 of file `accu/histo.hh`.

10.24 mln::accu::label_used< L> Struct Template Reference

References all the labels used.

`#include <label_used.hh>`

Inherits `mln::accu::internal::base< const fun::i2v::array< bool> &, label_used< L>>`.

Public Member Functions

- void `init` ()
Initialize accumulator attributes.
- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
Take as initialization the value t.
- void `take_n_times` (unsigned n, const T &t)
Take n times the value t.
- const fun::i2v::array< bool> & `to_result` () const
Get the value of the accumulator.
- void `take` (const argument &)
Manipulators.

10.24.1 Detailed Description

`template<typename L> struct mln::accu::label_used< L>`

References all the labels used.

The parameter *L* is the label type.

Definition at line 53 of file label_used.hh.

10.24.2 Member Function Documentation

10.24.2.1 `template<typename L> void mln::accu::label_used<L>::init () [inline]`

Initialize accumulator attributes.

Definition at line 110 of file label_used.hh.

10.24.2.2 `template<typename L> bool mln::accu::label_used<L>::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 150 of file label_used.hh.

10.24.2.3 `template<typename L> void mln::accu::label_used<L>::take (const argument & /) [inline]`

Manipulators.

Definition at line 118 of file label_used.hh.

10.24.2.4 `void mln::Accumulator<label_used<L>>::take_as_init (const T & t) [inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.24.2.5 `void mln::Accumulator<label_used<L>>::take_n_times (unsigned n, const T & t) [inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.24.2.6 `template<typename L> const fun::i2v::array<bool> & mln::accu::label_used<L>::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 142 of file label_used.hh.

10.25 mln::accu::logic::land Struct Reference

"Logical-and" accumulator.

`#include <land.hh>`

Inherits `mln::accu::internal::base<bool, land>`.

Public Member Functions

- `bool is_valid () const`

Check whether this accu is able to return a result.

- void `take_as_init` (const T &t)

Take as initialization the value t.

- void `take_n_times` (unsigned n, const T &t)

Take n times the value t.

- bool `to_result` () const

Get the value of the accumulator.

- void `init` ()

Manipulators.

10.25.1 Detailed Description

"Logical-and" accumulator.

Definition at line 96 of file `accu/logic/land.hh`.

10.25.2 Member Function Documentation

10.25.2.1 void `mln::accu::logic::land::init` () `[inline]`

Manipulators.

Definition at line 136 of file `accu/logic/land.hh`.

10.25.2.2 bool `mln::accu::logic::land::is_valid` () const `[inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 185 of file `accu/logic/land.hh`.

10.25.2.3 void `mln::Accumulator< land >::take_as_init` (const T &t) `[inherited]`

Take as initialization the value t.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.25.2.4 void `mln::Accumulator< land >::take_n_times` (unsigned n, const T &t) `[inherited]`

Take n times the value t.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.25.2.5 bool `mln::accu::logic::land::to_result` () const `[inline]`

Get the value of the accumulator.

Definition at line 178 of file `accu/logic/land.hh`.

10.26 mln::accu::logic::land_basic Struct Reference

"Logical-and" accumulator.

```
#include <land_basic.hh>
```

Inherits mln::accu::internal::base< bool, land_basic >.

Public Member Functions

- bool [can_stop](#) () const
Test if it is worth for this accumulator to take extra data.
- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- bool [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.26.1 Detailed Description

"Logical-and" accumulator.

Conversely to [accu::logic::land](#), this version does not have the 'untake' method but features the 'can_stop' method.

Definition at line 99 of file land_basic.hh.

10.26.2 Member Function Documentation

10.26.2.1 bool mln::accu::logic::land_basic::can_stop () const [inline]

Test if it is worth for this accumulator to take extra data.

If the result is already 'false' (because this accumulator has already taken a 'false' value), can_stop returns true.

Definition at line 181 of file land_basic.hh.

10.26.2.2 void mln::accu::logic::land_basic::init () [inline]

Manipulators.

Definition at line 140 of file land_basic.hh.

10.26.2.3 bool mln::accu::logic::land_basic::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 174 of file land_basic.hh.

10.26.2.4 `void mln::Accumulator< land_basic >::take_as_init (const T & t)` `[inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.26.2.5 `void mln::Accumulator< land_basic >::take_n_times (unsigned n, const T & t)` `[inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.26.2.6 `bool mln::accu::logic::land_basic::to_result () const` `[inline]`

Get the value of the accumulator.

Definition at line 167 of file `land_basic.hh`.

10.27 mln::accu::logic::lor Struct Reference

"Logical-or" accumulator.

```
#include <lor.hh>
```

Inherits `mln::accu::internal::base< bool, lor >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value `t`.
- `void take_n_times (unsigned n, const T &t)`
Take `n` times the value `t`.
- `bool to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.27.1 Detailed Description

"Logical-or" accumulator.

Definition at line 95 of file `accu/logic/lor.hh`.

10.27.2 Member Function Documentation

10.27.2.1 `void mln::accu::logic::lor::init ()` `[inline]`

Manipulators.

Definition at line 134 of file `accu/logic/lor.hh`.

10.27.2.2 `bool mln::accu::logic::lor::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 183 of file accu/logic/lor.hh.

10.27.2.3 `void mln::Accumulator< lor >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.27.2.4 `void mln::Accumulator< lor >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.27.2.5 `bool mln::accu::logic::lor::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 176 of file accu/logic/lor.hh.

10.28 mln::accu::logic::lor_basic Struct Reference

"Logical-or" accumulator class.

```
#include <lor_basic.hh>
```

Inherits `mln::accu::internal::base< bool, lor_basic >`.

Public Member Functions

- `bool can_stop () const`
Test if it is worth for this accumulator to take extra data.
- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value `t`.
- `void take_n_times (unsigned n, const T &t)`
Take `n` times the value `t`.
- `bool to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.28.1 Detailed Description

"Logical-or" accumulator class.

Conversely to `accu::logic::lor`, this version does not have the `'untake'` method but features the `'can_stop'` method.

Definition at line 98 of file `lor_basic.hh`.

10.28.2 Member Function Documentation

10.28.2.1 `bool mln::accu::logic::lor_basic::can_stop () const [inline]`

Test if it is worth for this accumulator to take extra data.

If the result is already 'true' (because this accumulator has already taken a 'true' value), `can_stop` returns true.

Definition at line 180 of file `lor_basic.hh`.

10.28.2.2 `void mln::accu::logic::lor_basic::init () [inline]`

Manipulators.

Definition at line 139 of file `lor_basic.hh`.

10.28.2.3 `bool mln::accu::logic::lor_basic::is_valid () const [inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 173 of file `lor_basic.hh`.

10.28.2.4 `void mln::Accumulator< lor_basic >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.28.2.5 `void mln::Accumulator< lor_basic >::take_n.times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.28.2.6 `bool mln::accu::logic::lor_basic::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 166 of file `lor_basic.hh`.

10.29 `mln::accu::maj_h< T >` Struct Template Reference

Compute the majority value.

```
#include <maj_h.hh>
```

Inherits `mln::accu::internal::base< const T &, maj_h< T > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value `t`.

- void `take_n_times` (unsigned *n*, const *T* &*t*)

*Take *n* times the value *t*.*

- const *T* & `to_result` () const

Get the value of the accumulator.

- void `init` ()

Manipulators.

10.29.1 Detailed Description

`template<typename T> struct mln::accu::maj_h< T >`

Compute the majority value.

It is based on a histogram. The parameter *T* is the type of values.

Definition at line 57 of file `maj_h.hh`.

10.29.2 Member Function Documentation

10.29.2.1 `template<typename T> void mln::accu::maj_h< T >::init () [inline]`

Manipulators.

Definition at line 129 of file `maj_h.hh`.

10.29.2.2 `template<typename T> bool mln::accu::maj_h< T >::is_valid () const [inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 197 of file `maj_h.hh`.

10.29.2.3 `void mln::Accumulator< maj_h< T > >::take_as_init (const T & t) [inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.29.2.4 `void mln::Accumulator< maj_h< T > >::take_n_times (unsigned n, const T & t) [inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.29.2.5 `template<typename T> const T & mln::accu::maj_h< T >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 187 of file `maj_h.hh`.

10.30 mln::accu::math::count< T > Struct Template Reference

Generic counter accumulator.

```
#include <count.hh>
```

Inherits mln::accu::internal::base< unsigned, count< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- unsigned [to_result](#) () const
Get the value of the accumulator.

- void [init](#) ()
Manipulators.
- void [set_value](#) (unsigned c)
Force the value of the counter to c.

10.30.1 Detailed Description

```
template<typename T>struct mln::accu::math::count< T >
```

Generic counter accumulator.

The parameter *T* is the type to be count.

Definition at line 100 of file count.hh.

10.30.2 Member Function Documentation

10.30.2.1 `template<typename T> void count< T >::init () [inline]`

Manipulators.

Definition at line 145 of file count.hh.

10.30.2.2 `template<typename T> bool count< T >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 203 of file count.hh.

10.30.2.3 `template<typename T> void count< T >::set_value (unsigned c) [inline]`

Force the value of the counter to c.

Definition at line 195 of file count.hh.

10.30.2.4 `void mln::Accumulator< count< T > >::take_as_init (const T & t)` [inherited]

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.30.2.5 `void mln::Accumulator< count< T > >::take_n_times (unsigned n, const T & t)` [inherited]

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.30.2.6 `template<typename T> unsigned count< T >::to_result () const` [inline]

Get the value of the accumulator.

Definition at line 187 of file `count.hh`.

10.31 mln::accu::math::inf< T > Struct Template Reference

Generic inf accumulator class.

`#include <inf.hh>`

Inherits `mln::accu::internal::base< const T &, inf< T > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value `t`.
- `void take_n_times (unsigned n, const T &t)`
Take `n` times the value `t`.
- `const T & to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.31.1 Detailed Description

`template<typename T> struct mln::accu::math::inf< T >`

Generic inf accumulator class.

The parameter `T` is the type of values.

Definition at line 56 of file `accu/math/inf.hh`.

10.31.2 Member Function Documentation

10.31.2.1 `template<typename T> void mln::accu::math::inf< T >::init ()` [inline]

Manipulators.

Definition at line 126 of file `accu/math/inf.hh`.

10.31.2.2 `template<typename T> bool mln::accu::math::inf<T>::is_valid () const` `[inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 164 of file `accu/math/inf.hh`.

10.31.2.3 `void mln::Accumulator< inf<T>>::take_as_init (const T & t)` `[inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.31.2.4 `void mln::Accumulator< inf<T>>::take_n.times (unsigned n, const T & t)` `[inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.31.2.5 `template<typename T> const T & mln::accu::math::inf<T>::to_result () const` `[inline]`

Get the value of the accumulator.

Definition at line 156 of file `accu/math/inf.hh`.

10.32 mln::accu::math::sum< T, S > Struct Template Reference

Generic sum accumulator class.

```
#include <sum.hh>
```

Inherits `mln::accu::internal::base< const S &, sum< T, S > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value `t`.
- `void take_n.times (unsigned n, const T &t)`
Take `n` times the value `t`.
- `const S & to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.32.1 Detailed Description

```
template<typename T, typename S = typename mln::value::props< T >::sum> struct mln::accu::math::sum< T, S >
```

Generic sum accumulator class.

Parameter `T` is the type of values that we sum. Parameter `S` is the type to store the value sum; the default type of `S` is the summation type (property) of `T`.

Definition at line 112 of file `accu/math/sum.hh`.

10.32.2 Member Function Documentation

10.32.2.1 `template<typename T, typename S > void sum< T, S >::init () [inline]`

Manipulators.

Definition at line 161 of file `accu/math/sum.hh`.

References `mln::literal::zero`.

10.32.2.2 `template<typename T, typename S > bool sum< T, S >::is_valid () const [inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 222 of file `accu/math/sum.hh`.

10.32.2.3 `void mln::Accumulator< sum< T, S > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.32.2.4 `void mln::Accumulator< sum< T, S > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.32.2.5 `template<typename T, typename S > const S & sum< T, S >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 206 of file `accu/math/sum.hh`.

10.33 mln::accu::math::sup< T > Struct Template Reference

Generic sup accumulator class.

```
#include <sup.hh>
```

Inherits `mln::accu::internal::base< const T &, sup< T > >`.

Public Member Functions

- bool [is_valid](#) () const

- Check whether this accu is able to return a result.*

 - void `take_as_init` (const T &t)
Take as initialization the value t .
 - void `take_n_times` (unsigned n, const T &t)
Take n times the value t .
 - const T & `to_result` () const
Get the value of the accumulator.
- void `init` ()
Manipulators.

10.33.1 Detailed Description

`template<typename T>struct mln::accu::math::sup< T >`

Generic sup accumulator class.

The parameter `T` is the type of values.

Definition at line 56 of file `accu/math/sup.hh`.

10.33.2 Member Function Documentation

10.33.2.1 `template<typename T> void mln::accu::math::sup< T >::init () [inline]`

Manipulators.

Definition at line 128 of file `accu/math/sup.hh`.

10.33.2.2 `template<typename T> bool mln::accu::math::sup< T >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 166 of file `accu/math/sup.hh`.

10.33.2.3 `void mln::Accumulator< sup< T > >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.33.2.4 `void mln::Accumulator< sup< T > >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.33.2.5 `template<typename T> const T & mln::accu::math::sup< T >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 158 of file `accu/math/sup.hh`.

10.34 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, max_site< I > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- I::psite [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.34.1 Detailed Description

```
template<typename I>struct mln::accu::max_site< I >
```

Define an accumulator that computes the first site with the maximum value in an image.

Definition at line 53 of file max_site.hh.

10.34.2 Member Function Documentation

10.34.2.1 `template<typename I > void mln::accu::max_site< I >::init () [inline]`

Manipulators.

Definition at line 114 of file max_site.hh.

10.34.2.2 `template<typename I > bool mln::accu::max_site< I >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 174 of file max_site.hh.

10.34.2.3 `void mln::Accumulator< max_site< I > >::take_as_init (const T &t) [inherited]`

Take as initialization the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.34.2.4 `void mln::Accumulator< max_site< I > >::take_n_times (unsigned n, const T &t) [inherited]`

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.34.2.5 `template<typename I> I::psite mln::accu::max_site<I>::to_result () const` `[inline]`

Get the value of the accumulator.

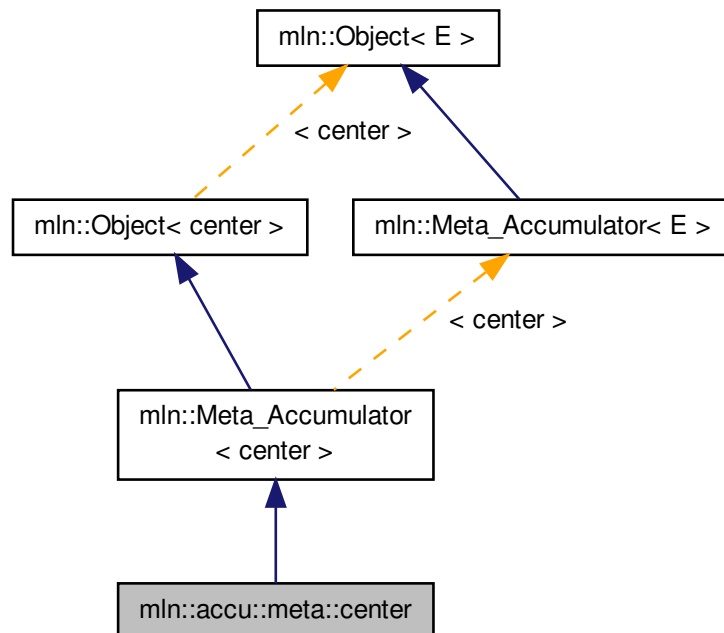
Definition at line 150 of file `max_site.hh`.

10.35 mln::accu::meta::center Struct Reference

Meta accumulator for center.

`#include <center.hh>`

Inheritance diagram for `mln::accu::meta::center`:



10.35.1 Detailed Description

Meta accumulator for center.

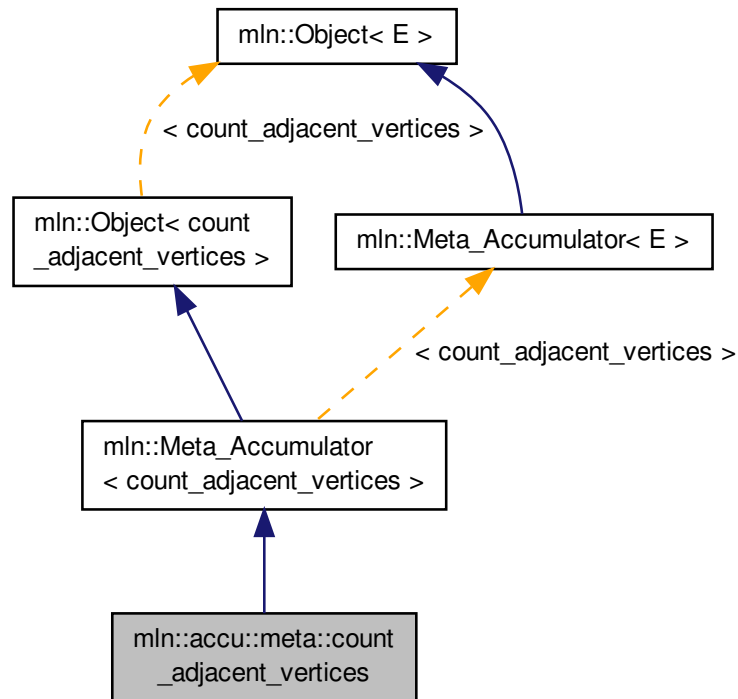
Definition at line 89 of file `center.hh`.

10.36 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:



10.36.1 Detailed Description

Meta accumulator for [count_adjacent_vertices](#).

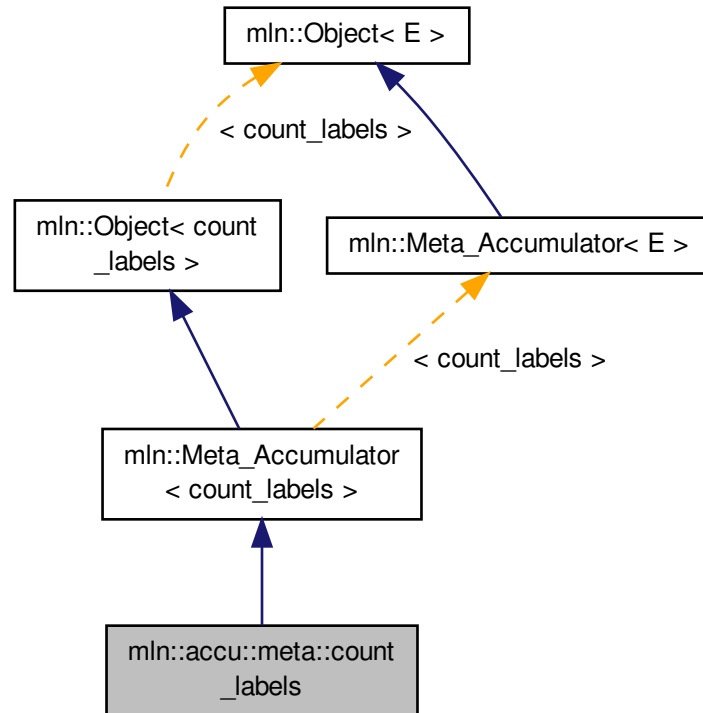
Definition at line 98 of file `accu/count_adjacent_vertices.hh`.

10.37 mln::accu::meta::count_labels Struct Reference

Meta accumulator for [count_labels](#).

```
#include <count_labels.hh>
```

Inheritance diagram for `mln::accu::meta::count_labels`:



10.37.1 Detailed Description

Meta accumulator for [count_labels](#).

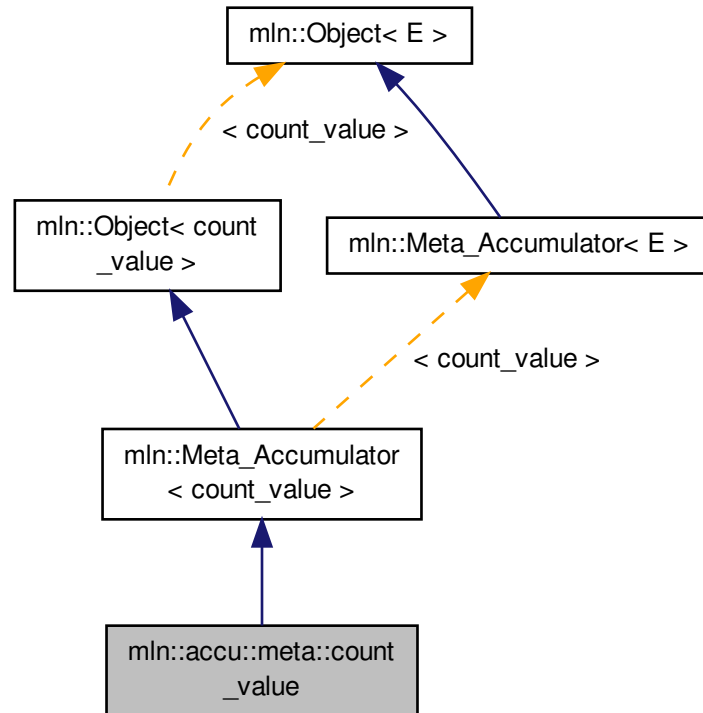
Definition at line 88 of file `count_labels.hh`.

10.38 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:



10.38.1 Detailed Description

FIXME: How to write a meta accumulator with a constructor taking a generic argument? Meta accumulator for [count_value](#).

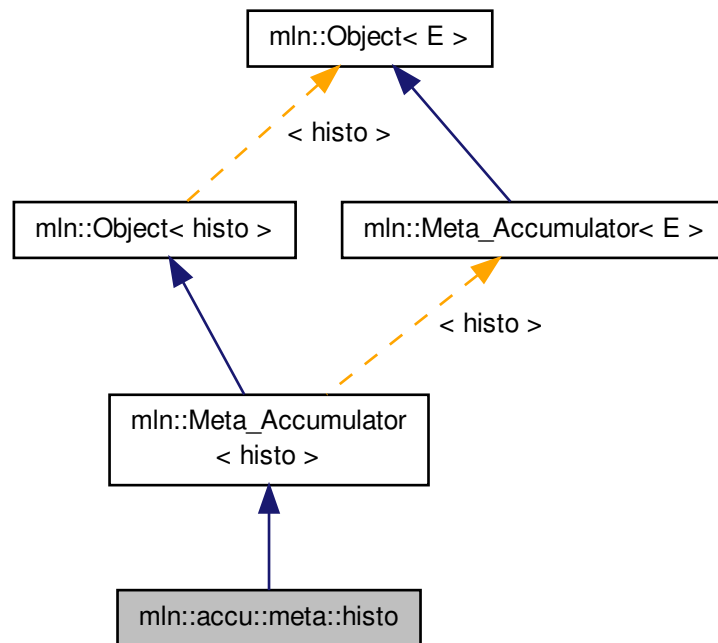
Definition at line 116 of file count_value.hh.

10.39 mln::accu::meta::histo Struct Reference

Meta accumulator for histo.

```
#include <histo.hh>
```

Inheritance diagram for `mln::accu::meta::histo`:



10.39.1 Detailed Description

Meta accumulator for `histo`.

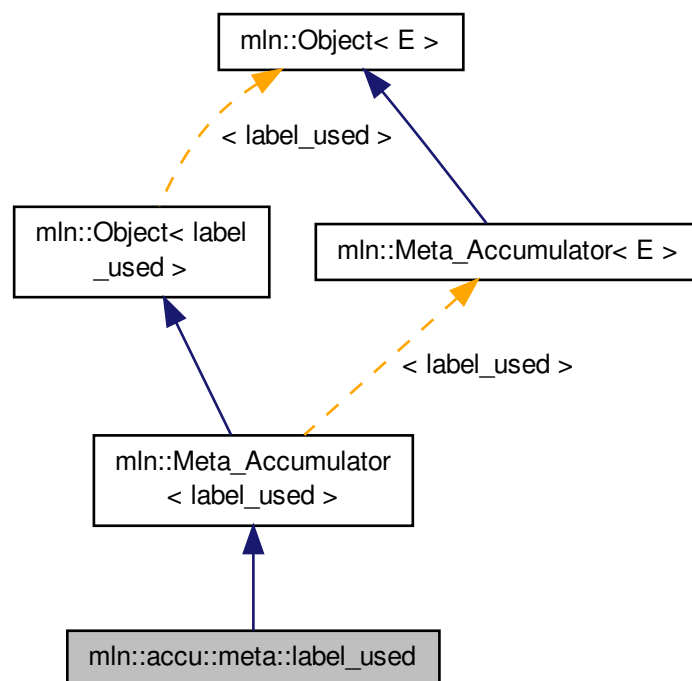
Definition at line 102 of file `accu/histo.hh`.

10.40 `mln::accu::meta::label_used` Struct Reference

Meta accumulator for `label_used`.

```
#include <label_used.hh>
```

Inheritance diagram for mln::accu::meta::label_used:



10.40.1 Detailed Description

Meta accumulator for [label_used](#).

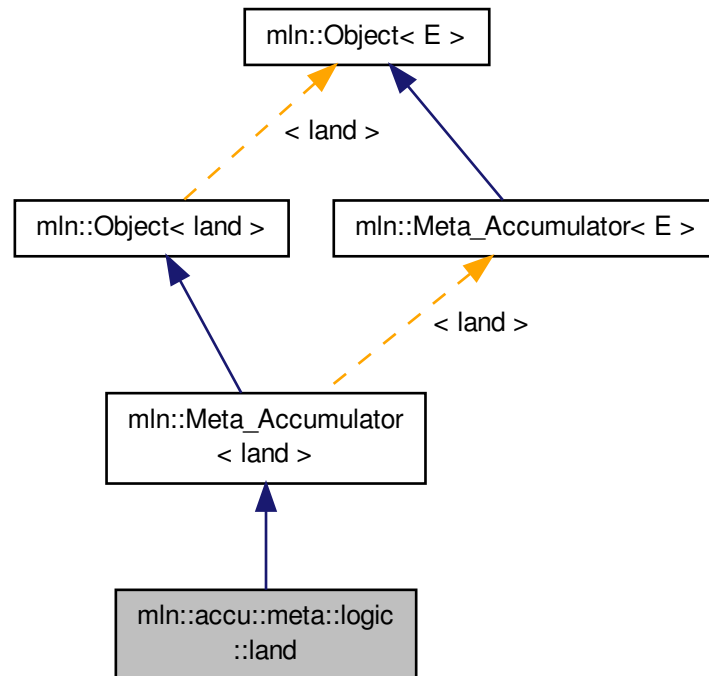
Definition at line 85 of file `label_used.hh`.

10.41 mln::accu::meta::logic::land Struct Reference

Meta accumulator for `land`.

```
#include <land.hh>
```

Inheritance diagram for mln::accu::meta::logic::land:



10.41.1 Detailed Description

Meta accumulator for land.

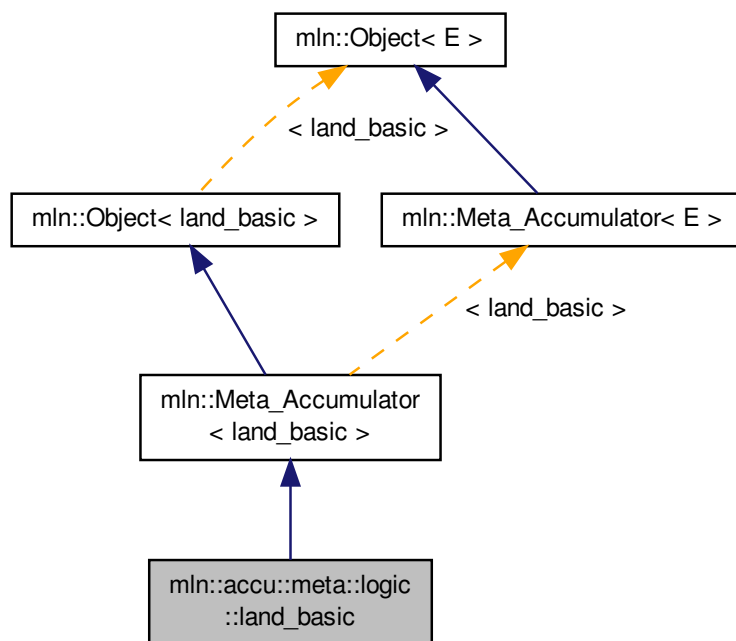
Definition at line 76 of file accu/logic/land.hh.

10.42 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:



10.42.1 Detailed Description

Meta accumulator for [land_basic](#).

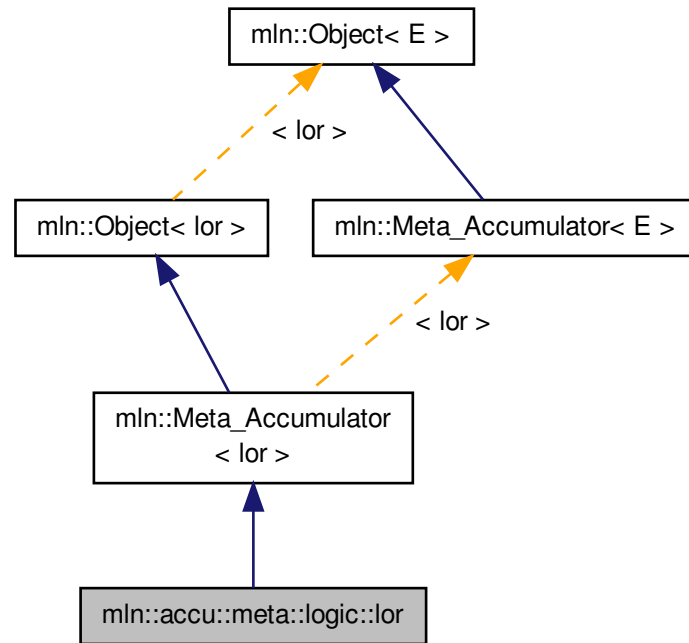
Definition at line 77 of file land_basic.hh.

10.43 mln::accu::meta::logic::lor Struct Reference

Meta accumulator for lor.

```
#include <lor.hh>
```

Inheritance diagram for mln::accu::meta::logic::lor:



10.43.1 Detailed Description

Meta accumulator for lor.

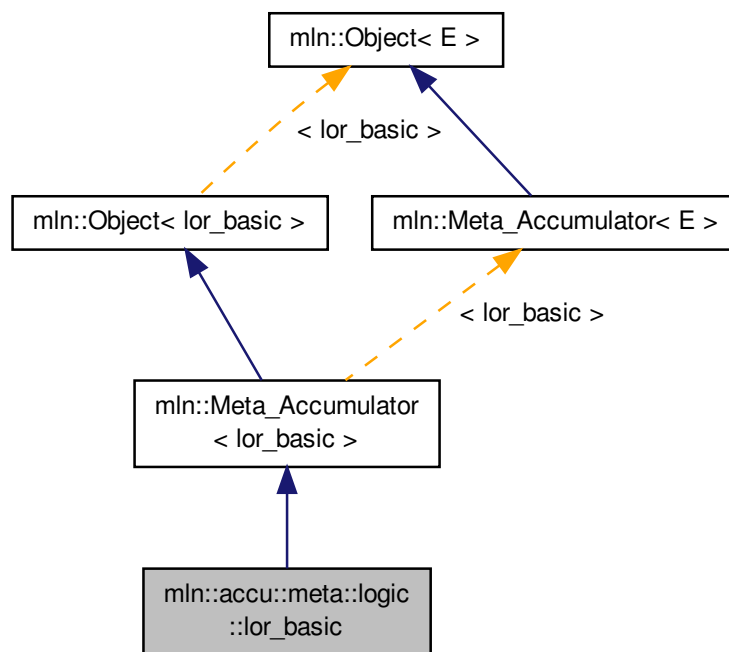
Definition at line 76 of file accu/logic/lor.hh.

10.44 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:



10.44.1 Detailed Description

Meta accumulator for [lor_basic](#).

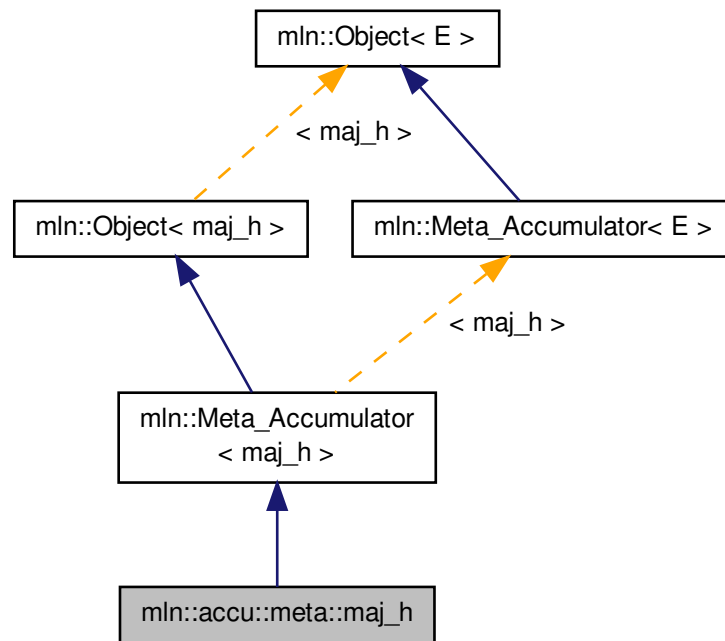
Definition at line 76 of file `lor_basic.hh`.

10.45 mln::accu::meta::maj_h Struct Reference

Meta accumulator for [maj_h](#).

```
#include <maj_h.hh>
```

Inheritance diagram for mln::accu::meta::maj_h:



10.45.1 Detailed Description

Meta accumulator for [maj_h](#).

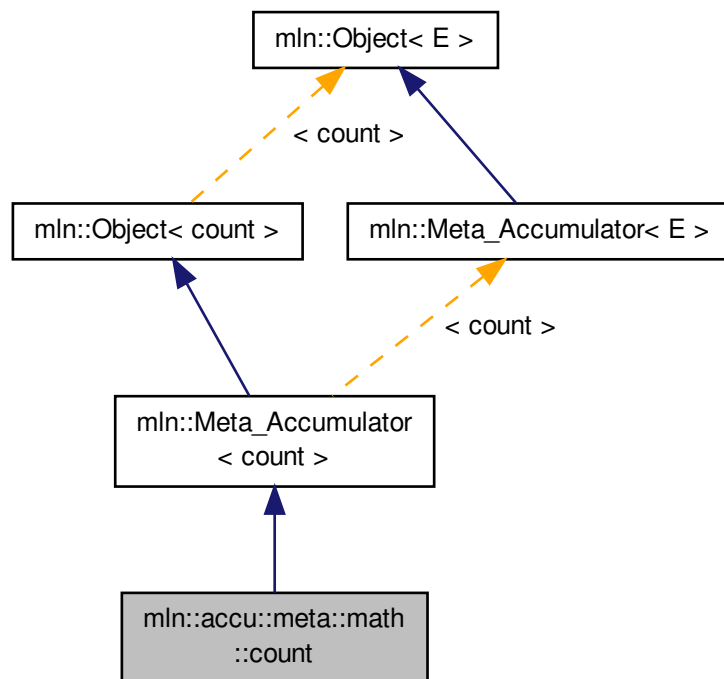
Definition at line 101 of file maj_h.hh.

10.46 mln::accu::meta::math::count Struct Reference

Meta accumulator for count.

```
#include <count.hh>
```

Inheritance diagram for mln::accu::meta::math::count:



10.46.1 Detailed Description

Meta accumulator for count.

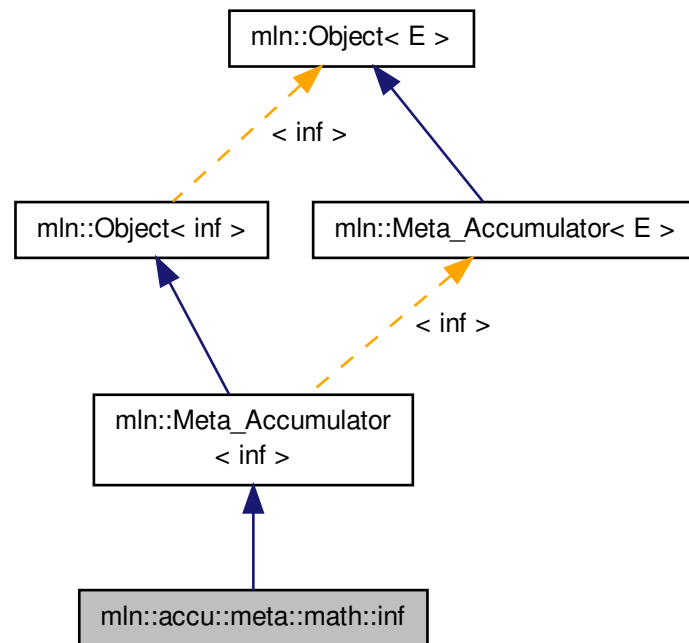
Definition at line 77 of file count.hh.

10.47 mln::accu::meta::math::inf Struct Reference

Meta accumulator for inf.

```
#include <inf.hh>
```

Inheritance diagram for `mln::accu::meta::math::inf`:



10.47.1 Detailed Description

Meta accumulator for `inf`.

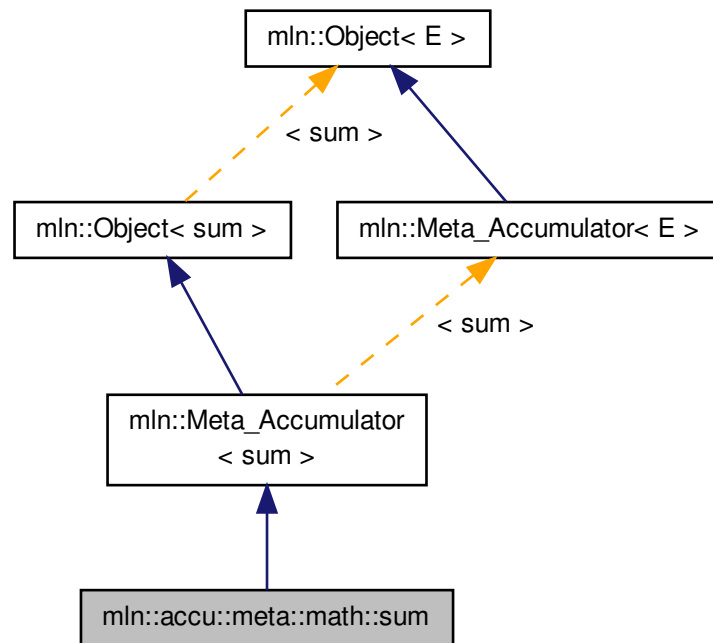
Definition at line 97 of file `accu/math/inf.hh`.

10.48 mln::accu::meta::math::sum Struct Reference

Meta accumulator for `sum`.

```
#include <sum.hh>
```

Inheritance diagram for mln::accu::meta::math::sum:



10.48.1 Detailed Description

Meta accumulator for sum.

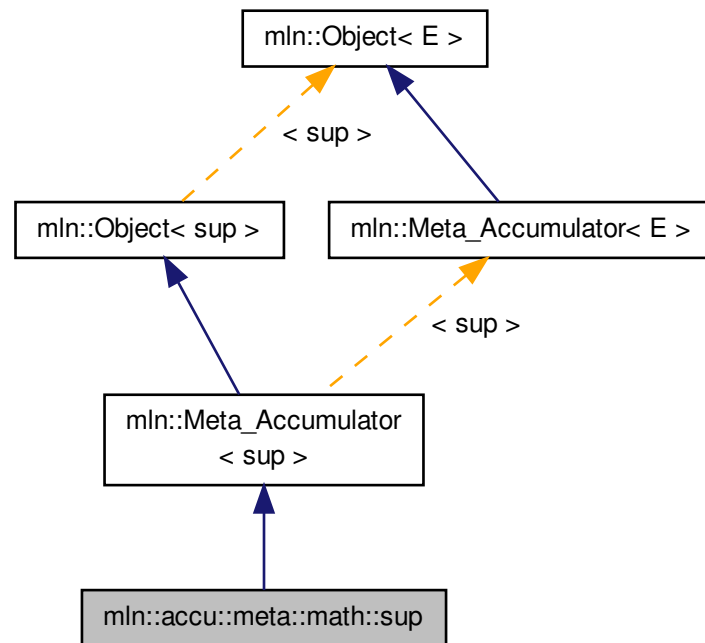
Definition at line 66 of file accu/math/sum.hh.

10.49 mln::accu::meta::math::sup Struct Reference

Meta accumulator for sup.

```
#include <sup.hh>
```

Inheritance diagram for `mln::accu::meta::math::sup`:



10.49.1 Detailed Description

Meta accumulator for `sup`.

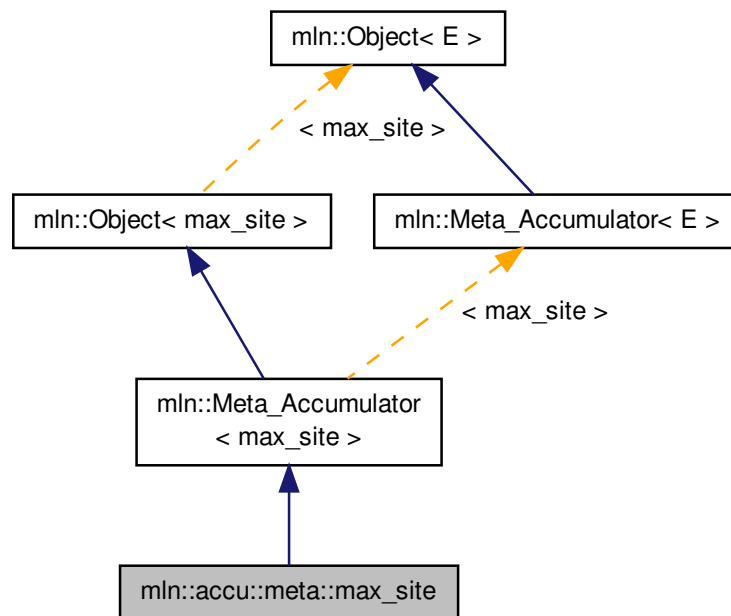
Definition at line 98 of file `accu/math/sup.hh`.

10.50 mln::accu::meta::max_site Struct Reference

Meta accumulator for [max_site](#).

```
#include <max_site.hh>
```


Inheritance diagram for mln::accu::meta::max_site:



10.50.1 Detailed Description

Meta accumulator for [max_site](#).

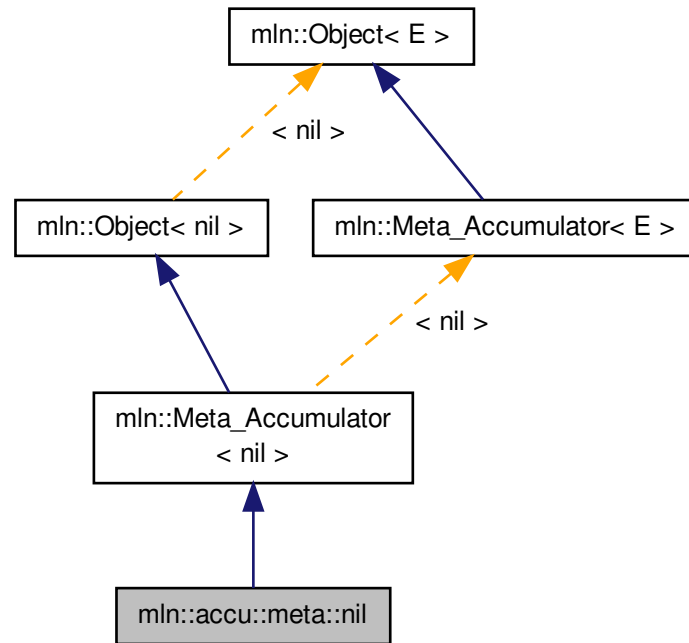
Definition at line 90 of file max_site.hh.

10.51 mln::accu::meta::nil Struct Reference

Meta accumulator for nil.

```
#include <nil.hh>
```

Inheritance diagram for `mln::accu::meta::nil`:



10.51.1 Detailed Description

Meta accumulator for nil.

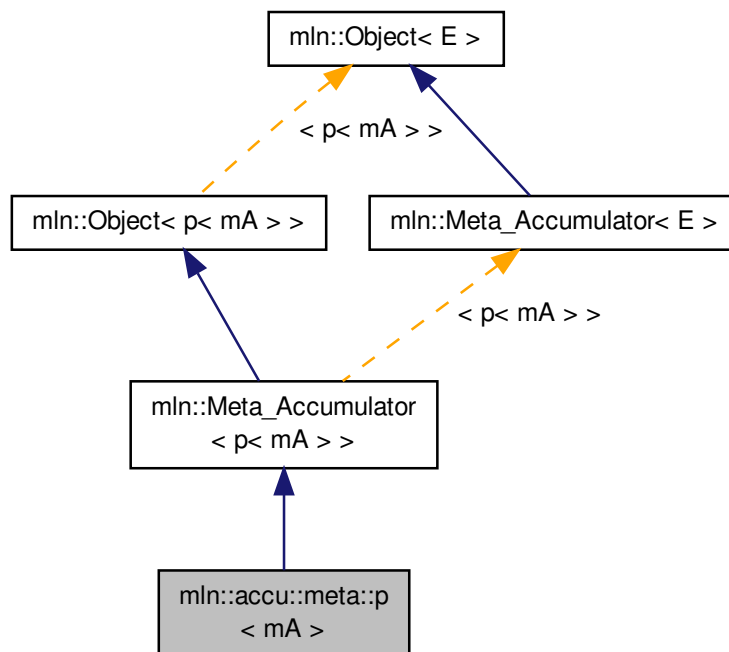
Definition at line 77 of file `accu/nil.hh`.

10.52 `mln::accu::meta::p< mA >` Struct Template Reference

Meta accumulator for p.

```
#include <p.hh>
```

Inheritance diagram for mln::accu::meta::p< mA >:



10.52.1 Detailed Description

```
template<typename mA>struct mln::accu::meta::p< mA >
```

Meta accumulator for p.

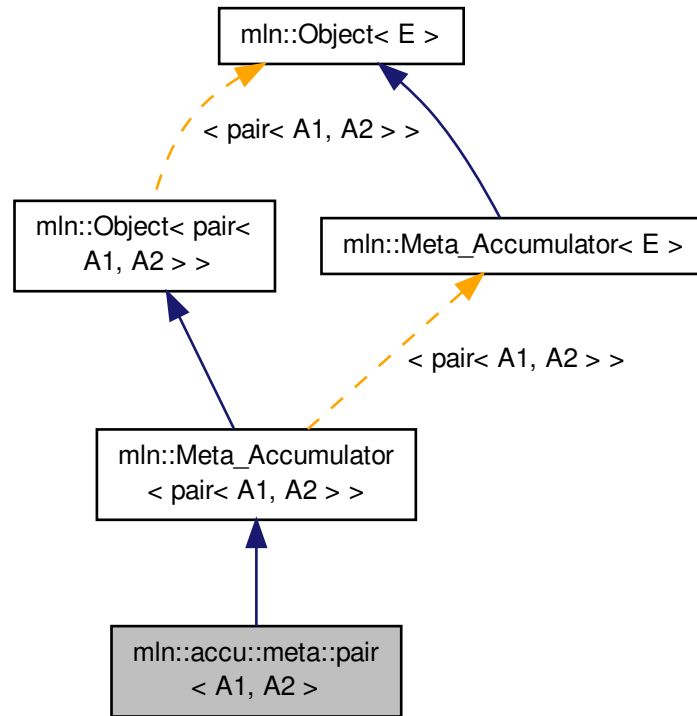
Definition at line 83 of file p.hh.

10.53 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 >`:



10.53.1 Detailed Description

```
template<typename A1, typename A2>struct mln::accu::meta::pair< A1, A2 >
```

Meta accumulator for pair.

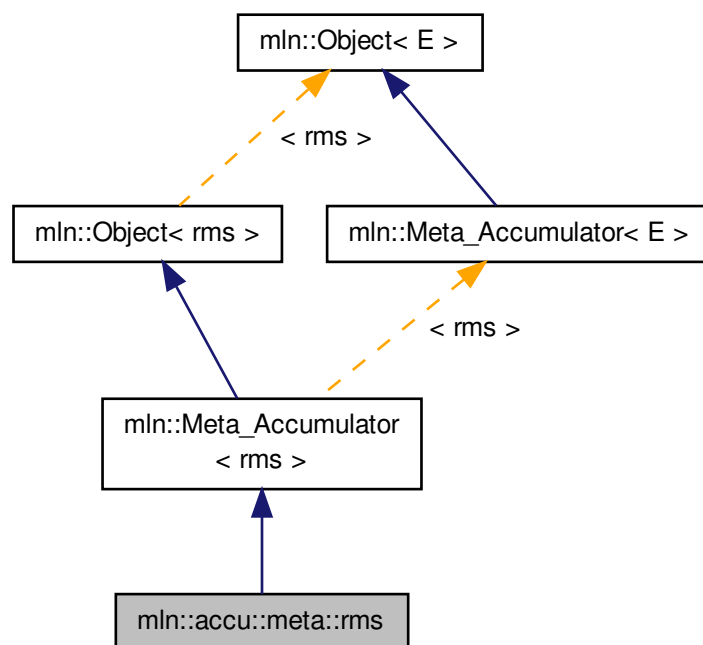
Definition at line 110 of file pair.hh.

10.54 mln::accu::meta::rms Struct Reference

Meta accumulator for rms.

```
#include <rms.hh>
```

Inheritance diagram for mln::accu::meta::rms:



10.54.1 Detailed Description

Meta accumulator for rms.

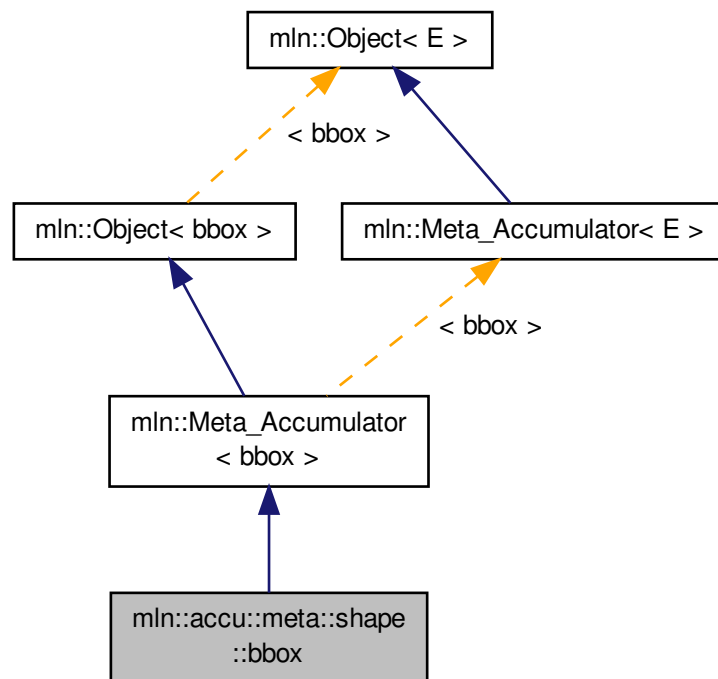
Definition at line 88 of file accu/rms.hh.

10.55 mln::accu::meta::shape::bbox Struct Reference

Meta accumulator for bbox.

```
#include <bbox.hh>
```

Inheritance diagram for mln::accu::meta::shape::bbox:



10.55.1 Detailed Description

Meta accumulator for bbox.

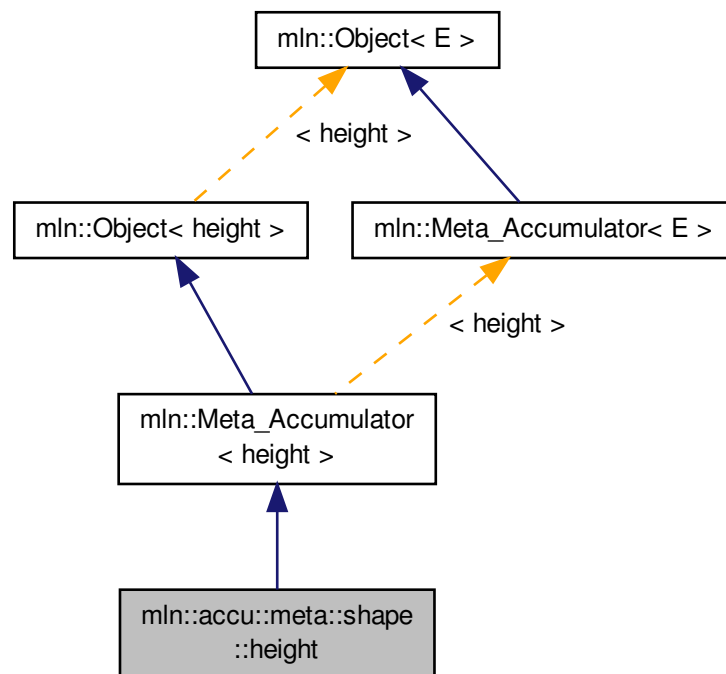
Definition at line 94 of file accu/shape/bbox.hh.

10.56 mln::accu::meta::shape::height Struct Reference

Meta accumulator for height.

```
#include <height.hh>
```

Inheritance diagram for mln::accu::meta::shape::height:



10.56.1 Detailed Description

Meta accumulator for height.

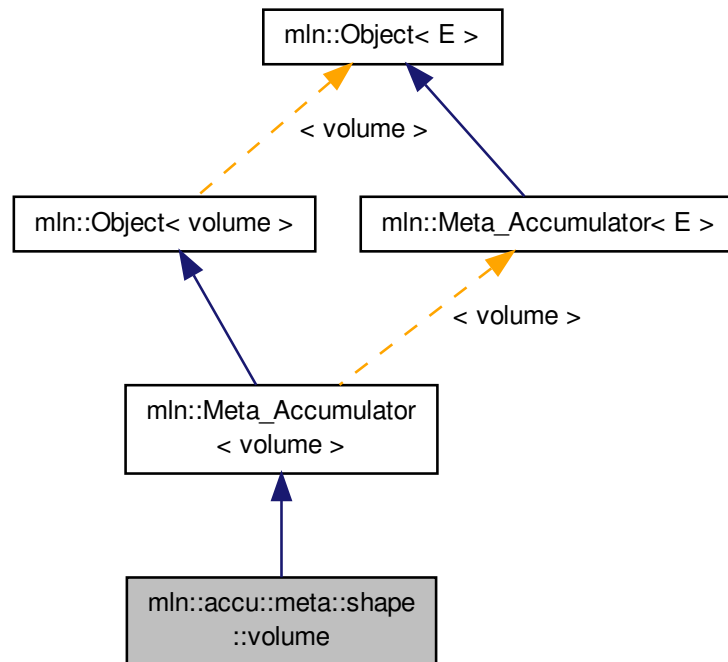
Definition at line 121 of file accu/shape/height.hh.

10.57 mln::accu::meta::shape::volume Struct Reference

Meta accumulator for volume.

```
#include <volume.hh>
```

Inheritance diagram for `mln::accu::meta::shape::volume`:



10.57.1 Detailed Description

Meta accumulator for volume.

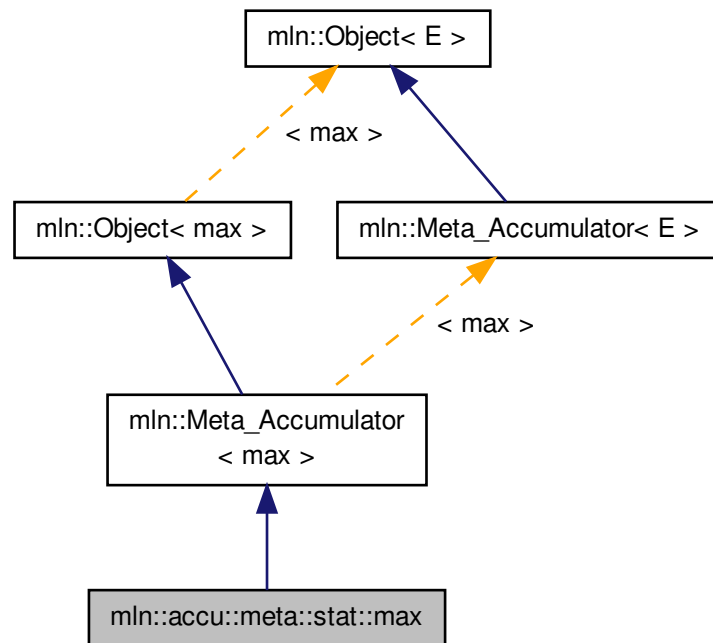
Definition at line 120 of file `accu/shape/volume.hh`.

10.58 mln::accu::meta::stat::max Struct Reference

Meta accumulator for max.

```
#include <max.hh>
```


Inheritance diagram for mln::accu::meta::stat::max:



10.58.1 Detailed Description

Meta accumulator for max.

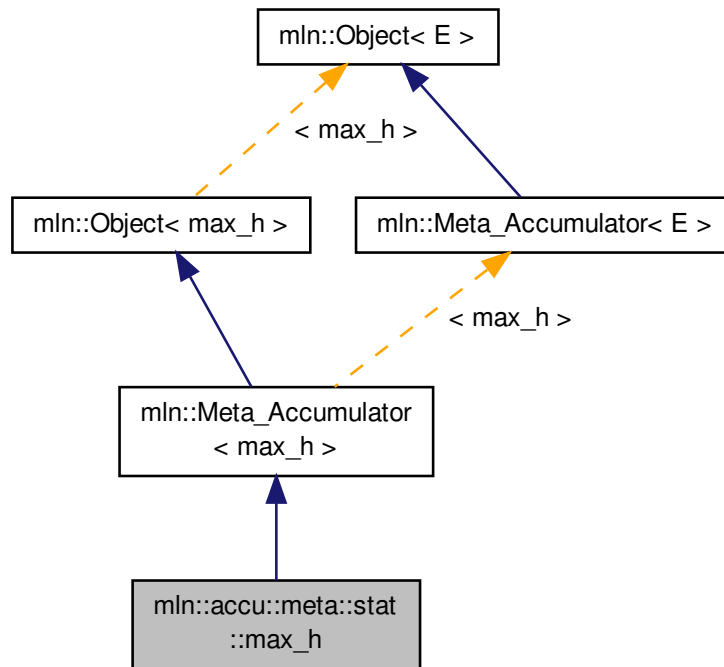
Definition at line 78 of file accu/stat/max.hh.

10.59 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`:



10.59.1 Detailed Description

Meta accumulator for max.

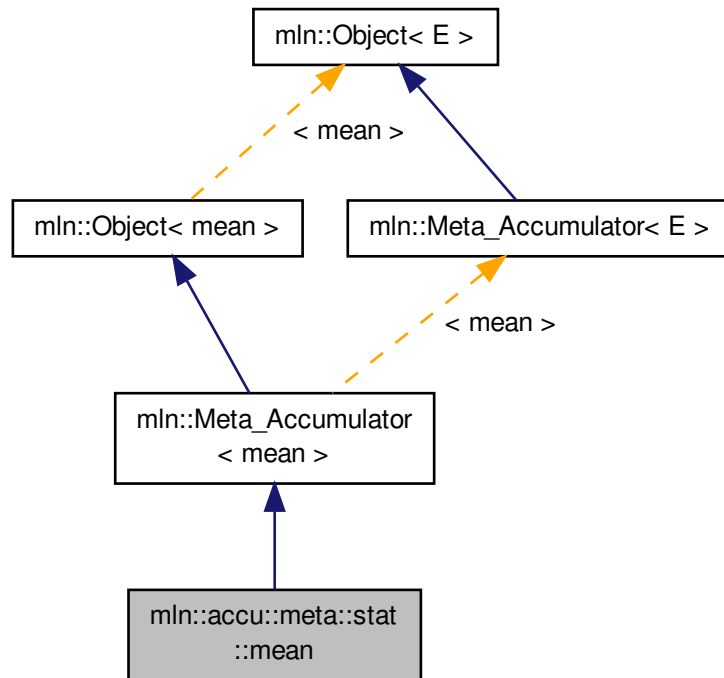
Definition at line 77 of file `max_h.hh`.

10.60 `mln::accu::meta::stat::mean` Struct Reference

Meta accumulator for mean.

```
#include <mean.hh>
```

Inheritance diagram for mln::accu::meta::stat::mean:



10.60.1 Detailed Description

Meta accumulator for mean.

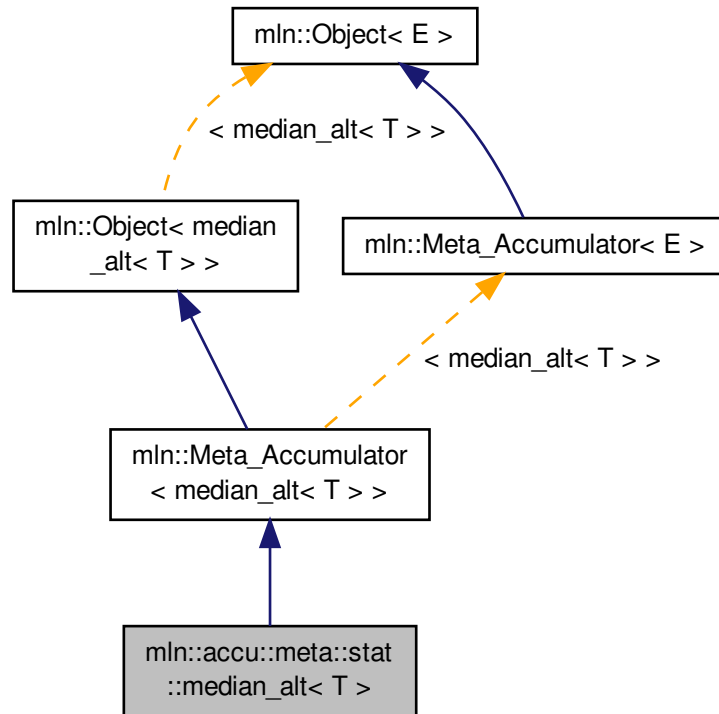
Definition at line 65 of file accu/stat/mean.hh.

10.61 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 >`:



10.61.1 Detailed Description

```
template<typename T>struct mln::accu::meta::stat::median_alt< T >
```

Meta accumulator for [median_alt](#).

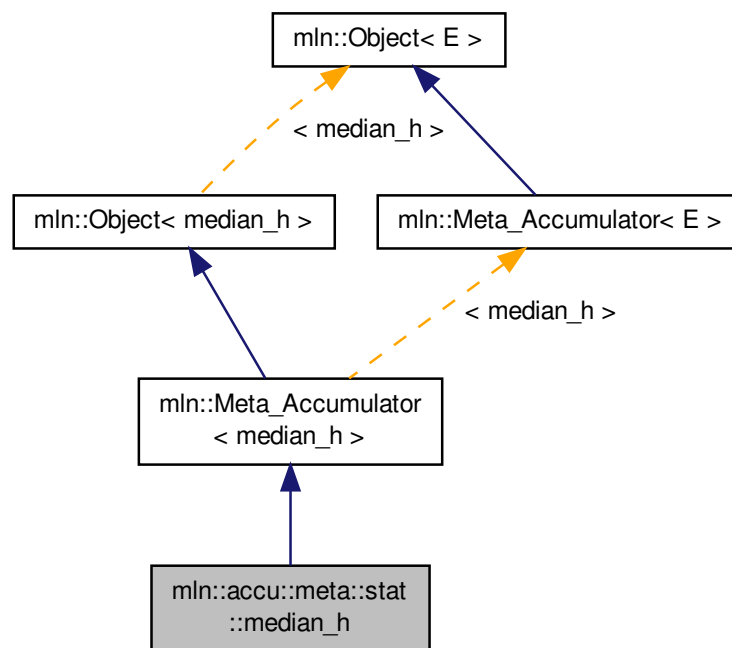
Definition at line 122 of file `median_alt.hh`.

10.62 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:



10.62.1 Detailed Description

Meta accumulator for [median_h](#).

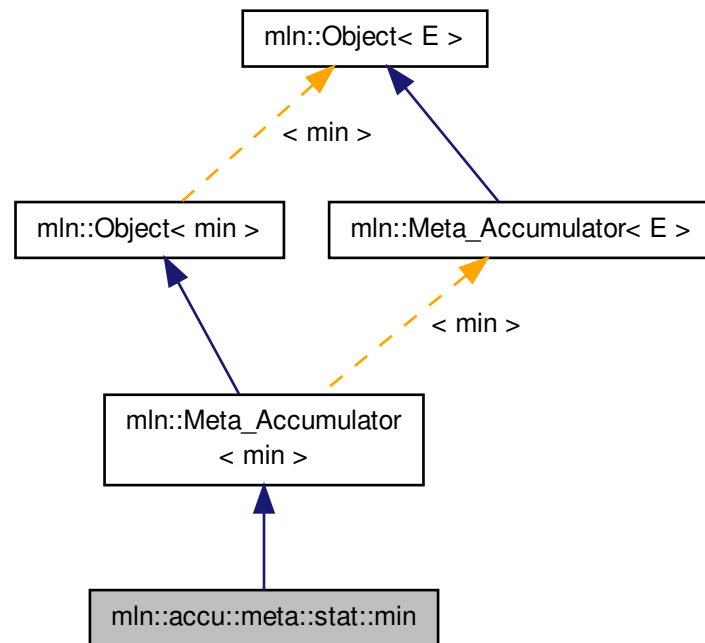
Definition at line 61 of file median_h.hh.

10.63 mln::accu::meta::stat::min Struct Reference

Meta accumulator for min.

```
#include <min.hh>
```

Inheritance diagram for mln::accu::meta::stat::min:



10.63.1 Detailed Description

Meta accumulator for min.

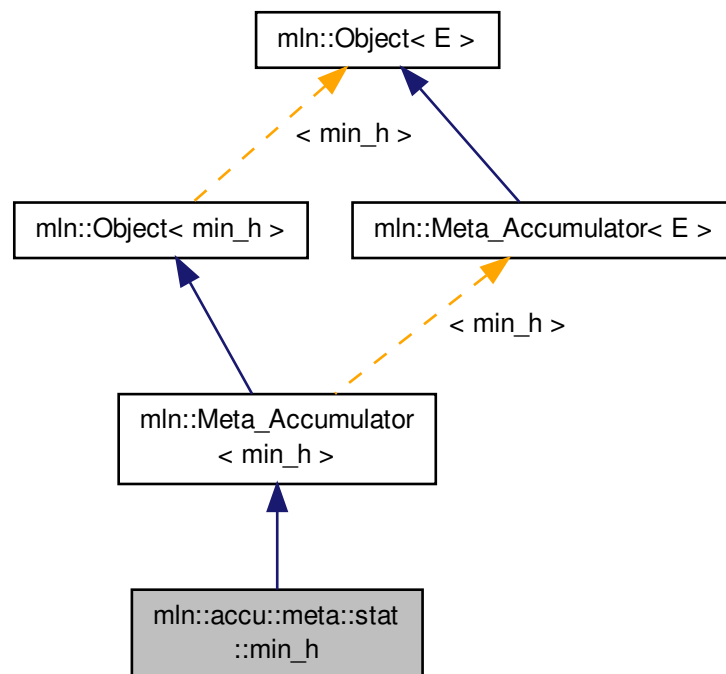
Definition at line 78 of file accu/stat/min.hh.

10.64 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:



10.64.1 Detailed Description

Meta accumulator for min.

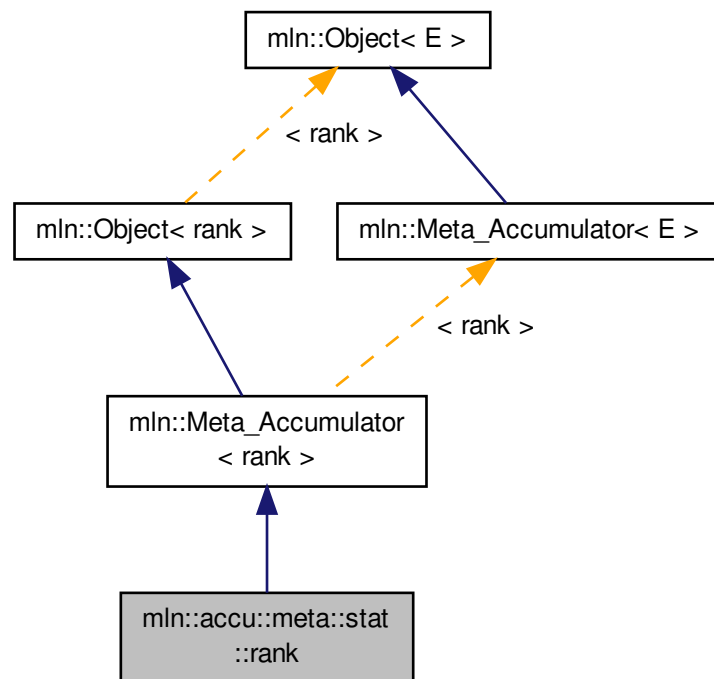
Definition at line 77 of file `min_h.hh`.

10.65 mln::accu::meta::stat::rank Struct Reference

Meta accumulator for rank.

```
#include <rank.hh>
```

Inheritance diagram for mln::accu::meta::stat::rank:



10.65.1 Detailed Description

Meta accumulator for rank.

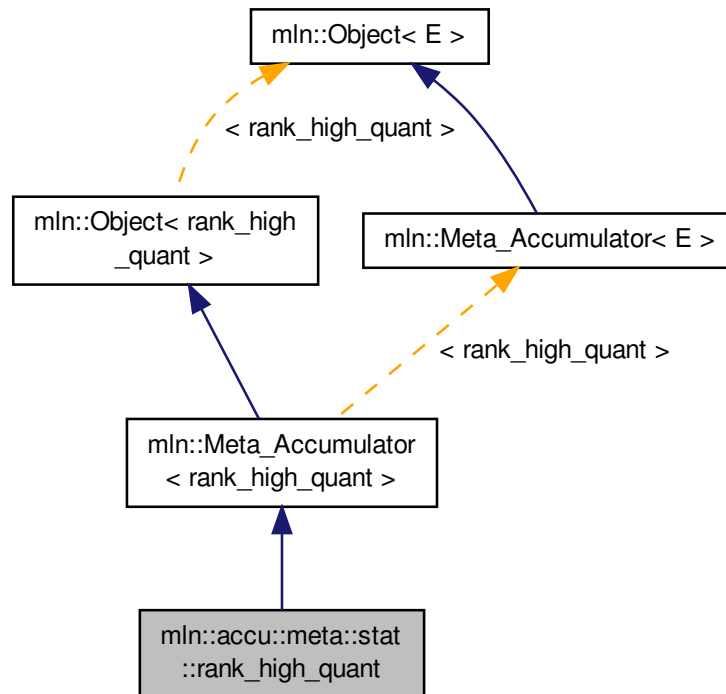
Definition at line 123 of file rank.hh.

10.66 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:



10.66.1 Detailed Description

Meta accumulator for [rank_high_quant](#).

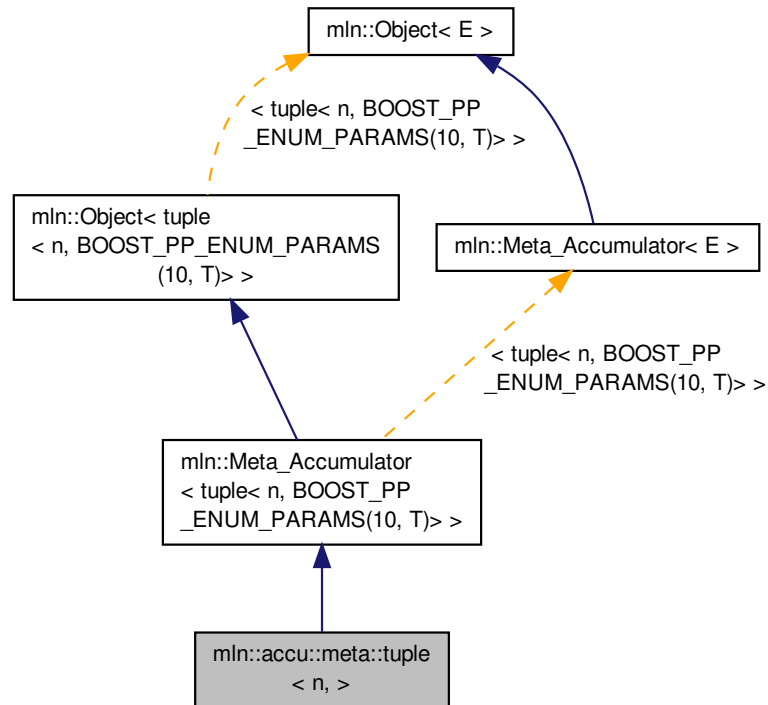
Definition at line 102 of file rank_high_quant.hh.

10.67 mln::accu::meta::tuple< n, > Struct Template Reference

Meta accumulator for tuple.

```
#include <tuple.hh>
```

Inheritance diagram for `mln::accu::meta::tuple< n, >`:



10.67.1 Detailed Description

```
template<unsigned n, BOOST_PP_ENUM_PARAMS_WITH_A_DEFAULT(10, typename T, boost::tuples::null_type)>struct mln::accu::meta::tuple< n, >
```

Meta accumulator for tuple.

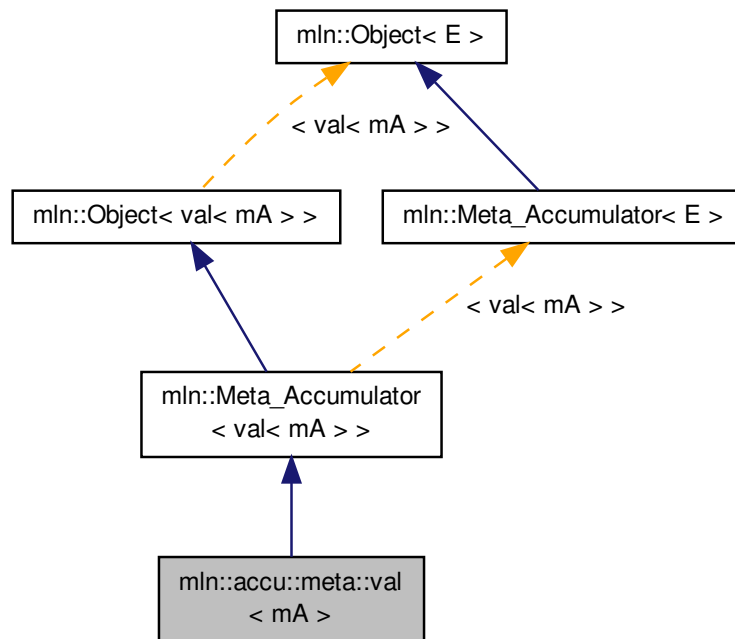
Definition at line 111 of file tuple.hh.

10.68 mln::accu::meta::val< mA > Struct Template Reference

Meta accumulator for val.

```
#include <v.hh>
```

Inheritance diagram for mln::accu::meta::val< mA >:



10.68.1 Detailed Description

```
template<typename mA>struct mln::accu::meta::val< mA >
```

Meta accumulator for val.

Definition at line 87 of file v.hh.

10.69 mln::accu::nil< T > Struct Template Reference

Define an accumulator that does nothing.

```
#include <nil.hh>
```

Inherits `mln::accu::internal::base< util::ignore, nil< T > >`.

Public Member Functions

- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
Take as initialization the value t.
- void `take_n_times` (unsigned n, const T &t)
Take n times the value t.
- `util::ignore to_result` () const

Get the value of the accumulator.

- void `init` ()

Manipulators.

10.69.1 Detailed Description

```
template<typename T>struct mln::accu::nil< T >
```

Define an accumulator that does nothing.

Definition at line 49 of file `accu/nil.hh`.

10.69.2 Member Function Documentation

10.69.2.1 `template<typename T> void mln::accu::nil< T >::init () [inline]`

Manipulators.

Definition at line 100 of file `accu/nil.hh`.

10.69.2.2 `template<typename T> bool mln::accu::nil< T >::is_valid () const [inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 136 of file `accu/nil.hh`.

10.69.2.3 `void mln::Accumulator< nil< T > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.69.2.4 `void mln::Accumulator< nil< T > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.69.2.5 `template<typename T> util::ignore mln::accu::nil< T >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 128 of file `accu/nil.hh`.

10.70 mln::accu::p< A > Struct Template Reference

Generic `p` of accumulators.

```
#include <p.hh>
```

Inherits `mln::accu::internal::base< const A::result &, p< A > >`.

Public Member Functions

- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
Take as initialization the value t.
- void `take_n_times` (unsigned n, const T &t)
Take n times the value t.
- const A::result & `to_result` () const
Get the value of the accumulator.
- void `init` ()
Manipulators.

10.70.1 Detailed Description

template<typename A>struct mln::accu::p< A >

Generic p of accumulators.

The parameter V is the type of values.

Definition at line 50 of file p.hh.

10.70.2 Member Function Documentation

10.70.2.1 template<typename A > void mln::accu::p< A >::init () [inline]

Manipulators.

Definition at line 115 of file p.hh.

10.70.2.2 template<typename A > bool mln::accu::p< A >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 155 of file p.hh.

10.70.2.3 void mln::Accumulator< p< A > >::take_as_init (const T & t) [inherited]

Take as initialization the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.70.2.4 void mln::Accumulator< p< A > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.70.2.5 template<typename A > const A::result & mln::accu::p< A >::to_result () const [inline]

Get the value of the accumulator.

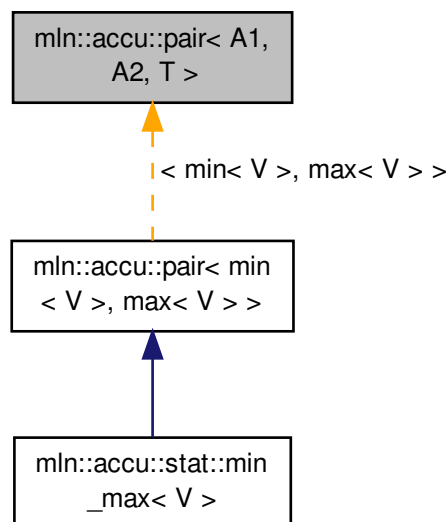
Definition at line 147 of file p.hh.

10.71 mln::accu::pair< A1, A2, T > Struct Template Reference

Generic pair of accumulators.

```
#include <pair.hh>
```

Inheritance diagram for mln::accu::pair< A1, A2, T >:



Public Member Functions

- `A1::result first () const`
Return the result of the first accumulator.
- `A1 first_accu () const`
Return the first accumulator.
- `bool is_valid () const`
Check whether this accu is able to return a result.
- `A2::result second () const`
Return the result of the second accumulator.
- `A2 second_accu () const`
Return the second accumulator.
- `void take_as_init (const T &t)`
Take as initialization the value t.
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t.
- `void init ()`
Manipulators.
- `std::pair< typename A1::result, typename A2::result > to_result () const`
Get the value of the accumulator.

10.71.1 Detailed Description

```
template<typename A1, typename A2, typename T = mln_argument(A1)> struct mln::accu::pair< A1, A2, T >
```

Generic pair of accumulators.

The parameter `T` is the type of values.

Definition at line 58 of file pair.hh.

10.71.2 Member Function Documentation

10.71.2.1 `template<typename A1 , typename A2 , typename T > A1::result mln::accu::pair< A1, A2, T >::first () const`
`[inline]`

Return the result of the first accumulator.

Definition at line 191 of file pair.hh.

References `mln::accu::pair< A1, A2, T >::to_result()`.

10.71.2.2 `template<typename A1 , typename A2 , typename T > A1 mln::accu::pair< A1, A2, T >::first_accu () const`
`[inline]`

Return the first accumulator.

Definition at line 209 of file pair.hh.

10.71.2.3 `template<typename A1 , typename A2 , typename T > void mln::accu::pair< A1, A2, T >::init ()` `[inline]`

Manipulators.

Definition at line 136 of file pair.hh.

10.71.2.4 `template<typename A1 , typename A2 , typename T > bool mln::accu::pair< A1, A2, T >::is_valid () const`
`[inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 226 of file pair.hh.

10.71.2.5 `template<typename A1 , typename A2 , typename T > A2::result mln::accu::pair< A1, A2, T >::second () const`
`[inline]`

Return the result of the second accumulator.

Definition at line 199 of file pair.hh.

References `mln::accu::pair< A1, A2, T >::to_result()`.

10.71.2.6 `template<typename A1 , typename A2 , typename T > A2 mln::accu::pair< A1, A2, T >::second_accu () const`
`[inline]`

Return the second accumulator.

Definition at line 217 of file pair.hh.

10.71.2.7 `void mln::Accumulator< pair< A1, A2, T > >::take_as_init (const T & t)` `[inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.71.2.8 `void mln::Accumulator< pair< A1, A2, T > >::take_n_times (unsigned n, const T & t)` `[inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.71.2.9 `template<typename A1, typename A2, typename T> std::pair< typename A1::result, typename A2::result > mln::accu::pair< A1, A2, T >::to_result () const` `[inline]`

Get the value of the accumulator.

Definition at line 172 of file `pair.hh`.

Referenced by `mln::accu::pair< A1, A2, T >::first()`, and `mln::accu::pair< A1, A2, T >::second()`.

10.72 mln::accu::rms< T, V > Struct Template Reference

Generic root mean square accumulator class.

```
#include <rms.hh>
```

Inherits `mln::accu::internal::base< V, rms< T, V > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value `t`.
- `void take_n_times (unsigned n, const T &t)`
Take `n` times the value `t`.
- `V to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.72.1 Detailed Description

```
template<typename T, typename V> struct mln::accu::rms< T, V >
```

Generic root mean square accumulator class.

The parameter `T` is the type of the root mean square value.

Definition at line 52 of file `accu/rms.hh`.

10.72.2 Member Function Documentation

10.72.2.1 `template<typename T, typename V> void mln::accu::rms< T, V >::init () [inline]`

Manipulators.

Definition at line 112 of file `accu/rms.hh`.

References `mln::literal::zero`.

10.72.2.2 `template<typename T, typename V> bool mln::accu::rms< T, V >::is_valid () const [inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 166 of file `accu/rms.hh`.

10.72.2.3 `void mln::Accumulator< rms< T, V > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.72.2.4 `void mln::Accumulator< rms< T, V > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.72.2.5 `template<typename T, typename V> V mln::accu::rms< T, V >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 148 of file `accu/rms.hh`.

10.73 mln::accu::shape::bbox< P > Struct Template Reference

Generic bounding box accumulator class.

```
#include <bbox.hh>
```

Inherits `mln::accu::internal::base< const box< P > &, bbox< P > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value `t`.
- `void take_n_times (unsigned n, const T &t)`
Take `n` times the value `t`.
- `const box< P > & to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.73.1 Detailed Description

`template<typename P> struct mln::accu::shape::bbox< P >`

Generic bounding box accumulator class.

The parameter `P` is the type of points.

Definition at line 55 of file `accu/shape/bbox.hh`.

10.73.2 Member Function Documentation

10.73.2.1 `template<typename P> void bbox< P >::init () [inline]`

Manipulators.

Definition at line 124 of file `accu/shape/bbox.hh`.

10.73.2.2 `template<typename P> bool bbox< P >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 224 of file `accu/shape/bbox.hh`.

10.73.2.3 `void mln::Accumulator< bbox< P > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.73.2.4 `void mln::Accumulator< bbox< P > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.73.2.5 `template<typename P> const box< P > & bbox< P >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 215 of file `accu/shape/bbox.hh`.

Referenced by `mln::geom::rotate()`.

10.74 mln::accu::shape::height< I > Struct Template Reference

Height accumulator.

```
#include <height.hh>
```

Inherits `mln::accu::internal::base< unsigned, height< I > >`.

Public Types

- typedef [util::pix< I >](#) [argument](#)

The accumulated data type.

- typedef [argument::value](#) [value](#)

The value type associated to the pixel type.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- unsigned [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.
- void [set_value](#) (unsigned h)
Force the value of the counter to h.

10.74.1 Detailed Description

```
template<typename I>struct mln::accu::shape::height< I >
```

Height accumulator.

The parameter I is the image type on which the accumulator of pixels is built.

Definition at line 68 of file `accu/shape/height.hh`.

10.74.2 Member Typedef Documentation

10.74.2.1 `template<typename I> typedef util::pix<I> mln::accu::shape::height< I >::argument`

The accumulated data type.

The height of component is represented by the height of its root pixel. See `mln::morpho::closing_height` and `mln::morpho::opening_height` for actual uses of this accumulator. FIXME: Replaced by [mln::morpho::attribute::height](#)

Definition at line 78 of file `accu/shape/height.hh`.

10.74.2.2 `template<typename I> typedef argument::value mln::accu::shape::height< I >::value`

The value type associated to the pixel type.

Definition at line 80 of file `accu/shape/height.hh`.

10.74.3 Member Function Documentation

10.74.3.1 `template<typename I> void height< I >::init() [inline]`

Manipulators.

Definition at line 150 of file `accu/shape/height.hh`.

10.74.3.2 `template<typename I> bool height<I>::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 199 of file `accu/shape/height.hh`.

10.74.3.3 `template<typename I> void height<I>::set_value (unsigned h) [inline]`

Force the value of the counter to *h*.

Definition at line 188 of file `accu/shape/height.hh`.

10.74.3.4 `void mln::Accumulator< height<I>>::take_as_init (const T & t) [inherited]`

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.74.3.5 `void mln::Accumulator< height<I>>::take_n_times (unsigned n, const T & t) [inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.74.3.6 `template<typename I> unsigned height<I>::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 180 of file `accu/shape/height.hh`.

10.75 mln::accu::shape::volume< I > Struct Template Reference

Volume accumulator class.

`#include <volume.hh>`

Inherits `mln::accu::internal::base< unsigned, volume< I >>`.

Public Types

- typedef `util::pix< I > argument`
The accumulated data type.
- typedef `argument::value value`
The value type associated to the pixel type.

Public Member Functions

- bool `is_valid () const`
Check whether this accu is able to return a result.
- void `take_as_init (const T &t)`
Take as initialization the value t.
- void `take_n_times (unsigned n, const T &t)`
Take n times the value t.

- unsigned [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.
- void [set_value](#) (unsigned v)
Force the value of the counter to v.

10.75.1 Detailed Description

`template<typename I> struct mln::accu::shape::volume< I >`

Volume accumulator class.

The parameter `I` is the image type on which the accumulator of pixels is built.

Definition at line 66 of file `accu/shape/volume.hh`.

10.75.2 Member Typedef Documentation

10.75.2.1 `template<typename I> typedef util::pix<I> mln::accu::shape::volume< I >::argument`

The accumulated data type.

The volume of component is represented by the volume of its root pixel. See `mln::morpho::closing_volume` and `mln::morpho::opening_volume` for actual uses of this accumulator. FIXME: Replaced by [mln::morpho::attribute::volume](#)

Definition at line 76 of file `accu/shape/volume.hh`.

10.75.2.2 `template<typename I> typedef argument::value mln::accu::shape::volume< I >::value`

The value type associated to the pixel type.

Definition at line 78 of file `accu/shape/volume.hh`.

10.75.3 Member Function Documentation

10.75.3.1 `template<typename I> void volume< I >::init () [inline]`

Manipulators.

Definition at line 148 of file `accu/shape/volume.hh`.

References `mln::literal::zero`.

10.75.3.2 `template<typename I> bool volume< I >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 215 of file `accu/shape/volume.hh`.

10.75.3.3 `template<typename I> void volume< I >::set_value (unsigned v) [inline]`

Force the value of the counter to v.

Definition at line 204 of file `accu/shape/volume.hh`.

References `mln::literal::zero`.

10.75.3.4 `void mln::Accumulator< volume< I > >::take_as_init (const T & t)` `[inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.75.3.5 `void mln::Accumulator< volume< I > >::take_n_times (unsigned n, const T & t)` `[inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.75.3.6 `template<typename I> unsigned volume< I >::to_result () const` `[inline]`

Get the value of the accumulator.

Definition at line 196 of file `accu/shape/volume.hh`.

10.76 mln::accu::site_set::rectangularity< P > Class Template Reference

Compute the rectangularity of a site set.

`#include <rectangularity.hh>`

Inherits `mln::accu::internal::couple< accu::shape::bbox< P >, accu::math::count< P >, float, rectangularity< P >`.

Public Member Functions

- `A2::result area () const`
Return the site set area.
- `A1::result bbox () const`
Return the site set bounding box.
- `rectangularity ()`
Constructor.
- `template<typename T>`
`void take_as_init (const T &t)`
Take as initialization the value t.
- `template<typename T>`
`void take_n_times (unsigned n, const T &t)`
Take n times the value t.
- `result to_result () const`
Return the rectangularity value.

10.76.1 Detailed Description

`template<typename P>class mln::accu::site_set::rectangularity< P >`

Compute the rectangularity of a site set.

Definition at line 51 of file `rectangularity.hh`.

10.76.2 Constructor & Destructor Documentation

10.76.2.1 `template<typename P> mln::accu::site_set::rectangularity< P>::rectangularity () [inline]`

Constructor.

Definition at line 91 of file rectangularity.hh.

10.76.3 Member Function Documentation

10.76.3.1 `template<typename P> rectangularity< P>::A2::result mln::accu::site_set::rectangularity< P>::area () const [inline]`

Return the site set area.

Definition at line 107 of file rectangularity.hh.

10.76.3.2 `template<typename P> rectangularity< P>::A1::result mln::accu::site_set::rectangularity< P>::bbox () const [inline]`

Return the site set bounding box.

Definition at line 98 of file rectangularity.hh.

10.76.3.3 `template<typename E> template<typename T> void Accumulator< E>::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 186 of file accumulator.hh.

References `mln::mln_exact()`.

10.76.3.4 `template<typename E> template<typename T> void Accumulator< E>::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 213 of file accumulator.hh.

References `mln::mln_exact()`.

10.76.3.5 `template<typename P> rectangularity< P>::result mln::accu::site_set::rectangularity< P>::to_result () const [inline]`

Return the rectangularity value.

Definition at line 116 of file rectangularity.hh.

10.77 mln::accu::stat::deviation< T, S, M > Struct Template Reference

Generic standard deviation accumulator class.

```
#include <deviation.hh>
```

Inherits `mln::accu::internal::base< M, deviation< T, S, M > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value `t`.
- `void take_n_times (unsigned n, const T &t)`
Take `n` times the value `t`.
- `M to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.77.1 Detailed Description

```
template<typename T, typename S = typename mln::value::props< T >::sum, typename M = S>struct mln::accu::stat-
::deviation< T, S, M >
```

Generic standard deviation accumulator class.

Parameter `T` is the type of values that we sum. Parameter `S` is the type to store the standard deviation; the default type of `S` is the summation type (property) of `T`. Parameter `M` is the type of the mean value; the default type of `M` is `S`.

Definition at line 62 of file `deviation.hh`.

10.77.2 Member Function Documentation

10.77.2.1 `template<typename T, typename S, typename M > void mln::accu::stat::deviation< T, S, M >::init ()`
`[inline]`

Manipulators.

Definition at line 132 of file `deviation.hh`.

10.77.2.2 `template<typename T, typename S, typename M > bool mln::accu::stat::deviation< T, S, M >::is_valid ()`
`const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 177 of file `deviation.hh`.

10.77.2.3 `void mln::Accumulator< deviation< T, S, M > >::take_as_init (const T &t)` `[inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.77.2.4 void mln::Accumulator< deviation< T, S, M > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.77.2.5 template<typename T, typename S, typename M > M mln::accu::stat::deviation< T, S, M >::to_result ()
const [inline]

Get the value of the accumulator.

Definition at line 159 of file deviation.hh.

10.78 mln::accu::stat::histo3d_rgb< V > Struct Template Reference

Define a histogram as accumulator which returns an [image3d](#).

```
#include <histo3d_rgb.hh>
```

Inherits mln::accu::internal::base< image3d< unsigned >, histo3d_rgb< V > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accumulator is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- [histo3d_rgb](#) ()
Constructors.
- void [init](#) ()
Manipulators.
- void [take](#) (const argument &t)
Update the histogram with the RGB pixel t.
- void [take](#) (const [histo3d_rgb](#)< V > &other)
Update the histogram with an other histogram.
- [result to_result](#) () const
Accessors.

10.78.1 Detailed Description

```
template<typename V>struct mln::accu::stat::histo3d_rgb< V >
```

Define a histogram as accumulator which returns an [image3d](#).

Param V defines the type of the input image value. It is in this space that we count the values. For instance, this histogram works well for [image2d](#)< rgb<2> > or with [image2d](#)< rgb<7> >. The number of bins depends directly the values V. For 8 bits there is 256x3 bins. Note that less quantification works too.

Definition at line 167 of file histo3d_rgb.hh.

10.78.2 Constructor & Destructor Documentation

10.78.2.1 `template<typename V> histo3d_rgb<V>::histo3d_rgb() [inline]`

Constructors.

Infer the size of the resulting [image3d](#) domain.

By evaluating the minimum and the maximum of V, we define the domain of the resulting [image3d](#).

Definition at line 244 of file `histo3d_rgb.hh`.

10.78.3 Member Function Documentation

10.78.3.1 `template<typename V> void histo3d_rgb<V>::init() [inline]`

Manipulators.

Initialize the histogram with zero value.

This method must be called just before starting the use of the histogram. If it's not, resulting values won't converge to the density.

Definition at line 268 of file `histo3d_rgb.hh`.

References `mln::data::fill()`, and `mln::literal::zero`.

10.78.3.2 `template<typename V> bool histo3d_rgb<V>::is_valid() const [inline]`

Check whether this accumulator is able to return a result.

Depends if the resulting [image1d](#) is valid. We can assume it is quite always the case.

Definition at line 307 of file `histo3d_rgb.hh`.

10.78.3.3 `template<typename V> void histo3d_rgb<V>::take(const argument & t) [inline]`

Update the histogram with the RGB pixel t.

Parameters

<code>in</code>	<code>t</code>	a graylevel pixel of type V.
-----------------	----------------	------------------------------

The end user shouldn't call this method. In place of it, he can go through the data compute interface.

Definition at line 275 of file `histo3d_rgb.hh`.

10.78.3.4 `template<typename V> void histo3d_rgb<V>::take(const histo3d_rgb<V> & other) [inline]`

Update the histogram with an other histogram.

Parameters

<code>in</code>	<code>other</code>	the other histogram.
-----------------	--------------------	----------------------

The end user shouldn't call this method. This is part of data compute interface mechanism.

Definition at line 286 of file `histo3d_rgb.hh`.

10.78.3.5 void mln::Accumulator< histo3d_rgb< V > >::take_as_init (const T & t) [inherited]

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.78.3.6 void mln::Accumulator< histo3d_rgb< V > >::take_n_times (unsigned n, const T & t) [inherited]

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.78.3.7 template<typename V> histo3d_rgb< V >::result histo3d_rgb< V >::to_result () const [inline]

Accessors.

Return the histogram as an RGB [image3d](#).

This is the machinery to communicate with data compute interface. The end user should'nt use it.

Definition at line 293 of file histo3d_rgb.hh.

10.79 mln::accu::stat::max< T > Struct Template Reference

Generic max accumulator class.

```
#include <max.hh>
```

Inherits mln::accu::internal::base< const T &, max< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [set_value](#) (const T &t)
Force the value of the min to t.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- const T & [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.79.1 Detailed Description

```
template<typename T>struct mln::accu::stat::max< T >
```

Generic max accumulator class.

The parameter *T* is the type of values.

Definition at line 101 of file accu/stat/max.hh.

10.79.2 Member Function Documentation

10.79.2.1 `template<typename T> void max<T>::init () [inline]`

Manipulators.

Definition at line 146 of file `accu/stat/max.hh`.

10.79.2.2 `template<typename T> bool max<T>::is_valid () const [inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 196 of file `accu/stat/max.hh`.

10.79.2.3 `template<typename T> void max<T>::set_value (const T & t) [inline]`

Force the value of the min to `t`.

Definition at line 180 of file `accu/stat/max.hh`.

10.79.2.4 `void mln::Accumulator<max<T>>::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.79.2.5 `void mln::Accumulator<max<T>>::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.79.2.6 `template<typename T> const T & max<T>::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 188 of file `accu/stat/max.hh`.

10.80 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 &, max_h<V>>`.

Public Member Functions

- bool `is_valid` () const
Check whether this `accu` is able to return a result.
- void `take_as_init` (const T &t)
Take as initialization the value `t`.
- void `take_n_times` (unsigned n, const T &t)
Take `n` times the value `t`.

- const argument & [to_result](#) () const
Get the value of the accumulator.

- void [init](#) ()
Manipulators.

10.80.1 Detailed Description

`template<typename V>struct mln::accu::stat::max_h< V >`

Generic max function based on histogram over a value set with type V .

Definition at line 100 of file max_h.hh.

10.80.2 Member Function Documentation

10.80.2.1 `template<typename V > void max_h< V >::init () [inline]`

Manipulators.

Definition at line 280 of file max_h.hh.

10.80.2.2 `template<typename V > bool max_h< V >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 323 of file max_h.hh.

10.80.2.3 `void mln::Accumulator< max_h< V > >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.80.2.4 `void mln::Accumulator< max_h< V > >::take_n.times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.80.2.5 `template<typename V > const max_h< V >::argument & max_h< V >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 304 of file max_h.hh.

10.81 mln::accu::stat::mean< T, S, M > Struct Template Reference

Generic mean accumulator class.

```
#include <mean.hh>
```

Inherits `mln::accu::internal::base< M, mean< T, S, M > >`.

Public Member Functions

- `accu::math::count< T >::result count () const`
Get the cardinality.
- `bool is_valid () const`
Check whether this accu is able to return a result.
- `accu::math::sum< T >::result sum () const`
Get the sum of values.
- `void take_as_init (const T &t)`
Take as initialization the value t.
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t.
- `M to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.81.1 Detailed Description

`template<typename T, typename S = typename mln::value::props< T >::sum, typename M = S> struct mln::accu::stat::mean< T, S, M >`

Generic mean accumulator class.

Parameter `T` is the type of values that we sum. Parameter `S` is the type to store the sum of values; the default type of `S` is the summation type (property) of `T`. Parameter `M` is the type of the mean value; the default type of `M` is `S`.

Definition at line 119 of file `accu/stat/mean.hh`.

10.81.2 Member Function Documentation

10.81.2.1 `template<typename T, typename S, typename M > accu::math::count< T >::result mean< T, S, M >::count () const [inline]`

Get the cardinality.

Definition at line 242 of file `accu/stat/mean.hh`.

References `mln::accu::stat::mean< T, S, M >::to_result()`.

10.81.2.2 `template<typename T, typename S, typename M > void mean< T, S, M >::init () [inline]`

Manipulators.

Definition at line 173 of file `accu/stat/mean.hh`.

10.81.2.3 `template<typename T, typename S, typename M > bool mean< T, S, M >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 234 of file `accu/stat/mean.hh`.

10.81.2.4 `template<typename T, typename S, typename M> mln::math::sum< T >::result mean< T, S, M >::sum ()`
`const [inline]`

Get the sum of values.

Definition at line 251 of file `accu/stat/mean.hh`.

References `mln::accu::stat::mean< T, S, M >::to_result()`.

10.81.2.5 `void mln::Accumulator< mean< T, S, M > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.81.2.6 `void mln::Accumulator< mean< T, S, M > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.81.2.7 `template<typename T, typename S, typename M> M mean< T, S, M >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 216 of file `accu/stat/mean.hh`.

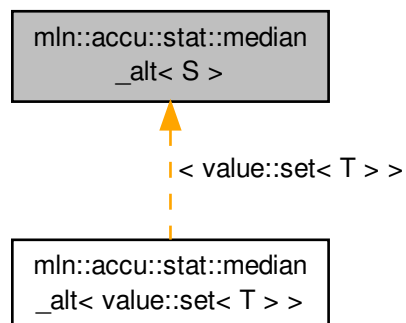
Referenced by `mln::accu::stat::mean< T, S, M >::count()`, and `mln::accu::stat::mean< T, S, M >::sum()`.

10.82 mln::accu::stat::median_alt< S > Struct Template Reference

Generic `median_alt` function based on histogram over a value set with type `S`.

`#include <median_alt.hh>`

Inheritance diagram for `mln::accu::stat::median_alt< S >`:



Public Member Functions

- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
Take as initialization the value t.
- void `take_n_times` (unsigned n, const T &t)
Take n times the value t.
- const argument & `to_result` () const
Get the value of the accumulator.
- void `take` (const argument &t)
Manipulators.

10.82.1 Detailed Description

template<typename S>struct mln::accu::stat::median_alt< S >

Generic `median_alt` function based on histogram over a value set with type S.

Definition at line 54 of file median_alt.hh.

10.82.2 Member Function Documentation

10.82.2.1 template<typename S > bool mln::accu::stat::median_alt< S >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 282 of file median_alt.hh.

10.82.2.2 template<typename S > void mln::accu::stat::median_alt< S >::take (const argument & t) [inline]

Manipulators.

Definition at line 165 of file median_alt.hh.

References mln::accu::stat::median_alt< S >::take().

Referenced by mln::accu::stat::median_alt< S >::take().

10.82.2.3 void mln::Accumulator< median_alt< S > >::take_as_init (const T & t) [inherited]

Take as initialization the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.82.2.4 void mln::Accumulator< median_alt< S > >::take_n_times (unsigned n, const T & t) [inherited]

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.82.2.5 `template<typename S> const median_alt< S >::argument & mln::accu::stat::median_alt< S >::to_result () const [inline]`

Get the value of the accumulator.

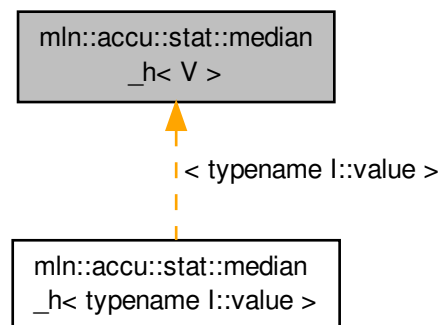
Definition at line 274 of file median_alt.hh.

10.83 mln::accu::stat::median_h< V > Struct Template Reference

Generic median function based on histogram over a value set with type V .

`#include <median_h.hh>`

Inheritance diagram for mln::accu::stat::median_h< V >:



Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `void take_as_init (const T &t)`
Take as initialization the value 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.

10.83.1 Detailed Description

`template<typename V> struct mln::accu::stat::median_h< V >`

Generic median function based on histogram over a value set with type V .

Definition at line 82 of file median_h.hh.

10.83.2 Member Function Documentation

10.83.2.1 `template<typename V> void median_h< V >::init () [inline]`

Manipulators.

Definition at line 267 of file median_h.hh.

10.83.2.2 `template<typename V> bool median_h< V >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 299 of file median_h.hh.

10.83.2.3 `void mln::Accumulator< median_h< V > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.83.2.4 `void mln::Accumulator< median_h< V > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.83.2.5 `template<typename V> const median_h< V >::argument & median_h< V >::to_result () const [inline]`

Get the value of the accumulator.

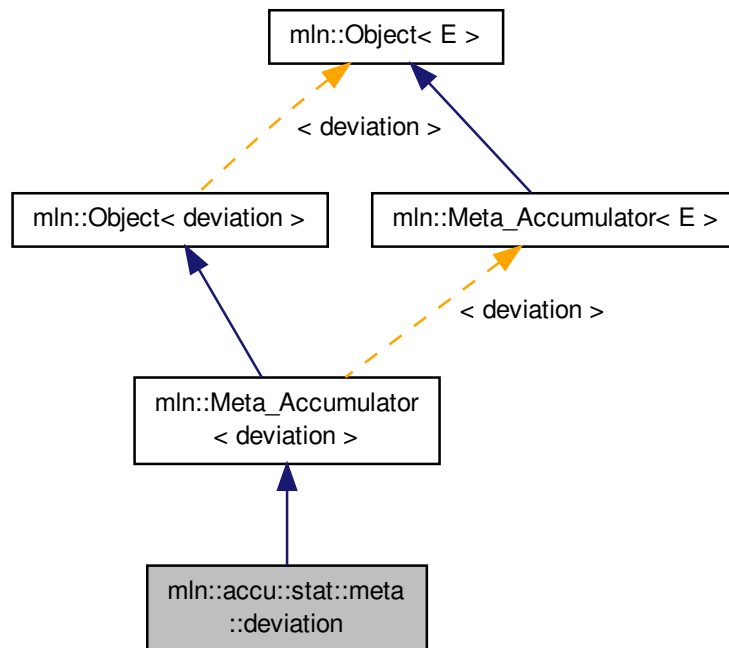
Definition at line 281 of file median_h.hh.

10.84 mln::accu::stat::meta::deviation Struct Reference

Meta accumulator for deviation.

```
#include <deviation.hh>
```

Inheritance diagram for mln::accu::stat::meta::deviation:



10.84.1 Detailed Description

Meta accumulator for deviation.

Definition at line 105 of file deviation.hh.

10.85 mln::accu::stat::min< T > Struct Template Reference

Generic min accumulator class.

```
#include <min.hh>
```

Inherits mln::accu::internal::base< const T &, min< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [set_value](#) (const T &t)
Force the value of the min to t.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- const T & [to_result](#) () const

Get the value of the accumulator.

- void `init()`

Manipulators.

10.85.1 Detailed Description

```
template<typename T> struct mln::accu::stat::min< T >
```

Generic min accumulator class.

The parameter `T` is the type of values.

Definition at line 102 of file `accu/stat/min.hh`.

10.85.2 Member Function Documentation

10.85.2.1 `template<typename T> void min< T >::init()` `[inline]`

Manipulators.

Definition at line 147 of file `accu/stat/min.hh`.

10.85.2.2 `template<typename T> bool min< T >::is_valid()` `const` `[inline]`

Check whether this `accu` is able to return a result.

Always true here.

Definition at line 195 of file `accu/stat/min.hh`.

10.85.2.3 `template<typename T> void min< T >::set_value(const T & t)` `[inline]`

Force the value of the `min` to `t`.

Definition at line 179 of file `accu/stat/min.hh`.

10.85.2.4 `void mln::Accumulator< min< T > >::take_as_init(const T & t)` `[inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.85.2.5 `void mln::Accumulator< min< T > >::take_n_times(unsigned n, const T & t)` `[inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.85.2.6 `template<typename T> const T & min< T >::to_result()` `const` `[inline]`

Get the value of the accumulator.

Definition at line 187 of file `accu/stat/min.hh`.

10.86 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 &, min_h< V > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T & t)
Take as initialization the value t .
- void [take_n_times](#) (unsigned n , const T & t)
Take n times the value t .
- const argument & [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.86.1 Detailed Description

```
template<typename V>struct mln::accu::stat::min_h< V >
```

Generic min function based on histogram over a value set with type V .

Definition at line 100 of file min_h.hh.

10.86.2 Member Function Documentation

10.86.2.1 template<typename V > void min_h< V >::init () [inline]

Manipulators.

Definition at line 256 of file min_h.hh.

10.86.2.2 template<typename V > bool min_h< V >::is_valid () const [inline]

Check whether this accu is able to return a result.

Always true here.

Definition at line 298 of file min_h.hh.

10.86.2.3 void mln::Accumulator< min_h< V > >::take_as_init (const T & t) [inherited]

Take as initialization the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.86.2.4 void mln::Accumulator< min_h< V > >::take_n_times (unsigned n , const T & t) [inherited]

Take n times the value t .

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.86.2.5 `template<typename V> const min_h< V >::argument & min_h< V >::to_result () const` `[inline]`

Get the value of the accumulator.

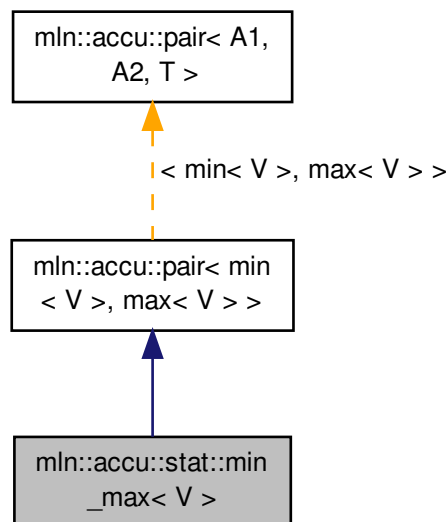
Definition at line 280 of file min_h.hh.

10.87 mln::accu::stat::min_max< V > Struct Template Reference

Generic min and max accumulator class.

```
#include <min_max.hh>
```

Inheritance diagram for mln::accu::stat::min_max< V >:



Public Member Functions

- `min< V >::result first () const`
Return the result of the first accumulator.
- `min< V > first_accu () const`
Return the first accumulator.
- `bool is_valid () const`
Check whether this accu is able to return a result.
- `max< V >::result second () const`
Return the result of the second accumulator.
- `max< V > second_accu () const`
Return the second accumulator.
- `template<typename T>`
`void take_as_init (const T &t)`
Take as initialization the value t.
- `template<typename T>`
`void take_n_times (unsigned n, const T &t)`

Take n times the value t .

- void `init()`

Manipulators.

- `std::pair< typename min< V > ::result, typename max< V > ::result > to_result()` const

Get the value of the accumulator.

10.87.1 Detailed Description

```
template<typename V>struct mln::accu::stat::min_max< V >
```

Generic min and max accumulator class.

The parameter V is the type of values.

Definition at line 61 of file `accu/stat/min_max.hh`.

10.87.2 Member Function Documentation

10.87.2.1 `min< V > ::result mln::accu::pair< min< V > , max< V > , mln_argument(min< V >) > ::first()` const
[inherited]

Return the result of the first accumulator.

10.87.2.2 `min< V > mln::accu::pair< min< V > , max< V > , mln_argument(min< V >) > ::first_accu()` const
[inherited]

Return the first accumulator.

10.87.2.3 `void mln::accu::pair< min< V > , max< V > , mln_argument(min< V >) > ::init()` [inherited]

Manipulators.

10.87.2.4 `bool mln::accu::pair< min< V > , max< V > , mln_argument(min< V >) > ::is_valid()` const
[inherited]

Check whether this accu is able to return a result.

Always true here.

10.87.2.5 `max< V > ::result mln::accu::pair< min< V > , max< V > , mln_argument(min< V >) > ::second()` const
[inherited]

Return the result of the second accumulator.

10.87.2.6 `max< V > mln::accu::pair< min< V > , max< V > , mln_argument(min< V >) > ::second_accu()` const
[inherited]

Return the second accumulator.

10.87.2.7 `template<typename E> template<typename T> void Accumulator< E >::take_as_init (const T & t)`
`[inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 186 of file `accumulator.hh`.

References `mln::mln_exact()`.

10.87.2.8 `template<typename E> template<typename T> void Accumulator< E >::take_n_times (unsigned n, const T & t)`
`[inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

Definition at line 213 of file `accumulator.hh`.

References `mln::mln_exact()`.

10.87.2.9 `std::pair<typename min< V > ::result, typename max< V > ::result> mln::accu::pair< min< V > , max< V > , mln_argument(min< V >)>::to_result () const` `[inherited]`

Get the value of the accumulator.

10.88 mln::accu::stat::rank< T > Struct Template Reference

Generic rank accumulator class.

`#include <rank.hh>`

Inherits `mln::accu::internal::base< const T &, rank< T > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this `accu` is able to return a result.
- `unsigned k () const`
Give the rank.
- `void take_as_init (const T &t)`
Take as initialization the value `t`.
- `void take_n_times (unsigned n, const T &t)`
Take `n` times the value `t`.
- `const T & to_result () const`
Get the value of the accumulator.
- `void init ()`
Manipulators.

10.88.1 Detailed Description

`template<typename T> struct mln::accu::stat::rank< T >`

Generic rank accumulator class.

The parameter `T` is the type of values.

Definition at line 60 of file rank.hh.

10.88.2 Member Function Documentation

10.88.2.1 `template<typename T> void mln::accu::stat::rank< T >::init () [inline]`

Manipulators.

Definition at line 319 of file rank.hh.

References `mln::accu::stat::rank< T >::init()`.

Referenced by `mln::accu::stat::rank< T >::init()`.

10.88.2.2 `template<typename T> bool mln::accu::stat::rank< T >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 343 of file rank.hh.

10.88.2.3 `template<typename T> unsigned mln::accu::stat::rank< T >::k () const [inline]`

Give the rank.

Definition at line 178 of file rank.hh.

10.88.2.4 `void mln::Accumulator< rank< T > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.88.2.5 `void mln::Accumulator< rank< T > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.88.2.6 `template<typename T> const T & mln::accu::stat::rank< T >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 333 of file rank.hh.

10.89 mln::accu::stat::rank< bool > Struct Template Reference

rank accumulator class for Boolean.

```
#include <rank_bool.hh>
```

Inherits `mln::accu::internal::base< bool, rank< bool > >`.

Public Member Functions

- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
Take as initialization the value t.
- void `take_n_times` (unsigned n, const T &t)
Take n times the value t.
- bool `to_result` () const
Get the value of the accumulator.

- void `init` ()
Manipulators.

10.89.1 Detailed Description

`template<> struct mln::accu::stat::rank< bool >`

rank accumulator class for Boolean.

Definition at line 58 of file rank_bool.hh.

10.89.2 Member Function Documentation

10.89.2.1 `void mln::accu::stat::rank< bool >::init () [inline]`

Manipulators.

Definition at line 105 of file rank_bool.hh.

10.89.2.2 `bool mln::accu::stat::rank< bool >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 157 of file rank_bool.hh.

10.89.2.3 `void mln::Accumulator< rank< bool > >::take_as_init (const T &t) [inherited]`

Take as initialization the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.89.2.4 `void mln::Accumulator< rank< bool > >::take_n_times (unsigned n, const T &t) [inherited]`

Take n times the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.89.2.5 `bool mln::accu::stat::rank< bool >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 150 of file rank_bool.hh.

10.90 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 &, rank_high_quant< T > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- const T & [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.90.1 Detailed Description

```
template<typename T>struct mln::accu::stat::rank_high_quant< T >
```

Generic rank accumulator class.

The parameter T is the type of values.

Definition at line 57 of file rank_high_quant.hh.

10.90.2 Member Function Documentation

10.90.2.1 `template<typename T> void mln::accu::stat::rank_high_quant< T >::init () [inline]`

Manipulators.

Definition at line 148 of file rank_high_quant.hh.

10.90.2.2 `template<typename T> bool mln::accu::stat::rank_high_quant< T >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 197 of file rank_high_quant.hh.

10.90.2.3 `void mln::Accumulator< rank_high_quant< T > >::take_as_init (const T & t) [inherited]`

Take as initialization the value t.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.90.2.4 `void mln::Accumulator< rank_high_quant< T > >::take_n_times (unsigned n, const T & t)`
`[inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.90.2.5 `template<typename T> const T & mln::accu::stat::rank_high_quant< T >::to_result () const`
`[inline]`

Get the value of the accumulator.

Definition at line 183 of file `rank_high_quant.hh`.

10.91 mln::accu::stat::var< T > Struct Template Reference

Var accumulator class.

`#include <var.hh>`

Inherits `mln::accu::internal::base< algebra::mat< T::dim, T::dim, float >, var< T > >`.

Public Types

- typedef `algebra::vec< dim, float >` `mean_t`
Type equipment.

Public Member Functions

- bool `is_valid ()` const
Check whether this accu returns a valid result.
- `mean_t mean ()` const
Get the mean vector.
- unsigned `n_items ()` const
Get the number of items.
- void `take_as_init (const T &t)`
Take as initialization the value t .
- void `take_n_times (unsigned n, const T &t)`
Take n times the value t .
- result `to_result ()` const
Get the accumulator result (the var value).
- result `variance ()` const
Get the variance matrix.
- void `init ()`
Manipulators.

10.91.1 Detailed Description

`template<typename T> struct mln::accu::stat::var< T >`

Var accumulator class.

Parameter `T` is the type of vectors

Definition at line 58 of file `accu/stat/var.hh`.

10.91.2 Member Typedef Documentation

10.91.2.1 `template<typename T> typedef algebra::vec<dim,float> mln::accu::stat::var< T >::mean_t`

Type equipment.

Definition at line 88 of file `accu/stat/var.hh`.

10.91.3 Member Function Documentation

10.91.3.1 `template<typename T> void mln::accu::stat::var< T >::init() [inline]`

Manipulators.

Definition at line 118 of file `accu/stat/var.hh`.

10.91.3.2 `template<typename T> bool mln::accu::stat::var< T >::is_valid() const [inline]`

Check whether this `accu` returns a valid result.

Definition at line 213 of file `accu/stat/var.hh`.

10.91.3.3 `template<typename T> var< T >::mean_t mln::accu::stat::var< T >::mean() const [inline]`

Get the mean vector.

Definition at line 200 of file `accu/stat/var.hh`.

References `mln::literal::zero`.

10.91.3.4 `template<typename T> unsigned mln::accu::stat::var< T >::n_items() const [inline]`

Get the number of items.

Definition at line 192 of file `accu/stat/var.hh`.

10.91.3.5 `void mln::Accumulator< var< T > >::take_as_init(const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.91.3.6 `void mln::Accumulator< var< T > >::take_n_times(unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.91.3.7 `template<typename T> var< T >::result mln::accu::stat::var< T >::to_result() const [inline]`

Get the accumulator result (the `var` value).

Definition at line 168 of file `accu/stat/var.hh`.

References `mln::literal::zero`.

10.91.3.8 `template<typename T> var< T >::result mln::accu::stat::var< T >::variance () const` `[inline]`

Get the variance matrix.

Definition at line 184 of file `accu/stat/var.hh`.

10.92 `mln::accu::stat::variance< T, S, R >` Struct Template Reference

Variance accumulator class.

`#include <variance.hh>`

Inherits `mln::accu::internal::base< R, variance< T, S, R > >`.

Public Member Functions

- `bool is_valid () const`
Check whether this accu is able to return a result.
- `R mean () const`
Get the mean value.
- `unsigned n_items () const`
Get the number of items.
- `R standard_deviation () const`
Get the standard deviation value.
- `S sum () const`
Get the sum value.
- `void take_n_times (unsigned n, const T &t)`
Take n times the value t .
- `R to_result () const`
Get the accumulator result (the variance value).
- `R var () const`
Get the variance value.
- `void init ()`
Manipulators.

10.92.1 Detailed Description

`template<typename T, typename S = typename mln::value::props< T >::sum, typename R = S> struct mln::accu::stat::variance< T, S, R >`

Variance accumulator class.

Parameter `T` is the type of values that we sum. Parameter `S` is the type to store the value sum and the sum of value

- `value`; the default type of `S` is the summation type (property) of `T`. Parameter `R` is the type of the mean and variance values; the default type of `R` is `S`.

Definition at line 61 of file `variance.hh`.

10.92.2 Member Function Documentation

10.92.2.1 `template<typename T, typename S, typename R> void mln::accu::stat::variance< T, S, R >::init ()`
`[inline]`

Manipulators.

Definition at line 125 of file variance.hh.

References `mln::literal::zero`.

10.92.2.2 `template<typename T, typename S, typename R> bool mln::accu::stat::variance< T, S, R >::is_valid ()`
`const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 229 of file variance.hh.

10.92.2.3 `template<typename T, typename S, typename R> R mln::accu::stat::variance< T, S, R >::mean () const`
`[inline]`

Get the mean value.

Definition at line 187 of file variance.hh.

References `mln::literal::zero`.

10.92.2.4 `template<typename T, typename S, typename R> unsigned mln::accu::stat::variance< T, S, R >::n_items () const`
`[inline]`

Get the number of items.

Definition at line 205 of file variance.hh.

10.92.2.5 `template<typename T, typename S, typename R> R mln::accu::stat::variance< T, S, R >::standard_deviation () const`
`[inline]`

Get the standard deviation value.

Definition at line 221 of file variance.hh.

10.92.2.6 `template<typename T, typename S, typename R> S mln::accu::stat::variance< T, S, R >::sum () const`
`[inline]`

Get the sum value.

Definition at line 197 of file variance.hh.

10.92.2.7 `void mln::Accumulator< variance< T, S, R > >::take_n_times (unsigned n, const T & t)` `[inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.92.2.8 `template<typename T , typename S , typename R > R mln::accu::stat::variance< T, S, R >::to_result () const`
`[inline]`

Get the accumulator result (the variance value).

Definition at line 176 of file variance.hh.

10.92.2.9 `template<typename T , typename S , typename R > R mln::accu::stat::variance< T, S, R >::var () const`
`[inline]`

Get the variance value.

Definition at line 213 of file variance.hh.

10.93 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!) >, tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >.

Public Member Functions

- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
Take as initialization the value t.
- void `take_n_times` (unsigned n, const T &t)
Take n times the value t.
- res `to_result` () const
Get the value of the accumulator.
- void `init` ()
Manipulators.

10.93.1 Detailed Description

`template<typename A, unsigned n, BOOST_PP_ENUM_PARAMS_WITH_A_DEFAULT(10, typename T, boost::tuples::null_type)> struct mln::accu::tuple< A, n, >`

Generic tuple of accumulators.

The parameter `T` is the type of values.

Definition at line 74 of file tuple.hh.

10.93.2 Member Function Documentation

10.93.2.1 `template<typename A , unsigned n, BOOST_PP_ENUM_PARAMS(10, typename T) > void mln::accu::tuple< A, n, >::init ()` `[inline]`

Manipulators.

Definition at line 197 of file tuple.hh.


```
10.93.2.2  template<typename A , unsigned n, BOOST_PP_ENUM_PARAMS(10, typename T) > bool mln::accu::tuple< A, n,
>::is_valid ( ) const  [inline]
```

Check whether this accu is able to return a result.

Always true here.

Definition at line 239 of file tuple.hh.

```
10.93.2.3  void mln::Accumulator< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >::take_as_init ( const T & t )
[inherited]
```

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

```
10.93.2.4  void mln::Accumulator< tuple< A, n, BOOST_PP_ENUM_PARAMS(10, T)> >::take_n_times ( unsigned n, const
T & t )  [inherited]
```

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

```
10.93.2.5  template<typename A , unsigned n, BOOST_PP_ENUM_PARAMS(10, typename T) > tuple< A, n,
BOOST_PP_ENUM_PARAMS(10, T) >::res mln::accu::tuple< A, n, >::to_result ( ) const  [inline]
```

Get the value of the accumulator.

Definition at line 229 of file tuple.hh.

10.94 mln::accu::val< A > Struct Template Reference

Generic val of accumulators.

```
#include <v.hh>
```

Inherits `mln::accu::internal::base< const A::result &, val< A > >`.

Public Member Functions

- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
*Take as initialization the value *t*.*
- void `take_n_times` (unsigned n, const T &t)
*Take *n* times the value *t*.*
- const A::result & `to_result` () const
Get the value of the accumulator.
- void `init` ()
Manipulators.

10.94.1 Detailed Description

```
template<typename A>struct mln::accu::val< A >
```

Generic val of accumulators.

Definition at line 50 of file v.hh.

10.94.2 Member Function Documentation

10.94.2.1 `template<typename A> void mln::accu::val< A >::init () [inline]`

Manipulators.

Definition at line 119 of file v.hh.

10.94.2.2 `template<typename A> bool mln::accu::val< A >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 177 of file v.hh.

10.94.2.3 `void mln::Accumulator< val< A > >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.94.2.4 `void mln::Accumulator< val< A > >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.94.2.5 `template<typename A> const A::result & mln::accu::val< A >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 169 of file v.hh.

10.95 mln::Accumulator< E > Struct Template Reference

Base class for implementation of accumulators.

```
#include <accumulator.hh>
```

[illegible]

- `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 .

10.95.1 Detailed Description

`template<typename E> struct mln::Accumulator< E >`

Base class for implementation of accumulators.

The parameter E is the exact type.

See Also

[mln::doc::Accumulator](#) for a complete documentation of this class contents.

Definition at line 78 of file accumulator.hh.

10.95.2 Member Function Documentation

10.95.2.1 `template<typename E > template<typename T > void Accumulator< E >::take_as_init (const T & t)`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

Definition at line 186 of file accumulator.hh.

References `mln::mln_exact()`.

10.95.2.2 `template<typename E > template<typename T > void Accumulator< E >::take_n.times (unsigned n, const T & t)`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

Definition at line 213 of file accumulator.hh.

References `mln::mln_exact()`.

10.96 mln::algebra::h_mat< d, T > Struct Template Reference

N-Dimensional matrix with homogeneous coordinates.

`#include <h_mat.hh>`

Inherits `mln::algebra::mat< n, m, T >`.

Public Types

- enum
 - Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1)*

Public Member Functions

- `mat< n, m, T > _1 () const`
 - Return the inverse of the matrix.*
- [h_mat \(\)](#)

Constructor without argument.

- `h_mat` (const mat< d+1, d+1, T > &x)

Constructor with the underlying matrix.

- `mat< m, n, T > t () const`

Return the transpose of the matrix.

10.96.1 Detailed Description

```
template<unsigned d, typename T>struct mln::algebra::h_mat< d, T >
```

N-Dimensional matrix with homogeneous coordinates.

Definition at line 49 of file algebra/h_mat.hh.

10.96.2 Member Enumeration Documentation

10.96.2.1 `template<unsigned d, typename T> anonymous enum`

Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1)

Definition at line 52 of file algebra/h_mat.hh.

10.96.3 Constructor & Destructor Documentation

10.96.3.1 `template<unsigned d, typename T > mln::algebra::h_mat< d, T >::h_mat () [inline]`

Constructor without argument.

Definition at line 67 of file algebra/h_mat.hh.

10.96.3.2 `template<unsigned d, typename T > mln::algebra::h_mat< d, T >::h_mat (const mat< d+1, d+1, T > & x) [inline]`

Constructor with the underlying matrix.

Definition at line 74 of file algebra/h_mat.hh.

10.96.4 Member Function Documentation

10.96.4.1 `template<unsigned n, unsigned m, typename T > mat< n, m, T > mat< n, m, T >::1 () const [inline], [inherited]`

Return the inverse of the matrix.

Only compile on square matrix.

Definition at line 604 of file algebra/mat.hh.

10.96.4.2 `template<unsigned n, unsigned m, typename T > mat< m, n, T > mat< n, m, T >::t () const [inline], [inherited]`

Return the transpose of the matrix.

Definition at line 538 of file algebra/mat.hh.

10.97 mln::algebra::h_vec< d, C > Class Template Reference

N-Dimensional vector with homogeneous coordinates.

```
#include <h_vec.hh>
```

Inherits mln::algebra::vec< n, T >.

Public Types

- enum
 - Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).*

Public Member Functions

- [h_vec](#) ()
 - Constructor without argument.*
- [h_vec](#) (const vec< d+1, C > &other)
 - Constructor with the underlying vector.*
- template<typename U >
 operator mat< n, 1, U > () const
 - Conversion to a matrix.*
- mat< 1, n, T > t** () const
 - Transposition.*
- vec< d, C > [to_vec](#) () const
 - Back to the natural (non-homogeneous) space.*

Static Public Attributes

- static const vec< n, T > **origin** = all_to(0)
 - Origin value.*
- static const vec< n, T > **zero** = all_to(0)
 - Zero value.*

10.97.1 Detailed Description

```
template<unsigned d, typename C>class mln::algebra::h_vec< d, C >
```

N-Dimensional vector with homogeneous coordinates.

Definition at line 94 of file h_vec.hh.

10.97.2 Member Enumeration Documentation

10.97.2.1 template<unsigned d, typename C> anonymous enum

Dimension is the 'natural' one (3 for 3D), not the one of the vector (dim + 1).

Definition at line 98 of file h_vec.hh.

10.97.3 Constructor & Destructor Documentation

10.97.3.1 `template<unsigned d, typename C> h_vec< d, C >::h_vec () [inline]`

Constructor without argument.

Definition at line 117 of file h_vec.hh.

References mln::literal::one.

10.97.3.2 `template<unsigned d, typename C> h_vec< d, C >::h_vec (const vec< d+1, C> & other) [inline]`

Constructor with the underlying vector.

Definition at line 128 of file h_vec.hh.

10.97.4 Member Function Documentation

10.97.4.1 `template<unsigned n, typename T> template<typename U> vec< n, T>::operator mat< n, 1, U> () const [inline], [inherited]`

Conversion to a matrix.

Definition at line 368 of file algebra/mat.hh.

10.97.4.2 `template<unsigned n, typename T> mat< 1, n, T> vec< n, T>::t () const [inline], [inherited]`

Transposition.

Definition at line 795 of file mat_base.hh.

10.97.4.3 `template<unsigned d, typename C> vec< d, C> h_vec< d, C>::to_vec () const [inline]`

Back to the natural (non-homogeneous) space.

Definition at line 145 of file h_vec.hh.

10.97.5 Member Data Documentation

10.97.5.1 `template<unsigned n, typename T> const vec< n, T> vec< n, T>::origin = all.to(0) [static], [inherited]`

Origin value.

Definition at line 248 of file algebra/vec.hh.

10.97.5.2 `template<unsigned n, typename T> const vec< n, T> vec< n, T>::zero = all.to(0) [static], [inherited]`

Zero value.

Definition at line 245 of file algebra/vec.hh.

10.98 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, 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.

10.98.1 Detailed Description

```
template<typename I>class mln::bkd_pixter1d< I >
```

Backward pixel iterator on a 1-D image with border.

Definition at line 69 of file `pixter1d.hh`.

10.98.2 Member Typedef Documentation

10.98.2.1 `template<typename I > typedef I mln::bkd_pixter1d< I >::image`

`Image` type.

Definition at line 76 of file `pixter1d.hh`.

10.98.3 Constructor & Destructor Documentation

10.98.3.1 `template<typename I > mln::bkd_pixter1d< I >::bkd_pixter1d (I & image)` `[inline]`

Constructor.

Parameters

<code>in</code>	<code>image</code>	The image this pixel iterator is bound to.
-----------------	--------------------	--

Definition at line 117 of file `pixter1d.hh`.

10.98.4 Member Function Documentation

10.98.4.1 `void mln::iterator< bkd_pixter1d< I > >::next ()` `[inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.99 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, 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.

10.99.1 Detailed Description

```
template<typename I>class mln::bkd_pixter2d< I >
```

Backward pixel iterator on a 2-D image with border.

Definition at line 87 of file pixter2d.hh.

10.99.2 Member Typedef Documentation

10.99.2.1 `template<typename I > typedef I mln::bkd_pixter2d< I >::image`

[Image](#) type.

Definition at line 94 of file pixter2d.hh.

10.99.3 Constructor & Destructor Documentation

10.99.3.1 `template<typename I > mln::bkd_pixter2d< I >::bkd_pixter2d (I & image) [inline]`

Constructor.

Parameters

<i>in</i>	<i>image</i>	The image this pixel iterator is bound to.
-----------	--------------	--

Definition at line 169 of file pixter2d.hh.

10.99.4 Member Function Documentation

10.99.4.1 `void mln::iterator< bkd_pixter2d< I > >::next ()` [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.100 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, 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.

10.100.1 Detailed Description

```
template<typename I>class mln::bkd_pixter3d< I >
```

Backward pixel iterator on a 3-D image with border.

Definition at line 100 of file pixter3d.hh.

10.100.2 Member Typedef Documentation

10.100.2.1 `template<typename I > typedef I mln::bkd_pixter3d< I >::image`

[Image](#) type.

Definition at line 107 of file pixter3d.hh.

10.100.3 Constructor & Destructor Documentation

10.100.3.1 `template<typename I > mln::bkd_pixter3d< I >::bkd_pixter3d (I & image)` [inline]

Constructor.

Parameters

<code>in</code>	<code>image</code>	The image this pixel iterator is bound to.
-----------------	--------------------	--

Definition at line 207 of file pixter3d.hh.

10.100.4 Member Function Documentation

10.100.4.1 `void mln::iterator< bkd_pixter3d< I > >::next ()` [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

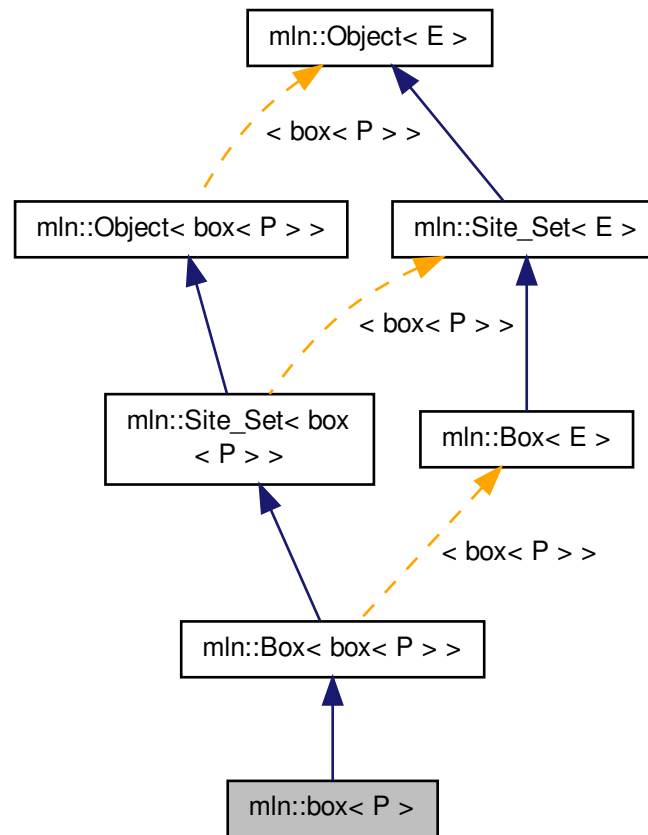
The iterator is valid.

10.101 mln::box< P > Class Template Reference

Generic box class: site set containing points of a regular grid.

```
#include <box.hh>
```

Inheritance diagram for `mln::box< P >`:



Public Types

- enum
Dimension.
- typedef `box_bkd_piter_< P >` `bkd_piter`
Backward [Site_Iterator](#) associated type.
- typedef `P` `element`
Element associated type.
- typedef `box_fwd_piter_< P >` `fwd_piter`
Forward [Site_Iterator](#) associated type.
- typedef `fwd_piter` `piter`
[Site_Iterator](#) associated type.
- typedef `P` `psite`
Psite associated type.
- typedef `P` `site`
[Site](#) associated type.

Public Member Functions

- const `box< P > & bbox ()` const
Give the bounding box of this site set.
- `box ()`
Constructor without argument.
- `box (const site &pmin, const site &pmax)`
Constructor of a box going from `pmin` to `pmax`.
- void `crop_wrt (const box< P > &b)`
Crop this bbox in order to fit in the reference box `b`.
- void `enlarge (unsigned b)`
Enlarge the box with a border `b`.
- void `enlarge (unsigned dim, unsigned b)`
Enlarge the box with a border `b` for dimension `dim`.
- bool `has (const P &p)` const
Test if `p` belongs to the box.
- bool `is_empty ()` const
Test if this box is empty.
- bool `is_valid ()` const
Test that the box owns valid data, i.e., is initialized and with `pmin` being 'less-than' `pmax`.
- unsigned `len (unsigned i)` const
Give the length of the `i`-th side of the box.
- `std::size_t memory_size ()` const
Return the size of this site set in memory.
- void `merge (const box< P > &b)`
Merge inplace with another box.
- unsigned `nsites ()` const
Give the number of sites of this box.
- `P pcenter ()` const
Return the approximated central site of this box.
- `P pmax ()` const
Maximum point.
- `P & pmax ()`
Reference to the maximum point.
- `P pmin ()` const
Minimum point.
- `P & pmin ()`
Reference to the minimum point.
- `box< P > to_larger (unsigned b)` const
Give a larger box.
- `box (typename P::coord ninds)`

Related Functions

(Note that these are not member functions.)

- `template<typename P >`
`std::ostream & operator<< (std::ostream &ostr, const box< P > &b)`
Print a generic box `b` into the output stream `ostr`.

10.101.1 Detailed Description

`template<typename P> class mln::box< P >`

Generic box class: site set containing points of a regular grid.

Parameter P is the corresponding type of point.

Definition at line 81 of file core/site_set/box.hh.

10.101.2 Member Typedef Documentation

10.101.2.1 `template<typename P> typedef box_bkd_piter_<P> mln::box< P >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 105 of file core/site_set/box.hh.

10.101.2.2 `template<typename P> typedef P mln::box< P >::element`

Element associated type.

Definition at line 90 of file core/site_set/box.hh.

10.101.2.3 `template<typename P> typedef box_fwd_piter_<P> mln::box< P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 99 of file core/site_set/box.hh.

10.101.2.4 `template<typename P> typedef fwd_piter mln::box< P >::piter`

[Site_Iterator](#) associated type.

Definition at line 102 of file core/site_set/box.hh.

10.101.2.5 `template<typename P> typedef P mln::box< P >::psite`

Psite associated type.

Definition at line 93 of file core/site_set/box.hh.

10.101.2.6 `template<typename P> typedef P mln::box< P >::site`

[Site](#) associated type.

Definition at line 96 of file core/site_set/box.hh.

