

# FreeLing Reference Manual

## 1.4

Generated by Doxygen 1.4.4

Wed Apr 26 12:55:30 2006



# Contents

<b>1</b>	<b>FreeLing Directory Hierarchy</b>	<b>1</b>
1.1	FreeLing Directories . . . . .	1
<b>2</b>	<b>FreeLing Namespace Index</b>	<b>3</b>
2.1	FreeLing Namespace List . . . . .	3
<b>3</b>	<b>FreeLing Hierarchical Index</b>	<b>5</b>
3.1	FreeLing Class Hierarchy . . . . .	5
<b>4</b>	<b>FreeLing Class Index</b>	<b>9</b>
4.1	FreeLing Class List . . . . .	9
<b>5</b>	<b>FreeLing File Index</b>	<b>13</b>
5.1	FreeLing File List . . . . .	13
<b>6</b>	<b>FreeLing Directory Documentation</b>	<b>15</b>
6.1	/home/padro/freeling-1.4/src/include/ Directory Reference . . . . .	15
6.2	/home/padro/freeling-1.4/src/libmorfo/ Directory Reference . . . . .	17
6.3	/home/padro/freeling-1.4/src/ Directory Reference . . . . .	18
<b>7</b>	<b>FreeLing Namespace Documentation</b>	<b>19</b>
7.1	std Namespace Reference . . . . .	19
<b>8</b>	<b>FreeLing Class Documentation</b>	<b>21</b>
8.1	accents Class Reference . . . . .	21
8.2	accents_default Class Reference . . . . .	23
8.3	accents_es Class Reference . . . . .	25
8.4	accents_module Class Reference . . . . .	28
8.5	adaboost Class Reference . . . . .	30
8.6	adaboost::wr_holder Struct Reference . . . . .	34
8.7	analysis Class Reference . . . . .	35

8.8	automat Class Reference . . . . .	39
8.9	cell Class Reference . . . . .	43
8.10	chart Class Reference . . . . .	44
8.11	chart_parser Class Reference . . . . .	48
8.12	check_A_lemma Class Reference . . . . .	50
8.13	check_A_wordclass Class Reference . . . . .	52
8.14	check_and Class Reference . . . . .	54
8.15	check_not Class Reference . . . . .	56
8.16	check_S_category Class Reference . . . . .	58
8.17	check_S_lemma Class Reference . . . . .	60
8.18	check_side Class Reference . . . . .	62
8.19	completer Class Reference . . . . .	64
8.20	completerRule Class Reference . . . . .	66
8.21	condition Class Reference . . . . .	68
8.22	constraint Class Reference . . . . .	72
8.23	constraint_grammar Class Reference . . . . .	74
8.24	dataset Class Reference . . . . .	75
8.25	dataset::iterator Class Reference . . . . .	78
8.26	dataset::mlDatasetNode Struct Reference . . . . .	80
8.27	dates Class Reference . . . . .	81
8.28	dates_ca Class Reference . . . . .	83
8.29	dates_default Class Reference . . . . .	85
8.30	dates_es Class Reference . . . . .	87
8.31	dates_module Class Reference . . . . .	89
8.32	dep_info Class Reference . . . . .	92
8.33	dep_tree Class Reference . . . . .	94
8.34	dependencyMaker Class Reference . . . . .	96
8.35	depLabeler Class Reference . . . . .	99
8.36	depnoder Class Reference . . . . .	101
8.37	dictionary Class Reference . . . . .	104
8.38	document Class Reference . . . . .	107
8.39	edge Class Reference . . . . .	108
8.40	emission_states Class Reference . . . . .	110
8.41	example Class Reference . . . . .	111
8.42	fex Class Reference . . . . .	113
8.43	FlexLexer Class Reference . . . . .	114

8.44 grammar Class Reference . . . . .	116
8.45 hmm_tagger Class Reference . . . . .	121
8.46 iFeature Class Reference . . . . .	124
8.47 label Class Reference . . . . .	125
8.48 list Class Reference . . . . .	127
8.49 locutions Class Reference . . . . .	128
8.50 maco Class Reference . . . . .	131
8.51 maco_options Class Reference . . . . .	133
8.52 map_input Class Reference . . . . .	136
8.53 map_input::const_iterator Class Reference . . . . .	139
8.54 mlABTree Class Reference . . . . .	140
8.55 mlOutput Class Reference . . . . .	144
8.56 mlOutput::label Struct Reference . . . . .	147
8.57 multimap Class Reference . . . . .	148
8.58 nec Class Reference . . . . .	149
8.59 node Class Reference . . . . .	151
8.60 np Class Reference . . . . .	155
8.61 numbers Class Reference . . . . .	159
8.62 numbers_ca Class Reference . . . . .	161
8.63 numbers_default Class Reference . . . . .	163
8.64 numbers_en Class Reference . . . . .	165
8.65 numbers_es Class Reference . . . . .	167
8.66 numbers_gl Class Reference . . . . .	169
8.67 numbers_module Class Reference . . . . .	171
8.68 paragraph Class Reference . . . . .	174
8.69 parse_tree Class Reference . . . . .	175
8.70 POS_tagger Class Reference . . . . .	177
8.71 probabilities Class Reference . . . . .	179
8.72 problem Class Reference . . . . .	182
8.73 punts Class Reference . . . . .	184
8.74 quantities Class Reference . . . . .	185
8.75 quantities_default Class Reference . . . . .	187
8.76 quantities_es Class Reference . . . . .	189
8.77 quantities_module Class Reference . . . . .	191
8.78 RegEx Class Reference . . . . .	194
8.79 relax Class Reference . . . . .	198

8.80 relax_tagger Class Reference . . . . .	201
8.81 RGF Struct Reference . . . . .	204
8.82 rule Class Reference . . . . .	208
8.83 rule_expression Class Reference . . . . .	211
8.84 ruleCG Class Reference . . . . .	213
8.85 ruleLabeler Class Reference . . . . .	215
8.86 senses Class Reference . . . . .	217
8.87 Sensor Class Reference . . . . .	219
8.88 SensorCheckMwRE Class Reference . . . . .	221
8.89 SensorCheckRE Class Reference . . . . .	223
8.90 SensorData Class Reference . . . . .	225
8.91 SensorMap Class Reference . . . . .	226
8.92 SensorMatchRE Class Reference . . . . .	228
8.93 SensorSet Class Reference . . . . .	230
8.94 SensorSetPart Class Reference . . . . .	232
8.95 sentence Class Reference . . . . .	234
8.96 set_input Class Reference . . . . .	237
8.97 splitter Class Reference . . . . .	241
8.98 suffixes Class Reference . . . . .	244
8.99 sufrule Class Reference . . . . .	247
8.100tokenizer Class Reference . . . . .	248
8.101traces Class Reference . . . . .	250
8.102tree< T > Class Template Reference . . . . .	252
8.103tree< T >::generic_iterator Class Reference . . . . .	256
8.104tree< T >::preorder_iterator Class Reference . . . . .	258
8.105tree< T >::sibling_iterator Class Reference . . . . .	260
8.106util Class Reference . . . . .	262
8.107vector Class Reference . . . . .	265
8.108viterbi Class Reference . . . . .	266
8.109word Class Reference . . . . .	268
8.110yyFlexLexer Class Reference . . . . .	275
<b>9 FreeLing File Documentation</b>	<b>281</b>
9.1 accents.cc File Reference . . . . .	281
9.2 accents.h File Reference . . . . .	283
9.3 accents_modules.cc File Reference . . . . .	284
9.4 accents_modules.h File Reference . . . . .	285

9.5	adaboost.h File Reference . . . . .	286
9.6	automat.cc File Reference . . . . .	287
9.7	automat.h File Reference . . . . .	288
9.8	chart.h File Reference . . . . .	290
9.9	chart_parser.h File Reference . . . . .	291
9.10	constraint_grammar.h File Reference . . . . .	292
9.11	dataset.h File Reference . . . . .	293
9.12	dates.cc File Reference . . . . .	294
9.13	dates.h File Reference . . . . .	295
9.14	dates_modules.cc File Reference . . . . .	296
9.15	dates_modules.h File Reference . . . . .	302
9.16	dep_rules.cc File Reference . . . . .	305
9.17	dep_rules.h File Reference . . . . .	306
9.18	dependencies.cc File Reference . . . . .	308
9.19	dependencies.h File Reference . . . . .	310
9.20	dictionary.cc File Reference . . . . .	311
9.21	dictionary.h File Reference . . . . .	313
9.22	example.h File Reference . . . . .	314
9.23	fex.h File Reference . . . . .	315
9.24	FlexLexer.h File Reference . . . . .	316
9.25	grammar.h File Reference . . . . .	317
9.26	hmm_tagger.cc File Reference . . . . .	318
9.27	hmm_tagger.h File Reference . . . . .	319
9.28	language.cc File Reference . . . . .	320
9.29	language.h File Reference . . . . .	321
9.30	locutions.cc File Reference . . . . .	324
9.31	locutions.h File Reference . . . . .	326
9.32	maco.cc File Reference . . . . .	327
9.33	maco.h File Reference . . . . .	328
9.34	maco_options.cc File Reference . . . . .	329
9.35	maco_options.h File Reference . . . . .	330
9.36	nec.cc File Reference . . . . .	331
9.37	nec.h File Reference . . . . .	332
9.38	np.cc File Reference . . . . .	334
9.39	np.h File Reference . . . . .	336
9.40	numbers.cc File Reference . . . . .	337

9.41	numbers.h File Reference . . . . .	338
9.42	numbers_modules.cc File Reference . . . . .	339
9.43	numbers_modules.h File Reference . . . . .	345
9.44	probabilities.cc File Reference . . . . .	347
9.45	probabilities.h File Reference . . . . .	348
9.46	punts.cc File Reference . . . . .	349
9.47	punts.h File Reference . . . . .	350
9.48	quantities.cc File Reference . . . . .	351
9.49	quantities.h File Reference . . . . .	352
9.50	quantities_modules.cc File Reference . . . . .	353
9.51	quantities_modules.h File Reference . . . . .	356
9.52	readFeatures.h File Reference . . . . .	357
9.53	regexp.h File Reference . . . . .	359
9.54	relax.h File Reference . . . . .	360
9.55	relax_tagger.h File Reference . . . . .	361
9.56	RGF.h File Reference . . . . .	362
9.57	senses.cc File Reference . . . . .	364
9.58	senses.h File Reference . . . . .	365
9.59	sensor.h File Reference . . . . .	366
9.60	splitter.cc File Reference . . . . .	368
9.61	splitter.h File Reference . . . . .	369
9.62	suffixes.cc File Reference . . . . .	370
9.63	suffixes.h File Reference . . . . .	372
9.64	sufrule.h File Reference . . . . .	373
9.65	tagger.cc File Reference . . . . .	374
9.66	tagger.h File Reference . . . . .	375
9.67	tok_features.h File Reference . . . . .	376
9.68	tokenizer.cc File Reference . . . . .	380
9.69	tokenizer.h File Reference . . . . .	381
9.70	tokens.h File Reference . . . . .	382
9.71	traces.cc File Reference . . . . .	384
9.72	traces.h File Reference . . . . .	385
9.73	tree.h File Reference . . . . .	390
9.74	util.cc File Reference . . . . .	391
9.75	util.h File Reference . . . . .	392



# Chapter 1

## FreeLing Directory Hierarchy

### 1.1 FreeLing Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

src . . . . .	18
include . . . . .	15
libmorfo . . . . .	17



## Chapter 2

# FreeLing Namespace Index

### 2.1 FreeLing Namespace List

Here is a list of all namespaces with brief descriptions:

<b>std</b> . . . . .	19
----------------------	----



# Chapter 3

## FreeLing Hierarchical Index

### 3.1 FreeLing Class Hierarchy

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

accents . . . . .	21
accents_module . . . . .	28
accents_default . . . . .	23
accents_es . . . . .	25
adaboost . . . . .	30
adaboost::wr_holder . . . . .	34
analysis . . . . .	35
automat . . . . .	39
dates_module . . . . .	89
dates_ca . . . . .	83
dates_default . . . . .	85
dates_es . . . . .	87
locutions . . . . .	128
np . . . . .	155
numbers_module . . . . .	171
numbers_ca . . . . .	161
numbers_default . . . . .	163
numbers_en . . . . .	165
numbers_es . . . . .	167
numbers_gl . . . . .	169
quantities_module . . . . .	191
quantities_default . . . . .	187
quantities_es . . . . .	189
chart_parser . . . . .	48
completer . . . . .	64
completerRule . . . . .	66
condition . . . . .	68
dataset . . . . .	75
dataset::iterator . . . . .	78
dataset::mlDatasetNode . . . . .	80
dates . . . . .	81
dep_info . . . . .	92

dependencyMaker . . . . .	96
depLabeler . . . . .	99
dictionary . . . . .	104
emission_states . . . . .	110
fex . . . . .	113
FlexLexer . . . . .	114
yyFlexLexer . . . . .	275
iFeature . . . . .	124
label . . . . .	125
list . . . . .	127
cell . . . . .	43
document . . . . .	107
ruleCG . . . . .	213
word . . . . .	268
maco . . . . .	131
maco_options . . . . .	133
map_input . . . . .	136
example . . . . .	111
map_input::const_iterator . . . . .	139
mlABTree . . . . .	140
mlOutput . . . . .	144
example . . . . .	111
mlOutput::label . . . . .	147
multimap . . . . .	148
constraint_grammar . . . . .	74
grammar . . . . .	116
nec . . . . .	149
node . . . . .	151
depnode . . . . .	101
numbers . . . . .	159
paragraph . . . . .	174
POS_tagger . . . . .	177
hmm_tagger . . . . .	121
relax_tagger . . . . .	201
probabilities . . . . .	179
punts . . . . .	184
quantities . . . . .	185
RegEx . . . . .	194
relax . . . . .	198
RGF . . . . .	204
rule . . . . .	208
edge . . . . .	108
rule_expression . . . . .	211
check_A_lemma . . . . .	50
check_A_wordclass . . . . .	52
check_and . . . . .	54
check_not . . . . .	56
check_S_category . . . . .	58
check_S_lemma . . . . .	60
check_side . . . . .	62
ruleLabeler . . . . .	215

senses . . . . .	217
Sensor . . . . .	219
SensorCheckMwRE . . . . .	221
SensorCheckRE . . . . .	223
SensorData . . . . .	225
SensorMap . . . . .	226
SensorMatchRE . . . . .	228
SensorSet . . . . .	230
SensorSetPart . . . . .	232
set_input . . . . .	237
splitter . . . . .	241
suffixes . . . . .	244
sufrule . . . . .	247
tokenizer . . . . .	248
traces . . . . .	250
tree< T > . . . . .	252
tree< T >::generic_iterator . . . . .	256
tree< T >::preorder_iterator . . . . .	258
tree< T >::sibling_iterator . . . . .	260
tree< depnode > . . . . .	252
dep_tree . . . . .	94
tree< node > . . . . .	252
parse_tree . . . . .	175
util . . . . .	262
vector . . . . .	265
chart . . . . .	44
constraint . . . . .	72
problem . . . . .	182
sentence . . . . .	234
viterbi . . . . .	266
bool . . . . .	??





## Chapter 4

# FreeLing Class Index

### 4.1 FreeLing Class List

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

<b>accents</b> (The class accents provides a wrapper to transparently create and access an <b>accents_module</b> (p. 28) to handle accentuation for the appropriate language )	21
<b>accents_default</b> (Derived <b>accents_module</b> (p. 28) for null accentuation (eg english) )	23
<b>accents_es</b> (Derived <b>accents_module</b> (p. 28) for Spanish accentuation ) . . . . .	25
<b>accents_module</b> (The abstract class accents_module generalizes accentuation rules for different languages ) . . . . .	28
<b>adaboost</b> . . . . .	30
<b>adaboost::wr_holder</b> . . . . .	34
<b>analysis</b> (Class analysis stores a possible reading (lemma, PoS, probability) for a word )	35
<b>automat</b> (Abstract class to implement a Finite-State Automaton which is used by modules recognizing multiwords (dates, numbers, quantities, ) . . . . .	39
<b>cell</b> (Class cell stores all information in a chart cell ) . . . . .	43
<b>chart</b> (Class chart contains an array of cells that constitute a chart ) . . . . .	44
<b>chart_parser</b> (Class chart_parser implements a chart parser ) . . . . .	48
<b>check_A_lemma</b> (Ancestor lemma in list of lemmas (separator character is  ) ) . . .	50
<b>check_A_wordclass</b> (Ancestor lemma belongs to a verb class ) . . . . .	52
<b>check_and</b> (And of basic constraints ) . . . . .	54
<b>check_not</b> (Negation ) . . . . .	56
<b>check_S_category</b> (Ancestor category ) . . . . .	58
<b>check_S_lemma</b> (Descendant lemma in list of lemmas (separator character is  ) ) . .	60
<b>check_side</b> (Descendant is to the right of its ancestor ) . . . . .	62
<b>completer</b> (The class completer implements a parse tree completer, which given a partial parse tree (chunker output), completes the full parse according to some grammar rules ) . . . . .	64
<b>completerRule</b> (The class completerRule stores rules used by the completer of parse trees ) . . . . .	66
<b>condition</b> (Class condition implements a condition of a CG rule ) . . . . .	68
<b>constraint</b> (The class constraint implements a constraint for the relaxation labelling algorithm ) . . . . .	72
<b>constraint_grammar</b> (Class constraint_grammar implements a pseudo CG, ready to be used from a relax PoS tagger ) . . . . .	74
<b>dataset</b> . . . . .	75
<b>dataset::iterator</b> . . . . .	78

<b>dataset::mlDatasetNode</b> . . . . .	80
<b>dates</b> (The class dates provides a wrapper to transparently create and access a <b>dates_</b> - <b>module</b> (p. 89), a temporal expression recognizer for the appropriate language ) . . . . .	81
<b>dates_ca</b> (The derived class dates_ca implements a Catalan date/time recognizer ) . .	83
<b>dates_default</b> (The derived class dates_default implements a default date/time recognizer (only simple patterns are recognized) ) . . . . .	85
<b>dates_es</b> (The derived class dates_es implements a Spanish date/time recognizer ) . .	87
<b>dates_module</b> (The abstract class dates_module generalizes temporal expression recognizer for different languages ) . . . . .	89
<b>dep_info</b> (Auxiliary class to store information about dependency building ) . . . . .	92
<b>dep_tree</b> (Class dep_tree stores a dependency tree ) . . . . .	94
<b>dependencyMaker</b> (DependencyMaker is a class for obtaining a dependency tree from chunks ) . . . . .	96
<b>depLabeler</b> (DepLabeler is class to set labels into a dependency tree ) . . . . .	99
<b>depnode</b> (Class denode stores nodes of a dependency tree and parse tree <-> deptime relations ) . . . . .	101
<b>dictionary</b> (The class dictionary implements dictionary search and suffix analysis for word forms ) . . . . .	104
<b>document</b> (Class document is a list of paragraphs ) . . . . .	107
<b>edge</b> (Class edge stores all information in a chart edge ) . . . . .	108
<b>emission_states</b> (The class emission_states stores the list of states in the HMM that *may* be generating a given word given the two previous words (and their valid tags) ) . . . . .	110
<b>example</b> . . . . .	111
<b>flex</b> . . . . .	113
<b>FlexLexer</b> . . . . .	114
<b>grammar</b> (Class grammar implements a CFG, ready to be used from a chart parser ) .	116
<b>hmm_tagger</b> (The class hmm_tagger implements the syntactic analyzer and is the main class, which uses all the others ) . . . . .	121
<b>iFeature</b> . . . . .	124
<b>label</b> (The class label stores all information related to a variable label in the relaxation labelling algorithm ) . . . . .	125
<b>list</b> . . . . .	127
<b>locutions</b> (Class locutions recognizes multiwords belonging to a list obtained from a configuration file ) . . . . .	128
<b>maco</b> (Class maco implements the morphological analyzer, which uses all the specific analyzers: dates, numbers, dictionary, etc ) . . . . .	131
<b>maco_options</b> (Class maco_options implements a set of specific options of the morphological analyzer ) . . . . .	133
<b>map_input</b> . . . . .	136
<b>map_input::const_iterator</b> . . . . .	139
<b>mlABTree</b> . . . . .	140
<b>mlOutput</b> . . . . .	144
<b>mlOutput::label</b> . . . . .	147
<b>multimap</b> . . . . .	148
<b>nec</b> (The class nec implements a ML-based NE classifier ) . . . . .	149
<b>node</b> (Class node stores nodes of a <b>parse_tree</b> (p. 175) Each node in the tree is either a label (intermediate node) or a word (leaf node) ) . . . . .	151
<b>np</b> (The class np implements a dummy proper noun recognizer ) . . . . .	155
<b>numbers</b> (Class numbers implements a wrapper to transparently create and access a <b>numbers_module</b> (p. 171) number recognizer for the appropriate language ) .	159
<b>numbers_ca</b> (The derived class numbers_ca implements a Catalan number recognizer )	161

<b>numbers_default</b> (The derived class <b>numbers_default</b> implements a default number recognizer (only numbers in digits are recognized) ) . . . . .	163
<b>numbers_en</b> (The derived class <b>numbers_en</b> implements an English number recognizer )	165
<b>numbers_es</b> (The derived class <b>numbers_es</b> implements a Spanish number recognizer )	167
<b>numbers_gl</b> (The derived class <b>numbers_ca</b> (p.161) implements a Galician number recognizer ) . . . . .	169
<b>numbers_module</b> (The abstract class <b>numbers_module</b> generalizes numeric expression recognizer for different languages ) . . . . .	171
<b>paragraph</b> (Class <b>paragraph</b> is just a list of sentences that someone has validated it as a paragraph ) . . . . .	174
<b>parse_tree</b> (Class <b>parse tree</b> is used to store the results of parsing ) . . . . .	175
<b>POS_tagger</b> (The class <b>POS_tagger</b> is just an abstract class generalizing a PoS tagger )	177
<b>probabilities</b> (Class <b>probabilities</b> sets lexical probabilities for each PoS tag of each word in a sentence ) . . . . .	179
<b>problem</b> (The class <b>problem</b> stores the structure of a problem, namely, a vector with a position for each variable to consider, and for each variable, a list of initial weights for each possible label ) . . . . .	182
<b>punts</b> (Class <b>numbers</b> implements a punctuation sign recognizer ) . . . . .	184
<b>quantities</b> (Class <b>quantities</b> implements a wrapper to transparently create and access a <b>quantities_module</b> (p. 191) monetary expressions detector for the appropriate language ) . . . . .	185
<b>quantities_default</b> (The derived class <b>quantities_default</b> implements a default quantities recognizer (only percentages are recognized) ) . . . . .	187
<b>quantities_es</b> (The derived class <b>quantities_es</b> implements a Spanish (and Catalan) quantities recognizer ) . . . . .	189
<b>quantities_module</b> (The abstract class <b>quantities_module</b> generalizes a percentage, ratios, and currency expression recognizer for different languages ) . . . . .	191
<b>RegEx</b> (Class <b>RegEx</b> , a simple and small API wrapper for PCRE ) . . . . .	194
<b>relax</b> (The class <b>relax</b> implements a generic solver for consistent labelling problems, using relaxation labelling algorithm ) . . . . .	198
<b>relax_tagger</b> (The class <b>relax_tagger</b> implements a PoS tagger based on relaxation labelling algorithm ) . . . . .	201
<b>RGF</b> . . . . .	204
<b>rule</b> (Class <b>rule</b> implements a rule of a grammar ) . . . . .	208
<b>rule_expression</b> (The class <b>rule_expression</b> is an abstract class (interface) for building dynamic restriction on a <b>ruleLabeler</b> (p.215) which are used by class <b>depLabeler</b> (p. 99) ) . . . . .	211
<b>ruleCG</b> (Class <b>rule</b> implements a rule of a CG ) . . . . .	213
<b>ruleLabeler</b> (RuleLabeler is an auxiliary class for the <b>depLabeler</b> (p. 99) ) . . . . .	215
<b>senses</b> (Class <b>senses</b> implements a sense annotator ) . . . . .	217
<b>Sensor</b> . . . . .	219
<b>SensorCheckMwRE</b> . . . . .	221
<b>SensorCheckRE</b> . . . . .	223
<b>SensorData</b> . . . . .	225
<b>SensorMap</b> . . . . .	226
<b>SensorMatchRE</b> . . . . .	228
<b>SensorSet</b> . . . . .	230
<b>SensorSetPart</b> . . . . .	232
<b>sentence</b> (Class <b>sentence</b> is just a list of words that someone (the splitter) has validated it as a complete sentence ) . . . . .	234
<b>set_input</b> . . . . .	237
<b>splitter</b> (Class <b>splitter</b> implements a sentence splitter, which accumulates lists of words until a sentence is completed, and then returns a list of sentence objects ) . . .	241

<b>suffixes</b> (Class suffixes implements suffixation rules and dictionary search for suffixed word forms ) . . . . .	244
<b>sufrule</b> (Class sufrule contains a suffixation rule, and is used by class suffixes ) . . . . .	247
<b>tokenizer</b> (Class tokenizer implements a token splitter, which converts a string into a sequence of word objects, according to a set of tokenization rules read from aconfiguration file ) . . . . .	248
<b>traces</b> (Class traces implements trace and error handling utilities ) . . . . .	250
<b>tree&lt; T &gt;</b> . . . . .	252
<b>tree&lt; T &gt;::generic_iterator</b> . . . . .	256
<b>tree&lt; T &gt;::preorder_iterator</b> (Traverse the tree in preorder (parent first, then children) ) . . . . .	258
<b>tree&lt; T &gt;::sibling_iterator</b> (Traverse all children of the same node ) . . . . .	260
<b>util</b> (Class util implements some utilities for NLP analyzers: "tolower" for latin alphabets, parole tags manipulation, string2number and viceversa conversions, etc ) . . . . .	262
<b>vector</b> . . . . .	265
<b>viterbi</b> (The class viterbi stores the two maps for each observation: The delta map stores the maximum probability for each state in that observation, the phi map stores the backpath to maximize the probability ) . . . . .	266
<b>word</b> (Class word stores all info related to a word: form, list of analysis, list of tokens (if multiword) ) . . . . .	268
<b>yyFlexLexer</b> . . . . .	275

# Chapter 5

## FreeLing File Index

### 5.1 FreeLing File List

Here is a list of all files with brief descriptions:

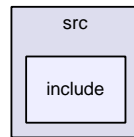
accents.cc . . . . .	281
accents.h . . . . .	283
accents_modules.cc . . . . .	284
accents_modules.h . . . . .	285
adaboost.h . . . . .	286
automat.cc . . . . .	287
automat.h . . . . .	288
chart.h . . . . .	290
chart_parser.h . . . . .	291
constraint_grammar.h . . . . .	292
dataset.h . . . . .	293
dates.cc . . . . .	294
dates.h . . . . .	295
dates_modules.cc . . . . .	296
dates_modules.h . . . . .	302
dep_rules.cc . . . . .	305
dep_rules.h . . . . .	306
dependencies.cc . . . . .	308
dependencies.h . . . . .	310
dictionary.cc . . . . .	311
dictionary.h . . . . .	313
example.h . . . . .	314
fex.h . . . . .	315
FlexLexer.h . . . . .	316
grammar.h . . . . .	317
hmm_tagger.cc . . . . .	318
hmm_tagger.h . . . . .	319
language.cc . . . . .	320
language.h . . . . .	321
locutions.cc . . . . .	324
locutions.h . . . . .	326
maco.cc . . . . .	327
maco.h . . . . .	328

maco_options.cc	329
maco_options.h	330
nec.cc	331
nec.h	332
np.cc	334
np.h	336
numbers.cc	337
numbers.h	338
numbers_modules.cc	339
numbers_modules.h	345
probabilities.cc	347
probabilities.h	348
punts.cc	349
punts.h	350
quantities.cc	351
quantities.h	352
quantities_modules.cc	353
quantities_modules.h	356
readFeatures.h	357
regex.h	359
relax.h	360
relax_tagger.h	361
RGF.h	362
senses.cc	364
senses.h	365
sensor.h	366
splitter.cc	368
splitter.h	369
suffixes.cc	370
suffixes.h	372
sufrule.h	373
tagger.cc	374
tagger.h	375
tok_features.h	376
tokenizer.cc	380
tokenizer.h	381
tokens.h	382
traces.cc	384
traces.h	385
tree.h	390
util.cc	391
util.h	392

## Chapter 6

# FreeLing Directory Documentation

### 6.1 /home/padro/freeling-1.4/src/include/ Directory Reference



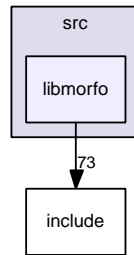
#### Files

- file accents.h
- file accents\_modules.h
- file adaboost.h
- file automat.h
- file chart.h
- file chart\_parser.h
- file constraint\_grammar.h
- file dataset.h
- file dates.h
- file dates\_modules.h
- file dep\_rules.h
- file dependencies.h
- file dictionary.h
- file example.h
- file fex.h
- file FlexLexer.h
- file grammar.h
- file hmm\_tagger.h
- file language.h
- file locutions.h
- file maco.h

- file **maco\_options.h**
- file **nec.h**
- file **np.h**
- file **numbers.h**
- file **numbers\_modules.h**
- file **probabilities.h**
- file **punts.h**
- file **quantities.h**
- file **quantities\_modules.h**
- file **readFeatures.h**
- file **regexp.h**
- file **relax.h**
- file **relax\_tagger.h**
- file **RGF.h**
- file **senses.h**
- file **sensor.h**
- file **splitter.h**
- file **suffixes.h**
- file **sufrule.h**
- file **tagger.h**
- file **tok\_features.h**
- file **tokenizer.h**
- file **tokens.h**
- file **traces.h**
- file **tree.h**
- file **util.h**



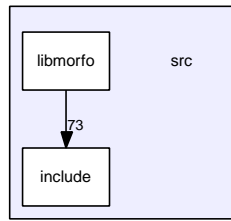
## 6.2 /home/padro/freeling-1.4/src/libmorfo/ Directory Reference



### Files

- file **accents.cc**
- file **accents\_modules.cc**
- file **automat.cc**
- file **dates.cc**
- file **dates\_modules.cc**
- file **dep\_rules.cc**
- file **dependencies.cc**
- file **dictionary.cc**
- file **hmm\_tagger.cc**
- file **language.cc**
- file **locutions.cc**
- file **maco.cc**
- file **maco\_options.cc**
- file **nec.cc**
- file **np.cc**
- file **numbers.cc**
- file **numbers\_modules.cc**
- file **probabilities.cc**
- file **punts.cc**
- file **quantities.cc**
- file **quantities\_modules.cc**
- file **senses.cc**
- file **splitter.cc**
- file **suffixes.cc**
- file **tagger.cc**
- file **tokenizer.cc**
- file **traces.cc**
- file **util.cc**

## 6.3 /home/padro/freeling-1.4/src/ Directory Reference



### Directories

- directory **include**
- directory **libmorfo**

## Chapter 7

# FreeLing Namespace Documentation

### 7.1 std Namespace Reference



## Chapter 8

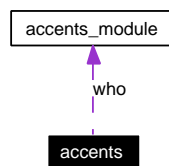
# FreeLing Class Documentation

### 8.1 accents Class Reference

The class accents provides a wrapper to transparently create and access an **accents\_module**(p. 28) to handle accentuation for the appropriate language.

```
#include <accents.h>
```

Collaboration diagram for accents:



#### Public Member Functions

- **accents** (const **maco\_options** &)  
*Constructor.*
- **~accents** ()  
*Destructor.*
- void **fix\_accentuation** (vector< string > &, const **sufrule** &) const  
*fix accentuation patterns of a list of root candidates, according to a suffix rule*

#### Private Attributes

- **accents\_module** \* **who**  
*remember which module is doing the real work.*

### 8.1.1 Detailed Description

The class `accents` provides a wrapper to transparently create and access an `accents__module`(p. 28) to handle accentuation for the appropriate language.

### 8.1.2 Constructor & Destructor Documentation

#### 8.1.2.1 `accents::accents (const maco_options & opts)`

Constructor.

Create Spanish accent handler

Create Galician accent handler !! (equal to Spanish ??) !!

Create Catalan accent handler !! To be fixed !!

Create Default (null) accent handler. Ok for english.

#### 8.1.2.2 `accents::~~accents ()`

Destructor.

### 8.1.3 Member Function Documentation

#### 8.1.3.1 `void accents::fix_accentuation (vector< string > &, const sufrule &) const`

fix accentuation patterns of a list of root candidates, according to a suffix rule

### 8.1.4 Member Data Documentation

#### 8.1.4.1 `accents__module* accents::who` [private]

remember which module is doing the real work.

The documentation for this class was generated from the following files:

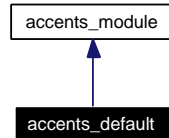
- `accents.h`
- `accents.cc`

## 8.2 accents\_\_default Class Reference

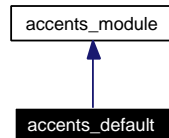
Derived **accents\_\_module**(p. 28) for null accentuation (eg english).

```
#include <accents_modules.h>
```

Inheritance diagram for accents\_\_default:



Collaboration diagram for accents\_\_default:



### Public Member Functions

- **accents\_\_default** ()  
*Constructor.*
- void **fix\_\_accentuation** (vector< string > &, const **sufrule** &) const  
*default accentuation patterns*

### 8.2.1 Detailed Description

Derived **accents\_\_module**(p. 28) for null accentuation (eg english).

### 8.2.2 Constructor & Destructor Documentation

#### 8.2.2.1 accents\_\_default::accents\_\_default ()

Constructor.

### 8.2.3 Member Function Documentation

#### 8.2.3.1 void accents\_\_default::fix\_\_accentuation (vector< string > &, const **sufrule** &) const [virtual]

default accentuation patterns

Implements **accents\_\_module** (p. 28).

The documentation for this class was generated from the following files:

- accents\_modules.h
- accents\_modules.cc

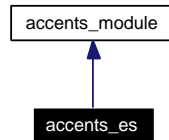


## 8.3 accents\_\_es Class Reference

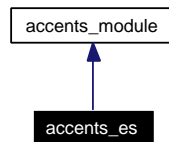
Derived **accents\_\_module**(p. 28) for Spanish accentuation.

```
#include <accents_modules.h>
```

Inheritance diagram for accents\_\_es:



Collaboration diagram for accents\_\_es:



### Public Member Functions

- **accents\_\_es** ()  
*Constructor.*
- void **fix\_\_accentuation** (vector< string > &, const **sufrule** &) const  
*Specific accentuation patterns for Spanish.*

### Static Private Member Functions

- static **bool** **is\_\_vowel** (char)  
*check for a vowel (maybe with latin accents)*
- static **bool** **is\_\_vowel\_\_notacc** (char)  
*check for a vowel (strictly)*
- static **bool** **is\_\_monosyllabic** (const string &)  
*check monosyllabous word*
- static **bool** **is\_\_open** (char)  
*check for weak spanish vowels*
- static **bool** **is\_\_accentued\_\_esp** (char)  
*check for spanish accents*
- static **bool** **is\_\_accentued\_\_esp** (const string &)  
*check for spanish accents*

- static void **remove\_\_accent\_\_esp** (string &)  
*remove spanish accents*
- static void **remove\_\_accent\_\_esp** (char &)  
*remove spanish accents*
- static void **put\_\_accent\_\_esp** (char &)  
*set spanish accents*
- static bool **put\_\_accent\_\_esp** (string &)  
*set spanish accents*
- static bool **llana\_\_acentuada** (const string &)  
*check for spanish accent pattern: "palabras llanas acentuadas"*

### 8.3.1 Detailed Description

Derived **accents\_\_module**(p. 28) for Spanish accentuation.

### 8.3.2 Constructor & Destructor Documentation

#### 8.3.2.1 accents\_\_es::accents\_\_es ()

Constructor.

### 8.3.3 Member Function Documentation

#### 8.3.3.1 void accents\_\_es::fix\_\_accentuation (vector< string > & *candidates*, const sufrule & *suf*) const [virtual]

Specific accentuation patterns for Spanish.

Roots are obtained after suffix removal and may require accent fixing, which is done here.

Implements **accents\_\_module** (p. 28).

#### 8.3.3.2 bool accents\_\_es::is\_\_accentued\_\_esp (const string &) [static, private]

check for spanish accents

#### 8.3.3.3 bool accents\_\_es::is\_\_accentued\_\_esp (char) [static, private]

check for spanish accents

#### 8.3.3.4 bool accents\_\_es::is\_\_monosyllabic (const string &) [static, private]

check monosyllabous word

**8.3.3.5 bool accents\_\_es::is\_\_open (char) [static, private]**

check for weak spanish vowels

**8.3.3.6 bool accents\_\_es::is\_\_vowel (char) [static, private]**

check for a vowel (maybe with latin accents)

**8.3.3.7 bool accents\_\_es::is\_\_vowel\_\_notacc (char) [static, private]**

check for a vowel (strictly)

**8.3.3.8 bool accents\_\_es::llana\_\_acentuada (const string &) [static, private]**

check for spanish accent pattern: "palabras llanas acentuadas"

**8.3.3.9 bool accents\_\_es::put\_\_accent\_\_esp (string &) [static, private]**

set spanish accents

**8.3.3.10 void accents\_\_es::put\_\_accent\_\_esp (char &) [static, private]**

set spanish accents

**8.3.3.11 void accents\_\_es::remove\_\_accent\_\_esp (char &) [static, private]**

remove spanish accents

**8.3.3.12 void accents\_\_es::remove\_\_accent\_\_esp (string &) [static, private]**

remove spanish accents

The documentation for this class was generated from the following files:

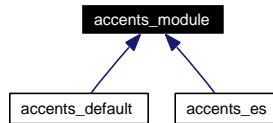
- accents\_\_modules.h
- accents\_\_modules.cc

## 8.4 accents\_\_module Class Reference

The abstract class accents\_\_module generalizes accentuation rules for different languages.

```
#include <accents_modules.h>
```

Inheritance diagram for accents\_\_module:



### Public Member Functions

- **accents\_\_module ()**  
*Constructor.*
- virtual void **fix\_\_accentuation** (**vector**< string > &, const **sufrule** &) const =0  
*Specific accentuation patterns.*
- virtual ~**accents\_\_module** ()  
*Destructor.*

#### 8.4.1 Detailed Description

The abstract class accents\_\_module generalizes accentuation rules for different languages.

#### 8.4.2 Constructor & Destructor Documentation

##### 8.4.2.1 accents\_\_module::accents\_\_module ()

Constructor.

##### 8.4.2.2 virtual accents\_\_module::~~accents\_\_module () [inline, virtual]

Destructor.

#### 8.4.3 Member Function Documentation

##### 8.4.3.1 virtual void accents\_\_module::fix\_\_accentuation (vector< string > &, const sufrule &) const [pure virtual]

Specific accentuation patterns.

Implemented in **accents\_\_default** (p. 23), and **accents\_\_es** (p. 26).

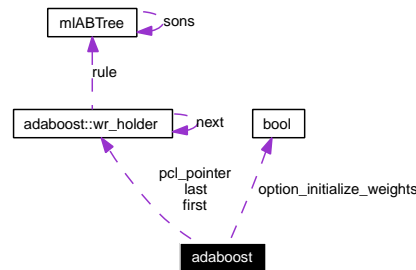
The documentation for this class was generated from the following files:

- accents\_modules.h
- accents\_modules.cc

## 8.5 adaboost Class Reference

```
#include <adaboost.h>
```

Collaboration diagram for adaboost:



### Public Member Functions

- **adaboost** (int nl)
- **adaboost** (const string &file)
- **~adaboost** ()
- int **n\_rules** ()
- int **get\_nlabels** ()
- string **get\_label** (int lb)
- string **default\_class** ()
- void **classify** (input \*i, double pred[])
- void **pcl\_ini\_pointer** ()
- int **pcl\_advance\_pointer** (int steps)
- void **pcl\_classify** (input \*i, double \*pred, int nrules)
- void **learn** (dataset \*ds, int nrounds, int maxdepth)
- void **set\_output** (ofstream &os)
- void **read\_from\_stream** (ifstream &in)
- void **read\_from\_file** (char \*file)

### Static Public Member Functions

- static void **set\_verbose** (int level)
- static void **set\_epsilon** (double eps)
- static void **set\_initialize\_weights** (bool b)

### Private Member Functions

- int **stopping\_criterion** (int nrounds)
- void **initialize\_weights** (dataset \*ds)
- void **update\_weights** (mABTree \*wr, double Z, dataset \*ds)
- void **add\_weak\_rule** (mABTree \*wr)
- **adaboost** (const **adaboost** &old\_bab)

## Private Attributes

- `wr_holder * first`
- `wr_holder * last`
- `wr_holder * pcl_pointer`
- `int nrules`
- `int nlabels`
- `vector< string > labels`
- `string label_others`
- `ofstream * out`
- `struct {  
    int n_rounds  
    int max_depth  
} SC`

## Static Private Attributes

- `static double epsilon`
- `static int verbose`
- `static bool option_initialize_weights`

## Classes

- `struct wr_holder`

### 8.5.1 Constructor & Destructor Documentation

8.5.1.1 `adaboost::adaboost (const adaboost & old_bab)` [private]

8.5.1.2 `adaboost::adaboost (int nl)`

8.5.1.3 `adaboost::adaboost (const string & file)`

8.5.1.4 `adaboost::~~adaboost ()`

### 8.5.2 Member Function Documentation

8.5.2.1 `void adaboost::add_weak_rule (mlABTree * wr)` [private]

8.5.2.2 `void adaboost::classify (input * i, double pred[])`

8.5.2.3 `string adaboost::default_class ()` [inline]

8.5.2.4 `string adaboost::get_label (int lb)` [inline]

8.5.2.5 `int adaboost::get_nlabels ()` [inline]

8.5.2.6 `void adaboost::initialize_weights (dataset * ds)` [private]

8.5.2.7 `void adaboost::learn (dataset * ds, int nrounds, int maxdepth)`

8.5.2.8 `int adaboost::n_rules ()`

8.5.2.9 `int adaboost::pcl_advance_pointer (int steps)`

8.5.2.10 `void adaboost::pcl_classify (input * i, double * pred, int nrules)`

8.5.2.11 `void adaboost::pcl_ini_pointer ()`

8.5.2.12 `void adaboost::read_from_file (char * file)`

8.5.2.13 `void adaboost::read_from_stream (ifstream & in)`

8.5.2.14 `static void adaboost::set_epsilon (double eps)` [static]

8.5.2.15 `static void adaboost::set_initialize_weights (bool b)` [static]

8.5.2.16 `void adaboost::set_output (ofstream & os)`

8.5.2.17 `static void adaboost::set_verbose (int level)` [static]

8.5.2.18 `int adaboost::stopping_criterion (int nrounds)` [private]

8.5.2.19 `void adaboost::update_weights (mlABTree * wr, double Z, dataset * ds)`  
[private]

### 8.5.3 Member Data Documentation

8.5.3.1 `double adaboost::epsilon` [static, private]

8.5.3.2 `wr_holder* adaboost::first` [private]

8.5.3.3 `string adaboost::label_others` [private]

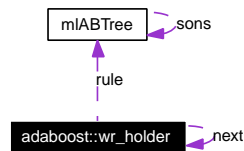
8.5.3.4 `vector<string> adaboost::labels` [private]



- `adaboost.h`

## 8.6 adaboost::wr\_holder Struct Reference

Collaboration diagram for adaboost::wr\_holder:



### Public Attributes

- `mlABTree * rule`
- `wr_holder * next`

### 8.6.1 Member Data Documentation

8.6.1.1 `wr_holder* adaboost::wr_holder::next`

8.6.1.2 `mlABTree* adaboost::wr_holder::rule`

The documentation for this struct was generated from the following file:

- `adaboost.h`

## 8.7 analysis Class Reference

Class analysis stores a possible reading (lemma, PoS, probability) for a word.

```
#include <language.h>
```

### Public Member Functions

- **analysis** ()  
*constructor*
- **analysis** (const string &, const string &)  
*constructor*
- void **set\_lemma** (const string &)  
*Set lemma for analysis.*
- void **set\_parole** (const string &)  
*Set parole PoS tag for analysis.*
- void **set\_prob** (double)  
*Set probability for analysis.*
- void **set\_retokenizable** (const list< word > &)  
*Set retokenization info for analysis.*
- bool **has\_prob** (void) const  
*Check whether probability has been set.*
- string **get\_lemma** (void) const  
*Get lemma value for analysis.*
- string **get\_parole** (void) const  
*Get parole PoS tag value for analysis.*
- string **get\_short\_parole** (const string &) const  
*Get short version of parole PoS tag value for analysis.*
- double **get\_prob** (void) const  
*Get probability value for analysis (-1 if not set).*
- bool **is\_retokenizable** (void) const  
*Find out if the analysis may imply retokenization.*
- list< word > **get\_retokenizable** (void) const  
*Get retokenization info for analysis.*
- list< string > **get\_senses** (void) const  
*get analysis sense list*

- void **set\_senses** (const **list**< string > &)  
*set analysis sense list*

## Private Attributes

- string **lemma**  
*lemma*
- string **parole**  
*PoS tag.*
- double **prob**  
*probability of that lemma-tag given the word*
- **list**< string > **senses**  
*list of possible senses for that analysis*
- **list**< word > **retok**  
*information to retokenize the word after tagging if this analysis is selected*

### 8.7.1 Detailed Description

Class analysis stores a possible reading (lemma, PoS, probability) for a word.

### 8.7.2 Constructor & Destructor Documentation

#### 8.7.2.1 analysis::analysis ()

constructor

#### 8.7.2.2 analysis::analysis (const string &, const string &)

constructor

### 8.7.3 Member Function Documentation

#### 8.7.3.1 string analysis::get\_lemma (void) const

Get lemma value for analysis.

#### 8.7.3.2 string analysis::get\_parole (void) const

Get parole PoS tag value for analysis.

**8.7.3.3 double analysis::get\_\_prob (void) const**

Get probability value for analysis (-1 if not set).

**8.7.3.4 list< word > analysis::get\_\_retokenizable (void) const**

Get retokenization info for analysis.

**8.7.3.5 list< string > analysis::get\_\_senses (void) const**

get analysis sense list

**8.7.3.6 string analysis::get\_\_short\_\_parole (const string &) const**

Get short version of parole PoS tag value for analysis.

**8.7.3.7 bool analysis::has\_\_prob (void) const**

Check whether probability has been set.

**8.7.3.8 bool analysis::is\_\_retokenizable (void) const**

Find out if the analysis may imply retokenization.

**8.7.3.9 void analysis::set\_\_lemma (const string &)**

Set lemma for analysis.

**8.7.3.10 void analysis::set\_\_parole (const string &)**

Set parole PoS tag for analysis.

**8.7.3.11 void analysis::set\_\_prob (double)**

Set probability for analysis.

**8.7.3.12 void analysis::set\_\_retokenizable (const list< word > &)**

Set retokenization info for analysis.

**8.7.3.13 void analysis::set\_\_senses (const list< string > &)**

set analysis sense list

## 8.7.4 Member Data Documentation

### 8.7.4.1 `string analysis::lemma` [private]

lemma

### 8.7.4.2 `string analysis::parole` [private]

PoS tag.