10.101.3 Member Enumeration Documentation

10.101.3.1 `template<typename P> anonymous enum`

Dimension.

Definition at line 87 of file core/site_set/box.hh.

10.101.4 Constructor & Destructor Documentation

10.101.4.1 `template<typename P> box< P >::box () [inline]`

Constructor without argument.

Definition at line 275 of file core/site_set/box.hh.

10.101.4.2 `template<typename P> box< P >::box (const site & pmin, const site & pmax) [inline]`

Constructor of a box going from pmin to pmax.

Definition at line 284 of file core/site_set/box.hh.

References mln::box< P >::is_valid().

10.101.4.3 `template<typename P> box< P >::box (typename P::coord ninds) [inline], [explicit]`

Constructors with different numbers of arguments (sizes) w.r.t. the dimension.

Definition at line 293 of file core/site_set/box.hh.

References mln::literal::origin.

10.101.5 Member Function Documentation

10.101.5.1 `const box< P > & mln::Box< box< P > >::bbox () const [inherited]`

Give the bounding box of this site set.

Return the bounding box of this site set, so that is itself. This method is declared by the [mln::Site_Set](#) concept.

Warning

This method is final for all box classes.

10.101.5.2 `template<typename P> void box< P >::crop_wrt (const box< P > & b) [inline]`

Crop this bbox in order to fit in the reference box b.

Definition at line 205 of file core/site_set/box.hh.

References mln::box< P >::pmax(), and mln::box< P >::pmin().

Referenced by mln::debug::draw_graph(), and mln::make_debug_graph_image().

10.101.5.3 `template<typename P> void box< P >::enlarge (unsigned b) [inline]`

Enlarge the box with a border b.

Definition at line 337 of file core/site_set/box.hh.

Referenced by mln::registration::icp().

10.101.5.4 `template<typename P> void box< P >::enlarge (unsigned dim, unsigned b) [inline]`

Enlarge the box with a border b for dimension dim.

Definition at line 351 of file core/site_set/box.hh.

10.101.5.5 `template<typename P> bool box< P >::has (const P & p) const` `[inline]`

Test if `p` belongs to the box.

Parameters

<code>in</code>	<code>p</code>	A point site.
-----------------	----------------	---------------

Definition at line 325 of file `core/site_set/box.hh`.

10.101.5.6 `bool mln::Box< box< P > >::is_empty () const` `[inherited]`

Test if this box is empty.

10.101.5.7 `template<typename P> bool box< P >::is_valid () const` `[inline]`

Test that the box owns valid data, i.e., is initialized and with `pmin` being 'less-than' `pmax`.

Definition at line 195 of file `core/site_set/box.hh`.

References `mln::util::ord_weak()`.

Referenced by `mln::box< P >::box()`, `mln::transform::distance_and_closest_point_geodesic()`, `mln::box< P >::merge()`, and `mln::box< P >::to_larger()`.

10.101.5.8 `unsigned mln::Box< box< P > >::len (unsigned i) const` `[inherited]`

Give the length of the `i`-th side of the box.

Precondition

`i < site::dim`

Warning

This method is final for all box classes.

10.101.5.9 `template<typename P> std::size_t box< P >::memory_size () const` `[inline]`

Return the size of this site set in memory.

Definition at line 407 of file `core/site_set/box.hh`.

10.101.5.10 `template<typename P> void box< P >::merge (const box< P > & b)` `[inline]`

Merge inplace with another box.

Definition at line 221 of file `core/site_set/box.hh`.

References `mln::box< P >::is_valid()`, `mln::box< P >::pmax()`, and `mln::box< P >::pmin()`.

10.101.5.11 `unsigned mln::Box< box< P > >::nsites () const` `[inherited]`

Give the number of sites of this box.

Return the number of sites of this box. This method is declared by the [mln::Site_Set](#) concept.

Warning

This method is final for all box classes.

10.101.5.12 `template<typename P> P box< P >::pcenter () const [inline]`

Return the approximated central site of this box.

Definition at line 395 of file core/site_set/box.hh.

10.101.5.13 `template<typename P> P box< P >::pmax () const [inline]`

Maximum point.

Definition at line 259 of file core/site_set/box.hh.

Referenced by mln::box< P >::crop_wrt(), mln::make::image3d(), mln::larger_than(), mln::io::fld::load(), and mln::box< P >::merge().

10.101.5.14 `template<typename P> P & box< P >::pmax () [inline]`

Reference to the maximum point.

Definition at line 268 of file core/site_set/box.hh.

10.101.5.15 `template<typename P> P box< P >::pmin () const [inline]`

Minimum point.

Definition at line 242 of file core/site_set/box.hh.

Referenced by mln::box< P >::crop_wrt(), mln::make::image3d(), mln::larger_than(), mln::io::fld::load(), and mln::box< P >::merge().

10.101.5.16 `template<typename P> P & box< P >::pmin () [inline]`

Reference to the minimum point.

Definition at line 251 of file core/site_set/box.hh.

10.101.5.17 `template<typename P> box< P > box< P >::to_larger (unsigned b) const [inline]`

Give a larger box.

Definition at line 378 of file core/site_set/box.hh.

References mln::box< P >::is_valid().

10.101.6 Friends And Related Function Documentation

10.101.6.1 `template<typename P> std::ostream & operator<< (std::ostream & ostr, const box< P > & b) [related]`

Print a generic box *b* into the output stream *ostr*.

Parameters

<code>in, out</code>	<code>ostr</code>	An output stream.
<code>in</code>	<code>b</code>	A generic box.

Returns

The modified output stream `ostr`.

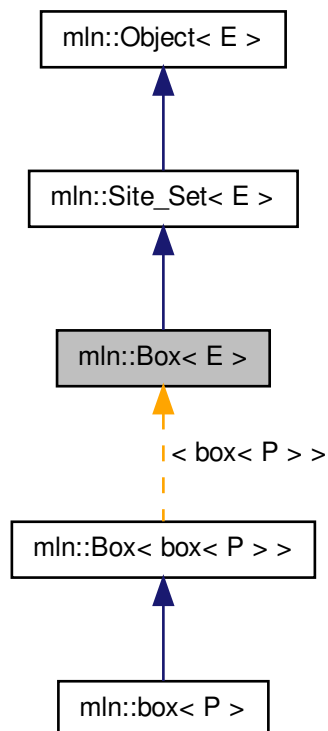
Definition at line 414 of file `core/site_set/box.hh`.

10.102 mln::Box< E > Struct Template Reference

Base class for implementation classes of boxes.

```
#include <box.hh>
```

Inheritance diagram for `mln::Box< E >`:



Public Member Functions

- `const E & bbox () const`
Give the bounding box of this site set.
- `bool is_empty () const`
Test if this box is empty.

- unsigned [len](#) (unsigned i) const
Give the length of the i -th side of the box.
- unsigned [nsites](#) () const
Give the number of sites of this box.

Related Functions

(Note that these are not member functions.)

- template<typename SI, typename Sr >
[p_set](#)< typename SI::site > [diff](#) (const [Site_Set](#)< SI > &lhs, const [Site_Set](#)< Sr > &rhs)
Set theoretic difference of lhs and rhs .
- template<typename SI, typename Sr >
[p_set](#)< typename SI::site > [inter](#) (const [Site_Set](#)< SI > &lhs, const [Site_Set](#)< Sr > &rhs)
Intersection between a couple of point sets.
- template<typename SI, typename Sr >
bool [operator](#)< (const [Site_Set](#)< SI > &lhs, const [Site_Set](#)< Sr > &rhs)
Strict inclusion test between site sets lhs and rhs .
- template<typename BI, typename Br >
bool [operator](#)< (const [Box](#)< BI > &lhs, const [Box](#)< Br > &rhs)
Strict inclusion test between boxes lhs and rhs .
- template<typename S >
std::ostream & [operator](#)<< (std::ostream &ostr, const [Site_Set](#)< S > &set)
Print a site set set into the output stream $ostr$.
- template<typename SI, typename Sr >
bool [operator](#)<= (const [Site_Set](#)< SI > &lhs, const [Site_Set](#)< Sr > &rhs)
Inclusion test between site sets lhs and rhs .
- template<typename BI, typename Br >
bool [operator](#)<= (const [Box](#)< BI > &lhs, const [Box](#)< Br > &rhs)
Inclusion test between boxes lhs and rhs .
- template<typename SI, typename Sr >
bool [operator](#)== (const [Site_Set](#)< SI > &lhs, const [Site_Set](#)< Sr > &rhs)
Equality test between site sets lhs and rhs .
- template<typename SI, typename Sr >
[p_set](#)< typename SI::site > [sym_diff](#) (const [Site_Set](#)< SI > &lhs, const [Site_Set](#)< Sr > &rhs)
Set theoretic symmetrical difference of lhs and rhs .
- template<typename SI, typename Sr >
[p_set](#)< typename SI::site > [uni](#) (const [Site_Set](#)< SI > &lhs, const [Site_Set](#)< Sr > &rhs)
Union of a couple of point sets.
- template<typename S >
[p_set](#)< typename S::site > [unique](#) (const [Site_Set](#)< S > &s)
Give the unique set of s .

10.102.1 Detailed Description

template<typename E>struct mln::Box< E >

Base class for implementation classes of boxes.

Boxes are particular site sets useful to bound any set of sites defined on a regular grid.

See Also

[mln::doc::Box](#) for a complete documentation of this class contents.

Definition at line 48 of file core/concept/box.hh.

10.102.2 Member Function Documentation

10.102.2.1 `template<typename E> const E & mln::Box< E >::bbox () const` `[inline]`

Give the bounding box of this site set.

Return the bounding box of this site set, so that is itself. This method is declared by the [mln::Site_Set](#) concept.

Warning

This method is final for all box classes.

Definition at line 124 of file core/concept/box.hh.

10.102.2.2 `template<typename E> bool mln::Box< E >::is_empty () const` `[inline]`

Test if this box is empty.

Definition at line 168 of file core/concept/box.hh.

10.102.2.3 `template<typename E> unsigned mln::Box< E >::len (unsigned i) const` `[inline]`

Give the length of the *i*-th side of the box.

Precondition

$i < \text{site::dim}$

Warning

This method is final for all box classes.

Definition at line 131 of file core/concept/box.hh.

10.102.2.4 `template<typename E> unsigned mln::Box< E >::nsites () const` `[inline]`

Give the number of sites of this box.

Return the number of sites of this box. This method is declared by the [mln::Site_Set](#) concept.

Warning

This method is final for all box classes.

Definition at line 154 of file core/concept/box.hh.

10.102.3 Friends And Related Function Documentation

10.102.3.1 `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*.

Definition at line 66 of file set/diff.hh.

10.102.3.2 `template<typename SI, typename Sr > p_set< typename SI::site > inter (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [related]

Intersection between a couple of point sets.

Definition at line 62 of file set/inter.hh.

10.102.3.3 `template<typename SI, typename Sr > bool operator< (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [related]

Strict inclusion test between site sets `lhs` and `rhs`.

Parameters

<code>in</code>	<code>lhs</code>	A site set (strictly included?).
<code>in</code>	<code>rhs</code>	Another site set (includer?).

Definition at line 479 of file operators.hh.

10.102.3.4 `template<typename BI, typename Br > bool operator< (const Box< BI > & lhs, const Box< Br > & rhs)` [related]

Strict inclusion test between boxes `lhs` and `rhs`.

Parameters

<code>in</code>	<code>lhs</code>	A box (strictly included?).
<code>in</code>	<code>rhs</code>	Another box (includor?).

Definition at line 194 of file core/concept/box.hh.

10.102.3.5 `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

<code>in, out</code>	<code>ostr</code>	An output stream.
<code>in</code>	<code>set</code>	A site set.

Returns

The modified output stream `ostr`.

Definition at line 505 of file operators.hh.

10.102.3.6 `template<typename SI, typename Sr > bool operator<= (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [related]

Inclusion test between site sets `lhs` and `rhs`.

Parameters

<code>in</code>	<code>lhs</code>	A site set (included?).
<code>in</code>	<code>rhs</code>	Another site set (includer?).

Definition at line 491 of file operators.hh.

10.102.3.7 `template<typename BI, typename Br > bool operator<= (const Box< BI > & lhs, const Box< Br > & rhs)`
[related]

Inclusion test between boxes `lhs` and `rhs`.

Parameters

<code>in</code>	<code>lhs</code>	A box (included?).
<code>in</code>	<code>rhs</code>	Another box (includor?).

Definition at line 179 of file core/concept/box.hh.

10.102.3.8 `template<typename SI, typename Sr > bool operator== (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [related]

Equality test between site sets `lhs` and `rhs`.

Parameters

<code>in</code>	<code>lhs</code>	A site set.
<code>in</code>	<code>rhs</code>	Another site set.

Definition at line 467 of file operators.hh.

10.102.3.9 `template<typename SI, typename Sr > p_set< typename SI::site > sym.diff (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [related]

Set theoretic symmetrical difference of `lhs` and `rhs`.

Definition at line 65 of file sym_diff.hh.

10.102.3.10 `template<typename SI, typename Sr > p_set< typename SI::site > uni (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [related]

Union of a couple of point sets.

Definition at line 61 of file uni.hh.

10.102.3.11 `template<typename S > p_set< typename S::site > unique (const Site_Set< S > & s)` [related]

Give the unique set of `s`.

Definition at line 61 of file unique.hh.

10.103 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< box< P >, box_runend_piter< P > >`.

Public Member Functions

- void [next](#) ()
Go to the next element.
- unsigned [run_length](#) () const
Give the lenght of the run.

10.103.1 Detailed Description

template<typename P>class mln::box_runend_piter< P >

A generic backward iterator on points by lines.

The parameter P is the type of points.

Definition at line 49 of file box_runend_piter.hh.

10.103.2 Member Function Documentation

10.103.2.1 void mln::Site_Iterator< box_runend_piter< P > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.103.2.2 template<typename P > unsigned mln::box_runend_piter< P >::run_length () const [inline]

Give the lenght of the run.

Definition at line 167 of file box_runend_piter.hh.

10.104 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< box< P >, box_runstart_piter< P > >.

Public Member Functions

- [box_runstart_piter](#) (const [box](#)< P > &b)
Constructor.
- void [next](#) ()
Go to the next element.
- unsigned [run_length](#) () const
Give the lenght of the run.

10.104.1 Detailed Description

```
template<typename P>class mln::box_runstart_piter< P >
```

A generic forward iterator on points by lines.

The parameter `P` is the type of points.

Definition at line 49 of file `box_runstart_piter.hh`.

10.104.2 Constructor & Destructor Documentation

10.104.2.1 `template<typename P > mln::box_runstart_piter< P >::box_runstart_piter (const box< P > & b)`
`[inline]`

Constructor.

Parameters

<code>in</code>	<code>b</code>	A box.
-----------------	----------------	--------

Definition at line 105 of file `box_runstart_piter.hh`.

10.104.3 Member Function Documentation

10.104.3.1 `void mln::Site_Itorator< box_runstart_piter< P > >::next ()` `[inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.104.3.2 `template<typename P > unsigned mln::box_runstart_piter< P >::run_length () const` `[inline]`

Give the lenght of the run.

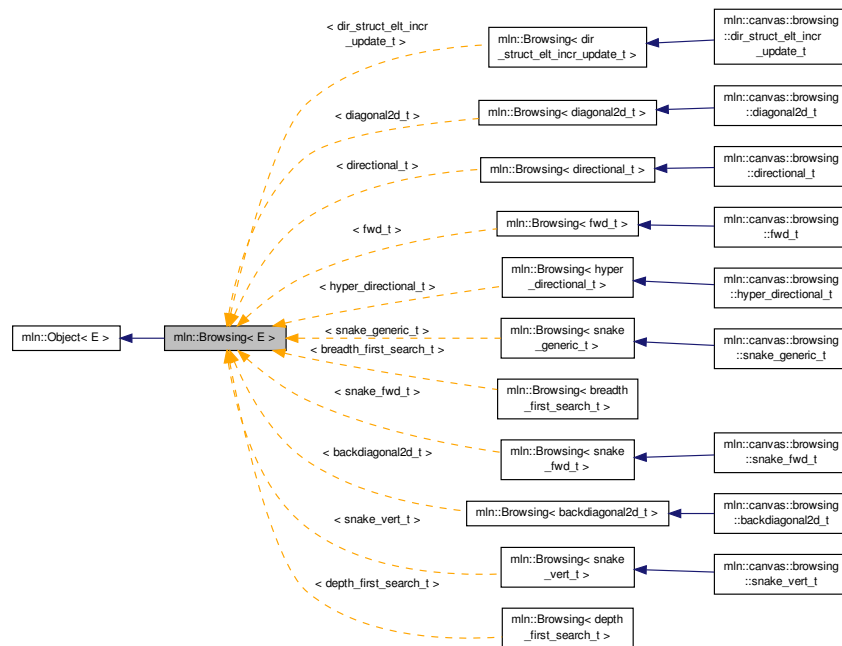
Definition at line 169 of file `box_runstart_piter.hh`.

10.105 mln::Browsing< E > Struct Template Reference

Base class for implementation classes that are browsings.

```
#include <browsing.hh>
```


Inheritance diagram for mln::Browsing< E >:



10.105.1 Detailed Description

```
template<typename E>struct mln::Browsing< E >
```

Base class for implementation classes that are browsings.

See Also

`mln::doc::Browsing` for a complete documentation of this class contents.

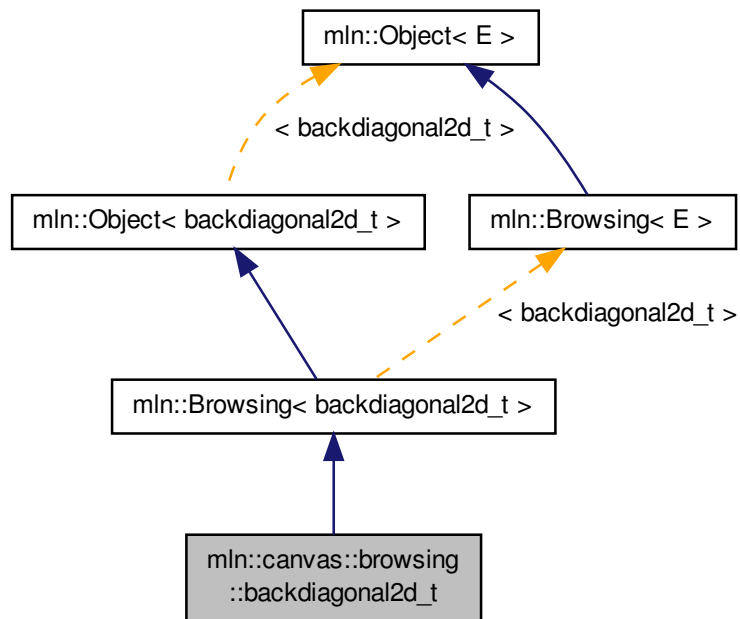
Definition at line 56 of file `browsing.hh`.

10.106 mln::canvas::browsing::backdiagonal2d_t Struct Reference

[Browsing](#) in a certain direction.

```
#include <backdiagonal2d.hh>
```

Inheritance diagram for `mln::canvas::browsing::backdiagonal2d_t`:



10.106.1 Detailed Description

Browsing in a certain direction.

This canvas browse all the point of an image 'input' of type 'l' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'l', '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:

```
l;
```

— as attributes:

```
dim;
```

```
dir; // and test dir < dim
```

```
input;
```

```
p;
```

— as methods:

```
void init();
void next();
void final();
}
```

Example :

```
-----> | 4 7 9 | 2 5 8 | 1 3 6
```

Definition at line 83 of file backdiagonal2d.hh.

10.107 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< breadth_first_search_t, std::queue >.

10.107.1 Detailed Description

Breadth-first search algorithm for graph, on vertices.

Definition at line 80 of file breadth_first_search.hh.

10.108 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< depth_first_search_t, std::stack >.

10.108.1 Detailed Description

Breadth-first search algorithm for graph, on vertices.

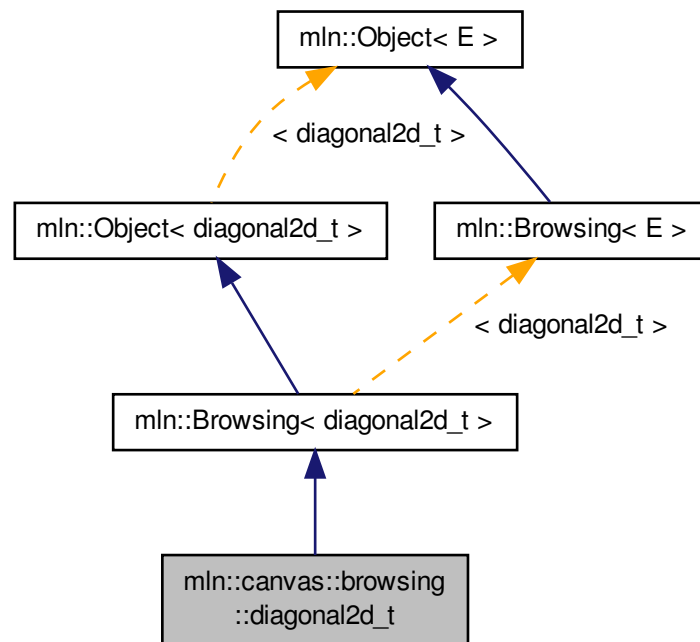
Definition at line 80 of file depth_first_search.hh.

10.109 mln::canvas::browsing::diagonal2d_t Struct Reference

[Browsing](#) in a certain direction.

```
#include <diagonal2d.hh>
```

Inheritance diagram for mln::canvas::browsing::diagonal2d_t:



10.109.1 Detailed Description

[Browsing](#) in a certain direction.

This canvas browse all the point of an image 'input' of type 'l' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'l', '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:

```
l;
```

— as attributes:

```
dim;
```

```
dir; // and test dir < dim
```

```
input;
```

```
p;
```

— as methods:

```

void init();
void next();
void final();
}

```

Example :

```
| 1 3 6 | 2 5 8 | 4 7 9 L--->
```

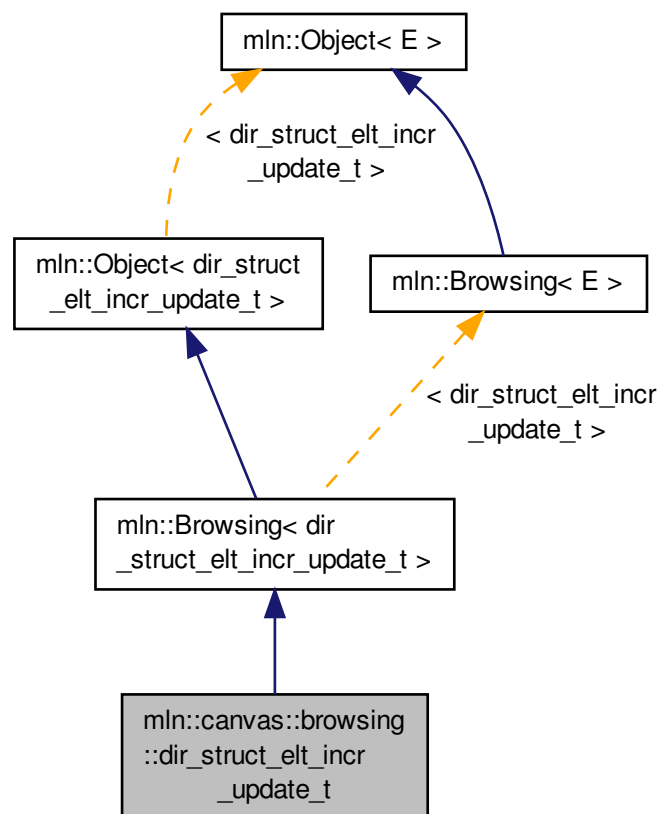
Definition at line 82 of file diagonal2d.hh.

10.110 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:



10.110.1 Detailed Description

[Browsing](#) in a certain direction with a segment.

This canvas browse all the point of an image 'input' of type 'l', of dimension 'dim' in the direction 'dir' with considering weigh the 'length' nearest points.

The functor should provide (In addition to 'input', 'l', '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:
l;
— as attributes:
dim;
dir; // and test dir < dim
input;
p;
length;
— as methods:
void init();
void init_line();
void add_point(q)
void remove_point(q)
void next();
void final();
}
```

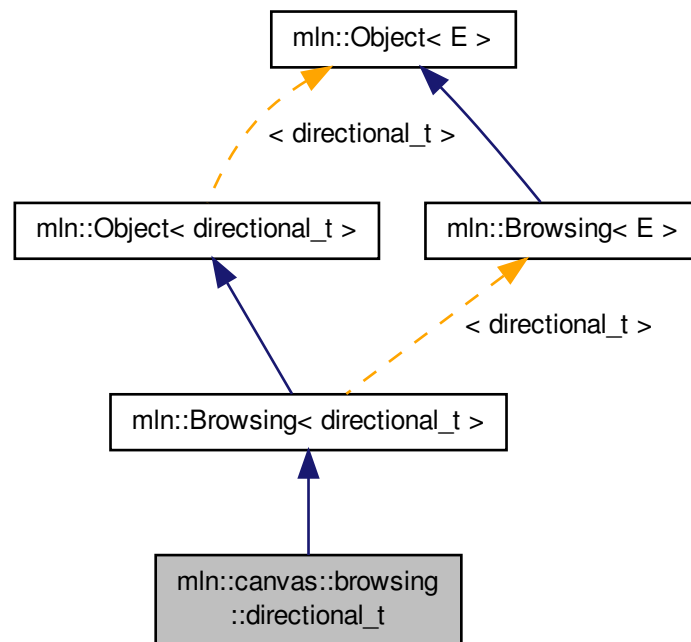
Definition at line 87 of file `dir_struct_elt_incr_update.hh`.

10.111 `mln::canvas::browsing::directional_t` Struct Reference

[Browsing](#) in a certain direction.

```
#include <directional.hh>
```

Inheritance diagram for mIn::canvas::browsing::directional_t:



10.111.1 Detailed Description

[Browsing](#) in a certain direction.

This canvas browse all the point of an image 'input' of type 'I' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'I', 'dim' and 'dir') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```
{
```

— as types:

```
I;
```

— as attributes:

```
dim;
```

```
dir; // and test dir < dim
```

```
input;
```

```
p;
```

— as methods:

```

void init();
void next();
void final();
}

```

Example :

```

1 0 0 2 0 0 3 0 0
4 0 0 5 0 0 6 0 0
7 0 0 8 0 0 9 0 0

```

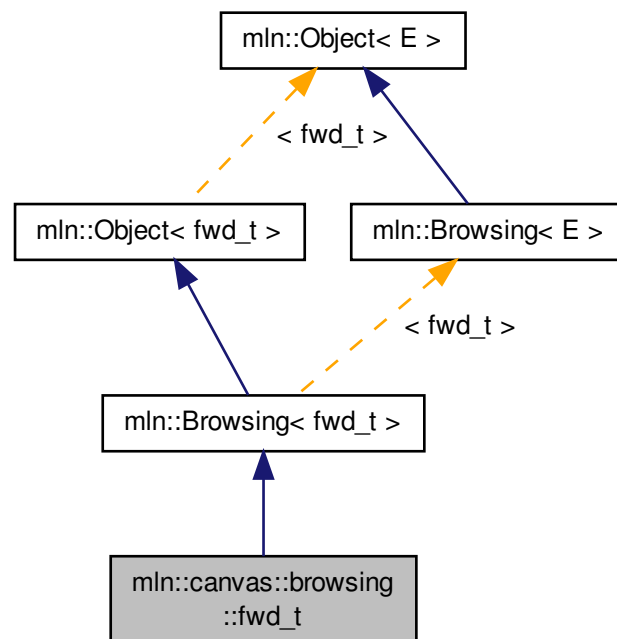
Definition at line 90 of file directional.hh.

10.112 mln::canvas::browsing::fwd_t Struct Reference

Canvas for forward browsing.

```
#include <fwd.hh>
```

Inheritance diagram for mln::canvas::browsing::fwd_t:



10.112.1 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:  
l;  
— as attributes:  
input;  
p;  
— as method:  
void init();  
void next();  
void final();  
}
```

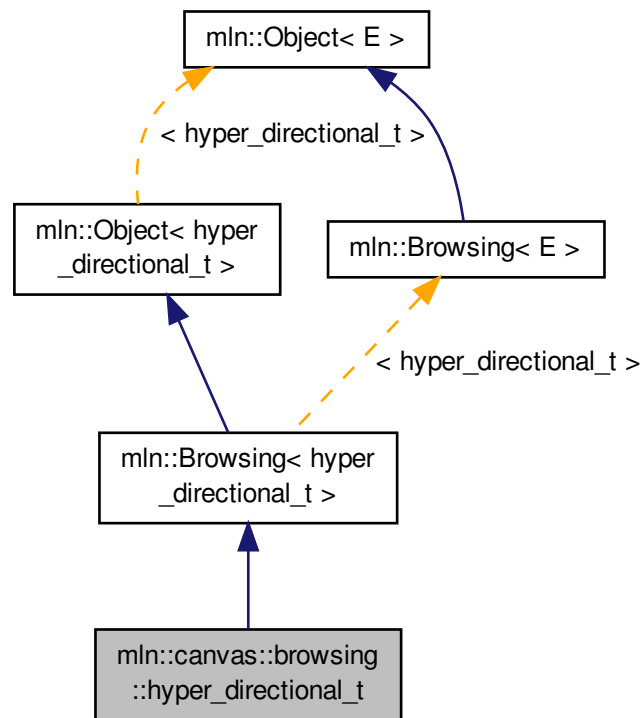
Definition at line 73 of file fwd.hh.

10.113 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`:



10.113.1 Detailed Description

Browsing in a certain direction.

This canvas browse all the point of an image 'input' of type 'I' and of dimension 'dim' in the direction 'dir'.

The functor should provide (In addition to 'input', 'I', 'dim' and 'dir') three methods :

- `init()` : Will be called at the beginning.
- `next()` : Will be called at each point 'p' (also provided by the functor).
- `final()` : Will be called at the end.

F shall features :

```
{
```

— as types:

```
I;
```

— as attributes:

```
dim;
```

```
dir; // and test dir < dim
```

```
input;
```

```

p;
— as methods:
void init();
void next();
void final();
}

```

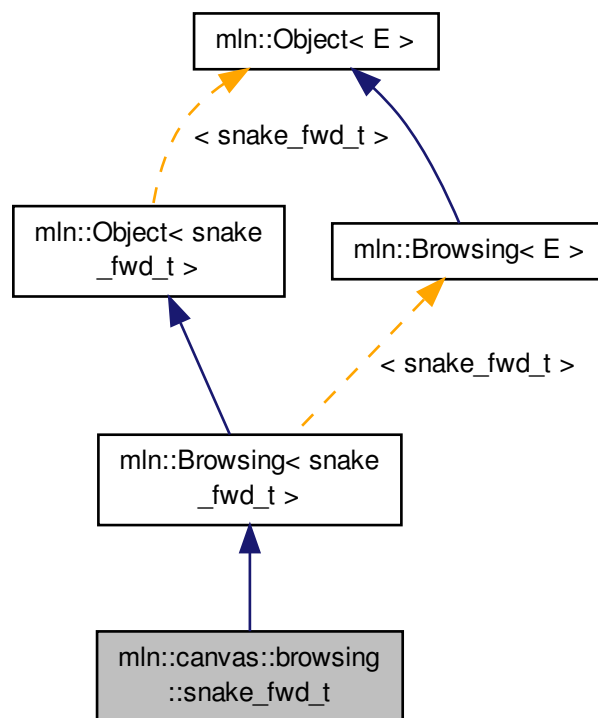
Definition at line 75 of file hyper_directional.hh.

10.114 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:



10.114.1 Detailed Description

[Browsing](#) in a snake-way, forward.

This canvas browse all the point of an image 'input' like this :

```

----->
<-----'

```

'----->

The functor should provide (In addition to 'input') four methods :

- `init()` : Will be called at the beginning.
- `down()` : Will be called after each moving down. (will also be called once at the first point).
- `fwd()` : Will be called after each moving right.
- `bwd()` : Will be called after each moving left.

This methods should access to the current working point 'p' also provided by the functor.

Warning: This canvas works only on 2D.

F shall feature:

```
{
— as attributes:
input;
p;
— as methods:
void init();
void down();
void fwd();
void bkd();
}
```

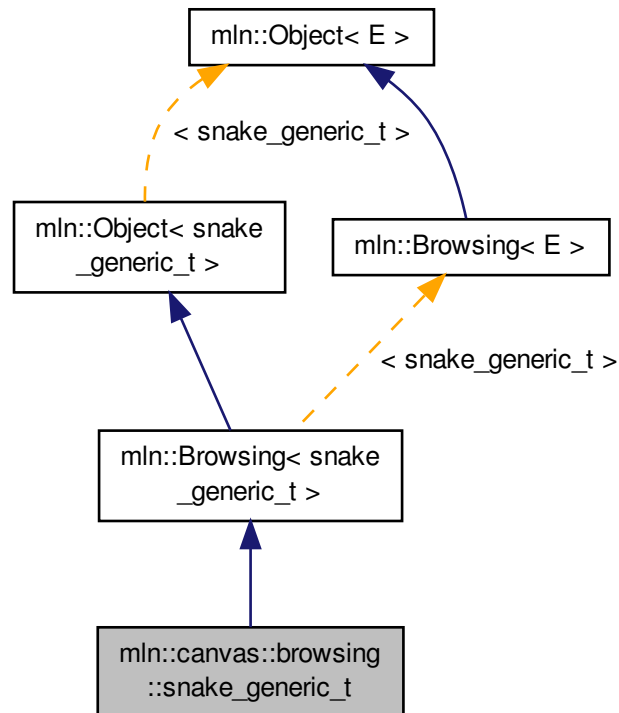
Definition at line 84 of file `snake_fwd.hh`.

10.115 `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:



10.115.1 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 called in each time the snake progress in the correct dimension :

`moves[i]` is called at each move following the delta psize `dps[i]`

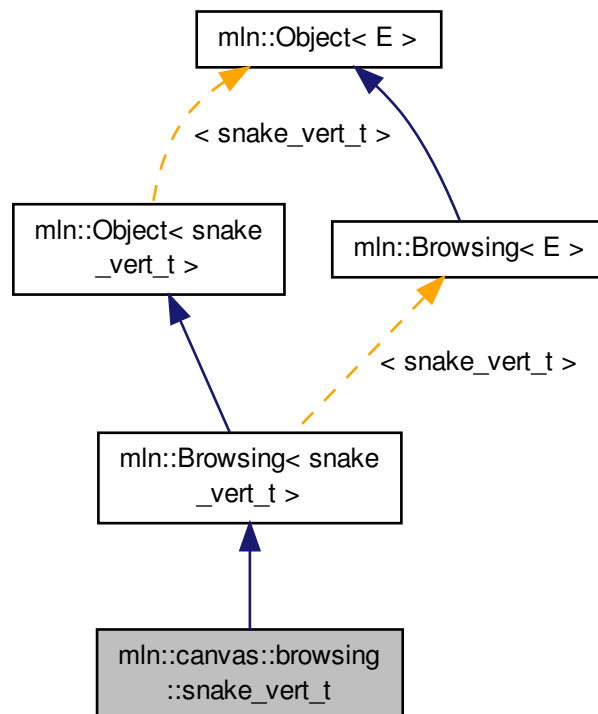
Definition at line 76 of file `snake_generic.hh`.

10.116 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`:



10.116.1 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();
}
```

Definition at line 84 of file snake_vert.hh.

10.117 mln::canvas::chamfer< F > Struct Template Reference

Compute chamfer distance.

```
#include <chamfer.hh>
```

10.117.1 Detailed Description

```
template<typename F>struct mln::canvas::chamfer< F >
```

Compute chamfer distance.

Definition at line 47 of file canvas/chamfer.hh.

10.118 mln::category< R(*) (A) > Struct Template Reference

Category declaration for a unary C function.

```
#include <c.hh>
```

10.118.1 Detailed Description

```
template<typename R, typename A>struct mln::category< R(*) (A) >
```

Category declaration for a unary C function.

Definition at line 51 of file c.hh.

10.119 mln::complex_image< D, G, V > Class Template Reference

[Image](#) based on a complex.

```
#include <complex_image.hh>
```

Inherits mln::internal::image_primary< V, p_complex< D, G >, complex_image< D, G, V > >.

Public Types

- typedef G [geom](#)
The geometry type of the complex.
- typedef V & [lvalue](#)
Return type of read-write access.
- typedef const V & [rvalue](#)
Return type of read-only access.
- typedef [complex_image](#)< D, tag::psite_< G >, tag::value_< V > > [skeleton](#)
Skeleton.
- typedef V [value](#)
[Value](#) associated type.

Public Member Functions

- [rvalue operator\(\)](#) (const [complex_psite](#)< D, G > &p) const
Read-only access of face value at point site p .
- [lvalue operator\(\)](#) (const [complex_psite](#)< D, G > &p)
Read-write access of face value at point site p .
- [complex_image](#) ()
Constructors.
- const [p_complex](#)< D, G > & [domain](#) () const
Accessors.
- const metal::vec< D+1, std::vector< mlc_unbool(V) > > & [values](#) () const
Return the array of values associated to the faces.

Static Public Attributes

- static const unsigned [dim](#) = D
The dimension of the complex.

10.119.1 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

<i>D</i>	The dimension of the complex.
<i>G</i>	The geometry type of the complex.
<i>V</i>	The value type of the image.

Definition at line 164 of file mln/core/image/complex_image.hh.

10.119.2 Member Typedef Documentation

10.119.2.1 `template<unsigned D, typename G, typename V> typedef G mln::complex_image< D, G, V >::geom`

The geometry type of the complex.

Definition at line 172 of file mln/core/image/complex_image.hh.

10.119.2.2 `template<unsigned D, typename G, typename V> typedef V& mln::complex_image< D, G, V >::lvalue`

Return type of read-write access.

Definition at line 177 of file mln/core/image/complex_image.hh.

10.119.2.3 `template<unsigned D, typename G, typename V> typedef const V& mln::complex_image< D, G, V >::rvalue`

Return type of read-only access.

Definition at line 180 of file mln/core/image/complex_image.hh.

10.119.2.4 `template<unsigned D, typename G, typename V> typedef complex_image< D, tag::psite_<G>, tag::value_<V> > mln::complex_image< D, G, V >::skeleton`

Skeleton.

Definition at line 183 of file mln/core/image/complex_image.hh.

10.119.2.5 `template<unsigned D, typename G, typename V> typedef V mln::complex_image< D, G, V >::value`

[Value](#) associated type.

Definition at line 174 of file mln/core/image/complex_image.hh.

10.119.3 Constructor & Destructor Documentation

10.119.3.1 `template<unsigned D, typename G , typename V > complex_image< D, G, V >::complex_image ()`
`[inline]`

Constructors.

Definition at line 276 of file mln/core/image/complex_image.hh.

10.119.4 Member Function Documentation

10.119.4.1 `template<unsigned D, typename G , typename V > const p_complex< D, G > & complex_image< D, G, V >::domain () const` `[inline]`

Accessors.

Return the domain of psites of the image.

Definition at line 343 of file `mln/core/image/complex_image.hh`.

10.119.4.2 `template<unsigned D, typename G, typename V> complex_image< D, G, V>::rvalue complex_image< D, G, V>::operator() (const complex_psite< D, G> & p) const [inline]`

Read-only access of face value at point site `p`.

Definition at line 317 of file `mln/core/image/complex_image.hh`.

References `mln::complex_psite< D, G>::face_id()`, and `mln::complex_psite< D, G>::n()`.

10.119.4.3 `template<unsigned D, typename G, typename V> complex_image< D, G, V>::lvalue complex_image< D, G, V>::operator() (const complex_psite< D, G> & p) [inline]`

Read-write access of face value at point site `p`.

Definition at line 326 of file `mln/core/image/complex_image.hh`.

References `mln::complex_psite< D, G>::face_id()`, and `mln::complex_psite< D, G>::n()`.

10.119.4.4 `template<unsigned D, typename G, typename V> const metal::vec< D+1, std::vector< mlc_unbool(V)>> & complex_image< D, G, V>::values () const [inline]`

Return the array of values associated to the faces.

Definition at line 335 of file `mln/core/image/complex_image.hh`.

10.119.5 Member Data Documentation

10.119.5.1 `template<unsigned D, typename G, typename V> const unsigned mln::complex_image< D, G, V>::dim = D [static]`

The dimension of the complex.

Definition at line 170 of file `mln/core/image/complex_image.hh`.

10.120 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, 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.

10.120.1 Detailed Description

template<typename I, typename G, typename N>class mln::complex_neighborhood_bkd_piter< I, G, N >

Backward iterator on complex neighborhood.

Definition at line 124 of file complex_neighborhood_piter.hh.

10.120.2 Member Typedef Documentation

10.120.2.1 template<typename I, typename G, typename N> typedef N::complex_bkd_iter
mln::complex_neighborhood_bkd_piter< I, G, N >::iter_type

The type of the underlying complex iterator.

Definition at line 135 of file complex_neighborhood_piter.hh.

10.120.2.2 template<typename I, typename G, typename N> typedef N::psite mln::complex_neighborhood_bkd_piter< I, G, N >::psite

The [Pseudo_Site](#) type.

Definition at line 133 of file complex_neighborhood_piter.hh.

10.120.3 Constructor & Destructor Documentation

10.120.3.1 template<typename I , typename G , typename N > mln::complex_neighborhood_bkd_piter< I, G, N
>::complex_neighborhood_bkd_piter () [inline]

Construction.

Definition at line 305 of file complex_neighborhood_piter.hh.

10.120.4 Member Function Documentation

10.120.4.1 template<typename I , typename G , typename N > const N::complex_bkd_iter &
mln::complex_neighborhood_bkd_piter< I, G, N >::iter () const [inline]

Accessors.

Definition at line 382 of file complex_neighborhood_piter.hh.

10.120.4.2 `void mln::Site_iterator< complex_neighborhood_bkd_piter< I, G, N > >::next ()` [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.121 `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, 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.

10.121.1 Detailed Description

```
template<typename I, typename G, typename N>class mln::complex_neighborhood_fwd_piter< I, G, N >
```

Forward iterator on complex neighborhood.

Definition at line 53 of file `complex_neighborhood_piter.hh`.

10.121.2 Member Typedef Documentation

10.121.2.1 `template<typename I, typename G, typename N> typedef N::complex_fwd_iter`
`mln::complex_neighborhood_fwd_piter< I, G, N >::iter_type`

The type of the underlying complex iterator.

Definition at line 64 of file `complex_neighborhood_piter.hh`.

10.121.2.2 `template<typename I, typename G, typename N> typedef N::psite mln::complex_neighborhood_fwd_piter< I, G, N >::psite`

The [Pseudo_Site](#) type.

Definition at line 62 of file `complex_neighborhood_piter.hh`.

10.121.3 Constructor & Destructor Documentation

10.121.3.1 `template<typename I, typename G, typename N> mln::complex_neighborhood_fwd_piter< I, G, N >::complex_neighborhood_fwd_piter () [inline]`

Construction.

Definition at line 198 of file `complex_neighborhood_piter.hh`.

10.121.4 Member Function Documentation

10.121.4.1 `template<typename I, typename G, typename N> const N::complex_fwd_iter & mln::complex_neighborhood_fwd_piter< I, G, N >::iter () const [inline]`

Accessors.

Definition at line 275 of file `complex_neighborhood_piter.hh`.

10.121.4.2 `void mln::Site_Iterator< complex_neighborhood_fwd_piter< I, G, N > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.122 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 &, complex_psite< D, G > >`.

Public Member Functions

- [complex_psite](#) ()
Construction and assignment.
- [complex_psite](#) (const [p_complex](#)< D, G > &pc, const [topo::face](#)< D > &face)
- bool [is_valid](#) () const
Psite manipulators.
- void [invalidate](#) ()

Invalidate this psite.

- const [target](#) & [site_set](#) () const
Site set manipulators.
- void [change_target](#) (const [target](#) &new_target)
Set the target site_set.
- const [topo::face](#)< D > & [face](#) () const
Face handle manipulators.
- unsigned [n](#) () const
Return the dimension of the face of this psite.
- unsigned [face_id](#) () const
Return the id of the face of this psite.

10.122.1 Detailed Description

template<unsigned D, typename G>class mln::complex_psite< D, G >

[Point](#) site associated to a [mln::p_complex](#).

Template Parameters

<i>D</i>	The dimension of the complex this psite belongs to.
<i>G</i>	The geometry of the complex.

Definition at line 60 of file complex_psite.hh.

10.122.2 Constructor & Destructor Documentation

10.122.2.1 template<unsigned D, typename G > mln::complex_psite< D, G >::complex_psite () [inline]

Construction and assignment.

Definition at line 203 of file complex_psite.hh.

References [mln::complex_psite< D, G >::invalidate\(\)](#).

10.122.2.2 template<unsigned D, typename G > mln::complex_psite< D, G >::complex_psite (const [p_complex](#)< D, G > &pc, const [topo::face](#)< D > &face) [inline]

Precondition

[pc.cplx\(\)](#) == [face.cplx\(\)](#).

Definition at line 211 of file complex_psite.hh.

References [mln::topo::face< D >::cplx\(\)](#), [mln::p_complex< D, G >::cplx\(\)](#), and [mln::complex_psite< D, G >::is_valid\(\)](#).

10.122.3 Member Function Documentation

10.122.3.1 template<unsigned D, typename G > void mln::complex_psite< D, G >::change_target (const [target](#) &
new_target) [inline]

Set the target site_set.

Definition at line 280 of file complex_psite.hh.

References mln::p_complex< D, G >::cplx().

10.122.3.2 `template<unsigned D, typename G > const topo::face< D > & mln::complex_psite< D, G >::face () const [inline]`

Face handle manipulators.

Return the face handle of this point site.

Definition at line 301 of file complex_psite.hh.

Referenced by mln::operator!==(), and mln::operator==().

10.122.3.3 `template<unsigned D, typename G > unsigned mln::complex_psite< D, G >::face_id () const [inline]`

Return the id of the face of this psite.

Definition at line 317 of file complex_psite.hh.

Referenced by mln::complex_image< D, G, V >::operator()().

10.122.3.4 `template<unsigned D, typename G > void mln::complex_psite< D, G >::invalidate () [inline]`

Invalidate this psite.

Definition at line 251 of file complex_psite.hh.

Referenced by mln::complex_psite< D, G >::complex_psite().

10.122.3.5 `template<unsigned D, typename G > bool mln::complex_psite< D, G >::is_valid () const [inline]`

Psite manipulators.

Is this psite valid?

Definition at line 239 of file complex_psite.hh.

Referenced by mln::complex_psite< D, G >::complex_psite(), and mln::p_complex< D, G >::has().

10.122.3.6 `template<unsigned D, typename G > unsigned mln::complex_psite< D, G >::n () const [inline]`

Return the dimension of the face of this psite.

Definition at line 309 of file complex_psite.hh.

Referenced by mln::make::cell(), and mln::complex_image< D, G, V >::operator()().

10.122.3.7 `template<unsigned D, typename G > const p_complex< D, G > & mln::complex_psite< D, G >::site_set () const [inline]`

Site set manipulators.

Return the [mln::p_complex](#) this site is built on. (shortcut for *target()).

Precondition

Member face_ is valid.

Definition at line 259 of file complex_psite.hh.

Referenced by mln::p_complex< D, G >::has(), mln::operator!==(), and mln::operator==().

10.123 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, 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.

10.123.1 Detailed Description

```
template<typename I, typename G, typename W>class mln::complex_window_bkd_piter< I, G, W >
```

Backward iterator on complex window.

Definition at line 124 of file complex_window_piter.hh.

10.123.2 Member Typedef Documentation

10.123.2.1 `template<typename I, typename G, typename W> typedef W::complex_bkd_iter
mln::complex_window_bkd_piter< I, G, W >::iter_type`

The type of the underlying complex iterator.

Definition at line 135 of file complex_window_piter.hh.

10.123.2.2 `template<typename I, typename G, typename W> typedef W::psite mln::complex_window_bkd_piter< I, G,
W >::psite`

The [Pseudo_Site](#) type.

Definition at line 133 of file complex_window_piter.hh.

10.123.3 Constructor & Destructor Documentation

10.123.3.1 `template<typename I , typename G , typename W > mln::complex_window_bkd_piter< I, G, W >::complex_window_bkd_piter() [inline]`

Construction.

Definition at line 305 of file complex_window_piter.hh.

10.123.4 Member Function Documentation

10.123.4.1 `template<typename I , typename G , typename W > const W::complex_bkd_iter & mln::complex_window_bkd_piter< I, G, W >::iter() const [inline]`

Accessors.

Definition at line 385 of file complex_window_piter.hh.

10.123.4.2 `void mln::Site_Iterator< complex_window_bkd_piter< I, G, W > >::next() [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.124 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, 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.

10.124.1 Detailed Description

```
template<typename I, typename G, typename W>class mln::complex_window_fwd_piter< I, G, W >
```

Forward iterator on complex window.

Definition at line 54 of file complex_window_piter.hh.

10.124.2 Member Typedef Documentation

10.124.2.1 `template<typename I, typename G, typename W> typedef W::complex_fwd_iter
mln::complex_window_fwd_piter< I, G, W >::iter_type`

The type of the underlying complex iterator.

Definition at line 65 of file complex_window_piter.hh.

10.124.2.2 `template<typename I, typename G, typename W> typedef W::psite mln::complex_window_fwd_piter< I, G,
W >::psite`

The [Pseudo_Site](#) type.

Definition at line 63 of file complex_window_piter.hh.

10.124.3 Constructor & Destructor Documentation

10.124.3.1 `template<typename I , typename G , typename W > mln::complex_window_fwd_piter< I, G, W
>::complex_window_fwd_piter () [inline]`

Construction.

Definition at line 197 of file complex_window_piter.hh.

10.124.4 Member Function Documentation

10.124.4.1 `template<typename I , typename G , typename W > const W::complex_fwd_iter &
mln::complex_window_fwd_piter< I, G, W >::iter () const [inline]`

Accessors.

Definition at line 275 of file complex_window_piter.hh.

10.124.4.2 `void mln::Site_Iterator< complex_window_fwd_piter< I, G, W > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.125 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, decorated_image< I, D > >, and mln::internal::image_identity< I, I::domain_t, decorated_image< I, D > >.

Public Types

- typedef [impl_::lvalue](#) [lvalue](#)
Return type of read-write access.
- typedef I::psite [psite](#)
Type of the psite.
- typedef I::rvalue [rvalue](#)
Return type of read-only access.
- typedef [decorated_image](#)
< tag::image_< I >, tag::data_
< D > > [skeleton](#)
Skeleton.

Public Member Functions

- [decorated_image](#) ()
Ctors.
- const D & [decoration](#) () const
Give the decoration.
- D & [decoration](#) ()
Give the decoration.
- operator [decorated_image](#)< const I, D > () const
Const promotion via conversion.
- [rvalue operator](#)() (const [psite](#) &p) const
Read-only access of pixel value at point site p.
- [lvalue operator](#)() (const [psite](#) &p)
Read-write access of pixel value at point site p.

10.125.1 Detailed Description

```
template<typename I, typename D>struct mln::decorated_image< I, D >
```

[Image](#) that can have additional features.

Definition at line 81 of file decorated_image.hh.

10.125.2 Member Typedef Documentation

10.125.2.1 template<typename I, typename D> typedef impl_::lvalue mln::decorated_image< I, D >::lvalue

Return type of read-write access.

Definition at line 95 of file decorated_image.hh.

10.125.2.2 `template<typename I, typename D> typedef I::psite mln::decorated_image< I, D >::psite`

Type of the psite.

Definition at line 90 of file decorated_image.hh.

10.125.2.3 `template<typename I, typename D> typedef I::rvalue mln::decorated_image< I, D >::rvalue`

Return type of read-only access.

Definition at line 93 of file decorated_image.hh.

10.125.2.4 `template<typename I, typename D> typedef decorated_image< tag::image_<I>, tag::data_<D> > mln::decorated_image< I, D >::skeleton`

Skeleton.

Definition at line 108 of file decorated_image.hh.

10.125.3 Constructor & Destructor Documentation

10.125.3.1 `template<typename I, typename D> decorated_image< I, D >::decorated_image () [inline]`

Ctors.

Definition at line 161 of file decorated_image.hh.

10.125.4 Member Function Documentation

10.125.4.1 `template<typename I, typename D> const D & decorated_image< I, D >::decoration () const [inline]`

Give the decoration.

Definition at line 249 of file decorated_image.hh.

10.125.4.2 `template<typename I, typename D> D & decorated_image< I, D >::decoration () [inline]`

Give the decoration.

Definition at line 257 of file decorated_image.hh.

10.125.4.3 `template<typename I, typename D> decorated_image< I, D >::operator decorated_image< const I, D > () const [inline]`

Const promotion via conversion.

Definition at line 239 of file decorated_image.hh.

10.125.4.4 `template<typename I, typename D> decorated_image< I, D >::rvalue decorated_image< I, D >::operator() (const psite & p) const [inline]`

Read-only access of pixel value at point site p.

Definition at line 197 of file decorated_image.hh.

10.125.4.5 `template<typename I , typename D > decorated_image< I, D >::lvalue decorated_image< I, D >::operator() (const psite & p) [inline]`

Read-write access of pixel value at point site p.

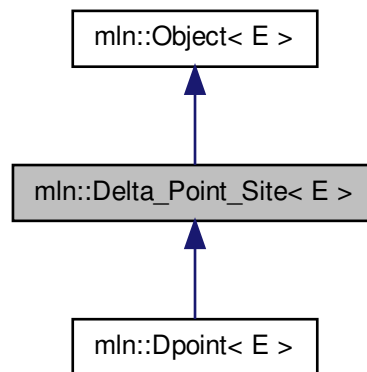
Definition at line 206 of file decorated_image.hh.

10.126 mln::Delta_Point_Site< E > Struct Template Reference

FIXME: Doc!

```
#include <delta_point_site.hh>
```

Inheritance diagram for mln::Delta_Point_Site< E >:



10.126.1 Detailed Description

```
template<typename E>struct mln::Delta_Point_Site< E >
```

FIXME: Doc!

Definition at line 79 of file delta_point_site.hh.

10.127 mln::Delta_Point_Site< void > Struct Template Reference

Delta point site category flag type.

```
#include <delta_point_site.hh>
```

10.127.1 Detailed Description

```
template<>struct mln::Delta_Point_Site< void >
```

Delta point site category flag type.

Definition at line 70 of file delta_point_site.hh.

10.128 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 [argument](#) &t)
Take into account a argument t (an element).
- void [take](#) (const E &other)
*Take into account another accumulator *other*.*

10.128.1 Detailed Description

```
template<typename E>struct mln::doc::Accumulator< E >
```

Documentation class for [mln::Accumulator](#).

See Also

[mln::Accumulator](#)

Definition at line 36 of file doc/accumulator.hh.

10.128.2 Member Typedef Documentation

10.128.2.1 `template<typename E > typedef void mln::doc::Accumulator< E >::argument`

The argument type of elements to accumulate.

Definition at line 39 of file doc/accumulator.hh.

10.128.3 Member Function Documentation

10.128.3.1 `template<typename E > void mln::doc::Accumulator< E >::init ()`

Initialize the accumulator.

10.128.3.2 `template<typename E > void mln::doc::Accumulator< E >::take (const argument & t)`

Take into account a argument t (an element).

10.128.3.3 `template<typename E > void mln::doc::Accumulator< E >::take (const E & other)`

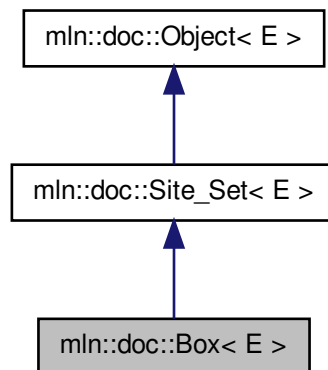
Take into account another accumulator *other*.

10.129 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.

10.129.1 Detailed Description

`template<typename E> struct mln::doc::Box< E >`

Documentation class for [mln::Box](#).

See Also

[mln::Box](#)

Definition at line 36 of file `core/concept/doc/box.hh`.

10.129.2 Member Typedef Documentation

10.129.2.1 `template<typename E > typedef void mln::doc::Site_Set< E >::bkd_piter` [inherited]

Backward [Site_Iterator](#) associated type.

Definition at line 53 of file `mln/core/concept/doc/site_set.hh`.

10.129.2.2 `template<typename E > typedef void mln::doc::Site_Set< E >::fwd_piter` [inherited]

Forward [Site_Iterator](#) associated type.

Definition at line 49 of file `mln/core/concept/doc/site_set.hh`.

10.129.2.3 `template<typename E > typedef void mln::doc::Site_Set< E >::psite` [inherited]

Psite associated type.

Definition at line 45 of file `mln/core/concept/doc/site_set.hh`.

10.129.2.4 `template<typename E > typedef void mln::doc::Site_Set< E >::site` [inherited]

[Site](#) associated type.

Definition at line 41 of file `mln/core/concept/doc/site_set.hh`.

10.129.3 Member Function Documentation

10.129.3.1 `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.

10.129.3.2 `template<typename E > bool mln::doc::Site_Set< E >::has (const psite & p) const` [inherited]

Test if *p* belongs to this site set.

Parameters

<i>in</i>	<i>p</i>	A psite.
-----------	----------	----------

Returns

True if *p* is an element of the site set.

10.129.3.3 `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.

10.129.3.4 `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.

10.129.3.5 `template<typename E > const site& mln::doc::Box< E >::pmin () const`

Give the box "minimum" point.

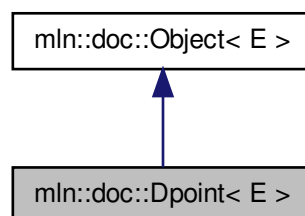
Return the "minimum" point w.r.t. the ordering between points. For instance, with [mln::box2d](#), this minimum is the top left point of the box.

10.130 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.

10.130.1 Detailed Description

template<typename E>struct mln::doc::Dpoint< E >

Documentation class for [mln::Dpoint](#).

See Also

[mln::Dpoint](#)

Definition at line 36 of file concept/doc/dpoint.hh.

10.130.2 Member Typedef Documentation

10.130.2.1 template<typename E > typedef void mln::doc::Dpoint< E >::coord

Coordinate associated type.

Definition at line 56 of file concept/doc/dpoint.hh.

10.130.2.2 template<typename E > typedef void mln::doc::Dpoint< E >::dpoint

Dpsite associated type.

Invariant

This type has to derive from [mln::Dpoint](#).

Definition at line 52 of file concept/doc/dpoint.hh.

10.130.2.3 template<typename E > typedef void mln::doc::Dpoint< E >::point

[Site](#) associated type.

Invariant

This type has to derive from [mln::Point](#).

Definition at line 47 of file concept/doc/dpoint.hh.

10.130.3 Member Enumeration Documentation

10.130.3.1 template<typename E > anonymous enum

Enumerator

dim Dimension of the space.

Invariant

$\text{dim} > 0$

Definition at line 42 of file concept/doc/dpoint.hh.

10.130.4 Member Function Documentation

10.130.4.1 template<typename E > coord mln::doc::Dpoint< E >::operator[] (unsigned *i*) const

Read-only access to the *i*-th coordinate value.

Parameters

<i>in</i>	<i>i</i>	The coordinate index.
-----------	----------	-----------------------

Precondition

$i < \text{dim}$

Returns

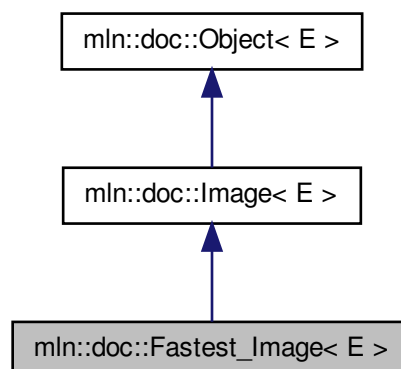
The value of the *i*-th coordinate.

10.131 mln::doc::Fastest_Image< E > Struct Template Reference

Documentation class for the concept of images that have the speed property set to "fastest".

```
#include <image_fastest.hh>
```

Inheritance diagram for mln::doc::Fastest_Image< E >:



Public Types

- typedef void [bkd_piter](#)
Backward point iterator associated type.
- typedef void [coord](#)
Coordinate associated type.
- typedef void [dpoint](#)
Dpsite associated type.
- typedef void [fwd_piter](#)
Forward point iterator associated type.
- typedef void [lvalue](#)
Type returned by the read-write pixel value operator.
- typedef void [point](#)
Site associated type.
- typedef void [pset](#)
Point set associated type.
- typedef void [psite](#)
Point_Site associated type.
- typedef void [rvalue](#)
Type returned by the read pixel value operator.
- typedef void [skeleton](#)
Associate type that describes how this type of image is constructed.
- typedef void [value](#)
Value associated type.
- typedef void [vset](#)
Value set associated type.

Public Member Functions

- const [box](#)< [point](#) > & [bbox](#) () const
Give a bounding box of the image domain.
- unsigned [border](#) ()
Give the border thickness.
- const [value](#) * [buffer](#) () const
Give a hook to the value buffer.
- int [delta_index](#) (const [dpoint](#) &dp)
Give the offset corresponding to the delta-point dp.
- const [pset](#) & [domain](#) () const
Give the definition domain of the image.
- bool [has](#) (const [psite](#) &p) const
Test if the image owns the point site p.
- bool [has](#) (const [psite](#) &p) const
Test if p belongs to the image domain.
- bool [is_valid](#) () const
Test if the image have been initialized.
- unsigned [nelements](#) () const
Give the number of pixels of the image including those of the virtual border.
- unsigned [nsites](#) () const
Give the number of points of the image domain.
- [rvalue operator](#)() (const [psite](#) &p) const
Read-only access to the image value located at p.

- [lvalue operator\(\)](#) (const [psite](#) &p)
Read-write access to the image value located at p.
- [rvalue operator\[\]](#) (unsigned o) const
Read-only access to the image value at offset o.
- [lvalue operator\[\]](#) (unsigned o)
Read-write access to the image value at offset o.
- [point point_at_index](#) (unsigned o) const
Give the point at offset o.
- const [vset](#) & [values](#) () const
Give the set of values of the image.

10.131.1 Detailed Description

`template<typename E>struct mln::doc::Fastest_Image< E >`

Documentation class for the concept of images that have the speed property set to "fastest".

Definition at line 36 of file `concept/doc/image_fastest.hh`.

10.131.2 Member Typedef Documentation

10.131.2.1 `template<typename E > typedef void mln::doc::Image< E >::bkd_piter` [inherited]

Backward point iterator associated type.

Invariant

This type has to derive from [mln::Site_Iterator](#).

Definition at line 147 of file `core/concept/doc/image.hh`.

10.131.2.2 `template<typename E > typedef void mln::doc::Image< E >::coord` [inherited]

Coordinate associated type.

Definition at line 131 of file `core/concept/doc/image.hh`.

10.131.2.3 `template<typename E > typedef void mln::doc::Image< E >::dpoint` [inherited]

Dpsite associated type.

Invariant

This type has to derive from [mln::Dpoint](#).

Definition at line 136 of file `core/concept/doc/image.hh`.

10.131.2.4 `template<typename E > typedef void mln::doc::Image< E >::fwd_piter` [inherited]

Forward point iterator associated type.

Invariant

This type has to derive from [mln::Site_Iterator](#).

Definition at line 142 of file `core/concept/doc/image.hh`.

10.131.2.5 `template<typename E> typedef void mln::doc::Image< E>::lvalue` [inherited]

Type returned by the read-write pixel value operator.

Definition at line 52 of file core/concept/doc/image.hh.

10.131.2.6 `template<typename E> typedef void mln::doc::Image< E>::point` [inherited]

[Site](#) associated type.

Invariant

This type has to derive from [mln::Point](#).

Definition at line 121 of file core/concept/doc/image.hh.

10.131.2.7 `template<typename E> typedef void mln::doc::Image< E>::pset` [inherited]

[Point](#) set associated type.

Invariant

This type has to derive from [mln::Site_Set](#).

Definition at line 116 of file core/concept/doc/image.hh.

10.131.2.8 `template<typename E> typedef void mln::doc::Image< E>::psite` [inherited]

[Point_Site](#) associated type.

Invariant

This type has to derive from [mln::Point_Site](#).

Definition at line 126 of file core/concept/doc/image.hh.

10.131.2.9 `template<typename E> typedef void mln::doc::Image< E>::rvalue` [inherited]

Type returned by the read pixel value operator.

Definition at line 48 of file core/concept/doc/image.hh.

10.131.2.10 `template<typename E> typedef void mln::doc::Image< E>::skeleton` [inherited]

Associate type that describes how this type of image is constructed.

Definition at line 64 of file core/concept/doc/image.hh.

10.131.2.11 `template<typename E> typedef void mln::doc::Image< E>::value` [inherited]

[Value](#) associated type.

Invariant

This type is neither qualified by const, nor by reference.

Definition at line 44 of file core/concept/doc/image.hh.

10.131.2.12 `template<typename E> typedef void mln::doc::Image< E >::vset` [inherited]

Value set associated type.

Invariant

This type has to derive from mln::Value_Set.

Definition at line 57 of file core/concept/doc/image.hh.

10.131.3 Member Function Documentation

10.131.3.1 `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.

10.131.3.2 `template<typename E> unsigned mln::doc::Fastest_Image< E >::border ()`

Give the border thickness.

Precondition

The image has to be initialized.

10.131.3.3 `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.

10.131.3.4 `template<typename E> int mln::doc::Fastest_Image< E >::delta_index (const dpoint & dp)`

Give the offset corresponding to the delta-point dp.

Parameters

in	dp	A delta-point.
----	----	----------------

Precondition

The image has to be initialized.

10.131.3.5 `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.

10.131.3.6 `template<typename E> bool mln::doc::Image< E>::has (const psite & p) const` `[inherited]`

Test if the image owns the point site p .

Returns

True if accessing the image value at p is possible, that is, does not abort the execution.

10.131.3.7 `template<typename E> bool mln::doc::Image< E>::has (const psite & p) const` `[inherited]`

Test if p belongs to the image domain.

Parameters

in	p	A point site.
------	-----	---------------

Returns

True if p belongs to the image domain.

Invariant

$has(p)$ is true \Rightarrow $has(p)$ is also true.

10.131.3.8 `template<typename E> bool mln::doc::Image< E>::is_valid () const` `[inherited]`

Test if the image have been initialized.

10.131.3.9 `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.

10.131.3.10 `template<typename E> unsigned mln::doc::Image< E>::nsites () const` `[inherited]`

Give the number of points of the image domain.

10.131.3.11 `template<typename E> rvalue mln::doc::Image< E>::operator() (const psite & p) const`
`[inherited]`

Read-only access to the image value located at p .

Parameters

in	p	A point site.
------	-----	---------------

Precondition

The image has to own the site p .

Returns

The value at p (not assignable).

10.131.3.12 `template<typename E> lvalue mln::doc::Image< E >::operator() (const psite & p) [inherited]`

Read-write access to the image value located at p .

Parameters

in	p	A point site.
------	-----	---------------

Precondition

The image has to own the site p .

Returns

The value at p (assignable).

10.131.3.13 `template<typename E> rvalue mln::doc::Fastest_Image< E >::operator[] (unsigned o) const`

Read-only access to the image value at offset o .

Parameters

in	o	An offset.
------	-----	------------

Precondition

$o < \text{nelements}()$

Returns

The value at o (not assignable).

10.131.3.14 `template<typename E> lvalue mln::doc::Fastest_Image< E >::operator[] (unsigned o)`

Read-write access to the image value at offset o .

Parameters

in	o	An offset.
------	-----	------------

Precondition

$o < \text{nelements}()$

Returns

The value at `o` (assignable).

10.131.3.15 `template<typename E> point mln::doc::Fastest_Image< E >::point_at_index (unsigned o) const`

Give the point at offset `o`.

Parameters

<code>in</code>	<code>o</code>	An offset.
-----------------	----------------	------------

Precondition

The image has to be initialized.

`o < nelements\(\)`

10.131.3.16 `template<typename E> const vset& mln::doc::Image< E >::values () const` `[inherited]`

Give the set of values of the image.

Returns

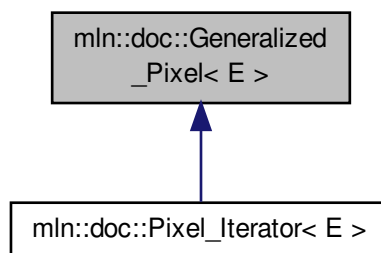
A reference to the value set.

10.132 `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.

10.132.1 Detailed Description

template<typename E>struct mln::doc::Generalized_Pixel< E >

Documentation class for [mln::Generalized_Pixel](#).

See Also

[mln::Generalized_Pixel](#)

Definition at line 45 of file doc/generalized_pixel.hh.

10.132.2 Member Typedef Documentation

10.132.2.1 template<typename E > typedef void mln::doc::Generalized_Pixel< E >::image

[Image](#) associated type (with possible const qualification).

Definition at line 49 of file doc/generalized_pixel.hh.

10.132.2.2 template<typename E > typedef void mln::doc::Generalized_Pixel< E >::rvalue

Read-only value associated type.

Definition at line 55 of file doc/generalized_pixel.hh.

10.132.2.3 template<typename E > typedef void mln::doc::Generalized_Pixel< E >::value

[Value](#) associated type.

Definition at line 52 of file doc/generalized_pixel.hh.

10.132.3 Member Function Documentation

10.132.3.1 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.

10.132.3.2 `template<typename E > rvalue mln::doc::Generalized_Pixel< E >::val () const`

Give the value of this generalized pixel.

Returns

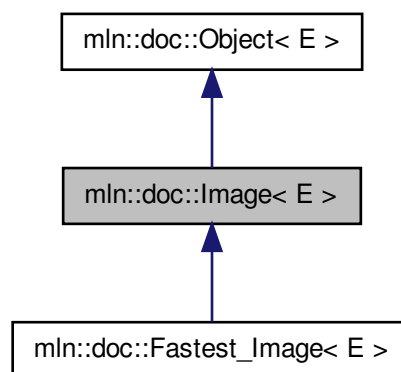
A read-only value.

10.133 `mln::doc::Image< E >` Struct Template Reference

Documentation class for `mln::Image`.

```
#include <image.hh>
```

Inheritance diagram for `mln::doc::Image< E >`:



Public Types

- typedef void `bkd_piter`
Backward point iterator associated type.
- typedef void `coord`
Coordinate associated type.
- typedef void `dpoint`
Dpsite associated type.
- typedef void `fwd_piter`
Forward point iterator associated type.
- typedef void `lvalue`
Type returned by the read-write pixel value operator.
- typedef void `point`
Site associated type.
- typedef void `pset`
Point set associated type.
- typedef void `psite`
Point_Site associated type.
- typedef void `rvalue`

Type returned by the read pixel value operator.

- typedef void [skeleton](#)

Associate type that describes how this type of image is constructed.

- typedef void [value](#)

Value associated type.

- typedef void [vset](#)

Value set associated type.

Public Member Functions

- const [box](#)< [point](#) > & [bbox](#) () const
Give a bounding box of the image domain.
- const [pset](#) & [domain](#) () const
Give the definition domain of the image.
- bool [has](#) (const [psite](#) &p) const
Test if the image owns the point site p .
- bool [has](#) (const [psite](#) &p) const
Test if p belongs to the image domain.
- bool [is_valid](#) () const
Test if the image have been initialized.
- unsigned [nsites](#) () const
Give the number of points of the image domain.
- [rvalue operator](#)() (const [psite](#) &p) const
Read-only access to the image value located at p .
- [lvalue operator](#)() (const [psite](#) &p)
Read-write access to the image value located at p .
- const [vset](#) & [values](#) () const
Give the set of values of the image.

10.133.1 Detailed Description

template<typename E>struct mln::doc::Image< E >

Documentation class for [mln::Image](#).

See Also

[mln::Image](#)

Definition at line 36 of file core/concept/doc/image.hh.

10.133.2 Member Typedef Documentation

10.133.2.1 template<typename E > typedef void mln::doc::Image< E >::bkd_piter

Backward point iterator associated type.

Invariant

This type has to derive from [mln::Site_iterator](#).

Definition at line 147 of file core/concept/doc/image.hh.

10.133.2.2 `template<typename E > typedef void mln::doc::Image< E >::coord`

Coordinate associated type.

Definition at line 131 of file core/concept/doc/image.hh.

10.133.2.3 `template<typename E > typedef void mln::doc::Image< E >::dpoint`

Dpsite associated type.

Invariant

This type has to derive from [mln::Dpoint](#).

Definition at line 136 of file core/concept/doc/image.hh.

10.133.2.4 `template<typename E > typedef void mln::doc::Image< E >::fwd_piter`

Forward point iterator associated type.

Invariant

This type has to derive from [mln::Site_Iterator](#).

Definition at line 142 of file core/concept/doc/image.hh.

10.133.2.5 `template<typename E > typedef void mln::doc::Image< E >::lvalue`

Type returned by the read-write pixel value operator.

Definition at line 52 of file core/concept/doc/image.hh.

10.133.2.6 `template<typename E > typedef void mln::doc::Image< E >::point`

[Site](#) associated type.

Invariant

This type has to derive from [mln::Point](#).

Definition at line 121 of file core/concept/doc/image.hh.

10.133.2.7 `template<typename E > typedef void mln::doc::Image< E >::pset`

[Point](#) set associated type.

Invariant

This type has to derive from [mln::Site_Set](#).

Definition at line 116 of file core/concept/doc/image.hh.

10.133.2.8 `template<typename E > typedef void mln::doc::Image< E >::psite`

[Point_Site](#) associated type.

Invariant

This type has to derive from mln::Point_Site.

Definition at line 126 of file core/concept/doc/image.hh.

10.133.2.9 `template<typename E > typedef void mln::doc::Image< E >::rvalue`

Type returned by the read pixel value operator.

Definition at line 48 of file core/concept/doc/image.hh.

10.133.2.10 `template<typename E > typedef void mln::doc::Image< E >::skeleton`

Associate type that describes how this type of image is constructed.

Definition at line 64 of file core/concept/doc/image.hh.

10.133.2.11 `template<typename E > typedef void mln::doc::Image< E >::value`

[Value](#) associated type.

Invariant

This type is neither qualified by const, nor by reference.

Definition at line 44 of file core/concept/doc/image.hh.

10.133.2.12 `template<typename E > typedef void mln::doc::Image< E >::vset`

[Value](#) set associated type.

Invariant

This type has to derive from mln::Value_Set.

Definition at line 57 of file core/concept/doc/image.hh.

10.133.3 Member Function Documentation

10.133.3.1 `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.

10.133.3.2 `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.

10.133.3.3 `template<typename E> bool mln::doc::Image< E>::has (const psite & p) const`

Test if the image owns the point site *p*.

Returns

True if accessing the image value at *p* is possible, that is, does not abort the execution.

10.133.3.4 `template<typename E> bool mln::doc::Image< E>::has (const psite & p) const`

Test if *p* belongs to the image domain.

Parameters

<i>in</i>	<i>p</i>	A point site.
-----------	----------	---------------

Returns

True if *p* belongs to the image domain.

Invariant

has(*p*) is true => has(*p*) is also true.

10.133.3.5 `template<typename E> bool mln::doc::Image< E>::is_valid () const`

Test if the image have been initialized.

10.133.3.6 `template<typename E> unsigned mln::doc::Image< E>::nsites () const`

Give the number of points of the image domain.

10.133.3.7 `template<typename E> rvalue mln::doc::Image< E>::operator() (const psite & p) const`

Read-only access to the image value located at *p*.

Parameters

<i>in</i>	<i>p</i>	A point site.
-----------	----------	---------------

Precondition

The image has to own the site *p*.

Returns

The value at *p* (not assignable).

10.133.3.8 `template<typename E> lvalue mln::doc::Image< E >::operator() (const psite & p)`

Read-write access to the image value located at *p*.

Parameters

<i>in</i>	<i>p</i>	A point site.
-----------	----------	---------------

Precondition

The image has to own the site *p*.

Returns

The value at *p* (assignable).

10.133.3.9 `template<typename E> const vset& mln::doc::Image< E >::values () const`

Give the set of values of the image.

Returns

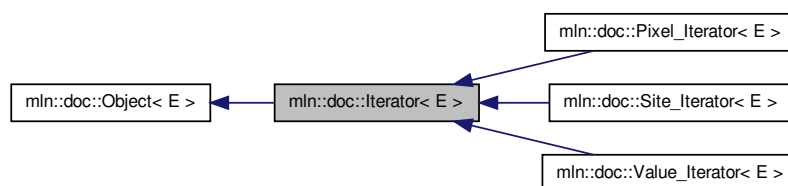
A reference to the value set.

10.134 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.

10.134.1 Detailed Description

`template<typename E> struct mln::doc::iterator< E >`

Documentation class for [mln::iterator](#).

See Also

[mln::iterator](#)

Definition at line 36 of file doc/iterator.hh.

10.134.2 Member Function Documentation

10.134.2.1 `template<typename E > void mln::doc::iterator< E >::invalidate ()`

Invalidate the iterator.

10.134.2.2 `template<typename E > bool mln::doc::iterator< E >::is_valid () const`

Returns true if the iterator is valid, that is, designates an element.

10.134.2.3 `template<typename E > void mln::doc::iterator< E >::start ()`

Start an iteration.

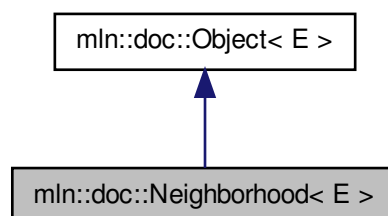
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

10.135 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.

10.135.1 Detailed Description

template<typename E>struct mln::doc::Neighborhood< E >

Documentation class for [mln::Neighborhood](#).

See Also

[mln::Neighborhood](#)

Definition at line 37 of file core/concept/doc/neighborhood.hh.

10.135.2 Member Typedef Documentation

10.135.2.1 template<typename E > typedef void mln::doc::Neighborhood< E >::bkd_niter

[Site_Iterator](#) type associated to this neighborhood to browse neighbors in a backward way.

Definition at line 52 of file core/concept/doc/neighborhood.hh.

10.135.2.2 template<typename E > typedef void mln::doc::Neighborhood< E >::dpoint

Dpsite associated type.

Definition at line 55 of file core/concept/doc/neighborhood.hh.

10.135.2.3 template<typename E > typedef void mln::doc::Neighborhood< E >::fwd_niter

[Site_Iterator](#) type associated to this neighborhood to browse neighbors in a forward way.

Definition at line 47 of file core/concept/doc/neighborhood.hh.

10.135.2.4 template<typename E > typedef void mln::doc::Neighborhood< E >::niter

[Site_Iterator](#) type associated to this neighborhood to browse neighbors.

Definition at line 42 of file core/concept/doc/neighborhood.hh.

10.135.2.5 template<typename E > typedef void mln::doc::Neighborhood< E >::point

[Site](#) associated type.

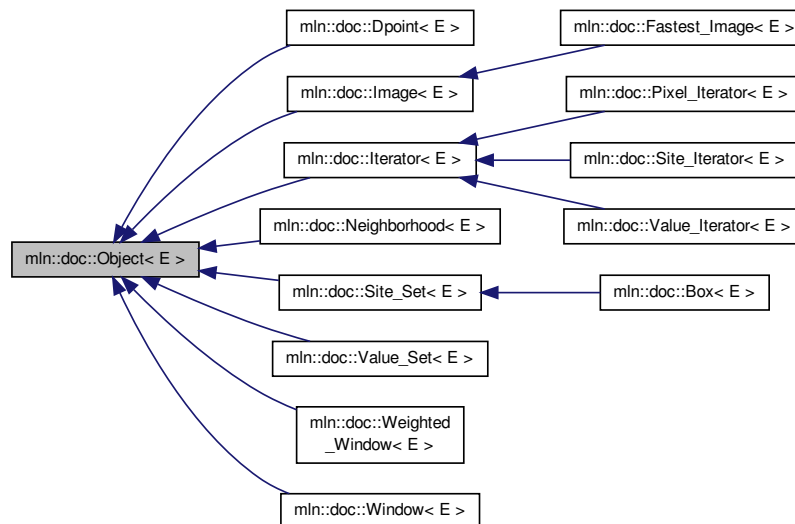
Definition at line 58 of file core/concept/doc/neighborhood.hh.

10.136 mln::doc::Object< E > Struct Template Reference

Documentation class for [mln::Object](#).

```
#include <object.hh>
```

Inheritance diagram for mln::doc::Object< E >:



10.136.1 Detailed Description

```
template<typename E>struct mln::doc::Object< E >
```

Documentation class for [mln::Object](#).

See Also

[mln::Object](#)

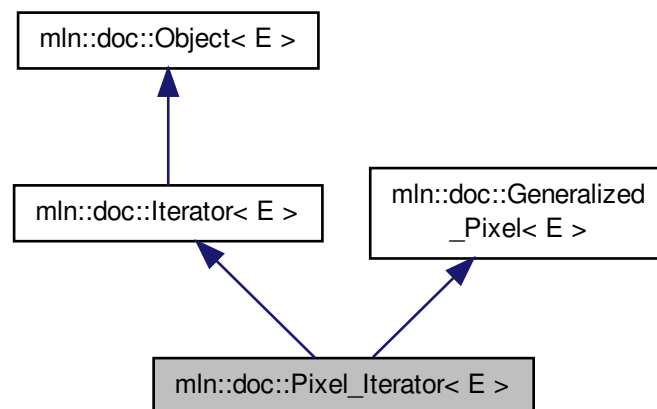
Definition at line 46 of file `doc/object.hh`.

10.137 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.

10.137.1 Detailed Description

template<typename E> struct mln::doc::Pixel_Iterator< E >

Documentation class for [mln::Iterator](#).

See Also

`mln::Pixel_Iterator`

Definition at line 36 of file `doc/pixel_iterator.hh`.

10.137.2 Member Typedef Documentation

10.137.2.1 `template<typename E > typedef void mln::doc::Generalized_Pixel< E >::image` [inherited]

[Image](#) associated type (with possible const qualification).

Definition at line 49 of file `doc/generalized_pixel.hh`.

10.137.2.2 `template<typename E > typedef void mln::doc::Pixel_Iterator< E >::lvalue`

Type returned by the read-write dereference operator.

Definition at line 41 of file `doc/pixel_iterator.hh`.

10.137.2.3 `template<typename E > typedef void mln::doc::Generalized_Pixel< E >::rvalue` [inherited]

Read-only value associated type.

Definition at line 55 of file `doc/generalized_pixel.hh`.

10.137.2.4 `template<typename E > typedef void mln::doc::Generalized_Pixel< E >::value` [inherited]

[Value](#) associated type.

Definition at line 52 of file `doc/generalized_pixel.hh`.

10.137.3 Member Function Documentation

10.137.3.1 `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.

10.137.3.2 `template<typename E > void mln::doc::Iterator< E >::invalidate ()` [inherited]

Invalidate the iterator.

10.137.3.3 `template<typename E > bool mln::doc::Iterator< E >::is_valid () const` [inherited]

Returns true if the iterator is valid, that is, designates an element.

10.137.3.4 `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.

10.137.3.5 `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.

10.138 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.

10.138.1 Detailed Description

```
template<typename E> struct mln::doc::Point_Site< E >
```

Documentation class for mln::Point_Site.

See Also

[mln::Point_Site](#)

Definition at line 37 of file doc/point_site.hh.

10.138.2 Member Typedef Documentation

10.138.2.1 `template<typename E> typedef void mln::doc::Point_Site< E >::coord`

Coordinate associated type.

Definition at line 62 of file doc/point_site.hh.

10.138.2.2 `template<typename E > typedef void mln::doc::Point_Site< E >::dpoint`

Dpsite associated type.

Invariant

This type has to derive from [mln::Dpoint](#).

Definition at line 58 of file doc/point_site.hh.

10.138.2.3 `template<typename E > typedef void mln::doc::Point_Site< E >::mesh`

[Mesh](#) associated type.

Invariant

This type has to derive from [mln::Mesh](#).

Definition at line 48 of file doc/point_site.hh.

10.138.2.4 `template<typename E > typedef void mln::doc::Point_Site< E >::point`

[Site](#) associated type.

Invariant

This type has to derive from [mln::Point](#).

Definition at line 53 of file doc/point_site.hh.

10.138.3 Member Enumeration Documentation

10.138.3.1 `template<typename E > anonymous enum`

Enumerator

dim Dimension of the space.

Invariant

$\text{dim} > 0$

Definition at line 43 of file doc/point_site.hh.

10.138.4 Member Function Documentation

10.138.4.1 `template<typename E > coord mln::doc::Point_Site< E >::operator[] (unsigned i) const`

Read-only access to the `i-th` coordinate value.

Parameters

<code>in</code>	<code>i</code>	The coordinate index.
-----------------	----------------	-----------------------

Precondition

`i < dim`

Returns

The value of the `i-th` coordinate.

10.138.4.2 `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

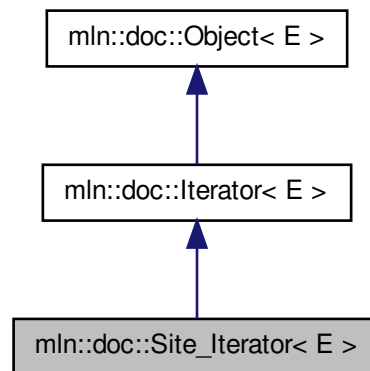
A point constant reference.

10.139 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.

10.139.1 Detailed Description

`template<typename E> struct mln::doc::Site_iterator< E >`

Documentation class for `mln::Site_iterator`.

See Also

`mln::Site_iterator`

Definition at line 37 of file `point_iterator.hh`.

10.139.2 Member Typedef Documentation

10.139.2.1 `template<typename E > typedef void mln::doc::Site_iterator< E >::psite`

`Point_Site` associated type.

Invariant

This type has to derive from `mln::Point_Site`.

Definition at line 43 of file `point_iterator.hh`.

10.139.3 Member Function Documentation

10.139.3.1 `template<typename E > void mln::doc::iterator< E >::invalidate () [inherited]`

Invalidate the iterator.

10.139.3.2 `template<typename E > bool mln::doc::iterator< E >::is_valid () const [inherited]`

Returns true if the iterator is valid, that is, designates an element.

10.139.3.3 `template<typename E > mln::doc::Site_iterator< E >::operator psite () const`

Conversion into a point-site.

Returns

A point site.

10.139.3.4 `template<typename E > void mln::doc::iterator< E >::start () [inherited]`

Start an iteration.

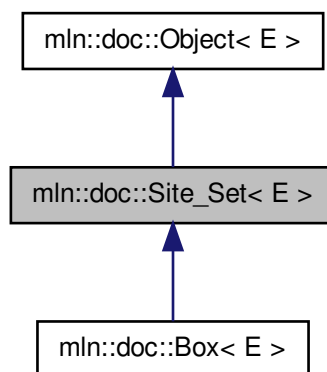
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

10.140 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.

10.140.1 Detailed Description

```
template<typename E>struct mln::doc::Site_Set< E >
```

Documentation class for [mln::Site_Set](#).

See Also

[mln::Site_Set](#)

Definition at line 37 of file mln/core/concept/doc/site_set.hh.

10.140.2 Member Typedef Documentation

10.140.2.1 `template<typename E > typedef void mln::doc::Site_Set< E >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 53 of file `mln/core/concept/doc/site_set.hh`.

10.140.2.2 `template<typename E > typedef void mln::doc::Site_Set< E >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 49 of file `mln/core/concept/doc/site_set.hh`.

10.140.2.3 `template<typename E > typedef void mln::doc::Site_Set< E >::psite`

PSite associated type.

Definition at line 45 of file `mln/core/concept/doc/site_set.hh`.

10.140.2.4 `template<typename E > typedef void mln::doc::Site_Set< E >::site`

[Site](#) associated type.

Definition at line 41 of file `mln/core/concept/doc/site_set.hh`.

10.140.3 Member Function Documentation

10.140.3.1 `template<typename E > bool mln::doc::Site_Set< E >::has (const psite & p) const`

Test if `p` belongs to this site set.

Parameters

<code>in</code>	<code>p</code>	A psite.
-----------------	----------------	----------

Returns

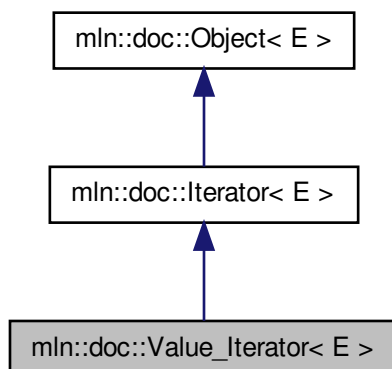
True if `p` is an element of the site set.

10.141 `mln::doc::Value_Iterator< E >` Struct Template Reference

Documentation class for `mln::Value_Iterator`.

```
#include <value_iterator.hh>
```

Inheritance diagram for mIn::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.

10.141.1 Detailed Description

```
template<typename E>struct mIn::doc::Value_Iterator< E >
```

Documentation class for mIn::Value_Iterator.

See Also

mIn::Value_Iterator

Definition at line 37 of file doc/value_iterator.hh.

10.141.2 Member Typedef Documentation

10.141.2.1 `template<typename E > typedef void mln::doc::Value_Iterator< E >::value`

[Value](#) associated type.

Definition at line 41 of file doc/value_iterator.hh.

10.141.3 Member Function Documentation

10.141.3.1 `template<typename E > void mln::doc::Iterator< E >::invalidate () [inherited]`

Invalidate the iterator.

10.141.3.2 `template<typename E > bool mln::doc::Iterator< E >::is_valid () const [inherited]`

Returns true if the iterator is valid, that is, designates an element.

10.141.3.3 `template<typename E > mln::doc::Value_Iterator< E >::operator value () const`

Conversion into a value.

Returns

A value.

10.141.3.4 `template<typename E > void mln::doc::Iterator< E >::start () [inherited]`

Start an iteration.

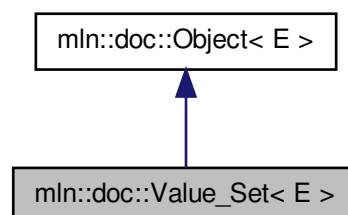
Make the iterator designate the first element if it exists. If this first element does not exist, the iterator is not valid.

10.142 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.

10.142.1 Detailed Description

template<typename E>struct mln::doc::Value_Set< E >

Documentation class for mln::Value_Set.

See Also

mln::Value_Set

Definition at line 37 of file doc/value_set.hh.

10.142.2 Member Typedef Documentation

10.142.2.1 template<typename E > typedef void mln::doc::Value_Set< E >::bkd_viter

Backward [Value_Iterator](#) associated type.

Definition at line 49 of file doc/value_set.hh.

10.142.2.2 template<typename E > typedef void mln::doc::Value_Set< E >::fwd_viter

Forward [Value_Iterator](#) associated type.

Definition at line 45 of file doc/value_set.hh.

10.142.2.3 template<typename E > typedef void mln::doc::Value_Set< E >::value

[Value](#) associated type.

Definition at line 41 of file doc/value_set.hh.

10.142.3 Member Function Documentation

10.142.3.1 `template<typename E> bool mln::doc::Value_Set< E >::has (const value & v) const`

Test if `v` belongs to this set of values.

Parameters

<code>in</code>	<code>v</code>	A value.
-----------------	----------------	----------

Returns

True if `v` is an element of the set of values.

10.142.3.2 `template<typename E> unsigned mln::doc::Value_Set< E >::index_of (const value & v) const`

Give the index of value `v` in this set.

10.142.3.3 `template<typename E> unsigned mln::doc::Value_Set< E >::nvalues () const`

Give the number of values in this set.

10.142.3.4 `template<typename E> value mln::doc::Value_Set< E >::operator[] (unsigned i) const`

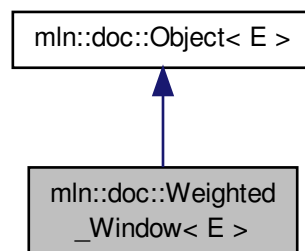
Give the `i`-th value of this set.

10.143 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.

10.143.1 Detailed Description

template<typename E> struct mln::doc::Weighted_Window< E >

Documentation class for [mln::Weighted_Window](#).

A weighted_window is the definition of a set of points located around a central point, with a weight associated to each point.

See Also

[mln::Weighted_Window](#)

Definition at line 40 of file doc/weighted_window.hh.

10.143.2 Member Typedef Documentation

10.143.2.1 template<typename E > typedef void mln::doc::Weighted_Window< E >::bkd_qiter

[Site_Iterator](#) type associated to this weighted_window to browse its points in a backward way.

Definition at line 51 of file doc/weighted_window.hh.

10.143.2.2 template<typename E > typedef void mln::doc::Weighted_Window< E >::dpoint

Dpsite associated type.

Definition at line 57 of file doc/weighted_window.hh.

10.143.2.3 `template<typename E> typedef void mln::doc::Weighted_Window< E >::fwd_qiter`

[Site_iterator](#) type associated to this weighted_window to browse its points in a forward way.

Definition at line 46 of file doc/weighted_window.hh.

10.143.2.4 `template<typename E> typedef void mln::doc::Weighted_Window< E >::point`

[Site](#) associated type.

Definition at line 54 of file doc/weighted_window.hh.

10.143.2.5 `template<typename E> typedef void mln::doc::Weighted_Window< E >::weight`

Weight associated type.

Definition at line 60 of file doc/weighted_window.hh.

10.143.2.6 `template<typename E> typedef void mln::doc::Weighted_Window< E >::window`

[Window](#) associated type.

Definition at line 63 of file doc/weighted_window.hh.

10.143.3 Member Function Documentation

10.143.3.1 `template<typename E> unsigned mln::doc::Weighted_Window< E >::delta () const`

Give the maximum coordinate gap between the window center and a window point.

10.143.3.2 `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.

10.143.3.3 `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.

10.143.3.4 `template<typename E> E& mln::doc::Weighted_Window< E >::sym ()`

Apply a central symmetry to the target weighted window.

10.143.3.5 `template<typename E> const window& mln::doc::Weighted_Window< E >::win () const`

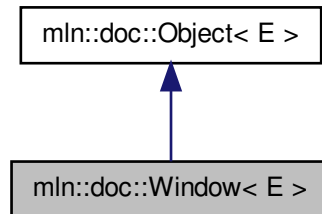
Give the corresponding window.

10.144 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.

10.144.1 Detailed Description

```
template<typename E>struct mln::doc::Window< E >
```

Documentation class for [mln::Window](#).

A window is the definition of a set of points located around a central point.

See Also

[mln::Window](#)

Definition at line 40 of file `concept/doc/window.hh`.

10.144.2 Member Typedef Documentation

10.144.2.1 `template<typename E > typedef void mln::doc::Window< E >::bkd_qiter`

[Site_iterator](#) type associated to this window to browse its points in a backward way.

Definition at line 55 of file `concept/doc/window.hh`.

10.144.2.2 `template<typename E > typedef void mln::doc::Window< E >::fwd_qiter`

[Site_iterator](#) type associated to this window to browse its points in a forward way.

Definition at line 50 of file `concept/doc/window.hh`.

10.144.2.3 `template<typename E > typedef void mln::doc::Window< E >::qiter`

[Site_iterator](#) type associated to this window to browse its points.

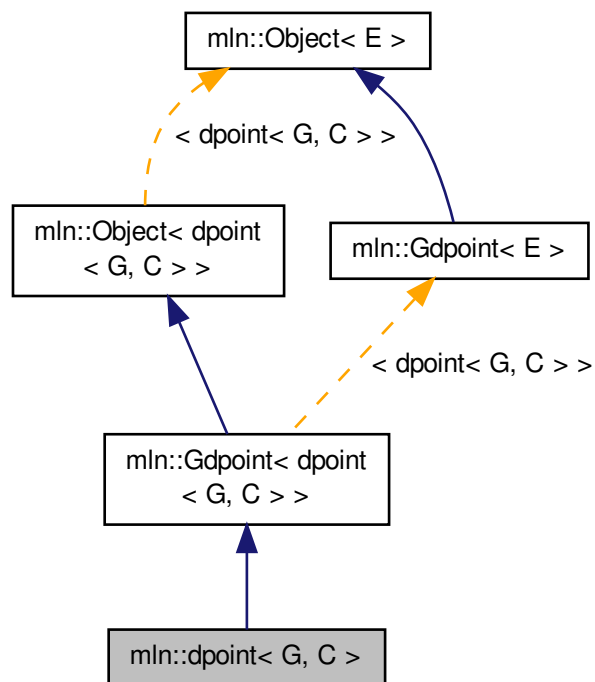
Definition at line 45 of file `concept/doc/window.hh`.

10.145 `mln::dpoint< G, C >` Struct Template Reference

Generic delta-point class.

```
#include <dpoint.hh>
```

Inheritance diagram for `mln::dpoint< G, C >`:



Public Types

- enum { `dim` = `G::dim` }
- typedef `C` `coord`
Coordinate associated type.
- typedef `G` `grid`
Grid associated type.
- typedef `point< G, C >` `psite`
Psite associated type.
- typedef `point< G, C >` `site`
Site associated type.
- typedef `algebra::vec< G::dim, C >` `vec`
Algebra vector (vec) associated type.

Public Member Functions

- `dpoint ()`
Constructor without argument.
- `template<typename C2 >`
`dpoint (const algebra::vec< dim, C2 > &v)`
Constructor from an algebra vector.
- `template<typename F >`
`dpoint (const Function_v2v< F > &f)`
Constructor; coordinates are set by function f .
- `template<typename Q >`
`operator mln::algebra::vec< dpoint< G, C >::dim, Q > () const`
Conversion towards a algebra::vec.
- `C operator[] (unsigned i) const`
Read-only access to the i -th coordinate value.
- `C & operator[] (unsigned i)`
Read-write access to the i -th coordinate value.
- `void set_all (C c)`
Set all coordinates to the value c .
- `vec to_vec () const`
Explicit conversion.
- `dpoint (C ind)`
- `dpoint (const literal::zero_t &)`
Constructors/assignments with literals.

10.145.1 Detailed Description

`template<typename G, typename C> struct mln::dpoint< G, C >`

Generic delta-point class.

Parameters are `G` the dimension of the space and `C` the coordinate type in this space.

Definition at line 58 of file `dpoint.hh`.

10.145.2 Member Typedef Documentation

10.145.2.1 `template<typename G, typename C> typedef C mln::dpoint< G, C >::coord`

Coordinate associated type.

Definition at line 76 of file `dpoint.hh`.

10.145.2.2 `template<typename G, typename C> typedef G mln::dpoint< G, C >::grid`

Grid associated type.

Definition at line 67 of file `dpoint.hh`.

10.145.2.3 `template<typename G, typename C> typedef point<G,C> mln::dpoint< G, C >::psite`

Psite associated type.

Definition at line 70 of file `dpoint.hh`.

10.145.2.4 `template<typename G, typename C> typedef point<G,C> mIn::dpoint< G, C >::site`

[Site](#) associated type.

Definition at line 73 of file dpoint.hh.

10.145.2.5 `template<typename G, typename C> typedef algebra::vec<G::dim, C> mIn::dpoint< G, C >::vec`

Algebra vector (vec) associated type.

Definition at line 79 of file dpoint.hh.

10.145.3 Member Enumeration Documentation

10.145.3.1 `template<typename G, typename C> anonymous enum`

Enumerator

dim Dimension of the space.
Invariant

`dim > 0`

Definition at line 64 of file dpoint.hh.

10.145.4 Constructor & Destructor Documentation

10.145.4.1 `template<typename G , typename C > dpoint< G, C >::dpoint () [inline]`

Constructor without argument.

Definition at line 152 of file dpoint.hh.

10.145.4.2 `template<typename G , typename C > template<typename C2 > dpoint< G, C >::dpoint (const algebra::vec< dim, C2 > & v) [inline]`

Constructor from an algebra vector.

Definition at line 159 of file dpoint.hh.

10.145.4.3 `template<typename G , typename C> dpoint< G, C >::dpoint (C ind) [inline]`

Constructors with different numbers of arguments (coordinates) w.r.t. the dimension.

Definition at line 176 of file dpoint.hh.

10.145.4.4 `template<typename G , typename C> dpoint< G, C >::dpoint (const literal::zero_t &) [inline]`

Constructors/assignments with literals.

Definition at line 203 of file dpoint.hh.

10.145.4.5 `template<typename G , typename C > template<typename F > dpoint< G, C >::dpoint (const Function_v2v< F > & f) [inline]`

Constructor; coordinates are set by function `f`.

Definition at line 238 of file dpoint.hh.

10.145.5 Member Function Documentation

10.145.5.1 `template<typename G , typename C > template<typename Q > dpoint< G, C >::operator mln::algebra::vec< dpoint< G, C >::dim, Q > () const [inline]`

Conversion towards a algebra::vec.

Definition at line 257 of file dpoint.hh.

10.145.5.2 `template<typename G , typename C > C dpoint< G, C >::operator[] (unsigned i) const [inline]`

Read-only access to the `i`-th coordinate value.

Parameters

<code>in</code>	<code>i</code>	The coordinate index.
-----------------	----------------	-----------------------

Precondition

`i < dim`

Definition at line 136 of file dpoint.hh.

10.145.5.3 `template<typename G , typename C > C & dpoint< G, C >::operator[] (unsigned i) [inline]`

Read-write access to the `i`-th coordinate value.

Parameters

<code>in</code>	<code>i</code>	The coordinate index.
-----------------	----------------	-----------------------

Precondition

`i < dim`

Definition at line 144 of file dpoint.hh.

10.145.5.4 `template<typename G , typename C > void dpoint< G, C >::set_all (C c) [inline]`

Set all coordinates to the value `c`.

Definition at line 248 of file dpoint.hh.

Referenced by `mln::win::line< M, i, C >::line()`.

10.145.5.5 `template<typename G , typename C > dpoint< G, C >::vec dpoint< G, C >::to_vec () const [inline]`

Explicit conversion.

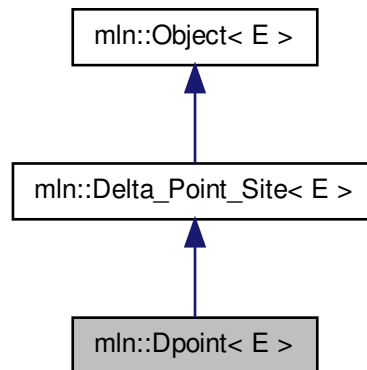
Definition at line 265 of file dpoint.hh.

10.146 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.

10.146.1 Detailed Description

```
template<typename E>struct mln::Dpoint< E >
```

Base class for implementation of delta-point classes.

A delta-point is a vector defined by a couple of points.

Given two points, A and B, the vector AB is mapped into the delta-point $D = AB$. Practically one can write: $D = B - A$.

See Also

[mln::doc::Dpoint](#) for a complete documentation of this class contents.

Definition at line 63 of file `concept/dpoint.hh`.

10.146.2 Member Function Documentation

10.146.2.1 `template<typename E > const E & Dpoint< E >::to_dpoint () const` `[inline]`

It is a [Dpoint](#) so it returns itself.

Definition at line 88 of file `concept/dpoint.hh`.

10.147 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>
```


Inherits mln::Pixel_Iterator< dpoints_bkd_pixter< I > >, and mln::internal::pixel_impl_< I, 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](#) (I &image, const Dps &dps, const Pref &p_ref)
Constructor (using an image).
- template<typename Dps , typename Pref >
[dpoints_bkd_pixter](#) (const [Generalized_Pixel](#)< Pref > &pxl_ref, const Dps &dps)
Constructor (using a generalized pixel).
- void [next](#) ()
Go to the next element.
- void [start](#) ()
Manipulation.
- void [invalidate](#) ()
Invalidate the iterator.
- bool [is_valid](#) () const
Test the iterator validity.
- void [update](#) ()
Force this iterator to update its location to take into account that its center point may have moved.

10.147.1 Detailed Description

```
template<typename I>class mln::dpoints_bkd_pixter< I >
```

A generic backward iterator on the pixels of a dpoint-based window or neighborhood.

Parameter *I* is the image type.

Definition at line 141 of file dpoints_pixter.hh.

10.147.2 Constructor & Destructor Documentation

10.147.2.1 `template<typename I > template<typename Dps , typename Pref > mln::dpoints_bkd_pixter< I >::dpoints_bkd_pixter (I &image, const Dps &dps, const Pref &p_ref) [inline]`

Constructor (using an image).

Parameters

in	<i>image</i>	The image to iterate over.
in	<i>dps</i>	An object (neighborhood or window) that can provide a set of delta-points.
in	<i>p_ref</i>	Center (resp. reference) point of the neighborhood (resp. window).

Definition at line 339 of file dpoints_pixter.hh.

10.147.2.2 `template<typename I > template<typename Dps , typename Pref > mln::dpoints_bkd_pixter< I >::dpoints_bkd_pixter (const Generalized_Pixel< Pref > & pxl_ref, const Dps & dps) [inline]`

Constructor (using a generalized pixel).

Parameters

in	<i>pxl_ref</i>	Center (generalized) pixel to iterate around.
in	<i>dps</i>	An object (neighborhood or window) that can provide a set of delta-points.

Definition at line 353 of file `dpoints_pixter.hh`.

10.147.3 Member Function Documentation

10.147.3.1 `template<typename I > const I::value & mln::dpoints_bkd_pixter< I >::center_val () const [inline]`

The value around which this iterator moves.

Definition at line 368 of file `dpoints_pixter.hh`.

10.147.3.2 `template<typename I > void mln::dpoints_bkd_pixter< I >::invalidate () [inline]`

Invalidate the iterator.

Definition at line 437 of file `dpoints_pixter.hh`.

10.147.3.3 `template<typename I > bool mln::dpoints_bkd_pixter< I >::is_valid () const [inline]`

Test the iterator validity.

Definition at line 429 of file `dpoints_pixter.hh`.

10.147.3.4 `void mln::literator< dpoints_bkd_pixter< I > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.147.3.5 `template<typename I > void mln::dpoints_bkd_pixter< I >::start () [inline]`

Manipulation.

Start an iteration.

Definition at line 410 of file `dpoints_pixter.hh`.

10.147.3.6 `template<typename I > void mln::dpoints_bkd_pixter< I >::update () [inline]`

Force this iterator to update its location to take into account that its center point may have moved.

Definition at line 396 of file `dpoints_pixter.hh`.

10.148 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>
```

Inherits mln::Pixel_iterator< dpoints_fwd_pixter< I > >, and mln::internal::pixel_impl_< I, 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](#) (I &image, const Dps &dps, const Pref &p_ref)
Constructor (using an image).
- template<typename Dps , typename Pref >
[dpoints_fwd_pixter](#) (const [Generalized_Pixel](#)< Pref > &pxl_ref, const Dps &dps)
Constructor (using a generalized pixel).
- void [next](#) ()
Go to the next element.
- void [start](#) ()
Manipulation.
- void [invalidate](#) ()
Invalidate the iterator.
- bool [is_valid](#) () const
Test the iterator validity.
- void [update](#) ()
Force this iterator to update its location to take into account that its center point may have moved.

10.148.1 Detailed Description

```
template<typename I>class mln::dpoints_fwd_pixter< I >
```

A generic forward iterator on the pixels of a dpoint-based window or neighborhood.

Parameter *I* is the image type.

Definition at line 58 of file dpoints_pixter.hh.

10.148.2 Constructor & Destructor Documentation

10.148.2.1 `template<typename I > template<typename Dps , typename Pref > mln::dpoints_fwd_pixter< I >::dpoints_fwd_pixter (I &image, const Dps &dps, const Pref &p_ref) [inline]`

Constructor (using an image).

Parameters

in	<i>image</i>	The image to iterate over.
in	<i>dps</i>	An object (neighborhood or window) that can provide a set of delta-points.
in	<i>p_ref</i>	Center (resp. reference) point of the neighborhood (resp. window).

Definition at line 225 of file dpoints_pixter.hh.

10.148.2.2 `template<typename I > template<typename Dps , typename Pref > mln::dpoints_fwd_pixter< I >::dpoints_fwd_pixter (const Generalized_Pixel< Pref > & pxl_ref, const Dps & dps) [inline]`

Constructor (using a generalized pixel).

Parameters

in	<i>pxl_ref</i>	Center (generalized) pixel to iterate around.
in	<i>dps</i>	An object (neighborhood or window) that can provide a set of delta-points.

Definition at line 242 of file `dpoints_pixter.hh`.

10.148.3 Member Function Documentation

10.148.3.1 `template<typename I > const I::value & mln::dpoints_fwd_pixter< I >::center_val () const [inline]`

The value around which this iterator moves.

Definition at line 257 of file `dpoints_pixter.hh`.

10.148.3.2 `template<typename I > void mln::dpoints_fwd_pixter< I >::invalidate () [inline]`

Invalidate the iterator.

Definition at line 326 of file `dpoints_pixter.hh`.

10.148.3.3 `template<typename I > bool mln::dpoints_fwd_pixter< I >::is_valid () const [inline]`

Test the iterator validity.

Definition at line 318 of file `dpoints_pixter.hh`.

10.148.3.4 `void mln::literator< dpoints_fwd_pixter< I > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.148.3.5 `template<typename I > void mln::dpoints_fwd_pixter< I >::start () [inline]`

Manipulation.

Start an iteration.

Definition at line 299 of file `dpoints_pixter.hh`.

10.148.3.6 `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.

Definition at line 285 of file `dpoints_pixter.hh`.

10.149 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, dpsites_bkd_piter< V > >.

Public Member Functions

- `template<typename P > dpsites_bkd_piter (const V &v, const P &c)`
Constructor.
- `dpsites_bkd_piter ()`
Constructor without argument.
- `void next ()`
Go to the next element.

10.149.1 Detailed Description

```
template<typename V>class mln::dpsites_bkd_piter< V >
```

A generic backward iterator on points of windows and of neighborhoods.

The parameter `V` is the type of `std::vector` enclosing structure.

Definition at line 94 of file `dpsites_piter.hh`.

10.149.2 Constructor & Destructor Documentation

10.149.2.1 `template<typename V > template<typename P > mln::dpsites_bkd_piter< V >::dpsites_bkd_piter (const V & v, const P & c) [inline]`

Constructor.

Parameters

<code>in</code>	<code>v</code>	<code>Object</code> that can provide an array of delta-points.
<code>in</code>	<code>c</code>	Center point to iterate around.

Definition at line 217 of file `dpsites_piter.hh`.

10.149.2.2 `template<typename V > mln::dpsites_bkd_piter< V >::dpsites_bkd_piter () [inline]`

Constructor without argument.

Definition at line 210 of file `dpsites_piter.hh`.

10.149.3 Member Function Documentation

10.149.3.1 `void mln::Site_Iterator< dpsites_bkd_piter< V > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.150 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, dpsites_fwd_piter< V > >.

Public Member Functions

- `template<typename P >`
`dpsites_fwd_piter` (const V &v, const P &c)
Constructor.
- `dpsites_fwd_piter` ()
Constructor without argument.
- `void next` ()
Go to the next element.

10.150.1 Detailed Description

```
template<typename V>class mln::dpsites_fwd_piter< V >
```

A generic forward iterator on points of windows and of neighborhoods.

The parameter *V* is the type of `std::vector` enclosing structure.

Definition at line 48 of file `dpsites_piter.hh`.

10.150.2 Constructor & Destructor Documentation

10.150.2.1 `template<typename V > template<typename P > mln::dpsites_fwd_piter< V >::dpsites_fwd_piter (`
`const V & v, const P & c) [inline]`

Constructor.

Parameters

<i>in</i>	<i>v</i>	Object that can provide an array of delta-points.
<i>in</i>	<i>c</i>	Center point to iterate around.

Definition at line 149 of file `dpsites_piter.hh`.

10.150.2.2 `template<typename V > mln::dpsites_fwd_piter< V >::dpsites_fwd_piter () [inline]`

Constructor without argument.

Definition at line 142 of file dpsites_piter.hh.

10.150.3 Member Function Documentation

10.150.3.1 void mln::Site_Iterator< dpsites_fwd_piter< V > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.151 mln::Edge< E > Struct Template Reference

edge category flag type.

```
#include <edge.hh>
```

10.151.1 Detailed Description

```
template<typename E>struct mln::Edge< E >
```

edge category flag type.

Definition at line 52 of file edge.hh.

10.152 mln::edge_image< P, V, G > Class Template Reference

[Image](#) based on graph edges.

```
#include <edge_image.hh>
```

Inherits mln::pw::internal::image_base< fun::i2v::array< V >, p_edges< G, internal::efsite_selector< P, G >::site_function_t >, edge_image< P, V, G > >.

Public Types

- typedef [graph_elt_neighborhood](#)
< G, [p_edges](#)< G,
[site_function_t](#) > > [edge_nbh_t](#)
Neighborhood type.
- typedef [graph_elt_window](#)< G,
[p_edges](#)< G, [site_function_t](#) > > [edge_win_t](#)
Edge Window type.
- typedef G [graph_t](#)
The type of the underlying graph.
- typedef [edge_nbh_t](#) [nbh_t](#)
Default Neighborhood type.

- typedef
internal::efsite_selector< P,
G >::site_function_t site_function_t
Function mapping graph elements to sites.
- typedef edge_image
< tag::psite_< P >
, tag::value_< V >
, tag::graph_< G > > skeleton
Skeleton type.
- typedef edge_win_t win_t
Default Window type.

Public Member Functions

- edge_image ()
Constructors.
- rvalue operator() (unsigned e_id) const
Value accessors/operators overloads.

10.152.1 Detailed Description

template<typename P, typename V, typename G = util::graph>class mln::edge_image< P, V, G >

[Image](#) based on graph edges.

Definition at line 123 of file core/image/edge_image.hh.

10.152.2 Member Typedef Documentation

10.152.2.1 template<typename P, typename V, typename G = util::graph> typedef graph_elt -
neighborhood<G,p_edges<G,site_function_t> > mln::edge_image< P, V, G
>::edge_nbh_t

[Neighborhood](#) type.

Definition at line 153 of file core/image/edge_image.hh.

10.152.2.2 template<typename P, typename V, typename G = util::graph> typedef graph_elt_window<G,p_
edges<G,site_function_t> > mln::edge_image< P, V, G >::edge_win_t

[Edge Window](#) type.

Definition at line 151 of file core/image/edge_image.hh.

10.152.2.3 template<typename P, typename V, typename G = util::graph> typedef G mln::edge_image< P, V, G
>::graph_t

The type of the underlying graph.

Definition at line 138 of file core/image/edge_image.hh.

10.152.2.4 `template<typename P, typename V, typename G = util::graph> typedef edge_nbh_t mln::edge_image< P, V, G >::nbh_t`

Default [Neighborhood](#) type.

Definition at line 159 of file core/image/edge_image.hh.

10.152.2.5 `template<typename P, typename V, typename G = util::graph> typedef internal::efsite_selector<P,G>::site_function_t mln::edge_image< P, V, G >::site_function_t`

[Function](#) mapping graph elements to sites.

Definition at line 147 of file core/image/edge_image.hh.

10.152.2.6 `template<typename P, typename V, typename G = util::graph> typedef edge_image< tag::psite_<P>, tag::value_<V>, tag::graph_<G> > mln::edge_image< P, V, G >::skeleton`

Skeleton type.

Definition at line 143 of file core/image/edge_image.hh.

10.152.2.7 `template<typename P, typename V, typename G = util::graph> typedef edge_win_t mln::edge_image< P, V, G >::win_t`

Default [Window](#) type.

Definition at line 156 of file core/image/edge_image.hh.

10.152.3 Constructor & Destructor Documentation

10.152.3.1 `template<typename P, typename V, typename G> edge_image< P, V, G >::edge_image () [inline]`

Constructors.

Definition at line 248 of file core/image/edge_image.hh.

10.152.4 Member Function Documentation

10.152.4.1 `template<typename P, typename V, typename G> edge_image< P, V, G >::rvalue edge_image< P, V, G >::operator() (unsigned e_id) const`

[Value](#) accessors/operators overloads.

Definition at line 302 of file core/image/edge_image.hh.

10.153 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, box< I::site >, extended< I > >`, `mlc_not_equalmln_trait_image_ext_domainI`, and `check_t`.

Public Types

- typedef tag::image_< I > [skeleton](#)
Skeleton.
- typedef I::value [value](#)
Value type.

Public Member Functions

- const [box](#)< typename I::site > & [domain](#) () const
Give the definition domain.
- [extended](#) ()
Constructor without argument.
- [extended](#) (I &ima, const [box](#)< typename I::site > &b)
Constructor.

10.153.1 Detailed Description

`template<typename I>struct mln::extended< I >`

Makes an image become restricted by a point set.

Definition at line 93 of file extended.hh.

10.153.2 Member Typedef Documentation

10.153.2.1 `template<typename I> typedef tag::image_<I> mln::extended< I >::skeleton`

Skeleton.

Definition at line 103 of file extended.hh.

10.153.2.2 `template<typename I> typedef I::value mln::extended< I >::value`

[Value](#) type.

Definition at line 100 of file extended.hh.

10.153.3 Constructor & Destructor Documentation

10.153.3.1 `template<typename I> extended< I >::extended () [inline]`

Constructor without argument.

Definition at line 170 of file extended.hh.

10.153.3.2 `template<typename I> extended< I >::extended (I & ima, const box< typename I::site > &b) [inline]`

Constructor.

Definition at line 176 of file extended.hh.

10.153.4 Member Function Documentation

10.153.4.1 `template<typename I> const box< typename I::site > & extended< I >::domain () const` `[inline]`

Give the definition domain.

Definition at line 193 of file extended.hh.

10.154 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, extension_fun< I, F > >, result, and check_t.

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](#) ()
Constructor without argument.
- [extension_fun](#) (I &ima, const F &fun)
Constructor from an image ima and a function fun.
- template<typename P>
 bool [has](#) (const P &p) const
Test if p is valid.
- I::value [operator\(\)](#) (const typename I::psite &p) const
Read-only access to the image value located at site p.
- internal::morpher_lvalue_< I >::ret [operator\(\)](#) (const typename I::psite &p)
Read-write access to the image value located at site p.

10.154.1 Detailed Description

`template<typename I, typename F>class mln::extension_fun< I, F >`

Extends the domain of an image with a function.

Definition at line 100 of file extension_fun.hh.

10.154.2 Member Typedef Documentation

10.154.2.1 `template<typename I, typename F> typedef I::value mln::extension_fun< I, F >::rvalue`

Return type of read-only access.

Definition at line 113 of file `extension_fun.hh`.

10.154.2.2 `template<typename I, typename F> typedef extension_fun< tag::image_<I>, tag::function_<F> > mln::extension_fun< I, F >::skeleton`

Skeleton.

Definition at line 107 of file `extension_fun.hh`.

10.154.2.3 `template<typename I, typename F> typedef I::value mln::extension_fun< I, F >::value`

[Image](#) value type.

Definition at line 110 of file `extension_fun.hh`.

10.154.3 Constructor & Destructor Documentation

10.154.3.1 `template<typename I, typename F> extension_fun< I, F >::extension_fun () [inline]`

Constructor without argument.

Definition at line 179 of file `extension_fun.hh`.

10.154.3.2 `template<typename I, typename F> extension_fun< I, F >::extension_fun (I & ima, const F & fun) [inline]`

Constructor from an image `ima` and a function `fun`.

Definition at line 185 of file `extension_fun.hh`.

10.154.4 Member Function Documentation

10.154.4.1 `template<typename I, typename F> const F & extension_fun< I, F >::extension () const [inline]`

Give the extension function.

Definition at line 244 of file `extension_fun.hh`.

10.154.4.2 `template<typename I, typename F> template<typename P> bool extension_fun< I, F >::has (const P & p) const [inline]`

Test if `p` is valid.

It returns always true, assuming that the function is valid for any `p`.

Definition at line 202 of file `extension_fun.hh`.

10.154.4.3 `template<typename I, typename F> I::value extension_fun< I, F >::operator() (const typename I::psite & p) const [inline]`

Read-only access to the image value located at site `p`;

Definition at line 210 of file extension_fun.hh.

```
10.154.4.4  template<typename I , typename F > internal::morpher_lvalue_< I >::ret extension_fun< I, F >::operator() (
            const typename I::psite & p )  [inline]
```

Read-write access to the image value located at site *p*.

Definition at line 224 of file extension_fun.hh.

10.155 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, extension_ima< I, J > >, value, and check_t.

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](#) ()
Constructor without argument.
- [extension_ima](#) (I &ima, const J &ext)
Constructor from an image ima and a function ext.
- template<typename P >
 bool [has](#) (const P &p) const
Test if p is valid.
- I::value [operator\(\)](#) (const typename I::psite &p) const
Read-only access to the image value located at site p;.
- internal::morpher_lvalue_< I >::ret [operator\(\)](#) (const typename I::psite &p)
Read-write access to the image value located at site p.

10.155.1 Detailed Description

```
template<typename I, typename J>class mln::extension_ima< I, J >
```

Extends the domain of an image with an image.

Definition at line 97 of file extension_ima.hh.

10.155.2 Member Typedef Documentation

10.155.2.1 `template<typename I, typename J> typedef I::value mln::extension_ima< I, J >::rvalue`

Return type of read-only access.

Definition at line 111 of file `extension_ima.hh`.

10.155.2.2 `template<typename I, typename J> typedef extension_ima< tag::image_<I>, tag::ext_<J> > mln::extension_ima< I, J >::skeleton`

Skeleton.

Definition at line 105 of file `extension_ima.hh`.

10.155.2.3 `template<typename I, typename J> typedef I::value mln::extension_ima< I, J >::value`

[Image](#) value type.

Definition at line 108 of file `extension_ima.hh`.

10.155.3 Constructor & Destructor Documentation

10.155.3.1 `template<typename I, typename J> extension_ima< I, J >::extension_ima () [inline]`

Constructor without argument.

Definition at line 173 of file `extension_ima.hh`.

10.155.3.2 `template<typename I, typename J> extension_ima< I, J >::extension_ima (I & ima, const J & ext) [inline]`

Constructor from an image `ima` and a function `ext`.

Definition at line 179 of file `extension_ima.hh`.

10.155.4 Member Function Documentation

10.155.4.1 `template<typename I, typename J> const J & extension_ima< I, J >::extension () const [inline]`

Read-only access to the extension domain (image).

Definition at line 244 of file `extension_ima.hh`.

10.155.4.2 `template<typename I, typename J> template<typename P> bool extension_ima< I, J >::has (const P & p) const [inline]`

Test if `p` is valid.

Definition at line 196 of file `extension_ima.hh`.

10.155.4.3 `template<typename I, typename J> I::value extension_ima< I, J >::operator() (const typename I::psite & p) const [inline]`

Read-only access to the image value located at site `p`.

Definition at line 208 of file `extension_ima.hh`.

10.155.4.4 `template<typename I, typename J> internal::morpher_lvalue_< I >::ret extension_ima< I, J >::operator() (const typename I::psite & p) [inline]`

Read-write access to the image value located at site *p*.

Definition at line 223 of file extension_ima.hh.

10.156 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, 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](#) ()
Constructor without argument.
- [extension_val](#) (I &ima, const typename I::value &val)
Constructor from an image ima and a value val.
- template<typename P >
 bool [has](#) (const P &p) const
Test if p is valid. It returns always true.
- I::value [operator\(\)](#) (const typename I::psite &p) const
Read-only access to the image value located at site p.
- internal::morpher_lvalue_< I >::ret [operator\(\)](#) (const typename I::psite &p)
Read-write access to the image value located at site p.

10.156.1 Detailed Description

`template<typename I>class mln::extension_val< I >`

Extends the domain of an image with a value.

Definition at line 99 of file extension_val.hh.

10.156.2 Member Typedef Documentation

10.156.2.1 `template<typename I> typedef I::value mln::extension_val<I>::rvalue`

Return type of read-only access.

Definition at line 111 of file `extension_val.hh`.

10.156.2.2 `template<typename I> typedef extension_val< tag::image_<I> > mln::extension_val<I>::skeleton`

Skeleton.

Definition at line 105 of file `extension_val.hh`.

10.156.2.3 `template<typename I> typedef I::value mln::extension_val<I>::value`

[Image](#) value type.

Definition at line 108 of file `extension_val.hh`.

10.156.3 Constructor & Destructor Documentation

10.156.3.1 `template<typename I> extension_val<I>::extension_val() [inline]`

Constructor without argument.

Definition at line 177 of file `extension_val.hh`.

10.156.3.2 `template<typename I> extension_val<I>::extension_val(I & ima, const typename I::value & val) [inline]`

Constructor from an image `ima` and a value `val`.

Definition at line 183 of file `extension_val.hh`.

10.156.4 Member Function Documentation

10.156.4.1 `template<typename I> void extension_val<I>::change_extension(const typename I::value & val) [inline]`

Change the value of the extension domain.

Definition at line 248 of file `extension_val.hh`.

10.156.4.2 `template<typename I> const I::value & extension_val<I>::extension() const [inline]`

Read-only access to the value of the extension domain.

Definition at line 239 of file `extension_val.hh`.

10.156.4.3 `template<typename I> template<typename P> bool extension_val<I>::has(const P & p) const [inline]`

Test if `p` is valid. It returns always true.

Definition at line 200 of file `extension_val.hh`.

10.156.4.4 `template<typename I > I::value extension_val< I >::operator() (const typename I::psite & p) const`
`[inline]`

Read-only access to the image value located at site *p*.

Definition at line 208 of file extension_val.hh.

10.156.4.5 `template<typename I > internal::morpher_lvalue< I >::ret extension_val< I >::operator() (const typename I::psite & p)` `[inline]`

Read-write access to the image value located at site *p*.

Definition at line 221 of file extension_val.hh.

10.157 mln::flat_image< T, S > Struct Template Reference

[Image](#) with a single value.

`#include <flat_image.hh>`

Inherits `mln::internal::image_primary< T, S, flat_image< T, S > >`.

Public Types

- typedef `T & lvalue`
Return type of read-write access.
- typedef `const T & rvalue`
Return type of read-only access.
- typedef `flat_image`
`< tag::value_< T >`
`, tag::domain_< S > > skeleton`
Skeleton.
- typedef `T value`
Value associated type.

Public Member Functions

- `const S & domain () const`
Give the definition domain.
- `flat_image ()`
Constructor without argument.
- `flat_image (const T &val, const S &pset)`
Constructor.
- `bool has (const typename S::psite &p) const`
*Test if *p* is valid: always return true.*
- `const T & operator() (const typename S::psite &p) const`
*Read-only access to the image value located at point *p*.*
- `T & operator() (const typename S::psite &p)`
*Read-write access to the image value located at point *p*.*

10.157.1 Detailed Description

`template<typename T, typename S> struct mln::flat_image< T, S >`

[Image](#) with a single value.

Definition at line 106 of file flat_image.hh.

10.157.2 Member Typedef Documentation

10.157.2.1 `template<typename T, typename S> typedef T& mln::flat_image< T, S >::lvalue`

Return type of read-write access.

Definition at line 119 of file flat_image.hh.

10.157.2.2 `template<typename T, typename S> typedef const T& mln::flat_image< T, S >::rvalue`

Return type of read-only access.

Definition at line 116 of file flat_image.hh.

10.157.2.3 `template<typename T, typename S> typedef flat_image< tag::value_<T>, tag::domain_<S> > mln::flat_image< T, S >::skeleton`

Skeleton.

Definition at line 109 of file flat_image.hh.

10.157.2.4 `template<typename T, typename S> typedef T mln::flat_image< T, S >::value`

[Value](#) associated type.

Definition at line 113 of file flat_image.hh.

10.157.3 Constructor & Destructor Documentation

10.157.3.1 `template<typename T , typename S > flat_image< T, S >::flat_image () [inline]`

Constructor without argument.

Definition at line 192 of file flat_image.hh.

10.157.3.2 `template<typename T , typename S > flat_image< T, S >::flat_image (const T & val, const S & pset) [inline]`

Constructor.

Definition at line 198 of file flat_image.hh.

10.157.4 Member Function Documentation

10.157.4.1 `template<typename T , typename S > const S & flat_image< T, S >::domain () const [inline]`

Give the definition domain.

Definition at line 215 of file flat_image.hh.

```
10.157.4.2  template<typename T , typename S > bool flat_image< T, S >::has ( const typename S::psite & p ) const
           [inline]
```

Test if p is valid: always return true.

Definition at line 223 of file flat_image.hh.

```
10.157.4.3  template<typename T , typename S > const T & flat_image< T, S >::operator() ( const typename S::psite & p )
           const [inline]
```

Read-only access to the image value located at point p.

Definition at line 231 of file flat_image.hh.

```
10.157.4.4  template<typename T , typename S > T & flat_image< T, S >::operator() ( const typename S::psite & p )
           [inline]
```

Read-write access to the image value located at point p.

Definition at line 240 of file flat_image.hh.

10.158 mln::fun::from_accu< A > Struct Template Reference

Wrap an accumulator into a function.

```
#include <from_accu.hh>
```

Inherits mln::fun::unary_param< F, Param, Storage, E >.

10.158.1 Detailed Description

```
template<typename A>struct mln::fun::from_accu< A >
```

Wrap an accumulator into a function.

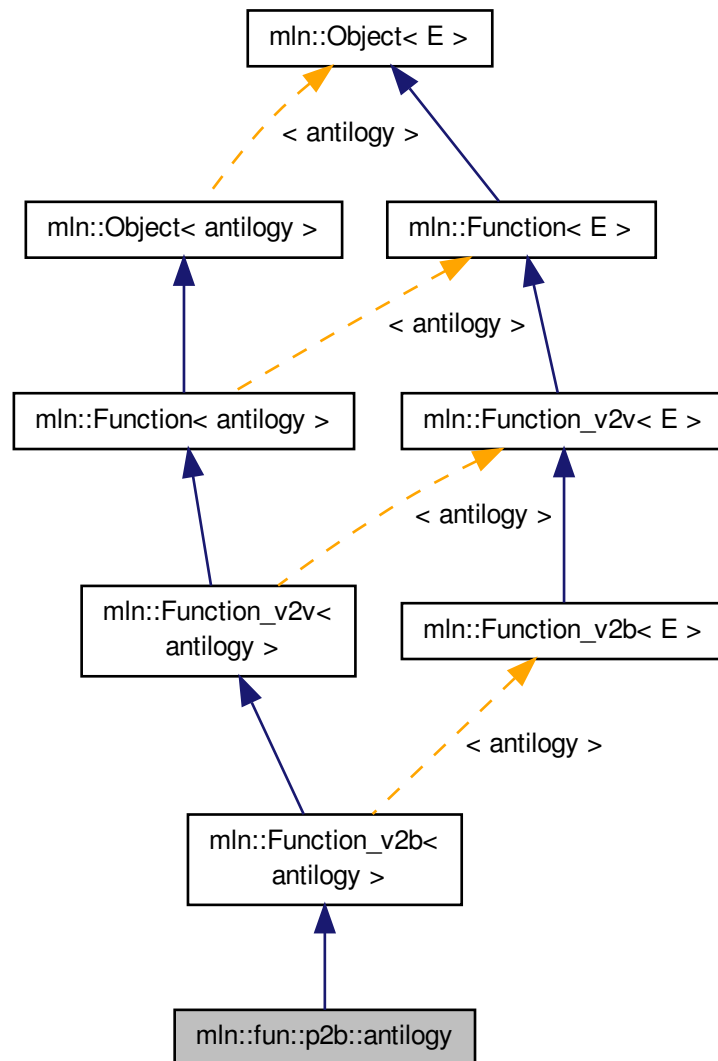
Definition at line 49 of file from_accu.hh.

10.159 mln::fun::n2v::white_gaussian< V > Struct Template Reference

Generate a White Gaussian Noise.

```
#include <white_gaussian.hh>
```


Inheritance diagram for mln::fun::p2b::antilogy:



10.160.1 Detailed Description

A **p2b** function always returning `false`.

A simpler name would be 'false', but this is not a valid C++ identifier, as `false` is a keyword of the language.

Definition at line 50 of file `antilogy.hh`.

10.161 mln::fun::p2b::tautology Struct Reference

A **p2b** function always returning `true`.

```
#include <tautology.hh>
```


[illegible]

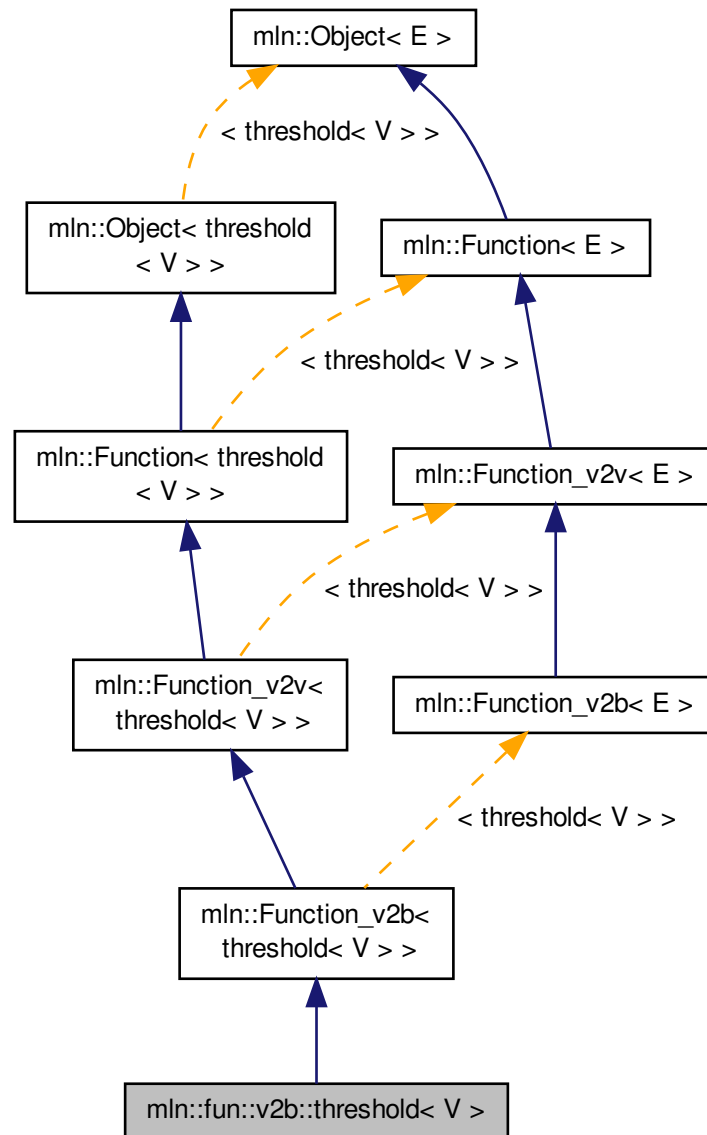
Definition at line 47 of file Inot.hh.

10.163 mln::fun::v2b::threshold< V > Struct Template Reference

Threshold function.

```
#include <threshold.hh>
```

Inheritance diagram for mln::fun::v2b::threshold< V >:



10.163.1 Detailed Description

```
template<typename V>struct mln::fun::v2b::threshold< V >
```

Threshold function.

$f(v) = (v \geq \text{threshold})$.

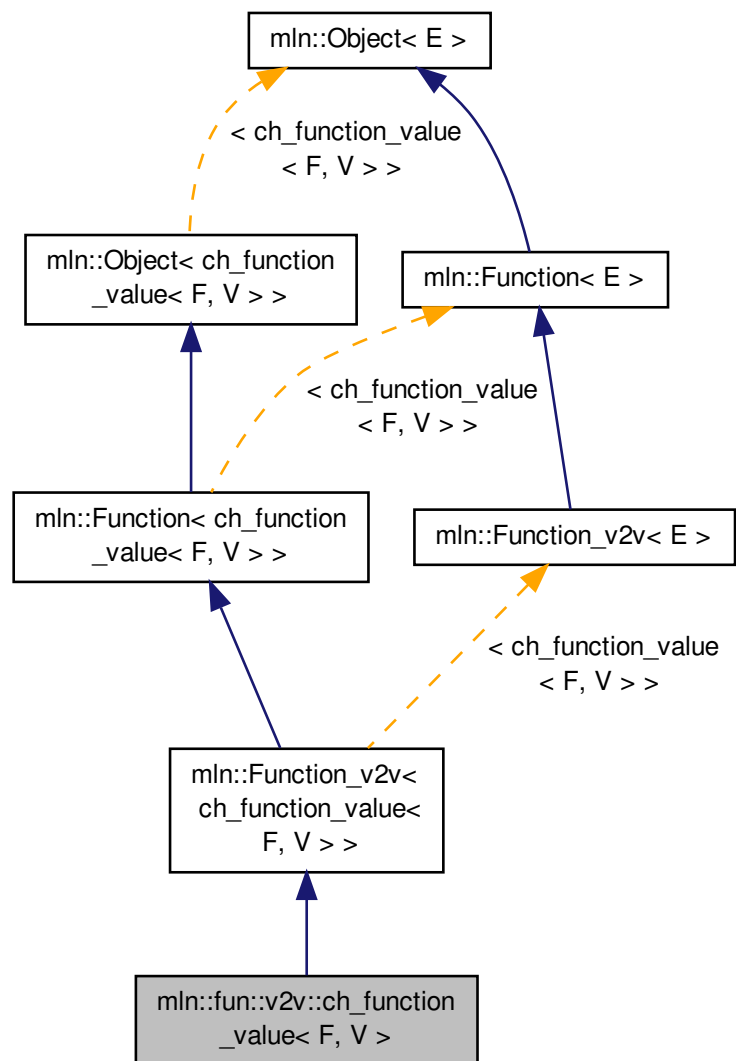
Definition at line 48 of file fun/v2b/threshold.hh.

10.164 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 >`:



10.164.1 Detailed Description

```
template<typename F, typename V>class mln::fun::v2v::ch_function_value< F, V >
```

Wrap a function `v2v` and convert its result to another type.

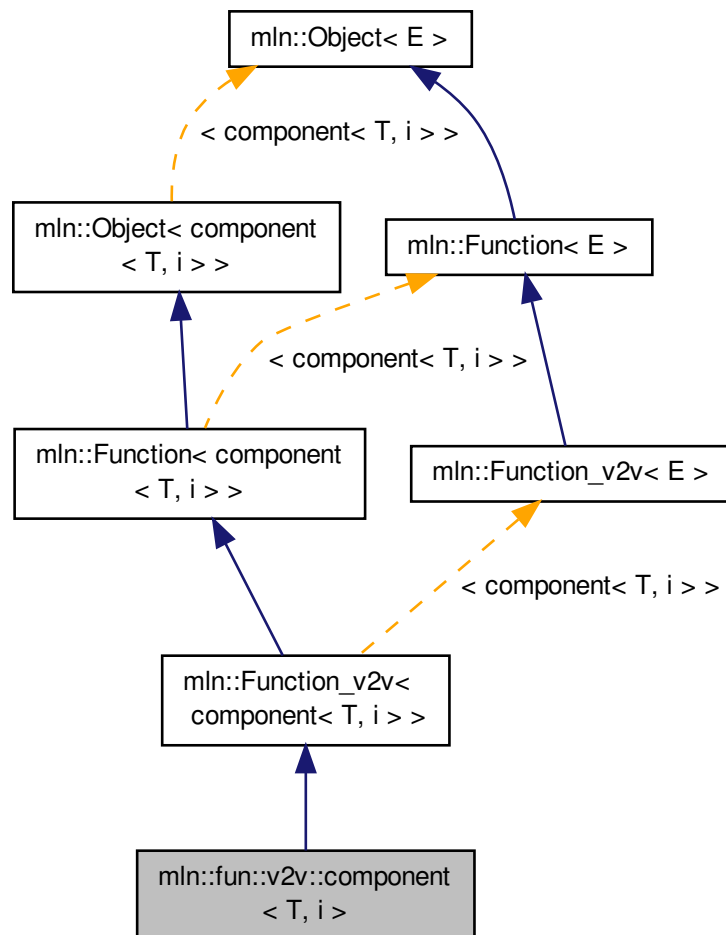
Definition at line 52 of file `fun/v2v/ch_function_value.hh`.

10.165 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 >`:



10.165.1 Detailed Description

```
template<typename T, unsigned i>struct mln::fun::v2v::component< T, i >
```

Functor that accesses the i-th component of a value.

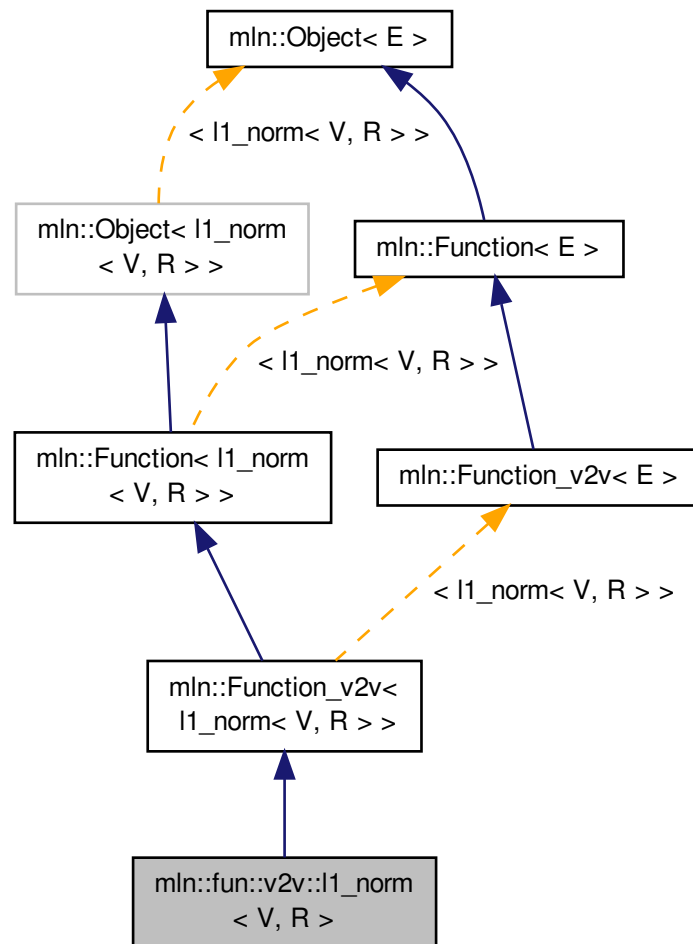
Definition at line 51 of file component.hh.

10.166 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 >:



10.166.1 Detailed Description

```
template<typename V, typename R>struct mln::fun::v2v::l1_norm< V, R >
```

L1-norm.

V is the type of input values; R is the result type.

See Also

[mln::norm::l1.](#)

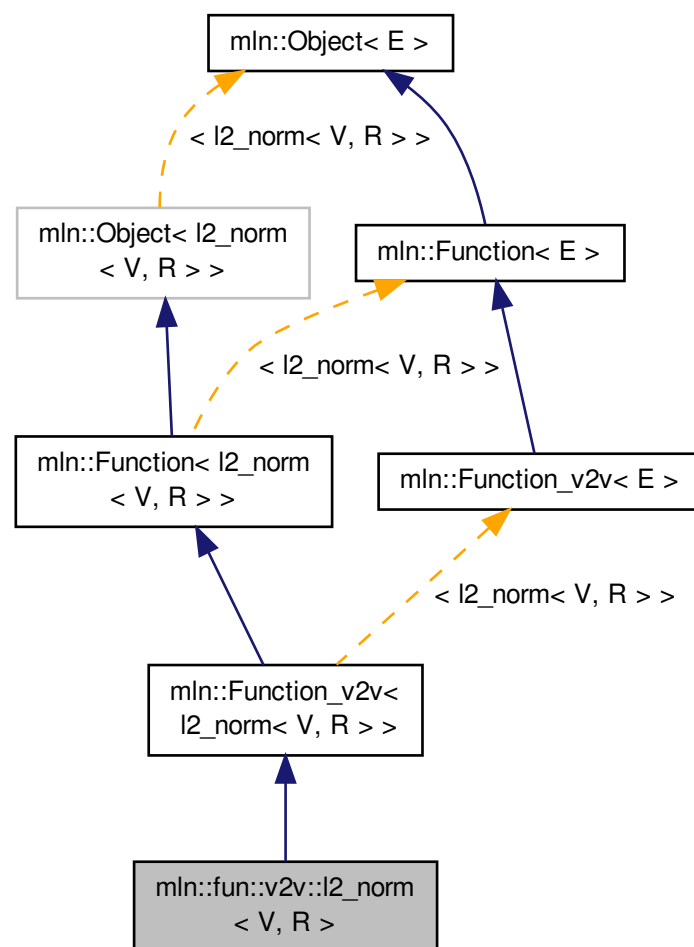
Definition at line 57 of file v2v/norm.hh.

10.167 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 >:



10.167.1 Detailed Description

```
template<typename V, typename R>struct mln::fun::v2v::l2_norm< V, R >
```

L2-norm.

V is the type of input values; R is the result type.

See Also

mln::norm::l2.

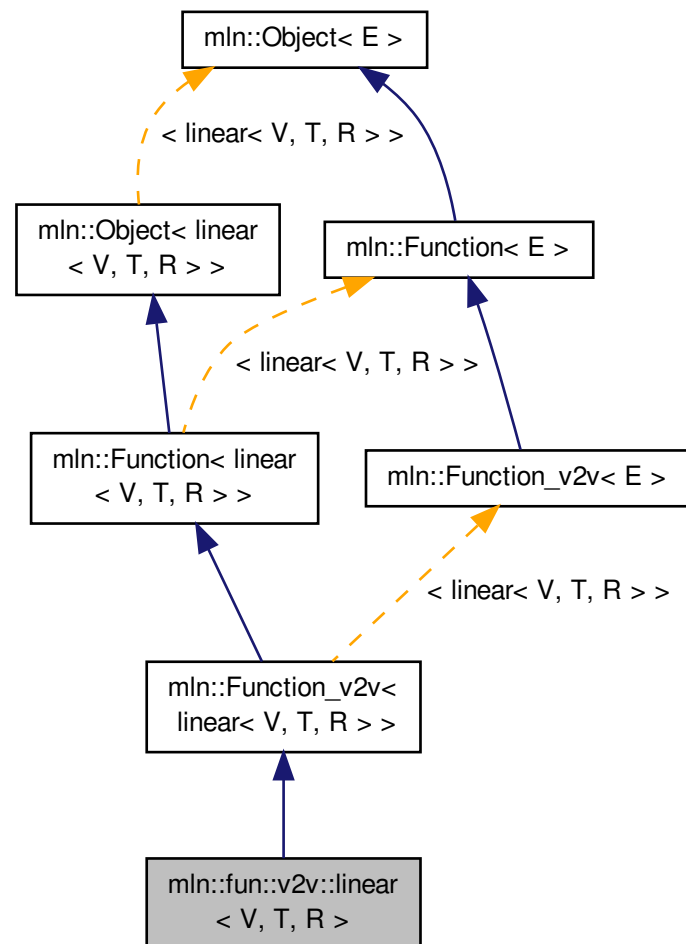
Definition at line 69 of file v2v/norm.hh.

10.168 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 >:



10.168.1 Detailed Description

```
template<typename V, typename T = V, typename R = T>struct mln::fun::v2v::linear< V, T, R >
```

Linear function. $f(v) = a * v + b$. V is the type of input values; T is the type used to compute the result; R is the result type.

By default, T is V and R is T .

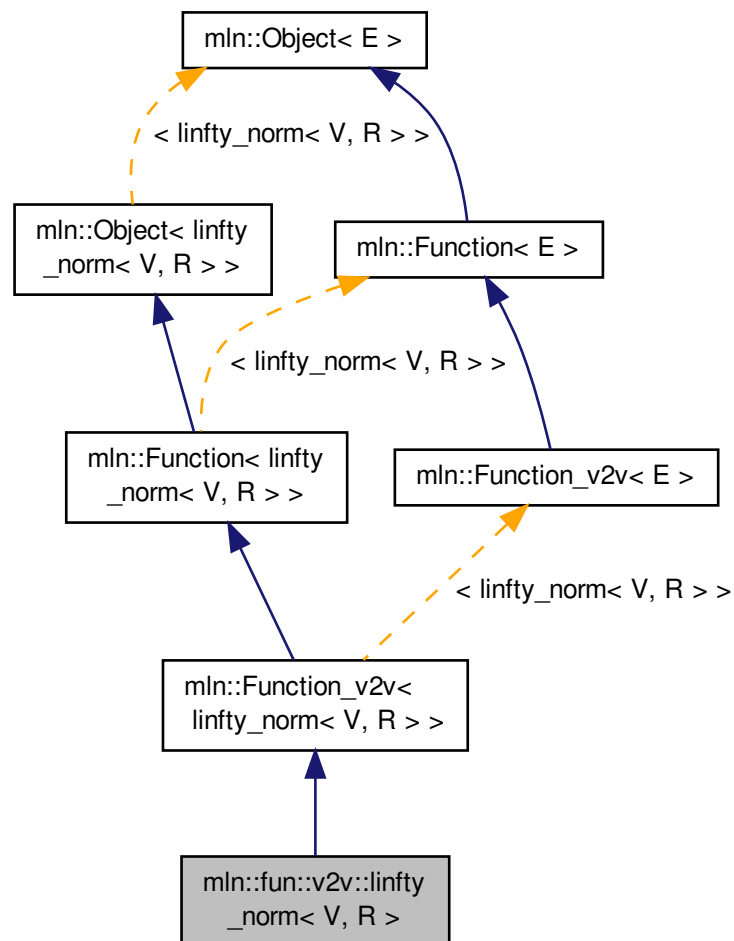
Definition at line 55 of file v2v/linear.hh.

10.169 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 >:



10.169.1 Detailed Description

```
template<typename V, typename R>struct mln::fun::v2v::linfty_norm< V, R >
```

L-infty norm.

V is the type of input values; R is the result type.

See Also

[mln::norm::linfty](#).

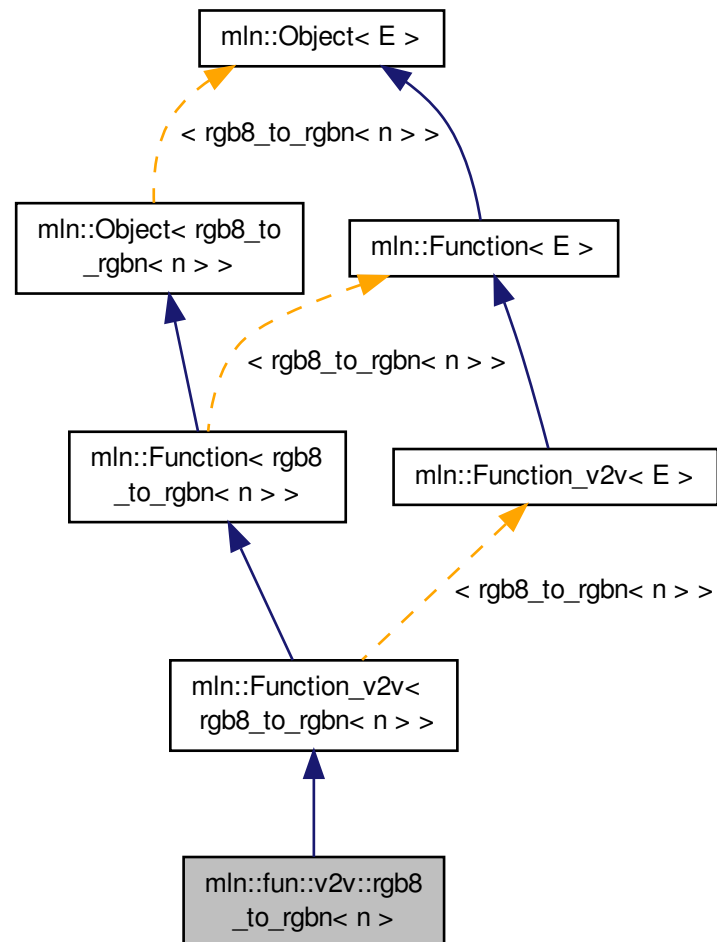
Definition at line 81 of file v2v/norm.hh.

10.170 mln::fun::v2v::rgb8_to_rgbn< n > Struct Template Reference

Convert a rgb8 value to a rgn, $n < 8$.

```
#include <rgb8_to_rgbn.hh>
```

Inheritance diagram for mln::fun::v2v::rgb8_to_rgbn< n >:



Public Member Functions

- [result operator\(\)](#) (const [argument](#) &c) const
Convert a *rgb8* value to a *rgn*, $n < 8$.

10.170.1 Detailed Description

template<unsigned n> struct mln::fun::v2v::rgb8_to_rgn< n >

Convert a *rgb8* value to a *rgn*, $n < 8$.

Parameters

<i>n</i>	defines the output quantification used for the transformation.
----------	--

Definition at line 56 of file *rgb8_to_rgn.hh*.

10.170.2 Member Function Documentation

10.170.2.1 template<unsigned n> *rgb8_to_rgn*< n >::result mln::fun::v2v::rgb8_to_rgn< n >::operator() (const *argument* & *c*) const

Convert a *rgb8* value to a *rgn*, $n < 8$.

Parameters

<i>in</i>	<i>v</i>	the <i>rgb8</i> value to convert.
-----------	----------	-----------------------------------

Conversion is done by computing the size by which we divide each *rgb* component.

Parameters

<i>n</i>	defines the output quantification used for the transformation.
----------	--

Definition at line 83 of file *rgb8_to_rgn.hh*.

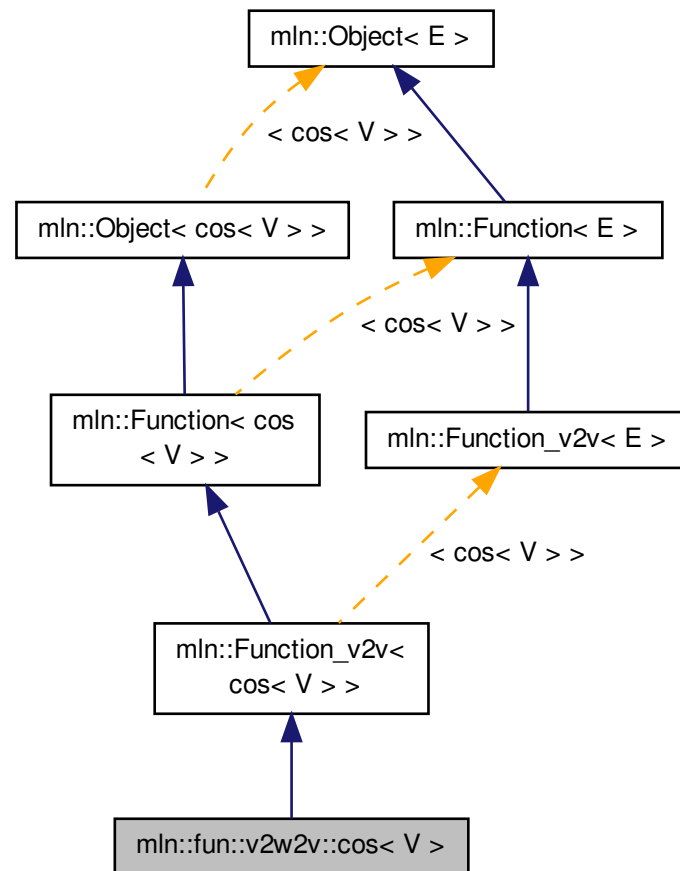
References *mln::value::rgb*< n >::red().

10.171 mln::fun::v2w2v::cos< V > Struct Template Reference

Cosinus bijective functor.

```
#include <cos.hh>
```


Inheritance diagram for mln::fun::v2w2v::cos< V >:



10.171.1 Detailed Description

```
template<typename V>struct mln::fun::v2w2v::cos< V >
```

Cosinus bijective functor.

`V` is the type of input values and the result type.

See Also

`mln::math::cos`.

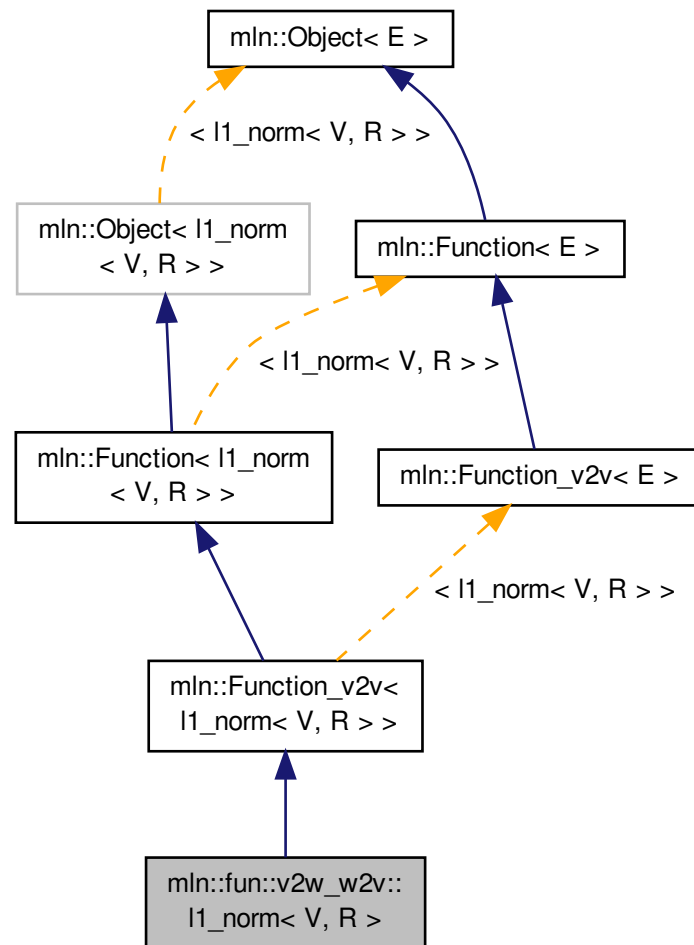
Definition at line 59 of file `fun/v2w2v/cos.hh`.

10.172 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 >`:



10.172.1 Detailed Description

```
template<typename V, typename R>struct mln::fun::v2w_w2v::l1_norm< V, R >
```

L1-norm.

`V` is the type of input values; `R` is the result type.

See Also

[mln::norm::l1](#).

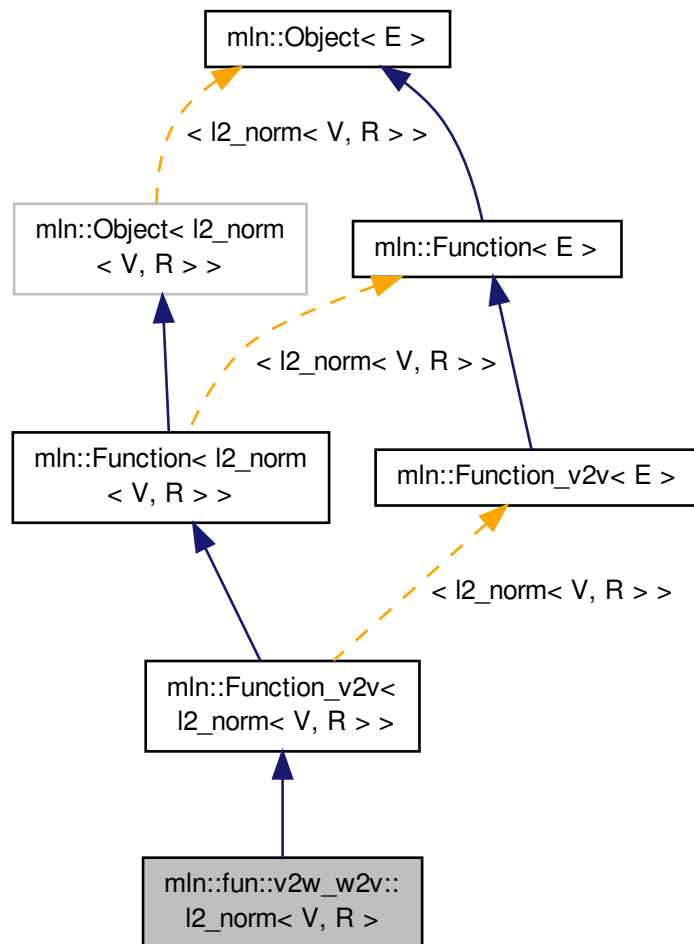
Definition at line 56 of file `v2w_w2v/norm.hh`.

10.173 `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 >:



10.173.1 Detailed Description

```
template<typename V, typename R>struct mln::fun::v2w_w2v::l2_norm< V, R >
```

L2-norm.

V is the type of input values; R is the result type.

See Also

`mln::norm::l2`.

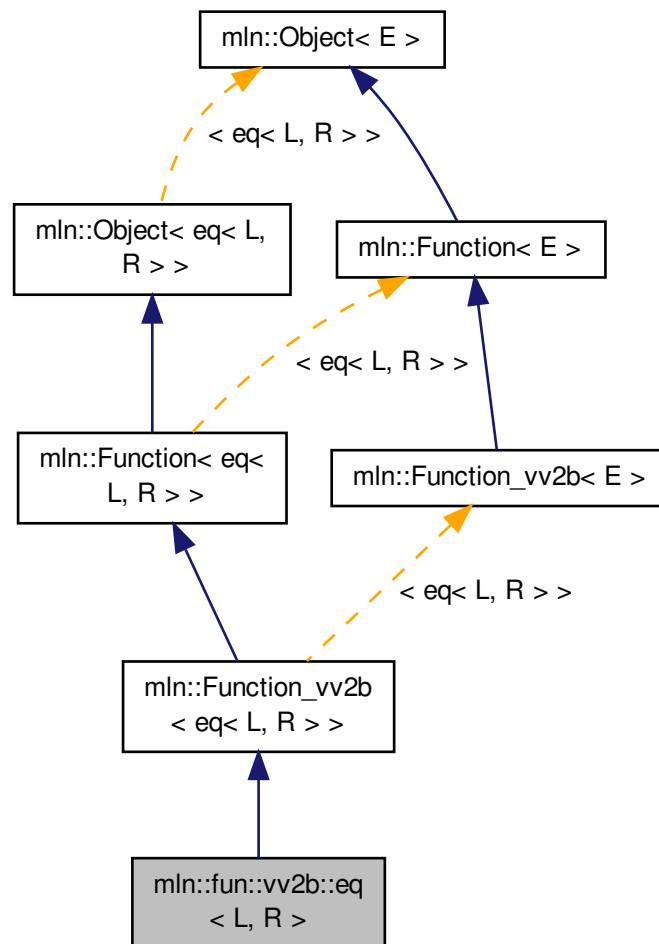
Definition at line 69 of file `v2w_w2v/norm.hh`.

10.175 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 >:



10.175.1 Detailed Description

```
template<typename L, typename R = L>struct mln::fun::vv2b::eq< L, R >
```

Functor computing equal between two values.

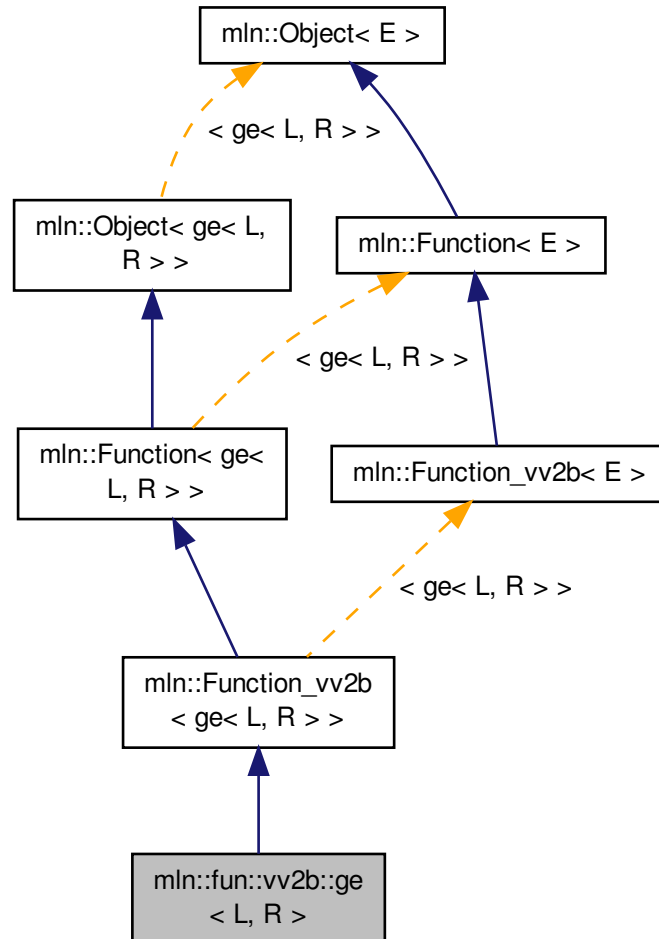
Definition at line 48 of file `fun/vv2b/eq.hh`.

10.176 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 >`:



10.176.1 Detailed Description

```
template<typename L, typename R = L>struct mln::fun::vv2b::ge< L, R >
```

Functor computing "greater or equal than" between two values.

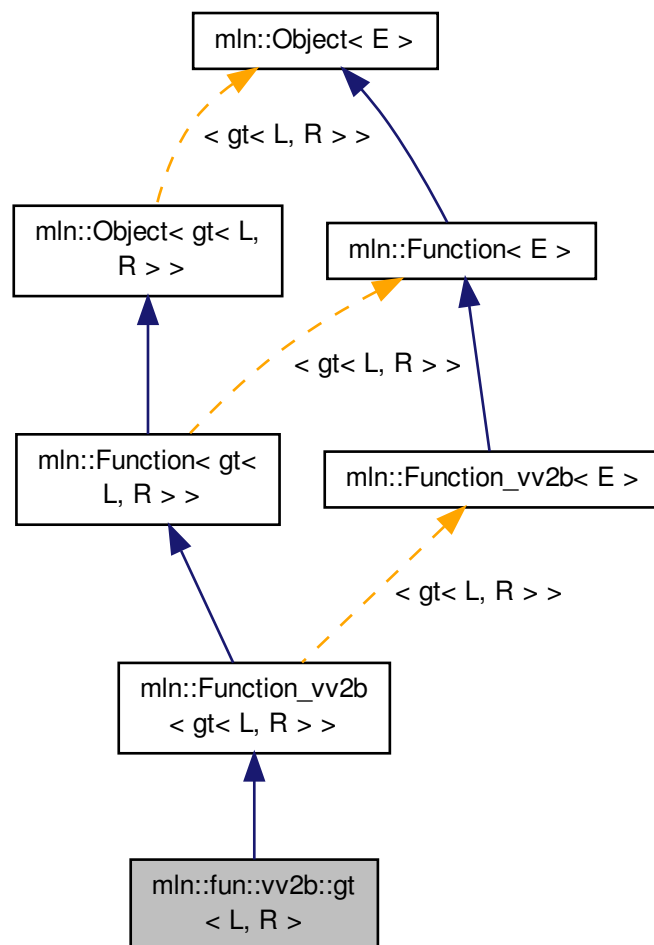
Definition at line 48 of file `ge.hh`.

10.177 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 >:



10.177.1 Detailed Description

```
template<typename L, typename R = L>struct mln::fun::vv2b::gt< L, R >
```

Functor computing "greater than" between two values.

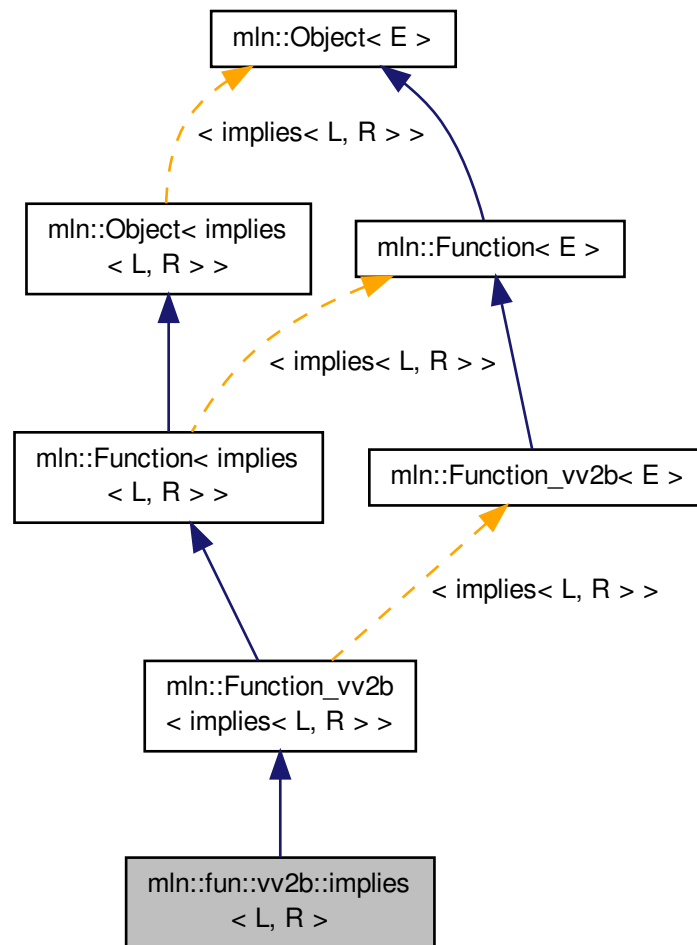
Definition at line 48 of file `gt.hh`.

10.178 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 >`:



10.178.1 Detailed Description

```
template<typename L, typename R = L>struct mln::fun::vv2b::implies< L, R >
```

Functor computing logical-implies between two values.

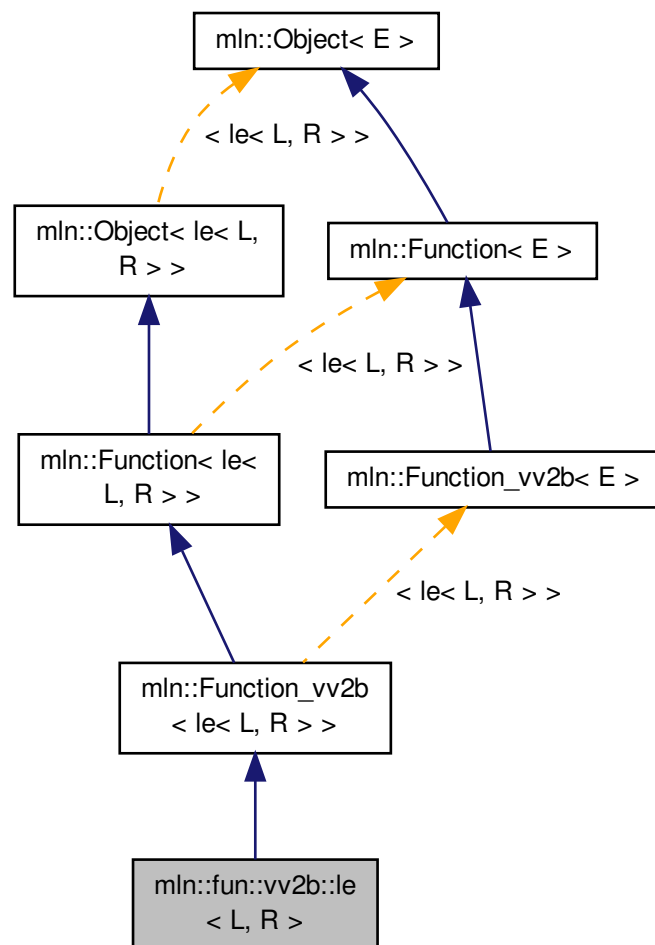
Definition at line 48 of file `implies.hh`.

10.179 `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 >:



10.179.1 Detailed Description

```
template<typename L, typename R = L>struct mln::fun::vv2b::le< L, R >
```

Functor computing "lower or equal than" between two values.

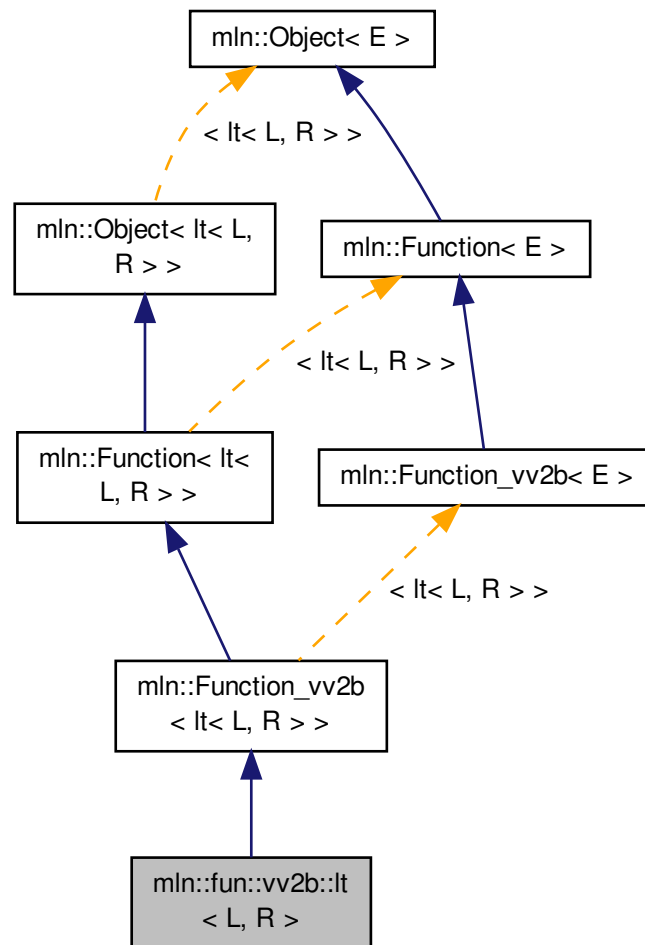
Definition at line 48 of file `le.hh`.

10.180 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 >`:



10.180.1 Detailed Description

```
template<typename L, typename R = L>struct mln::fun::vv2b::lt< L, R >
```

Functor computing "lower than" between two values.

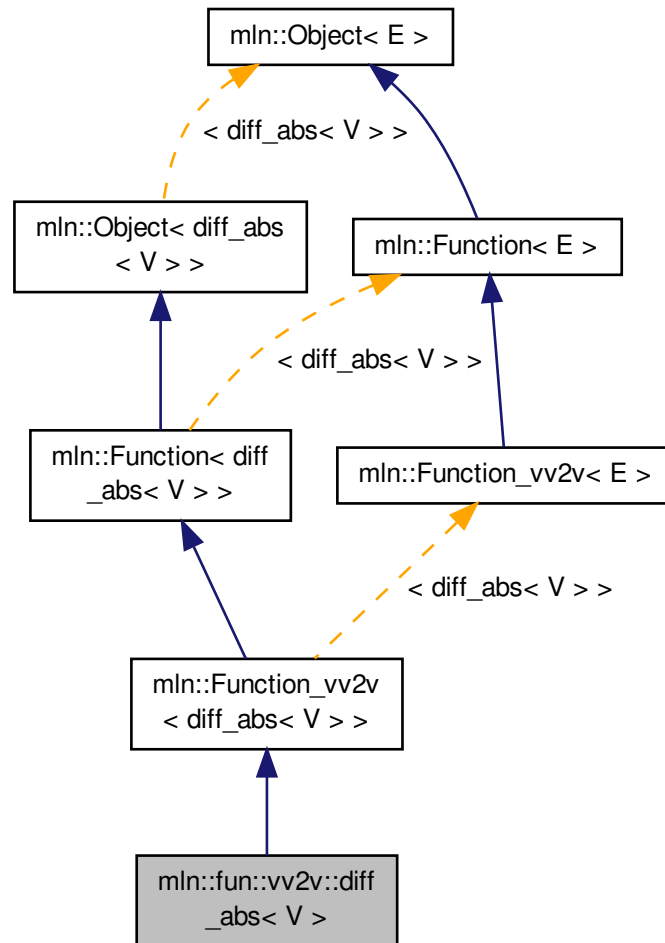
Definition at line 48 of file `lt.hh`.

10.181 `mln::fun::vv2v::diff_abs< V >` Struct Template Reference

A functor computing the `diff_abs`imum of two values.

```
#include <diff_abs.hh>
```

Inheritance diagram for mln::fun::vv2v::diff_abs< V >:



10.181.1 Detailed Description

```
template<typename V>struct mln::fun::vv2v::diff_abs< V >
```

A functor computing the diff_absimum of two values.

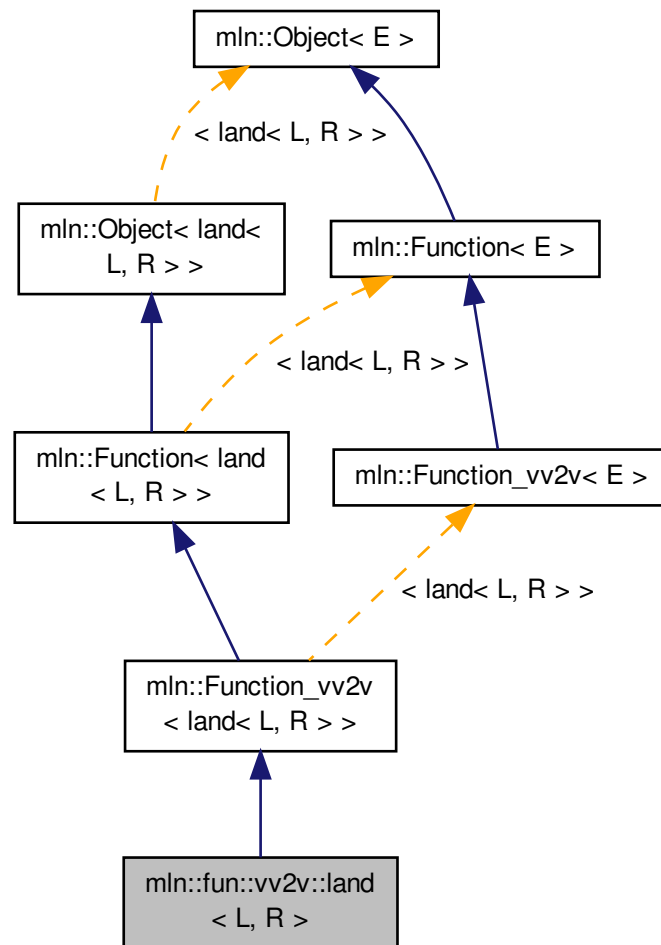
Definition at line 50 of file fun/vv2v/diff_abs.hh.

10.182 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 >`:



10.182.1 Detailed Description

```
template<typename L, typename R = L>struct mln::fun::vv2v::land< L, R >
```

Functor computing logical-and between two values.

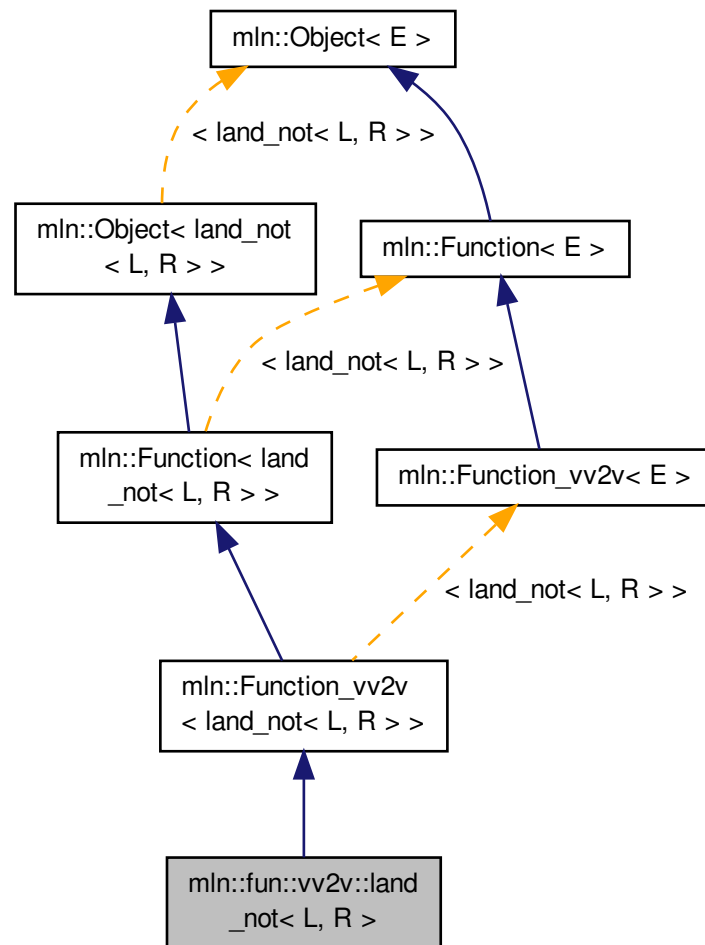
Definition at line 48 of file `fun/vv2v/land.hh`.

10.183 `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 >:



10.183.1 Detailed Description

```
template<typename L, typename R = L>struct mln::fun::vv2v::land_not< L, R >
```

Functor computing logical and-not between two values.

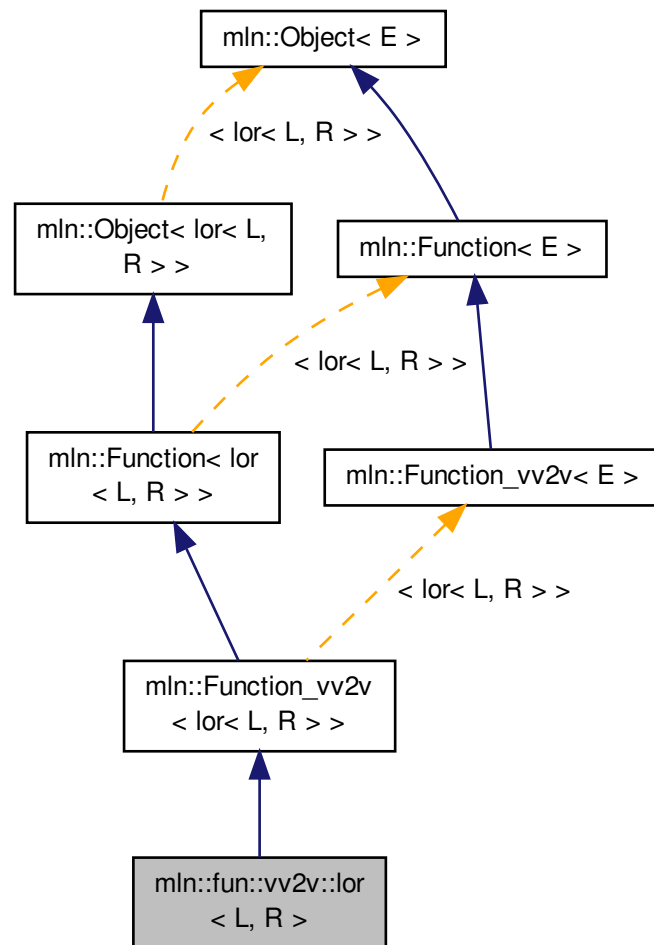
Definition at line 49 of file `land_not.hh`.

10.184 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 >`:



10.184.1 Detailed Description

```
template<typename L, typename R = L>struct mln::fun::vv2v::lor< L, R >
```

Functor computing logical-or between two values.

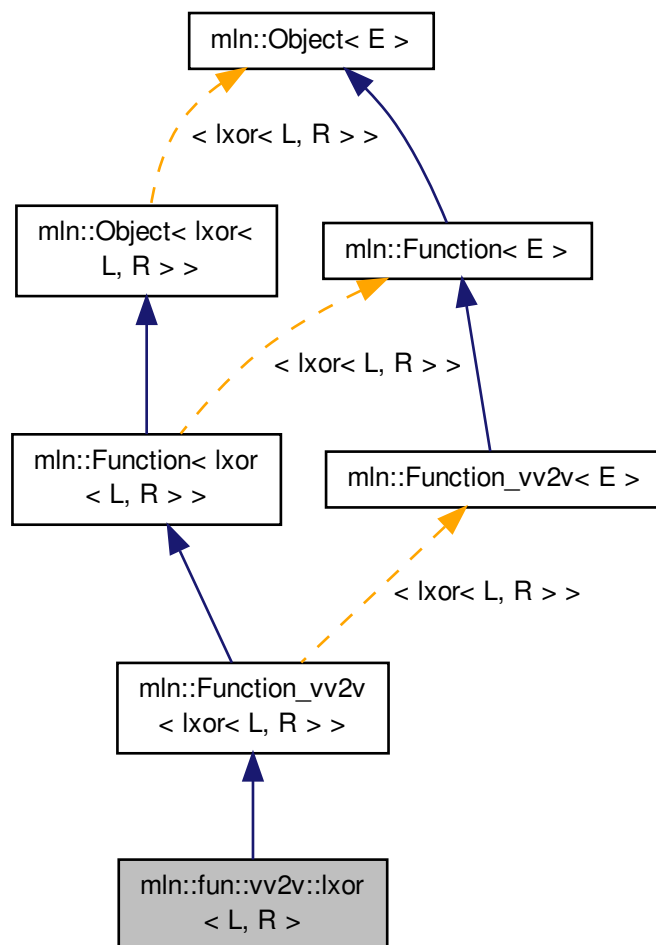
Definition at line 48 of file `fun/vv2v/lor.hh`.

10.185 `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 >:



10.185.1 Detailed Description

```
template<typename L, typename R = L>struct mln::fun::vv2v::lxor< L, R >
```

Functor computing logical-xor between two values.

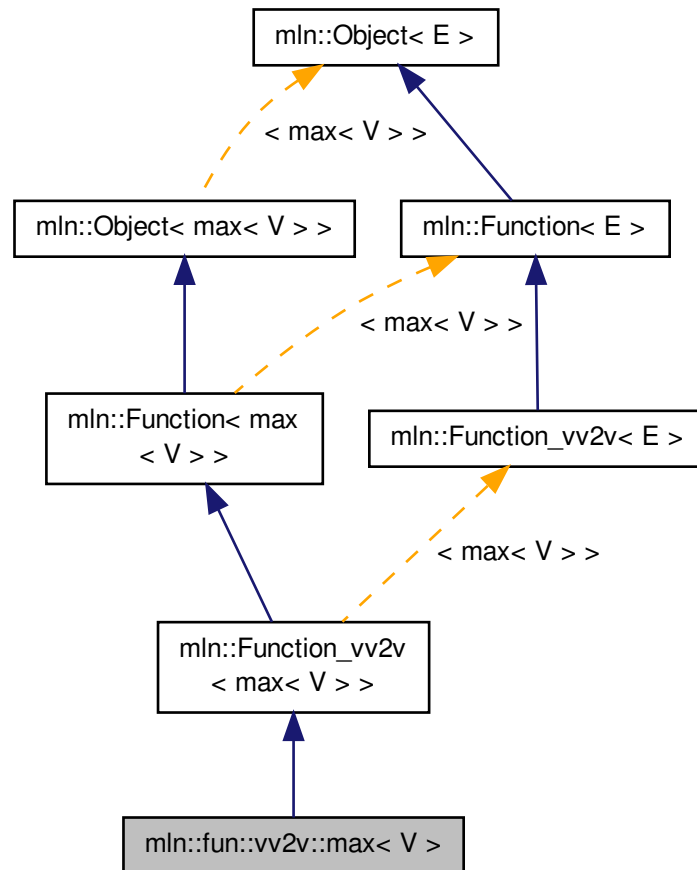
Definition at line 48 of file `lxor.hh`.

10.186 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 >`:



10.186.1 Detailed Description

```
template<typename V>struct mln::fun::vv2v::max< V >
```

A functor computing the maximum of two values.

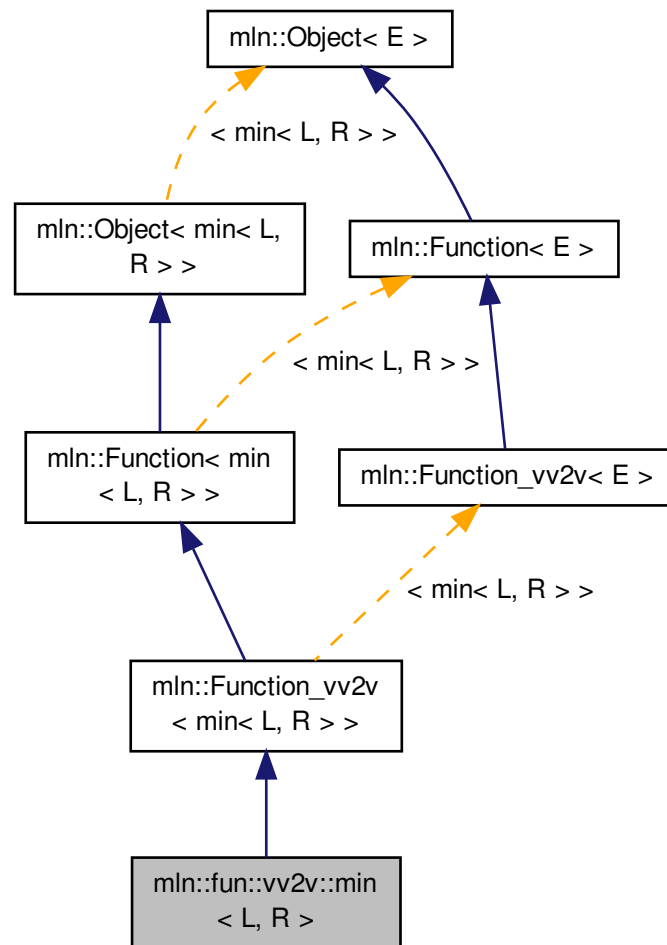
Definition at line 49 of file `fun/vv2v/max.hh`.

10.187 `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 >:



10.187.1 Detailed Description

```
template<typename L, typename R = L>struct mln::fun::vv2v::min< L, R >
```

A functor computing the minimum of two values.

Definition at line 50 of file `fun/vv2v/min.hh`.

10.188 mln::fun::vv2v::vec< V > Struct Template Reference

A functor computing the vecimum of two values.

```
#include <vec.hh>
```



```
template<typename P>struct mln::fun::x2p::closest_point< P >
```

FIXME: doxygen + concept checking.

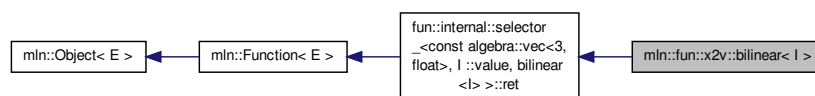
Definition at line 45 of file closest_point.hh.

10.190 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< 2, T > &v) const
Bilinear filtering on 2d images.
- template<typename T >
I::value [operator\(\)](#) (const algebra::vec< 3, T > &v) const
Bilinear filtering on 3d images. Work on slices.

10.190.1 Detailed Description

```
template<typename I>struct mln::fun::x2v::bilinear< I >
```

Represent a bilinear interolation of values from an underlying image.

Definition at line 52 of file bilinear.hh.

10.190.2 Member Function Documentation

10.190.2.1 template<typename I > template<typename T > I::value mln::fun::x2v::bilinear< I >::operator() (const algebra::vec< 2, T > & v) const

Bilinear filtering on 2d images.

Definition at line 86 of file bilinear.hh.

10.190.2.2 template<typename I > template<typename T > I::value mln::fun::x2v::bilinear< I >::operator() (const algebra::vec< 3, T > & v) const

Bilinear filtering on 3d images. Work on slices.

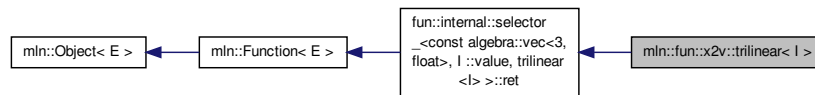
Definition at line 132 of file bilinear.hh.

10.191 mln::fun::x2v::trilinear< I > Struct Template Reference

Represent a trilinear interolation of values from an underlying image.

```
#include <trilinear.hh>
```

Inheritance diagram for mln::fun::x2v::trilinear< I >:



10.191.1 Detailed Description

```
template<typename I>struct mln::fun::x2v::trilinear< I >
```

Represent a trilinear interolation of values from an underlying image.

Definition at line 53 of file trilinear.hh.

10.192 mln::fun::x2x::composed< T2, T1 > Struct Template Reference

Represent a composition of two transformations.

```
#include <composed.hh>
```

Inherits helper_composed_< T2, T1, composed< T2, T1 >, (mlc_is(T2, Function_v2v< T2 >)::value &&mlc_is(T1, Function_v2v< T1 >)::value) >, and check_t< metal::bool_<(T2::dim==T1::dim)>, metal::is< mln_argument(T2), T1::result > >.

Public Member Functions

- [composed](#) ()
Constructor without argument.
- [composed](#) (const T2 &f, const T1 &g)
Constructor with the two transformation to be composed.

10.192.1 Detailed Description

```
template<typename T2, typename T1>struct mln::fun::x2x::composed< T2, T1 >
```

Represent a composition of two transformations.

Definition at line 144 of file composed.hh.

10.192.2 Constructor & Destructor Documentation

10.192.2.1 `template<typename T2, typename T1> mln::fun::x2x::composed< T2, T1 >::composed () [inline]`

Constructor without argument.

Definition at line 153 of file composed.hh.

10.192.2.2 `template<typename T2, typename T1> mln::fun::x2x::composed< T2, T1 >::composed (const T2 & f, const T1 & g) [inline]`

Constructor with the two transformation to be composed.

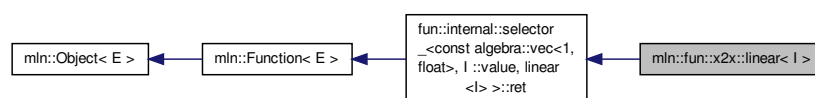
Definition at line 156 of file composed.hh.

10.193 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.

10.193.1 Detailed Description

```
template<typename I> struct mln::fun::x2x::linear< I >
```

Represent a linear interolation of values from an underlying image.

Definition at line 53 of file x2v/linear.hh.

10.193.2 Constructor & Destructor Documentation

10.193.2.1 `template<typename I > mln::fun::x2x::linear< I >::linear (const I & ima)`

Constructor with the underlying image.

Definition at line 77 of file x2v/linear.hh.

10.193.3 Member Function Documentation

10.193.3.1 `template<typename I > template<typename C > inline value mln::fun::x2x::linear< I >::operator() (const algebra::vec< 1, C > & v) const`

Return the interpolated value in the underlying image at the given 'point' v.

Definition at line 85 of file x2v/linear.hh.

10.193.4 Member Data Documentation

10.193.4.1 `template<typename I > const I& mln::fun::x2x::linear< I >::ima`

Underlying image.

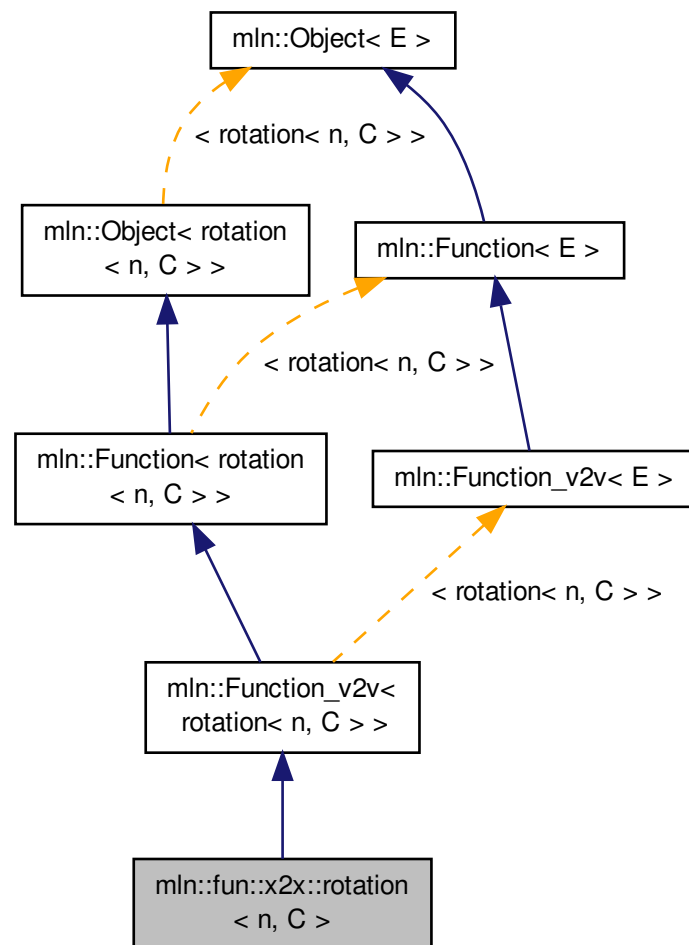
Definition at line 70 of file x2v/linear.hh.

10.194 mln::fun::x2x::rotation< n, C > Struct Template Reference

Represent a rotation function.

```
#include <rotation.hh>
```

Inheritance diagram for mln::fun::x2x::rotation< n, C >:



Public Types

- typedef C [data_t](#)
Type of the underlying data stored in vectors and matrices.
- typedef [rotation< n, C >](#) [invert](#)
Type of the inverse function.

Public Member Functions

- [invert inv](#) () const
Return the inverse function.
- `algebra::vec< n, C >` [operator\(\)](#) (const algebra::vec< n, C > &v) const
Perform the rotation of the given vector.
- [rotation](#) ()
Constructor without argument.

- [rotation](#) (C alpha, const algebra::vec< n, C > &axis)
Constructor with radian alpha and a facultative direction (rotation axis).
- [rotation](#) (const algebra::quat &q)
Constructor with quaternion.
- [rotation](#) (const algebra::h_mat< n, C > &m)
Constructor with h_mat.
- void [set_alpha](#) (C alpha)
Set a new grade alpha.
- void [set_axis](#) (const algebra::vec< n, C > &axis)
Set a new rotation axis.

10.194.1 Detailed Description

```
template<unsigned n, typename C>struct mln::fun::x2x::rotation< n, C >
```

Represent a rotation function.

Definition at line 150 of file rotation.hh.

10.194.2 Member Typedef Documentation

10.194.2.1 `template<unsigned n, typename C> typedef C mln::fun::x2x::rotation< n, C >::data_t`

Type of the underlying data stored in vectors and matrices.

Definition at line 155 of file rotation.hh.

10.194.2.2 `template<unsigned n, typename C> typedef rotation<n,C> mln::fun::x2x::rotation< n, C >::invert`

Type of the inverse function.

Definition at line 158 of file rotation.hh.

10.194.3 Constructor & Destructor Documentation

10.194.3.1 `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation () [inline]`

Constructor without argument.

Definition at line 195 of file rotation.hh.

10.194.3.2 `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation (C alpha, const algebra::vec< n, C > & axis) [inline]`

Constructor with radian alpha and a facultative direction (rotation axis).

Definition at line 201 of file rotation.hh.

10.194.3.3 `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation (const algebra::quat & q) [inline]`

Constructor with quaternion.

Definition at line 211 of file rotation.hh.

References `mln::make::h_mat()`.

10.194.3.4 `template<unsigned n, typename C > mln::fun::x2x::rotation< n, C >::rotation (const algebra::h_mat< n, C > & m) [inline]`

Constructor with h_mat.

Definition at line 241 of file rotation.hh.

10.194.4 Member Function Documentation

10.194.4.1 `template<unsigned n, typename C > rotation< n, C > mln::fun::x2x::rotation< n, C >::inv () const [inline]`

Return the inverse function.

Definition at line 268 of file rotation.hh.

10.194.4.2 `template<unsigned n, typename C > algebra::vec< n, C > mln::fun::x2x::rotation< n, C >::operator() (const algebra::vec< n, C > & v) const [inline]`

Perform the rotation of the given vector.

Definition at line 250 of file rotation.hh.

10.194.4.3 `template<unsigned n, typename C > void mln::fun::x2x::rotation< n, C >::set_alpha (C alpha) [inline]`

Set a new grade alpha.

Definition at line 277 of file rotation.hh.

10.194.4.4 `template<unsigned n, typename C > void mln::fun::x2x::rotation< n, C >::set_axis (const algebra::vec< n, C > & axis) [inline]`

Set a new rotation axis.

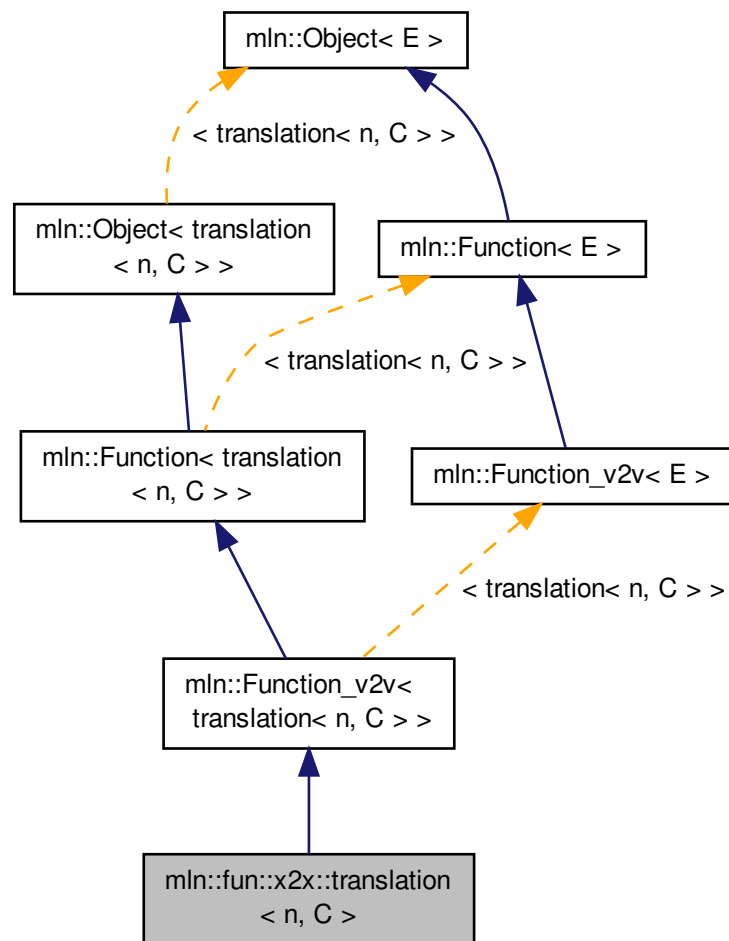
Definition at line 286 of file rotation.hh.

10.195 mln::fun::x2x::translation< n, C > Struct Template Reference

Translation function-object.

`#include <translation.hh>`

Inheritance diagram for `mIn::fun::x2x::translation< n, C >`:



Public Types

- typedef `C` `data_t`
Type of the underlying data stored in vectors and matrices.
- typedef `translation< n, C >` `invert`
Type of the inverse function.

Public Member Functions

- `invert inv () const`
Return the inverse function.
- `algebra::vec< n, C > operator() (const algebra::vec< n, C > &v) const`
Perform the translation of the given vector.
- void `set_t (const algebra::vec< n, C > &t)`
Set a net translation vector.

- `const algebra::vec< n, C > & t () const`
Return the translation vector.
- `translation ()`
Constructor without argument.
- `translation (const algebra::vec< n, C > &t)`
Constructor with the translation vector.

10.195.1 Detailed Description

`template<unsigned n, typename C>struct mln::fun::x2x::translation< n, C >`

Translation function-object.

Definition at line 52 of file x2x/translation.hh.

10.195.2 Member Typedef Documentation

10.195.2.1 `template<unsigned n, typename C> typedef C mln::fun::x2x::translation< n, C >::data_t`

Type of the underlying data stored in vectors and matrices.

Definition at line 59 of file x2x/translation.hh.

10.195.2.2 `template<unsigned n, typename C> typedef translation<n,C> mln::fun::x2x::translation< n, C >::invert`

Type of the inverse function.

Definition at line 62 of file x2x/translation.hh.

10.195.3 Constructor & Destructor Documentation

10.195.3.1 `template<unsigned n, typename C> mln::fun::x2x::translation< n, C >::translation () [inline]`

Constructor without argument.

Definition at line 93 of file x2x/translation.hh.

10.195.3.2 `template<unsigned n, typename C> mln::fun::x2x::translation< n, C >::translation (const algebra::vec< n, C > & t) [inline]`

Constructor with the translation vector.

Definition at line 99 of file x2x/translation.hh.

10.195.4 Member Function Documentation

10.195.4.1 `template<unsigned n, typename C> translation< n, C > mln::fun::x2x::translation< n, C >::inv () const [inline]`

Return the inverse function.

Definition at line 124 of file x2x/translation.hh.

10.195.4.2 `template<unsigned n, typename C > algebra::vec< n, C > mln::fun::x2x::translation< n, C >::operator() (const algebra::vec< n, C > & v) const [inline]`

Perform the translation of the given vector.

Definition at line 108 of file x2x/translation.hh.

10.195.4.3 `template<unsigned n, typename C > void mln::fun::x2x::translation< n, C >::set_t (const algebra::vec< n, C > & t) [inline]`

Set a net translation vector.

Definition at line 134 of file x2x/translation.hh.

10.195.4.4 `template<unsigned n, typename C > const algebra::vec< n, C > & mln::fun::x2x::translation< n, C >::t () const [inline]`

Return the translation vector.

Definition at line 143 of file x2x/translation.hh.

10.196 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, fun_image< F, I > >.

Public Types

- typedef F::result [lvalue](#)
Return type of read-write access.
- typedef F::result [rvalue](#)
Return type of read-only access.
- typedef [fun_image](#)< tag::value_
< typename F::result >
, tag::image_< I > > [skeleton](#)
Skeleton.
- typedef F::result [value](#)
Value associated type.

Public Member Functions

- [fun_image](#) ()
Constructor.
- [fun_image](#) (const [Function_v2v](#)< F > &f, const [Image](#)< I > &ima)
Constructor.
- [fun_image](#) (const [Image](#)< I > &ima)
Constructor.
- F::result [operator\(\)](#) (const typename I::psite &p) const
Read-only access of pixel value at point site p.
- F::result [operator\(\)](#) (const typename I::psite &p)
Mutable access is for reading only.

10.196.1 Detailed Description

template<typename F, typename I> struct mln::fun_image< F, I >

[Image](#) read through a function.

Definition at line 101 of file fun_image.hh.

10.196.2 Member Typedef Documentation

10.196.2.1 template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::lvalue

Return type of read-write access.

Definition at line 111 of file fun_image.hh.

10.196.2.2 template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::rvalue

Return type of read-only access.

Definition at line 108 of file fun_image.hh.

10.196.2.3 template<typename F, typename I> typedef fun_image< tag::value_<typename F ::result>, tag::image_<I> > mln::fun_image< F, I >::skeleton

Skeleton.

Definition at line 115 of file fun_image.hh.

10.196.2.4 template<typename F, typename I> typedef F ::result mln::fun_image< F, I >::value

[Value](#) associated type.

Definition at line 105 of file fun_image.hh.

10.196.3 Constructor & Destructor Documentation

10.196.3.1 template<typename F, typename I> fun_image< F, I >::fun_image () [inline]

Constructor.

Definition at line 177 of file fun_image.hh.

10.196.3.2 template<typename F, typename I> fun_image< F, I >::fun_image (const Function_v2v< F > & f, const Image< I > & ima) [inline]

Constructor.

Definition at line 184 of file fun_image.hh.

10.196.3.3 template<typename F, typename I> fun_image< F, I >::fun_image (const Image< I > & ima) [inline]

Constructor.

Definition at line 191 of file fun_image.hh.

10.196.4 Member Function Documentation

10.196.4.1 `template<typename F, typename I> F::result fun_image< F, I >::operator() (const typename I::psite & p)
const [inline]`

Read-only access of pixel value at point site *p*.

Definition at line 209 of file `fun_image.hh`.

10.196.4.2 `template<typename F, typename I> F::result fun_image< F, I >::operator() (const typename I::psite & p)
[inline]`

Mutable access is for reading only.

Definition at line 218 of file `fun_image.hh`.

10.197 `mln::Function< E >` Struct Template Reference

Base class for implementation of function-objects.

`#include <function.hh>`

Inherits `mln::Object< E >`.

Inherited by `mln::Function_n2v< E >`, `mln::Function_v2v< E >`, `mln::Function_vv2b< E >`, and `mln::Function_vv2v< E >`.

Protected Member Functions

- `Function ()`
An operator() has to be provided.

10.197.1 Detailed Description

`template<typename E> struct mln::Function< E >`

Base class for implementation of function-objects.

The parameter *E* is the exact type.

Definition at line 64 of file `function.hh`.

10.197.2 Constructor & Destructor Documentation

10.197.2.1 `template<typename E> Function< E >::Function () [inline], [protected]`

An operator() has to be provided.

Its signature depends on the particular function-object one considers.

Definition at line 219 of file `function.hh`.

10.198 `mln::Function< void >` Struct Template Reference

`Function` category flag type.

`#include <function.hh>`

10.198.1 Detailed Description

`template<> struct mIn::Function< void >`

[Function](#) category flag type.

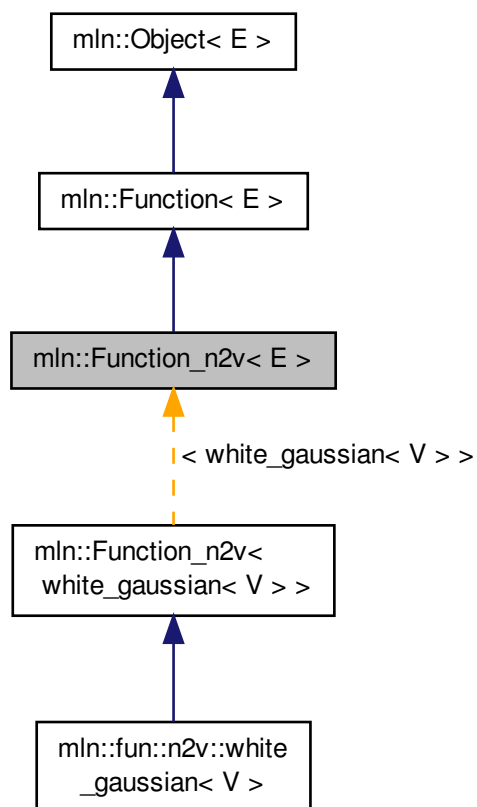
Definition at line 51 of file function.hh.

10.199 mIn::Function_n2v< E > Struct Template Reference

Base class for implementation of function-objects from Nil to value.

`#include <function.hh>`

Inheritance diagram for mIn::Function_n2v< E >:



10.199.1 Detailed Description

`template<typename E> struct mIn::Function_n2v< E >`

Base class for implementation of function-objects from Nil to value.

The parameter *E* is the exact type.

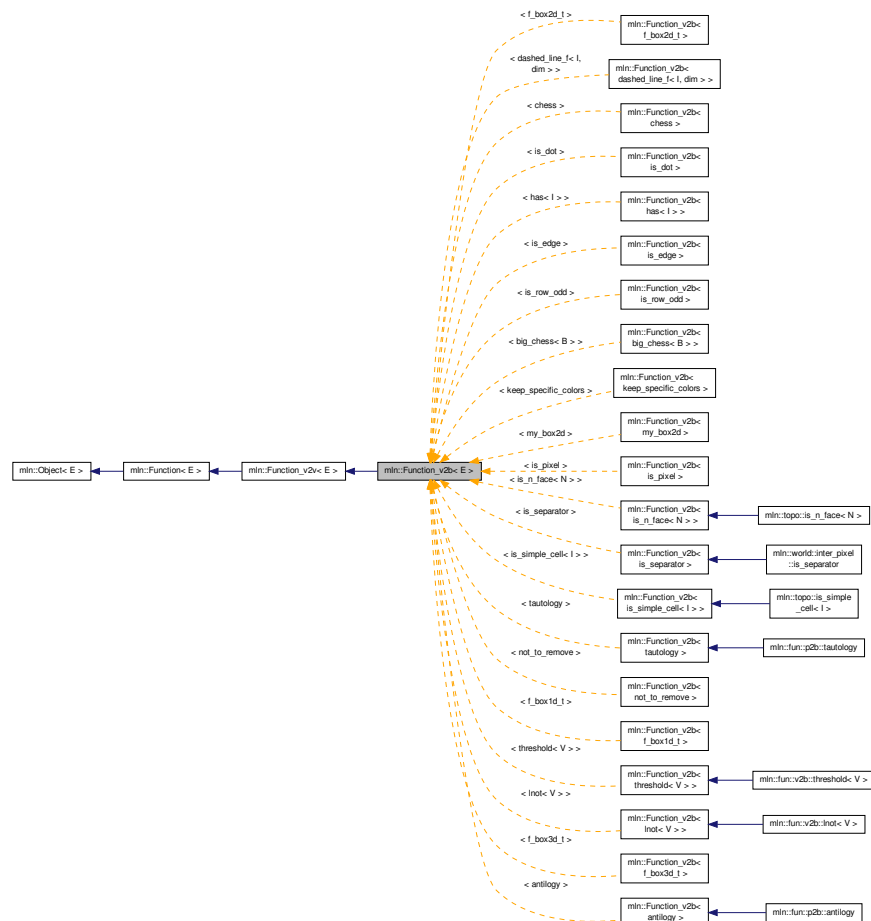
Definition at line 99 of file function.hh.

10.200 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 >:



10.200.1 Detailed Description

```
template<typename E> struct mln::Function_v2b< E >
```

Base class for implementation of function-objects from a value to a Boolean.

The parameter *E* is the exact type.

Definition at line 150 of file function.hh.

10.201 mln::Function_v2v< E > Struct Template Reference

Base class for implementation of function-objects from value to value.


```
#include <function.hh>
```

Inherits [mln::Function< E >](#).

Inherited by [mln::fun::C< R\(*\) \(A\) >](#), [mln::fun::v2v::dec< T >](#), [mln::fun::v2v::id< T >](#), [mln::fun::v2v::inc< T >](#), [mln::fun::x2v::bilinear< I >](#), [mln::fun::x2v::trilinear< I >](#), [mln::fun::x2x::internal::helper_composed_< T2, T1, E, false >](#), [mln::fun::x2x::internal::helper_composed_< T2, T1, E, true >](#), [mln::fun::x2x::linear< I >](#), [mln::fun::x2x::neighbor< I >](#), and [mln::Function_v2b< E >](#) [virtual].

10.201.1 Detailed Description

```
template<typename E> struct mln::Function_v2v< E >
```

Base class for implementation of function-objects from value to value.

The parameter *E* is the exact type.

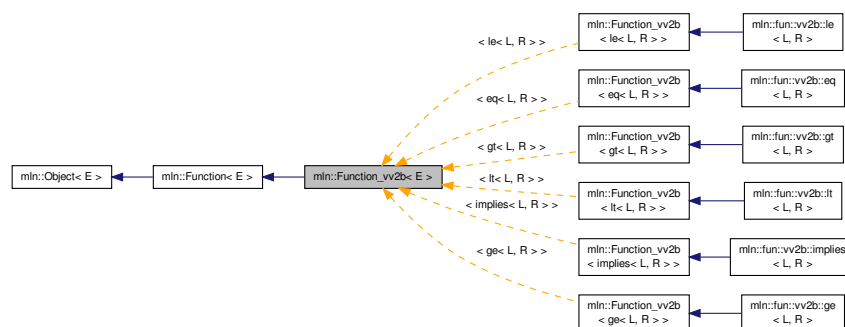
Definition at line 124 of file function.hh.

10.202 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 >](#):



10.202.1 Detailed Description

```
template<typename E> struct mln::Function_vv2b< E >
```

Base class for implementation of function-objects from a couple of values to a Boolean.

The parameter *E* is the exact type.

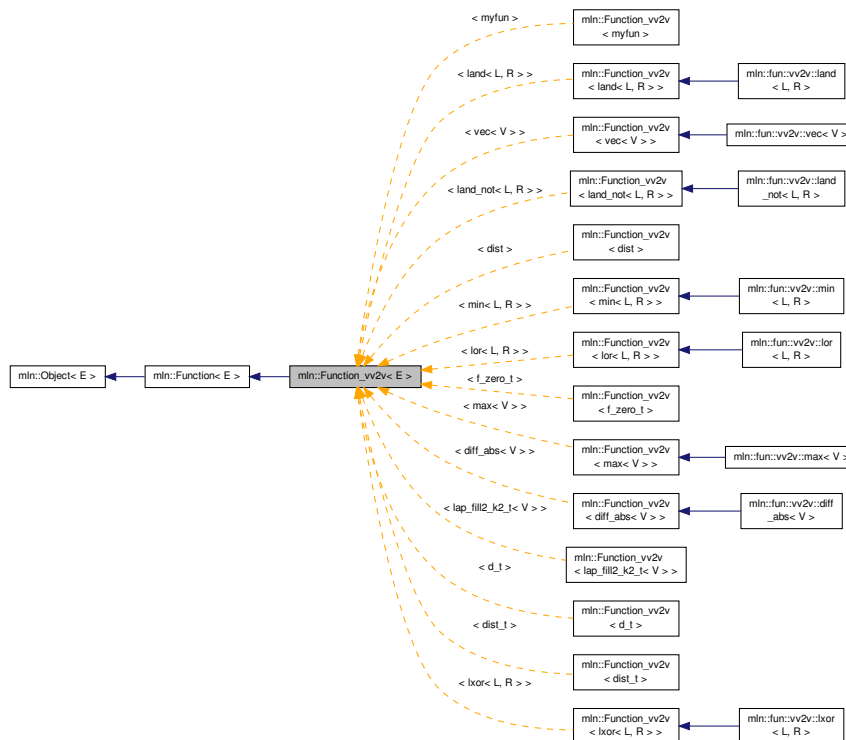
Definition at line 202 of file function.hh.

10.203 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 >`:



10.203.1 Detailed Description

```
template<typename E> struct mln::Function_vv2v< E >
```

Base class for implementation of function-objects from a couple of values to a value.

The parameter *E* is the exact type.

Definition at line 177 of file `function.hh`.

10.204 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, 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.

10.204.1 Detailed Description

```
template<typename I>class mln::fwd_pixter1d< I >
```

Forward pixel iterator on a 1-D image with border.

Definition at line 45 of file pixter1d.hh.

10.204.2 Member Typedef Documentation

10.204.2.1 `template<typename I > typedef I mln::fwd_pixter1d< I >::image`

[Image](#) type.

Definition at line 52 of file pixter1d.hh.

10.204.3 Constructor & Destructor Documentation

10.204.3.1 `template<typename I > mln::fwd_pixter1d< I >::fwd_pixter1d (I & image) [inline]`

Constructor.

Parameters

<code>in</code>	<code>image</code>	The image this pixel iterator is bound to.
-----------------	--------------------	--

Definition at line 96 of file pixter1d.hh.

10.204.4 Member Function Documentation

10.204.4.1 `void mln::Iterator< fwd_pixter1d< I > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.205 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, 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.

10.205.1 Detailed Description

template<typename I>class mln::fwd_pixter2d< I >

Forward pixel iterator on a 2-D image with border.

Definition at line 47 of file pixter2d.hh.

10.205.2 Member Typedef Documentation

10.205.2.1 template<typename I > typedef I mln::fwd_pixter2d< I >::image

[Image](#) type.

Definition at line 54 of file pixter2d.hh.

10.205.3 Constructor & Destructor Documentation

10.205.3.1 template<typename I > mln::fwd_pixter2d< I >::fwd_pixter2d (I & *image*) [inline]

Constructor.

Parameters

in	<i>image</i>	The image this pixel iterator is bound to.
--------------------	--------------	--

Definition at line 130 of file pixter2d.hh.

10.205.4 Member Function Documentation

10.205.4.1 void mln::Iterator< fwd_pixter2d< I > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.206 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, 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.

10.206.1 Detailed Description

```
template<typename I>class mln::fwd_pixter3d< I >
```

Forward pixel iterator on a 3-D image with border.

Definition at line 48 of file pixter3d.hh.

10.206.2 Member Typedef Documentation

10.206.2.1 template<typename I > typedef I mln::fwd_pixter3d< I >::image

[Image](#) type.

Definition at line 55 of file pixter3d.hh.

10.206.3 Constructor & Destructor Documentation

10.206.3.1 template<typename I > mln::fwd_pixter3d< I >::fwd_pixter3d (I & *image*) [inline]

Constructor.

Parameters

in	<i>image</i>	The image this pixel iterator is bound to.
--------------------	--------------	--

Definition at line 154 of file pixter3d.hh.

10.206.4 Member Function Documentation

10.206.4.1 void mln::Iterator< fwd_pixter3d< I > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

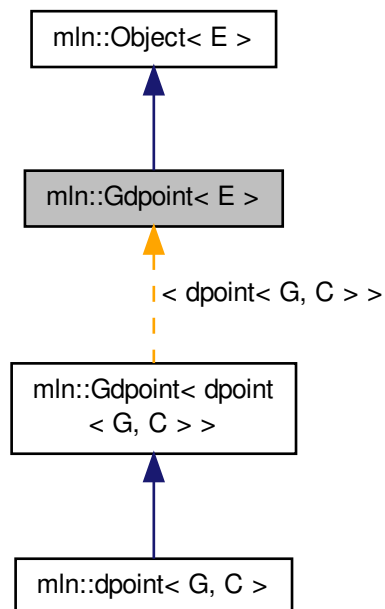
The iterator is valid.

10.207 mln::Gdpoint< E > Struct Template Reference

FIXME: Doc!

```
#include <gdpoint.hh>
```

Inheritance diagram for mln::Gdpoint< E >:

**10.207.1 Detailed Description**

```
template<typename E>struct mln::Gdpoint< E >
```

FIXME: Doc!

Definition at line 95 of file `gdpoint.hh`.

10.208 mln::Gdpoint< void > Struct Template Reference

Delta point site category flag type.

```
#include <gdpoint.hh>
```

10.208.1 Detailed Description

```
template<> struct mln::Gdpoint< void >
```

Delta point site category flag type.

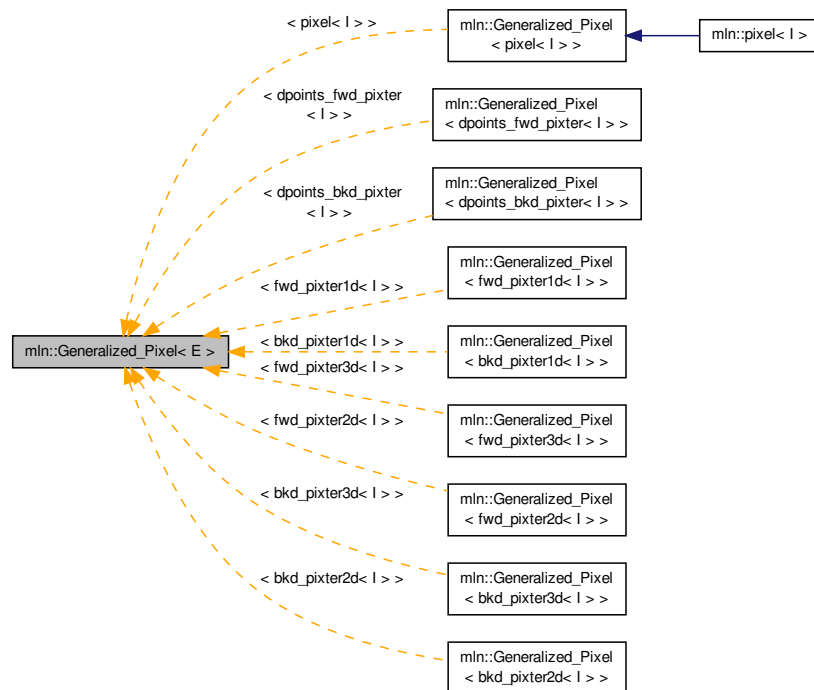
Definition at line 87 of file gdpoint.hh.

10.209 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 >:



10.209.1 Detailed Description

```
template<typename E> struct mln::Generalized_Pixel< E >
```

Base class for implementation classes that are pixels or that have the behavior of pixels.

Warning

This class does *not* derive from [mln::Object](#); it is for use as a parallel hierarchy.

See Also

[mln::doc::Generalized_Pixel](#) for a complete documentation of this class contents.

Definition at line 53 of file `generalized_pixel.hh`.

10.210 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.

10.210.1 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

<i>D</i>	The dimension of the complex.
<i>P</i>	The type of the location of a 0-face.

Locations of 0-face are usually points (hence the *P* above), but can possibly be any (default-constructible) values.

The functor returns a `std::vector` of locations: 0-faces are singletons, 1-faces are (usually) pairs, faces of higher dimensions are arrays of locations.

Note that for consistency reasons w.r.t. the return type of `operator()`, returned sites are always *arrays* of locations attached to 0-faces; hence the returned singletons (of locations) for 0-faces.

Definition at line 88 of file `geom/complex_geometry.hh`.

10.210.2 Constructor & Destructor Documentation

```
10.210.2.1 template<unsigned D, typename P > mln::geom::complex_geometry< D, P >::complex_geometry ( )  
[inline]
```

Build a complex geometry object.

Definition at line 132 of file `geom/complex_geometry.hh`.

10.210.3 Member Function Documentation

```
10.210.3.1 template<unsigned D, typename P > unsigned mln::geom::complex_geometry< D, P >::add_location (   
const P & p ) [inline]
```

Populate the set of locations.

Append a new location *p*. Return the index of the newly created location (which should semantically match the id of the corresponding 0-face in the complex).

Definition at line 140 of file `geom/complex_geometry.hh`.

10.210.3.2 `template<unsigned D, typename P > util::multi_site< P > mln::geom::complex_geometry< D, P >::operator() (const mln::topo::face< D > & f) const [inline]`

Retrieve the site associated to *f*.

Definition at line 151 of file `geom/complex_geometry.hh`.

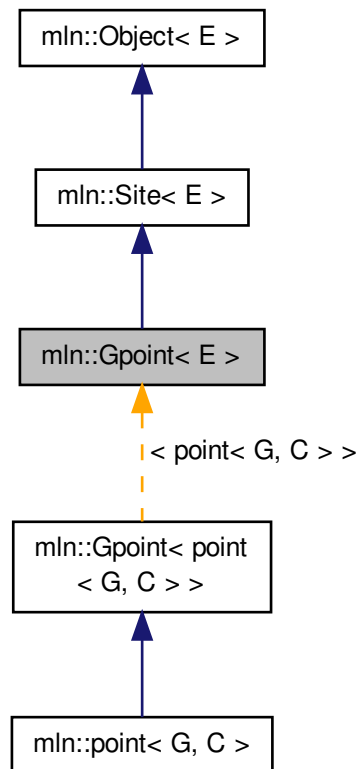
References `mln::topo::face< D >::face_id()`, and `mln::topo::face< D >::n()`.

10.211 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)`
Divise 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.

10.211.1 Detailed Description

`template<typename E>struct mln::Gpoint< E >`

Base class for implementation of point classes.

A point is an element of a space.

For instance, `mln::point2d` is the type of elements defined on the discrete square grid of the 2D plane.

Definition at line 115 of file `gpoint.hh`.

10.211.2 Friends And Related Function Documentation

10.211.2.1 `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

<code>in</code>	<code>p</code>	A grid point.
<code>in</code>	<code>dp</code>	A delta-point.

The type of `dp` has to compatible with the type of `p`.

Returns

A point (temporary object).

\see `mln::Gdpoint`

Definition at line 385 of file `gpoint.hh`.

10.211.2.2 `template<typename P , typename D > P & operator+= (Gpoint< P > & p, const Gdpoint< D > & dp)`
 [related]

Shift a point by a delta-point `dp`.

Parameters

<code>in, out</code>	<code>p</code>	The targeted point.
<code>in</code>	<code>dp</code>	A delta-point.

Returns

A reference to the point `p` once translated by `dp`.

Precondition

The type of `dp` has to be compatible with the type of `p`.

Definition at line 428 of file `gpoint.hh`.

10.211.2.3 `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

<code>in</code>	<code>lhs</code>	A first grid point.
<code>in</code>	<code>rhs</code>	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 `mln::Gdpoint`

Definition at line 374 of file `gpoint.hh`.

10.211.2.4 `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

<code>in, out</code>	<code>p</code>	The targeted point.
<code>in</code>	<code>dp</code>	A delta-point.

Returns

A reference to the point `p` once translated by `- dp`.

Precondition

The type of `dp` has to be compatible with the type of `p`.

Definition at line 436 of file `gpoint.hh`.

10.211.2.5 `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

in, out	<i>p</i>	The targeted point.
in	<i>dp</i>	A scalar.

Returns

A reference to the point `p` once divided by `s`.

10.211.2.6 `template<typename P > std::ostream & operator<< (std::ostream & ostr, const Gpoint< P > & p)`
[related]

Print a grid point `p` into the output stream `ostr`.

Parameters

in, out	<i>ostr</i>	An output stream.
in	<i>p</i>	A grid point.

Returns

The modified output stream `ostr`.

Definition at line 417 of file `gpoint.hh`.

References `mln::debug::format()`.

10.211.2.7 `template<typename L , typename R > bool operator== (const Gpoint< L > & lhs, const Gpoint< R > & rhs)`
[related]

Equality comparison between a couple of grid point `lhs` and `rhs`.

Parameters

in	<i>lhs</i>	A first grid point.
in	<i>rhs</i>	A second grid point.

Precondition

Both `lhs` and `rhs` have to be defined on the same topology; otherwise this test does not compile.

Returns

True if both grid points have the same coordinates, otherwise false.

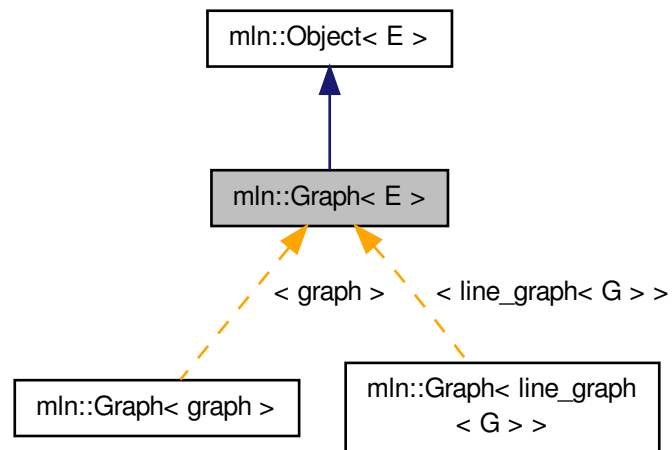
Definition at line 365 of file gpoint.hh.

10.212 mln::Graph< E > Struct Template Reference

Base class for implementation of graph classes.

```
#include <graph.hh>
```

Inheritance diagram for mln::Graph< E >:

**10.212.1 Detailed Description**

```
template<typename E>struct mln::Graph< E >
```

Base class for implementation of graph classes.

See Also

`mln::doc::Graph` for a complete documentation of this class contents.

Definition at line 57 of file `mln/core/concept/graph.hh`.

10.213 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.

10.213.1 Detailed Description

Compute the cardinality of every component in a graph.

Returns

An array with the cardinality for each component. Components are labeled from 0.

Definition at line 61 of file graph/attribute/card.hh.

10.213.2 Member Typedef Documentation

10.213.2.1 typedef [util::array](#)< unsigned > [mln::graph::attribute::card_t::result](#)

Type of the computed value.

Definition at line 64 of file graph/attribute/card.hh.

10.214 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.

10.214.1 Detailed Description

Compute the representative vertex of every component in a graph.

Returns

An array with the representative for each component. Components are labeled from 0.

Definition at line 63 of file representative.hh.

10.214.2 Member Typedef Documentation

10.214.2.1 typedef [util::array](#)< unsigned > [mln::graph::attribute::representative_t::result](#)

Type of the computed value.

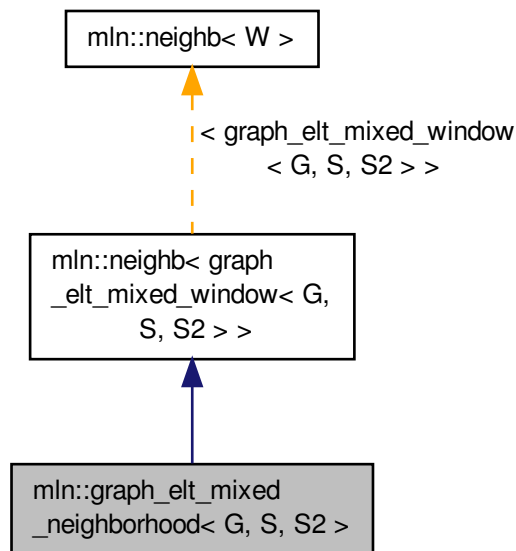
Definition at line 66 of file representative.hh.

10.215 mln::graph_elt_mixed_neighborhood< G, S, S2 > Struct Template Reference

Elementary neighborhood on graph class.

```
#include <graph_elt_mixed_neighborhood.hh>
```

Inheritance diagram for mln::graph_elt_mixed_neighborhood< G, S, S2 >:



Public Types

- typedef `neighb_bkd_niter`
`< graph_elt_mixed_window< G, S, S2 > > bkd_niter`
Backward site iterator associated type.
- typedef `neighb_fwd_niter`
`< graph_elt_mixed_window< G, S, S2 > > fwd_niter`
Forward site iterator associated type.
- typedef `fwd_niter niter`
Site iterator associated type.

10.215.1 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

<code>G</code>	is a graph type.
<code>S</code>	is a site set type.
<code>S2</code>	is the site set type of the neighbors.

Definition at line 48 of file graph_elt_mixed_neighborhood.hh.

10.215.2 Member Typedef Documentation

10.215.2.1 `typedef neighb_bkd_niter<graph_elt_mixed_window< G, S, S2 > > mln::neighb<graph_elt_mixed_window< G, S, S2 > >::bkd_niter` [inherited]

Backward site iterator associated type.

Definition at line 87 of file mln/core/neighb.hh.

10.215.2.2 `typedef neighb_fwd_niter<graph_elt_mixed_window< G, S, S2 > > mln::neighb<graph_elt_mixed_window< G, S, S2 > >::fwd_niter` [inherited]

Forward site iterator associated type.

Definition at line 84 of file mln/core/neighb.hh.

10.215.2.3 `typedef fwd_niter mln::neighb< graph_elt_mixed_window< G, S, S2 > >::niter` [inherited]

Site iterator associated type.

Definition at line 90 of file mln/core/neighb.hh.

10.216 mln::graph_elt_mixed_window< G, S, S2 > Class Template Reference

Elementary window on graph class.

```
#include <graph_elt_mixed_window.hh>
```



```

graph BT
    A["mIn::graph_elt_mixed_window< G, S, S2 >"]
    B["mIn::graph_window_base< S2::fun_t::result, graph_elt_mixed_window< G, S, S2 > >"]
    C["mIn::Window< graph_elt_mixed_window< G, S, S2 > >"]
    D["mIn::graph_window_base< P, E >"]
    E["mIn::Object< graph_elt_mixed_window< G, S, S2 > >"]
    F["mIn::Object< E >"]
    G["mIn::Window< E >"]

    A --> B
    B --> C
    B --> D
    C --> E
    E --> F
    D --> G
    G --> F

    B -.-> C
    B -.-> D
    C -.-> E
    D -.-> G
  
```

- typedef super_::target **target**
Associated types.
- typedef target::psite **psite**
The type of psite corresponding to the window.
- typedef S::psite **center_t**
Type of the window center element.
- typedef target::graph_element **graph_element**
Type of the graph element pointed by this iterator.
- typedef **graph_window_piter**
< **target**, **self_**, nbh_fwd_iter_ > **fwd_qiter**
***Site Iterator** type to browse the psites of the window w.r.t.*

- typedef [graph_window_piter](#)
`< target, self_, nbh_bkd_iter_ > bkd_qiter`
[Site_iterator](#) type to browse the psites of the window w.r.t.
- typedef [fwd_qiter](#) qiter
The default qiter type.
- typedef S2::fun_t::result [site](#)
Associated types.

Public Member Functions

- bool [is_valid](#) () const
Return true by default.
- bool [is_empty](#) () const
Interface of the concept Window.
- bool [is_centered](#) () const
Is the window centered?
- bool [is_symmetric](#) () const
Is the window symmetric?
- unsigned [delta](#) () const
Return the maximum coordinate gap between the window center and a window point.
- [self_ & sym](#) ()
Apply a central symmetry to the target window.

10.216.1 Detailed Description

`template<typename G, typename S, typename S2>class mln::graph_elt_mixed_window< G, S, S2 >`

Elementary window on graph class.

G is the graph type. S is an image site set from where the center is extracted. S2 is an image site set from where the neighbors are extracted.

Definition at line 110 of file `graph_elt_mixed_window.hh`.

10.216.2 Member Typedef Documentation

10.216.2.1 `template<typename G , typename S , typename S2 > typedef graph_window_piter<target,self_,nbh_bkd_iter_> mln::graph_elt_mixed_window< G, S, S2 >::bkd_qiter`

[Site_iterator](#) type to browse the psites of the window w.r.t.

the reverse ordering of vertices.

Definition at line 140 of file `graph_elt_mixed_window.hh`.

10.216.2.2 `template<typename G , typename S , typename S2 > typedef S ::psite mln::graph_elt_mixed_window< G, S, S2 >::center_t`

Type of the window center element.

Definition at line 129 of file `graph_elt_mixed_window.hh`.

10.216.2.3 `template<typename G , typename S , typename S2 > typedef graph_window_piter<target,self_,nbh_fwd_iter-> mln::graph_elt_mixed_window< G, S, S2 >::fwd_qiter`

[Site_iterator](#) type to browse the psites of the window w.r.t.

the ordering of vertices.

Definition at line 136 of file graph_elt_mixed_window.hh.

10.216.2.4 `template<typename G , typename S , typename S2 > typedef target ::graph_element mln::graph_elt_mixed_window< G, S, S2 >::graph_element`

Type of the graph element pointed by this iterator.

Definition at line 132 of file graph_elt_mixed_window.hh.

10.216.2.5 `template<typename G , typename S , typename S2 > typedef target ::psite mln::graph_elt_mixed_window< G, S, S2 >::psite`

The type of psite corresponding to the window.

Definition at line 126 of file graph_elt_mixed_window.hh.

10.216.2.6 `template<typename G , typename S , typename S2 > typedef fwd_qiter mln::graph_elt_mixed_window< G, S, S2 >::qiter`

The default qiter type.

Definition at line 143 of file graph_elt_mixed_window.hh.

10.216.2.7 `typedef S2::fun_t::result mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::site [inherited]`

Associated types.

The type of site corresponding to the window.

Definition at line 48 of file graph_window_base.hh.

10.216.2.8 `template<typename G , typename S , typename S2 > typedef super_::target mln::graph_elt_mixed_window< G, S, S2 >::target`

Associated types.

Definition at line 124 of file graph_elt_mixed_window.hh.

10.216.3 Member Function Documentation

10.216.3.1 `unsigned mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::delta ()const [inherited]`

Return the maximum coordinate gap between the window center and a window point.

10.216.3.2 `bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_centered ()const [inherited]`

Is the window centered?

```
10.216.3.3  bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_empty (
            ) const  [inherited]
```

Interface of the concept Window.

Is the window is empty?

```
10.216.3.4  bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 >
            >::is_symmetric ( ) const  [inherited]
```

Is the window symmetric?

```
10.216.3.5  bool mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::is_valid (
            ) const  [inherited]
```

Return true by default.

```
10.216.3.6  self_& mln::graph_window_base< S2::fun_t::result , graph_elt_mixed_window< G, S, S2 > >::sym ( )
            [inherited]
```

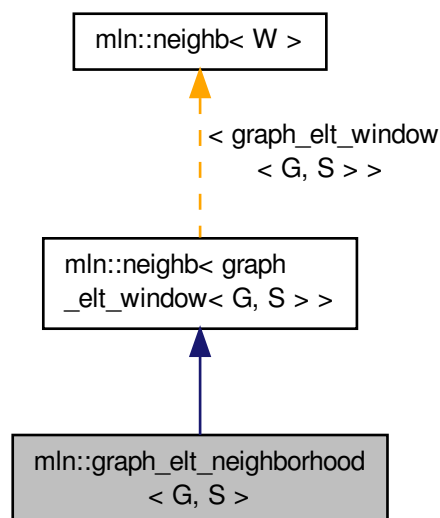
Apply a central symmetry to the target window.

10.217 mln::graph_elt_neighborhood< G, S > Struct Template Reference

Elementary neighborhood on graph class.

```
#include <graph_elt_neighborhood.hh>
```

Inheritance diagram for mln::graph_elt_neighborhood< G, S >:



Public Types

- typedef neighb_bkd_niter
 < [graph_elt_window](#)< G, S > > [bkd_niter](#)
 Backward site iterator associated type.
- typedef neighb_fwd_niter
 < [graph_elt_window](#)< G, S > > [fwd_niter](#)
 Forward site iterator associated type.
- typedef [fwd_niter](#) niter
 Site iterator associated type.

10.217.1 Detailed Description

template<typename G, typename S>struct mln::graph_elt_neighborhood< G, S >

Elementary neighborhood on graph class.

Template Parameters

G	is a graph type.
S	is a site set type.

Definition at line 47 of file graph_elt_neighborhood.hh.

10.217.2 Member Typedef Documentation

10.217.2.1 typedef neighb_bkd_niter<[graph_elt_window](#)< G, S > > mln::neighb< [graph_elt_window](#)< G, S > >::bkd_niter [\[inherited\]](#)

Backward site iterator associated type.

Definition at line 87 of file mln/core/neighb.hh.

10.217.2.2 typedef neighb_fwd_niter<[graph_elt_window](#)< G, S > > mln::neighb< [graph_elt_window](#)< G, S > >::fwd_niter [\[inherited\]](#)

Forward site iterator associated type.

Definition at line 84 of file mln/core/neighb.hh.

10.217.2.3 typedef fwd_niter mln::neighb< [graph_elt_window](#)< G, S > >::niter [\[inherited\]](#)

Site iterator associated type.

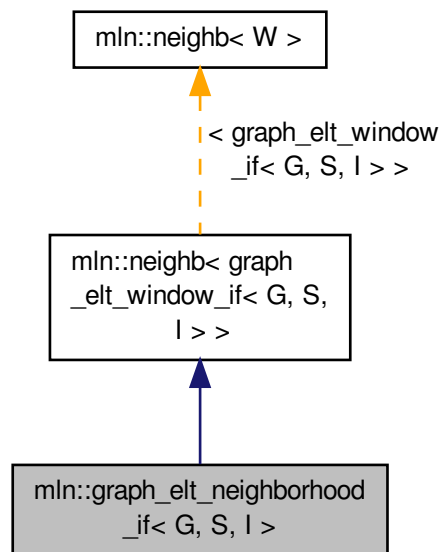
Definition at line 90 of file mln/core/neighb.hh.

10.218 mln::graph_elt_neighborhood_if< G, S, I > Struct Template Reference

Elementary neighborhood_if on graph class.

```
#include <graph_elt_neighborhood_if.hh>
```

Inheritance diagram for `mln::graph_elt_neighborhood_if< G, S, I >`:



Public Types

- typedef `neighb_bkd_niter`
`< graph_elt_window_if< G, S, I > > bkd_niter`
Backward site iterator associated type.
- typedef `neighb_fwd_niter`
`< graph_elt_window_if< G, S, I > > fwd_niter`
Forward site iterator associated type.
- typedef `fwd_niter niter`
Site iterator associated type.

Public Member Functions

- `graph_elt_neighborhood_if ()`
Constructors @ { Construct an invalid neighborhood.
- `graph_elt_neighborhood_if (const Image< I > &mask)`
- `const I &mask () const`
@ }

10.218.1 Detailed Description

```
template<typename G, typename S, typename I>struct mln::graph_elt_neighborhood_if< G, S, I >
```

Elementary neighborhood_if on graph class.

Definition at line 43 of file `graph_elt_neighborhood_if.hh`.

10.218.2 Member Typedef Documentation

10.218.2.1 `typedef neighb_bkd_niter<graph_elt_window_if< G, S, I > > mln::neighb< graph_elt_window_if< G, S, I > >::bkd_niter` `[inherited]`

Backward site iterator associated type.

Definition at line 87 of file mln/core/neighb.hh.

10.218.2.2 `typedef neighb_fwd_niter<graph_elt_window_if< G, S, I > > mln::neighb< graph_elt_window_if< G, S, I > >::fwd_niter` `[inherited]`

Forward site iterator associated type.

Definition at line 84 of file mln/core/neighb.hh.

10.218.2.3 `typedef fwd_niter mln::neighb< graph_elt_window_if< G, S, I > >::niter` `[inherited]`

Site iterator associated type.

Definition at line 90 of file mln/core/neighb.hh.

10.218.3 Constructor & Destructor Documentation

10.218.3.1 `template<typename G , typename S , typename I > mln::graph_elt_neighborhood_if< G, S, I >::graph_elt_neighborhood_if()` `[inline]`

Constructors @ { Construct an invalid neighborhood.

Definition at line 67 of file graph_elt_neighborhood_if.hh.

10.218.3.2 `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

<code>in</code>	<code>mask</code>	A graph image of Boolean.
-----------------	-------------------	---------------------------

Definition at line 74 of file graph_elt_neighborhood_if.hh.

10.218.4 Member Function Documentation

10.218.4.1 `template<typename G , typename S , typename I > const I & mln::graph_elt_neighborhood_if< G, S, I >::mask() const` `[inline]`

@ {

Return the graph image used as mask.

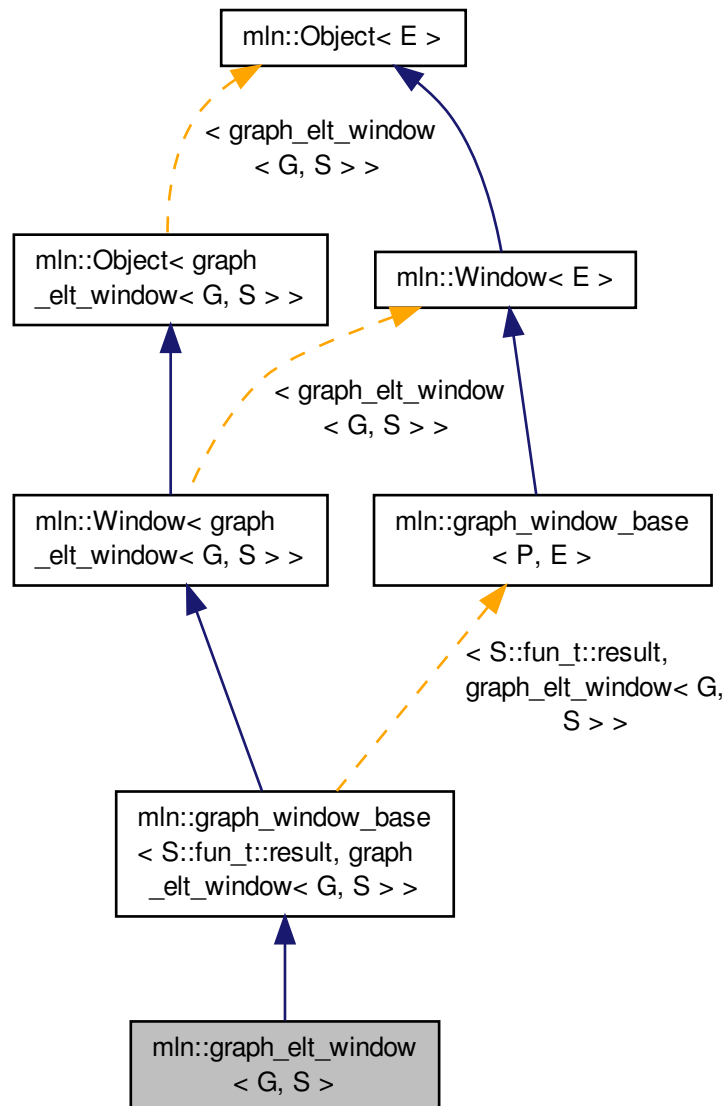
Definition at line 83 of file graph_elt_neighborhood_if.hh.

10.219 mln::graph_elt_window< G, S > Class Template Reference

Elementary window on graph class.

`#include <graph_elt_window.hh>`

Inheritance diagram for `mln::graph_elt_window< G, S >`:



Public Types

- typedef `S` [target](#)
Associated types.
- typedef `S::psite` [psite](#)
The type of psite corresponding to the window.
- typedef `S::psite` [center_t](#)
Type of the window center element.
- typedef `S::graph_element` [graph_element](#)
Type of the graph element pointed by this iterator.

- typedef [graph_window_piter](#)< S, [self_](#), [nbh_fwd_iter_](#) > [fwd_qiter](#)
[Site_iterator](#) type to browse the psites of the window w.r.t.
- typedef [graph_window_piter](#)< S, [self_](#), [nbh_bkd_iter_](#) > [bkd_qiter](#)
[Site_iterator](#) type to browse the psites of the window w.r.t.
- typedef [fwd_qiter](#) [qiter](#)
The default qiter type.
- typedef S::fun_t::result [site](#)
Associated types.

Public Member Functions

- bool [is_valid](#) () const
Return true by default.
- bool [is_empty](#) () const
Interface of the concept Window.
- bool [is_centered](#) () const
Is the window centered?
- bool [is_symmetric](#) () const
Is the window symmetric?
- unsigned [delta](#) () const
Return the maximum coordinate gap between the window center and a window point.
- [self_](#) & [sym](#) ()
Apply a central symmetry to the target window.

10.219.1 Detailed Description

template<typename G, typename S>class mln::graph_elt_window< G, S >

Elementary window on graph class.

G is the graph type. S is an image site set from where the center is extracted. S2 is an image site set from where the neighbors are extracted.

Definition at line 111 of file graph_elt_window.hh.

10.219.2 Member Typedef Documentation

10.219.2.1 template<typename G , typename S > typedef [graph_window_piter](#)<S,[self_](#),[nbh_bkd_iter_](#)>
mln::graph_elt_window< G, S >::bkd_qiter

[Site_iterator](#) type to browse the psites of the window w.r.t.

the reverse ordering of vertices.

Definition at line 142 of file graph_elt_window.hh.

10.219.2.2 template<typename G , typename S > typedef S ::psite mln::graph_elt_window< G, S >::center_t

Type of the window center element.

Definition at line 131 of file graph_elt_window.hh.

10.219.2.3 `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.

Definition at line 138 of file `graph_elt_window.hh`.

10.219.2.4 `template<typename G , typename S > typedef S ::graph_element mln::graph_elt_window< G, S
>::graph_element`

Type of the graph element pointed by this iterator.

Definition at line 134 of file `graph_elt_window.hh`.

10.219.2.5 `template<typename G , typename S > typedef S ::psite mln::graph_elt_window< G, S >::psite`

The type of psite corresponding to the window.

Definition at line 128 of file `graph_elt_window.hh`.

10.219.2.6 `template<typename G , typename S > typedef fwd_qiter mln::graph_elt_window< G, S >::qiter`

The default qiter type.

Definition at line 145 of file `graph_elt_window.hh`.

10.219.2.7 `typedef S::fun_t::result mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::site
[inherited]`

Associated types.

The type of site corresponding to the window.

Definition at line 48 of file `graph_window_base.hh`.

10.219.2.8 `template<typename G , typename S > typedef S mln::graph_elt_window< G, S >::target`

Associated types.

Definition at line 125 of file `graph_elt_window.hh`.

10.219.3 Member Function Documentation

10.219.3.1 `unsigned mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::delta () const
[inherited]`

Return the maximum coordinate gap between the window center and a window point.

10.219.3.2 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::is_centered () const
[inherited]`

Is the window centered?

10.219.3.3 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::is_empty () const`
[inherited]

Interface of the concept Window.

Is the window is empty?

10.219.3.4 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::is_symmetric () const`
[inherited]

Is the window symmetric?

10.219.3.5 `bool mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::is_valid () const`
[inherited]

Return true by default.

10.219.3.6 `self_& mln::graph_window_base< S::fun_t::result , graph_elt_window< G, S > >::sym ()`
[inherited]

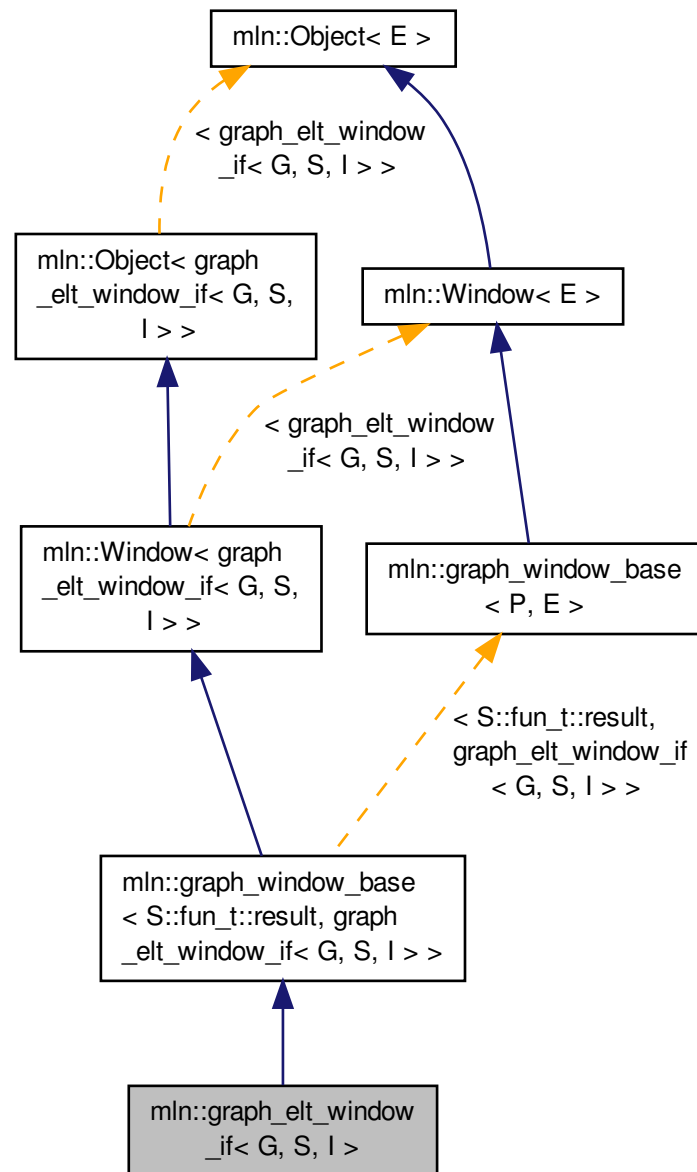
Apply a central symmetry to the target window.

10.220 mln::graph_elt_window_if< G, S, I > Class Template Reference

Custom window on graph class.

```
#include <graph_elt_window_if.hh>
```

Inheritance diagram for mln::graph_elt_window_if< G, S, I >:



Public Types

- typedef I [mask_t](#)
The type of the image used as mask.
- typedef S [target](#)
@}
- typedef target::psite [psite](#)
The type of psite corresponding to the window.

- typedef [graph_window_if_piter](#)
 < [target](#), [self_](#), [nbh_fwd_iter_](#) > [fwd_qiter](#)
Site_iterator type to browse the psites of the window w.r.t.
- typedef [graph_window_if_piter](#)
 < [target](#), [self_](#), [nbh_bkd_iter_](#) > [bkd_qiter](#)
Site_iterator type to browse the psites of the window w.r.t.
- typedef [fwd_qiter](#) [qiter](#)
The default qiter type.
- typedef S::fun_t::result [site](#)
Associated types.

Public Member Functions

- void [change_mask](#) (const [Image](#)< I > &[mask](#))
Change mask image.
- [graph_elt_window_if](#) ()
Constructor.
- [graph_elt_window_if](#) (const [Image](#)< I > &[mask](#))
- bool [is_valid](#) () const
Return true by default.
- const I & [mask](#) () const
Return the graph image used as mask.
- bool [is_empty](#) () const
Interface of the concept Window.
- bool [is_centered](#) () const
Is the window centered?
- bool [is_symmetric](#) () const
Is the window symmetric?
- unsigned [delta](#) () const
Return the maximum coordinate gap between the window center and a window point.
- [self_](#) & [sym](#) ()
Apply a central symmetry to the target window.

10.220.1 Detailed Description

template<typename G, typename S, typename I>class mln::graph_elt_window_if< G, S, I >

Custom window on graph class.

It is defined thanks to a mask.

G is the graph type. S is the image site set. I is the graph image the type used as mask.

Definition at line 106 of file [graph_elt_window_if.hh](#).

10.220.2 Member Typedef Documentation

10.220.2.1 `template<typename G , typename S , typename I > typedef graph_window_if_piter<target,self_nbh_bkd_iter_> mln::graph_elt_window_if< G, S, I >::bkd_qiter`

[Site_iterator](#) type to browse the psites of the window w.r.t.

the reverse ordering of vertices.

Definition at line 148 of file `graph_elt_window_if.hh`.

10.220.2.2 `template<typename G , typename S , typename I > typedef graph_window_if_piter<target,self_nbh_fwd_iter_> mln::graph_elt_window_if< G, S, I >::fwd_qiter`

[Site_iterator](#) type to browse the psites of the window w.r.t.

the ordering of vertices.

Definition at line 144 of file `graph_elt_window_if.hh`.

10.220.2.3 `template<typename G , typename S , typename I > typedef I mln::graph_elt_window_if< G, S, I >::mask_t`

The type of the image used as mask.

Definition at line 120 of file `graph_elt_window_if.hh`.

10.220.2.4 `template<typename G , typename S , typename I > typedef target::psite mln::graph_elt_window_if< G, S, I >::psite`

The type of psite corresponding to the window.

Definition at line 140 of file `graph_elt_window_if.hh`.

10.220.2.5 `template<typename G , typename S , typename I > typedef fwd_qiter mln::graph_elt_window_if< G, S, I >::qiter`

The default qiter type.

Definition at line 151 of file `graph_elt_window_if.hh`.

10.220.2.6 `typedef S::fun_t::result mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::site [inherited]`

Associated types.

The type of site corresponding to the window.

Definition at line 48 of file `graph_window_base.hh`.

10.220.2.7 `template<typename G , typename S , typename I > typedef S mln::graph_elt_window_if< G, S, I >::target`

@}

Associated types. The image domain on which this window iterates on.

Definition at line 137 of file `graph_elt_window_if.hh`.

10.220.3 Constructor & Destructor Documentation

10.220.3.1 `template<typename G , typename S , typename I > graph_elt_window_if< G, S, I >::graph_elt_window_if () [inline]`

Constructor.

@{ Default. Construct an invalid window.

Definition at line 175 of file `graph_elt_window_if.hh`.

10.220.3.2 `template<typename G , typename S , typename I > graph_elt_window_if< G, S, I >::graph_elt_window_if (const Image<I > & mask) [inline]`

Parameters

<code>in</code>	<code>mask</code>	A graph image of bool.
-----------------	-------------------	------------------------

See Also

[vertex_image](#), [edge_image](#).

Definition at line 182 of file `graph_elt_window_if.hh`.

10.220.4 Member Function Documentation

10.220.4.1 `template<typename G , typename S , typename I > void graph_elt_window_if< G, S, I >::change_mask (const Image<I > & mask) [inline]`

Change mask image.

Definition at line 200 of file `graph_elt_window_if.hh`.

10.220.4.2 `unsigned mln::graph_window_base< S::fun.t::result , graph_elt_window_if< G, S, I > >::delta () const [inherited]`

Return the maximum coordinate gap between the window center and a window point.

10.220.4.3 `bool mln::graph_window_base< S::fun.t::result , graph_elt_window_if< G, S, I > >::is_centered () const [inherited]`

Is the window centered?

10.220.4.4 `bool mln::graph_window_base< S::fun.t::result , graph_elt_window_if< G, S, I > >::is_empty () const [inherited]`

Interface of the concept Window.

Is the window is empty?

10.220.4.5 `bool mln::graph_window_base< S::fun.t::result , graph_elt_window_if< G, S, I > >::is_symmetric () const [inherited]`

Is the window symmetric?

10.220.4.6 `template<typename G , typename S , typename I > bool graph_elt_window_if< G, S, I >::is_valid () const`
`[inline]`

Return true by default.

Definition at line 209 of file `graph_elt_window_if.hh`.

10.220.4.7 `template<typename G , typename S , typename I > const I & graph_elt_window_if< G, S, I >::mask () const`
`[inline]`

Return the graph image used as mask.

Definition at line 191 of file `graph_elt_window_if.hh`.

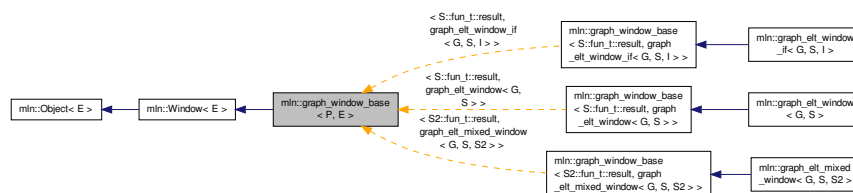
10.220.4.8 `self_& mln::graph_window_base< S::fun_t::result , graph_elt_window_if< G, S, I > >::sym ()`
`[inherited]`

Apply a central symmetry to the target window.

10.221 mln::graph_window_base< P, E > Class Template Reference

`#include <graph_window_base.hh>`

Inheritance diagram for `mln::graph_window_base< P, E >`:



Public Types

- typedef `P` [site](#)
Associated types.

Public Member Functions

- bool [is_valid](#) () const
Return true by default.
- bool [is_empty](#) () const
Interface of the concept [Window](#).
- bool [is_centered](#) () const
Is the window centered?
- bool [is_symmetric](#) () const
Is the window symmetric?
- unsigned [delta](#) () const
Return the maximum coordinate gap between the window center and a window point.

- [self_ & sym \(\)](#)

Apply a central symmetry to the target window.

10.221.1 Detailed Description

```
template<typename P, typename E>class mln::graph_window_base< P, E >
```

Template Parameters

<i>P</i>	Site type.
----------	----------------------------

Definition at line 40 of file graph_window_base.hh.

10.221.2 Member Typedef Documentation

10.221.2.1 `template<typename P, typename E> typedef P mln::graph_window_base< P, E >::site`

Associated types.

The type of site corresponding to the window.

Definition at line 48 of file graph_window_base.hh.

10.221.3 Member Function Documentation

10.221.3.1 `template<typename P , typename E > unsigned mln::graph_window_base< P, E >::delta () const`
[inline]

Return the maximum coordinate gap between the window center and a window point.

Definition at line 128 of file graph_window_base.hh.

10.221.3.2 `template<typename P , typename E > bool mln::graph_window_base< P, E >::is_centered () const`
[inline]

Is the window centered?

Definition at line 112 of file graph_window_base.hh.

10.221.3.3 `template<typename P , typename E > bool mln::graph_window_base< P, E >::is_empty () const`
[inline]

Interface of the concept [Window](#).

Is the window is empty?

Definition at line 104 of file graph_window_base.hh.

10.221.3.4 `template<typename P , typename E > bool mln::graph_window_base< P, E >::is_symmetric () const`
[inline]

Is the window symmetric?

Definition at line 120 of file graph_window_base.hh.

10.221.3.5 `template<typename P, typename E> bool mln::graph_window_base< P, E >::is_valid () const`
`[inline]`

Return true by default.

Definition at line 153 of file `graph_window_base.hh`.

10.221.3.6 `template<typename P, typename E> graph_window_base< P, E > & mln::graph_window_base< P, E >::sym () [inline]`

Apply a central symmetry to the target window.

Definition at line 137 of file `graph_window_base.hh`.

10.222 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, graph_window_if_piter< S, W, I > >`, and `mln::internal::is_masked_impl_selector< S, W::mask_t::domain_t, graph_window_if_piter< S, W, I > >`.

Public Types

- `typedef S::fun_t::result P`

Associated types.

Public Member Functions

- `void next ()`

Go to the next element.

- `graph_window_if_piter ()`

Construction.

- `const S::graph_element & element () const`

Return the graph element pointed by this iterator.

- `unsigned id () const`

Return the graph element id.

10.222.1 Detailed Description

`template<typename S, typename W, typename I> class mln::graph_window_if_piter< S, W, I >`

Forward iterator on line graph window.

Definition at line 47 of file `graph_window_if_piter.hh`.

10.222.2 Member Typedef Documentation

10.222.2.1 `template<typename S, typename W, typename I> typedef S::fun_t::result mln::graph_window_if_piter< S, W, I>::P`

Associated types.

Definition at line 60 of file graph_window_if_piter.hh.

10.222.3 Constructor & Destructor Documentation

10.222.3.1 `template<typename S, typename W, typename I> mln::graph_window_if_piter< S, W, I>::graph_window_if_piter() [inline]`

Construction.

Definition at line 122 of file graph_window_if_piter.hh.

10.222.4 Member Function Documentation

10.222.4.1 `template<typename S, typename W, typename I> const S::graph_element & mln::graph_window_if_piter< S, W, I>::element() const [inline]`

Return the graph element pointed by this iterator.

Definition at line 213 of file graph_window_if_piter.hh.

10.222.4.2 `template<typename S, typename W, typename I> unsigned mln::graph_window_if_piter< S, W, I>::id() const [inline]`

Return the graph element id.

FIXME: we do not want to have this member since there is an automatic conversion to the graph element. C++ does not seem to use this conversion operator.

Definition at line 221 of file graph_window_if_piter.hh.

10.222.4.3 `void mln::Site_Iterator< graph_window_if_piter< S, W, I> >::next() [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.223 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, graph_window_piter< S, W, I >, W::center_t >, and mln::internal::impl_selector< W::center_t, W::psite, graph_window_piter< S, W, I > >.

Public Types

- typedef S::fun_t::result [P](#)
Associated types
Type of the window elements.
- typedef W::center_t [center_t](#)
Type of the window center.
- typedef W::graph_element [graph_element](#)
Type of the graph element pointed by this iterator.

Public Member Functions

- void [change_target_site_set](#) (const S &s)
Change the target site set.
- void [next](#) ()
Go to the next element.
- const S & [target_site_set](#) () const
Return the target site set.
- [graph_window_piter](#) ()
Construction.
- template<typename Pref >
[graph_window_piter](#) (const [Window](#)< W > &win, const Pref &p_ref)
To be used in case the center and neighbor sites have the same type and belong to the same site set.
- template<typename Pref >
[graph_window_piter](#) (const [Window](#)< W > &win, const [Site_Set](#)< S > &target_site_set, const Pref &p_ref)
To be used in case center and neighbors sites do not have the same type and do not belong to the same site set.
- const [graph_element](#) & [element](#) () const
Return the graph element pointed by this iterator.
- unsigned [id](#) () const
Return the graph element id.

10.223.1 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

<i>S</i>	is the site set type.
<i>W</i>	is the window type.
<i>I</i>	is the underlying iterator type.

Definition at line 99 of file graph_window_piter.hh.

10.223.2 Member Typedef Documentation

10.223.2.1 `template<typename S , typename W , typename I > typedef W::center_t mln::graph_window_piter< S, W, I >::center_t`

Type of the window center.

Definition at line 120 of file graph_window_piter.hh.

10.223.2.2 `template<typename S , typename W , typename I > typedef W::graph_element mln::graph_window_piter< S, W, I >::graph_element`

Type of the graph element pointed by this iterator.

Definition at line 122 of file graph_window_piter.hh.

10.223.2.3 `template<typename S , typename W , typename I > typedef S::fun_t ::result mln::graph_window_piter< S, W, I >::P`

Associated types

Type of the window elements.

Definition at line 118 of file graph_window_piter.hh.

10.223.3 Constructor & Destructor Documentation

10.223.3.1 `template<typename S , typename W , typename I > mln::graph_window_piter< S, W, I >::graph_window_piter () [inline]`

Construction.

Definition at line 226 of file graph_window_piter.hh.

10.223.3.2 `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

<i>win</i>	The underlying window.
<i>p_ref</i>	Window center.

Definition at line 235 of file graph_window_piter.hh.

10.223.3.3 `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

<i>win</i>	The underlying window.
<i>target_site_set</i>	Site set in which neighbor sites are extracted.
<i>p_ref</i>	Window center.

Definition at line 249 of file graph_window_piter.hh.

10.223.4 Member Function Documentation

10.223.4.1 `template<typename S , typename W , typename I > void mln::graph_window_piter< S, W, I >::change_target_site_set (const S & s) [inline]`

Change the target site set.

[Window](#) elements different from the center come from the target site set.

Definition at line 357 of file `graph_window_piter.hh`.

10.223.4.2 `template<typename S , typename W , typename I > const graph_window_piter< S, W, I >::graph_element & mln::graph_window_piter< S, W, I >::element () const [inline]`

Return the graph element pointed by this iterator.

Definition at line 341 of file `graph_window_piter.hh`.

10.223.4.3 `template<typename S , typename W , typename I > unsigned mln::graph_window_piter< S, W, I >::id () const [inline]`

Return the graph element id.

FIXME: we do not want to have this member since there is an automatic conversion to the graph element. C++ does not seem to use this conversion operator.

Definition at line 349 of file `graph_window_piter.hh`.

10.223.4.4 `void mln::Site_Iterator< graph_window_piter< S, W, I > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.223.4.5 `template<typename S , typename W , typename I > const S & mln::graph_window_piter< S, W, I >::target_site_set () const [inline]`

Return the target site set.

[Window](#) elements different from the center come from the target site set.

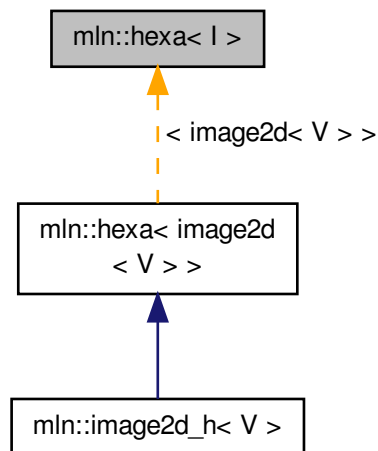
Definition at line 366 of file `graph_window_piter.hh`.

10.224 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 `l::lvalue lvalue`
Lvalue associated type.
- typedef `point2d_h psite`
Point site type.
- typedef `l::rvalue rvalue`
Return type of read-only access.
- typedef `hexa< tag::image_< I >> skeleton`
Skeleton.
- typedef `l::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 ()`
Constructor without argument.
- `hexa (I &ima)`
Constructor with an base image.
- `rvalue operator() (const point2d_h &p) const`

Read-only access of pixel value at hexa point site p .

- `lvalue operator()` (const `point2d_h` & p)

Read-write access of pixel value at hexa point site p .

10.224.1 Detailed Description

```
template<typename I> struct mln::hexa< I >
```

hexagonal image class.

The parameter I is the type of the base image. This image class which handles hexagonal grid.

Ex : 1 3 5 7 9 11

0 2 4 6 8 10

0 XX| | | | |XX

2 XX| | | | |XX

4 XX| | | | |XX

6 XX| | | | |XX

8 XX| | | | |XX

Definition at line 117 of file hexa.hh.

10.224.2 Member Typedef Documentation

10.224.2.1 `template<typename I> typedef hexa_bkd_piter_<box2d> mln::hexa< I >::bkd_piter`

FIXME : should it be in box2d_h? Backward [Site_iterator](#) associated type.

Definition at line 141 of file hexa.hh.

10.224.2.2 `template<typename I> typedef hexa_fwd_piter_<box2d> mln::hexa< I >::fwd_piter`

FIXME : should it be in box2d_h? Forward [Site_iterator](#) associated type.

Definition at line 137 of file hexa.hh.

10.224.2.3 `template<typename I> typedef I::lvalue mln::hexa< I >::lvalue`

Lvalue associated type.

Definition at line 127 of file hexa.hh.

10.224.2.4 `template<typename I> typedef point2d_h mln::hexa< I >::psite`

[Point](#) site type.

Definition at line 133 of file hexa.hh.

10.224.2.5 `template<typename I> typedef I ::rvalue mln::hexa< I >::rvalue`

Return type of read-only access.

Definition at line 130 of file hexa.hh.

10.224.2.6 `template<typename I> typedef hexa< tag::image_<I> > mln::hexa< I >::skeleton`

Skeleton.

Definition at line 121 of file hexa.hh.

10.224.2.7 `template<typename I> typedef I ::value mln::hexa< I >::value`

[Value](#) associated type.

Definition at line 124 of file hexa.hh.

10.224.3 Constructor & Destructor Documentation

10.224.3.1 `template<typename I> hexa< I >::hexa () [inline]`

Constructor without argument.

Definition at line 216 of file hexa.hh.

10.224.3.2 `template<typename I> hexa< I >::hexa (I & ima) [inline]`

Constructor with an base image.

Definition at line 223 of file hexa.hh.

10.224.4 Member Function Documentation

10.224.4.1 `template<typename I> const box2d_h & hexa< I >::domain () const [inline]`

Give the definition domain.

Definition at line 251 of file hexa.hh.

10.224.4.2 `template<typename I> bool hexa< I >::has (const psite & p) const [inline]`

Test if *p* belongs to the image domain.

Definition at line 260 of file hexa.hh.

10.224.4.3 `template<typename I> hexa< I >::rvalue hexa< I >::operator() (const point2d_h & p) const [inline]`

Read-only access of pixel value at hexa point site *p*.

Definition at line 231 of file hexa.hh.

10.224.4.4 `template<typename I> hexa<I>::lvalue hexa<I>::operator()(const point2d_h & p)` `[inline]`

Read-write access of pixel value at hexa point site `p`.

Definition at line 241 of file `hexa.hh`.

10.225 `mln::histo::array< T >` Struct Template Reference

Generic histogram class over a value set with type `T`.

```
#include <array.hh>
```

10.225.1 Detailed Description

```
template<typename T> struct mln::histo::array< T >
```

Generic histogram class over a value set with type `T`.