### 8.7.4.3 `double analysis::prob` [private]

probability of that lemma-tag given the word

### 8.7.4.4 `list<word> analysis::retok` [private]

information to retokenize the word after tagging if this analysis is selected

### 8.7.4.5 `list<string> analysis::senses` [private]

list of possible senses for that analysis

The documentation for this class was generated from the following files:

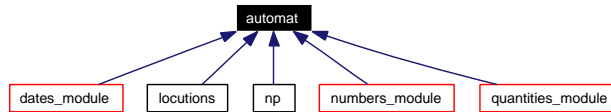
- `language.h`
- `language.cc`

## 8.8 automat Class Reference

Abstract class to implement a Finite-State Automaton which is used by modules recognizing multiwords (dates, numbers, quantities, .

```
#include <automat.h>
```

Inheritance diagram for automat:



### Public Member Functions

- **automat** ()  
*Constructor.*
- virtual **~automat** ()  
*Destructor.*
- void **annotate** (sentence &)  
*Detect patterns in sentence using default options.*

### Protected Attributes

- int **initialState**  
*state code of initial state*
- int **stopState**  
*state code for stop State*
- int **trans** [MAX\_STATES][MAX\_TOKENS]  
*Transition tables.*
- set< int > **Final**  
*set of final states*

### Private Member Functions

- virtual int **ComputeToken** (int, sentence::const\_iterator, const **sentence** &)=0  
*pure virtual function to be provided by the child class.*
- virtual void **ResetActions** ()=0  
*pure virtual function to be provided by the child class .*
- virtual void **StateActions** (int, int, int, sentence::const\_iterator)=0

*pure virtual function to be provided by the child class.*

- virtual void **SetMultiwordAnalysis** (sentence::iterator) const =0  
*pure virtual function to be provided by the child class.*
- virtual bool **ValidMultiWord** (const word &) const  
*virtual function (true by default).*
- sentence::iterator **BuildMultiword** (sentence &, sentence::iterator, sentence::iterator) const  
*Private function to re-arrange sentence when match found.*

### 8.8.1 Detailed Description

Abstract class to implement a Finite-State Automaton which is used by modules recognizing multiwords (dates, numbers, quantities, .

..).

Details:

- Child classes must implement a DFA, by means of a transition table.
- IMPORTANT: Since many matches may be found in a sentence, when arriving to a error or final state, the transition table must provide a default transition to "stopState" where the automata will stop, and check for longest match found.
- Specific actions may be associated to each state using the "StateActions" virtual function.
- Token codes for a word must be provided by the "ComputeToken" virtual function.

Child classes must provide a constructor that:

- fills the "final" set
- fills the "trans" table
- sets initialState and stopState
- initializes any other information required by the child

Child classes must provide the virtual functions:

- virtual int ComputeToken(int,sentence::iterator, sentence &); (computes the token code, given the state and word. Sentence reference is provided just in case extra info is necessary)
- virtual void **ResetActions**()(p. 41); (reset any variables required by state actions)
- virtual void StateActions(int, int, int, sentence::iterator); (perform specific state actions)
- virtual void SetMultiwordAnalysis(sentence::iterator); (once a mw has been detected and build, set its lemma&parole)

Child classes must declare and manage any private attribute or function they may need to perform the expected computations



## 8.8.2 Constructor & Destructor Documentation

### 8.8.2.1 `automat::automat ()`

Constructor.

Since `automat` is an abstract class, this is called always from child constructors.

### 8.8.2.2 `virtual automat::~~automat () [inline, virtual]`

Destructor.

## 8.8.3 Member Function Documentation

### 8.8.3.1 `void automat::annotate (sentence & se)`

Detect patterns in sentence using default options.

Recognize the longest pattern starting at first possible start found. Repeat the process starting from first word after recognized pattern, until sentence ends.

### 8.8.3.2 `sentence::iterator automat::BuildMultiword (sentence &, sentence::iterator, sentence::iterator) const [private]`

Private function to re-arrange sentence when match found.

### 8.8.3.3 `virtual int automat::ComputeToken (int, sentence::const_iterator, const sentence &) [private, pure virtual]`

pure virtual function to be provided by the child class.

Computes token code for current word in current state.

Implemented in `dates_default` (p. 86), `dates_es` (p. 88), `dates_ca` (p. 84), `locutions` (p. 129), `np` (p. 156), `numbers_default` (p. 164), `numbers_es` (p. 168), `numbers_ca` (p. 162), `numbers_gl` (p. 170), `numbers_en` (p. 166), `quantities_default` (p. 188), and `quantities_es` (p. 190).

### 8.8.3.4 `virtual void automat::ResetActions () [private, pure virtual]`

pure virtual function to be provided by the child class .

Resets automaton internal variables when a new search is started.

Implemented in `dates_default` (p. 86), `dates_es` (p. 88), `dates_ca` (p. 84), `locutions` (p. 129), `np` (p. 156), `numbers_default` (p. 164), `numbers_es` (p. 168), `numbers_ca` (p. 162), `numbers_gl` (p. 170), `numbers_en` (p. 166), `quantities_default` (p. 188), and `quantities_es` (p. 190).

### 8.8.3.5 `virtual void automat::SetMultiwordAnalysis (sentence::iterator) const [private, pure virtual]`

pure virtual function to be provided by the child class.

Sets analysis for pattern identified as a multiword.

Implemented in **dates\_default** (p. 86), **dates\_es** (p. 88), **dates\_ca** (p. 84), **locutions** (p. 129), **np** (p. 157), **numbers\_default** (p. 164), **numbers\_es** (p. 168), **numbers\_ca** (p. 162), **numbers\_gl** (p. 170), **numbers\_en** (p. 166), **quantities\_default** (p. 188), and **quantities\_es** (p. 190).

#### 8.8.3.6 **virtual void automat::StateActions (int, int, int, sentence::const\_iterator)** [private, pure virtual]

pure virtual function to be provided by the child class.

Performs appropriate internal actions, given origin and destination states, token code and word.

Implemented in **dates\_default** (p. 86), **dates\_es** (p. 88), **dates\_ca** (p. 84), **locutions** (p. 130), **np** (p. 157), **numbers\_default** (p. 164), **numbers\_es** (p. 168), **numbers\_ca** (p. 162), **numbers\_gl** (p. 170), **numbers\_en** (p. 166), **quantities\_default** (p. 188), and **quantities\_es** (p. 190).

#### 8.8.3.7 **bool automat::ValidMultiWord (const word &) const** [private, virtual]

virtual function (true by default).

Allows the child class to perform a last-minute check before effectively building the multiword.

Reimplemented in **np** (p. 157).

### 8.8.4 Member Data Documentation

#### 8.8.4.1 **set<int> automat::Final** [protected]

set of final states

#### 8.8.4.2 **int automat::initialState** [protected]

state code of initial state

#### 8.8.4.3 **int automat::stopState** [protected]

state code for stop State

#### 8.8.4.4 **int automat::trans[MAX\_STATES][MAX\_TOKENS]** [protected]

Transition tables.

The documentation for this class was generated from the following files:

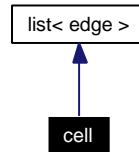
- **automat.h**
- **automat.cc**

## 8.9 cell Class Reference

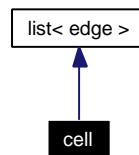
Class cell stores all information in a chart cell.

```
#include <chart.h>
```

Inheritance diagram for cell:



Collaboration diagram for cell:



### 8.9.1 Detailed Description

Class cell stores all information in a chart cell.

The documentation for this class was generated from the following file:

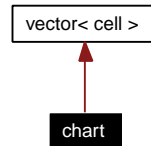
- **chart.h**

## 8.10 chart Class Reference

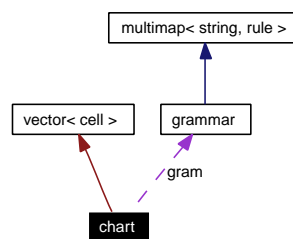
Class chart contains an array of cells that constitute a chart.

```
#include <chart.h>
```

Inheritance diagram for chart:



Collaboration diagram for chart:



### Public Member Functions

- **chart** ()  
*constructor*
- **int** **get\_\_size** () const  
*Get size of the table.*
- **cell** **get\_\_cell** (int, int) const  
*get the contents of a cell in a given position*
- **void** **load\_\_sentence** (const **sentence** &)  
*load sentece and init parsing (fill up first row of chart)*
- **void** **set\_\_grammar** (const **grammar** &)  
*set grammar to use in parsing*
- **void** **parse** ()  
*Do the parsing.*
- **parse\_\_tree** **get\_\_tree** (int, int, const string &="") const  
*navigate through the chart and obtain a parse tree.*

## Private Member Functions

- **bool better\_edge** (const **edge** &, const **edge** &) const  
*compare two edges when extracting a tree*
- **list< pair< int, int > > cover** (int a, int b) const  
*obtain a list of cells that cover the subtree under cell (a,b)*
- **int index** (int i, int j) const  
*compute position of the cell inside the vector*
- **bool can\_extend** (const string &, int, int) const  
*find out whether the cell (i,j) has some inactive edge whose head is the given category*
- **void find\_all\_rules** (const **edge** &, **cell** &, int, int) const  
*Complete edges in a cell (ce) after inserting a terminal or an inactive edge, using rules whose right part starts with the right token (which may be wildcarded).*
- **bool check\_match** (const string &, const string &) const  
*check match between a (possibly) wildcarded string and a literal.*
- **void dump** () const

## Private Attributes

- **int size**  
*dimension of the chart table (length of the sentence to parse)*
- **const grammar \* gram**

### 8.10.1 Detailed Description

Class chart contains an array of cells that constitute a chart.

### 8.10.2 Constructor & Destructor Documentation

#### 8.10.2.1 chart::chart ()

constructor

### 8.10.3 Member Function Documentation

#### 8.10.3.1 bool chart::better\_edge (const edge &, const edge &) const [private]

compare two edges when extracting a tree

#### 8.10.3.2 bool chart::can\_extend (const string &, int, int) const [private]

find out whether the cell (i,j) has some inactive edge whose head is the given category

**8.10.3.3    `bool chart::check_match (const string &, const string &) const`    [private]**

check match between a (possibly) wildcarded string and a literal.

**8.10.3.4    `list<pair<int,int> > chart::cover (int a, int b) const`    [private]**

obtain a list of cells that cover the subtree under cell (*a*,*b*)

**8.10.3.5    `void chart::dump () const`    [private]****8.10.3.6    `void chart::find_all_rules (const edge &, cell &, int, int) const`    [private]**

Complete edges in a cell (*ce*) after inserting a terminal or an inactive edge, using rules whose right part starts with the right token (which may be wildcarded).

**8.10.3.7    `cell chart::get_cell (int, int) const`**

get the contents of a cell in a given position

**8.10.3.8    `int chart::get_size () const`**

Get size of the table.

**8.10.3.9    `parse_tree chart::get_tree (int, int, const string & = "") const`**

navigate through the chart and obtain a parse tree.

**8.10.3.10    `int chart::index (int i, int j) const`    [private]**

compute position of the cell inside the vector

**8.10.3.11    `void chart::load_sentence (const sentence &)`**

load sentence and init parsing (fill up first row of chart)

**8.10.3.12    `void chart::parse ()`**

Do the parsing.

**8.10.3.13    `void chart::set_grammar (const grammar &)`**

set grammar to use in parsing

## 8.10.4 Member Data Documentation

**8.10.4.1** `const grammar* chart::gram` [private]

**8.10.4.2** `int chart::size` [private]

dimension of the chart table (length of the sentece to parse)

The documentation for this class was generated from the following file:

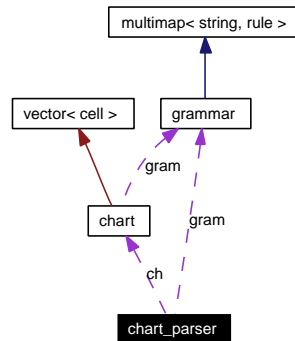
- `chart.h`

## 8.11 chart\_parser Class Reference

Class chart\_parser implements a chart parser.

```
#include <chart_parser.h>
```

Collaboration diagram for chart\_parser:



### Public Member Functions

- **chart\_parser** (const string &)  
*Constructors.*
- string **get\_start\_symbol** (void) const  
*Get the start symbol of the grammar.*
- void **analyze** (list< sentence > &)  
*Parse sentences in list.*
- list< sentence > **analyze** (const list< sentence > &)  
*Parse sentences in list, return copy.*

### Private Attributes

- **chart ch**  
*Part of the rule already matched.*
- **grammar gram**

#### 8.11.1 Detailed Description

Class chart\_parser implements a chart parser.

#### 8.11.2 Constructor & Destructor Documentation

##### 8.11.2.1 chart\_parser::chart\_parser (const string &)

Constructors.



### 8.11.3 Member Function Documentation

#### 8.11.3.1 list<sentence> chart\_parser::analyze (const list< sentence > &)

Parse sentences in list, return copy.

#### 8.11.3.2 void chart\_parser::analyze (list< sentence > &)

Parse sentences in list.

#### 8.11.3.3 string chart\_parser::get\_start\_symbol (void) const

Get the start symbol of the grammar.

### 8.11.4 Member Data Documentation

#### 8.11.4.1 chart chart\_parser::ch [private]

Part of the rule already matched.

#### 8.11.4.2 grammar chart\_parser::gram [private]

The documentation for this class was generated from the following file:

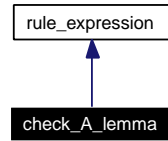
- chart\_parser.h

## 8.12 check\_A\_lemma Class Reference

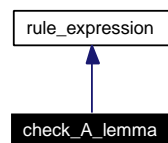
ancestor lemma in list of lemmas (separator character is |)

```
#include <dep_rules.h>
```

Inheritance diagram for check\_A\_lemma:



Collaboration diagram for check\_A\_lemma:



### Public Member Functions

- **check\_A\_lemma** (const string &)  
*check\_A\_lemma*
- **bool eval** (dep\_tree::iterator, dep\_tree::iterator) const

### Static Public Member Functions

- static **rule\_expression \* create** (const string &)

#### 8.12.1 Detailed Description

ancestor lemma in list of lemmas (separator character is |)

#### 8.12.2 Constructor & Destructor Documentation

##### 8.12.2.1 check\_A\_lemma::check\_A\_lemma (const string &)

`check_A_lemma`

### 8.12.3 Member Function Documentation

**8.12.3.1** `rule_expression * check_A_lemma::create (const string &) [static]`

**8.12.3.2** `bool check_A_lemma::eval (dep_tree::iterator, dep_tree::iterator) const [virtual]`

Implements `rule_expression` (p. 212).

The documentation for this class was generated from the following files:

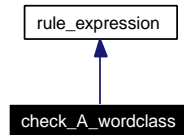
- `dep_rules.h`
- `dep_rules.cc`

## 8.13 check\_A\_wordclass Class Reference

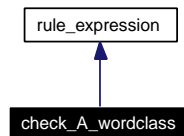
ancestor lemma belongs to a verb class

```
#include <dep_rules.h>
```

Inheritance diagram for check\_A\_wordclass:



Collaboration diagram for check\_A\_wordclass:



### Public Member Functions

- **check\_A\_wordclass** (const string &)  
*check\_A\_wordclass*
- **bool eval** (dep\_tree::iterator, dep\_tree::iterator) const

### Static Public Member Functions

- static **rule\_expression \* create** (const string &)

### Static Public Attributes

- static set< string > **wordclasses**

### Private Attributes

- string **wclass**

#### 8.13.1 Detailed Description

ancestor lemma belongs to a verb class

#### 8.13.2 Constructor & Destructor Documentation

##### 8.13.2.1 check\_A\_wordclass::check\_A\_wordclass (const string &)

check\_A\_wordclass

### 8.13.3 Member Function Documentation

8.13.3.1 `rule_expression * check_A_wordclass::create (const string &) [static]`

8.13.3.2 `bool check_A_wordclass::eval (dep_tree::iterator, dep_tree::iterator)  
const [virtual]`

Implements `rule_expression` (p. 212).

### 8.13.4 Member Data Documentation

8.13.4.1 `string check_A_wordclass::wclass [private]`

8.13.4.2 `set< string > check_A_wordclass::wordclasses [static]`

The documentation for this class was generated from the following files:

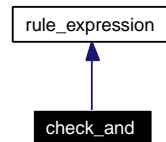
- `dep_rules.h`
- `dep_rules.cc`
- `dependencies.cc`

## 8.14 check\_and Class Reference

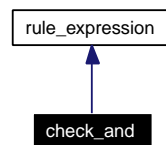
And of basic constraints.

```
#include <dep_rules.h>
```

Inheritance diagram for check\_and:



Collaboration diagram for check\_and:



### Public Member Functions

- void **add** (**rule\_expression** \*)  
*check\_and*
- bool **eval** (**dep\_tree::iterator** ancestor, **dep\_tree::iterator** descendant) const

### Private Attributes

- list< **rule\_expression** \* > **check\_list**

#### 8.14.1 Detailed Description

And of basic constraints.

#### 8.14.2 Member Function Documentation

##### 8.14.2.1 void check\_and::add (rule\_expression \*)

*check\_and*

##### 8.14.2.2 bool check\_and::eval (dep\_tree::iterator *ancestor*, dep\_tree::iterator *descendant*) const [virtual]

Implements **rule\_expression** (p. 212).

### 8.14.3 Member Data Documentation

#### 8.14.3.1 list<rule\_expression \*> check\_and::check\_list [private]

The documentation for this class was generated from the following files:

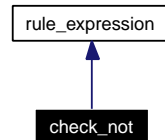
- dep\_rules.h
- dep\_rules.cc

## 8.15 check\_not Class Reference

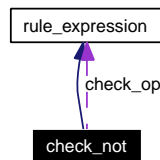
negation

```
#include <dep_rules.h>
```

Inheritance diagram for check\_not:



Collaboration diagram for check\_not:



### Public Member Functions

- `check_not (rule_expression *)`  
*check\_not*
- `bool eval (dep_tree::iterator, dep_tree::iterator) const`

### Private Attributes

- `rule_expression * check_op`

#### 8.15.1 Detailed Description

negation

#### 8.15.2 Constructor & Destructor Documentation

##### 8.15.2.1 check\_not::check\_not (rule\_expression \*)

`check_not`

#### 8.15.3 Member Function Documentation

##### 8.15.3.1 bool check\_not::eval (dep\_tree::iterator, dep\_tree::iterator) const [virtual]

Implements `rule_expression` (p. 212).



## 8.15.4 Member Data Documentation

### 8.15.4.1 `rule_expression* check_not::check_op` [private]

The documentation for this class was generated from the following files:

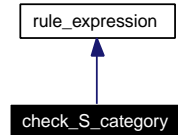
- `dep_rules.h`
- `dep_rules.cc`

## 8.16 check\_\_S\_\_category Class Reference

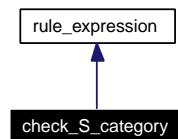
ancestor category

```
#include <dep_rules.h>
```

Inheritance diagram for check\_\_S\_\_category:



Collaboration diagram for check\_\_S\_\_category:



### Public Member Functions

- **check\_\_S\_\_category** (const string &)  
*check\_\_S\_\_category*
- **bool eval** (dep\_\_tree::iterator, dep\_\_tree::iterator) const

### Static Public Member Functions

- static **rule\_\_expression \* create** (const string &)

### Private Attributes

- string **catg**

#### 8.16.1 Detailed Description

ancestor category

#### 8.16.2 Constructor & Destructor Documentation

##### 8.16.2.1 check\_\_S\_\_category::check\_\_S\_\_category (const string &)

check\_\_S\_\_category

### 8.16.3 Member Function Documentation

**8.16.3.1** `rule_expression * check_S_category::create (const string &) [static]`

**8.16.3.2** `bool check_S_category::eval (dep_tree::iterator, dep_tree::iterator)  
const [virtual]`

Implements `rule_expression` (p. 212).

### 8.16.4 Member Data Documentation

**8.16.4.1** `string check_S_category::catg [private]`

The documentation for this class was generated from the following files:

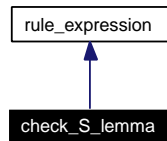
- `dep_rules.h`
- `dep_rules.cc`

## 8.17 check\_S\_lemma Class Reference

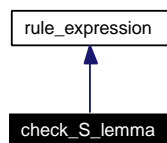
descendant lemma in list of lemmas (separator character is |)

```
#include <dep_rules.h>
```

Inheritance diagram for check\_S\_lemma:



Collaboration diagram for check\_S\_lemma:



### Public Member Functions

- **check\_S\_lemma** (const string &)  
*check\_S\_lemma*
- **bool eval** (dep\_tree::iterator, dep\_tree::iterator) const

### Static Public Member Functions

- static **rule\_expression \* create** (const string &)

#### 8.17.1 Detailed Description

descendant lemma in list of lemmas (separator character is |)

#### 8.17.2 Constructor & Destructor Documentation

##### 8.17.2.1 check\_S\_lemma::check\_S\_lemma (const string &)

check\_S\_lemma

### 8.17.3 Member Function Documentation

**8.17.3.1** `rule_expression * check_S_lemma::create (const string &) [static]`

**8.17.3.2** `bool check_S_lemma::eval (dep_tree::iterator, dep_tree::iterator) const [virtual]`

Implements `rule_expression` (p. 212).

The documentation for this class was generated from the following files:

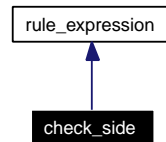
- `dep_rules.h`
- `dep_rules.cc`

## 8.18 check\_side Class Reference

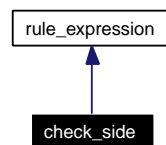
descendant is to the right of its ancestor

```
#include <dep_rules.h>
```

Inheritance diagram for check\_side:



Collaboration diagram for check\_side:



### Public Member Functions

- **check\_side** (const string &)  
*check\_side*
- **bool eval** (dep\_tree::iterator, dep\_tree::iterator) const

### Static Public Member Functions

- static **rule\_expression \* create** (const string &)

### Private Attributes

- string **side**

#### 8.18.1 Detailed Description

descendant is to the right of its ancestor

#### 8.18.2 Constructor & Destructor Documentation

##### 8.18.2.1 check\_side::check\_side (const string &)

check\_side

### 8.18.3 Member Function Documentation

**8.18.3.1** `rule_expression * check_side::create (const string &) [static]`

**8.18.3.2** `bool check_side::eval (dep_tree::iterator, dep_tree::iterator) const [virtual]`

Implements `rule_expression` (p. 212).

### 8.18.4 Member Data Documentation

**8.18.4.1** `string check_side::side [private]`

The documentation for this class was generated from the following files:

- `dep_rules.h`
- `dep_rules.cc`

## 8.19 completer Class Reference

The class completer implements a parse tree completer, which given a partial parse tree (chunker output), completes the full parse according to some grammar rules.

```
#include <dependencies.h>
```

### Public Member Functions

- **completer** (const string &)  
*Constructor. Load a tree-completion grammar.*
- **parse\_tree complete** (parse\_tree &, const string &)  
*complete given parse tree*

### Private Member Functions

- **parse\_tree::iterator find\_last\_label** (parse\_tree \*, const string &) const  
*find last node with given label in a subtree*
- **completerRule gramatica** (const string &, const string &)  
*retrieve rule from grammar*
- **parse\_tree \* applyRule** (const completerRule &, parse\_tree \*, parse\_tree \*)  
*apply a completion rule*

### Private Attributes

- map< pair< string, string >, completerRule > **chgram**  
*set of rules, indexed by labels of nodes*

#### 8.19.1 Detailed Description

The class completer implements a parse tree completer, which given a partial parse tree (chunker output), completes the full parse according to some grammar rules.

#### 8.19.2 Constructor & Destructor Documentation

##### 8.19.2.1 completer::completer (const string & *filename*)

Constructor. Load a tree-completion grammar.

Load a tree-completion grammar



### 8.19.3 Member Function Documentation

**8.19.3.1** `parse_tree * completer::applyRule (const completerRule &, parse_tree *, parse_tree *)` [private]

apply a completion rule

**8.19.3.2** `parse_tree completer::complete (parse_tree &, const string &)`

complete given parse tree

**8.19.3.3** `parse_tree::iterator completer::find_last_label (parse_tree *, const string &) const` [private]

find last node with given label in a subtree

**8.19.3.4** `completerRule completer::gramatica (const string &, const string &)` [private]

retrieve rule from grammar

### 8.19.4 Member Data Documentation

**8.19.4.1** `map<pair<string,string>,completerRule> completer::chgram` [private]

set of rules, indexed by labels of nodes

The documentation for this class was generated from the following files:

- `dependencies.h`
- `dependencies.cc`

## 8.20 completerRule Class Reference

The class `completerRule` stores rules used by the completer of parse trees.

```
#include <dep_rules.h>
```

### Public Member Functions

- **completerRule ()**  
*constructors*
- **completerRule (const string &, const string &)**  
*Constructor.*
- **completerRule (const completerRule &)**  
*Constructor.*
- **completerRule & operator= (const completerRule &)**  
*assignment*
- **int operator< (const completerRule &a) const**  
*Comparison. The more weight the higher priority.*

### Public Attributes

- string **chunk**
- string **leftChunk**
- string **newNode**
- string **operation**
- string **dependenceLabel**
- int **weight**

#### 8.20.1 Detailed Description

The class `completerRule` stores rules used by the completer of parse trees.

#### 8.20.2 Constructor & Destructor Documentation

##### 8.20.2.1 completerRule::completerRule ()

constructors

##### 8.20.2.2 completerRule::completerRule (const string &, const string &)

Constructor.

### 8.20.2.3 completerRule::completerRule (const completerRule &)

Constructor.

## 8.20.3 Member Function Documentation

### 8.20.3.1 int completerRule::operator< (const completerRule & a) const

Comparison. The more weight the higher priority.

The more weight the higher priority

### 8.20.3.2 completerRule & completerRule::operator= (const completerRule &)

assignment

## 8.20.4 Member Data Documentation

### 8.20.4.1 string completerRule::chunk

### 8.20.4.2 string completerRule::dependenceLabel

### 8.20.4.3 string completerRule::leftChunk

### 8.20.4.4 string completerRule::newNode

### 8.20.4.5 string completerRule::operation

### 8.20.4.6 int completerRule::weight

The documentation for this class was generated from the following files:

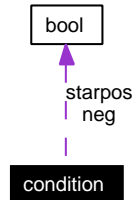
- `dep_rules.h`
- `dep_rules.cc`

## 8.21 condition Class Reference

Class condition implements a condition of a CG rule.

```
#include <constraint_grammar.h>
```

Collaboration diagram for condition:



### Public Member Functions

- **condition** ()  
*constructor*
- void **clear** ()  
*empty condition*
- void **set\_neg** (bool)  
*set negative condition flag*
- void **set\_pos** (int, bool)  
*set position value*
- void **set\_terms** (const list< string > &)  
*set terms list*
- void **set\_barrier** (const list< string > &)  
*set barrier terms list*
- **bool is\_neg** () const  
*find out whether it is a negative condition*
- **int get\_pos** () const  
*get position*
- **bool has\_star** () const  
*find out whethe position has a "\*"*
- **list< string > get\_terms** () const  
*get terms to check*
- **bool has\_barrier** () const  
*find out if there are barrier terms*

- **list< string > get\_\_barrier () const**  
*get barrier terms*

## Private Attributes

- **bool neg**  
*is it a negative condition?*
- **int pos**  
*position to check for*
- **bool starpos**  
*star in the position*
- **list< string > terms**  
*terms ORed in the condition (each term is a <lemma>, (form), or TAG)*
- **list< string > barrier**  
*terms in barrier (if any)*

### 8.21.1 Detailed Description

Class condition implements a condition of a CG rule.

### 8.21.2 Constructor & Destructor Documentation

#### 8.21.2.1 condition::condition ()

constructor

### 8.21.3 Member Function Documentation

#### 8.21.3.1 void condition::clear ()

empty condition

#### 8.21.3.2 list<string> condition::get\_\_barrier () const

get barrier terms

#### 8.21.3.3 int condition::get\_\_pos () const

get position

**8.21.3.4** `list<string> condition::get _terms () const`

get terms to check

**8.21.3.5** `bool condition::has _barrier () const`

find out if there are barrier terms

**8.21.3.6** `bool condition::has _star () const`

find out whethe position has a "\*"

**8.21.3.7** `bool condition::is _neg () const`

find out whether it is a negative condition

**8.21.3.8** `void condition::set _barrier (const list< string > &)`

set barrier terms list

**8.21.3.9** `void condition::set _neg (bool)`

set negative condition flag

**8.21.3.10** `void condition::set _pos (int, bool)`

set position value

**8.21.3.11** `void condition::set _terms (const list< string > &)`

set terms list

**8.21.4** **Member Data Documentation****8.21.4.1** `list<string> condition::barrier [private]`

terms in barrier (if any)

**8.21.4.2** `bool condition::neg [private]`

is it a negative condition?

**8.21.4.3** `int condition::pos [private]`

position to check for

**8.21.4.4** `bool condition::starpos` [private]

star in the position

**8.21.4.5** `list<string> condition::terms` [private]

terms ORed in the condition (each term is a <lemma>, (form), or TAG)

The documentation for this class was generated from the following file:

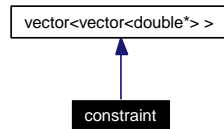
- `constraint_grammar.h`

## 8.22 constraint Class Reference

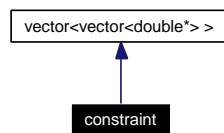
The class constraint implements a constraint for the relaxation labelling algorithm.

```
#include <relax.h>
```

Inheritance diagram for constraint:



Collaboration diagram for constraint:



### Public Member Functions

- **constraint** ()  
*Constructor.*
- void **set\_**compatibility (double)  
*set/get compatibility value*
- double **get\_**compatibility () const

### Private Attributes

- double **compatibility**

#### 8.22.1 Detailed Description

The class constraint implements a constraint for the relaxation labelling algorithm.

#### 8.22.2 Constructor & Destructor Documentation

##### 8.22.2.1 constraint::constraint ()

Constructor.



### 8.22.3 Member Function Documentation

**8.22.3.1** `double constraint::get_compatibility () const`

**8.22.3.2** `void constraint::set_compatibility (double)`

set/get compatibility value

### 8.22.4 Member Data Documentation

**8.22.4.1** `double constraint::compatibility [private]`

The documentation for this class was generated from the following file:

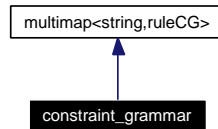
- `relax.h`

## 8.23 constraint\_grammar Class Reference

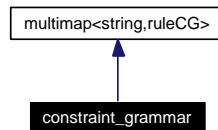
Class `constraint_grammar` implements a pseudo CG, ready to be used from a relax PoS tagger.

```
#include <constraint_grammar.h>
```

Inheritance diagram for `constraint_grammar`:



Collaboration diagram for `constraint_grammar`:



### Public Member Functions

- **constraint\_grammar** (const string &)  
*Create a grammar loading it from a file.*
- void **get\_rules\_head** (const string &, list< ruleCG > &) const  
*add to the given list all rules with a head starting with the given string*

#### 8.23.1 Detailed Description

Class `constraint_grammar` implements a pseudo CG, ready to be used from a relax PoS tagger.

#### 8.23.2 Constructor & Destructor Documentation

##### 8.23.2.1 constraint\_grammar::constraint\_grammar (const string &)

Create a grammar loading it from a file.

#### 8.23.3 Member Function Documentation

##### 8.23.3.1 void constraint\_grammar::get\_rules\_head (const string &, list< ruleCG > &) const

add to the given list all rules with a head starting with the given string

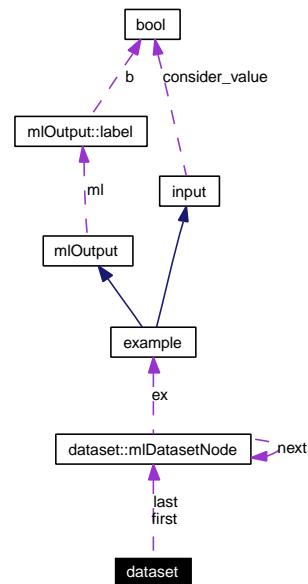
The documentation for this class was generated from the following file:

- **constraint\_grammar.h**

## 8.24 dataset Class Reference

```
#include <dataset.h>
```

Collaboration diagram for dataset:



### Public Member Functions

- **dataset** (int nlabels)
- **~dataset** ()
- void **delete\_examples** ()
- void **read\_stream** (istream &)
- void **add\_example** (example \*, mlDatasetNode \* = NULL)
- int **size** () const
- int **negative\_size** (int l) const
- int **positive\_size** (int l) const
- int **nlabels** () const
- int **dimension** () const
- **iterator begin** () const
- **iterator end** () const
- void **split** (int feature, dataset \*ds0, dataset \*ds1, bool create\_ds\_nodes)
- void **print** (ostream &) const
- void **print\_sizes** (ostream &) const

### Private Attributes

- mlDatasetNode \* **first**
- mlDatasetNode \* **last**
- int **\_size**
- int \* **\_sizes**
- int **\_dimension**
- int **\_nlabels**

## Classes

- class **iterator**

- struct **mlDatasetNode**

## 8.24.1 Constructor & Destructor Documentation

8.24.1.1 `dataset::dataset (int nlabels)`

8.24.1.2 `dataset::~~dataset ()`

## 8.24.2 Member Function Documentation

8.24.2.1 `void dataset::add_example (example *, mlDatasetNode * = NULL)`

8.24.2.2 `iterator dataset::begin () const [inline]`

8.24.2.3 `void dataset::delete_examples ()`

8.24.2.4 `int dataset::dimension () const [inline]`

8.24.2.5 `iterator dataset::end () const [inline]`

8.24.2.6 `int dataset::negative_size (int l) const [inline]`

8.24.2.7 `int dataset::nlabels () const [inline]`

8.24.2.8 `int dataset::positive_size (int l) const [inline]`

8.24.2.9 `void dataset::print (ostream &) const`

8.24.2.10 `void dataset::print_sizes (ostream &) const`

8.24.2.11 `void dataset::read_stream (istream &)`

8.24.2.12 `int dataset::size () const [inline]`

8.24.2.13 `void dataset::split (int feature, dataset * ds0, dataset * ds1, bool create_ds_nodes)`

## 8.24.3 Member Data Documentation

8.24.3.1 `int dataset::_dimension [private]`

8.24.3.2 `int dataset::_nlabels [private]`

8.24.3.3 `int dataset::_size [private]`

8.24.3.4 `int* dataset::_sizes [private]`

8.24.3.5 `mlDatasetNode* dataset::first [private]`

8.24.3.6 `mlDatasetNode* dataset::last [private]`

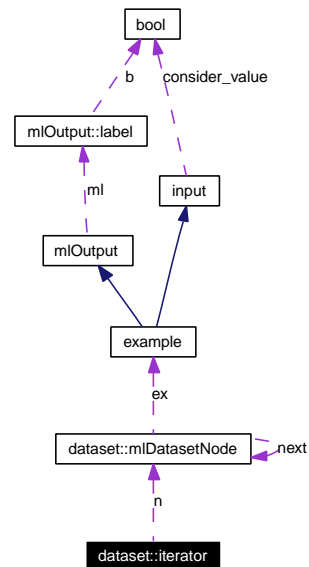
The documentation for this class was generated from the following file:

- `dataset.h`

## 8.25 dataset::iterator Class Reference

```
#include <dataset.h>
```

Collaboration diagram for dataset::iterator:



### Public Member Functions

- `iterator ()`
- `iterator (mlDatasetNode *n0)`
- `example * operator → () const`
- `example & operator * () const`
- `iterator & operator ++ ()`
- `iterator operator ++ (int)`
- `bool operator == (const iterator &rhs)`
- `bool operator != (const iterator &rhs)`

### Private Attributes

- `mlDatasetNode * n`

## 8.25.1 Constructor & Destructor Documentation

8.25.1.1 dataset::iterator::iterator () [inline]

8.25.1.2 dataset::iterator::iterator (mlDatasetNode \* *n0*) [inline]

## 8.25.2 Member Function Documentation

8.25.2.1 example& dataset::iterator::operator \* () const [inline]

8.25.2.2 bool dataset::iterator::operator!= (const iterator & *rhs*) [inline]

8.25.2.3 iterator dataset::iterator::operator++ (int) [inline]

8.25.2.4 iterator& dataset::iterator::operator++ () [inline]

8.25.2.5 example\* dataset::iterator::operator → () const [inline]

8.25.2.6 bool dataset::iterator::operator== (const iterator & *rhs*) [inline]

## 8.25.3 Member Data Documentation

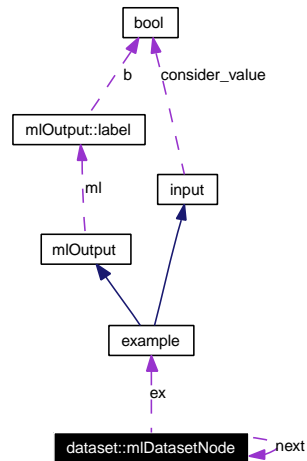
8.25.3.1 mlDatasetNode\* dataset::iterator::n [private]

The documentation for this class was generated from the following file:

- dataset.h

## 8.26 dataset::mlDatasetNode Struct Reference

Collaboration diagram for dataset::mlDatasetNode:



### Public Member Functions

- `mlDatasetNode ()`
- `mlDatasetNode (example *e)`

### Public Attributes

- `example * ex`
- `mlDatasetNode * next`

### 8.26.1 Constructor & Destructor Documentation

8.26.1.1 `dataset::mlDatasetNode::mlDatasetNode () [inline]`

8.26.1.2 `dataset::mlDatasetNode::mlDatasetNode (example * e) [inline]`

### 8.26.2 Member Data Documentation

8.26.2.1 `example* dataset::mlDatasetNode::ex`

8.26.2.2 `mlDatasetNode* dataset::mlDatasetNode::next`

The documentation for this struct was generated from the following file:

- `dataset.h`

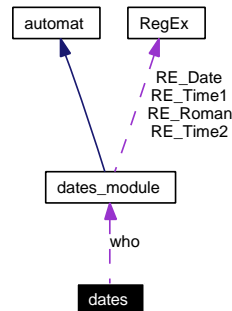


## 8.27 dates Class Reference

The class `dates` provides a wrapper to transparently create and access a `dates_module`(p.89), a temporal expression recognizer for the appropriate language.

```
#include <dates.h>
```

Collaboration diagram for `dates`:



### Public Member Functions

- `dates (const maco_options &)`  
*Constructor.*
- `~dates ()`  
*Destructor.*
- `void annotate (sentence &)`  
*Detect date/time expressions in sentence using default options.*

### Private Attributes

- `dates_module * who`  
*remember which module is doing the real work.*

#### 8.27.1 Detailed Description

The class `dates` provides a wrapper to transparently create and access a `dates_module`(p.89), a temporal expression recognizer for the appropriate language.

#### 8.27.2 Constructor & Destructor Documentation

##### 8.27.2.1 `dates::dates (const maco_options &)`

Constructor.

### 8.27.2.2 `dates::~~dates ()`

Destructor.

## 8.27.3 Member Function Documentation

### 8.27.3.1 `void dates::annotate (sentence &)`

Detect date/time expressions in sentence using default options.

## 8.27.4 Member Data Documentation

### 8.27.4.1 `dates_module* dates::who` [private]

remember which module is doing the real work.

The documentation for this class was generated from the following files:

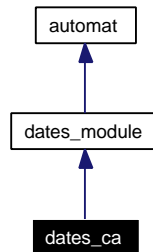
- `dates.h`
- `dates.cc`

## 8.28 dates\_\_ca Class Reference

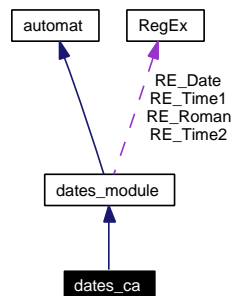
The derived class `dates__ca` implements a Catalan date/time recognizer.

```
#include <dates_modules.h>
```

Inheritance diagram for `dates__ca`:



Collaboration diagram for `dates__ca`:



### Public Member Functions

- `dates__ca ()`  
*Constructor.*

### Private Member Functions

- `int ComputeToken (int, sentence::const_iterator, const sentence &)`  
*Compute the right token code for word j from given state.*
- `void ResetActions ()`  
*Reset accumulators used by state actions: day, year, month, hour, minute, etc.*
- `void StateActions (int, int, int, sentence::const_iterator)`  
*Perform necessary actions in "state" reached from state "origin" via word j interpreted as code "token": Basically, when reaching a state with an informative token (day, year, month, etc) store that part of the date.*
- `void SetMultiwordAnalysis (sentence::iterator) const`  
*Set the appropriate lemma and parole for the new multiword.*

### 8.28.1 Detailed Description

The derived class `dates_ca` implements a Catalan date/time recognizer.

### 8.28.2 Constructor & Destructor Documentation

#### 8.28.2.1 `dates_ca::dates_ca ()`

Constructor.

### 8.28.3 Member Function Documentation

#### 8.28.3.1 `int dates_ca::ComputeToken (int, sentence::const_iterator, const sentence &) [private, virtual]`

Compute the right token code for word `j` from given state.

Implements **automat** (p. 41).

#### 8.28.3.2 `void dates_ca::ResetActions () [private, virtual]`

Reset accumulators used by state actions: day, year, month, hour, minute, etc.

Implements **automat** (p. 41).

#### 8.28.3.3 `void dates_ca::SetMultiwordAnalysis (sentence::iterator) const [private, virtual]`

Set the appropriate lemma and parole for the new multiword.

Implements **automat** (p. 41).

#### 8.28.3.4 `void dates_ca::StateActions (int, int, int, sentence::const_iterator) [private, virtual]`

Perform necessary actions in "state" reached from state "origin" via word `j` interpreted as code "token": Basically, when reaching a state with an informative token (day, year, month, etc) store that part of the date.

Implements **automat** (p. 42).

The documentation for this class was generated from the following files:

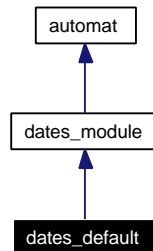
- `dates_modules.h`
- `dates_modules.cc`

## 8.29 dates\_\_default Class Reference

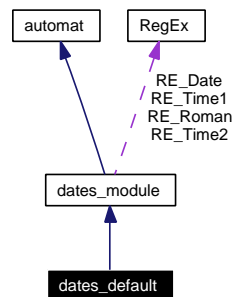
The derived class `dates__default` implements a default date/time recognizer (only simple patterns are recognized).

```
#include <dates_modules.h>
```

Inheritance diagram for `dates__default`:



Collaboration diagram for `dates__default`:



### Public Member Functions

- `dates__default ()`  
*Constructor.*

### Private Member Functions

- `int ComputeToken (int, sentence::const_iterator, const sentence &)`  
*Compute the right token code for word j from given state.*
- `void ResetActions ()`  
*Reset accumulators used by state actions: day, year, month, hour, minute, etc.*
- `void StateActions (int, int, int, sentence::const_iterator)`  
*Perform necessary actions in "state" reached from state "origin" via word j interpreted as code "token": Basically, when reaching a state with an informative token (day, year, month, etc) store that part of the date.*
- `void SetMultiwordAnalysis (sentence::iterator) const`

*Set the appropriate lemma and parole for the new multiword.*

### 8.29.1 Detailed Description

The derived class `dates__default` implements a default date/time recognizer (only simple patterns are recognized).

### 8.29.2 Constructor & Destructor Documentation

#### 8.29.2.1 `dates__default::dates__default ()`

Constructor.

### 8.29.3 Member Function Documentation

#### 8.29.3.1 `int dates__default::ComputeToken (int, sentence::const_iterator, const sentence &) [private, virtual]`

Compute the right token code for word `j` from given state.

Implements **automat** (p. 41).

#### 8.29.3.2 `void dates__default::ResetActions () [private, virtual]`

Reset accumulators used by state actions: day, year, month, hour, minute, etc.

Implements **automat** (p. 41).

#### 8.29.3.3 `void dates__default::SetMultiwordAnalysis (sentence::iterator) const [private, virtual]`

Set the appropriate lemma and parole for the new multiword.

Implements **automat** (p. 41).

#### 8.29.3.4 `void dates__default::StateActions (int, int, int, sentence::const_iterator) [private, virtual]`

Perform necessary actions in "state" reached from state "origin" via word `j` interpreted as code "token": Basically, when reaching a state with an informative token (day, year, month, etc) store that part of the date.

Implements **automat** (p. 42).

The documentation for this class was generated from the following files:

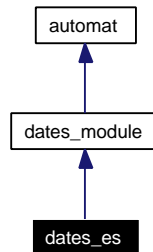
- `dates__modules.h`
- `dates__modules.cc`

## 8.30 dates\_es Class Reference

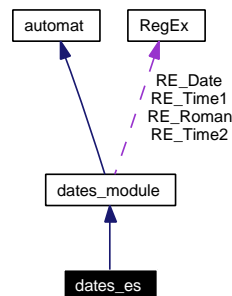
The derived class `dates_es` implements a Spanish date/time recognizer.

```
#include <dates_modules.h>
```

Inheritance diagram for `dates_es`:



Collaboration diagram for `dates_es`:



### Public Member Functions

- `dates_es ()`  
*Constructor.*

### Private Member Functions

- `int ComputeToken (int, sentence::const_iterator, const sentence &)`  
*Compute the right token code for word j from given state.*
- `void ResetActions ()`  
*Reset accumulators used by state actions: day, year, month, hour, minute, etc.*
- `void StateActions (int, int, int, sentence::const_iterator)`  
*Perform necessary actions in "state" reached from state "origin" via word j interpreted as code "token": Basically, when reaching a state with an informative token (day, year, month, etc) store that part of the date.*
- `void SetMultiwordAnalysis (sentence::iterator) const`  
*Set the appropriate lemma and parole for the new multiword.*

### 8.30.1 Detailed Description

The derived class `dates_es` implements a Spanish date/time recognizer.

### 8.30.2 Constructor & Destructor Documentation

#### 8.30.2.1 `dates_es::dates_es ()`

Constructor.

### 8.30.3 Member Function Documentation

#### 8.30.3.1 `int dates_es::ComputeToken (int, sentence::const_iterator, const sentence &) [private, virtual]`

Compute the right token code for word `j` from given state.

Implements **automat** (p. 41).

#### 8.30.3.2 `void dates_es::ResetActions () [private, virtual]`

Reset accumulators used by state actions: day, year, month, hour, minute, etc.

Implements **automat** (p. 41).

#### 8.30.3.3 `void dates_es::SetMultiwordAnalysis (sentence::iterator) const [private, virtual]`

Set the appropriate lemma and parole for the new multiword.

Implements **automat** (p. 41).

#### 8.30.3.4 `void dates_es::StateActions (int, int, int, sentence::const_iterator) [private, virtual]`

Perform necessary actions in "state" reached from state "origin" via word `j` interpreted as code "token": Basically, when reaching a state with an informative token (day, year, month, etc) store that part of the date.

Implements **automat** (p. 42).

The documentation for this class was generated from the following files:

- `dates_modules.h`
- `dates_modules.cc`

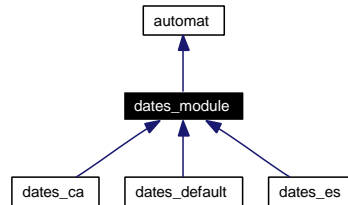


## 8.31 dates\_\_module Class Reference

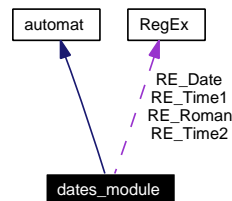
The abstract class `dates__module` generalizes temporal expression recognizer for different languages.

```
#include <dates_modules.h>
```

Inheritance diagram for `dates__module`:



Collaboration diagram for `dates__module`:



### Public Member Functions

- **dates\_\_module** (const string &, const string &, const string &, const string &)  
*Constructor.*

### Protected Attributes

- map< string, int > **nMes**  
*translate month names to numbers*
- map< string, string > **nDia**  
*translate weekday names to unified codes*
- string **century**  
*interpretation of the date-time expression*
- string **year**
- string **month**
- string **day**
- string **weekday**
- string **hour**
- string **minute**
- string **meridian**

- **int temp**  
*auxiliary for interpretation building*
- **int sign**
- **map< string, int > tok**  
*translate particular strings to token codes*
- **RegEx RE\_Date**
- **RegEx RE\_Time1**
- **RegEx RE\_Time2**
- **RegEx RE\_Roman**

### 8.31.1 Detailed Description

The abstract class `dates_module` generalizes temporal expression recognizer for different languages.

### 8.31.2 Constructor & Destructor Documentation

#### 8.31.2.1 `dates_module::dates_module (const string &, const string &, const string &, const string &)`

Constructor.