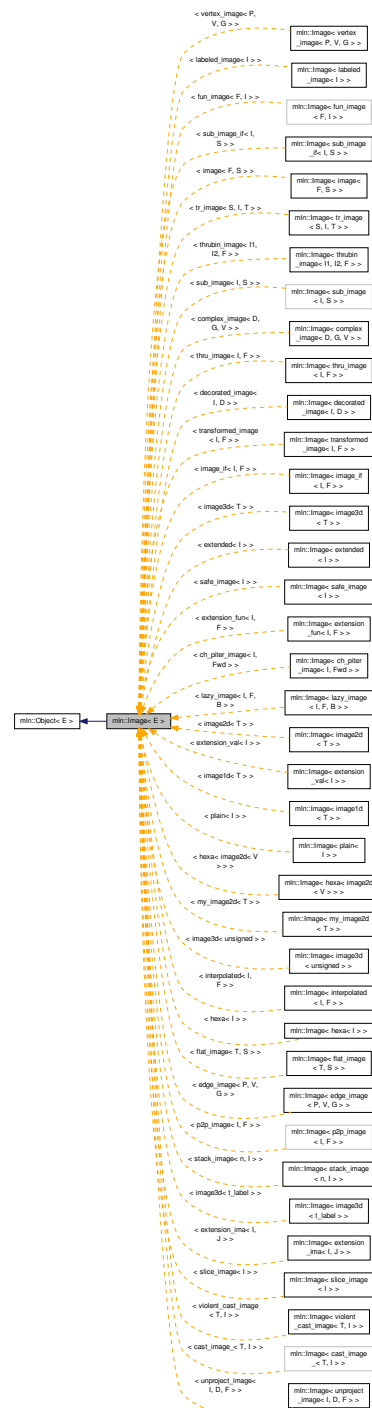
Definition at line 48 of file `histo/array.hh`.

10.226 `mln::Image< E >` Struct Template Reference

Base class for implementation of image classes.

```
#include <image.hh>
```

Inheritance diagram for mln::Image< E >:



10.226.1 Detailed Description

template<typename E>struct mln::Image< E >

Base class for implementation of image classes.

See Also

[mln::doc::Image](#) for a complete documentation of this class contents.

Definition at line 73 of file core/concept/image.hh.

10.227 mln::image1d< T > Struct Template Reference

Basic 1D image class.

```
#include <image1d.hh>
```

Inherits mln::internal::image_primary< T, box1d, image1d< T > >.

Public Types

- typedef T & [lvalue](#)
Return type of read-write access.
- typedef const T & [rvalue](#)
Return type of read-only access.
- typedef [image1d](#)< tag::value_< T > > [skeleton](#)
Skeleton.
- typedef T [value](#)
Value associated type.

Public Member Functions

- const [box1d](#) & [bbox](#) () const
Give the bounding box domain.
- unsigned [border](#) () const
Give the border thickness.
- const T * [buffer](#) () const
Give a hook to the value buffer.
- T * [buffer](#) ()
Give a hook to the value buffer.
- int [delta_index](#) (const [dpoint1d](#) &dp) const
Give the offset corresponding to the delta-point dp .
- const [box1d](#) & [domain](#) () const
Give the definition domain.
- const T & [element](#) (unsigned i) const
Read-only access to the i -th image value (including the border).
- T & [element](#) (unsigned i)
Read-write access to the i -th image value (including the border).
- bool [has](#) (const [point1d](#) &p) const
Test if p is valid.
- [image1d](#) ()
Constructor without argument.
- [image1d](#) (unsigned [ninds](#), unsigned bdr=border::thickness)
Constructor with the number of indices and the border thickness.
- [image1d](#) (const [box1d](#) &b, unsigned bdr=border::thickness)
Constructor with a box and the border thickness.
- unsigned [nelements](#) () const

- Give the number of cells (points including border ones).*
 - unsigned [ninds](#) () const
 - Give the number of indexes.*
 - const T & [operator\(\)](#) (const [point1d](#) &p) const
 - Read-only access to the image value located at point *p*.*
 - T & [operator\(\)](#) (const [point1d](#) &p)
 - Read-write access to the image value located at point *p*.*
 - [point1d point_at_index](#) (unsigned i) const
 - Give the point corresponding to the offset *o*.*
 - const [box1d](#) & [vbbox](#) () const
- virtual box, i.e., box including the virtual border*

10.227.1 Detailed Description

`template<typename T> struct mln::image1d< T >`

Basic 1D image class.

The parameter `T` is the type of pixel values. This image class stores data in memory and has a virtual border with constant thickness before and after data.

Definition at line 155 of file `image1d.hh`.

10.227.2 Member Typedef Documentation

10.227.2.1 `template<typename T> typedef T& mln::image1d< T >::lvalue`

Return type of read-write access.

Definition at line 167 of file `image1d.hh`.

10.227.2.2 `template<typename T> typedef const T& mln::image1d< T >::rvalue`

Return type of read-only access.

Definition at line 164 of file `image1d.hh`.

10.227.2.3 `template<typename T> typedef image1d< tag::value_<T> > mln::image1d< T >::skeleton`

Skeleton.

Definition at line 170 of file `image1d.hh`.

10.227.2.4 `template<typename T> typedef T mln::image1d< T >::value`

[Value](#) associated type.

Definition at line 161 of file `image1d.hh`.

10.227.3 Constructor & Destructor Documentation

10.227.3.1 `template<typename T> image1d< T >::image1d () [inline]`

Constructor without argument.

Definition at line 371 of file `image1d.hh`.

10.227.3.2 `template<typename T> image1d< T >::image1d (unsigned ninds, unsigned bdr = border::thickness) [inline]`

Constructor with the number of indices and the border thickness.

Definition at line 384 of file image1d.hh.

10.227.3.3 `template<typename T> image1d< T >::image1d (const box1d & b, unsigned bdr = border::thickness) [inline]`

Constructor with a box and the border thickness.

Definition at line 377 of file image1d.hh.

10.227.4 Member Function Documentation

10.227.4.1 `template<typename T> const box1d & image1d< T >::bbox () const [inline]`

Give the bounding box domain.

Definition at line 411 of file image1d.hh.

10.227.4.2 `template<typename T> unsigned image1d< T >::border () const [inline]`

Give the border thickness.

Definition at line 429 of file image1d.hh.

10.227.4.3 `template<typename T> const T * image1d< T >::buffer () const [inline]`

Give a hook to the value buffer.

Definition at line 520 of file image1d.hh.

10.227.4.4 `template<typename T> T * image1d< T >::buffer () [inline]`

Give a hook to the value buffer.

Definition at line 529 of file image1d.hh.

10.227.4.5 `template<typename T> int image1d< T >::delta_index (const dpoint1d & dp) const [inline]`

Give the offset corresponding to the delta-point *dp*.

Definition at line 538 of file image1d.hh.

10.227.4.6 `template<typename T> const box1d & image1d< T >::domain () const [inline]`

Give the definition domain.

Definition at line 402 of file image1d.hh.

10.227.4.7 `template<typename T> const T & image1d< T >::element (unsigned i) const [inline]`

Read-only access to the *i-th* image value (including the border).

Definition at line 502 of file image1d.hh.

10.227.4.8 `template<typename T> T & image1d< T >::element (unsigned i) [inline]`

Read-write access to the *i*-th image value (including the border).

Definition at line 511 of file image1d.hh.

10.227.4.9 `template<typename T> bool image1d< T >::has (const point1d & p) const [inline]`

Test if *p* is valid.

Definition at line 447 of file image1d.hh.

10.227.4.10 `template<typename T> unsigned image1d< T >::nelements () const [inline]`

Give the number of cells (points including border ones).

Definition at line 438 of file image1d.hh.

10.227.4.11 `template<typename T> unsigned image1d< T >::ninds () const [inline]`

Give the number of indexes.

Definition at line 483 of file image1d.hh.

10.227.4.12 `template<typename T> const T & image1d< T >::operator() (const point1d & p) const [inline]`

Read-only access to the image value located at point *p*.

Definition at line 456 of file image1d.hh.

10.227.4.13 `template<typename T> T & image1d< T >::operator() (const point1d & p) [inline]`

Read-write access to the image value located at point *p*.

Definition at line 465 of file image1d.hh.

10.227.4.14 `template<typename T> point1d image1d< T >::point_at_index (unsigned i) const [inline]`

Give the point corresponding to the offset *o*.

Definition at line 548 of file image1d.hh.

10.227.4.15 `template<typename T> const box1d & image1d< T >::vbbox () const [inline]`

virtual box, i.e., box including the virtual border

Definition at line 420 of file image1d.hh.

10.228 mln::image2d< T > Class Template Reference

Basic 2D image class.

```
#include <image2d.hh>
```

Inherits mln::internal::image_primary< T, mln::box2d, image2d< T > >.

Public Types

- typedef T & [lvalue](#)
Return type of read-write access.
- typedef const T & [rvalue](#)
Return type of read-only access.
- typedef [image2d](#)< tag::value_< T > > [skeleton](#)
Skeleton.
- typedef T [value](#)
[Value](#) associated type.

Public Member Functions

- const [box2d](#) & [bbox](#) () const
Give the bounding box domain.
- unsigned [border](#) () const
Give the border thickness.
- const T * [buffer](#) () const
Give a hook to the value buffer.
- T * [buffer](#) ()
Give a hook to the value buffer.
- int [delta_index](#) (const [dpoint2d](#) &dp) const
Give the delta-index corresponding to the delta-point dp .
- const [box2d](#) & [domain](#) () const
Give the definition domain.
- const T & [element](#) (unsigned i) const
Read-only access to the image value located at index i .
- T & [element](#) (unsigned i)
Read-write access to the image value located at index i .
- bool [has](#) (const [point2d](#) &p) const
Test if p is valid.
- [image2d](#) ()
Constructor without argument.
- [image2d](#) (int [nrows](#), int [ncols](#), unsigned bdr=[border::thickness](#))
Constructor with the numbers of rows and columns and the border thickness.
- [image2d](#) (const [box2d](#) &b, unsigned bdr=[border::thickness](#))
Constructor with a box and the border thickness (default is 3).
- unsigned [ncols](#) () const
Give the number of columns.
- unsigned [nelements](#) () const
Give the number of elements (points including border ones).
- unsigned [nrows](#) () const
Give the number of rows.
- const T & [operator\(\)](#) (const [point2d](#) &p) const
Read-only access to the image value located at point p .
- T & [operator\(\)](#) (const [point2d](#) &p)
Read-write access to the image value located at point p .
- [point2d](#) [point_at_index](#) (unsigned i) const
Give the point corresponding to the index i .

10.228.1 Detailed Description

`template<typename T>class mln::image2d< T >`

Basic 2D image class.

The parameter `T` is the type of pixel values. This image class stores data in memory and has a virtual border with constant thickness around data.

Definition at line 136 of file `core/image/image2d.hh`.

10.228.2 Member Typedef Documentation

10.228.2.1 `template<typename T> typedef T& mln::image2d< T >::lvalue`

Return type of read-write access.

Definition at line 148 of file `core/image/image2d.hh`.

10.228.2.2 `template<typename T> typedef const T& mln::image2d< T >::rvalue`

Return type of read-only access.

Definition at line 145 of file `core/image/image2d.hh`.

10.228.2.3 `template<typename T> typedef image2d< tag::value_<T> > mln::image2d< T >::skeleton`

Skeleton.

Definition at line 152 of file `core/image/image2d.hh`.

10.228.2.4 `template<typename T> typedef T mln::image2d< T >::value`

[Value](#) associated type.

Definition at line 142 of file `core/image/image2d.hh`.

10.228.3 Constructor & Destructor Documentation

10.228.3.1 `template<typename T > image2d< T >::image2d () [inline]`

Constructor without argument.

Definition at line 394 of file `core/image/image2d.hh`.

10.228.3.2 `template<typename T > image2d< T >::image2d (int nrows, int ncols, unsigned bdr = border::thickness) [inline]`

Constructor with the numbers of rows and columns and the border thickness.

Definition at line 400 of file `core/image/image2d.hh`.

10.228.3.3 `template<typename T > image2d< T >::image2d (const box2d & b, unsigned bdr = border::thickness) [inline]`

Constructor with a box and the border thickness (default is 3).

Definition at line 407 of file core/image/image2d.hh.

10.228.4 Member Function Documentation

10.228.4.1 `template<typename T> const box2d & image2d<T>::bbox () const` `[inline]`

Give the bounding box domain.

Definition at line 433 of file core/image/image2d.hh.

10.228.4.2 `template<typename T> unsigned image2d<T>::border () const` `[inline]`

Give the border thickness.

Definition at line 520 of file core/image/image2d.hh.

10.228.4.3 `template<typename T> const T * image2d<T>::buffer () const` `[inline]`

Give a hook to the value buffer.

Definition at line 556 of file core/image/image2d.hh.

10.228.4.4 `template<typename T> T * image2d<T>::buffer ()` `[inline]`

Give a hook to the value buffer.

Definition at line 565 of file core/image/image2d.hh.

10.228.4.5 `template<typename T> int image2d<T>::delta_index (const dpoint2d & dp) const` `[inline]`

Give the delta-index corresponding to the delta-point `dp`.

Definition at line 574 of file core/image/image2d.hh.

10.228.4.6 `template<typename T> const box2d & image2d<T>::domain () const` `[inline]`

Give the definition domain.

Definition at line 424 of file core/image/image2d.hh.

Referenced by `mln::morpho::line_gradient()`, `mln::make_debug_graph_image()`, and `mln::io::txt::save()`.

10.228.4.7 `template<typename T> const T & image2d<T>::element (unsigned i) const` `[inline]`

Read-only access to the image value located at index `i`.

Definition at line 538 of file core/image/image2d.hh.

10.228.4.8 `template<typename T> T & image2d<T>::element (unsigned i)` `[inline]`

Read-write access to the image value located at index `i`.

Definition at line 547 of file core/image/image2d.hh.

10.228.4.9 `template<typename T> bool image2d< T >::has (const point2d & p) const` `[inline]`

Test if *p* is valid.

Definition at line 451 of file core/image/image2d.hh.

Referenced by `mln::debug::put_word()`.

10.228.4.10 `template<typename T> unsigned image2d< T >::ncols () const` `[inline]`

Give the number of columns.

Definition at line 508 of file core/image/image2d.hh.

10.228.4.11 `template<typename T> unsigned image2d< T >::nelements () const` `[inline]`

Give the number of elements (points including border ones).

Definition at line 529 of file core/image/image2d.hh.

10.228.4.12 `template<typename T> unsigned image2d< T >::nrows () const` `[inline]`

Give the number of rows.

Definition at line 499 of file core/image/image2d.hh.

10.228.4.13 `template<typename T> const T & image2d< T >::operator() (const point2d & p) const` `[inline]`

Read-only access to the image value located at point *p*.

Definition at line 460 of file core/image/image2d.hh.

10.228.4.14 `template<typename T> T & image2d< T >::operator() (const point2d & p)` `[inline]`

Read-write access to the image value located at point *p*.

Definition at line 469 of file core/image/image2d.hh.

10.228.4.15 `template<typename T> point2d image2d< T >::point_at_index (unsigned i) const` `[inline]`

Give the point corresponding to the index *i*.

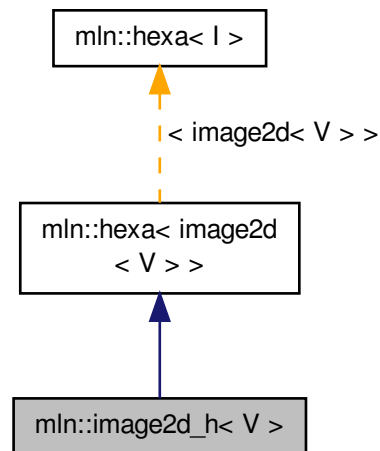
Definition at line 584 of file core/image/image2d.hh.

10.229 mln::image2d_h< V > Struct Template Reference

2d image based on an hexagonal mesh.

```
#include <image2d_h.hh>
```

Inheritance diagram for `mln::image2d_h< V >`:



Public Types

- typedef `hexa_bkd_piter_< box2d > bkd_piter`
FIXME : should it be in box2d_h? Backward Site_Iterator associated type.
- typedef `hexa_fwd_piter_< box2d > fwd_piter`
FIXME : should it be in box2d_h? Forward Site_Iterator associated type.
- typedef `image2d< V >::lvalue lvalue`
Lvalue associated type.
- typedef `point2d_h psite`
Point site type.
- typedef `image2d< V >::rvalue rvalue`
Return type of read-only access.
- typedef `hexa< tag::image_< image2d< V >> > skeleton`
Skeleton.
- typedef `image2d< V >::value value`
Value associated type.

Public Member Functions

- const `box2d_h & domain ()` const
Give the definition domain.
- bool `has (const psite &p)` const
Test if p belongs to the image domain.
- `image2d_h (int nrow, int ncol, unsigned bdr=border::thickness)`
Constructor with the numbers of rows and columns border thickness.
- `rvalue operator() (const point2d_h &p)` const
Read-only access of pixel value at hexa point site p .
- `lvalue operator() (const point2d_h &p)`
Read-write access of pixel value at hexa point site p .

10.229.1 Detailed Description

`template<typename V> struct mln::image2d_h< V >`

2d image based on an hexagonal mesh.

Definition at line 50 of file image2d_h.hh.

10.229.2 Member Typedef Documentation

10.229.2.1 `typedef hexa_bkd_piter_<box2d> mln::hexa< image2d< V > >::bkd_piter` [inherited]

FIXME : should it be in box2d_h? Backward Site_iterator associated type.

Definition at line 141 of file hexa.hh.

10.229.2.2 `typedef hexa_fwd_piter_<box2d> mln::hexa< image2d< V > >::fwd_piter` [inherited]

FIXME : should it be in box2d_h? Forward Site_iterator associated type.

Definition at line 137 of file hexa.hh.

10.229.2.3 `typedef image2d< V >::lvalue mln::hexa< image2d< V > >::lvalue` [inherited]

Lvalue associated type.

Definition at line 127 of file hexa.hh.

10.229.2.4 `template<typename V> typedef point2d_h mln::image2d_h< V >::psite`

[Point](#) site type.

Definition at line 56 of file image2d_h.hh.

10.229.2.5 `typedef image2d< V >::rvalue mln::hexa< image2d< V > >::rvalue` [inherited]

Return type of read-only access.

Definition at line 130 of file hexa.hh.

10.229.2.6 `typedef hexa< tag::image_<image2d< V > > > mln::hexa< image2d< V > >::skeleton`
[inherited]

Skeleton.

Definition at line 121 of file hexa.hh.

10.229.2.7 `typedef image2d< V >::value mln::hexa< image2d< V > >::value` [inherited]

Value associated type.

Definition at line 124 of file hexa.hh.

10.229.3 Constructor & Destructor Documentation

10.229.3.1 `template<typename V > mln::image2d_h< V >::image2d_h (int nrows, int ncols, unsigned bdr = border::thickness) [inline]`

Constructor with the numbers of rows and columns border thickness.

`image2d_h(3,6)` will build this hexa image :

```

  1   3   5
0  2   4
```

0| x x x | 2| x x x | 4| x x x

Definition at line 82 of file `image2d_h.hh`.

10.229.4 Member Function Documentation

10.229.4.1 `const box2d_h& mln::hexa< image2d< V > >::domain () const [inherited]`

Give the definition domain.

10.229.4.2 `bool mln::hexa< image2d< V > >::has (const psite & p) const [inherited]`

Test if `p` belongs to the image domain.

10.229.4.3 `rvalue mln::hexa< image2d< V > >::operator() (const point2d_h & p) const [inherited]`

Read-only access of pixel value at hexa point site `p`.

10.229.4.4 `lvalue mln::hexa< image2d< V > >::operator() (const point2d_h & p) [inherited]`

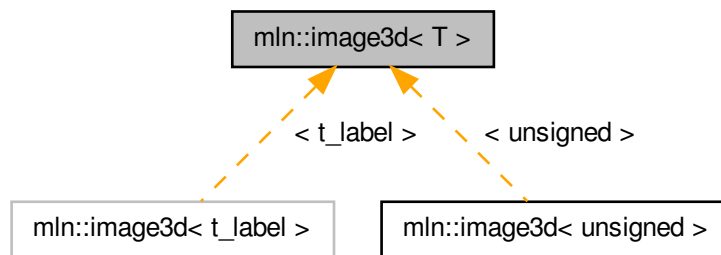
Read-write access of pixel value at hexa point site `p`.

10.230 mln::image3d< T > Struct Template Reference

Basic 3D image class.

```
#include <image3d.hh>
```

Inheritance diagram for mln::image3d< T >:



Public Types

- typedef T & [lvalue](#)
Return type of read-write access.
- typedef const T & [rvalue](#)
Return type of read-only access.
- typedef [image3d](#)< tag::value_< T > > [skeleton](#)
Skeleton.
- typedef T [value](#)
Value associated type.

Public Member Functions

- const [box3d](#) & [bbox](#) () const
Give the bounding box domain.
- unsigned [border](#) () const
Give the border thickness.
- const T * [buffer](#) () const
Give a hook to the value buffer.
- T * [buffer](#) ()
Give a hook to the value buffer.
- int [delta_index](#) (const [dpoint3d](#) &dp) const
Fast [Image](#) method.
- const [box3d](#) & [domain](#) () const
Give the definition domain.
- const T & [element](#) (unsigned i) const
*Read-only access to the image value located at index *i*.*
- T & [element](#) (unsigned i)
*Read-write access to the image value located at index *i*.*
- bool [has](#) (const [point3d](#) &p) const
*Test if *p* is valid.*
- [image3d](#) ()
Constructor without argument.
- [image3d](#) (const [box3d](#) &b, unsigned bdr=[border](#)::thickness)

- Constructor with a box and the border thickness (default is 3).*
- `image3d` (int `nslics`, int `nrows`, int `ncols`, unsigned `bdr=border::thickness`)
Constructor with the numbers of indexes and the border thickness.
- unsigned `ncols` () const
Give the number of columns.
- unsigned `nelements` () const
Give the number of cells (points including border ones).
- unsigned `nrows` () const
Give the number of rows.
- unsigned `nslics` () const
Give the number of slices.
- const T & `operator()` (const `point3d` &p) const
*Read-only access to the image value located at point *p*.*
- T & `operator()` (const `point3d` &p)
*Read-write access to the image value located at point *p*.*
- `point3d point_at_index` (unsigned o) const
*Give the point corresponding to the offset *o*.*
- const `box3d` & `vbbox` () const
virtual box, i.e., box including the virtual border

10.230.1 Detailed Description

```
template<typename T> struct mln::image3d< T >
```

Basic 3D image class.

The parameter `T` is the type of pixel values. This image class stores data in memory and has a virtual border with constant thickness around data.

Definition at line 130 of file `core/image/image3d.hh`.

10.230.2 Member Typedef Documentation

10.230.2.1 `template<typename T> typedef T& mln::image3d< T >::lvalue`

Return type of read-write access.

Definition at line 153 of file `core/image/image3d.hh`.

10.230.2.2 `template<typename T> typedef const T& mln::image3d< T >::rvalue`

Return type of read-only access.

Definition at line 150 of file `core/image/image3d.hh`.

10.230.2.3 `template<typename T> typedef image3d< tag::value_<T> > mln::image3d< T >::skeleton`

Skeleton.

Definition at line 157 of file `core/image/image3d.hh`.

10.230.2.4 `template<typename T> typedef T mln::image3d< T >::value`

`Value` associated type.

Definition at line 147 of file `core/image/image3d.hh`.

10.230.3 Constructor & Destructor Documentation

10.230.3.1 `template<typename T > image3d< T >::image3d () [inline]`

Constructor without argument.

Definition at line 389 of file core/image/image3d.hh.

10.230.3.2 `template<typename T > image3d< T >::image3d (const box3d & b, unsigned bdr = border::thickness) [inline]`

Constructor with a box and the border thickness (default is 3).

Definition at line 395 of file core/image/image3d.hh.

10.230.3.3 `template<typename T > image3d< T >::image3d (int nslis, int nrows, int ncols, unsigned bdr = border::thickness) [inline]`

Constructor with the numbers of indexes and the border thickness.

Definition at line 402 of file core/image/image3d.hh.

10.230.4 Member Function Documentation

10.230.4.1 `template<typename T > const box3d & image3d< T >::bbox () const [inline]`

Give the bounding box domain.

Definition at line 428 of file core/image/image3d.hh.

10.230.4.2 `template<typename T > unsigned image3d< T >::border () const [inline]`

Give the border thickness.

Definition at line 446 of file core/image/image3d.hh.

10.230.4.3 `template<typename T > const T * image3d< T >::buffer () const [inline]`

Give a hook to the value buffer.

Definition at line 554 of file core/image/image3d.hh.

10.230.4.4 `template<typename T > T * image3d< T >::buffer () [inline]`

Give a hook to the value buffer.

Definition at line 563 of file core/image/image3d.hh.

10.230.4.5 `template<typename T > int image3d< T >::delta_index (const dpoint3d & dp) const [inline]`

Fast [Image](#) method.

Give the offset corresponding to the delta-point dp.

Definition at line 572 of file core/image/image3d.hh.

10.230.4.6 `template<typename T> const box3d & image3d<T>::domain () const` `[inline]`

Give the definition domain.

Definition at line 419 of file core/image/image3d.hh.

Referenced by `mln::accu::stat::operator==()`.

10.230.4.7 `template<typename T> const T & image3d<T>::element (unsigned i) const` `[inline]`

Read-only access to the image value located at index `i`.

Definition at line 491 of file core/image/image3d.hh.

10.230.4.8 `template<typename T> T & image3d<T>::element (unsigned i)` `[inline]`

Read-write access to the image value located at index `i`.

Definition at line 500 of file core/image/image3d.hh.

10.230.4.9 `template<typename T> bool image3d<T>::has (const point3d & p) const` `[inline]`

Test if `p` is valid.

Definition at line 464 of file core/image/image3d.hh.

10.230.4.10 `template<typename T> unsigned image3d<T>::ncols () const` `[inline]`

Give the number of columns.

Definition at line 545 of file core/image/image3d.hh.

10.230.4.11 `template<typename T> unsigned image3d<T>::nelements () const` `[inline]`

Give the number of cells (points including border ones).

Definition at line 455 of file core/image/image3d.hh.

10.230.4.12 `template<typename T> unsigned image3d<T>::nrows () const` `[inline]`

Give the number of rows.

Definition at line 536 of file core/image/image3d.hh.

10.230.4.13 `template<typename T> unsigned image3d<T>::nslis () const` `[inline]`

Give the number of slices.

Definition at line 527 of file core/image/image3d.hh.

10.230.4.14 `template<typename T> const T & image3d<T>::operator() (const point3d & p) const` `[inline]`

Read-only access to the image value located at point `p`.

Definition at line 473 of file core/image/image3d.hh.

10.230.4.15 `template<typename T> T & image3d< T >::operator()(const point3d & p) [inline]`

Read-write access to the image value located at point p.

Definition at line 482 of file core/image/image3d.hh.

10.230.4.16 `template<typename T> point3d image3d< T >::point_at_index (unsigned o) const [inline]`

Give the point corresponding to the offset o.

Definition at line 583 of file core/image/image3d.hh.

10.230.4.17 `template<typename T> const box3d & image3d< T >::vbbox () const [inline]`

virtual box, i.e., box including the virtual border

Definition at line 437 of file core/image/image3d.hh.

10.231 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, interpolated< I, F > >.

Public Types

- typedef I::lvalue [lvalue](#)
Return type of read-write access.
- typedef I::psite [psite](#)
Point_Site associated type.
- typedef I::rvalue [rvalue](#)
Return type of read-only access.
- typedef [interpolated](#)
 < tag::image_< I >, F > [skeleton](#)
Skeleton.
- typedef I::value [value](#)
Value associated type.

Public Member Functions

- template<typename C>
 bool [has](#) (const mln::algebra::vec< I::psite::dim, C > &v) const
Test if a pixel value is accessible at v.
- [interpolated](#) (I &ima)
Constructors.
- bool [is_valid](#) () const
Test if this image has been initialized.

10.231.1 Detailed Description

`template<typename I, template< class > class F> struct mln::interpolated< I, F >`

Makes the underlying image being accessed with floating coordinates.

Definition at line 84 of file interpolated.hh.

10.231.2 Member Typedef Documentation

10.231.2.1 `template<typename I, template< class > class F> typedef I ::lvalue mln::interpolated< I, F >::lvalue`

Return type of read-write access.

Definition at line 98 of file interpolated.hh.

10.231.2.2 `template<typename I, template< class > class F> typedef I ::psite mln::interpolated< I, F >::psite`

Point_Site associated type.

Definition at line 92 of file interpolated.hh.

10.231.2.3 `template<typename I, template< class > class F> typedef I ::rvalue mln::interpolated< I, F >::rvalue`

Return type of read-only access.

Definition at line 101 of file interpolated.hh.

10.231.2.4 `template<typename I, template< class > class F> typedef interpolated< tag::image_<I>, F > mln::interpolated< I, F >::skeleton`

Skeleton.

Definition at line 104 of file interpolated.hh.

10.231.2.5 `template<typename I, template< class > class F> typedef I ::value mln::interpolated< I, F >::value`

[Value](#) associated type.

Definition at line 95 of file interpolated.hh.

10.231.3 Constructor & Destructor Documentation

10.231.3.1 `template<typename I , template< class > class F> interpolated< I, F >::interpolated (I & ima)
[inline]`

Constructors.

FIXME: don't we want a 'const' here?

Definition at line 156 of file interpolated.hh.

10.231.4 Member Function Documentation

10.231.4.1 `template<typename I, template< class > class F> template<typename C > bool interpolated< I, F >::has (const mln::algebra::vec< I::psite::dim, C > & v) const [inline]`

Test if a pixel value is accessible at v.

Definition at line 189 of file interpolated.hh.

10.231.4.2 `template<typename I, template< class > class F> bool interpolated< I, F >::is_valid () const [inline]`

Test if this image has been initialized.

Definition at line 180 of file interpolated.hh.

10.232 mln::io::dicom::dicom_header Struct Reference

Store dicom file header.

```
#include <get_header.hh>
```

10.232.1 Detailed Description

Store dicom file header.

Definition at line 59 of file dicom/get_header.hh.

10.233 mln::io::dump::dump_header Struct Reference

Store dump file header.

```
#include <get_header.hh>
```

10.233.1 Detailed Description

Store dump file header.

Definition at line 54 of file dump/get_header.hh.

10.234 mln::io::fld::fld_header Struct Reference

Define the header structure of an AVS field data file.

```
#include <header.hh>
```

10.234.1 Detailed Description

Define the header structure of an AVS field data file.

Definition at line 46 of file header.hh.

10.235 mln::io::raw::raw_header Struct Reference

Store raw file header.

```
#include <get_header.hh>
```

10.235.1 Detailed Description

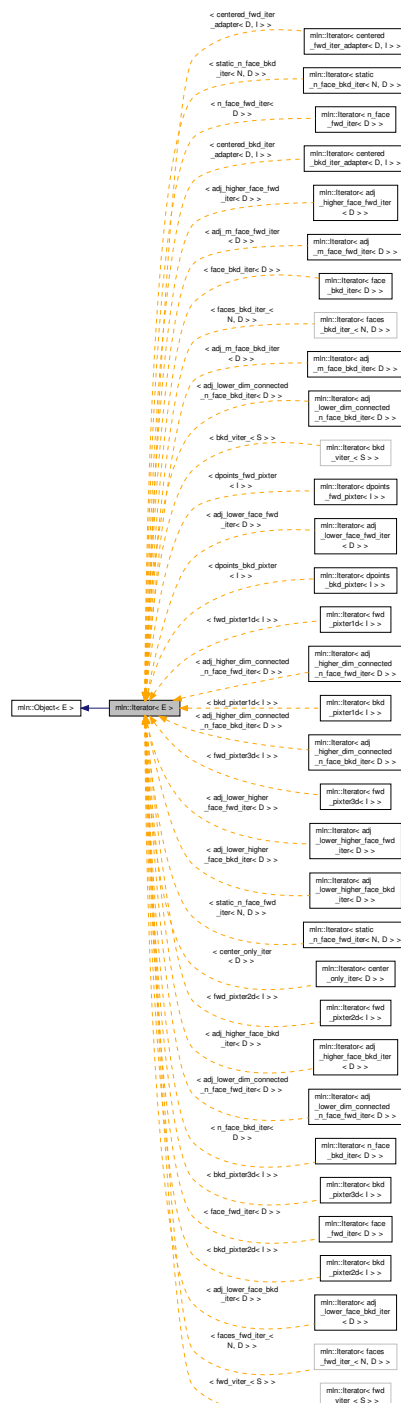
Store raw file header.

Definition at line 53 of file raw/get_header.hh.

10.236 mIn::Iterator< E > Struct Template Reference

Base class for implementation classes that are iterators.

```
#include <iterator.hh>
```



Public Member Functions

- void **next** ()
Go to the next element.

10.236.1 Detailed Description

```
template<typename E>struct mln::Iterator< E >
```

Base class for implementation classes that are iterators.

See Also

[mln::doc::Iterator](#) for a complete documentation of this class contents.

Definition at line 75 of file iterator.hh.

10.236.2 Member Function Documentation

10.236.2.1 `template<typename E > void mln::Iterator< E >::next ()`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

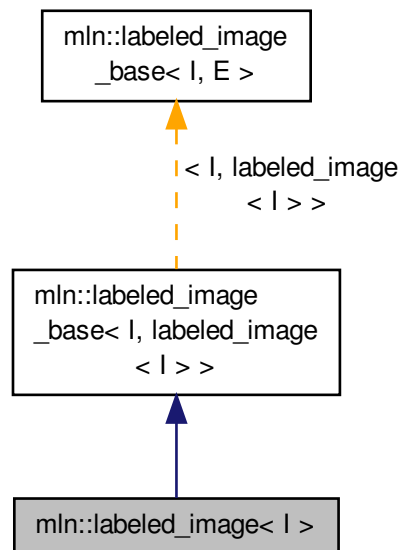
Definition at line 102 of file iterator.hh.

10.237 `mln::labeled_image< I >` Class Template Reference

Morpher providing an improved interface for labeled image.

```
#include <labeled_image.hh>
```


Inheritance diagram for mln::labeled_image< I >:



Public Types

- typedef `accu::shape::bbox`
`< typename I::psite >::result bbox_t`
Type of the bounding component bounding boxes.
- typedef `labeled_image`
`< tag::image_< I > > skeleton`
Skeleton.

Public Member Functions

- `const bbox_t & bbox (const typename I::value &label) const`
Return the bounding box of the component `label`.
- `const util::array< bbox_t > & bboxes () const`
Return the component bounding boxes.
- `I::value nlabels () const`
Return the number of labels;.
- `p_if< mln_box(I), fun::eq_v2b_expr< pw::value_< I >, pw::cst_< typename I::value > > > subdomain (const typename I::value &label) const`
Return the domain of the component with label `label`.
- `labeled_image ()`
Constructors
Constructor without argument.
- `labeled_image (const I &ima, const typename I::value &nlabels)`

Constructor from an image *ima* and the number of labels *nlabels*.

- `labeled_image` (const I &ima, const typename I::value &nlabels, const util::array< mIn_box(I)> &bboxes)

Constructor from an image *ima*, the number of labels *nlabels* and the object bounding boxes.

- void `relabel` (const Function_v2v< F > &f)

Relabel according to a function.

- void `relabel` (const Function_v2b< F > &f)

Labels may be removed.

Protected Member Functions

- void `update_data` (const fun::i2v::array< typename I::value > &relabel_fun)

Update bounding boxes information.

10.237.1 Detailed Description

```
template<typename I>class mIn::labeled_image< I >
```

Morpher providing an improved interface for labeled image.

Template Parameters

/	The label image type.
---	-----------------------

This image type allows to access every site set at a given label.

This image type guaranties that labels are contiguous (from 1 to n).

Definition at line 106 of file labeled_image.hh.

10.237.2 Member Typedef Documentation

10.237.2.1 `typedef accu::shape::bbox<typename I::psite>::result mIn::labeled_image_base< I, labeled_image< I >::bbox_t [inherited]`

Type of the bounding component bounding boxes.

Definition at line 124 of file labeled_image_base.hh.

10.237.2.2 `template<typename I> typedef labeled_image< tag::image_<I> > mIn::labeled_image< I >::skeleton`

Skeleton.

Definition at line 114 of file labeled_image.hh.

10.237.3 Constructor & Destructor Documentation

10.237.3.1 `template<typename I> labeled_image< I >::labeled_image () [inline]`

Constructors

Constructor without argument.

Definition at line 194 of file labeled_image.hh.

10.237.3.2 `template<typename I> labeled_image< I >::labeled_image (const I & ima, const typename I::value & nlabels) [inline]`

Constructor from an image *ima* and the number of labels *nlabels*.

Definition at line 200 of file labeled_image.hh.

10.237.3.3 `template<typename I> labeled_image< I >::labeled_image (const I & ima, const typename I::value & nlabels, const util::array< mln_box(I)> & bboxes) [inline]`

Constructor from an image *ima*, the number of labels *nlabels* and the object bounding boxes.

Definition at line 207 of file labeled_image.hh.

10.237.4 Member Function Documentation

10.237.4.1 `const bbox_t& mln::labeled_image_base< I, labeled_image< I > >::bbox (const typename I::value & label) const [inherited]`

Return the bounding box of the component *label*.

10.237.4.2 `const util::array< bbox_t>& mln::labeled_image_base< I, labeled_image< I > >::bboxes () const [inherited]`

Return the component bounding boxes.

10.237.4.3 `I::value mln::labeled_image_base< I, labeled_image< I > >::nlabels () const [inherited]`

Return the number of labels;.

10.237.4.4 `void mln::labeled_image_base< I, labeled_image< I > >::relabel (const Function_v2v< F> & f) [inherited]`

Relabel according to a function.

Merge or delete labels according to the given function. This method ensures that the labeling remains contiguous.

10.237.4.5 `void mln::labeled_image_base< I, labeled_image< I > >::relabel (const Function_v2b< F> & f) [inherited]`

Labels may be removed.

This overload make sure the labeling is still contiguous.

10.237.4.6 `p_if<mln_box(I), fun::eq_v2b_expr_<pw::value_<I>, pw::cst_<typename I::value> > > mln::labeled_image_base< I, labeled_image< I > >::subdomain (const typename I::value & label) const [inherited]`

Return the domain of the component with label *label*.

10.237.4.7 `void mln::labeled_image_base< I, labeled_image< I > >::update_data (const fun::i2v::array< typename I::value> & relabel_fun) [protected], [inherited]`

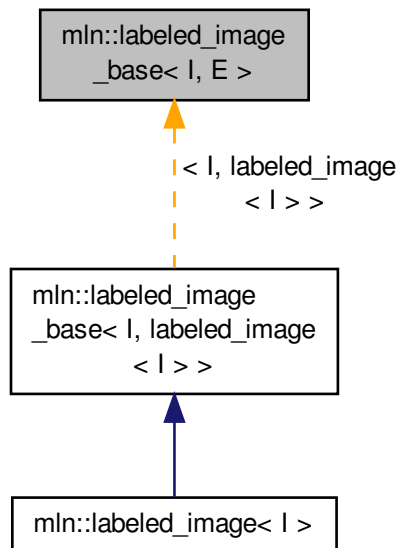
Update bounding boxes information.

10.238 mln::labeled_image_base< I, E > Class Template Reference

Base class Morpher providing an improved interface for labeled image.

```
#include <labeled_image_base.hh>
```

Inheritance diagram for mln::labeled_image_base< I, E >:



Public Types

- typedef [accu::shape::bbox](#)
`< typename I::psite >::result` [bbox_t](#)
Type of the bounding component bounding boxes.

Public Member Functions

- const [bbox_t](#) & [bbox](#) (const typename I::value &[label](#)) const
Return the bounding box of the component `label`.
- const [util::array< bbox_t >](#) & [bboxes](#) () const
Return the component bounding boxes.
- I::value [nlabels](#) () const
Return the number of labels;.
- [p_if< mln_box\(I\), fun::eq_v2b_expr< pw::value_< I >, pw::cst_< typename I::value > >](#) [subdomain](#) (const typename I::value &[label](#)) const
Return the domain of the component with label `label`.
- [labeled_image_base](#) ()

*Constructors**Constructor without argument.*

- template<typename F >
void [relabel](#) (const [Function_v2v](#)< F > &f)
Relabel according to a function.
- template<typename F >
void [relabel](#) (const [Function_v2b](#)< F > &f)
Labels may be removed.

Protected Member Functions

- void [update_data](#) (const fun::i2v::array< typename I::value > &relabel_fun)
Update bounding boxes information.

10.238.1 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

/	The label image type.
---	-----------------------

This image type allows to access every site set at a given label.

This image type guaranties that labels are contiguous (from 1 to n).

Definition at line 116 of file labeled_image_base.hh.

10.238.2 Member Typedef Documentation

10.238.2.1 `template<typename I, typename E> typedef accu::shape::bbox<typename I::psite>::result
mln::labeled_image_base< I, E >::bbox_t`

Type of the bounding component bounding boxes.

Definition at line 124 of file labeled_image_base.hh.

10.238.3 Constructor & Destructor Documentation

10.238.3.1 `template<typename I, typename E > labeled_image_base< I, E >::labeled_image_base ()
[inline]`

Constructors

Constructor without argument.

Definition at line 217 of file labeled_image_base.hh.

10.238.4 Member Function Documentation

10.238.4.1 `template<typename I, typename E> const labeled_image_base< I, E>::bbox_t & labeled_image_base< I, E>::bbox (const typename I::value & label) const`

Return the bounding box of the component `label`.

Definition at line 305 of file `labeled_image_base.hh`.

10.238.4.2 `template<typename I, typename E> const util::array< typename labeled_image_base< I, E>::bbox_t> & labeled_image_base< I, E>::bboxes () const`

Return the component bounding boxes.

Definition at line 313 of file `labeled_image_base.hh`.

10.238.4.3 `template<typename I, typename E> I::value labeled_image_base< I, E>::nlabels () const [inline]`

Return the number of labels;

Definition at line 273 of file `labeled_image_base.hh`.

10.238.4.4 `template<typename I, typename E> template<typename F> void labeled_image_base< I, E>::relabel (const Function_v2v< F> & f) [inline]`

Relabel according to a function.

Merge or delete labels according to the given function. This method ensures that the labeling remains contiguous.

Definition at line 226 of file `labeled_image_base.hh`.

References `mln::labeling::relabel_inplace()`, and `mln::make::relabelfun()`.

10.238.4.5 `template<typename I, typename E> template<typename F> void labeled_image_base< I, E>::relabel (const Function_v2b< F> & f) [inline]`

Labels may be removed.

This overload make sure the labeling is still contiguous.

Definition at line 252 of file `labeled_image_base.hh`.

References `mln::labeling::relabel_inplace()`, and `mln::make::relabelfun()`.

10.238.4.6 `template<typename I, typename E> p_if< mln_box(I), fun::eq_v2b_expr.< pw::value.< I>, pw::cst.< typename I::value>>> labeled_image_base< I, E>::subdomain (const typename I::value & label) const`

Return the domain of the component with label `label`.

Definition at line 322 of file `labeled_image_base.hh`.

10.238.4.7 `template<typename I, typename E> void labeled_image_base< I, E>::update_data (const fun::i2v::array< typename I::value> & relabel_fun) [protected]`

Update bounding boxes information.

Definition at line 281 of file `labeled_image_base.hh`.

References `mln::util::array< T>::size()`.

10.239 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, 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](#) ()
Constructors.
- [lazy_image](#) (const F &fun, const B &[box](#))
Constructors.
- F::result [operator\(\)](#) (const typename F::input &x) const
Read-only access of pixel value at F::input x.
- F::result [operator\(\)](#) (const typename F::input &x)
Read and "write if possible" access of pixel value at F::input x.
- [rvalue operator\(\)](#) (const typename I::psite &p) const
Read-only access of pixel value at point site p.
- [lvalue operator\(\)](#) (const typename I::psite &p)
Read and "write if possible" access of pixel value at point site p.

10.239.1 Detailed Description

```
template<typename I, typename F, typename B>struct mln::lazy_image< I, F, B >
```

Image values are computed on the fly.

The parameter `I` is the type of image. The parameter `F` is the type of function. The parameter `B` is the type of box.

This image class take a functor `fun` and a box `box`. Access to `ima(p)` where `p` include `box` return `fun(b)` lazily.

Definition at line 92 of file `lazy_image.hh`.

10.239.2 Member Typedef Documentation

10.239.2.1 `template<typename I, typename F, typename B> typedef F ::result mln::lazy_image< I, F, B >::lvalue`

Return type of read-write access.

Definition at line 104 of file lazy_image.hh.

10.239.2.2 `template<typename I, typename F, typename B> typedef F ::result mln::lazy_image< I, F, B >::rvalue`

Return type of read access.

Definition at line 101 of file lazy_image.hh.

10.239.2.3 `template<typename I, typename F, typename B> typedef lazy_image< tag::image_<I>, F, B > mln::lazy_image< I, F, B >::skeleton`

Skeleton.

Definition at line 107 of file lazy_image.hh.

10.239.3 Constructor & Destructor Documentation

10.239.3.1 `template<typename I, typename F, typename B> mln::lazy_image< I, F, B >::lazy_image ()`

Constructors.

10.239.3.2 `template<typename I , typename F, typename B> lazy_image< I, F, B >::lazy_image (const F & fun, const B & box) [inline]`

Constructors.

Definition at line 161 of file lazy_image.hh.

10.239.4 Member Function Documentation

10.239.4.1 `template<typename I , typename F , typename B > const box< typename I::psite > & lazy_image< I, F, B >::domain () const [inline]`

Return domain of lazyd_image.

Definition at line 226 of file lazy_image.hh.

10.239.4.2 `template<typename I, typename F, typename B > bool lazy_image< I, F, B >::has (const typename I::psite & p) const [inline]`

Test if a pixel value is accessible at p.

Definition at line 175 of file lazy_image.hh.

10.239.4.3 `template<typename I , typename F, typename B > F::result lazy_image< I, F, B >::operator() (const typename F::input & x) const [inline]`

Read-only access of pixel value at F::input x.

Definition at line 183 of file lazy_image.hh.


```
10.239.4.4  template<typename I, typename F, typename B > F::result lazy_image< I, F, B >::operator() ( const typename
F::input & x )  [inline]
```

Read and "write if possible" access of pixel value at F::input x.

Definition at line 197 of file lazy_image.hh.

```
10.239.4.5  template<typename I, typename F, typename B > lazy_image< I, F, B >::rvalue lazy_image< I, F, B
>::operator() ( const typename I::psite & p ) const  [inline]
```

Read-only access of pixel value at point site p.

Definition at line 210 of file lazy_image.hh.

```
10.239.4.6  template<typename I, typename F, typename B > lazy_image< I, F, B >::lvalue lazy_image< I, F, B
>::operator() ( const typename I::psite & p )  [inline]
```

Read and "write if possible" access of pixel value at point site p.

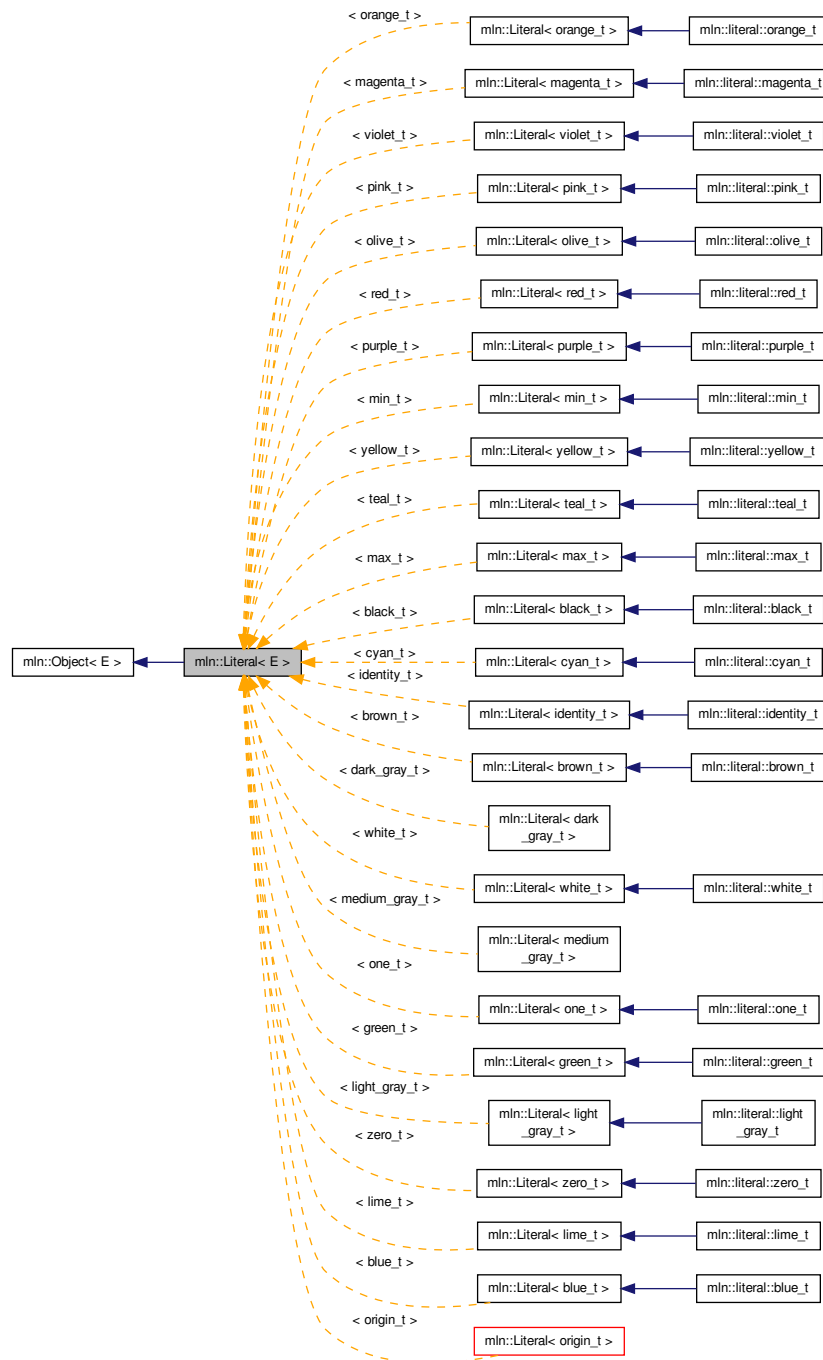
Definition at line 218 of file lazy_image.hh.

10.240 mIn::Literal< E > Struct Template Reference

Base class for implementation classes of literals.

```
#include <literal.hh>
```

Inheritance diagram for `mln::Literal< E >`:



10.240.1 Detailed Description

```
template<typename E>struct mln::Literal< E >
```

Base class for implementation classes of literals.

See Also

mIn::doc::Literal for a complete documentation of this class contents.

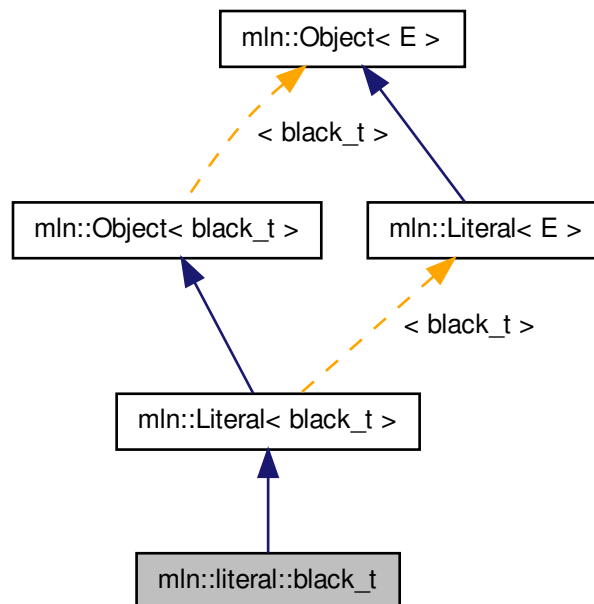
Definition at line 56 of file literal.hh.

10.241 mIn::literal::black_t Struct Reference

Type of literal black.

```
#include <black.hh>
```

Inheritance diagram for mIn::literal::black_t:



10.241.1 Detailed Description

Type of literal black.

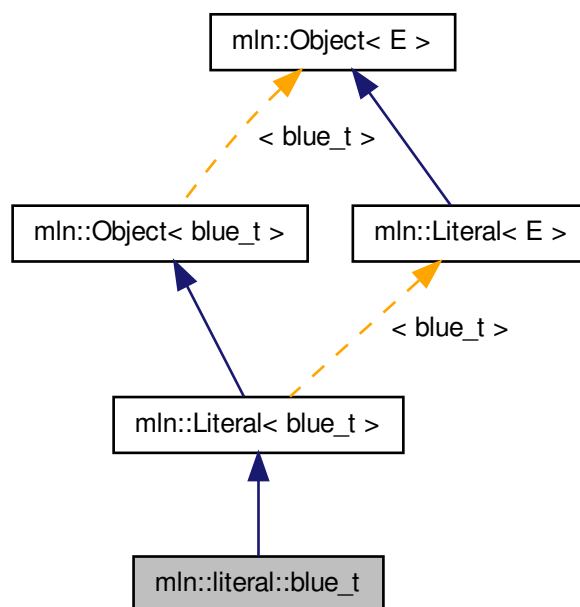
Definition at line 43 of file black.hh.

10.242 mIn::literal::blue_t Struct Reference

Type of literal blue.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::blue_t:



10.242.1 Detailed Description

Type of literal blue.

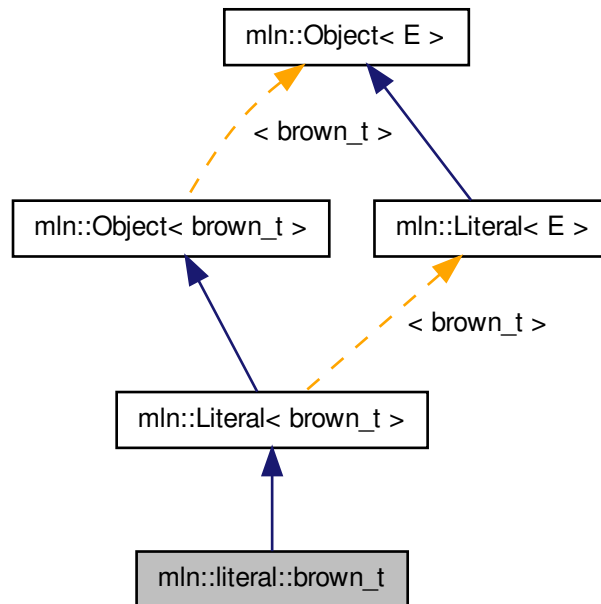
Definition at line 59 of file colors.hh.

10.243 mln::literal::brown_t Struct Reference

Type of literal brown.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::brown_t:



10.243.1 Detailed Description

Type of literal brown.

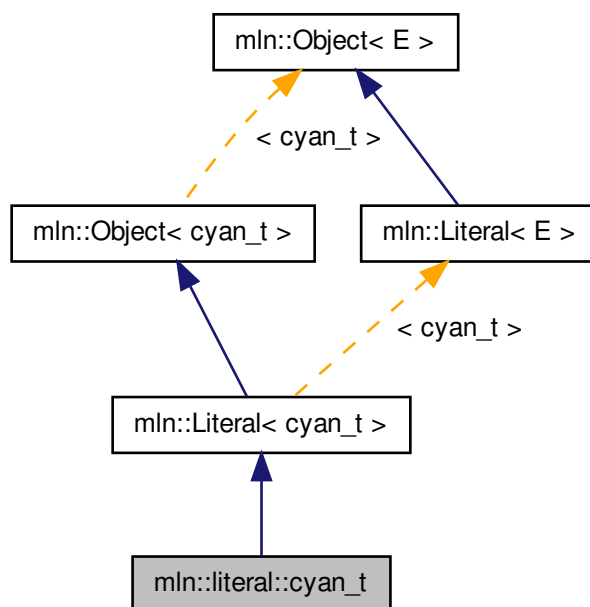
Definition at line 67 of file colors.hh.

10.244 mln::literal::cyan_t Struct Reference

Type of literal cyan.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::cyan_t:



10.244.1 Detailed Description

Type of literal cyan.

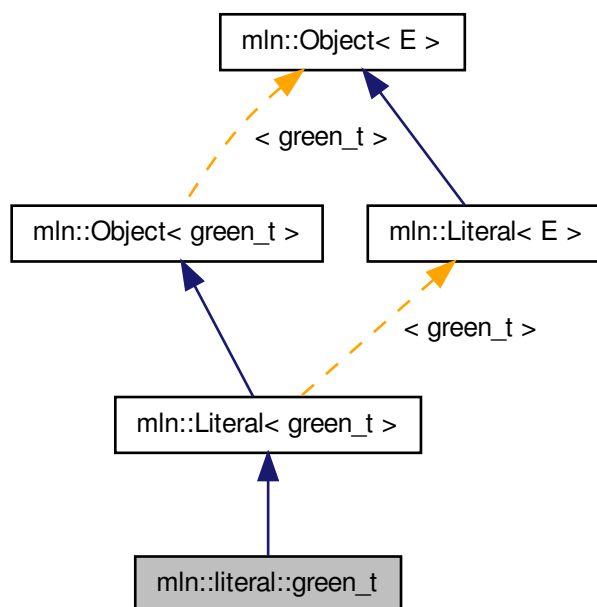
Definition at line 123 of file colors.hh.

10.245 mln::literal::green_t Struct Reference

Type of literal green.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::green_t:



10.245.1 Detailed Description

Type of literal green.

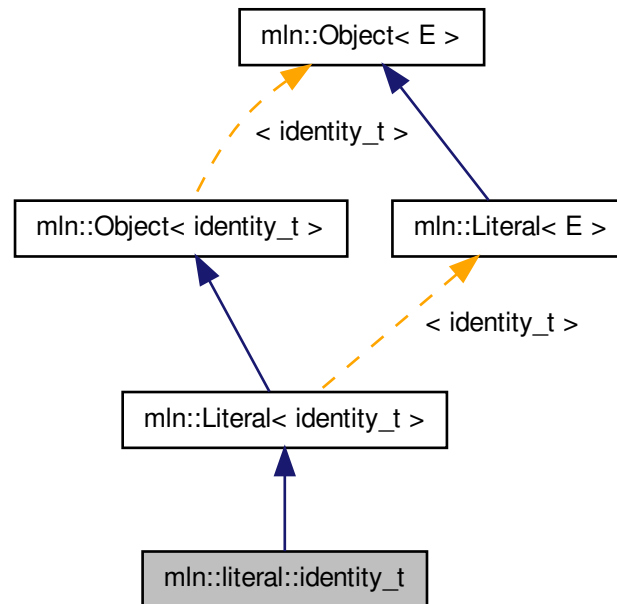
Definition at line 51 of file colors.hh.

10.246 mln::literal::identity_t Struct Reference

Type of literal identity.

```
#include <identity.hh>
```

Inheritance diagram for mln::literal::identity_t:



10.246.1 Detailed Description

Type of literal identity.

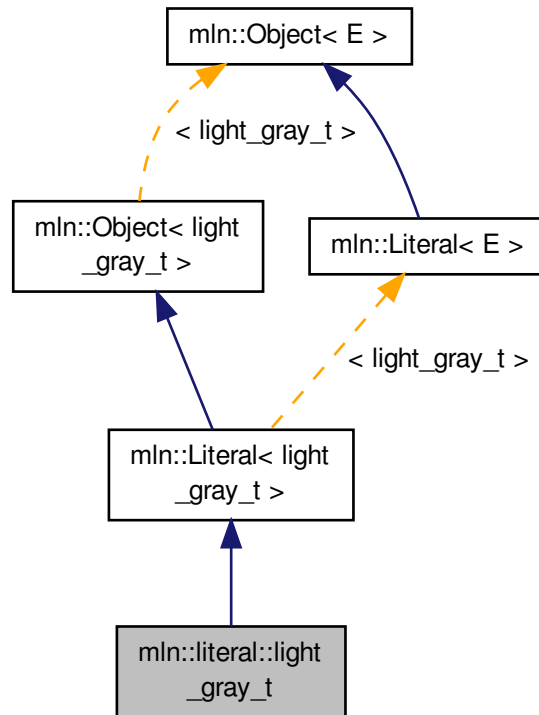
Definition at line 44 of file `identity.hh`.

10.247 mln::literal::light_gray_t Struct Reference

Type of literal grays.

```
#include <grays.hh>
```


Inheritance diagram for mln::literal::light_gray_t:



10.247.1 Detailed Description

Type of literal grays.

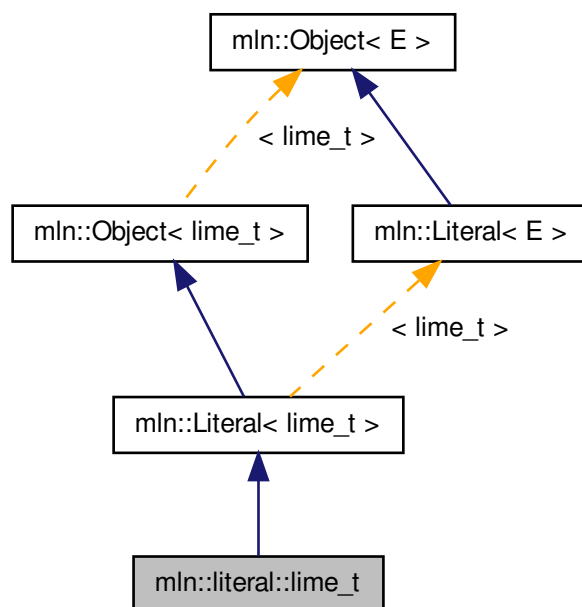
Definition at line 44 of file `grays.hh`.

10.248 mln::literal::lime_t Struct Reference

Type of literal lime.

```
#include <colors.hh>
```

Inheritance diagram for `mIn::literal::lime_t`:



10.248.1 Detailed Description

Type of literal lime.

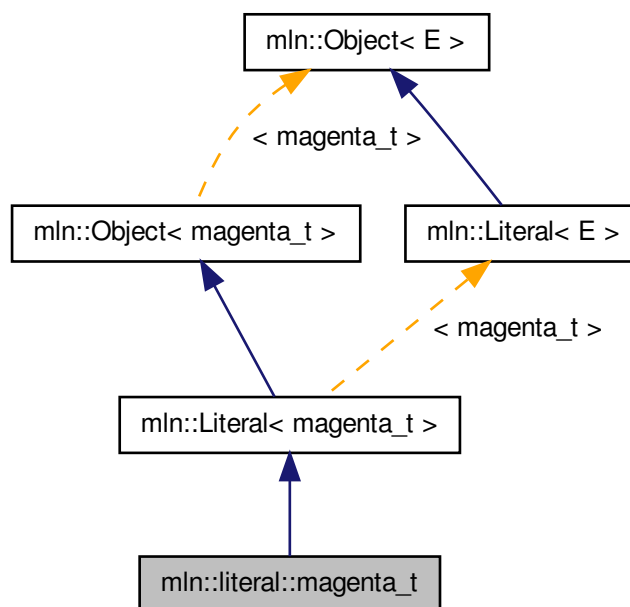
Definition at line 75 of file `colors.hh`.

10.249 mIn::literal::magenta_t Struct Reference

Type of literal magenta.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::magenta_t:



10.249.1 Detailed Description

Type of literal magenta.

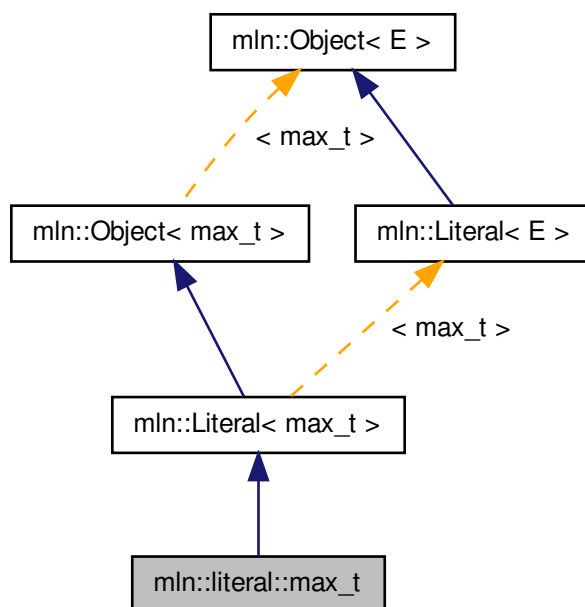
Definition at line 131 of file colors.hh.

10.250 mln::literal::max_t Struct Reference

Type of literal max.

```
#include <max.hh>
```

Inheritance diagram for mln::literal::max_t:



10.250.1 Detailed Description

Type of literal max.

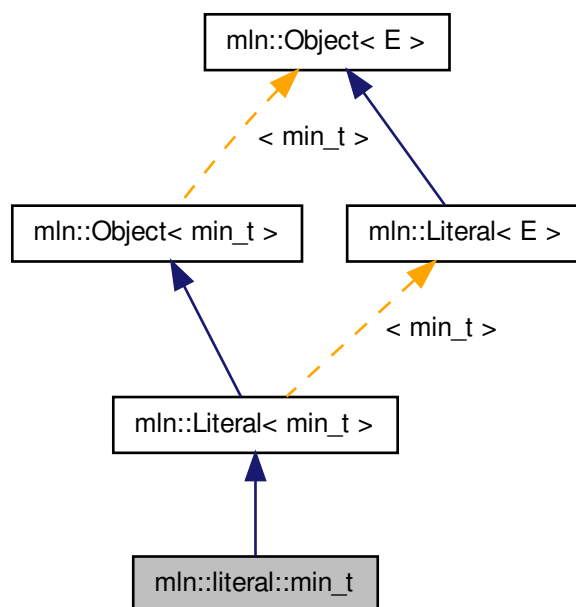
Definition at line 45 of file literal/max.hh.

10.251 mln::literal::min_t Struct Reference

Type of literal min.

```
#include <min.hh>
```

Inheritance diagram for mln::literal::min_t:



10.251.1 Detailed Description

Type of literal min.

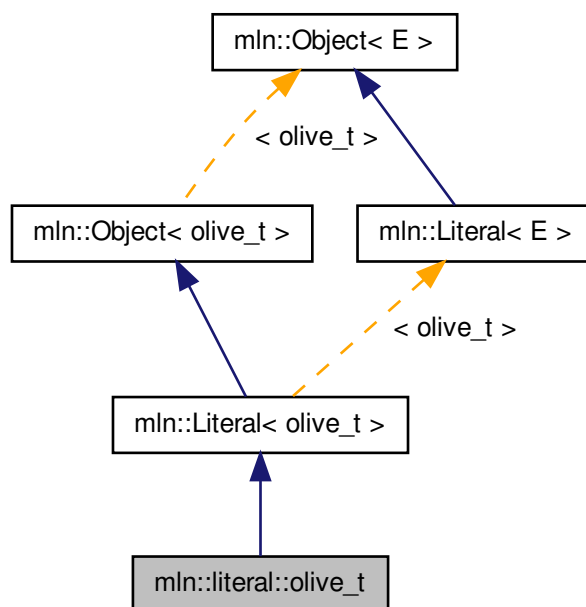
Definition at line 44 of file literal/min.hh.

10.252 mln::literal::olive_t Struct Reference

Type of literal olive.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::olive_t:



10.252.1 Detailed Description

Type of literal olive.

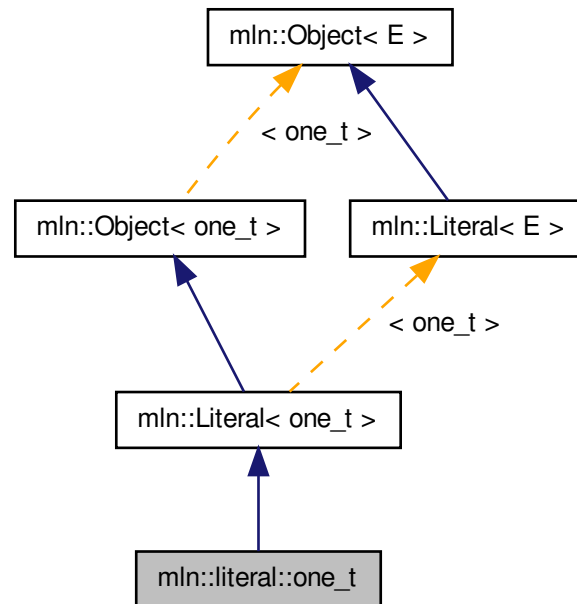
Definition at line 147 of file colors.hh.

10.253 mln::literal::one_t Struct Reference

Type of literal one.

```
#include <one.hh>
```

Inheritance diagram for mln::literal::one_t:



10.253.1 Detailed Description

Type of literal one.

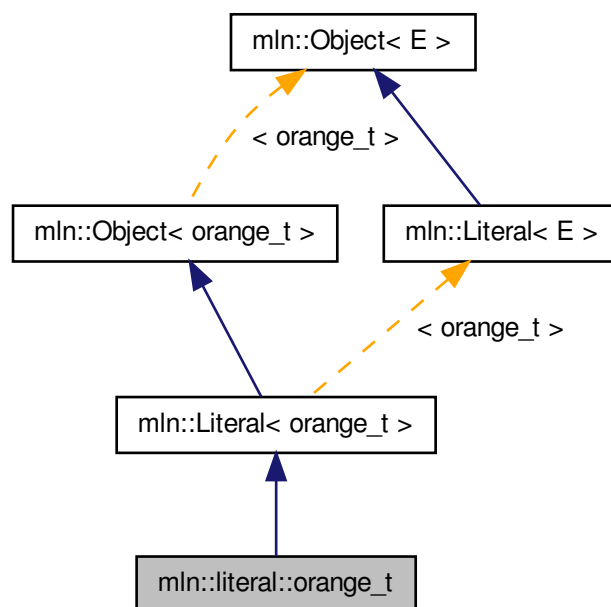
Definition at line 45 of file one.hh.

10.254 mln::literal::orange_t Struct Reference

Type of literal orange.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::orange_t:



10.254.1 Detailed Description

Type of literal orange.

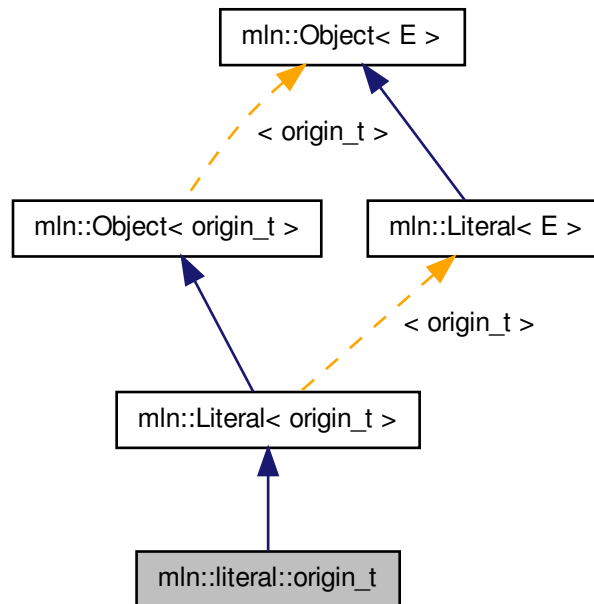
Definition at line 83 of file colors.hh.

10.255 mln::literal::origin_t Struct Reference

Type of literal origin.

```
#include <origin.hh>
```


Inheritance diagram for mln::literal::origin_t:



10.255.1 Detailed Description

Type of literal origin.

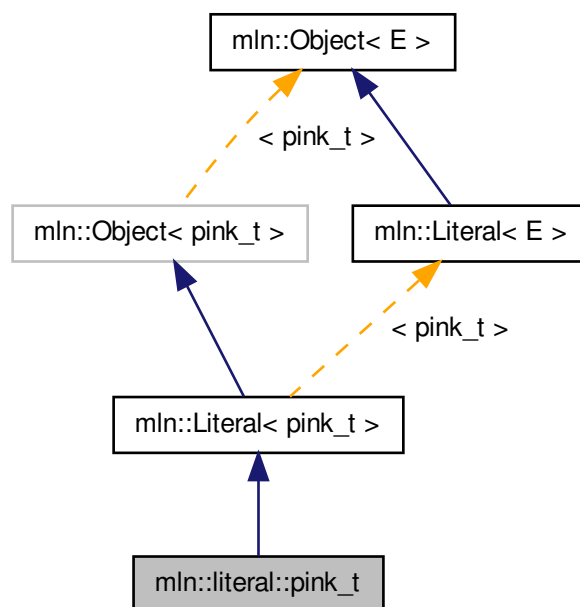
Definition at line 45 of file `origin.hh`.

10.256 mln::literal::pink_t Struct Reference

Type of literal pink.

```
#include <colors.hh>
```

Inheritance diagram for `mln::literal::pink_t`:



10.256.1 Detailed Description

Type of literal pink.

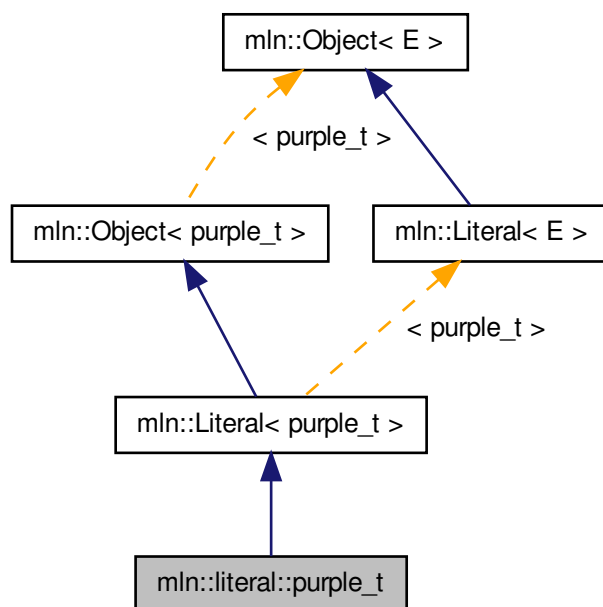
Definition at line 91 of file `colors.hh`.

10.257 mln::literal::purple_t Struct Reference

Type of literal purple.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::purple_t:



10.257.1 Detailed Description

Type of literal purple.

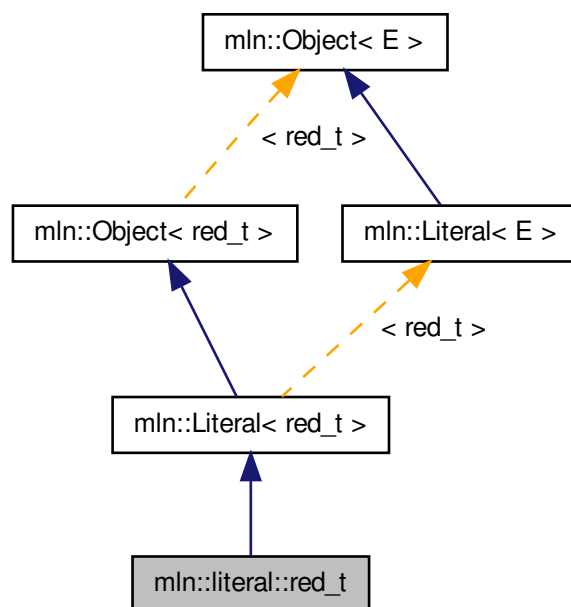
Definition at line 99 of file colors.hh.

10.258 mln::literal::red_t Struct Reference

Type of literal red.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::red_t:



10.258.1 Detailed Description

Type of literal red.

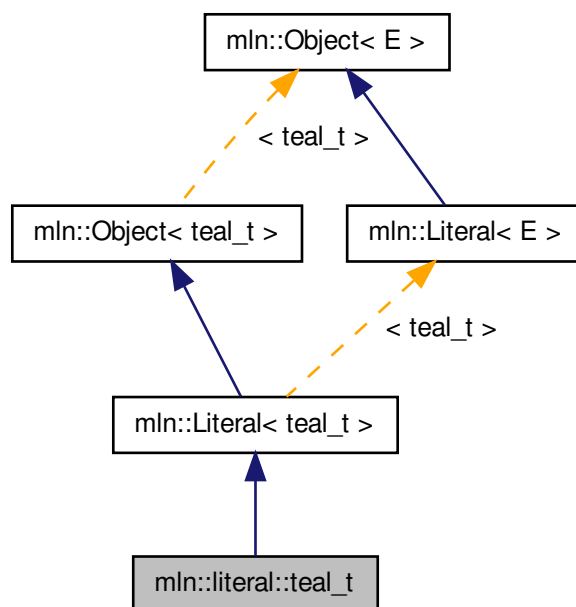
Definition at line 43 of file colors.hh.

10.259 mln::literal::teal_t Struct Reference

Type of literal teal.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::teal_t:



10.259.1 Detailed Description

Type of literal teal.

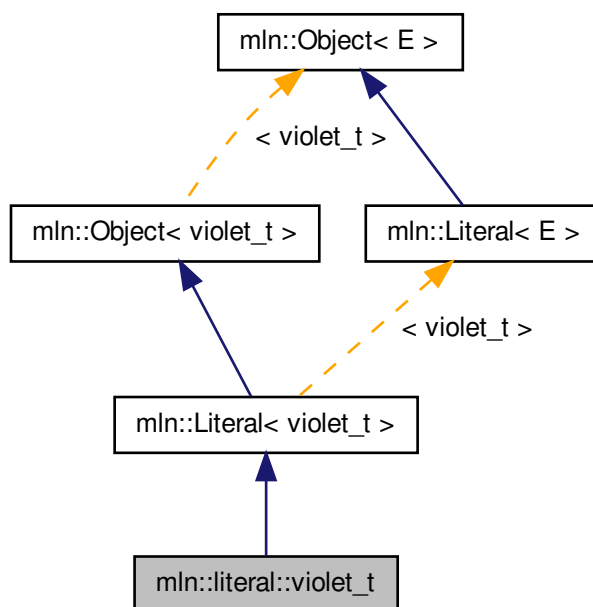
Definition at line 107 of file colors.hh.

10.260 mln::literal::violet_t Struct Reference

Type of literal violet.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::violet_t:



10.260.1 Detailed Description

Type of literal violet.

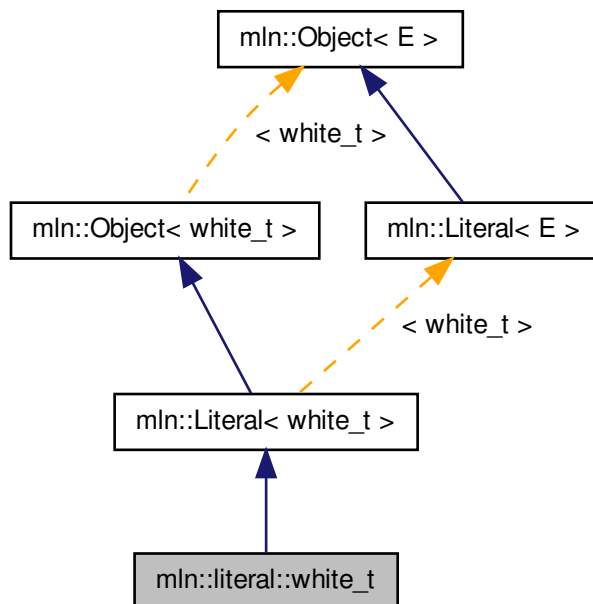
Definition at line 115 of file colors.hh.

10.261 mln::literal::white_t Struct Reference

Type of literal white.

```
#include <white.hh>
```

Inheritance diagram for mln::literal::white_t:



10.261.1 Detailed Description

Type of literal white.

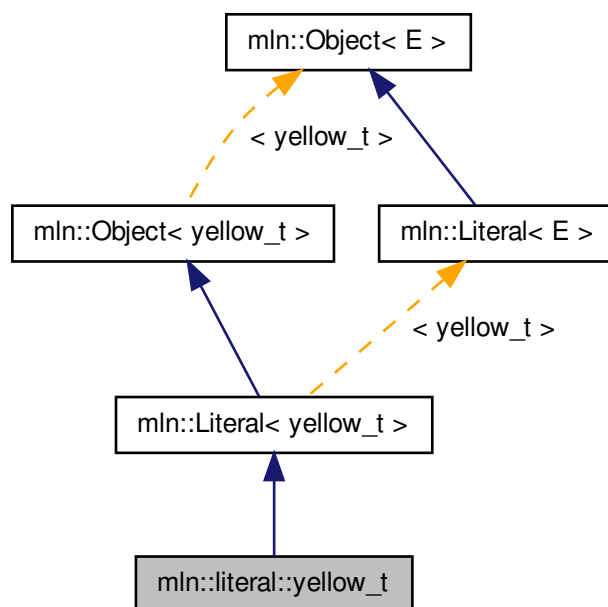
Definition at line 43 of file white.hh.

10.262 mln::literal::yellow_t Struct Reference

Type of literal yellow.

```
#include <colors.hh>
```

Inheritance diagram for mln::literal::yellow_t:



10.262.1 Detailed Description

Type of literal yellow.

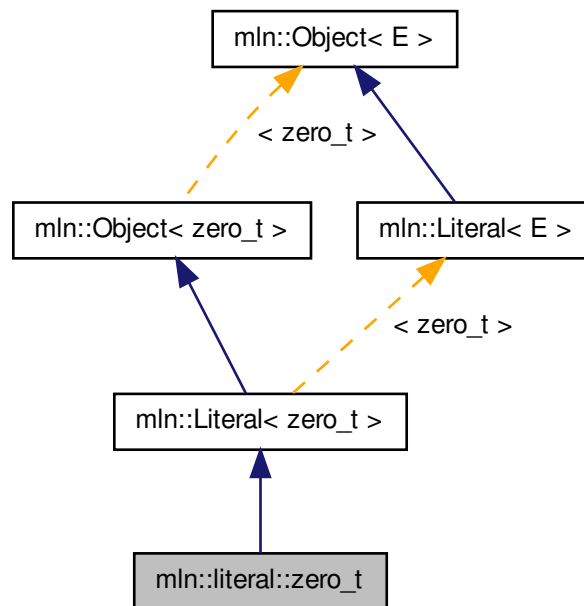
Definition at line 139 of file colors.hh.

10.263 mln::literal::zero_t Struct Reference

Type of literal zero.

```
#include <zero.hh>
```


Inheritance diagram for mln::literal::zero_t:



10.263.1 Detailed Description

Type of literal zero.

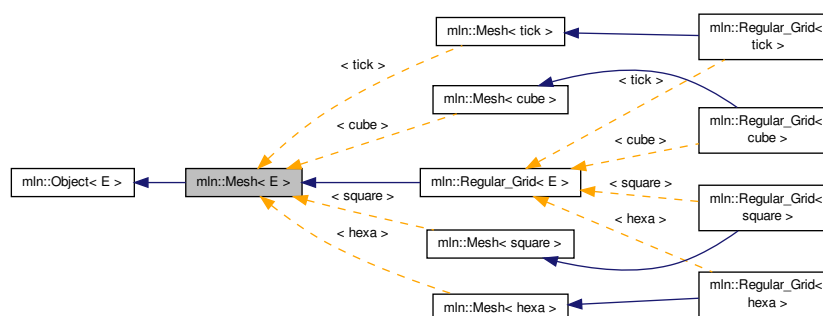
Definition at line 47 of file `zero.hh`.

10.264 mln::Mesh< E > Struct Template Reference

Base class for implementation classes of meshes.

```
#include <mesh.hh>
```

Inheritance diagram for `mln::Mesh< E >`:



10.264.1 Detailed Description

```
template<typename E>struct mln::Mesh< E >
```

Base class for implementation classes of meshes.

See Also

mln::doc::Mesh for a complete documentation of this class contents.

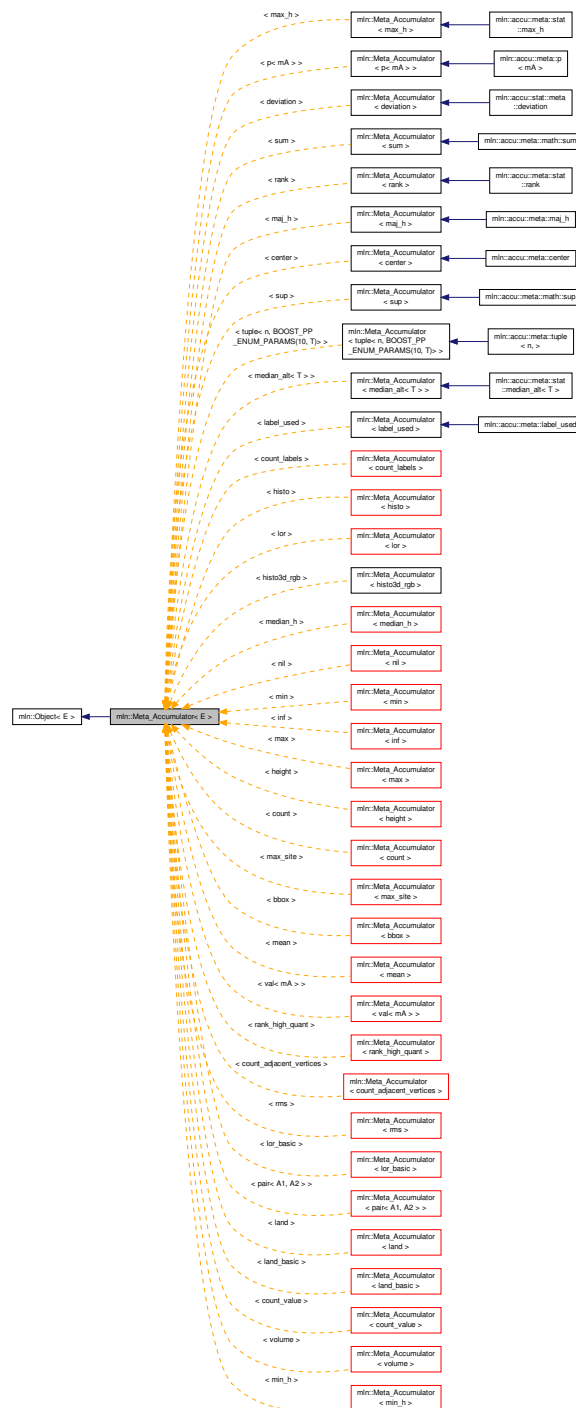
Definition at line 45 of file mesh.hh.

10.265 mln::Meta_Accumulator< E > Struct Template Reference

Base class for implementation of meta accumulators.

```
#include <meta_accumulator.hh>
```

Inheritance diagram for mln::Meta_Accumulator< E >:



10.265.1 Detailed Description

`template<typename E> struct mln::Meta_Accumulator< E >`

Base class for implementation of meta accumulators.

The parameter *E* is the exact type.

See Also

`mln::doc::Meta_Accumulator` for a complete documentation of this class contents.

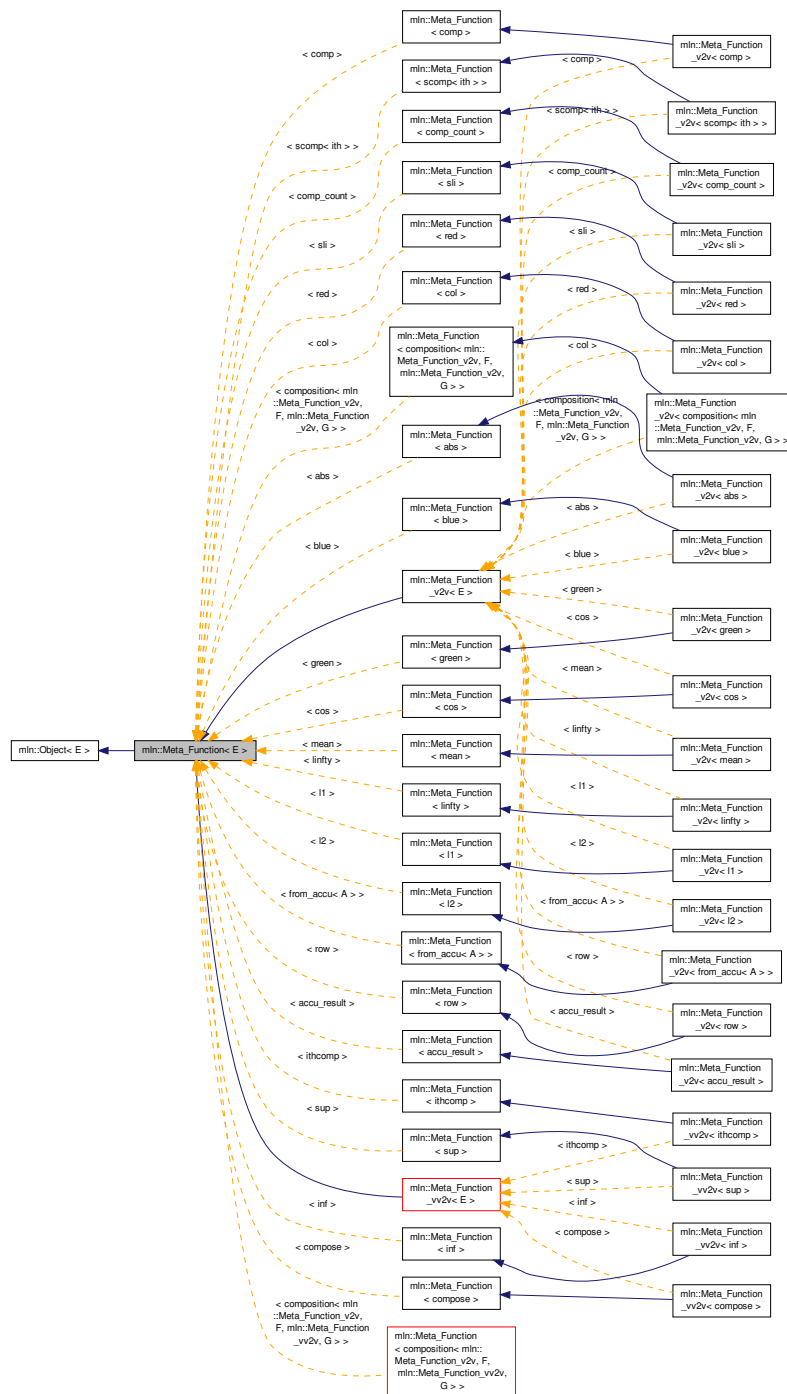
Definition at line 103 of file `meta_accumulator.hh`.

10.266 `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 >:



10.266.1 Detailed Description

```
template<typename E> struct mln::Meta_Function< E >
```

Base class for implementation of meta functions.

The parameter *E* is the exact type.

See Also

`mln::doc::Meta_Function` for a complete documentation of this class contents.

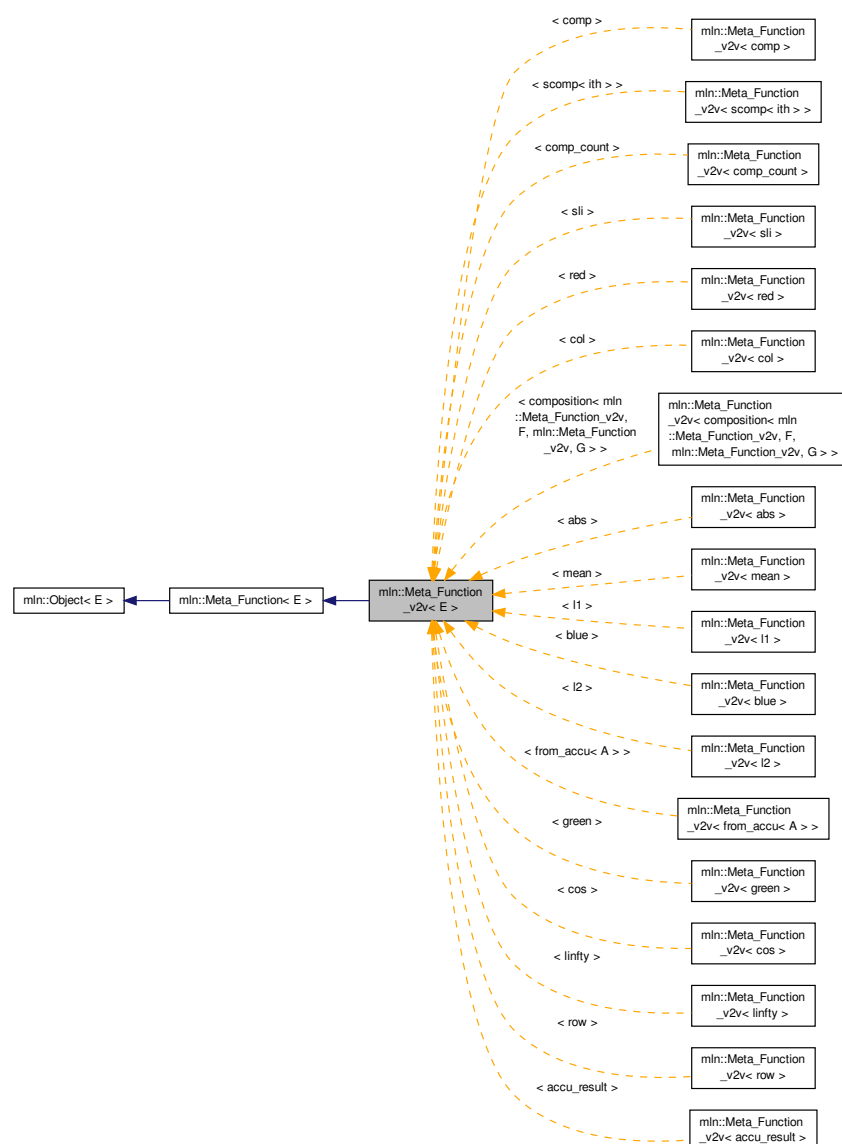
Definition at line 78 of file `meta_function.hh`.

10.267 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 >`:



10.267.1 Detailed Description

```
template<typename E>struct mln::Meta_Function_v2v< E >
```

Base class for implementation of function-objects from value to value.

The parameter *E* is the exact type.

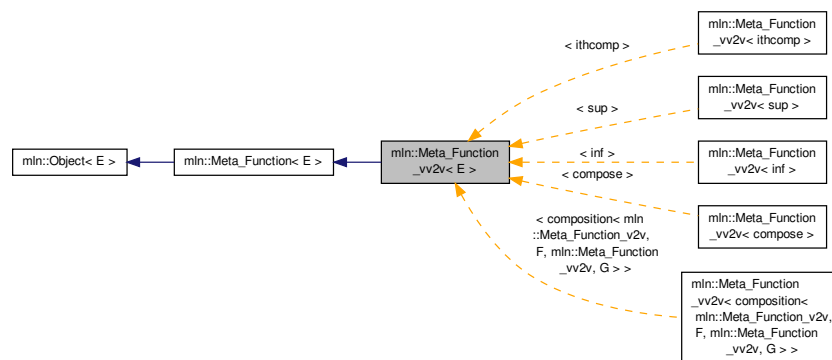
Definition at line 98 of file meta_function.hh.

10.268 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 >:



10.268.1 Detailed Description

```
template<typename E>struct mln::Meta_Function_vv2v< E >
```

Base class for implementation of function-objects from value to value.

The parameter *E* is the exact type.

Definition at line 120 of file meta_function.hh.

10.269 mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 > Struct Template Reference

Ands type.

```
#include <ands.hh>
```

Inherits `bool_<(E1::value &&E2::value &&E3::value &&E4::value &&E5::value &&E6::value &&E7::value &&E8::value)>`.

10.269.1 Detailed Description

```
template<typename E1, typename E2, typename E3, typename E4 = true_, typename E5 = true_, typename E6 = true_, typename E7 = true_, typename E8 = true_>struct mln::metal::ands< E1, E2, E3, E4, E5, E6, E7, E8 >
```

Ands type.

Definition at line 51 of file ands.hh.

10.270 mln::metal::converts_to< T, U > Struct Template Reference

"converts-to" check.

```
#include <converts_to.hh>
```

Inherits `bool_<(sizeof(internal::helper_convert_to_< T, U >::selector(*internal::make_< mln::metal::const_< T >::ret >::ptr(), 0))==sizeof(internal::yes_))>`.

Inherited by `mln::metal::converts_to< T *, U * >`.

10.270.1 Detailed Description

```
template<typename T, typename U>struct mln::metal::converts_to< T, U >
```

"converts-to" check.

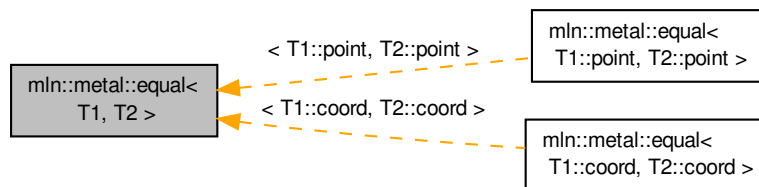
Definition at line 78 of file converts_to.hh.

10.271 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 >`:



10.271.1 Detailed Description

```
template<typename T1, typename T2>struct mln::metal::equal< T1, T2 >
```

Definition of a static 'equal' test.

Check whether type T1 is exactly type T2.

Definition at line 49 of file equal.hh.

10.272 mln::metal::goes_to< T, U > Struct Template Reference

"goes-to" check.


```
#include <goes_to.hh>
```

Inherits eval< mlc_converts_to(T, U), mlc_is(T, U) >.

10.272.1 Detailed Description

```
template<typename T, typename U>struct mln::metal::goes_to< T, U >
```

"goes-to" check.

FIXME: Doc!

Definition at line 53 of file goes_to.hh.

10.273 mln::metal::is< T, U > Struct Template Reference

"is" check.

```
#include <is.hh>
```

Inherits bool_<(sizeof(internal::helper_is_< T, U >::selector(internal::make_< T >::ptr()))==sizeof(internal::yes_ -))>.

10.273.1 Detailed Description

```
template<typename T, typename U>struct mln::metal::is< T, U >
```

"is" check.

Check whether T inherits from U.

Definition at line 64 of file is.hh.

10.274 mln::metal::is_a< T, M > Struct Template Reference

"is_a" check.

```
#include <is_a.hh>
```

Inherits bool_<(sizeof(internal::helper_is_a_< T, M >::selector(internal::make_< T >::ptr()))==sizeof(internal::yes_ -))>.

10.274.1 Detailed Description

```
template<typename T, template< class > class M>struct mln::metal::is_a< T, M >
```

"is_a" check.

Check whether T inherits from *CONCEPT* M.

Definition at line 95 of file is_a.hh.

10.275 mln::metal::is_not< T, U > Struct Template Reference

"is_not" check.

```
#include <is_not.hh>
```

Inherits eval< is< T, U > >.

10.275.1 Detailed Description

```
template<typename T, typename U>struct mln::metal::is_not< T, U >
```

"is_not" check.

FIXME: Doc!

Definition at line 52 of file is_not.hh.

10.276 mln::metal::is_not_a< T, M > Struct Template Reference

"is_not_a" static Boolean expression.

```
#include <is_not_a.hh>
```

Inherits eval< is_a< T, M > >.

10.276.1 Detailed Description

```
template<typename T, template< class > class M>struct mln::metal::is_not_a< T, M >
```

"is_not_a" static Boolean expression.

Definition at line 48 of file is_not_a.hh.

10.277 mln::morpho::attribute::card< I > Class Template Reference

Cardinality accumulator class.

```
#include <card.hh>
```

Inherits mln::accu::internal::base< unsigned, card< I > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- unsigned [to_result](#) () const
Get the value of the accumulator.
- void [init](#) ()
Manipulators.

10.277.1 Detailed Description

`template<typename I>class mln::morpho::attribute::card< I >`

Cardinality accumulator class.

Definition at line 80 of file morpho/attribute/card.hh.

10.277.2 Member Function Documentation

10.277.2.1 `template<typename I> void card< I >::init () [inline]`

Manipulators.

Definition at line 128 of file morpho/attribute/card.hh.

10.277.2.2 `template<typename I> bool card< I >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 197 of file morpho/attribute/card.hh.

10.277.2.3 `void mln::Accumulator< card< I > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.277.2.4 `void mln::Accumulator< card< I > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.277.2.5 `template<typename I> unsigned card< I >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 189 of file morpho/attribute/card.hh.

10.278 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, count_adjacent_vertices< I > >`.

Public Member Functions

- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
Take as initialization the value t.

- void `take_n_times` (unsigned `n`, const `T` &`t`)
Take `n` times the value `t`.
- unsigned `to_result` () const
Get the value of the accumulator.
- void `init` ()
Manipulators.

10.278.1 Detailed Description

`template<typename I>struct mln::morpho::attribute::count_adjacent_vertices< I >`

Count_Adjacent_Vertices accumulator class.

The parameter `I` is the image type on which the accumulator of pixels is built.

Definition at line 83 of file `morpho/attribute/count_adjacent_vertices.hh`.

10.278.2 Member Function Documentation

10.278.2.1 `template<typename I > void count_adjacent_vertices< I >::init () [inline]`

Manipulators.

Definition at line 132 of file `morpho/attribute/count_adjacent_vertices.hh`.

10.278.2.2 `template<typename I > bool count_adjacent_vertices< I >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Definition at line 185 of file `morpho/attribute/count_adjacent_vertices.hh`.

10.278.2.3 `void mln::Accumulator< count_adjacent_vertices< I > >::take_as_init (const T & t) [inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.278.2.4 `void mln::Accumulator< count_adjacent_vertices< I > >::take_n_times (unsigned n, const T & t) [inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.278.2.5 `template<typename I > unsigned count_adjacent_vertices< I >::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 169 of file `morpho/attribute/count_adjacent_vertices.hh`.

10.279 mln::morpho::attribute::height< I > Struct Template Reference

Height accumulator class.

```
#include <height.hh>
```

Inherits mln::accu::internal::base< unsigned, height< I > >.

Public Member Functions

- unsigned [base_level](#) () const
Get base & current level of the accumulator.
- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
Take as initialization the value 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.

10.279.1 Detailed Description

```
template<typename I>struct mln::morpho::attribute::height< I >
```

Height accumulator class.

The parameter `I` is the image type on which the accumulator of pixels is built.

Definition at line 81 of file morpho/attribute/height.hh.

10.279.2 Member Function Documentation

10.279.2.1 `template<typename I> unsigned height< I >::base_level () const` `[inline]`

Get base & current level of the accumulator.

Definition at line 215 of file morpho/attribute/height.hh.

10.279.2.2 `template<typename I> void height< I >::init ()` `[inline]`

Manipulators.

Definition at line 132 of file morpho/attribute/height.hh.

10.279.2.3 `template<typename I> bool height< I >::is_valid () const` `[inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 232 of file morpho/attribute/height.hh.

10.279.2.4 `void mln::Accumulator< height< I > >::take_as_init (const T & t)` `[inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.279.2.5 `void mln::Accumulator< height< I > >::take_n_times (unsigned n, const T & t)` `[inherited]`

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.279.2.6 `template<typename I > unsigned height< I >::to_result () const` `[inline]`

Get the value of the accumulator.

Definition at line 205 of file `morpho/attribute/height.hh`.

10.280 `mln::morpho::attribute::sharpness< I >` Struct Template Reference

Sharpness accumulator class.

`#include <sharpness.hh>`

Inherits `mln::accu::internal::base< double, sharpness< I > >`.

Public Member Functions

- unsigned `area` () const
Give the area of the component.
- unsigned `height` () const
Give the height.
- bool `is_valid` () const
Check whether this accu is able to return a result.
- void `take_as_init` (const T &t)
*Take as initialization the value *t*.*
- void `take_n_times` (unsigned n, const T &t)
*Take *n* times the value *t*.*
- double `to_result` () const
Get the value of the accumulator.
- unsigned `volume` () const
Give the volume of the component.
- void `init` ()
Manipulators.

10.280.1 Detailed Description

`template<typename I> struct mln::morpho::attribute::sharpness< I >`

Sharpness accumulator class.

The parameter *I* is the image type on which the accumulator of pixels is built.

Definition at line 81 of file `sharpness.hh`.

10.280.2 Member Function Documentation

10.280.2.1 `template<typename I> unsigned sharpness<I>::area () const [inline]`

Give the area of the component.

Definition at line 191 of file sharpness.hh.

10.280.2.2 `template<typename I> unsigned sharpness<I>::height () const [inline]`

Give the height.

Definition at line 207 of file sharpness.hh.

10.280.2.3 `template<typename I> void sharpness<I>::init () [inline]`

Manipulators.

Definition at line 135 of file sharpness.hh.

10.280.2.4 `template<typename I> bool sharpness<I>::is_valid () const [inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 215 of file sharpness.hh.

10.280.2.5 `void mln::Accumulator< sharpness<I> >::take_as_init (const T & t) [inherited]`

Take as initialization the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.280.2.6 `void mln::Accumulator< sharpness<I> >::take_n_times (unsigned n, const T & t) [inherited]`

Take n times the value t .

Dev note: this is a final method; override if needed by `take_as_init_` (ending with '_').

10.280.2.7 `template<typename I> double sharpness<I>::to_result () const [inline]`

Get the value of the accumulator.

Definition at line 176 of file sharpness.hh.

10.280.2.8 `template<typename I> unsigned sharpness<I>::volume () const [inline]`

Give the volume of the component.

Definition at line 199 of file sharpness.hh.

10.281 mln::morpho::attribute::sum< I, S > Class Template Reference

Suminality accumulator class.

```
#include <sum.hh>
```

Inherits mln::accu::internal::base< S, sum< I, S > >.

Public Member Functions

- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [set_value](#) (const argument &v)
Set the return value of the accumulator.
- void [take_as_init](#) (const T &t)
Take as initialization the value t.
- void [take_n_times](#) (unsigned n, const T &t)
Take n times the value t.
- S [to_result](#) () const
Get the value of the accumulator.
- void [untake](#) (const argument &v)
Untake a value from the accumulator.
- void [init](#) ()
Manipulators.

10.281.1 Detailed Description

```
template<typename I, typename S = typename mln::value::props< typename I ::value >::sum>class mln::morpho::attribute-  
::sum< I, S >
```

Suminality accumulator class.

Definition at line 80 of file morpho/attribute/sum.hh.

10.281.2 Member Function Documentation

10.281.2.1 `template<typename I , typename S > void sum< I, S >::init () [inline]`

Manipulators.

Definition at line 137 of file morpho/attribute/sum.hh.

References mln::literal::zero.

10.281.2.2 `template<typename I , typename S > bool sum< I, S >::is_valid () const [inline]`

Check whether this accu is able to return a result.

Return always true.

Definition at line 229 of file morpho/attribute/sum.hh.

10.281.2.3 `template<typename I , typename S > void sum< I, S >::set_value (const argument & v) [inline]`

Set the return value of the accumulator.

Definition at line 205 of file morpho/attribute/sum.hh.

10.281.2.4 void mln::Accumulator< sum< I, S > >::take_as_init (const T & t) [inherited]

Take as initialization the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.281.2.5 void mln::Accumulator< sum< I, S > >::take_n_times (unsigned n, const T & t) [inherited]

Take *n* times the value *t*.

Dev note: this is a final method; override if needed by take_as_init_ (ending with '_').

10.281.2.6 template<typename I, typename S> S sum< I, S >::to_result () const [inline]

Get the value of the accumulator.

Definition at line 221 of file morpho/attribute/sum.hh.

10.281.2.7 template<typename I, typename S> void sum< I, S >::untake (const argument & v) [inline]

Untake a value from the accumulator.

Definition at line 189 of file morpho/attribute/sum.hh.

10.282 mln::morpho::attribute::volume< I > Struct Template Reference

Volume accumulator class.

```
#include <volume.hh>
```

Inherits mln::accu::internal::base< unsigned, volume< I > >.

Public Member Functions

- unsigned [area](#) () const
Give the area.
- bool [is_valid](#) () const
Check whether this accu is able to return a result.
- void [take_as_init](#) (const T &t)
*Take as initialization the value *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.

10.282.1 Detailed Description

```
template<typename I>struct mln::morpho::attribute::volume< I >
```

Volume accumulator class.

The parameter `I` is the image type on which the accumulator of pixels is built.

Definition at line 79 of file `morpho/attribute/volume.hh`.

10.282.2 Member Function Documentation

10.282.2.1 `template<typename I> unsigned volume<I>::area () const` `[inline]`

Give the area.

Definition at line 203 of file `morpho/attribute/volume.hh`.

10.282.2.2 `template<typename I> void volume<I>::init ()` `[inline]`

Manipulators.

Definition at line 131 of file `morpho/attribute/volume.hh`.

10.282.2.3 `template<typename I> bool volume<I>::is_valid () const` `[inline]`

Check whether this accu is able to return a result.

Always true here.

Definition at line 211 of file `morpho/attribute/volume.hh`.

10.282.2.4 `void mln::Accumulator< volume<I> >::take_as_init (const T & t)` `[inherited]`

Take as initialization the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.282.2.5 `void mln::Accumulator< volume<I> >::take_n_times (unsigned n, const T & t)` `[inherited]`

Take `n` times the value `t`.

Dev note: this is a final method; override if needed by `take_as_init_` (ending with `'_'`).

10.282.2.6 `template<typename I> unsigned volume<I>::to_result () const` `[inline]`

Get the value of the accumulator.

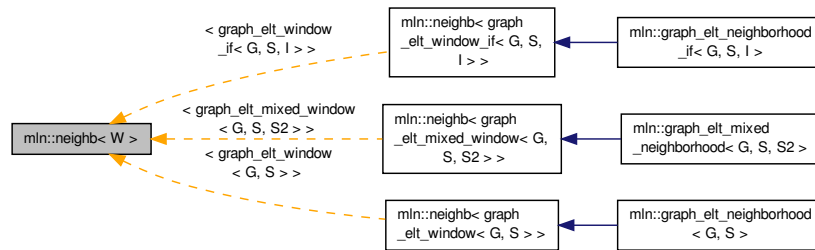
Definition at line 195 of file `morpho/attribute/volume.hh`.

10.283 mln::neighb< W > Class Template Reference

Adapter class from window to neighborhood.

```
#include <neighb.hh>
```

Inheritance diagram for mln::neighb< W >:



Public Types

- typedef `neighb_bkd_niter< W >` `bkd_niter`
Backward site iterator associated type.
- typedef `neighb_fwd_niter< W >` `fwd_niter`
Forward site iterator associated type.
- typedef `fwd_niter` `niter`
Site iterator associated type.

Public Member Functions

- `neighb()`
Constructor without argument.
- `neighb(const W &win)`
Constructor from a window `win`.

10.283.1 Detailed Description

```
template<typename W>class mln::neighb< W >
```

Adapter class from window to neighborhood.

Definition at line 76 of file `mln/core/neighb.hh`.

10.283.2 Member Typedef Documentation

10.283.2.1 `template<typename W> typedef neighb_bkd_niter<W> mln::neighb< W >::bkd_niter`

Backward site iterator associated type.

Definition at line 87 of file `mln/core/neighb.hh`.

10.283.2.2 `template<typename W> typedef neighb_fwd_niter<W> mln::neighb< W >::fwd_niter`

Forward site iterator associated type.

Definition at line 84 of file `mln/core/neighb.hh`.

10.283.2.3 `template<typename W> typedef fwd_niter mln::neighb< W >::niter`

[Site](#) iterator associated type.

Definition at line 90 of file `mln/core/neighb.hh`.

10.283.3 Constructor & Destructor Documentation

10.283.3.1 `template<typename W> neighb< W >::neighb () [inline]`

Constructor without argument.

Definition at line 150 of file `mln/core/neighb.hh`.

10.283.3.2 `template<typename W> neighb< W >::neighb (const W & win) [inline]`

Constructor from a window `win`.

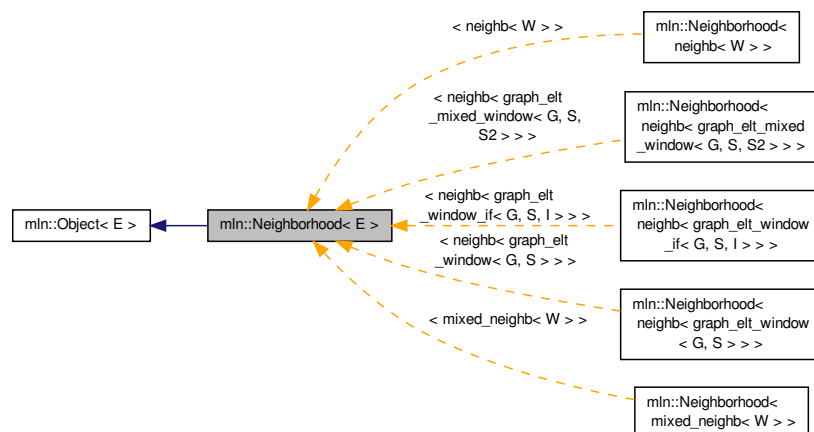
Definition at line 156 of file `mln/core/neighb.hh`.

10.284 mln::Neighborhood< E > Struct Template Reference

Base class for implementation classes that are neighborhoods.

`#include <neighborhood.hh>`

Inheritance diagram for `mln::Neighborhood< E >`:



10.284.1 Detailed Description

`template<typename E> struct mln::Neighborhood< E >`

Base class for implementation classes that are neighborhoods.

See Also

[mln::doc::Neighborhood](#) for a complete documentation of this class contents.

Definition at line 66 of file core/concept/neighborhood.hh.

10.285 mln::Neighborhood< void > Struct Template Reference

[Neighborhood](#) category flag type.

```
#include <neighborhood.hh>
```

10.285.1 Detailed Description

```
template<> struct mln::Neighborhood< void >
```

[Neighborhood](#) category flag type.

Definition at line 54 of file core/concept/neighborhood.hh.

10.286 mln::Object< E > Struct Template Reference

Base class for almost every class defined in Milena.

```
#include <object.hh>
```

Inherited by [mln::algebra::internal::mat_base< n, m, T, E >](#), [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::lterator< E >](#), [mln::Literal< E >](#), [mln::Mesh< E >](#), [mln::Meta_Accumulator< E >](#), [mln::Meta_Function< E >](#), [mln::Neighborhood< E >](#), [mln::Point_Site< E >](#), [mln::Proxy< E >](#), [mln::Site< E >](#), [mln::Site_Set< E >](#), [mln::Value< E >](#), [mln::value::HSL< E >](#), [mln::Value_Set< E >](#), [mln::Weighted_Window< E >](#), and [mln::Window< E >](#).

10.286.1 Detailed Description

```
template<typename E> struct mln::Object< E >
```

Base class for almost every class defined in Milena.

The parameter *E* is the exact type.

Definition at line 171 of file object.hh.

10.287 mln::p2p_image< I, F > Struct Template Reference

FIXME: Doc!

```
#include <p2p_image.hh>
```

Inherits [mln::internal::image_domain_morpher< I, I::domain_t, p2p_image< I, F > >](#).

Public Types

- typedef [p2p_image< tag::image_< I >, tag::function_< F > >](#) [skeleton](#)
Skeleton.

Public Member Functions

- `const I::domain_t & domain () const`
Give the definition domain.
- `const F & fun () const`
Give the p2p function.
- `I::rvalue operator() (const typename I::psite &p) const`
*Read-only access to the image value located at point *p*.*
- `internal::morpher_lvalue_< I >::ret operator() (const typename I::psite &p)`
*Read-write access to the image value located at point *p*.*
- `p2p_image ()`
Constructor without argument.
- `p2p_image (I &ima, const F &f)`
*Constructor from an image *ima* and a predicate *f*.*

10.287.1 Detailed Description

```
template<typename I, typename F>struct mln::p2p_image< I, F >
```

FIXME: Doc!

Definition at line 90 of file p2p_image.hh.

10.287.2 Member Typedef Documentation

10.287.2.1 `template<typename I, typename F> typedef p2p_image< tag::image_<I>, tag::function_<F> > mln::p2p_image< I, F >::skeleton`

Skeleton.

Definition at line 95 of file p2p_image.hh.

10.287.3 Constructor & Destructor Documentation

10.287.3.1 `template<typename I , typename F > p2p_image< I, F >::p2p_image () [inline]`

Constructor without argument.

Definition at line 178 of file p2p_image.hh.

10.287.3.2 `template<typename I , typename F > p2p_image< I, F >::p2p_image (I & ima, const F & f) [inline]`

Constructor from an image *ima* and a predicate *f*.

Definition at line 184 of file p2p_image.hh.

10.287.4 Member Function Documentation

10.287.4.1 `template<typename I , typename F > const I::domain_t & p2p_image< I, F >::domain () const [inline]`

Give the definition domain.

Definition at line 201 of file p2p_image.hh.

10.287.4.2 `template<typename I , typename F > const F & p2p_image< I, F >::fun () const` `[inline]`

Give the p2p function.

Definition at line 210 of file p2p_image.hh.

10.287.4.3 `template<typename I , typename F > l::rvalue p2p_image< I, F >::operator() (const typename l::psite & p) const` `[inline]`

Read-only access to the image value located at point p.

Definition at line 219 of file p2p_image.hh.

10.287.4.4 `template<typename I , typename F > internal::morpher_lvalue_< I >::ret p2p_image< I, F >::operator() (const typename l::psite & p)` `[inline]`

Read-write access to the image value located at point p.

Definition at line 229 of file p2p_image.hh.

10.288 mln::p_array< P > Class Template Reference

Multi-set of sites.

`#include <p_array.hh>`

Inherits `mln::internal::site_set_base_< P, p_array< P > >`.

Public Types

- typedef `p_indexed_bkd_piter`
`< self_ > bkd_piter`
Backward [Site_Iterator](#) associated type.
- typedef P `element`
Element associated type.
- typedef `p_indexed_fwd_piter`
`< self_ > fwd_piter`
Forward [Site_Iterator](#) associated type.
- typedef P `i_element`
Insertion element associated type.
- typedef `fwd_piter` `piter`
[Site_Iterator](#) associated type.
- typedef `p_indexed_psite``< self_ > psite`
Psite associated type.

Public Member Functions

- `p_array< P > & append` (const P &p)
*Append a point *p*.*
- `p_array< P > & append` (const `p_array< P >` &other)
*Append an array *other* of points.*
- void `change` (const `psite` &p, const P &new_p)
*Change site *p* into *new_p*.*

- void [clear](#) ()
Clear this set.
- bool [has](#) (const [psite](#) &p) const
Test if p belongs to this site set.
- bool [has](#) (const util::index &i) const
Test if index i belongs to this site set.
- void [insert](#) (const P &p)
Insert a 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\[\]](#) (unsigned i) const
Return the i -th site (constant).
- P & [operator\[\]](#) (unsigned i)
Return the i -th site (mutable).
- const P & [operator\[\]](#) (const util::index &i) const
Return the i -th element.
- [p_array](#) ()
Constructor.
- [p_array](#) (const std::vector< P > &vect)
Constructor from a vector $vect$.
- void [reserve](#) (size_type n)
Reserve n cells.
- void [resize](#) (size_t size)
Update the size of this array.
- const std::vector< P > & [std_vector](#) () const
Return the corresponding std::vector of points.

10.288.1 Detailed Description

template<typename P>class mln::p_array< P >

Multi-set of sites.

[Site](#) set class based on std::vector.

Definition at line 84 of file p_array.hh.

10.288.2 Member Typedef Documentation

10.288.2.1 template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_array< P >::bkd_piter

Backward [Site_iterator](#) associated type.

Definition at line 100 of file p_array.hh.

10.288.2.2 template<typename P> typedef P mln::p_array< P >::element

Element associated type.

Definition at line 91 of file p_array.hh.

10.288.2.3 `template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_array< P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 97 of file p_array.hh.

10.288.2.4 `template<typename P> typedef P mln::p_array< P >::i_element`

Insertion element associated type.

Definition at line 141 of file p_array.hh.

10.288.2.5 `template<typename P> typedef fwd_piter mln::p_array< P >::piter`

[Site_Iterator](#) associated type.

Definition at line 103 of file p_array.hh.

10.288.2.6 `template<typename P> typedef p_indexed_psite<self_> mln::p_array< P >::psite`

Psite associated type.

Definition at line 94 of file p_array.hh.

10.288.3 Constructor & Destructor Documentation

10.288.3.1 `template<typename P> p_array< P >::p_array () [inline]`

Constructor.

Definition at line 340 of file p_array.hh.

10.288.3.2 `template<typename P> p_array< P >::p_array (const std::vector< P > & vect) [inline]`

Constructor from a vector `vect`.

Definition at line 346 of file p_array.hh.

10.288.4 Member Function Documentation

10.288.4.1 `template<typename P> p_array< P > & p_array< P >::append (const P & p) [inline]`

Append a point `p`.

Definition at line 408 of file p_array.hh.

Referenced by `mln::convert::to_p_array()`.

10.288.4.2 `template<typename P> p_array< P > & p_array< P >::append (const p_array< P > & other) [inline]`

Append an array `other` of points.

Definition at line 425 of file p_array.hh.

References `mln::p_array< P >::std_vector()`.

10.288.4.3 `template<typename P> void p_array<P>::change (const psite & p, const P & new_p)` `[inline]`

Change site `p` into `new_p`.

Definition at line 472 of file `p_array.hh`.

10.288.4.4 `template<typename P> void p_array<P>::clear ()` `[inline]`

Clear this set.

Definition at line 436 of file `p_array.hh`.

10.288.4.5 `template<typename P> bool p_array<P>::has (const psite & p) const` `[inline]`

Test is `p` belongs to this site set.

Definition at line 362 of file `p_array.hh`.

10.288.4.6 `template<typename P> bool p_array<P>::has (const util::index & i) const` `[inline]`

Test is index `i` belongs to this site set.

Definition at line 375 of file `p_array.hh`.

10.288.4.7 `template<typename P> void p_array<P>::insert (const P & p)` `[inline]`

Insert a point `p` (equivalent as 'append').

Definition at line 417 of file `p_array.hh`.

10.288.4.8 `template<typename P> bool p_array<P>::is_valid () const` `[inline]`

Test this set validity so returns always true.

Definition at line 383 of file `p_array.hh`.

10.288.4.9 `template<typename P> std::size_t p_array<P>::memory_size () const` `[inline]`

Return the size of this site set in memory.

Definition at line 481 of file `p_array.hh`.

10.288.4.10 `template<typename P> unsigned p_array<P>::nsites () const` `[inline]`

Give the number of sites.

Definition at line 400 of file `p_array.hh`.

Referenced by `mln::registration::get_rot()`.

10.288.4.11 `template<typename P> const P & p_array<P>::operator[] (unsigned i) const` `[inline]`

Return the `i-th` site (constant).

Definition at line 454 of file `p_array.hh`.

10.288.4.12 `template<typename P> P & p_array< P >::operator[] (unsigned i) [inline]`

Return the *i*-th site (mutable).

Definition at line 463 of file `p_array.hh`.

10.288.4.13 `template<typename P> const P & p_array< P >::operator[] (const util::index & i) const [inline]`

Return the *i*-th element.

Definition at line 391 of file `p_array.hh`.

10.288.4.14 `template<typename P> void p_array< P >::reserve (size_type n) [inline]`

Reserve *n* cells.

Definition at line 354 of file `p_array.hh`.

Referenced by `mln::convert::to_p_array()`.

10.288.4.15 `template<typename P> void p_array< P >::resize (size_t size) [inline]`

Update the size of this array.

Definition at line 445 of file `p_array.hh`.

10.288.4.16 `template<typename P> const std::vector< P > & p_array< P >::std_vector () const [inline]`

Return the corresponding `std::vector` of points.

Definition at line 489 of file `p_array.hh`.

Referenced by `mln::p_array< P >::append()`.

10.289 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, p_centered< W > >`, `mlc_is_aW`, and `check_t`.

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 ()`
Constructor without argument.
- `p_centered (const W &win, const typename W::psite &c)`
Constructor from a window win and a center c .
- `const W & window () const`
Give the window this site set is defined upon.

10.289.1 Detailed Description

`template<typename W>class mln::p_centered< W >`

[Site](#) set corresponding to a window centered on a site.

Definition at line 77 of file `p_centered.hh`.

10.289.2 Member Typedef Documentation

10.289.2.1 `template<typename W> typedef p_centered_piter<W> mln::p_centered< W >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 97 of file `p_centered.hh`.

10.289.2.2 `template<typename W> typedef psite mln::p_centered< W >::element`

Element associated type.

Definition at line 90 of file `p_centered.hh`.

10.289.2.3 `template<typename W> typedef p_centered_piter<W> mln::p_centered< W >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 94 of file `p_centered.hh`.

10.289.2.4 `template<typename W> typedef fwd_piter mln::p_centered< W >::piter`

[Site_Iterator](#) associated type.

Definition at line 100 of file `p_centered.hh`.

10.289.2.5 `template<typename W> typedef W ::psite mln::p_centered< W >::psite`

Psite associated type.

Definition at line 83 of file p_centered.hh.

10.289.2.6 `template<typename W> typedef W ::site mln::p_centered< W >::site`

Site associated type.

Definition at line 86 of file p_centered.hh.

10.289.3 Constructor & Destructor Documentation

10.289.3.1 `template<typename W> p_centered< W >::p_centered () [inline]`

Constructor without argument.

Definition at line 182 of file p_centered.hh.

10.289.3.2 `template<typename W> p_centered< W >::p_centered (const W & win, const typename W::psite & c) [inline]`

Constructor from a window `win` and a center `c`.

Definition at line 188 of file p_centered.hh.

References `mln::p_centered< W >::is_valid()`.

10.289.4 Member Function Documentation

10.289.4.1 `template<typename W> const W::psite & p_centered< W >::center () const [inline]`

Give the center of this site set.

Definition at line 216 of file p_centered.hh.

10.289.4.2 `template<typename W> template<typename P> bool p_centered< W >::has (const P & p) const [inline]`

Test if `p` belongs to the box.

Definition at line 199 of file p_centered.hh.

10.289.4.3 `template<typename W> bool p_centered< W >::is_valid () const [inline]`

Test if this site set is initialized.

Definition at line 175 of file p_centered.hh.

Referenced by `mln::p_centered< W >::p_centered()`.

10.289.4.4 `template<typename W> std::size_t p_centered< W >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 208 of file p_centered.hh.

10.289.4.5 `template<typename W> const W & p_centered< W>::window () const` `[inline]`

Give the window this site set is defined upon.

Definition at line 224 of file `p_centered.hh`.

10.290 `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< complex_psite< D, G>, p_complex< D, G>>`.

Public Types

- `typedef super_::site element`
Associated types.
- `typedef complex_psite< D, G> psite`
Point_Site associated type.
- `typedef p_complex_fwd_piter_< D, G> fwd_piter`
Forward Site_Iterator associated type.
- `typedef p_complex_bkd_piter_< D, G> bkd_piter`
Backward Site_Iterator associated type.
- `typedef fwd_piter piter`
Site_Iterator associated type.

Public Member Functions

- `bool has (const psite &p) const`
Does this site set has p?
- `bool is_valid () const`
Is this site set valid?
- `unsigned nfaces () const`
Return the number of faces in the complex.
- `unsigned nfaces_of_dim (unsigned n) const`
Return the number of n-faces in the complex.
- `unsigned nsites () const`
Return The number of sites of the set, i.e., the number of faces.
- `p_complex (const topo::complex< D> &cplx, const G &geom)`
Construct a complex psite set from a complex.
- `topo::complex< D> & cplx () const`
Accessors.
- `topo::complex< D> & cplx ()`
Return the complex associated to the p_complex domain (mutable version).
- `const G & geom () const`
Return the geometry of the complex.

10.290.1 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

<i>D</i>	The dimension of the complex.
<i>G</i>	A function object type, associating localization information (geometry) to each face of the complex.

See Also

[mln::geom::complex_geometry](#). A complex [psite set](#) based on the N-faces of a complex.

Definition at line 116 of file p_complex.hh.

10.290.2 Member Typedef Documentation

10.290.2.1 template<unsigned D, typename G> typedef p_complex_bkd_piter_<D, G> mln::p_complex< D, G >::bkd_piter

Backward [Site_Iterator](#) associated type.

Definition at line 141 of file p_complex.hh.

10.290.2.2 template<unsigned D, typename G> typedef super_::site mln::p_complex< D, G >::element

Associated types.

Element associated type.

Definition at line 132 of file p_complex.hh.

10.290.2.3 template<unsigned D, typename G> typedef p_complex_fwd_piter_<D, G> mln::p_complex< D, G >::fwd_piter

Forward [Site_Iterator](#) associated type.

Definition at line 138 of file p_complex.hh.

10.290.2.4 template<unsigned D, typename G> typedef fwd_piter mln::p_complex< D, G >::piter

[Site_Iterator](#) associated type.

Definition at line 144 of file p_complex.hh.

10.290.2.5 template<unsigned D, typename G> typedef complex_psite<D, G> mln::p_complex< D, G >::psite

Point_Site associated type.

Definition at line 135 of file p_complex.hh.

10.290.3 Constructor & Destructor Documentation

10.290.3.1 `template<unsigned D, typename G > p_complex< D, G >::p_complex (const topo::complex< D > & cplx, const G & geom) [inline]`

Construct a complex psite set from a complex.

Parameters

<i>cplx</i>	The complex upon which the complex psite set is built.
<i>geom</i>	FIXME

Definition at line 231 of file p_complex.hh.

10.290.4 Member Function Documentation

10.290.4.1 `template<unsigned D, typename G > topo::complex< D > & p_complex< D, G >::cplx () const`

Accessors.

Return the complex associated to the [p_complex](#) domain (const version)

Definition at line 293 of file p_complex.hh.

Referenced by `mln::complex_psite< D, G >::change_target()`, `mln::complex_psite< D, G >::complex_psite()`, and `mln::operator==()`.

10.290.4.2 `template<unsigned D, typename G > topo::complex< D > & p_complex< D, G >::cplx ()`

Return the complex associated to the [p_complex](#) domain (mutable version).

Definition at line 301 of file p_complex.hh.

10.290.4.3 `template<unsigned D, typename G > const G & p_complex< D, G >::geom () const`

Return the geometry of the complex.

Definition at line 309 of file p_complex.hh.

10.290.4.4 `template<unsigned D, typename G > bool p_complex< D, G >::has (const psite & p) const [inline]`

Does this site set has *p*?

Definition at line 271 of file p_complex.hh.

References `mln::complex_psite< D, G >::is_valid()`, and `mln::complex_psite< D, G >::site_set()`.

10.290.4.5 `template<unsigned D, typename G > bool p_complex< D, G >::is_valid () const [inline]`

Is this site set valid?

Definition at line 263 of file p_complex.hh.

10.290.4.6 `template<unsigned D, typename G > unsigned p_complex< D, G >::nfaces () const [inline]`

Return the number of faces in the complex.

Definition at line 247 of file p_complex.hh.

10.290.4.7 `template<unsigned D, typename G > unsigned p_complex< D, G >::nfaces_of_dim (unsigned n) const`
`[inline]`

Return the number of *n*-faces in the complex.

Definition at line 255 of file p_complex.hh.

10.290.4.8 `template<unsigned D, typename G > unsigned p_complex< D, G >::nsites () const` `[inline]`

Return The number of sites of the set, i.e., the number of *faces*.

(Required by the [mln::Site_Set](#) concept, since the property trait::site_set::nsites::known of this site set is set to 'known'.)

Definition at line 239 of file p_complex.hh.

10.291 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, p_edges< G, F > >`.

Public Types

- `typedef util::edge< G > edge`
Type of graph edge.
- `typedef F fun_t`
Function associated type.
- `typedef util::edge< G > graph_element`
Type of graph element this site set focuses on.
- `typedef G graph_t`
Graph associated type.
- `typedef super_::site element`
Associated types.
- `typedef p_edges_psite< G, F > psite`
Point_Site associated type.
- `typedef p_graph_piter< self_, mln_edge_fwd_iter(G) > fwd_piter`
Forward Site_Iterator associated type.
- `typedef p_graph_piter< self_, mln_edge_bkd_iter(G) > bkd_piter`
Backward Site_Iterator associated type.
- `typedef fwd_piter piter`
Site_Iterator associated type.

Public Member Functions

- `bool has (const psite &p) const`
Does this site set has site p?
- `template<typename G2 > bool has (const util::edge< G2 > &e) const`

- Does this site set has edge e?*

 - void `invalidate` ()

Invalidate this site set.
- bool `is_valid` () const

Is this site set valid?
- std::size_t `memory_size` () const

Does this site set has vertex_id? FIXME: causes ambiguities while calling has(mln::neighb_fwd_niter<>); bool has(unsigned vertex_id) const;.
- unsigned `nedges` () const

Return The number of edges in the graph.
- unsigned `nsites` () const

Return The number of points (sites) of the set, i.e., the number of edges.
- `p_edges` ()

Constructors
Default constructor.
- `p_edges` (const `Graph`< G > &gr)

Construct a graph edge psite set from a graph.
- `p_edges` (const `Graph`< G > &gr, const `Function`< F > &f)

Construct a graph edge psite set from a graph and a function.
- template<typename F2 >
`p_edges` (const `Graph`< G > &gr, const `Function`< F2 > &f)

Construct a graph edge psite set from a graph and a function.
- const G & `graph` () const

Accessors.
- const F & `function` () const

Return the mapping function.

10.291.1 Detailed Description

```
template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> class mln::p_edges< G, F >
```

`Site` set mapping graph edges and image sites.

Definition at line 71 of file `p_edges.hh`.

10.291.2 Member Typedef Documentation

10.291.2.1 `template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> typedef p_graph_piter< self_, mln::edge_bkd_iter(G) > mln::p_edges< G, F >::bkd_piter`

Backward `Site_iterator` associated type.

Definition at line 129 of file `p_edges.hh`.

10.291.2.2 `template<typename G, typename F = util::internal::id2element<G,util::edge<G> >> typedef util::edge<G> mln::p_edges< G, F >::edge`

Type of graph edge.

Definition at line 87 of file `p_edges.hh`.

10.291.2.3 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>> typedef super_::site
mln::p_edges< G, F >::element`

Associated types.

Element associated type.

Definition at line 120 of file p_edges.hh.

10.291.2.4 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>> typedef F mln::p_edges<
G, F >::fun_t`

[Function](#) associated type.

Definition at line 84 of file p_edges.hh.

10.291.2.5 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>> typedef p_graph_piter<
self_, mln::edge_fwd_iter(G) > mln::p_edges< G, F >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 126 of file p_edges.hh.

10.291.2.6 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>> typedef util::edge<G>
mln::p_edges< G, F >::graph_element`

Type of graph element this site set focuses on.

Definition at line 90 of file p_edges.hh.

10.291.2.7 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>> typedef G mln::p_edges<
G, F >::graph_t`

[Graph](#) associated type.

Definition at line 81 of file p_edges.hh.

10.291.2.8 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>> typedef fwd_piter
mln::p_edges< G, F >::piter`

[Site_Iterator](#) associated type.

Definition at line 132 of file p_edges.hh.

10.291.2.9 `template<typename G, typename F = util::internal::id2element<G,util::edge<G>>> typedef p_edges_psite<G,
F> mln::p_edges< G, F >::psite`

Point_Site associated type.

Definition at line 123 of file p_edges.hh.

10.291.3 Constructor & Destructor Documentation

10.291.3.1 `template<typename G , typename F > p_edges< G, F >::p_edges () [inline]`

Constructors

Default constructor.

Definition at line 204 of file p_edges.hh.

10.291.3.2 `template<typename G , typename F > p_edges< G, F >::p_edges (const Graph< G > & gr) [inline]`

Construct a graph edge psite set from a graph.

Parameters

<i>gr</i>	The graph upon which the graph edge psite set is built.
-----------	---

Definition at line 210 of file p_edges.hh.

References `mln::p_edges< G, F >::is_valid()`.

10.291.3.3 `template<typename G , typename F > 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

<i>gr</i>	The graph upon which the graph edge psite set is built.
<i>f</i>	the function mapping edges and sites.

Definition at line 222 of file p_edges.hh.

References `mln::p_edges< G, F >::is_valid()`.

10.291.3.4 `template<typename G , typename F > template<typename F2 > 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

<i>gr</i>	The graph upon which the graph edge psite set is built.
<i>f</i>	the function mapping edges and sites. It must be convertible towards the function type <code>F</code> .

Definition at line 232 of file p_edges.hh.

References `mln::p_edges< G, F >::is_valid()`.

10.291.4 Member Function Documentation

10.291.4.1 `template<typename G , typename F > const F & p_edges< G, F >::function () const [inline]`

Return the mapping function.

Definition at line 325 of file p_edges.hh.

10.291.4.2 `template<typename G , typename F > const G & p_edges< G, F >::graph () const [inline]`

Accessors.

Return the graph associated to this site set

Definition at line 316 of file p_edges.hh.

References mln::p_edges< G, F >::is_valid().

Referenced by mln::operator==().

10.291.4.3 `template<typename G , typename F > bool p_edges< G, F >::has (const psite & p) const` `[inline]`

Does this site set has site *p*?

Definition at line 277 of file p_edges.hh.

References mln::p_edges< G, F >::is_valid().

10.291.4.4 `template<typename G , typename F > template<typename G2 > bool p_edges< G, F >::has (const util::edge< G2 > & e) const` `[inline]`

Does this site set has edge *e*?

Definition at line 287 of file p_edges.hh.

References mln::util::edge< G >::graph(), mln::util::edge< G >::is_valid(), and mln::p_edges< G, F >::is_valid().

10.291.4.5 `template<typename G , typename F > void p_edges< G, F >::invalidate ()` `[inline]`

Invalidate this site set.

Definition at line 269 of file p_edges.hh.

10.291.4.6 `template<typename G , typename F > bool p_edges< G, F >::is_valid () const` `[inline]`

Is this site set valid?

Definition at line 261 of file p_edges.hh.

Referenced by mln::p_edges< G, F >::graph(), mln::p_edges< G, F >::has(), and mln::p_edges< G, F >::p_edges().

10.291.4.7 `template<typename G , typename F > std::size_t p_edges< G, F >::memory_size () const` `[inline]`

Does this site set has *vertex_id*? **FIXME:** causes ambiguities while calling has(mln::neighb_fwd_niter<>); bool has(unsigned vertex_id) const;.

Definition at line 306 of file p_edges.hh.

10.291.4.8 `template<typename G , typename F > unsigned p_edges< G, F >::nedges () const` `[inline]`

Return The number of edges in the graph.

Definition at line 253 of file p_edges.hh.

Referenced by mln::p_edges< G, F >::nsites().

10.291.4.9 `template<typename G , typename F > unsigned p_edges< G, F >::nsites () const` `[inline]`

Return The number of points (sites) of the set, i.e., the number of *edges*.

Definition at line 245 of file p_edges.hh.

References mln::p_edges< G, F >::nedges().

10.292 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_< faces_psite< N, D, P >, p_faces< N, D, P > >.

Public Types

- typedef super_::site [element](#)
Associated types.
- typedef faces_psite< N, D, P > [psite](#)
Point_Site associated type.
- typedef p_faces_fwd_piter_< N, D, P > [fwd_piter](#)
Forward Site_Iterator associated type.
- typedef p_faces_bkd_piter_< N, D, P > [bkd_piter](#)
Backward Site_Iterator associated type.
- typedef [fwd_piter](#) piter
Site_Iterator associated type.

Public Member Functions

- bool [is_valid](#) () const
Is this site set valid?
- unsigned [nfaces](#) () const
Return The number of faces in the complex.
- unsigned [nsites](#) () const
Return The number of sites of the set, i.e., the number of faces.
- [p_faces](#) (const [topo::complex](#)< D > &cplx)
Construct a faces psite set from an mln::complex.
- [p_faces](#) (const [p_complex](#)< D, P > &pc)
Construct a faces psite set from an mln::p_complex.
- [topo::complex](#)< D > & [cplx](#) () const
Accessors.
- [topo::complex](#)< D > & [cplx](#) ()
Return the complex associated to the p_faces domain (mutable version).

10.292.1 Detailed Description

```
template<unsigned N, unsigned D, typename P>struct mln::p_faces< N, D, P >
```

A complex psite set based on a the N-faces of a complex of dimension D (a D-complex).

Definition at line 78 of file p_faces.hh.

10.292.2 Member Typedef Documentation

10.292.2.1 `template<unsigned N, unsigned D, typename P> typedef p_faces_bkd_piter_<N, D, P> mln::p_faces< N, D, P >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 111 of file p_faces.hh.

10.292.2.2 `template<unsigned N, unsigned D, typename P> typedef super_::site mln::p_faces< N, D, P >::element`

Associated types.

Element associated type.

Definition at line 100 of file p_faces.hh.

10.292.2.3 `template<unsigned N, unsigned D, typename P> typedef p_faces_fwd_piter_<N, D, P> mln::p_faces< N, D, P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 107 of file p_faces.hh.

10.292.2.4 `template<unsigned N, unsigned D, typename P> typedef fwd_piter mln::p_faces< N, D, P >::piter`

[Site_Iterator](#) associated type.

Definition at line 114 of file p_faces.hh.

10.292.2.5 `template<unsigned N, unsigned D, typename P> typedef faces_psite<N, D, P> mln::p_faces< N, D, P >::psite`

Point_Site associated type.

Definition at line 103 of file p_faces.hh.

10.292.3 Constructor & Destructor Documentation

10.292.3.1 `template<unsigned N, unsigned D, typename P > p_faces< N, D, P >::p_faces (const topo::complex< D > & cplx) [inline]`

Construct a faces psite set from an mln::complex.

Parameters

<i>cplx</i>	The complex upon which the complex psite set is built.
-------------	--

Definition at line 193 of file p_faces.hh.

10.292.3.2 `template<unsigned N, unsigned D, typename P > p_faces< N, D, P >::p_faces (const p_complex< D, P > & pc) [inline]`

Construct a faces psite set from an [mln::p_complex](#).

Parameters

<i>pc</i>	The complex upon which the complex psite set is built.
-----------	--

Definition at line 202 of file `p_faces.hh`.

10.292.4 Member Function Documentation

10.292.4.1 `template<unsigned N, unsigned D, typename P> topo::complex< D> & p_faces< N, D, P>::cplx () const`

Accessors.

Return the complex associated to the `p_faces` domain (const version).

Definition at line 258 of file `p_faces.hh`.

Referenced by `mln::operator==()`.

10.292.4.2 `template<unsigned N, unsigned D, typename P> topo::complex< D> & p_faces< N, D, P>::cplx ()`

Return the complex associated to the `p_faces` domain (mutable version).

Definition at line 266 of file `p_faces.hh`.

10.292.4.3 `template<unsigned N, unsigned D, typename P> bool p_faces< N, D, P>::is_valid () const [inline]`

Is this site set valid?

Definition at line 228 of file `p_faces.hh`.

10.292.4.4 `template<unsigned N, unsigned D, typename P> unsigned p_faces< N, D, P>::nfaces () const [inline]`

Return The number of faces in the complex.

Definition at line 220 of file `p_faces.hh`.

10.292.4.5 `template<unsigned N, unsigned D, typename P> unsigned p_faces< N, D, P>::nsites () const [inline]`

Return The number of sites of the set, i.e., the number of *faces*.

(Required by the `mln::Site_Set` concept, since the property trait `site_set::nsites::known` of this site set is set to 'known'.)

Definition at line 212 of file `p_faces.hh`.

10.293 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, 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.

10.293.1 Detailed Description

template<typename S, typename I>class mln::p_graph_piter< S, I >

Generic iterator on point sites of a mln::S.

Definition at line 55 of file p_graph_piter.hh.

10.293.2 Constructor & Destructor Documentation

10.293.2.1 template<typename S , typename I > [p_graph_piter](#)< S, I >::p_graph_piter () [inline]

Constructors.

Definition at line 151 of file p_graph_piter.hh.

10.293.3 Member Function Documentation

10.293.3.1 template<typename S , typename I > const S::graph_t & [p_graph_piter](#)< S, I >::graph () const [inline]

Return the graph associated to the target S.

Definition at line 212 of file p_graph_piter.hh.

10.293.3.2 template<typename S , typename I > unsigned [p_graph_piter](#)< S, I >::id () const [inline]

Return the graph element id.

Definition at line 228 of file p_graph_piter.hh.

10.293.3.3 template<typename S , typename I > mln::p_graph_piter< S, I >::mln_q_subject (iter)

Return the underlying graph element.

10.293.3.4 void mln::Site_Iterator< [p_graph_piter](#)< S, I > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.294 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, 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](#) (const S &s, const F &f)
Constructor with a site set s and a predicate f .
- [p_if](#) ()
Constructor without argument.
- bool [pred](#) (const [psite](#) &p) const
Test predicate on point site p .
- const F & [predicate](#) () const
Give the predicate function.

10.294.1 Detailed Description

template<typename S, typename F>class mln::p_if< S, F >

[Site](#) set restricted w.r.t.

a predicate.

Parameter *S* is a site set type; parameter *F* is a function from point to Boolean.

Definition at line 83 of file p_if.hh.

10.294.2 Member Typedef Documentation

10.294.2.1 template<typename S, typename F> typedef p_if_piter_<typename S ::bkd_piter, S, F> mln::p_if< S, F >::bkd_piter

Backward [Site_iterator](#) associated type.

Definition at line 100 of file p_if.hh.

10.294.2.2 template<typename S, typename F> typedef S ::element mln::p_if< S, F >::element

Element associated type.

Definition at line 90 of file p_if.hh.

10.294.2.3 template<typename S, typename F> typedef p_if_piter_<typename S ::fwd_piter, S, F> mln::p_if< S, F >::fwd_piter

Forward [Site_iterator](#) associated type.

Definition at line 97 of file p_if.hh.

10.294.2.4 template<typename S, typename F> typedef fwd_piter mln::p_if< S, F >::piter

[Site_iterator](#) associated type.

Definition at line 103 of file p_if.hh.

10.294.2.5 template<typename S, typename F> typedef S ::psite mln::p_if< S, F >::psite

Psite associated type.

Definition at line 94 of file p_if.hh.

10.294.3 Constructor & Destructor Documentation

10.294.3.1 template<typename S , typename F > p_if< S, F >::p_if (const S & s, const F & f) [inline]

Constructor with a site set *s* and a predicate *f*.

Definition at line 190 of file p_if.hh.

10.294.3.2 template<typename S , typename F > p_if< S, F >::p_if () [inline]

Constructor without argument.

Definition at line 198 of file p_if.hh.

10.294.4 Member Function Documentation

10.294.4.1 `template<typename S, typename F> bool p_if< S, F>::has (const psite & p) const` `[inline]`

Test if `p` belongs to the subset.

Definition at line 159 of file p_if.hh.

10.294.4.2 `template<typename S, typename F> bool p_if< S, F>::is_valid () const` `[inline]`

Test if this site set is valid.

Definition at line 167 of file p_if.hh.

10.294.4.3 `template<typename S, typename F> std::size_t p_if< S, F>::memory_size () const` `[inline]`

Return the size of this site set in memory.

Definition at line 213 of file p_if.hh.

10.294.4.4 `template<typename S, typename F> const S & p_if< S, F>::overset () const` `[inline]`

Give the primary overset.

Definition at line 175 of file p_if.hh.

10.294.4.5 `template<typename S, typename F> bool p_if< S, F>::pred (const psite & p) const` `[inline]`

Test predicate on point site `p`.

Definition at line 183 of file p_if.hh.

10.294.4.6 `template<typename S, typename F> const F & p_if< S, F>::predicate () const` `[inline]`

Give the predicate function.

Definition at line 205 of file p_if.hh.

10.295 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, p_image< I > >`.

Public Types

- typedef [S::bkd_piter](#) `bkd_piter`
Backward [Site_Iterator](#) associated type.
- typedef `I::psite` [element](#)
Element associated type.

- typedef [S::fwd_piter](#) [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef [psite](#) [i_element](#)
Insertion element associated type.
- typedef [S::piter](#) [piter](#)
[Site_Iterator](#) associated type.
- typedef [I::psite](#) [psite](#)
Psite associated type.
- typedef [psite](#) [r_element](#)
Removal element associated type.
- typedef
internal::p_image_site_set< I >
::ret [S](#)
Equivalent [site_set](#) type.

Public Member Functions

- void [clear](#) ()
Clear this set.
- bool [has](#) (const [psite](#) &) const
*Test if the [psite](#) *p* belongs to this site set.*
- void [insert](#) (const [psite](#) &p)
*Insert a site *p*.*
- bool [is_valid](#) () const
Test if this site set is valid, i.e., initialized.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- unsigned [nsites](#) () const
Give the number of sites.
- operator typename internal::p_image_site_set< I >::ret () const
Conversion towards the equivalent site set.
- [p_image](#) ()
Constructor without argument.
- [p_image](#) (const I &ima)
Constructor.
- void [remove](#) (const [psite](#) &p)
*Remove a site *p*.*
- void [toggle](#) (const [psite](#) &p)
*Change the status in/out of a site *p*.*

10.295.1 Detailed Description

template<typename I>class mln::p_image< I >

[Site](#) set based on an image of Booleans.

Definition at line 88 of file [p_image.hh](#).

10.295.2 Member Typedef Documentation

10.295.2.1 `template<typename I> typedef S::bkd_piter mln::p_image<I>::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 110 of file p_image.hh.

10.295.2.2 `template<typename I> typedef I::psite mln::p_image<I>::element`

Element associated type.

Definition at line 100 of file p_image.hh.

10.295.2.3 `template<typename I> typedef S::fwd_piter mln::p_image<I>::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 107 of file p_image.hh.

10.295.2.4 `template<typename I> typedef psite mln::p_image<I>::i_element`

Insertion element associated type.

Definition at line 136 of file p_image.hh.

10.295.2.5 `template<typename I> typedef S::piter mln::p_image<I>::piter`

[Site_Iterator](#) associated type.

Definition at line 113 of file p_image.hh.

10.295.2.6 `template<typename I> typedef I::psite mln::p_image<I>::psite`

Psite associated type.

Definition at line 104 of file p_image.hh.

10.295.2.7 `template<typename I> typedef psite mln::p_image<I>::r_element`

Removal element associated type.

Definition at line 142 of file p_image.hh.

10.295.2.8 `template<typename I> typedef internal::p_image_site_set<I>::ret mln::p_image<I>::S`

Equivalent site_set type.

Definition at line 93 of file p_image.hh.

10.295.3 Constructor & Destructor Documentation

10.295.3.1 `template<typename I> p_image<I>::p_image() [inline]`

Constructor without argument.

Definition at line 182 of file p_image.hh.

10.295.3.2 `template<typename I> p_image< I >::p_image (const I & ima) [inline]`

Constructor.

Definition at line 189 of file p_image.hh.

10.295.4 Member Function Documentation

10.295.4.1 `template<typename I> void p_image< I >::clear () [inline]`

Clear this set.

Definition at line 283 of file p_image.hh.

References mln::data::fill_with_value().

10.295.4.2 `template<typename I> bool p_image< I >::has (const psite & p) const [inline]`

Test is the psite *p* belongs to this site set.

Definition at line 199 of file p_image.hh.

10.295.4.3 `template<typename I> void p_image< I >::insert (const psite & p) [inline]`

Insert a site *p*.

Definition at line 224 of file p_image.hh.

10.295.4.4 `template<typename I> bool p_image< I >::is_valid () const [inline]`

Test if this site set is valid, i.e., initialized.

Definition at line 208 of file p_image.hh.

10.295.4.5 `template<typename I> std::size_t p_image< I >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 273 of file p_image.hh.

10.295.4.6 `template<typename I> unsigned p_image< I >::nsites () const [inline]`

Give the number of sites.

Definition at line 216 of file p_image.hh.

10.295.4.7 `template<typename I> p_image< I >::operator typename internal::p_image_site_set< I >::ret () const [inline]`

Conversion towards the equivalent site set.

Definition at line 174 of file p_image.hh.

10.295.4.8 `template<typename I> void p_image< I >::remove (const psite & p) [inline]`

Remove a site *p*.

Definition at line 237 of file p_image.hh.

10.295.4.9 `template<typename I> void p_image< I>::toggle (const psite & p) [inline]`

Change the status in/out of a site p.

Definition at line 251 of file p_image.hh.

10.296 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, 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 ()`
Constructor with no argument.
- `p_indexed_bkd_piter (const S &s)`
Constructor.

10.296.1 Detailed Description

`template<typename S>class mln::p_indexed_bkd_piter< S >`

Backward iterator on sites of an indexed site set.

Definition at line 276 of file p_array.hh.

10.296.2 Constructor & Destructor Documentation

10.296.2.1 `template<typename S> p_indexed_bkd_piter< S>::p_indexed_bkd_piter () [inline]`

Constructor with no argument.

Definition at line 686 of file p_array.hh.

10.296.2.2 `template<typename S> p_indexed_bkd_piter< S>::p_indexed_bkd_piter (const S &s) [inline]`

Constructor.

Definition at line 692 of file p_array.hh.

10.296.3 Member Function Documentation

10.296.3.1 `template<typename S> int p_indexed_bkd_piter< S>::index () const [inline]`

Return the current index.

Definition at line 733 of file p_array.hh.

10.296.3.2 void mln::Site_Iterator< p_indexed_bkd_piter< S > >::next() [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.297 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, 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](#) ()
Constructor with no argument.
- [p_indexed_fwd_piter](#) (const S &s)
Constructor.

10.297.1 Detailed Description

```
template<typename S>class mln::p_indexed_fwd_piter< S >
```

Forward iterator on sites of an indexed site set.

Definition at line 235 of file p_array.hh.

10.297.2 Constructor & Destructor Documentation

10.297.2.1 template<typename S > p_indexed_fwd_piter< S >::p_indexed_fwd_piter() [inline]

Constructor with no argument.

Definition at line 629 of file p_array.hh.

10.297.2.2 template<typename S > p_indexed_fwd_piter< S >::p_indexed_fwd_piter(const S &s) [inline]

Constructor.

Definition at line 635 of file p_array.hh.

10.297.3 Member Function Documentation

10.297.3.1 `template<typename S> int p_indexed_fwd_piter< S>::index () const [inline]`

Return the current index.

Definition at line 676 of file `p_array.hh`.

10.297.3.2 `void mln::Site_Iterator< p_indexed_fwd_piter< S> >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.298 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 &, p_indexed_psite< S> >`.

10.298.1 Detailed Description

```
template<typename S>class mln::p_indexed_psite< S>
```

Psite class for indexed site sets such as [p_array](#).

.

Definition at line 183 of file `p_array.hh`.

10.299 mln::p_key< K, P > Class Template Reference

Priority queue class.

```
#include <p_key.hh>
```

Inherits `mln::internal::site_set_base_< P, p_key< K, P> >`.

Public Types

- typedef `p_double_piter< self_, mln_bkd_eiter(util::set< K>), typename p_set< P> ::bkd_piter > bkd_piter`
Backward [Site_Iterator](#) associated type.
- typedef `P element`

- *Element associated type.*
 • typedef p_double_piter< self_,
 mln_fwd_eiter(util::set< K >
), typename p_set< P >
 ::fwd_piter > fwd_piter
 Forward *Site_Iterator* associated type.
- typedef std::pair< K, P > i_element
 Insertion element associated type.
- typedef fwd_piter piter
Site_Iterator associated type.
- typedef p_double_psite< self_,
 p_set< P > > psite
 Psite associated type.
- typedef P r_element
 Removal element associated type.

Public Member Functions

- void change_key (const K &k, const K &new_k)
 Change the key *k* into a new value *new_k*.
- template<typename F >
 void change_keys (const Function_v2v< F > &f)
 Change the keys by applying the function *f*.
- void clear ()
 Clear this site set.
- bool exists_key (const K &key) const
 Test if the *priority* exists.
- bool has (const psite &) const
 Test is the psite *p* belongs to this site set.
- bool has (const P &p) const
 Test is the psite *p* belongs to this site set.
- void insert (const i_element &k_p)
 Insert a pair *k_p* (key *k*, site *p*).
- void insert (const K &k, const P &p)
 Insert a pair (key *k*, site *p*).
- bool is_valid () const
 Test this set validity so returns always true.
- const K & key (const P &p) const
 Give the key associated with site *p*.
- const util::set< K > & keys () const
 Give the set of keys.
- std::size_t memory_size () const
 Return the size of this site set in memory.
- unsigned nsites () const
 Give the number of sites.
- const p_set< P > & operator() (const K &key) const
 Give the queue with the priority *priority*.
- p_key ()
 Constructor.
- void remove (const P &p)
 Remove a site *p*.
- void remove_key (const K &k)
 Remove all sites with key *k*.

10.299.1 Detailed Description

`template<typename K, typename P> class mln::p_key< K, P >`

Priority queue class.

Definition at line 72 of file p_key.hh.

10.299.2 Member Typedef Documentation

10.299.2.1 `template<typename K , typename P > typedef p_double_piter<self_, mln_bkd_eiter(util::set<K>), typename p_set<P>::bkd_piter> mln::p_key< K, P >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 93 of file p_key.hh.

10.299.2.2 `template<typename K , typename P > typedef P mln::p_key< K, P >::element`

Element associated type.

Definition at line 79 of file p_key.hh.

10.299.2.3 `template<typename K , typename P > typedef p_double_piter<self_, mln_fwd_eiter(util::set<K>), typename p_set<P>::fwd_piter> mln::p_key< K, P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 88 of file p_key.hh.

10.299.2.4 `template<typename K , typename P > typedef std::pair<K,P> mln::p_key< K, P >::i_element`

Insertion element associated type.

Definition at line 118 of file p_key.hh.

10.299.2.5 `template<typename K , typename P > typedef fwd_piter mln::p_key< K, P >::piter`

[Site_Iterator](#) associated type.

Definition at line 96 of file p_key.hh.

10.299.2.6 `template<typename K , typename P > typedef p_double_psite< self_, p_set<P> > mln::p_key< K, P >::psite`

Psite associated type.

Definition at line 83 of file p_key.hh.

10.299.2.7 `template<typename K , typename P > typedef P mln::p_key< K, P >::r_element`

Removal element associated type.

Definition at line 132 of file p_key.hh.

10.299.3 Constructor & Destructor Documentation

10.299.3.1 `template<typename K , typename P > p_key< K, P >::p_key () [inline]`

Constructor.

Definition at line 208 of file p_key.hh.

10.299.4 Member Function Documentation

10.299.4.1 `template<typename K , typename P > void p_key< K, P >::change_key (const K & k, const K & new_k) [inline]`

Change the key `k` into a new value `new_k`.

Definition at line 382 of file p_key.hh.

References `mln::p_set< P >::nsites()`.

10.299.4.2 `template<typename K , typename P > template<typename F > void p_key< K, P >::change_keys (const Function_v2v< F > & f) [inline]`

Change the keys by applying the function `f`.

Definition at line 428 of file p_key.hh.

References `mln::util::set< T >::insert()`.

10.299.4.3 `template<typename K , typename P > void p_key< K, P >::clear () [inline]`

Clear this site set.

Definition at line 462 of file p_key.hh.

10.299.4.4 `template<typename K , typename P > bool p_key< K, P >::exists_key (const K & key) const [inline]`

Test if the `priority` exists.

Definition at line 520 of file p_key.hh.

10.299.4.5 `template<typename K , typename P > bool p_key< K, P >::has (const psite &) const [inline]`

Test is the `psite p` belongs to this site set.

Definition at line 217 of file p_key.hh.

10.299.4.6 `template<typename K , typename P > bool p_key< K, P >::has (const P & p) const [inline]`

Test is the `psite p` belongs to this site set.

Definition at line 227 of file p_key.hh.

10.299.4.7 `template<typename K , typename P > void p_key< K, P >::insert (const i_element & k.p) [inline]`

Insert a pair `k_p` (key `k`, site `p`).

Definition at line 301 of file p_key.hh.

10.299.4.8 `template<typename K , typename P > void p_key< K, P >::insert (const K & k, const P & p)` `[inline]`

Insert a pair (key *k*, site *p*).

Definition at line 268 of file `p_key.hh`.

10.299.4.9 `template<typename K , typename P > bool p_key< K, P >::is_valid () const` `[inline]`

Test this set validity so returns always true.

Definition at line 236 of file `p_key.hh`.

10.299.4.10 `template<typename K , typename P > const K & p_key< K, P >::key (const P & p) const` `[inline]`

Give the key associated with site *p*.

Definition at line 501 of file `p_key.hh`.

10.299.4.11 `template<typename K , typename P > const util::set< K > & p_key< K, P >::keys () const` `[inline]`

Give the set of keys.

Definition at line 511 of file `p_key.hh`.

10.299.4.12 `template<typename K , typename P > std::size_t p_key< K, P >::memory_size () const` `[inline]`

Return the size of this site set in memory.

Definition at line 475 of file `p_key.hh`.

10.299.4.13 `template<typename K , typename P > unsigned p_key< K, P >::nsites () const` `[inline]`

Give the number of sites.

Definition at line 245 of file `p_key.hh`.

10.299.4.14 `template<typename K , typename P > const p_set< P > & p_key< K, P >::operator() (const K & key) const` `[inline]`

Give the queue with the priority *priority*.

This method always works: if the priority is not in this set, an empty queue is returned.

Definition at line 489 of file `p_key.hh`.

10.299.4.15 `template<typename K , typename P > void p_key< K, P >::remove (const P & p)` `[inline]`

Remove a site *p*.

Definition at line 309 of file `p_key.hh`.

10.299.4.16 `template<typename K , typename P > void p_key< K, P >::remove_key (const K & k)` `[inline]`

Remove all sites with key *k*.

Definition at line 351 of file `p_key.hh`.

References `mln::p_set< P >::nsites()`.

10.300 mln::p_line2d Class Reference

2D discrete line of points.

```
#include <p_line2d.hh>
```

Inherits mln::internal::site_set_base_< point2d, p_line2d >.

Public Types

- typedef [p_indexed_bkd_piter](#)
< [self_](#) > [bkd_piter](#)
Backward Site_Iterator associated type.
- typedef [point2d](#) [element](#)
Element associated type.
- typedef [p_indexed_fwd_piter](#)
< [self_](#) > [fwd_piter](#)
Forward Site_Iterator associated type.
- typedef [p_indexed_fwd_piter](#)
< [self_](#) > [piter](#)
Site_Iterator associated type.
- typedef [p_indexed_psite](#)< [self_](#) > [psite](#)
Psite associated type.
- typedef const [box2d](#) & [q_box](#)
Box (qualified) associated type.

Public Member Functions

- const [box2d](#) & [bbox](#) () const
Give the exact bounding box.
- const [point2d](#) & [begin](#) () const
Give the point that begins the line.
- const [point2d](#) & [end](#) () const
Give the point that ends the line.
- bool [has](#) (const [psite](#) &p) const
Test if p belongs to this point set.
- bool [has](#) (const util::index &i) const
Test if index i 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](#) ()
Constructor without argument.
- [p_line2d](#) (const [point2d](#) &beg, const [point2d](#) &end, bool is_end_excluded=false)
Constructor from point beg to point end .
- const std::vector< [point2d](#) > & [std_vector](#) () const
Return the corresponding std::vector of points.

10.300.1 Detailed Description

2D discrete line of points.

It is based on [p_array](#).

Definition at line 79 of file p_line2d.hh.

10.300.2 Member Typedef Documentation

10.300.2.1 `typedef p_indexed_bkd_piter<self_> mln::p_line2d::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 97 of file p_line2d.hh.

10.300.2.2 `typedef point2d mln::p_line2d::element`

Element associated type.

Definition at line 85 of file p_line2d.hh.

10.300.2.3 `typedef p_indexed_fwd_piter<self_> mln::p_line2d::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 94 of file p_line2d.hh.

10.300.2.4 `typedef p_indexed_fwd_piter<self_> mln::p_line2d::piter`

[Site_Iterator](#) associated type.

Definition at line 91 of file p_line2d.hh.

10.300.2.5 `typedef p_indexed_psite<self_> mln::p_line2d::psite`

Psite associated type.

Definition at line 88 of file p_line2d.hh.

10.300.2.6 `typedef const box2d& mln::p_line2d::q_box`

[Box](#) (qualified) associated type.

Definition at line 132 of file p_line2d.hh.

10.300.3 Constructor & Destructor Documentation

10.300.3.1 `mln::p_line2d::p_line2d () [inline]`

Constructor without argument.

Definition at line 161 of file p_line2d.hh.

References [is_valid\(\)](#).

10.300.3.2 `mln::p_line2d::p_line2d (const point2d & beg, const point2d & end, bool is_end_excluded = false)`
[inline]

Constructor from point `beg` to point `end`.

Definition at line 167 of file `p_line2d.hh`.

References `is_valid()`.

10.300.4 Member Function Documentation

10.300.4.1 `const box2d & mln::p_line2d::bbox () const` [inline]

Give the exact bounding box.

Definition at line 273 of file `p_line2d.hh`.

References `is_valid()`.

10.300.4.2 `const point2d & mln::p_line2d::begin () const` [inline]

Give the point that begins the line.

Definition at line 307 of file `p_line2d.hh`.

References `is_valid()`.

Referenced by `mln::debug::draw_graph()`.

10.300.4.3 `const point2d & mln::p_line2d::end () const` [inline]

Give the point that ends the line.

Definition at line 315 of file `p_line2d.hh`.

References `is_valid()`, and `nsites()`.

Referenced by `mln::debug::draw_graph()`.

10.300.4.4 `bool mln::p_line2d::has (const psite & p) const` [inline]

Test if `p` belongs to this point set.

Definition at line 240 of file `p_line2d.hh`.

10.300.4.5 `bool mln::p_line2d::has (const util::index & i) const` [inline]

Test if index `i` belongs to this point set.

Definition at line 251 of file `p_line2d.hh`.

References `nsites()`.

10.300.4.6 `bool mln::p_line2d::is_valid () const` [inline]

Test if this line is valid, i.e., initialized.

Definition at line 258 of file `p_line2d.hh`.

References `mln::implies()`.

Referenced by `bbox()`, `begin()`, `end()`, and `p_line2d()`.

10.300.4.7 `std::size_t mln::p_line2d::memory_size () const` `[inline]`

Return the size of this site set in memory.

Definition at line 323 of file `p_line2d.hh`.

10.300.4.8 `unsigned mln::p_line2d::nsites () const` `[inline]`

Give the number of points.

Definition at line 266 of file `p_line2d.hh`.

Referenced by `end()`, `has()`, and `operator[]()`.

10.300.4.9 `const point2d & mln::p_line2d::operator[] (unsigned i) const` `[inline]`

Return the `i`-th point of the line.

Definition at line 299 of file `p_line2d.hh`.

References `nsites()`.

10.300.4.10 `const std::vector< point2d > & mln::p_line2d::std_vector () const` `[inline]`

Return the corresponding `std::vector` of points.

Definition at line 281 of file `p_line2d.hh`.

10.301 `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, p_mutable_array_of< S > >`, `mlc_is_aS`, and `check_t`.

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.
- const S & [operator\[\]](#) (unsigned i) const
Return the i -th site set (const version).
- S & [operator\[\]](#) (unsigned i)
Return the i -th site set (mutable version).
- [p_mutable_array_of](#) ()
Constructor without arguments.
- void [reserve](#) (unsigned n)
Reserve memory for n elements.

10.301.1 Detailed Description

template<typename S>class mln::p_mutable_array_of< S >

[p_mutable_array_of](#) is a mutable array of site sets.

Parameter S is the type of the contained site sets.

Definition at line 76 of file p_mutable_array_of.hh.

10.301.2 Member Typedef Documentation

10.301.2.1 template<typename S > typedef p_double_piter<self_, mln_bkd_eiter(array_), typename S ::bkd_piter>
mln::p_mutable_array_of< S >::bkd_piter

Backward [Site_Iterator](#) associated type.

Definition at line 99 of file p_mutable_array_of.hh.

10.301.2.2 template<typename S > typedef S mln::p_mutable_array_of< S >::element

Element associated type.

Definition at line 85 of file p_mutable_array_of.hh.

10.301.2.3 template<typename S > typedef p_double_piter<self_, mln_fwd_eiter(array_), typename S ::fwd_piter>
mln::p_mutable_array_of< S >::fwd_piter

Forward [Site_Iterator](#) associated type.

Definition at line 94 of file p_mutable_array_of.hh.

10.301.2.4 `template<typename S > typedef S mln::p_mutable_array_of< S >::i_element`

Insertion element associated type.

Definition at line 121 of file `p_mutable_array_of.hh`.

10.301.2.5 `template<typename S > typedef fwd_piter mln::p_mutable_array_of< S >::piter`

[Site_iterator](#) associated type.

Definition at line 102 of file `p_mutable_array_of.hh`.

10.301.2.6 `template<typename S > typedef p_double_psite<self_element> mln::p_mutable_array_of< S >::psite`

Psite associated type.

Definition at line 89 of file `p_mutable_array_of.hh`.

10.301.3 Constructor & Destructor Documentation

10.301.3.1 `template<typename S > p_mutable_array_of< S >::p_mutable_array_of() [inline]`

Constructor without arguments.

Definition at line 175 of file `p_mutable_array_of.hh`.

10.301.4 Member Function Documentation

10.301.4.1 `template<typename S > void p_mutable_array_of< S >::clear() [inline]`

Clear this set.

Definition at line 241 of file `p_mutable_array_of.hh`.

10.301.4.2 `template<typename S > bool p_mutable_array_of< S >::has(const psite & p) const [inline]`

Test if `p` belongs to this point set.

Definition at line 190 of file `p_mutable_array_of.hh`.

10.301.4.3 `template<typename S > void p_mutable_array_of< S >::insert(const S & s) [inline]`

Insert a site set `s`.

Precondition

`s` is valid.

Definition at line 206 of file `p_mutable_array_of.hh`.

10.301.4.4 `template<typename S > bool p_mutable_array_of< S >::is_valid() const [inline]`

Test this set validity so returns always true.

Definition at line 198 of file `p_mutable_array_of.hh`.

10.301.4.5 `template<typename S> std::size_t p_mutable_array_of< S >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 258 of file p_mutable_array_of.hh.

10.301.4.6 `template<typename S> unsigned p_mutable_array_of< S >::nelements () const [inline]`

Give the number of elements (site sets) of this composite.

Definition at line 233 of file p_mutable_array_of.hh.

10.301.4.7 `template<typename S> const S & p_mutable_array_of< S >::operator[] (unsigned i) const [inline]`

Return the *i*-th site set (const version).

Definition at line 215 of file p_mutable_array_of.hh.

10.301.4.8 `template<typename S> S & p_mutable_array_of< S >::operator[] (unsigned i) [inline]`

Return the *i*-th site set (mutable version).

Definition at line 224 of file p_mutable_array_of.hh.

10.301.4.9 `template<typename S> void p_mutable_array_of< S >::reserve (unsigned n) [inline]`

Reserve memory for *n* elements.

Definition at line 182 of file p_mutable_array_of.hh.

10.302 mln::p_n_faces_bkd_piter< D, G > Class Template Reference

Backward iterator on the *n*-faces sites of an mln::p_complex<D, G>.

`#include <p_n_faces_piter.hh>`

Inherits mln::internal::p_complex_piter_base_< topo::n_face_bkd_iter< D >, p_complex< D, G >, G::site, p_n_faces_bkd_piter< D, G > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [p_n_faces_bkd_piter](#) ()
Construction and assignment.
- unsigned [n](#) () const
Accessors.

10.302.1 Detailed Description

`template<unsigned D, typename G> class mln::p_n_faces_bkd_piter< D, G >`

Backward iterator on the *n*-faces sites of an mln::p_complex<D, G>.

Definition at line 92 of file `p_n_faces_piter.hh`.

10.302.2 Constructor & Destructor Documentation

10.302.2.1 `template<unsigned D, typename G > mln::p_n_faces_bkd_piter< D, G >::p_n_faces_bkd_piter ()`
`[inline]`

Construction and assignment.

Definition at line 169 of file `p_n_faces_piter.hh`.

10.302.3 Member Function Documentation

10.302.3.1 `template<unsigned D, typename G > unsigned mln::p_n_faces_bkd_piter< D, G >::n () const`
`[inline]`

Accessors.

Shortcuts to `face_`'s accessors.

Definition at line 186 of file `p_n_faces_piter.hh`.

10.302.3.2 `void mln::Site_Iterator< p_n_faces_bkd_piter< D, G > >::next ()` `[inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.303 `mln::p_n_faces_fwd_piter< D, G >` Class Template Reference

Forward iterator on the n-faces sites of an `mln::p_complex<D, G>`.

```
#include <p_n_faces_piter.hh>
```

Inherits `mln::internal::p_complex_piter_base_< topo::n_face_fwd_iter< D >, p_complex< D, G >, G::site, p_n_faces_fwd_piter< D, G > >`.

Public Member Functions

- void `next ()`
Go to the next element.
- `p_n_faces_fwd_piter ()`
Construction and assignment.
- unsigned `n () const`
Accessors.

10.303.1 Detailed Description

template<unsigned D, typename G>class mln::p_n_faces_fwd_piter< D, G >

Forward iterator on the n-faces sites of an mln::p_complex<D, G>.

Definition at line 56 of file p_n_faces_piter.hh.

10.303.2 Constructor & Destructor Documentation

10.303.2.1 template<unsigned D, typename G > mln::p_n_faces_fwd_piter< D, G >::p_n_faces_fwd_piter ()
[inline]

Construction and assignment.

Definition at line 132 of file p_n_faces_piter.hh.

10.303.3 Member Function Documentation

10.303.3.1 template<unsigned D, typename G > unsigned mln::p_n_faces_fwd_piter< D, G >::n () const
[inline]

Accessors.

Shortcuts to face_'s accessors.

Definition at line 149 of file p_n_faces_piter.hh.

10.303.3.2 void mln::Site_Iterator< p_n_faces_fwd_piter< D, G > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_t* method.

Precondition

The iterator is valid.

10.304 mln::p_priority< P, Q > Class Template Reference

Priority queue.

#include <p_priority.hh>

Inherits mln::internal::site_set_base_< Q::site, p_priority< P, Q > >, mlc_is_aQ, and check_t.

Public Types

- typedef p_double_piter< [self_](#), mln_fwd_eiter([util::set](#)< P >), typename Q::bkd_piter > [bkd_piter](#)
Backward [Site_Iterator](#) associated type.
- typedef Q::element [element](#)

- Element associated type.*
- typedef p_double_piter< [self_](#),
mln_bkd_eiter(util::set< P >
>, typename Q::fwd_piter > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef std::pair< P, [element](#) > [i_element](#)
Insertion element associated type.
- typedef [fwd_piter](#) piter
[Site_Iterator](#) associated type.
- typedef p_double_psite< [self_](#), Q > [psite](#)
Psite associated type.

Public Member Functions

- void [clear](#) ()
Clear the queue.
- bool [exists_priority](#) (const P &priority) const
*Test if the *priority* exists.*
- const Q::element & [front](#) () const
Give an element with highest priority.
- bool [has](#) (const [psite](#) &) const
*Test is the psite *p* belongs to this site set.*
- const P [highest_priority](#) () const
Give the highest priority.
- void [insert](#) (const [i_element](#) &p_e)
*Insert a pair *p_e* (priority *p*, element *e*).*
- void [insert](#) (const [p_priority](#)< P, Q > &other)
Insert elements from another priority queue.
- bool [is_valid](#) () const
Test this set validity so returns always true.
- const P [lowest_priority](#) () const
Give the lowest priority.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- unsigned [nsites](#) () const
Give the number of sites.
- const Q & [operator](#)() (const P &priority) const
*Give the queue with the priority *priority*.*
- [p_priority](#) ()
Constructor.
- void [pop](#) ()
Pop (remove) from the queue an element with highest priority.
- Q::element [pop_front](#) ()
Return an element with highest priority and remove it from the set.
- const util::set< P > & [priorities](#) () const
Give the set of priorities.
- void [push](#) (const P &priority, const [element](#) &e)
*Push in the queue with *priority* the element *e*.*

10.304.1 Detailed Description

`template<typename P, typename Q>class mln::p_priority< P, Q >`

Priority queue.

The parameter `P` is the type of the priorities (for instance `unsigned`).

The parameter `Q` is a type of queue (for instance `p_queue<point2d>`).

Definition at line 76 of file `p_priority.hh`.

10.304.2 Member Typedef Documentation

10.304.2.1 `template<typename P, typename Q> typedef p_double_piter< self_, mln_fwd_eiter(util::set<P>), typename Q ::bkd_piter > mln::p_priority< P, Q >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 98 of file `p_priority.hh`.

10.304.2.2 `template<typename P, typename Q> typedef Q ::element mln::p_priority< P, Q >::element`

Element associated type.

Definition at line 84 of file `p_priority.hh`.

10.304.2.3 `template<typename P, typename Q> typedef p_double_piter< self_, mln_bkd_eiter(util::set<P>), typename Q ::fwd_piter > mln::p_priority< P, Q >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 93 of file `p_priority.hh`.

10.304.2.4 `template<typename P, typename Q> typedef std::pair<P, element> mln::p_priority< P, Q >::i_element`

Insertion element associated type.

Definition at line 121 of file `p_priority.hh`.

10.304.2.5 `template<typename P, typename Q> typedef fwd_piter mln::p_priority< P, Q >::piter`

[Site_Iterator](#) associated type.

Definition at line 101 of file `p_priority.hh`.

10.304.2.6 `template<typename P, typename Q> typedef p_double_psite<self_, Q> mln::p_priority< P, Q >::psite`

Psite associated type.

Definition at line 88 of file `p_priority.hh`.

10.304.3 Constructor & Destructor Documentation

10.304.3.1 `template<typename P , typename Q > p_priority< P, Q >::p_priority () [inline]`

Constructor.

Definition at line 202 of file `p_priority.hh`.

10.304.4 Member Function Documentation

10.304.4.1 `template<typename P, typename Q> void p_priority<P, Q>::clear () [inline]`

Clear the queue.

Definition at line 316 of file `p_priority.hh`.

10.304.4.2 `template<typename P, typename Q> bool p_priority<P, Q>::exists_priority (const P & priority) const [inline]`

Test if the `priority` exists.

Definition at line 366 of file `p_priority.hh`.

Referenced by `mln::p_priority<P, Q>::operator()()`.

10.304.4.3 `template<typename P, typename Q> const Q::element & p_priority<P, Q>::front () const [inline]`

Give an element with highest priority.

If several elements have this priority, the least recently inserted is chosen.

Precondition

`! is_empty()`

Definition at line 294 of file `p_priority.hh`.

References `mln::p_priority<P, Q>::highest_priority()`.

Referenced by `mln::morpho::meyer_wst()`, `mln::p_priority<P, Q>::pop_front()`, and `mln::morpho::watershed::topological()`.

10.304.4.4 `template<typename P, typename Q> bool p_priority<P, Q>::has (const psite &) const [inline]`

Test is the `psite p` belongs to this site set.

Definition at line 211 of file `p_priority.hh`.

10.304.4.5 `template<typename P, typename Q> const P p_priority<P, Q>::highest_priority () const [inline]`

Give the highest priority.

Precondition

`! is_empty()`

Definition at line 375 of file `p_priority.hh`.

Referenced by `mln::p_priority<P, Q>::front()`, and `mln::p_priority<P, Q>::pop()`.

10.304.4.6 `template<typename P, typename Q> void p_priority<P, Q>::insert (const i_element & p_e) [inline]`

Insert a pair `p_e` (priority `p`, element `e`).

Definition at line 251 of file `p_priority.hh`.

References `mln::p_priority<P, Q>::push()`.

10.304.4.7 `template<typename P , typename Q > void p_priority< P, Q >::insert (const p_priority< P, Q > & other)`
`[inline]`

Insert elements from another priority queue.

Definition at line 259 of file p_priority.hh.

10.304.4.8 `template<typename P , typename Q > bool p_priority< P, Q >::is_valid () const` `[inline]`

Test this set validity so returns always true.

Definition at line 221 of file p_priority.hh.

10.304.4.9 `template<typename P , typename Q > const P p_priority< P, Q >::lowest_priority () const` `[inline]`

Give the lowest priority.

Precondition

`! is_empty()`

Definition at line 384 of file p_priority.hh.

10.304.4.10 `template<typename P , typename Q > std::size_t p_priority< P, Q >::memory_size () const` `[inline]`

Return the size of this site set in memory.

Definition at line 328 of file p_priority.hh.

10.304.4.11 `template<typename P , typename Q > unsigned p_priority< P, Q >::nsites () const` `[inline]`

Give the number of sites.

Definition at line 230 of file p_priority.hh.

Referenced by `mln::p_priority< P, Q >::operator()()`.

10.304.4.12 `template<typename P , typename Q > const Q & p_priority< P, Q >::operator() (const P & priority) const`
`[inline]`

Give the queue with the priority `priority`.

This method always works: if the priority is not in this set, an empty queue is returned.

Definition at line 341 of file p_priority.hh.

References `mln::p_priority< P, Q >::exists_priority()`, and `mln::p_priority< P, Q >::nsites()`.

10.304.4.13 `template<typename P , typename Q > void p_priority< P, Q >::pop ()` `[inline]`

Pop (remove) from the queue an element with highest priority.

If several elements have this priority, the least recently inserted is chosen.

Precondition

`! is_empty()`

Definition at line 277 of file `p_priority.hh`.

References `mln::p_priority< P, Q >::highest_priority()`.

Referenced by `mln::morpho::meyer_wst()`, `mln::p_priority< P, Q >::pop_front()`, and `mln::morpho::watershed::topological()`.

10.304.4.14 `template<typename P , typename Q > Q::element p_priority< P, Q >::pop_front () [inline]`

Return an element with highest priority and remove it from the set.

If several elements have this priority, the least recently inserted is chosen.

Precondition

`! is_empty()`

Definition at line 304 of file `p_priority.hh`.

References `mln::p_priority< P, Q >::front()`, and `mln::p_priority< P, Q >::pop()`.

10.304.4.15 `template<typename P , typename Q > const util::set< P > & p_priority< P, Q >::priorities () const [inline]`

Give the set of priorities.

Definition at line 357 of file `p_priority.hh`.

10.304.4.16 `template<typename P , typename Q > void p_priority< P, Q >::push (const P & priority, const element & e) [inline]`

Push in the queue with `priority` the element `e`.

Definition at line 239 of file `p_priority.hh`.

Referenced by `mln::p_priority< P, Q >::insert()`, `mln::morpho::meyer_wst()`, and `mln::morpho::watershed::topological()`.

10.305 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, p_queue< P > >`.

Public Types

- typedef `p_indexed_bkd_piter`
`< self_ > bkd_piter`
Backward Site_Iterator associated type.
- typedef `P element`
Element associated type.
- typedef `p_indexed_fwd_piter`
`< self_ > fwd_piter`

Forward [Site_Iterator](#) associated type.

- typedef P [i_element](#)
Insertion element associated type.
- typedef [fwd_piter](#) piter
[Site_Iterator](#) associated type.
- typedef [p_indexed_psite](#)< [self_](#) > [psite](#)
[Psite](#) associated type.

Public Member Functions

- void [clear](#) ()
Clear the queue.
- const P & [front](#) () const
Give the front site *p* of the queue; *p* is the least recently inserted site.
- bool [has](#) (const [psite](#) &p) const
Test if *p* belongs to this site set.
- bool [has](#) (const util::index &i) const
Test if index *i* belongs to this site set.
- void [insert](#) (const P &p)
Insert a site *p* (equivalent as 'push').
- bool [is_valid](#) () const
This set is always valid so it returns true.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- unsigned [nsites](#) () const
Give the number of sites.
- const P & [operator\[\]](#) (unsigned i) const
Return the *i*-th site.
- [p_queue](#) ()
Constructor without argument.
- void [pop](#) ()
Pop (remove) the front site *p* from the queue; *p* is the least recently inserted site.
- P [pop_front](#) ()
Pop (remove) the front site *p* from the queue; *p* is the least recently inserted site and give the front site *p* of the queue; *p* is the least recently inserted site.
- void [push](#) (const P &p)
Push a site *p* in the queue.
- const std::deque< P > & [std_deque](#) () const
Return the corresponding std::deque of sites.

10.305.1 Detailed Description

template<typename P>class mln::p_queue< P >

Queue of sites (based on std::deque).

The parameter *P* shall be a site or pseudo-site type.

Definition at line 74 of file p_queue.hh.

10.305.2 Member Typedef Documentation

10.305.2.1 `template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_queue< P >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 90 of file p_queue.hh.

10.305.2.2 `template<typename P> typedef P mln::p_queue< P >::element`

Element associated type.

Definition at line 80 of file p_queue.hh.

10.305.2.3 `template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_queue< P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 87 of file p_queue.hh.

10.305.2.4 `template<typename P> typedef P mln::p_queue< P >::i_element`

Insertion element associated type.

Definition at line 118 of file p_queue.hh.

10.305.2.5 `template<typename P> typedef fwd_piter mln::p_queue< P >::piter`

[Site_Iterator](#) associated type.

Definition at line 93 of file p_queue.hh.

10.305.2.6 `template<typename P> typedef p_indexed_psite<self_> mln::p_queue< P >::psite`

Psite associated type.

Definition at line 84 of file p_queue.hh.

10.305.3 Constructor & Destructor Documentation

10.305.3.1 `template<typename P> p_queue< P >::p_queue () [inline]`

Constructor without argument.

Definition at line 163 of file p_queue.hh.

10.305.4 Member Function Documentation

10.305.4.1 `template<typename P> void p_queue< P >::clear () [inline]`

Clear the queue.

Definition at line 244 of file p_queue.hh.

10.305.4.2 `template<typename P> const P & p_queue< P >::front () const` `[inline]`

Give the front site `p` of the queue; `p` is the least recently inserted site.

Definition at line 224 of file `p_queue.hh`.

Referenced by `mln::geom::impl::seeds2tiling()`.

10.305.4.3 `template<typename P> bool p_queue< P >::has (const psite & p) const` `[inline]`

Test if `p` belongs to this site set.

Definition at line 170 of file `p_queue.hh`.

10.305.4.4 `template<typename P> bool p_queue< P >::has (const util::index & i) const` `[inline]`

Test if index `i` belongs to this site set.

Definition at line 183 of file `p_queue.hh`.

10.305.4.5 `template<typename P> void p_queue< P >::insert (const P & p)` `[inline]`

Insert a site `p` (equivalent as 'push').

Definition at line 261 of file `p_queue.hh`.

10.305.4.6 `template<typename P> bool p_queue< P >::is_valid () const` `[inline]`

This set is always valid so it returns true.

Definition at line 191 of file `p_queue.hh`.

10.305.4.7 `template<typename P> std::size_t p_queue< P >::memory_size () const` `[inline]`

Return the size of this site set in memory.

Definition at line 277 of file `p_queue.hh`.

10.305.4.8 `template<typename P> unsigned p_queue< P >::nsites () const` `[inline]`

Give the number of sites.

Definition at line 199 of file `p_queue.hh`.

10.305.4.9 `template<typename P> const P & p_queue< P >::operator[] (unsigned i) const` `[inline]`

Return the `i`-th site.

Definition at line 252 of file `p_queue.hh`.

10.305.4.10 `template<typename P> void p_queue< P >::pop ()` `[inline]`

Pop (remove) the front site `p` from the queue; `p` is the least recently inserted site.

Definition at line 215 of file `p_queue.hh`.

Referenced by `mln::geom::impl::seeds2tiling()`.

10.305.4.11 `template<typename P> P p_queue<P>::pop_front () [inline]`

Pop (remove) the front site `p` from the queue; `p` is the least recently inserted site and give the front site `p` of the queue; `p` is the least recently inserted site.

Definition at line 233 of file `p_queue.hh`.

10.305.4.12 `template<typename P> void p_queue<P>::push (const P & p) [inline]`

Push a site `p` in the queue.

Definition at line 207 of file `p_queue.hh`.

Referenced by `mln::geom::impl::seeds2tiling()`.

10.305.4.13 `template<typename P> const std::deque<P> & p_queue<P>::std_deque () const [inline]`

Return the corresponding `std::deque` of sites.

Definition at line 269 of file `p_queue.hh`.

10.306 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, 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 **psite** &p) const
Test if p belongs to this site set.
- bool **has** (const util::index &i) const
Test if index i belongs to this site set.
- void **insert** (const P &p)
Insert a site p (equivalent as 'push').
- bool **is_valid** () const
This set is always valid so it returns true.
- std::size_t **memory_size** () const
Return the size of this site set in memory.
- unsigned **nsites** () const
Give the number of sites.
- const P & **operator[]** (unsigned i) const
Return the i -th site.
- **p_queue_fast** ()
Constructor without argument.
- void **pop** ()
Pop (remove) the front site p from the queue; p is the least recently inserted site.
- const P & **pop_front** ()
Pop (remove) the front site p from the queue; p is the least recently inserted site and give the front site p of the queue; p is the least recently inserted site.
- void **purge** ()
Purge the queue to save (free) some memory.
- void **push** (const P &p)
Push a site p in the queue.
- void **reserve** (typename **p_array**< P >::size_type n)
Reserve n cells.
- const std::vector< P > & **std_vector** () const
Return the corresponding std::vector of sites.

10.306.1 Detailed Description

template<typename P>class mln::p_queue_fast< P >

Queue of sites class (based on **p_array**).

This container is efficient; FIXME: explain...

The parameter **P** shall be a site or pseudo-site type.

Definition at line 72 of file p_queue_fast.hh.

10.306.2 Member Typedef Documentation

10.306.2.1 template<typename P > typedef **p_indexed_bkd_piter**<self_> mln::p_queue_fast< P >::bkd_piter

Backward **Site_iterator** associated type.

Definition at line 87 of file p_queue_fast.hh.

10.306.2.2 `template<typename P > typedef P mln::p_queue_fast< P >::element`

Element associated type.

Definition at line 78 of file `p_queue_fast.hh`.

10.306.2.3 `template<typename P > typedef p_indexed_fwd_piter<self_> mln::p_queue_fast< P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 84 of file `p_queue_fast.hh`.

10.306.2.4 `template<typename P > typedef P mln::p_queue_fast< P >::i_element`

Insertion element associated type.

Definition at line 121 of file `p_queue_fast.hh`.

10.306.2.5 `template<typename P > typedef fwd_piter mln::p_queue_fast< P >::piter`

[Site_Iterator](#) associated type.

Definition at line 90 of file `p_queue_fast.hh`.

10.306.2.6 `template<typename P > typedef p_indexed_psite<self_> mln::p_queue_fast< P >::psite`

Psite associated type.

Definition at line 81 of file `p_queue_fast.hh`.

10.306.3 Constructor & Destructor Documentation

10.306.3.1 `template<typename P > p_queue_fast< P >::p_queue_fast () [inline]`

Constructor without argument.

Definition at line 170 of file `p_queue_fast.hh`.

10.306.4 Member Function Documentation

10.306.4.1 `template<typename P > void p_queue_fast< P >::clear () [inline]`

Clear the queue.

Definition at line 297 of file `p_queue_fast.hh`.

10.306.4.2 `template<typename P > bool p_queue_fast< P >::compute_has (const P & p) const [inline]`

Test if `p` belongs to this site set.

Definition at line 222 of file `p_queue_fast.hh`.

10.306.4.3 `template<typename P > bool p_queue_fast< P >::empty () const [inline]`

Test if the queue is empty.

Definition at line 250 of file `p_queue_fast.hh`.

10.306.4.4 `template<typename P> const P & p_queue_fast< P >::front () const` `[inline]`

Give the front site `p` of the queue; `p` is the least recently inserted site.

Definition at line 277 of file `p_queue_fast.hh`.

10.306.4.5 `template<typename P> bool p_queue_fast< P >::has (const psite & p) const` `[inline]`

Test if `p` belongs to this site set.

Definition at line 201 of file `p_queue_fast.hh`.

10.306.4.6 `template<typename P> bool p_queue_fast< P >::has (const util::index & i) const` `[inline]`

Test if index `i` belongs to this site set.

Definition at line 214 of file `p_queue_fast.hh`.

10.306.4.7 `template<typename P> void p_queue_fast< P >::insert (const P & p)` `[inline]`

Insert a site `p` (equivalent as 'push').

Definition at line 314 of file `p_queue_fast.hh`.

10.306.4.8 `template<typename P> bool p_queue_fast< P >::is_valid () const` `[inline]`

This set is always valid so it returns true.

Definition at line 233 of file `p_queue_fast.hh`.

10.306.4.9 `template<typename P> std::size_t p_queue_fast< P >::memory_size () const` `[inline]`

Return the size of this site set in memory.

Definition at line 330 of file `p_queue_fast.hh`.

10.306.4.10 `template<typename P> unsigned p_queue_fast< P >::nsites () const` `[inline]`

Give the number of sites.

Definition at line 241 of file `p_queue_fast.hh`.

10.306.4.11 `template<typename P> const P & p_queue_fast< P >::operator[] (unsigned i) const` `[inline]`

Return the `i`-th site.

Definition at line 305 of file `p_queue_fast.hh`.

10.306.4.12 `template<typename P> void p_queue_fast< P >::pop ()` `[inline]`

Pop (remove) the front site `p` from the queue; `p` is the least recently inserted site.

Definition at line 268 of file `p_queue_fast.hh`.

10.306.4.13 `template<typename P> const P & p_queue_fast< P >::pop_front () [inline]`

Pop (remove) the front site `p` from the queue; `p` is the least recently inserted site and give the front site `p` of the queue; `p` is the least recently inserted site.

Definition at line 286 of file `p_queue_fast.hh`.

10.306.4.14 `template<typename P> void p_queue_fast< P >::purge () [inline]`

Purge the queue to save (free) some memory.

Definition at line 187 of file `p_queue_fast.hh`.

10.306.4.15 `template<typename P> void p_queue_fast< P >::push (const P & p) [inline]`

Push a site `p` in the queue.

Definition at line 259 of file `p_queue_fast.hh`.

10.306.4.16 `template<typename P> void p_queue_fast< P >::reserve (typename p_array< P >::size_type n) [inline]`

Reserve `n` cells.

Definition at line 179 of file `p_queue_fast.hh`.

10.306.4.17 `template<typename P> const std::vector< P > & p_queue_fast< P >::std_vector () const [inline]`

Return the corresponding `std::vector` of sites.

Definition at line 322 of file `p_queue_fast.hh`.

10.307 `mln::p_run< P >` Class Template Reference

[Point](#) set class in run.

```
#include <p_run.hh>
```

Inherits `mln::internal::site_set_base< P, p_run< P > >`.

Public Types

- `typedef p_run_bkd_piter< P > bkd_piter`
Backward [Site_Iterator](#) associated type.
- `typedef P element`
Element associated type.
- `typedef p_run_fwd_piter< P > fwd_piter`
Forward [Site_Iterator](#) associated type.
- `typedef fwd_piter piter`
[Site_Iterator](#) associated type.
- `typedef p_run_psite< P > psite`
Psite associated type.
- `typedef mln::box< P > q_box`
[Box](#) associated type.

Public Member Functions

- `mln::box< P > bbox () const`
Give the exact bounding box.
- `P end () const`
Return (compute) the ending point.
- `bool has (const psite &p) const`
Test if p belongs to this point set.
- `bool has (const P &p) const`
Test if p belongs to this point set.
- `bool has_index (unsigned short i) const`
Test if index i belongs to this point set.
- `void init (const P &start, unsigned short len)`
Set the starting point.
- `bool is_valid () const`
Test if this run is valid, i.e., with length > 0 .
- `unsigned short length () const`
Give the length of the run.
- `std::size_t memory_size () const`
Return the size of this site set in memory.
- `unsigned nsites () const`
Give the number of sites.
- `P operator[] (unsigned short i) const`
Return the i -th point.
- `p_run ()`
Constructor without argument.
- `p_run (const P &start, unsigned short len)`
Constructor.
- `p_run (const P &start, const P &end)`
Constructor.
- `const P & start () const`
Return the starting point.

10.307.1 Detailed Description

`template<typename P>class mln::p_run< P >`

[Point](#) set class in run.

This is a mathematical set of points (not a multi-set). The parameter P shall be a [Point](#) type.

Definition at line 86 of file `p_run.hh`.

10.307.2 Member Typedef Documentation

10.307.2.1 `template<typename P> typedef p_run_bkd_piter_<P> mln::p_run< P >::bkd_piter`

Backward [Site_iterator](#) associated type.

Definition at line 101 of file `p_run.hh`.

10.307.2.2 `template<typename P> typedef P mln::p_run< P >::element`

Element associated type.

Definition at line 91 of file p_run.hh.

10.307.2.3 `template<typename P> typedef p_run_fwd_piter_<P> mln::p_run< P >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 98 of file p_run.hh.

10.307.2.4 `template<typename P> typedef fwd_piter mln::p_run< P >::piter`

[Site_Iterator](#) associated type.

Definition at line 104 of file p_run.hh.

10.307.2.5 `template<typename P> typedef p_run_psite<P> mln::p_run< P >::psite`

Psite associated type.

Definition at line 95 of file p_run.hh.

10.307.2.6 `template<typename P> typedef mln::box<P> mln::p_run< P >::q_box`

[Box](#) associated type.

Definition at line 149 of file p_run.hh.

10.307.3 Constructor & Destructor Documentation

10.307.3.1 `template<typename P> p_run< P >::p_run () [inline]`

Constructor without argument.

Definition at line 223 of file p_run.hh.

10.307.3.2 `template<typename P> p_run< P >::p_run (const P & start, unsigned short len) [inline]`

Constructor.

Definition at line 230 of file p_run.hh.

10.307.3.3 `template<typename P> p_run< P >::p_run (const P & start, const P & end) [inline]`

Constructor.

Definition at line 238 of file p_run.hh.

10.307.4 Member Function Documentation

10.307.4.1 `template<typename P> mln::box< P > p_run< P >::bbox () const [inline]`

Give the exact bounding box.

Definition at line 267 of file p_run.hh.

10.307.4.2 `template<typename P> P p_run< P >::end () const [inline]`

Return (compute) the ending point.

Definition at line 348 of file p_run.hh.

References mln::point< G, C >::last_coord().

10.307.4.3 `template<typename P> bool p_run< P >::has (const psite & p) const [inline]`

Test if *p* belongs to this point set.

Definition at line 276 of file p_run.hh.

10.307.4.4 `template<typename P> bool p_run< P >::has (const P & p) const [inline]`

Test if *p* belongs to this point set.

Definition at line 289 of file p_run.hh.

10.307.4.5 `template<typename P> bool p_run< P >::has_index (unsigned short i) const [inline]`

Test if index *i* belongs to this point set.

Definition at line 302 of file p_run.hh.

10.307.4.6 `template<typename P> void p_run< P >::init (const P & start, unsigned short len) [inline]`

Set the starting point.

Definition at line 249 of file p_run.hh.

10.307.4.7 `template<typename P> bool p_run< P >::is_valid () const [inline]`

Test if this run is valid, i.e., with length > 0.

Definition at line 259 of file p_run.hh.

10.307.4.8 `template<typename P> unsigned short p_run< P >::length () const [inline]`

Give the length of the run.

Definition at line 319 of file p_run.hh.

10.307.4.9 `template<typename P> std::size_t p_run< P >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 358 of file p_run.hh.

10.307.4.10 `template<typename P> unsigned p_run< P >::nsites () const [inline]`

Give the number of sites.

Definition at line 310 of file p_run.hh.

10.307.4.11 `template<typename P> P p_run<P>::operator[] (unsigned short i) const` `[inline]`

Return the i -th point.

Definition at line 328 of file `p_run.hh`.

References `mln::point<G, C>::last_coord()`.

10.307.4.12 `template<typename P> const P & p_run<P>::start () const` `[inline]`

Return the starting point.

Definition at line 340 of file `p_run.hh`.

10.308 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, p_set<P>>`.

Public Types

- typedef [p_indexed_bkd_piter](#)
`< self_ > bkd_piter`
Backward [Site_Iterator](#) associated type.
- typedef P [element](#)
Element associated type.
- typedef [p_indexed_fwd_piter](#)
`< self_ > fwd_piter`
Forward [Site_Iterator](#) associated type.
- typedef P [i_element](#)
Insertion element associated type.
- typedef [fwd_piter](#) [piter](#)
[Site_Iterator](#) associated type.
- typedef [p_indexed_psite](#)`< self_ > psite`
Psite associated type.
- typedef P [r_element](#)
Removal element associated type.

Public Member Functions

- void [clear](#) ()
Clear this set.
- bool [has](#) (const [psite](#) &p) const
Test if psite p belongs to this point set.
- bool [has](#) (const P &p) const
Test if p belongs to this point set.
- bool [has](#) (const util::index &i) const
Test if index i belongs to this point set.
- void [insert](#) (const P &p)
Insert a site p .

- bool [is_valid](#) () const
Test this set validity so returns always true.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- unsigned [nsites](#) () const
Give the number of sites.
- const P & [operator\[\]](#) (unsigned i) const
*Return the *i*-th site.*
- [p_set](#) ()
Constructor.
- void [remove](#) (const P &p)
*Remove a site *p*.*
- const std::vector< P > & [std_vector](#) () const
Return the corresponding std::vector of sites.
- const [util::set](#)< P > & [util_set](#) () const
Return the corresponding [util::set](#) of sites.

10.308.1 Detailed Description

template<typename P>class mln::p_set< P >

Mathematical set of sites (based on [util::set](#)).

This is a mathematical set of sites (not a multi-set).

The parameter P shall be a site or pseudo-site type.

Definition at line 70 of file p_set.hh.

10.308.2 Member Typedef Documentation

10.308.2.1 template<typename P> typedef p_indexed_bkd_piter<self_> mln::p_set< P >::bkd_piter

Backward [Site_Iterator](#) associated type.

Definition at line 85 of file p_set.hh.

10.308.2.2 template<typename P> typedef P mln::p_set< P >::element

Element associated type.

Definition at line 76 of file p_set.hh.

10.308.2.3 template<typename P> typedef p_indexed_fwd_piter<self_> mln::p_set< P >::fwd_piter

Forward [Site_Iterator](#) associated type.

Definition at line 82 of file p_set.hh.

10.308.2.4 template<typename P> typedef P mln::p_set< P >::i_element

Insertion element associated type.

Definition at line 113 of file p_set.hh.

10.308.2.5 `template<typename P> typedef fwd_piter mln::p_set< P >::piter`

[Site_iterator](#) associated type.

Definition at line 88 of file `p_set.hh`.

10.308.2.6 `template<typename P> typedef p_indexed_psite<self_> mln::p_set< P >::psite`

Psite associated type.

Definition at line 79 of file `p_set.hh`.

10.308.2.7 `template<typename P> typedef P mln::p_set< P >::r_element`

Removal element associated type.

Definition at line 119 of file `p_set.hh`.

10.308.3 Constructor & Destructor Documentation

10.308.3.1 `template<typename P> p_set< P >::p_set() [inline]`

Constructor.

Definition at line 152 of file `p_set.hh`.

10.308.4 Member Function Documentation

10.308.4.1 `template<typename P> void p_set< P >::clear() [inline]`

Clear this set.

Definition at line 219 of file `p_set.hh`.

10.308.4.2 `template<typename P> bool p_set< P >::has(const psite & p) const [inline]`

Test if psite `p` belongs to this point set.

Definition at line 167 of file `p_set.hh`.

10.308.4.3 `template<typename P> bool p_set< P >::has(const P & p) const [inline]`

Test if `p` belongs to this point set.

Definition at line 159 of file `p_set.hh`.

10.308.4.4 `template<typename P> bool p_set< P >::has(const util::index & i) const [inline]`

Test if index `i` belongs to this point set.

Definition at line 179 of file `p_set.hh`.

10.308.4.5 `template<typename P> void p_set< P >::insert(const P & p) [inline]`

Insert a site `p`.

Definition at line 203 of file p_set.hh.

Referenced by mln::convert::to_p_set().

10.308.4.6 `template<typename P> bool p_set<P>::is_valid () const [inline]`

Test this set validity so returns always true.

Definition at line 187 of file p_set.hh.

10.308.4.7 `template<typename P> std::size_t p_set<P>::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 236 of file p_set.hh.

10.308.4.8 `template<typename P> unsigned p_set<P>::nsites () const [inline]`

Give the number of sites.

Definition at line 195 of file p_set.hh.

Referenced by mln::p_key< K, P >::change_key(), and mln::p_key< K, P >::remove_key().

10.308.4.9 `template<typename P> const P & p_set<P>::operator[] (unsigned i) const [inline]`

Return the *i*-th site.

Definition at line 227 of file p_set.hh.

10.308.4.10 `template<typename P> void p_set<P>::remove (const P & p) [inline]`

Remove a site *p*.

Definition at line 211 of file p_set.hh.

10.308.4.11 `template<typename P> const std::vector<P> & p_set<P>::std_vector () const [inline]`

Return the corresponding std::vector of sites.

Definition at line 244 of file p_set.hh.

10.308.4.12 `template<typename P> const util::set<P> & p_set<P>::util_set () const [inline]`

Return the corresponding [util::set](#) of sites.

Definition at line 252 of file p_set.hh.

10.309 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, p_transformed< S, F > >, result, and check_t.

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](#) (const S &s, const F &f)
Constructor with a site set s and a predicate f .
- [p_transformed](#) ()
Constructor without argument.
- const S & [primary_set](#) () const
Return the primary set.

10.309.1 Detailed Description

```
template<typename S, typename F>class mln::p_transformed< S, F >
```

[Site](#) set transformed through a function.

Parameter S is a site set type; parameter F is a function from site to site.

Definition at line 83 of file `p_transformed.hh`.

10.309.2 Member Typedef Documentation

10.309.2.1 `template<typename S, typename F> typedef p_transformed_piter<typename S::bkd_piter, S, F> mln::p_transformed< S, F >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 102 of file `p_transformed.hh`.

10.309.2.2 `template<typename S, typename F> typedef S::element mln::p_transformed< S, F >::element`

Element associated type.

Definition at line 92 of file p_transformed.hh.

10.309.2.3 `template<typename S, typename F> typedef p_transformed_piter<typename S::fwd_piter, S, F> mln::p_transformed< S, F >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 99 of file p_transformed.hh.

10.309.2.4 `template<typename S, typename F> typedef fwd_piter mln::p_transformed< S, F >::piter`

[Site_Iterator](#) associated type.

Definition at line 105 of file p_transformed.hh.

10.309.2.5 `template<typename S, typename F> typedef S::psite mln::p_transformed< S, F >::psite`

Psite associated type.

Definition at line 96 of file p_transformed.hh.

10.309.3 Constructor & Destructor Documentation

10.309.3.1 `template<typename S, typename F> p_transformed< S, F >::p_transformed (const S & s, const F & f) [inline]`

Constructor with a site set *s* and a predicate *f*.

Definition at line 164 of file p_transformed.hh.

10.309.3.2 `template<typename S, typename F> p_transformed< S, F >::p_transformed () [inline]`

Constructor without argument.

Definition at line 158 of file p_transformed.hh.

10.309.4 Member Function Documentation

10.309.4.1 `template<typename S, typename F> const F & p_transformed< S, F >::function () const [inline]`

Return the transformation function.

Definition at line 207 of file p_transformed.hh.

10.309.4.2 `template<typename S, typename F> bool p_transformed< S, F >::has (const psite & p) const [inline]`

Test if *p* belongs to the subset.

Definition at line 173 of file p_transformed.hh.

10.309.4.3 `template<typename S, typename F> bool p_transformed< S, F >::is_valid () const` `[inline]`

Test if this site set is valid.

Definition at line 183 of file `p_transformed.hh`.

10.309.4.4 `template<typename S, typename F> std::size_t p_transformed< S, F >::memory_size () const`
`[inline]`

Return the size of this site set in memory.

Definition at line 191 of file `p_transformed.hh`.

10.309.4.5 `template<typename S, typename F> const S & p_transformed< S, F >::primary_set () const` `[inline]`

Return the primary set.

Definition at line 199 of file `p_transformed.hh`.

Referenced by `mln::p_transformed_piter< Pi, S, F >::change_target()`.

10.310 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< S, E >`.

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](#) ()
Constructor without argument.
- [p_transformed_piter](#) (const [p_transformed](#)< S, F > &s)
Constructor from a site set.

10.310.1 Detailed Description

`template<typename Pi, typename S, typename F> struct mln::p_transformed_piter< Pi, S, F >`

[Iterator](#) on `p_transformed<S,F>`.

Parameter *S* is a site set type; parameter *F* is a function from point to Boolean.

See Also

[mln::p_transformed](#)

Definition at line 50 of file `p_transformed_piter.hh`.

10.310.2 Constructor & Destructor Documentation

10.310.2.1 `template<typename Pi, typename S, typename F> p_transformed_piter< Pi, S, F >::p_transformed_piter () [inline]`

Constructor without argument.

Definition at line 93 of file p_transformed_piter.hh.

10.310.2.2 `template<typename Pi, typename S, typename F> p_transformed_piter< Pi, S, F >::p_transformed_piter (const p_transformed< S, F > & s) [inline]`

Constructor from a site set.

Definition at line 99 of file p_transformed_piter.hh.

10.310.3 Member Function Documentation

10.310.3.1 `template<typename Pi, typename S, typename F> void p_transformed_piter< Pi, S, F >::change_target (const p_transformed< S, F > & s) [inline]`

Change the set site targeted by this iterator.

Definition at line 143 of file p_transformed_piter.hh.

References `mln::p_transformed< S, F >::primary_set()`.

10.310.3.2 `template<typename E> void mln::Site_Iterator< E >::next () [inline], [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

Definition at line 92 of file site_iterator.hh.

10.311 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, p_vaccess< V, S > >`, and `mln::internal::site_set_impl< S >`.

Public Types

- `typedef p_double_piter< self_,
typename vset::bkd_viter,
typename S::bkd_piter > bkd_piter`

Backward [Site_Iterator](#) associated type.

- typedef S::element [element](#)
Element associated type.
- typedef p_double_piter< [self_](#),
typename vset::fwd_viter,
typename S::fwd_piter > [fwd_piter](#)
Forward [Site_Iterator](#) associated type.
- typedef std::pair< V, [element](#) > [i_element](#)
Insertion element associated type.
- typedef [fwd_piter](#) piter
[Site_Iterator](#) associated type.
- typedef S [pset](#)
Inner site set associated type.
- typedef p_double_psite< [self_](#), S > [psite](#)
Psite associated type.
- typedef V [value](#)
[Value](#) associated type.
- typedef [mln::value::set](#)< V > [vset](#)
[Value_Set](#) associated type.

Public Member Functions

- bool [has](#) (const [psite](#) &p) const
Test if p belongs to this site set.
- bool [has](#) (const V &v, const typename S::psite &p) const
Test if the couple (value v , psite p) belongs to this site set.
- void [insert](#) (const [i_element](#) &v_e)
Insert a pair v_e (value v , element e).
- void [insert](#) (const V &v, const [element](#) &e)
Insert e at value v .
- bool [is_valid](#) () const
Test if this site set is valid.
- std::size_t [memory_size](#) () const
Return the size of this site set in memory.
- const S & [operator\(\)](#) (const V &v) const
Return the site set at value v .
- [p_vaccess](#) ()
Constructor.
- const [mln::value::set](#)< V > & [values](#) () const
Give the set of values.

10.311.1 Detailed Description

template<typename V, typename S>class mln::p_vaccess< V, S >

[Site](#) set in which sites are grouped by their associated value.

Definition at line 70 of file p_vaccess.hh.

10.311.2 Member Typedef Documentation

10.311.2.1 `template<typename V, typename S > typedef p_double_piter<self_, typename vset ::bkd_viter, typename S ::bkd_piter> mln::p_vaccess< V, S >::bkd_piter`

Backward [Site_Iterator](#) associated type.

Definition at line 94 of file p_vaccess.hh.

10.311.2.2 `template<typename V, typename S > typedef S ::element mln::p_vaccess< V, S >::element`

Element associated type.

Definition at line 117 of file p_vaccess.hh.

10.311.2.3 `template<typename V, typename S > typedef p_double_piter<self_, typename vset ::fwd_viter, typename S ::fwd_piter> mln::p_vaccess< V, S >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 91 of file p_vaccess.hh.

10.311.2.4 `template<typename V, typename S > typedef std::pair<V, element> mln::p_vaccess< V, S >::i_element`

Insertion element associated type.

Definition at line 120 of file p_vaccess.hh.

10.311.2.5 `template<typename V, typename S > typedef fwd_piter mln::p_vaccess< V, S >::piter`

[Site_Iterator](#) associated type.

Definition at line 97 of file p_vaccess.hh.

10.311.2.6 `template<typename V, typename S > typedef S mln::p_vaccess< V, S >::pset`

Inner site set associated type.

Definition at line 85 of file p_vaccess.hh.

10.311.2.7 `template<typename V, typename S > typedef p_double_psite<self_, S> mln::p_vaccess< V, S >::psite`

Psite associated type.

Definition at line 88 of file p_vaccess.hh.

10.311.2.8 `template<typename V, typename S > typedef V mln::p_vaccess< V, S >::value`

[Value](#) associated type.

Definition at line 78 of file p_vaccess.hh.

10.311.2.9 `template<typename V, typename S > typedef mln::value::set<V> mln::p_vaccess< V, S >::vset`

Value_Set associated type.

Definition at line 81 of file p_vaccess.hh.

10.311.3 Constructor & Destructor Documentation

10.311.3.1 `template<typename V , typename S > p_vaccess< V, S >::p_vaccess () [inline]`

Constructor.

Definition at line 163 of file p_vaccess.hh.

10.311.4 Member Function Documentation

10.311.4.1 `template<typename V , typename S > bool p_vaccess< V, S >::has (const psite & p) const [inline]`

Test if *p* belongs to this site set.

Definition at line 180 of file p_vaccess.hh.

10.311.4.2 `template<typename V , typename S > bool p_vaccess< V, S >::has (const V & v, const typename S::psite & p) const [inline]`

Test if the couple (value *v*, psite *p*) belongs to this site set.

Definition at line 189 of file p_vaccess.hh.

10.311.4.3 `template<typename V , typename S > void p_vaccess< V, S >::insert (const i_element & v.e) [inline]`

Insert a pair *v_e* (value *v*, element *e*).

Definition at line 216 of file p_vaccess.hh.

10.311.4.4 `template<typename V , typename S > void p_vaccess< V, S >::insert (const V & v, const element & e) [inline]`

Insert *e* at value *v*.

Definition at line 206 of file p_vaccess.hh.

10.311.4.5 `template<typename V , typename S > bool p_vaccess< V, S >::is_valid () const [inline]`

Test if this site set is valid.

Definition at line 197 of file p_vaccess.hh.

10.311.4.6 `template<typename V , typename S > std::size_t p_vaccess< V, S >::memory_size () const [inline]`

Return the size of this site set in memory.

Definition at line 242 of file p_vaccess.hh.

10.311.4.7 `template<typename V , typename S > const S & p_vaccess< V, S >::operator() (const V & v) const [inline]`

Return the site set at value *v*.

Definition at line 234 of file p_vaccess.hh.

10.311.4.8 `template<typename V, typename S> const mln::value::set< V> & p_vaccess< V, S>::values () const`
`[inline]`

Give the set of values.

Definition at line 254 of file p_vaccess.hh.

10.312 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, p_vertices< G, F> >`.

Public Types

- typedef F [fun_t](#)
Function associated type.
- typedef `util::vertex< G>` [graph_element](#)
Type of graph element this site set focuses on.
- typedef G [graph_t](#)
Graph associated type.
- typedef `util::vertex< G>` [vertex](#)
Type of graph vertex.
- typedef `super_::site` [element](#)
Associated types.
- typedef `p_vertices_psite< G, F>` [psite](#)
Point_Site associated type.
- typedef `p_graph_piter< self_, mln_vertex_fwd_iter(G)>` [fwd_piter](#)
Forward Site_Iterator associated type.
- typedef `p_graph_piter< self_, mln_vertex_bkd_iter(G)>` [bkd_piter](#)
Backward Site_Iterator associated type.
- typedef `fwd_piter` [piter](#)
Site_Iterator associated type.

Public Member Functions

- bool [has](#) (const [psite](#) &p) const
Does this site set has p?
- template<typename G2>
 bool [has](#) (const `util::vertex< G2>` &v) const
Does this site set has v?
- void [invalidate](#) ()
Invalidate this site set.
- bool [is_valid](#) () const
Test this site set validity.
- std::size_t [memory_size](#) () const

Does this site set has vertex_id? FIXME: causes ambiguities while calling has(mln::neighb_fwd_niter<>); bool has(unsigned vertex_id) const;.

- unsigned [nsites](#) () const

Return The number of points (sites) of the set, i.e., the number of vertices.

- unsigned [nvertices](#) () const

Return The number of vertices in the graph.

- [p_vertices](#) ()

Constructor without argument.

- [p_vertices](#) (const [Graph](#)< G > &gr)

Construct a graph psite set from a graph of points.

- [p_vertices](#) (const [Graph](#)< G > &gr, const [Function](#)< F > &f)

Construct a graph psite set from a graph of points.

- template<typename F2 >

[p_vertices](#) (const [Graph](#)< G > &gr, const [Function](#)< F2 > &f)

Construct a graph psite set from a graph of points.

- template<typename F2 >

[p_vertices](#) (const [p_vertices](#)< G, F2 > &other)

Copy constructor.

- F::result [operator](#)() (const [psite](#) &p) const

Return the value associated to an element of this site set.

- const G & [graph](#) () const

Accessors.

- const F & [function](#) () const

Return the association function.

10.312.1 Detailed Description

```
template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>>class mln::p_vertices< G, F >
```

[Site](#) set based mapping graph vertices to sites.

Definition at line 71 of file p_vertices.hh.

10.312.2 Member Typedef Documentation

10.312.2.1 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef p_graph_piter< self_, mln_vertex_bkd_iter(G) > mln::p_vertices< G, F >::bkd_piter`

Backward [Site_iterator](#) associated type.

Definition at line 132 of file p_vertices.hh.

10.312.2.2 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G>>> typedef super_::site mln::p_vertices< G, F >::element`

Associated types.

Element associated type.

Definition at line 123 of file p_vertices.hh.

10.312.2.3 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> typedef F
mln::p_vertices< G, F >::fun_t`

[Function](#) associated type.

Definition at line 84 of file p_vertices.hh.

10.312.2.4 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> typedef p_graph_piter<
self_, mln_vertex_fwd_iter(G) > mln::p_vertices< G, F >::fwd_piter`

Forward [Site_Iterator](#) associated type.

Definition at line 129 of file p_vertices.hh.

10.312.2.5 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> typedef util::vertex<G>
mln::p_vertices< G, F >::graph_element`

Type of graph element this site set focuses on.

Definition at line 91 of file p_vertices.hh.

10.312.2.6 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> typedef G
mln::p_vertices< G, F >::graph_t`

[Graph](#) associated type.

Definition at line 81 of file p_vertices.hh.

10.312.2.7 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> typedef fwd_piter
mln::p_vertices< G, F >::piter`

[Site_Iterator](#) associated type.

Definition at line 135 of file p_vertices.hh.

10.312.2.8 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> typedef
p_vertices_psite<G,F> mln::p_vertices< G, F >::psite`

[Point_Site](#) associated type.

Definition at line 126 of file p_vertices.hh.

10.312.2.9 `template<typename G, typename F = util::internal::id2element<G,util::vertex<G> >> typedef util::vertex<G>
mln::p_vertices< G, F >::vertex`

Type of graph vertex.

Definition at line 87 of file p_vertices.hh.

10.312.3 Constructor & Destructor Documentation

10.312.3.1 `template<typename G , typename F > p_vertices< G, F >::p_vertices () [inline]`

Constructor without argument.

Definition at line 220 of file p_vertices.hh.

10.312.3.2 `template<typename G , typename F > p_vertices< G, F >::p_vertices (const Graph< G > & gr)`
`[inline]`

Construct a graph psite set from a graph of points.

Parameters

<i>gr</i>	The graph upon which the graph psite set is built. The identity function is used.
-----------	---

Definition at line 226 of file p_vertices.hh.

References mln::p_vertices< G, F >::is_valid().

10.312.3.3 `template<typename G , typename F > 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

<i>gr</i>	The graph upon which the graph psite set is built.
<i>f</i>	the function which maps a vertex to a site.

Definition at line 238 of file p_vertices.hh.

References mln::p_vertices< G, F >::is_valid().

10.312.3.4 `template<typename G , typename F > template<typename F2 > 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

<i>gr</i>	The graph upon which the graph psite set is built.
<i>f</i>	the function which maps a vertex to a site. It must be convertible to the function type F.

Definition at line 248 of file p_vertices.hh.

References mln::p_vertices< G, F >::is_valid().

10.312.3.5 `template<typename G , typename F > template<typename F2 > p_vertices< G, F >::p_vertices (const p_vertices< G, F2 > & other)` `[inline]`

Copy constructor.

Definition at line 260 of file p_vertices.hh.

References mln::p_vertices< G, F >::function(), mln::p_vertices< G, F >::graph(), and mln::p_vertices< G, F >::is_valid().

10.312.4 Member Function Documentation

10.312.4.1 `template<typename G , typename F > const F & p_vertices< G, F >::function () const` `[inline]`

Return the association function.

Definition at line 385 of file p_vertices.hh.

Referenced by mln::p_vertices< G, F >::p_vertices().

10.312.4.2 `template<typename G , typename F > const G & p_vertices< G, F >::graph () const [inline]`

Accessors.

Return the graph associated to this site set (const version)

Definition at line 376 of file p_vertices.hh.

References mln::p_vertices< G, F >::is_valid().

Referenced by mln::debug::draw_graph(), mln::operator==(), and mln::p_vertices< G, F >::p_vertices().

10.312.4.3 `template<typename G , typename F > bool p_vertices< G, F >::has (const psite & p) const [inline]`

Does this site set has p ?

Definition at line 304 of file p_vertices.hh.

References mln::p_vertices< G, F >::is_valid().

10.312.4.4 `template<typename G , typename F > template<typename G2 > bool p_vertices< G, F >::has (const util::vertex< G2 > & v) const [inline]`

Does this site set has v ?

Definition at line 314 of file p_vertices.hh.

References mln::util::vertex< G >::graph(), mln::util::vertex< G >::is_valid(), and mln::p_vertices< G, F >::is_valid().

10.312.4.5 `template<typename G , typename F > void p_vertices< G, F >::invalidate () [inline]`

Invalidate this site set.

Definition at line 296 of file p_vertices.hh.

10.312.4.6 `template<typename G , typename F > bool p_vertices< G, F >::is_valid () const [inline]`

Test this site set validity.

Definition at line 288 of file p_vertices.hh.

Referenced by mln::p_vertices< G, F >::graph(), mln::p_vertices< G, F >::has(), and mln::p_vertices< G, F >::p_vertices().

10.312.4.7 `template<typename G , typename F > std::size_t p_vertices< G, F >::memory_size () const [inline]`

Does this site set has *vertex_id*? FIXME: causes ambiguities while calling has(mln::neighb_fwd_niter<>); bool has(unsigned vertex_id) const;.

Definition at line 339 of file p_vertices.hh.

10.312.4.8 `template<typename G , typename F > unsigned p_vertices< G, F >::nsites () const [inline]`

Return The number of points (sites) of the set, i.e., the number of *vertices*.

Required by the mln::Point_Set concept.

Definition at line 272 of file p_vertices.hh.

References mln::p_vertices< G, F >::nvertices().

10.312.4.9 `template<typename G , typename F > unsigned p_vertices< G, F >::nvertices () const` `[inline]`

Return The number of vertices in the graph.

Definition at line 280 of file p_vertices.hh.

Referenced by mln::p_vertices< G, F >::nsites().

10.312.4.10 `template<typename G , typename F > F::result p_vertices< G, F >::operator() (const psite & p) const`
`[inline]`

Return the value associated to an element of this site set.

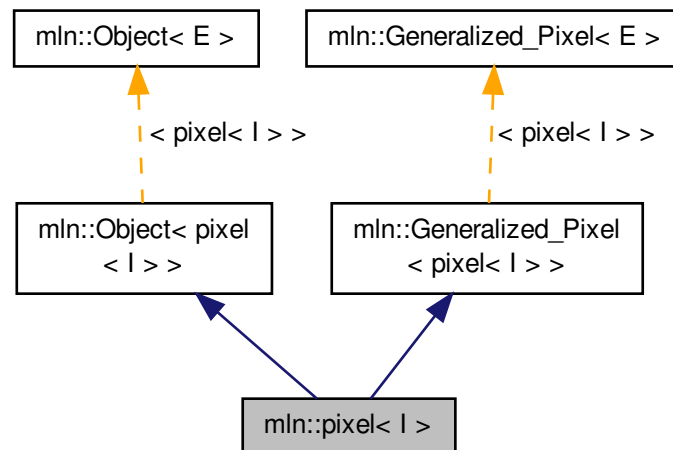
Definition at line 349 of file p_vertices.hh.

10.313 mln::pixel< I > Struct Template Reference

Generic pixel class.

```
#include <pixel.hh>
```

Inheritance diagram for mln::pixel< I >:



Public Member Functions

- void `change_to` (const typename I::psite &p)
Change the pixel to the one at point p.
- bool `is_valid` () const
Test if this pixel is valid.
- `pixel` (I &image)

Constructor.

- `pixel` (I &image, const typename I::psite &p)

Constructor.

10.313.1 Detailed Description

```
template<typename I>struct mln::pixel< I >
```

Generic pixel class.

The parameter is `I` the type of the image it belongs to.

Definition at line 50 of file core/pixel.hh.

10.313.2 Constructor & Destructor Documentation

10.313.2.1 `template<typename I> mln::pixel< I >::pixel (I & image)` `[inline]`

Constructor.

Definition at line 75 of file core/pixel.hh.

10.313.2.2 `template<typename I> mln::pixel< I >::pixel (I & image, const typename I::psite & p)` `[inline]`

Constructor.

Definition at line 82 of file core/pixel.hh.

References `mln::pixel< I >::change_to()`.

10.313.3 Member Function Documentation

10.313.3.1 `template<typename I> void mln::pixel< I >::change.to (const typename I::psite & p)` `[inline]`

Change the pixel to the one at point `p`.

Definition at line 92 of file core/pixel.hh.

Referenced by `mln::pixel< I >::pixel()`.

10.313.3.2 `template<typename I> bool mln::pixel< I >::is_valid () const` `[inline]`

Test if this pixel is valid.

Definition at line 101 of file core/pixel.hh.

10.314 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, plain< I > >`, and `check_t`.

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 `plain`< I > &rhs)
Assignment operator.
- `plain`< I > & `operator=` (const I &ima)
Assignment operator from an image ima.
- `plain` ()
Constructor without argument.
- `plain` (const `plain`< I > &rhs)
Copy constructor.
- `plain` (const I &ima)
Copy constructor from an image ima.

10.314.1 Detailed Description

`template<typename I>class mln::plain< I >`

Prevents an image from sharing its data.

While assigned to another image, its data is duplicated.

Definition at line 83 of file plain.hh.

10.314.2 Member Typedef Documentation

10.314.2.1 `template<typename I> typedef plain< tag::image_<I> > mln::plain< I >::skeleton`

Skeleton.

Definition at line 94 of file plain.hh.

10.314.3 Constructor & Destructor Documentation

10.314.3.1 `template<typename I> plain< I >::plain ()` `[inline]`

Constructor without argument.

Definition at line 142 of file plain.hh.

10.314.3.2 `template<typename I> plain< I >::plain (const plain< I > & rhs)` `[inline]`

Copy constructor.

Definition at line 148 of file plain.hh.

10.314.3.3 `template<typename I> plain<I>::plain (const I & ima) [inline]`

Copy constructor from an image `ima`.

Definition at line 157 of file `plain.hh`.

10.314.4 Member Function Documentation

10.314.4.1 `template<typename I> plain<I>::operator I () const [inline]`

Conversion into an image with type `I`.

Definition at line 198 of file `plain.hh`.

References `mln::duplicate()`.

10.314.4.2 `template<typename I> plain<I> & plain<I>::operator= (const plain<I> & rhs) [inline]`

Assignment operator.

Definition at line 175 of file `plain.hh`.

10.314.4.3 `template<typename I> plain<I> & plain<I>::operator= (const I & ima) [inline]`

Assignment operator from an image `ima`.

Definition at line 188 of file `plain.hh`.

10.315 mln::Point< P > Struct Template Reference

Base class for implementation of point classes.

```
#include <point.hh>
```

Inherits `mln::Point_Site< 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 negated 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.

10.315.1 Detailed Description

`template<typename P>struct mln::Point< P >`

Base class for implementation of point classes.

A point is an element of a space.

For instance, `mln::point2d` is the type of elements defined on the discrete square grid of the 2D plane.

Definition at line 62 of file `concept/point.hh`.

10.315.2 Member Typedef Documentation

10.315.2.1 `template<typename P > typedef P mln::Point< P >::point`

The associated point type is itself.

Definition at line 66 of file `concept/point.hh`.

10.315.3 Member Function Documentation

10.315.3.1 `template<typename P > const P & Point< P >::to_point () const` `[inline]`

It is a `Point` so it returns itself.

Definition at line 130 of file `concept/point.hh`.

10.315.4 Friends And Related Function Documentation

10.315.4.1 `template<typename P , typename D > P & operator+= (Point< P > & p, const Dpoint< D > & dp)`
`[related]`

Shift a point by a delta-point dp.

Parameters

<code>in, out</code>	<code>p</code>	The targeted point.
<code>in</code>	<code>dp</code>	A delta-point.

Returns

A reference to the point `p` once translated by `dp`.

Precondition

The type of `dp` has to be compatible with the type of `p`.

Definition at line 137 of file `concept/point.hh`.

10.315.4.2 `template<typename P , typename D > P & operator-= (Point< P > & p, const Dpoint< D > & dp)`
[related]

Shift a point by the negate of a delta-point dp.

Parameters

in, out	<i>p</i>	The targeted point.
in	<i>dp</i>	A delta-point.

Returns

A reference to the point *p* once translated by - dp.

Precondition

The type of *dp* has to be compatible with the type of *p*.

Definition at line 149 of file concept/point.hh.

10.315.4.3 `template<typename P , typename D > P & operator/ (Point< P > & p, const value::Scalar< D > & dp)`
[related]

Divise a point by a scalar s.

Parameters

in, out	<i>p</i>	The targeted point.
in	<i>dp</i>	A scalar.

Returns

A reference to the point *p* once divided by *s*.

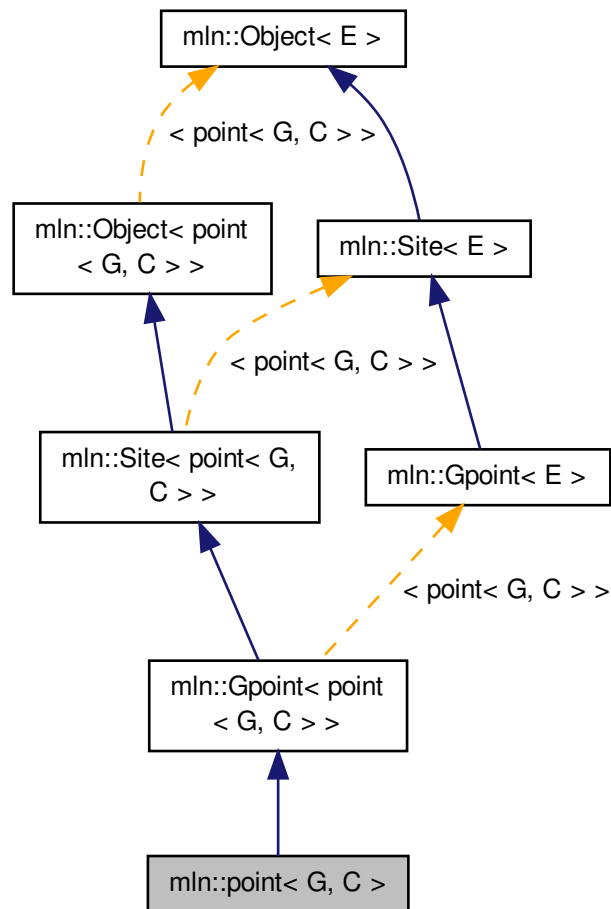
Definition at line 163 of file concept/point.hh.

10.316 mln::point< G, C > Struct Template Reference

Generic point class.

```
#include <point.hh>
```

Inheritance diagram for `mln::point< G, C >`:



Public Types

- enum { `dim` = `G::dim` }
- typedef `C coord`
Coordinate associated type.
- typedef `dpoint< G, C > delta`
Delta associated type.
- typedef `dpoint< G, C > dpsite`
DPSite associated type.
- typedef `G grid`
Grid associated type.
- typedef `mln::algebra::h_vec< G::dim, float > h_vec`
Algebra hexagonal vector (hvec) associated type.
- typedef `mln::algebra::vec< G::dim, float > vec`
Algebra vector (vec) associated type.

Public Member Functions

- const C & [last_coord](#) () const
Read-only access to the last coordinate.
- C & [last_coord](#) ()
Read-write access to the last coordinate.
- [point](#)< G, C > & [operator+=](#) (const [delta](#) &dp)
Shifting by dp .
- [point](#)< G, C > & [operator-=](#) (const [delta](#) &dp)
Shifting by the inverse of dp .
- const C & [operator\[\]](#) (unsigned i) const
Read-only access to the i -th coordinate value.
- C & [operator\[\]](#) (unsigned i)
Read-write access to the i -th coordinate value.
- [point](#) ()
Constructor without argument.
- template<typename C2 >
[point](#) (const mln::algebra::vec< [dim](#), C2 > &v)
Constructor from an algebra vector.
- template<typename F >
[point](#) (const [Function_v2v](#)< F > &f)
Constructor; coordinates are set by function f .
- void [set_all](#) (C c)
Set all coordinates to the value c .
- [h_vec to_h_vec](#) () const
Transform to point in homogeneous coordinate system.
- [vec to_vec](#) () const
Explicit conversion towards mln::algebra::vec.
- [point](#) (C ind)
- [point](#) (const [literal::origin_t](#) &)
Constructors/assignments with literals.

Static Public Member Functions

- static const [point](#)< G, C > & [minus_infty](#) ()
Point with all coordinates set to the minimum value.
- static const [point](#)< G, C > & [plus_infty](#) ()
Point with all coordinates set to the maximum value.

Static Public Attributes

- static const [point](#)< G, C > [origin](#) = [all_to](#)(0)
Origin point (all coordinates are 0).

10.316.1 Detailed Description

template<typename G, typename C>struct mln::point< G, C >

Generic point class.

Parameters are n the dimension of the space and C the coordinate type in this space.

Definition at line 108 of file point.hh.

10.316.2 Member Typedef Documentation

10.316.2.1 `template<typename G, typename C> typedef C mln::point< G, C >::coord`

Coordinate associated type.

Definition at line 131 of file point.hh.

10.316.2.2 `template<typename G, typename C> typedef dpoint<G,C> mln::point< G, C >::delta`

Delta associated type.

Definition at line 125 of file point.hh.

10.316.2.3 `template<typename G, typename C> typedef dpoint<G,C> mln::point< G, C >::dpsite`

DPsite associated type.

Definition at line 128 of file point.hh.

10.316.2.4 `template<typename G, typename C> typedef G mln::point< G, C >::grid`

Grid associated type.

Definition at line 122 of file point.hh.

10.316.2.5 `template<typename G, typename C> typedef mln::algebra::h_vec<G::dim, float> mln::point< G, C >::h_vec`

Algebra hexagonal vector (hvec) associated type.

Definition at line 137 of file point.hh.

10.316.2.6 `template<typename G, typename C> typedef mln::algebra::vec<G::dim, float> mln::point< G, C >::vec`

Algebra vector (vec) associated type.

Definition at line 134 of file point.hh.

10.316.3 Member Enumeration Documentation

10.316.3.1 `template<typename G, typename C> anonymous enum`

Enumerator

dim Dimension of the space.

Invariant

`dim > 0`

Definition at line 119 of file point.hh.

10.316.4 Constructor & Destructor Documentation

10.316.4.1 `template<typename G, typename C> point< G, C >::point () [inline]`

Constructor without argument.

Definition at line 420 of file point.hh.

10.316.4.2 `template<typename G , typename C > template<typename C2 > point< G, C >::point (const mln::algebra::vec< dim, C2 > & v) [inline]`

Constructor from an algebra vector.

Definition at line 427 of file point.hh.

10.316.4.3 `template<typename G , typename C> point< G, C >::point (C ind) [inline], [explicit]`

Constructors with different numbers of arguments (coordinates) w.r.t. the dimension.

Definition at line 443 of file point.hh.

10.316.4.4 `template<typename G , typename C> point< G, C >::point (const literal::origin_t &) [inline]`

Constructors/assignments with literals.

Definition at line 481 of file point.hh.

10.316.4.5 `template<typename G , typename C > template<typename F > point< G, C >::point (const Function_v2v< F > & f) [inline]`

Constructor; coordinates are set by function *f*.

Definition at line 471 of file point.hh.

10.316.5 Member Function Documentation

10.316.5.1 `template<typename G , typename C > const C & point< G, C >::last_coord () const [inline]`

Read-only access to the last coordinate.

Definition at line 402 of file point.hh.

Referenced by `mln::p_run< P >::end()`, `mln::p_run< P >::operator[]()`, and `mln::debug::put_word()`.

10.316.5.2 `template<typename G , typename C > C & point< G, C >::last_coord () [inline]`

Read-write access to the last coordinate.

Definition at line 410 of file point.hh.

10.316.5.3 `template<typename G , typename C > const point< G, C > & point< G, C >::minus_infty () [inline], [static]`

[Point](#) with all coordinates set to the minimum value.

Definition at line 627 of file point.hh.

10.316.5.4 `template<typename G , typename C > point< G, C > & point< G, C >::operator+= (const delta & dp) [inline]`

Shifting by *dp*.

Definition at line 544 of file point.hh.

10.316.5.5 `template<typename G , typename C > point< G, C > & point< G, C >::operator-= (const delta & dp)`
`[inline]`

Shifting by the inverse of dp.

Definition at line 554 of file point.hh.

10.316.5.6 `template<typename G , typename C > const C & point< G, C >::operator[] (unsigned i) const` `[inline]`

Read-only access to the *i*-th coordinate value.

Parameters

<i>in</i>	<i>i</i>	The coordinate index.
-----------	----------	-----------------------

Precondition

`i < dim`

Definition at line 385 of file point.hh.

10.316.5.7 `template<typename G , typename C > C & point< G, C >::operator[] (unsigned i)` `[inline]`

Read-write access to the *i*-th coordinate value.

Parameters

<i>in</i>	<i>i</i>	The coordinate index.
-----------	----------	-----------------------

Precondition

`i < dim`

Definition at line 393 of file point.hh.

10.316.5.8 `template<typename G , typename C > const point< G, C > & point< G, C >::plus_infty ()` `[inline]`,
`[static]`

[Point](#) with all coordinates set to the maximum value.

Definition at line 618 of file point.hh.

10.316.5.9 `template<typename G , typename C> void point< G, C >::set_all (C c)` `[inline]`

Set all coordinates to the value *c*.

Definition at line 533 of file point.hh.

10.316.5.10 `template<typename G , typename C > point< G, C >::h_vec point< G, C >::to_h_vec () const`
`[inline]`

Transform to point in homogeneous coordinate system.

Definition at line 592 of file point.hh.

10.316.5.11 `template<typename G, typename C> point< G, C >::vec point< G, C >::to_vec () const [inline]`

Explicit conversion towards mln::algebra::vec.

Definition at line 571 of file point.hh.

10.316.6 Member Data Documentation

10.316.6.1 `template<typename G, typename C> const point< G, C > point< G, C >::origin = all.to(0) [static]`

Origin point (all coordinates are 0).

Definition at line 192 of file point.hh.

10.317 mln::Proxy< E > Struct Template Reference

Base class for implementation classes of the notion of "proxy".

`#include <proxy.hh>`

Inherits [mln::Object< E >](#).

Inherited by [mln::Accumulator< E >](#), [mln::internal::graph_iter_base< G, Elt, E >](#), [mln::internal::nbh_iterator_base< G, C, Elt, E >](#), and [mln::Site_Proxy< E >](#).

10.317.1 Detailed Description

`template<typename E>struct mln::Proxy< E >`

Base class for implementation classes of the notion of "proxy".

Definition at line 232 of file core/concept/proxy.hh.

10.318 mln::Proxy< void > Struct Template Reference

[Proxy](#) category flag type.

`#include <proxy.hh>`

10.318.1 Detailed Description

`template<>struct mln::Proxy< void >`

[Proxy](#) category flag type.

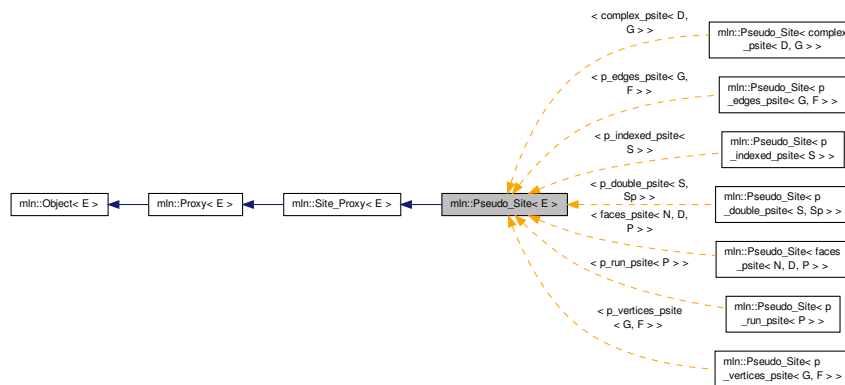
Definition at line 222 of file core/concept/proxy.hh.

10.319 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 >`:



10.319.1 Detailed Description

```
template<typename E>struct mln::Pseudo_Site< E >
```

Base class for implementation classes of the notion of "pseudo site".

FIXME: Explain...

Definition at line 64 of file `pseudo_site.hh`.

10.320 mln::Pseudo_Site< void > Struct Template Reference

[Pseudo_Site](#) category flag type.

```
#include <pseudo_site.hh>
```

10.320.1 Detailed Description

```
template<>struct mln::Pseudo_Site< void >
```

[Pseudo_Site](#) category flag type.

Definition at line 52 of file `pseudo_site.hh`.

10.321 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, image< F, S > >`.

Public Types

- typedef [image](#)< tag::function_
< F >, tag::domain_
S > > [skeleton](#)
Skeleton.

Public Member Functions

- [image](#) ()
Constructor without argument.
- [image](#) (const [Function_v2v](#)< F > &f, const [Site_Set](#)< S > &ps)
Constructor.

10.321.1 Detailed Description

```
template<typename F, typename S>class mln::pw::image< F, S >
```

A generic point-wise image implementation.

Parameter `F` is a function restricting the domain. Parameter `S` is the domain type.

Definition at line 92 of file pw/image.hh.

10.321.2 Member Typedef Documentation

10.321.2.1 `template<typename F, typename S> typedef image< tag::function_<F>, tag::domain_<S> > mln::pw::image< F, S >::skeleton`

Skeleton.

Definition at line 99 of file pw/image.hh.

10.321.3 Constructor & Destructor Documentation

10.321.3.1 `template<typename F, typename S> image< F, S >::image () [inline]`

Constructor without argument.

Definition at line 169 of file pw/image.hh.

10.321.3.2 `template<typename F, typename S> image< F, S >::image (const Function_v2v< F > &f, const Site_Set< S > &ps) [inline]`

Constructor.

Definition at line 175 of file pw/image.hh.

10.322 mln::registration::closest_point_basic< P > Class Template Reference

Closest point functor based on map distance.

```
#include <icp.hh>
```

10.322.1 Detailed Description

```
template<typename P>class mln::registration::closest_point_basic< P >
```

Closest point functor based on map distance.

Definition at line 240 of file icp.hh.

10.323 mln::registration::closest_point_with_map< P > Class Template Reference

Closest point functor based on map distance.

```
#include <icp.hh>
```

10.323.1 Detailed Description

```
template<typename P>class mln::registration::closest_point_with_map< P >
```

Closest point functor based on map distance.

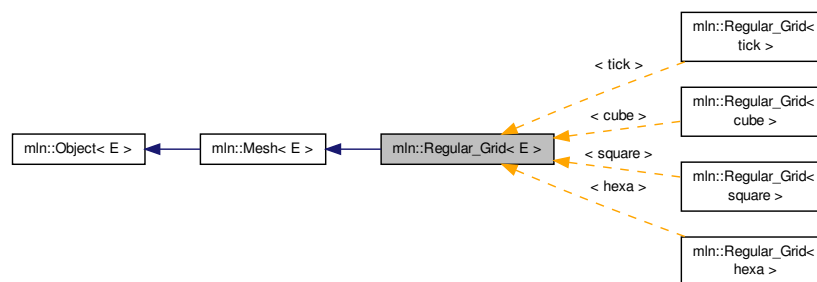
Definition at line 145 of file icp.hh.

10.324 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 >:



10.324.1 Detailed Description

```
template<typename E>struct mln::Regular_Grid< E >
```

Base class for implementation classes of regular grids.

Definition at line 42 of file regular_grid.hh.

10.325 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, 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.

10.325.1 Detailed Description

```
template<typename I>class mln::safe_image< I >
```

Makes an image accessible at undefined location.

Definition at line 84 of file safe.hh.

10.325.2 Member Typedef Documentation

```
10.325.2.1 template<typename I> typedef safe_image< tag::image_<I> > mln::safe_image< I >::skeleton
```

Skeleton.

Definition at line 89 of file safe.hh.

10.325.3 Member Function Documentation

```
10.325.3.1 template<typename I> safe_image< I >::operator safe_image< const I > ( ) const [inline]
```

Const promotion via conversion.

Definition at line 196 of file safe.hh.

10.326 mln::select::p_of< P > Struct Template Reference

Structure [p_of](#).

```
#include <pix.hh>
```

Inherits P.

10.326.1 Detailed Description

```
template<typename P>struct mln::select::p_of< P >
```

Structure [p_of](#).

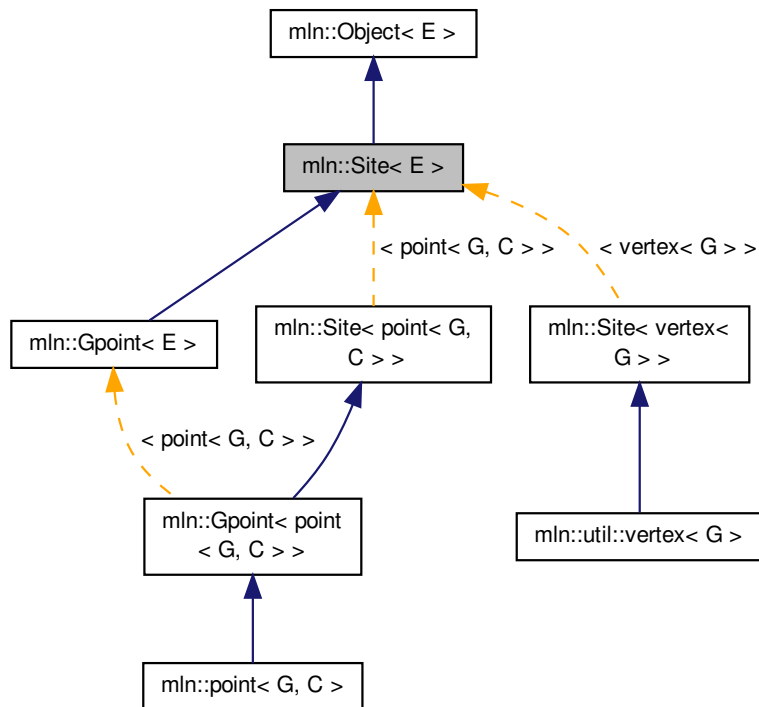
Definition at line 52 of file util/pix.hh.

10.327 mln::Site< E > Struct Template Reference

Base class for classes that are explicitly sites.

```
#include <site.hh>
```

Inheritance diagram for `mln::Site< E >`:



10.327.1 Detailed Description

```
template<typename E>struct mln::Site< E >
```

Base class for classes that are explicitly sites.

Definition at line 55 of file `site.hh`.

10.328 mln::Site< void > Struct Template Reference

[Site](#) category flag type.

```
#include <site.hh>
```

10.328.1 Detailed Description

```
template<>struct mln::Site< void >
```

[Site](#) category flag type.

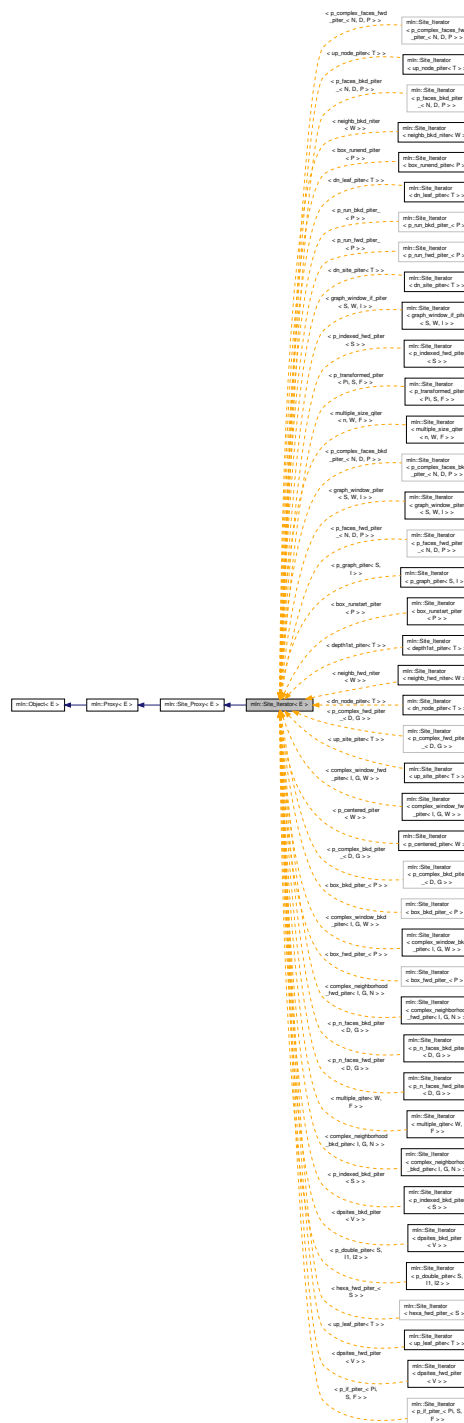
Definition at line 46 of file `site.hh`.

10.329 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.

10.329.1 Detailed Description

```
template<typename E>struct mln::Site_Iterator< E >
```

Base class for implementation of classes of iterator on points.

An iterator on points is an iterator that browse over a set of points.

See Also

[mln::doc::Site_Iterator](#) for a complete documentation of this class contents.

Definition at line 53 of file `site_iterator.hh`.

10.329.2 Member Function Documentation

10.329.2.1 `template<typename E > void mln::Site_Iterator< E >::next () [inline]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

Definition at line 92 of file `site_iterator.hh`.

10.330 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 >](#).

10.330.1 Detailed Description

```
template<typename E>struct mln::Site_Proxy< E >
```

Base class for implementation classes of the notion of "site proxy".

FIXME: Explain...

Definition at line 61 of file `site_proxy.hh`.

10.331 mln::Site_Proxy< void > Struct Template Reference

[Site_Proxy](#) category flag type.

```
#include <site_proxy.hh>
```

10.331.1 Detailed Description

```
template<> struct mln::Site_Proxy< void >
```

[Site_Proxy](#) category flag type.

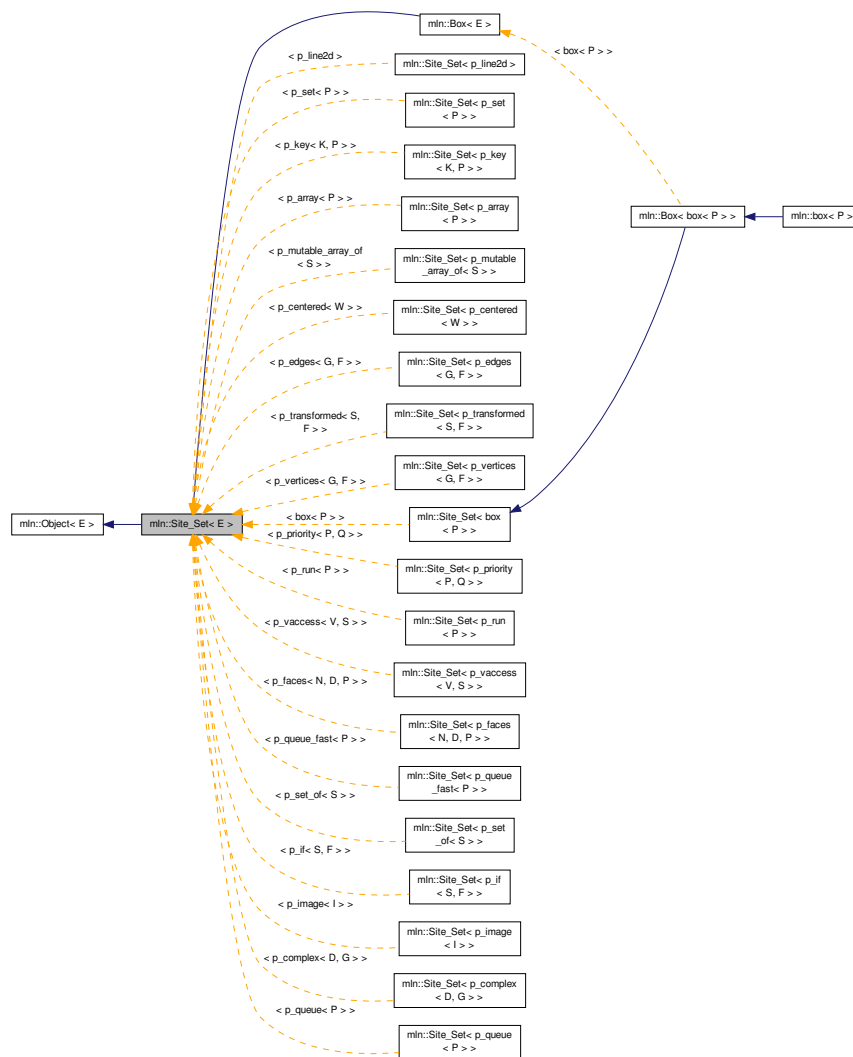
Definition at line 49 of file site_proxy.hh.

10.332 mln::Site_Set< E > Struct Template Reference

Base class for implementation classes of site sets.

```
#include <site_set.hh>
```

Inheritance diagram for `min::Site_Set< E >`:



Related Functions

(Note that these are not member functions.)

- template<typename SI, typename Sr>
`p_set< typename SI::site > diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic difference of lhs and rhs.
- template<typename SI, typename Sr>
`p_set< typename SI::site > inter (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Intersection between a couple of point sets.
- template<typename SI, typename Sr>
`bool operator< (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Strict inclusion test between site sets lhs and rhs.
- template<typename S>
`std::ostream & operator<< (std::ostream &ostr, const Site_Set< S > &set)`
Print a site set set into the output stream ostr.

- `template<typename SI , typename Sr >`
`bool operator<= (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Inclusion test between site sets lhs and rhs.
- `template<typename SI , typename Sr >`
`bool operator== (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Equality test between site sets lhs and rhs.
- `template<typename SI , typename Sr >`
`p_set< typename SI::site > sym_diff (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Set theoretic symmetrical difference of lhs and rhs.
- `template<typename SI , typename Sr >`
`p_set< typename SI::site > uni (const Site_Set< SI > &lhs, const Site_Set< Sr > &rhs)`
Union of a couple of point sets.
- `template<typename S >`
`p_set< typename S::site > unique (const Site_Set< S > &s)`
Give the unique set of s.

10.332.1 Detailed Description

`template<typename E>struct mln::Site_Set< E >`

Base class for implementation classes of site sets.

See Also

[mln::doc::Site_Set](#) for a complete documentation of this class contents.

Definition at line 65 of file `mln/core/concept/site_set.hh`.

10.332.2 Friends And Related Function Documentation

10.332.2.1 `template<typename SI , typename Sr > p_set< typename SI::site > diff (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [\[related\]](#)

Set theoretic difference of lhs and rhs.

Definition at line 66 of file `set/diff.hh`.

10.332.2.2 `template<typename SI , typename Sr > p_set< typename SI::site > inter (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [\[related\]](#)

Intersection between a couple of point sets.

Definition at line 62 of file `set/inter.hh`.

10.332.2.3 `template<typename SI , typename Sr > bool operator< (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [\[related\]](#)

Strict inclusion test between site sets lhs and rhs.

Parameters

<code>in</code>	<code>lhs</code>	A site set (strictly included?).
<code>in</code>	<code>rhs</code>	Another site set (includer?).

Definition at line 479 of file `operators.hh`.

10.332.2.4 `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

<code>in, out</code>	<code><i>ostr</i></code>	An output stream.
<code>in</code>	<code><i>set</i></code>	A site set.

Returns

The modified output stream `ostr`.

Definition at line 505 of file operators.hh.

10.332.2.5 `template<typename SI , typename Sr > bool operator<= (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [\[related\]](#)

Inclusion test between site sets `lhs` and `rhs`.

Parameters

<code>in</code>	<code><i>lhs</i></code>	A site set (included?).
<code>in</code>	<code><i>rhs</i></code>	Another site set (includer?).

Definition at line 491 of file operators.hh.

10.332.2.6 `template<typename SI , typename Sr > bool operator== (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [\[related\]](#)

Equality test between site sets `lhs` and `rhs`.

Parameters

<code>in</code>	<code><i>lhs</i></code>	A site set.
<code>in</code>	<code><i>rhs</i></code>	Another site set.

Definition at line 467 of file operators.hh.

10.332.2.7 `template<typename SI , typename Sr > p_set< typename SI::site > sym_diff (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [\[related\]](#)

Set theoretic symmetrical difference of `lhs` and `rhs`.

Definition at line 65 of file sym_diff.hh.

10.332.2.8 `template<typename SI , typename Sr > p_set< typename SI::site > uni (const Site_Set< SI > & lhs, const Site_Set< Sr > & rhs)` [\[related\]](#)

Union of a couple of point sets.

Definition at line 61 of file uni.hh.

10.332.2.9 `template<typename S > p_set< typename S::site > unique (const Site_Set< S > & s)` [related]

Give the unique set of `s`.

Definition at line 61 of file `unique.hh`.

10.333 mln::Site_Set< void > Struct Template Reference

[Site_Set](#) category flag type.

```
#include <site_set.hh>
```

10.333.1 Detailed Description

```
template<> struct mln::Site_Set< void >
```

[Site_Set](#) category flag type.

Definition at line 54 of file `mln/core/concept/site_set.hh`.

10.334 mln::sub_image< I, S > Class Template Reference

[Image](#) having its domain restricted by a site set.

```
#include <sub_image.hh>
```

Inherits `mln::internal::image_domain_morpher< I, S, sub_image< I, S > >`.

Public Types

- typedef [sub_image](#)< tag::image_
< I >, tag::domain_< S > > [skeleton](#)
Skeleton.

Public Member Functions

- const S & [domain](#) () const
Give the definition domain.
- [operator sub_image](#)< const I, S > () const
Const promotion via conversion.
- [sub_image](#) ()
Constructor without argument.
- [sub_image](#) (const I &ima, const S &pset)
Constructor.

10.334.1 Detailed Description

```
template<typename I, typename S> class mln::sub_image< I, S >
```

[Image](#) having its domain restricted by a site set.

Definition at line 102 of file `sub_image.hh`.

10.334.2 Member Typedef Documentation

10.334.2.1 `template<typename I, typename S> typedef sub_image< tag::image_<I>, tag::domain_<S> > mln::sub_image< I, S >::skeleton`

Skeleton.

Definition at line 108 of file sub_image.hh.

10.334.3 Constructor & Destructor Documentation

10.334.3.1 `template<typename I, typename S> sub_image< I, S >::sub_image () [inline]`

Constructor without argument.

Definition at line 182 of file sub_image.hh.

10.334.3.2 `template<typename I, typename S> sub_image< I, S >::sub_image (const I & ima, const S & pset) [inline]`

Constructor.

Definition at line 188 of file sub_image.hh.

10.334.4 Member Function Documentation

10.334.4.1 `template<typename I, typename S> const S & sub_image< I, S >::domain () const [inline]`

Give the definition domain.

Definition at line 205 of file sub_image.hh.

10.334.4.2 `template<typename I, typename S> sub_image< I, S >::operator sub_image< const I, S > () const [inline]`

Const promotion via conversion.

Definition at line 212 of file sub_image.hh.

10.335 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, p_if< S, fun::p2b::has< I > >, sub_image_if< I, S > >`.

Public Types

- typedef [sub_image_if](#)
`< tag::image_< I >
, tag::domain_< S > >` [skeleton](#)

Skeleton.

Public Member Functions

- const [p_if](#)< S, fun::p2b::has< I > > & [domain](#) () const
Give the definition domain.
- [sub_image_if](#) ()
Constructor without argument.
- [sub_image_if](#) (I & ima, const S & s)
Constructor.

10.335.1 Detailed Description

template<typename I, typename S>struct mln::sub_image_if< I, S >

[Image](#) having its domain restricted by a site set and a function.

Definition at line 102 of file sub_image_if.hh.

10.335.2 Member Typedef Documentation

10.335.2.1 template<typename I, typename S> typedef [sub_image_if](#)< tag::image_<I>, tag::domain_<S> > [mln::sub_image_if](#)< I, S >::skeleton

Skeleton.

Definition at line 107 of file sub_image_if.hh.

10.335.3 Constructor & Destructor Documentation

10.335.3.1 template<typename I, typename S> [sub_image_if](#)< I, S >::[sub_image_if](#) () [inline]

Constructor without argument.

Definition at line 182 of file sub_image_if.hh.

10.335.3.2 template<typename I, typename S> [sub_image_if](#)< I, S >::[sub_image_if](#) (I & ima, const S & s) [inline]

Constructor.

Definition at line 188 of file sub_image_if.hh.

10.335.4 Member Function Documentation

10.335.4.1 template<typename I, typename S> const [p_if](#)< S, fun::p2b::has< I > > & [sub_image_if](#)< I, S >::[domain](#) () const [inline]

Give the definition domain.

Definition at line 205 of file sub_image_if.hh.

10.336 mln::thru_image< I, F > Class Template Reference

Morph image values through a function.

```
#include <thru_image.hh>
```

Inherits `ret< I, F >`.

Public Member Functions

- [operator thru_image< const I, F > \(\) const](#)
Const promotion via conversion.

10.336.1 Detailed Description

```
template<typename I, typename F>class mln::thru_image< I, F >
```

Morph image values through a function.

Definition at line 156 of file `thru_image.hh`.

10.336.2 Member Function Documentation

10.336.2.1 `template<typename I, typename F> thru_image< I, F >::operator thru_image< const I, F > () const`
[inline]

Const promotion via conversion.

Definition at line 239 of file `thru_image.hh`.

10.337 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, 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.

10.337.1 Detailed Description

```
template<typename I1, typename I2, typename F>class mln::thrubin_image< I1, I2, F >
```

Morphes values from two images through a binary function.

Definition at line 82 of file thrubin_image.hh.

10.337.2 Member Typedef Documentation

10.337.2.1 `template<typename I1, typename I2, typename F> typedef I1 ::psite mln::thrubin_image< I1, I2, F >::psite`

Point_Site associated type.

Definition at line 94 of file thrubin_image.hh.

10.337.2.2 `template<typename I1, typename I2, typename F> typedef value mln::thrubin_image< I1, I2, F >::rvalue`

Return type of read-only access.

Definition at line 100 of file thrubin_image.hh.

10.337.2.3 `template<typename I1, typename I2, typename F> typedef thrubin_image<tag::image_<I1>, tag::image_<I2>, F> mln::thrubin_image< I1, I2, F >::skeleton`

Skeleton.

Definition at line 91 of file thrubin_image.hh.

10.337.2.4 `template<typename I1, typename I2, typename F> typedef F ::result mln::thrubin_image< I1, I2, F >::value`

[Value](#) associated type.

Definition at line 97 of file thrubin_image.hh.

10.337.3 Member Function Documentation

10.337.3.1 `template<typename I1 , typename I2, typename F> thrubin_image< I1, I2, F >::operator thrubin_image< const I1, const I2, F > () const [inline]`

Const promotion via conversion.

Definition at line 186 of file thrubin_image.hh.

10.338 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< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_bkd_iter< D > >, and mln::topo::internal::adj_higher_dim_connected_n_face_iterator< D >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_higher_dim_connected_n_face_bkd_iter](#) ()
Construction.

10.338.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

Definition at line 104 of file `adj_higher_dim_connected_n_face_iter.hh`.

10.338.2 Constructor & Destructor Documentation

10.338.2.1 `template<unsigned D> mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D >::adj_higher_dim_connected_n_face_bkd_iter () [inline]`

Construction.

Definition at line 196 of file `adj_higher_dim_connected_n_face_iter.hh`.

10.338.3 Member Function Documentation

10.338.3.1 `void mln::iterator< adj_higher_dim_connected_n_face_bkd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next* method.

Precondition

The iterator is valid.

10.339 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< topo::face< D >, algebraic_face< D >, adj_higher_dim_connected_n_face_fwd_iter< D > >`, and `mln::topo::internal::adj_higher_dim_connected_n_face_iterator< D >`.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_higher_dim_connected_n_face_fwd_iter](#) ()
Construction.

10.339.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

Definition at line 65 of file adj_higher_dim_connected_n_face_iter.hh.

10.339.2 Constructor & Destructor Documentation

10.339.2.1 template<unsigned D> mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D >::adj_higher_dim_connected_n_face_fwd_iter () [inline]

Construction.

Definition at line 162 of file adj_higher_dim_connected_n_face_iter.hh.

10.339.3 Member Function Documentation

10.339.3.1 void mln::iterator< adj_higher_dim_connected_n_face_fwd_iter< D > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.340 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< topo::face< D >, algebraic_face< D >, adj_higher_face_bkd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_higher_face_bkd_iter](#) ()
Construction.

10.340.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

Definition at line 106 of file `adj_higher_face_iter.hh`.

10.340.2 Constructor & Destructor Documentation

10.340.2.1 `template<unsigned D> mln::topo::adj_higher_face_bkd_iter< D >::adj_higher_face_bkd_iter ()`
[`inline`]

Construction.

Definition at line 167 of file `adj_higher_face_iter.hh`.

10.340.3 Member Function Documentation

10.340.3.1 `void mln::iterator< adj_higher_face_bkd_iter< D > >::next ()` [`inherited`]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.341 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< topo::face< D >, algebraic_face< D >, adj_higher_face_fwd_iter< D > >`.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_higher_face_fwd_iter](#) ()
Construction.

10.341.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

Definition at line 72 of file adj_higher_face_iter.hh.

10.341.2 Constructor & Destructor Documentation

10.341.2.1 template<unsigned D> mln::topo::adj_higher_face_fwd_iter< D >::adj_higher_face_fwd_iter ()
[inline]

Construction.

Definition at line 139 of file adj_higher_face_iter.hh.

10.341.3 Member Function Documentation

10.341.3.1 void mln::iterator< adj_higher_face_fwd_iter< D > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next*_method.

Precondition

The iterator is valid.

10.342 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< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_bkd_iter< D > >, and mln::topo::internal::adj_lower_dim_connected_n_face_iterator< D >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_lower_dim_connected_n_face_bkd_iter](#) ()
Construction.

10.342.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

Definition at line 104 of file `adj_lower_dim_connected_n_face_iter.hh`.

10.342.2 Constructor & Destructor Documentation

```
10.342.2.1 template<unsigned D> mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D
>::adj_lower_dim_connected_n_face_bkd_iter( ) [inline]
```

Construction.

Definition at line 196 of file `adj_lower_dim_connected_n_face_iter.hh`.

10.342.3 Member Function Documentation

```
10.342.3.1 void mln::iterator< adj_lower_dim_connected_n_face_bkd_iter< D > >::next( ) [inherited]
```

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next* method.

Precondition

The iterator is valid.

10.343 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< topo::face< D >, algebraic_face< D >, adj_lower_dim_connected_n_face_fwd_iter< D > >`, and `mln::topo::internal::adj_lower_dim_connected_n_face_iterator< D >`.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_lower_dim_connected_n_face_fwd_iter](#) ()
Construction.

10.343.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

Definition at line 65 of file adj_lower_dim_connected_n_face_iter.hh.

10.343.2 Constructor & Destructor Documentation

10.343.2.1 template<unsigned D> mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D
>::adj_lower_dim_connected_n_face_fwd_iter() [inline]

Construction.

Definition at line 162 of file adj_lower_dim_connected_n_face_iter.hh.

10.343.3 Member Function Documentation

10.343.3.1 void mln::iterator< adj_lower_dim_connected_n_face_fwd_iter< D > >::next() [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.344 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< topo::face< D >, algebraic_face< D >, adj_lower_face_bkd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_lower_face_bkd_iter](#) ()
Construction.

10.344.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

Definition at line 108 of file `adj_lower_face_iter.hh`.

10.344.2 Constructor & Destructor Documentation

10.344.2.1 `template<unsigned D> mln::topo::adj_lower_face_bkd_iter< D >::adj_lower_face_bkd_iter ()`
[`inline`]

Construction.

Definition at line 169 of file `adj_lower_face_iter.hh`.

10.344.3 Member Function Documentation

10.344.3.1 `void mln::iterator< adj_lower_face_bkd_iter< D > >::next ()` [`inherited`]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.345 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< topo::face< D >, algebraic_face< D >, adj_lower_face_fwd_iter< D > >`.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_lower_face_fwd_iter](#) ()
Construction.

10.345.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

Definition at line 73 of file adj_lower_face_iter.hh.

10.345.2 Constructor & Destructor Documentation

10.345.2.1 template<unsigned D> mln::topo::adj_lower_face_fwd_iter< D >::adj_lower_face_fwd_iter ()
[inline]

Construction.

Definition at line 141 of file adj_lower_face_iter.hh.

10.345.3 Member Function Documentation

10.345.3.1 void mln::iterator< adj_lower_face_fwd_iter< D > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.346 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< adj_higher_face_bkd_iter< D >, adj_lower_face_bkd_iter< D >, adj_lower_higher_face_bkd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_lower_higher_face_bkd_iter](#) ()
Construction.

10.346.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

Definition at line 102 of file `adj_lower_higher_face_iter.hh`.

10.346.2 Constructor & Destructor Documentation

10.346.2.1 `template<unsigned D> mln::topo::adj_lower_higher_face_bkd_iter< D >::adj_lower_higher_face_bkd_iter () [inline]`

Construction.

Definition at line 152 of file `adj_lower_higher_face_iter.hh`.

10.346.3 Member Function Documentation

10.346.3.1 `void mln::iterator< adj_lower_higher_face_bkd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.347 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< adj_lower_face_fwd_iter< D >, adj_higher_face_fwd_iter< D >, adj_lower_higher_face_fwd_iter< D > >`.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [adj_lower_higher_face_fwd_iter](#) ()
Construction.

10.347.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

Definition at line 71 of file adj_lower_higher_face_iter.hh.

10.347.2 Constructor & Destructor Documentation

10.347.2.1 template<unsigned D> mln::topo::adj_lower_higher_face_fwd_iter< D
>::adj_lower_higher_face_fwd_iter () [inline]

Construction.

Definition at line 133 of file adj_lower_higher_face_iter.hh.

10.347.3 Member Function Documentation

10.347.3.1 void mln::iterator< adj_lower_higher_face_fwd_iter< D > >::next () [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.348 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< topo::face< D >, algebraic_face< D >, adj_m_face_bkd_iter< D > >, and mln::topo::internal::adj_m_face_iterator< D >.

Public Member Functions

- void `next` ()
Go to the next element.
- `adj_m_face_bkd_iter` ()
Construction.
- `template<typename Fref > adj_m_face_bkd_iter` (const Fref &f_ref, unsigned m)
Constructs an iterator, with f_ref as reference face, and a target dimension equal to m.

10.348.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

The dimension parameter (*m_*) must be lower or equal to *D*.

If *m_* is equal to the dimension of the reference face, then the iterated set is empty.

Definition at line 118 of file `adj_m_face_iter.hh`.

10.348.2 Constructor & Destructor Documentation

10.348.2.1 `template<unsigned D> mln::topo::adj_m_face_bkd_iter< D >::adj_m_face_bkd_iter ()`
[inline]

Construction.

Construct an iterator, with an invalid reference face, and a target dimension equal to 0.

Definition at line 223 of file `adj_m_face_iter.hh`.

10.348.2.2 `template<unsigned D> template<typename Fref > mln::topo::adj_m_face_bkd_iter< D >::adj_m_face_bkd_iter (const Fref & f_ref, unsigned m)` [inline]

Constructs an iterator, with *f_ref* as reference face, and a target dimension equal to *m*.

Definition at line 230 of file `adj_m_face_iter.hh`.

10.348.3 Member Function Documentation

10.348.3.1 `void mln::iterator< adj_m_face_bkd_iter< D > >::next ()` [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.349 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< topo::face< D >, algebraic_face< D >, adj_m_face_fwd_iter< D > >, and mln::topo::internal::adj_m_face_iterator< D >.

Public Member Functions

- void [next](#) ()

Go to the next element.

- [adj_m_face_fwd_iter](#) ()

Construction.

- template<typename Fref >

[adj_m_face_fwd_iter](#) (const Fref &f_ref, unsigned m)

Constructs an iterator, with f_ref as reference face, and a target dimension equal to m.

10.349.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

The dimension parameter (*m_*) must be lower or equal to *D*.

If *m_* is equal to the dimension of the reference face, then the iterated set is empty.

Definition at line 70 of file adj_m_face_iter.hh.

10.349.2 Constructor & Destructor Documentation

10.349.2.1 template<unsigned D> mln::topo::adj_m_face_fwd_iter< D >::adj_m_face_fwd_iter ()
[inline]

Construction.

Construct an iterator, with an invalid reference face, and a target dimension equal to 0.

Definition at line 194 of file adj_m_face_iter.hh.

10.349.2.2 template<unsigned D> template<typename Fref > mln::topo::adj_m_face_fwd_iter< D >::adj_m_face_fwd_iter (const Fref &f_ref, unsigned m) [inline]

Constructs an iterator, with *f_ref* as reference face, and a target dimension equal to *m*.

Definition at line 201 of file adj_m_face_iter.hh.

10.349.3 Member Function Documentation

10.349.3.1 `void mln::iterator< adj_m_face_fwd_iter< D > >::next ()` [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

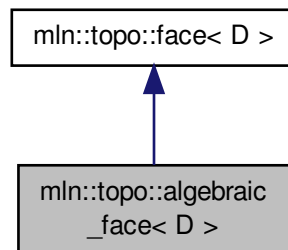
The iterator is valid.