### 8.31.3 Member Data Documentation

#### 8.31.3.1 `string dates_module::century` [protected]

interpretation of the date-time expression

#### 8.31.3.2 `string dates_module::day` [protected]

#### 8.31.3.3 `string dates_module::hour` [protected]

#### 8.31.3.4 `string dates_module::meridian` [protected]

#### 8.31.3.5 `string dates_module::minute` [protected]

#### 8.31.3.6 `string dates_module::month` [protected]

#### 8.31.3.7 `map<string,string> dates_module::nDia` [protected]

translate weekday names to unified codes

#### 8.31.3.8 `map<string,int> dates_module::nMes` [protected]

translate month names to numbers

**8.31.3.9**   `RegEx dates__module::RE_Date`   [protected]

**8.31.3.10**   `RegEx dates__module::RE_Roman`   [protected]

**8.31.3.11**   `RegEx dates__module::RE_Time1`   [protected]

**8.31.3.12**   `RegEx dates__module::RE_Time2`   [protected]

**8.31.3.13**   `int dates__module::sign`   [protected]

**8.31.3.14**   `int dates__module::temp`   [protected]

auxiliary for interpretation building

**8.31.3.15**   `map<string,int> dates__module::tok`   [protected]

translate particular strings to token codes

**8.31.3.16**   `string dates__module::weekday`   [protected]

**8.31.3.17**   `string dates__module::year`   [protected]

The documentation for this class was generated from the following files:

- `dates__modules.h`
- `dates__modules.cc`

## 8.32 dep\_info Class Reference

auxiliary class to store information about dependency building

```
#include <dep_rules.h>
```

### Public Member Functions

- **dep\_info** (void)  
*Constructor.*
- **dep\_info** (const string &, const string &, const string &)  
*Constructor.*
- string **get\_dep\_source** (void)  
*get/set dependency attributes*
- string **get\_dep\_target** (void)
- string **get\_dep\_result** (void)
- void **set\_dep\_source** (const string &)
- void **set\_dep\_target** (const string &)
- void **set\_dep\_result** (const string &)

### Private Attributes

- string **dep\_source**
- string **dep\_target**
- string **dep\_result**

#### 8.32.1 Detailed Description

auxiliary class to store information about dependency building

#### 8.32.2 Constructor & Destructor Documentation

##### 8.32.2.1 dep\_info::dep\_info (void)

Constructor.

##### 8.32.2.2 dep\_info::dep\_info (const string &, const string &, const string &)

Constructor.

#### 8.32.3 Member Function Documentation

##### 8.32.3.1 string dep\_info::get\_dep\_result (void)

##### 8.32.3.2 string dep\_info::get\_dep\_source (void)

get/set dependency attributes

8.32.3.3 string dep\_info::get\_dep\_target (void)

8.32.3.4 void dep\_info::set\_dep\_result (const string &)

8.32.3.5 void dep\_info::set\_dep\_source (const string &)

8.32.3.6 void dep\_info::set\_dep\_target (const string &)

## 8.32.4 Member Data Documentation

8.32.4.1 string dep\_info::dep\_result [private]

8.32.4.2 string dep\_info::dep\_source [private]

8.32.4.3 string dep\_info::dep\_target [private]

The documentation for this class was generated from the following files:

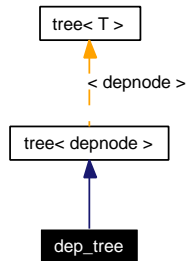
- dep\_rules.h
- dep\_rules.cc

## 8.33 dep\_tree Class Reference

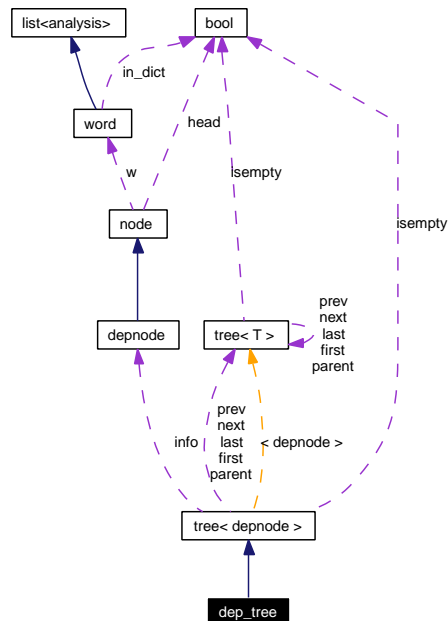
class dep\_tree stores a dependency tree

```
#include <language.h>
```

Inheritance diagram for dep\_tree:



Collaboration diagram for dep\_tree:



### Public Member Functions

- `dep_tree ()`  
*Constructors for dep\_tree.*
- `dep_tree (const depnode &)`

#### 8.33.1 Detailed Description

class dep\_tree stores a dependency tree

## 8.33.2 Constructor & Destructor Documentation

### 8.33.2.1 dep\_tree::dep\_tree ()

Constructors for dep\_tree.

### 8.33.2.2 dep\_tree::dep\_tree (const depnode &)

The documentation for this class was generated from the following files:

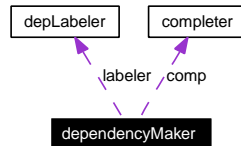
- language.h
- language.cc

## 8.34 dependencyMaker Class Reference

dependencyMaker is a class for obtaining a dependency tree from chunks.

```
#include <dependencies.h>
```

Collaboration diagram for dependencyMaker:



### Public Member Functions

- **dependencyMaker** (const string &, const string &)  
*constructor*
- void **analyze** (list< sentence > &)  
*Enrich all sentences in given list with a dependency tree.*
- list< sentence > **analyze** (const list< sentence > &)  
*Enrich all sentences in given list, return a copy.*

### Private Member Functions

- void **registerFunction** (const string &, expcreator \*)  
*register a new tree joining operation*
- dep\_tree \* **dependencies** (parse\_tree &tr)  
*compute dependency tree*

### Private Attributes

- **completer comp**  
*tree completer*
- **depLabeler labeler**  
*dependency labeler*
- string **start**

#### 8.34.1 Detailed Description

dependencyMaker is a class for obtaining a dependency tree from chunks.

this implementation uses two subclasses: **completer**: to complete the chunk analysis in a full parse tree **depLabeler**(p. 99): to set the labels once the class has build a dependency tree



## 8.34.2 Constructor & Destructor Documentation

### 8.34.2.1 `dependencyMaker::dependencyMaker (const string & fullgram, const string & startSymbol)`

constructor

Load a dependency rule file.

## 8.34.3 Member Function Documentation

### 8.34.3.1 `list< sentence > dependencyMaker::analyze (const list< sentence > & ls)`

Enrich all sentences in given list, return a copy.

Useful for Perl API

### 8.34.3.2 `void dependencyMaker::analyze (list< sentence > &)`

Enrich all sentences in given list with a dependency tree.

### 8.34.3.3 `dep_tree * dependencyMaker::dependencies (parse_tree & tr)` [private]

compute dependency tree

Auxiliar structure to keep and update dependencies

inverse index get node Id from a pointer to the node

Create dependency nodes

Build tree with created depnodes according to dependency list

### 8.34.3.4 `void dependencyMaker::registerFunction (const string &, expcreator *)` [private]

register a new tree joining operation

## 8.34.4 Member Data Documentation

### 8.34.4.1 `completer dependencyMaker::comp` [private]

tree completer

### 8.34.4.2 `depLabeler dependencyMaker::labeler` [private]

dependency labeler

### 8.34.4.3 `string dependencyMaker::start` [private]

The documentation for this class was generated from the following files:

- `dependencies.h`
- `dependencies.cc`

## 8.35 depLabeler Class Reference

depLabeler is class to set labels into a dependency tree

```
#include <dependencies.h>
```

### Public Member Functions

- **depLabeler** (const string &)  
*Constructor. Load dependency labelling rules.*
- void **label** (dep\_tree \*)  
*Label nodes in a dependcendy tree. (Initial call).*
- void **label** (dep\_tree \*, dep\_tree::iterator)  
*Label nodes in a dependcendy tree. (recursive).*
- void **registerFunction** (const string, expcreator \*)  
*register a new tree joining operation to be used in rule file*

### Private Attributes

- map< string, list< ruleLabeler > > **rules**
- map< string, expcreator \* > **expressions**

#### 8.35.1 Detailed Description

depLabeler is class to set labels into a dependency tree

#### 8.35.2 Constructor & Destructor Documentation

##### 8.35.2.1 depLabeler::depLabeler (const string &)

Constructor. Load dependency labelling rules.

#### 8.35.3 Member Function Documentation

##### 8.35.3.1 void depLabeler::label (dep\_tree \* *dependency*, dep\_tree::iterator *ancestor*)

Label nodes in a dependcendy tree. (recursive).

(recursive)

##### 8.35.3.2 void depLabeler::label (dep\_tree \* *dependency*)

Label nodes in a dependcendy tree. (Initial call).

(Initial call)

### 8.35.3.3 void depLabeler::registerFunction (const *string*, expcreator \*)

register a new tree joining operation to be used in rule file

## 8.35.4 Member Data Documentation

### 8.35.4.1 map<string, expcreator \*> depLabeler::expressions [private]

### 8.35.4.2 map<string, list <ruleLabeler> > depLabeler::rules [private]

The documentation for this class was generated from the following files:

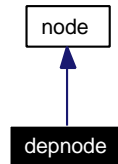
- dependencies.h
- dependencies.cc

## 8.36 depnode Class Reference

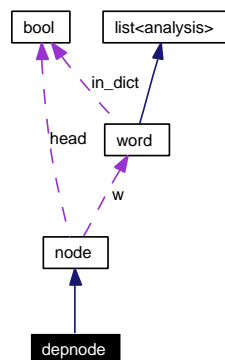
class depnode stores nodes of a dependency tree and parse tree <-> deptree relations

```
#include <language.h>
```

Inheritance diagram for depnode:



Collaboration diagram for depnode:



### Public Member Functions

- **depnode** ()  
*Methods for dependency tree nodes.*
- **depnode** (const string &)
- **depnode** (const **node** &)
- void **set\_link** (const **parse\_tree::iterator**)
- **parse\_tree::iterator** **get\_link** (void)
- void **set\_dep\_source** (const string &)
- void **set\_dep\_target** (const string &)
- void **set\_dep\_result** (const string &)
- string **get\_dep\_source** (void) const
- string **get\_dep\_target** (void) const
- string **get\_dep\_result** (void) const

### Private Attributes

- **parse\_tree::iterator** itree  
*corresponding node of the parse tree in the same sentence.*

- string `dep_source`

*information about categories which support the building of dependencies*

- string `dep_target`
- string `dep_result`

### 8.36.1 Detailed Description

class `denode` stores nodes of a dependency tree and parse tree <-> `deptree` relations

### 8.36.2 Constructor & Destructor Documentation

#### 8.36.2.1 `depnode::depnode ()`

Methods for dependency tree nodes.

#### 8.36.2.2 `depnode::depnode (const string &)`

#### 8.36.2.3 `depnode::depnode (const node &)`

### 8.36.3 Member Function Documentation

#### 8.36.3.1 `string depnode::get_dep_result (void) const`

#### 8.36.3.2 `string depnode::get_dep_source (void) const`

#### 8.36.3.3 `string depnode::get_dep_target (void) const`

#### 8.36.3.4 `parse_tree::iterator depnode::get_link (void)`

#### 8.36.3.5 `void depnode::set_dep_result (const string &)`

#### 8.36.3.6 `void depnode::set_dep_source (const string &)`

#### 8.36.3.7 `void depnode::set_dep_target (const string &)`

#### 8.36.3.8 `void depnode::set_link (const parse_tree::iterator)`

### 8.36.4 Member Data Documentation

#### 8.36.4.1 `string depnode::dep_result [private]`

#### 8.36.4.2 `string depnode::dep_source [private]`

*information about categories which support the building of dependencies*

**8.36.4.3** `string depnode::dep_target` [private]

**8.36.4.4** `parse_tree::iterator depnode::itree` [private]

corresponding node of the parse tree in the same sentence.

The documentation for this class was generated from the following files:

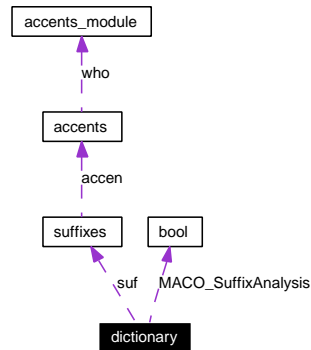
- `language.h`
- `language.cc`

## 8.37 dictionary Class Reference

The class dictionary implements dictionary search and suffix analysis for word forms.

```
#include <dictionary.h>
```

Collaboration diagram for dictionary:



### Public Member Functions

- **dictionary** (const **maco\_options** &)  
*Constructor.*
- **~dictionary** ()  
*Destructor.*
- void **search\_form** (const string &, list< **analysis** > &)  
*Get the analysis list from a given form (default options).*
- void **annotate** (**sentence** &)  
*Search words in sentence using default options.*

### Private Member Functions

- void **annotate\_word** (**word** &)  
*Fills the analysis list of a word and checks for suffixes.*
- bool **check\_contracted** (const **word** &, list< **word** > &)  
*check whether the word is a contraction, and if so, fill the list with the contracted words*

### Private Attributes

- bool **MACO\_SuffixAnalysis**  
*configuration options*
- **suffixes** **suf**



*suffix analyzer*

- **Db morfodb**

*C++ Interface to BerkeleyDB C API.*

## 8.37.1 Detailed Description

The class dictionary implements dictionary search and suffix analysis for word forms.

## 8.37.2 Constructor & Destructor Documentation

### 8.37.2.1 dictionary::dictionary (const maco\_options &)

Constructor.

### 8.37.2.2 dictionary::~dictionary ()

Destructor.

## 8.37.3 Member Function Documentation

### 8.37.3.1 void dictionary::annotate (sentence &)

Search words in sentence using default options.

### 8.37.3.2 void dictionary::annotate\_word (word &) [private]

Fills the analysis list of a word and checks for suffixes.

### 8.37.3.3 bool dictionary::check\_contracted (const word &, list< word > &) [private]

check whether the word is a contraction, and if so, fill the list with the contracted words

### 8.37.3.4 void dictionary::search\_form (const string &, list< analysis > &)

Get the analysis list from a given form (default options).

## 8.37.4 Member Data Documentation

### 8.37.4.1 bool dictionary::MACO\_SuffixAnalysis [private]

configuration options

### 8.37.4.2 Db dictionary::morfodb [private]

C++ Interface to BerkeleyDB C API.

#### 8.37.4.3 suffixes dictionary::suf [private]

suffix analyzer

The documentation for this class was generated from the following files:

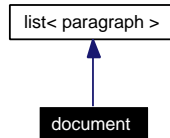
- **dictionary.h**
- **dictionary.cc**

## 8.38 document Class Reference

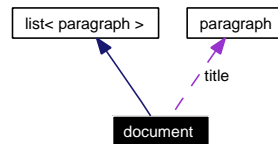
Class document is a list of paragraphs.

```
#include <language.h>
```

Inheritance diagram for document:



Collaboration diagram for document:



### Private Attributes

- paragraph title

### 8.38.1 Detailed Description

Class document is a list of paragraphs.

It may have additional information (such as title)

### 8.38.2 Member Data Documentation

#### 8.38.2.1 paragraph document::title [private]

The documentation for this class was generated from the following file:

- language.h

## 8.39 edge Class Reference

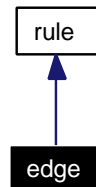
Class edge stores all information in a chart edge.

```
#include <chart.h>
```

Inheritance diagram for edge:



Collaboration diagram for edge:



### Public Member Functions

- **edge** (const string &, const **list**< string > &, const int posgov)  
*Constructors of the subclass edge.*
- **edge** ()
- const **list**< string > **get\_matched** () const  
*get matched part of the rule.*
- const **list**< pair< int, int > > **get\_backpath** () const  
*get list of cells used to satisfy edge rule*
- **bool active** () const  
*Check if the edge is complete (inactive) or not.*
- void **shift** (int, int)  
*Advance the edge one position.*

### Private Attributes

- **list**< string > **matched**  
*Part of the rule already matched.*
- **list**< pair< int, int > > **backpath**

### 8.39.1 Detailed Description

Class edge stores all information in a chart edge.

### 8.39.2 Constructor & Destructor Documentation

#### 8.39.2.1 `edge::edge (const string &, const list< string > &, const int posgov)`

Constructors of the subclass edge.

#### 8.39.2.2 `edge::edge ()`

### 8.39.3 Member Function Documentation

#### 8.39.3.1 `bool edge::active () const`

Check if the edge is complete (inactive) or not.

#### 8.39.3.2 `const list<pair<int,int> > edge::get_backpath () const`

get list of cells used to satisfy edge rule

#### 8.39.3.3 `const list<string> edge::get_matched () const`

get matched part of the rule.

#### 8.39.3.4 `void edge::shift (int, int)`

Advance the edge one position.

### 8.39.4 Member Data Documentation

#### 8.39.4.1 `list<pair<int,int> > edge::backpath [private]`

#### 8.39.4.2 `list<string> edge::matched [private]`

Part of the rule already matched.

The documentation for this class was generated from the following file:

- `chart.h`

## 8.40 emission\_states Class Reference

The class `emission_states` stores the list of states in the HMM that *may* be generating a given word given the two previous words (and their valid tags).

```
#include <hmm_tagger.h>
```

### 8.40.1 Detailed Description

The class `emission_states` stores the list of states in the HMM that *may* be generating a given word given the two previous words (and their valid tags).

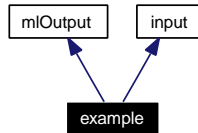
The documentation for this class was generated from the following file:

- `hmm_tagger.h`

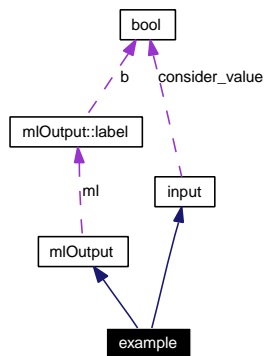
## 8.41 example Class Reference

```
#include <example.h>
```

Inheritance diagram for example:



Collaboration diagram for example:



### Public Types

- `typedef input::const_iterator feature_iterator`

### Public Member Functions

- `example (string ml, string f)`
- `example ()`
- `void print (ostream &o) const`

### 8.41.1 Member Typedef Documentation

8.41.1.1 `typedef input::const_iterator example::feature_iterator`

### 8.41.2 Constructor & Destructor Documentation

8.41.2.1 `example::example (string ml, string f) [inline]`

8.41.2.2 `example::example () [inline]`

### 8.41.3 Member Function Documentation

8.41.3.1 `void example::print (ostream & o) const`

Reimplemented from `map_input` (p. 137).

The documentation for this class was generated from the following file:

- `example.h`



## 8.42 fex Class Reference

```
#include <fex.h>
```

### Public Member Functions

- **fex** (const string &)  
*Constructor.*
- void **encode** (const **sentence** &, **vector**< set< string > > &)  
*encode given sentence in features*
- **vector**< set< string > > **encode** (const **sentence** &)  
*encode given sentence in features, return result as vector*

### Private Attributes

- **vector**< **RGF** > **rules**

#### 8.42.1 Constructor & Destructor Documentation

##### 8.42.1.1 fex::fex (const string &)

Constructor.

#### 8.42.2 Member Function Documentation

##### 8.42.2.1 vector<set<string> > fex::encode (const sentence &)

encode given sentence in features, return result as vector

##### 8.42.2.2 void fex::encode (const sentence &, vector< set< string > > &)

encode given sentence in features

#### 8.42.3 Member Data Documentation

##### 8.42.3.1 vector<RGF> fex::rules [private]

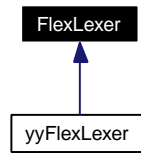
The documentation for this class was generated from the following file:

- **fex.h**

## 8.43 FlexLexer Class Reference

```
#include <FlexLexer.h>
```

Inheritance diagram for FlexLexer:



### Public Member Functions

- virtual `~FlexLexer ()`
- `const char * YYText ()`
- `int YYLeng ()`
- virtual void `yy_switch_to_buffer` (struct yy\_buffer\_state \*new\_buffer)=0
- virtual struct yy\_buffer\_state \* `yy_create_buffer` (istream \*s, int size)=0
- virtual void `yy_delete_buffer` (struct yy\_buffer\_state \*b)=0
- virtual void `yyrestart` (istream \*s)=0
- virtual int `yylex ()`=0
- int `yylex` (istream \*new\_in, ostream \*new\_out=0)
- virtual void `switch_streams` (istream \*new\_in=0, ostream \*new\_out=0)=0
- int `lineno ()` const
- int `debug ()` const
- void `set_debug` (int flag)

### Protected Attributes

- `char * yytext`
- `int yyleng`
- `int yylineno`
- `int yy_flex_debug`

#### 8.43.1 Constructor & Destructor Documentation

8.43.1.1 `virtual FlexLexer::~FlexLexer ()` [inline, virtual]

#### 8.43.2 Member Function Documentation

8.43.2.1 `int FlexLexer::debug ()` const [inline]

8.43.2.2 `int FlexLexer::lineno ()` const [inline]

8.43.2.3 `void FlexLexer::set_debug` (int *flag*) [inline]

8.43.2.4 `virtual void FlexLexer::switch_streams` (istream \* *new\_in* = 0, ostream \* *new\_out* = 0) [pure virtual]

Implemented in `yyFlexLexer` (p. 276).

**8.43.2.5** virtual struct yy\_buffer\_state\* FlexLexer::yy\_create\_buffer (istream \* *s*, int *size*) [pure virtual]

Implemented in yyFlexLexer (p. 277).

**8.43.2.6** virtual void FlexLexer::yy\_delete\_buffer (struct yy\_buffer\_state \* *b*) [pure virtual]

Implemented in yyFlexLexer (p. 277).

**8.43.2.7** virtual void FlexLexer::yy\_switch\_to\_buffer (struct yy\_buffer\_state \* *new\_buffer*) [pure virtual]

Implemented in yyFlexLexer (p. 277).

**8.43.2.8** int FlexLexer::YYLeng () [inline]

**8.43.2.9** int FlexLexer::yylex (istream \* *new\_in*, ostream \* *new\_out* = 0) [inline]

**8.43.2.10** virtual int FlexLexer::yylex () [pure virtual]

Implemented in yyFlexLexer (p. 277).

**8.43.2.11** virtual void FlexLexer::yyrestart (istream \* *s*) [pure virtual]

Implemented in yyFlexLexer (p. 277).

**8.43.2.12** const char\* FlexLexer::YYText () [inline]

### 8.43.3 Member Data Documentation

**8.43.3.1** int FlexLexer::yy\_flex\_debug [protected]

**8.43.3.2** int FlexLexer::yyleng [protected]

**8.43.3.3** int FlexLexer::yylineno [protected]

**8.43.3.4** char\* FlexLexer::yytext [protected]

The documentation for this class was generated from the following file:

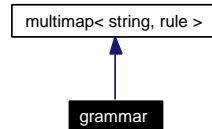
- FlexLexer.h

## 8.44 grammar Class Reference

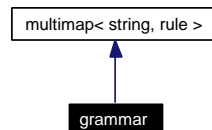
Class grammar implements a CFG, ready to be used from a chart parser.

```
#include <grammar.h>
```

Inheritance diagram for grammar:



Collaboration diagram for grammar:



### Public Member Functions

- **grammar** (const string &)  
*Create a grammar loading it from a file.*
- int **get\_\_specificity** (const string &) const
- int **get\_\_priority** (const string &) const
- string **get\_\_start\_\_symbol** () const  
*obtain the start symbol of the grammar*
- bool **is\_\_hidden** (const string &) const  
*Check whether a symbol must disappear of final tree.*
- bool **is\_\_flat** (const string &) const  
*Check whether a symbol must be flattened when recursive.*
- bool **is\_\_notop** (const string &) const  
*Check whether a symbol can not be used as a tree root.*
- bool **is\_\_onlytop** (const string &) const  
*Check whether a symbol is hidden unless when at tree root.*
- bool **is\_\_terminal** (const string &) const  
*Check whether a symbol is terminal or not.*
- list< rule > **get\_\_rules\_\_right** (const string &) const  
*Get all rules with a right part beggining with the given category.*
- list< rule > **get\_\_rules\_\_right\_\_wildcard** (const string &) const

*Get all rules with a right part beginning with a wildcarded category.*

- **bool in\_filemap** (const string &, const string &) const  
*search given string in filemap, and check whether it maps to the second*

## Static Public Attributes

- static unsigned int **NOGOV**
- static unsigned int **DEFGOV**

## Private Member Functions

- void **new\_rule** (const string &, const list< string > &, bool, const int rgov)  
*Create and store a new rule, indexed by 1st category in its right part.*

## Private Attributes

- set< string > **nonterminal**  
*Non-terminal symbols in the grammar.*
- multimap< string, rule > **wild**  
*rules starting with a wildcarded token, indexed by first char in category.*
- multimap< string, string > **filemap**  
*map to store files appearing in grammar rules*
- map< string, int > **prior**  
*symbol priorities to build the tree*
- set< string > **hidden**  
*Non-terminal symbols that must not be seen in the tree.*
- set< string > **flat**  
*Non-terminal symbols that must be flattened in final tree when recursive.*
- set< string > **notop**  
*Non-terminal symbols that must not be considered tree roots.*
- set< string > **onlytop**  
*Non-terminal symbols that are visible only when are at tree root.*
- string **start**  
*start symbol*

### 8.44.1 Detailed Description

Class grammar implements a CFG, ready to be used from a chart parser.

## 8.44.2 Constructor & Destructor Documentation

### 8.44.2.1 `grammar::grammar (const string &)`

Create a grammar loading it from a file.

## 8.44.3 Member Function Documentation

### 8.44.3.1 `int grammar::get_priority (const string &) const`

### 8.44.3.2 `list<rule> grammar::get_rules_right (const string &) const`

Get all rules with a right part beginning with the given category.

### 8.44.3.3 `list<rule> grammar::get_rules_right_wildcard (const string &) const`

Get all rules with a right part beginning with a wildcarded category.

### 8.44.3.4 `int grammar::get_specificity (const string &) const`

### 8.44.3.5 `string grammar::get_start_symbol () const`

obtain the start symbol of the grammar

### 8.44.3.6 `bool grammar::in_filemap (const string &, const string &) const`

search given string in filemap, and check whether it maps to the second

### 8.44.3.7 `bool grammar::is_flat (const string &) const`

Check whether a symbol must be flattened when recursive.

### 8.44.3.8 `bool grammar::is_hidden (const string &) const`

Check whether a symbol must disappear of final tree.

### 8.44.3.9 `bool grammar::is_notop (const string &) const`

Check whether a symbol can not be used as a tree root.

### 8.44.3.10 `bool grammar::is_onlytop (const string &) const`

Check whether a symbol is hidden unless when at tree root.

### 8.44.3.11 `bool grammar::is_terminal (const string &) const`

Check whether a symbol is terminal or not.

**8.44.3.12** `void grammar::new_rule (const string &, const list< string > &, bool, const int rgov)` [private]

Create and store a new rule, indexed by 1st category in its right part.

## 8.44.4 Member Data Documentation

**8.44.4.1** `unsigned int grammar::DEFGOV` [static]

**8.44.4.2** `multimap<string,string> grammar::filemap` [private]

map to store files appearing in grammar rules

**8.44.4.3** `set<string> grammar::flat` [private]

Non-terminal symbols that must be flattened in final tree when recursive.

**8.44.4.4** `set<string> grammar::hidden` [private]

Non-terminal symbols that must not be seen in the tree.

**8.44.4.5** `unsigned int grammar::NOGOV` [static]

**8.44.4.6** `set<string> grammar::nonterminal` [private]

Non-terminal symbols in the grammar.

**8.44.4.7** `set<string> grammar::notop` [private]

Non-terminal symbols that must not be considered tree roots.

**8.44.4.8** `set<string> grammar::onlytop` [private]

Non-terminal symbols that are visible only when are at tree root.

**8.44.4.9** `map<string,int> grammar::prior` [private]

symbol priorities to build the tree

**8.44.4.10** `string grammar::start` [private]

start symbol

**8.44.4.11** `multimap<string,rule> grammar::wild` [private]

rules starting with a wildcarded token, indexed by first char in category.

The documentation for this class was generated from the following file:

- `grammar.h`

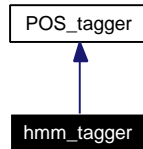


## 8.45 hmm\_tagger Class Reference

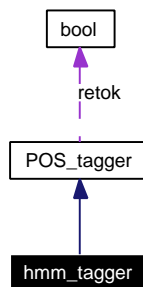
The class `hmm_tagger` implements the syntactic analyzer and is the main class, which uses all the others.

```
#include <hmm_tagger.h>
```

Inheritance diagram for `hmm_tagger`:



Collaboration diagram for `hmm_tagger`:



### Public Member Functions

- **hmm\_tagger** (const string &, const string &, bool)  
*Constructor.*
- void **analyze** (list< sentence > &)  
*analyze given sentences*
- list< sentence > **analyze** (const list< sentence > &)  
*analyze sentences and return analyzed copy*

### Private Member Functions

- double **ProbA\_log** (const string &, const string &) const  
*Compute transition log\_probability from state\_i to state\_j, returning appropriate smoothed values if no evidence is available.*
- double **ProbB\_log** (const string &, const word &) const  
*Compute emission log\_probability for observation obs from state\_i.*
- double **ProbPi\_log** (const string &) const  
*Compute initial log\_probability for state\_i.*

- **list< emission\_states > FindStates (const sentence &) const**  
*compute possible emission states for each word in sentence.*

## Private Attributes

- string **Language**
- map< string, double > **PTag**  
*maps to store the probabilities*
- map< string, double > **PBg**
- map< string, double > **PTrg**
- map< string, double > **PInitial**
- map< string, double > **PWord**
- double **c** [3]  
*coefficients to compute linear interpolation*

### 8.45.1 Detailed Description

The class `hmm_tagger` implements the syntactic analyzer and is the main class, which uses all the others.

### 8.45.2 Constructor & Destructor Documentation

#### 8.45.2.1 `hmm_tagger::hmm_tagger (const string &, const string &, bool)`

Constructor.

### 8.45.3 Member Function Documentation

#### 8.45.3.1 `list< sentence > hmm_tagger::analyze (const list< sentence > &)`

analyze sentences and return analyzed copy

#### 8.45.3.2 `void hmm_tagger::analyze (list< sentence > &) [virtual]`

analyze given sentences

Implements **POS\_tagger** (p.178).

#### 8.45.3.3 `list< emission_states > hmm_tagger::FindStates (const sentence &)` `const [private]`

compute possible emission states for each word in sentence.

#### 8.45.3.4 double hmm\_tagger::ProbA\_log (const string &, const string &) const [private]

Compute transition log\_probability from state\_i to state\_j, returning appropriate smoothed values if no evidence is available.

#### 8.45.3.5 double hmm\_tagger::ProbB\_log (const string & *state\_i*, const word & *obs*) const [private]

Compute emission log\_probability for observation obs from state\_i.

$P_b = P(\text{state}|\text{word}) * P(\text{word}) / P(\text{state})$  approximating  $P(s|w)$  by:  $P(s|w) \sim P(t3|w) * P(s) / P(t3)$   
thus,  $P_b \sim P(t3|w) * P(w) / P(t3)$

#### 8.45.3.6 double hmm\_tagger::ProbPi\_log (const string &) const [private]

Compute initial log\_probability for state\_i.

### 8.45.4 Member Data Documentation

#### 8.45.4.1 double hmm\_tagger::c[3] [private]

coefficients to compute linear interpolation

#### 8.45.4.2 string hmm\_tagger::Language [private]

#### 8.45.4.3 map<string, double> hmm\_tagger::PBg [private]

#### 8.45.4.4 map<string, double> hmm\_tagger::PInitial [private]

#### 8.45.4.5 map<string, double> hmm\_tagger::PTag [private]

maps to store the probabilities

#### 8.45.4.6 map<string, double> hmm\_tagger::PTrg [private]

#### 8.45.4.7 map<string, double> hmm\_tagger::PWord [private]

The documentation for this class was generated from the following files:

- `hmm_tagger.h`
- `hmm_tagger.cc`

## 8.46 iFeature Class Reference

```
#include <example.h>
```

### Public Member Functions

- **iFeature** (int *l*, double *v*=1.0)
- int **label** () const
- double **value** () const
- void **set\_value** (double *v*)
- bool **operator**< (const **iFeature** &*rhs*) const

### Private Attributes

- int **\_label**
- double **\_value**

#### 8.46.1 Constructor & Destructor Documentation

8.46.1.1 **iFeature::iFeature** (int *l*, double *v* = 1.0) [inline]

#### 8.46.2 Member Function Documentation

8.46.2.1 int **iFeature::label** () const [inline]

8.46.2.2 bool **iFeature::operator**< (const **iFeature** & *rhs*) const [inline]

8.46.2.3 void **iFeature::set\_value** (double *v*) [inline]

8.46.2.4 double **iFeature::value** () const [inline]

#### 8.46.3 Member Data Documentation

8.46.3.1 int **iFeature::\_label** [private]

8.46.3.2 double **iFeature::\_value** [private]

The documentation for this class was generated from the following file:

- **example.h**

## 8.47 label Class Reference

The class label stores all information related to a variable label in the relaxation labelling algorithm.

```
#include <relax.h>
```

### Public Member Functions

- **label ()**  
*Constructor.*

### Protected Attributes

- double **weight** [2]  
*label weigth at current and next iterations*
- **list< constraint > constraints**  
*list of constraints for the label*

### Friends

- class **relax**

#### 8.47.1 Detailed Description

The class label stores all information related to a variable label in the relaxation labelling algorithm.

#### 8.47.2 Constructor & Destructor Documentation

##### 8.47.2.1 label::label ()

Constructor.

#### 8.47.3 Friends And Related Function Documentation

##### 8.47.3.1 friend class relax [friend]

#### 8.47.4 Member Data Documentation

##### 8.47.4.1 list<constraint> label::constraints [protected]

list of constraints for the label

#### 8.47.4.2 `double label::weight[2]` [protected]

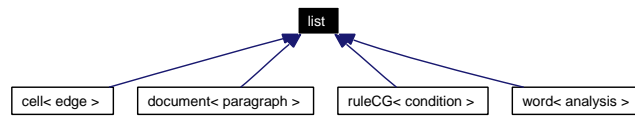
label weigth at current and next iterations

The documentation for this class was generated from the following file:

- `relax.h`

## 8.48 list Class Reference

Inheritance diagram for list:



The documentation for this class was generated from the following file:

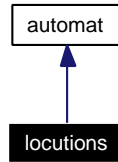
- **language.h**

## 8.49 locutions Class Reference

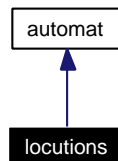
Class locutions recognizes multiwords belonging to a list obtained from a configuration file.

```
#include <locutions.h>
```

Inheritance diagram for locutions:



Collaboration diagram for locutions:



### Public Member Functions

- **locutions** (const **maco\_\_options** &)  
*Constructor.*

### Private Member Functions

- void **check** (const string, list< string > &, bool &, bool &)
- int **ComputeToken** (int, sentence::const\_iterator, const **sentence** &)  
*Compute the right token code for word j from given state.*
- void **ResetActions** ()  
*Reset current multiword acumulator.*
- void **StateActions** (int, int, int, sentence::const\_iterator)  
*Perform necessary actions in "state" reached from state "origin" via word j interpreted as code "token": Basically, when reaching a state, update accumulated multiword.*
- void **SetMultiwordAnalysis** (sentence::iterator) const  
*Set the appropriate lemma and parole for the new multiword.*

### Private Attributes

- map< string, string > **locut**



*store multiword list*

- `set< string > prefixes`

*store multiword prefixes*

- `list< string > acc_mw`

*partially build multiword.*

- `list< string > longest_mw`

- `vector< word > components`

*store mw components in case we need to recover them*

### 8.49.1 Detailed Description

Class locutions recognizes multiwords belonging to a list obtained from a configuration file.

### 8.49.2 Constructor & Destructor Documentation

#### 8.49.2.1 `locutions::locutions (const maco_options &)`

Constructor.

### 8.49.3 Member Function Documentation

#### 8.49.3.1 `void locutions::check (const string, list< string > &, bool &, bool &)` [private]

#### 8.49.3.2 `int locutions::ComputeToken (int, sentence::const_iterator, const sentence &)` [private, virtual]

Compute the right token code for word j from given state.

Implements **automat** (p. 41).

#### 8.49.3.3 `void locutions::ResetActions ()` [private, virtual]

Reset current multiword accumulator.

Implements **automat** (p. 41).

#### 8.49.3.4 `void locutions::SetMultiwordAnalysis (sentence::iterator) const` [private, virtual]

Set the appropriate lemma and parole for the new multiword.

Implements **automat** (p. 41).

### 8.49.3.5 void locutions::StateActions (int, int, int, sentence::const\_iterator) [private, virtual]

Perform necessary actions in "state" reached from state "origin" via word j interpreted as code "token": Basically, when reaching a state, update accumulated multiword.

Implements **automat** (p. 42).

## 8.49.4 Member Data Documentation

### 8.49.4.1 list<string> locutions::acc\_mw [private]

partially build multiword.

### 8.49.4.2 vector<word> locutions::components [private]

store mw components in case we need to recover them

### 8.49.4.3 map<string,string> locutions::locut [private]

store multiword list

### 8.49.4.4 list<string> locutions::longest\_mw [private]

### 8.49.4.5 set<string> locutions::prefixes [private]

store multiword prefixes

The documentation for this class was generated from the following files:

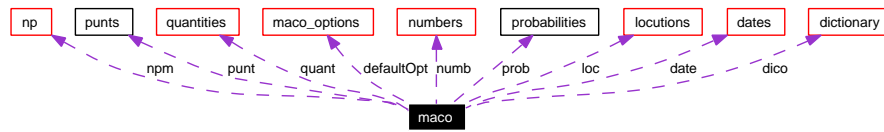
- **locutions.h**
- **locutions.cc**

## 8.50 maco Class Reference

Class maco implements the morphological analyzer, which uses all the specific analyzers: dates, numbers, dictionary, etc.

```
#include <maco.h>
```

Collaboration diagram for maco:



### Public Member Functions

- **maco** (const **maco\_options** &)  
*Constructor.*
- void **analyze** (list< sentence > &)  
*analyze given sentences*
- list< sentence > **analyze** (const list< sentence > &)  
*analyze sentences, return analyzed copy*

### Private Attributes

- **maco\_options** **defaultOpt**  
*creation options*
- **locutions** **loc**
- **dictionary** **dico**
- **numbers** **numb**
- **dates** **date**
- **quantities** **quant**
- **punts** **punt**
- **probabilities** **prob**
- **np** **npm**

#### 8.50.1 Detailed Description

Class maco implements the morphological analyzer, which uses all the specific analyzers: dates, numbers, dictionary, etc.

#### 8.50.2 Constructor & Destructor Documentation

##### 8.50.2.1 maco::maco (const maco\_options &)

Constructor.

### 8.50.3 Member Function Documentation

#### 8.50.3.1 `list< sentence > maco::analyze (const list< sentence > &)`

analyze sentences, return analyzed copy

#### 8.50.3.2 `void maco::analyze (list< sentence > &)`

analyze given sentences

### 8.50.4 Member Data Documentation

#### 8.50.4.1 `dates maco::date [private]`

#### 8.50.4.2 `maco_options maco::defaultOpt [private]`

creation options

#### 8.50.4.3 `dictionary maco::dico [private]`

#### 8.50.4.4 `locutions maco::loc [private]`

#### 8.50.4.5 `np maco::npm [private]`

#### 8.50.4.6 `numbers maco::numb [private]`

#### 8.50.4.7 `probabilities maco::prob [private]`

#### 8.50.4.8 `punts maco::punt [private]`

#### 8.50.4.9 `quantities maco::quant [private]`

The documentation for this class was generated from the following files:

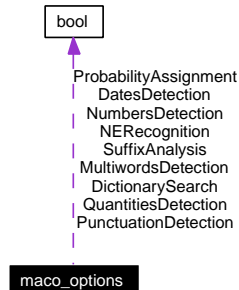
- `maco.h`
- `maco.cc`

## 8.51 maco\_options Class Reference

Class maco\_options implements a set of specific options of the morphological analyzer.

```
#include <maco_options.h>
```

Collaboration diagram for maco\_options:



### Public Member Functions

- **maco\_options** (const string &)  
*constructor*
- void **set\_active\_modules** (bool, bool, bool, bool, bool, bool, bool, bool, bool)  
*Option setting methods provided to ease perl interface generation.*
- void **set\_numerical\_points** (const string &, const string &)
- void **set\_data\_files** (const string &, const string &, const string &, const string &, const string &, const string &, const string &)
- void **set\_threshold** (double)

### Public Attributes

- string **Lang**
- bool **SuffixAnalysis**  
*Morphological analyzer options.*
- bool **MultiwordsDetection**
- bool **NumbersDetection**
- bool **PunctuationDetection**
- bool **DatesDetection**
- bool **QuantitiesDetection**
- bool **DictionarySearch**
- bool **ProbabilityAssignment**
- bool **NERecognition**
- string **Decimal**  
*Morphological analyzer options.*
- string **Thousand**
- string **LocutionsFile**

*Morphological analyzer options.*

- string **CurrencyFile**
- string **SuffixFile**
- string **ProbabilityFile**
- string **DictionaryFile**
- string **NPdataFile**
- string **PunctuationFile**
- double **ProbabilityThreshold**

### 8.51.1 Detailed Description

Class `maco_options` implements a set of specific options of the morphological analyzer.

Other modules do not have such a class because they deal with a reduced number of options

### 8.51.2 Constructor & Destructor Documentation

#### 8.51.2.1 `maco_options::maco_options (const string & lg)`

constructor

Initialize with default values.

### 8.51.3 Member Function Documentation

#### 8.51.3.1 `void maco_options::set_active_modules (bool, bool, bool, bool, bool, bool, bool, bool)`

Option setting methods provided to ease perl interface generation.

Since option data members are public and can be accessed directly from C++, the following methods are not necessary, but may become convenient sometimes.

#### 8.51.3.2 `void maco_options::set_data_files (const string &, const string &, const string &, const string &, const string &, const string &)`

#### 8.51.3.3 `void maco_options::set_numerical_points (const string &, const string &)`

#### 8.51.3.4 `void maco_options::set_threshold (double)`

### 8.51.4 Member Data Documentation

#### 8.51.4.1 `string maco_options::CurrencyFile`

#### 8.51.4.2 `bool maco_options::DatesDetection`

#### 8.51.4.3 `string maco_options::Decimal`

Morphological analyzer options.

8.51.4.4 string maco\_options::DictionaryFile

8.51.4.5 bool maco\_options::DictionarySearch

8.51.4.6 string maco\_options::Lang

8.51.4.7 string maco\_options::LocutionsFile

Morphological analyzer options.

8.51.4.8 bool maco\_options::MultiwordsDetection

8.51.4.9 bool maco\_options::NERecognition

8.51.4.10 string maco\_options::NPdataFile

8.51.4.11 bool maco\_options::NumbersDetection

8.51.4.12 bool maco\_options::ProbabilityAssignment

8.51.4.13 string maco\_options::ProbabilityFile

8.51.4.14 double maco\_options::ProbabilityThreshold

8.51.4.15 bool maco\_options::PunctuationDetection

8.51.4.16 string maco\_options::PunctuationFile

8.51.4.17 bool maco\_options::QuantitiesDetection

8.51.4.18 bool maco\_options::SuffixAnalysis

Morphological analyzer options.

8.51.4.19 string maco\_options::SuffixFile

8.51.4.20 string maco\_options::Thousand

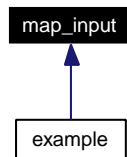
The documentation for this class was generated from the following files:

- maco\_options.h
- maco\_options.cc

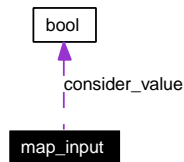
## 8.52 map\_input Class Reference

```
#include <example.h>
```

Inheritance diagram for map\_input:



Collaboration diagram for map\_input:



### Public Member Functions

- **map\_input** ()
- **map\_input** (string)
- **map\_input** (double f1, const **map\_input** &i1, double f2, const **map\_input** &i2)
- **~map\_input** ()
- void **add\_feature** (const **iFeature** &f)
- void **add\_feature** (int l, double v=1.0)
- void **parse\_string** (string)
- double **feature\_value** (int **label**) const
- int **size** () const
- int **dimension** () const
- double **inner\_product** (**map\_input** \*) const
- **const\_iterator** **begin** () const
- **const\_iterator** **end** () const
- void **print** (ostream &o) const
- void **write** (ostream &o) const

### Static Public Member Functions

- static void **set\_consider\_value** (bool)
- static void **set\_label\_value\_separator** (char)

### Private Attributes

- map< int, **iFeature** > **mf**
- int **\_dimension**



## Static Private Attributes

- static bool consider\_value
- static char label\_value\_separator

## Classes

- class const\_iterator

### 8.52.1 Constructor & Destructor Documentation

8.52.1.1 map\_input::map\_input () [inline]

8.52.1.2 map\_input::map\_input (string)

8.52.1.3 map\_input::map\_input (double *f1*, const map\_input & *i1*, double *f2*, const map\_input & *i2*)

8.52.1.4 map\_input::~~map\_input ()

### 8.52.2 Member Function Documentation

8.52.2.1 void map\_input::add\_feature (int *l*, double *v* = 1.0)

8.52.2.2 void map\_input::add\_feature (const iFeature & *f*)

8.52.2.3 const\_iterator map\_input::begin () const [inline]

8.52.2.4 int map\_input::dimension () const [inline]

8.52.2.5 const\_iterator map\_input::end () const [inline]

8.52.2.6 double map\_input::feature\_value (int *label*) const

8.52.2.7 double map\_input::inner\_product (map\_input \*) const

8.52.2.8 void map\_input::parse\_string (string)

8.52.2.9 void map\_input::print (ostream & *o*) const

Reimplemented in [example](#) (p. 112).