10.350 mln::topo::algebraic_face< D > Class Template Reference

Algebraic face handle in a complex; the face dimension is dynamic.

```
#include <algebraic_face.hh>
```

Inheritance diagram for `mln::topo::algebraic_face< D >`:



Public Member Functions

- [algebraic_face](#) ()
Build a non-initialized algebraic face handle.
- [algebraic_face](#) ([complex](#)< D > &[complex](#), unsigned [n](#), unsigned [face_id](#), bool [sign](#))
Build an algebraic face handle from complex and face_id.
- [algebraic_face](#) (const [face](#)< D > &[f](#), bool [sign](#))
Build an algebraic face handle from an mln::face.
- [template](#)<unsigned N>
[algebraic_face](#) (const [algebraic_n_face](#)< N, D > &[f](#))
Build a face handle from an mln::topo::algebraic_n_face.
- void [invalidate](#) ()
Invalidate this handle.
- bool [is_valid](#) () const
Is this handle valid?

- bool [sign](#) () const
Accessors.
- void [set_sign](#) (bool [sign](#))
Set the sign of this face.
- [complex](#)< D > [cplx](#) () const
Accessors.
- unsigned [n](#) () const
Return the dimension of the face.
- unsigned [face_id](#) () const
Return the id of the face.
- void [set_cplx](#) (const [complex](#)< D > &[cplx](#))
Set the complex the face belongs to.
- void [set_n](#) (unsigned [n](#))
Set the dimension of the face.
- void [inc_n](#) ()
Increment the dimension of the face.
- void [dec_n](#) ()
Decrement the dimension of the face.
- void [set_face_id](#) (unsigned [face_id](#))
Set the id of the face.
- void [inc_face_id](#) ()
Increment the id of the face.
- void [dec_face_id](#) ()
Decrement the id of the face.
- template<unsigned N>
[face_data](#)< N, D > & [data](#) () const
Return the mln::topo::face_data pointed by this handle.
- std::vector< [algebraic_face](#)< D > > [lower_dim_adj_faces](#) () const
Return an array of face handles pointing to adjacent (n-1)-faces.
- std::vector< [algebraic_face](#)< D > > [higher_dim_adj_faces](#) () const
Return an array of face handles pointing to adjacent (n+1)-faces.

10.350.1 Detailed Description

```
template<unsigned D>class mln::topo::algebraic_face< D >
```

Algebraic face handle in a complex; the face dimension is dynamic.

Contrary to an [mln::topo::algebraic_n_face](#), the dimension of an [mln::topo::algebraic_face](#) is not fixed.

Definition at line 60 of file [algebraic_face.hh](#).

10.350.2 Constructor & Destructor Documentation

10.350.2.1 `template<unsigned D> algebraic_face< D >::algebraic_face () [inline]`

Build a non-initialized algebraic face handle.

Definition at line 157 of file [algebraic_face.hh](#).

10.350.2.2 `template<unsigned D> algebraic_face< D >::algebraic_face (complex< D > & complex, unsigned n, unsigned face_id, bool sign) [inline]`

Build an algebraic face handle from *complex* and *face_id*.

Definition at line 164 of file `algebraic_face.hh`.

10.350.2.3 `template<unsigned D> algebraic_face< D >::algebraic_face (const face< D > & f, bool sign) [inline]`

Build an algebraic face handle from an `mln::face`.

Definition at line 174 of file `algebraic_face.hh`.

References `mln::topo::face< D >::n()`.

10.350.2.4 `template<unsigned D> template<unsigned N> algebraic_face< D >::algebraic_face (const algebraic_n_face< N, D > & f) [inline]`

Build a face handle from an `mln::topo::algebraic_n_face`.

Definition at line 184 of file `algebraic_face.hh`.

10.350.3 Member Function Documentation

10.350.3.1 `template<unsigned D> complex< D > face< D >::cplx () const [inline],[inherited]`

Accessors.

Return the complex the face belongs to.

Definition at line 224 of file `face.hh`.

Referenced by `mln::complex_psite< D, G >::complex_psite()`, `mln::topo::operator!=()`, and `mln::topo::operator==()`.

10.350.3.2 `template<unsigned D> template<unsigned N> face_data< N, D > & face< D >::data () const [inline],[inherited]`

Return the `mln::topo::face_data` pointed by this handle.

Definition at line 305 of file `face.hh`.

10.350.3.3 `template<unsigned D> void face< D >::dec_face_id () [inline],[inherited]`

Decrement the id of the face.

Definition at line 296 of file `face.hh`.

10.350.3.4 `template<unsigned D> void face< D >::dec_n () [inline],[inherited]`

Decrement the dimension of the face.

Definition at line 272 of file `face.hh`.

10.350.3.5 `template<unsigned D> unsigned face< D >::face_id () const [inline],[inherited]`

Return the id of the face.

Definition at line 240 of file face.hh.

Referenced by mln::geom::complex_geometry< D, P >::operator>(), and mln::topo::operator==().

10.350.3.6 `template<unsigned D> std::vector< algebraic_face< D > > face< D >::higher_dim_adj_faces () const`
`[inline], [inherited]`

Return an array of face handles pointing to adjacent (n+1)-faces.

Definition at line 370 of file face.hh.

10.350.3.7 `template<unsigned D> void face< D >::inc_face_id () [inline], [inherited]`

Increment the id of the face.

Definition at line 288 of file face.hh.

10.350.3.8 `template<unsigned D> void face< D >::inc_n () [inline], [inherited]`

Increment the dimension of the face.

Definition at line 264 of file face.hh.

10.350.3.9 `template<unsigned D> void face< D >::invalidate () [inline], [inherited]`

Invalidate this handle.

Definition at line 215 of file face.hh.

10.350.3.10 `template<unsigned D> bool face< D >::is_valid () const [inline], [inherited]`

Is this handle valid?

Definition at line 207 of file face.hh.

10.350.3.11 `template<unsigned D> std::vector< algebraic_face< D > > face< D >::lower_dim_adj_faces () const`
`[inline], [inherited]`

Return an array of face handles pointing to adjacent (n-1)-faces.

Definition at line 357 of file face.hh.

10.350.3.12 `template<unsigned D> unsigned face< D >::n () const [inline], [inherited]`

Return the dimension of the face.

Definition at line 232 of file face.hh.

Referenced by mln::topo::algebraic_face< D >::algebraic_face(), mln::geom::complex_geometry< D, P >::operator>(), and mln::topo::operator==().

10.350.3.13 `template<unsigned D> void face< D >::set_cplx (const complex< D > & cplx) [inline],`
`[inherited]`

Set the complex the face belongs to.

Definition at line 248 of file face.hh.

10.350.3.14 `template<unsigned D> void face< D >::set_face_id (unsigned face_id)` `[inline],[inherited]`

Set the id of the face.

Definition at line 280 of file face.hh.

10.350.3.15 `template<unsigned D> void face< D >::set_n (unsigned n)` `[inline],[inherited]`

Set the dimension of the face.

Definition at line 256 of file face.hh.

10.350.3.16 `template<unsigned D> void algebraic_face< D >::set_sign (bool sign)` `[inline]`

Set the sign of this face.

Definition at line 203 of file algebraic_face.hh.

10.350.3.17 `template<unsigned D> bool algebraic_face< D >::sign () const` `[inline]`

Accessors.

Return the sign of this face.

Definition at line 195 of file algebraic_face.hh.

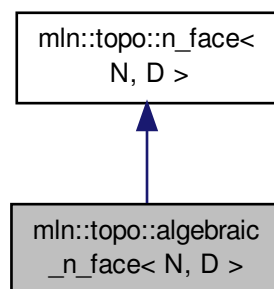
Referenced by `mln::topo::operator==()`.

10.351 `mln::topo::algebraic_n_face< N, D >` Class Template Reference

Algebraic *N*-face handle in a complex.

`#include <algebraic_n_face.hh>`

Inheritance diagram for `mln::topo::algebraic_n_face< N, D >`:



Public Member Functions

- [`algebraic_n_face \(\)`](#)
Build a non-initialized algebraic face handle.

- [algebraic_n_face](#) ([complex](#)< D > &[complex](#), unsigned [face_id](#), bool [sign](#))

Build an algebraic face handle from complex and face_id.

- [algebraic_n_face](#) (const [n_face](#)< N, D > &[f](#), bool [sign](#))

Build an algebraic face handle from an mln::n_face.

- void [invalidate](#) ()

Invalidate this handle.

- bool [is_valid](#) () const

Is this handle valid?

- bool [sign](#) () const

Accessors.

- void [set_sign](#) (bool [sign](#))

Set the sign of this face.

- [complex](#)< D > [cplx](#) () const

Accessors.

- unsigned [face_id](#) () const

Return the id of the face.

- void [set_cplx](#) (const [complex](#)< D > &[cplx](#))

Set the complex the face belongs to.

- unsigned [n](#) () const

Return the dimension of the face.

- void [set_face_id](#) (unsigned [face_id](#))

Set the id of the face.

- void [inc_face_id](#) ()

Increment the id of the face.

- void [dec_face_id](#) ()

Decrement the id of the face.

- [face_data](#)< N, D > &[data](#) () const

Return the mln::topo::face_data pointed by this handle.

- std::vector< [algebraic_n_face](#)
< N-1, D > > [lower_dim_adj_faces](#) () const

Return an array of face handles pointing to adjacent (n-1)-faces.

- std::vector< [algebraic_n_face](#)
< N+1, D > > [higher_dim_adj_faces](#) () const

Return an array of face handles pointing to adjacent (n+1)-faces.

10.351.1 Detailed Description

```
template<unsigned N, unsigned D>class mln::topo::algebraic_n_face< N, D >
```

Algebraic N-face handle in a complex.

Contrary to an [mln::topo::algebraic_face](#), the dimension of an [mln::topo::algebraic_n_face](#) is fixed.

Definition at line 50 of file [algebraic_n_face.hh](#).

10.351.2 Constructor & Destructor Documentation

10.351.2.1 `template<unsigned N, unsigned D> mln::topo::algebraic_n_face< N, D >::algebraic_n_face ()`
[inline]

Build a non-initialized algebraic face handle.

Definition at line 166 of file algebraic_n_face.hh.

References `mln::topo::n_face< N, D >::is_valid()`.

10.351.2.2 `template<unsigned N, unsigned D> mln::topo::algebraic_n_face< N, D >::algebraic_n_face (`
`complex< D > & complex, unsigned face_id, bool sign)` [inline]

Build an algebraic face handle from *complex* and *face_id*.

Definition at line 176 of file algebraic_n_face.hh.

10.351.2.3 `template<unsigned N, unsigned D> mln::topo::algebraic_n_face< N, D >::algebraic_n_face (const`
`n_face< N, D > & f, bool sign)` [inline]

Build an algebraic face handle from an `mln::n_face`.

Definition at line 186 of file algebraic_n_face.hh.

10.351.3 Member Function Documentation

10.351.3.1 `template<unsigned N, unsigned D> complex< D > n_face< N, D >::cplx () const` [inline],
[inherited]

Accessors.

Return the complex the face belongs to.

Definition at line 195 of file n_face.hh.

Referenced by `mln::topo::n_faces_set< N, D >::add()`, `mln::topo::operator!=()`, and `mln::topo::operator==()`.

10.351.3.2 `template<unsigned N, unsigned D> face_data< N, D > & n_face< N, D >::data () const` [inline],
[inherited]

Return the `mln::topo::face_data` pointed by this handle.

Definition at line 251 of file n_face.hh.

10.351.3.3 `template<unsigned N, unsigned D> void n_face< N, D >::dec_face_id ()` [inline], [inherited]

Decrement the id of the face.

Definition at line 243 of file n_face.hh.

10.351.3.4 `template<unsigned N, unsigned D> unsigned n_face< N, D >::face_id () const` [inline],
[inherited]

Return the id of the face.

Definition at line 211 of file n_face.hh.

Referenced by `mln::topo::operator==()`.

10.351.3.5 `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N+1, D > > n_face< N, D >::higher_dim_adj_faces () const [inline],[inherited]`

Return an array of face handles pointing to adjacent (n+1)-faces.

Definition at line 270 of file n_face.hh.

Referenced by mln::topo::edge().

10.351.3.6 `template<unsigned N, unsigned D> void n_face< N, D >::inc_face_id () [inline],[inherited]`

Increment the id of the face.

Definition at line 235 of file n_face.hh.

10.351.3.7 `template<unsigned N, unsigned D> void n_face< N, D >::invalidate () [inline],[inherited]`

Invalidate this handle.

Definition at line 187 of file n_face.hh.

10.351.3.8 `template<unsigned N, unsigned D> bool n_face< N, D >::is_valid () const [inline],[inherited]`

Is this handle valid?

Definition at line 179 of file n_face.hh.

Referenced by mln::topo::algebraic_n_face< N, D >::algebraic_n_face(), and mln::topo::n_face< N, D >::n_face().

10.351.3.9 `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N-1, D > > n_face< N, D >::lower_dim_adj_faces () const [inline],[inherited]`

Return an array of face handles pointing to adjacent (n-1)-faces.

Definition at line 260 of file n_face.hh.

10.351.3.10 `template<unsigned N, unsigned D> unsigned n_face< N, D >::n () const [inline],[inherited]`

Return the dimension of the face.

Definition at line 203 of file n_face.hh.

10.351.3.11 `template<unsigned N, unsigned D> void n_face< N, D >::set_cplx (const complex< D > & cplx) [inline],[inherited]`

Set the complex the face belongs to.

Definition at line 219 of file n_face.hh.

10.351.3.12 `template<unsigned N, unsigned D> void n_face< N, D >::set_face_id (unsigned face_id) [inline],[inherited]`

Set the id of the face.

Definition at line 227 of file n_face.hh.

10.351.3.13 `template<unsigned N, unsigned D> void mln::topo::algebraic_n_face< N, D >::set_sign (bool sign)`
`[inline]`

Set the sign of this face.

Definition at line 205 of file `algebraic_n_face.hh`.

10.351.3.14 `template<unsigned N, unsigned D> bool mln::topo::algebraic_n_face< N, D >::sign () const`
`[inline]`

Accessors.

Return the sign of this face.

Definition at line 197 of file `algebraic_n_face.hh`.

Referenced by `mln::topo::operator==()`.

10.352 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< topo::face< D >, algebraic_face< D >, center_only_iter< D > >`.

Public Member Functions

- `void next ()`
Go to the next element.
- `center_only_iter ()`
Construction.

10.352.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

`mln::topo::center_only_iter` inherits from `mln::topo::internal::forward_complex_relative_iterator_base`, but it could inherit from `mln::topo::internal::backward_complex_relative_iterator_base` as well, since it always contains a single element, the center/reference face (and the traversal order is meaningless).

This iterator is essentially used to implement other iterators.

See Also

`mln::topo::centered_iter_adapter`
`mln::complex_lower_window`
`mln::complex_higher_window`
`mln::complex_lower_higher_window`

Definition at line 74 of file center_only_iter.hh.

10.352.2 Constructor & Destructor Documentation

10.352.2.1 `template<unsigned D> mln::topo::center_only_iter< D >::center_only_iter () [inline]`

Construction.

Definition at line 108 of file center_only_iter.hh.

10.352.3 Member Function Documentation

10.352.3.1 `void mln::iterator< center_only_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.353 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, center_only_iter< D >, centered_bkd_iter_adapter< D, I > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [centered_bkd_iter_adapter](#) ()
Construction.

10.353.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
<i>I</i>	The adapted complex relative iterator.

Definition at line 91 of file centered_iter_adapter.hh.

10.353.2 Constructor & Destructor Documentation

10.353.2.1 `template<unsigned D, typename I > mln::topo::centered_bkd_iter_adapter< D, I >::centered_bkd_iter_adapter () [inline]`

Construction.

Definition at line 141 of file `centered_iter_adapter.hh`.

10.353.3 Member Function Documentation

10.353.3.1 `void mln::iterator< centered_bkd_iter_adapter< D, I > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.354 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< center_only_iter< D >, I, centered_fwd_iter_adapter< D, I > >`.

Public Member Functions

- `void next ()`
Go to the next element.
- `centered_fwd_iter_adapter ()`
Construction.

10.354.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
<i>I</i>	The adapted complex relative iterator.

Definition at line 57 of file `centered_iter_adapter.hh`.

10.354.2 Constructor & Destructor Documentation

10.354.2.1 `template<unsigned D, typename I > mln::topo::centered_fwd_iter_adapter< D, I >::centered_fwd_iter_adapter () [inline]`

Construction.

Definition at line 122 of file `centered_iter_adapter.hh`.

10.354.3 Member Function Documentation

10.354.3.1 `void mln::iterator< centered_fwd_iter_adapter< D, I > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.355 mln::topo::complex< D > Class Template Reference

General complex of dimension D.

```
#include <complex.hh>
```

Public Types

- `typedef face_bkd_iter< D > bkd_citer`
Backward [mln::iterator](#) type iterating on all faces.
- `typedef face_fwd_iter< D > fwd_citer`
Forward [mln::iterator](#) type iterating on all faces.

Public Member Functions

- `const void * addr () const`
Get the address of the data of this complex.
- `complex ()`
Complex construction.
- `n_face< 0u, D > add_face ()`
Add a 0-face to the complex.
- `template<unsigned N>`
`n_face< N+1, D > add_face (const n_faces_set< N, D > &adjacent_faces)`
Add a (N+1)-face to the complex (with $N \geq 0$).
- `unsigned nfaces () const`
Static manipulators.

- `template<unsigned N>`
`unsigned nfaces_of_static_dim () const`
Return the number of N -faces.
- `unsigned nfaces_of_dim (unsigned n) const`
Dynamic manipulators.
- `void print (std::ostream &ostr) const`
Pretty-printing.
- `template<unsigned N>`
`void print_faces (std::ostream &ostr) const`
Print the faces of dimension N .

10.355.1 Detailed Description

`template<unsigned D>class mln::topo::complex< D >`

General complex of dimension D .

Definition at line 87 of file `complex.hh`.

10.355.2 Member Typedef Documentation

10.355.2.1 `template<unsigned D> typedef face_bkd_iter<D> mln::topo::complex< D >::bkd_citer`

Backward `mln::Iterator` type iterating on all faces.

Definition at line 93 of file `complex.hh`.

10.355.2.2 `template<unsigned D> typedef face_fwd_iter<D> mln::topo::complex< D >::fwd_citer`

Forward `mln::Iterator` type iterating on all faces.

Definition at line 91 of file `complex.hh`.

10.355.3 Constructor & Destructor Documentation

10.355.3.1 `template<unsigned D> complex< D >::complex () [inline]`

Complex construction.

Create a new D -complex.

Definition at line 471 of file `complex.hh`.

10.355.4 Member Function Documentation

10.355.4.1 `template<unsigned D> n_face< 0u, D > complex< D >::add_face () [inline]`

Add a 0-face to the complex.

Definition at line 480 of file `complex.hh`.

10.355.4.2 `template<unsigned D> template<unsigned N> n_face< N+1, D > complex< D >::add_face (const n_faces_set< N, D > & adjacent_faces) [inline]`

Add a (N+1)-face to the complex (with N >= 0).

Parameters

<code>adjacent_faces</code>	The (N-1)-faces adjacent to the new N-face.
-----------------------------	---

Definition at line 493 of file complex.hh.

References `mln::topo::n_faces_set< N, D >::faces()`.

10.355.4.3 `template<unsigned D> const void * complex< D >::addr () const [inline]`

Get the address of the data of this complex.

This address is a concise and useful information to print and track the actual content of this complex.

Definition at line 699 of file complex.hh.

10.355.4.4 `template<unsigned D> unsigned complex< D >::nfaces () const [inline]`

Static manipulators.

These methods use statically-known input.

Return the total number of faces, whatever their dimension.

Definition at line 579 of file complex.hh.

10.355.4.5 `template<unsigned D> unsigned complex< D >::nfaces_of_dim (unsigned n) const [inline]`

Dynamic manipulators.

These methods use input know as run time.

Return the number of *n*-faces.

Warning, this function has a complexity linear in term of N, since each `n_faces_set` is checked (the present implementation does not provide a direct access to `n_faces_set` through a dynamic value of the dimension).

Definition at line 601 of file complex.hh.

10.355.4.6 `template<unsigned D> template<unsigned N> unsigned complex< D >::nfaces_of_static_dim () const [inline]`

Return the number of N-faces.

Definition at line 588 of file complex.hh.

10.355.4.7 `template<unsigned D> void complex< D >::print (std::ostream & ostr) const [inline]`

Pretty-printing.

Print the complex.

Definition at line 679 of file complex.hh.

Referenced by `mln::topo::operator<<()`.

10.355.4.8 `template<unsigned D> template<unsigned N> void complex< D >::print_faces (std::ostream & ostr) const`
`[inline]`

Print the faces of dimension N.

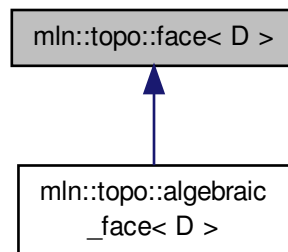
Definition at line 688 of file complex.hh.

10.356 `mln::topo::face< D >` Class Template Reference

Face handle in a complex; the face dimension is dynamic.

`#include <face.hh>`

Inheritance diagram for `mln::topo::face< D >`:



Public Member Functions

- `face ()`
Build a non-initialized face handle.
- `face (complex< D > &complex, unsigned n, unsigned face_id)`
Build a face handle from complex and face_id.
- `template<unsigned N>`
`face (const n_face< N, D > &f)`
Build a face handle from an mln::topo::n_face.
- `void invalidate ()`
Invalidate this handle.
- `bool is_valid () const`
Is this handle valid?
- `complex< D > cplx () const`
Accessors.
- `unsigned n () const`
Return the dimension of the face.
- `unsigned face_id () const`
Return the id of the face.
- `void set_cplx (const complex< D > &cplx)`
Set the complex the face belongs to.
- `void set_n (unsigned n)`

- Set the dimension of the face.*

 - void [inc_n](#) ()

Increment the dimension of the face.
- void [dec_n](#) ()

Decrement the dimension of the face.
- void [set_face_id](#) (unsigned [face_id](#))

Set the id of the face.
- void [inc_face_id](#) ()

Increment the id of the face.
- void [dec_face_id](#) ()

Decrement the id of the face.
- template<unsigned N>
[face_data](#)< N, D > & [data](#) () const

Return the mln::topo::face_data pointed by this handle.
- std::vector< [algebraic_face](#)< D > > [lower_dim_adj_faces](#) () const

Return an array of face handles pointing to adjacent (n-1)-faces.
- std::vector< [algebraic_face](#)< D > > [higher_dim_adj_faces](#) () const

Return an array of face handles pointing to adjacent (n+1)-faces.

10.356.1 Detailed Description

template<unsigned D>class mln::topo::face< D >

Face handle in a complex; the face dimension is dynamic.

Contrary to an [mln::topo::n_face](#), the dimension of an [mln::topo::face](#) is not fixed.

Definition at line 64 of file face.hh.

10.356.2 Constructor & Destructor Documentation

10.356.2.1 template<unsigned D> face< D >::face () [inline]

Build a non-initialized face handle.

Definition at line 178 of file face.hh.

10.356.2.2 template<unsigned D> face< D >::face (complex< D > & complex, unsigned n, unsigned face_id) [inline]

Build a face handle from *complex* and *face_id*.

Definition at line 187 of file face.hh.

10.356.2.3 template<unsigned D> template<unsigned N> face< D >::face (const n_face< N, D > & f) [inline]

Build a face handle from an [mln::topo::n_face](#).

Definition at line 197 of file face.hh.

10.356.3 Member Function Documentation

10.356.3.1 `template<unsigned D> complex< D > face< D >::cplx () const` `[inline]`

Accessors.

Return the complex the face belongs to.

Definition at line 224 of file face.hh.

Referenced by `mln::complex_psite< D, G >::complex_psite()`, `mln::topo::operator!=()`, and `mln::topo::operator==()`.

10.356.3.2 `template<unsigned D> template<unsigned N> face_data< N, D > & face< D >::data () const`
`[inline]`

Return the `mln::topo::face_data` pointed by this handle.

Definition at line 305 of file face.hh.

10.356.3.3 `template<unsigned D> void face< D >::dec_face_id ()` `[inline]`

Decrement the id of the face.

Definition at line 296 of file face.hh.

10.356.3.4 `template<unsigned D> void face< D >::dec_n ()` `[inline]`

Decrement the dimension of the face.

Definition at line 272 of file face.hh.

10.356.3.5 `template<unsigned D> unsigned face< D >::face_id () const` `[inline]`

Return the id of the face.

Definition at line 240 of file face.hh.

Referenced by `mln::geom::complex_geometry< D, P >::operator()()`, and `mln::topo::operator==()`.

10.356.3.6 `template<unsigned D> std::vector< algebraic_face< D > > face< D >::higher_dim_adj_faces () const`
`[inline]`

Return an array of face handles pointing to adjacent (n+1)-faces.

Definition at line 370 of file face.hh.

10.356.3.7 `template<unsigned D> void face< D >::inc_face_id ()` `[inline]`

Increment the id of the face.

Definition at line 288 of file face.hh.

10.356.3.8 `template<unsigned D> void face< D >::inc_n ()` `[inline]`

Increment the dimension of the face.

Definition at line 264 of file face.hh.

10.356.3.9 `template<unsigned D> void face< D >::invalidate () [inline]`

Invalidate this handle.

Definition at line 215 of file face.hh.

10.356.3.10 `template<unsigned D> bool face< D >::is_valid () const [inline]`

Is this handle valid?

Definition at line 207 of file face.hh.

10.356.3.11 `template<unsigned D> std::vector< algebraic_face< D > > face< D >::lower_dim_adj_faces () const [inline]`

Return an array of face handles pointing to adjacent (n-1)-faces.

Definition at line 357 of file face.hh.

10.356.3.12 `template<unsigned D> unsigned face< D >::n () const [inline]`

Return the dimension of the face.

Definition at line 232 of file face.hh.

Referenced by `mln::topo::algebraic_face< D >::algebraic_face()`, `mln::geom::complex_geometry< D, P >::operator()`, and `mln::topo::operator==()`.

10.356.3.13 `template<unsigned D> void face< D >::set_cplx (const complex< D > & cplx) [inline]`

Set the complex the face belongs to.

Definition at line 248 of file face.hh.

10.356.3.14 `template<unsigned D> void face< D >::set_face_id (unsigned face_id) [inline]`

Set the id of the face.

Definition at line 280 of file face.hh.

10.356.3.15 `template<unsigned D> void face< D >::set_n (unsigned n) [inline]`

Set the dimension of the face.

Definition at line 256 of file face.hh.

10.357 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< topo::face< D >, face_bkd_iter< D > >`.

Public Member Functions

- void `next` ()
Go to the next element.
- `face_bkd_iter` ()
Construction and assignment.
- void `start` ()
Manipulation.

10.357.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

Definition at line 112 of file `face_iter.hh`.

10.357.2 Constructor & Destructor Documentation

10.357.2.1 `template<unsigned D> face_bkd_iter< D >::face_bkd_iter () [inline]`

Construction and assignment.

Definition at line 207 of file `face_iter.hh`.

10.357.3 Member Function Documentation

10.357.3.1 `void mln::Iterator< face_bkd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the `next_` method.

Precondition

The iterator is valid.

10.357.3.2 `template<unsigned D> void face_bkd_iter< D >::start () [inline]`

Manipulation.

Start an iteration.

Definition at line 224 of file `face_iter.hh`.

10.358 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< topo::face< D >, face_fwd_iter< D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [face_fwd_iter](#) ()
Construction and assignment.
- void [start](#) ()
Manipulation.

10.358.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

Definition at line 69 of file face_iter.hh.

10.358.2 Constructor & Destructor Documentation

10.358.2.1 `template<unsigned D> face_fwd_iter< D >::face_fwd_iter () [inline]`

Construction and assignment.

Definition at line 155 of file face_iter.hh.

10.358.3 Member Function Documentation

10.358.3.1 `void mln::Iterator< face_fwd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.358.3.2 `template<unsigned D> void face_fwd_iter< D >::start () [inline]`

Manipulation.

Test if the iterator is valid.

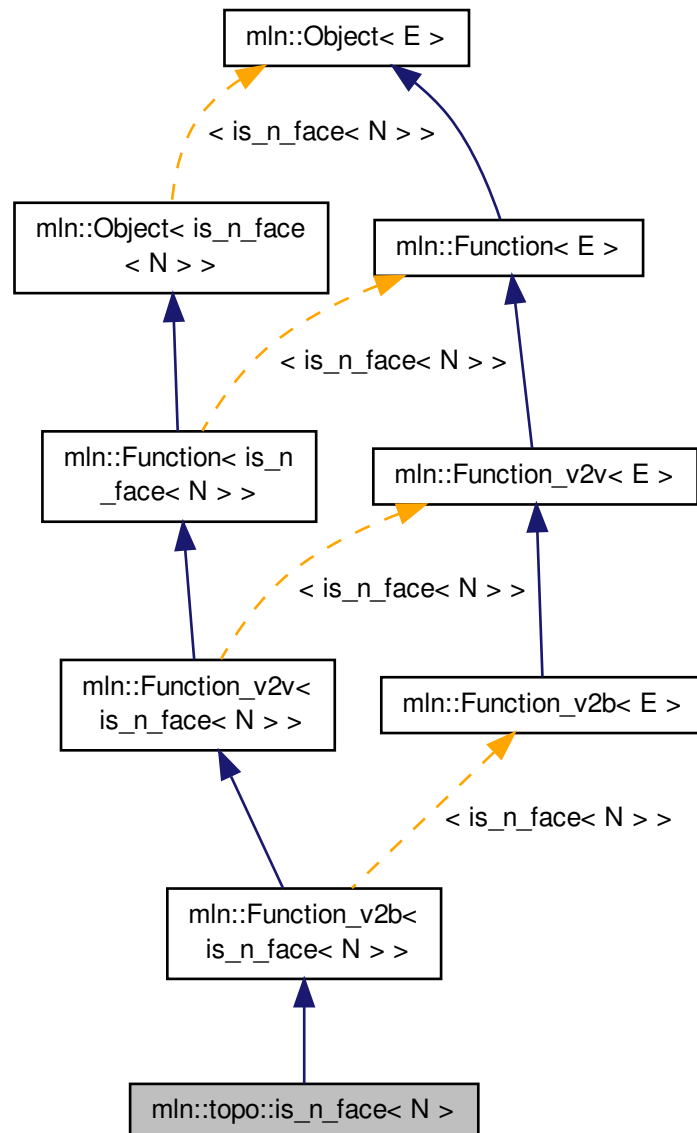
Definition at line 172 of file `face_iter.hh`.

10.359 `mln::topo::is_n_face< N >` Struct Template Reference

A functor testing wheter a `mln::complex_psite` is an N -face.

`#include <is_n_face.hh>`

Inheritance diagram for `mln::topo::is_n_face< N >`:



10.359.1 Detailed Description

```
template<unsigned N>struct mln::topo::is_n_face< N >
```

A functor testing wheter a [mln::complex_psite](#) is an N -face.

Definition at line 48 of file is_n_face.hh.

10.360 mln::topo::is_simple_2d_t< N > Struct Template Reference

Test if a point is simple or not.

```
#include <is_simple_2d.hh>
```

10.360.1 Detailed Description

```
template<typename N>struct mln::topo::is_simple_2d_t< N >
```

Test if a point is simple or not.

A point of an object is simple if in its c8 neiborhood, there is exactly one connected component of the object, and only one connected component of the background Examples : (| == object, - = background)

```
- - |
| P | Here p is simple in the c4 and c8 case.
| | |

- | -
| P | Here p is never simple.
| | |
```

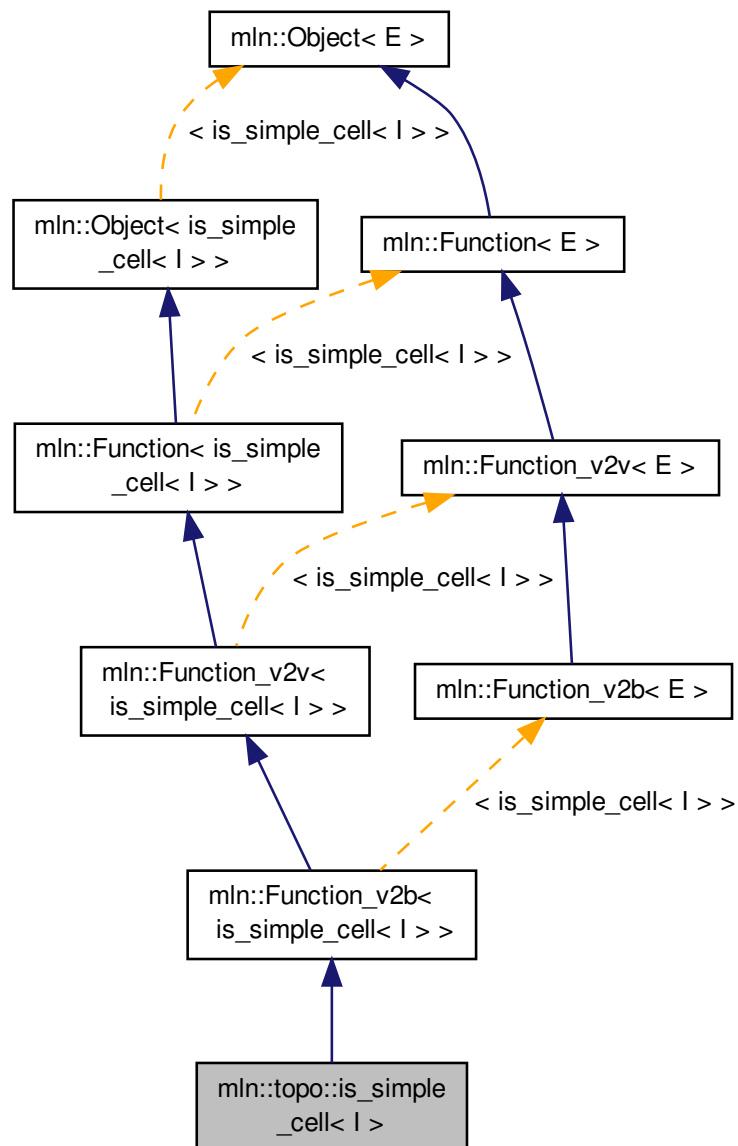
Definition at line 66 of file is_simple_2d.hh.

10.361 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).

10.361.1 Detailed Description

```
template<typename I>class mln::topo::is_simple_cell< I >
```

A predicate for the simplicity of a point based on the collapse property of the attachment.

The functor does not actually take a cell as input, but a face that is expected to be a D-facet.

Definition at line 57 of file `is_simple_cell.hh`.

10.361.2 Member Typedef Documentation

10.361.2.1 `template<typename I > typedef mln::complex_psite<D, G> mln::topo::is_simple_cell< I >::psite`

Psite type.

Definition at line 65 of file `is_simple_cell.hh`.

10.361.2.2 `template<typename I > typedef bool mln::topo::is_simple_cell< I >::result`

Result type of the functor.

Definition at line 68 of file `is_simple_cell.hh`.

10.361.3 Member Function Documentation

10.361.3.1 `template<typename I > typedef mln::topo::is_simple_cell< I >::mln_geom (I)`

Geometry of the image.

10.361.3.2 `template<typename I > bool mln::topo::is_simple_cell< I >::operator() (const mln::complex_psite< I::dim, mln_geom(I)> &p) const` `[inline]`

Based on the algorithm A2 from couprie.08.pami.

Definition at line 115 of file `is_simple_cell.hh`.

References `mln::make::attachment()`.

10.361.3.3 `template<typename I > void mln::topo::is_simple_cell< I >::set_image (const mln::Image< I > &ima)` `[inline]`

Set the underlying image.

Definition at line 107 of file `is_simple_cell.hh`.

10.361.4 Member Data Documentation

10.361.4.1 `template<typename I> const unsigned mln::topo::is_simple_cell<I>::D = I::dim` [static]

Dimension of the image (and therefore of the complex).

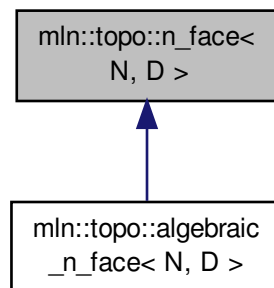
Definition at line 61 of file `is_simple_cell.hh`.

10.362 `mln::topo::n_face< N, D >` Class Template Reference

N-face handle in a complex.

```
#include <n_face.hh>
```

Inheritance diagram for `mln::topo::n_face< N, D >`:



Public Member Functions

- `void invalidate ()`
Invalidate this handle.
- `bool is_valid () const`
Is this handle valid?
- `n_face ()`
Build a non-initialized face handle.
- `n_face (complex< D > &complex, unsigned face_id)`
Build a face handle from complex and face_id.
- `complex< D > cplx () const`
Accessors.
- `unsigned face_id () const`
Return the id of the face.
- `void set_cplx (const complex< D > &cplx)`
Set the complex the face belongs to.
- `unsigned n () const`
Return the dimension of the face.
- `void set_face_id (unsigned face_id)`
Set the id of the face.

- void [inc_face_id](#) ()
Increment the id of the face.
- void [dec_face_id](#) ()
Decrement the id of the face.
- [face_data](#)< N, D > & [data](#) () const
Return the mln::topo::face_data pointed by this handle.
- std::vector< [algebraic_n_face](#)
< N-1, D > > [lower_dim_adj_faces](#) () const
Return an array of face handles pointing to adjacent (n-1)-faces.
- std::vector< [algebraic_n_face](#)
< N+1, D > > [higher_dim_adj_faces](#) () const
Return an array of face handles pointing to adjacent (n+1)-faces.

10.362.1 Detailed Description

template<unsigned N, unsigned D>class mln::topo::n_face< N, D >

N-face handle in a complex.

Contrary to an [mln::topo::face](#), the dimension of an [mln::topo::n_face](#) is fixed.

Definition at line 61 of file n_face.hh.

10.362.2 Constructor & Destructor Documentation

10.362.2.1 template<unsigned N, unsigned D> [n_face](#)< N, D >::n_face () [inline]

Build a non-initialized face handle.

Definition at line 159 of file n_face.hh.

References [mln::topo::n_face](#)< N, D >::is_valid().

10.362.2.2 template<unsigned N, unsigned D> [n_face](#)< N, D >::n_face ([complex](#)< D > & *complex*, unsigned *face_id*) [inline]

Build a face handle from *complex* and *face_id*.

Definition at line 169 of file n_face.hh.

10.362.3 Member Function Documentation

10.362.3.1 template<unsigned N, unsigned D> [complex](#)< D > [n_face](#)< N, D >::cplx () const [inline]

Accessors.

Return the complex the face belongs to.

Definition at line 195 of file n_face.hh.

Referenced by [mln::topo::n_faces_set](#)< N, D >::add(), [mln::topo::operator!=\(\)](#), and [mln::topo::operator==\(\)](#).

10.362.3.2 template<unsigned N, unsigned D> [face_data](#)< N, D > & [n_face](#)< N, D >::data () const [inline]

Return the mln::topo::face_data pointed by this handle.

Definition at line 251 of file n_face.hh.

10.362.3.3 `template<unsigned N, unsigned D> void n_face< N, D >::dec_face_id () [inline]`

Decrement the id of the face.

Definition at line 243 of file `n_face.hh`.

10.362.3.4 `template<unsigned N, unsigned D> unsigned n_face< N, D >::face_id () const [inline]`

Return the id of the face.

Definition at line 211 of file `n_face.hh`.

Referenced by `mln::topo::operator==()`.

10.362.3.5 `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N+1, D > > n_face< N, D >::higher_dim_adj_faces () const [inline]`

Return an array of face handles pointing to adjacent (n+1)-faces.

Definition at line 270 of file `n_face.hh`.

Referenced by `mln::topo::edge()`.

10.362.3.6 `template<unsigned N, unsigned D> void n_face< N, D >::inc_face_id () [inline]`

Increment the id of the face.

Definition at line 235 of file `n_face.hh`.

10.362.3.7 `template<unsigned N, unsigned D> void n_face< N, D >::invalidate () [inline]`

Invalidate this handle.

Definition at line 187 of file `n_face.hh`.

10.362.3.8 `template<unsigned N, unsigned D> bool n_face< N, D >::is_valid () const [inline]`

Is this handle valid?

Definition at line 179 of file `n_face.hh`.

Referenced by `mln::topo::algebraic_n_face< N, D >::algebraic_n_face()`, and `mln::topo::n_face< N, D >::n_face()`.

10.362.3.9 `template<unsigned N, unsigned D> std::vector< algebraic_n_face< N-1, D > > n_face< N, D >::lower_dim_adj_faces () const [inline]`

Return an array of face handles pointing to adjacent (n-1)-faces.

Definition at line 260 of file `n_face.hh`.

10.362.3.10 `template<unsigned N, unsigned D> unsigned n_face< N, D >::n () const [inline]`

Return the dimension of the face.

Definition at line 203 of file `n_face.hh`.

10.362.3.11 `template<unsigned N, unsigned D> void n_face< N, D >::set_cplx (const complex< D > & cplx)`
`[inline]`

Set the complex the face belongs to.

Definition at line 219 of file n_face.hh.

10.362.3.12 `template<unsigned N, unsigned D> void n_face< N, D >::set_face_id (unsigned face_id)` `[inline]`

Set the id of the face.

Definition at line 227 of file n_face.hh.

10.363 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< topo::face< D >, n_face_bkd_iter< D > >.

Public Member Functions

- void `next` ()
Go to the next element.
- void `start` ()
Manipulation.
- unsigned `n` () const
Accessors.

10.363.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

Definition at line 127 of file n_face_iter.hh.

10.363.2 Member Function Documentation

10.363.2.1 `template<unsigned D> unsigned mln::topo::n_face_bkd_iter< D >::n () const` `[inline]`

Accessors.

Shortcuts to face_'s accessors.

Definition at line 308 of file n_face_iter.hh.

10.363.2.2 `void mln::iterator< n_face_bkd_iter< D > >::next ()` [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.363.2.3 `template<unsigned D> void mln::topo::n_face_bkd_iter< D >::start ()` [inline]

Manipulation.

Start an iteration.

Definition at line 275 of file `n_face_iter.hh`.

10.364 `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< topo::face< D >, n_face_fwd_iter< D > >`.

Public Member Functions

- `void next ()`
Go to the next element.
- `void start ()`
Manipulation.
- `unsigned n () const`
Accessors.

10.364.1 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

<i>D</i>	The dimension of the complex this iterator belongs to.
----------	--

Definition at line 72 of file `n_face_iter.hh`.

10.364.2 Member Function Documentation

10.364.2.1 `template<unsigned D> unsigned mln::topo::n_face_fwd_iter< D >::n () const [inline]`

Accessors.

Shortcuts to face_'s accessors.

Definition at line 235 of file n_face_iter.hh.

10.364.2.2 `void mln::iterator< n_face_fwd_iter< D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.364.2.3 `template<unsigned D> void mln::topo::n_face_fwd_iter< D >::start () [inline]`

Manipulation.

Test if the iterator is valid.

Definition at line 202 of file n_face_iter.hh.

10.365 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.

10.365.1 Detailed Description

`template<unsigned N, unsigned D> class mln::topo::n_faces_set< N, D >`

Set of face handles of dimension N .

Definition at line 56 of file `n_faces_set.hh`.

10.365.2 Member Typedef Documentation

10.365.2.1 `template<unsigned N, unsigned D> typedef std::vector< algebraic_n_face<N, D> > mln::topo::n_faces_set< N, D >::faces_type`

The type of the set of face handles.

Definition at line 70 of file `n_faces_set.hh`.

10.365.3 Member Function Documentation

10.365.3.1 `template<unsigned N, unsigned D> void n_faces_set< N, D >::add (const algebraic_n_face< N, D > & f) [inline]`

Append an algebraic face f to the set.

Definition at line 171 of file `n_faces_set.hh`.

References `mln::topo::n_face< N, D >::cplx()`.

Referenced by `mln::topo::operator+()`, and `mln::topo::operator-()`.

10.365.3.2 `template<unsigned N, unsigned D> const std::vector< algebraic_n_face< N, D > > & n_faces_set< N, D >::faces () const [inline]`

Accessors.

Return the set of handles.

Definition at line 190 of file `n_faces_set.hh`.

Referenced by `mln::topo::complex< D >::add_face()`.

10.365.3.3 `template<unsigned N, unsigned D> void n_faces_set< N, D >::reserve (size_t n) [inline]`

Reserve n cells in the set.

This methods does not change the content of `faces_`; it only pre-allocate memory. Method `reserve` is provided for efficiency purpose, and its use is completely optional.

Definition at line 182 of file `n_faces_set.hh`.

10.366 mln::topo::skeleton::is_simple_point< N > Struct Template Reference

`#include <is_simple_point.hh>`

10.366.1 Detailed Description

```
template<typename N>struct mln::topo::skeleton::is_simple_point< N >
```

Tell if a point is simple or not. A point of an object is simple if in its c8 neighborhood, there is exactly one connected component of the object, and only one connected component of the background Examples : (| == object, - = background)

```
- - |
| P | Here p is simple in the c4 and c8 case.
| | |

- | -
| P | Here p is never simple.
| | |
```

Definition at line 68 of file is_simple_point.hh.

10.367 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< topo::face< D >, static_n_face_bkd_iter< N, D > >.

Public Member Functions

- void [next](#) ()
Go to the next element.
- [static_n_face_bkd_iter](#) ()
Construction and assignment.
- void [start](#) ()
Manipulation.

10.367.1 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

<i>N</i>	The dimension of the face associated to this iterator.
<i>D</i>	The dimension of the complex this iterator belongs to.

Definition at line 101 of file static_n_face_iter.hh.

10.367.2 Constructor & Destructor Documentation

10.367.2.1 `template<unsigned N, unsigned D> mln::topo::static_n_face_bkd_iter< N, D >::static_n_face_bkd_iter () [inline]`

Construction and assignment.

Definition at line 194 of file static_n_face_iter.hh.

10.367.3 Member Function Documentation

10.367.3.1 `void mln::iterator< static_n_face_bkd_iter< N, D > >::next ()` [inherited]

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next_* method.

Precondition

The iterator is valid.

10.367.3.2 `template<unsigned N, unsigned D> void mln::topo::static_n_face_bkd_iter< N, D >::start ()`
[inline]

Manipulation.

Start an iteration.

Definition at line 217 of file `static_n_face_iter.hh`.

10.368 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< topo::face< D >, static_n_face_fwd_iter< N, D > >`.

Public Member Functions

- `void next ()`
Go to the next element.
- `static_n_face_fwd_iter ()`
Construction and assignment.
- `void start ()`
Manipulation.

10.368.1 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

<i>N</i>	The dimension of the face associated to this iterator.
<i>D</i>	The dimension of the complex this iterator belongs to.

Definition at line 55 of file static_n_face_iter.hh.

10.368.2 Constructor & Destructor Documentation

10.368.2.1 `template<unsigned N, unsigned D> mln::topo::static_n_face_fwd_iter< N, D >::static_n_face_fwd_iter () [inline]`

Construction and assignment.

Definition at line 145 of file static_n_face_iter.hh.

10.368.3 Member Function Documentation

10.368.3.1 `void mln::iterator< static_n_face_fwd_iter< N, D > >::next () [inherited]`

Go to the next element.

Warning

This is a final method; iterator classes should not re-defined this method. The actual "next" operation has to be defined through the *next* method.

Precondition

The iterator is valid.

10.368.3.2 `template<unsigned N, unsigned D> void mln::topo::static_n_face_fwd_iter< N, D >::start () [inline]`

Manipulation.

Test if the iterator is valid.

Definition at line 168 of file static_n_face_iter.hh.

10.369 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, 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.

10.369.1 Detailed Description

template<typename S, typename I, typename T> struct mln::tr_image< S, I, T >

Transform an image by a given transformation.

Definition at line 83 of file tr_image.hh.

10.369.2 Member Typedef Documentation

10.369.2.1 template<typename S, typename I, typename T> typedef I::value mln::tr_image< S, I, T >::lvalue

Return type of read-write access.

Definition at line 101 of file tr_image.hh.

10.369.2.2 template<typename S, typename I, typename T> typedef I::psite mln::tr_image< S, I, T >::psite

Point_Site associated type.

Definition at line 92 of file tr_image.hh.

10.369.2.3 template<typename S, typename I, typename T> typedef I::value mln::tr_image< S, I, T >::rvalue

Return type of read-only access.

Definition at line 104 of file tr_image.hh.

10.369.2.4 `template<typename S, typename I, typename T> typedef I ::site mln::tr_image< S, I, T >::site`

[Site](#) associated type.

Definition at line 95 of file tr_image.hh.

10.369.2.5 `template<typename S, typename I, typename T> typedef tr_image< S, tag::image_<I>, T> mln::tr_image< S, I, T >::skeleton`

Skeleton.

Definition at line 107 of file tr_image.hh.

10.369.2.6 `template<typename S, typename I, typename T> typedef I ::value mln::tr_image< S, I, T >::value`

[Value](#) associated type.

Definition at line 98 of file tr_image.hh.

10.369.3 Constructor & Destructor Documentation

10.369.3.1 `template<typename S, typename I, typename T> tr_image< S, I, T >::tr_image (const S & s, const I & ima, const T & tr) [inline]`

Constructors.

Definition at line 174 of file tr_image.hh.

10.369.4 Member Function Documentation

10.369.4.1 `template<typename S, typename I, typename T> const S & tr_image< S, I, T >::domain () const [inline]`

Return the domain morpher.

Definition at line 247 of file tr_image.hh.

10.369.4.2 `template<typename S, typename I, typename T> bool tr_image< S, I, T >::has (const vec.t & v) const [inline]`

Test if a pixel value is accessible at v.

Definition at line 200 of file tr_image.hh.

10.369.4.3 `template<typename S, typename I, typename T> bool tr_image< S, I, T >::is_valid () const [inline]`

Test if this image has been initialized.

Definition at line 191 of file tr_image.hh.

10.369.4.4 `template<typename S, typename I, typename T> I::value tr_image< S, I, T >::operator() (const psite & p) const [inline]`

Read-only access of pixel value at point site p.

Mutable access is only OK for reading (not writing).

Definition at line 213 of file tr_image.hh.

10.369.4.5 `template<typename S, typename I, typename T> void tr_image< S, I, T>::set_tr (T & tr) [inline]`

Set the transformation.

Definition at line 231 of file tr_image.hh.

10.369.4.6 `template<typename S, typename I, typename T> const T & tr_image< S, I, T>::tr () const [inline]`

Return the underlying transformation.

Definition at line 239 of file tr_image.hh.

10.370 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, p_transformed< I::domain_t, F >, 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.
- I::rvalue [operator\(\)](#) (const typename I::psite &p) const
Read-only access of pixel value at point site p.
- internal::morpher_lvalue_< I >::ret [operator\(\)](#) (const typename I::psite &p)
Read and "write if possible" access of pixel value at point site p.
- [transformed_image](#) ()
Constructor without argument.
- [transformed_image](#) (I &ima, const F &f)
Constructor.

10.370.1 Detailed Description

```
template<typename I, typename F> struct mln::transformed_image< I, F >
```

[Image](#) having its domain restricted by a site set.

Definition at line 96 of file transformed_image.hh.

10.370.2 Member Typedef Documentation

10.370.2.1 `template<typename I, typename F> typedef transformed_image< tag::image_<I>, tag::function_<F> > mln::transformed_image< I, F >::skeleton`

Skeleton.

Definition at line 101 of file transformed_image.hh.

10.370.3 Constructor & Destructor Documentation

10.370.3.1 `template<typename I, typename F> transformed_image< I, F >::transformed_image () [inline]`

Constructor without argument.

Definition at line 187 of file transformed_image.hh.

10.370.3.2 `template<typename I, typename F> transformed_image< I, F >::transformed_image (I & ima, const F & f) [inline]`

Constructor.

Definition at line 193 of file transformed_image.hh.

10.370.4 Member Function Documentation

10.370.4.1 `template<typename I, typename F> const p_transformed< typename I::domain_t, F > & transformed_image< I, F >::domain () const [inline]`

Give the definition domain.

Definition at line 210 of file transformed_image.hh.

10.370.4.2 `template<typename I, typename F> transformed_image< I, F >::operator transformed_image< const I, F > () const [inline]`

Const promotion via conversion.

Definition at line 240 of file transformed_image.hh.

10.370.4.3 `template<typename I, typename F> I::rvalue transformed_image< I, F >::operator() (const typename I::psite & p) const [inline]`

Read-only access of pixel value at point site *p*.

Definition at line 218 of file transformed_image.hh.

10.370.4.4 `template<typename I, typename F> internal::morpher_lvalue_< I >::ret transformed_image< I, F >::operator() (const typename I::psite & p) [inline]`

Read and "write if possible" access of pixel value at point site *p*.

Definition at line 229 of file transformed_image.hh.

10.371 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, unproject_image< I, D, F > >.

Public Member Functions

- const D & [domain](#) () const
Give the definition domain.
- I::rvalue [operator\(\)](#) (const typename D::psite &p) const
Read-only access to the image value located at point p.
- internal::morpher_lvalue_< I >::ret [operator\(\)](#) (const typename D::psite &p)
Read-write access to the image value located at point p.
- [unproject_image](#) ()
Constructor without argument.
- [unproject_image](#) (I &ima, const D &dom, const F &f)
Constructor from an image ima, a domain dom, and a function f.

10.371.1 Detailed Description

```
template<typename I, typename D, typename F>struct mln::unproject_image< I, D, F >
```

Un-projects an image.

Definition at line 96 of file unproject_image.hh.

10.371.2 Constructor & Destructor Documentation

10.371.2.1 `template<typename I , typename D , typename F > unproject_image< I, D, F >::unproject_image ()`
[inline]

Constructor without argument.

Definition at line 182 of file unproject_image.hh.

10.371.2.2 `template<typename I , typename D , typename F > unproject_image< I, D, F >::unproject_image (I & ima, const D & dom, const F & f)` [inline]

Constructor from an image ima, a domain dom, and a function f.

Definition at line 188 of file unproject_image.hh.

10.371.3 Member Function Documentation

10.371.3.1 `template<typename I , typename D , typename F > const D & unproject_image< I, D, F >::domain () const`
[inline]

Give the definition domain.

Definition at line 205 of file unproject_image.hh.

10.371.3.2 `template<typename I , typename D , typename F > I::rvalue unproject_image< I, D, F >::operator() (const typename D::psite & p) const [inline]`

Read-only access to the image value located at point `p`.

Definition at line 214 of file `unproject_image.hh`.

10.371.3.3 `template<typename I , typename D , typename F > internal::morpher_lvalue< I >::ret unproject_image< I, D, F >::operator() (const typename D::psite & p) [inline]`

Read-write access to the image value located at point `p`.

Definition at line 225 of file `unproject_image.hh`.

10.372 mln::util::adjacency_matrix< V > Class Template Reference

A class of adjacency matrix.

```
#include <adjacency_matrix.hh>
```

Inherits `mlc_converts_toV`, `check_t`, and `mln::util::internal::adjacency_matrix_impl_selector< V, mln::metal::equal< mln_trait_value_quant(V), trait::value::quant::low >::eval >`.

Public Member Functions

- [adjacency_matrix](#) ()
Constructors.
- [adjacency_matrix](#) (const V &nelements)
Construct an adjacency matrix with `nelements` elements maximum.

10.372.1 Detailed Description

```
template<typename V = def::coord>class mln::util::adjacency_matrix< V >
```

A class of adjacency matrix.

Support low and high quantification value types. In case of low quantification value type, it uses an [image2d](#) to store adjacency information. In case of high quantification value type, it uses a [util::set](#) to store the adjacency information.

Definition at line 136 of file `adjacency_matrix.hh`.

10.372.2 Constructor & Destructor Documentation

10.372.2.1 `template<typename V > mln::util::adjacency_matrix< V >::adjacency_matrix ()`

Constructors.

```
@{
```

Default

Definition at line 308 of file `adjacency_matrix.hh`.

10.372.2.2 `template<typename V > mln::util::adjacency_matrix< V >::adjacency_matrix (const V & nelements)`

Construct an adjacency matrix with `nelements` elements maximum.

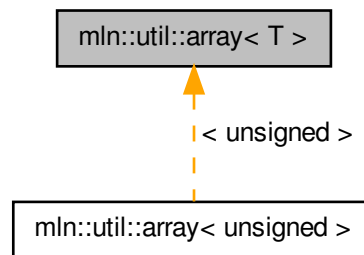
Definition at line 315 of file adjacency_matrix.hh.

10.373 mln::util::array< T > Class Template Reference

A dynamic array class.

```
#include <array.hh>
```

Inheritance diagram for mln::util::array< T >:



Public Types

- typedef T [element](#)
Element associated type.
- typedef T [result](#)
Returned value types.
- typedef array_fwd_iter< T > [fwd_eiter](#)
Iterator types
Forward iterator associated type.
- typedef array_bkd_iter< T > [bkd_eiter](#)
Backward iterator associated type.
- typedef [fwd_eiter](#) [eiter](#)
Iterator associated type.

Public Member Functions

- [array](#)< T > & [append](#) (const T &elt)
Add the element `elt` at the end of this array.
- template<typename U >
[array](#)< T > & [append](#) (const [array](#)< U > &other)
Add the elements of `other` at the end of this array.
- void [clear](#) ()
Empty the array.
- void [fill](#) (const T &value)
Fill the whole array with value `value`.

- `bool is_empty () const`
Test if the array is empty.
- `ro_result last () const`
Return the last element.
- `mutable_result last ()`
Return the last element.
- `std::size_t memory_size () const`
Return the size of this array in memory.
- `unsigned nelements () const`
Return the number of elements of the array.
- `ro_result operator() (unsigned i) const`
*Return the *i*-th element of the array.*
- `mutable_result operator() (unsigned i)`
*Return the *i*-th element of the array.*
- `ro_result operator[] (unsigned i) const`
*Return the *i*-th element of the array.*
- `mutable_result operator[] (unsigned i)`
*Return the *i*-th element of the array.*
- `void reserve (unsigned n)`
*Reserve memory for *n* elements.*
- `void resize (unsigned n)`
*Resize this array to *n* elements.*
- `void resize (unsigned n, const T &value)`
*Resize this array to *n* elements with *value* as value.*
- `unsigned size () const`
Return the number of elements of the array.
- `const std::vector< T > & std_vector () const`
Return the corresponding `std::vector` of elements.
- `array ()`
Constructors
Constructor without arguments.
- `array (unsigned n)`
*Construct a new array and resize it to *n* elements.*
- `array (unsigned n, const T &value)`
*Construct a new array, resize it to *n* elements and fill it with *default_value*.*

10.373.1 Detailed Description

`template<typename T>class mln::util::array< T >`

A dynamic array class.

Elements are stored by copy. Implementation is lazy.

The parameter `T` is the element type, which shall not be const-qualified.

Definition at line 99 of file `util/array.hh`.

10.373.2 Member Typedef Documentation

10.373.2.1 `template<typename T> typedef array_bkd_iter<T> mln::util::array< T >::bkd_eiter`

Backward iterator associated type.

Definition at line 124 of file util/array.hh.

10.373.2.2 `template<typename T> typedef fwd_eiter mln::util::array< T >::eiter`

[Iterator](#) associated type.

Definition at line 127 of file util/array.hh.

10.373.2.3 `template<typename T> typedef T mln::util::array< T >::element`

Element associated type.

Definition at line 107 of file util/array.hh.

10.373.2.4 `template<typename T> typedef array_fwd_iter<T> mln::util::array< T >::fwd_eiter`

[Iterator](#) types

Forward iterator associated type.

Definition at line 121 of file util/array.hh.

10.373.2.5 `template<typename T> typedef T mln::util::array< T >::result`

Returned value types.

Related to the [Function_v2v](#) concept.

Definition at line 112 of file util/array.hh.

10.373.3 Constructor & Destructor Documentation

10.373.3.1 `template<typename T> array< T >::array () [inline]`

Constructors

Constructor without arguments.

Definition at line 427 of file util/array.hh.

10.373.3.2 `template<typename T> array< T >::array (unsigned n) [inline]`

Construct a new array and resize it to elements.

Definition at line 433 of file util/array.hh.

10.373.3.3 `template<typename T> array< T >::array (unsigned n, const T & value) [inline]`

Construct a new array, resize it to elements and fill it with `default_value`.

Definition at line 440 of file util/array.hh.

10.373.4 Member Function Documentation

10.373.4.1 `template<typename T> array< T > & array< T >::append (const T & elt) [inline]`

Add the element `elt` at the end of this array.

Definition at line 472 of file util/array.hh.

Referenced by `mln::io::dicom::get_header()`, and `mln::io::plot::load()`.

10.373.4.2 `template<typename T> template<typename U> array< T > & array< T >::append (const array< U > & other) [inline]`

Add the elements of `other` at the end of this array.

Definition at line 482 of file util/array.hh.

References `mln::util::array< T >::is_empty()`, and `mln::util::array< T >::std_vector()`.

10.373.4.3 `template<typename T> void array< T >::clear () [inline]`

Empty the array.

All elements contained in the array are destroyed.

Postcondition

`is_empty() == true`

Definition at line 495 of file util/array.hh.

Referenced by `mln::io::plot::load()`.

10.373.4.4 `template<typename T> void array< T >::fill (const T & value) [inline]`

Fill the whole array with value `value`.

Definition at line 504 of file util/array.hh.

10.373.4.5 `template<typename T> bool array< T >::is_empty () const [inline]`

Test if the array is empty.

Definition at line 578 of file util/array.hh.

Referenced by `mln::util::array< T >::append()`, `mln::make::image3d()`, and `mln::io::pnms::load()`.

10.373.4.6 `template<typename T> array< T >::ro_result array< T >::last () const [inline]`

Return the last element.

Definition at line 562 of file util/array.hh.

10.373.4.7 `template<typename T> array< T >::mutable_result array< T >::last () [inline]`

Return the last element.

Definition at line 570 of file util/array.hh.

10.373.4.8 `template<typename T> std::size_t array< T>::memory_size () const` `[inline]`

Return the size of this array in memory.

Definition at line 602 of file util/array.hh.

10.373.4.9 `template<typename T> unsigned array< T>::nelements () const` `[inline]`

Return the number of elements of the array.

Definition at line 520 of file util/array.hh.

Referenced by `mln::labeling::fill_holes()`, `mln::make::image3d()`, `mln::io::pnms::load()`, `mln::util::operator<<()`, and `mln::io::plot::save()`.

10.373.4.10 `template<typename T> array< T>::ro_result array< T>::operator() (unsigned i) const` `[inline]`

Return the `i`-th element of the array.

Precondition

`i < nelements()`

Definition at line 528 of file util/array.hh.

10.373.4.11 `template<typename T> array< T>::mutable_result array< T>::operator() (unsigned i)` `[inline]`

Return the `i`-th element of the array.

Precondition

`i < nelements()`

Definition at line 536 of file util/array.hh.

10.373.4.12 `template<typename T> array< T>::ro_result array< T>::operator[] (unsigned i) const` `[inline]`

Return the `i`-th element of the array.

Precondition

`i < nelements()`

Definition at line 544 of file util/array.hh.

10.373.4.13 `template<typename T> array< T>::mutable_result array< T>::operator[] (unsigned i)` `[inline]`

Return the `i`-th element of the array.

Precondition

`i < nelements()`

Definition at line 553 of file util/array.hh.

10.373.4.14 `template<typename T> void array< T >::reserve (unsigned n) [inline]`

Reserve memory for `n` elements.

Definition at line 448 of file `util/array.hh`.

10.373.4.15 `template<typename T> void array< T >::resize (unsigned n) [inline]`

Resize this array to `n` elements.

Definition at line 456 of file `util/array.hh`.

Referenced by `mln::labeling::impl::generic::compute()`, `mln::labeling::impl::compute_fastest()`, `mln::io::raw::get_header()`, and `mln::io::dump::get_header()`.

10.373.4.16 `template<typename T> void array< T >::resize (unsigned n, const T & value) [inline]`

Resize this array to `n` elements with `value` as value.

Definition at line 464 of file `util/array.hh`.

10.373.4.17 `template<typename T> unsigned array< T >::size () const [inline]`

Return the number of elements of the array.

Added for compatibility with `fun::i2v::array`.

See Also

[nelements](#)

Definition at line 512 of file `util/array.hh`.

Referenced by `mln::labeling::impl::generic::compute()`, `mln::labeling::impl::compute_fastest()`, `mln::value::lut_vec< S, T >::lut_vec()`, and `mln::labeled_image_base< I, E >::update_data()`.

10.373.4.18 `template<typename T> const std::vector< T > & array< T >::std_vector () const [inline]`

Return the corresponding `std::vector` of elements.

Definition at line 586 of file `util/array.hh`.

Referenced by `mln::util::array< T >::append()`, `mln::value::lut_vec< S, T >::lut_vec()`, and `mln::util::operator==()`.

10.374 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.

10.374.1 Detailed Description

`template<typename T>class mln::util::branch< T >`

Class of generic branch.

Definition at line 249 of file tree.hh.

10.374.2 Constructor & Destructor Documentation

10.374.2.1 `template<typename T > branch< T >::branch (util::tree< T > & tree, util::tree_node< T > & apex)`
`[inline]`

Constructor.

Parameters

<code>in</code>	<code>tree</code>	The tree of the branch.
<code>in</code>	<code>apex</code>	The apex of the branch.

Definition at line 537 of file tree.hh.

10.374.3 Member Function Documentation

10.374.3.1 `template<typename T > util::tree_node< T > & branch< T >::apex ()` `[inline]`

The getter of the appex.

Returns

The [tree_node](#) appex of the current branch.

Definition at line 548 of file tree.hh.

10.374.3.2 `template<typename T > mln::util::tree< T > & branch< T >::util_tree ()` `[inline]`

The getter of the tree.

Returns

The tree of the current branch.

Definition at line 556 of file tree.hh.

10.375 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.

10.375.1 Detailed Description

`template<typename T>class mln::util::branch_iter< T >`

Basic 2D image class.

The parameter `T` is the type of node's data. [branch_iter](#) is used to pre-order walk a branch.

Definition at line 52 of file `branch_iter.hh`.

10.375.2 Member Function Documentation

10.375.2.1 `template<typename T > unsigned mln::util::branch_iter< T >::deepness () const` `[inline]`

Give how deep is the iterator in the branch.

Definition at line 119 of file `branch_iter.hh`.

References `mln::util::tree_node< T >::parent()`.

10.375.2.2 `template<typename T > void mln::util::branch_iter< T >::invalidate ()` `[inline]`

Invalidate the iterator.

Definition at line 143 of file `branch_iter.hh`.

10.375.2.3 `template<typename T > bool mln::util::branch_iter< T >::is_valid () const` `[inline]`

Test the iterator validity.

Definition at line 135 of file `branch_iter.hh`.

10.375.2.4 `template<typename T > void mln::util::branch_iter< T >::next ()` `[inline]`

Go to the next point.

Definition at line 162 of file `branch_iter.hh`.

10.375.2.5 `template<typename T > mln::util::branch_iter< T >::operator util::tree_node< T > & () const` `[inline]`

Conversion to node.

Definition at line 101 of file `branch_iter.hh`.

10.375.2.6 `template<typename T> void mln::util::branch_iter< T>::start () [inline]`

Start an iteration.

Definition at line 152 of file `branch_iter.hh`.

10.376 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.

10.376.1 Detailed Description

```
template<typename T> class mln::util::branch_iter_ind< T>
```

Basic 2D image class.

The parameter `T` is the type of node's data. `branch_iter_ind` is used to pre-order walk a branch.

Definition at line 66 of file `branch_iter_ind.hh`.

10.376.2 Member Function Documentation

10.376.2.1 `template<typename T> unsigned mln::util::branch_iter_ind< T>::deepness () const [inline]`

Give how deep is the iterator in the branch.

Definition at line 131 of file `branch_iter_ind.hh`.

References `mln::util::tree_node< T>::parent()`.

10.376.2.2 `template<typename T> void mln::util::branch_iter_ind< T>::invalidate () [inline]`

Invalidate the iterator.

Definition at line 155 of file `branch_iter_ind.hh`.

10.376.2.3 `template<typename T> bool mln::util::branch_iter_ind< T >::is_valid () const [inline]`

Test the iterator validity.

Definition at line 147 of file `branch_iter_ind.hh`.

10.376.2.4 `template<typename T> void mln::util::branch_iter_ind< T >::next () [inline]`

Go to the next point.

Definition at line 174 of file `branch_iter_ind.hh`.

10.376.2.5 `template<typename T> mln::util::branch_iter_ind< T >::operator util::tree_node< T > & () const [inline]`

Conversion to node.

Definition at line 113 of file `branch_iter_ind.hh`.

10.376.2.6 `template<typename T> void mln::util::branch_iter_ind< T >::start () [inline]`

Start an iteration.

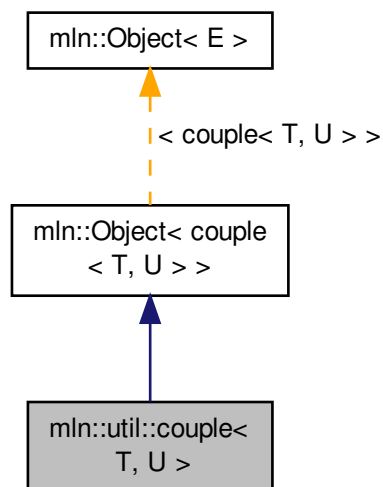
Definition at line 164 of file `branch_iter_ind.hh`.

10.377 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.

10.377.1 Detailed Description

`template<typename T, typename U>class mln::util::couple< T, U >`

Definition of a couple.

Definition at line 48 of file util/couple.hh.

10.377.2 Member Function Documentation

10.377.2.1 `template<typename T, typename U > void mln::util::couple< T, U >::change_both (const T & first, const U & second) [inline]`

Replace both members of the couple by *val*.

Definition at line 182 of file util/couple.hh.

10.377.2.2 `template<typename T, typename U > void mln::util::couple< T, U >::change_first (const T & val) [inline]`

Replace the first member of the couple by *val*.

Definition at line 166 of file util/couple.hh.

10.377.2.3 `template<typename T, typename U > void mln::util::couple< T, U >::change_second (const U & val) [inline]`

Replace the second member of the couple by *val*.

Definition at line 174 of file util/couple.hh.

10.377.2.4 `template<typename T, typename U > const T & mln::util::couple< T, U >::first () const [inline]`

Get the first member of the couple.

Definition at line 134 of file util/couple.hh.

10.377.2.5 `template<typename T , typename U > const U & mln::util::couple< T, U >::second () const` `[inline]`

Get the second member of the couple.

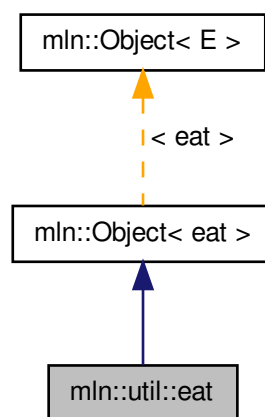
Definition at line 150 of file util/couple.hh.

10.378 mln::util::eat Struct Reference

Eat structure.

```
#include <eat.hh>
```

Inheritance diagram for mln::util::eat:



10.378.1 Detailed Description

Eat structure.

Definition at line 46 of file eat.hh.

10.379 mln::util::edge< G > Class Template Reference

[Edge](#) of a graph [G](#).

```
#include <edge.hh>
```

Inherits mln::util::internal::edge_impl< G >.

Public Types

- typedef [Edge](#)< void > [category](#)
Object category.
- typedef G [graph_t](#)
Graph associated type.

- typedef [edge_id_t id_t](#)
The edge type id.
- typedef [edge_id_t::value_t id_value_t](#)
The underlying type used to store edge ids.

Public Member Functions

- [edge](#) ()
Constructors.
- bool [is_valid](#) () const
Misc.
- void [invalidate](#) ()
Invalidate that vertex.
- [edge_id_t id](#) () const
Return the edge id.
- void [update_id](#) (const [edge_id_t](#) &id)
Set id_ with id;.
- [operator edge_id_t](#) () const
Conversion to the edge id.
- const G & [graph](#) () const
Return a reference to the graph holding this edge.
- void [change_graph](#) (const G &g)
Set g_ with g;.
- [vertex_id_t v_other](#) (const [vertex_id_t](#) &id_v) const
Vertex and edges oriented.
- [vertex_id_t v1](#) () const
Edge oriented.
- [vertex_id_t v2](#) () const
Return the highest vertex id adjacent to this edge.
- size_t [nmax_nbh_edges](#) () const
Return the number max of adjacent edges.
- [edge_id_t ith_nbh_edge](#) (unsigned i) const
Return the i th adjacent edge.

10.379.1 Detailed Description

template<typename G>class mln::util::edge< G >

[Edge](#) of a graph G.

Definition at line 69 of file edge.hh.

10.379.2 Member Typedef Documentation

10.379.2.1 template<typename G> typedef Edge<void> mln::util::edge< G >::category

[Object](#) category.

Definition at line 73 of file edge.hh.

10.379.2.2 `template<typename G> typedef G mln::util::edge< G >::graph_t`

[Graph](#) associated type.

Definition at line 82 of file edge.hh.

10.379.2.3 `template<typename G> typedef edge_id_t mln::util::edge< G >::id_t`

The edge type id.

Definition at line 79 of file edge.hh.

10.379.2.4 `template<typename G> typedef edge_id_t::value_t mln::util::edge< G >::id_value_t`

The underlying type used to store edge ids.

Definition at line 76 of file edge.hh.

10.379.3 Constructor & Destructor Documentation

10.379.3.1 `template<typename G> edge< G >::edge () [inline]`

Constructors.

Definition at line 227 of file edge.hh.

10.379.4 Member Function Documentation

10.379.4.1 `template<typename G> void edge< G >::change_graph (const G & g) [inline]`

Set `g_` with `g`;

Definition at line 290 of file edge.hh.

10.379.4.2 `template<typename G> const G & edge< G >::graph () const [inline]`

Return a reference to the graph holding this edge.

Definition at line 282 of file edge.hh.

Referenced by `mln::p_edges< G, F >::has()`, and `mln::util::line_graph< G >::has()`.

10.379.4.3 `template<typename G> edge_id_t edge< G >::id () const [inline]`

Return the edge id.

Definition at line 259 of file edge.hh.

Referenced by `mln::util::line_graph< G >::has()`.

10.379.4.4 `template<typename G> void edge< G >::invalidate () [inline]`

Invalidate that vertex.

Definition at line 306 of file edge.hh.

10.379.4.5 `template<typename G> bool edge< G >::is_valid () const [inline]`

Misc.

Return whether is points to a known edge.

Definition at line 298 of file edge.hh.

Referenced by `mln::p_edges< G, F >::has()`.

10.379.4.6 `template<typename G> edge_id_t edge< G >::ith_nbh_edge (unsigned i) const [inline]`

Return the `i` th adjacent edge.

Definition at line 351 of file edge.hh.

10.379.4.7 `template<typename G> size_t edge< G >::nmax_nbh_edges () const [inline]`

Return the number max of adjacent edges.

Definition at line 342 of file edge.hh.

10.379.4.8 `template<typename G> edge< G >::operator edge_id_t () const [inline]`

Conversion to the edge id.

Definition at line 274 of file edge.hh.

10.379.4.9 `template<typename G> void edge< G >::update_id (const edge_id_t & id) [inline]`

Set `id_` with `id`;

Definition at line 267 of file edge.hh.

10.379.4.10 `template<typename G> vertex_id_t edge< G >::v1 () const [inline]`

[Edge](#) oriented.

Return the lowest vertex id adjacent to this edge.

Definition at line 324 of file edge.hh.

10.379.4.11 `template<typename G> vertex_id_t edge< G >::v2 () const [inline]`

Return the highest vertex id adjacent to this edge.

Definition at line 333 of file edge.hh.

10.379.4.12 `template<typename G> vertex_id_t edge< G >::v_other (const vertex_id_t & id_v) const [inline]`

[Vertex](#) and edges oriented.

Return the vertex id of this edge which is different from `id_v`.

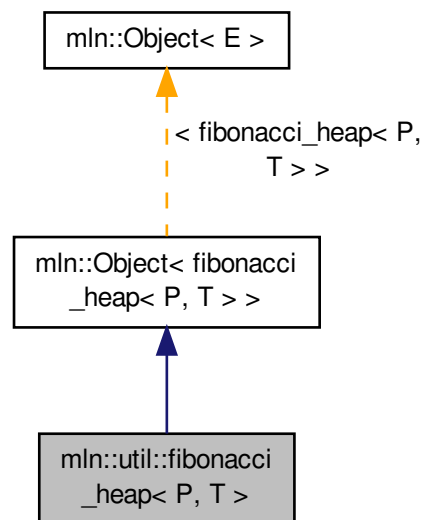
Definition at line 315 of file edge.hh.

10.380 mln::util::fibonacci_heap< P, T > Class Template Reference

Fibonacci heap.

```
#include <fibonacci_heap.hh>
```

Inheritance diagram for mln::util::fibonacci_heap< P, T >:



Public Member Functions

- void `clear` ()
Clear all elements in the heap and make the heap empty.
- `fibonacci_heap` ()
Default constructor.
- `fibonacci_heap` (const `fibonacci_heap`< P, T > &node)
Copy constructor Be ware that once this heap is constructed, the argument node is cleared and all its elements are part of this new heap.
- const T & `front` () const
Return the minimum value in the heap.
- bool `is_empty` () const
Is it empty?
- bool `is_valid` () const
return false if it is empty.
- unsigned `nelements` () const
Return the number of elements.
- `fibonacci_heap`< P, T > & `operator=` (`fibonacci_heap`< P, T > &rhs)
Assignment operator.
- T `pop_front` ()
Return and remove the minimum value in the heap.
- void `push` (const P &priority, const T &value)

Push a new element in the heap.

- void `push (fibonacci_heap< P, T > &other_heap)`

Take `other_heap`'s elements and insert them in this heap.

10.380.1 Detailed Description

```
template<typename P, typename T>class mln::util::fibonacci_heap< P, T >
```

Fibonacci heap.

Definition at line 117 of file fibonacci_heap.hh.

10.380.2 Constructor & Destructor Documentation

10.380.2.1 `template<typename P , typename T > mln::util::fibonacci_heap< P, T >::fibonacci_heap ()`
[inline]

Default constructor.

Definition at line 472 of file fibonacci_heap.hh.

10.380.2.2 `template<typename P , typename T > mln::util::fibonacci_heap< P, T >::fibonacci_heap (const fibonacci_heap< P, T > & node)` [inline]

Copy constructor Be ware that once this heap is constructed, the argument `node` is cleared and all its elements are part of this new heap.

Definition at line 480 of file fibonacci_heap.hh.

10.380.3 Member Function Documentation

10.380.3.1 `template<typename P , typename T > void mln::util::fibonacci_heap< P, T >::clear ()` [inline]

Clear all elements in the heap and make the heap empty.

Definition at line 723 of file fibonacci_heap.hh.

10.380.3.2 `template<typename P , typename T > const T & mln::util::fibonacci_heap< P, T >::front () const`
[inline]

Return the minimum value in the heap.

Definition at line 569 of file fibonacci_heap.hh.

10.380.3.3 `template<typename P , typename T > bool mln::util::fibonacci_heap< P, T >::is_empty () const`
[inline]

Is it empty?

Definition at line 705 of file fibonacci_heap.hh.

Referenced by `mln::util::fibonacci_heap< P, T >::push()`.

10.380.3.4 `template<typename P , typename T > bool mln::util::fibonacci_heap< P, T >::is_valid () const`
`[inline]`

return false if it is empty.

Definition at line 714 of file fibonacci_heap.hh.

10.380.3.5 `template<typename P , typename T > unsigned mln::util::fibonacci_heap< P, T >::nelements () const`
`[inline]`

Return the number of elements.

Definition at line 745 of file fibonacci_heap.hh.

10.380.3.6 `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 cleared and all its elements are part of this new heap.

Definition at line 754 of file fibonacci_heap.hh.

10.380.3.7 `template<typename P , typename T > T mln::util::fibonacci_heap< P, T >::pop_front () [inline]`

Return and remove the minimum value in the heap.

Definition at line 578 of file fibonacci_heap.hh.

10.380.3.8 `template<typename P , typename T > void mln::util::fibonacci_heap< P, T >::push (const P & priority, const`
`T & value) [inline]`

Push a new element in the heap.

See Also

`insert`

Definition at line 508 of file fibonacci_heap.hh.

10.380.3.9 `template<typename P , typename T > void mln::util::fibonacci_heap< P, T >::push (fibonacci_heap< P,`
`T > & other_heap) [inline]`

Take `other_heap`'s elements and insert them in this heap.

After this call `other_heap` is cleared.

Definition at line 520 of file fibonacci_heap.hh.

References `mln::util::fibonacci_heap< P, T >::is_empty()`.

10.381 mln::util::graph Class Reference

Undirected graph.

`#include <graph.hh>`

Inherits `mln::util::internal::graph_base< graph >`.

Public Types

- typedef std::set< edge_data_t > [edges_set_t](#)
A set to test the presence of a given edge.
- typedef std::vector< edge_data_t > [edges_t](#)
The type of the set of edges.
- typedef std::vector
 < vertex_data_t > [vertices_t](#)
The type of the set of vertices.
- typedef
 mln::internal::vertex_nbh_edge_fwd_iterator
 < [graph](#) > [vertex_nbh_edge_fwd_iter](#)
Vertex centered edge iterators.
- typedef
 mln::internal::vertex_nbh_vertex_fwd_iterator
 < [graph](#) > [vertex_nbh_vertex_fwd_iter](#)
Vertex centered vertex iterators.
- typedef
 mln::internal::edge_fwd_iterator
 < [graph](#) > [edge_fwd_iter](#)
Edge iterators.
- typedef
 mln::internal::edge_nbh_edge_fwd_iterator
 < [graph](#) > [edge_nbh_edge_fwd_iter](#)
Edge centered edge iterators.

Public Member Functions

- [graph](#) ()
- [graph](#) (unsigned nvertices)
Construct a graph with `nvertices` vertices.
- bool [has_v](#) (const [vertex_id_t](#) &id_v) const
Check whether a vertex `id_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.
- unsigned [add_vertex](#) ()
Vertex oriented.
- std::pair< [vertex_id_t](#),
[vertex_id_t](#) > [add_vertices](#) (unsigned n)
Add `n` vertices to the graph.
- vertex_t [vertex](#) ([vertex_id_t](#) id_v) const
Return the vertex whose `id` is `v`.

- `edge_id_t add_edge` (const `vertex_id_t` &id_v1, const `vertex_id_t` &id_v2)
Edge oriented.
- `edge_t edge` (const `edge_id_t` &e) const
Return the edge whose id is e.
- `const std::vector`
`< util::ord_pair< vertex_id_t > > & edges` () const
Return the list of all edges.
- `size_t e_nmax` () const
Return the number of edges in the graph.
- `bool has_e` (const `edge_id_t` &id_e) const
Return whether id_e is in the graph.
- `edge_t edge` (const `vertex_t` &v1, const `vertex_t` &v2) const
Return the corresponding edge id if exists.
- `vertex_id_t v1` (const `edge_id_t` &id_e) const
Return the first vertex associated to the edge id_e.
- `vertex_id_t v2` (const `edge_id_t` &id_e) const
Return the second vertex associated to edge id_e.
- `edge_id_t e_ith_nbh_edge` (const `edge_id_t` &id_e, unsigned i) const
Return the i th edge adjacent to the edge id_e.
- `template<typename G2 >`
`bool is_subgraph_of` (const G2 &g) const
*Return whether this graph is a subgraph Return true if g and *this have the same graph_id.*

10.381.1 Detailed Description

Undirected graph.

Definition at line 87 of file mln/util/graph.hh.

10.381.2 Member Typedef Documentation

10.381.2.1 `typedef mln::internal::edge_fwd_iterator<graph> mln::util::graph::edge_fwd_iter`

`Edge` iterators.

Definition at line 129 of file mln/util/graph.hh.

10.381.2.2 `typedef mln::internal::edge_nbh_edge_fwd_iterator<graph> mln::util::graph::edge_nbh_edge_fwd_iter`

`Edge` centered edge iterators.

Definition at line 136 of file mln/util/graph.hh.

10.381.2.3 `typedef std::set<edge_data_t> mln::util::graph::edges_set_t`

A set to test the presence of a given edge.

Definition at line 102 of file mln/util/graph.hh.

10.381.2.4 `typedef std::vector<edge_data_t> mln::util::graph::edges_t`

The type of the set of edges.

Definition at line 100 of file mln/util/graph.hh.

10.381.2.5 `typedef mln::internal::vertex_nbh_edge_fwd_iterator<graph> mln::util::graph::vertex_nbh_edge_fwd_iter`

[Vertex](#) centered edge iterators.

Definition at line 115 of file `mln/util/graph.hh`.

10.381.2.6 `typedef mln::internal::vertex_nbh_vertex_fwd_iterator<graph> mln::util::graph::vertex_nbh_vertex_fwd_iter`

[Vertex](#) centered vertex iterators.

Definition at line 122 of file `mln/util/graph.hh`.

10.381.2.7 `typedef std::vector<vertex_data_t> mln::util::graph::vertices_t`

The type of the set of vertices.

Definition at line 97 of file `mln/util/graph.hh`.

10.381.3 Constructor & Destructor Documentation

10.381.3.1 `mln::util::graph::graph ()` `[inline]`

Constructor.

Definition at line 282 of file `mln/util/graph.hh`.

10.381.3.2 `mln::util::graph::graph (unsigned nvertices)` `[inline]`

Construct a graph with `nvertices` vertices.

Definition at line 288 of file `mln/util/graph.hh`.

10.381.4 Member Function Documentation

10.381.4.1 `edge_id_t mln::util::graph::add_edge (const vertex_id_t & id_v1, const vertex_id_t & id_v2)` `[inline]`

[Edge](#) oriented.

Add an edge.

Returns

The id of the new edge if it does not exist yet; otherwise, return `mln_max(unsigned)`.

Definition at line 386 of file `mln/util/graph.hh`.

References `edge()`, and `has_v()`.

Referenced by `mln::make::voronoi()`.

10.381.4.2 `unsigned mln::util::graph::add_vertex ()` `[inline]`

[Vertex](#) oriented.

Shortcuts factoring the insertion of vertices and edges. Add a vertex.

Returns

The id of the new vertex.

Definition at line 299 of file mln/util/graph.hh.

References `v_nmax()`.

Referenced by `mln::make::voronoi()`.

10.381.4.3 `std::pair< vertex_id_t, vertex_id_t > mln::util::graph::add_vertices (unsigned n) [inline]`

Add `n` vertices to the graph.

Returns

A range of vertex ids.

Definition at line 310 of file mln/util/graph.hh.

References `v_nmax()`.

10.381.4.4 `edge_id_t mln::util::graph::e_ith_nbh_edge (const edge_id_t & id_e, unsigned i) const [inline]`

Return the `i` th edge adjacent to the edge `id_e`.

Definition at line 503 of file mln/util/graph.hh.

References `e_nmax()`, `has_e()`, `v1()`, `v2()`, and `v_ith_nbh_edge()`.

10.381.4.5 `size_t mln::util::graph::e_nmax () const [inline]`

Return the number of edges in the graph.

Definition at line 443 of file mln/util/graph.hh.

Referenced by `e_ith_nbh_edge()`, and `edge()`.

10.381.4.6 `graph::edge_t mln::util::graph::edge (const edge_id_t & e) const [inline]`

Return the edge whose id is `e`.

Definition at line 435 of file mln/util/graph.hh.

References `e_nmax()`.

Referenced by `add_edge()`.

10.381.4.7 `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.

Definition at line 457 of file mln/util/graph.hh.

References `has_v()`.

10.381.4.8 `const std::vector< util::ord_pair< vertex_id_t > > & mln::util::graph::edges () const [inline]`

Return the list of all edges.

Definition at line 428 of file mln/util/graph.hh.

10.381.4.9 `bool mln::util::graph::has_e (const edge_id_t & id_e) const [inline]`

Return whether `id_e` is in the graph.

Definition at line 450 of file `mln/util/graph.hh`.

Referenced by `e_ith_nbh_edge()`, `v1()`, and `v2()`.

10.381.4.10 `bool mln::util::graph::has_v (const vertex_id_t & id_v) const [inline]`

Check whether a vertex id `id_v` exists in the graph.

Definition at line 338 of file `mln/util/graph.hh`.

Referenced by `add_edge()`, `edge()`, `v_ith_nbh_edge()`, `v_ith_nbh_vertex()`, and `vertex()`.

10.381.4.11 `template<typename G2> bool mln::util::graph::is_subgraph_of (const G2 & g) const [inline]`

Return whether this graph is a subgraph Return true if `g` and `*this` have the same `graph_id`.

Definition at line 519 of file `mln/util/graph.hh`.

10.381.4.12 `vertex_id_t mln::util::graph::v1 (const edge_id_t & id_e) const [inline]`

Return the first vertex associated to the edge `id_e`.

Definition at line 479 of file `mln/util/graph.hh`.

References `has_e()`.

Referenced by `e_ith_nbh_edge()`.

10.381.4.13 `vertex_id_t mln::util::graph::v2 (const edge_id_t & id_e) const [inline]`

Return the second vertex associated to edge `id_e`.

Definition at line 487 of file `mln/util/graph.hh`.

References `has_e()`.

Referenced by `e_ith_nbh_edge()`.

10.381.4.14 `edge_id_t mln::util::graph::v_ith_nbh_edge (const vertex_id_t & id_v, unsigned i) const [inline]`

Returns the `i` th edge adjacent to the vertex `id_v`.

Definition at line 353 of file `mln/util/graph.hh`.

References `has_v()`.

Referenced by `e_ith_nbh_edge()`, and `v_ith_nbh_vertex()`.

10.381.4.15 `vertex_id_t mln::util::graph::v_ith_nbh_vertex (const vertex_id_t & id_v, unsigned i) const [inline]`

Returns the `i` th vertex adjacent to the vertex `id_v`.

Definition at line 371 of file `mln/util/graph.hh`.

References `has_v()`, and `v_ith_nbh_edge()`.

10.381.4.16 `size_t mln::util::graph::v_nmax () const` `[inline]`

Return the number of vertices in the graph.

Definition at line 331 of file mln/util/graph.hh.

Referenced by `add_vertex()`, and `add_vertices()`.

10.381.4.17 `graph::vertex_t mln::util::graph::vertex (vertex_id_t id_v) const` `[inline]`

Return the vertex whose id is *v*.

Definition at line 322 of file mln/util/graph.hh.

References `has_v()`.

10.382 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?

10.382.1 Detailed Description

```
template<typename I>class mln::util::greater_point< I >
```

A “greater than” functor comparing points w.r.t.

the values they refer to in an image.

This functor used in useful to implement ordered queues of points.

Definition at line 42 of file `greater_point.hh`.

10.382.2 Member Function Documentation

10.382.2.1 `template<typename I > bool mln::util::greater_point< I >::operator() (const point & x, const point & y)`

Is *x* greater than *y*?

Definition at line 74 of file `greater_point.hh`.

10.383 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?

10.383.1 Detailed Description

```
template<typename I>class mln::util::greater_psite< I >
```

A “greater than” functor comparing psites w.r.t.
the values they refer to in an image.

This functor used in useful to implement ordered queues of psites.

Definition at line 42 of file greater_psite.hh.

10.383.2 Member Function Documentation

10.383.2.1 `template<typename I > bool mln::util::greater_psite< I >::operator() (const psite & x, const psite & y)`

Is x greater than y?

Definition at line 74 of file greater_psite.hh.

10.384 mln::util::head< T, R > Class Template Reference

Top structure of the soft heap.

```
#include <soft_heap.hh>
```

10.384.1 Detailed Description

```
template<typename T, typename R>class mln::util::head< T, R >
```

Top structure of the soft heap.

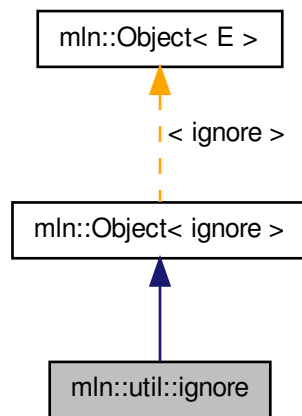
Definition at line 138 of file soft_heap.hh.

10.385 mln::util::ignore Struct Reference

Ignore structure.

```
#include <ignore.hh>
```


Inheritance diagram for mln::util::ignore:



10.385.1 Detailed Description

Ignore structure.

Definition at line 46 of file ignore.hh.

10.386 mln::util::ilcell< T > Struct Template Reference

Element of an item list. Store the data (key) used in [soft_heap](#).

```
#include <soft_heap.hh>
```

10.386.1 Detailed Description

```
template<typename T>struct mln::util::ilcell< T >
```

Element of an item list. Store the data (key) used in [soft_heap](#).

Definition at line 76 of file soft_heap.hh.

10.387 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< 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_nbh_vertex_fwd_iterator
< [line_graph](#)< G > > [vertex_nbh_vertex_fwd_iter](#)
[Vertex](#) nbh vertex iterators.
- typedef
mln::internal::vertex_nbh_edge_fwd_iterator
< [line_graph](#)< G > > [vertex_nbh_edge_fwd_iter](#)
[Vertex](#) nbh edge iterators.

Public Member Functions

- template<typename G2 >
bool [has](#) (const [util::vertex](#)< G2 > &v) const
Check whether a vertex v exists in the line graph.
- bool [has_v](#) (const [vertex_id_t](#) &id_v) const
Check whether a vertex id id_v exists in the line graph.
- [edge_id_t v_ith_nbh_edge](#) (const [vertex_id_t](#) &id_v, unsigned i) const
Returns the i th edge adjacent to the vertex id_v .
- [vertex_id_t v_ith_nbh_vertex](#) (const [vertex_id_t](#) &id_v, unsigned i) const
Returns the i th vertex adjacent to the vertex id_v .
- size_t [v_nmax](#) () const
Return the number of vertices in the graph.
- [vertex_t vertex](#) (const [vertex_id_t](#) &id_v) const
[Vertex](#) oriented.
- [edge_t edge](#) (const [edge_id_t](#) &e) const
[Edge](#) oriented.
- size_t [e_nmax](#) () const
Return the number of edges in the graph.
- bool [has_e](#) (const [util::edge_id_t](#) &id_e) const
Return whether id_e is in the line graph.
- template<typename G2 >
bool [has](#) (const [util::edge](#)< G2 > &e) const
Return whether e is in the line graph.
- [vertex_id_t v1](#) (const [edge_id_t](#) &id_e) const

- Return the first vertex associated to the edge `id_e`.*
- `vertex_id_t v2` (const `edge_id_t` &`id_e`) const
- Return the second vertex associated to edge `id_e`.*
- `edge_id_t e_ith_nbh_edge` (const `edge_id_t` &`id_e`, unsigned `i`) const
- Return the `i` th edge adjacent to the edge `id_e`.*
- `template<typename G2 >`
`bool is_subgraph_of` (const `G2` &`g`) const
- Return whether this graph is a subgraph Return true if `g` and *this have the same `graph_id`.*
- `const G & graph` () const
- Return the underlying graph.*

10.387.1 Detailed Description

`template<typename G>class mln::util::line_graph< G >`

Undirected line graph of a graph of type `G`.

Definition at line 82 of file `line_graph.hh`.

10.387.2 Member Typedef Documentation

10.387.2.1 `template<typename G> typedef mln::internal::edge_fwd_iterator< line_graph<G> >`
`mln::util::line_graph< G >::edge_fwd_iter`

Edge iterators.

Definition at line 114 of file `line_graph.hh`.

10.387.2.2 `template<typename G> typedef mln::internal::edge_nbh_edge_fwd_iterator< line_graph<G> >`
`mln::util::line_graph< G >::edge_nbh_edge_fwd_iter`

Edge nbh edge iterators.

Definition at line 123 of file `line_graph.hh`.

10.387.2.3 `template<typename G> typedef std::vector<edge_data_t> mln::util::line_graph< G >::edges_t`

The type of the set of edges.

Definition at line 98 of file `line_graph.hh`.

10.387.2.4 `template<typename G> typedef mln::internal::vertex_nbh_edge_fwd_iterator< line_graph<G> >`
`mln::util::line_graph< G >::vertex_nbh_edge_fwd_iter`

Vertex nbh edge iterators.

Definition at line 141 of file `line_graph.hh`.

10.387.2.5 `template<typename G> typedef mln::internal::vertex_nbh_vertex_fwd_iterator< line_graph<G> >`
`mln::util::line_graph< G >::vertex_nbh_vertex_fwd_iter`

Vertex nbh vertex iterators.

Definition at line 132 of file `line_graph.hh`.

10.387.2.6 `template<typename G> typedef std::vector<vertex_data_t> mln::util::line_graph< G >::vertices_t`

The type of the set of vertices.

Definition at line 95 of file line_graph.hh.

10.387.3 Member Function Documentation

10.387.3.1 `template<typename G> edge_id_t line_graph< G >::e_ith_nbh_edge (const edge_id_t & id_e, unsigned i) const [inline]`

Return the *i* th edge adjacent to the edge *id_e*.

Definition at line 460 of file line_graph.hh.

10.387.3.2 `template<typename G> size_t line_graph< G >::e_nmax () const [inline]`

Return the number of edges in the graph.

Definition at line 408 of file line_graph.hh.

10.387.3.3 `template<typename G> line_graph< G >::edge_t line_graph< G >::edge (const edge_id_t & e) const [inline]`

[Edge](#) oriented.

Return the edge whose id is *e*.

Definition at line 399 of file line_graph.hh.

10.387.3.4 `template<typename G> const G & line_graph< G >::graph () const [inline]`

Return the underlying graph.

Definition at line 485 of file line_graph.hh.

10.387.3.5 `template<typename G> template<typename G2> bool line_graph< G >::has (const util::vertex< G2 > & v) const [inline]`

Check whether a vertex *v* exists in the line graph.

Definition at line 345 of file line_graph.hh.

References `mln::util::vertex< G >::graph()`, and `mln::util::vertex< G >::id()`.

10.387.3.6 `template<typename G> template<typename G2> bool line_graph< G >::has (const util::edge< G2 > & e) const [inline]`

Return whether *e* is in the line graph.

Definition at line 425 of file line_graph.hh.

References `mln::util::edge< G >::graph()`, and `mln::util::edge< G >::id()`.

10.387.3.7 `template<typename G> bool line_graph< G >::has_e (const util::edge_id_t & id_e) const [inline]`

Return whether *id_e* is in the line graph.

Definition at line 416 of file line_graph.hh.

10.387.3.8 `template<typename G> bool line_graph< G >::has_v (const vertex_id_t & id_v) const` `[inline]`

Check whether a vertex id `id_v` exists in the line graph.

Definition at line 336 of file `line_graph.hh`.

10.387.3.9 `template<typename G> template<typename G2> bool line_graph< G >::is_subgraph_of (const G2 & g) const` `[inline]`

Return whether this graph is a subgraph Return true if `g` and `*this` have the same `graph_id`.

Definition at line 477 of file `line_graph.hh`.

10.387.3.10 `template<typename G> vertex_id_t line_graph< G >::v1 (const edge_id_t & id_e) const` `[inline]`

Return the first vertex associated to the edge `id_e`.

Definition at line 433 of file `line_graph.hh`.

10.387.3.11 `template<typename G> vertex_id_t line_graph< G >::v2 (const edge_id_t & id_e) const` `[inline]`

Return the second vertex associated to edge `id_e`.

Definition at line 442 of file `line_graph.hh`.

10.387.3.12 `template<typename G> edge_id_t line_graph< G >::v_ith_nbh_edge (const vertex_id_t & id_v, unsigned i) const` `[inline]`

Returns the `i` th edge adjacent to the vertex `id_v`.

Definition at line 363 of file `line_graph.hh`.

10.387.3.13 `template<typename G> vertex_id_t line_graph< G >::v_ith_nbh_vertex (const vertex_id_t & id_v, unsigned i) const` `[inline]`

Returns the `i` th vertex adjacent to the vertex `id_v`.

Definition at line 383 of file `line_graph.hh`.

10.387.3.14 `template<typename G> size_t line_graph< G >::v_nmax () const` `[inline]`

Return the number of vertices in the graph.

Definition at line 328 of file `line_graph.hh`.

10.387.3.15 `template<typename G> line_graph< G >::vertex_t line_graph< G >::vertex (const vertex_id_t & id_v) const` `[inline]`

[Vertex](#) oriented.

Shortcuts factoring the insertion of vertices and edges.

Return the vertex whose id is `v`.

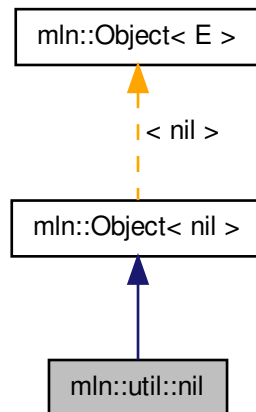
Definition at line 318 of file `line_graph.hh`.

10.388 mln::util::nil Struct Reference

Nil structure.

```
#include <nil.hh>
```

Inheritance diagram for mln::util::nil:



10.388.1 Detailed Description

Nil structure.

Definition at line 46 of file `util/nil.hh`.

10.389 mln::util::node< T, R > Class Template Reference

Meta-data of an element in the heap.

```
#include <soft_heap.hh>
```

10.389.1 Detailed Description

```
template<typename T, typename R>class mln::util::node< T, R >
```

Meta-data of an element in the heap.

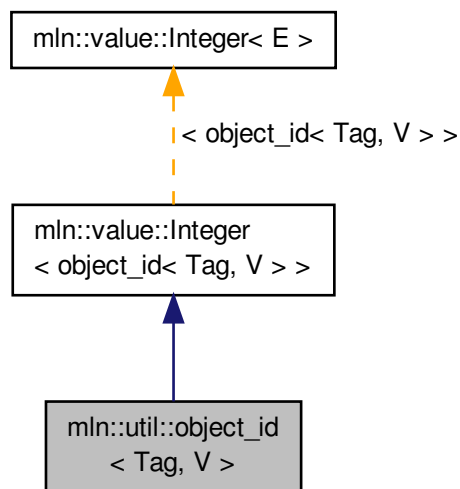
Definition at line 97 of file `soft_heap.hh`.

10.390 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.

10.390.1 Detailed Description

```
template<typename Tag, typename V>class mln::util::object_id< Tag, V >
```

Base class of an object id.

Template Parameters

<i>Tag</i>	the tag type
<i>Equiv</i>	the equivalent value.

Definition at line 67 of file `object_id.hh`.

10.390.2 Member Typedef Documentation

10.390.2.1 `template<typename Tag, typename V> typedef V mln::util::object_id< Tag, V >::value_t`

The underlying type id.

Definition at line 71 of file object_id.hh.

10.390.3 Constructor & Destructor Documentation

10.390.3.1 `template<typename Tag , typename V > object_id< Tag, V >::object_id ()` `[inline]`

Constructors.

Definition at line 121 of file object_id.hh.

10.391 mln::util::ord< T > Struct Template Reference

Function-object that defines an ordering between objects with type `T`: *lhs R rhs*.

```
#include <ord.hh>
```

10.391.1 Detailed Description

```
template<typename T>struct mln::util::ord< T >
```

Function-object that defines an ordering between objects with type `T`: *lhs R rhs*.

Its meaning is "lhs less-than rhs."

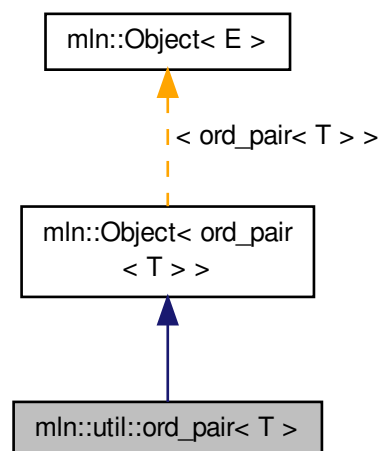
Definition at line 48 of file util/ord.hh.

10.392 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.

10.392.1 Detailed Description

template<typename T>struct mln::util::ord_pair< T >

Ordered pair structure s.a.

this->first <= this->second; ordered pairs are partially ordered using lexicographical ordering.

Definition at line 50 of file ord_pair.hh.

10.392.2 Member Function Documentation

10.392.2.1 template<typename T > void mln::util::ord_pair< T >::change_both (const T & [first](#), const T & [second](#))
[inline]

Replace both members of the pair by *val*, while keeping the relative order.

Postcondition

first_ <= second_ (with <= being the [mln::util::ord_weak](#) relationship).

Definition at line 211 of file ord_pair.hh.

References [mln::util::ord_strict\(\)](#), and [mln::util::ord_weak\(\)](#).

10.392.2.2 template<typename T > void mln::util::ord_pair< T >::change_first (const T & [val](#)) [inline]

Replace the first member of the pair by *val*, while keeping the relative order.

Postcondition

first_ <= second_ (with <= being the [mln::util::ord_weak](#) relationship).

Definition at line 181 of file ord_pair.hh.

References [mln::util::ord_strict\(\)](#), and [mln::util::ord_weak\(\)](#).

10.392.2.3 `template<typename T> void mln::util::ord_pair< T>::change_second (const T & val) [inline]`

Replace the second member of the pair by *val*, while keeping the relative order.

Postcondition

first_ <= *second_* (with <= being the [mln::util::ord_weak](#) relationship).

Definition at line 196 of file `ord_pair.hh`.

References `mln::util::ord_strict()`, and `mln::util::ord_weak()`.

10.392.2.4 `template<typename T> const T & mln::util::ord_pair< T>::first () const [inline]`

Get the first (lowest) member of the pair.

Definition at line 149 of file `ord_pair.hh`.

10.392.2.5 `template<typename T> const T & mln::util::ord_pair< T>::second () const [inline]`

Get the second (highest) member of the pair.

Definition at line 165 of file `ord_pair.hh`.

10.393 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.

10.393.1 Detailed Description

```
template<typename I> struct mln::util::pix< I>
```

Structure `pix`.

Definition at line 69 of file `util/pix.hh`.

10.393.2 Member Typedef Documentation

10.393.2.1 `template<typename I> typedef I::psite mln::util::pix< I >::psite`

Point_Site associated type.

Definition at line 73 of file util/pix.hh.

10.393.2.2 `template<typename I> typedef I::value mln::util::pix< I >::value`

Value associated type.

Definition at line 76 of file util/pix.hh.

10.393.3 Constructor & Destructor Documentation

10.393.3.1 `template<typename I> mln::util::pix< I >::pix (const Image< I > & ima, const typename I::psite & p) [inline]`

Constructor.

Parameters

in	<i>ima</i>	The image.
in	<i>p</i>	The p_site.

Definition at line 121 of file util/pix.hh.

10.393.4 Member Function Documentation

10.393.4.1 `template<typename I> const I & mln::util::pix< I >::ima () const [inline]`

The getter of the image associate to pix structure.

Returns

The image ima_.

Definition at line 131 of file util/pix.hh.

10.393.4.2 `template<typename I> const I::psite & mln::util::pix< I >::p () const [inline]`

The getter of psite associate to pix structure.

Returns

The psite p_.

Definition at line 140 of file util/pix.hh.

10.393.4.3 `template<typename I> I::rvalue mln::util::pix< I >::v () const [inline]`

The getter of value associate to pix structure.

Returns

The value of pix.

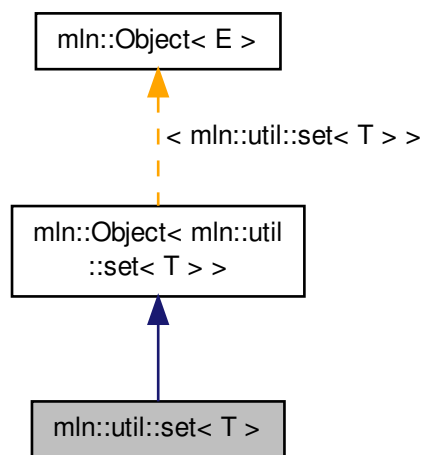
Definition at line 150 of file util/pix.hh.

10.394 mln::util::set< T > Class Template Reference

An "efficient" mathematical set class.

```
#include <set.hh>
```

Inheritance diagram for mln::util::set< T >:

**Public Types**

- typedef set_bkd_iter< T > [bkd_eiter](#)
Backward iterator associated type.
- typedef [fwd_eiter](#) eiter
Iterator associated type.
- typedef T [element](#)
Element associated type.
- typedef set_fwd_iter< T > [fwd_eiter](#)
Forward iterator associated type.

Public Member Functions

- void [clear](#) ()
Empty the set.
- const T [first_element](#) () const
Return the first element of the set.
- bool [has](#) (const T &elt) const

- Test if the object `elt` belongs to the set.*

 - `set< T > & insert (const T &elt)`

Insert an element `elt` into the set.
- `template<typename U >`
`set< T > & insert (const set< U > &other)`

Insert the elements of `other` into the set.
- `bool is_empty () const`

Test if the set is empty.
- `const T last_element () const`

Return the last element of the set.
- `std::size_t memory_size () const`

Return the size of this set in memory.
- `unsigned nelements () const`

Return the number of elements of the set.
- `const T & operator[] (unsigned i) const`

*Return the *i*-th element of the set.*
- `set< T > & remove (const T &elt)`

Remove an element `elt` into the set.
- `set ()`

Constructor without arguments.
- `const std::vector< T > & std_vector () const`

Give access to the set elements.

10.394.1 Detailed Description

`template<typename T>class mln::util::set< T >`

An "efficient" mathematical set class.

This set class is designed to store a mathematical set and to present it to the user as a linear array (`std::vector`).

Elements are stored by copy. Implementation is lazy.

The set has two states: frozen or not. There is an automatic switch of state when the user modifies its contents (insert, remove, or clear) or access to its contents (`op[i]`).

The parameter `T` is the element type, which shall not be const-qualified.

The unicity of set elements is handled by the `mln::util::ord` mechanism.

See Also

[mln::util::ord](#)

Definition at line 81 of file `util/set.hh`.

10.394.2 Member Typedef Documentation

10.394.2.1 `template<typename T> typedef set_bkd_iter<T> mln::util::set< T >::bkd_eiter`

Backward iterator associated type.

Definition at line 93 of file `util/set.hh`.

10.394.2.2 `template<typename T> typedef fwd_eiter mln::util::set< T >::eiter`

[Iterator](#) associated type.

Definition at line 96 of file util/set.hh.

10.394.2.3 `template<typename T> typedef T mln::util::set< T >::element`

Element associated type.

Definition at line 86 of file util/set.hh.

10.394.2.4 `template<typename T> typedef set_fwd_iter<T> mln::util::set< T >::fwd_eiter`

Forward iterator associated type.

Definition at line 90 of file util/set.hh.

10.394.3 Constructor & Destructor Documentation

10.394.3.1 `template<typename T > mln::util::set< T >::set () [inline]`

Constructor without arguments.

Definition at line 348 of file util/set.hh.

10.394.4 Member Function Documentation

10.394.4.1 `template<typename T > void mln::util::set< T >::clear () [inline]`

Empty the set.

All elements contained in the set are destroyed so the set is emptied.

Postcondition

`is_empty() == true`

Definition at line 390 of file util/set.hh.

10.394.4.2 `template<typename T > const T mln::util::set< T >::first_element () const [inline]`

Return the first element of the set.

Precondition

not `is_empty()`

Definition at line 427 of file util/set.hh.

10.394.4.3 `template<typename T > bool mln::util::set< T >::has (const T & elt) const [inline]`

Test if the object `elt` belongs to the set.

Parameters

<code>in</code>	<code>elt</code>	A possible element of the set.
-----------------	------------------	--------------------------------

Returns

True is `elt` is in the set.

Definition at line 445 of file `util/set.hh`.

10.394.4.4 `template<typename T> set< T> & mln::util::set< T>::insert (const T & elt) [inline]`

Insert an element `elt` into the set.

Parameters

<code>in</code>	<code>elt</code>	The element to be inserted.
-----------------	------------------	-----------------------------

If `elt` is already in the set, this method is a no-op.

Returns

The set itself after insertion.

Definition at line 356 of file `util/set.hh`.

Referenced by `mln::p_key< K, P>::change_keys()`.

10.394.4.5 `template<typename T> template<typename U> set< T> & mln::util::set< T>::insert (const set< U> & other) [inline]`

Insert the elements of `other` into the set.

Parameters

<code>in</code>	<code>other</code>	The set containing the elements to be inserted.
-----------------	--------------------	---

Returns

The set itself after insertion.

Definition at line 367 of file `util/set.hh`.

References `mln::util::set< T>::is_empty()`, and `mln::util::set< T>::std_vector()`.

10.394.4.6 `template<typename T> bool mln::util::set< T>::is_empty () const [inline]`

Test if the set is empty.

Definition at line 453 of file `util/set.hh`.

Referenced by `mln::util::set< T>::insert()`.

10.394.4.7 `template<typename T> const T mln::util::set< T>::last_element () const [inline]`

Return the last element of the set.

Precondition

not `is_empty()`

Definition at line 436 of file `util/set.hh`.

10.394.4.8 `template<typename T> std::size_t mln::util::set<T>::memory_size () const [inline]`

Return the size of this set in memory.

Definition at line 494 of file util/set.hh.

10.394.4.9 `template<typename T> unsigned mln::util::set<T>::nelements () const [inline]`

Return the number of elements of the set.

Definition at line 409 of file util/set.hh.

10.394.4.10 `template<typename T> const T & mln::util::set<T>::operator[] (unsigned i) const [inline]`

Return the i-th element of the set.

Parameters

<code>in</code>	<code>i</code>	Index of the element to retrieve.
-----------------	----------------	-----------------------------------

Precondition

`i < nelements()`

The element is returned by reference and is constant.

Definition at line 417 of file util/set.hh.

10.394.4.11 `template<typename T> set<T> & mln::util::set<T>::remove (const T & elt) [inline]`

Remove an element `elt` into the set.

Parameters

<code>in</code>	<code>elt</code>	The element to be inserted.
-----------------	------------------	-----------------------------

If `elt` is already in the set, this method is a no-op.

Returns

The set itself after suppression.

Definition at line 380 of file util/set.hh.

10.394.4.12 `template<typename T> const std::vector<T> & mln::util::set<T>::std_vector () const [inline]`

Give access to the set elements.

The complexity of this method is O(1).

Postcondition

The set is frozen.

Returns

An array (std::vector) of elements.

Definition at line 461 of file util/set.hh.

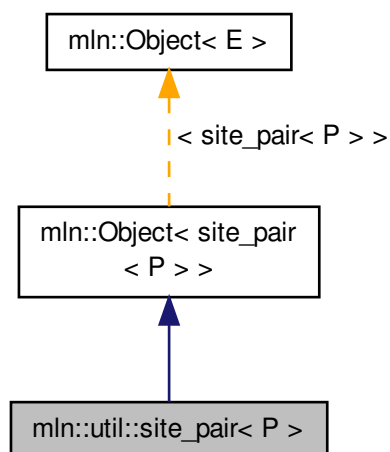
Referenced by `mln::util::set<T>::insert()`.

10.395 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.

10.395.1 Detailed Description

```
template<typename P>class mln::util::site_pair< P >
```

A pair of sites.

It can be used as site.

Definition at line 52 of file `site_pair.hh`.

10.395.2 Member Function Documentation

10.395.2.1 `template<typename P > const P & mln::util::site_pair< P >::first () const` `[inline]`

Return the first site.

Definition at line 142 of file `site_pair.hh`.

10.395.2.2 `template<typename P> const util::ord_pair< P> & mln::util::site_pair< P>::pair () const`
`[inline]`

Return the underlying pair.

Definition at line 158 of file site_pair.hh.

10.395.2.3 `template<typename P> const P & mln::util::site_pair< P>::second () const` `[inline]`

Return the second site.

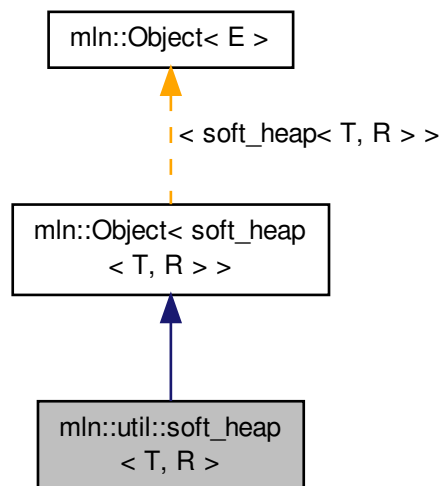
Definition at line 150 of file site_pair.hh.

10.396 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` (const T &`element`)
Add a new element `element`.
- void `push` (soft_heap< T, R > &`sh`)
Merge `sh` with this heap.
- `soft_heap` (unsigned `r`=20)
Default constructor.
- `~soft_heap` ()
Destructor.

10.396.1 Detailed Description

template<typename T, typename R>class mln::util::soft_heap< T, R >

Soft heap.

T key, the data to store in the heap. For instance a point 2d. R rank, for instance int_u8

Definition at line 178 of file soft_heap.hh.

10.396.2 Member Typedef Documentation

10.396.2.1 template<typename T, typename R> typedef T mln::util::soft_heap< T, R >::element

Element associated type.

Definition at line 185 of file soft_heap.hh.

10.396.3 Constructor & Destructor Documentation

10.396.3.1 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.

Definition at line 619 of file soft_heap.hh.

10.396.3.2 template<typename T , typename R > mln::util::soft_heap< T, R >::~~soft_heap () [inline]

Destructor.

Definition at line 631 of file soft_heap.hh.

10.396.4 Member Function Documentation

10.396.4.1 `template<typename T , typename R > void mln::util::soft_heap< T, R >::clear () [inline]`

Clear the heap.

Definition at line 771 of file `soft_heap.hh`.

10.396.4.2 `template<typename T , typename R > bool mln::util::soft_heap< T, R >::is_empty () const [inline]`

Return true if there is at least one element.

Definition at line 753 of file `soft_heap.hh`.

10.396.4.3 `template<typename T , typename R > bool mln::util::soft_heap< T, R >::is_valid () const [inline]`

Return true if there is at least one element.

Definition at line 744 of file `soft_heap.hh`.

10.396.4.4 `template<typename T , typename R > int mln::util::soft_heap< T, R >::nelements () const [inline]`

Return the number of element in the heap.

Definition at line 762 of file `soft_heap.hh`.

Referenced by `mln::util::soft_heap< T, R >::push()`.

10.396.4.5 `template<typename T , typename R > T mln::util::soft_heap< T, R >::pop_front () [inline]`

Returns the element with the lowest priority and remove it from the heap.

Definition at line 675 of file `soft_heap.hh`.

10.396.4.6 `template<typename T , typename R > void mln::util::soft_heap< T, R >::push (const T & element) [inline]`

Add a new element `element`.

Definition at line 646 of file `soft_heap.hh`.

10.396.4.7 `template<typename T , typename R > void mln::util::soft_heap< T, R >::push (soft_heap< T, R > & sh) [inline]`

Merge `sh` with this heap.

Be ware that after this call, `sh` will be empty. This heap will hold the elements which were part of `sh`.

Definition at line 658 of file `soft_heap.hh`.

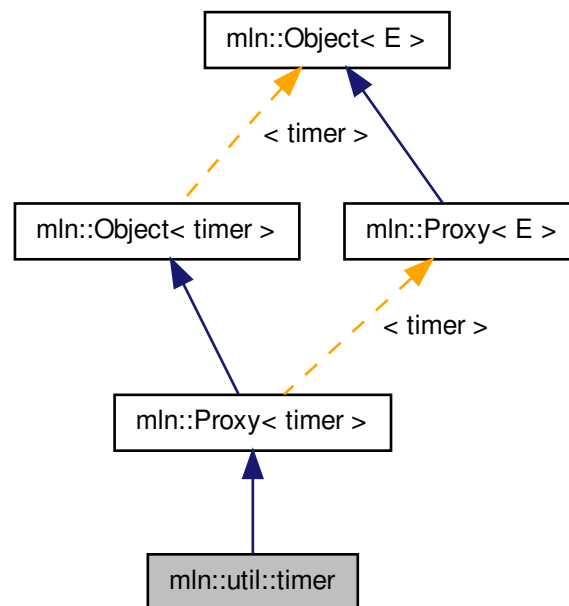
References `mln::util::soft_heap< T, R >::nelements()`.

10.397 mln::util::timer Class Reference

Timer structure.

```
#include <timer.hh>
```

Inheritance diagram for mln::util::timer:



10.397.1 Detailed Description

Timer structure.

Definition at line 45 of file mln/util/timer.hh.

10.398 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).
- `const T * operator-> () const`
Mimics the behavior of op-> for a pointer in the const case.
- `T * operator-> ()`
Mimics the behavior of op-> for a pointer in the mutable case.
- `tracked_ptr< T > & operator= (const tracked_ptr< T > &rhs)`
Assignment.
- `tracked_ptr< T > & operator= (T *ptr)`

Assignment.

- `~tracked_ptr()`

Destructor.

- `tracked_ptr()`

Constructors.

- `tracked_ptr(const tracked_ptr< T > &rhs)`

Copy constructor.

10.398.1 Detailed Description

```
template<typename T>struct mln::util::tracked_ptr< T >
```

Smart pointer for shared data with tracking.

Definition at line 52 of file tracked_ptr.hh.

10.398.2 Constructor & Destructor Documentation

10.398.2.1 `template<typename T > mln::util::tracked_ptr< T >::tracked_ptr()` `[inline]`

Constructors.

Definition at line 140 of file tracked_ptr.hh.

10.398.2.2 `template<typename T > mln::util::tracked_ptr< T >::tracked_ptr(const tracked_ptr< T > & rhs)`
`[inline]`

Copy constructor.

Definition at line 164 of file tracked_ptr.hh.

10.398.2.3 `template<typename T > mln::util::tracked_ptr< T >::~~tracked_ptr()` `[inline]`

Destructor.

Definition at line 216 of file tracked_ptr.hh.

10.398.3 Member Function Documentation

10.398.3.1 `template<typename T > mln::util::tracked_ptr< T >::operator bool() const` `[inline]`

Coercion towards Boolean (for arithmetical tests).

Definition at line 106 of file tracked_ptr.hh.

10.398.3.2 `template<typename T > bool mln::util::tracked_ptr< T >::operator!() const` `[inline]`

Negation (for arithmetical tests).

Definition at line 114 of file tracked_ptr.hh.

10.398.3.3 `template<typename T > const T * mln::util::tracked_ptr< T >::operator-> () const` `[inline]`

Mimics the behavior of `op->` for a pointer in the const case.

Invariant

Pointer proxy exists.

Definition at line 122 of file `tracked_ptr.hh`.

10.398.3.4 `template<typename T > T * mln::util::tracked_ptr< T >::operator-> ()` `[inline]`

Mimics the behavior of `op->` for a pointer in the mutable case.

Invariant

Pointer proxy exists.

Definition at line 131 of file `tracked_ptr.hh`.

10.398.3.5 `template<typename T > tracked_ptr< T > & mln::util::tracked_ptr< T >::operator= (const tracked_ptr< T > & rhs)` `[inline]`

Assignment.

Definition at line 176 of file `tracked_ptr.hh`.

10.398.3.6 `template<typename T > tracked_ptr< T > & mln::util::tracked_ptr< T >::operator= (T * ptr)` `[inline]`

Assignment.

Definition at line 195 of file `tracked_ptr.hh`.

10.399 mln::util::tree< T > Class Template Reference

Class of generic tree.

```
#include <tree.hh>
```

Public Member Functions

- void `add_tree_down` (T &elt)
Bind a new tree downer the current.
- void `add_tree_up` (T &elt)
Bind a new tree upper the current.
- bool `check_consistency` ()
Check the consistency of the tree.
- `branch`< T > `main_branch` ()
Convert the tree into brach.
- `tree_node`< T > * `root` ()
The getter of the root.
- `tree` ()
Constructor.
- `tree` (`tree_node`< T > *root)
Constructor.

10.399.1 Detailed Description

`template<typename T>class mln::util::tree< T >`

Class of generic tree.

Definition at line 187 of file tree.hh.

10.399.2 Constructor & Destructor Documentation

10.399.2.1 `template<typename T > tree< T >::tree () [inline]`

Constructor.

Definition at line 285 of file tree.hh.

10.399.2.2 `template<typename T > tree< T >::tree (tree_node< T > * root) [inline]`

Constructor.

Parameters

<i>in</i>	<i>root</i>	The root of the tree.
-----------	-------------	-----------------------

Definition at line 292 of file tree.hh.

10.399.3 Member Function Documentation

10.399.3.1 `template<typename T > void tree< T >::add_tree_down (T & elt) [inline]`

Bind a new tree downner the current.

Parameters

<i>in</i>	<i>elt</i>	The new value of the new tree_node of the new tree add downner the current.
-----------	------------	---

Definition at line 328 of file tree.hh.

10.399.3.2 `template<typename T > void tree< T >::add_tree_up (T & elt) [inline]`

Bind a new tree upper the current.

Parameters

<i>in</i>	<i>elt</i>	The new value of the new tree_node of the new tree add upper the current.
-----------	------------	---

Definition at line 317 of file tree.hh.

References `mln::util::tree_node< T >::children()`, and `mln::util::tree_node< T >::set_parent()`.

10.399.3.3 `template<typename T > bool tree< T >::check_consistency () [inline]`

Check the consistency of the tree.

Returns

true if no error, else false.

Definition at line 338 of file tree.hh.

10.399.3.4 `template<typename T> branch< T> tree< T>::main_branch () [inline]`

Convert the tree into brach.

Returns

The root's [tree_node](#) of the the current tree.

Definition at line 309 of file tree.hh.

10.399.3.5 `template<typename T> tree_node< T> * tree< T>::root () [inline]`

The getter of the root.

Returns

The root's [tree_node](#) of the the current tree.

Definition at line 301 of file tree.hh.

Referenced by `mln::util::display_tree()`, and `mln::util::tree_to_fast()`.

10.400 mln::util::tree_node< T > Class Template Reference

Class of generic [tree_node](#) for tree.

```
#include <tree.hh>
```

Public Member Functions

- [tree_node](#)< T > * [add_child](#) (T elt)
Create a [tree_node](#) with elt which become the child of the current [tree_node](#).
- [tree_node](#)< T > * [add_child](#) ([tree_node](#)< T > *[tree_node](#))
Bind [tree_node](#) to the current [tree_node](#) and become its child.
- bool [check_consistency](#) ()
Check the consistency of the [tree_node](#).
- children_t & [children](#) ()
The getter of the children.
- const children_t & [children](#) () const
The getter of the children.
- [tree_node](#)< T > * [delete_tree_node](#) ()
Delete the current [tree_node](#).
- T & [elt](#) ()
The getter of the element.
- const T & [elt](#) () const
The const getter of the element.
- [tree_node](#)< T > * [parent](#) ()
The getter of the parent.

- void [print](#) (std::ostream &ostr, int level=0)
Print on `ostr` the arborescence with the current `tree_node` as root.
- [tree_node](#)< T > * [search](#) (T &elt)
Search the `tree_node` with value `elt` in the arborescence of the current `tree_node`.
- int [search_rec](#) ([tree_node](#)< T > **res, T &elt)
The using method for method search.
- void [set_parent](#) ([tree_node](#)< T > *parent)
Bind `tree_node` to the current `tree_node` and become its parent.
- [tree_node](#) ()
Constructor.
- [tree_node](#) (T elt)
Constructor.

10.400.1 Detailed Description

template<typename T>class mln::util::tree_node< T >

Class of generic [tree_node](#) for tree.

Definition at line 58 of file tree.hh.

10.400.2 Constructor & Destructor Documentation

10.400.2.1 template<typename T > [tree_node](#)< T >::tree_node () [inline]

Constructor.

Definition at line 345 of file tree.hh.

10.400.2.2 template<typename T > [tree_node](#)< T >::tree_node (T elt) [inline]

Constructor.

Parameters

in	elt	The element of tree_node .
--------------------	---------------------	--

Definition at line 352 of file tree.hh.

10.400.3 Member Function Documentation

10.400.3.1 template<typename T > [tree_node](#)< T > * [tree_node](#)< T >::add_child (T elt) [inline]

Create a [tree_node](#) with `elt` which become the child of the current [tree_node](#).

Parameters

in	elt	The element of the new child to add.
--------------------	---------------------	--------------------------------------

Returns

The new [tree_node](#) created.

Definition at line 394 of file tree.hh.

10.400.3.2 `template<typename T> tree_node< T> * tree_node< T>::add_child (tree_node< T> * tree_node)`
`[inline]`

Bind `tree_node` to the current `tree_node` and become its child.

Parameters

<code>in</code>	<code>tree_node</code>	The new child <code>tree_node</code> .
-----------------	------------------------	--

Returns

The child `tree_node`.

Definition at line 407 of file `tree.hh`.

References `mln::util::tree_node< T>::parent()`.

10.400.3.3 `template<typename T> bool tree_node< T>::check_consistency ()` `[inline]`

Check the consistency of the `tree_node`.

Returns

true if no error, else false.

Definition at line 519 of file `tree.hh`.

10.400.3.4 `template<typename T> std::vector< tree_node< T>*> & tree_node< T>::children ()` `[inline]`

The getter of the children.

Returns

The children of the `tree_node`.

Definition at line 378 of file `tree.hh`.

Referenced by `mln::util::tree< T>::add_tree_up()`.

10.400.3.5 `template<typename T> const std::vector< tree_node< T>*> & tree_node< T>::children () const`
`[inline]`

The getter of the children.

Returns

The children of the `tree_node` in const.

Definition at line 386 of file `tree.hh`.

10.400.3.6 `template<typename T> tree_node< T> * tree_node< T>::delete_tree_node ()` `[inline]`

Delete the current `tree_node`.

Definition at line 427 of file `tree.hh`.

10.400.3.7 `template<typename T> T & tree_node< T >::elt () [inline]`

The getter of the element.

Returns

The element of the [tree_node](#).

Definition at line 369 of file tree.hh.

10.400.3.8 `template<typename T> const T & tree_node< T >::elt () const [inline]`

The const getter of the element.

Returns

The element of the [tree_node](#) in const.

Definition at line 361 of file tree.hh.

10.400.3.9 `template<typename T> tree_node< T > * tree_node< T >::parent () [inline]`

The getter of the parent.

Returns

The parent of the [tree_node](#).

Definition at line 477 of file tree.hh.

Referenced by `mln::util::tree_node< T >::add_child()`, `mln::util::branch_iter< T >::deepness()`, and `mln::util::branch_iter_ind< T >::deepness()`.

10.400.3.10 `template<typename T> void tree_node< T >::print (std::ostream & ostr, int level = 0) [inline]`

Print on `ostr` the arborescence with the current [tree_node](#) as root.

Parameters

<code>in</code>	<code>ostr</code>	The output stream.
<code>in</code>	<code>level</code>	The deep level

Definition at line 449 of file tree.hh.

10.400.3.11 `template<typename T> tree_node< T > * tree_node< T >::search (T & elt) [inline]`

Search the [tree_node](#) with value `elt` in the arborescence of the current [tree_node](#).

Parameters

<code>in</code>	<code>elt</code>	The value of the searched tree_node .
-----------------	------------------	---

Returns

If not found 0 else the [tree_node](#) with `elt` value.

Definition at line 507 of file tree.hh.

10.400.3.12 `template<typename T> int tree_node< T >::search_rec (tree_node< T > ** res, T & elt) [inline]`

The using method for method search.

Definition at line 485 of file tree.hh.

10.400.3.13 `template<typename T> void tree_node< T >::set_parent (tree_node< T > * parent) [inline]`

Bind `tree_node` to the current `tree_node` and become its parent.

Parameters

<code>in</code>	<code>parent</code>	The new parent <code>tree_node</code> .
-----------------	---------------------	---

Definition at line 467 of file tree.hh.

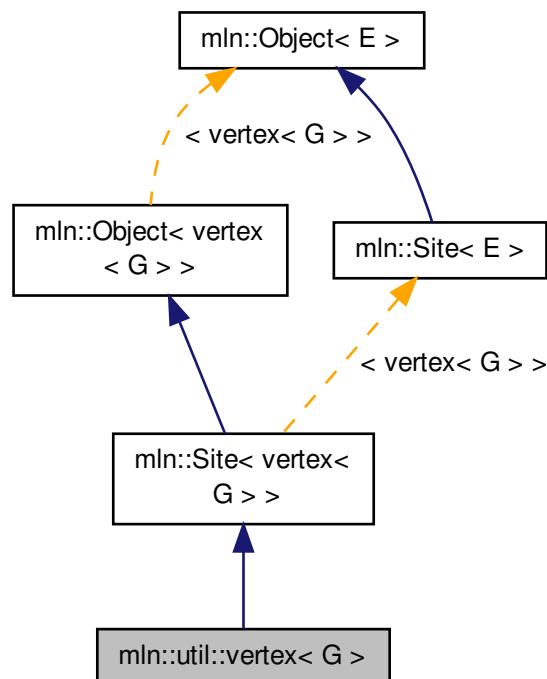
Referenced by `mln::util::tree< T >::add_tree_up()`.

10.401 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.

10.401.1 Detailed Description

```
template<typename G>class mln::util::vertex< G >
```

[Vertex](#) of a graph G.

Definition at line 71 of file vertex.hh.

10.401.2 Member Typedef Documentation

10.401.2.1 `template<typename G> typedef Vertex<void> mln::util::vertex< G >::Category`

[Object](#) category.

Definition at line 77 of file vertex.hh.

10.401.2.2 `template<typename G> typedef G mln::util::vertex< G >::graph_t`

[Graph](#) associated type.

Definition at line 86 of file vertex.hh.

10.401.2.3 `template<typename G> typedef vertex_id_t mln::util::vertex< G >::id_t`

The vertex type id.

Definition at line 83 of file vertex.hh.

10.401.2.4 `template<typename G> typedef vertex_id_t::value_t mln::util::vertex< G >::id_value_t`

The underlying type used to store vertex ids.

Definition at line 80 of file vertex.hh.

10.401.3 Constructor & Destructor Documentation

10.401.3.1 `template<typename G> vertex< G >::vertex () [inline]`

Constructors.

Definition at line 226 of file vertex.hh.

10.401.4 Member Function Documentation

10.401.4.1 `template<typename G> void vertex< G >::change_graph (const G & g) [inline]`

Change the parent graph of that vertex.

Definition at line 331 of file vertex.hh.

10.401.4.2 `template<typename G> edge< G > vertex< G >::edge_with (const vertex< G > & v_id) const [inline]`

Returns true if this vertex has an edge with the given vertex.

Definition at line 321 of file vertex.hh.

10.401.4.3 `template<typename G> const G & vertex< G >::graph () const [inline]`

Returns the graph pointer this vertex belongs to.

Definition at line 348 of file vertex.hh.

Referenced by `mln::p_vertices< G, F >::has()`, `mln::util::line_graph< G >::has()`, and `mln::util::operator==()`.

10.401.4.4 `template<typename G> const vertex_id_t & vertex<G>::id () const [inline]`

Returns the vertex id.

Definition at line 356 of file vertex.hh.

Referenced by `mln::util::line_graph<G>::has()`, and `mln::util::operator==()`.

10.401.4.5 `template<typename G> void vertex<G>::invalidate () [inline]`

Invalidate that vertex.

Definition at line 266 of file vertex.hh.

10.401.4.6 `template<typename G> bool vertex<G>::is_valid () const [inline]`

Check whether the vertex is still part of the graph.

Definition at line 258 of file vertex.hh.

Referenced by `mln::p_vertices<G, F>::has()`.

10.401.4.7 `template<typename G> edge_id_t vertex<G>::ith_nbh_edge (unsigned i) const [inline]`

Returns the *i*th edge starting from this vertex.

Definition at line 285 of file vertex.hh.

10.401.4.8 `template<typename G> vertex_id_t vertex<G>::ith_nbh_vertex (unsigned i) const [inline]`

Returns the *i*th vertex adjacent to this vertex.

Definition at line 303 of file vertex.hh.

10.401.4.9 `template<typename G> unsigned 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 retrived from the initial graph.

Definition at line 294 of file vertex.hh.

10.401.4.10 `template<typename G> unsigned vertex<G>::nmax_nbh_vertices () const [inline]`

Returns the number max of vertices adjacent to this vertex.

Definition at line 312 of file vertex.hh.

10.401.4.11 `template<typename G> vertex<G>::operator vertex_id_t () const [inline]`

Conversion to the vertex id.

FIXME: May cause ambiguities... :(

Definition at line 363 of file vertex.hh.

10.401.4.12 `template<typename G > vertex_id_t vertex< G >::other(const edge_id_t & id_e) const` `[inline]`

Returns the other vertex located on edge `id_e`.

Definition at line 274 of file `vertex.hh`.

10.401.4.13 `template<typename G > void vertex< G >::update_id(const vertex_id_t & id)` `[inline]`

Update the vertex id.