8.52.2.10 static void map\_input::set\_consider\_value (bool) [static]

8.52.2.11 static void map\_input::set\_label\_value\_separator (char) [static]

8.52.2.12 int map\_input::size () const [inline]

8.52.2.13 void map\_input::write (ostream & o) const

### 8.52.3 Member Data Documentation

8.52.3.1 int map\_input::\_dimension [private]

8.52.3.2 bool map\_input::consider\_value [static, private]

8.52.3.3 char map\_input::label\_value\_separator [static, private]

8.52.3.4 map<int,iFeature> map\_input::mf [private]

The documentation for this class was generated from the following file:

- **example.h**

## 8.53 map\_input::const\_iterator Class Reference

```
#include <example.h>
```

Collaboration diagram for map\_input::const\_iterator:



### Public Member Functions

- `const_iterator ()`
- `const_iterator (map< int, iFeature >::const_iterator i0)`
- `const iFeature * operator → () const`
- `const_iterator & operator++ ()`
- `bool operator== (const const_iterator &rhs)`
- `bool operator!= (const const_iterator &rhs)`

### Private Attributes

- `map< int, iFeature >::const_iterator i`

### 8.53.1 Constructor & Destructor Documentation

8.53.1.1 `map_input::const_iterator::const_iterator () [inline]`

8.53.1.2 `map_input::const_iterator::const_iterator (map< int, iFeature >::const_iterator i0) [inline]`

### 8.53.2 Member Function Documentation

8.53.2.1 `bool map_input::const_iterator::operator!= (const const_iterator & rhs) [inline]`

8.53.2.2 `const_iterator& map_input::const_iterator::operator++ () [inline]`

8.53.2.3 `const iFeature* map_input::const_iterator::operator → () const [inline]`

8.53.2.4 `bool map_input::const_iterator::operator== (const const_iterator & rhs) [inline]`

### 8.53.3 Member Data Documentation

8.53.3.1 `map<int,iFeature>::const_iterator map_input::const_iterator::i [private]`

The documentation for this class was generated from the following file:

- `example.h`

## 8.54 mlABTree Class Reference

```
#include <adaboost.h>
```

Collaboration diagram for mlABTree:



### Public Member Functions

- **mlABTree** (double \*p0)
- **mlABTree** (int f, **mlABTree** \*wrFalse, **mlABTree** \*wrTrue)
- **~mlABTree** ()
- void **classify** (input \*i, double \*pred)
- void **print** (char \*carry)
- void **write\_to\_stream** (ofstream &os)

### Static Public Member Functions

- static void **set\_nlabels** (int nl)
- static void **set\_verbose** (int level)
- static void **set\_epsilon** (double eps)
- static **mlABTree** \* **read\_from\_stream** (istream &is)
- static **mlABTree** \* **learn** (dataset \*ds, double \*Z, int max\_depth0)

### Private Member Functions

- **mlABTree** (const **mlABTree** &wr0)

### Static Private Member Functions

- static **mlABTree** \* **learn\_0** (dataset \*ds, double \*Z, int depth)
- static int **stopping\_criterion** (dataset \*ds, int depth)
- static int **best\_feature** (dataset \*ds, double \*W)
- static double **Zcalculus** (double \*W, int ndim)
- static void **Cprediction** (int v, double \*W, double result[])

### Private Attributes

- int **feature**
- **mlABTree** \*\* **sons**
- double \* **predictions**

## Static Private Attributes

- static int **nlabels**
- static double **epsilon**
- static int **max\_\_depth**
- static int \* **used\_\_features**
- static int **verbose**

### 8.54.1 Constructor & Destructor Documentation

- 8.54.1.1 `mlABTree::mlABTree (const mlABTree & wr0)` [private]
- 8.54.1.2 `mlABTree::mlABTree (double * p0)`
- 8.54.1.3 `mlABTree::mlABTree (int f, mlABTree * wrFalse, mlABTree * wrTrue)`
- 8.54.1.4 `mlABTree::~~mlABTree ()`

### 8.54.2 Member Function Documentation

- 8.54.2.1 `static int mlABTree::best_feature (dataset * ds, double * W)` [static, private]
- 8.54.2.2 `void mlABTree::classify (input * i, double * pred)`
- 8.54.2.3 `static void mlABTree::Cprediction (int v, double * W, double result[])` [static, private]
- 8.54.2.4 `static mlABTree* mlABTree::learn (dataset * ds, double * Z, int max_depth0)` [static]
- 8.54.2.5 `static mlABTree* mlABTree::learn_0 (dataset * ds, double * Z, int depth)` [static, private]
- 8.54.2.6 `void mlABTree::print (char * carry)`
- 8.54.2.7 `static mlABTree* mlABTree::read_from_stream (istream & is)` [static]
- 8.54.2.8 `static void mlABTree::set_epsilon (double eps)` [static]
- 8.54.2.9 `static void mlABTree::set_nlabels (int nl)` [static]
- 8.54.2.10 `static void mlABTree::set_verbose (int level)` [static]
- 8.54.2.11 `static int mlABTree::stopping_criterion (dataset * ds, int depth)` [static, private]
- 8.54.2.12 `void mlABTree::write_to_stream (ofstream & os)`
- 8.54.2.13 `static double mlABTree::Zcalculus (double * W, int ndim)` [static, private]

### 8.54.3 Member Data Documentation

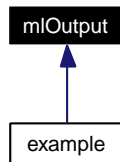
- 8.54.3.1 `double mlABTree::epsilon` [static, private]
- 8.54.3.2 `int mlABTree::feature` [private]
- 8.54.3.3 `int mlABTree::max_depth` [static, private]
- 8.54.3.4 `int mlABTree::nlabels` [static, private]
- ~~8.54.3.5 `double* mlABTree::predictions` [private]~~
- 8.54.3.6 `mlABTree** mlABTree::sons` [private]
- 8.54.3.7 `int* mlABTree::used_features` [static, private]
- 8.54.3.8 `int mlABTree::verbose` [static, private]

- `adaboost.h`

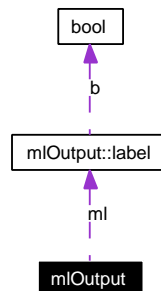
## 8.55 mlOutput Class Reference

```
#include <example.h>
```

Inheritance diagram for mlOutput:



Collaboration diagram for mlOutput:



### Public Member Functions

- **mlOutput** ()
- **mlOutput** (string)
- **~mlOutput** ()
- void **set\_label** (int l, bool b)
- bool **belongs\_to** (int l) const
- int **sign** (int l) const
- void **set\_weight** (int l, double w)
- double **weight** (int l) const
- void **set\_prediction** (int l, double pr)
- double **prediction** (int l)

### Static Public Member Functions

- static void **set\_separator** (char)
- static void **set\_nlabels** (int)
- static int **nlabels** ()

### Static Protected Attributes

- static int **\_nlabels**



## Private Member Functions

- void `parse_string` (string)

## Private Attributes

- `label * ml`

## Static Private Attributes

- static char `_separator`

## Classes

- struct `label`

### 8.55.1 Constructor & Destructor Documentation

8.55.1.1 `mlOutput::mlOutput ()`

8.55.1.2 `mlOutput::mlOutput (string)`

8.55.1.3 `mlOutput::~~mlOutput ()`

### 8.55.2 Member Function Documentation

8.55.2.1 `bool mlOutput::belongs_to (int l) const` [inline]

8.55.2.2 `static int mlOutput::nlabels ()` [static]

8.55.2.3 `void mlOutput::parse_string (string)` [private]

8.55.2.4 `double mlOutput::prediction (int l)` [inline]

8.55.2.5 `void mlOutput::set_label (int l, bool b)` [inline]

8.55.2.6 `static void mlOutput::set_nlabels (int)` [static]

8.55.2.7 `void mlOutput::set_prediction (int l, double pr)` [inline]

8.55.2.8 `static void mlOutput::set_separator (char)` [static]

8.55.2.9 `void mlOutput::set_weight (int l, double w)` [inline]

8.55.2.10 `int mlOutput::sign (int l) const` [inline]

8.55.2.11 `double mlOutput::weight (int l) const` [inline]

### 8.55.3 Member Data Documentation

8.55.3.1 `int mlOutput::_nlabels` [static, protected]

8.55.3.2 `char mlOutput::_separator` [static, private]

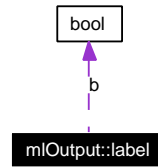
8.55.3.3 `label* mlOutput::ml` [private]

The documentation for this class was generated from the following file:

- `example.h`

## 8.56 mlOutput::label Struct Reference

Collaboration diagram for mlOutput::label:



### Public Member Functions

- `label ()`

### Public Attributes

- `bool b`
- `double weight`
- `double prediction`

### 8.56.1 Constructor & Destructor Documentation

8.56.1.1 `mlOutput::label::label ()` [inline]

### 8.56.2 Member Data Documentation

8.56.2.1 `bool mlOutput::label::b`

8.56.2.2 `double mlOutput::label::prediction`

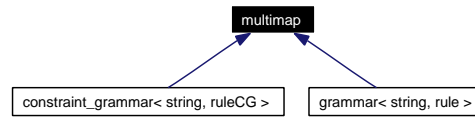
8.56.2.3 `double mlOutput::label::weight`

The documentation for this struct was generated from the following file:

- `example.h`

## 8.57 multimap Class Reference

Inheritance diagram for multimap:



The documentation for this class was generated from the following file:

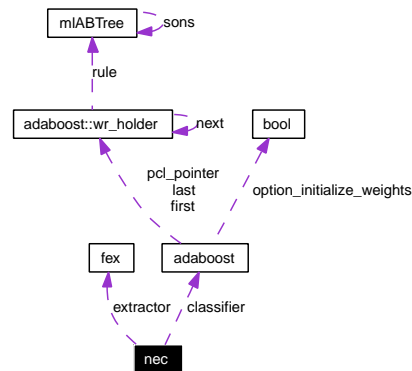
- **constraint\_grammar.h**

## 8.58 nec Class Reference

The class nec implements a ML-based NE classifier.

```
#include <nec.h>
```

Collaboration diagram for nec:



### Public Member Functions

- **nec** (const string &, const string &)  
*Constructor.*
- void **analyze** (list< sentence > &) const  
*Classify NEs in given sentence.*

### Private Attributes

- **fex \* extractor**  
*feature extractor*
- map< string, int > **lexicon**  
*lexicon to translate symbolic features to integer indexes*
- **adaboost \* classifier**  
*adaboost classifier*
- string **NPtag**

#### 8.58.1 Detailed Description

The class nec implements a ML-based NE classifier.

## 8.58.2 Constructor & Destructor Documentation

### 8.58.2.1 `nec::nec (const string &, const string &)`

Constructor.

## 8.58.3 Member Function Documentation

### 8.58.3.1 `void nec::analyze (list< sentence > &) const`

Classify NEs in given sentence.

## 8.58.4 Member Data Documentation

### 8.58.4.1 `adaboost* nec::classifier [private]`

adaboost classifier

### 8.58.4.2 `fex* nec::extractor [private]`

feature extractor

### 8.58.4.3 `map<string,int> nec::lexicon [private]`

lexicon to translate symbolic features to integer indexes

### 8.58.4.4 `string nec::NPtag [private]`

The documentation for this class was generated from the following files:

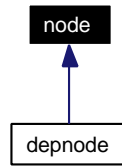
- `nec.h`
- `nec.cc`

## 8.59 node Class Reference

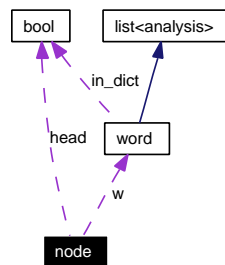
Class node stores nodes of a **parse\_tree**(p. 175) Each node in the tree is either a label (intermediate node) or a word (leaf node).

```
#include <language.h>
```

Inheritance diagram for node:



Collaboration diagram for node:



### Public Member Functions

- **node** ()  
*constructors*
- **node** (const string &)  
*Methods for parse\_tree(p. 175) nodes.*
- string **get\_label** (void) const  
*get node label*
- **word** **get\_word** (void) const  
*get node word*
- void **set\_label** (const string &)  
*set node label*
- void **set\_word** (**word** &)  
*set node word*
- bool **is\_head** (void) const  
*find out whether node is a head*

- void **set\_head** (const **bool**)  
*set whether node is a head*
- **bool** **is\_chunk** (void) const  
*find out whether node is a chunk*
- void **set\_chunk** (const int)  
*set position of the chunk in the sentence*
- int **get\_chunk\_ord** (void) const  
*get position of the chunk in the sentence*

## Protected Attributes

- **bool** **head**  
*is the node the head of the rule?*
- int **chunk**  
*is the node the root of a chunk? which?*
- string **label**  
*node label*
- **word** \* **w**  
*sentence word related to the node (if leaf)*

### 8.59.1 Detailed Description

Class node stores nodes of a **parse\_tree**(p.175) Each node in the tree is either a label (intermediate node) or a word (leaf node).

### 8.59.2 Constructor & Destructor Documentation

#### 8.59.2.1 node::node ()

constructors

#### 8.59.2.2 node::node (const string &)

Methods for **parse\_tree**(p.175) nodes.

### 8.59.3 Member Function Documentation

#### 8.59.3.1 int node::get\_chunk\_ord (void) const

get position of the chunk in the sentence



**8.59.3.2 string node::get\_\_label (void) const**

get node label

**8.59.3.3 word node::get\_\_word (void) const**

get node word

**8.59.3.4 bool node::is\_\_chunk (void) const**

find out whether node is a chunk

**8.59.3.5 bool node::is\_\_head (void) const**

find out whether node is a head

**8.59.3.6 void node::set\_\_chunk (const *int*)**

set position of the chunk in the sentence

**8.59.3.7 void node::set\_\_head (const *bool*)**

set whether node is a head

**8.59.3.8 void node::set\_\_label (const string &)**

set node label

**8.59.3.9 void node::set\_\_word (word &)**

set node word

**8.59.4 Member Data Documentation****8.59.4.1 int node::chunk [protected]**

is the node the root of a chunk? which?

**8.59.4.2 bool node::head [protected]**

is the node the head of the rule?

**8.59.4.3 string node::label [protected]**

node label

#### 8.59.4.4 word\* node::w [protected]

sentence word related to the node (if leaf)

The documentation for this class was generated from the following files:

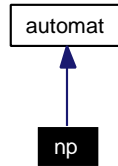
- **language.h**
- **language.cc**

## 8.60 np Class Reference

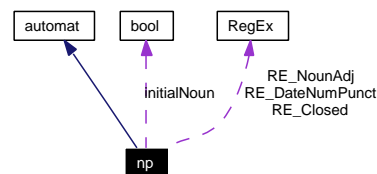
The class np implements a dummy proper noun recognizer.

```
#include <np.h>
```

Inheritance diagram for np:



Collaboration diagram for np:



### Public Member Functions

- **np** (const **maco\_options** &)  
*Constructor.*

### Private Member Functions

- int **ComputeToken** (int, sentence::const\_iterator, const **sentence** &)  
*Compute the right token code for word j from given state.*
- void **ResetActions** ()  
*Reset flag about capitalized noun at sentence start.*
- void **StateActions** (int, int, int, sentence::const\_iterator)  
*Perform necessary actions in "state" reached from state "origin" via word j interpreted as code "token": Basically, set flag about capitalized noun at sentence start.*
- void **SetMultiwordAnalysis** (sentence::iterator) const  
*Set the appropriate lemma and parole for the new multiword.*
- bool **ValidMultiWord** (const **word** &) const  
*Perform last minute validation before effectively building multiword.*
- bool **find\_tag\_match** (**RegEx** &, const sentence::const\_iterator)  
*Private method to look for a tag in a list of analysis.*

## Private Attributes

- **set< string > func**  
*set of function words*
- **set< string > punct**  
*set of special punctuation tags*
- **string NE\_tag**  
*Tag to assign to detected NEs.*
- **bool initialNoun**  
*it is a noun at the beggining of the sentence*
- **unsigned int Title\_length**  
*length beyond which a multiword made of all capitalized words ("WRECKAGE: TITANIC DIS-APPEARS IN NORTHERN SEA") will be considered a title and not a proper noun.*
- **RegEx RE\_NounAdj**
- **RegEx RE\_Closed**
- **RegEx RE\_DateNumPunct**

### 8.60.1 Detailed Description

The class `np` implements a dummy proper noun recognizer.

### 8.60.2 Constructor & Destructor Documentation

#### 8.60.2.1 `np::np (const maco_options &)`

Constructor.

### 8.60.3 Member Function Documentation

#### 8.60.3.1 `int np::ComputeToken (int, sentence::const_iterator, const sentence &)` [private, virtual]

Compute the right token code for word `j` from given state.

Implements **automat** (p. 41).

#### 8.60.3.2 `bool np::find_tag_match (RegEx &, const sentence::const_iterator)` [private]

Private method to look for a tag in a list of analysis.

#### 8.60.3.3 `void np::ResetActions ()` [private, virtual]

Reset flag about capitalized noun at sentence start.

Implements **automat** (p. 41).

**8.60.3.4** `void np::SetMultiwordAnalysis (sentence::iterator) const` [private, virtual]

Set the appropriate lemma and parole for the new multiword.

Implements **automat** (p. 41).

**8.60.3.5** `void np::StateActions (int, int, int, sentence::const_iterator)` [private, virtual]

Perform necessary actions in "state" reached from state "origin" via word j interpreted as code "token": Basically, set flag about capitalized noun at sentence start.

Implements **automat** (p. 42).

**8.60.3.6** `bool np::ValidMultiWord (const word &) const` [private, virtual]

Perform last minute validation before effectively building multiword.

Reimplemented from **automat** (p. 42).

## 8.60.4 Member Data Documentation

**8.60.4.1** `set<string> np::func` [private]

set of function words

**8.60.4.2** `bool np::initialNoun` [private]

it is a noun at the beggining of the sentence

**8.60.4.3** `string np::NE_tag` [private]

Tag to assign to detected NEs.

**8.60.4.4** `set<string> np::punct` [private]

set of special punctuation tags

**8.60.4.5** `RegEx np::RE_Closed` [private]

**8.60.4.6** `RegEx np::RE_DateNumPunct` [private]

**8.60.4.7** `RegEx np::RE_NounAdj` [private]

**8.60.4.8** `unsigned int np::Title_length` [private]

length beyond which a multiword made of all capitalizated words ("WRECKAGE: TITANIC DIS-APPEARS IN NORTHERN SEA") will be considered a title and not a proper noun.

A value of zero deactivates this behaviour.

The documentation for this class was generated from the following files:

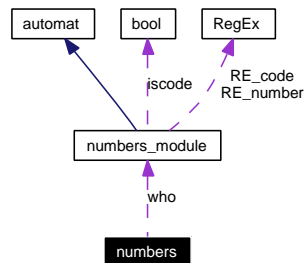
- **np.h**
- **np.cc**

## 8.61 numbers Class Reference

Class numbers implements a wrapper to transparently create and access a **numbers\_module**(p.171) number recognizer for the appropriate language.

```
#include <numbers.h>
```

Collaboration diagram for numbers:



### Public Member Functions

- **numbers** (const **maco\_options** &)  
*Constructor.*
- **~numbers** ()  
*Destructor.*
- void **annotate** (sentence &)  
*Detect number expressions in sentence using default options.*

### Private Attributes

- **numbers\_module** \* **who**  
*remember which module is doing the real work.*

#### 8.61.1 Detailed Description

Class numbers implements a wrapper to transparently create and access a **numbers\_module**(p.171) number recognizer for the appropriate language.

#### 8.61.2 Constructor & Destructor Documentation

##### 8.61.2.1 numbers::numbers (const maco\_options &)

Constructor.

##### 8.61.2.2 numbers::~~numbers ()

Destructor.

### 8.61.3 Member Function Documentation

#### 8.61.3.1 `void numbers::annotate (sentence &)`

Detect number expressions in sentence using default options.

### 8.61.4 Member Data Documentation

#### 8.61.4.1 `numbers_module* numbers::who` [private]

remember which module is doing the real work.

The documentation for this class was generated from the following files:

- `numbers.h`
- `numbers.cc`

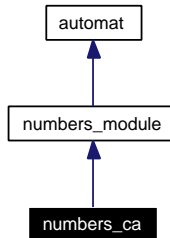


## 8.62 numbers\_ca Class Reference

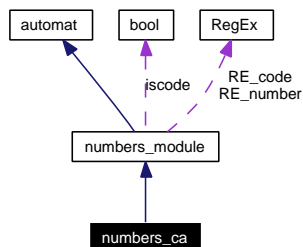
The derived class `numbers_ca` implements a Catalan number recognizer.

```
#include <numbers_modules.h>
```

Inheritance diagram for `numbers_ca`:



Collaboration diagram for `numbers_ca`:



### Public Member Functions

- **numbers\_ca** (const string &, const string &)  
*Constructor.*

### Private Member Functions

- int **ComputeToken** (int, sentence::const\_iterator, const sentence &)  
*Compute the right token code for word j from given state.*
- void **ResetActions** ()  
*Reset acumulators used by state actions: bilion, milion, units.*
- void **StateActions** (int, int, int, sentence::const\_iterator)  
*Perform necessary actions in "state" reached from state "origin" via word j interpreted as code "token": Basically, when reaching a state, update current numerical value.*
- void **SetMultiwordAnalysis** (sentence::iterator) const  
*Set the appropriate lemma and parole for the new multiword.*

### 8.62.1 Detailed Description

The derived class `numbers_ca` implements a Catalan number recognizer.

### 8.62.2 Constructor & Destructor Documentation

#### 8.62.2.1 `numbers_ca::numbers_ca (const string &, const string &)`

Constructor.

### 8.62.3 Member Function Documentation

#### 8.62.3.1 `int numbers_ca::ComputeToken (int, sentence::const_iterator, const sentence &) [private, virtual]`

Compute the right token code for word `j` from given state.

Implements **automat** (p. 41).

#### 8.62.3.2 `void numbers_ca::ResetActions () [private, virtual]`

Reset accumulators used by state actions: `bilion`, `million`, `units`.

Implements **automat** (p. 41).

#### 8.62.3.3 `void numbers_ca::SetMultiwordAnalysis (sentence::iterator) const [private, virtual]`

Set the appropriate lemma and parole for the new multiword.

Implements **automat** (p. 41).

#### 8.62.3.4 `void numbers_ca::StateActions (int, int, int, sentence::const_iterator) [private, virtual]`

Perform necessary actions in "state" reached from state "origin" via word `j` interpreted as code "token": Basically, when reaching a state, update current numerical value.

Implements **automat** (p. 42).

The documentation for this class was generated from the following files:

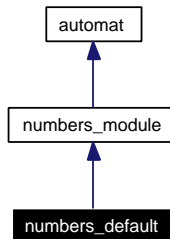
- `numbers_modules.h`
- `numbers_modules.cc`

## 8.63 numbers\_\_default Class Reference

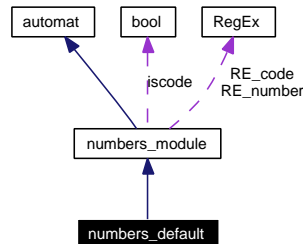
The derived class numbers\_\_default implements a default number recognizer (only numbers in digits are recognized).

```
#include <numbers_modules.h>
```

Inheritance diagram for numbers\_\_default:



Collaboration diagram for numbers\_\_default:



### Public Member Functions

- **numbers\_\_default** (const string &, const string &)  
*Constructor.*

### Private Member Functions

- int **ComputeToken** (int, sentence::const\_iterator, const sentence &)  
*Compute the right token code for word j from given state.*
- void **ResetActions** ()  
*Reset acumulators used by state actions: bilion, milion, units.*
- void **StateActions** (int, int, int, sentence::const\_iterator)  
*Perform necessary actions in "state" reached from state "origin" via word j interpreted as code "token": Basically, when reaching a state, update current numerical value.*
- void **SetMultiwordAnalysis** (sentence::iterator) const  
*Set the appropriate lemma and parole for the new multiword.*

### 8.63.1 Detailed Description

The derived class `numbers_default` implements a default number recognizer (only numbers in digits are recognized).

### 8.63.2 Constructor & Destructor Documentation

#### 8.63.2.1 `numbers_default::numbers_default (const string &, const string &)`

Constructor.

### 8.63.3 Member Function Documentation

#### 8.63.3.1 `int numbers_default::ComputeToken (int, sentence::const_iterator, const sentence &) [private, virtual]`

Compute the right token code for word `j` from given state.

Implements **automat** (p. 41).

#### 8.63.3.2 `void numbers_default::ResetActions () [private, virtual]`

Reset accumulators used by state actions: billion, million, units.

Implements **automat** (p. 41).

#### 8.63.3.3 `void numbers_default::SetMultiwordAnalysis (sentence::iterator) const [private, virtual]`

Set the appropriate lemma and parole for the new multiword.

Implements **automat** (p. 41).

#### 8.63.3.4 `void numbers_default::StateActions (int, int, int, sentence::const_iterator) [private, virtual]`

Perform necessary actions in "state" reached from state "origin" via word `j` interpreted as code "token": Basically, when reaching a state, update current numerical value.

Implements **automat** (p. 42).

The documentation for this class was generated from the following files:

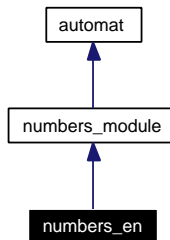
- `numbers_modules.h`
- `numbers_modules.cc`

## 8.64 numbers\_en Class Reference

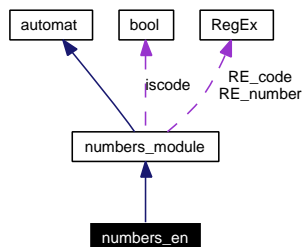
The derived class `numbers_en` implements an English number recognizer.

```
#include <numbers_modules.h>
```

Inheritance diagram for `numbers_en`:



Collaboration diagram for `numbers_en`:



### Public Member Functions

- **numbers\_en** (const string &, const string &)  
*Constructor.*

### Private Member Functions

- int **ComputeToken** (int, sentence::const\_iterator, const sentence &)  
*Compute the right token code for word j from given state.*
- void **ResetActions** ()  
*Reset acumulators used by state actions: bilion, milion, units.*
- void **StateActions** (int, int, int, sentence::const\_iterator)  
*Perform necessary actions in "state" reached from state "origin" via word j interpreted as code "token": Basically, when reaching a state, update current numerical value.*
- void **SetMultiwordAnalysis** (sentence::iterator) const  
*Set the appropriate lemma and parole for the new multiword.*

### 8.64.1 Detailed Description

The derived class `numbers_en` implements an English number recognizer.

### 8.64.2 Constructor & Destructor Documentation

#### 8.64.2.1 `numbers_en::numbers_en (const string &, const string &)`

Constructor.

### 8.64.3 Member Function Documentation

#### 8.64.3.1 `int numbers_en::ComputeToken (int, sentence::const_iterator, const sentence &) [private, virtual]`

Compute the right token code for word `j` from given state.

Implements **automat** (p. 41).

#### 8.64.3.2 `void numbers_en::ResetActions () [private, virtual]`

Reset accumulators used by state actions: billion, million, units.

Implements **automat** (p. 41).

#### 8.64.3.3 `void numbers_en::SetMultiwordAnalysis (sentence::iterator) const [private, virtual]`

Set the appropriate lemma and parole for the new multiword.

Implements **automat** (p. 41).

#### 8.64.3.4 `void numbers_en::StateActions (int, int, int, sentence::const_iterator) [private, virtual]`

Perform necessary actions in "state" reached from state "origin" via word `j` interpreted as code "token": Basically, when reaching a state, update current numerical value.

Implements **automat** (p. 42).

The documentation for this class was generated from the following files:

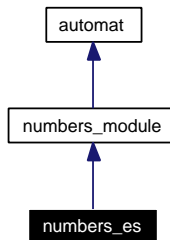
- `numbers_modules.h`
- `numbers_modules.cc`

## 8.65 numbers\_es Class Reference

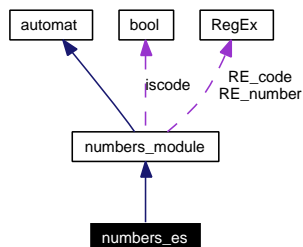
The derived class numbers\_es implements a Spanish number recognizer.

```
#include <numbers_modules.h>
```

Inheritance diagram for numbers\_es:



Collaboration diagram for numbers\_es:



### Public Member Functions

- **numbers\_es** (const string &, const string &)  
*Constructor.*

### Private Member Functions

- int **ComputeToken** (int, sentence::const\_iterator, const sentence &)  
*Compute the right token code for word j from given state.*
- void **ResetActions** ()  
*Reset acumulators used by state actions: bilion, milion, units.*
- void **StateActions** (int, int, int, sentence::const\_iterator)  
*Perform necessary actions in "state" reached from state "origin" via word j interpreted as code "token": Basically, when reaching a state, update current numerical value.*
- void **SetMultiwordAnalysis** (sentence::iterator) const  
*Set the appropriate lemma and parole for the new multiword.*

### 8.65.1 Detailed Description

The derived class `numbers_es` implements a Spanish number recognizer.

### 8.65.2 Constructor & Destructor Documentation

#### 8.65.2.1 `numbers_es::numbers_es (const string &, const string &)`

Constructor.

### 8.65.3 Member Function Documentation

#### 8.65.3.1 `int numbers_es::ComputeToken (int, sentence::const_iterator, const sentence &) [private, virtual]`

Compute the right token code for word `j` from given state.

Implements **automat** (p. 41).

#### 8.65.3.2 `void numbers_es::ResetActions () [private, virtual]`

Reset accumulators used by state actions: billion, million, units.

Implements **automat** (p. 41).

#### 8.65.3.3 `void numbers_es::SetMultiwordAnalysis (sentence::iterator) const [private, virtual]`

Set the appropriate lemma and parole for the new multiword.

Implements **automat** (p. 41).

#### 8.65.3.4 `void numbers_es::StateActions (int, int, int, sentence::const_iterator) [private, virtual]`

Perform necessary actions in "state" reached from state "origin" via word `j` interpreted as code "token": Basically, when reaching a state, update current numerical value.

Implements **automat** (p. 42).

The documentation for this class was generated from the following files:

- `numbers_modules.h`
- `numbers_modules.cc`

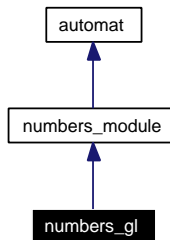


## 8.66 numbers\_gl Class Reference

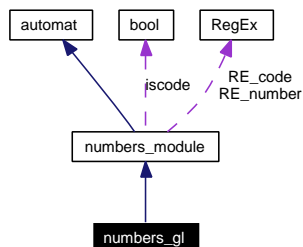
The derived class **numbers\_ca**(p. 161) implements a Galician number recognizer.

```
#include <numbers_modules.h>
```

Inheritance diagram for numbers\_gl:



Collaboration diagram for numbers\_gl:



### Public Member Functions

- **numbers\_gl** (const string &, const string &)  
*Constructor.*

### Private Member Functions

- int **ComputeToken** (int, sentence::const\_iterator, const **sentence** &)  
*Compute the right token code for word j from given state.*
- void **ResetActions** ()  
*Reset acumulators used by state actions: bilion, milion, units.*
- void **StateActions** (int, int, int, sentence::const\_iterator)  
*Perform necessary actions in "state" reached from state "origin" via word j interpreted as code "token": Basically, when reaching a state, update current numerical value.*
- void **SetMultiwordAnalysis** (sentence::iterator) const  
*Set the appropriate lemma and parole for the new multiword.*

### 8.66.1 Detailed Description

The derived class **numbers\_ca**(p. 161) implements a Galician number recognizer.

### 8.66.2 Constructor & Destructor Documentation

#### 8.66.2.1 **numbers\_gl::numbers\_gl** (const string &, const string &)

Constructor.

### 8.66.3 Member Function Documentation

#### 8.66.3.1 **int numbers\_gl::ComputeToken** (int, sentence::const\_iterator, const sentence &) [private, virtual]

Compute the right token code for word j from given state.

Implements **automat** (p. 41).

#### 8.66.3.2 **void numbers\_gl::ResetActions** () [private, virtual]

Reset acumulators used by state actions: bilion, milion, units.

Implements **automat** (p. 41).

#### 8.66.3.3 **void numbers\_gl::SetMultiwordAnalysis** (sentence::iterator) const [private, virtual]

Set the appropriate lemma and parole for the new multiword.

Implements **automat** (p. 41).

#### 8.66.3.4 **void numbers\_gl::StateActions** (int, int, int, sentence::const\_iterator) [private, virtual]

Perform necessary actions in "state" reached from state "origin" via word j interpreted as code "token": Basically, when reaching a state, update current numerical value.

Implements **automat** (p. 42).

The documentation for this class was generated from the following files:

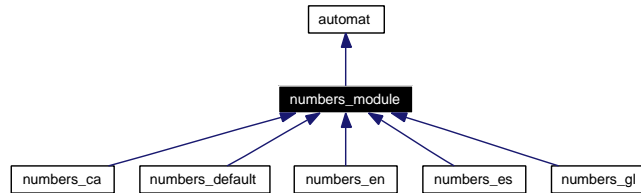
- **numbers\_modules.h**
- **numbers\_modules.cc**

## 8.67 numbers\_ module Class Reference

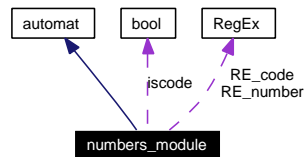
The abstract class numbers\_ module generalizes numeric expression recognizer for different languages.

```
#include <numbers_modules.h>
```

Inheritance diagram for numbers\_ module:



Collaboration diagram for numbers\_ module:



### Public Member Functions

- **numbers\_ module** (const string &, const string &)  
*Constructor.*

### Protected Attributes

- string **MACO\_Decimal**
- string **MACO\_Thousand**
- map< string, float > **value**  
*to map words into numerical values*
- map< string, int > **tok**  
*to map words into token codes*
- map< int, long double > **power**  
*to map value of power words (billion, million)*
- long double **bilion**  
*partial value of partially build number expression*
- long double **milion**
- long double **units**
- int **block**

- `bool iscode`
- `RegEx RE_number`
- `RegEx RE_code`

### 8.67.1 Detailed Description

The abstract class `numbers_module` generalizes numeric expression recognizer for different languages.

### 8.67.2 Constructor & Destructor Documentation

#### 8.67.2.1 `numbers_module::numbers_module (const string &, const string &)`

Constructor.

### 8.67.3 Member Data Documentation

#### 8.67.3.1 `long double numbers_module::billion` [protected]

partial value of partially build number expression

#### 8.67.3.2 `int numbers_module::block` [protected]

#### 8.67.3.3 `bool numbers_module::iscode` [protected]

#### 8.67.3.4 `string numbers_module::MACO_Decimal` [protected]

#### 8.67.3.5 `string numbers_module::MACO_Thousand` [protected]

#### 8.67.3.6 `long double numbers_module::million` [protected]

#### 8.67.3.7 `map<int,long double> numbers_module::power` [protected]

to map value of power words (billion, million)

#### 8.67.3.8 `RegEx numbers_module::RE_code` [protected]

#### 8.67.3.9 `RegEx numbers_module::RE_number` [protected]

#### 8.67.3.10 `map<string,int> numbers_module::tok` [protected]

to map words into token codes

#### 8.67.3.11 `long double numbers_module::units` [protected]

#### 8.67.3.12 `map<string,float> numbers_module::value` [protected]

to map words into numerical values

The documentation for this class was generated from the following files:

- numbers\_\_modules.h
- numbers\_\_modules.cc

## 8.68 paragraph Class Reference

Class paragraph is just a list of sentences that someone has validated it as a paragraph.

```
#include <language.h>
```

### 8.68.1 Detailed Description

Class paragraph is just a list of sentences that someone has validated it as a paragraph.

The documentation for this class was generated from the following file:

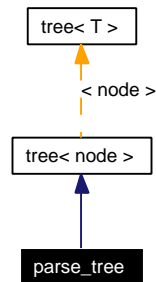
- **language.h**

## 8.69 parse\_tree Class Reference

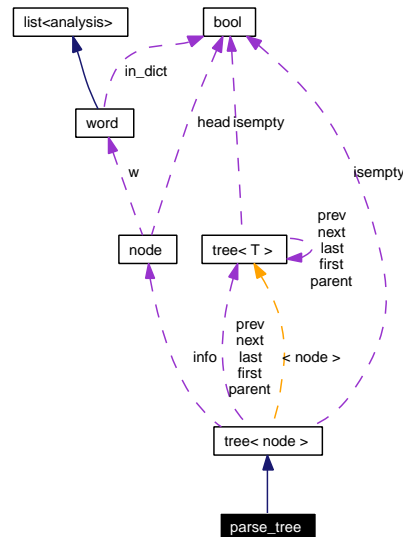
Class parse tree is used to store the results of parsing.

```
#include <language.h>
```

Inheritance diagram for parse\_tree:



Collaboration diagram for parse\_tree:



### Public Member Functions

- `parse_tree ()`  
*Constructors for parse\_tree.*
- `parse_tree (parse_tree::iterator p)`
- `parse_tree (const node &)`

### 8.69.1 Detailed Description

Class parse tree is used to store the results of parsing.

## 8.69.2 Constructor & Destructor Documentation

### 8.69.2.1 `parse_tree::parse_tree ()`

Constructors for `parse_tree`.

### 8.69.2.2 `parse_tree::parse_tree (parse_tree::iterator p)`

### 8.69.2.3 `parse_tree::parse_tree (const node &)`

The documentation for this class was generated from the following files:

- `language.h`
- `language.cc`

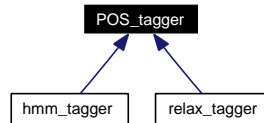


## 8.70 POS\_tagger Class Reference

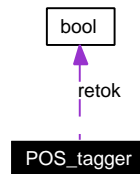
The class POS\_tagger is just an abstract class generalizing a PoS tagger.

```
#include <tagger.h>
```

Inheritance diagram for POS\_tagger:



Collaboration diagram for POS\_tagger:



### Public Member Functions

- **POS\_tagger (bool)**  
*Create an instance of the class, initializing options member.*
- virtual **~POS\_tagger ()**
- virtual void **analyze** (list< sentence > &)=0

### Static Protected Member Functions

- static void **retokenize** (list< sentence > &)  
*Look for words whose selected tag has retokenizing rules, and retokenize them appropriately.*

### Protected Attributes

- bool **retok**

#### 8.70.1 Detailed Description

The class POS\_tagger is just an abstract class generalizing a PoS tagger.

#### 8.70.2 Constructor & Destructor Documentation

##### 8.70.2.1 POS\_tagger::POS\_tagger (bool r)

Create an instance of the class, initializing options member.

Since `tagger` is an abstract class, this is called always from child constructors.

**8.70.2.2** `virtual POS_tagger::~~POS_tagger () [inline, virtual]`

### 8.70.3 Member Function Documentation

**8.70.3.1** `virtual void POS_tagger::analyze (list< sentence > &) [pure virtual]`

Implemented in `hmm_tagger` (p. 122), and `relax_tagger` (p. 202).

**8.70.3.2** `void POS_tagger::retokenize (list< sentence > &) [static, protected]`

Look for words whose selected tag has retokenizing rules, and retokenize them appropriately.

### 8.70.4 Member Data Documentation

**8.70.4.1** `bool POS_tagger::retok [protected]`

The documentation for this class was generated from the following files:

- `tagger.h`
- `tagger.cc`

## 8.71 probabilities Class Reference

Class probabilities sets lexical probabilities for each PoS tag of each word in a sentence.

```
#include <probabilities.h>
```

### Public Member Functions

- **probabilities** (const **maco\_options** &)  
*Constructor.*
- void **annotate** (**sentence** &)  
*Assign probabilities to tags using default options.*

### Private Member Functions

- double **compute\_\_probability** (const string &, double, string)  
*Compute  $p(\text{tag}|\text{suffix})$  using recursively shorter suffixes.*

### Private Attributes

- double **MACO\_ProbabilityThreshold**
- string **Language**
- map< string, double > **single\_tags**  
*unigram probabilities*
- map< string, map< string, double > > **class\_tags**  
*probabilities for usual ambiguity classes*
- map< string, map< string, double > > **lexical\_tags**  
*lexical probabilities for frequent words*
- map< string, double > **unk\_tags**  
*list of tags and probabilities to assign to unknown words*
- map< string, map< string, double > > **unk\_suffs**  
*list of tag frequencies for unknown word suffixes*
- double **theeta**  
*unknown words suffix smoothing parameter;*
- unsigned int **long\_suff**  
*length of longest suffix*

#### 8.71.1 Detailed Description

Class probabilities sets lexical probabilities for each PoS tag of each word in a sentence.

## 8.71.2 Constructor & Destructor Documentation

### 8.71.2.1 `probabilities::probabilities (const maco_options &)`

Constructor.

## 8.71.3 Member Function Documentation

### 8.71.3.1 `void probabilities::annotate (sentence &)`

Assign probabilities to tags using default options.

### 8.71.3.2 `double probabilities::compute_probability (const string &, double, string) [private]`

Compute  $p(\text{tag}|\text{suffix})$  using recursively shorter suffixes.

## 8.71.4 Member Data Documentation

### 8.71.4.1 `map<string,map<string,double> > probabilities::class_tags [private]`

probabilities for usual ambiguity classes

### 8.71.4.2 `string probabilities::Language [private]`

### 8.71.4.3 `map<string,map<string,double> > probabilities::lexical_tags [private]`

lexical probabilities for frequent words

### 8.71.4.4 `unsigned int probabilities::long_suff [private]`

length of longest suffix

### 8.71.4.5 `double probabilities::MACO_ProbabilityThreshold [private]`

### 8.71.4.6 `map<string,double> probabilities::single_tags [private]`

unigram probabilities

### 8.71.4.7 `double probabilities::theeta [private]`

unknown words suffix smoothing parameter;

### 8.71.4.8 `map<string,map<string,double> > probabilities::unk_suffs [private]`

list of tag frequencies for unknown word suffixes

#### 8.71.4.9 `map<string,double> probabilities::unk_tags` [private]

list of tags and probabilities to assign to unknown words

The documentation for this class was generated from the following files:

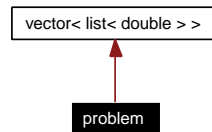
- `probabilities.h`
- `probabilities.cc`

## 8.72 problem Class Reference

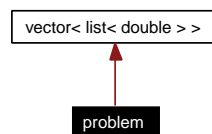
The class `problem` stores the structure of a problem, namely, a vector with a position for each variable to consider, and for each variable, a list of initial weights for each possible label.

```
#include <relax.h>
```

Inheritance diagram for `problem`:



Collaboration diagram for `problem`:



### Public Member Functions

- **problem** (int)  
*Constructor.*
- void **add\_label** (int, double)  
*add a label (and its weight) to the given variable*

### Friends

- class **relax**

#### 8.72.1 Detailed Description

The class `problem` stores the structure of a problem, namely, a vector with a position for each variable to consider, and for each variable, a list of initial weights for each possible label.

Variables and labels are unnamed, and sequentially numbered. The caller application must keep track of the meaning of each variable and label position.

#### 8.72.2 Constructor & Destructor Documentation

##### 8.72.2.1 `problem::problem` (int)

Constructor.

### 8.72.3 Member Function Documentation

#### 8.72.3.1 void problem::add\_label (int, double)

add a label (and its weight) to the given variable

### 8.72.4 Friends And Related Function Documentation

#### 8.72.4.1 friend class relax [friend]

The documentation for this class was generated from the following file:

- relax.h

## 8.73 punts Class Reference

Class numbers implements a punctuation sign recognizer.

```
#include <punts.h>
```

### Public Member Functions

- **punts** (const **maco\_options** &)  
*Constructor.*
- void **annotate** (**sentence** &)  
*Detect numbers in sentence with default options.*

### Private Attributes

- map< string, string > **punct**  
*map of punctuation signs and corresponding parole tags*

#### 8.73.1 Detailed Description

Class numbers implements a punctuation sign recognizer.

#### 8.73.2 Constructor & Destructor Documentation

##### 8.73.2.1 punts::punts (const maco\_options &)

Constructor.

#### 8.73.3 Member Function Documentation

##### 8.73.3.1 void punts::annotate (sentence &)

Detect numbers in sentence with default options.

#### 8.73.4 Member Data Documentation

##### 8.73.4.1 map<string,string> punts::punct [private]

map of punctuation signs and corresponding parole tags

The documentation for this class was generated from the following files:

- **punts.h**
- **punts.cc**

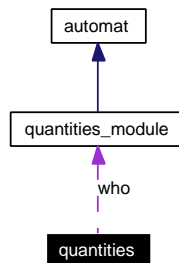


## 8.74 quantities Class Reference

Class quantities implements a wrapper to transparently create and access a **quantities\_module**(p.191) monetary expressions detector for the appropriate language.

```
#include <quantities.h>
```

Collaboration diagram for quantities:



### Public Member Functions

- **quantities** (const **maco\_options** &)  
*Constructor.*
- **~quantities** ()  
*Destructor.*
- void **annotate** (sentence &)  
*Detect time expressions in sentence using default options.*

### Private Attributes

- **quantities\_module** \* **who**  
*remember which module is doing the real work.*

#### 8.74.1 Detailed Description

Class quantities implements a wrapper to transparently create and access a **quantities\_module**(p.191) monetary expressions detector for the appropriate language.

#### 8.74.2 Constructor & Destructor Documentation

##### 8.74.2.1 quantities::quantities (const **maco\_options** &)

Constructor.

##### 8.74.2.2 quantities::~~quantities ()

Destructor.

### 8.74.3 Member Function Documentation

#### 8.74.3.1 `void quantities::annotate (sentence &)`

Detect time expressions in sentence using default options.

### 8.74.4 Member Data Documentation

#### 8.74.4.1 `quantities_module* quantities::who` [private]

remember which module is doing the real work.

The documentation for this class was generated from the following files:

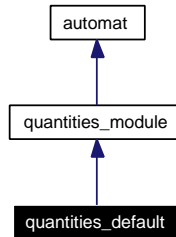
- `quantities.h`
- `quantities.cc`

## 8.75 quantities\_\_default Class Reference

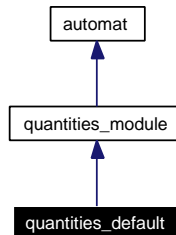
The derived class quantities\_\_default implements a default quantities recognizer (only percentages are recognized).

```
#include <quantities_modules.h>
```

Inheritance diagram for quantities\_\_default:



Collaboration diagram for quantities\_\_default:



### Public Member Functions

- **quantities\_\_default** (const string &)  
*Constructor.*

### Private Member Functions

- int **ComputeToken** (int, sentence::const\_iterator, const **sentence** &)  
*Compute the right token code for word j from given state.*
- void **ResetActions** ()  
*Reset acumulators used by state actions: value1, value2 (for ratios).*
- void **StateActions** (int, int, int, sentence::const\_iterator)  
*Perform necessary actions in "state" reached from state "origin" via word j interpreted as code "token": Basically, when reaching a state with an informative.*
- void **SetMultiwordAnalysis** (sentence::iterator) const  
*Set the appropriate lemma and parole for the new multiword.*

### 8.75.1 Detailed Description

The derived class `quantities_default` implements a default quantities recognizer (only percentages are recognized).

### 8.75.2 Constructor & Destructor Documentation

#### 8.75.2.1 `quantities_default::quantities_default (const string &)`

Constructor.

### 8.75.3 Member Function Documentation

#### 8.75.3.1 `int quantities_default::ComputeToken (int, sentence::const_iterator, const sentence &) [private, virtual]`

Compute the right token code for word `j` from given state.

Implements **automat** (p. 41).

#### 8.75.3.2 `void quantities_default::ResetActions () [private, virtual]`

Reset accumulators used by state actions: `value1`, `value2` (for ratios).

Implements **automat** (p. 41).

#### 8.75.3.3 `void quantities_default::SetMultiwordAnalysis (sentence::iterator) const [private, virtual]`

Set the appropriate lemma and parole for the new multiword.

Implements **automat** (p. 41).

#### 8.75.3.4 `void quantities_default::StateActions (int, int, int, sentence::const_iterator) [private, virtual]`

Perform necessary actions in "state" reached from state "origin" via word `j` interpreted as code "token": Basically, when reaching a state with an informative.

Implements **automat** (p. 42).

The documentation for this class was generated from the following files:

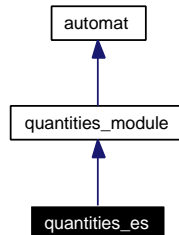
- `quantities_modules.h`
- `quantities_modules.cc`

## 8.76 quantities\_\_es Class Reference

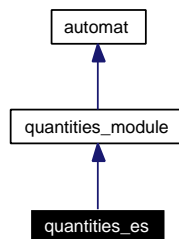
The derived class quantities\_\_es implements a Spanish (and Catalan) quantities recognizer.

```
#include <quantities_modules.h>
```

Inheritance diagram for quantities\_\_es:



Collaboration diagram for quantities\_\_es:



### Public Member Functions

- **quantities\_\_es** (const string &)  
*Constructor.*

### Private Member Functions

- int **ComputeToken** (int, sentence::const\_iterator, const sentence &)  
*Compute the right token code for word j from given state.*
- void **ResetActions** ()  
*Reset accumulators used by state actions: value1, value2 (for ratios).*
- void **StateActions** (int, int, int, sentence::const\_iterator)  
*Perform necessary actions in "state" reached from state "origin" via word j interpreted as code "token": Basically, when reaching a state with an informative.*
- void **SetMultiwordAnalysis** (sentence::iterator) const  
*Set the appropriate lemma and parole for the new multiword.*

### 8.76.1 Detailed Description

The derived class `quantities_es` implements a Spanish (and Catalan) quantities recognizer.

### 8.76.2 Constructor & Destructor Documentation

#### 8.76.2.1 `quantities_es::quantities_es (const string &)`

Constructor.

### 8.76.3 Member Function Documentation

#### 8.76.3.1 `int quantities_es::ComputeToken (int, sentence::const_iterator, const sentence &) [private, virtual]`

Compute the right token code for word `j` from given state.

Implements **automat** (p. 41).

#### 8.76.3.2 `void quantities_es::ResetActions () [private, virtual]`

Reset accumulators used by state actions: `value1`, `value2` (for ratios).

Implements **automat** (p. 41).