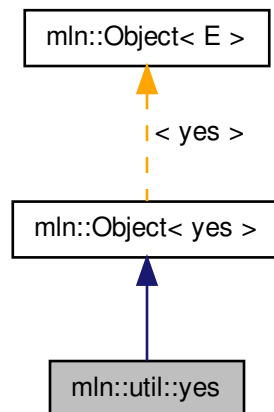
Definition at line 340 of file `vertex.hh`.

10.402 mln::util::yes Struct Reference

[Object](#) that always says "yes".

```
#include <yes.hh>
```

Inheritance diagram for `mln::util::yes`:



10.402.1 Detailed Description

[Object](#) that always says "yes".

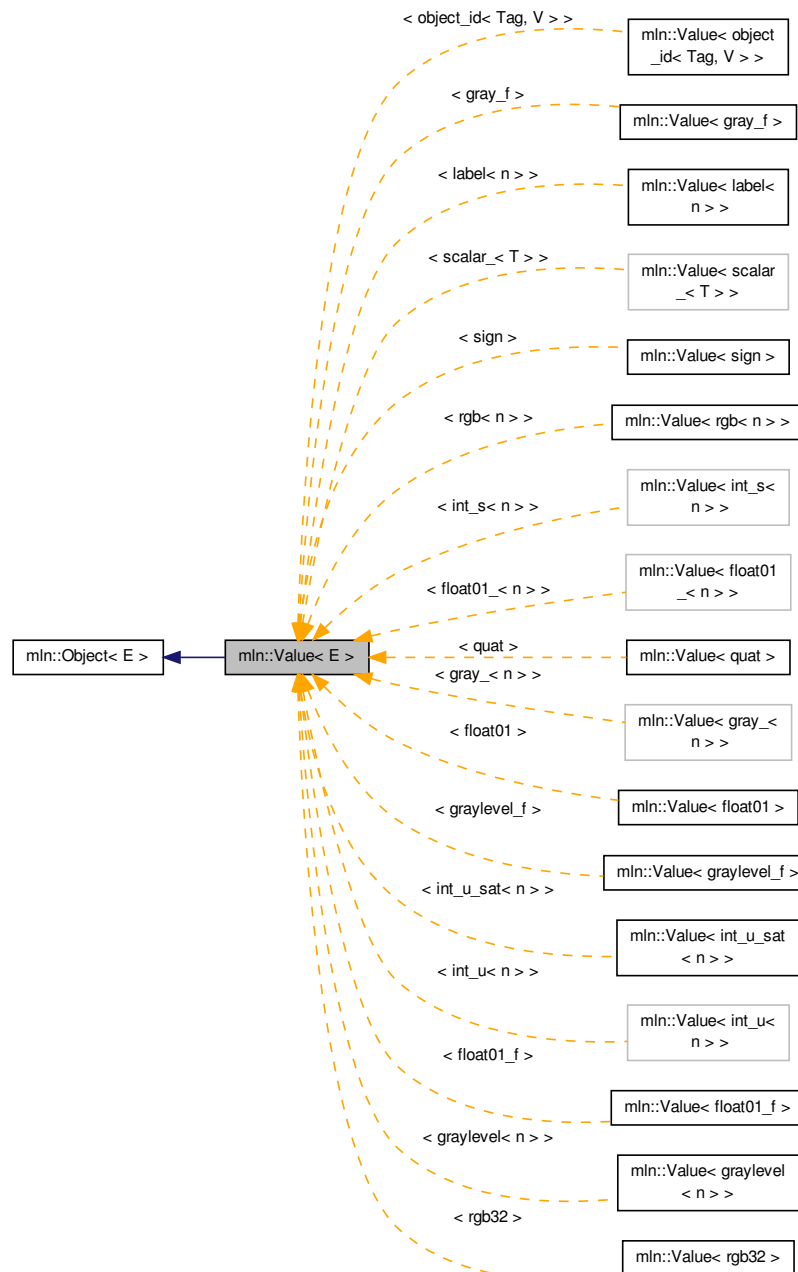
Definition at line 76 of file `yes.hh`.

10.403 mln::Value< E > Struct Template Reference

Base class for implementation classes of values.

```
#include <value.hh>
```

Inheritance diagram for `mln::Value< E >`:



10.403.1 Detailed Description

```
template<typename E>struct mln::Value< E >
```

Base class for implementation classes of values.

See Also

mln::doc::Value for a complete documentation of this class contents.

Definition at line 57 of file core/concept/value.hh.

10.404 mln::value::float01 Class Reference

Class for floating values restricted to the interval [0..1] and discretized with n bits.

```
#include <float01.hh>
```

Inherits mln::value::Floating< E >.

Public Types

- typedef std::pair< unsigned, unsigned long > [enc](#)

Encoding associated type.

- typedef float [equiv](#)

Equivalent associated type.

Public Member Functions

- [float01](#) ()
Ctor.
- template<unsigned n>
[float01](#) (const float01_< n > &val)
Ctor.
- [float01](#) (unsigned [nbits](#), float val)
Ctor.
- unsigned [nbits](#) () const
Access to the encoding size.
- operator float () const
Conversion to float.
- [float01](#) & [set_nbits](#) (unsigned [nbits](#))
Set the encoding size to nbits.
- const [float01](#) [to_nbits](#) (unsigned [nbits](#)) const
Return an equivalent gray encoded on nbits bits.
- float [value](#) () const
Access to std type.
- unsigned long [value_ind](#) () const
Access to the position in the quantized interval.

10.404.1 Detailed Description

Class for floating values restricted to the interval [0..1] and discretized with n bits.

Definition at line 57 of file float01.hh.

10.404.2 Member Typedef Documentation

10.404.2.1 `typedef std::pair<unsigned, unsigned long> mln::value::float01::enc`

Encoding associated type.

Definition at line 62 of file float01.hh.

10.404.2.2 `typedef float mln::value::float01::equiv`

Equivalent associated type.

Definition at line 65 of file float01.hh.

10.404.3 Constructor & Destructor Documentation

10.404.3.1 `mln::value::float01::float01 () [inline]`

Ctor.

Definition at line 152 of file float01.hh.

10.404.3.2 `template<unsigned n> mln::value::float01::float01 (const float01_<n> & val) [inline]`

Ctor.

Definition at line 159 of file float01.hh.

10.404.3.3 `mln::value::float01::float01 (unsigned nbits, float val) [inline]`

Ctor.

Definition at line 166 of file float01.hh.

10.404.4 Member Function Documentation

10.404.4.1 `unsigned mln::value::float01::nbits () const [inline]`

Access to the encoding size.

Definition at line 187 of file float01.hh.

Referenced by `set_nbits()`.

10.404.4.2 `mln::value::float01::operator float () const [inline]`

Conversion to float.

Definition at line 225 of file float01.hh.

10.404.4.3 `float01 & mln::value::float01::set_nbits (unsigned nbits) [inline]`

Set the encoding size to `nbits`.

Definition at line 194 of file float01.hh.

References `nbits()`.

Referenced by to_nbits().

10.404.4.4 `const float01 mln::value::float01::to_nbits (unsigned nbits) const` `[inline]`

Return an equivalent gray encoded on *nbits* bits.

Definition at line 215 of file float01.hh.

References set_nbits().

10.404.4.5 `float mln::value::float01::value () const` `[inline]`

Access to std type.

Definition at line 173 of file float01.hh.

10.404.4.6 `unsigned long mln::value::float01::value_ind () const` `[inline]`

Access to the position in the quantized interval.

Definition at line 180 of file float01.hh.

10.405 mln::value::float01_f Struct Reference

Class for floating values restricted to the interval [0..1].

`#include <float01_f.hh>`

Inherits mln::value::Floating< E >, and mln::value::internal::value_like_< V, C, N, E >.

Public Member Functions

- [float01_f](#) ()
Constructor without argument.
- [float01_f](#) (float val)
Constructor from a float.
- [operator float](#) () const
Conversion to a float.
- [float01_f](#) & [operator=](#) (const float val)
Assignment from a float.
- float [value](#) () const
Access to float value.

10.405.1 Detailed Description

Class for floating values restricted to the interval [0..1].

Definition at line 85 of file float01_f.hh.

10.405.2 Constructor & Destructor Documentation

10.405.2.1 `mln::value::float01_f::float01_f ()` `[inline]`

Constructor without argument.

Definition at line 116 of file float01_f.hh.

10.405.2.2 `mln::value::float01_f::float01_f(float val)` `[inline]`

Constructor from a float.

Definition at line 121 of file float01_f.hh.

10.405.3 Member Function Documentation

10.405.3.1 `mln::value::float01_f::operator float () const` `[inline]`

Conversion to a float.

Definition at line 146 of file float01_f.hh.

10.405.3.2 `float01_f & mln::value::float01_f::operator= (const float val)` `[inline]`

Assignment from a float.

Definition at line 137 of file float01_f.hh.

10.405.3.3 `float mln::value::float01_f::value () const` `[inline]`

Access to float value.

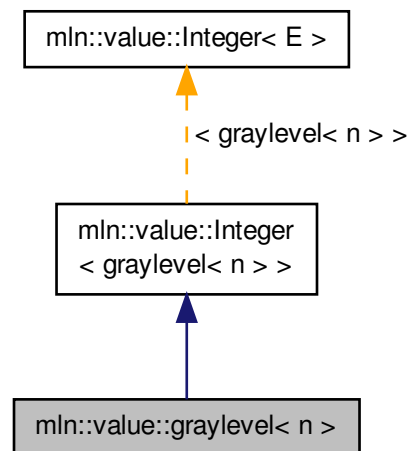
Definition at line 130 of file float01_f.hh.

10.406 `mln::value::graylevel< n >` Struct Template Reference

General gray-level class on n bits.

```
#include <graylevel.hh>
```

Inheritance diagram for mln::value::graylevel< n >:



Public Member Functions

- `graylevel ()`
Constructor without argument.
- `graylevel (const graylevel< n > &rhs)`
Copy constructor.
- `graylevel (int val)`
Constructor from int.
- `template<unsigned m>`
`graylevel (const graylevel< m > &rhs)`
Constructor from any graylevel.
- `graylevel< n > & operator= (const graylevel< n > &rhs)`
Assignment.
- `graylevel< n > & operator= (int val)`
Assignment with int.
- `template<unsigned m>`
`graylevel< n > & operator= (const graylevel< m > &rhs)`
Assignment with any graylevel.
- `float to_float () const`
Conversion to float between 0 and 1.
- `unsigned value () const`
Access to std type.
- `graylevel (const mln::literal::black_t &)`
Ctors with literals.
- `graylevel< n > & operator= (const mln::literal::black_t &)`
Assignment with literals.

10.406.1 Detailed Description

`template<unsigned n> struct mln::value::graylevel< n >`

General gray-level class on n bits.

Forward declarations.

Definition at line 257 of file graylevel.hh.

10.406.2 Constructor & Destructor Documentation

10.406.2.1 `template<unsigned n> graylevel< n >::graylevel () [inline]`

Constructor without argument.

Definition at line 436 of file graylevel.hh.

10.406.2.2 `template<unsigned n> graylevel< n >::graylevel (const graylevel< n > & rhs) [inline]`

Copy constructor.

Definition at line 463 of file graylevel.hh.

10.406.2.3 `template<unsigned n> graylevel< n >::graylevel (int val) [inline]`

Constructor from int.

Definition at line 443 of file graylevel.hh.

10.406.2.4 `template<unsigned n> template<unsigned m> graylevel< n >::graylevel (const graylevel< m > & rhs) [inline]`

Constructor from any graylevel.

Definition at line 481 of file graylevel.hh.

References `mln::value::graylevel< n >::value()`.

10.406.2.5 `template<unsigned n> graylevel< n >::graylevel (const mln::literal::black_t &) [inline]`

Ctors with literals.

Definition at line 499 of file graylevel.hh.

10.406.3 Member Function Documentation

10.406.3.1 `template<unsigned n> graylevel< n > & graylevel< n >::operator= (const graylevel< n > & rhs) [inline]`

Assignment.

Definition at line 472 of file graylevel.hh.

10.406.3.2 `template<unsigned n> graylevel< n > & graylevel< n >::operator= (int val) [inline]`

Assignment with int.

Definition at line 453 of file graylevel.hh.

10.406.3.3 `template<unsigned n> template<unsigned m> graylevel< n > & graylevel< n >::operator= (const graylevel< m > & rhs) [inline]`

Assignment with any graylevel.

Definition at line 490 of file graylevel.hh.

References `mln::value::graylevel< n >::value()`.

10.406.3.4 `template<unsigned n> graylevel< n > & graylevel< n >::operator= (const mln::literal::black_t &) [inline]`

Assignment with literals.

Definition at line 507 of file graylevel.hh.

10.406.3.5 `template<unsigned n> float graylevel< n >::to_float () const [inline]`

Conversion to float between 0 and 1.

Definition at line 557 of file graylevel.hh.

Referenced by `mln::value::graylevel_f::graylevel_f()`, and `mln::value::graylevel_f::operator=()`.

10.406.3.6 `template<unsigned n> unsigned graylevel< n >::value () const [inline]`

Access to std type.

Definition at line 549 of file graylevel.hh.

Referenced by `mln::value::graylevel< n >::graylevel()`, and `mln::value::graylevel< n >::operator=()`.

10.407 mln::value::graylevel_f Struct Reference

General gray-level class on n bits.

`#include <graylevel_f.hh>`

Inherits `mln::value::Floating< E >`, and `mln::value::internal::value_like_< V, C, N, E >`.

Public Member Functions

- [graylevel_f](#) ()
Constructor without argument.
- [graylevel_f](#) (const [graylevel_f](#) &rhs)
Copy constructor.
- [graylevel_f](#) (float val)
Constructor from float.
- `template<unsigned n>`
[graylevel_f](#) (const [graylevel](#)< n > &rhs)
Constructor from graylevel.
- `template<unsigned n>`
[operator graylevel](#)< n > () const
Conversion to graylevel<n>.

- `graylevel_f & operator= (const graylevel_f &rhs)`
Assignment.
- `graylevel_f & operator= (float val)`
Assignment with float.
- `template<unsigned n> graylevel_f & operator= (const graylevel< n > &rhs)`
Assignment with graylevel.
- `float value () const`
Access to std type.
- `graylevel_f (const mln::literal::black_t &)`
Ctors with literals.
- `graylevel_f & operator= (const mln::literal::black_t &)`
Assignment with literals.

10.407.1 Detailed Description

General gray-level class on n bits.

Definition at line 194 of file `graylevel_f.hh`.

10.407.2 Constructor & Destructor Documentation

10.407.2.1 `mln::value::graylevel_f::graylevel_f ()` `[inline]`

Constructor without argument.

Definition at line 342 of file `graylevel_f.hh`.

10.407.2.2 `mln::value::graylevel_f::graylevel_f (const graylevel_f & rhs)` `[inline]`

Copy constructor.

Definition at line 384 of file `graylevel_f.hh`.

10.407.2.3 `mln::value::graylevel_f::graylevel_f (float val)` `[inline]`

Constructor from float.

Definition at line 348 of file `graylevel_f.hh`.

10.407.2.4 `template<unsigned n> mln::value::graylevel_f::graylevel_f (const graylevel< n > & rhs)`

Constructor from graylevel.

Definition at line 366 of file `graylevel_f.hh`.

References `mln::value::graylevel< n >::to_float()`.

10.407.2.5 `mln::value::graylevel_f::graylevel_f (const mln::literal::black_t &)` `[inline]`

Ctors with literals.

Definition at line 401 of file `graylevel_f.hh`.

10.407.3 Member Function Documentation

10.407.3.1 `template<unsigned n> mln::value::graylevel_f::operator graylevel< n > () const` `[inline]`

Conversion to graylevel<n>.

Definition at line 444 of file graylevel_f.hh.

10.407.3.2 `graylevel_f & mln::value::graylevel_f::operator= (const graylevel_f & rhs)` `[inline]`

Assignment.

Definition at line 392 of file graylevel_f.hh.

10.407.3.3 `graylevel_f & mln::value::graylevel_f::operator= (float val)` `[inline]`

Assignment with float.

Definition at line 357 of file graylevel_f.hh.

10.407.3.4 `template<unsigned n> graylevel_f & mln::value::graylevel_f::operator= (const graylevel< n > & rhs)`

Assignment with graylevel.

Definition at line 375 of file graylevel_f.hh.

References mln::value::graylevel< n >::to_float().

10.407.3.5 `graylevel_f & mln::value::graylevel_f::operator= (const mln::literal::black_t &)` `[inline]`

Assignment with literals.

Definition at line 408 of file graylevel_f.hh.

10.407.3.6 `float mln::value::graylevel_f::value () const` `[inline]`

Access to std type.

Definition at line 451 of file graylevel_f.hh.

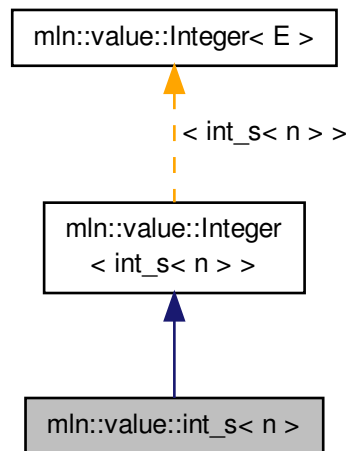
Referenced by mln::value::operator<<().

10.408 mln::value::int_s< n > Struct Template Reference

Signed integer value class.

```
#include <int_s.hh>
```

Inheritance diagram for `mln::value::int_s< n >`:



Public Member Functions

- `int_s ()`
Constructor without argument.
- `int_s (int i)`
Constructor from an integer.
- `operator int () const`
Conversion to an integer.
- `int_s< n > & operator= (int i)`
Assignment from an integer.
- `int_s (const mln::literal::zero_t &)`
Constructors/assignments with literals.

Static Public Attributes

- static const `int_s< n > one = 1`
Unit value.
- static const `int_s< n > zero = 0`
Zero value.

10.408.1 Detailed Description

```
template<unsigned n>struct mln::value::int_s< n >
```

Signed integer value class.

The parameter is `n` the number of encoding bits.

Definition at line 115 of file `int_s.hh`.

10.408.2 Constructor & Destructor Documentation

10.408.2.1 `template<unsigned n> int_s< n >::int_s() [inline]`

Constructor without argument.

Definition at line 179 of file int_s.hh.

10.408.2.2 `template<unsigned n> int_s< n >::int_s(int i) [inline]`

Constructor from an integer.

Definition at line 192 of file int_s.hh.

10.408.2.3 `template<unsigned n> int_s< n >::int_s(const mln::literal::zero_t &) [inline]`

Constructors/assignments with literals.

Definition at line 222 of file int_s.hh.

10.408.3 Member Function Documentation

10.408.3.1 `template<unsigned n> int_s< n >::operator int() const [inline]`

Conversion to an integer.

Definition at line 185 of file int_s.hh.

10.408.3.2 `template<unsigned n> int_s< n > & int_s< n >::operator=(int i) [inline]`

Assignment from an integer.

Definition at line 207 of file int_s.hh.

10.408.4 Member Data Documentation

10.408.4.1 `template<unsigned n> const int_s< n > int_s< n >::one = 1 [static]`

Unit value.

Definition at line 149 of file int_s.hh.

10.408.4.2 `template<unsigned n> const int_s< n > int_s< n >::zero = 0 [static]`

Zero value.

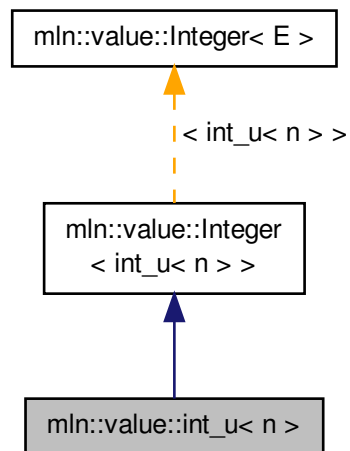
Definition at line 146 of file int_s.hh.

10.409 mln::value::int_u< n > Struct Template Reference

Unsigned integer value class.

```
#include <int_u.hh>
```

Inheritance diagram for `mln::value::int_u< n >`:



Public Member Functions

- `int_u ()`
Constructor without argument.
- `int_u (int i)`
Constructor from an integer.
- `int_u< n > next () const`
Give the next value (i.e., $i + 1$).
- `operator unsigned () const`
Conversion to an unsigned integer.
- `int operator- () const`
Unary operator minus.
- `int_u< n > & operator= (int i)`
Assignment from an integer.
- `int_u (const mln::literal::zero_t &)`
Constructors/assignments with literals.

10.409.1 Detailed Description

```
template<unsigned n>struct mln::value::int_u< n >
```

Unsigned integer value class.

The parameter is `n` the number of encoding bits.

Definition at line 156 of file `int_u.hh`.

10.409.2 Constructor & Destructor Documentation

10.409.2.1 `template<unsigned n> int_u< n >::int_u () [inline]`

Constructor without argument.

Definition at line 276 of file int_u.hh.

10.409.2.2 `template<unsigned n> int_u< n >::int_u (int i) [inline]`

Constructor from an integer.

Definition at line 282 of file int_u.hh.

10.409.2.3 `template<unsigned n> int_u< n >::int_u (const mln::literal::zero_t &) [inline]`

Constructors/assignments with literals.

Definition at line 291 of file int_u.hh.

10.409.3 Member Function Documentation

10.409.3.1 `template<unsigned n> int_u< n > int_u< n >::next () const [inline]`

Give the next value (i.e., i + 1).

Definition at line 350 of file int_u.hh.

10.409.3.2 `template<unsigned n> int_u< n >::operator unsigned () const [inline]`

Conversion to an unsigned integer.

Definition at line 323 of file int_u.hh.

10.409.3.3 `template<unsigned n> int int_u< n >::operator- () const [inline]`

Unary operator minus.

Definition at line 331 of file int_u.hh.

10.409.3.4 `template<unsigned n> int_u< n > & int_u< n >::operator= (int i) [inline]`

Assignment from an integer.

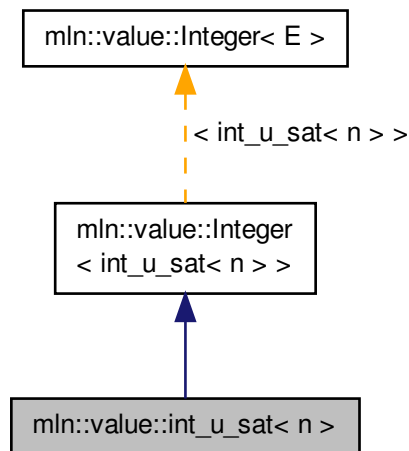
Definition at line 339 of file int_u.hh.

10.410 mln::value::int_u_sat< n > Struct Template Reference

Unsigned integer value class with saturation behavior.

```
#include <int_u_sat.hh>
```

Inheritance diagram for `mln::value::int_u_sat< n >`:



Public Member Functions

- `int_u_sat ()`
Constructor without argument.
- `int_u_sat (int i)`
Constructor from an integer.
- `operator int () const`
Conversion to an integer.
- `int_u_sat< n > & operator+= (int i)`
Self addition.
- `int_u_sat< n > & operator-= (int i)`
Self subtraction.
- `int_u_sat< n > & operator= (int i)`
Assignment from an integer.

Static Public Attributes

- static const `int_u_sat< n > one = 1`
Unit value.
- static const `int_u_sat< n > zero = 0`
Zero value.

10.410.1 Detailed Description

```
template<unsigned n>struct mln::value::int_u_sat< n >
```

Unsigned integer value class with saturation behavior.

The parameter is `n` the number of encoding bits.

Definition at line 90 of file `int_u_sat.hh`.

10.410.2 Constructor & Destructor Documentation

10.410.2.1 `template<unsigned n> int_u_sat< n >::int_u_sat () [inline]`

Constructor without argument.

Definition at line 149 of file int_u_sat.hh.

10.410.2.2 `template<unsigned n> int_u_sat< n >::int_u_sat (int i) [inline]`

Constructor from an integer.

Definition at line 155 of file int_u_sat.hh.

10.410.3 Member Function Documentation

10.410.3.1 `template<unsigned n> int_u_sat< n >::operator int () const [inline]`

Conversion to an integer.

Definition at line 170 of file int_u_sat.hh.

10.410.3.2 `template<unsigned n> int_u_sat< n > & int_u_sat< n >::operator+= (int i) [inline]`

Self addition.

Definition at line 195 of file int_u_sat.hh.

10.410.3.3 `template<unsigned n> int_u_sat< n > & int_u_sat< n >::operator-= (int i) [inline]`

Self subtraction.

Definition at line 205 of file int_u_sat.hh.

10.410.3.4 `template<unsigned n> int_u_sat< n > & int_u_sat< n >::operator= (int i) [inline]`

Assignment from an integer.

Definition at line 178 of file int_u_sat.hh.

10.410.4 Member Data Documentation

10.410.4.1 `template<unsigned n> const int_u_sat< n > int_u_sat< n >::one = 1 [static]`

Unit value.

Definition at line 115 of file int_u_sat.hh.

10.410.4.2 `template<unsigned n> const int_u_sat< n > int_u_sat< n >::zero = 0 [static]`

Zero value.

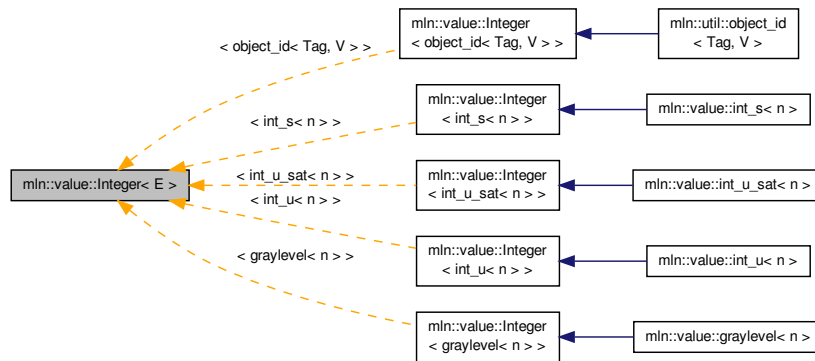
Definition at line 112 of file int_u_sat.hh.

10.411 mln::value::Integer< E > Struct Template Reference

Concept of integer.

```
#include <integer.hh>
```

Inheritance diagram for mln::value::Integer< E >:



10.411.1 Detailed Description

```
template<typename E>struct mln::value::Integer< E >
```

Concept of integer.

Definition at line 58 of file `concept/integer.hh`.

10.412 mln::value::Integer< void > Struct Template Reference

Category flag type.

```
#include <integer.hh>
```

10.412.1 Detailed Description

```
template<>struct mln::value::Integer< void >
```

Category flag type.

Definition at line 50 of file `concept/integer.hh`.

10.413 mln::value::label< n > Struct Template Reference

Label value class.

```
#include <label.hh>
```

Inherits `mln::value::Symbolic< E >`, and `mln::value::internal::value_like_< V, C, N, E >`.

Public Types

- typedef
internal::encoding_unsigned_
< n >::ret **enc**
Encoding associated type.

Public Member Functions

- **label** ()
Constructor without argument.
- **label** (unsigned i)
Constructor from an (unsigned) integer.
- **label** (const literal::zero_t &v)
*Constructor from *literal::zero*.*
- **label**< n > **next** () const
Return the next value.
- **operator unsigned** () const
Conversion to an unsigned integer.
- **label**< n > & **operator++** ()
Self increment.
- **label**< n > & **operator--** ()
Self decrement.
- **label**< n > & **operator=** (unsigned i)
Assignment from an (unsigned) integer.
- **label**< n > & **operator=** (const literal::zero_t &v)
*Assignment from *literal::zero*.*
- **label**< n > **prev** () const
Return the previous value.

10.413.1 Detailed Description

template<unsigned n>struct mIn::value::label< n >

Label value class.

The parameter *n* is the number of encoding bits.

Definition at line 140 of file label.hh.

10.413.2 Member Typedef Documentation

10.413.2.1 template<unsigned n> typedef internal::encoding_unsigned_<n>::ret mIn::value::label< n >::enc

Encoding associated type.

Definition at line 150 of file label.hh.

10.413.3 Constructor & Destructor Documentation

10.413.3.1 template<unsigned n> label< n >::label () [inline]

Constructor without argument.

Definition at line 271 of file label.hh.

10.413.3.2 `template<unsigned n> label< n >::label (unsigned i) [inline]`

Constructor from an (unsigned) integer.

Definition at line 277 of file label.hh.

10.413.3.3 `template<unsigned n> label< n >::label (const literal::zero_t & v) [inline]`

Constructor from [literal::zero](#).

Definition at line 284 of file label.hh.

10.413.4 Member Function Documentation

10.413.4.1 `template<unsigned n> label< n > label< n >::next () const [inline]`

Return the next value.

Definition at line 338 of file label.hh.

10.413.4.2 `template<unsigned n> label< n >::operator unsigned () const [inline]`

Conversion to an unsigned integer.

Definition at line 291 of file label.hh.

10.413.4.3 `template<unsigned n> label< n > & label< n >::operator++ () [inline]`

Self increment.

Definition at line 318 of file label.hh.

10.413.4.4 `template<unsigned n> label< n > & label< n >::operator-- () [inline]`

Self decrement.

Definition at line 328 of file label.hh.

10.413.4.5 `template<unsigned n> label< n > & label< n >::operator= (unsigned i) [inline]`

Assignment from an (unsigned) integer.

Definition at line 299 of file label.hh.

10.413.4.6 `template<unsigned n> label< n > & label< n >::operator= (const literal::zero_t & v) [inline]`

Assignment from [literal::zero](#).

Definition at line 309 of file label.hh.

10.413.4.7 `template<unsigned n> label< n > label< n >::prev () const [inline]`

Return the previous value.

Definition at line 346 of file label.hh.

10.414 mln::value::lut_vec< S, T > Struct Template Reference

Class that defines FIXME.

```
#include <lut_vec.hh>
```

Inherits mln::Value_Set< lut_vec< S, T > >.

Public Types

- typedef bkd_viter_< lut_vec< S, T > > bkd_viter
Backward Value_Iterator associated type.
- typedef fwd_viter_< lut_vec< S, T > > fwd_viter
Forward Value_Iterator associated type.
- typedef T value
Value associated type.

Public Member Functions

- bool has (const value &v) const
Test if v belongs to this set.
- unsigned index_of (const value &v) const
Give the index of value v in this set.
- unsigned nvalues () const
Give the number of values.
- T operator[] (unsigned i) const
Give the i -th value.
- template<typename F >
lut_vec (const S &vset, const Function_v2v< F > &f)
Constructors
Constructor from a value set and any Function_v2v.
- template<typename V >
lut_vec (const S &vset, const Function_v2v< fun::i2v::array< V > > &f)
Constructor from a value set and any fun::i2v::array.
- template<typename V >
lut_vec (const S &vset, const Function_v2v< util::array< V > > &f)
Constructor from a value set and any util::array.

10.414.1 Detailed Description

```
template<typename S, typename T>struct mln::value::lut_vec< S, T >
```

Class that defines FIXME.

Warning

This is a multi-set!!! FIXME

Definition at line 71 of file lut_vec.hh.

10.414.2 Member Typedef Documentation

10.414.2.1 `template<typename S, typename T> typedef bkd_viter_< lut_vec<S,T> > mln::value::lut_vec< S, T >::bkd_viter`

Backward Value_iterator associated type.

Definition at line 80 of file lut_vec.hh.

10.414.2.2 `template<typename S, typename T> typedef fwd_viter_< lut_vec<S,T> > mln::value::lut_vec< S, T >::fwd_viter`

Forward Value_iterator associated type.

Definition at line 77 of file lut_vec.hh.

10.414.2.3 `template<typename S, typename T> typedef T mln::value::lut_vec< S, T >::value`

[Value](#) associated type.

Definition at line 74 of file lut_vec.hh.

10.414.3 Constructor & Destructor Documentation

10.414.3.1 `template<typename S, typename T> template<typename F> mln::value::lut_vec< S, T >::lut_vec (const S & vset, const Function_v2v< F> & f) [inline]`

Constructors

Constructor from a value set and any [Function_v2v](#).

Definition at line 149 of file lut_vec.hh.

10.414.3.2 `template<typename S, typename T> template<typename V> mln::value::lut_vec< S, T >::lut_vec (const S & vset, const Function_v2v< fun::i2v::array< V> > & f) [inline]`

Constructor from a value set and any `fun::i2v::array`.

Definition at line 162 of file lut_vec.hh.

10.414.3.3 `template<typename S, typename T> template<typename V> mln::value::lut_vec< S, T >::lut_vec (const S & vset, const Function_v2v< util::array< V> > & f) [inline]`

Constructor from a value set and any `util::array`.

Definition at line 173 of file lut_vec.hh.

References `mln::util::array< T>::size()`, and `mln::util::array< T>::std_vector()`.

10.414.4 Member Function Documentation

10.414.4.1 `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.

Definition at line 130 of file lut_vec.hh.

10.414.4.2 `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.

Definition at line 139 of file lut_vec.hh.

10.414.4.3 `template<typename S, typename T> unsigned mln::value::lut_vec< S, T >::nvalues () const [inline]`

Give the number of values.

Definition at line 203 of file lut_vec.hh.

10.414.4.4 `template<typename S, typename T> T mln::value::lut_vec< S, T >::operator[] (unsigned i) const [inline]`

Give the i -th value.

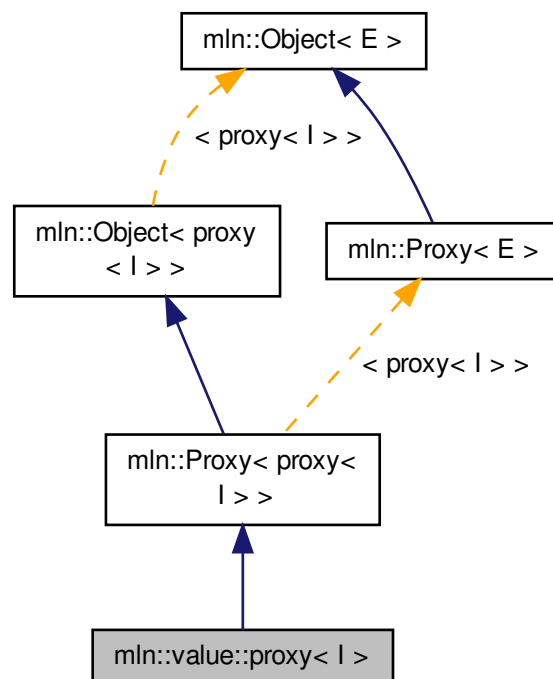
Definition at line 194 of file lut_vec.hh.

10.415 mln::value::proxy< I > Class Template Reference

Generic proxy class for an image pixel value.

```
#include <proxy.hh>
```

Inheritance diagram for mln::value::proxy< I >:



Public Types

- typedef void [enc](#)
Encoding associated type.
- typedef I::value [equiv](#)
Equivalent associated type.

Public Member Functions

- [proxy](#)< I > & [operator=](#) (const [proxy](#)< I > &rhs)
Assignment (write access); replacement for default op.
- template<typename J >
[proxy](#)< I > & [operator=](#) (const [proxy](#)< J > &rhs)
Assignment (write access); with other proxy.
- [proxy](#) ()
Constructor.
- [proxy](#) (I &ima, const typename I::psite &p)
Constructor.
- I::value [to_value](#) () const
Explicit read access.
- [~proxy](#) ()
Destructor.

10.415.1 Detailed Description

template<typename I>class mln::value::proxy< I >

Generic proxy class for an image pixel value.

The parameter I is an image type.

Definition at line 85 of file value/proxy.hh.

10.415.2 Member Typedef Documentation

10.415.2.1 template<typename I> typedef void mln::value::proxy< I >::enc

Encoding associated type.

Definition at line 91 of file value/proxy.hh.

10.415.2.2 template<typename I> typedef I::value mln::value::proxy< I >::equiv

Equivalent associated type.

Definition at line 94 of file value/proxy.hh.

10.415.3 Constructor & Destructor Documentation

10.415.3.1 template<typename I > [proxy](#)< I >::proxy () [inline]

Constructor.

Definition at line 150 of file value/proxy.hh.

10.415.3.2 `template<typename I> proxy<I>::proxy (I & ima, const typename I::psite & p)` `[inline]`

Constructor.

Definition at line 157 of file value/proxy.hh.

10.415.3.3 `template<typename I> proxy<I>::~~proxy ()` `[inline]`

Destructor.

Definition at line 165 of file value/proxy.hh.

10.415.4 Member Function Documentation

10.415.4.1 `template<typename I> proxy<I> & proxy<I>::operator= (const proxy<I> & rhs)` `[inline]`

Assignment (write access); replacement for default op.

Definition at line 186 of file value/proxy.hh.

References `mln::value::proxy<I>::to_value()`.

10.415.4.2 `template<typename I> template<typename J> proxy<I> & proxy<I>::operator= (const proxy<J> & rhs)` `[inline]`

Assignment (write access); with other proxy.

Definition at line 199 of file value/proxy.hh.

References `mln::value::proxy<I>::to_value()`.

10.415.4.3 `template<typename I> I::value proxy<I>::to_value () const` `[inline]`

Explicit read access.

Definition at line 226 of file value/proxy.hh.

Referenced by `mln::value::proxy<I>::operator=()`.

10.416 mln::value::qt::rgb32 Struct Reference

Color class for red-green-blue where every component is n-bit encoded.

`#include <rgb32.hh>`

Inherits `mln::value::Vectorial<E>`, and `mln::value::internal::value_like_<V, C, N, E>`.

Public Member Functions

- `rgb32 & operator= (const rgb32 &rhs)`
Assignment.
- `rgb32 ()`
Constructor without argument.
- `rgb32 (int r, int g, int b)`
Constructor from component values.
- `rgb32 (const algebra::vec<3, int> &rhs)`

Constructor from a algebra::vec.

- `int_u < 8 > red () const`

Acces to red/green/blue component.

- `rgb32 (const mln::literal::zero_t &)`

Constructors with literals.

Static Public Attributes

- static const `rgb32 zero`

Zero value.

10.416.1 Detailed Description

Color class for red-green-blue where every component is n-bit encoded.

Definition at line 197 of file rgb32.hh.

10.416.2 Constructor & Destructor Documentation

10.416.2.1 `mln::value::qt::rgb32::rgb32 ()` `[inline]`

Constructor without argument.

Definition at line 385 of file rgb32.hh.

10.416.2.2 `mln::value::qt::rgb32::rgb32 (int r, int g, int b)` `[inline]`

Constructor from component values.

Definition at line 427 of file rgb32.hh.

10.416.2.3 `mln::value::qt::rgb32::rgb32 (const algebra::vec< 3, int > & rhs)` `[inline]`

Constructor from a algebra::vec.

Definition at line 391 of file rgb32.hh.

10.416.2.4 `mln::value::qt::rgb32::rgb32 (const mln::literal::zero_t &)` `[inline]`

Constructors with literals.

Definition at line 442 of file rgb32.hh.

10.416.3 Member Function Documentation

10.416.3.1 `rgb32 & mln::value::qt::rgb32::operator= (const rgb32 & rhs)` `[inline]`

Assignment.

Definition at line 623 of file rgb32.hh.

10.416.3.2 `int_u<8> mIn::value::qt::rgb32::red () const [inline]`

Access to red/green/blue component.

Definition at line 212 of file rgb32.hh.

10.416.4 Member Data Documentation

10.416.4.1 `const rgb32 mIn::value::qt::rgb32::zero [static]`

Zero value.

Definition at line 272 of file rgb32.hh.

10.417 mIn::value::rgb< n > Struct Template Reference

Color class for red-green-blue where every component is n-bit encoded.

`#include <rgb.hh>`

Inherits `mIn::value::Vectorial< E >`, and `mIn::value::internal::value_like_< V, C, N, E >`.

Public Member Functions

- `rgb< n > & operator= (const rgb< n > &rhs)`
Assignment.
- `rgb ()`
Constructor without argument.
- `rgb (int r, int g, int b)`
Constructor from component values.
- `rgb (const algebra::vec< 3, int > &rhs)`
Constructor from a algebra::vec.
- `int_u< n > red () const`
Access to red/green/blue component.
- `rgb (const mIn::literal::white_t &)`
Constructors with literals.

Static Public Attributes

- `static const rgb< n > zero`
Zero value.

10.417.1 Detailed Description

`template<unsigned n>struct mIn::value::rgb< n >`

Color class for red-green-blue where every component is n-bit encoded.

Definition at line 248 of file value/rgb.hh.

10.417.2 Constructor & Destructor Documentation

10.417.2.1 `template<unsigned n> rgb< n >::rgb () [inline]`

Constructor without argument.

Definition at line 422 of file value/rgb.hh.

10.417.2.2 `template<unsigned n> rgb< n >::rgb (int r, int g, int b) [inline]`

Constructor from component values.

Definition at line 458 of file value/rgb.hh.

10.417.2.3 `template<unsigned n> rgb< n >::rgb (const algebra::vec< 3, int > & rhs) [inline]`

Constructor from a algebra::vec.

Definition at line 428 of file value/rgb.hh.

10.417.2.4 `template<unsigned n> rgb< n >::rgb (const mln::literal::white_t &) [inline]`

Constructors with literals.

Definition at line 473 of file value/rgb.hh.

10.417.3 Member Function Documentation

10.417.3.1 `template<unsigned n> rgb< n > & rgb< n >::operator= (const rgb< n > & rhs) [inline]`

Assignment.

Definition at line 645 of file value/rgb.hh.

10.417.3.2 `template<unsigned n> int_u<n> mln::value::rgb< n >::red () const [inline]`

Acces to red/green/blue component.

Definition at line 264 of file value/rgb.hh.

Referenced by mln::fun::v2v::rgb8_to_rgbn< n >::operator()().

10.417.4 Member Data Documentation

10.417.4.1 `template<unsigned n> const rgb< n > rgb< n >::zero [static]`

Zero value.

Definition at line 322 of file value/rgb.hh.

10.418 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, 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.

10.418.1 Detailed Description

```
template<typename T>struct mln::value::set< T >
```

Class that defines the set of values of type T.

This is the exhaustive set of values obtainable from type T.

Definition at line 65 of file value/set.hh.

10.418.2 Member Function Documentation

10.418.2.1 `template<typename T > const set< T > & mln::value::set< T >::the () [inline], [static]`

Return a singleton.

Definition at line 80 of file value/set.hh.

10.419 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< sign >.

Public Types

- typedef int [enc](#)
FIXME Are these typedefs correct?
- typedef int [equiv](#)
Define the equivalent type.

Public Member Functions

- [operator int](#) () const
Conversion to an integer.
- [sign](#) & [operator=](#) (int i)
Assignment from an integer.
- [sign](#) ()
Constructor without argument.
- [sign](#) (int i)
Constructor from an integer.

- `sign` (const `mln::literal::zero_t` &)

Constructors/assignments with literals.

Static Public Attributes

- static const `sign one` = 1

Unit value.

- static const `sign zero` = 0

Zero value.

10.419.1 Detailed Description

The sign class represents the value type composed by the set (-1, 0, 1) sign value type is a subset of the int value type.

Definition at line 49 of file value/sign.hh.

10.419.2 Member Typedef Documentation

10.419.2.1 `typedef int mln::value::sign::enc`

FIXME Are these typedefs correct?

Define the encoding type

Definition at line 55 of file value/sign.hh.

10.419.2.2 `typedef int mln::value::sign::equiv`

Define the equivalent type.

Definition at line 58 of file value/sign.hh.

10.419.3 Constructor & Destructor Documentation

10.419.3.1 `mln::value::sign::sign ()` `[inline]`

Constructor without argument.

Definition at line 119 of file value/sign.hh.

10.419.3.2 `mln::value::sign::sign (int i)` `[inline]`

Constructor from an integer.

Definition at line 137 of file value/sign.hh.

10.419.3.3 `mln::value::sign::sign (const mln::literal::zero_t &)` `[inline]`

Constructors/assignments with literals.

Definition at line 155 of file value/sign.hh.

10.419.4 Member Function Documentation

10.419.4.1 mln::value::sign::operator int () const [inline]

Conversion to an integer.

Definition at line 124 of file value/sign.hh.

10.419.4.2 sign & mln::value::sign::operator= (int i) [inline]

Assignment from an integer.

Definition at line 146 of file value/sign.hh.

10.419.5 Member Data Documentation

10.419.5.1 const sign mln::value::sign::one = 1 [static]

Unit value.

Definition at line 88 of file value/sign.hh.

10.419.5.2 const sign mln::value::sign::zero = 0 [static]

Zero value.

Definition at line 85 of file value/sign.hh.

10.420 mln::value::stack_image< n, I > Struct Template Reference

Stack image class.

```
#include <stack.hh>
```

Inherits mln::internal::image_value_morpher< I, algebra::vec< n, I::value >, stack_image< n, I > >.

Public Types

- typedef I::domain_t [domain_t](#)
[Site_Set](#) associated type.
- typedef
internal::helper_stack_image_lvalue_
< n, I >::ret [lvalue](#)
Return type of read-write access.
- typedef I::psite [psite](#)
Point_Site associated type.
- typedef [value](#) [rvalue](#)
Return type of read-only access.
- typedef [stack_image](#)< n,
tag::image_< I > > [skeleton](#)
Skeleton.
- typedef algebra::vec< n,
typename I::value > [value](#)
[Value](#) associated type.

Public Member Functions

- `bool is_valid () const`
Test if this image has been initialized.
- `rvalue operator() (const psite &p) const`
Read-only access of pixel value at point site p .
- `lvalue operator() (const psite &)`
Read-write access of pixel value at point site p .
- `stack_image (const algebra::vec< n, I > &imas)`
Constructors.

10.420.1 Detailed Description

`template<unsigned n, typename I> struct mln::value::stack_image< n, I >`

Stack image class.

`mln::value::stack_image` stores a vector of n images of the same domain.

The parameter n is the number of images, I is the type of a stack element. Acces a value will compute a vector which contains n coordinates : `[stack[0](p), stack[1](p), ... , stack[n](p)]`

Definition at line 145 of file `stack.hh`.

10.420.2 Member Typedef Documentation

10.420.2.1 `template<unsigned n, typename I> typedef I ::domain_t mln::value::stack_image< n, I >::domain_t`

`Site_Set` associated type.

Definition at line 154 of file `stack.hh`.

10.420.2.2 `template<unsigned n, typename I> typedef internal::helper_stack_image_lvalue_<n,I>::ret mln::value::stack_image< n, I >::lvalue`

Return type of read-write access.

Definition at line 166 of file `stack.hh`.

10.420.2.3 `template<unsigned n, typename I> typedef I ::psite mln::value::stack_image< n, I >::psite`

`Point_Site` associated type.

Definition at line 151 of file `stack.hh`.

10.420.2.4 `template<unsigned n, typename I> typedef value mln::value::stack_image< n, I >::rvalue`

Return type of read-only access.

The rvalue type is not a const reference, since the value type is built on the fly, and return by value (copy).

Definition at line 163 of file `stack.hh`.

10.420.2.5 `template<unsigned n, typename I> typedef stack_image< n, tag::image_<I> > mln::value::stack_image< n, I>::skeleton`

Skeleton.

Definition at line 170 of file stack.hh.

10.420.2.6 `template<unsigned n, typename I> typedef algebra::vec<n, typename I::value> mln::value::stack_image< n, I>::value`

[Value](#) associated type.

Definition at line 157 of file stack.hh.

10.420.3 Constructor & Destructor Documentation

10.420.3.1 `template<unsigned n, typename I> stack_image< n, I>::stack_image (const algebra::vec< n, I> & imas) [inline]`

Constructors.

Definition at line 236 of file stack.hh.

10.420.4 Member Function Documentation

10.420.4.1 `template<unsigned n, typename I> bool stack_image< n, I>::is_valid () const [inline]`

Test if this image has been initialized.

Definition at line 255 of file stack.hh.

10.420.4.2 `template<unsigned n, typename I> stack_image< n, I>::rvalue stack_image< n, I>::operator() (const psite & p) const [inline]`

Read-only access of pixel value at point site *p*.

Definition at line 277 of file stack.hh.

10.420.4.3 `template<unsigned n, typename I> stack_image< n, I>::lvalue stack_image< n, I>::operator() (const psite & p) [inline]`

Read-write access of pixel value at point site *p*.

Definition at line 296 of file stack.hh.

10.421 mln::value::super_value< sign > Struct Template Reference

Specializations:

```
#include <super_value.hh>
```

10.421.1 Detailed Description

```
template<> struct mln::value::super_value< sign >
```

Specializations:

Sign type is a subset of the short value type.

Definition at line 56 of file super_value.hh.

10.422 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
}

10.422.1 Detailed Description

```
template<typename T, typename V> struct mln::value::value_array< T, V >
```

Generic array class over indexed by a value set with type T.

Definition at line 45 of file value_array.hh.

10.422.2 Constructor & Destructor Documentation

10.422.2.1 `template<typename T , typename V > mln::value::value_array< T, V >::value_array () [inline]`

Constructors.

```
{
```

Definition at line 89 of file value_array.hh.

10.422.3 Member Function Documentation

10.422.3.1 `template<typename T , typename V > const V & mln::value::value_array< T, V >::operator() (const T & v) const [inline]`

```
}
```

Access elements through a value of T. {

Definition at line 128 of file value_array.hh.

```
10.422.3.2  template<typename T , typename V > const V & mln::value::value_array< T, V >::operator[] ( unsigned i )
           const [inline]

}
```

Access elements through array indexes. {

Definition at line 152 of file value_array.hh.

```
10.422.3.3  template<typename T , typename V > const mln::value::set< T > & mln::value::value_array< T, V >::vset
           ( )const [inline]

}
```

Reference to the set of T.

Definition at line 144 of file value_array.hh.

10.423 mln::Vertex< E > Struct Template Reference

[Vertex](#) category flag type.

```
#include <vertex.hh>
```

10.423.1 Detailed Description

```
template<typename E>struct mln::Vertex< E >
```

[Vertex](#) category flag type.

Definition at line 53 of file vertex.hh.

10.424 mln::vertex_image< P, V, G > Class Template Reference

[Image](#) based on graph vertices.

```
#include <vertex_image.hh>
```

Inherits mln::pw::internal::image_base< fun::i2v::array< V >, p_vertices< G, internal::vfsite_selector< P, G >::site_function_t >, vertex_image< P, V, G > >.

Public Types

- typedef G [graph_t](#)
The type of the underlying graph.
- typedef [vertex_nbh_t](#) nbh_t
Neighborhood type.
- typedef
internal::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 [vertex_win_t](#) win_t

Window type.

Public Member Functions

- [vertex_image](#) ()

Constructors.

- rvalue [operator](#)() (unsigned v_id) const

Value accessors/operators overloads.

10.424.1 Detailed Description

```
template<typename P, typename V, typename G = util::graph>class mln::vertex_image< P, V, G >
```

[Image](#) based on graph vertices.

Definition at line 126 of file core/image/vertex_image.hh.

10.424.2 Member Typedef Documentation

10.424.2.1 `template<typename P, typename V, typename G = util::graph> typedef G mln::vertex_image< P, V, G >::graph_t`

The type of the underlying graph.

Definition at line 141 of file core/image/vertex_image.hh.

10.424.2.2 `template<typename P, typename V, typename G = util::graph> typedef vertex_nbh_t mln::vertex_image< P, V, G >::nbh_t`

[Neighborhood](#) type.

Definition at line 165 of file core/image/vertex_image.hh.

10.424.2.3 `template<typename P, typename V, typename G = util::graph> typedef internal::vfsite_selector<P,G>::site_function_t mln::vertex_image< P, V, G >::site_function_t`

[Function](#) mapping graph elements to sites.

Definition at line 145 of file core/image/vertex_image.hh.

10.424.2.4 `template<typename P, typename V, typename G = util::graph> typedef vertex_image< tag::psite_<P>, tag::value_<V>, tag::graph_<G> > mln::vertex_image< P, V, G >::skeleton`

Skeleton type.

Definition at line 152 of file core/image/vertex_image.hh.

10.424.2.5 `template<typename P, typename V, typename G = util::graph> typedef vertex_win_t mln::vertex_image< P, V, G >::win_t`

[Window](#) type.

Definition at line 163 of file core/image/vertex_image.hh.

10.424.3 Constructor & Destructor Documentation

10.424.3.1 `template<typename P, typename V, typename G > vertex_image< P, V, G >::vertex_image ()`
`[inline]`

Constructors.

Definition at line 247 of file core/image/vertex_image.hh.

10.424.4 Member Function Documentation

10.424.4.1 `template<typename P, typename V, typename G > vertex_image< P, V, G >::rvalue vertex_image< P, V, G >::operator() (unsigned v_id) const`

[Value](#) accessors/operators overloads.

Definition at line 292 of file core/image/vertex_image.hh.

10.425 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, violent_cast_image< T, I > >.

Public Types

- typedef T [lvalue](#)
Return type of read-write access.
- typedef T [rvalue](#)
Return type of read-only access.
- typedef [violent_cast_image](#)
`< tag::value_< T >`
`, tag::image_< I > >` [skeleton](#)
Skeleton.
- typedef T [value](#)
[Value](#) associated type.

Public Member Functions

- T [operator\(\)](#) (const typename I::psite &p) const
Read-only access of pixel value at point site p.
- T [operator\(\)](#) (const typename I::psite &p)
Mutable access is only OK for reading (not writing).
- [violent_cast_image](#) (const [Image](#)< I > &ima)
Constructor.

10.425.1 Detailed Description

```
template<typename T, typename I> struct mln::violent_cast_image< T, I >
```

Violently cast image values to a given type.

Definition at line 113 of file violent_cast_image.hh.

10.425.2 Member Typedef Documentation

10.425.2.1 `template<typename T, typename I> typedef T mln::violent_cast_image< T, I >::lvalue`

Return type of read-write access.

Definition at line 123 of file violent_cast_image.hh.

10.425.2.2 `template<typename T, typename I> typedef T mln::violent_cast_image< T, I >::rvalue`

Return type of read-only access.

Definition at line 120 of file violent_cast_image.hh.

10.425.2.3 `template<typename T, typename I> typedef violent_cast_image< tag::value_<T>, tag::image_<I> > mln::violent_cast_image< T, I >::skeleton`

Skeleton.

Definition at line 126 of file violent_cast_image.hh.

10.425.2.4 `template<typename T, typename I> typedef T mln::violent_cast_image< T, I >::value`

[Value](#) associated type.

Definition at line 117 of file violent_cast_image.hh.

10.425.3 Constructor & Destructor Documentation

10.425.3.1 `template<typename T, typename I> violent_cast_image< T, I >::violent_cast_image (const Image< I > & ima) [inline]`

Constructor.

Definition at line 173 of file violent_cast_image.hh.

10.425.4 Member Function Documentation

10.425.4.1 `template<typename T, typename I> T violent_cast_image< T, I >::operator() (const typename I::psite & p) const [inline]`

Read-only access of pixel value at point site *p*.

Definition at line 192 of file violent_cast_image.hh.

10.425.4.2 `template<typename T, typename I> T violent_cast_image< T, I >::operator() (const typename I::psite & p)`
`[inline]`

Mutable access is only OK for reading (not writing).

Definition at line 201 of file `violent_cast_image.hh`.

10.426 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 >, w_window< D, W > >`.

Public Types

- typedef with_w_
`< dpsites_bkd_piter< w_window
< D, W > >, W > bkd_qiter`
Site_iterator type to browse (backward) the points of a generic w_window.
- typedef D dpsite
Dpsite associated type.
- typedef with_w_
`< dpsites_fwd_piter< w_window
< D, W > >, W > fwd_qiter`
Site_iterator type to browse (forward) the points of a generic w_window.
- typedef W weight
Weight associated type.

Public Member Functions

- void `clear` ()
Clear this window.
- `w_window< D, W > & insert` (const W &w, const D &d)
Insert a couple of weight w and delta-point d.
- bool `is_symmetric` () const
Test if the window is symmetric.
- const `std::vector< D > & std_vector` () const
Give access to the vector of delta-points.
- void `sym` ()
Apply a central symmetry to the window.
- W `w` (unsigned i) const
Give the i-th weight.
- `w_window` ()
Constructor without argument.
- const `std::vector< W > & weights` () const
Give access to the vector of weights.
- const `mln::window< D > & win` () const
Give the corresponding window.

Related Functions

(Note that these are not member functions.)

- `template<typename D , typename W >`
`std::ostream & operator<< (std::ostream &ostr, const w_window< D, W > &w_win)`
Print a weighted window `w_win` into an output stream `ostr`.
- `template<typename D , typename Wl , typename Wr >`
`bool operator== (const w_window< D, Wl > &lhs, const w_window< D, Wr > &rhs)`
Equality test between two weighted windows `lhs` and `rhs`.

10.426.1 Detailed Description

`template<typename D, typename W> struct mln::w_window< D, W >`

Generic [w_window](#) class.

This type of [w_window](#) is just like a set of delta-points. The parameter `D` is the type of delta-points; the parameter `W` is the type of weights.

Definition at line 100 of file `core/w_window.hh`.

10.426.2 Member Typedef Documentation

10.426.2.1 `template<typename D, typename W> typedef with_w_< dpsites_bkd_piter< w_window<D, W> >, W > mln::w_window< D, W >::bkd_qiter`

[Site_iterator](#) type to browse (backward) the points of a generic [w_window](#).

Definition at line 114 of file `core/w_window.hh`.

10.426.2.2 `template<typename D, typename W> typedef D mln::w_window< D, W >::dpsite`

Dpsite associated type.

Definition at line 104 of file `core/w_window.hh`.

10.426.2.3 `template<typename D, typename W> typedef with_w_< dpsites_fwd_piter< w_window<D, W> >, W > mln::w_window< D, W >::fwd_qiter`

[Site_iterator](#) type to browse (forward) the points of a generic [w_window](#).

Definition at line 111 of file `core/w_window.hh`.

10.426.2.4 `template<typename D, typename W> typedef W mln::w_window< D, W >::weight`

Weight associated type.

Definition at line 107 of file `core/w_window.hh`.

10.426.3 Constructor & Destructor Documentation

10.426.3.1 `template<typename D , typename W > w_window< D, W >::w_window () [inline]`

Constructor without argument.

Definition at line 213 of file `core/w_window.hh`.

10.426.4 Member Function Documentation

10.426.4.1 `template<typename D , typename W > void w_window< D, W >::clear () [inline]`

Clear this window.

Definition at line 308 of file core/w_window.hh.

10.426.4.2 `template<typename D , typename W > w_window< D, W > & w_window< D, W >::insert (const W & w, const D & d) [inline]`

Insert a couple of weight *w* and delta-point *d*.

Definition at line 254 of file core/w_window.hh.

Referenced by `mln::w_window< D, W >::sym()`, `mln::make::w_window()`, `mln::make::w_window1d()`, `mln::make::w_window3d()`, and `mln::make::w_window_directional()`.

10.426.4.3 `template<typename D , typename W > bool w_window< D, W >::is_symmetric () const [inline]`

Test if the window is symmetric.

Definition at line 284 of file core/w_window.hh.

References `mln::w_window< D, W >::sym()`.

10.426.4.4 `template<typename D , typename W > const std::vector< D > & w_window< D, W >::std_vector () const [inline]`

Give access to the vector of delta-points.

Definition at line 228 of file core/w_window.hh.

10.426.4.5 `template<typename D , typename W > void w_window< D, W >::sym () [inline]`

Apply a central symmetry to the window.

Definition at line 296 of file core/w_window.hh.

References `mln::w_window< D, W >::insert()`.

Referenced by `mln::w_window< D, W >::is_symmetric()`.

10.426.4.6 `template<typename D , typename W > W w_window< D, W >::w (unsigned i) const [inline]`

Give the *i*-th weight.

Definition at line 244 of file core/w_window.hh.

10.426.4.7 `template<typename D , typename W > const std::vector< W > & w_window< D, W >::weights () const [inline]`

Give access to the vector of weights.

Definition at line 236 of file core/w_window.hh.

Referenced by `mln::w_window< D, W >::operator==()`.

10.426.4.8 `template<typename D , typename W > const mln::window< D > & w_window< D, W >::win () const`
`[inline]`

Give the corresponding window.

Definition at line 220 of file `core/w_window.hh`.

Referenced by `mln::w_window< D, W >::operator==()`.

10.426.5 Friends And Related Function Documentation

10.426.5.1 `template<typename D , typename W > std::ostream & operator<< (std::ostream & ostr, const w_window< D, W > & w_win)` `[related]`

Print a weighted window `w_win` into an output stream `ostr`.

Definition at line 420 of file `core/w_window.hh`.

10.426.5.2 `template<typename D , typename Wl , typename Wr > bool operator==(const w_window< D, Wl > & lhs, const w_window< D, Wr > & rhs)` `[related]`

Equality test between two weighted windows `lhs` and `rhs`.

Definition at line 430 of file `core/w_window.hh`.

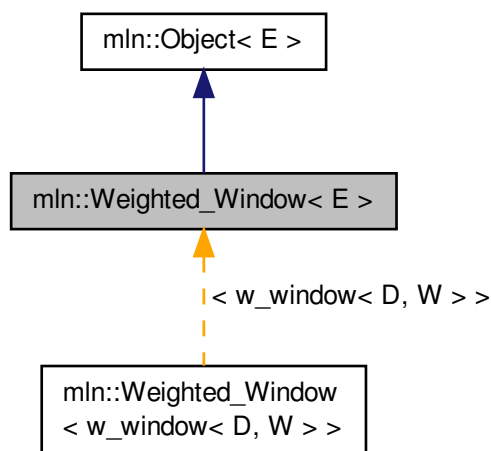
References `mln::w_window< D, W >::weights()`, and `mln::w_window< D, W >::win()`.

10.427 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`.

10.427.1 Detailed Description

```
template<typename E>struct mln::Weighted_Window< E >
```

Base class for implementation classes that are `weighted_windows`.

See Also

[mln::doc::Weighted_Window](#) for a complete documentation of this class contents.

Definition at line 68 of file `weighted_window.hh`.

10.427.2 Friends And Related Function Documentation

10.427.2.1 `template<typename W > W operator- (const Weighted_Window< W > & rhs)` [\[related\]](#)

Compute the symmetrical weighted window of `rhs`.

10.428 mln::win::backdiag2d Struct Reference

Diagonal line window defined on the 2D square grid.

```
#include <backdiag2d.hh>
```

Inherits `mln::internal::classical_window_base< dpoint2d, backdiag2d >`.

Public Member Functions

- `backdiag2d (unsigned length)`
Constructor.
- `unsigned length () const`
Give the diagonal length, that is, its width.

10.428.1 Detailed Description

Diagonal line window defined on the 2D square grid.

An `backdiag2d` is centered and symmetric. its width (length) is odd.

For instance:

```
*  ○
*  ○
*  x
*  ○
*  ○
*
```

is defined with length = 5.

Definition at line 63 of file backdiag2d.hh.

10.428.2 Constructor & Destructor Documentation

10.428.2.1 mln::win::backdiag2d (unsigned *length*) [inline]

Constructor.

Parameters

<i>in</i>	<i>length</i>	Length, thus width, of the diagonal line.
-----------	---------------	---

Precondition

length is odd.

Definition at line 93 of file backdiag2d.hh.

10.428.3 Member Function Documentation

10.428.3.1 unsigned mln::win::backdiag2d::length () const [inline]

Give the diagonal length, that is, its width.

Definition at line 105 of file backdiag2d.hh.

10.429 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< dpoint< G, C >, ball< G, C > >.

Public Member Functions

- [ball](#) (unsigned [diameter](#))
Constructor.
- unsigned [diameter](#) () const
Give the ball diameter.

10.429.1 Detailed Description

```
template<typename G, typename C>struct mln::win::ball< G, C >
```

Generic ball window defined on a given grid.

A ball is centered and symmetric; so its diameter is odd.

G is the given grid on which the ball is defined and C is the type of coordinates.

Definition at line 71 of file ball.hh.

10.429.2 Constructor & Destructor Documentation

10.429.2.1 `template<typename G , typename C > ball< G, C >::ball (unsigned diameter)` `[inline]`

Constructor.

Parameters

<code>in</code>	<code><i>diameter</i></code>	Diameter of the ball.
-----------------	------------------------------	-----------------------

Precondition

`diameter` is odd.

Definition at line 99 of file ball.hh.

References `mln::literal::origin`.

10.429.3 Member Function Documentation

10.429.3.1 `template<typename G , typename C > unsigned ball< G, C >::diameter () const` `[inline]`

Give the ball diameter.

Definition at line 122 of file ball.hh.

10.430 mln::win::cube3d Struct Reference

Cube window defined on the 3D grid.

```
#include <cube3d.hh>
```

Inherits `mln::internal::classical_window_base< dpoint3d, cube3d >`.

Public Member Functions

- `cube3d` (unsigned `length`)
Constructor.
- unsigned `length` () const
Give the cube length, that is, its height.

10.430.1 Detailed Description

Cube window defined on the 3D grid.

An `cube3d` is centered and symmetric; so its height (length) is odd.

For instance:

```
*   o o o
*   o o o
*   o o o

*   o o o
*   o x o
*   o o o

*   o o o
*   o o o
```

```
*  o  o  o
*
```

is defined with length = 3.

Definition at line 69 of file cube3d.hh.

10.430.2 Constructor & Destructor Documentation

10.430.2.1 mln::win::cube3d::cube3d (unsigned *length*) [inline]

Constructor.

Parameters

<i>in</i>	<i>length</i>	Length, thus height, of the cube3d .
-----------	---------------	--

Precondition

length is odd.

Definition at line 99 of file cube3d.hh.

10.430.3 Member Function Documentation

10.430.3.1 unsigned mln::win::cube3d::length () const [inline]

Give the cube length, that is, its height.

Definition at line 113 of file cube3d.hh.

10.431 mln::win::cuboid3d Struct Reference

Cuboid defined on the 3-D square grid.

```
#include <cuboid3d.hh>
```

Inherits mln::internal::classical_window_base< dpoint3d, 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.

10.431.1 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 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
      o o o o o o o
      o o o o o o o

```

is defined with depth = 3, height = 5 and width = 7.

Reference: <http://en.wikipedia.org/wiki/Cuboid>

Definition at line 80 of file cuboid3d.hh.

10.431.2 Constructor & Destructor Documentation

10.431.2.1 `mln::win::cuboid3d::cuboid3d (unsigned depth, unsigned height, unsigned width)` `[inline]`

Constructor.

Parameters

<code>in</code>	<i>depth</i>	The depth of the cuboid3d .
<code>in</code>	<i>height</i>	The height of the cuboid3d .
<code>in</code>	<i>width</i>	The width of the cuboid3d .

Precondition

Argument *depth*, *height* and *width* must be odd.

Definition at line 125 of file cuboid3d.hh.

10.431.3 Member Function Documentation

10.431.3.1 `unsigned mln::win::cuboid3d::depth () const` `[inline]`

Accessors.

Return the depth of the cuboid.

Definition at line 146 of file cuboid3d.hh.

10.431.3.2 unsigned mln::win::cuboid3d::height () const [inline]

Return the height of the cuboid.

Definition at line 153 of file cuboid3d.hh.

10.431.3.3 unsigned mln::win::cuboid3d::volume () const [inline]

Return the volume of the cuboid.

Definition at line 167 of file cuboid3d.hh.

10.431.3.4 unsigned mln::win::cuboid3d::width () const [inline]

Return the width of the cuboid.

Definition at line 160 of file cuboid3d.hh.

10.432 mln::win::diag2d Struct Reference

Diagonal line window defined on the 2D square grid.

```
#include <diag2d.hh>
```

Inherits mln::internal::classical_window_base< dpoint2d, diag2d >.

Public Member Functions

- [diag2d](#) (unsigned [length](#))
Constructor.
- unsigned [length](#) () const
Give the diagonal length, that is, its width.

10.432.1 Detailed Description

Diagonal line window defined on the 2D square grid.

An [diag2d](#) is centered and symmetric. its width (length) is odd.

For instance:

```

*           o
*           o
*       x
*   o
*   o
*   o
*

```

is defined with length = 5.

Definition at line 63 of file diag2d.hh.

10.432.2 Constructor & Destructor Documentation

10.432.2.1 mln::win::diag2d::diag2d (unsigned [length](#)) [inline]

Constructor.

Parameters

in	length	Length, thus width, of the diagonal line.
----	--------	---

Precondition

length is odd.

Definition at line 93 of file diag2d.hh.

References length().

10.432.3 Member Function Documentation

10.432.3.1 unsigned mln::win::diag2d::length () const [inline]

Give the diagonal length, that is, its width.

Definition at line 106 of file diag2d.hh.

Referenced by diag2d().

10.433 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< dpoint< M, C >, line< M, i, C > >.

Public Types

- enum
Direction.

Public Member Functions

- unsigned [length](#) () const
Give the line length.
- [line](#) (unsigned [length](#))
Constructor.
- unsigned [size](#) () const
Give the line size, that is, its length.

10.433.1 Detailed Description

```
template<typename M, unsigned i, typename C>struct mln::win::line< M, i, C >
```

Generic line window defined on a given grid in the given dimension.

An line is centered and symmetric; so its length is odd.

M is the given grid on which the line is defined, i is the given dimension of the line end C is the type of the coordinates.

See Also

[mln::win::hline2d](#) for an exemple of his use.

Definition at line 73 of file win/line.hh.

10.433.2 Member Enumeration Documentation

10.433.2.1 `template<typename M , unsigned i, typename C > anonymous enum`

Direction.

Definition at line 76 of file win/line.hh.

10.433.3 Constructor & Destructor Documentation

10.433.3.1 `template<typename M , unsigned i, typename C > line< M, i, C >::line (unsigned length) [inline]`

Constructor.

Parameters

<i>in</i>	<i>length</i>	Length of the line.
-----------	---------------	---------------------

Precondition

length is odd.

Definition at line 106 of file win/line.hh.

References `mln::dpoint< G, C >::set_all()`.

10.433.4 Member Function Documentation

10.433.4.1 `template<typename M , unsigned i, typename C > unsigned line< M, i, C >::length () const [inline]`

Give the line length.

Definition at line 125 of file win/line.hh.

10.433.4.2 `template<typename M , unsigned i, typename C > unsigned line< M, i, C >::size () const [inline]`

Give the line size, that is, its length.

Definition at line 132 of file win/line.hh.

10.434 mln::win::multiple< W, F > Class Template Reference

Multiple window.

```
#include <multiple.hh>
```

Inherits `mln::internal::window_base< W::dpsite, multiple< W, F > >`, and `check_t< mlc_is(mln_trait_window_ - size(W), trait::window::size::fixed), mlc_is(mln_trait_window_support(W), trait::window::support::regular) >`.

10.434.1 Detailed Description

```
template<typename W, typename F>class mln::win::multiple< W, F >
```

Multiple window.

Definition at line 76 of file multiple.hh.

10.435 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, multiple_size< n, W, F > >, and check_t< mlc_bool(n > 1), mlc_is(mln_trait_window_size(W), trait::window::size::fixed), mlc_is(mln_trait_window_support(W), trait::window::support::regular) >.

10.435.1 Detailed Description

```
template<unsigned n, typename W, typename F>class mln::win::multiple_size< n, W, F >
```

Definition of a multiple-size window.

Definition at line 78 of file multiple_size.hh.

10.436 mln::win::octagon2d Struct Reference

Octagon window defined on the 2D square grid.

```
#include <octagon2d.hh>
```

Inherits mln::internal::classical_window_base< dpoint2d, octagon2d >.

Public Member Functions

- unsigned [area](#) () const
Give the area.
- unsigned [length](#) () const
Give the octagon length, that is, its width.
- [octagon2d](#) (unsigned [length](#))
Constructor.

10.436.1 Detailed Description

Octagon window defined on the 2D square grid.

An [octagon2d](#) is centered and symmetric.

The length L of the octagon is such as $L = 6 * l + 1$ where $l \geq 0$.

For instance:

```

*      o o o
*    o o o o o
*  o o o o o o o
* o o o x o o o
* o o o o o o o
*   o o o o o
*     o o o
*

```

is defined with $L = 7$ ($l = 1$).

Definition at line 67 of file octagon2d.hh.

10.436.2 Constructor & Destructor Documentation

10.436.2.1 mln::win::octagon2d::octagon2d (unsigned *length*) [inline]

Constructor.

Parameters

in	<i>length</i>	Length, of the octagon.
----	---------------	-------------------------

Precondition

length is such as $length = 6 \cdot x + 1$ where $x \geq 0$.

Definition at line 101 of file octagon2d.hh.

10.436.3 Member Function Documentation

10.436.3.1 unsigned mln::win::octagon2d::area () const [inline]

Give the area.

Definition at line 157 of file octagon2d.hh.

10.436.3.2 unsigned mln::win::octagon2d::length () const [inline]

Give the octagon length, that is, its width.

Definition at line 145 of file octagon2d.hh.

10.437 mln::win::rectangle2d Struct Reference

Rectangular window defined on the 2D square grid.

```
#include <rectangle2d.hh>
```

Inherits mln::internal::classical_window_base< dpoint2d, rectangle2d >.

Public Member Functions

- unsigned [area](#) () const
Give the rectangle area.
- unsigned [height](#) () const
Give the rectangle height.
- [rectangle2d](#) (unsigned [height](#), unsigned [width](#))
Constructor.
- const std::vector< [dpoint2d](#) > & [std_vector](#) () const
Give the std vector of delta-points.
- unsigned [width](#) () const
Give the rectangle width.

10.437.1 Detailed Description

Rectangular window defined on the 2D square grid.

A [rectangle2d](#) is a 2D window with rectangular shape. It is centered and symmetric.

For instance:

```
*  o o o o o
*  o o x o o
*  o o o o o
*
```

is defined with height = 3 and width = 5.

Definition at line 64 of file rectangle2d.hh.

10.437.2 Constructor & Destructor Documentation

10.437.2.1 `mln::win::rectangle2d::rectangle2d (unsigned height, unsigned width)` `[inline]`

Constructor.

Parameters

<code>in</code>	<code><i>height</i></code>	Height of the rectangle2d .
<code>in</code>	<code><i>width</i></code>	Width of the rectangle2d .

Precondition

Height and width are odd.

Definition at line 106 of file rectangle2d.hh.

10.437.3 Member Function Documentation

10.437.3.1 `unsigned mln::win::rectangle2d::area () const` `[inline]`

Give the rectangle area.

Definition at line 132 of file rectangle2d.hh.

10.437.3.2 `unsigned mln::win::rectangle2d::height () const` `[inline]`

Give the rectangle height.

Definition at line 120 of file rectangle2d.hh.

10.437.3.3 `const std::vector< dpoint2d > & mln::win::rectangle2d::std_vector () const` `[inline]`

Give the std vector of delta-points.

Definition at line 145 of file rectangle2d.hh.

10.437.3.4 `unsigned mln::win::rectangle2d::width () const` `[inline]`

Give the rectangle width.

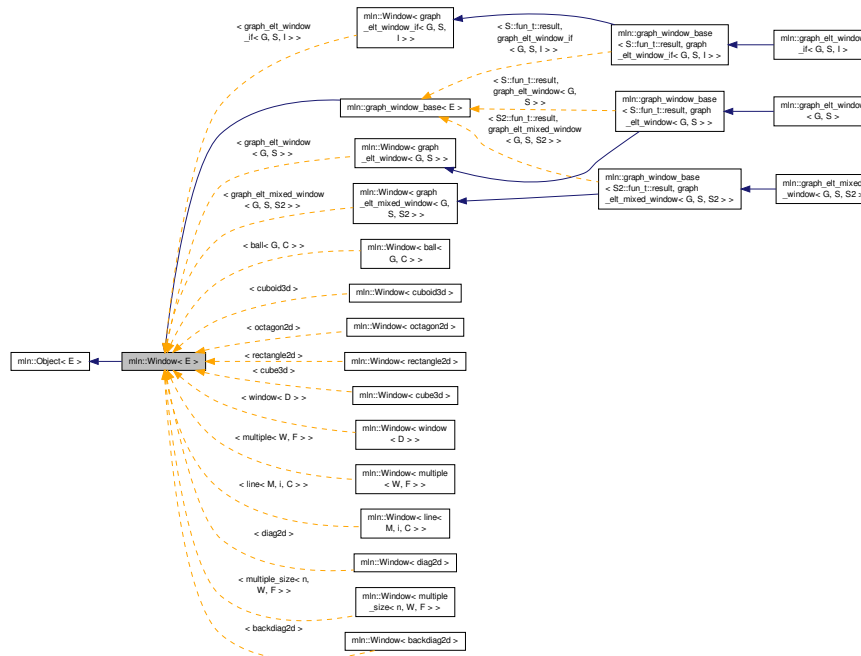
Definition at line 126 of file rectangle2d.hh.

10.438 mln::Window< E > Struct Template Reference

Base class for implementation classes that are windows.

```
#include <window.hh>
```

Inheritance diagram for mln::Window< E >:



10.438.1 Detailed Description

```
template<typename E>struct mln::Window< E >
```

Base class for implementation classes that are windows.

See Also

[mln::doc::Window](#) for a complete documentation of this class contents.

Definition at line 87 of file concept/window.hh.

10.439 mln::window< D > Class Template Reference

Generic window class.

```
#include <window.hh>
```

Inherits mln::internal::window_base< D, window< D > >.

Public Types

- typedef [dpsites_bkd_piter](#)
[< window< D > >](#) [bkd_qiter](#)

- *Site_iterator* type to browse the points of a basic window w.r.t. the reverse ordering of delta-points.
- typedef `dpsites_fwd_piter`
`< window< D > > fwd_qiter`
Site_iterator type to browse the points of a basic window w.r.t. the ordering of delta-points.
- typedef `fwd_qiter` `qiter`
Site_iterator type to browse the points of a basic window whatever the ordering of delta-points.
- typedef `window< D > regular`
Regular window associated type.

Public Member Functions

- void `clear` ()
Clear the window.
- unsigned `delta` () const
Give the maximum coordinate gap between the window center and a window point.
- const D & `dp` (unsigned i) const
*Give the *i*-th delta-point.*
- bool `has` (const D & `dp`) const
*Test if *dp* is in this window definition.*
- `window< D > & insert` (const D & `dp`)
*Insert a delta-point *dp*.*
- template<typename W >
`window< D > & insert` (const `Window< W >` & `win`)
*Insert another window *win*.*
- bool `is_centered` () const
Test if the window is centered.
- bool `is_empty` () const
Test if the window is empty (null size; no delta-point).
- bool `is_symmetric` () const
- void `print` (std::ostream & `ostr`) const
*Print the window definition into *ostr*.*
- unsigned `size` () const
Give the window size, i.e., the number of delta-sites.
- const std::vector< D > & `std_vector` () const
Give the std vector of delta-points.
- void `sym` ()
Apply a central symmetry to the target window.
- `window` ()
Constructor without argument.
- `window< D > & insert` (const typename D::coord & `dind`)

Related Functions

(Note that these are not member functions.)

- template<typename D >
bool `operator==` (const `window< D >` & `lhs`, const `window< D >` & `rhs`)
*Equality comparison between windows *lhs* and *rhs*.*

10.439.1 Detailed Description

`template<typename D> class mln::window< D >`

Generic window class.

This type of window is just like a set of delta-points. The parameter is `D`, type of delta-point.

Definition at line 85 of file window.hh.

10.439.2 Member Typedef Documentation

10.439.2.1 `template<typename D> typedef dpsites_bkd_piter< window<D> > mln::window< D >::bkd_qiter`

[Site_Iterator](#) type to browse the points of a basic window w.r.t. the reverse ordering of delta-points.

Definition at line 124 of file window.hh.

10.439.2.2 `template<typename D> typedef dpsites_fwd_piter< window<D> > mln::window< D >::fwd_qiter`

[Site_Iterator](#) type to browse the points of a basic window w.r.t. the ordering of delta-points.

Definition at line 119 of file window.hh.

10.439.2.3 `template<typename D> typedef fwd_qiter mln::window< D >::qiter`

[Site_Iterator](#) type to browse the points of a basic window whatever the ordering of delta-points.

Definition at line 129 of file window.hh.

10.439.2.4 `template<typename D> typedef window<D> mln::window< D >::regular`

Regular window associated type.

Definition at line 90 of file window.hh.

10.439.3 Constructor & Destructor Documentation

10.439.3.1 `template<typename D> window< D >::window () [inline]`

Constructor without argument.

The constructed window is empty.

Definition at line 207 of file window.hh.

10.439.4 Member Function Documentation

10.439.4.1 `template<typename D> void window< D >::clear () [inline]`

Clear the window.

Definition at line 254 of file window.hh.

10.439.4.2 `template<typename D> unsigned window< D >::delta () const [inline]`

Give the maximum coordinate gap between the window center and a window point.

Definition at line 262 of file window.hh.

10.439.4.3 `template<typename D> const D & window< D >::dp (unsigned i) const [inline]`

Give the *i*-th delta-point.

Definition at line 302 of file window.hh.

10.439.4.4 `template<typename D> bool window< D >::has (const D & dp) const [inline]`

Test if *dp* is in this window definition.

Definition at line 311 of file window.hh.

10.439.4.5 `template<typename D> window< D > & window< D >::insert (const D & dp) [inline]`

Insert a delta-point *dp*.

Definition at line 327 of file window.hh.

Referenced by `mln::c18()`, `mln::c26()`, `mln::c2_3d_sli()`, `mln::c4_3d()`, `mln::c6()`, `mln::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()`.

10.439.4.6 `template<typename D> template<typename W> window< D > & window< D >::insert (const Window< W > & win) [inline]`

Insert another window *win*.

Definition at line 337 of file window.hh.

10.439.4.7 `template<typename D> window< D > & window< D >::insert (const typename D::coord & dind) [inline]`

Insertion of a delta-point with different numbers of arguments (coordinates) w.r.t. the dimension.

Definition at line 349 of file window.hh.

10.439.4.8 `template<typename D> bool window< D >::is_centered () const [inline]`

Test if the window is centered.

Returns

True if the delta-point 0 belongs to the window.

Definition at line 226 of file window.hh.

References `mln::literal::zero`.

10.439.4.9 `template<typename D> bool window< D >::is_empty () const [inline]`

Test if the window is empty (null size; no delta-point).

Definition at line 246 of file window.hh.

10.439.4.10 `template<typename D> bool window< D >::is_symmetric () const [inline]`

Test if the window is symmetric.

```
\return True if for every dp of this window, -dp is also in
this window.
```

Definition at line 216 of file window.hh.

References `mln::window< D >::sym()`.

10.439.4.11 `template<typename D> void window< D >::print (std::ostream & ostr) const [inline]`

Print the window definition into `ostr`.

Definition at line 390 of file window.hh.

10.439.4.12 `template<typename D> unsigned window< D >::size () const [inline]`

Give the window size, i.e., the number of delta-sites.

Definition at line 294 of file window.hh.

Referenced by `mln::win_c4p()`, `mln::win_c4p_3d()`, `mln::win_c8p()`, and `mln::win_c8p_3d()`.

10.439.4.13 `template<typename D> const std::vector< D > & window< D >::std_vector () const [inline]`

Give the std vector of delta-points.

Definition at line 319 of file window.hh.

10.439.4.14 `template<typename D> void window< D >::sym () [inline]`

Apply a central symmetry to the target window.

Definition at line 234 of file window.hh.

References `mln::window< D >::insert()`.

Referenced by `mln::window< D >::is_symmetric()`.

10.439.5 Friends And Related Function Documentation

10.439.5.1 `template<typename D> bool operator==(const window< D > & lhs, const window< D > & rhs) [related]`

Equality comparison between windows `lhs` and `rhs`.

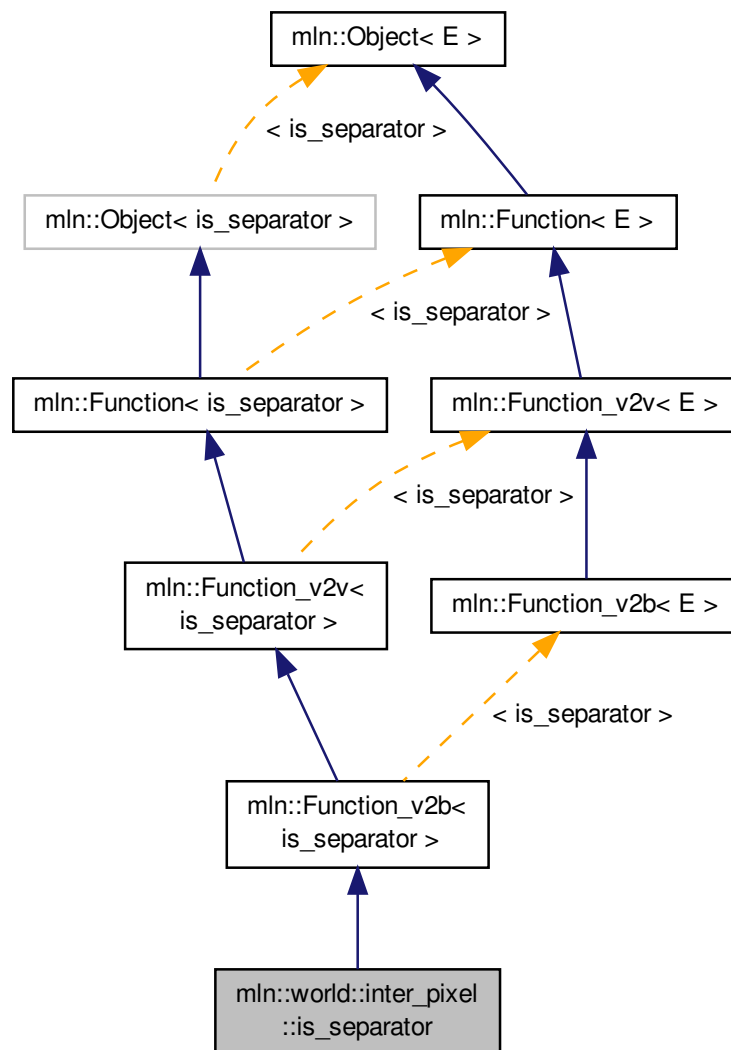
Definition at line 400 of file window.hh.

10.440 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:



10.440.1 Detailed Description

Functor returning whether a site is a separator in an inter-pixel image.

Definition at line 52 of file is_separator.hh.

10.441 point< G, C > Struct Template Reference

Forward declarations.

```
#include <point.hh>
```

10.441.1 Detailed Description

```
template<typename G, typename C>struct point< G, C >
```

Forward declarations.

Definition at line 58 of file point.hh.

10.442 point< G, C > Struct Template Reference

Forward declarations.

```
#include <point.hh>
```

10.442.1 Detailed Description

```
template<typename G, typename C>struct point< G, C >
```

Forward declarations.

Definition at line 58 of file point.hh.

10.443 site_set_impl< Sc > Struct Template Reference

The facade.

```
#include <site_set_impl.hh>
```

10.443.1 Detailed Description

```
template<typename Sc>struct site_set_impl< Sc >
```

The facade.

Parameter *Sc* is the site set component.

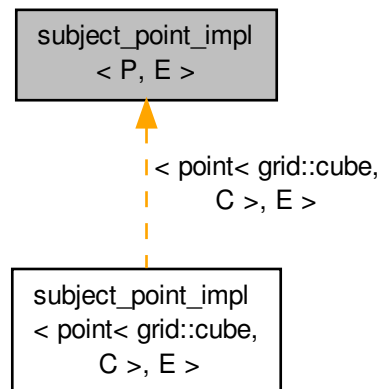
Definition at line 46 of file site_set_impl.hh.

10.444 subject_point_impl< P, E > Struct Template Reference

subject_impl specialization (Proxy)

```
#include <point.hh>
```

Inheritance diagram for subject_point_impl< P, E >:



10.444.1 Detailed Description

```
template<typename P, typename E>struct subject_point_impl< P, E >
```

subject_impl specialization (Proxy)

Definition at line 234 of file point.hh.

10.445 trait::graph< I > Struct Template Reference

Graph traits.

```
#include <morpho.hh>
```

10.445.1 Detailed Description

```
template<typename I>struct trait::graph< I >
```

Graph traits.

Definition at line 73 of file morpho.hh.

10.446 trait::graph< mln::complex_image< 1, G, V > > Struct Template Reference

Graph traits for 1-complexes images.

```
#include <morpho.hh>
```

10.446.1 Detailed Description

```
template<typename G, typename V>struct trait::graph< mln::complex_image< 1, G, V > >
```

Graph traits for 1-complexes images.

Definition at line 133 of file morpho.hh.

10.447 trait::graph< mln::image2d< T > > Struct Template Reference

Graph traits for [mln::image2d](#).

```
#include <morpho.hh>
```

10.447.1 Detailed Description

```
template<typename T>struct trait::graph< mln::image2d< T > >
```

Graph traits for [mln::image2d](#).

Definition at line 94 of file morpho.hh.

10.448 tree_node< T > Class Template Reference

Fwd declarations.

```
#include <tree.hh>
```

10.448.1 Detailed Description

```
template<typename T>class tree_node< T >
```

Fwd declarations.

Definition at line 49 of file tree.hh.

10.449 up_leaf_piter< T > Struct Template Reference

Iterate on tree's leaves in the same way of [up_node_piter](#).

```
#include <data.hh>
```

10.449.1 Detailed Description

```
template<typename T>struct up_leaf_piter< T >
```

Iterate on tree's leaves in the same way of [up_node_piter](#).

Definition at line 90 of file morpho/tree/data.hh.

10.450 up_node_piter< T > Struct Template Reference

Iterate on tree's nodes (component representants) from leaves to roots.

```
#include <data.hh>
```

10.450.1 Detailed Description

```
template<typename T>struct up_node_piter< T >
```

Iterate on tree's nodes (component representants) from leaves to roots.

Definition at line 84 of file morpho/tree/data.hh.

10.451 up_site_piter< T > Struct Template Reference

Iterate on tree's sites from leaves to roots.