#### 8.76.3.3 `void quantities_es::SetMultiwordAnalysis (sentence::iterator) const [private, virtual]`

Set the appropriate lemma and parole for the new multiword.

Implements **automat** (p. 41).

#### 8.76.3.4 `void quantities_es::StateActions (int, int, int, sentence::const_iterator) [private, virtual]`

Perform necessary actions in "state" reached from state "origin" via word `j` interpreted as code "token": Basically, when reaching a state with an informative.

Implements **automat** (p. 42).

The documentation for this class was generated from the following files:

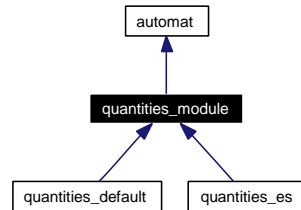
- `quantities_modules.h`
- `quantities_modules.cc`

## 8.77 quantities\_\_module Class Reference

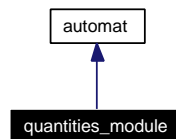
The abstract class quantities\_\_module generalizes a percentage, ratios, and currency expression recognizer for different languages.

```
#include <quantities_modules.h>
```

Inheritance diagram for quantities\_\_module:



Collaboration diagram for quantities\_\_module:



### Public Member Functions

- **quantities\_\_module ()**  
*Constructor.*

### Protected Attributes

- map< string, int > **tok**  
*translate particular strings to token codes*
- map< string, long double > **fract**  
*translate fraction strings to their numerical values*
- map< string, string > **currency**  
*list of currency names and their international codes*
- map< string, string > **units**  
*list of measure units and their lexical realizations*
- long double **value1**  
*values for ratios*
- long double **value2**
- string **currCodes**

*auxiliary for nation-specific currencies*

- string **unitCode**

*auxiliary for storing standarized unit description*

### 8.77.1 Detailed Description

The abstract class `quantities_module` generalizes a percentage, ratios, and currency expression recognizer for different languages.

### 8.77.2 Constructor & Destructor Documentation

#### 8.77.2.1 `quantities_module::quantities_module ()`

Constructor.

### 8.77.3 Member Data Documentation

#### 8.77.3.1 `string quantities_module::currCodes` [protected]

*auxiliary for nation-specific currencies*

#### 8.77.3.2 `map<string,string> quantities_module::currency` [protected]

list of currency names and their international codes

#### 8.77.3.3 `map<string,long double> quantities_module::fract` [protected]

translate fraction strings to their nummerical values

#### 8.77.3.4 `map<string,int> quantities_module::tok` [protected]

translate particular strings to token codes

#### 8.77.3.5 `string quantities_module::unitCode` [protected]

*auxiliary for storing standarized unit description*

#### 8.77.3.6 `map<string,string> quantities_module::units` [protected]

list of measure units and their lexical realizations

#### 8.77.3.7 `long double quantities_module::value1` [protected]

values for ratios



### 8.77.3.8 long double quantities\_\_module::value2 [protected]

The documentation for this class was generated from the following files:

- quantities\_\_modules.h
- quantities\_\_modules.cc

## 8.78 RegEx Class Reference

class RegEx, a simple and small API wrapper for PCRE.

```
#include <regex.h>
```

### Public Member Functions

- **RegEx** (const string &regex, int options=0)  
*Constructor.*
- **RegEx** (const **RegEx** &y)  
*Copy constructor.*
- **RegEx & operator=** (const **RegEx** &y)  
*Assignment.*
- **~RegEx** ()  
*Destructor.*
- int **SubStrings** (void) const  
*Get number of substrings.*
- bool **Search** (const char \*subject, int len=-1, int options=0)  
*Search RE in string.*
- bool **SearchAgain** (int options=0)  
*Search RE in string again.*
- string **Match** (int i=1)  
*Get match for i-th substring in regexp.*

### Private Member Functions

- void **clone** (const **RegEx** &y)  
*auxiliary for copy and assignation constructors*
- void **ClearMatchList** (void)  
*Clear match list.*

### Private Attributes

- pcre \* **re**
- pcre\_extra \* **pe**
- int **substrcount**
- int \* **ovector**
- const char \* **lastsubject**
- int **slen**
- const char \*\* **matchlist**

### 8.78.1 Detailed Description

class RegEx, a simple and small API wrapper for PCRE.

**RegEx::RegEx**(p. 195)(string regex, int options = 0)

The constructor's first parameter is the regular expression the created object shall implement. Optional parameter options can be any combination of PCRE options accepted by `pcre_compile()`. If compiling the regular expression fails, an error message string is thrown as an exception.

**RegEx::~~RegEx**()(p. 196)

The destructor frees all resources held by the RegEx object.

int **RegEx::SubStrings**(void) const(p. 196)

Method **SubStrings**()(p. 196) returns the number of substrings defined by the regular expression. The match of the entire expression is also considered a substring, so the return value will always be  $\geq 1$ .

bool **RegEx::Search**(p. 196)(const char \* subject, int len = -1, int options = 0)

Method **Search**()(p. 196) applies the regular expression to parameter subject. Optional parameter len can be used to pass the subject's length to **Search**()(p. 196). If not specified (or less than 0), `strlen()` is used internally to determine the length. Parameter options can contain any combination of options PCRE\_ANCHORED, PCRE\_NOTBOL, PCRE\_NOTEOL, PCRE\_NOTEMPTY. **Search**()(p. 196) returns true if a match is found.

bool **RegEx::SearchAgain**(p. 196)(int options = 0)

**SearchAgain**()(p. 196) again applies the regular expression to parameter subject last passed to a successful call of **Search**()(p. 196). It returns true if a further match is found. Subsequent calls to **SearchAgain**()(p. 196) will find all matches in subject. Example:

```
if (Pattern.Search(astring)) { do { printf("%s\n", Pattern.Match()); } while (Pattern.SearchAgain()); }
```

Parameter options is interpreted as for method **Search**()(p. 196).

string **RegEx::Match**(p. 196)(int i = 1)

Method **Match**()(p. 196) returns a pointer to the matched substring specified with parameter i. **Match**()(p. 196) may only be called after a successful call to **Search**()(p. 196) or **SearchAgain**()(p. 196) and applies to that last **Search**()(p. 196)/**SearchAgain**() call. Parameter i must be less than **SubStrings**()(p. 196). **Match**(-1) returns the last searched subject. **Match**(0) returns the match of the complete regular expression. **Match**(1) returns \$1, etc.

The bottom of this file contains an example using class RegEx. It's the simplest version of grep I could come with. You can compile it by defining REGEX\_DEMO on the compiler command line.

### 8.78.2 Constructor & Destructor Documentation

#### 8.78.2.1 RegEx::RegEx (const string & regex, int options = 0) [inline]

Constructor.

#### 8.78.2.2 RegEx::RegEx (const RegEx & y) [inline]

Copy constructor.

**8.78.2.3    `RegEx::~~RegEx ()`    [inline]**

Destructor.

**8.78.3    Member Function Documentation****8.78.3.1    `void RegEx::ClearMatchList (void)`    [inline, private]**

Clear match list.

**8.78.3.2    `void RegEx::clone (const RegEx & y)`    [inline, private]**

auxiliary for copy and assignation constructors

**8.78.3.3    `string RegEx::Match (int i = 1)`    [inline]**

Get match for i-th substring in regexp.

**8.78.3.4    `RegEx& RegEx::operator= (const RegEx & y)`    [inline]**

Assignation.

**8.78.3.5    `bool RegEx::Search (const char * subject, int len = -1, int options = 0)`  
              [inline]**

Search RE in string.

**8.78.3.6    `bool RegEx::SearchAgain (int options = 0)`    [inline]**

Search RE in string again.

**8.78.3.7    `int RegEx::SubStrings (void) const`    [inline]**

Get number of substrings.

## 8.78.4 Member Data Documentation

8.78.4.1 `const char* RegEx::lastsubject` [private]

8.78.4.2 `const char* * RegEx::matchlist` [private]

8.78.4.3 `int* RegEx::ovector` [private]

8.78.4.4 `pcre_extra* RegEx::pe` [private]

8.78.4.5 `pcre* RegEx::re` [private]

8.78.4.6 `int RegEx::slen` [private]

8.78.4.7 `int RegEx::substrcount` [private]

The documentation for this class was generated from the following file:

- `regexp.h`

## 8.79 relax Class Reference

The class relax implements a generic solver for consistent labelling problems, using relaxation labelling algorithm.

```
#include <relax.h>
```

### Public Member Functions

- **relax** (int, double, double)  
*Constructor.*
- void **reset** (const **problem** &)  
*Prepare for a new problem (i.e. free tables and alloc for the new problem).*
- void **add\_constraint** (int, int, const **list**< **list**< pair< int, int > > &, double)  
*add a new constraint to the problem*
- void **solve** ()  
*solve consistent labelling problem*
- **list**< int > **best\_label** (int) const  
*get best label(s) –hopefully only one– for given variable*

### Private Member Functions

- double **NormalizeSupport** (double) const  
*private methods*
- bool **there\_are\_changes** () const

### Private Attributes

- **vector**< **vector**< **label** > > **vars**  
*table with variable-labels in the CLP.*
- int **CURRENT**  
*which of both weight sets are we using and which are we computing*
- int **NEXT**
- int **MaxIter**  
*Maximum number of iterations in case of not converging.*
- double **ScaleFactor**  
*Scale factor for label supports.*
- double **Epsilon**  
*epsilon value to decide whether or not an iteration has caused relevant weight changes*

### 8.79.1 Detailed Description

The class relax implements a generic solver for consistent labelling problems, using relaxation labelling algorithm.

### 8.79.2 Constructor & Destructor Documentation

#### 8.79.2.1 relax::relax (int, double, double)

Constructor.

### 8.79.3 Member Function Documentation

#### 8.79.3.1 void relax::add\_constraint (int, int, const list< list< pair< int, int > > > &, double)

add a new constraint to the problem

#### 8.79.3.2 list<int> relax::best\_label (int) const

get best label(s) –hopefully only one– for given variable

#### 8.79.3.3 double relax::NormalizeSupport (double) const [private]

private methods

#### 8.79.3.4 void relax::reset (const problem &)

Prepare for a new problem (i.e. free tables and alloc for the new problem).

#### 8.79.3.5 void relax::solve ()

solve consistent labelling problem

#### 8.79.3.6 bool relax::there\_are\_changes () const [private]

### 8.79.4 Member Data Documentation

#### 8.79.4.1 int relax::CURRENT [private]

which of both weight sets are we using and which are we computing

#### 8.79.4.2 double relax::Epsilon [private]

epsilon value to decide whether or not an iteration has caused relevant weight changes

**8.79.4.3** `int relax::MaxIter` [private]

Maximum number of iterations in case of not converging.

**8.79.4.4** `int relax::NEXT` [private]**8.79.4.5** `double relax::ScaleFactor` [private]

Scale factor for label supports.

**8.79.4.6** `vector<vector<label> > relax::vars` [private]

table with variable-labels in the CLP.

The documentation for this class was generated from the following file:

- `relax.h`

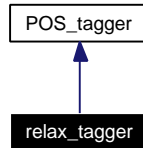


## 8.80 relax\_tagger Class Reference

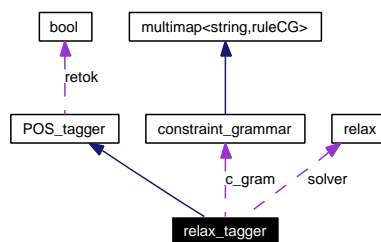
The class relax\_tagger implements a PoS tagger based on relaxation labelling algorithm.

```
#include <relax_tagger.h>
```

Inheritance diagram for relax\_tagger:



Collaboration diagram for relax\_tagger:



### Public Member Functions

- **relax\_tagger** (const string &, int, double, double, **bool**)  
*Constructor, given the constraints file and config parameters.*
- void **analyze** (list< sentence > &)  
*analyze sentences with default options*
- list< sentence > **analyze** (const list< sentence > &)  
*analyze sentences, return analyzed copy*

### Private Member Functions

- **bool CheckCondition** (const sentence &, sentence::const\_iterator, int, const **condition** &, list< list< pair< int, int > > > &) const  
*check a condition of a RuleCG.*
- **bool CheckWordMatchCondition** (const list< string > &, **bool**, int, sentence::const\_iterator, list< pair< int, int > > &) const  
*check whether a word matches a simple list of terms.*
- **bool check\_possible\_matching** (const string &, word::const\_iterator, sentence::const\_iterator) const  
*check whether a word matches one of all possible condition patterns*

- **bool check\_match** (const string &, const string &) const  
*check a match of two literals, taking into account lemma, tag, wildcards, etc.*

## Private Attributes

- **relax solver**  
*Generic solver instance.*
- **constraint\_grammar c\_gram**  
*PoS constraints.*

### 8.80.1 Detailed Description

The class `relax_tagger` implements a PoS tagger based on relaxation labelling algorithm. It does so using the generic solver implemented by class `relax`.

### 8.80.2 Constructor & Destructor Documentation

#### 8.80.2.1 `relax_tagger::relax_tagger` (const string &, int, double, double, bool)

Constructor, given the constraints file and config parameters.

### 8.80.3 Member Function Documentation

#### 8.80.3.1 `list<sentence> relax_tagger::analyze` (const list< sentence > &)

analyze sentences, return analyzed copy

#### 8.80.3.2 `void relax_tagger::analyze` (list< sentence > &) [virtual]

analyze sentences with default options

Implements **POS\_tagger** (p.178).

#### 8.80.3.3 `bool relax_tagger::check_match` (const string &, const string &) const [private]

check a match of two literals, taking into account lemma, tag, wildcards, etc.

#### 8.80.3.4 `bool relax_tagger::check_possible_matching` (const string &, word::const\_iterator, sentence::const\_iterator) const [private]

check whether a word matches one of all possible condition patterns

**8.80.3.5** `bool relax_tagger::CheckCondition (const sentence &, sentence::const_iterator, int, const condition &, list< list< pair< int, int > > > &) const` [private]

check a condition of a RuleCG.

Add to the given constraint& solver-encoded constraint info for the condition

**8.80.3.6** `bool relax_tagger::CheckWordMatchCondition (const list< string > &, bool, int, sentence::const_iterator, list< pair< int, int > > &) const` [private]

check whether a word matches a simple list of terms.

Return (via list<pair<int,int>>&) a solver-encoded term for the condition

## 8.80.4 Member Data Documentation

**8.80.4.1** `constraint_grammar relax_tagger::c_gram` [private]

PoS constraints.

**8.80.4.2** `relax relax_tagger::solver` [private]

Generic solver instance.

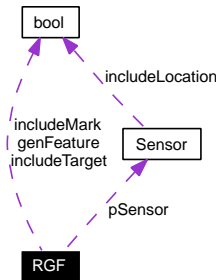
The documentation for this class was generated from the following file:

- `relax_tagger.h`

## 8.81 RGF Struct Reference

```
#include <RGF.h>
```

Collaboration diagram for RGF:



### Public Member Functions

- **RGF** ()
- **RGF** (char \*sensorName, char \*p1, char \*p2, char \*p3, char \*dpath)
- **RGF** (char \*sensorName)
- **RGF** (SubRGF &subR)
- ~**RGF** ()
- **bool operator==** (const **RGF** &rhs) const
- **bool operator<=** (const **RGF** &rhs) const
- **bool operator<** (const **RGF** &rhs) const
- **bool operator>** (const **RGF** &rhs) const
- **bool operator>=** (const **RGF** &rhs) const
- set< string > **Extract** (const **sentence** &sent, int targIndex)
- set< string > **Extract** (const **sentence** &sent, int targIndex, int targLength)
- int **TargetIndex** ()
- void **Target** (char \*targ)
- const char \* **Target** () const
- void **IncludeLocation** (bool val)
- void **LocationOffset** (int val)
- int **LocationOffset** () const
- void **LeftOffset** (int val)
- int **LeftOffset** () const
- void **RightOffset** (int val)
- int **RightOffset** () const
- void **IncludeTarget** (bool val)
- **bool IncludeTarget** () const
- void **IncludeMark** (bool val)
- **bool IncludeMark** () const
- void **Mode** (ExtractMode mode)
- **ExtractMode Mode** ()
- void **Mask** (char \*val)
- char \* **Mask** ()
- void **Param** (char \*val)
- const char \* **Param** ()

- void **GenFeature** (bool val)
- void **Insert** (RGF rel)
- void **Show** ()

## Protected Member Functions

- set< string > **Process** (const **sentence** &sent, int rec, int targIndex, int start, int end)

## Protected Attributes

- SubRGF subRGFs
- char \* **optParam**
- bool **includeTarget**
- bool **includeMark**
- int **targetIndex**
- int **leftOffset**
- int **rightOffset**
- int **locationOffset**
- ExtractMode **extractMode**
- char \* **target**
- char \* **mask**
- bool **genFeature**
- Sensor \* **pSensor**

## 8.81.1 Constructor & Destructor Documentation

8.81.1.1 RGF::RGF ()

8.81.1.2 RGF::RGF (char \* *sensorName*, char \* *p1*, char \* *p2*, char \* *p3*, char \* *dpath*)

8.81.1.3 RGF::RGF (char \* *sensorName*)

8.81.1.4 RGF::RGF (SubRGF & *subR*)

8.81.1.5 RGF::~~RGF ()

## 8.81.2 Member Function Documentation

8.81.2.1 set<string> RGF::Extract (const sentence & *sent*, int *targIndex*, int *targLength*)

8.81.2.2 set<string> RGF::Extract (const sentence & *sent*, int *targIndex*)

8.81.2.3 void RGF::GenFeature (bool *val*) [inline]

8.81.2.4 void RGF::IncludeLocation (bool *val*)

8.81.2.5 bool RGF::IncludeMark () const [inline]

8.81.2.6 void RGF::IncludeMark (bool *val*)

8.81.2.7 bool RGF::IncludeTarget () const [inline]

8.81.2.8 void RGF::IncludeTarget (bool *val*)

8.81.2.9 void RGF::Insert (RGF *rel*) [inline]

8.81.2.10 int RGF::LeftOffset () const [inline]

8.81.2.11 void RGF::LeftOffset (int *val*) [inline]

8.81.2.12 int RGF::LocationOffset () const [inline]

8.81.2.13 void RGF::LocationOffset (int *val*)

8.81.2.14 char\* RGF::Mask () [inline]

8.81.2.15 void RGF::Mask (char \* *val*) [inline]

8.81.2.16 ExtractMode RGF::Mode () [inline]

8.81.2.17 void RGF::Mode (ExtractMode *mode*)

8.81.2.18 bool RGF::operator< (const RGF & *rhs*) const

8.81.2.19 bool RGF::operator<= (const RGF & *rhs*) const

~~8.81.2.20 bool RGF::operator== (const RGF & *rhs*) const~~

Generated on Wed Apr 26 12:55:30 2006 for FreeLing by Doxygen

8.81.2.21 bool RGF::operator> (const RGF & *rhs*) const

8.81.2.22 bool RGF::operator>= (const RGF & *rhs*) const

8.81.2.23 const char\* RGF::Param () [inline]

- RGF.h

## 8.82 rule Class Reference

Class rule implements a rule of a grammar.

```
#include <grammar.h>
```

Inheritance diagram for rule:



### Public Member Functions

- **rule** (const string &, const **list**< string > &, const int)

*Constructors of the subclass rule.*

- **rule** (const **rule** &r)
- **rule** ()
- **rule** & **operator=** (const **rule** &)
- void **set\_governor** (const int)

*set rule governor*

- unsigned int **get\_governor** (void) const

*get rule governor*

- string **get\_head** () const

*get rule head.*

- **list**< string > **get\_right** () const

*get right part of the rule.*

### Protected Attributes

- string **head**

*Head of the rule.*

- **list**< string > **right**

*Right part of the rule.*

- int **gov**

*Position of the rule Governor of.*



### 8.82.1 Detailed Description

Class rule implements a rule of a grammar.

### 8.82.2 Constructor & Destructor Documentation

#### 8.82.2.1 rule::rule (const string &, const list< string > &, const int)

Constructors of the subclass rule.

#### 8.82.2.2 rule::rule (const rule & r)

#### 8.82.2.3 rule::rule ()

### 8.82.3 Member Function Documentation

#### 8.82.3.1 unsigned int rule::get\_governor (void) const

get rule governor

#### 8.82.3.2 string rule::get\_head () const

get rule head.

#### 8.82.3.3 list<string> rule::get\_right () const

get right part of the rule.

#### 8.82.3.4 rule& rule::operator= (const rule &)

#### 8.82.3.5 void rule::set\_governor (const int)

set rule governor

### 8.82.4 Member Data Documentation

#### 8.82.4.1 int rule::gov [protected]

Position of the rule Governor of.

#### 8.82.4.2 string rule::head [protected]

Head of the rule.

#### 8.82.4.3 list<string> rule::right [protected]

Right part of the rule.

The documentation for this class was generated from the following file:

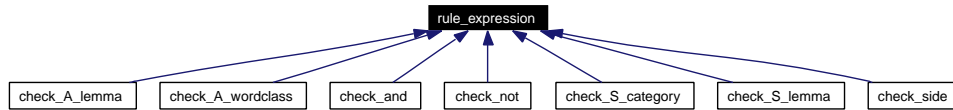
- `grammar.h`

## 8.83 rule\_expression Class Reference

The class `rule_expression` is an abstract class (interface) for building dynamic restriction on a `ruleLabeler`(p. 215) which are used by class `depLabeler`(p. 99).

```
#include <dep_rules.h>
```

Inheritance diagram for `rule_expression`:



### Public Member Functions

- **rule\_expression** ()  
*Constructor.*
- **rule\_expression** (const string &)  
*Constructor.*
- virtual **~rule\_expression** ()
- **bool find** (const string &) const  
*Search for a value in the list of a expression.*
- virtual **bool eval** (dep\_tree::iterator, dep\_tree::iterator) const =0

### Protected Attributes

- set< string > **valueList**

#### 8.83.1 Detailed Description

The class `rule_expression` is an abstract class (interface) for building dynamic restriction on a `ruleLabeler`(p. 215) which are used by class `depLabeler`(p. 99).

#### 8.83.2 Constructor & Destructor Documentation

##### 8.83.2.1 rule\_expression::rule\_expression ()

Constructor.

##### 8.83.2.2 rule\_expression::rule\_expression (const string &)

Constructor.

**8.83.2.3** `virtual rule_expression::~~rule_expression ()` [inline, virtual]

### 8.83.3 Member Function Documentation

**8.83.3.1** `virtual bool rule_expression::eval (dep_tree::iterator, dep_tree::iterator) const` [pure virtual]

Implemented in `check_and` (p. 54), `check_not` (p. 56), `check_side` (p. 63), `check_A_lemma` (p. 51), `check_S_lemma` (p. 61), `check_S_category` (p. 59), and `check_A_wordclass` (p. 53).

**8.83.3.2** `bool rule_expression::find (const string &) const`

Search for a value in the list of a expression.

### 8.83.4 Member Data Documentation

**8.83.4.1** `set<string> rule_expression::valueList` [protected]

The documentation for this class was generated from the following files:

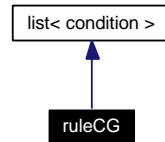
- `dep_rules.h`
- `dep_rules.cc`

## 8.84 ruleCG Class Reference

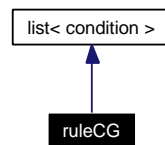
Class rule implements a rule of a CG.

```
#include <constraint_grammar.h>
```

Inheritance diagram for ruleCG:



Collaboration diagram for ruleCG:



### Public Member Functions

- **ruleCG ()**  
*Constructors of the subclass rule.*
- void **set\_head** (const string &)  
*set rule head.*
- void **set\_weight** (double)  
*set rule compatibility.*
- string **get\_head** () const  
*get rule head.*
- double **get\_weight** () const  
*set rule compatibility.*

### Protected Attributes

- double **weight**  
*compatibility value*
- string **head**  
*Head of the rule.*

### 8.84.1 Detailed Description

Class rule implements a rule of a CG.

### 8.84.2 Constructor & Destructor Documentation

#### 8.84.2.1 ruleCG::ruleCG ()

Constructors of the subclass rule.

### 8.84.3 Member Function Documentation

#### 8.84.3.1 string ruleCG::get\_head () const

get rule head.

#### 8.84.3.2 double ruleCG::get\_weight () const

set rule compatibility.

#### 8.84.3.3 void ruleCG::set\_head (const string &)

set rule head.

#### 8.84.3.4 void ruleCG::set\_weight (double)

set rule compatibility.

### 8.84.4 Member Data Documentation

#### 8.84.4.1 string ruleCG::head [protected]

Head of the rule.

#### 8.84.4.2 double ruleCG::weight [protected]

compatibility value

The documentation for this class was generated from the following file:

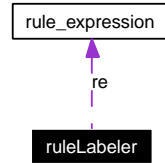
- **constraint\_grammar.h**

## 8.85 ruleLabeler Class Reference

ruleLabeler is an auxiliary class for the **depLabeler**(p. 99)

```
#include <dep_rules.h>
```

Collaboration diagram for ruleLabeler:



### Public Member Functions

- **ruleLabeler** (void)  
*Constructor.*
- **ruleLabeler** (const string &, **rule\_expression** \*)  
*Constructor.*
- **bool eval** (**dep\_tree::iterator**, **dep\_tree::iterator**) const  
*Evaluate rule conditions.*

### Public Attributes

- string **label**
- **rule\_expression** \* **re**
- string **ancestorLabel**

#### 8.85.1 Detailed Description

ruleLabeler is an auxiliary class for the **depLabeler**(p. 99)

#### 8.85.2 Constructor & Destructor Documentation

##### 8.85.2.1 ruleLabeler::ruleLabeler (void)

Constructor.

##### 8.85.2.2 ruleLabeler::ruleLabeler (const string &, rule\_expression \*)

Constructor.

### 8.85.3 Member Function Documentation

#### 8.85.3.1 `bool ruleLabeler::eval (dep_tree::iterator, dep_tree::iterator) const`

Evaluate rule conditions.

### 8.85.4 Member Data Documentation

#### 8.85.4.1 `string ruleLabeler::ancestorLabel`

#### 8.85.4.2 `string ruleLabeler::label`

#### 8.85.4.3 `rule_expression* ruleLabeler::re`

The documentation for this class was generated from the following files:

- `dep_rules.h`
- `dep_rules.cc`



## 8.86 senses Class Reference

Class senses implements a sense annotator.

```
#include <senses.h>
```

### Public Member Functions

- **senses** (const string &)  
*Constructor.*
- **~senses** ()  
*Destructor.*
- void **analyze** (list< sentence > &)  
*sense annotate selected analysis for each word in given sentences*

### Private Attributes

- Db **sensesdb**  
*C++ Interface to BerkeleyDB C API.*

#### 8.86.1 Detailed Description

Class senses implements a sense annotator.

#### 8.86.2 Constructor & Destructor Documentation

##### 8.86.2.1 senses::senses (const string &)

Constructor.

##### 8.86.2.2 senses::~~senses ()

Destructor.

#### 8.86.3 Member Function Documentation

##### 8.86.3.1 void senses::analyze (list< sentence > &)

sense annotate selected analysis for each word in given sentences

## 8.86.4 Member Data Documentation

### 8.86.4.1 Db senses::sensesdb [private]

C++ Interface to BerkeleyDB C API.

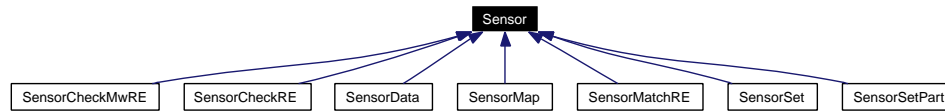
The documentation for this class was generated from the following files:

- **senses.h**
- **senses.cc**

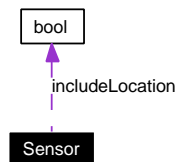
## 8.87 Sensor Class Reference

```
#include <sensor.h>
```

Inheritance diagram for Sensor:



Collaboration diagram for Sensor:



### Public Member Functions

- **Sensor** (string targ)
- virtual **~Sensor** ()
- **bool IncludeLocation** ()
- void **IncludeLocation** (bool val)
- **SensorType** **getSensorType** ()
- string **solve\_target** (const **sentence** &sent, int rec)
- virtual void **Extract** (const **sentence** &sent, set< string > &outSet, int rec\_OR\_start, int targLoc\_OR\_end)=0

### Protected Member Functions

- void **Output** (set< string > &outSet, string feat, int loc)

### Protected Attributes

- **SensorType** **sensorType**
- string **target**

### Private Member Functions

- string & **PostProcess** (string &feat, const char \*checkFeat)

### Private Attributes

- **bool** **includeLocation**

## 8.87.1 Constructor & Destructor Documentation

8.87.1.1 `Sensor::Sensor (string targ)` [inline]

8.87.1.2 `virtual Sensor::~~Sensor ()` [inline, virtual]

## 8.87.2 Member Function Documentation

8.87.2.1 `virtual void Sensor::Extract (const sentence & sent, set< string > & outSet, int rec_OR_start, int targLoc_OR_end)` [pure virtual]

Implemented in `SensorData` (p. 225), `SensorMap` (p. 226), `SensorSet` (p. 230), `SensorSetPart` (p. 232), `SensorMatchRE` (p. 228), `SensorCheckMwRE` (p. 221), and `SensorCheckRE` (p. 223).

8.87.2.2 `SensorType Sensor::getSensorType ()` [inline]

8.87.2.3 `void Sensor::IncludeLocation (bool val)` [inline]

8.87.2.4 `bool Sensor::IncludeLocation ()` [inline]

8.87.2.5 `void Sensor::Output (set< string > & outSet, string feat, int loc)`  
[protected]

8.87.2.6 `string& Sensor::PostProcess (string & feat, const char * checkFeat)`  
[private]

8.87.2.7 `string Sensor::solve_target (const sentence & sent, int rec)` [inline]

## 8.87.3 Member Data Documentation

8.87.3.1 `bool Sensor::includeLocation` [private]

8.87.3.2 `SensorType Sensor::sensorType` [protected]

8.87.3.3 `string Sensor::target` [protected]

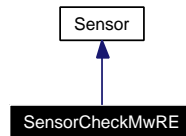
The documentation for this class was generated from the following file:

- `sensor.h`

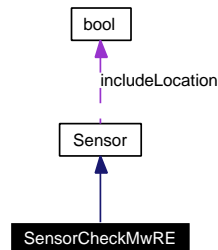
## 8.88 SensorCheckMwRE Class Reference

```
#include <sensor.h>
```

Inheritance diagram for SensorCheckMwRE:



Collaboration diagram for SensorCheckMwRE:



### Public Member Functions

- **SensorCheckMwRE** (char \*targ, char \*expr)
- void **Extract** (const **sentence** &sent, set< string > &outSet, int rec, int targLoc)

### Private Attributes

- list< string > mw\_patrons

### 8.88.1 Constructor & Destructor Documentation

**8.88.1.1** SensorCheckMwRE::SensorCheckMwRE (char \* *targ*, char \* *expr*)  
[inline]

### 8.88.2 Member Function Documentation

**8.88.2.1** void SensorCheckMwRE::Extract (const sentence & *sent*, set< string > & *outSet*, int *rec*, int *targLoc*) [virtual]

Implements **Sensor** (p. 220).

### 8.88.3 Member Data Documentation

**8.88.3.1** list<string> SensorCheckMwRE::mw\_patrons [private]

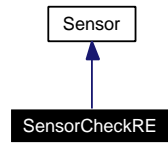
The documentation for this class was generated from the following file:

- `sensor.h`

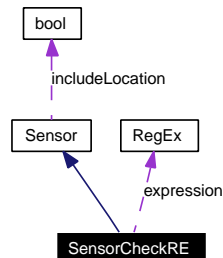
## 8.89 SensorCheckRE Class Reference

```
#include <sensor.h>
```

Inheritance diagram for SensorCheckRE:



Collaboration diagram for SensorCheckRE:



### Public Member Functions

- **SensorCheckRE** (char \*targ, char \*expr)
- void **Extract** (const **sentence** &sent, set< string > &outSet, int rec, int targLoc)

### Private Attributes

- **RegEx** expression

### 8.89.1 Constructor & Destructor Documentation

8.89.1.1 **SensorCheckRE::SensorCheckRE** (char \* *targ*, char \* *expr*) [inline]

### 8.89.2 Member Function Documentation

8.89.2.1 void **SensorCheckRE::Extract** (const **sentence** & *sent*, set< string > & *outSet*, int *rec*, int *targLoc*) [virtual]

Implements **Sensor** (p. 220).

### 8.89.3 Member Data Documentation

8.89.3.1 **RegEx** **SensorCheckRE::expression** [private]

The documentation for this class was generated from the following file:

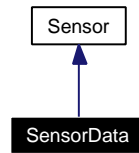
- `sensor.h`



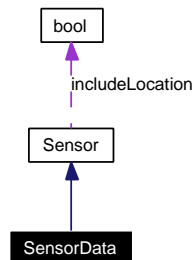
## 8.90 SensorData Class Reference

```
#include <sensor.h>
```

Inheritance diagram for SensorData:



Collaboration diagram for SensorData:



### Public Member Functions

- **SensorData** (string *targ*)
- void **Extract** (const **sentence** &*sent*, set< string > &*outSet*, int *rec*, int *targLoc*)

### 8.90.1 Constructor & Destructor Documentation

8.90.1.1 **SensorData::SensorData** (string *targ*) [inline]

### 8.90.2 Member Function Documentation

8.90.2.1 void **SensorData::Extract** (const **sentence** & *sent*, set< string > & *outSet*, int *rec*, int *targLoc*) [virtual]

Implements **Sensor** (p. 220).

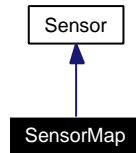
The documentation for this class was generated from the following file:

- **sensor.h**

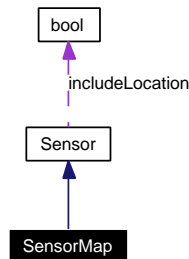
## 8.91 SensorMap Class Reference

```
#include <sensor.h>
```

Inheritance diagram for SensorMap:



Collaboration diagram for SensorMap:



### Public Member Functions

- **SensorMap** (char \*targ, char \*fname, char \*dpath)
- void **Extract** (const **sentence** &sent, set< string > &outSet, int rec, int targLoc)

### Private Attributes

- map< string, string > **content**

#### 8.91.1 Constructor & Destructor Documentation

**8.91.1.1** **SensorMap::SensorMap** (char \* *targ*, char \* *fname*, char \* *dpath*) [inline]

#### 8.91.2 Member Function Documentation

**8.91.2.1** void **SensorMap::Extract** (const **sentence** & *sent*, set< string > & *outSet*, int *rec*, int *targLoc*) [virtual]

Implements **Sensor** (p. 220).

#### 8.91.3 Member Data Documentation

**8.91.3.1** map<string,string> **SensorMap::content** [private]

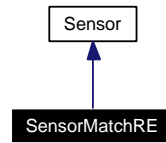
The documentation for this class was generated from the following file:

- `sensor.h`

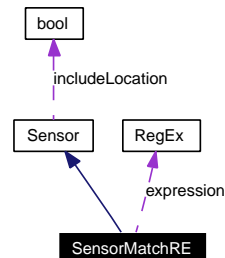
## 8.92 SensorMatchRE Class Reference

```
#include <sensor.h>
```

Inheritance diagram for SensorMatchRE:



Collaboration diagram for SensorMatchRE:



### Public Member Functions

- **SensorMatchRE** (`char *targ, char *expr`)
- `void Extract` (`const sentence &sent, set< string > &outSet, int rec, int targLoc`)

### Private Attributes

- **RegEx** `expression`

### 8.92.1 Constructor & Destructor Documentation

**8.92.1.1** `SensorMatchRE::SensorMatchRE (char * targ, char * expr)` [inline]

### 8.92.2 Member Function Documentation

**8.92.2.1** `void SensorMatchRE::Extract (const sentence & sent, set< string > & outSet, int rec, int targLoc)` [virtual]

Implements **Sensor** (p. 220).

### 8.92.3 Member Data Documentation

**8.92.3.1** `RegEx SensorMatchRE::expression` [private]

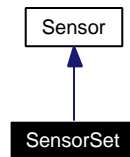
The documentation for this class was generated from the following file:

- `sensor.h`

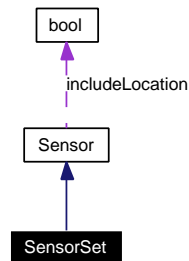
## 8.93 SensorSet Class Reference

```
#include <sensor.h>
```

Inheritance diagram for SensorSet:



Collaboration diagram for SensorSet:



### Public Member Functions

- **SensorSet** (char \*targ, char \*fname, char \*dpath)
- void **Extract** (const **sentence** &sent, set< string > &outSet, int rec, int targLoc)

### Private Attributes

- set< string > **content**

### 8.93.1 Constructor & Destructor Documentation

**8.93.1.1** **SensorSet::SensorSet** (char \* *targ*, char \* *fname*, char \* *dpath*) [inline]

### 8.93.2 Member Function Documentation

**8.93.2.1** void **SensorSet::Extract** (const *sentence* & *sent*, set< string > & *outSet*, int *rec*, int *targLoc*) [virtual]

Implements **Sensor** (p. 220).

### 8.93.3 Member Data Documentation

**8.93.3.1** set<string> **SensorSet::content** [private]

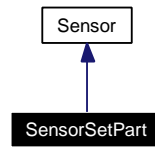
The documentation for this class was generated from the following file:

- `sensor.h`

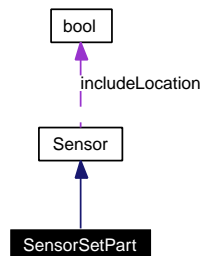
## 8.94 SensorSetPart Class Reference

```
#include <sensor.h>
```

Inheritance diagram for SensorSetPart:



Collaboration diagram for SensorSetPart:



### Public Member Functions

- **SensorSetPart** (char \*targ, char \*fname, char \*dpath)
- void **Extract** (const **sentence** &sent, set< string > &outSet, int rec, int targLoc)

### Private Attributes

- set< string > **content**

### 8.94.1 Constructor & Destructor Documentation

**8.94.1.1** **SensorSetPart::SensorSetPart** (char \* *targ*, char \* *fname*, char \* *dpath*)  
[inline]

### 8.94.2 Member Function Documentation

**8.94.2.1** void **SensorSetPart::Extract** (const **sentence** & *sent*, set< string > & *outSet*, int *rec*, int *targLoc*) [virtual]

Implements **Sensor** (p. 220).

### 8.94.3 Member Data Documentation

**8.94.3.1** set<string> **SensorSetPart::content** [private]

The documentation for this class was generated from the following file:



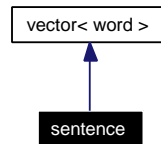
- `sensor.h`

## 8.95 sentence Class Reference

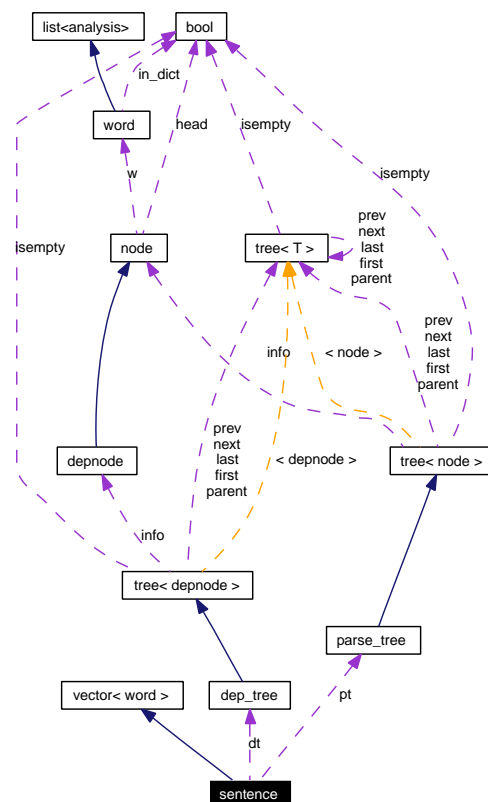
Class sentence is just a list of words that someone (the splitter) has validated it as a complete sentence.

```
#include <language.h>
```

Inheritance diagram for sentence:



Collaboration diagram for sentence:



### Public Member Functions

- `sentence ()`  
*Create a new sentence.*
- `void set_parse_tree (const parse_tree &)`  
*Set the parse tree.*

- **parse\_tree & get\_parse\_tree** (void)  
*Obtain the parse tree.*
- **bool is\_parsed** () const  
*Find out whether the sentence is parsed.*
- **dep\_tree & get\_dep\_tree** ()  
*Obtain the parse dependency tree.*
- **void set\_dep\_tree** (const **dep\_tree** &)  
*Set the dependency tree.*
- **bool is\_dep\_parsed** () const  
*Find out whether the sentence is dependency parsed.*
- **vector< word > get\_words** () const  
*get word list (useful for perl API)*
- **sentence::iterator words\_begin** (void)  
*get iterators to word list (useful for perl/java API)*
- **sentence::const\_iterator words\_begin** (void) const
- **sentence::iterator words\_end** (void)
- **sentence::const\_iterator words\_end** (void) const

## Private Attributes

- **parse\_tree pt**
- **dep\_tree dt**

### 8.95.1 Detailed Description

Class sentence is just a list of words that someone (the splitter) has validated it as a complete sentence.

It may include a parse tree.

### 8.95.2 Constructor & Destructor Documentation

#### 8.95.2.1 sentence::sentence ()

Create a new sentence.

It may include a parse tree.

### 8.95.3 Member Function Documentation

#### 8.95.3.1 dep\_tree & sentence::get\_dep\_tree ()

Obtain the parse dependency tree.

**8.95.3.2 parse\_tree & sentence::get\_parse\_tree (void)**

Obtain the parse tree.

**8.95.3.3 vector< word > sentence::get\_words () const**

get word list (useful for perl API)

**8.95.3.4 bool sentence::is\_dep\_parsed () const**

Find out whether the sentence is dependency parsed.

**8.95.3.5 bool sentence::is\_parsed () const**

Find out whether the sentence is parsed.

**8.95.3.6 void sentence::set\_dep\_tree (const dep\_tree &)**

Set the dependency tree.

**8.95.3.7 void sentence::set\_parse\_tree (const parse\_tree &)**

Set the parse tree.

**8.95.3.8 sentence::const\_iterator sentence::words\_begin (void) const****8.95.3.9 sentence::iterator sentence::words\_begin (void)**

get iterators to word list (useful for perl/java API)

**8.95.3.10 sentence::const\_iterator sentence::words\_end (void) const****8.95.3.11 sentence::iterator sentence::words\_end (void)****8.95.4 Member Data Documentation****8.95.4.1 dep\_tree sentence::dt [private]****8.95.4.2 parse\_tree sentence::pt [private]**

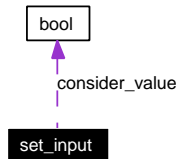
The documentation for this class was generated from the following files:

- language.h
- language.cc

## 8.96 set\_input Class Reference

```
#include <example.h>
```

Collaboration diagram for set\_input:



### Public Types

- typedef set< iFeature >::const\_iterator const\_iterator

### Public Member Functions

- set\_input ()
- set\_input (string)
- set\_input (char \*s)
- set\_input (const set\_input &i1)
- set\_input (double f1, const set\_input &i1, double f2, const set\_input &i2)
- ~set\_input ()
- void add\_feature (const iFeature &f)
- void add\_feature (int l, double v=1.0)
- void multiply\_by\_factor (double f)
- void normalize ()
- void parse\_string (string)
- double feature\_value (int label) const
- int size () const
- int dimension () const
- double inner\_product (set\_input \*) const
- double norm () const
- const\_iterator begin () const
- const\_iterator end () const
- void print (ostream &o) const
- void write (ostream &o) const

### Static Public Member Functions

- static void set\_consider\_value (bool)
- static void set\_label\_value\_separator (char)

### Private Attributes

- set< iFeature > sf
- int \_dimension

- Generated on Wed Apr 26 12:55:30 2006 for FreeLing by Doxygen

## 8.96.1 Member Typedef Documentation

8.96.1.1 `typedef set<iFeature>::const_iterator set_input::const_iterator`

## 8.96.2 Constructor & Destructor Documentation

8.96.2.1 `set_input::set_input () [inline]`

8.96.2.2 `set_input::set_input (string)`

8.96.2.3 `set_input::set_input (char * s) [inline]`

8.96.2.4 `set_input::set_input (const set_input & i1)`

8.96.2.5 `set_input::set_input (double f1, const set_input & i1, double f2, const set_input & i2)`

8.96.2.6 `set_input::~~set_input ()`

## 8.96.3 Member Function Documentation

8.96.3.1 `void set_input::add_feature (int l, double v = 1.0)`

8.96.3.2 `void set_input::add_feature (const iFeature & f)`

8.96.3.3 `const_iterator set_input::begin () const [inline]`

8.96.3.4 `int set_input::dimension () const [inline]`

8.96.3.5 `const_iterator set_input::end () const [inline]`

8.96.3.6 `double set_input::feature_value (int label) const`

8.96.3.7 `double set_input::inner_product (set_input *) const`

8.96.3.8 `void set_input::multiply_by_factor (double f)`

8.96.3.9 `double set_input::norm () const`

8.96.3.10 `void set_input::normalize ()`

8.96.3.11 `void set_input::parse_string (string)`

8.96.3.12 `void set_input::print (ostream & o) const`

8.96.3.13 `static void set_input::set_consider_value (bool) [static]`

8.96.3.14 `static void set_input::set_label_value_separator (char) [static]`

8.96.3.15 `int set_input::size () const [inline]`

8.96.3.16 `void set_input::write (ostream & o) const`

## 8.96.4 Member Data Documentation

---

Generated on Wed Apr 26 12:55:30 2006 for FreeLing by Doxygen

8.96.4.1 `int set_input::_dimension [private]`

8.96.4.2 `bool set_input::consider_value [static, private]`

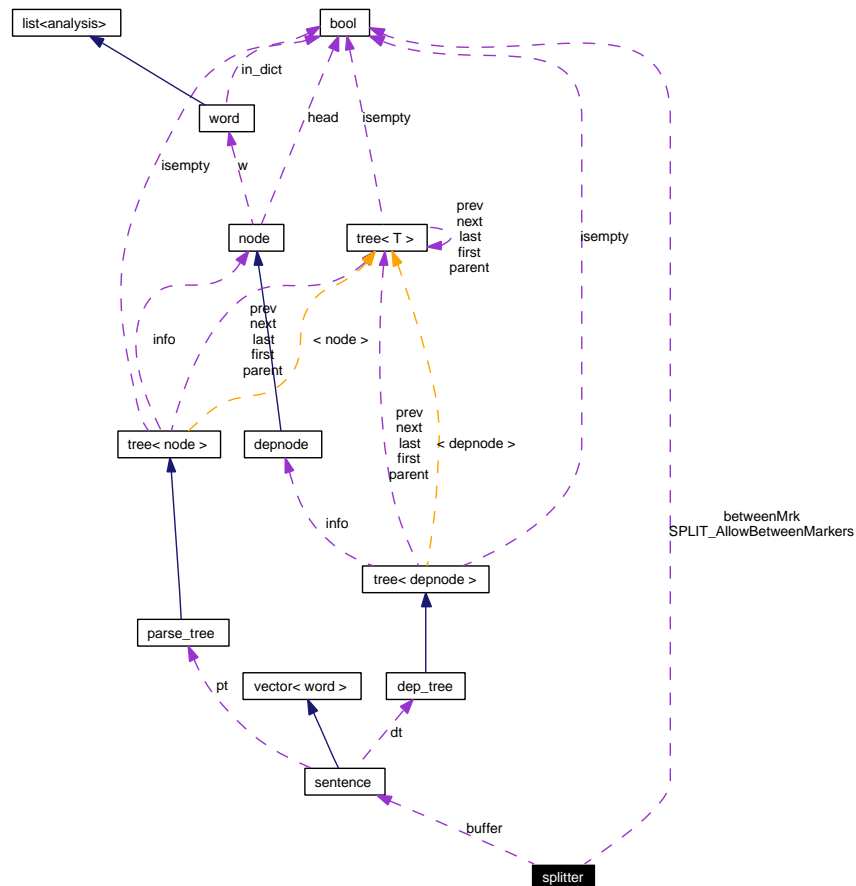
8.96.4.3 `char set_input::label_value_separator [static, private]`

- `example.h`



Class splitter implements a sentence splitter, which accumulates lists of words until a sentence is completed, and then returns a list of sentence objects.

Collaboration diagram for splitter:



- **splitter** (const string &)  
*Constructor.*
- **list< sentence > split** (const **list< word >** &, **bool**)  
*split sentences with default options*

- `bool end_of_sentence (list< word >::const_iterator, const list< word > &) const`  
*check for sentence markers*

## Private Attributes

- **bool SPLIT\_AllowBetweenMarkers**

*configuration options*

- **int SPLIT\_MaxLines**
- **set< string > starters**

*Sentence delimiters.*

- **map< string, bool > enders**
- **map< string, int > markers**

*Open-close marker pairs (parenthesis, etc).*

- **bool betweenMrk**
- **int no\_split\_count**
- **int mark\_type**
- **list< sentence > ls**

*accumulated list of returned sentences*

- **sentence buffer**

*accumulated words of current sentence*

### 8.97.1 Detailed Description

Class splitter implements a sentence splitter, which accumulates lists of words until a sentence is completed, and then returns a list of sentence objects.

### 8.97.2 Constructor & Destructor Documentation

#### 8.97.2.1 splitter::splitter (const string &)

Constructor.

### 8.97.3 Member Function Documentation

#### 8.97.3.1 bool splitter::end\_of\_sentence (list< word >::const\_iterator, const list< word > &) const [private]

check for sentence markers

#### 8.97.3.2 list< sentence > splitter::split (const list< word > & v, bool flush)

split sentences with default options

If a sentence marker is reached (or flush flag is set), return all sentences currently in buffer, and clean buffer. If a new sentence is started but not completed, keep in buffer, and wait for further calls with more data.

## 8.97.4 Member Data Documentation

**8.97.4.1** `bool splitter::betweenMrk` [private]

**8.97.4.2** `sentence splitter::buffer` [private]

accumulated words of current sentence

**8.97.4.3** `map<string,bool> splitter::enders` [private]

**8.97.4.4** `list<sentence> splitter::ls` [private]

accumulated list of returned sentences

**8.97.4.5** `int splitter::mark_type` [private]

**8.97.4.6** `map<string,int> splitter::markers` [private]

Open-close marker pairs (parenthesis, etc).

**8.97.4.7** `int splitter::no_split_count` [private]

**8.97.4.8** `bool splitter::SPLIT_AllowBetweenMarkers` [private]

configuration options

**8.97.4.9** `int splitter::SPLIT_MaxLines` [private]

**8.97.4.10** `set<string> splitter::starters` [private]

Sentence delimiters.

The documentation for this class was generated from the following files:

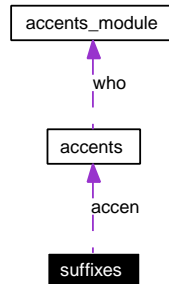
- `splitter.h`
- `splitter.cc`

## 8.98 suffixes Class Reference

Class suffixes implements suffixation rules and dictionary search for suffixed word forms.

```
#include <suffixes.h>
```

Collaboration diagram for suffixes:



### Public Member Functions

- **suffixes** (const **maco\_options** &)  
*Constructor.*
- void **look\_for\_suffixes** (**word** &, **dictionary** &) const  
*look up possible roots of a suffixed form*

### Private Member Functions

- void **look\_for\_suffixes\_in\_list** (const **multimap**< string, **sufrule** > &, **word** &, **dictionary** &) const  
*auxiliary methods to deal with suffixing*
- **vector**< string > **GenerateRoots** (const **sufrule** &, const string &) const  
*auxiliary methods to deal with suffixing*
- void **SearchRootsList** (const **vector**< string > &, const **sufrule** &, **word** &, **dictionary** &) const  
*auxiliary methods to deal with suffixing*
- void **CheckRetokenizable** (const **sufrule** &, const string &, const string &, const string &, **dictionary** &, **list**< **word** > &) const  
*auxiliary method to deal with retokenization*

### Private Attributes

- **accents accen**  
*Language-specific accent handler.*

- **multimap**< string, **sufrule** > **suffix**  
*all suffixation rules*
- **multimap**< string, **sufrule** > **suffix\_always**  
*suffixation rules applied unconditionally*
- **set**< unsigned int > **ExistingLength**  
*array of existing suffix lengths.*
- unsigned int **LongestSuf**  
*Length of longest suffix.*

### 8.98.1 Detailed Description

Class suffixes implements suffixation rules and dictionary search for suffixed word forms.

### 8.98.2 Constructor & Destructor Documentation

#### 8.98.2.1 **suffixes::suffixes** (const **maco\_options** &)

Constructor.

### 8.98.3 Member Function Documentation

#### 8.98.3.1 **void suffixes::CheckRetokenizable** (const **sufrule** &, const string &, const string &, const string &, dictionary &, list< word > &) const [private]

auxiliary method to deal with retokenization

#### 8.98.3.2 **vector**< string > **suffixes::GenerateRoots** (const **sufrule** &, const string &) const [private]

auxiliary methods to deal with suffixing

#### 8.98.3.3 **void suffixes::look\_for\_suffixes** (word & *w*, dictionary & *dic*) const

look up possible roots of a suffixed form

Words already analyzed are only applied the "always"-marked suffix rules. So-far unrecognized words, are applied all the suffix rules.

#### 8.98.3.4 **void suffixes::look\_for\_suffixes\_in\_list** (const **multimap**< string, **sufrule** > &, word &, dictionary &) const [private]

auxiliary methods to deal with suffixing

**8.98.3.5** `void suffixes::SearchRootsList (const vector< string > &, const sufrule &, word &, dictionary &) const` [private]

auxiliary methods to deal with suffixing

## 8.98.4 Member Data Documentation

**8.98.4.1** `accents suffixes::accen` [private]

Language-specific accent handler.

**8.98.4.2** `set<unsigned int> suffixes::ExistingLength` [private]

array of existing suffix lengths.

**8.98.4.3** `unsigned int suffixes::LongestSuf` [private]

Length of longest suffix.

**8.98.4.4** `multimap<string,sufrule> suffixes::suffix` [private]

all suffixation rules

**8.98.4.5** `multimap<string,sufrule> suffixes::suffix_always` [private]

suffixation rules applied unconditionally

The documentation for this class was generated from the following files:

- `suffixes.h`
- `suffixes.cc`

## 8.99 sufrule Class Reference

Class sufrule contains a suffixation rule, and is used by class suffixes.

```
#include <sufrule.h>
```

### Public Attributes

- string **term**
- string **cond**
- string **output**
- string **retok**
- int **last\_acc**
- int **not\_acc**
- int **lem\_form**
- int **enc**
- int **always**

### 8.99.1 Detailed Description

Class sufrule contains a suffixation rule, and is used by class suffixes.

### 8.99.2 Member Data Documentation

**8.99.2.1** int sufrule::always

**8.99.2.2** string sufrule::cond

**8.99.2.3** int sufrule::enc

**8.99.2.4** int sufrule::last\_acc

**8.99.2.5** int sufrule::lem\_form

**8.99.2.6** int sufrule::not\_acc

**8.99.2.7** string sufrule::output

**8.99.2.8** string sufrule::retok

**8.99.2.9** string sufrule::term

The documentation for this class was generated from the following file:

- **sufrule.h**

## 8.100 tokenizer Class Reference

Class tokenizer implements a token splitter, which converts a string into a sequence of word objects, according to a set of tokenization rules read from aconfiguration file.

```
#include <tokenizer.h>
```

### Public Member Functions

- **tokenizer** (const string &)  
*Constructor.*
- **list< word > tokenize** (const string &)  
*tokenize string with default options*
- **list< word > tokenize** (const string &, int &)  
*tokenize string with default options, tracking offset*

### Private Attributes

- **set< string > abrevs**  
*abbreviations set (Dr. Mrs. etc. period is not separated)*
- **vector< pair< string, RegEx > > rules**  
*tokenization rules*
- **map< string, int > matches**  
*substrings to convert into tokens in each rule*

### 8.100.1 Detailed Description

Class tokenizer implements a token splitter, which converts a string into a sequence of word objects, according to a set of tokenization rules read from aconfiguration file.

### 8.100.2 Constructor & Destructor Documentation

#### 8.100.2.1 tokenizer::tokenizer (const string &)

Constructor.

### 8.100.3 Member Function Documentation

#### 8.100.3.1 list< word > tokenizer::tokenize (const string &, int &)

tokenize string with default options, tracking offset



**8.100.3.2** `list< word > tokenizer::tokenize (const string &)`

tokenize string with default options

**8.100.4** Member Data Documentation**8.100.4.1** `set<string> tokenizer::abrevs` [private]

abbreviations set (Dr. Mrs. etc. period is not separated)

**8.100.4.2** `map<string,int> tokenizer::matches` [private]

substrings to convert into tokens in each rule

**8.100.4.3** `vector<pair<string,Regex> > tokenizer::rules` [private]

tokenization rules

The documentation for this class was generated from the following files:

- `tokenizer.h`
- `tokenizer.cc`

## 8.101 traces Class Reference

Class traces implements trace and error handling utilities.

```
#include <traces.h>
```

### Static Public Member Functions

- static void **error\_\_crash** (const string &, const string &, unsigned long)  
*static trace methods definition. Inlined for efficiency*
- static void **warning** (const string &, const string &, unsigned long)
- static void **trace** (int, const string &, const string &, unsigned long)
- static void **trace\_\_word** (int lv, const **word** &, const string &, unsigned long)
- static void **trace\_\_word\_list** (int, const **list**< **word** > &, const string &, unsigned long)
- static void **trace\_\_sentence** (int, const **sentence** &, const string &, unsigned long)
- static void **trace\_\_sentence\_list** (int, const **list**< **sentence** > &, const string &, unsigned long)

### Static Public Attributes

- static int **TraceLevel** = 0  
*Static data members.*
- static unsigned long **TraceModule** = 0x0000000

#### 8.101.1 Detailed Description

Class traces implements trace and error handling utilities.

#### 8.101.2 Member Function Documentation

- 8.101.2.1 void traces::error\_\_crash (const string &, const string &, unsigned long)**  
[inline, static]

static trace methods definition. Inlined for efficiency

- 8.101.2.2 `void traces::trace (int, const string &, const string &, unsigned long)`  
[inline, static]
- 8.101.2.3 `void traces::trace_sentence (int, const sentence &, const string &, unsigned long)` [inline, static]
- 8.101.2.4 `void traces::trace_sentence_list (int, const list< sentence > &, const string &, unsigned long)` [inline, static]
- 8.101.2.5 `void traces::trace_word (int lv, const word &, const string &, unsigned long)` [inline, static]
- 8.101.2.6 `void traces::trace_word_list (int, const list< word > &, const string &, unsigned long)` [inline, static]
- 8.101.2.7 `void traces::warning (const string &, const string &, unsigned long)`  
[inline, static]

### 8.101.3 Member Data Documentation

- 8.101.3.1 `int traces::TraceLevel = 0` [static]

Static data members.

They are global variables, declared here in a separate module so it is linked only once.

- 8.101.3.2 `unsigned long traces::TraceModule = 0x0000000` [static]

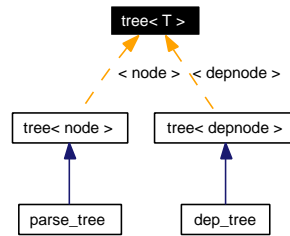
The documentation for this class was generated from the following files:

- `traces.h`
- `traces.cc`

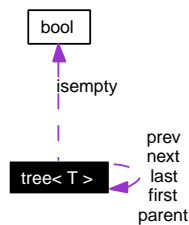
## 8.102 tree< T > Class Template Reference

```
#include <tree.h>
```

Inheritance diagram for tree< T >:



Collaboration diagram for tree< T >:



### Public Types

- typedef `preorder_iterator` `iterator`

### Public Member Functions

- `tree ()`  
*constructor: empty tree doesn't exist, it's a one-node tree with no info. Be careful*
- `tree (const T &)`  
*constructor: one-node tree*
- `tree (const tree< T > &)`  
*copy constructor*
- `tree (const preorder_iterator &)`
- `~tree ()`  
*destructor*
- `tree< T > & operator= (const tree< T > &)`  
*assignment*
- `unsigned int num_children () const`  
*number of children*

- **sibling\_iterator nth\_child** (unsigned int) const  
*access nth child*
- void **append\_child** (const tree< T > &)  
*append child to a tree*
- void **hang\_child** (tree< T > &)  
*hang a tree as child of another (no copying!!)*
- void **clear** ()
- **bool empty** () const  
*detect empty tree*
- **sibling\_iterator sibling\_begin** ()  
*begin/end sibling iterator*
- **sibling\_iterator sibling\_end** () const
- **preorder\_iterator begin** ()  
*begin/end preorder iterator*
- **preorder\_iterator end** () const

## Public Attributes

- T info

## Private Member Functions

- void **clone** (const tree< T > &)  
*clone an entire tree*

## Private Attributes

- **bool isempty**
- **tree \* parent**
- **tree \* first**
- **tree \* last**
- **tree \* prev**
- **tree \* next**

## Classes

- class **generic\_iterator**
- class **preorder\_iterator**  
*traverse the tree in preorder (parent first, then children)*
- class **sibling\_iterator**  
*traverse all children of the same node*

```
template<class T> class tree< T >
```

### 8.102.1 Member Typedef Documentation

8.102.1.1 `template<class T> typedef preorder_iterator tree< T >::iterator`

### 8.102.2 Constructor & Destructor Documentation

8.102.2.1 `template<class T> tree< T >::tree ()`

constructor: empty tree doesn't exist, it's a one-node tree with no info. Be careful

8.102.2.2 `template<class T> tree< T >::tree (const T &)`

constructor: one-node tree

8.102.2.3 `template<class T> tree< T >::tree (const tree< T > &)`

copy constructor

8.102.2.4 `template<class T> tree< T >::tree (const preorder_iterator &)`

8.102.2.5 `template<class T> tree< T >::~~tree ()`

destructor

### 8.102.3 Member Function Documentation

8.102.3.1 `template<class T> void tree< T >::append_child (const tree< T > &)`

append child to a tree

8.102.3.2 `template<class T> tree< T >::preorder_iterator tree< T >::begin ()`

begin/end preorder iterator

8.102.3.3 `template<class T> void tree< T >::clear ()`

8.102.3.4 `template<class T> void tree< T >::clone (const tree< T > &) [private]`

clone an entire tree

8.102.3.5 `template<class T> bool tree< T >::empty () const`

detect empty tree

**8.102.3.6** `template<class T> tree< T >::preorder_iterator tree< T >::end () const`

**8.102.3.7** `template<class T> void tree< T >::hang_child (tree< T > &)`

hang a tree as child of another (no copying!!)

**8.102.3.8** `template<class T> tree< T >::sibling_iterator tree< T >::nth_child  
(unsigned int) const`

access nth child

**8.102.3.9** `template<class T> unsigned int tree< T >::num_children () const`

number of children

**8.102.3.10** `template<class T> tree< T > & tree< T >::operator= (const tree< T >  
&)`

assignment

**8.102.3.11** `template<class T> tree< T >::sibling_iterator tree< T >::sibling_begin  
()`

begin/end sibling iterator

**8.102.3.12** `template<class T> tree< T >::sibling_iterator tree< T >::sibling_end  
() const`

## 8.102.4 Member Data Documentation

**8.102.4.1** `template<class T> tree* tree< T >::first [private]`

**8.102.4.2** `template<class T> T tree< T >::info`

**8.102.4.3** `template<class T> bool tree< T >::isempty [private]`

**8.102.4.4** `template<class T> tree * tree< T >::last [private]`

**8.102.4.5** `template<class T> tree * tree< T >::next [private]`

**8.102.4.6** `template<class T> tree* tree< T >::parent [private]`

**8.102.4.7** `template<class T> tree* tree< T >::prev [private]`

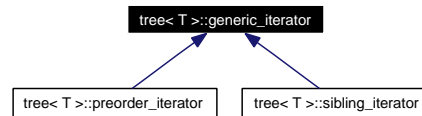
The documentation for this class was generated from the following file:

- tree.h

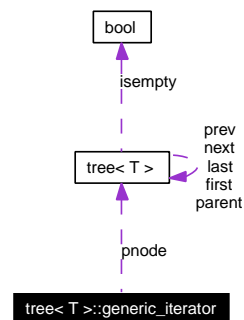
## 8.103 `tree< T >::generic_iterator` Class Reference

```
#include <tree.h>
```

Inheritance diagram for `tree< T >::generic_iterator`:



Collaboration diagram for `tree< T >::generic_iterator`:



### Public Member Functions

- `generic_iterator ()`
- `generic_iterator (tree *)`
- `tree< T > & operator * () const`
- `tree< T > * operator → () const`
- `bool operator== (const generic_iterator &) const`
- `bool operator!= (const generic_iterator &) const`

### Protected Attributes

- `tree * pnode`



```
template<class T> class tree< T >::generic_iterator
```

### 8.103.1 Constructor & Destructor Documentation

8.103.1.1 `template<class T> tree< T >::generic_iterator::generic_iterator ()`

8.103.1.2 `template<class T> tree< T >::generic_iterator::generic_iterator (tree *)`

### 8.103.2 Member Function Documentation

8.103.2.1 `template<class T> tree< T > & tree< T >::generic_iterator::operator *  
() const`

8.103.2.2 `template<class T> bool tree< T >::generic_iterator::operator!= (const  
generic_iterator &) const`

8.103.2.3 `template<class T> tree< T > * tree< T >::generic_iterator::operator →  
() const`

8.103.2.4 `template<class T> bool tree< T >::generic_iterator::operator== (const  
generic_iterator &) const`

### 8.103.3 Member Data Documentation

8.103.3.1 `template<class T> tree* tree< T >::generic_iterator::pnode [protected]`

The documentation for this class was generated from the following file:

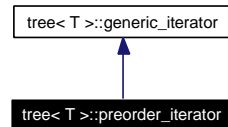
- tree.h

## 8.104 `tree< T >::preorder_iterator` Class Reference

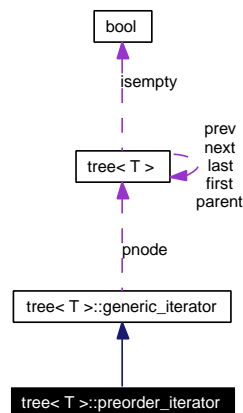
traverse the tree in preorder (parent first, then children)

```
#include <tree.h>
```

Inheritance diagram for `tree< T >::preorder_iterator`:



Collaboration diagram for `tree< T >::preorder_iterator`:



### Public Member Functions

- `preorder_iterator ()`
- `preorder_iterator (tree *)`
- `preorder_iterator (sibling_iterator &)`
- `preorder_iterator & operator++ ()`
- `preorder_iterator & operator-- ()`
- `preorder_iterator & operator+= (unsigned int)`
- `preorder_iterator & operator-= (unsigned int)`

### 8.104.1 Detailed Description

```
template<class T> class tree< T >::preorder_iterator
```

traverse the tree in preorder (parent first, then children)

## 8.104.2 Constructor & Destructor Documentation

8.104.2.1 `template<class T> tree< T >::preorder_iterator::preorder_iterator ()`

8.104.2.2 `template<class T> tree< T >::preorder_iterator::preorder_iterator (tree *)`

8.104.2.3 `template<class T> tree< T >::preorder_iterator::preorder_iterator (sibling_iterator &)`

## 8.104.3 Member Function Documentation

8.104.3.1 `template<class T> tree< T >::preorder_iterator & tree< T >::preorder_iterator::operator++ ()`

8.104.3.2 `template<class T> tree< T >::preorder_iterator & tree< T >::preorder_iterator::operator+= (unsigned int)`

8.104.3.3 `template<class T> tree< T >::preorder_iterator & tree< T >::preorder_iterator::operator- ()`

8.104.3.4 `template<class T> tree< T >::preorder_iterator & tree< T >::preorder_iterator::operator-= (unsigned int)`

The documentation for this class was generated from the following file:

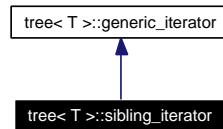
- `tree.h`

## 8.105 tree< T >::sibling\_iterator Class Reference

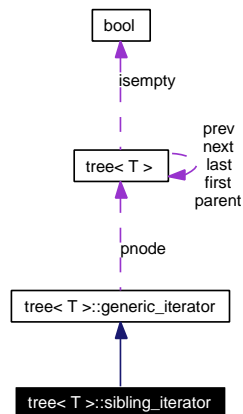
traverse all children of the same node

```
#include <tree.h>
```

Inheritance diagram for tree< T >::sibling\_iterator:



Collaboration diagram for tree< T >::sibling\_iterator:



### Public Member Functions

- `sibling_iterator ()`

*Method implementations for class sibling\_iterator (p. 260).*

- `sibling_iterator (tree *)`
- `sibling_iterator & operator++ ()`
- `sibling_iterator & operator-- ()`
- `sibling_iterator & operator+=( unsigned int )`
- `sibling_iterator & operator-=( unsigned int )`

### Friends

- `class preorder_iterator`

#### 8.105.1 Detailed Description

```
template<class T> class tree< T >::sibling_iterator
```

traverse all children of the same node

## 8.105.2 Constructor & Destructor Documentation

### 8.105.2.1 `template<class T> tree< T >::sibling_iterator::sibling_iterator ()`

Method implementations for class `sibling_iterator`(p. 260).

### 8.105.2.2 `template<class T> tree< T >::sibling_iterator::sibling_iterator (tree *)`

## 8.105.3 Member Function Documentation

### 8.105.3.1 `template<class T> tree< T >::sibling_iterator & tree< T >::sibling_iterator::operator++ ()`

### 8.105.3.2 `template<class T> tree< T >::sibling_iterator & tree< T >::sibling_iterator::operator+= (unsigned int)`

### 8.105.3.3 `template<class T> tree< T >::sibling_iterator & tree< T >::sibling_iterator::operator- ()`

### 8.105.3.4 `template<class T> tree< T >::sibling_iterator & tree< T >::sibling_iterator::operator-= (unsigned int)`

## 8.105.4 Friends And Related Function Documentation

### 8.105.4.1 `template<class T> friend class preorder_iterator [friend]`

The documentation for this class was generated from the following file:

- `tree.h`

## 8.106 util Class Reference

Class util implements some utilities for NLP analyzers: "tolower" for latin alphabets, parole tags manipulation, string2number and viceversa conversions, etc.

```
#include <util.h>
```

### Static Public Member Functions

- static string **lowercase** (const string &)  
*Lowercase a string, even with latin characters.*
- static bool **isalphanum** (const char)  
*find out whether a char is alphanumeric, even latin characters*
- static bool **has\_\_lowercase** (const string &)  
*find out whether a string contains a lowercase char, even latin characters*
- static int **string2int** (const string &)  
*conversion utilities*
- static string **int2string** (const int)  
*Type conversion.*
- static double **string2double** (const string &)  
*Type conversion.*
- static string **double2string** (const double)  
*Type conversion.*
- static long double **string2longdouble** (const string &)  
*Type conversion.*
- static string **longdouble2string** (const long double)  
*Type conversion.*
- static string **list2string** (const list< string > &, const char)  
*Create a single string concatenating all strings in given list with given separator char.*
- static list< string > **string2list** (const string &, const char)  
*Split a string into a list of strings given a separator char.*

### 8.106.1 Detailed Description

Class util implements some utilities for NLP analyzers: "tolower" for latin alphabets, parole tags manipulation, string2number and viceversa conversions, etc.

## 8.106.2 Member Function Documentation

### 8.106.2.1 string util::double2string (const *double*) [static]

Type conversion.

### 8.106.2.2 bool util::has\_lowercase (const string &) [static]

find out whether a string contains a lowercase char, even latin characters

### 8.106.2.3 string util::int2string (const *int*) [static]

Type conversion.

### 8.106.2.4 bool util::isalphanum (const *char*) [static]

find out whether a char is alphanumeric, even latin characters

### 8.106.2.5 string util::list2string (const list< string > &, const *char*) [static]

Create a single string concatenating all strings in given list with given separator char.

### 8.106.2.6 string util::longdouble2string (const long *double*) [static]

Type conversion.

### 8.106.2.7 string util::lowercase (const string &) [static]

Lowercase a string, even with latin characters.

### 8.106.2.8 double util::string2double (const string &) [static]

Type conversion.

### 8.106.2.9 int util::string2int (const string &) [static]

conversion utilities

### 8.106.2.10 list< string > util::string2list (const string &, const *char*) [static]

Split a string into a list of strings given a separator char.

### 8.106.2.11 long double util::string2longdouble (const string &) [static]

Type conversion.

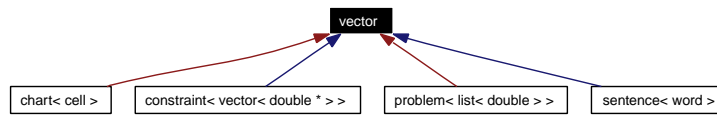
The documentation for this class was generated from the following files:

- `util.h`
- `util.cc`



## 8.107 vector Class Reference

Inheritance diagram for vector:



The documentation for this class was generated from the following file:

- **relax.h**

## 8.108 viterbi Class Reference

The class viterbi stores the two maps for each observation: The delta map stores the maximum probability for each state in that observation, the phi map stores the backpath to maximize the probability.

```
#include <hmm_tagger.h>
```

### Public Member Functions

- **viterbi** (int)  
*Constructor.*
- **~viterbi** ()  
*Destructor.*

### Public Attributes

- **map< string, double > \* delta\_log**  
*Space for delta tables used in Viterbi algorithm.*
- **map< string, string > \* phi**  
*Space for phi tables used in Viterbi algorithm.*

### 8.108.1 Detailed Description

The class viterbi stores the two maps for each observation: The delta map stores the maximum probability for each state in that observation, the phi map stores the backpath to maximize the probability.

An instance of this class is created for each sentence to be tagged, and destroyed when work is finished.

### 8.108.2 Constructor & Destructor Documentation

#### 8.108.2.1 viterbi::viterbi (int)

Constructor.

#### 8.108.2.2 viterbi::~viterbi ()

Destructor.

### 8.108.3 Member Data Documentation

#### 8.108.3.1 map<string, double>\* viterbi::delta\_log

Space for delta tables used in Viterbi algorithm.

### 8.108.3.2 map<string, string>\* viterbi::phi

Space for phi tables used in Viterbi algorithm.

The documentation for this class was generated from the following files:

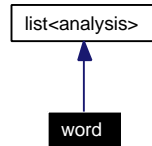
- `hmm_tagger.h`
- `hmm_tagger.cc`

## 8.109 word Class Reference

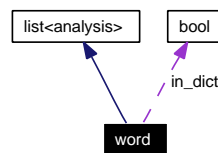
Class word stores all info related to a word: form, list of analysis, list of tokens (if multiword).

```
#include <language.h>
```

Inheritance diagram for word:



Collaboration diagram for word:



### Public Member Functions

- **word** ()  
*constructor*
- **word** (const string &)  
*constructor*
- **word** (const string &, const **list**< **word** > &)  
*constructor*
- **word** (const string &, const **list**< **analysis** > &, const **list**< **word** > &)  
*constructor*
- **word** (const **word** &)  
*Copy constructor.*
- **word** & **operator=** (const **word** &)  
*assignment*
- **bool** **is\_\_ambiguous** (void) const  
*true iff the word has more than one analysis*
- **bool** **is\_\_multiword** (void) const  
*true iff the word is a multiword compound*
- **int** **get\_\_n\_\_words\_\_mw** (void) const  
*get number of words in compound*

- **list< word > get\_words\_mw** (void) const  
*get word objects that compound the multiword*
- **string get\_form** (void) const  
*get word form*
- **analysis get\_selected\_analysis** (void) const  
*Get the selected analysis.*
- **word::iterator selected\_analysis** (void) const  
*Get an iterator to the selected analysis.*
- **string get\_lemma** (void) const  
*get lemma for the selected analysis in list*
- **string get\_parole** (void) const  
*get parole for the selected analysis*
- **string get\_short\_parole** (const string &) const  
*get parole (short version) for the selected analysis*
- **list< string > get\_senses** (void) const  
*get sense list for the selected analysis*
- **void set\_senses** (const list< string > &)  
*set sense list for the selected analysis*
- **unsigned int get\_span\_start** (void) const  
*get token span.*
- **unsigned int get\_span\_finish** (void) const
- **bool found\_in\_dict** (void) const  
*get in\_dict*
- **void set\_found\_in\_dict** (bool)  
*set in\_dict*
- **void add\_analysis** (const analysis &)  
*add one analysis to current analysis list (no duplicate check!)*
- **void set\_analysis** (const analysis &)  
*set analysis list to one single analysis, overwriting current values*
- **void set\_analysis** (const list< analysis > &)  
*set analysis list, overwriting current values*
- **void set\_form** (const string &)  
*set word form*

- void **set\_\_span** (unsigned int, unsigned int)  
*set token span*
- void **set\_\_user\_\_data** (void \*)  
*set user data*
- int **get\_\_n\_\_analysis** (void) const  
*get number of analysis in current list*
- void **copy\_\_analysis** (const **word** &)  
*copy analysis list*
- void **select\_\_analysis** (word::iterator)  
*mark the given analysis as the selected one.*
- **list< analysis > get\_\_analysis** (void) const  
*get list of analysis (useful for perl API)*
- word::iterator **analysis\_\_begin** (void)  
*get begin iterator to analysis list (useful for perl/java API)*
- word::const\_iterator **analysis\_\_begin** (void) const
- word::iterator **analysis\_\_end** (void)  
*get end iterator to analysis list (useful for perl/java API)*
- word::const\_iterator **analysis\_\_end** (void) const

## Private Attributes

- string **form**  
*lexical form*
- word::iterator **selected**  
*selected analysis (if any)*
- **list< word > multiword**  
*empty list if not a multiword*
- unsigned int **start**  
*token span*
- unsigned int **finish**
- bool **in\_\_dict**  
*word form found in dictionary*
- void \* **user**  
*Private data.*

### 8.109.1 Detailed Description

Class word stores all info related to a word: form, list of analysis, list of tokens (if multiword).

### 8.109.2 Constructor & Destructor Documentation

#### 8.109.2.1 word::word ()

constructor

#### 8.109.2.2 word::word (const string &)

constructor

#### 8.109.2.3 word::word (const string &, const list< word > &)

constructor

#### 8.109.2.4 word::word (const string &, const list< analysis > &, const list< word > &)

constructor

#### 8.109.2.5 word::word (const word &)

Copy constructor.

### 8.109.3 Member Function Documentation

#### 8.109.3.1 void word::add\_\_analysis (const analysis &)

add one analysis to current analysis list (no duplicate check!)

#### 8.109.3.2 word::const\_iterator word::analysis\_\_begin (void) const

#### 8.109.3.3 word::iterator word::analysis\_\_begin (void)

get begin iterator to analysis list (useful for perl/java API)

#### 8.109.3.4 word::const\_iterator word::analysis\_\_end (void) const

#### 8.109.3.5 word::iterator word::analysis\_\_end (void)

get end iterator to analysis list (useful for perl/java API)

#### 8.109.3.6 void word::copy\_\_analysis (const word &)

copy analysis list

**8.109.3.7 bool word::found\_in\_dict (void) const**

get in\_dict

**8.109.3.8 list< analysis > word::get\_analysis (void) const**

get list of analysis (useful for perl API)

**8.109.3.9 string word::get\_form (void) const**

get word form

**8.109.3.10 string word::get\_lemma (void) const**

get lemma for the selected analysis in list

**8.109.3.11 int word::get\_n\_analysis (void) const**

get number of analysis in current list

**8.109.3.12 int word::get\_n\_words\_mw (void) const**

get number of words in compound

**8.109.3.13 string word::get\_parole (void) const**

get parole for the selected analysis

**8.109.3.14 analysis word::get\_selected\_analysis (void) const**

Get the selected analysis.

**8.109.3.15 list< string > word::get\_senses (void) const**

get sense list for the selected analysis

**8.109.3.16 string word::get\_short\_parole (const string &) const**

get parole (short version) for the selected analysis

**8.109.3.17 unsigned int word::get\_span\_finish (void) const****8.109.3.18 unsigned int word::get\_span\_start (void) const**

get token span.



**8.109.3.19** `list< word > word::get__words__mw (void) const`

get word objects that compound the multiword

**8.109.3.20** `bool word::is__ambiguous (void) const`

true iff the word has more than one analysis

**8.109.3.21** `bool word::is__multiword (void) const`

true iff the word is a multiword compound

**8.109.3.22** `word & word::operator= (const word &)`

assignment

**8.109.3.23** `void word::select__analysis (word::iterator)`

mark the given analysis as the selected one.

**8.109.3.24** `word::iterator word::selected__analysis (void) const`

Get an iterator to the selected analysis.

**8.109.3.25** `void word::set__analysis (const list< analysis > &)`

set analysis list, overwriting current values

**8.109.3.26** `void word::set__analysis (const analysis &)`

set analysis list to one single analysis, overwriting current values

**8.109.3.27** `void word::set__form (const string &)`

set word form

**8.109.3.28** `void word::set__found__in__dict (bool)`

set in\_\_dict

**8.109.3.29** `void word::set__senses (const list< string > &)`

set sense list for the selected analysis

**8.109.3.30** `void word::set__span (unsigned int, unsigned int)`

set token span

**8.109.3.31 void word::set\_\_user\_\_data (void \*)**

set user data

**8.109.4 Member Data Documentation****8.109.4.1 unsigned int word::finish [private]****8.109.4.2 string word::form [private]**

lexical form

**8.109.4.3 bool word::in\_\_dict [private]**

word form found in dictionary

**8.109.4.4 list<word> word::multiword [private]**

empty list if not a multiword

**8.109.4.5 word::iterator word::selected [private]**

selected analysis (if any)

**8.109.4.6 unsigned int word::start [private]**

token span

**8.109.4.7 void\* word::user [private]**

Private data.

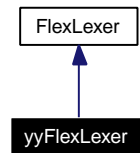
The documentation for this class was generated from the following files:

- **language.h**
- **language.cc**

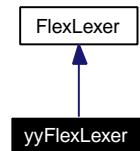
## 8.110 yyFlexLexer Class Reference

```
#include <FlexLexer.h>
```

Inheritance diagram for yyFlexLexer:



Collaboration diagram for yyFlexLexer:



### Public Member Functions

- **yyFlexLexer** (istream \*arg\_yyin=0, ostream \*arg\_yyout=0)
- virtual **~yyFlexLexer** ()
- void **yy\_switch\_to\_buffer** (struct yy\_buffer\_state \*new\_buffer)
- yy\_buffer\_state \* **yy\_create\_buffer** (istream \*s, int size)
- void **yy\_delete\_buffer** (struct yy\_buffer\_state \*b)
- void **yyrestart** (istream \*s)
- virtual int **yylex** ()
- virtual void **switch\_streams** (istream \*new\_in, ostream \*new\_out)

### Protected Member Functions

- virtual int **LexerInput** (char \*buf, int max\_size)
- virtual void **LexerOutput** (const char \*buf, int size)
- virtual void **LexerError** (const char \*msg)
- void **yyunput** (int c, char \*buf\_ptr)
- int **yyinput** ()
- void **yy\_load\_buffer\_state** ()
- void **yy\_init\_buffer** (struct yy\_buffer\_state \*b, istream \*s)
- void **yy\_flush\_buffer** (struct yy\_buffer\_state \*b)
- void **yy\_push\_state** (int new\_state)
- void **yy\_pop\_state** ()
- int **yy\_top\_state** ()
- **yy\_state\_type** **yy\_get\_previous\_state** ()
- **yy\_state\_type** **yy\_try\_NUL\_trans** (**yy\_state\_type** current\_state)
- int **yy\_get\_next\_buffer** ()

## Protected Attributes

- `int yy_start_stack_ptr`
- `int yy_start_stack_depth`
- `int * yy_start_stack`
- `istream * yyin`
- `ostream * yyout`
- `yy_buffer_state * yy_current_buffer`
- `char yy_hold_char`
- `int yy_n_chars`
- `char * yy_c_buf_p`
- `int yy_init`
- `int yy_start`
- `int yy_did_buffer_switch_on_eof`
- `yy_state_type yy_last_accepting_state`
- `char * yy_last_accepting_cpos`
- `yy_state_type * yy_state_buf`
- `yy_state_type * yy_state_ptr`
- `char * yy_full_match`
- `int * yy_full_state`
- `int yy_full_lp`
- `int yy_lp`
- `int yy_looking_for_trail_begin`
- `int yy_more_flag`
- `int yy_more_len`
- `int yy_more_offset`
- `int yy_prev_more_offset`

### 8.110.1 Constructor & Destructor Documentation

8.110.1.1 `yyFlexLexer::yyFlexLexer (istream * arg_yyin = 0, ostream * arg_yyout = 0)`

8.110.1.2 `virtual yyFlexLexer::~yyFlexLexer ()` [virtual]

### 8.110.2 Member Function Documentation

8.110.2.1 `virtual void yyFlexLexer::LexerError (const char * msg)` [protected, virtual]

8.110.2.2 `virtual int yyFlexLexer::LexerInput (char * buf, int max_size)` [protected, virtual]

8.110.2.3 `virtual void yyFlexLexer::LexerOutput (const char * buf, int size)` [protected, virtual]

8.110.2.4 `virtual void yyFlexLexer::switch_streams (istream * new_in, ostream * new_out)` [virtual]

Implements `FlexLexer` (p. 114).

**8.110.2.5** `struct yy_buffer_state* yyFlexLexer::yy_create_buffer (istream * s, int size)` [virtual]

Implements **FlexLexer** (p. 115).

**8.110.2.6** `void yyFlexLexer::yy_delete_buffer (struct yy_buffer_state * b)` [virtual]

Implements **FlexLexer** (p. 115).

**8.110.2.7** `void yyFlexLexer::yy_flush_buffer (struct yy_buffer_state * b)` [protected]

**8.110.2.8** `int yyFlexLexer::yy_get_next_buffer ()` [protected]

**8.110.2.9** `yy_state_type yyFlexLexer::yy_get_previous_state ()` [protected]

**8.110.2.10** `void yyFlexLexer::yy_init_buffer (struct yy_buffer_state * b, istream * s)` [protected]

**8.110.2.11** `void yyFlexLexer::yy_load_buffer_state ()` [protected]

**8.110.2.12** `void yyFlexLexer::yy_pop_state ()` [protected]

**8.110.2.13** `void yyFlexLexer::yy_push_state (int new_state)` [protected]

**8.110.2.14** `void yyFlexLexer::yy_switch_to_buffer (struct yy_buffer_state * new_buffer)` [virtual]

Implements **FlexLexer** (p. 115).

**8.110.2.15** `int yyFlexLexer::yy_top_state ()` [protected]

**8.110.2.16** `yy_state_type yyFlexLexer::yy_try_NUL_trans (yy_state_type current_state)` [protected]

**8.110.2.17** `int yyFlexLexer::yyinput ()` [protected]

**8.110.2.18** `virtual int yyFlexLexer::yylex ()` [virtual]

Implements **FlexLexer** (p. 115).

**8.110.2.19** `void yyFlexLexer::yyrestart (istream * s)` [virtual]

Implements **FlexLexer** (p. 115).

8.110.2.20 void yyFlexLexer::yyunput (int *c*, char \* *buf\_ptr*) [protected]

### 8.110.3 Member Data Documentation

8.110.3.1 char\* yyFlexLexer::yy\_c\_buf\_p [protected]

8.110.3.2 struct yy\_buffer\_state\* yyFlexLexer::yy\_current\_buffer [protected]

8.110.3.3 int yyFlexLexer::yy\_did\_buffer\_switch\_on\_eof [protected]

8.110.3.4 int yyFlexLexer::yy\_full\_lp [protected]

8.110.3.5 char\* yyFlexLexer::yy\_full\_match [protected]

8.110.3.6 int\* yyFlexLexer::yy\_full\_state [protected]

8.110.3.7 char yyFlexLexer::yy\_hold\_char [protected]

8.110.3.8 int yyFlexLexer::yy\_init [protected]

8.110.3.9 char\* yyFlexLexer::yy\_last\_accepting\_cpos [protected]

8.110.3.10 yy\_state\_type yyFlexLexer::yy\_last\_accepting\_state [protected]

8.110.3.11 int yyFlexLexer::yy\_looking\_for\_trail\_begin [protected]

8.110.3.12 int yyFlexLexer::yy\_lp [protected]

8.110.3.13 int yyFlexLexer::yy\_more\_flag [protected]

8.110.3.14 int yyFlexLexer::yy\_more\_len [protected]

8.110.3.15 int yyFlexLexer::yy\_more\_offset [protected]

8.110.3.16 int yyFlexLexer::yy\_n\_chars [protected]

8.110.3.17 int yyFlexLexer::yy\_prev\_more\_offset [protected]

8.110.3.18 int yyFlexLexer::yy\_start [protected]

8.110.3.19 int\* yyFlexLexer::yy\_start\_stack [protected]

8.110.3.20 int yyFlexLexer::yy\_start\_stack\_depth [protected]

8.110.3.21 int yyFlexLexer::yy\_start\_stack\_ptr [protected]

8.110.3.22 yy\_state\_type\* yyFlexLexer::yy\_state\_buf [protected]

8.110.3.23 yy\_state\_type\* yyFlexLexer::yy\_state\_ptr [protected]

8.110.3.24 istream\* yyFlexLexer::yyin [protected]

8.110.3.25 ostream\* yyFlexLexer::yyout [protected]

The documentation for this class was generated from the following file:

Generated on Wed Apr 26 12:55:30 2006 for FreeLing by Doxygen

- FlexLexer.h





## Chapter 9

# FreeLing File Documentation

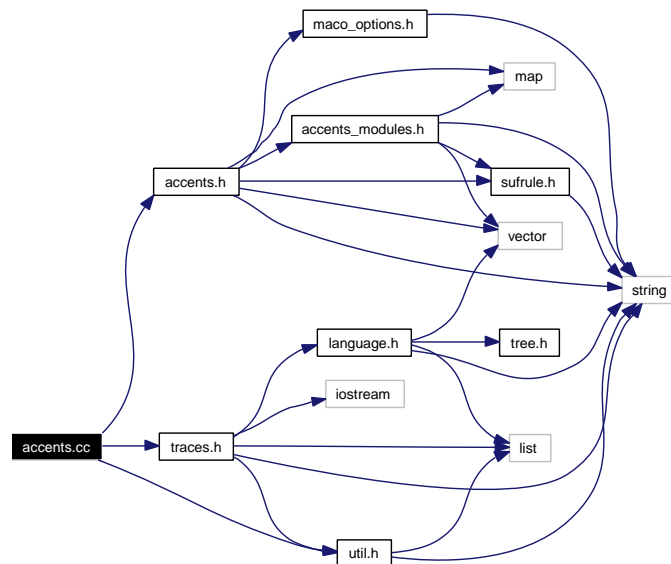
### 9.1 accents.cc File Reference

```
#include "accents.h"
```

```
#include "util.h"
```

```
#include "traces.h"
```

Include dependency graph for accents.cc:



### Defines

- `#define MOD_TRACENAME "ACCENTS"`
- `#define MOD_TRACECODE SUFF_TRACE`

### 9.1.1 Define Documentation

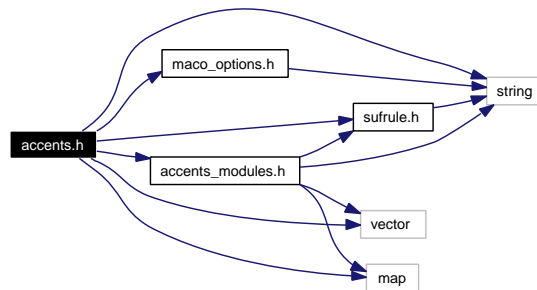
9.1.1.1 `#define MOD_TRACECODE SUFF_TRACE`

9.1.1.2 `#define MOD_TRACENAME "ACCENTS"`

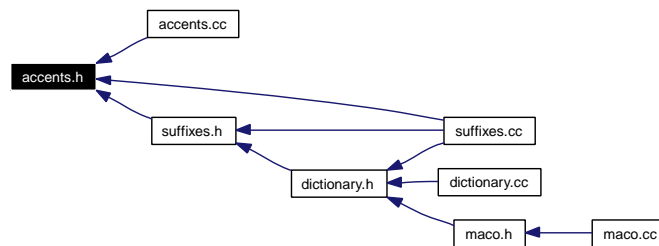
## 9.2 accents.h File Reference

```
#include <string>
#include <vector>
#include <map>
#include "maco_options.h"
#include "sufrule.h"
#include "accents_modules.h"
```

Include dependency graph for accents.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class `accents`

*The class `accents` provides a wrapper to transparently create and access an `accents__module`(p.28) to handle accentuation for the appropriate language.*

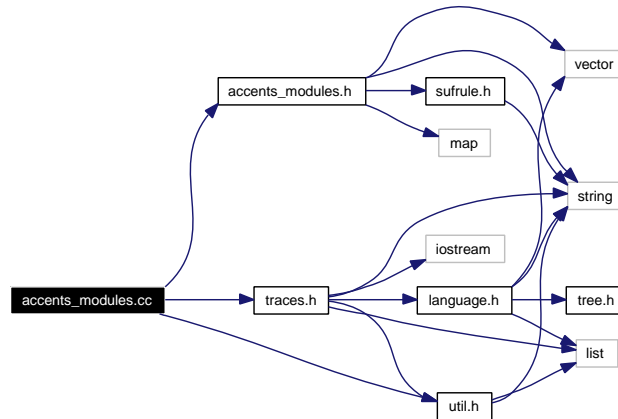
## 9.3 accents\_modules.cc File Reference

```
#include "accents_modules.h"
```

```
#include "util.h"
```

```
#include "traces.h"
```

Include dependency graph for accents\_modules.cc:



### Defines

- `#define MOD_TRACENAME "ACCENTS"`
- `#define MOD_TRACECODE SUFF_TRACE`

### 9.3.1 Define Documentation

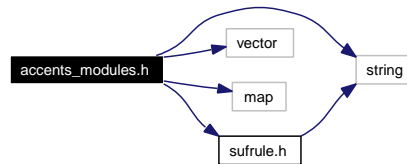
9.3.1.1 `#define MOD_TRACECODE SUFF_TRACE`

9.3.1.2 `#define MOD_TRACENAME "ACCENTS"`

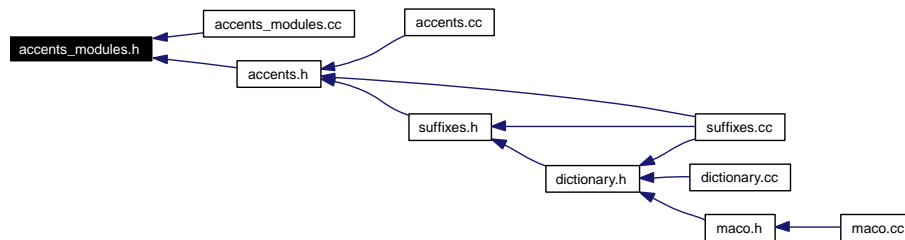
## 9.4 accents\_modules.h File Reference

```
#include <string>
#include <vector>
#include <map>
#include "sufrule.h"
```

Include dependency graph for accents\_modules.h:



This graph shows which files directly or indirectly include this file:



## Namespaces

- namespace `std`

## Classes

- class `accents_module`

*The abstract class `accents_module` generalizes accentuation rules for different languages.*

- class `accents_default`

*Derived `accents_module`(p.28) for null accentuation (eg english).*

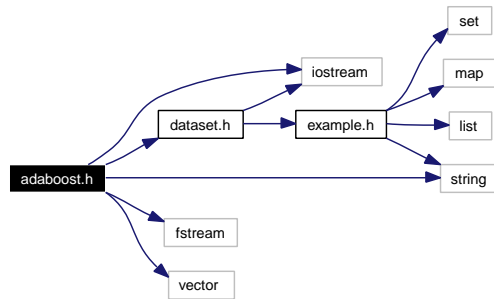
- class `accents_es`

*Derived `accents_module`(p.28) for Spanish accentuation.*

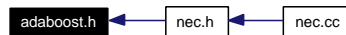
## 9.5 adaboost.h File Reference

```
#include "dataset.h"  
#include <fstream>  
#include <iostream>  
#include <string>  
#include <vector>
```