```
#include <data.hh>
```

10.451.1 Detailed Description

```
template<typename T>struct up_site_piter< T >
```

Iterate on tree's sites from leaves to roots.

Definition at line 78 of file morpho/tree/data.hh.

Index

- ~proxy
 - mln::value::proxy, [1009](#)
- ~soft_heap
 - mln::util::soft_heap, [971](#)
- ~tracked_ptr
 - mln::util::tracked_ptr, [974](#)
- _1
 - mln::algebra::h_mat, [501](#)
- 1D neighborhoods, [109](#)
 - c2, [109](#)
 - neighb1d, [109](#)
- 1D windows, [123](#)
 - segment1d, [123](#)
 - window1d, [123](#)
- 2D neighborhoods, [110](#)
 - c2_col, [110](#)
 - c2_row, [110](#)
 - c4, [111](#)
 - c8, [111](#)
 - neighb2d, [110](#)
- 2D windows, [124](#)
 - disk2d, [124](#)
 - hline2d, [124](#)
 - vline2d, [125](#)
 - win_c4p, [125](#)
 - win_c8p, [125](#)
 - window2d, [125](#)
- 3D neighborhoods, [112](#)
 - c18, [112](#)
 - c26, [113](#)
 - c2_3d_sli, [113](#)
 - c4_3d, [113](#)
 - c6, [114](#)
 - c8_3d, [114](#)
 - neighb3d, [112](#)
- 3D windows, [127](#)
 - sline3d, [127](#)
 - sphere3d, [128](#)
 - win_c4p_3d, [128](#)
 - win_c8p_3d, [128](#)
 - window3d, [128](#)
- a_point_of
 - mln, [155](#)
- abs
 - mln::data, [204](#)
 - mln::math, [330](#)
- abs_inplace
 - mln::data, [204](#)
- Accumulators, [104](#)
- add
 - mln::topo::n_faces_set, [918](#)
- add_child
 - mln::util::tree_node, [978](#)
- add_edge
 - mln::util::graph, [948](#)
- add_face
 - mln::topo::complex, [900](#)
- add_location
 - mln::geom::complex_geometry, [664](#)
- add_tree_down
 - mln::util::tree, [976](#)
- add_tree_up
 - mln::util::tree, [976](#)
- add_vertex
 - mln::util::graph, [948](#)
- add_vertices
 - mln::util::graph, [949](#)
- addr
 - mln::topo::complex, [901](#)
- adj_higher_face_bkd_iter
 - mln::topo::adj_higher_face_bkd_iter, [878](#)
- adj_higher_face_fwd_iter
 - mln::topo::adj_higher_face_fwd_iter, [879](#)
- adj_lower_face_bkd_iter
 - mln::topo::adj_lower_face_bkd_iter, [882](#)
- adj_lower_face_fwd_iter
 - mln::topo::adj_lower_face_fwd_iter, [883](#)
- adj_lower_higher_face_bkd_iter
 - mln::topo::adj_lower_higher_face_bkd_iter, [884](#)
- adj_lower_higher_face_fwd_iter
 - mln::topo::adj_lower_higher_face_fwd_iter, [885](#)
- adj_m_face_bkd_iter
 - mln::topo::adj_m_face_bkd_iter, [886](#)
- adj_m_face_fwd_iter
 - mln::topo::adj_m_face_fwd_iter, [887](#)
- adjacency_matrix
 - mln::util::adjacency_matrix, [927](#)
- adjust
 - mln::border, [190](#)
 - mln::extension, [233](#)
- adjust_duplicate
 - mln::extension, [233](#)
- adjust_fill
 - mln::extension, [233](#)
- algebraic_face
 - mln::topo::algebraic_face, [889](#), [890](#)
- algebraic_n_face
 - mln::topo::algebraic_n_face, [894](#)

- and_inplace
 - mln::logical, [306](#)
- and_not
 - mln::logical, [306](#)
- and_not_inplace
 - mln::logical, [306](#)
- antialiased
 - mln::subsampling, [365](#)
- apex
 - mln::util::branch, [934](#)
- append
 - mln::p_array, [777](#)
 - mln::util::array, [931](#)
- apply
 - mln::data, [204](#)
- apply_p2p
 - mln, [156](#)
- area
 - mln::accu::site_set::rectangularity, [471](#)
 - mln::morpho::attribute::sharpness, [767](#)
 - mln::morpho::attribute::volume, [770](#)
 - mln::win::octagon2d, [1036](#)
 - mln::win::rectangle2d, [1037](#)
- argument
 - mln::accu::shape::height, [467](#)
 - mln::accu::shape::volume, [469](#)
 - mln::doc::Accumulator, [550](#)
- array
 - mln::util::array, [930](#)
- array< T >, [399](#)
- at
 - mln::opt, [358](#)
- attachment
 - mln::make, [312](#)
- backdiag2d
 - mln::win::backdiag2d, [1028](#)
- background
 - mln::labeling, [281](#)
- ball
 - mln::win::ball, [1029](#)
- base_level
 - mln::morpho::attribute::height, [765](#)
- Basic types, [98](#), [116](#)
- bbox
 - mln::accu::site_set::rectangularity, [471](#)
 - mln::Box, [516](#)
 - mln::box, [511](#)
 - mln::doc::Box, [552](#)
 - mln::doc::Fastest_Image, [559](#)
 - mln::doc::Image, [567](#)
 - mln::geom, [245](#)
 - mln::image1d, [702](#)
 - mln::image2d, [706](#)
 - mln::image3d, [713](#)
 - mln::labeled_image, [723](#)
 - mln::labeled_image_base, [725](#)
 - mln::p_line2d, [809](#)
 - mln::p_run, [830](#)
- bbox_t
 - mln::labeled_image, [722](#)
 - mln::labeled_image_base, [725](#)
- bboxes
 - mln::labeled_image, [723](#)
 - mln::labeled_image_base, [726](#)
- before
 - mln, [167](#)
- begin
 - mln::p_line2d, [809](#)
- bin_1complex_image2d
 - mln, [151](#)
- bin_2complex_image3df
 - mln, [152](#)
- binarization
 - mln::binarization, [189](#)
- bkd_citer
 - mln::topo::complex, [900](#)
- bkd_eiter
 - mln::util::array, [930](#)
 - mln::util::set, [965](#)
- bkd_niter
 - mln::doc::Neighborhood, [571](#)
 - mln::graph_elt_mixed_neighborhood, [672](#)
 - mln::graph_elt_neighborhood, [677](#)
 - mln::graph_elt_neighborhood_if, [679](#)
 - mln::neighb, [771](#)
- bkd_piter
 - mln::box, [510](#)
 - mln::doc::Box, [552](#)
 - mln::doc::Fastest_Image, [557](#)
 - mln::doc::Image, [565](#)
 - mln::doc::Site_Set, [580](#)
 - mln::hexa, [696](#)
 - mln::image2d_h, [709](#)
 - mln::p_array, [776](#)
 - mln::p_centered, [780](#)
 - mln::p_complex, [783](#)
 - mln::p_edges, [786](#)
 - mln::p_faces, [791](#)
 - mln::p_if, [795](#)
 - mln::p_image, [798](#)
 - mln::p_key, [804](#)
 - mln::p_line2d, [808](#)
 - mln::p_mutable_array_of, [811](#)
 - mln::p_priority, [817](#)
 - mln::p_queue, [822](#)
 - mln::p_queue_fast, [825](#)
 - mln::p_run, [829](#)
 - mln::p_set, [833](#)
 - mln::p_transformed, [836](#)
 - mln::p_vaccess, [841](#)
 - mln::p_vertices, [844](#)
- bkd_pixter1d
 - mln::bkd_pixter1d, [504](#)
- bkd_pixter2d
 - mln::bkd_pixter2d, [505](#)
- bkd_pixter3d

- mln::bkd_pixter3d, 506
- bkd_qiter
 - mln::doc::Weighted_Window, 585
 - mln::doc::Window, 587
 - mln::graph_elt_mixed_window, 674
 - mln::graph_elt_window, 681
 - mln::graph_elt_window_if, 686
 - mln::w_window, 1024
 - mln::window, 1040
- bkd_viter
 - mln::doc::Value_Set, 583
 - mln::value::lut_vec, 1006
- black
 - mln::literal, 302
- blobs
 - mln::canvas::labeling, 196
 - mln::labeling, 281
- blobs_and_compute
 - mln::labeling, 282
- blue
 - mln::literal, 302
- border
 - mln::doc::Fastest_Image, 559
 - mln::image1d, 702
 - mln::image2d, 706
 - mln::image3d, 713
- box
 - mln::box, 511
 - mln::draw, 228
- box1d
 - mln, 152
 - mln::make, 312
- box2d
 - mln, 152
 - mln::make, 313
- box2d_h
 - mln, 152
 - mln::make, 314
- box3d
 - mln, 152
 - mln::make, 314, 315
- box_plain
 - mln::draw, 228
- box_runstart_piter
 - mln::box_runstart_piter, 520
- branch
 - mln::util::branch, 934
- brown
 - mln::literal, 302
- buffer
 - mln::doc::Fastest_Image, 559
 - mln::image1d, 702
 - mln::image2d, 706
 - mln::image3d, 713
- c18
 - 3D neighborhoods, 112
- c2
 - 1D neighborhoods, 109
- c26
 - 3D neighborhoods, 113
- c2_3d_sli
 - 3D neighborhoods, 113
- c2_col
 - 2D neighborhoods, 110
- c2_row
 - 2D neighborhoods, 110
- c4
 - 2D neighborhoods, 111
- c4_3d
 - 3D neighborhoods, 113
- c6
 - 3D neighborhoods, 114
- c8
 - 2D neighborhoods, 111
- c8_3d
 - 3D neighborhoods, 114
- C_Function< E >, 399
- can_stop
 - mln::accu::logic::land_basic, 415
 - mln::accu::logic::lor_basic, 418
- Canvas, 106
- card
 - mln::set, 362
- cast
 - mln::value, 391
- Category
 - mln::util::vertex, 983
- category
 - mln::util::edge, 940
- cell
 - mln::make, 315
- center
 - mln::p_centered, 781
- center_only_iter
 - mln::topo::center_only_iter, 897
- center_t
 - mln::graph_elt_mixed_window, 674
 - mln::graph_elt_window, 681
 - mln::graph_window_piter, 692
- center_val
 - mln::dpoints_bkd_pixter, 594
 - mln::dpoints_fwd_pixter, 596
- centered_bkd_iter_adapter
 - mln::topo::centered_bkd_iter_adapter, 898
- centered_fwd_iter_adapter
 - mln::topo::centered_fwd_iter_adapter, 899
- chamfer
 - mln::geom, 245
- change
 - mln::p_array, 777
- change_both
 - mln::util::couple, 938
 - mln::util::ord_pair, 961
- change_extension
 - mln::extension_val, 608
- change_first

- mln::util::couple, 938
- mln::util::ord_pair, 961
- change_graph
 - mln::util::edge, 941
 - mln::util::vertex, 983
- change_key
 - mln::p_key, 805
- change_keys
 - mln::p_key, 805
- change_mask
 - mln::graph_elt_window_if, 687
- change_second
 - mln::util::couple, 938
 - mln::util::ord_pair, 961
- change_target
 - mln::complex_psite, 542
 - mln::p_transformed_piter, 839
- change_target_site_set
 - mln::graph_window_piter, 694
- change_to
 - mln::pixel, 849
- check_consistency
 - mln::util::tree, 976
 - mln::util::tree_node, 979
- children
 - mln::util::tree_node, 979
- clear
 - mln::p_array, 778
 - mln::p_image, 799
 - mln::p_key, 805
 - mln::p_mutable_array_of, 812
 - mln::p_priority, 818
 - mln::p_queue, 822
 - mln::p_queue_fast, 826
 - mln::p_set, 834
 - mln::util::array, 931
 - mln::util::fibonacci_heap, 944
 - mln::util::set, 966
 - mln::util::soft_heap, 972
 - mln::w_window, 1025
 - mln::window, 1040
- closing
 - mln::morpho::elementary, 341
- colorize
 - mln::labeling, 282, 283
- complementation
 - mln::morpho, 334
- complementation_inplace
 - mln::morpho, 334
- complex
 - mln::topo::complex, 900
- Complex based, 118
- complex_geometry
 - mln::geom::complex_geometry, 664
- complex_image
 - mln::complex_image, 537
- complex_neighborhood_bkd_piter
 - mln::complex_neighborhood_bkd_piter, 539
- complex_neighborhood_fwd_piter
 - mln::complex_neighborhood_fwd_piter, 541
- complex_psite
 - mln::complex_psite, 542
- complex_window_bkd_piter
 - mln::complex_window_bkd_piter, 545
- complex_window_fwd_piter
 - mln::complex_window_fwd_piter, 546
- compose
 - mln, 156
- composed
 - mln::fun::x2x::composed, 644
- compute
 - mln::accu, 169
 - mln::data, 204, 205
 - mln::graph, 253
 - mln::histo, 256
 - mln::labeling, 283, 284
 - mln::labeling::impl::generic, 294, 295
 - mln::set, 362
- compute_attribute_image
 - mln::morpho::tree, 346
- compute_attribute_image_from
 - mln::morpho::tree, 346
- compute_fastest
 - mln::labeling::impl, 293
- compute_has
 - mln::p_queue_fast, 826
- compute_image
 - mln::labeling, 285
- compute_in_window
 - mln::labeling, 286
- compute_parent
 - mln::morpho::tree, 347
- compute_with_weights
 - mln::set, 363
- contrast
 - mln::morpho, 334
- convert
 - mln::data, 205
- convolve
 - mln::linear::local, 299
- coord
 - mln::def, 225
 - mln::doc::Dpoint, 554
 - mln::doc::Fastest_Image, 557
 - mln::doc::Image, 565
 - mln::doc::Point_Site, 575
 - mln::dpoint, 589
 - mln::point, 856
- coordf
 - mln::def, 225
- count
 - mln::accu::stat::mean, 478
- couple
 - mln::make, 316
- cplx
 - mln::p_complex, 784

- mln::p_faces, 792
 - mln::topo::algebraic_face, 890
 - mln::topo::algebraic_n_face, 894
 - mln::topo::face, 904
 - mln::topo::n_face, 913
- crop_wrt
 - mln::box, 511
- cube3d
 - mln::win::cube3d, 1030
- cuboid3d
 - mln::win::cuboid3d, 1031
- cyan
 - mln::literal, 302
- D
 - mln::topo::is_simple_cell, 912
- dark_gray
 - mln::literal, 302
- dashed_line
 - mln::draw, 229
- data
 - mln::topo::algebraic_face, 890
 - mln::topo::algebraic_n_face, 894
 - mln::topo::face, 904
 - mln::topo::n_face, 913
- data< I >, 399
- data_t
 - mln::fun::x2x::rotation, 648
 - mln::fun::x2x::translation, 651
- dec_face_id
 - mln::topo::algebraic_face, 890
 - mln::topo::algebraic_n_face, 894
 - mln::topo::face, 904
 - mln::topo::n_face, 913
- dec_n
 - mln::topo::algebraic_face, 890
 - mln::topo::face, 904
- decorated_image
 - mln::decorated_image, 548
- decoration
 - mln::decorated_image, 548
- deepness
 - mln::util::branch_iter, 935
 - mln::util::branch_iter_ind, 936
- delete_tree_node
 - mln::util::tree_node, 979
- delta
 - mln::doc::Weighted_Window, 586
 - mln::geom, 245, 246
 - mln::graph_elt_mixed_window, 675
 - mln::graph_elt_window, 682
 - mln::graph_elt_window_if, 687
 - mln::graph_window_base, 689
 - mln::point, 856
 - mln::window, 1040
- delta_index
 - mln::doc::Fastest_Image, 559
 - mln::image1d, 702
 - mln::image2d, 706
 - mln::image3d, 713
- depth
 - mln::win::cuboid3d, 1031
- depth1st_piter< T >, 400
- det
 - mln::algebra, 176
- detach
 - mln::topo, 371
- detachment
 - mln::make, 316
- diag2d
 - mln::win::diag2d, 1032
- diameter
 - mln::win::ball, 1029
- diff
 - mln::Box, 516
 - mln::Site_Set, 869
 - mln::win, 397
- diff_abs
 - mln::arith, 179
- dilation
 - mln::morpho, 334
- dim
 - mln::complex_image, 538
 - mln::doc::Dpoint, 555
 - mln::doc::Point_Site, 576
 - mln::dpoint, 590
 - mln::point, 856
- direct
 - mln::morpho::tree::filter, 351
- discrete_plane_1complex_geometry
 - mln, 152
- discrete_plane_2complex_geometry
 - mln, 152
- disk2d
 - 2D windows, 124
- display_branch
 - mln::util, 383
- display_tree
 - mln::util, 383
- distance_and_closest_point_geodesic
 - mln::transform, 377
- distance_and_influence_zone_geodesic
 - mln::transform, 378
- distance_front
 - mln::canvas, 194
 - mln::transform, 378
- distance_geodesic
 - mln::canvas, 194
 - mln::transform, 379
- div
 - mln::arith, 179
- div_cst
 - mln::arith, 179
- div_inplace
 - mln::arith, 180
- dn_leaf_piter< T >, 400
- dn_node_piter< T >, 401

- dn_site_piter< T >, [401](#)
- domain
 - mln::complex_image, [537](#)
 - mln::doc::Fastest_Image, [559](#)
 - mln::doc::Image, [567](#)
 - mln::extended, [603](#)
 - mln::flat_image, [610](#)
 - mln::hexa, [697](#)
 - mln::image1d, [702](#)
 - mln::image2d, [706](#)
 - mln::image2d_h, [710](#)
 - mln::image3d, [713](#)
 - mln::lazy_image, [728](#)
 - mln::p2p_image, [774](#)
 - mln::sub_image, [872](#)
 - mln::sub_image_if, [873](#)
 - mln::tr_image, [923](#)
 - mln::transformed_image, [925](#)
 - mln::unproject_image, [926](#)
- Domain morphers, [101](#)
- domain_t
 - mln::value::stack_image, [1016](#)
- dp
 - mln::window, [1041](#)
- dpoint
 - mln::doc::Dpoint, [554](#)
 - mln::doc::Fastest_Image, [557](#)
 - mln::doc::Image, [566](#)
 - mln::doc::Neighborhood, [571](#)
 - mln::doc::Point_Site, [575](#)
 - mln::doc::Weighted_Window, [585](#)
 - mln::dpoint, [590](#)
- dpoint1d
 - mln, [153](#)
- dpoint2d
 - mln, [153](#)
- dpoint2d_h
 - mln, [153](#)
 - mln::make, [316](#)
- dpoint3d
 - mln, [153](#)
- dpoints_bkd_pixter
 - mln::dpoints_bkd_pixter, [593](#)
- dpoints_fwd_pixter
 - mln::dpoints_fwd_pixter, [595](#)
- dpsite
 - mln::point, [856](#)
 - mln::w_window, [1024](#)
- dpsites_bkd_piter
 - mln::dpsites_bkd_piter, [597](#)
- dpsites_fwd_piter
 - mln::dpsites_fwd_piter, [598](#)
- draw_graph
 - mln::debug, [221](#), [222](#)
- dual_input_max_tree
 - mln::morpho::tree, [348](#)
- dummy_p_edges
 - mln::make, [317](#)
- dummy_p_vertices
 - mln::make, [317](#)
- duplicate
 - mln, [156](#)
 - mln::border, [191](#)
 - mln::extension, [234](#)
- e_ith_nbh_edge
 - mln::util::graph, [949](#)
 - mln::util::line_graph, [956](#)
- e_nmax
 - mln::util::graph, [949](#)
 - mln::util::line_graph, [956](#)
- edge
 - mln::p_edges, [786](#)
 - mln::topo, [371](#)
 - mln::util::edge, [941](#)
 - mln::util::graph, [949](#)
 - mln::util::line_graph, [956](#)
- edge_fwd_iter
 - mln::util::graph, [947](#)
 - mln::util::line_graph, [955](#)
- edge_image
 - mln::edge_image, [601](#)
 - mln::make, [318](#), [319](#)
- edge_nbh_edge_fwd_iter
 - mln::util::graph, [947](#)
 - mln::util::line_graph, [955](#)
- edge_nbh_t
 - mln::edge_image, [600](#)
- edge_win_t
 - mln::edge_image, [600](#)
- edge_with
 - mln::util::vertex, [983](#)
- edges
 - mln::util::graph, [949](#)
- edges_set_t
 - mln::util::graph, [947](#)
- edges_t
 - mln::util::graph, [947](#)
 - mln::util::line_graph, [955](#)
- eiter
 - mln::util::array, [930](#)
 - mln::util::set, [965](#)
- element
 - mln::box, [510](#)
 - mln::graph_window_if_piter, [691](#)
 - mln::graph_window_piter, [694](#)
 - mln::image1d, [702](#)
 - mln::image2d, [706](#)
 - mln::image3d, [714](#)
 - mln::p_array, [776](#)
 - mln::p_centered, [780](#)
 - mln::p_complex, [783](#)
 - mln::p_edges, [786](#)
 - mln::p_faces, [791](#)
 - mln::p_if, [795](#)
 - mln::p_image, [798](#)
 - mln::p_key, [804](#)

- mln::p_line2d, [808](#)
- mln::p_mutable_array_of, [811](#)
- mln::p_priority, [817](#)
- mln::p_queue, [822](#)
- mln::p_queue_fast, [825](#)
- mln::p_run, [829](#)
- mln::p_set, [833](#)
- mln::p_transformed, [836](#)
- mln::p_vaccess, [841](#)
- mln::p_vertices, [844](#)
- mln::util::array, [930](#)
- mln::util::set, [966](#)
- mln::util::soft_heap, [971](#)
- elt
 - mln::util::tree_node, [979](#), [980](#)
- empty
 - mln::p_queue_fast, [826](#)
- enc
 - mln::value::float01, [988](#)
 - mln::value::label, [1003](#)
 - mln::value::proxy, [1008](#)
 - mln::value::sign, [1014](#)
- end
 - mln::p_line2d, [809](#)
 - mln::p_run, [830](#)
- enlarge
 - mln::box, [511](#)
- equalize
 - mln::border, [191](#)
 - mln::histo, [256](#)
- equiv
 - mln::value, [391](#)
 - mln::value::float01, [988](#)
 - mln::value::proxy, [1008](#)
 - mln::value::sign, [1014](#)
- erosion
 - mln::morpho, [334](#)
- erosion_tolerant
 - mln::morpho, [334](#)
- exists_key
 - mln::p_key, [805](#)
- exists_priority
 - mln::p_priority, [818](#)
- extend
 - mln, [157](#)
- extended
 - mln::extended, [602](#)
- extension
 - mln::extension_fun, [604](#)
 - mln::extension_ima, [606](#)
 - mln::extension_val, [608](#)
- extension_fun
 - mln::extension_fun, [604](#)
- extension_ima
 - mln::extension_ima, [606](#)
- extension_val
 - mln::extension_val, [608](#)
- f_hsi_to_rgb_3x8
 - mln::fun::v2v, [238](#)
- f_hsl_to_rgb_3x8
 - mln::fun::v2v, [238](#)
- f_rgb_to_hsi_f
 - mln::fun::v2v, [238](#)
- f_rgb_to_hsl_f
 - mln::fun::v2v, [239](#)
- face
 - mln::complex_psite, [543](#)
 - mln::topo::face, [903](#)
- face_bkd_iter
 - mln::topo::face_bkd_iter, [906](#)
- face_data< N, D >, [401](#)
- face_fwd_iter
 - mln::topo::face_fwd_iter, [907](#)
- face_id
 - mln::complex_psite, [543](#)
 - mln::topo::algebraic_face, [890](#)
 - mln::topo::algebraic_n_face, [894](#)
 - mln::topo::face, [904](#)
 - mln::topo::n_face, [914](#)
- faces
 - mln::topo::n_faces_set, [918](#)
- faces_set_mixin< N, D >, [401](#)
- faces_type
 - mln::topo::n_faces_set, [918](#)
- fast_median
 - mln::data, [205](#)
- fibonacci_heap
 - mln::util::fibonacci_heap, [944](#)
- filename
 - mln::debug, [222](#)
- fill
 - mln::border, [191](#)
 - mln::data, [206](#)
 - mln::extension, [234](#)
 - mln::util::array, [931](#)
- fill_holes
 - mln::labeling, [286](#)
- fill_with_image
 - mln::data, [206](#)
 - mln::data::impl::generic, [217](#)
- fill_with_value
 - mln::data, [206](#)
 - mln::data::impl::generic, [217](#)
- filter
 - mln::morpho::tree::filter, [351](#)
- find
 - mln::border, [192](#)
- first
 - mln::accu::pair, [463](#)
 - mln::accu::stat::min_max, [487](#)
 - mln::util::couple, [938](#)
 - mln::util::ord_pair, [962](#)
 - mln::util::site_pair, [969](#)
- first_accu
 - mln::accu::pair, [463](#)
 - mln::accu::stat::min_max, [487](#)

- first_element
 - mln::util::set, 966
- flat_image
 - mln::flat_image, 610
- flat_zones
 - mln::labeling, 286
- float01
 - mln::value::float01, 988
- float01_16
 - mln::value, 389
- float01_8
 - mln::value, 389
- float01_f
 - mln::value::float01_f, 989, 990
- float_2complex_image3df
 - mln, 153
- flooding
 - mln::morpho::watershed, 353, 354
- foreground
 - mln::labeling, 287
- format
 - mln::debug, 222
- from_to
 - mln::convert, 198, 199
- front
 - mln::p_priority, 818
 - mln::p_queue, 822
 - mln::p_queue_fast, 826
 - mln::util::fibonacci_heap, 944
- fun
 - mln::p2p_image, 774
- fun_image
 - mln::fun_image, 653
- fun_t
 - mln::p_edges, 787
 - mln::p_vertices, 844
- Function
 - mln::Function, 654
- function
 - mln::p_edges, 788
 - mln::p_transformed, 837
 - mln::p_vertices, 846
- Functions, 107
- fwd_citer
 - mln::topo::complex, 900
- fwd_eiter
 - mln::util::array, 930
 - mln::util::set, 966
- fwd_niter
 - mln::doc::Neighborhood, 571
 - mln::graph_elt_mixed_neighborhood, 672
 - mln::graph_elt_neighborhood, 677
 - mln::graph_elt_neighborhood_if, 679
 - mln::neighb, 771
- fwd_piter
 - mln::box, 510
 - mln::doc::Box, 552
 - mln::doc::Fastest_Image, 557
 - mln::doc::Image, 566
 - mln::doc::Site_Set, 580
 - mln::hexa, 696
 - mln::image2d_h, 709
 - mln::p_array, 776
 - mln::p_centered, 780
 - mln::p_complex, 783
 - mln::p_edges, 787
 - mln::p_faces, 791
 - mln::p_if, 795
 - mln::p_image, 798
 - mln::p_key, 804
 - mln::p_line2d, 808
 - mln::p_mutable_array_of, 811
 - mln::p_priority, 817
 - mln::p_queue, 822
 - mln::p_queue_fast, 826
 - mln::p_run, 830
 - mln::p_set, 833
 - mln::p_transformed, 837
 - mln::p_vaccess, 841
 - mln::p_vertices, 845
- fwd_pixter1d
 - mln::fwd_pixter1d, 659
- fwd_pixter2d
 - mln::fwd_pixter2d, 660
- fwd_pixter3d
 - mln::fwd_pixter3d, 661
- fwd_qiter
 - mln::doc::Weighted_Window, 585
 - mln::doc::Window, 587
 - mln::graph_elt_mixed_window, 674
 - mln::graph_elt_window, 681
 - mln::graph_elt_window_if, 686
 - mln::w_window, 1024
 - mln::window, 1040
- fwd_viter
 - mln::doc::Value_Set, 583
 - mln::value::lut_vec, 1006
- gaussian
 - mln::linear, 296
- gaussian_1st_derivative
 - mln::linear, 297
- gaussian_2nd_derivative
 - mln::linear, 297
- gaussian_subsampling
 - mln::subsampling, 365
- general
 - mln::morpho, 335
- geom
 - mln::complex_image, 537
 - mln::p_complex, 784
- get
 - mln::border, 192
 - mln::set, 363
- get_header
 - mln::io::dicom, 259
 - mln::io::dump, 260

- mln::io::raw, 276
- get_rot
 - mln::registration, 360
- gl16
 - mln::value, 389
- gl8
 - mln::value, 390
- glf
 - mln::value, 390
- gradient
 - mln::morpho, 335
- gradient_external
 - mln::morpho, 335
- gradient_internal
 - mln::morpho, 335
- graph
 - mln::p_edges, 788
 - mln::p_graph_piter, 793
 - mln::p_vertices, 847
 - mln::util::edge, 941
 - mln::util::graph, 948
 - mln::util::line_graph, 956
 - mln::util::vertex, 983
- Graph based, 117
- graph_element
 - mln::graph_elt_mixed_window, 675
 - mln::graph_elt_window, 682
 - mln::graph_window_piter, 693
 - mln::p_edges, 787
 - mln::p_vertices, 845
- graph_elt_mixed_window< G, S, S2 >, 402
- graph_elt_neighborhood_if
 - mln::graph_elt_neighborhood_if, 679
- graph_elt_window< G, S >, 402
- graph_elt_window_if
 - mln::graph_elt_window_if, 687
- graph_elt_window_if< G, S, I >, 402
- graph_mixed_window_iter_dispatch< G, S, S2 >, 403
- graph_t
 - mln::edge_image, 600
 - mln::p_edges, 787
 - mln::p_vertices, 845
 - mln::util::edge, 940
 - mln::util::vertex, 983
 - mln::vertex_image, 1020
- graph_window_if_iter_dispatch< G, S >, 403
- graph_window_if_piter
 - mln::graph_window_if_piter, 691
- graph_window_iter_dispatch< G, S >, 403
- graph_window_piter
 - mln::graph_window_piter, 693
- Graphes, 96
- graylevel
 - mln::value::graylevel, 992
- graylevel_f
 - mln::value::graylevel_f, 994
- green
 - mln::literal, 303
- grid
 - mln::dpoint, 589
 - mln::point, 856
- h_mat
 - mln::algebra::h_mat, 501
 - mln::make, 320
- h_vec
 - mln::algebra::h_vec, 503
 - mln::point, 856
- has
 - mln::box, 511
 - mln::doc::Box, 552
 - mln::doc::Fastest_Image, 560
 - mln::doc::Image, 568
 - mln::doc::Site_Set, 580
 - mln::doc::Value_Set, 584
 - mln::extension_fun, 604
 - mln::extension_ima, 606
 - mln::extension_val, 608
 - mln::flat_image, 611
 - mln::hexa, 697
 - mln::image1d, 703
 - mln::image2d, 706
 - mln::image2d_h, 710
 - mln::image3d, 714
 - mln::interpolated, 716
 - mln::lazy_image, 728
 - mln::p_array, 778
 - mln::p_centered, 781
 - mln::p_complex, 784
 - mln::p_edges, 789
 - mln::p_if, 796
 - mln::p_image, 799
 - mln::p_key, 805
 - mln::p_line2d, 809
 - mln::p_mutable_array_of, 812
 - mln::p_priority, 818
 - mln::p_queue, 823
 - mln::p_queue_fast, 827
 - mln::p_run, 831
 - mln::p_set, 834
 - mln::p_transformed, 837
 - mln::p_vaccess, 842
 - mln::p_vertices, 847
 - mln::set, 364
 - mln::tr_image, 923
 - mln::util::line_graph, 956
 - mln::util::set, 966
 - mln::value::lut_vec, 1006
 - mln::window, 1041
- has_e
 - mln::util::graph, 949
 - mln::util::line_graph, 956
- has_index
 - mln::p_run, 831
- has_v
 - mln::util::graph, 950
 - mln::util::line_graph, 956

- height
 - mln::morpho::attribute::sharpness, 767
 - mln::win::cuboid3d, 1031
 - mln::win::rectangle2d, 1037
- hexa
 - mln::hexa, 697
- higher_dim_adj_faces
 - mln::topo::algebraic_face, 891
 - mln::topo::algebraic_n_face, 894
 - mln::topo::face, 904
 - mln::topo::n_face, 914
- highest_priority
 - mln::p_priority, 818
- histo3d_rgb
 - mln::accu::stat::histo3d_rgb, 474
- hit_or_miss
 - mln::morpho, 335
- hit_or_miss_background_closing
 - mln::morpho, 335
- hit_or_miss_background_opening
 - mln::morpho, 336
- hit_or_miss_closing
 - mln::morpho, 336
- hit_or_miss_opening
 - mln::morpho, 336
- hline2d
 - 2D windows, 124
- hough
 - mln::transform, 379
- i_element
 - mln::p_array, 777
 - mln::p_image, 798
 - mln::p_key, 804
 - mln::p_mutable_array_of, 811
 - mln::p_priority, 817
 - mln::p_queue, 822
 - mln::p_queue_fast, 826
 - mln::p_set, 833
 - mln::p_vaccess, 841
- icp
 - mln::registration, 360, 361
- id
 - mln::graph_window_if_piter, 691
 - mln::graph_window_piter, 694
 - mln::p_graph_piter, 793
 - mln::util::edge, 941
 - mln::util::vertex, 983
- id_t
 - mln::util::edge, 941
 - mln::util::vertex, 983
- id_value_t
 - mln::util::edge, 941
 - mln::util::vertex, 983
- identity
 - mln::literal, 303
- Identity morphers, 102
- ima
 - mln::doc::Generalized_Pixel, 563
- mln::doc::Pixel_iterator, 574
- mln::fun::x2x::linear, 646
- mln::util::pix, 963
- image
 - mln::bkd_pixter1d, 504
 - mln::bkd_pixter2d, 505
 - mln::bkd_pixter3d, 506
 - mln::doc::Generalized_Pixel, 563
 - mln::doc::Pixel_iterator, 574
 - mln::fwd_pixter1d, 659
 - mln::fwd_pixter2d, 660
 - mln::fwd_pixter3d, 661
 - mln::make, 320
 - mln::pw::image, 861
- Image morphers, 99
- image1d
 - mln::image1d, 701, 702
- image2d
 - mln::image2d, 705
 - mln::make, 321
- image2d_h
 - mln::image2d_h, 710
- image3d
 - mln::image3d, 713
 - mln::make, 321
- Images, 97
- implies
 - mln, 157
- inc_face_id
 - mln::topo::algebraic_face, 891
 - mln::topo::algebraic_n_face, 895
 - mln::topo::face, 904
 - mln::topo::n_face, 914
- inc_n
 - mln::topo::algebraic_face, 891
 - mln::topo::face, 904
- index
 - mln::p_indexed_bkd_piter, 800
 - mln::p_indexed_fwd_piter, 802
- index_of
 - mln::doc::Value_Set, 584
 - mln::value::lut_vec, 1006
- influence_zone_adjacency_graph
 - mln::make, 321
- influence_zone_front
 - mln::transform, 379
- influence_zone_geodesic
 - mln::transform, 379
- influence_zone_geodesic_saturated
 - mln::transform, 380
- init
 - mln::accu::center, 405
 - mln::accu::convolve, 406
 - mln::accu::count_adjacent_vertices, 407
 - mln::accu::count_labels, 409
 - mln::accu::count_value, 410
 - mln::accu::label_used, 413
 - mln::accu::logic::land, 414

- mln::accu::logic::land_basic, 415
- mln::accu::logic::lor, 416
- mln::accu::logic::lor_basic, 418
- mln::accu::maj_h, 419
- mln::accu::math::count, 420
- mln::accu::math::inf, 421
- mln::accu::math::sum, 423
- mln::accu::math::sup, 424
- mln::accu::max_site, 425
- mln::accu::nil, 460
- mln::accu::p, 461
- mln::accu::pair, 463
- mln::accu::rms, 465
- mln::accu::shape::bbox, 466
- mln::accu::shape::height, 467
- mln::accu::shape::volume, 469
- mln::accu::stat::deviation, 472
- mln::accu::stat::histo3d_rgb, 474
- mln::accu::stat::max, 476
- mln::accu::stat::max_h, 477
- mln::accu::stat::mean, 478
- mln::accu::stat::median_h, 482
- mln::accu::stat::min, 484
- mln::accu::stat::min_h, 485
- mln::accu::stat::min_max, 487
- mln::accu::stat::rank, 489
- mln::accu::stat::rank< bool >, 490
- mln::accu::stat::rank_high_quant, 491
- mln::accu::stat::var, 493
- mln::accu::stat::variance, 495
- mln::accu::tuple, 496
- mln::accu::val, 498
- mln::doc::Accumulator, 550
- mln::morpho::attribute::card, 763
- mln::morpho::attribute::count_adjacent_vertices, 764
- mln::morpho::attribute::height, 765
- mln::morpho::attribute::sharpness, 767
- mln::morpho::attribute::sum, 768
- mln::morpho::attribute::volume, 770
- mln::p_run, 831
- initialize
 - mln, 157
- insert
 - mln::p_array, 778
 - mln::p_image, 799
 - mln::p_key, 805
 - mln::p_mutable_array_of, 812
 - mln::p_priority, 818
 - mln::p_queue, 823
 - mln::p_queue_fast, 827
 - mln::p_set, 834
 - mln::p_vaccess, 842
 - mln::util::set, 967
 - mln::w_window, 1025
 - mln::window, 1041
- int_s
 - mln::value::int_s, 997
- int_s< n >, 403
- int_s16
 - mln::value, 390
- int_s32
 - mln::value, 390
- int_s8
 - mln::value, 390
- int_u
 - mln::value::int_u, 999
- int_u12
 - mln::value, 390
- int_u16
 - mln::value, 390
- int_u32
 - mln::value, 390
- int_u8
 - mln::value, 390
- int_u8_1complex_image2d
 - mln, 153
- int_u8_2complex_image2d
 - mln, 153
- int_u8_2complex_image3df
 - mln, 153
- int_u_sat
 - mln::value::int_u_sat, 1001
- inter
 - mln::Box, 516
 - mln::Site_Set, 869
- interpolated
 - mln::interpolated, 716
- inv
 - mln::fun::x2x::rotation, 649
 - mln::fun::x2x::translation, 651
- invalidate
 - mln::complex_psite, 543
 - mln::doc::Iterator, 570
 - mln::doc::Pixel_Iterator, 574
 - mln::doc::Site_Iterator, 578
 - mln::doc::Value_Iterator, 582
 - mln::dpoints_bkd_pixter, 594
 - mln::dpoints_fwd_pixter, 596
 - mln::p_edges, 789
 - mln::p_vertices, 847
 - mln::topo::algebraic_face, 891
 - mln::topo::algebraic_n_face, 895
 - mln::topo::face, 904
 - mln::topo::n_face, 914
 - mln::util::branch_iter, 935
 - mln::util::branch_iter_ind, 936
 - mln::util::edge, 941
 - mln::util::vertex, 984
- invert
 - mln::fun::x2x::rotation, 648
 - mln::fun::x2x::translation, 651
- iota
 - mln::debug, 223
- is_centered
 - mln::doc::Weighted_Window, 586

- mln::graph_elt_mixed_window, 675
- mln::graph_elt_window, 682
- mln::graph_elt_window_if, 687
- mln::graph_window_base, 689
- mln::window, 1041
- is_empty
 - mln::Box, 516
 - mln::box, 512
 - mln::doc::Weighted_Window, 586
 - mln::graph_elt_mixed_window, 675
 - mln::graph_elt_window, 682
 - mln::graph_elt_window_if, 687
 - mln::graph_window_base, 689
 - mln::util::array, 931
 - mln::util::fibonacci_heap, 944
 - mln::util::set, 967
 - mln::util::soft_heap, 972
 - mln::window, 1041
- is_facet
 - mln::topo, 371
- is_subgraph_of
 - mln::util::graph, 950
 - mln::util::line_graph, 957
- is_symmetric
 - mln::graph_elt_mixed_window, 676
 - mln::graph_elt_window, 683
 - mln::graph_elt_window_if, 687
 - mln::graph_window_base, 689
 - mln::w_window, 1025
 - mln::window, 1041
- is_valid
 - mln::accu::center, 405
 - mln::accu::convolve, 406
 - mln::accu::count_adjacent_vertices, 407
 - mln::accu::count_labels, 409
 - mln::accu::count_value, 410
 - mln::accu::histo, 411
 - mln::accu::label_used, 413
 - mln::accu::logic::land, 414
 - mln::accu::logic::land_basic, 415
 - mln::accu::logic::lor, 416
 - mln::accu::logic::lor_basic, 418
 - mln::accu::maj_h, 419
 - mln::accu::math::count, 420
 - mln::accu::math::inf, 422
 - mln::accu::math::sum, 423
 - mln::accu::math::sup, 424
 - mln::accu::max_site, 425
 - mln::accu::nil, 460
 - mln::accu::p, 461
 - mln::accu::pair, 463
 - mln::accu::rms, 465
 - mln::accu::shape::bbox, 466
 - mln::accu::shape::height, 467
 - mln::accu::shape::volume, 469
 - mln::accu::stat::deviation, 472
 - mln::accu::stat::histo3d_rgb, 474
 - mln::accu::stat::max, 476
 - mln::accu::stat::max_h, 477
 - mln::accu::stat::mean, 478
 - mln::accu::stat::median_alt, 480
 - mln::accu::stat::median_h, 482
 - mln::accu::stat::min, 484
 - mln::accu::stat::min_h, 485
 - mln::accu::stat::min_max, 487
 - mln::accu::stat::rank, 489
 - mln::accu::stat::rank< bool >, 490
 - mln::accu::stat::rank_high_quant, 491
 - mln::accu::stat::var, 493
 - mln::accu::stat::variance, 495
 - mln::accu::tuple, 496
 - mln::accu::val, 498
 - mln::box, 512
 - mln::complex_psite, 543
 - mln::doc::Fastest_Image, 560
 - mln::doc::Image, 568
 - mln::doc::Iterator, 570
 - mln::doc::Pixel_Iterator, 574
 - mln::doc::Site_Iterator, 578
 - mln::doc::Value_Iterator, 582
 - mln::dpoints_bkd_pixter, 594
 - mln::dpoints_fwd_pixter, 596
 - mln::graph_elt_mixed_window, 676
 - mln::graph_elt_window, 683
 - mln::graph_elt_window_if, 687
 - mln::graph_window_base, 689
 - mln::interpolated, 717
 - mln::morpho::attribute::card, 763
 - mln::morpho::attribute::count_adjacent_vertices, 764
 - mln::morpho::attribute::height, 765
 - mln::morpho::attribute::sharpness, 767
 - mln::morpho::attribute::sum, 768
 - mln::morpho::attribute::volume, 770
 - mln::p_array, 778
 - mln::p_centered, 781
 - mln::p_complex, 784
 - mln::p_edges, 789
 - mln::p_faces, 792
 - mln::p_if, 796
 - mln::p_image, 799
 - mln::p_key, 806
 - mln::p_line2d, 809
 - mln::p_mutable_array_of, 812
 - mln::p_priority, 819
 - mln::p_queue, 823
 - mln::p_queue_fast, 827
 - mln::p_run, 831
 - mln::p_set, 835
 - mln::p_transformed, 837
 - mln::p_vaccess, 842
 - mln::p_vertices, 847
 - mln::pixel, 849
 - mln::topo::algebraic_face, 891
 - mln::topo::algebraic_n_face, 895
 - mln::topo::face, 905

- mln::topo::n_face, 914
- mln::tr_image, 923
- mln::util::branch_iter, 935
- mln::util::branch_iter_ind, 936
- mln::util::edge, 941
- mln::util::fibonacci_heap, 944
- mln::util::soft_heap, 972
- mln::util::vertex, 984
- mln::value::stack_image, 1017
- iter
 - mln::complex_neighborhood_bkd_piter, 539
 - mln::complex_neighborhood_fwd_piter, 541
 - mln::complex_window_bkd_piter, 545
 - mln::complex_window_fwd_piter, 546
- iter_type
 - mln::complex_neighborhood_bkd_piter, 539
 - mln::complex_neighborhood_fwd_piter, 540
 - mln::complex_window_bkd_piter, 544
 - mln::complex_window_fwd_piter, 546
- ith_nbh_edge
 - mln::util::edge, 942
 - mln::util::vertex, 984
- ith_nbh_vertex
 - mln::util::vertex, 984
- k
 - mln::accu::stat::rank, 489
- key
 - mln::p_key, 806
- keys
 - mln::p_key, 806
- l1
 - mln::norm, 356
- l1_distance
 - mln::norm, 356
- l2
 - mln::norm, 356
- l2_distance
 - mln::norm, 356
- label
 - mln::value::label, 1003, 1004
- label_16
 - mln::value, 391
- label_32
 - mln::value, 391
- label_8
 - mln::value, 391
- labeled_image
 - mln::labeled_image, 722, 723
- labeled_image_base
 - mln::labeled_image_base, 725
- labeling
 - mln::graph, 254
- laplacian
 - mln::morpho, 336
- larger_than
 - mln, 157
- last
 - mln::util::array, 931
- last_coord
 - mln::point, 857
- last_element
 - mln::util::set, 967
- lazy_image
 - mln::lazy_image, 728
- ldlt_decomp
 - mln::algebra, 176
- ldlt_solve
 - mln::algebra, 176
- lemmings
 - mln::util, 384
- len
 - mln::Box, 516
 - mln::box, 512
- length
 - mln::p_run, 831
 - mln::win::backdiag2d, 1028
 - mln::win::cube3d, 1030
 - mln::win::diag2d, 1033
 - mln::win::line, 1034
 - mln::win::octagon2d, 1036
- light_gray
 - mln::literal, 303
- lime
 - mln::literal, 303
- line
 - mln::accu, 169
 - mln::draw, 229
 - mln::win::line, 1034
- line_gradient
 - mln::morpho, 336
- line_graph< G >, 404
- linear
 - mln::fun::x2x::linear, 645
- linfty
 - mln::norm, 356
- linfty_distance
 - mln::norm, 356
- load
 - mln::io::cloud, 258
 - mln::io::dicom, 259
 - mln::io::dump, 260
 - mln::io::fits, 261
 - mln::io::fld, 262
 - mln::io::magick, 263
 - mln::io::off, 264
 - mln::io::pbm, 266
 - mln::io::pbms, 267
 - mln::io::pfm, 268
 - mln::io::pgm, 269
 - mln::io::pgms, 270
 - mln::io::plot, 271
 - mln::io::pnm, 272
 - mln::io::pnms, 274
 - mln::io::ppm, 275
 - mln::io::ppms, 276

- mln::io::raw, 276
 - mln::io::tiff, 277
- load_raw_2d
 - mln::io::pnm, 273
- lower_dim_adj_faces
 - mln::topo::algebraic_face, 891
 - mln::topo::algebraic_n_face, 895
 - mln::topo::face, 905
 - mln::topo::n_face, 914
- lowest_priority
 - mln::p_priority, 819
- lut_vec
 - mln::value::lut_vec, 1006
- lvalue
 - mln::complex_image, 537
 - mln::decorated_image, 547
 - mln::doc::Fastest_Image, 557
 - mln::doc::Image, 566
 - mln::doc::Pixel_iterator, 574
 - mln::flat_image, 610
 - mln::fun_image, 653
 - mln::hexa, 696
 - mln::image1d, 701
 - mln::image2d, 705
 - mln::image2d_h, 709
 - mln::image3d, 712
 - mln::interpolated, 716
 - mln::lazy_image, 728
 - mln::tr_image, 922
 - mln::value::stack_image, 1016
 - mln::violent_cast_image, 1022
- magenta
 - mln::literal, 303
- main_branch
 - mln::util::tree, 977
- make_algebraic_face
 - mln::topo, 371
- make_algebraic_n_face
 - mln::topo, 371
- make_debug_graph_image
 - mln, 158
- make_greater_point
 - mln::util, 384
- make_greater_psite
 - mln::util, 384
- mask
 - mln::graph_elt_neighborhood_if, 679
 - mln::graph_elt_window_if, 688
- mask_t
 - mln::graph_elt_window_if, 686
- mat
 - mln::make, 321
- max
 - mln::literal, 303
 - mln::morpho::tree::filter, 352
- max_col
 - mln::geom, 246
- max_component
 - mln::io::pnm, 273
- max_ind
 - mln::geom, 246
- max_row
 - mln::geom, 246
- max_sli
 - mln::geom, 246
- max_tree
 - mln::morpho::tree, 348
- mean
 - mln::accu::stat::var, 493
 - mln::accu::stat::variance, 495
 - mln::estim, 231
- mean_t
 - mln::accu::stat::var, 493
- median
 - mln::data, 207
 - mln::data::approx, 212, 213
 - mln::data::naive, 219
- medium_gray
 - mln::literal, 303
- memory_size
 - mln::box, 512
 - mln::p_array, 778
 - mln::p_centered, 781
 - mln::p_edges, 789
 - mln::p_if, 796
 - mln::p_image, 799
 - mln::p_key, 806
 - mln::p_line2d, 809
 - mln::p_mutable_array_of, 812
 - mln::p_priority, 819
 - mln::p_queue, 823
 - mln::p_queue_fast, 827
 - mln::p_run, 831
 - mln::p_set, 835
 - mln::p_transformed, 838
 - mln::p_vaccess, 842
 - mln::p_vertices, 847
 - mln::util::array, 931
 - mln::util::set, 967
- merge
 - mln::box, 512
- mesh
 - mln::doc::Point_Site, 576
- mesh_corner_point_area
 - mln::geom, 246
- mesh_curvature
 - mln::geom, 247
- mesh_normal
 - mln::geom, 247
- meyer_wst
 - mln::morpho, 337
- min
 - mln::arith, 180
 - mln::literal, 303
 - mln::morpho, 337
 - mln::morpho::tree::filter, 352

- min_col
 - mln::geom, [247](#), [248](#)
- min_ind
 - mln::geom, [248](#)
- min_inplace
 - mln::arith, [180](#)
 - mln::morpho, [337](#)
- min_max
 - mln::estim, [231](#)
- min_row
 - mln::geom, [248](#)
- min_sli
 - mln::geom, [248](#)
- min_tree
 - mln::morpho::tree, [348](#)
- minus
 - mln::arith, [181](#)
 - mln::morpho, [338](#)
- minus_cst
 - mln::arith, [182](#)
- minus_cst_inplace
 - mln::arith, [182](#)
- minus_infty
 - mln::point, [857](#)
- minus_inplace
 - mln::arith, [183](#)
- mirror
 - mln::border, [192](#)
- mln, [135](#)
 - a_point_of, [155](#)
 - apply_p2p, [156](#)
 - before, [167](#)
 - bin_1complex_image2d, [151](#)
 - bin_2complex_image3df, [152](#)
 - box1d, [152](#)
 - box2d, [152](#)
 - box2d_h, [152](#)
 - box3d, [152](#)
 - compose, [156](#)
 - discrete_plane_1complex_geometry, [152](#)
 - discrete_plane_2complex_geometry, [152](#)
 - dpoint1d, [153](#)
 - dpoint2d, [153](#)
 - dpoint2d_h, [153](#)
 - dpoint3d, [153](#)
 - duplicate, [156](#)
 - extend, [157](#)
 - float_2complex_image3df, [153](#)
 - implies, [157](#)
 - initialize, [157](#)
 - int_u8_1complex_image2d, [153](#)
 - int_u8_2complex_image2d, [153](#)
 - int_u8_2complex_image3df, [153](#)
 - larger_than, [157](#)
 - make_debug_graph_image, [158](#)
 - mln_exact, [158](#)
 - mln_gen_complex_neighborhood, [158](#)
 - mln_gen_complex_window, [159](#)
 - mln_gen_complex_window_p, [159](#), [160](#)
 - mln_regular, [160](#)
 - mln_trait_op_neq, [160](#)
 - operator<, [162](#)
 - operator<<, [163](#)
 - operator<=, [163](#), [164](#)
 - operator*, [161](#)
 - operator++, [161](#)
 - operator-, [161](#)
 - operator--, [162](#)
 - operator==, [164](#), [165](#)
 - p_run2d, [154](#)
 - p_runs2d, [154](#)
 - point1d, [154](#)
 - point1df, [154](#)
 - point2d, [154](#)
 - point2d_h, [154](#)
 - point2df, [154](#)
 - point3d, [154](#)
 - point3df, [154](#)
 - primary, [166](#)
 - ptransform, [166](#)
 - rgb8_2complex_image3df, [155](#)
 - sagittal_dec, [167](#)
 - space_2complex_geometry, [155](#)
 - unsigned_2complex_image3df, [155](#)
 - up, [167](#)
 - vec2d_d, [155](#)
 - vec2d_f, [155](#)
 - vec3d_d, [155](#)
 - vec3d_f, [155](#)
- mln::doc::Dpoint
 - dim, [555](#)
- mln::doc::Point_Site
 - dim, [576](#)
- mln::dpoint
 - dim, [590](#)
- mln::point
 - dim, [856](#)
- mln::Accumulator
 - take_as_init, [500](#)
 - take_n_times, [500](#)
- mln::Accumulator< E >, [498](#)
- mln::Box
 - bbox, [516](#)
 - diff, [516](#)
 - inter, [516](#)
 - is_empty, [516](#)
 - len, [516](#)
 - nsites, [516](#)
 - operator<, [517](#)
 - operator<<, [517](#)
 - operator<=, [517](#), [518](#)
 - operator==, [518](#)
 - sym_diff, [518](#)
 - uni, [518](#)
 - unique, [518](#)
- mln::Box< E >, [514](#)

mln::Browsing< E >, 520
 mln::Delta_Point_Site< E >, 549
 mln::Delta_Point_Site< void >, 549
 mln::Dpoint
 to_dpoint, 592
 mln::Dpoint< E >, 591
 mln::Edge< E >, 599
 mln::Function
 Function, 654
 mln::Function< E >, 654
 mln::Function< void >, 654
 mln::Function_n2v< E >, 655
 mln::Function_v2b< E >, 656
 mln::Function_v2v< E >, 656
 mln::Function_vv2b< E >, 657
 mln::Function_vv2v< E >, 657
 mln::Gdpoint< E >, 662
 mln::Gdpoint< void >, 662
 mln::Generalized_Pixel< E >, 663
 mln::Gpoint
 operator<<, 668
 operator+, 666
 operator+==, 666
 operator-, 667
 operator-=, 667
 operator/, 668
 operator==, 668
 mln::Gpoint< E >, 665
 mln::Graph< E >, 669
 mln::Image< E >, 698
 mln::Iterator
 next, 720
 mln::Iterator< E >, 718
 mln::Literal< E >, 729
 mln::Mesh< E >, 753
 mln::Meta_Accumulator< E >, 754
 mln::Meta_Function< E >, 756
 mln::Meta_Function_v2v< E >, 758
 mln::Meta_Function_vv2v< E >, 759
 mln::Neighborhood< E >, 772
 mln::Neighborhood< void >, 773
 mln::Object< E >, 773
 mln::Point
 operator+==, 852
 operator-=, 852
 operator/, 853
 point, 852
 to_point, 852
 mln::Point< P >, 851
 mln::Proxy< E >, 859
 mln::Proxy< void >, 859
 mln::Pseudo_Site< E >, 859
 mln::Pseudo_Site< void >, 860
 mln::Regular_Grid< E >, 862
 mln::Site< E >, 863
 mln::Site< void >, 864
 mln::Site_Iterator
 next, 866
 mln::Site_Iterator< E >, 865
 mln::Site_Proxy< E >, 866
 mln::Site_Proxy< void >, 867
 mln::Site_Set
 diff, 869
 inter, 869
 operator<, 869
 operator<<, 869
 operator<=, 870
 operator==, 870
 sym_diff, 870
 uni, 870
 unique, 870
 mln::Site_Set< E >, 867
 mln::Site_Set< void >, 871
 mln::Value< E >, 985
 mln::Vertex< E >, 1019
 mln::Weighted_Window
 operator-, 1027
 mln::Weighted_Window< E >, 1026
 mln::Window< E >, 1038
 mln::accu, 167
 compute, 169
 line, 169
 mln_meta_accu_result, 169
 take, 170
 mln::accu::center
 init, 405
 is_valid, 405
 nsites, 405
 take_as_init, 405
 take_n_times, 405
 to_result, 405
 mln::accu::center< P, V >, 404
 mln::accu::convolve
 init, 406
 is_valid, 406
 take_as_init, 406
 take_n_times, 406
 to_result, 406
 mln::accu::convolve< T1, T2, R >, 405
 mln::accu::count_adjacent_vertices
 init, 407
 is_valid, 407
 set_value, 407
 take_as_init, 408
 take_n_times, 408
 to_result, 408
 mln::accu::count_adjacent_vertices< F, S >, 407
 mln::accu::count_labels
 init, 409
 is_valid, 409
 set_value, 409
 take_as_init, 409
 take_n_times, 409
 to_result, 409
 mln::accu::count_labels< L >, 408
 mln::accu::count_value

- init, 410
- is_valid, 410
- set_value, 410
- take_as_init, 410
- take_n_times, 410
- to_result, 411
- mln::accu::count_value< V >, 409
- mln::accu::histo
 - is_valid, 411
 - take, 411
 - take_as_init, 412
 - take_n_times, 412
 - vect, 412
- mln::accu::histo< V >, 411
- mln::accu::image, 170
- mln::accu::impl, 170
- mln::accu::label_used
 - init, 413
 - is_valid, 413
 - take, 413
 - take_as_init, 413
 - take_n_times, 413
 - to_result, 413
- mln::accu::label_used< L >, 412
- mln::accu::logic, 170
- mln::accu::logic::land, 413
 - init, 414
 - is_valid, 414
 - take_as_init, 414
 - take_n_times, 414
 - to_result, 414
- mln::accu::logic::land_basic, 415
 - can_stop, 415
 - init, 415
 - is_valid, 415
 - take_as_init, 415
 - take_n_times, 416
 - to_result, 416
- mln::accu::logic::lor, 416
 - init, 416
 - is_valid, 416
 - take_as_init, 417
 - take_n_times, 417
 - to_result, 417
- mln::accu::logic::lor_basic, 417
 - can_stop, 418
 - init, 418
 - is_valid, 418
 - take_as_init, 418
 - take_n_times, 418
 - to_result, 418
- mln::accu::maj_h
 - init, 419
 - is_valid, 419
 - take_as_init, 419
 - take_n_times, 419
 - to_result, 419
- mln::accu::maj_h< T >, 418
- mln::accu::math, 171
- mln::accu::math::count
 - init, 420
 - is_valid, 420
 - set_value, 420
 - take_as_init, 420
 - take_n_times, 421
 - to_result, 421
- mln::accu::math::count< T >, 420
- mln::accu::math::inf
 - init, 421
 - is_valid, 422
 - take_as_init, 422
 - take_n_times, 422
 - to_result, 422
- mln::accu::math::inf< T >, 421
- mln::accu::math::sum
 - init, 423
 - is_valid, 423
 - take_as_init, 423
 - take_n_times, 423
 - to_result, 423
- mln::accu::math::sum< T, S >, 422
- mln::accu::math::sup
 - init, 424
 - is_valid, 424
 - take_as_init, 424
 - take_n_times, 424
 - to_result, 424
- mln::accu::math::sup< T >, 423
- mln::accu::max_site
 - init, 425
 - is_valid, 425
 - take_as_init, 425
 - take_n_times, 425
 - to_result, 425
- mln::accu::max_site< I >, 425
- mln::accu::meta::center, 426
- mln::accu::meta::count_adjacent_vertices, 426
- mln::accu::meta::count_labels, 427
- mln::accu::meta::count_value, 428
- mln::accu::meta::histo, 429
- mln::accu::meta::label_used, 430
- mln::accu::meta::logic, 171
- mln::accu::meta::logic::land, 431
- mln::accu::meta::logic::land_basic, 432
- mln::accu::meta::logic::lor, 433
- mln::accu::meta::logic::lor_basic, 434
- mln::accu::meta::maj_h, 435
- mln::accu::meta::math, 172
- mln::accu::meta::math::count, 436
- mln::accu::meta::math::inf, 437
- mln::accu::meta::math::sum, 438
- mln::accu::meta::math::sup, 439
- mln::accu::meta::max_site, 440
- mln::accu::meta::nil, 441
- mln::accu::meta::p< mA >, 442
- mln::accu::meta::pair< A1, A2 >, 443

- mln::accu::meta::rms, 444
- mln::accu::meta::shape, 172
- mln::accu::meta::shape::bbox, 445
- mln::accu::meta::shape::height, 446
- mln::accu::meta::shape::volume, 447
- mln::accu::meta::stat, 172
- mln::accu::meta::stat::max, 448
- mln::accu::meta::stat::max_h, 449
- mln::accu::meta::stat::mean, 450
- mln::accu::meta::stat::median_alt< T >, 451
- mln::accu::meta::stat::median_h, 452
- mln::accu::meta::stat::min, 453
- mln::accu::meta::stat::min_h, 454
- mln::accu::meta::stat::rank, 455
- mln::accu::meta::stat::rank_high_quant, 456
- mln::accu::meta::tuple< n, >, 457
- mln::accu::meta::val< mA >, 458
- mln::accu::nil
 - init, 460
 - is_valid, 460
 - take_as_init, 460
 - take_n_times, 460
 - to_result, 460
- mln::accu::nil< T >, 459
- mln::accu::p
 - init, 461
 - is_valid, 461
 - take_as_init, 461
 - take_n_times, 461
 - to_result, 461
- mln::accu::p< A >, 460
- mln::accu::pair
 - first, 463
 - first_accu, 463
 - init, 463
 - is_valid, 463
 - second, 463
 - second_accu, 463
 - take_as_init, 463
 - take_n_times, 464
 - to_result, 464
- mln::accu::pair< A1, A2, T >, 462
- mln::accu::rms
 - init, 465
 - is_valid, 465
 - take_as_init, 465
 - take_n_times, 465
 - to_result, 465
- mln::accu::rms< T, V >, 464
- mln::accu::shape, 173
- mln::accu::shape::bbox
 - init, 466
 - is_valid, 466
 - take_as_init, 466
 - take_n_times, 466
 - to_result, 466
- mln::accu::shape::bbox< P >, 465
- mln::accu::shape::height
 - argument, 467
 - init, 467
 - is_valid, 467
 - set_value, 468
 - take_as_init, 468
 - take_n_times, 468
 - to_result, 468
 - value, 467
- mln::accu::shape::height< I >, 466
- mln::accu::shape::volume
 - argument, 469
 - init, 469
 - is_valid, 469
 - set_value, 469
 - take_as_init, 470
 - take_n_times, 470
 - to_result, 470
 - value, 469
- mln::accu::shape::volume< I >, 468
- mln::accu::site_set::rectangularity
 - area, 471
 - bbox, 471
 - rectangularity, 471
 - take_as_init, 471
 - take_n_times, 471
 - to_result, 471
- mln::accu::site_set::rectangularity< P >, 470
- mln::accu::stat, 173
 - operator==, 174
- mln::accu::stat::deviation
 - init, 472
 - is_valid, 472
 - take_as_init, 472
 - take_n_times, 472
 - to_result, 473
- mln::accu::stat::deviation< T, S, M >, 471
- mln::accu::stat::histo3d_rgb
 - histo3d_rgb, 474
 - init, 474
 - is_valid, 474
 - take, 474
 - take_as_init, 474
 - take_n_times, 475
 - to_result, 475
- mln::accu::stat::histo3d_rgb< V >, 473
- mln::accu::stat::max
 - init, 476
 - is_valid, 476
 - set_value, 476
 - take_as_init, 476
 - take_n_times, 476
 - to_result, 476
- mln::accu::stat::max< T >, 475
- mln::accu::stat::max_h
 - init, 477
 - is_valid, 477
 - take_as_init, 477
 - take_n_times, 477

- to_result, 477
- mln::accu::stat::max_h< V >, 476
- mln::accu::stat::mean
 - count, 478
 - init, 478
 - is_valid, 478
 - sum, 478
 - take_as_init, 479
 - take_n_times, 479
 - to_result, 479
- mln::accu::stat::mean< T, S, M >, 477
- mln::accu::stat::median_alt
 - is_valid, 480
 - take, 480
 - take_as_init, 480
 - take_n_times, 480
 - to_result, 480
- mln::accu::stat::median_alt< S >, 479
- mln::accu::stat::median_h
 - init, 482
 - is_valid, 482
 - take_as_init, 482
 - take_n_times, 482
 - to_result, 482
- mln::accu::stat::median_h< V >, 481
- mln::accu::stat::meta::deviation, 482
- mln::accu::stat::min
 - init, 484
 - is_valid, 484
 - set_value, 484
 - take_as_init, 484
 - take_n_times, 484
 - to_result, 484
- mln::accu::stat::min< T >, 483
- mln::accu::stat::min_h
 - init, 485
 - is_valid, 485
 - take_as_init, 485
 - take_n_times, 485
 - to_result, 485
- mln::accu::stat::min_h< V >, 485
- mln::accu::stat::min_max
 - first, 487
 - first_accu, 487
 - init, 487
 - is_valid, 487
 - second, 487
 - second_accu, 487
 - take_as_init, 487
 - take_n_times, 488
 - to_result, 488
- mln::accu::stat::min_max< V >, 486
- mln::accu::stat::rank
 - init, 489
 - is_valid, 489
 - k, 489
 - take_as_init, 489
 - take_n_times, 489
- to_result, 489
- mln::accu::stat::rank< bool >, 489
 - init, 490
 - is_valid, 490
 - take_as_init, 490
 - take_n_times, 490
 - to_result, 490
- mln::accu::stat::rank< T >, 488
- mln::accu::stat::rank_high_quant
 - init, 491
 - is_valid, 491
 - take_as_init, 491
 - take_n_times, 491
 - to_result, 492
- mln::accu::stat::rank_high_quant< T >, 491
- mln::accu::stat::var
 - init, 493
 - is_valid, 493
 - mean, 493
 - mean_t, 493
 - n_items, 493
 - take_as_init, 493
 - take_n_times, 493
 - to_result, 493
 - variance, 494
- mln::accu::stat::var< T >, 492
- mln::accu::stat::variance
 - init, 495
 - is_valid, 495
 - mean, 495
 - n_items, 495
 - standard_deviation, 495
 - sum, 495
 - take_n_times, 495
 - to_result, 495
 - var, 496
- mln::accu::stat::variance< T, S, R >, 494
- mln::accu::tuple
 - init, 496
 - is_valid, 496
 - take_as_init, 497
 - take_n_times, 497
 - to_result, 497
- mln::accu::tuple< A, n, >, 496
- mln::accu::val
 - init, 498
 - is_valid, 498
 - take_as_init, 498
 - take_n_times, 498
 - to_result, 498
- mln::accu::val< A >, 497
- mln::algebra, 175
 - det, 176
 - ldlt_decomp, 176
 - ldlt_solve, 176
 - operator*, 176
 - tr, 176
 - vprod, 176

- mln::algebra::h_mat
 - _1, [501](#)
 - h_mat, [501](#)
 - t, [501](#)
- mln::algebra::h_mat< d, T >, [500](#)
- mln::algebra::h_vec
 - h_vec, [503](#)
 - operator mat< n, 1, U >, [503](#)
 - origin, [503](#)
 - t, [503](#)
 - to_vec, [503](#)
 - zero, [503](#)
- mln::algebra::h_vec< d, C >, [502](#)
- mln::arith, [177](#)
 - diff_abs, [179](#)
 - div, [179](#)
 - div_cst, [179](#)
 - div_inplace, [180](#)
 - min, [180](#)
 - min_inplace, [180](#)
 - minus, [181](#)
 - minus_cst, [182](#)
 - minus_cst_inplace, [182](#)
 - minus_inplace, [183](#)
 - plus, [183](#), [184](#)
 - plus_cst, [184](#), [185](#)
 - plus_cst_inplace, [186](#)
 - plus_inplace, [186](#)
 - revert, [186](#)
 - revert_inplace, [187](#)
 - times, [187](#)
 - times_cst, [187](#)
 - times_inplace, [188](#)
- mln::arith::impl, [188](#)
- mln::arith::impl::generic, [189](#)
- mln::binarization, [189](#)
 - binarization, [189](#)
 - threshold, [189](#)
- mln::bkd_pixter1d
 - bkd_pixter1d, [504](#)
 - image, [504](#)
 - next, [504](#)
- mln::bkd_pixter1d< I >, [503](#)
- mln::bkd_pixter2d
 - bkd_pixter2d, [505](#)
 - image, [505](#)
 - next, [505](#)
- mln::bkd_pixter2d< I >, [505](#)
- mln::bkd_pixter3d
 - bkd_pixter3d, [506](#)
 - image, [506](#)
 - next, [507](#)
- mln::bkd_pixter3d< I >, [506](#)
- mln::border, [190](#)
 - adjust, [190](#)
 - duplicate, [191](#)
 - equalize, [191](#)
 - fill, [191](#)
 - find, [192](#)
 - get, [192](#)
 - mirror, [192](#)
 - resize, [192](#)
- mln::border::impl, [193](#)
- mln::border::impl::generic, [193](#)
- mln::box
 - bbox, [511](#)
 - bkd_piter, [510](#)
 - box, [511](#)
 - crop_wrt, [511](#)
 - element, [510](#)
 - enlarge, [511](#)
 - fwd_piter, [510](#)
 - has, [511](#)
 - is_empty, [512](#)
 - is_valid, [512](#)
 - len, [512](#)
 - memory_size, [512](#)
 - merge, [512](#)
 - nsites, [512](#)
 - operator<<, [513](#)
 - pcenter, [513](#)
 - piter, [510](#)
 - pmax, [513](#)
 - pmin, [513](#)
 - psite, [510](#)
 - site, [510](#)
 - to_larger, [513](#)
- mln::box< P >, [507](#)
- mln::box_runend_piter
 - next, [519](#)
 - run_length, [519](#)
- mln::box_runend_piter< P >, [518](#)
- mln::box_runstart_piter
 - box_runstart_piter, [520](#)
 - next, [520](#)
 - run_length, [520](#)
- mln::box_runstart_piter< P >, [519](#)
- mln::canvas, [193](#)
 - distance_front, [194](#)
 - distance_geodesic, [194](#)
- mln::canvas::browsing, [194](#)
- mln::canvas::browsing::backdiagonal2d_t, [521](#)
- mln::canvas::browsing::breadth_first_search_t, [523](#)
- mln::canvas::browsing::depth_first_search_t, [523](#)
- mln::canvas::browsing::diagonal2d_t, [523](#)
- mln::canvas::browsing::dir_struct_elt_incr_update_t, [525](#)
- mln::canvas::browsing::directional_t, [526](#)
- mln::canvas::browsing::fwd_t, [528](#)
- mln::canvas::browsing::hyper_directional_t, [529](#)
- mln::canvas::browsing::snake_fwd_t, [531](#)
- mln::canvas::browsing::snake_generic_t, [532](#)
- mln::canvas::browsing::snake_vert_t, [534](#)
- mln::canvas::chamfer< F >, [535](#)
- mln::canvas::impl, [195](#)
- mln::canvas::labeling, [195](#)

- blobs, 196
- mln::canvas::labeling::impl, 196
- mln::canvas::morpho, 196
- mln::category< R(*) (A) >, 535
- mln::complex_image
 - complex_image, 537
 - dim, 538
 - domain, 537
 - geom, 537
 - lvalue, 537
 - operator(), 538
 - rvalue, 537
 - skeleton, 537
 - value, 537
 - values, 538
- mln::complex_image< D, G, V >, 536
- mln::complex_neighborhood_bkd_piter
 - complex_neighborhood_bkd_piter, 539
 - iter, 539
 - iter_type, 539
 - next, 539
 - psite, 539
- mln::complex_neighborhood_bkd_piter< I, G, N >, 538
- mln::complex_neighborhood_fwd_piter
 - complex_neighborhood_fwd_piter, 541
 - iter, 541
 - iter_type, 540
 - next, 541
 - psite, 540
- mln::complex_neighborhood_fwd_piter< I, G, N >, 540
- mln::complex_psite
 - change_target, 542
 - complex_psite, 542
 - face, 543
 - face_id, 543
 - invalidate, 543
 - is_valid, 543
 - n, 543
 - site_set, 543
- mln::complex_psite< D, G >, 541
- mln::complex_window_bkd_piter
 - complex_window_bkd_piter, 545
 - iter, 545
 - iter_type, 544
 - next, 545
 - psite, 544
- mln::complex_window_bkd_piter< I, G, W >, 544
- mln::complex_window_fwd_piter
 - complex_window_fwd_piter, 546
 - iter, 546
 - iter_type, 546
 - next, 546
 - psite, 546
- mln::complex_window_fwd_piter< I, G, W >, 545
- mln::convert, 196
 - from_to, 198, 199
 - mln_image_from_grid, 199
 - mln_window, 199
 - to, 199
 - to_dpoint, 199
 - to_fun, 200, 202
 - to_image, 200
 - to_p_array, 200
 - to_p_set, 200, 201
 - to_qimage, 201
 - to_upper_window, 201
 - to_window, 201, 202
- mln::data, 202
 - abs, 204
 - abs_inplace, 204
 - apply, 204
 - compute, 204, 205
 - convert, 205
 - fast_median, 205
 - fill, 206
 - fill_with_image, 206
 - fill_with_value, 206
 - median, 207
 - mln_meta_accu_result, 207
 - paste, 207
 - paste_without_localization, 208
 - replace, 208
 - saturate, 208
 - saturate_inplace, 209
 - sort_offsets_increasing, 209
 - sort_psites_decreasing, 209
 - sort_psites_increasing, 209
 - stretch, 210
 - to_enc, 210
 - transform, 210
 - transform_inplace, 211
 - update, 211
 - wrap, 211
- mln::data::approx, 212
 - median, 212, 213
- mln::data::approx::impl, 213
- mln::data::impl, 213
 - paste_without_localization_fast, 214
 - paste_without_localization_fastest, 214
 - paste_without_localization_lines, 215
 - stretch, 215
 - transform_inplace_lowq, 216
 - update_fastest, 216
- mln::data::impl::generic, 216
 - fill_with_image, 217
 - fill_with_value, 217
 - paste, 217
 - transform, 218
 - transform_inplace, 218
 - update, 218
- mln::data::naive, 219
 - median, 219
- mln::data::naive::impl, 220
- mln::debug, 220
 - draw_graph, 221, 222
 - filename, 222

- format, [222](#)
- iota, [223](#)
- mosaic, [223](#)
- println, [223](#)
- println_with_border, [223](#)
- put_word, [223](#)
- slices_2d, [224](#)
- superpose, [224](#)
- z_order, [225](#)
- mln::debug::impl, [225](#)
- mln::decorated_image
 - decorated_image, [548](#)
 - decoration, [548](#)
 - lvalue, [547](#)
 - operator decorated_image< const I, D >, [548](#)
 - operator(), [548](#)
 - psite, [547](#)
 - rvalue, [548](#)
 - skeleton, [548](#)
- mln::decorated_image< I, D >, [547](#)
- mln::def, [225](#)
 - coord, [225](#)
 - coordf, [225](#)
- mln::display, [226](#)
- mln::display::impl, [226](#)
- mln::display::impl::generic, [226](#)
- mln::doc, [227](#)
- mln::doc::Accumulator
 - argument, [550](#)
 - init, [550](#)
 - take, [550](#)
- mln::doc::Accumulator< E >, [550](#)
- mln::doc::Box
 - bbox, [552](#)
 - bkd_piter, [552](#)
 - fwd_piter, [552](#)
 - has, [552](#)
 - nsites, [553](#)
 - pmax, [553](#)
 - pmin, [553](#)
 - psite, [552](#)
 - site, [552](#)
- mln::doc::Box< E >, [551](#)
- mln::doc::Dpoint
 - coord, [554](#)
 - dpoint, [554](#)
 - point, [554](#)
- mln::doc::Dpoint< E >, [553](#)
- mln::doc::Fastest_Image
 - bbox, [559](#)
 - bkd_piter, [557](#)
 - border, [559](#)
 - buffer, [559](#)
 - coord, [557](#)
 - delta_index, [559](#)
 - domain, [559](#)
 - dpoint, [557](#)
 - fwd_piter, [557](#)
 - has, [560](#)
 - is_valid, [560](#)
 - lvalue, [557](#)
 - nelements, [560](#)
 - nsites, [560](#)
 - operator(), [560](#), [561](#)
 - point, [558](#)
 - point_at_index, [562](#)
 - pset, [558](#)
 - psite, [558](#)
 - rvalue, [558](#)
 - skeleton, [558](#)
 - value, [558](#)
 - values, [562](#)
 - vset, [558](#)
- mln::doc::Fastest_Image< E >, [555](#)
- mln::doc::Generalized_Pixel
 - ima, [563](#)
 - image, [563](#)
 - rvalue, [563](#)
 - val, [563](#)
 - value, [563](#)
- mln::doc::Generalized_Pixel< E >, [562](#)
- mln::doc::Image
 - bbox, [567](#)
 - bkd_piter, [565](#)
 - coord, [565](#)
 - domain, [567](#)
 - dpoint, [566](#)
 - fwd_piter, [566](#)
 - has, [568](#)
 - is_valid, [568](#)
 - lvalue, [566](#)
 - nsites, [568](#)
 - operator(), [568](#), [569](#)
 - point, [566](#)
 - pset, [566](#)
 - psite, [566](#)
 - rvalue, [567](#)
 - skeleton, [567](#)
 - value, [567](#)
 - values, [569](#)
 - vset, [567](#)
- mln::doc::Image< E >, [564](#)
- mln::doc::Iterator
 - invalidate, [570](#)
 - is_valid, [570](#)
 - start, [570](#)
- mln::doc::Iterator< E >, [569](#)
- mln::doc::Neighborhood
 - bkd_niter, [571](#)
 - dpoint, [571](#)
 - fwd_niter, [571](#)
 - niter, [571](#)
 - point, [571](#)
- mln::doc::Neighborhood< E >, [570](#)
- mln::doc::Object< E >, [572](#)
- mln::doc::Pixel_Iterator

- ima, [574](#)
- image, [574](#)
- invalidate, [574](#)
- is_valid, [574](#)
- lvalue, [574](#)
- rvalue, [574](#)
- start, [574](#)
- val, [574](#)
- value, [574](#)
- mln::doc::Pixel_Iterator< E >, [572](#)
- mln::doc::Point_Site
 - coord, [575](#)
 - dpoint, [575](#)
 - mesh, [576](#)
 - point, [576](#)
 - to_point, [577](#)
- mln::doc::Point_Site< E >, [575](#)
- mln::doc::Site_Iterator
 - invalidate, [578](#)
 - is_valid, [578](#)
 - operator psite, [578](#)
 - psite, [578](#)
 - start, [578](#)
- mln::doc::Site_Iterator< E >, [577](#)
- mln::doc::Site_Set
 - bkd_piter, [580](#)
 - fwd_piter, [580](#)
 - has, [580](#)
 - psite, [580](#)
 - site, [580](#)
- mln::doc::Site_Set< E >, [579](#)
- mln::doc::Value_Iterator
 - invalidate, [582](#)
 - is_valid, [582](#)
 - operator value, [582](#)
 - start, [582](#)
 - value, [582](#)
- mln::doc::Value_Iterator< E >, [580](#)
- mln::doc::Value_Set
 - bkd_viter, [583](#)
 - fwd_viter, [583](#)
 - has, [584](#)
 - index_of, [584](#)
 - nvalues, [584](#)
 - value, [583](#)
- mln::doc::Value_Set< E >, [582](#)
- mln::doc::Weighted_Window
 - bkd_qiter, [585](#)
 - delta, [586](#)
 - dpoint, [585](#)
 - fwd_qiter, [585](#)
 - is_centered, [586](#)
 - is_empty, [586](#)
 - point, [586](#)
 - sym, [586](#)
 - weight, [586](#)
 - win, [586](#)
 - window, [586](#)
- mln::doc::Weighted_Window< E >, [584](#)
- mln::doc::Window
 - bkd_qiter, [587](#)
 - fwd_qiter, [587](#)
 - qiter, [587](#)
- mln::doc::Window< E >, [586](#)
- mln::dpoint
 - coord, [589](#)
 - dpoint, [590](#)
 - grid, [589](#)
 - operator mln::algebra::vec< dpoint< G, C >::dim, Q >, [591](#)
 - psite, [589](#)
 - set_all, [591](#)
 - site, [589](#)
 - to_vec, [591](#)
 - vec, [590](#)
- mln::dpoint< G, C >, [588](#)
- mln::dpoints_bkd_paxter
 - center_val, [594](#)
 - dpoints_bkd_paxter, [593](#)
 - invalidate, [594](#)
 - is_valid, [594](#)
 - next, [594](#)
 - start, [594](#)
 - update, [594](#)
- mln::dpoints_bkd_paxter< I >, [592](#)
- mln::dpoints_fwd_paxter
 - center_val, [596](#)
 - dpoints_fwd_paxter, [595](#)
 - invalidate, [596](#)
 - is_valid, [596](#)
 - next, [596](#)
 - start, [596](#)
 - update, [596](#)
- mln::dpoints_fwd_paxter< I >, [595](#)
- mln::dpsites_bkd_piter
 - dpsites_bkd_piter, [597](#)
 - next, [597](#)
- mln::dpsites_bkd_piter< V >, [597](#)
- mln::dpsites_fwd_piter
 - dpsites_fwd_piter, [598](#)
 - next, [599](#)
- mln::dpsites_fwd_piter< V >, [598](#)
- mln::draw, [228](#)
 - box, [228](#)
 - box_plain, [228](#)
 - dashed_line, [229](#)
 - line, [229](#)
 - plot, [229](#)
 - polygon, [230](#)
 - site_set, [230](#)
- mln::edge_image
 - edge_image, [601](#)
 - edge_nbh_t, [600](#)
 - edge_win_t, [600](#)
 - graph_t, [600](#)
 - nbh_t, [600](#)

- operator(), 601
- site_function_t, 601
- skeleton, 601
- win_t, 601
- mln::edge_image< P, V, G >, 599
- mln::estim, 230
 - mean, 231
 - min_max, 231
 - sum, 232
- mln::extended
 - domain, 603
 - extended, 602
 - skeleton, 602
 - value, 602
- mln::extended< I >, 601
- mln::extension, 232
 - adjust, 233
 - adjust_duplicate, 233
 - adjust_fill, 233
 - duplicate, 234
 - fill, 234
- mln::extension_fun
 - extension, 604
 - extension_fun, 604
 - has, 604
 - operator(), 604, 605
 - rvalue, 604
 - skeleton, 604
 - value, 604
- mln::extension_fun< I, F >, 603
- mln::extension_ima
 - extension, 606
 - extension_ima, 606
 - has, 606
 - operator(), 606
 - rvalue, 606
 - skeleton, 606
 - value, 606
- mln::extension_ima< I, J >, 605
- mln::extension_val
 - change_extension, 608
 - extension, 608
 - extension_val, 608
 - has, 608
 - operator(), 608, 609
 - rvalue, 608
 - skeleton, 608
 - value, 608
- mln::extension_val< I >, 607
- mln::flat_image
 - domain, 610
 - flat_image, 610
 - has, 611
 - lvalue, 610
 - operator(), 611
 - rvalue, 610
 - skeleton, 610
 - value, 610
- mln::flat_image< T, S >, 609
- mln::fun, 234
 - mln::fun::access, 235
 - mln::fun::from_accu< A >, 611
 - mln::fun::i2v, 235
 - operator<<, 236
 - mln::fun::n2v, 236
 - mln::fun::n2v::white_gaussian< V >, 611
 - mln::fun::p2b, 236
 - mln::fun::p2b::antilogy, 612
 - mln::fun::p2b::tautology, 613
 - mln::fun::p2p, 237
 - mln::fun::p2v, 237
 - mln::fun::stat, 237
 - mln::fun::v2b, 237
 - mln::fun::v2b::lnot< V >, 614
 - mln::fun::v2b::threshold< V >, 616
 - mln::fun::v2i, 237
 - mln::fun::v2v, 238
 - f_hsi_to_rgb_3x8, 238
 - f_hsl_to_rgb_3x8, 238
 - f_rgb_to_hsi_f, 238
 - f_rgb_to_hsl_f, 239
 - mln::fun::v2v::ch_function_value< F, V >, 617
 - mln::fun::v2v::component< T, i >, 618
 - mln::fun::v2v::l1_norm< V, R >, 619
 - mln::fun::v2v::l2_norm< V, R >, 620
 - mln::fun::v2v::linear< V, T, R >, 621
 - mln::fun::v2v::linfty_norm< V, R >, 622
 - mln::fun::v2v::rgb8_to_rgbn
 - operator(), 624
 - mln::fun::v2v::rgb8_to_rgbn< n >, 623
 - mln::fun::v2w2v, 239
 - mln::fun::v2w2v::cos< V >, 624
 - mln::fun::v2w_w2v, 239
 - mln::fun::v2w_w2v::l1_norm< V, R >, 625
 - mln::fun::v2w_w2v::l2_norm< V, R >, 626
 - mln::fun::v2w_w2v::linfty_norm< V, R >, 628
 - mln::fun::vv2b, 239
 - mln::fun::vv2b::eq< L, R >, 629
 - mln::fun::vv2b::ge< L, R >, 629
 - mln::fun::vv2b::gt< L, R >, 630
 - mln::fun::vv2b::implies< L, R >, 631
 - mln::fun::vv2b::le< L, R >, 632
 - mln::fun::vv2b::lt< L, R >, 633
 - mln::fun::vv2v, 240
 - mln::fun::vv2v::diff_abs< V >, 634
 - mln::fun::vv2v::land< L, R >, 635
 - mln::fun::vv2v::land_not< L, R >, 636
 - mln::fun::vv2v::lor< L, R >, 637
 - mln::fun::vv2v::lxor< L, R >, 638
 - mln::fun::vv2v::max< V >, 639
 - mln::fun::vv2v::min< L, R >, 640
 - mln::fun::vv2v::vec< V >, 641
 - mln::fun::x2p, 240
 - mln::fun::x2p::closest_point< P >, 642
 - mln::fun::x2v, 241
 - mln::fun::x2v::bilinear

- operator(), 643
- mln::fun::x2v::bilinear< I >, 643
- mln::fun::x2v::trilinear< I >, 644
- mln::fun::x2x, 241
- mln::fun::x2x::composed
 - composed, 644
- mln::fun::x2x::composed< T2, T1 >, 644
- mln::fun::x2x::linear
 - ima, 646
 - linear, 645
 - operator(), 646
- mln::fun::x2x::linear< I >, 645
- mln::fun::x2x::rotation
 - data_t, 648
 - inv, 649
 - invert, 648
 - operator(), 649
 - rotation, 648
 - set_alpha, 649
 - set_axis, 649
- mln::fun::x2x::rotation< n, C >, 646
- mln::fun::x2x::translation
 - data_t, 651
 - inv, 651
 - invert, 651
 - operator(), 651
 - set_t, 652
 - t, 652
 - translation, 651
- mln::fun::x2x::translation< n, C >, 649
- mln::fun_image
 - fun_image, 653
 - lvalue, 653
 - operator(), 654
 - rvalue, 653
 - skeleton, 653
 - value, 653
- mln::fun_image< F, I >, 652
- mln::fwd_pixter1d
 - fwd_pixter1d, 659
 - image, 659
 - next, 659
- mln::fwd_pixter1d< I >, 658
- mln::fwd_pixter2d
 - fwd_pixter2d, 660
 - image, 660
 - next, 660
- mln::fwd_pixter2d< I >, 659
- mln::fwd_pixter3d
 - fwd_pixter3d, 661
 - image, 661
 - next, 661
- mln::fwd_pixter3d< I >, 661
- mln::geom, 241
 - bbox, 245
 - chamfer, 245
 - delta, 245, 246
 - max_col, 246
 - max_ind, 246
 - max_row, 246
 - max_sli, 246
 - mesh_corner_point_area, 246
 - mesh_curvature, 247
 - mesh_normal, 247
 - min_col, 247, 248
 - min_ind, 248
 - min_row, 248
 - min_sli, 248
 - ncols, 248
 - ninds, 249
 - nrows, 249
 - nsites, 249
 - nslis, 249
 - pmin_pmax, 249, 250
 - rotate, 250, 251
 - seeds2tiling, 251
 - translate, 251, 252
 - vertical_symmetry, 252
- mln::geom::complex_geometry
 - add_location, 664
 - complex_geometry, 664
 - operator(), 664
- mln::geom::complex_geometry< D, P >, 664
- mln::geom::impl, 252
 - seeds2tiling, 253
- mln::graph, 253
 - compute, 253
 - labeling, 254
 - to_neighb, 254
 - to_win, 255
- mln::graph::attribute::card_t, 669
 - result, 670
- mln::graph::attribute::representative_t, 670
 - result, 670
- mln::graph_elt_mixed_neighborhood
 - bkd_niter, 672
 - fwd_niter, 672
 - niter, 672
- mln::graph_elt_mixed_neighborhood< G, S, S2 >, 671
- mln::graph_elt_mixed_window
 - bkd_qiter, 674
 - center_t, 674
 - delta, 675
 - fwd_qiter, 674
 - graph_element, 675
 - is_centered, 675
 - is_empty, 675
 - is_symmetric, 676
 - is_valid, 676
 - psite, 675
 - qiter, 675
 - site, 675
 - sym, 676
 - target, 675
- mln::graph_elt_mixed_window< G, S, S2 >, 672
- mln::graph_elt_neighborhood

- bkd_niter, [677](#)
 - fwd_niter, [677](#)
 - niter, [677](#)
- mln::graph_elt_neighborhood< G, S >, [676](#)
- mln::graph_elt_neighborhood_if
 - bkd_niter, [679](#)
 - fwd_niter, [679](#)
 - graph_elt_neighborhood_if, [679](#)
 - mask, [679](#)
 - niter, [679](#)
- mln::graph_elt_neighborhood_if< G, S, I >, [677](#)
- mln::graph_elt_window
 - bkd_qiter, [681](#)
 - center_t, [681](#)
 - delta, [682](#)
 - fwd_qiter, [681](#)
 - graph_element, [682](#)
 - is_centered, [682](#)
 - is_empty, [682](#)
 - is_symmetric, [683](#)
 - is_valid, [683](#)
 - psite, [682](#)
 - qiter, [682](#)
 - site, [682](#)
 - sym, [683](#)
 - target, [682](#)
- mln::graph_elt_window< G, S >, [679](#)
- mln::graph_elt_window_if
 - bkd_qiter, [686](#)
 - change_mask, [687](#)
 - delta, [687](#)
 - fwd_qiter, [686](#)
 - graph_elt_window_if, [687](#)
 - is_centered, [687](#)
 - is_empty, [687](#)
 - is_symmetric, [687](#)
 - is_valid, [687](#)
 - mask, [688](#)
 - mask_t, [686](#)
 - psite, [686](#)
 - qiter, [686](#)
 - site, [686](#)
 - sym, [688](#)
 - target, [686](#)
- mln::graph_elt_window_if< G, S, I >, [683](#)
- mln::graph_window_base
 - delta, [689](#)
 - is_centered, [689](#)
 - is_empty, [689](#)
 - is_symmetric, [689](#)
 - is_valid, [689](#)
 - site, [689](#)
 - sym, [690](#)
- mln::graph_window_base< P, E >, [688](#)
- mln::graph_window_if_piter
 - element, [691](#)
 - graph_window_if_piter, [691](#)
 - id, [691](#)
 - next, [691](#)
 - P, [691](#)
- mln::graph_window_if_piter< S, W, I >, [690](#)
- mln::graph_window_piter
 - center_t, [692](#)
 - change_target_site_set, [694](#)
 - element, [694](#)
 - graph_element, [693](#)
 - graph_window_piter, [693](#)
 - id, [694](#)
 - next, [694](#)
 - P, [693](#)
 - target_site_set, [694](#)
- mln::graph_window_piter< S, W, I >, [691](#)
- mln::grid, [255](#)
- mln::hexa
 - bkd_piter, [696](#)
 - domain, [697](#)
 - fwd_piter, [696](#)
 - has, [697](#)
 - hexa, [697](#)
 - lvalue, [696](#)
 - operator(), [697](#)
 - psite, [696](#)
 - rvalue, [696](#)
 - skeleton, [697](#)
 - value, [697](#)
- mln::hexa< I >, [694](#)
- mln::histo, [255](#)
 - compute, [256](#)
 - equalize, [256](#)
- mln::histo::array< T >, [698](#)
- mln::histo::impl, [256](#)
- mln::histo::impl::generic, [257](#)
- mln::image1d
 - bbox, [702](#)
 - border, [702](#)
 - buffer, [702](#)
 - delta_index, [702](#)
 - domain, [702](#)
 - element, [702](#)
 - has, [703](#)
 - image1d, [701](#), [702](#)
 - lvalue, [701](#)
 - nelements, [703](#)
 - ninds, [703](#)
 - operator(), [703](#)
 - point_at_index, [703](#)
 - rvalue, [701](#)
 - skeleton, [701](#)
 - value, [701](#)
 - vbbox, [703](#)
- mln::image1d< T >, [700](#)
- mln::image2d
 - bbox, [706](#)
 - border, [706](#)
 - buffer, [706](#)
 - delta_index, [706](#)

- domain, 706
- element, 706
- has, 706
- image2d, 705
- lvalue, 705
- ncols, 707
- nelements, 707
- nrows, 707
- operator(), 707
- point_at_index, 707
- rvalue, 705
- skeleton, 705
- value, 705
- mln::image2d< T >, 703
- mln::image2d_h
 - bkd_piter, 709
 - domain, 710
 - fwd_piter, 709
 - has, 710
 - image2d_h, 710
 - lvalue, 709
 - operator(), 710
 - psite, 709
 - rvalue, 709
 - skeleton, 709
 - value, 709
- mln::image2d_h< V >, 707
- mln::image3d
 - bbox, 713
 - border, 713
 - buffer, 713
 - delta_index, 713
 - domain, 713
 - element, 714
 - has, 714
 - image3d, 713
 - lvalue, 712
 - ncols, 714
 - nelements, 714
 - nrows, 714
 - nslis, 714
 - operator(), 714
 - point_at_index, 715
 - rvalue, 712
 - skeleton, 712
 - value, 712
 - vbbox, 715
- mln::image3d< T >, 710
- mln::impl, 257
- mln::interpolated
 - has, 716
 - interpolated, 716
 - is_valid, 717
 - lvalue, 716
 - psite, 716
 - rvalue, 716
 - skeleton, 716
 - value, 716
- mln::interpolated< I, F >, 715
- mln::io, 257
- mln::io::cloud, 258
 - load, 258
 - save, 258
- mln::io::dicom, 259
 - get_header, 259
 - load, 259
- mln::io::dicom::dicom_header, 717
- mln::io::dump, 260
 - get_header, 260
 - load, 260
 - save, 260
- mln::io::dump::dump_header, 717
- mln::io::fits, 261
 - load, 261
- mln::io::fld, 261
 - load, 262
 - read_header, 262
 - write_header, 262
- mln::io::fld::fld_header, 717
- mln::io::magick, 263
 - load, 263
 - save, 263
- mln::io::off, 264
 - load, 264
 - save, 265
 - save_bin_alt, 265
- mln::io::pbm, 265
 - load, 266
 - save, 266
- mln::io::pbm::impl, 266
- mln::io::pbms, 267
 - load, 267
- mln::io::pbms::impl, 267
- mln::io::pfm, 267
 - load, 268
 - save, 268
- mln::io::pfm::impl, 269
- mln::io::pgm, 269
 - load, 269
 - save, 270
- mln::io::pgms, 270
 - load, 270
- mln::io::plot, 270
 - load, 271
 - save, 271, 272
- mln::io::pnm, 272
 - load, 272
 - load_raw_2d, 273
 - max_component, 273
 - save, 273
- mln::io::pnm::impl, 273
- mln::io::pnms, 273
 - load, 274
- mln::io::ppm, 274
 - load, 275
 - save, 275

- mln::io::ppms, 275
 - load, 276
- mln::io::raw, 276
 - get_header, 276
 - load, 276
 - save, 277
- mln::io::raw::raw_header, 717
- mln::io::tiff, 277
 - load, 277
- mln::io::txt, 277
 - save, 278
- mln::labeled_image
 - bbox, 723
 - bbox_t, 722
 - bboxes, 723
 - labeled_image, 722, 723
 - nlabels, 723
 - relabel, 723
 - skeleton, 722
 - subdomain, 723
 - update_data, 723
- mln::labeled_image< I >, 720
- mln::labeled_image_base
 - bbox, 725
 - bbox_t, 725
 - bboxes, 726
 - labeled_image_base, 725
 - nlabels, 726
 - relabel, 726
 - subdomain, 726
 - update_data, 726
- mln::labeled_image_base< I, E >, 724
- mln::labeling, 278
 - background, 281
 - blobs, 281
 - blobs_and_compute, 282
 - colorize, 282, 283
 - compute, 283, 284
 - compute_image, 285
 - compute_in_window, 286
 - fill_holes, 286
 - flat_zones, 286
 - foreground, 287
 - pack, 287, 288
 - pack_inplace, 288
 - regional_maxima, 288
 - regional_minima, 288
 - relabel, 289
 - relabel_inplace, 290
 - superpose, 290
 - value, 291
 - value_and_compute, 291
 - wrap, 291, 292
- mln::labeling::impl, 292
 - compute_fastest, 293
- mln::labeling::impl::generic, 293
 - compute, 294, 295
- mln::lazy_image
 - domain, 728
 - has, 728
 - lazy_image, 728
 - lvalue, 728
 - operator(), 728, 729
 - rvalue, 728
 - skeleton, 728
- mln::lazy_image< I, F, B >, 727
- mln::linear, 295
 - gaussian, 296
 - gaussian_1st_derivative, 297
 - gaussian_2nd_derivative, 297
 - mln_ch_convolve, 297, 298
 - mln_ch_convolve_grad, 298
- mln::linear::impl, 299
- mln::linear::local, 299
 - convolve, 299
- mln::linear::local::impl, 300
- mln::literal, 300
 - black, 302
 - blue, 302
 - brown, 302
 - cyan, 302
 - dark_gray, 302
 - green, 303
 - identity, 303
 - light_gray, 303
 - lime, 303
 - magenta, 303
 - max, 303
 - medium_gray, 303
 - min, 303
 - olive, 303
 - one, 304
 - orange, 304
 - origin, 304
 - pink, 304
 - purple, 304
 - red, 304
 - teal, 304
 - violet, 304
 - white, 305
 - yellow, 305
 - zero, 305
- mln::literal::black_t, 731
- mln::literal::blue_t, 731
- mln::literal::brown_t, 732
- mln::literal::cyan_t, 733
- mln::literal::green_t, 734
- mln::literal::identity_t, 735
- mln::literal::light_gray_t, 736
- mln::literal::lime_t, 737
- mln::literal::magenta_t, 738
- mln::literal::max_t, 739
- mln::literal::min_t, 740
- mln::literal::olive_t, 741
- mln::literal::one_t, 742
- mln::literal::orange_t, 743

- mln::literal::origin_t, 744
- mln::literal::pink_t, 745
- mln::literal::purple_t, 746
- mln::literal::red_t, 747
- mln::literal::teal_t, 748
- mln::literal::violet_t, 749
- mln::literal::white_t, 750
- mln::literal::yellow_t, 751
- mln::literal::zero_t, 752
- mln::logical, 305
 - and_inplace, 306
 - and_not, 306
 - and_not_inplace, 306
 - not_inplace, 307
 - or_inplace, 307
 - xor_inplace, 307
- mln::logical::impl, 307
- mln::logical::impl::generic, 308
- mln::make, 308
 - attachment, 312
 - box1d, 312
 - box2d, 313
 - box2d_h, 314
 - box3d, 314, 315
 - cell, 315
 - couple, 316
 - detachment, 316
 - dpoint2d_h, 316
 - dummy_p_edges, 317
 - dummy_p_vertices, 317
 - edge_image, 318, 319
 - h_mat, 320
 - image, 320
 - image2d, 321
 - image3d, 321
 - influence_zone_adjacency_graph, 321
 - mat, 321
 - ord_pair, 322
 - p_edges_with_mass_centers, 322
 - p_vertices_with_mass_centers, 322
 - pix, 323
 - pixel, 323
 - point2d_h, 323
 - rag_and_labeled_wsl, 323
 - region_adjacency_graph, 324
 - relabelfun, 324, 325
 - vec, 325, 326
 - vertex_image, 326, 327
 - voronoi, 327
 - w_window, 327
 - w_window1d, 328
 - w_window2d, 328
 - w_window3d, 328
 - w_window_directional, 329
- mln::math, 329
 - abs, 330
- mln::metal, 330
 - ands< E1, E2, E3, E4, E5, E6, E7, E8 >, 759
 - converts_to< T, U >, 760
 - equal< T1, T2 >, 760
 - goes_to< T, U >, 760
 - impl, 331
 - is< T, U >, 761
 - is_a< T, M >, 761
 - is_not< T, U >, 761
 - is_not_a< T, M >, 762
 - math, 331
 - math::impl, 331
 - morpho, 331
 - complementation, 334
 - complementation_inplace, 334
 - contrast, 334
 - dilation, 334
 - erosion, 334
 - erosion_tolerant, 334
 - general, 335
 - gradient, 335
 - gradient_external, 335
 - gradient_internal, 335
 - hit_or_miss, 335
 - hit_or_miss_background_closing, 335
 - hit_or_miss_background_opening, 336
 - hit_or_miss_closing, 336
 - hit_or_miss_opening, 336
 - laplacian, 336
 - line_gradient, 336
 - meyer_wst, 337
 - min, 337
 - min_inplace, 337
 - minus, 338
 - plus, 338
 - rank_filter, 338
 - thick_miss, 338
 - thickening, 338
 - thin_fit, 338
 - thinning, 339
 - top_hat_black, 339
 - top_hat_self_complementary, 339
 - top_hat_white, 339
 - morpho::approx, 340
 - morpho::attribute, 340
 - morpho::attribute::card
 - init, 763
 - is_valid, 763
 - take_as_init, 763
 - take_n_times, 763
 - to_result, 763
 - morpho::attribute::card< I >, 762
 - morpho::attribute::count_adjacent_vertices
 - init, 764
 - is_valid, 764
 - take_as_init, 764
 - take_n_times, 764
 - to_result, 764

- mln::morpho::attribute::count_adjacent_vertices< I >, 763
- mln::morpho::attribute::height
 - base_level, 765
 - init, 765
 - is_valid, 765
 - take_as_init, 765
 - take_n_times, 765
 - to_result, 766
- mln::morpho::attribute::height< I >, 764
- mln::morpho::attribute::sharpness
 - area, 767
 - height, 767
 - init, 767
 - is_valid, 767
 - take_as_init, 767
 - take_n_times, 767
 - to_result, 767
 - volume, 767
- mln::morpho::attribute::sharpness< I >, 766
- mln::morpho::attribute::sum
 - init, 768
 - is_valid, 768
 - set_value, 768
 - take_as_init, 768
 - take_n_times, 769
 - to_result, 769
 - untake, 769
- mln::morpho::attribute::sum< I, S >, 767
- mln::morpho::attribute::volume
 - area, 770
 - init, 770
 - is_valid, 770
 - take_as_init, 770
 - take_n_times, 770
 - to_result, 770
- mln::morpho::attribute::volume< I >, 769
- mln::morpho::closing::approx, 340
 - structural, 341
- mln::morpho::elementary, 341
 - closing, 341
 - mln_trait_op_minus_twice, 341
 - opening, 342
 - top_hat_black, 342
 - top_hat_self_complementary, 342
 - top_hat_white, 342
- mln::morpho::impl, 342
- mln::morpho::impl::generic, 343
- mln::morpho::opening::approx, 343
 - structural, 343
- mln::morpho::reconstruction, 343
- mln::morpho::reconstruction::by_dilation, 344
- mln::morpho::reconstruction::by_erosion, 344
- mln::morpho::tree, 344
 - compute_attribute_image, 346
 - compute_attribute_image_from, 346
 - compute_parent, 347
 - dual_input_max_tree, 348
 - max_tree, 348
 - min_tree, 348
 - propagate_if, 349
 - propagate_if_value, 349
 - propagate_node_to_ancestors, 349
 - propagate_node_to_descendants, 350
 - propagate_representative, 350
- mln::morpho::tree::filter, 351
 - direct, 351
 - filter, 351
 - max, 352
 - min, 352
 - subtractive, 352
- mln::morpho::watershed, 353
 - flooding, 353, 354
 - superpose, 354
 - topological, 354
- mln::morpho::watershed::watershed, 354
- mln::morpho::watershed::watershed::generic, 355
- mln::neighb
 - bkd_niter, 771
 - fwd_niter, 771
 - neighb, 772
 - niter, 771
- mln::neighb< W >, 770
- mln::norm, 355
 - l1, 356
 - l1_distance, 356
 - l2, 356
 - l2_distance, 356
 - linfty, 356
 - linfty_distance, 356
 - sqr_l2, 357
- mln::norm::impl, 357
- mln::opt, 357
 - at, 358
- mln::opt::impl, 358
- mln::p2p_image
 - domain, 774
 - fun, 774
 - operator(), 775
 - p2p_image, 774
 - skeleton, 774
- mln::p2p_image< I, F >, 773
- mln::p_array
 - append, 777
 - bkd_piter, 776
 - change, 777
 - clear, 778
 - element, 776
 - fwd_piter, 776
 - has, 778
 - i_element, 777
 - insert, 778
 - is_valid, 778
 - memory_size, 778
 - nsites, 778
 - p_array, 777

- ptiter, 777
 - psite, 777
 - reserve, 779
 - resize, 779
 - std_vector, 779
- mln::p_array< P >, 775
- mln::p_centered
 - bkd_ptiter, 780
 - center, 781
 - element, 780
 - fwd_ptiter, 780
 - has, 781
 - is_valid, 781
 - memory_size, 781
 - p_centered, 781
 - ptiter, 780
 - psite, 780
 - site, 781
 - window, 781
- mln::p_centered< W >, 779
- mln::p_complex
 - bkd_ptiter, 783
 - cplx, 784
 - element, 783
 - fwd_ptiter, 783
 - geom, 784
 - has, 784
 - is_valid, 784
 - nfaces, 784
 - nfaces_of_dim, 784
 - nsites, 785
 - p_complex, 784
 - ptiter, 783
 - psite, 783
- mln::p_complex< D, G >, 782
- mln::p_edges
 - bkd_ptiter, 786
 - edge, 786
 - element, 786
 - fun_t, 787
 - function, 788
 - fwd_ptiter, 787
 - graph, 788
 - graph_element, 787
 - graph_t, 787
 - has, 789
 - invalidate, 789
 - is_valid, 789
 - memory_size, 789
 - nedges, 789
 - nsites, 789
 - p_edges, 787, 788
 - ptiter, 787
 - psite, 787
- mln::p_edges< G, F >, 785
- mln::p_faces
 - bkd_ptiter, 791
 - cplx, 792
 - element, 791
 - fwd_ptiter, 791
 - is_valid, 792
 - nfaces, 792
 - nsites, 792
 - p_faces, 791
 - ptiter, 791
 - psite, 791
- mln::p_faces< N, D, P >, 790
- mln::p_graph_ptiter
 - graph, 793
 - id, 793
 - mln_q_subject, 793
 - next, 793
 - p_graph_ptiter, 793
- mln::p_graph_ptiter< S, I >, 792
- mln::p_if
 - bkd_ptiter, 795
 - element, 795
 - fwd_ptiter, 795
 - has, 796
 - is_valid, 796
 - memory_size, 796
 - overset, 796
 - p_if, 795
 - ptiter, 795
 - pred, 796
 - predicate, 796
 - psite, 795
- mln::p_if< S, F >, 794
- mln::p_image
 - bkd_ptiter, 798
 - clear, 799
 - element, 798
 - fwd_ptiter, 798
 - has, 799
 - i_element, 798
 - insert, 799
 - is_valid, 799
 - memory_size, 799
 - nsites, 799
 - operator typename internal::p_image_site_set< I >::ret, 799
 - p_image, 798
 - ptiter, 798
 - psite, 798
 - r_element, 798
 - remove, 799
 - S, 798
 - toggle, 800
- mln::p_image< I >, 796
- mln::p_indexed_bkd_ptiter
 - index, 800
 - next, 801
 - p_indexed_bkd_ptiter, 800
- mln::p_indexed_bkd_ptiter< S >, 800
- mln::p_indexed_fwd_ptiter
 - index, 802

- next, [802](#)
- p_indexed_fwd_piter, [801](#)
- mln::p_indexed_fwd_piter< S >, [801](#)
- mln::p_indexed_psite< S >, [802](#)
- mln::p_key
 - bkd_piter, [804](#)
 - change_key, [805](#)
 - change_keys, [805](#)
 - clear, [805](#)
 - element, [804](#)
 - exists_key, [805](#)
 - fwd_piter, [804](#)
 - has, [805](#)
 - i_element, [804](#)
 - insert, [805](#)
 - is_valid, [806](#)
 - key, [806](#)
 - keys, [806](#)
 - memory_size, [806](#)
 - nsites, [806](#)
 - operator(), [806](#)
 - p_key, [805](#)
 - piter, [804](#)
 - psite, [804](#)
 - r_element, [804](#)
 - remove, [806](#)
 - remove_key, [806](#)
- mln::p_key< K, P >, [802](#)
- mln::p_line2d, [807](#)
 - bbox, [809](#)
 - begin, [809](#)
 - bkd_piter, [808](#)
 - element, [808](#)
 - end, [809](#)
 - fwd_piter, [808](#)
 - has, [809](#)
 - is_valid, [809](#)
 - memory_size, [809](#)
 - nsites, [810](#)
 - p_line2d, [808](#)
 - piter, [808](#)
 - psite, [808](#)
 - q_box, [808](#)
 - std_vector, [810](#)
- mln::p_mutable_array_of
 - bkd_piter, [811](#)
 - clear, [812](#)
 - element, [811](#)
 - fwd_piter, [811](#)
 - has, [812](#)
 - i_element, [811](#)
 - insert, [812](#)
 - is_valid, [812](#)
 - memory_size, [812](#)
 - nelements, [813](#)
 - p_mutable_array_of, [812](#)
 - piter, [812](#)
 - psite, [812](#)
 - reserve, [813](#)
- mln::p_mutable_array_of< S >, [810](#)
- mln::p_n_faces_bkd_piter
 - n, [814](#)
 - next, [814](#)
 - p_n_faces_bkd_piter, [814](#)
- mln::p_n_faces_bkd_piter< D, G >, [813](#)
- mln::p_n_faces_fwd_piter
 - n, [815](#)
 - next, [815](#)
 - p_n_faces_fwd_piter, [815](#)
- mln::p_n_faces_fwd_piter< D, G >, [814](#)
- mln::p_priority
 - bkd_piter, [817](#)
 - clear, [818](#)
 - element, [817](#)
 - exists_priority, [818](#)
 - front, [818](#)
 - fwd_piter, [817](#)
 - has, [818](#)
 - highest_priority, [818](#)
 - i_element, [817](#)
 - insert, [818](#)
 - is_valid, [819](#)
 - lowest_priority, [819](#)
 - memory_size, [819](#)
 - nsites, [819](#)
 - operator(), [819](#)
 - p_priority, [817](#)
 - piter, [817](#)
 - pop, [819](#)
 - pop_front, [820](#)
 - priorities, [820](#)
 - psite, [817](#)
 - push, [820](#)
- mln::p_priority< P, Q >, [815](#)
- mln::p_queue
 - bkd_piter, [822](#)
 - clear, [822](#)
 - element, [822](#)
 - front, [822](#)
 - fwd_piter, [822](#)
 - has, [823](#)
 - i_element, [822](#)
 - insert, [823](#)
 - is_valid, [823](#)
 - memory_size, [823](#)
 - nsites, [823](#)
 - p_queue, [822](#)
 - piter, [822](#)
 - pop, [823](#)
 - pop_front, [823](#)
 - psite, [822](#)
 - push, [824](#)
 - std_deque, [824](#)
- mln::p_queue< P >, [820](#)
- mln::p_queue_fast
 - bkd_piter, [825](#)

- clear, 826
- compute_has, 826
- element, 825
- empty, 826
- front, 826
- fwd_piter, 826
- has, 827
- i_element, 826
- insert, 827
- is_valid, 827
- memory_size, 827
- nsites, 827
- p_queue_fast, 826
- piter, 826
- pop, 827
- pop_front, 827
- psite, 826
- purge, 828
- push, 828
- reserve, 828
- std_vector, 828
- mln::p_queue_fast< P >, 824
- mln::p_run
 - bbox, 830
 - bkd_piter, 829
 - element, 829
 - end, 830
 - fwd_piter, 830
 - has, 831
 - has_index, 831
 - init, 831
 - is_valid, 831
 - length, 831
 - memory_size, 831
 - nsites, 831
 - p_run, 830
 - piter, 830
 - psite, 830
 - q_box, 830
 - start, 832
- mln::p_run< P >, 828
- mln::p_set
 - bkd_piter, 833
 - clear, 834
 - element, 833
 - fwd_piter, 833
 - has, 834
 - i_element, 833
 - insert, 834
 - is_valid, 835
 - memory_size, 835
 - nsites, 835
 - p_set, 834
 - piter, 833
 - psite, 834
 - r_element, 834
 - remove, 835
 - std_vector, 835
 - util_set, 835
- mln::p_set< P >, 832
- mln::p_transformed
 - bkd_piter, 836
 - element, 836
 - function, 837
 - fwd_piter, 837
 - has, 837
 - is_valid, 837
 - memory_size, 838
 - p_transformed, 837
 - piter, 837
 - primary_set, 838
 - psite, 837
- mln::p_transformed< S, F >, 835
- mln::p_transformed_piter
 - change_target, 839
 - next, 839
 - p_transformed_piter, 839
- mln::p_transformed_piter< Pi, S, F >, 838
- mln::p_vaccess
 - bkd_piter, 841
 - element, 841
 - fwd_piter, 841
 - has, 842
 - i_element, 841
 - insert, 842
 - is_valid, 842
 - memory_size, 842
 - operator(), 842
 - p_vaccess, 842
 - piter, 841
 - pset, 841
 - psite, 841
 - value, 841
 - values, 842
 - vset, 841
- mln::p_vaccess< V, S >, 839
- mln::p_vertices
 - bkd_piter, 844
 - element, 844
 - fun_t, 844
 - function, 846
 - fwd_piter, 845
 - graph, 847
 - graph_element, 845
 - graph_t, 845
 - has, 847
 - invalidate, 847
 - is_valid, 847
 - memory_size, 847
 - nsites, 847
 - nvertices, 848
 - operator(), 848
 - p_vertices, 845, 846
 - piter, 845
 - psite, 845
 - vertex, 845

mln::p_vertices< G, F >, 843
 mln::pixel
 change_to, 849
 is_valid, 849
 pixel, 849
 mln::pixel< I >, 848
 mln::plain
 operator I, 851
 operator=, 851
 plain, 850
 skeleton, 850
 mln::plain< I >, 849
 mln::point
 coord, 856
 delta, 856
 dpsite, 856
 grid, 856
 h_vec, 856
 last_coord, 857
 minus_infty, 857
 operator+=, 857
 operator-=, 857
 origin, 859
 plus_infty, 858
 point, 856, 857
 set_all, 858
 to_h_vec, 858
 to_vec, 858
 vec, 856
 mln::point< G, C >, 853
 mln::pw, 359
 mln::pw::image
 image, 861
 skeleton, 861
 mln::pw::image< F, S >, 860
 mln::registration, 359
 get_rot, 360
 icp, 360, 361
 registration1, 361
 registration2, 361
 registration3, 361
 mln::registration::closest_point_basic< P >, 861
 mln::registration::closest_point_with_map< P >, 862
 mln::safe_image
 operator safe_image< const I >, 863
 skeleton, 863
 mln::safe_image< I >, 862
 mln::select, 361
 mln::select::p_of< P >, 863
 mln::set, 362
 card, 362
 compute, 362
 compute_with_weights, 363
 get, 363
 has, 364
 mln_meta_accu_result, 364
 mln::sub_image
 domain, 872
 operator sub_image< const I, S >, 872
 skeleton, 872
 sub_image, 872
 mln::sub_image< I, S >, 871
 mln::sub_image_if
 domain, 873
 skeleton, 873
 sub_image_if, 873
 mln::sub_image_if< I, S >, 872
 mln::subsampling, 364
 antialiased, 365
 gaussian_subsampling, 365
 subsampling, 365
 mln::tag, 365
 mln::test, 366
 positive, 366
 predicate, 366, 367
 mln::test::impl, 367
 mln::thru_image
 operator thru_image< const I, F >, 874
 mln::thru_image< I, F >, 873
 mln::thruin_image
 operator thruin_image< const I1, const I2, F >, 875
 psite, 875
 rvalue, 875
 skeleton, 875
 value, 875
 mln::thruin_image< I1, I2, F >, 874
 mln::topo, 367
 detach, 371
 edge, 371
 is_facet, 371
 make_algebraic_face, 371
 make_algebraic_n_face, 371
 operator<, 373
 operator<<, 374
 operator+, 372
 operator-, 372, 373
 operator==, 374, 375
 mln::topo::adj_higher_dim_connected_n_face_bkd_iter
 next, 876
 mln::topo::adj_higher_dim_connected_n_face_bkd_iter< D >, 875
 mln::topo::adj_higher_dim_connected_n_face_fwd_iter
 next, 877
 mln::topo::adj_higher_dim_connected_n_face_fwd_iter< D >, 876
 mln::topo::adj_higher_face_bkd_iter
 adj_higher_face_bkd_iter, 878
 next, 878
 mln::topo::adj_higher_face_bkd_iter< D >, 877
 mln::topo::adj_higher_face_fwd_iter
 adj_higher_face_fwd_iter, 879
 next, 879
 mln::topo::adj_higher_face_fwd_iter< D >, 878
 mln::topo::adj_lower_dim_connected_n_face_bkd_iter
 next, 880

mln::topo::adj_lower_dim_connected_n_face_bkd_iter< D >, 879
 mln::topo::adj_lower_dim_connected_n_face_fwd_iter next, 881
 mln::topo::adj_lower_dim_connected_n_face_fwd_iter< D >, 880
 mln::topo::adj_lower_face_bkd_iter adj_lower_face_bkd_iter, 882 next, 882
 mln::topo::adj_lower_face_bkd_iter< D >, 881
 mln::topo::adj_lower_face_fwd_iter adj_lower_face_fwd_iter, 883 next, 883
 mln::topo::adj_lower_face_fwd_iter< D >, 882
 mln::topo::adj_lower_higher_face_bkd_iter next, 884
 mln::topo::adj_lower_higher_face_bkd_iter< D >, 883
 mln::topo::adj_lower_higher_face_fwd_iter next, 885
 mln::topo::adj_lower_higher_face_fwd_iter< D >, 884
 mln::topo::adj_m_face_bkd_iter adj_m_face_bkd_iter, 886 next, 886
 mln::topo::adj_m_face_bkd_iter< D >, 885
 mln::topo::adj_m_face_fwd_iter adj_m_face_fwd_iter, 887 next, 888
 mln::topo::adj_m_face_fwd_iter< D >, 887
 mln::topo::algebraic_face algebraic_face, 889, 890 cplx, 890 data, 890 dec_face_id, 890 dec_n, 890 face_id, 890 higher_dim_adj_faces, 891 inc_face_id, 891 inc_n, 891 invalidate, 891 is_valid, 891 lower_dim_adj_faces, 891 n, 891 set_cplx, 891 set_face_id, 891 set_n, 892 set_sign, 892 sign, 892
 mln::topo::algebraic_face< D >, 888
 mln::topo::algebraic_n_face algebraic_n_face, 894 cplx, 894 data, 894 dec_face_id, 894 face_id, 894 higher_dim_adj_faces, 894 inc_face_id, 895 invalidate, 895 is_valid, 895 lower_dim_adj_faces, 895 n, 895 set_cplx, 895 set_face_id, 895 set_n, 896 set_sign, 895 sign, 896
 mln::topo::algebraic_n_face< N, D >, 892
 mln::topo::center_only_iter center_only_iter, 897 next, 897
 mln::topo::center_only_iter< D >, 896
 mln::topo::centered_bkd_iter_adapter centered_bkd_iter_adapter, 898 next, 898
 mln::topo::centered_bkd_iter_adapter< D, I >, 897
 mln::topo::centered_fwd_iter_adapter centered_fwd_iter_adapter, 899 next, 899
 mln::topo::centered_fwd_iter_adapter< D, I >, 898
 mln::topo::complex add_face, 900 addr, 901 bkd_citer, 900 complex, 900 fwd_citer, 900 nfaces, 901 nfaces_of_dim, 901 nfaces_of_static_dim, 901 print, 901 print_faces, 901
 mln::topo::complex< D >, 899
 mln::topo::face cplx, 904 data, 904 dec_face_id, 904 dec_n, 904 face, 903 face_id, 904 higher_dim_adj_faces, 904 inc_face_id, 904 inc_n, 904 invalidate, 904 is_valid, 905 lower_dim_adj_faces, 905 n, 905 set_cplx, 905 set_face_id, 905 set_n, 905
 mln::topo::face< D >, 902
 mln::topo::face_bkd_iter face_bkd_iter, 906 next, 906 start, 906
 mln::topo::face_bkd_iter< D >, 905
 mln::topo::face_fwd_iter face_fwd_iter, 907 next, 907 start, 907

- mln::topo::face_fwd_iter< D >, 907
- mln::topo::is_n_face< N >, 908
- mln::topo::is_simple_2d_t< N >, 909
- mln::topo::is_simple_cell
 - D, 912
 - mln_geom, 911
 - operator(), 911
 - psite, 911
 - result, 911
 - set_image, 911
- mln::topo::is_simple_cell< I >, 909
- mln::topo::n_face
 - cplx, 913
 - data, 913
 - dec_face_id, 913
 - face_id, 914
 - higher_dim_adj_faces, 914
 - inc_face_id, 914
 - invalidate, 914
 - is_valid, 914
 - lower_dim_adj_faces, 914
 - n, 914
 - n_face, 913
 - set_cplx, 914
 - set_face_id, 915
- mln::topo::n_face< N, D >, 912
- mln::topo::n_face_bkd_iter
 - n, 915
 - next, 915
 - start, 916
- mln::topo::n_face_bkd_iter< D >, 915
- mln::topo::n_face_fwd_iter
 - n, 916
 - next, 917
 - start, 917
- mln::topo::n_face_fwd_iter< D >, 916
- mln::topo::n_faces_set
 - add, 918
 - faces, 918
 - faces_type, 918
 - reserve, 918
- mln::topo::n_faces_set< N, D >, 917
- mln::topo::skeleton::is_simple_point< N >, 918
- mln::topo::static_n_face_bkd_iter
 - next, 920
 - start, 920
 - static_n_face_bkd_iter, 919
- mln::topo::static_n_face_bkd_iter< N, D >, 919
- mln::topo::static_n_face_fwd_iter
 - next, 921
 - start, 921
 - static_n_face_fwd_iter, 921
- mln::topo::static_n_face_fwd_iter< N, D >, 920
- mln::tr_image
 - domain, 923
 - has, 923
 - is_valid, 923
 - lvalue, 922
 - operator(), 923
 - psite, 922
 - rvalue, 922
 - set_tr, 924
 - site, 922
 - skeleton, 923
 - tr, 924
 - tr_image, 923
 - value, 923
- mln::tr_image< S, I, T >, 921
- mln::trace, 376
- mln::trait, 376
- mln::transform, 376
 - distance_and_closest_point_geodesic, 377
 - distance_and_influence_zone_geodesic, 378
 - distance_front, 378
 - distance_geodesic, 379
 - hough, 379
 - influence_zone_front, 379
 - influence_zone_geodesic, 379
 - influence_zone_geodesic_saturated, 380
- mln::transformed_image
 - domain, 925
 - operator transformed_image< const I, F >, 925
 - operator(), 925
 - skeleton, 925
 - transformed_image, 925
- mln::transformed_image< I, F >, 924
- mln::unproject_image
 - domain, 926
 - operator(), 926, 927
 - unproject_image, 926
- mln::unproject_image< I, D, F >, 926
- mln::util, 380
 - display_branch, 383
 - display_tree, 383
 - lemmings, 384
 - make_greater_point, 384
 - make_greater_psite, 384
 - operator<, 384
 - operator<<, 384
 - operator==, 384, 385
 - ord_strict, 385
 - ord_weak, 385
 - tree_fast_to_image, 385
 - tree_to_fast, 385
 - tree_to_image, 385
 - vertex_id_t, 383
- mln::util::adjacency_matrix
 - adjacency_matrix, 927
- mln::util::adjacency_matrix< V >, 927
- mln::util::array
 - append, 931
 - array, 930
 - bkd_eiter, 930
 - clear, 931
 - eiter, 930
 - element, 930

- fill, 931
- fwd_eiter, 930
- is_empty, 931
- last, 931
- memory_size, 931
- nelements, 932
- operator(), 932
- reserve, 932
- resize, 933
- result, 930
- size, 933
- std_vector, 933
- mln::util::array< T >, 928
- mln::util::branch
 - apex, 934
 - branch, 934
 - util_tree, 934
- mln::util::branch< T >, 933
- mln::util::branch_iter
 - deepness, 935
 - invalidate, 935
 - is_valid, 935
 - next, 935
 - operator util::tree_node< T > &, 935
 - start, 935
- mln::util::branch_iter< T >, 934
- mln::util::branch_iter_ind
 - deepness, 936
 - invalidate, 936
 - is_valid, 936
 - next, 937
 - operator util::tree_node< T > &, 937
 - start, 937
- mln::util::branch_iter_ind< T >, 936
- mln::util::couple
 - change_both, 938
 - change_first, 938
 - change_second, 938
 - first, 938
 - second, 938
- mln::util::couple< T, U >, 937
- mln::util::eat, 939
- mln::util::edge
 - category, 940
 - change_graph, 941
 - edge, 941
 - graph, 941
 - graph_t, 940
 - id, 941
 - id_t, 941
 - id_value_t, 941
 - invalidate, 941
 - is_valid, 941
 - ith_nbh_edge, 942
 - nmax_nbh_edges, 942
 - operator edge_id_t, 942
 - update_id, 942
 - v1, 942
 - v2, 942
 - v_other, 942
- mln::util::edge< G >, 939
- mln::util::fibonacci_heap
 - clear, 944
 - fibonacci_heap, 944
 - front, 944
 - is_empty, 944
 - is_valid, 944
 - nelements, 945
 - operator=, 945
 - pop_front, 945
 - push, 945
- mln::util::fibonacci_heap< P, T >, 943
- mln::util::graph, 945
 - add_edge, 948
 - add_vertex, 948
 - add_vertices, 949
 - e_ith_nbh_edge, 949
 - e_nmax, 949
 - edge, 949
 - edge_fwd_iter, 947
 - edge_nbh_edge_fwd_iter, 947
 - edges, 949
 - edges_set_t, 947
 - edges_t, 947
 - graph, 948
 - has_e, 949
 - has_v, 950
 - is_subgraph_of, 950
 - v1, 950
 - v2, 950
 - v_ith_nbh_edge, 950
 - v_ith_nbh_vertex, 950
 - v_nmax, 950
 - vertex, 951
 - vertex_nbh_edge_fwd_iter, 947
 - vertex_nbh_vertex_fwd_iter, 948
 - vertices_t, 948
- mln::util::greater_point
 - operator(), 951
- mln::util::greater_point< I >, 951
- mln::util::greater_psite
 - operator(), 952
- mln::util::greater_psite< I >, 951
- mln::util::head< T, R >, 952
- mln::util::ignore, 952
- mln::util::ilcell< T >, 953
- mln::util::impl, 386
- mln::util::line_graph
 - e_ith_nbh_edge, 956
 - e_nmax, 956
 - edge, 956
 - edge_fwd_iter, 955
 - edge_nbh_edge_fwd_iter, 955
 - edges_t, 955
 - graph, 956
 - has, 956

- has_e, 956
- has_v, 956
- is_subgraph_of, 957
- v1, 957
- v2, 957
- v_ith_nbh_edge, 957
- v_ith_nbh_vertex, 957
- v_nmax, 957
- vertex, 957
- vertex_nbh_edge_fwd_iter, 955
- vertex_nbh_vertex_fwd_iter, 955
- vertices_t, 955
- mln::util::line_graph< G >, 953
- mln::util::nil, 958
- mln::util::node< T, R >, 958
- mln::util::object_id
 - object_id, 960
 - value_t, 959
- mln::util::object_id< Tag, V >, 958
- mln::util::ord< T >, 960
- mln::util::ord_pair
 - change_both, 961
 - change_first, 961
 - change_second, 961
 - first, 962
 - second, 962
- mln::util::ord_pair< T >, 960
- mln::util::pix
 - ima, 963
 - p, 963
 - pix, 963
 - psite, 963
 - v, 963
 - value, 963
- mln::util::pix< I >, 962
- mln::util::set
 - bkd_eiter, 965
 - clear, 966
 - eiter, 965
 - element, 966
 - first_element, 966
 - fwd_eiter, 966
 - has, 966
 - insert, 967
 - is_empty, 967
 - last_element, 967
 - memory_size, 967
 - nelements, 968
 - remove, 968
 - set, 966
 - std_vector, 968
- mln::util::set< T >, 964
- mln::util::site_pair
 - first, 969
 - pair, 969
 - second, 970
- mln::util::site_pair< P >, 969
- mln::util::soft_heap
 - ~soft_heap, 971
 - clear, 972
 - element, 971
 - is_empty, 972
 - is_valid, 972
 - nelements, 972
 - pop_front, 972
 - push, 972
 - soft_heap, 971
- mln::util::soft_heap< T, R >, 970
- mln::util::timer, 972
- mln::util::tracked_ptr
 - ~tracked_ptr, 974
 - operator bool, 974
 - operator->, 974, 975
 - operator=, 975
 - tracked_ptr, 974
- mln::util::tracked_ptr< T >, 973
- mln::util::tree
 - add_tree_down, 976
 - add_tree_up, 976
 - check_consistency, 976
 - main_branch, 977
 - root, 977
 - tree, 976
- mln::util::tree< T >, 975
- mln::util::tree_node
 - add_child, 978
 - check_consistency, 979
 - children, 979
 - delete_tree_node, 979
 - elt, 979, 980
 - parent, 980
 - print, 980
 - search, 980
 - search_rec, 980
 - set_parent, 981
 - tree_node, 978
- mln::util::tree_node< T >, 977
- mln::util::vertex
 - Category, 983
 - change_graph, 983
 - edge_with, 983
 - graph, 983
 - graph_t, 983
 - id, 983
 - id_t, 983
 - id_value_t, 983
 - invalidate, 984
 - is_valid, 984
 - ith_nbh_edge, 984
 - ith_nbh_vertex, 984
 - nmax_nbh_edges, 984
 - nmax_nbh_vertices, 984
 - operator vertex_id_t, 984
 - other, 984
 - update_id, 985
 - vertex, 983

- mln::util::vertex< G >, 981
- mln::util::yes, 985
- mln::value, 386
 - cast, 391
 - equiv, 391
 - float01_16, 389
 - float01_8, 389
 - gl16, 389
 - gl8, 390
 - glf, 390
 - int_s16, 390
 - int_s32, 390
 - int_s8, 390
 - int_u12, 390
 - int_u16, 390
 - int_u32, 390
 - int_u8, 390
 - label_16, 391
 - label_32, 391
 - label_8, 391
 - operator<<, 392–395
 - operator*, 391, 392
 - operator+, 392
 - operator-, 392
 - operator/, 392
 - operator==, 395
 - other, 395
 - rgb16, 391
 - rgb8, 391
 - stack, 395
- mln::value::Integer< E >, 1002
- mln::value::Integer< void >, 1002
- mln::value::float01, 987
 - enc, 988
 - equiv, 988
 - float01, 988
 - nbits, 988
 - operator float, 988
 - set_nbits, 988
 - to_nbits, 989
 - value, 989
 - value_ind, 989
- mln::value::float01_f, 989
 - float01_f, 989, 990
 - operator float, 990
 - operator=, 990
 - value, 990
- mln::value::graylevel
 - graylevel, 992
 - operator=, 992, 993
 - to_float, 993
 - value, 993
- mln::value::graylevel< n >, 990
- mln::value::graylevel_f, 993
 - graylevel_f, 994
 - operator graylevel< n >, 995
 - operator=, 995
 - value, 995
- mln::value::impl, 396
- mln::value::int_s
 - int_s, 997
 - one, 997
 - operator int, 997
 - operator=, 997
 - zero, 997
- mln::value::int_s< n >, 995
- mln::value::int_u
 - int_u, 999
 - next, 999
 - operator unsigned, 999
 - operator-, 999
 - operator=, 999
- mln::value::int_u< n >, 997
- mln::value::int_u_sat
 - int_u_sat, 1001
 - one, 1001
 - operator int, 1001
 - operator+&, 1001
 - operator=, 1001
 - operator=, 1001
 - zero, 1001
- mln::value::int_u_sat< n >, 999
- mln::value::label
 - enc, 1003
 - label, 1003, 1004
 - next, 1004
 - operator unsigned, 1004
 - operator+&, 1004
 - operator--, 1004
 - operator=, 1004
 - prev, 1004
- mln::value::label< n >, 1002
- mln::value::lut_vec
 - bkd_viter, 1006
 - fwd_viter, 1006
 - has, 1006
 - index_of, 1006
 - lut_vec, 1006
 - nvalues, 1007
 - value, 1006
- mln::value::lut_vec< S, T >, 1005
- mln::value::proxy
 - ~proxy, 1009
 - enc, 1008
 - equiv, 1008
 - operator=, 1009
 - proxy, 1008
 - to_value, 1009
- mln::value::proxy< I >, 1007
- mln::value::qt::rgb32, 1009
 - operator=, 1010
 - red, 1010
 - rgb32, 1010
 - zero, 1011
- mln::value::rgb
 - operator=, 1012

- red, [1012](#)
- rgb, [1012](#)
- zero, [1012](#)
- mln::value::rgb< n >, [1011](#)
- mln::value::set
 - the, [1013](#)
- mln::value::set< T >, [1012](#)
- mln::value::sign, [1013](#)
 - enc, [1014](#)
 - equiv, [1014](#)
 - one, [1015](#)
 - operator int, [1015](#)
 - operator=, [1015](#)
 - sign, [1014](#)
 - zero, [1015](#)
- mln::value::stack_image
 - domain_t, [1016](#)
 - is_valid, [1017](#)
 - lvalue, [1016](#)
 - operator(), [1017](#)
 - psite, [1016](#)
 - rvalue, [1016](#)
 - skeleton, [1016](#)
 - stack_image, [1017](#)
 - value, [1017](#)
- mln::value::stack_image< n, l >, [1015](#)
- mln::value::super_value< sign >, [1017](#)
- mln::value::value_array
 - operator(), [1018](#)
 - value_array, [1018](#)
 - vset, [1019](#)
- mln::value::value_array< T, V >, [1018](#)
- mln::vertex_image
 - graph_t, [1020](#)
 - nbh_t, [1020](#)
 - operator(), [1021](#)
 - site_function_t, [1020](#)
 - skeleton, [1020](#)
 - vertex_image, [1021](#)
 - win_t, [1020](#)
- mln::vertex_image< P, V, G >, [1019](#)
- mln::violent_cast_image
 - lvalue, [1022](#)
 - operator(), [1022](#)
 - rvalue, [1022](#)
 - skeleton, [1022](#)
 - value, [1022](#)
 - violent_cast_image, [1022](#)
- mln::violent_cast_image< T, l >, [1021](#)
- mln::w_window
 - bkd_qiter, [1024](#)
 - clear, [1025](#)
 - dpsite, [1024](#)
 - fwd_qiter, [1024](#)
 - insert, [1025](#)
 - is_symmetric, [1025](#)
 - operator<<, [1026](#)
 - operator==, [1026](#)
 - std_vector, [1025](#)
 - sym, [1025](#)
 - w, [1025](#)
 - w_window, [1024](#)
 - weight, [1024](#)
 - weights, [1025](#)
 - win, [1025](#)
- mln::w_window< D, W >, [1023](#)
- mln::win, [396](#)
 - diff, [397](#)
 - mln_regular, [397](#)
 - sym, [398](#)
- mln::win::backdiag2d, [1027](#)
 - backdiag2d, [1028](#)
 - length, [1028](#)
- mln::win::ball
 - ball, [1029](#)
 - diameter, [1029](#)
- mln::win::ball< G, C >, [1028](#)
- mln::win::cube3d, [1029](#)
 - cube3d, [1030](#)
 - length, [1030](#)
- mln::win::cuboid3d, [1030](#)
 - cuboid3d, [1031](#)
 - depth, [1031](#)
 - height, [1031](#)
 - volume, [1032](#)
 - width, [1032](#)
- mln::win::diag2d, [1032](#)
 - diag2d, [1032](#)
 - length, [1033](#)
- mln::win::line
 - length, [1034](#)
 - line, [1034](#)
 - size, [1034](#)
- mln::win::line< M, i, C >, [1033](#)
- mln::win::multiple< W, F >, [1034](#)
- mln::win::multiple_size< n, W, F >, [1035](#)
- mln::win::octagon2d, [1035](#)
 - area, [1036](#)
 - length, [1036](#)
 - octagon2d, [1036](#)
- mln::win::rectangle2d, [1036](#)
 - area, [1037](#)
 - height, [1037](#)
 - rectangle2d, [1037](#)
 - std_vector, [1037](#)
 - width, [1037](#)
- mln::window
 - bkd_qiter, [1040](#)
 - clear, [1040](#)
 - delta, [1040](#)
 - dp, [1041](#)
 - fwd_qiter, [1040](#)
 - has, [1041](#)
 - insert, [1041](#)
 - is_centered, [1041](#)
 - is_empty, [1041](#)

- is_symmetric, 1041
- operator==, 1042
- print, 1042
- qiter, 1040
- regular, 1040
- size, 1042
- std_vector, 1042
- sym, 1042
- window, 1040
- mln::window< D >, 1038
- mln::world::inter_pixel::is_separator, 1042
- mln_ch_convolve
 - mln::linear, 297, 298
- mln_ch_convolve_grad
 - mln::linear, 298
- mln_exact
 - mln, 158
- mln_gen_complex_neighborhood
 - mln, 158
- mln_gen_complex_window
 - mln, 159
- mln_gen_complex_window_p
 - mln, 159, 160
- mln_geom
 - mln::topo::is_simple_cell, 911
- mln_image_from_grid
 - mln::convert, 199
- mln_meta_accu_result
 - mln::accu, 169
 - mln::data, 207
 - mln::set, 364
- mln_q_subject
 - mln::p_graph_piter, 793
- mln_regular
 - mln, 160
 - mln::win, 397
- mln_trait_op_minus_twice
 - mln::morpho::elementary, 341
- mln_trait_op_neq
 - mln, 160
- mln_window
 - mln::convert, 199
- mosaic
 - mln::debug, 223
- Multiple accumulators, 95
- Multiple windows, 131
- n
 - mln::complex_psite, 543
 - mln::p_n_faces_bkd_piter, 814
 - mln::p_n_faces_fwd_piter, 815
 - mln::topo::algebraic_face, 891
 - mln::topo::algebraic_n_face, 895
 - mln::topo::face, 905
 - mln::topo::n_face, 914
 - mln::topo::n_face_bkd_iter, 915
 - mln::topo::n_face_fwd_iter, 916
- N-D windows, 130
- n_face
 - mln::topo::n_face, 913
- n_items
 - mln::accu::stat::var, 493
 - mln::accu::stat::variance, 495
- nbh_t
 - mln::edge_image, 600
 - mln::vertex_image, 1020
- nbits
 - mln::value::float01, 988
- ncols
 - mln::geom, 248
 - mln::image2d, 707
 - mln::image3d, 714
- nedges
 - mln::p_edges, 789
- neighb
 - mln::neighb, 772
- neighb1d
 - 1D neighborhoods, 109
- neighb2d
 - 2D neighborhoods, 110
- neighb3d
 - 3D neighborhoods, 112
- Neighborhoods, 108
- nelements
 - mln::doc::Fastest_Image, 560
 - mln::image1d, 703
 - mln::image2d, 707
 - mln::image3d, 714
 - mln::p_mutable_array_of, 813
 - mln::util::array, 932
 - mln::util::fibonacci_heap, 945
 - mln::util::set, 968
 - mln::util::soft_heap, 972
- next
 - mln::bkd_pixter1d, 504
 - mln::bkd_pixter2d, 505
 - mln::bkd_pixter3d, 507
 - mln::box_runend_piter, 519
 - mln::box_runstart_piter, 520
 - mln::complex_neighborhood_bkd_piter, 539
 - mln::complex_neighborhood_fwd_piter, 541
 - mln::complex_window_bkd_piter, 545
 - mln::complex_window_fwd_piter, 546
 - mln::dpoints_bkd_pixter, 594
 - mln::dpoints_fwd_pixter, 596
 - mln::dpsites_bkd_piter, 597
 - mln::dpsites_fwd_piter, 599
 - mln::fwd_pixter1d, 659
 - mln::fwd_pixter2d, 660
 - mln::fwd_pixter3d, 661
 - mln::graph_window_if_piter, 691
 - mln::graph_window_piter, 694
 - mln::iterator, 720
 - mln::p_graph_piter, 793
 - mln::p_indexed_bkd_piter, 801
 - mln::p_indexed_fwd_piter, 802
 - mln::p_n_faces_bkd_piter, 814

- mln::p_n_faces_fwd_iter, 815
- mln::p_transformed_iter, 839
- mln::Site_iterator, 866
- mln::topo::adj_higher_dim_connected_n_face_bkd_iter, 876
- mln::topo::adj_higher_dim_connected_n_face_fwd_iter, 877
- mln::topo::adj_higher_face_bkd_iter, 878
- mln::topo::adj_higher_face_fwd_iter, 879
- mln::topo::adj_lower_dim_connected_n_face_bkd_iter, 880
- mln::topo::adj_lower_dim_connected_n_face_fwd_iter, 881
- mln::topo::adj_lower_face_bkd_iter, 882
- mln::topo::adj_lower_face_fwd_iter, 883
- mln::topo::adj_lower_higher_face_bkd_iter, 884
- mln::topo::adj_lower_higher_face_fwd_iter, 885
- mln::topo::adj_m_face_bkd_iter, 886
- mln::topo::adj_m_face_fwd_iter, 888
- mln::topo::center_only_iter, 897
- mln::topo::centered_bkd_iter_adapter, 898
- mln::topo::centered_fwd_iter_adapter, 899
- mln::topo::face_bkd_iter, 906
- mln::topo::face_fwd_iter, 907
- mln::topo::n_face_bkd_iter, 915
- mln::topo::n_face_fwd_iter, 917
- mln::topo::static_n_face_bkd_iter, 920
- mln::topo::static_n_face_fwd_iter, 921
- mln::util::branch_iter, 935
- mln::util::branch_iter_ind, 937
- mln::value::int_u, 999
- mln::value::label, 1004
- nfaces
 - mln::p_complex, 784
 - mln::p_faces, 792
 - mln::topo::complex, 901
- nfaces_of_dim
 - mln::p_complex, 784
 - mln::topo::complex, 901
- nfaces_of_static_dim
 - mln::topo::complex, 901
- ninds
 - mln::geom, 249
 - mln::image1d, 703
- niter
 - mln::doc::Neighborhood, 571
 - mln::graph_elt_mixed_neighborhood, 672
 - mln::graph_elt_neighborhood, 677
 - mln::graph_elt_neighborhood_if, 679
 - mln::neighb, 771
- nlabels
 - mln::labeled_image, 723
 - mln::labeled_image_base, 726
- nmax_nbh_edges
 - mln::util::edge, 942
 - mln::util::vertex, 984
- nmax_nbh_vertices
 - mln::util::vertex, 984
- not_inplace
 - mln::logical, 307
- nrows
 - mln::geom, 249
 - mln::image2d, 707
 - mln::image3d, 714
- nsites
 - mln::accu::center, 405
 - mln::Box, 516
 - mln::box, 512
 - mln::doc::Box, 553
 - mln::doc::Fastest_Image, 560
 - mln::doc::Image, 568
 - mln::geom, 249
 - mln::p_array, 778
 - mln::p_complex, 785
 - mln::p_edges, 789
 - mln::p_faces, 792
 - mln::p_image, 799
 - mln::p_key, 806
 - mln::p_line2d, 810
 - mln::p_priority, 819
 - mln::p_queue, 823
 - mln::p_queue_fast, 827
 - mln::p_run, 831
 - mln::p_set, 835
 - mln::p_vertices, 847
- nslis
 - mln::geom, 249
 - mln::image3d, 714
- nvalues
 - mln::doc::Value_Set, 584
 - mln::value::lut_vec, 1007
- nvertices
 - mln::p_vertices, 848
- object_id
 - mln::util::object_id, 960
- octagon2d
 - mln::win::octagon2d, 1036
- olive
 - mln::literal, 303
- On images, 92
- On site sets, 91
- On values, 93
- one
 - mln::literal, 304
 - mln::value::int_s, 997
 - mln::value::int_u_sat, 1001
 - mln::value::sign, 1015
- opening
 - mln::morpho::elementary, 342
- operator bool
 - mln::util::tracked_ptr, 974
- operator decorated_image< const I, D >
 - mln::decorated_image, 548
- operator edge_id_t
 - mln::util::edge, 942
- operator float

- mln::value::float01, 988
- mln::value::float01_f, 990
- operator graylevel< n >
 - mln::value::graylevel_f, 995
- operator I
 - mln::plain, 851
- operator int
 - mln::value::int_s, 997
 - mln::value::int_u_sat, 1001
 - mln::value::sign, 1015
- operator mat< n, 1, U >
 - mln::algebra::h_vec, 503
- operator mln::algebra::vec< dpoint< G, C >::dim, Q >
 - mln::dpoint, 591
- operator psite
 - mln::doc::Site_Iterator, 578
- operator safe_image< const I >
 - mln::safe_image, 863
- operator sub_image< const I, S >
 - mln::sub_image, 872
- operator thru_image< const I, F >
 - mln::thru_image, 874
- operator thrubin_image< const I1, const I2, F >
 - mln::thrubin_image, 875
- operator transformed_image< const I, F >
 - mln::transformed_image, 925
- operator typename internal::p_image_site_set< I >::ret
 - mln::p_image, 799
- operator unsigned
 - mln::value::int_u, 999
 - mln::value::label, 1004
- operator util::tree_node< T > &
 - mln::util::branch_iter, 935
 - mln::util::branch_iter_ind, 937
- operator value
 - mln::doc::Value_Iterator, 582
- operator vertex_id_t
 - mln::util::vertex, 984
- operator<
 - mln, 162
 - mln::Box, 517
 - mln::Site_Set, 869
 - mln::topo, 373
 - mln::util, 384
- operator<<
 - mln, 163
 - mln::Box, 517
 - mln::box, 513
 - mln::fun::i2v, 236
 - mln::Gpoint, 668
 - mln::Site_Set, 869
 - mln::topo, 374
 - mln::util, 384
 - mln::value, 392–395
 - mln::w_window, 1026
- operator<=
 - mln, 163, 164
 - mln::Box, 517, 518
- mln::Site_Set, 870
- operator*
 - mln, 161
 - mln::algebra, 176
 - mln::value, 391, 392
- operator()
 - mln::complex_image, 538
 - mln::decorated_image, 548
 - mln::doc::Fastest_Image, 560, 561
 - mln::doc::Image, 568, 569
 - mln::edge_image, 601
 - mln::extension_fun, 604, 605
 - mln::extension_ima, 606
 - mln::extension_val, 608, 609
 - mln::flat_image, 611
 - mln::fun::v2v::rgb8_to_rgbn, 624
 - mln::fun::x2v::bilinear, 643
 - mln::fun::x2x::linear, 646
 - mln::fun::x2x::rotation, 649
 - mln::fun::x2x::translation, 651
 - mln::fun_image, 654
 - mln::geom::complex_geometry, 664
 - mln::hexa, 697
 - mln::image1d, 703
 - mln::image2d, 707
 - mln::image2d_h, 710
 - mln::image3d, 714
 - mln::lazy_image, 728, 729
 - mln::p2p_image, 775
 - mln::p_key, 806
 - mln::p_priority, 819
 - mln::p_vaccess, 842
 - mln::p_vertices, 848
 - mln::topo::is_simple_cell, 911
 - mln::tr_image, 923
 - mln::transformed_image, 925
 - mln::unproject_image, 926, 927
 - mln::util::array, 932
 - mln::util::greater_point, 951
 - mln::util::greater_psite, 952
 - mln::value::stack_image, 1017
 - mln::value::value_array, 1018
 - mln::vertex_image, 1021
 - mln::violent_cast_image, 1022
- operator+
 - mln::Gpoint, 666
 - mln::topo, 372
 - mln::value, 392
- operator++
 - mln, 161
 - mln::value::label, 1004
- operator+=
 - mln::Gpoint, 666
 - mln::Point, 852
 - mln::point, 857
 - mln::value::int_u_sat, 1001
- operator-
 - mln, 161

- mln::Gpoint, 667
- mln::topo, 372, 373
- mln::value, 392
- mln::value::int_u, 999
- mln::Weighted_Window, 1027
- operator->
 - mln::util::tracked_ptr, 974, 975
- operator--
 - mln, 162
 - mln::value::label, 1004
- operator==
 - mln::Gpoint, 667
 - mln::Point, 852
 - mln::point, 857
 - mln::value::int_u_sat, 1001
- operator/
 - mln::Gpoint, 668
 - mln::Point, 853
 - mln::value, 392
- operator=
 - mln::plain, 851
 - mln::util::fibonacci_heap, 945
 - mln::util::tracked_ptr, 975
 - mln::value::float01_f, 990
 - mln::value::graylevel, 992, 993
 - mln::value::graylevel_f, 995
 - mln::value::int_s, 997
 - mln::value::int_u, 999
 - mln::value::int_u_sat, 1001
 - mln::value::label, 1004
 - mln::value::proxy, 1009
 - mln::value::qt::rgb32, 1010
 - mln::value::rgb, 1012
 - mln::value::sign, 1015
- operator==
 - mln, 164, 165
 - mln::accu::stat, 174
 - mln::Box, 518
 - mln::Gpoint, 668
 - mln::Site_Set, 870
 - mln::topo, 374, 375
 - mln::util, 384, 385
 - mln::value, 395
 - mln::w_window, 1026
 - mln::window, 1042
- or_inplace
 - mln::logical, 307
- orange
 - mln::literal, 304
- ord_pair
 - mln::make, 322
- ord_strict
 - mln::util, 385
- ord_weak
 - mln::util, 385
- origin
 - mln::algebra::h_vec, 503
 - mln::literal, 304
- mln::point, 859
- other
 - mln::util::vertex, 984
 - mln::value, 395
- overset
 - mln::p_if, 796
- P
 - mln::graph_window_if_piter, 691
 - mln::graph_window_piter, 693
- p
 - mln::util::pix, 963
- p2p_image
 - mln::p2p_image, 774
- p_array
 - mln::p_array, 777
- p_centered
 - mln::p_centered, 781
- p_complex
 - mln::p_complex, 784
- p_edges
 - mln::p_edges, 787, 788
- p_edges_with_mass_centers
 - mln::make, 322
- p_faces
 - mln::p_faces, 791
- p_graph_piter
 - mln::p_graph_piter, 793
- p_if
 - mln::p_if, 795
- p_image
 - mln::p_image, 798
- p_indexed_bkd_piter
 - mln::p_indexed_bkd_piter, 800
- p_indexed_fwd_piter
 - mln::p_indexed_fwd_piter, 801
- p_key
 - mln::p_key, 805
- p_line2d
 - mln::p_line2d, 808
- p_mutable_array_of
 - mln::p_mutable_array_of, 812
- p_n_faces_bkd_piter
 - mln::p_n_faces_bkd_piter, 814
- p_n_faces_fwd_piter
 - mln::p_n_faces_fwd_piter, 815
- p_priority
 - mln::p_priority, 817
- p_queue
 - mln::p_queue, 822
- p_queue_fast
 - mln::p_queue_fast, 826
- p_run
 - mln::p_run, 830
- p_run2d
 - mln, 154
- p_runs2d
 - mln, 154
- p_set

- mln::p_set, 834
- p_transformed
 - mln::p_transformed, 837
- p_transformed_piter
 - mln::p_transformed_piter, 839
- p_vaccess
 - mln::p_vaccess, 842
- p_vertices
 - mln::p_vertices, 845, 846
- p_vertices_with_mass_centers
 - mln::make, 322
- pack
 - mln::labeling, 287, 288
- pack_inplace
 - mln::labeling, 288
- pair
 - mln::util::site_pair, 969
- parent
 - mln::util::tree_node, 980
- paste
 - mln::data, 207
 - mln::data::impl::generic, 217
- paste_without_localization
 - mln::data, 208
- paste_without_localization_fast
 - mln::data::impl, 214
- paste_without_localization_fastest
 - mln::data::impl, 214
- paste_without_localization_lines
 - mln::data::impl, 215
- pcenter
 - mln::box, 513
- pink
 - mln::literal, 304
- piter
 - mln::box, 510
 - mln::p_array, 777
 - mln::p_centered, 780
 - mln::p_complex, 783
 - mln::p_edges, 787
 - mln::p_faces, 791
 - mln::p_if, 795
 - mln::p_image, 798
 - mln::p_key, 804
 - mln::p_line2d, 808
 - mln::p_mutable_array_of, 812
 - mln::p_priority, 817
 - mln::p_queue, 822
 - mln::p_queue_fast, 826
 - mln::p_run, 830
 - mln::p_set, 833
 - mln::p_transformed, 837
 - mln::p_vaccess, 841
 - mln::p_vertices, 845
- pix
 - mln::make, 323
 - mln::util::pix, 963
- pixel
 - mln::make, 323
 - mln::pixel, 849
- plain
 - mln::plain, 850
- plot
 - mln::draw, 229
- plus
 - mln::arith, 183, 184
 - mln::morpho, 338
- plus_cst
 - mln::arith, 184, 185
- plus_cst_inplace
 - mln::arith, 186
- plus_infty
 - mln::point, 858
- plus_inplace
 - mln::arith, 186
- pmax
 - mln::box, 513
 - mln::doc::Box, 553
- pmin
 - mln::box, 513
 - mln::doc::Box, 553
- pmin_pmax
 - mln::geom, 249, 250
- point
 - mln::doc::Dpoint, 554
 - mln::doc::Fastest_Image, 558
 - mln::doc::Image, 566
 - mln::doc::Neighborhood, 571
 - mln::doc::Point_Site, 576
 - mln::doc::Weighted_Window, 586
 - mln::Point, 852
 - mln::point, 856, 857
- point< G, C >, 1043, 1044
- point1d
 - mln, 154
- point1df
 - mln, 154
- point2d
 - mln, 154
- point2d_h
 - mln, 154
 - mln::make, 323
- point2df
 - mln, 154
- point3d
 - mln, 154
- point3df
 - mln, 154
- point_at_index
 - mln::doc::Fastest_Image, 562
 - mln::image1d, 703
 - mln::image2d, 707
 - mln::image3d, 715
- polygon
 - mln::draw, 230
- pop

- mln::p_priority, 819
- mln::p_queue, 823
- mln::p_queue_fast, 827
- pop_front
 - mln::p_priority, 820
 - mln::p_queue, 823
 - mln::p_queue_fast, 827
 - mln::util::fibonacci_heap, 945
 - mln::util::soft_heap, 972
- positive
 - mln::test, 366
- pred
 - mln::p_if, 796
- predicate
 - mln::p_if, 796
 - mln::test, 366, 367
- prev
 - mln::value::label, 1004
- primary
 - mln, 166
- primary_set
 - mln::p_transformed, 838
- print
 - mln::topo::complex, 901
 - mln::util::tree_node, 980
 - mln::window, 1042
- print_faces
 - mln::topo::complex, 901
- println
 - mln::debug, 223
- println_with_border
 - mln::debug, 223
- priorities
 - mln::p_priority, 820
- propagate_if
 - mln::morpho::tree, 349
- propagate_if_value
 - mln::morpho::tree, 349
- propagate_node_to_ancestors
 - mln::morpho::tree, 349
- propagate_node_to_descendants
 - mln::morpho::tree, 350
- propagate_representative
 - mln::morpho::tree, 350
- proxy
 - mln::value::proxy, 1008
- pset
 - mln::doc::Fastest_Image, 558
 - mln::doc::Image, 566
 - mln::p_vaccess, 841
- psite
 - mln::box, 510
 - mln::complex_neighborhood_bkd_piter, 539
 - mln::complex_neighborhood_fwd_piter, 540
 - mln::complex_window_bkd_piter, 544
 - mln::complex_window_fwd_piter, 546
 - mln::decorated_image, 547
 - mln::doc::Box, 552
 - mln::doc::Fastest_Image, 558
 - mln::doc::Image, 566
 - mln::doc::Site_Iterator, 578
 - mln::doc::Site_Set, 580
 - mln::dpoint, 589
 - mln::graph_elt_mixed_window, 675
 - mln::graph_elt_window, 682
 - mln::graph_elt_window_if, 686
 - mln::hexa, 696
 - mln::image2d_h, 709
 - mln::interpolated, 716
 - mln::p_array, 777
 - mln::p_centered, 780
 - mln::p_complex, 783
 - mln::p_edges, 787
 - mln::p_faces, 791
 - mln::p_if, 795
 - mln::p_image, 798
 - mln::p_key, 804
 - mln::p_line2d, 808
 - mln::p_mutable_array_of, 812
 - mln::p_priority, 817
 - mln::p_queue, 822
 - mln::p_queue_fast, 826
 - mln::p_run, 830
 - mln::p_set, 834
 - mln::p_transformed, 837
 - mln::p_vaccess, 841
 - mln::p_vertices, 845
 - mln::thrubin_image, 875
 - mln::topo::is_simple_cell, 911
 - mln::tr_image, 922
 - mln::util::pix, 963
 - mln::value::stack_image, 1016
- ptransform
 - mln, 166
- purge
 - mln::p_queue_fast, 828
- purple
 - mln::literal, 304
- push
 - mln::p_priority, 820
 - mln::p_queue, 824
 - mln::p_queue_fast, 828
 - mln::util::fibonacci_heap, 945
 - mln::util::soft_heap, 972
- put_word
 - mln::debug, 223
- q_box
 - mln::p_line2d, 808
 - mln::p_run, 830
- qiter
 - mln::doc::Window, 587
 - mln::graph_elt_mixed_window, 675
 - mln::graph_elt_window, 682
 - mln::graph_elt_window_if, 686
 - mln::window, 1040
- Queue based, 120

- r_element
 - mln::p_image, 798
 - mln::p_key, 804
 - mln::p_set, 834
- rag_and_labeled_wsl
 - mln::make, 323
- rank_filter
 - mln::morpho, 338
- read_header
 - mln::io::fld, 262
- rectangle2d
 - mln::win::rectangle2d, 1037
- rectangularity
 - mln::accu::site_set::rectangularity, 471
- red
 - mln::literal, 304
 - mln::value::qt::rgb32, 1010
 - mln::value::rgb, 1012
- region_adjacency_graph
 - mln::make, 324
- regional_maxima
 - mln::labeling, 288
- regional_minima
 - mln::labeling, 288
- registration1
 - mln::registration, 361
- registration2
 - mln::registration, 361
- registration3
 - mln::registration, 361
- regular
 - mln::window, 1040
- relabel
 - mln::labeled_image, 723
 - mln::labeled_image_base, 726
 - mln::labeling, 289
- relabel_inplace
 - mln::labeling, 290
- relabelfun
 - mln::make, 324, 325
- remove
 - mln::p_image, 799
 - mln::p_key, 806
 - mln::p_set, 835
 - mln::util::set, 968
- remove_key
 - mln::p_key, 806
- replace
 - mln::data, 208
- reserve
 - mln::p_array, 779
 - mln::p_mutable_array_of, 813
 - mln::p_queue_fast, 828
 - mln::topo::n_faces_set, 918
 - mln::util::array, 932
- resize
 - mln::border, 192
 - mln::p_array, 779
 - mln::util::array, 933
- result
 - mln::graph::attribute::card_t, 670
 - mln::graph::attribute::representative_t, 670
 - mln::topo::is_simple_cell, 911
 - mln::util::array, 930
- revert
 - mln::arith, 186
- revert_inplace
 - mln::arith, 187
- rgb
 - mln::value::rgb, 1012
- rgb16
 - mln::value, 391
- rgb32
 - mln::value::qt::rgb32, 1010
- rgb8
 - mln::value, 391
- rgb8_2complex_image3df
 - mln, 155
- root
 - mln::util::tree, 977
- rotate
 - mln::geom, 250, 251
- rotation
 - mln::fun::x2x::rotation, 648
- Routines, 105
- run_length
 - mln::box_runend_piter, 519
 - mln::box_runstart_piter, 520
- rvalue
 - mln::complex_image, 537
 - mln::decorated_image, 548
 - mln::doc::Fastest_Image, 558
 - mln::doc::Generalized_Pixel, 563
 - mln::doc::Image, 567
 - mln::doc::Pixel_iterator, 574
 - mln::extension_fun, 604
 - mln::extension_ima, 606
 - mln::extension_val, 608
 - mln::flat_image, 610
 - mln::fun_image, 653
 - mln::hexa, 696
 - mln::image1d, 701
 - mln::image2d, 705
 - mln::image2d_h, 709
 - mln::image3d, 712
 - mln::interpolated, 716
 - mln::lazy_image, 728
 - mln::thruvin_image, 875
 - mln::tr_image, 922
 - mln::value::stack_image, 1016
 - mln::violent_cast_image, 1022
- S
 - mln::p_image, 798
- sagittal_dec
 - mln, 167
- saturate

- mln::data, 208
- saturate_inplace
 - mln::data, 209
- save
 - mln::io::cloud, 258
 - mln::io::dump, 260
 - mln::io::magick, 263
 - mln::io::off, 265
 - mln::io::pbm, 266
 - mln::io::pfm, 268
 - mln::io::pgm, 270
 - mln::io::plot, 271, 272
 - mln::io::pnm, 273
 - mln::io::ppm, 275
 - mln::io::raw, 277
 - mln::io::txt, 278
- save_bin_alt
 - mln::io::off, 265
- search
 - mln::util::tree_node, 980
- search_rec
 - mln::util::tree_node, 980
- second
 - mln::accu::pair, 463
 - mln::accu::stat::min_max, 487
 - mln::util::couple, 938
 - mln::util::ord_pair, 962
 - mln::util::site_pair, 970
- second_accu
 - mln::accu::pair, 463
 - mln::accu::stat::min_max, 487
- seeds2tiling
 - mln::geom, 251
 - mln::geom::impl, 253
- segment1d
 - 1D windows, 123
- set
 - mln::util::set, 966
- set_all
 - mln::dpoint, 591
 - mln::point, 858
- set_alpha
 - mln::fun::x2x::rotation, 649
- set_axis
 - mln::fun::x2x::rotation, 649
- set_cplx
 - mln::topo::algebraic_face, 891
 - mln::topo::algebraic_n_face, 895
 - mln::topo::face, 905
 - mln::topo::n_face, 914
- set_face_id
 - mln::topo::algebraic_face, 891
 - mln::topo::algebraic_n_face, 895
 - mln::topo::face, 905
 - mln::topo::n_face, 915
- set_image
 - mln::topo::is_simple_cell, 911
- set_n
 - mln::topo::algebraic_face, 892
 - mln::topo::face, 905
- set_nbits
 - mln::value::float01, 988
- set_parent
 - mln::util::tree_node, 981
- set_sign
 - mln::topo::algebraic_face, 892
 - mln::topo::algebraic_n_face, 895
- set_t
 - mln::fun::x2x::translation, 652
- set_tr
 - mln::tr_image, 924
- set_value
 - mln::accu::count_adjacent_vertices, 407
 - mln::accu::count_labels, 409
 - mln::accu::count_value, 410
 - mln::accu::math::count, 420
 - mln::accu::shape::height, 468
 - mln::accu::shape::volume, 469
 - mln::accu::stat::max, 476
 - mln::accu::stat::min, 484
 - mln::morpho::attribute::sum, 768
- sign
 - mln::topo::algebraic_face, 892
 - mln::topo::algebraic_n_face, 896
 - mln::value::sign, 1014
- site
 - mln::box, 510
 - mln::doc::Box, 552
 - mln::doc::Site_Set, 580
 - mln::dpoint, 589
 - mln::graph_elt_mixed_window, 675
 - mln::graph_elt_window, 682
 - mln::graph_elt_window_if, 686
 - mln::graph_window_base, 689
 - mln::p_centered, 781
 - mln::tr_image, 922
- Site sets, 115
- site_function_t
 - mln::edge_image, 601
 - mln::vertex_image, 1020
- site_set
 - mln::complex_psite, 543
 - mln::draw, 230
- site_set_impl< Sc >, 1044
- size
 - mln::util::array, 933
 - mln::win::line, 1034
 - mln::window, 1042
- skeleton
 - mln::complex_image, 537
 - mln::decorated_image, 548
 - mln::doc::Fastest_Image, 558
 - mln::doc::Image, 567
 - mln::edge_image, 601
 - mln::extended, 602
 - mln::extension_fun, 604

- mln::extension_ima, 606
- mln::extension_val, 608
- mln::flat_image, 610
- mln::fun_image, 653
- mln::hexa, 697
- mln::image1d, 701
- mln::image2d, 705
- mln::image2d_h, 709
- mln::image3d, 712
- mln::interpolated, 716
- mln::labeled_image, 722
- mln::lazy_image, 728
- mln::p2p_image, 774
- mln::plain, 850
- mln::pw::image, 861
- mln::safe_image, 863
- mln::sub_image, 872
- mln::sub_image_if, 873
- mln::thruin_image, 875
- mln::tr_image, 923
- mln::transformed_image, 925
- mln::value::stack_image, 1016
- mln::vertex_image, 1020
- mln::violent_cast_image, 1022
- slices_2d
 - mln::debug, 224
- sline3d
 - 3D windows, 127
- soft_heap
 - mln::util::soft_heap, 971
- sort_offsets_increasing
 - mln::data, 209
- sort_psites_decreasing
 - mln::data, 209
- sort_psites_increasing
 - mln::data, 209
- space_2complex_geometry
 - mln, 155
- Sparse types, 119
- sphere3d
 - 3D windows, 128
- sqr_l2
 - mln::norm, 357
- stack
 - mln::value, 395
- stack_image
 - mln::value::stack_image, 1017
- standard_deviation
 - mln::accu::stat::variance, 495
- start
 - mln::doc::Iterator, 570
 - mln::doc::Pixel_Iterator, 574
 - mln::doc::Site_Iterator, 578
 - mln::doc::Value_Iterator, 582
 - mln::dpoints_bkd_pixter, 594
 - mln::dpoints_fwd_pixter, 596
 - mln::p_run, 832
 - mln::topo::face_bkd_iter, 906
 - mln::topo::face_fwd_iter, 907
 - mln::topo::n_face_bkd_iter, 916
 - mln::topo::n_face_fwd_iter, 917
 - mln::topo::static_n_face_bkd_iter, 920
 - mln::topo::static_n_face_fwd_iter, 921
 - mln::util::branch_iter, 935
 - mln::util::branch_iter_ind, 937
- static_n_face_bkd_iter
 - mln::topo::static_n_face_bkd_iter, 919
- static_n_face_fwd_iter
 - mln::topo::static_n_face_fwd_iter, 921
- std_deque
 - mln::p_queue, 824
- std_vector
 - mln::p_array, 779
 - mln::p_line2d, 810
 - mln::p_queue_fast, 828
 - mln::p_set, 835
 - mln::util::array, 933
 - mln::util::set, 968
 - mln::w_window, 1025
 - mln::win::rectangle2d, 1037
 - mln::window, 1042
- stretch
 - mln::data, 210
 - mln::data::impl, 215
- structural
 - mln::morpho::closing::approx, 341
 - mln::morpho::opening::approx, 343
- sub_image
 - mln::sub_image, 872
- sub_image_if
 - mln::sub_image_if, 873
- subdomain
 - mln::labeled_image, 723
 - mln::labeled_image_base, 726
- subject_point_impl< P, E >, 1044
- subsampling
 - mln::subsampling, 365
- subtractive
 - mln::morpho::tree::filter, 352
- sum
 - mln::accu::stat::mean, 478
 - mln::accu::stat::variance, 495
 - mln::estim, 232
- superpose
 - mln::debug, 224
 - mln::labeling, 290
 - mln::morpho::watershed, 354
- sym
 - mln::doc::Weighted_Window, 586
 - mln::graph_elt_mixed_window, 676
 - mln::graph_elt_window, 683
 - mln::graph_elt_window_if, 688
 - mln::graph_window_base, 690
 - mln::w_window, 1025
 - mln::win, 398
 - mln::window, 1042

- sym_diff
 - mln::Box, 518
 - mln::Site_Set, 870
- t
 - mln::algebra::h_mat, 501
 - mln::algebra::h_vec, 503
 - mln::fun::x2x::translation, 652
- take
 - mln::accu, 170
 - mln::accu::histo, 411
 - mln::accu::label_used, 413
 - mln::accu::stat::histo3d_rgb, 474
 - mln::accu::stat::median_alt, 480
 - mln::doc::Accumulator, 550
- take_as_init
 - mln::accu::center, 405
 - mln::accu::convolve, 406
 - mln::accu::count_adjacent_vertices, 408
 - mln::accu::count_labels, 409
 - mln::accu::count_value, 410
 - mln::accu::histo, 412
 - mln::accu::label_used, 413
 - mln::accu::logic::land, 414
 - mln::accu::logic::land_basic, 415
 - mln::accu::logic::lor, 417
 - mln::accu::logic::lor_basic, 418
 - mln::accu::maj_h, 419
 - mln::accu::math::count, 420
 - mln::accu::math::inf, 422
 - mln::accu::math::sum, 423
 - mln::accu::math::sup, 424
 - mln::accu::max_site, 425
 - mln::accu::nil, 460
 - mln::accu::p, 461
 - mln::accu::pair, 463
 - mln::accu::rms, 465
 - mln::accu::shape::bbox, 466
 - mln::accu::shape::height, 468
 - mln::accu::shape::volume, 470
 - mln::accu::site_set::rectangularity, 471
 - mln::accu::stat::deviation, 472
 - mln::accu::stat::histo3d_rgb, 474
 - mln::accu::stat::max, 476
 - mln::accu::stat::max_h, 477
 - mln::accu::stat::mean, 479
 - mln::accu::stat::median_alt, 480
 - mln::accu::stat::median_h, 482
 - mln::accu::stat::min, 484
 - mln::accu::stat::min_h, 485
 - mln::accu::stat::min_max, 487
 - mln::accu::stat::rank, 489
 - mln::accu::stat::rank< bool >, 490
 - mln::accu::stat::rank_high_quant, 491
 - mln::accu::stat::var, 493
 - mln::accu::tuple, 497
 - mln::accu::val, 498
 - mln::Accumulator, 500
 - mln::morpho::attribute::card, 763
 - mln::morpho::attribute::count_adjacent_vertices, 764
 - mln::morpho::attribute::height, 765
 - mln::morpho::attribute::sharpness, 767
 - mln::morpho::attribute::sum, 769
 - mln::morpho::attribute::volume, 770
- take_n_times
 - mln::accu::center, 405
 - mln::accu::convolve, 406
 - mln::accu::count_adjacent_vertices, 408
 - mln::accu::count_labels, 409
 - mln::accu::count_value, 410
 - mln::accu::histo, 412
 - mln::accu::label_used, 413
 - mln::accu::logic::land, 414
 - mln::accu::logic::land_basic, 416
 - mln::accu::logic::lor, 417
 - mln::accu::logic::lor_basic, 418
 - mln::accu::maj_h, 419
 - mln::accu::math::count, 421
 - mln::accu::math::inf, 422
 - mln::accu::math::sum, 423
 - mln::accu::math::sup, 424
 - mln::accu::max_site, 425
 - mln::accu::nil, 460
 - mln::accu::p, 461
 - mln::accu::pair, 464
 - mln::accu::rms, 465
 - mln::accu::shape::bbox, 466
 - mln::accu::shape::height, 468
 - mln::accu::shape::volume, 470
 - mln::accu::site_set::rectangularity, 471
 - mln::accu::stat::deviation, 472
 - mln::accu::stat::histo3d_rgb, 475
 - mln::accu::stat::max, 476
 - mln::accu::stat::max_h, 477
 - mln::accu::stat::mean, 479
 - mln::accu::stat::median_alt, 480
 - mln::accu::stat::median_h, 482
 - mln::accu::stat::min, 484
 - mln::accu::stat::min_h, 485
 - mln::accu::stat::min_max, 488
 - mln::accu::stat::rank, 489
 - mln::accu::stat::rank< bool >, 490
 - mln::accu::stat::rank_high_quant, 491
 - mln::accu::stat::var, 493
 - mln::accu::stat::variance, 495
 - mln::accu::tuple, 497
 - mln::accu::val, 498
 - mln::Accumulator, 500
 - mln::morpho::attribute::card, 763
 - mln::morpho::attribute::count_adjacent_vertices, 764
 - mln::morpho::attribute::height, 765
 - mln::morpho::attribute::sharpness, 767
 - mln::morpho::attribute::sum, 769
 - mln::morpho::attribute::volume, 770
- target

- mln::graph_elt_mixed_window, 675
- mln::graph_elt_window, 682
- mln::graph_elt_window_if, 686
- target_site_set
 - mln::graph_window_piter, 694
- teal
 - mln::literal, 304
- the
 - mln::value::set, 1013
- thick_miss
 - mln::morpho, 338
- thickening
 - mln::morpho, 338
- thin_fit
 - mln::morpho, 338
- thinning
 - mln::morpho, 339
- threshold
 - mln::binarization, 189
- times
 - mln::arith, 187
- times_cst
 - mln::arith, 187
- times_inplace
 - mln::arith, 188
- to
 - mln::convert, 199
- to_dpoint
 - mln::convert, 199
 - mln::Dpoint, 592
- to_enc
 - mln::data, 210
- to_float
 - mln::value::graylevel, 993
- to_fun
 - mln::convert, 200, 202
- to_h_vec
 - mln::point, 858
- to_image
 - mln::convert, 200
- to_larger
 - mln::box, 513
- to_nbits
 - mln::value::float01, 989
- to_neighb
 - mln::graph, 254
- to_p_array
 - mln::convert, 200
- to_p_set
 - mln::convert, 200, 201
- to_point
 - mln::doc::Point_Site, 577
 - mln::Point, 852
- to_qimage
 - mln::convert, 201
- to_result
 - mln::accu::center, 405
 - mln::accu::convolve, 406
 - mln::accu::count_adjacent_vertices, 408
 - mln::accu::count_labels, 409
 - mln::accu::count_value, 411
 - mln::accu::label_used, 413
 - mln::accu::logic::land, 414
 - mln::accu::logic::land_basic, 416
 - mln::accu::logic::lor, 417
 - mln::accu::logic::lor_basic, 418
 - mln::accu::maj_h, 419
 - mln::accu::math::count, 421
 - mln::accu::math::inf, 422
 - mln::accu::math::sum, 423
 - mln::accu::math::sup, 424
 - mln::accu::max_site, 425
 - mln::accu::nil, 460
 - mln::accu::p, 461
 - mln::accu::pair, 464
 - mln::accu::rms, 465
 - mln::accu::shape::bbox, 466
 - mln::accu::shape::height, 468
 - mln::accu::shape::volume, 470
 - mln::accu::site_set::rectangularity, 471
 - mln::accu::stat::deviation, 473
 - mln::accu::stat::histo3d_rgb, 475
 - mln::accu::stat::max, 476
 - mln::accu::stat::max_h, 477
 - mln::accu::stat::mean, 479
 - mln::accu::stat::median_alt, 480
 - mln::accu::stat::median_h, 482
 - mln::accu::stat::min, 484
 - mln::accu::stat::min_h, 485
 - mln::accu::stat::min_max, 488
 - mln::accu::stat::rank, 489
 - mln::accu::stat::rank< bool >, 490
 - mln::accu::stat::rank_high_quant, 492
 - mln::accu::stat::var, 493
 - mln::accu::stat::variance, 495
 - mln::accu::tuple, 497
 - mln::accu::val, 498
 - mln::morpho::attribute::card, 763
 - mln::morpho::attribute::count_adjacent_vertices, 764
 - mln::morpho::attribute::height, 766
 - mln::morpho::attribute::sharpness, 767
 - mln::morpho::attribute::sum, 769
 - mln::morpho::attribute::volume, 770
- to_upper_window
 - mln::convert, 201
- to_value
 - mln::value::proxy, 1009
- to_vec
 - mln::algebra::h_vec, 503
 - mln::dpoint, 591
 - mln::point, 858
- to_win
 - mln::graph, 255
- to_window
 - mln::convert, 201, 202

- toggle
 - mln::p_image, 800
- top_hat_black
 - mln::morpho, 339
 - mln::morpho::elementary, 342
- top_hat_self_complementary
 - mln::morpho, 339
 - mln::morpho::elementary, 342
- top_hat_white
 - mln::morpho, 339
 - mln::morpho::elementary, 342
- topological
 - mln::morpho::watershed, 354
- tr
 - mln::algebra, 176
 - mln::tr_image, 924
- tr_image
 - mln::tr_image, 923
- tracked_ptr
 - mln::util::tracked_ptr, 974
- trait::graph< I >, 1045
- trait::graph< mln::complex_image< 1, G, V > >, 1045
- trait::graph< mln::image2d< T > >, 1046
- transform
 - mln::data, 210
 - mln::data::impl::generic, 218
- transform_inplace
 - mln::data, 211
 - mln::data::impl::generic, 218
- transform_inplace_lowq
 - mln::data::impl, 216
- transformed_image
 - mln::transformed_image, 925
- translate
 - mln::geom, 251, 252
- translation
 - mln::fun::x2x::translation, 651
- tree
 - mln::util::tree, 976
- tree_fast_to_image
 - mln::util, 385
- tree_node
 - mln::util::tree_node, 978
- tree_node< T >, 1046
- tree_to_fast
 - mln::util, 385
- tree_to_image
 - mln::util, 385
- Types, 103
- uni
 - mln::Box, 518
 - mln::Site_Set, 870
- unique
 - mln::Box, 518
 - mln::Site_Set, 870
- unproject_image
 - mln::unproject_image, 926
- unsigned_2complex_image3df
 - mln, 155
- untake
 - mln::morpho::attribute::sum, 769
- up
 - mln, 167
- up_leaf_piter< T >, 1046
- up_node_piter< T >, 1046
- up_site_piter< T >, 1047
- update
 - mln::data, 211
 - mln::data::impl::generic, 218
 - mln::dpoints_bkd_pixter, 594
 - mln::dpoints_fwd_pixter, 596
- update_data
 - mln::labeled_image, 723
 - mln::labeled_image_base, 726
- update_fastest
 - mln::data::impl, 216
- update_id
 - mln::util::edge, 942
 - mln::util::vertex, 985
- util_set
 - mln::p_set, 835
- util_tree
 - mln::util::branch, 934
- Utilities, 121
- v
 - mln::util::pix, 963
- v1
 - mln::util::edge, 942
 - mln::util::graph, 950
 - mln::util::line_graph, 957
- v2
 - mln::util::edge, 942
 - mln::util::graph, 950
 - mln::util::line_graph, 957
- v2w2v functions, 132
- v2w_w2v functions, 133
- v_ith_nbh_edge
 - mln::util::graph, 950
 - mln::util::line_graph, 957
- v_ith_nbh_vertex
 - mln::util::graph, 950
 - mln::util::line_graph, 957
- v_nmax
 - mln::util::graph, 950
 - mln::util::line_graph, 957
- v_other
 - mln::util::edge, 942
- val
 - mln::doc::Generalized_Pixel, 563
 - mln::doc::Pixel_Iterator, 574
- value
 - mln::accu::shape::height, 467
 - mln::accu::shape::volume, 469
 - mln::complex_image, 537
 - mln::doc::Fastest_Image, 558
 - mln::doc::Generalized_Pixel, 563

- mln::doc::Image, 567
- mln::doc::Pixel_iterator, 574
- mln::doc::Value_iterator, 582
- mln::doc::Value_Set, 583
- mln::extended, 602
- mln::extension_fun, 604
- mln::extension_ima, 606
- mln::extension_val, 608
- mln::flat_image, 610
- mln::fun_image, 653
- mln::hexa, 697
- mln::image1d, 701
- mln::image2d, 705
- mln::image2d_h, 709
- mln::image3d, 712
- mln::interpolated, 716
- mln::labeling, 291
- mln::p_vaccess, 841
- mln::thruin_image, 875
- mln::tr_image, 923
- mln::util::pix, 963
- mln::value::float01, 989
- mln::value::float01_f, 990
- mln::value::graylevel, 993
- mln::value::graylevel_f, 995
- mln::value::lut_vec, 1006
- mln::value::stack_image, 1017
- mln::violent_cast_image, 1022
- value_and_compute
 - mln::labeling, 291
- value_array
 - mln::value::value_array, 1018
- value_ind
 - mln::value::float01, 989
- value_t
 - mln::util::object_id, 959
- values
 - mln::complex_image, 538
 - mln::doc::Fastest_Image, 562
 - mln::doc::Image, 569
 - mln::p_vaccess, 842
- Values morphers, 100
- var
 - mln::accu::stat::variance, 496
- variance
 - mln::accu::stat::var, 494
- vbbox
 - mln::image1d, 703
 - mln::image3d, 715
- vec
 - mln::dpoint, 590
 - mln::make, 325, 326
 - mln::point, 856
- vec2d_d
 - mln, 155
- vec2d_f
 - mln, 155
- vec3d_d
 - mln, 155
- vec3d_f
 - mln, 155
- vect
 - mln::accu::histo, 412
- vertex
 - mln::p_vertices, 845
 - mln::util::graph, 951
 - mln::util::line_graph, 957
 - mln::util::vertex, 983
- vertex_id_t
 - mln::util, 383
- vertex_image
 - mln::make, 326, 327
 - mln::vertex_image, 1021
- vertex_nbh_edge_fwd_iter
 - mln::util::graph, 947
 - mln::util::line_graph, 955
- vertex_nbh_vertex_fwd_iter
 - mln::util::graph, 948
 - mln::util::line_graph, 955
- vertical_symmetry
 - mln::geom, 252
- vertices_t
 - mln::util::graph, 948
 - mln::util::line_graph, 955
- violent_cast_image
 - mln::violent_cast_image, 1022
- violet
 - mln::literal, 304
- vline2d
 - 2D windows, 125
- volume
 - mln::morpho::attribute::sharpness, 767
 - mln::win::cuboid3d, 1032
- voronoi
 - mln::make, 327
- vprod
 - mln::algebra, 176
- vset
 - mln::doc::Fastest_Image, 558
 - mln::doc::Image, 567
 - mln::p_vaccess, 841
 - mln::value::value_array, 1019
- vv2b functions, 134
- w
 - mln::w_window, 1025
- w_window
 - mln::make, 327
 - mln::w_window, 1024
- w_window1d
 - mln::make, 328
- w_window2d
 - mln::make, 328
- w_window3d
 - mln::make, 328
- w_window_directional
 - mln::make, 329

- weight
 - mln::doc::Weighted_Window, 586
 - mln::w_window, 1024
- weights
 - mln::w_window, 1025
- white
 - mln::literal, 305
- width
 - mln::win::cuboid3d, 1032
 - mln::win::rectangle2d, 1037
- win
 - mln::doc::Weighted_Window, 586
 - mln::w_window, 1025
- win_c4p
 - 2D windows, 125
- win_c4p_3d
 - 3D windows, 128
- win_c8p
 - 2D windows, 125
- win_c8p_3d
 - 3D windows, 128
- win_t
 - mln::edge_image, 601
 - mln::vertex_image, 1020
- window
 - mln::doc::Weighted_Window, 586
 - mln::p_centered, 781
 - mln::window, 1040
- window1d
 - 1D windows, 123
- window2d
 - 2D windows, 125
- window3d
 - 3D windows, 128
- Windows, 122
- wrap
 - mln::data, 211
 - mln::labeling, 291, 292
- write_header
 - mln::io::fld, 262
- xor_inplace
 - mln::logical, 307
- yellow
 - mln::literal, 305
- z_order
 - mln::debug, 225
- zero
 - mln::algebra::h_vec, 503
 - mln::literal, 305
 - mln::value::int_s, 997
 - mln::value::int_u_sat, 1001
 - mln::value::qt::rgb32, 1011
 - mln::value::rgb, 1012
 - mln::value::sign, 1015