Include dependency graph for adaboost.h:



This graph shows which files directly or indirectly include this file:



### Classes

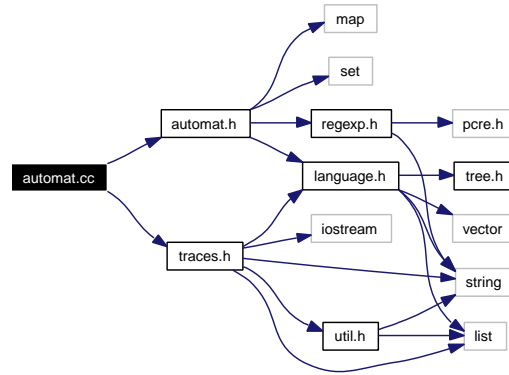
- class `adaboost`
- struct `adaboost::wr_holder`
- class `mlABTree`

## 9.6 automat.cc File Reference

```
#include "automat.h"
```

```
#include "traces.h"
```

Include dependency graph for automat.cc:



### Defines

- `#define MOD_TRACENAME "AUTOMAT"`
- `#define MOD_TRACECODE AUTOMAT_TRACE`

#### 9.6.1 Define Documentation

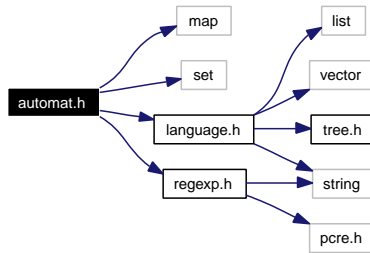
**9.6.1.1** `#define MOD_TRACECODE AUTOMAT_TRACE`

**9.6.1.2** `#define MOD_TRACENAME "AUTOMAT"`

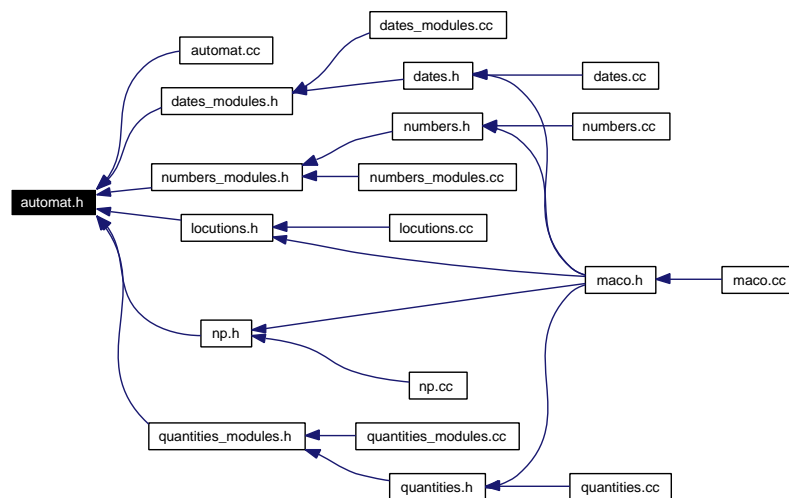
## 9.7 automat.h File Reference

```
#include <map>
#include <set>
#include "language.h"
#include "regexp.h"
```

Include dependency graph for automat.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **automat**

*Abstract class to implement a Finite-State Automaton which is used by modules recognizing multiwords (dates, numbers, quantities, .*

### Defines

- `#define MAX_STATES 100`
- `#define MAX_TOKENS 50`



## 9.7.1 Define Documentation

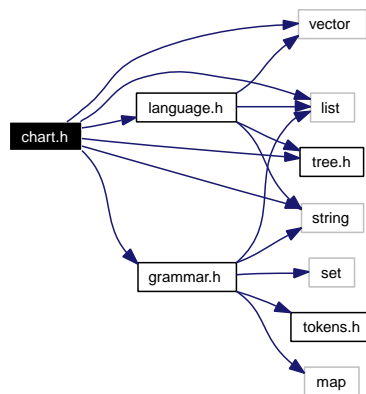
**9.7.1.1** `#define MAX_STATES 100`

**9.7.1.2** `#define MAX_TOKENS 50`

## 9.8 chart.h File Reference

```
#include <list>
#include <vector>
#include <string>
#include "tree.h"
#include "language.h"
#include "grammar.h"
```

Include dependency graph for chart.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **edge**  
*Class edge stores all information in a chart edge.*
- class **cell**  
*Class cell stores all information in a chart cell.*
- class **chart**  
*Class chart contains an array of cells that constitute a chart.*

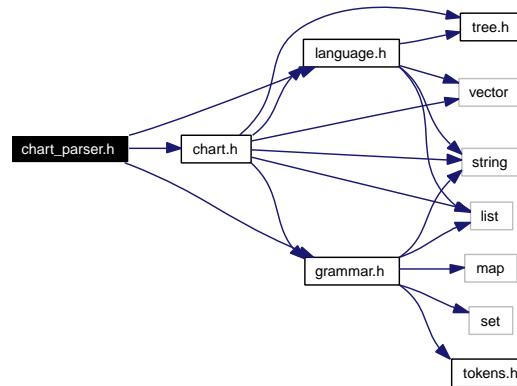
## 9.9 chart\_parser.h File Reference

```
#include "language.h"
```

```
#include "grammar.h"
```

```
#include "chart.h"
```

Include dependency graph for chart\_parser.h:



### Classes

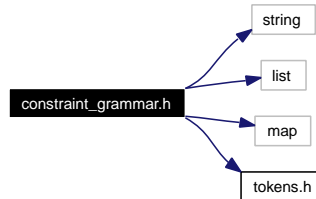
- class `chart_parser`

*Class `chart_parser` implements a chart parser.*

## 9.10 constraint\_grammar.h File Reference

```
#include <string>
#include <list>
#include <map>
#include "tokens.h"
```

Include dependency graph for constraint\_grammar.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **condition**

*Class condition implements a condition of a CG rule.*

- class **ruleCG**

*Class rule implements a rule of a CG.*

- class **constraint\_grammar**

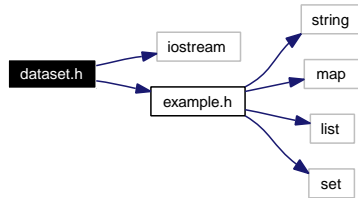
*Class constraint\_grammar implements a pseudo CG, ready to be used from a relax PoS tagger.*

## 9.11 dataset.h File Reference

```
#include <iostream>
```

```
#include "example.h"
```

Include dependency graph for dataset.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **dataset**
- struct **dataset::mlDatasetNode**
- class **dataset::iterator**

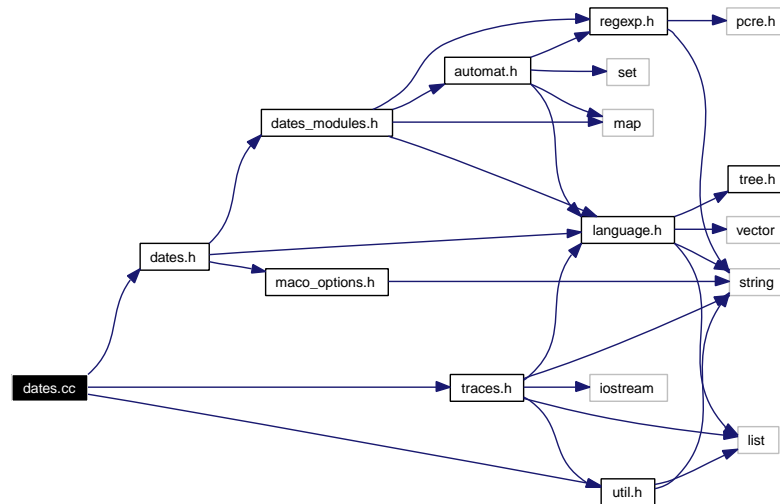
## 9.12 dates.cc File Reference

```
#include "dates.h"
```

```
#include "util.h"
```

```
#include "traces.h"
```

Include dependency graph for dates.cc:



### Defines

- `#define MOD_TRACENAME "DATES"`
- `#define MOD_TRACECODE DATES_TRACE`

### 9.12.1 Define Documentation

9.12.1.1 `#define MOD_TRACECODE DATES_TRACE`

9.12.1.2 `#define MOD_TRACENAME "DATES"`

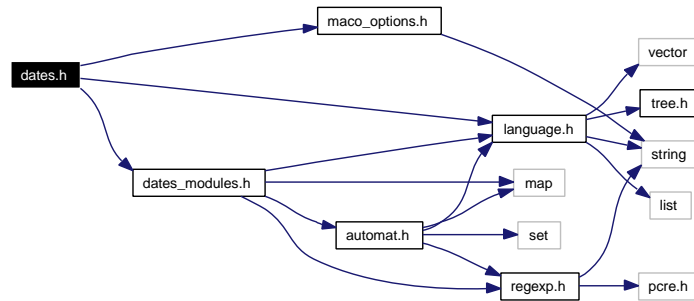
## 9.13 dates.h File Reference

```
#include "language.h"
```

```
#include "maco_options.h"
```

```
#include "dates_modules.h"
```

Include dependency graph for dates.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class **dates**

*The class `dates` provides a wrapper to transparently create and access a `dates_module`(p.89), a temporal expression recognizer for the appropriate language.*

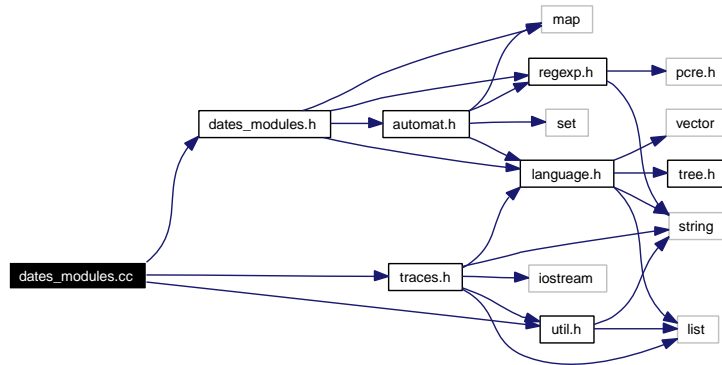
## 9.14 dates\_modules.cc File Reference

```
#include "dates_modules.h"
```

```
#include "util.h"
```

```
#include "traces.h"
```

Include dependency graph for dates\_modules.cc:



### Defines

- #define **MOD\_TRACENAME** "DATES"
- #define **MOD\_TRACECODE** DATES\_TRACE
- #define **A** 1
- #define **B** 2
- #define **C** 3
- #define **D** 4
- #define **STOP** 5
- #define **TK\_hour** 1
- #define **TK\_hhmm** 2
- #define **TK\_min** 3
- #define **TK\_date** 4
- #define **TK\_other** 5
- #define **A** 1
- #define **B** 2
- #define **C** 3
- #define **D** 4
- #define **E** 5
- #define **F** 6
- #define **G** 7
- #define **H** 8
- #define **I** 9
- #define **Ib** 10
- #define **J** 11
- #define **K** 12
- #define **L** 13
- #define **P** 14



- `#define S1` 15
- `#define S2` 16
- `#define Ha` 17
- `#define Hb` 18
- `#define Hc` 19
- `#define Hd` 20
- `#define He` 21
- `#define AH` 22
- `#define BH` 23
- `#define CH` 24
- `#define DH` 25
- `#define EH` 26
- `#define EHb` 27
- `#define FH` 28
- `#define GH` 29
- `#define BH1` 30
- `#define BH2` 31
- `#define CH1` 32
- `#define DH1` 33
- `#define EH1` 34
- `#define EH1b` 35
- `#define FH1` 36
- `#define FH1b` 37
- `#define GH1` 38
- `#define STOP` 39
- `#define TK_ weekday` 1
- `#define TK_ daynum` 2
- `#define TK_ month` 3
- `#define TK_ shmonth` 4
- `#define TK_ monthnum` 5
- `#define TK_ number` 6
- `#define TK_ comma` 7
- `#define TK_ dot` 8
- `#define TK_ colon` 9
- `#define TK_ wday` 10
- `#define TK_ wmonth` 11
- `#define TK_ wyear` 12
- `#define TK_ wpast` 13
- `#define TK_ wcentury` 14
- `#define TK_ roman` 15
- `#define TK_ wacdc` 16
- `#define TK_ wampm` 17
- `#define TK_ hournum` 18
- `#define TK_ minnum` 19
- `#define TK_ wquart` 20
- `#define TK_ wy` 21
- `#define TK_ wmenos` 22
- `#define TK_ wmorning` 23
- `#define TK_ wmidnight` 24
- `#define TK_ wen` 25

- `#define TK_wa` 26
- `#define TK_wde` 27
- `#define TK_wdel` 28
- `#define TK_wel` 29
- `#define TK_wpor` 30
- `#define TK_whacia` 31
- `#define TK_weso` 32
- `#define TK_wla` 33
- `#define TK_wpunto` 34
- `#define TK_whour` 35
- `#define TK_wmin` 36
- `#define TK_hour` 37
- `#define TK_hhmm` 38
- `#define TK_min` 39
- `#define TK_date` 40
- `#define TK_other` 41
- `#define A` 1
- `#define B` 2
- `#define C` 3
- `#define D` 4
- `#define E` 5
- `#define F` 6
- `#define G` 7
- `#define H` 8
- `#define I` 9
- `#define Ib` 10
- `#define J` 11
- `#define K` 12
- `#define L` 13
- `#define P` 14
- `#define S1` 15
- `#define S2` 16
- `#define Ha` 17
- `#define Hb` 18
- `#define Hc` 19
- `#define AH` 22
- `#define BH` 23
- `#define CH` 24
- `#define DH` 25
- `#define EH` 26
- `#define EHb` 27
- `#define FH` 28
- `#define GH` 29
- `#define BH1` 30
- `#define BH2` 31
- `#define CH1` 32
- `#define DH1` 33
- `#define EH1` 34
- `#define EH1b` 35
- `#define FH1` 36

- `#define FH1b` 37
- `#define GH1` 38
- `#define STOP` 39
- `#define IH` 40
- `#define JH` 41
- `#define KH` 42
- `#define LH` 43
- `#define MH` 44
- `#define MHb` 45
- `#define NH` 46
- `#define IH1` 47
- `#define JH1` 48
- `#define KH1` 49
- `#define LH1` 50
- `#define MH1` 51
- `#define MH1b` 52
- `#define NH1` 53
- `#define TK_ weekday` 1
- `#define TK_ daynum` 2
- `#define TK_ month` 3
- `#define TK_ shmonth` 4
- `#define TK_ monthnum` 5
- `#define TK_ number` 6
- `#define TK_ comma` 7
- `#define TK_ dot` 8
- `#define TK_ colon` 9
- `#define TK_ wday` 10
- `#define TK_ wmonth` 11
- `#define TK_ wyear` 12
- `#define TK_ wpast` 13
- `#define TK_ wcentury` 14
- `#define TK_ roman` 15
- `#define TK_ wacdc` 16
- `#define TK_ wampm` 17
- `#define TK_ hournum` 18
- `#define TK_ minnum` 19
- `#define TK_ wquart` 20
- `#define TK_ wi` 21
- `#define TK_ wmenys` 22
- `#define TK_ wmorning` 23
- `#define TK_ wmidnight` 24
- `#define TK_ wen` 25
- `#define TK_ wa` 26
- `#define TK_ wal` 27
- `#define TK_ wde` 28
- `#define TK_ wdel` 29
- `#define TK_ wel` 30
- `#define TK_ wpor` 31
- `#define TK_ wcap` 32
- `#define TK_ walla` 33

- `#define TK_wla 34`
- `#define TK_wpunto 35`
- `#define TK_whatour 36`
- `#define TK_wmin 37`
- `#define TK_hour 38`
- `#define TK_hhmm 39`
- `#define TK_min 40`
- `#define TK_date 41`
- `#define TK_other 42`
- `#define TK_wmitja 43`
- `#define TK_wmig 44`
- `#define TK_quartnum 45`

## 9.14.1 Define Documentation

9.14.1.1 `#define A 1`

9.14.1.2 `#define A 1`

9.14.1.3 `#define A 1`

9.14.1.4 `#define AH 22`

9.14.1.5 `#define AH 22`

9.14.1.6 `#define B 2`

9.14.1.7 `#define B 2`

9.14.1.8 `#define B 2`

9.14.1.9 `#define BH 23`

9.14.1.10 `#define BH 23`

9.14.1.11 `#define BH1 30`

9.14.1.12 `#define BH1 30`

9.14.1.13 `#define BH2 31`

9.14.1.14 `#define BH2 31`

9.14.1.15 `#define C 3`

9.14.1.16 `#define C 3`

9.14.1.17 `#define C 3`

9.14.1.18 `#define CH 24`

9.14.1.19 `#define CH 24`

9.14.1.20 `#define CH1 32`

9.14.1.21 `#define CH1 32`

9.14.1.22 `#define D 4`

9.14.1.23 `#define D 4`

9.14.1.24 `#define D 4`

9.14.1.25 `#define DH 25`

9.14.1.26 `#define DH 25`

9.14.1.27 `#define DH1 33`

---

Generated on Wed Apr 26 12:55:30 2006 for FreeLing by Doxygen

9.14.1.28 `#define DH1 33`

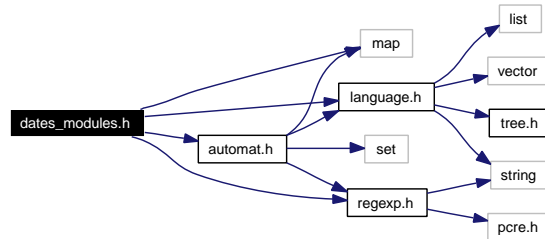
9.14.1.29 `#define E 5`

9.14.1.30 `#define E 5`

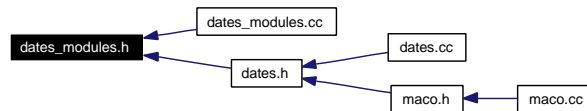
## 9.15 dates\_modules.h File Reference

```
#include <map>
#include "language.h"
#include "automat.h"
#include "regex.h"
```

Include dependency graph for dates\_modules.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class **dates\_module**

*The abstract class dates\_module generalizes temporal expression recognizer for different languages.*

- class **dates\_default**

*The derived class dates\_default implements a default date/time recognizer (only simple patterns are recognized).*

- class **dates\_es**

*The derived class dates\_es implements a Spanish date/time recognizer.*

- class **dates\_ca**

*The derived class dates\_ca implements a Catalan date/time recognizer.*

## Defines

- **#define RE\_ROMAN** `"^([IVXLCDM]+)$"`
- **#define RE\_DATE\_DF** `"^((?:((?:[0-3])?(?:\\d)))/(?:((?:[0-1])?(?:\\d)))?(\\d{1,4}))$"`
- **#define RE\_TIME1\_DF** `"^((?:((?:[0-1])?(?:\\d))|(?:2(?:[01234])))(?:h)((?:[0-5])?(?:\\d))(?:min|m)?)$"`

- `#define RE_TIME2_DF "^(?:((?:[0-5])?(?:\\d))(?:min\\.|min|m\\.|m))$"`
- `#define RE_DATE_ES "^(?:((?:[0-3])?(?:\\d)))/(?:((?:[0-1])?(?:\\d))|enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octubre|noviembre|diciembre|ene|feb|mar)"`
- `#define RE_TIME1_ES "^(?:((?:[0-1])?(?:\\d))|(?:2(?:[01234])))(?:h)((?:[0-5])?(?:\\d))(?:minutos|min|m)?)"`
- `#define RE_TIME2_ES "^(?:((?:[0-5])?(?:\\d))(?:minutos|min\\.|min|m\\.|m))$"`
- `#define RE_DATE_CA "^(?:((?:[0-3])?(?:\\d)))/(?:((?:[0-1])?(?:\\d))|gener|febrer|març|abril|maig|juny|juliol|agost|setembre|octubre|novembre|desembre|gen|feb|mar|abr)"`
- `#define RE_TIME1_CA "^(?:((?:[0-1])?(?:\\d))|(?:2(?:[01234])))(?:h)((?:[0-5])?(?:\\d))(?:minuts|min|m)?)"`
- `#define RE_TIME2_CA "^(?:((?:[0-5])?(?:\\d))(?:minuts|min\\.|min|m\\.|m))$"`
- `#define RE_DATE_EN "^(?:((?:[0-3])?(?:\\d)))/(?:((?:[0-1])?(?:\\d))|january|february|march|april|may|june|july|august|september|october|november|december|jan|feb|mar)"`
- `#define RE_TIME1_EN "^(?:((?:[0-1])?(?:\\d))|(?:2(?:[01234])))(?:h)((?:[0-5])?(?:\\d))(?:minutes|min|m)?)"`
- `#define RE_TIME2_EN "^(?:((?:[0-5])?(?:\\d))(?:minutes|min\\.|min|m\\.|m))$"`

### 9.15.1 Define Documentation

- 9.15.1.1 `#define RE_DATE_CA "^(?:((?:[0-3])?(?:\\d)))/(?:((?:[0-1])?(?:\\d))|gener|febrer|march|abril|maig|juny|juliol|agost|setembre|octubre|novembre|desembre)"`
- 9.15.1.2 `#define RE_DATE_DF "^(?:((?:[0-3])?(?:\\d)))/(?:((?:[0-1])?(?:\\d)))/(\\d{1,4}))$"`
- 9.15.1.3 `#define RE_DATE_EN "^(?:((?:[0-3])?(?:\\d)))/(?:((?:[0-1])?(?:\\d))|january|february|march|april|may|june|july|august|september|october|november|december)"`
- 9.15.1.4 `#define RE_DATE_ES "^(?:((?:[0-3])?(?:\\d)))/(?:((?:[0-1])?(?:\\d))|enero|febrero|marcho|abril|mayo|junio|julio|agosto|septiembre|octubre|noviembre|diciembre)"`
- 9.15.1.5 `#define RE_ROMAN "^(IVXLCDM|+)$"`
- 9.15.1.6 `#define RE_TIME1_CA "^(?:((?:[0-1])?(?:\\d))|(?:2(?:[01234])))(?:h)((?:[0-5])?(?:\\d))(?:minuts|min|m)?)"`
- 9.15.1.7 `#define RE_TIME1_DF "^(?:((?:[0-1])?(?:\\d))|(?:2(?:[01234])))(?:h)((?:[0-5])?(?:\\d))(?:min|m)?)"`
- 9.15.1.8 `#define RE_TIME1_EN "^(?:((?:[0-1])?(?:\\d))|(?:2(?:[01234])))(?:h)((?:[0-5])?(?:\\d))(?:minutes|min|m)?)"`
- 9.15.1.9 `#define RE_TIME1_ES "^(?:((?:[0-1])?(?:\\d))|(?:2(?:[01234])))(?:h)((?:[0-5])?(?:\\d))(?:minutos|min|m)?)"`
- 9.15.1.10 `#define RE_TIME2_CA "^(?:((?:[0-5])?(?:\\d)))(?:minuts|min\\.|min|m\\.|m)$"`
- 9.15.1.11 `#define RE_TIME2_DF "^(?:((?:[0-5])?(?:\\d)))(?:min\\.|min|m\\.|m)$"`
- 9.15.1.12 `#define RE_TIME2_EN "^(?:((?:[0-5])?(?:\\d)))(?:minutes|min\\.|min|m\\.|m)$"`
- 9.15.1.13 `#define RE_TIME2_ES "^(?:((?:[0-5])?(?:\\d)))(?:minutos|min\\.|min|m\\.|m)$"`

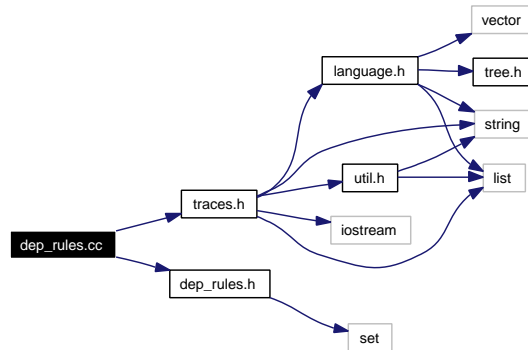


## 9.16 dep\_rules.cc File Reference

```
#include "traces.h"
```

```
#include "dep_rules.h"
```

Include dependency graph for dep\_rules.cc:



### Defines

- `#define MOD_TRACENAME "DEPENDENCIES"`
- `#define MOD_TRACECODE DEP_TRACE`

#### 9.16.1 Define Documentation

9.16.1.1 `#define MOD_TRACECODE DEP_TRACE`

9.16.1.2 `#define MOD_TRACENAME "DEPENDENCIES"`

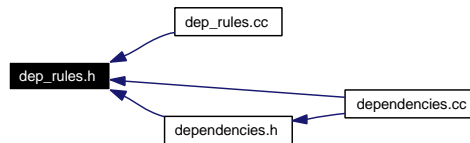
## 9.17 dep\_rules.h File Reference

```
#include <set>
```

Include dependency graph for dep\_rules.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **completerRule**  
*The class completerRule stores rules used by the completer of parse trees.*
- class **rule\_expression**  
*The class rule\_expression is an abstract class (interface) for building dynamic restriction on a ruleLabeler(p. 215) which are used by class depLabeler(p. 99).*
- class **check\_and**  
*And of basic constraints.*
- class **check\_not**  
*negation*
- class **check\_side**  
*descendant is to the right of its ancestor*
- class **check\_A\_lemma**  
*ancestor lemma in list of lemmas (separator character is |)*
- class **check\_S\_lemma**  
*descendant lemma in list of lemmas (separator character is |)*
- class **check\_S\_category**  
*ancestor category*
- class **check\_A\_wordclass**  
*ancestor lemma belongs to a verb class*
- class **ruleLabeler**  
*ruleLabeler is an auxiliary class for the depLabeler(p. 99)*

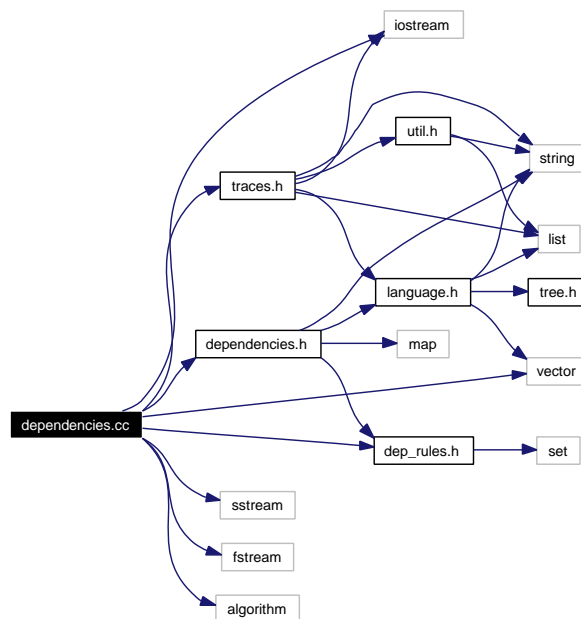
- class **dep\_info**

*auxiliary class to store information about dependency building*

## 9.18 dependencies.cc File Reference

```
#include <iostream>
#include <sstream>
#include <fstream>
#include <vector>
#include <algorithm>
#include "traces.h"
#include "dependencies.h"
#include "dep_rules.h"
```

Include dependency graph for dependencies.cc:



### Defines

- `#define MOD_TRACENAME "DEPENDENCIES"`
- `#define MOD_TRACECODE DEP_TRACE`

### Functions

- `void PrintTree (parse_tree::iterator n, int depth)`
- `void PrintDepTree (dep_tree::iterator n, int depth)`

## 9.18.1 Define Documentation

9.18.1.1 `#define MOD_TRACECODE DEP_TRACE`

9.18.1.2 `#define MOD_TRACENAME "DEPENDENCIES"`

## 9.18.2 Function Documentation

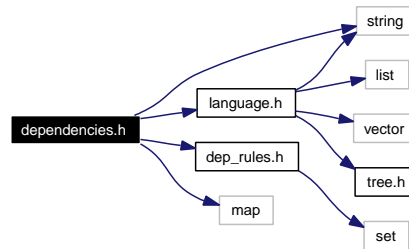
9.18.2.1 `void PrintDepTree (dep_tree::iterator n, int depth)`

9.18.2.2 `void PrintTree (parse_tree::iterator n, int depth)`

## 9.19 dependencies.h File Reference

```
#include "language.h"
#include "dep_rules.h"
#include <string>
#include <map>
```

Include dependency graph for dependencies.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class **completer**

*The class completer implements a parse tree completer, which given a partial parse tree (chunker output), completes the full parse according to some grammar rules.*

- class **depLabeler**

*depLabeler is class to set labels into a dependency tree*

- class **dependencyMaker**

*dependencyMaker is a class for obtaining a dependency tree from chunks.*

## Typedefs

- typedef **rule\_expression \* expcreator** (const string &)

*expression creator generic function, to allow registering new expressions*

### 9.19.1 Typedef Documentation

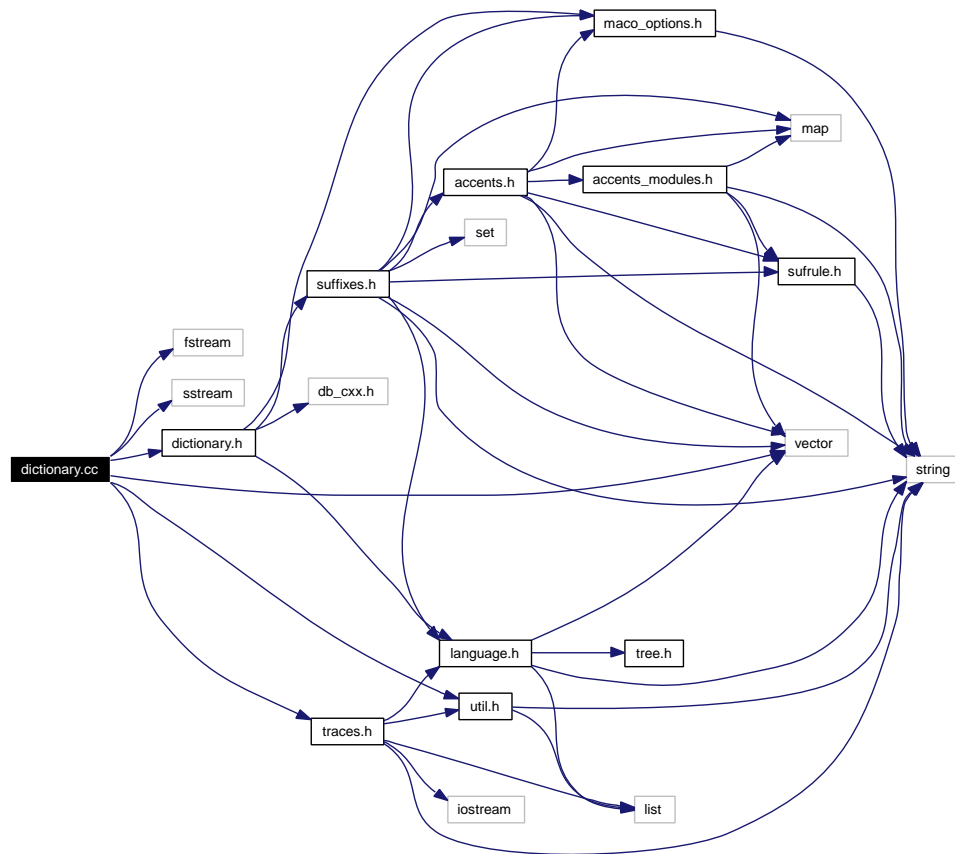
#### 9.19.1.1 typedef rule\_expression\* expcreator(const string &)

expression creator generic function, to allow registering new expressions

## 9.20 dictionary.cc File Reference

```
#include <fstream>
#include <sstream>
#include <vector>
#include "dictionary.h"
#include "util.h"
#include "traces.h"
```

Include dependency graph for dictionary.cc:



### Defines

- `#define MOD_TRACENAME "DICTIONARY"`
- `#define MOD_TRACECODE DICT_TRACE`

### 9.20.1 Define Documentation

9.20.1.1 `#define MOD_TRACECODE DICT_TRACE`

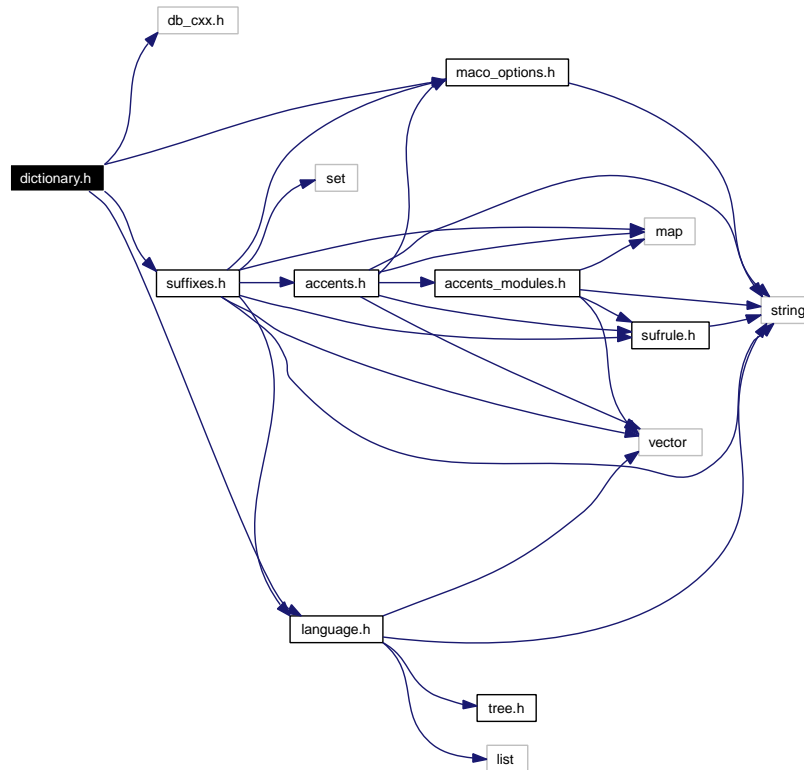
9.20.1.2 `#define MOD_TRACENAME "DICTIONARY"`



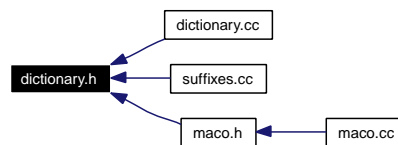
## 9.21 dictionary.h File Reference

```
#include <db_cxx.h>
#include "maco_options.h"
#include "language.h"
#include "suffixes.h"
```

Include dependency graph for dictionary.h:



This graph shows which files directly or indirectly include this file:



### Classes

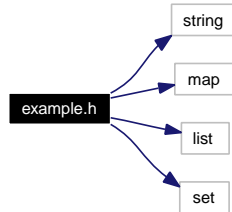
- class **dictionary**

*The class dictionary implements dictionary search and suffix analysis for word forms.*

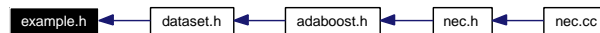
## 9.22 example.h File Reference

```
#include <string>
#include <map>
#include <list>
#include <set>
```

Include dependency graph for example.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **iFeature**
- class **map\_input**
- class **map\_input::const\_iterator**
- class **set\_input**
- class **mlOutput**
- struct **mlOutput::label**
- class **example**

### Typedefs

- typedef **map\_input** input

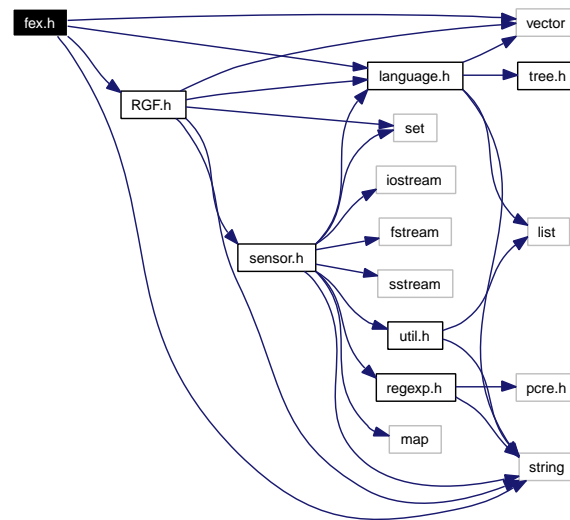
#### 9.22.1 Typedef Documentation

##### 9.22.1.1 typedef map\_input input

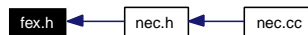
## 9.23 fex.h File Reference

```
#include <string>
#include <vector>
#include "language.h"
#include "RGF.h"
```

Include dependency graph for fex.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class `fex`

## 9.24 FlexLexer.h File Reference

```
#include <iostream>
```

Include dependency graph for FlexLexer.h:



### Classes

- class **FlexLexer**
- class **yyFlexLexer**

### Defines

- `#define yyFlexLexerOnce`

### Typedefs

- `typedef int yy_state_type`

#### 9.24.1 Define Documentation

##### 9.24.1.1 `#define yyFlexLexerOnce`

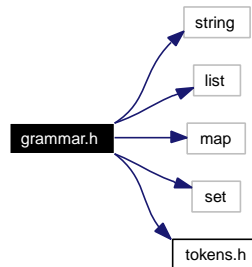
#### 9.24.2 Typedef Documentation

##### 9.24.2.1 `typedef int yy_state_type`

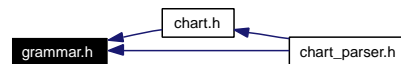
## 9.25 grammar.h File Reference

```
#include <string>
#include <list>
#include <map>
#include <set>
#include "tokens.h"
```

Include dependency graph for grammar.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **rule**

*Class rule implements a rule of a grammar.*

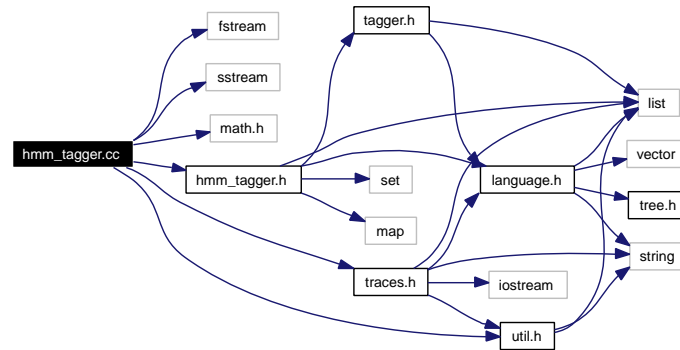
- class **grammar**

*Class grammar implements a CFG, ready to be used from a chart parser.*

## 9.26 hmm\_tagger.cc File Reference

```
#include <fstream>
#include <sstream>
#include <math.h>
#include "hmm_tagger.h"
#include "util.h"
#include "traces.h"
```

Include dependency graph for hmm\_tagger.cc:



### Defines

- `#define MOD_TRACENAME "HMMTAGGER"`
- `#define MOD_TRACECODE HMM_TRACE`

### 9.26.1 Define Documentation

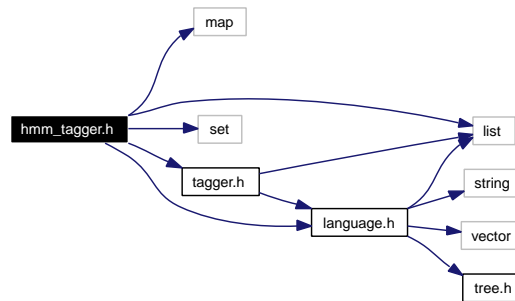
9.26.1.1 `#define MOD_TRACECODE HMM_TRACE`

9.26.1.2 `#define MOD_TRACENAME "HMMTAGGER"`

## 9.27 hmm\_tagger.h File Reference

```
#include <map>
#include <list>
#include <set>
#include "language.h"
#include "tagger.h"
```

Include dependency graph for hmm\_tagger.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **viterbi**

*The class viterbi stores the two maps for each observation: The delta map stores the maximum probability for each state in that observation, the phi map stores the backpath to maximize the probability.*

- class **emission\_states**

*The class emission\_states stores the list of states in the HMM that *may* be generating a given word given the two previous words (and their valid tags).*

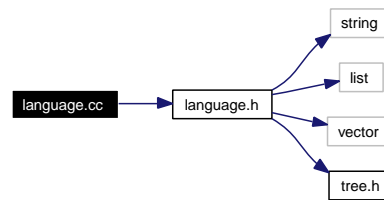
- class **hmm\_tagger**

*The class hmm\_tagger implements the syntactic analyzer and is the main class, which uses all the others.*

## 9.28 language.cc File Reference

```
#include "language.h"
```

Include dependency graph for language.cc:

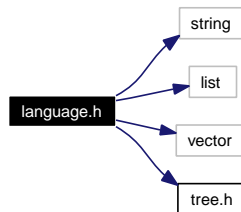




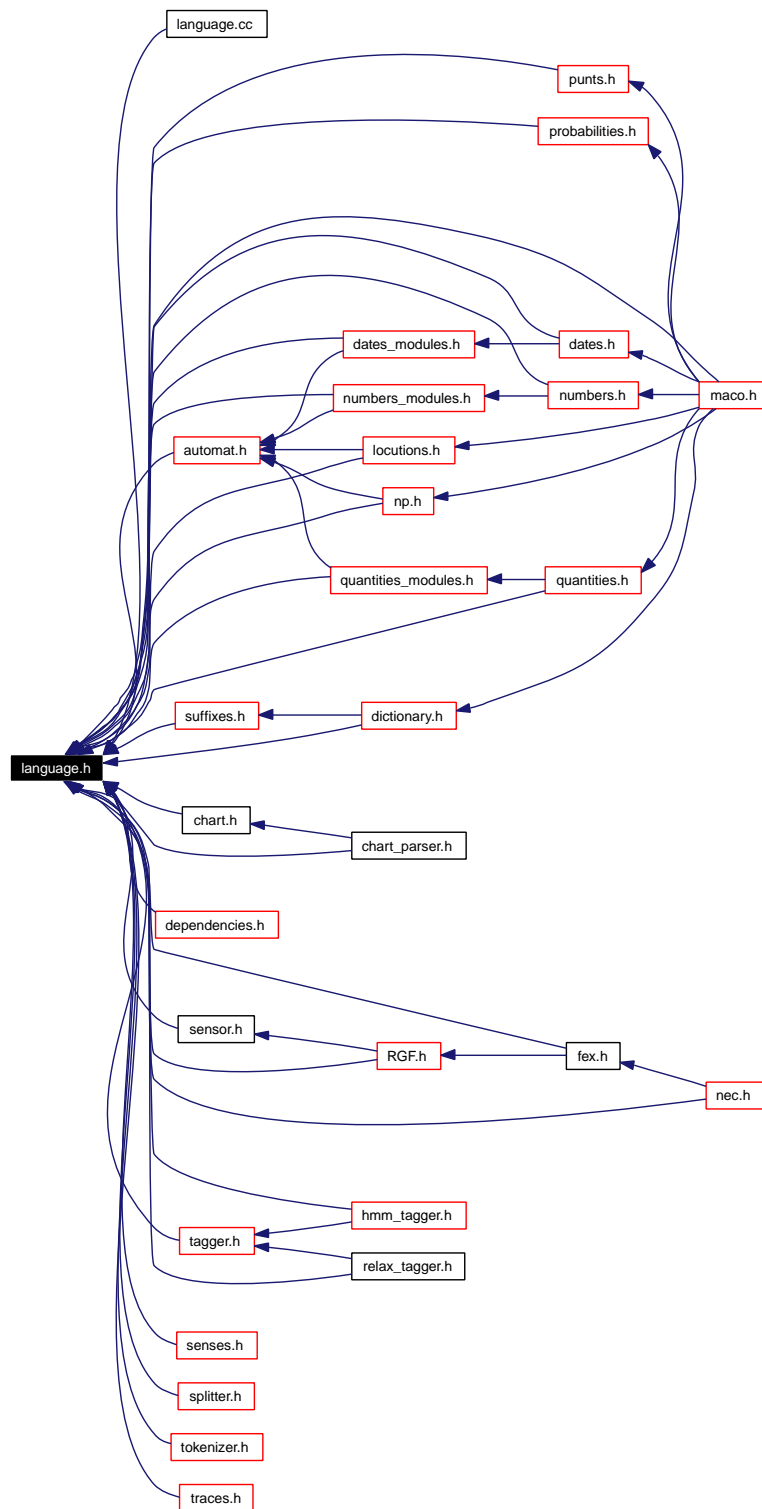
## 9.29 language.h File Reference

```
#include <string>
#include <list>
#include <vector>
#include "tree.h"
```

Include dependency graph for language.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class **analysis**

*Class analysis stores a possible reading (lemma, PoS, probability) for a word.*

- class **word**

*Class word stores all info related to a word: form, list of analysis, list of tokens (if multiword).*

- class **node**

*Class node stores nodes of a **parse\_tree**(p.175) Each node in the tree is either a label (intermediate node) or a word (leaf node).*

- class **parse\_tree**

*Class parse tree is used to store the results of parsing.*

- class **depnode**

*class denode stores nodes of a dependency tree and parse tree <-> deptree relations*

- class **dep\_tree**

*class dep\_tree stores a dependency tree*

- class **sentence**

*Class sentence is just a list of words that someone (the splitter) has validated it as a complete sentence.*

- class **paragraph**

*Class paragraph is just a list of sentences that someone has validated it as a paragraph.*

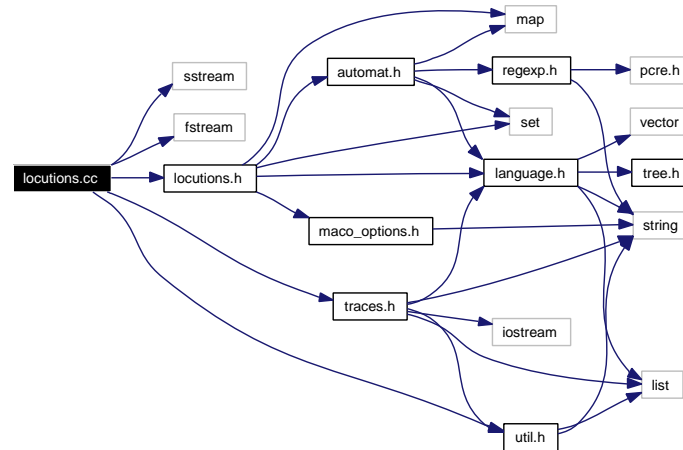
- class **document**

*Class document is a list of paragraphs.*

## 9.30 locutions.cc File Reference

```
#include <sstream>
#include <fstream>
#include "locutions.h"
#include "util.h"
#include "traces.h"
```

Include dependency graph for locutions.cc:



### Defines

- `#define MOD_TRACENAME "LOCUTIONS"`
- `#define MOD_TRACECODE LOCUT_TRACE`
- `#define P 1`
- `#define M 2`
- `#define STOP 3`
- `#define TK_pref 1`
- `#define TK_mw 2`
- `#define TK_prefL 3`
- `#define TK_mwL 4`
- `#define TK_prefP 5`
- `#define TK_mwP 6`
- `#define TK_other 7`

## 9.30.1 Define Documentation

9.30.1.1 `#define M 2`

9.30.1.2 `#define MOD_TRACECODE LOCUT_TRACE`

9.30.1.3 `#define MOD_TRACENAME "LOCUTIONS"`

9.30.1.4 `#define P 1`

9.30.1.5 `#define STOP 3`

9.30.1.6 `#define TK_mw 2`

9.30.1.7 `#define TK_mwL 4`

9.30.1.8 `#define TK_mwP 6`

9.30.1.9 `#define TK_other 7`

9.30.1.10 `#define TK_pref 1`

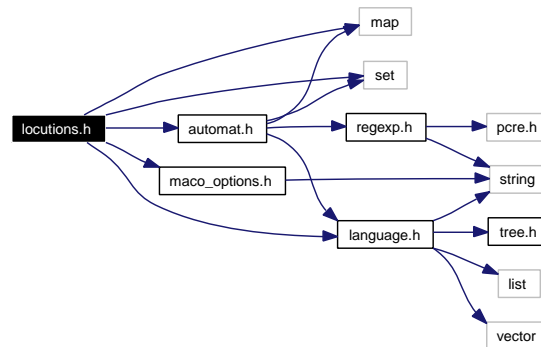
9.30.1.11 `#define TK_prefL 3`

9.30.1.12 `#define TK_prefP 5`

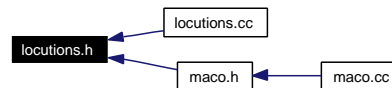
## 9.31 locutions.h File Reference

```
#include <map>
#include <set>
#include "maco_options.h"
#include "language.h"
#include "automat.h"
```

Include dependency graph for locutions.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class **locutions**

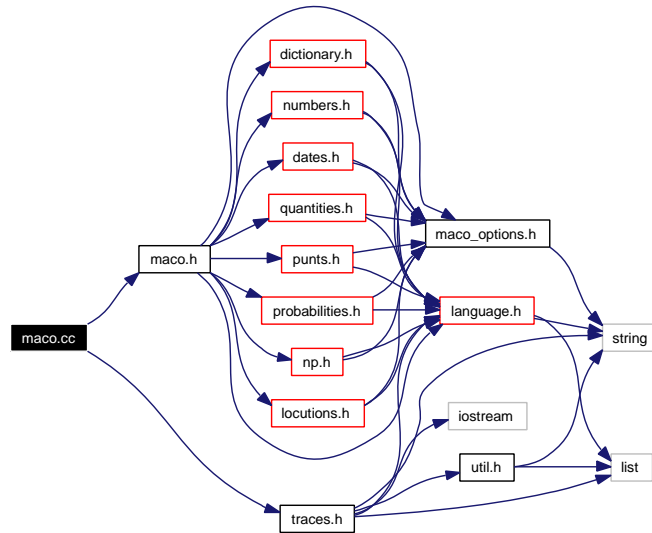
*Class locutions recognizes multiwords belonging to a list obtained from a configuration file.*

## 9.32 maco.cc File Reference

```
#include "maco.h"
```

```
#include "traces.h"
```

Include dependency graph for maco.cc:



### Defines

- `#define MOD_TRACENAME "MACO"`
- `#define MOD_TRACECODE MACO_TRACE`

### 9.32.1 Define Documentation

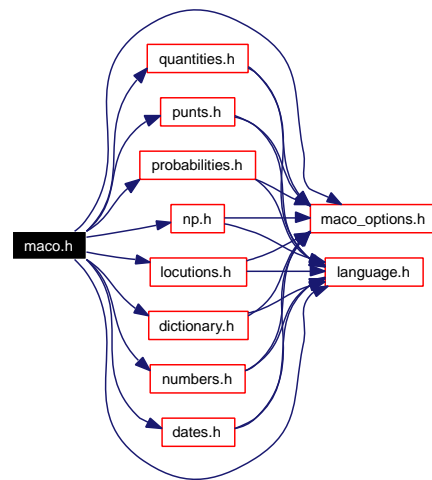
9.32.1.1 `#define MOD_TRACECODE MACO_TRACE`

9.32.1.2 `#define MOD_TRACENAME "MACO"`

### 9.33 maco.h File Reference

```
#include "maco_options.h"
#include "language.h"
#include "locutions.h"
#include "dictionary.h"
#include "numbers.h"
#include "dates.h"
#include "quantities.h"
#include "punts.h"
#include "probabilities.h"
#include "np.h"
```

Include dependency graph for maco.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **maco**

*Class maco implements the morphological analyzer, which uses all the specific analyzers: dates, numbers, dictionary, etc.*



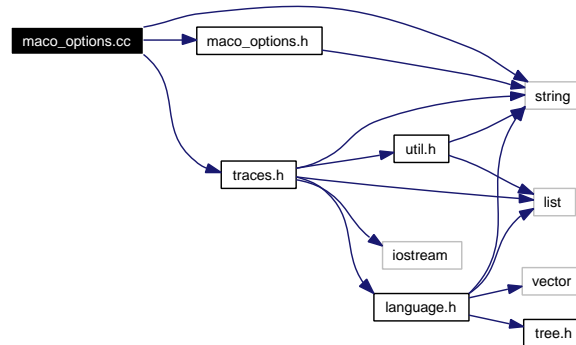
## 9.34 maco\_options.cc File Reference

```
#include <string>
```

```
#include "maco_options.h"
```

```
#include "traces.h"
```

Include dependency graph for maco\_options.cc:



### Defines

- `#define MOD_TRACENAME "MACO_OPTIONS"`
- `#define MOD_TRACECODE OPTIONS_TRACE`

#### 9.34.1 Define Documentation

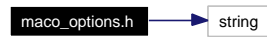
9.34.1.1 `#define MOD_TRACECODE OPTIONS_TRACE`

9.34.1.2 `#define MOD_TRACENAME "MACO_OPTIONS"`

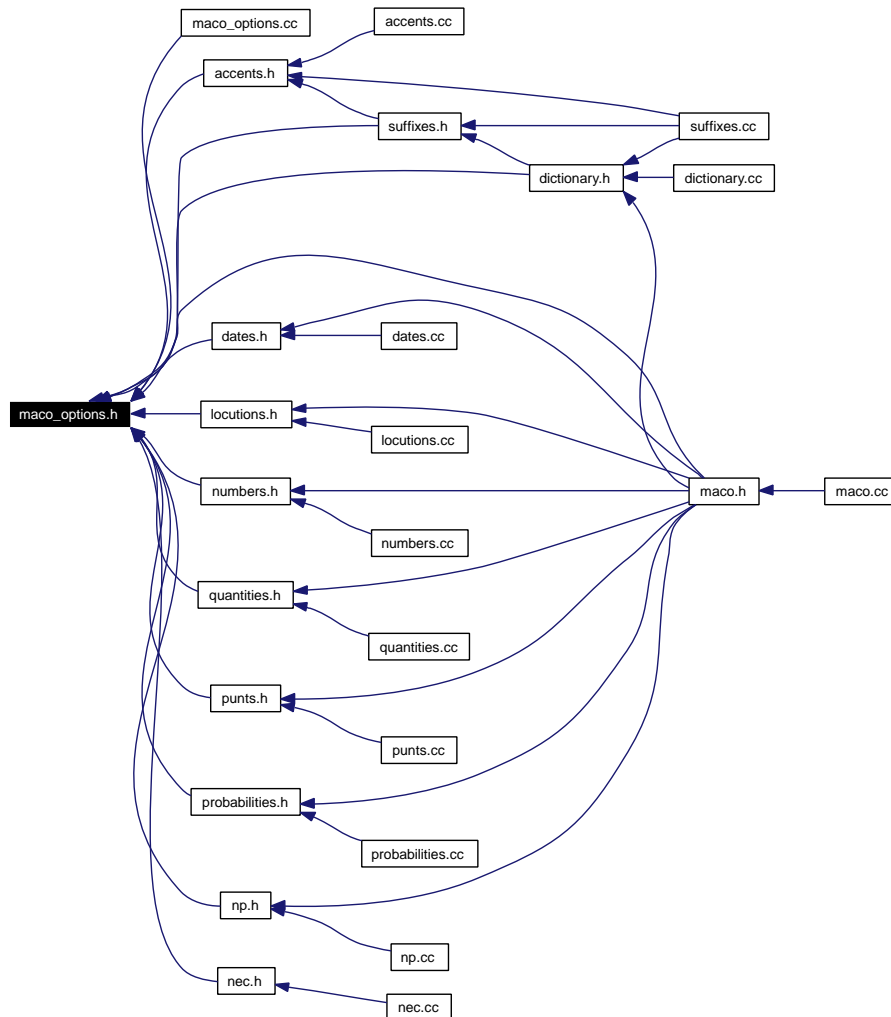
## 9.35 maco\_options.h File Reference

```
#include <string>
```

Include dependency graph for maco\_options.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class **maco\_options**

*Class maco\_options implements a set of specific options of the morphological analyzer.*

## 9.36 nec.cc File Reference

```
#include <fstream>
```

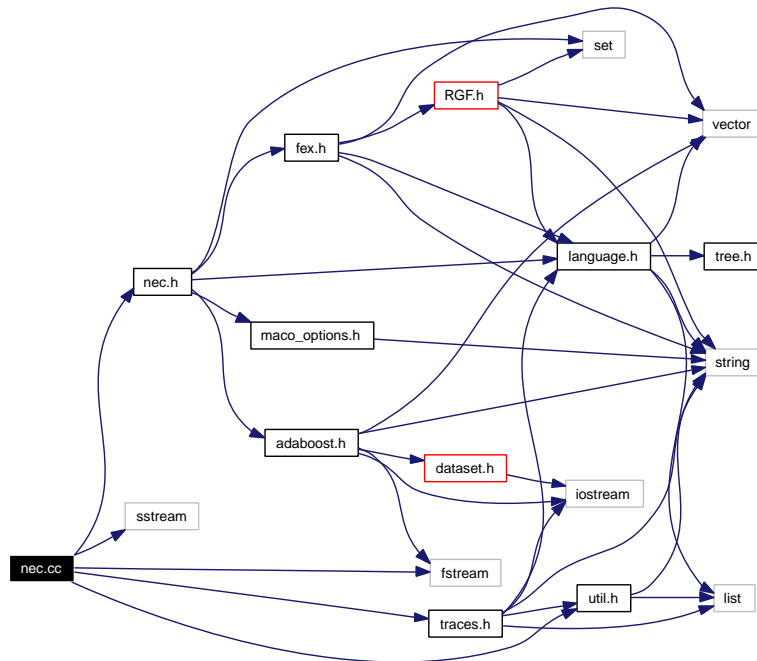
```
#include <sstream>
```

```
#include "nec.h"
```

```
#include "util.h"
```

```
#include "traces.h"
```

Include dependency graph for nec.cc:



### Defines

- `#define MOD_TRACENAME "NEC"`
- `#define MOD_TRACECODE NEC_TRACE`

### 9.36.1 Define Documentation

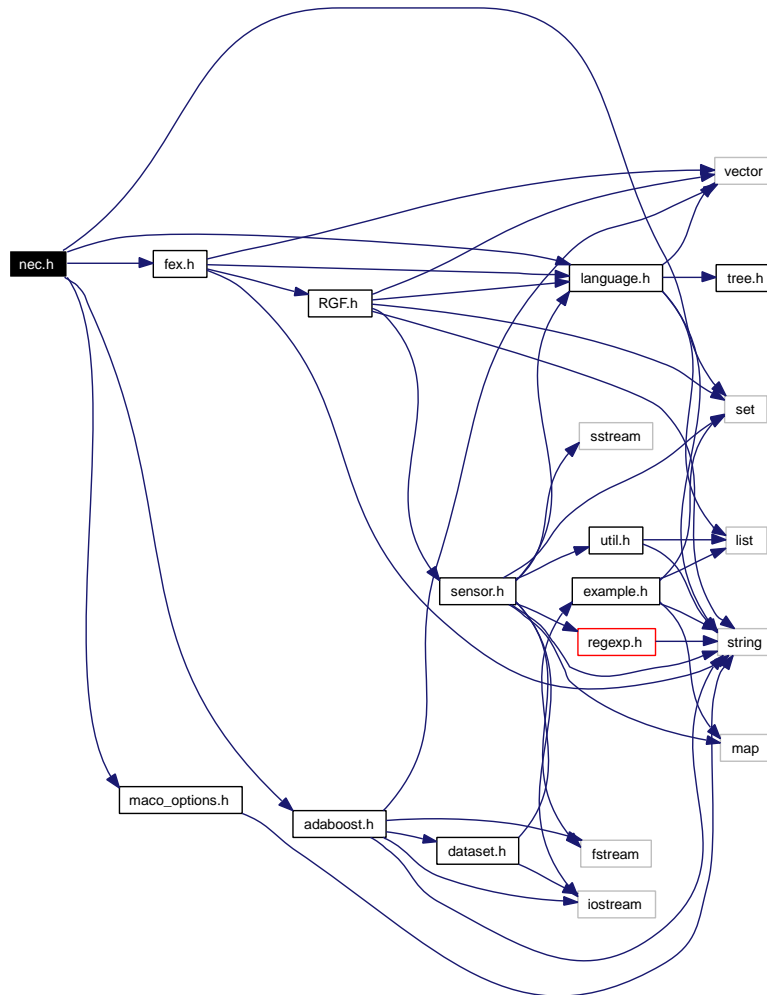
9.36.1.1 `#define MOD_TRACECODE NEC_TRACE`

9.36.1.2 `#define MOD_TRACENAME "NEC"`

## 9.37 nec.h File Reference

```
#include <set>
#include "language.h"
#include "maco_options.h"
#include "fex.h"
#include "adaboost.h"
```

Include dependency graph for nec.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class **nec**

*The class nec implements a ML-based NE classifier.*

## 9.38 np.cc File Reference

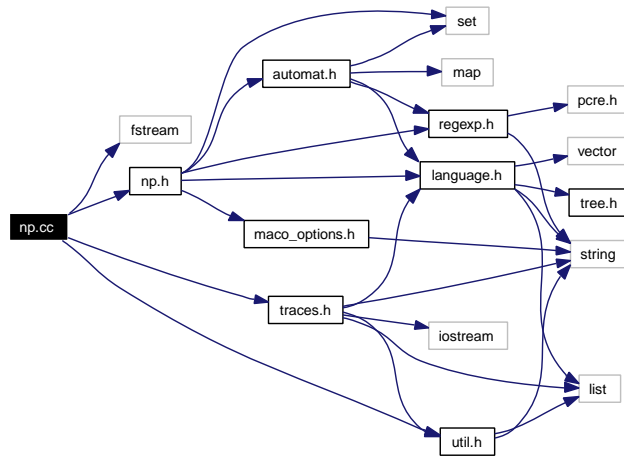
```
#include <fstream>
```

```
#include "np.h"
```

```
#include "util.h"
```

```
#include "traces.h"
```

Include dependency graph for np.cc:



## Defines

- `#define MOD_TRACENAME "NP"`
- `#define MOD_TRACECODE NP_TRACE`
- `#define IN 1`
- `#define NP 2`
- `#define FUN 3`
- `#define STOP 4`
- `#define TK_sUnkUpp 1`
- `#define TK_sNounUpp 2`
- `#define TK_mUpper 3`
- `#define TK_mFun 4`
- `#define TK_other 5`

## 9.38.1 Define Documentation

9.38.1.1 `#define FUN 3`

9.38.1.2 `#define IN 1`

9.38.1.3 `#define MOD_TRACECODE NP_TRACE`

9.38.1.4 `#define MOD_TRACENAME "NP"`

9.38.1.5 `#define NP 2`

9.38.1.6 `#define STOP 4`

9.38.1.7 `#define TK_mFun 4`

9.38.1.8 `#define TK_mUpper 3`

9.38.1.9 `#define TK_other 5`

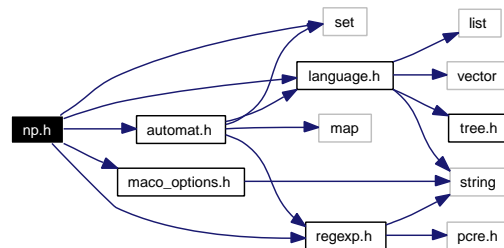
9.38.1.10 `#define TK_sNounUpp 2`

9.38.1.11 `#define TK_sUnkUpp 1`

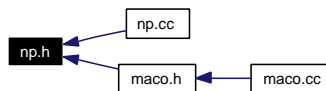
## 9.39 np.h File Reference

```
#include <set>
#include "language.h"
#include "automat.h"
#include "maco_options.h"
#include "regexp.h"
```

Include dependency graph for np.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **np**

*The class np implements a dummy proper noun recognizer.*

### Defines

- `#define RE_NA "^(NC|AQ)"`
- `#define RE_DNP "^[FWZ]"`
- `#define RE_CLO "^[DSC]"`

#### 9.39.1 Define Documentation

**9.39.1.1** `#define RE_CLO "^[DSC]"`

**9.39.1.2** `#define RE_DNP "^[FWZ]"`

**9.39.1.3** `#define RE_NA "^(NC|AQ)"`



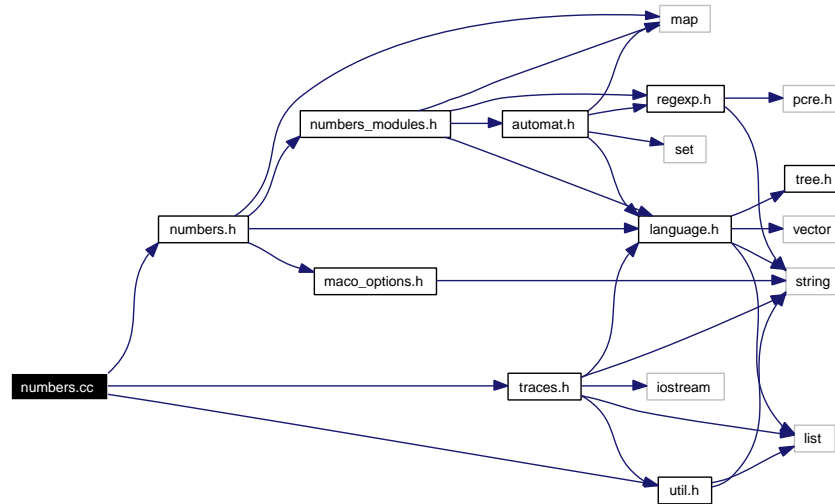
## 9.40 numbers.cc File Reference

```
#include "numbers.h"
```

```
#include "util.h"
```

```
#include "traces.h"
```

Include dependency graph for numbers.cc:



### Defines

- `#define MOD_TRACENAME "NUMBERS"`
- `#define MOD_TRACECODE NUMBERS_TRACE`

#### 9.40.1 Define Documentation

9.40.1.1 `#define MOD_TRACECODE NUMBERS_TRACE`

9.40.1.2 `#define MOD_TRACENAME "NUMBERS"`

## 9.41 numbers.h File Reference

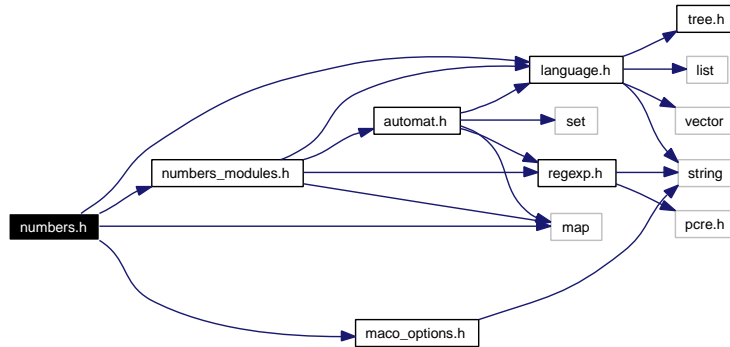
```
#include <map>
```

```
#include "language.h"
```

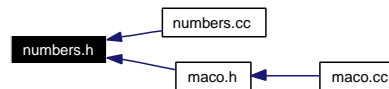
```
#include "maco_options.h"
```

```
#include "numbers_modules.h"
```

Include dependency graph for numbers.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class **numbers**

*Class numbers implements a wrapper to transparently create and access a **numbers\_module**(p. 171) number recognizer for the appropriate language.*

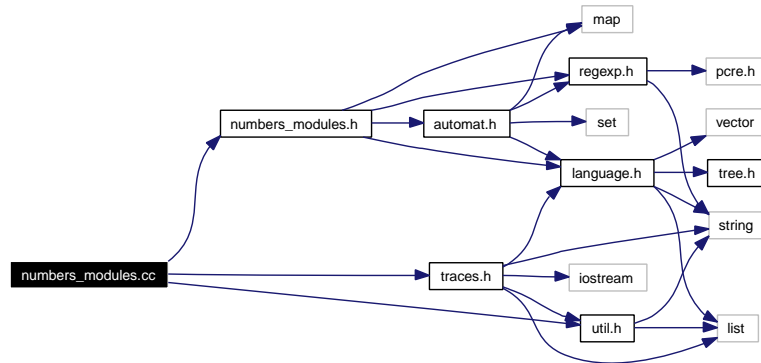
## 9.42 numbers\_modules.cc File Reference

```
#include "numbers_modules.h"
```

```
#include "util.h"
```

```
#include "traces.h"
```

Include dependency graph for numbers\_modules.cc:



### Defines

- #define **MOD\_TRACENAME** "NUMBERS"
- #define **MOD\_TRACECODE** NUMBERS\_TRACE
- #define **A** 1
- #define **NUM** 2
- #define **COD** 3
- #define **STOP** 4
- #define **TK\_num** 1
- #define **TK\_code** 2
- #define **TK\_other** 3
- #define **B1** 1
- #define **B2** 2
- #define **B3** 3
- #define **B4** 4
- #define **Bu** 5
- #define **B5** 6
- #define **B6** 7
- #define **B7** 8
- #define **B8** 9
- #define **Bk** 10
- #define **M1** 11
- #define **M1a** 12
- #define **M1b** 13
- #define **M2** 14
- #define **M3** 15
- #define **M4** 16
- #define **Mu** 17

- `#define M5` 18
- `#define M6` 19
- `#define M7` 20
- `#define M8` 21
- `#define Mk` 22
- `#define S1` 23
- `#define S1a` 24
- `#define S1b` 25
- `#define S2` 26
- `#define S3` 27
- `#define S4` 28
- `#define Su` 29
- `#define S5` 30
- `#define S6` 31
- `#define S7` 32
- `#define S8` 33
- `#define Sk` 34
- `#define COD` 35
- `#define X1` 36
- `#define X2` 37
- `#define X3` 38
- `#define X4` 39
- `#define X5` 40
- `#define STOP` 41
- `#define TK_c` 1
- `#define TK_d` 2
- `#define TK_u` 3
- `#define TK_wy` 4
- `#define TK_wmedio` 5
- `#define TK_wcuarto` 6
- `#define TK_mil` 7
- `#define TK_mill` 8
- `#define TK_bill` 9
- `#define TK_num` 10
- `#define TK_code` 11
- `#define TK_milr` 12
- `#define TK_cent` 13
- `#define TK_dec` 14
- `#define TK_doc` 15
- `#define TK_other` 16
- `#define B1` 1
- `#define B2` 2
- `#define B3` 3
- `#define B4` 4
- `#define Bu` 5
- `#define B5` 6
- `#define B6` 7
- `#define B7` 8
- `#define B8` 9
- `#define Bk` 10

- `#define M1 11`
- `#define M1a 12`
- `#define M1b 13`
- `#define M2 14`
- `#define M3 15`
- `#define M4 16`
- `#define Mu 17`
- `#define M5 18`
- `#define M6 19`
- `#define M7 20`
- `#define M8 21`
- `#define Mk 22`
- `#define S1 23`
- `#define S1a 24`
- `#define S1b 25`
- `#define S2 26`
- `#define S3 27`
- `#define S4 28`
- `#define Su 29`
- `#define S5 30`
- `#define S6 31`
- `#define S7 32`
- `#define S8 33`
- `#define Sk 34`
- `#define COD 35`
- `#define STOP 36`
- `#define TK_c 1`
- `#define TK_d 2`
- `#define TK_u 3`
- `#define TK_wy 4`
- `#define TK_wmedio 5`
- `#define TK_wcuarto 6`
- `#define TK_mil 7`
- `#define TK_mill 8`
- `#define TK_bill 9`
- `#define TK_num 10`
- `#define TK_code 11`
- `#define TK_other 12`
- `#define B1 1`
- `#define B2 2`
- `#define B3 3`
- `#define B4 4`
- `#define Bu 5`
- `#define B5 6`
- `#define B6 7`
- `#define B7 8`
- `#define B8 9`
- `#define Bk 10`
- `#define M1 11`
- `#define M1a 12`

- `#define M1b` 13
- `#define M2` 14
- `#define M3` 15
- `#define M4` 16
- `#define Mu` 17
- `#define M5` 18
- `#define M6` 19
- `#define M7` 20
- `#define M8` 21
- `#define Mk` 22
- `#define S1` 23
- `#define S1a` 24
- `#define S1b` 25
- `#define S2` 26
- `#define S3` 27
- `#define S4` 28
- `#define Su` 29
- `#define S5` 30
- `#define S6` 31
- `#define S7` 32
- `#define S8` 33
- `#define Sk` 34
- `#define COD` 35
- `#define X1` 36
- `#define X2` 37
- `#define X3` 38
- `#define X4` 39
- `#define X5` 40
- `#define STOP` 41
- `#define TK_c` 1
- `#define TK_d` 2
- `#define TK_u` 3
- `#define TK_wy` 4
- `#define TK_wmedio` 5
- `#define TK_wcuarto` 6
- `#define TK_mil` 7
- `#define TK_mill` 8
- `#define TK_bill` 9
- `#define TK_num` 10
- `#define TK_code` 11
- `#define TK_milr` 12
- `#define TK_cent` 13
- `#define TK_dec` 14
- `#define TK_doc` 15
- `#define TK_other` 16
- `#define B1` 1
- `#define B2` 2
- `#define B3` 3
- `#define B4` 4
- `#define B5` 5

- `#define B6 6`
- `#define B7 7`
- `#define B8 8`
- `#define M1 9`
- `#define M2 10`
- `#define M3 11`
- `#define M4 12`
- `#define M5 13`
- `#define M6 14`
- `#define M7 15`
- `#define M8 16`
- `#define S1 17`
- `#define S2 18`
- `#define S3 19`
- `#define S4 20`
- `#define S5 21`
- `#define S6 22`
- `#define S7 23`
- `#define S8 24`
- `#define COD 25`
- `#define STOP 26`
- `#define TK_a 1`
- `#define TK_u 1`
- `#define TK_hundr 2`
- `#define TK_thous 3`
- `#define TK_mill 4`
- `#define TK_bill 5`
- `#define TK_num 6`
- `#define TK_code 7`
- `#define TK_other 8`

## 9.42.1 Define Documentation

9.42.1.1 `#define A 1`

9.42.1.2 `#define B1 1`

9.42.1.3 `#define B1 1`

9.42.1.4 `#define B1 1`

9.42.1.5 `#define B1 1`

9.42.1.6 `#define B2 2`

9.42.1.7 `#define B2 2`

9.42.1.8 `#define B2 2`

9.42.1.9 `#define B2 2`

9.42.1.10 `#define B3 3`

9.42.1.11 `#define B3 3`

9.42.1.12 `#define B3 3`

9.42.1.13 `#define B3 3`

9.42.1.14 `#define B4 4`

9.42.1.15 `#define B4 4`

9.42.1.16 `#define B4 4`

9.42.1.17 `#define B4 4`

9.42.1.18 `#define B5 5`

9.42.1.19 `#define B5 6`

9.42.1.20 `#define B5 6`

9.42.1.21 `#define B5 6`

9.42.1.22 `#define B6 6`

9.42.1.23 `#define B6 7`

9.42.1.24 `#define B6 7`

9.42.1.25 `#define B6 7`

9.42.1.26 `#define B7 7`

9.42.1.27 `#define B7 8`

9.42.1.28 `#define B7 8`

9.42.1.29 `#define B7 8`

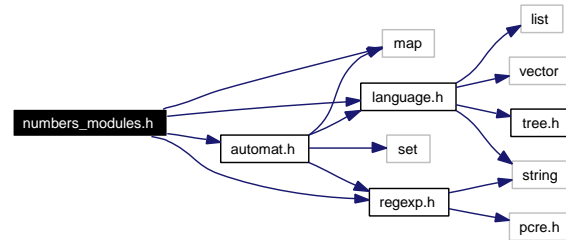
9.42.1.30 `#define B8 8`



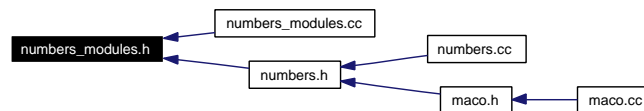
## 9.43 numbers\_modules.h File Reference

```
#include <map>
#include "language.h"
#include "automat.h"
#include "regexp.h"
```

Include dependency graph for numbers\_modules.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class **numbers\_module**

*The abstract class `numbers_module` generalizes numeric expression recognizer for different languages.*

- class **numbers\_default**

*The derived class `numbers_default` implements a default number recognizer (only numbers in digits are recognized).*

- class **numbers\_es**

*The derived class `numbers_es` implements a Spanish number recognizer.*

- class **numbers\_ca**

*The derived class `numbers_ca` implements a Catalan number recognizer.*

- class **numbers\_gl**

*The derived class `numbers_ca(p.161)` implements a Galician number recognizer.*

- class **numbers\_en**

*The derived class `numbers_en` implements an English number recognizer.*

## Defines

- `#define RE_NUM "^[0-9]+\\\"+MACO_Thousand+\"*[0-9]+(\\\"+MACO_Decimal+\")?[0-9]*$"`
- `#define RE_CODE "^[0-9]*$"`

### 9.43.1 Define Documentation

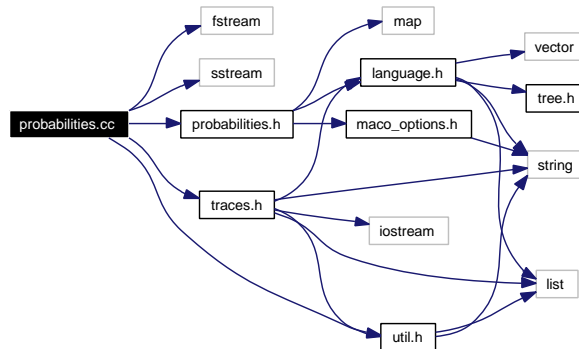
**9.43.1.1** `#define RE_CODE "^[0-9]*$"`

**9.43.1.2** `#define RE_NUM "^[0-9]+\\\"+MACO_Thousand+\"*[0-9]+(\\\"+MACO_Decimal+\")?[0-9]*$"`

## 9.44 probabilities.cc File Reference

```
#include <fstream>
#include <sstream>
#include "probabilities.h"
#include "util.h"
#include "traces.h"
```

Include dependency graph for probabilities.cc:



### Defines

- `#define MOD_TRACENAME "PROBABILITIES"`
- `#define MOD_TRACECODE PROB_TRACE`

#### 9.44.1 Define Documentation

9.44.1.1 `#define MOD_TRACECODE PROB_TRACE`

9.44.1.2 `#define MOD_TRACENAME "PROBABILITIES"`

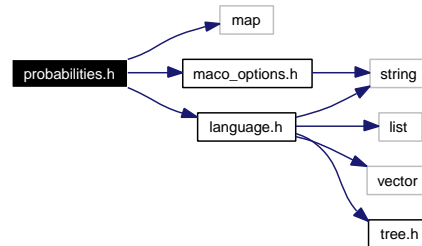
## 9.45 probabilities.h File Reference

```
#include <map>
```

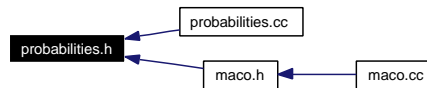
```
#include "maco_options.h"
```

```
#include "language.h"
```

Include dependency graph for probabilities.h:



This graph shows which files directly or indirectly include this file:



## Classes

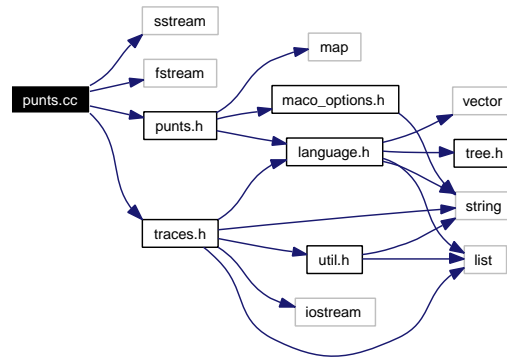
- class **probabilities**

*Class probabilities sets lexical probabilities for each PoS tag of each word in a sentence.*

## 9.46 punts.cc File Reference

```
#include <sstream>
#include <fstream>
#include "punts.h"
#include "traces.h"
```

Include dependency graph for punts.cc:



### Defines

- `#define MOD_TRACENAME "PUNCTUATION"`
- `#define MOD_TRACECODE PUNCT_TRACE`

#### 9.46.1 Define Documentation

9.46.1.1 `#define MOD_TRACECODE PUNCT_TRACE`

9.46.1.2 `#define MOD_TRACENAME "PUNCTUATION"`

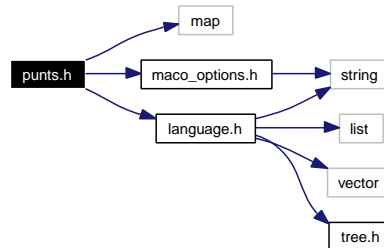
## 9.47 punts.h File Reference

```
#include <map>
```

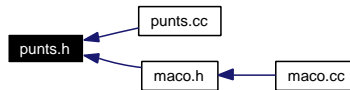
```
#include "maco_options.h"
```

```
#include "language.h"
```

Include dependency graph for punts.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **punts**  
*Class numbers implements a punctuation sign recognizer.*

### Defines

- `#define OTHER "<Other>"`

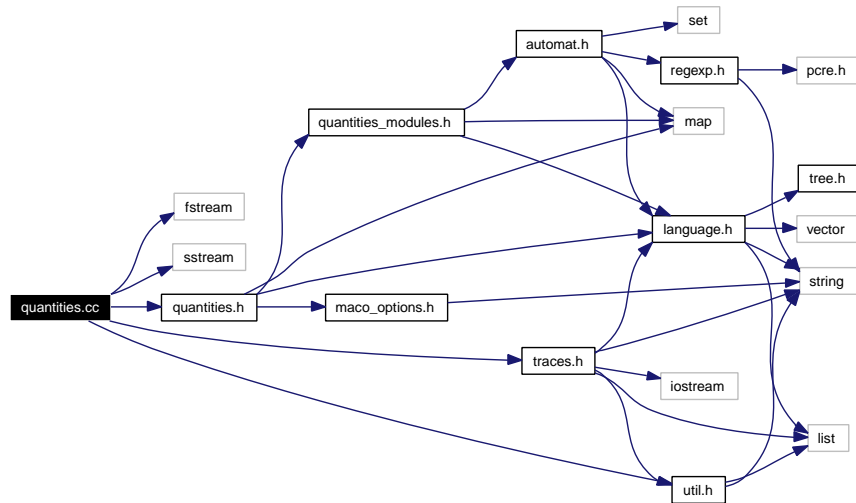
#### 9.47.1 Define Documentation

##### 9.47.1.1 `#define OTHER "<Other>"`

## 9.48 quantities.cc File Reference

```
#include <fstream>
#include <sstream>
#include "quantities.h"
#include "util.h"
#include "traces.h"
```

Include dependency graph for quantities.cc:



### Defines

- `#define MOD_TRACENAME "QUANTITIES"`
- `#define MOD_TRACECODE QUANT_TRACE`

### 9.48.1 Define Documentation

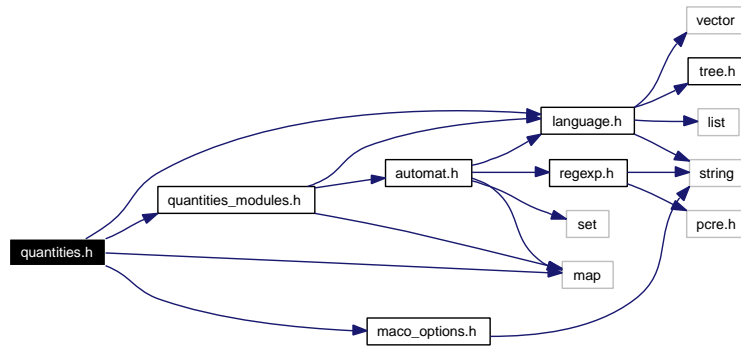
9.48.1.1 `#define MOD_TRACECODE QUANT_TRACE`

9.48.1.2 `#define MOD_TRACENAME "QUANTITIES"`

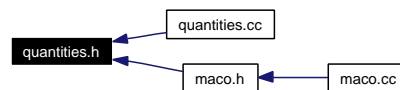
## 9.49 quantities.h File Reference

```
#include <map>
#include "maco_options.h"
#include "language.h"
#include "quantities_modules.h"
```

Include dependency graph for quantities.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class **quantities**

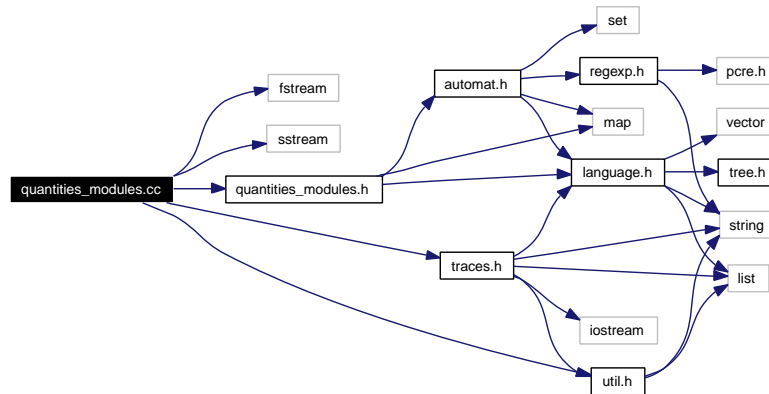
*Class `quantities` implements a wrapper to transparently create and access a `quantities_module(p.191)` monetary expressions detector for the appropriate language.*



## 9.50 quantities\_\_modules.cc File Reference

```
#include <fstream>
#include <sstream>
#include "quantities_modules.h"
#include "util.h"
#include "traces.h"
```

Include dependency graph for quantities\_\_modules.cc:



### Defines

- #define MOD\_TRACENAME "QUANTITIES"
- #define MOD\_TRACECODE QUANT\_TRACE
- #define A 1
- #define B 2
- #define C 3
- #define STOP 4
- #define TK\_number 1
- #define TK\_pc 2
- #define TK\_other 3
- #define A 1
- #define B 2
- #define C 3
- #define D 4
- #define E 5
- #define F 6
- #define G 7
- #define H 8
- #define I 9
- #define STOP 10
- #define TK\_number 1
- #define TK\_n100 2
- #define TK\_pc 3
- #define TK\_wde 4

- `#define TK_wcada 5`
- `#define TK_wpor 6`
- `#define TK_wsobre 7`
- `#define TK_avo 8`
- `#define TK_part 9`
- `#define TK_curr 10`
- `#define TK_country 11`
- `#define TK_unit 12`
- `#define TK_other 13`

## 9.50.1 Define Documentation

9.50.1.1 `#define A 1`

9.50.1.2 `#define A 1`

9.50.1.3 `#define B 2`

9.50.1.4 `#define B 2`

9.50.1.5 `#define C 3`

9.50.1.6 `#define C 3`

9.50.1.7 `#define D 4`

9.50.1.8 `#define E 5`

9.50.1.9 `#define F 6`

9.50.1.10 `#define G 7`

9.50.1.11 `#define H 8`

9.50.1.12 `#define I 9`

9.50.1.13 `#define MOD_TRACECODE QUANT_TRACE`

9.50.1.14 `#define MOD_TRACENAME "QUANTITIES"`

9.50.1.15 `#define STOP 10`

9.50.1.16 `#define STOP 4`

9.50.1.17 `#define TK_avo 8`

9.50.1.18 `#define TK_country 11`

9.50.1.19 `#define TK_curr 10`

9.50.1.20 `#define TK_n100 2`

9.50.1.21 `#define TK_number 1`

9.50.1.22 `#define TK_number 1`

9.50.1.23 `#define TK_other 13`

9.50.1.24 `#define TK_other 3`

9.50.1.25 `#define TK_part 9`

9.50.1.26 `#define TK_pc 3`

9.50.1.27 `#define TK_pc 2`

---

Generated on Wed Apr 26 12:55:30 2006 for FreeLing by Doxygen

9.50.1.28 `#define TK_unit 12`

9.50.1.29 `#define TK_wcada 5`

9.50.1.30 `#define TK_wde 4`

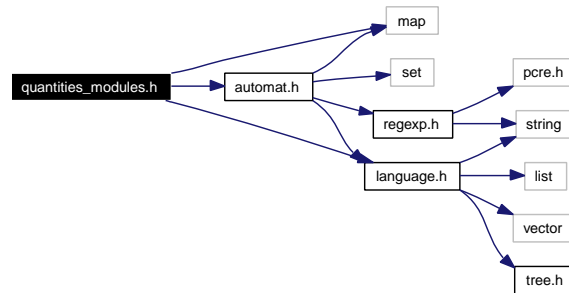
## 9.51 quantities\_modules.h File Reference

```
#include <map>
```

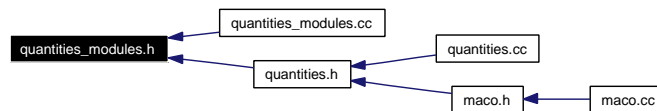
```
#include "automat.h"
```

```
#include "language.h"
```

Include dependency graph for quantities\_modules.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class **quantities\_module**

*The abstract class `quantities_module` generalizes a percentage, ratios, and currency expression recognizer for different languages.*

- class **quantities\_default**

*The derived class `quantities_default` implements a default quantities recognizer (only percentages are recognized).*

- class **quantities\_es**

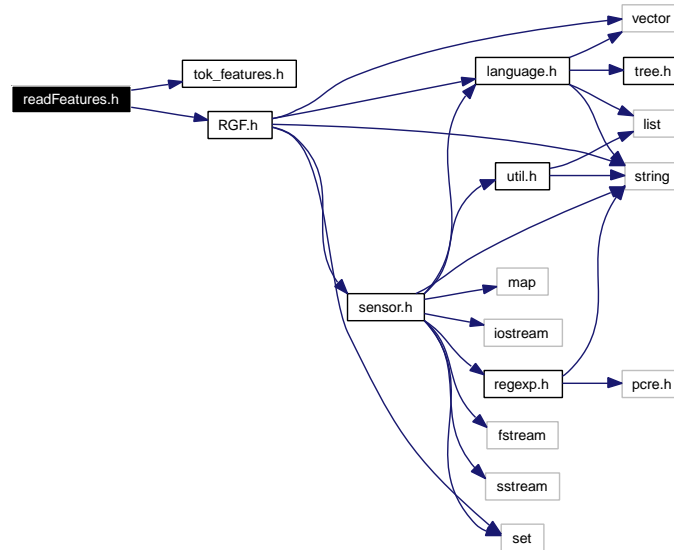
*The derived class `quantities_es` implements a Spanish (and Catalan) quantities recognizer.*

## 9.52 readFeatures.h File Reference

```
#include "tok_features.h"
```

```
#include "RGF.h"
```

Include dependency graph for readFeatures.h:



### Enumerations

- enum {  
     T\_COLOC, T\_LABEL, T\_NOT, T\_CONJ,  
     T\_SCOLOC, T\_DISJ, T\_LINK }

### Functions

- SubRGF \* DoParse (const char \*Scriptfile)

#### 9.52.1 Enumeration Type Documentation

##### 9.52.1.1 anonymous enum

Enumerator:

```

T_COLOC
T_LABEL
T_NOT
T_CONJ
T_SCOLOC
T_DISJ
T_LINK

```

## 9.52.2 Function Documentation

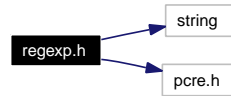
### 9.52.2.1 SubRGF\* DoParse (const char \* *Scriptfile*)

## 9.53 regexp.h File Reference

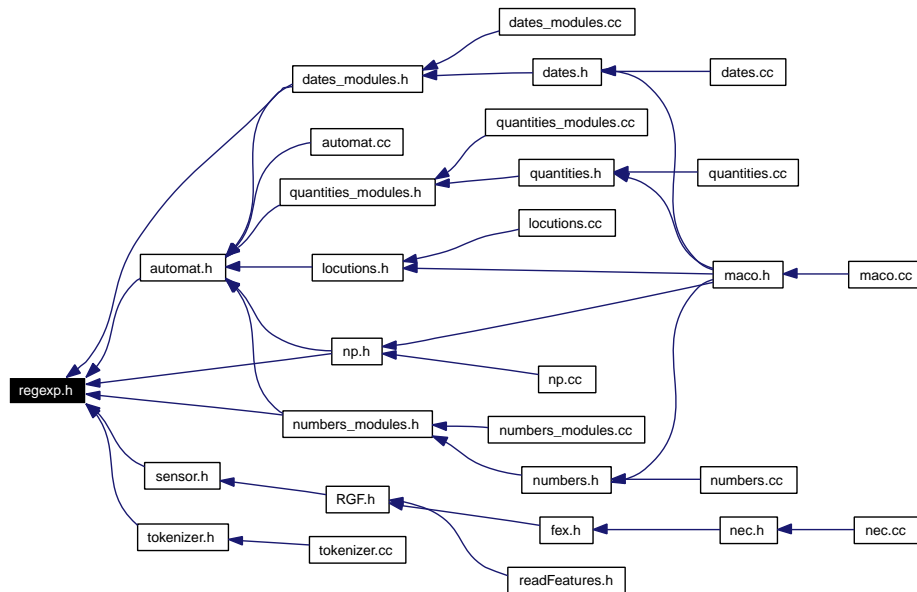
```
#include <string>
```

```
#include "pcre.h"
```

Include dependency graph for regexp.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **RegEx**

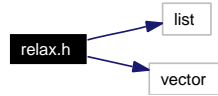
*class RegEx, a simple and small API wrapper for PCRE.*

## 9.54 relax.h File Reference

```
#include <list>
```

```
#include <vector>
```

Include dependency graph for relax.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **problem**

*The class problem stores the structure of a problem, namely, a vector with a position for each variable to consider, and for each variable, a list of initial weights for each possible label.*

- class **label**

*The class label stores all information related to a variable label in the relaxation labelling algorithm.*

- class **constraint**

*The class constraint implements a constraint for the relaxation labelling algorithm.*

- class **relax**

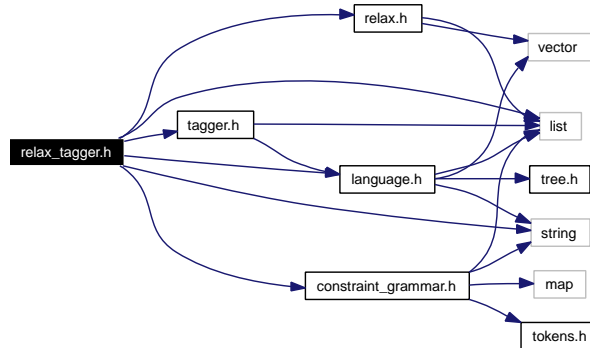
*The class relax implements a generic solver for consistent labelling problems, using relaxation labelling algorithm.*



## 9.55 relax\_tagger.h File Reference

```
#include <list>
#include <string>
#include "language.h"
#include "tagger.h"
#include "relax.h"
#include "constraint_grammar.h"
```

Include dependency graph for relax\_tagger.h:



### Classes

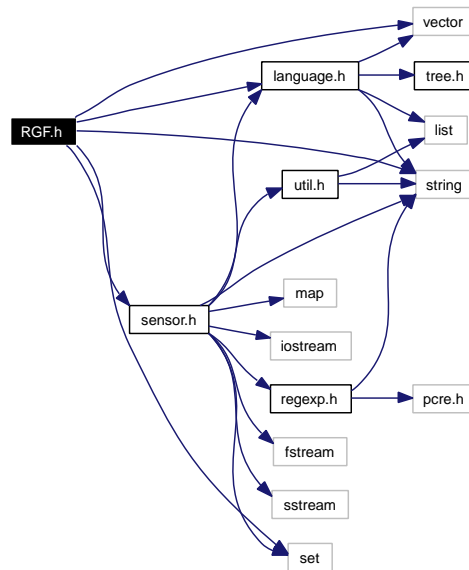
- class `relax_tagger`

*The class `relax_tagger` implements a PoS tagger based on relaxation labelling algorithm.*

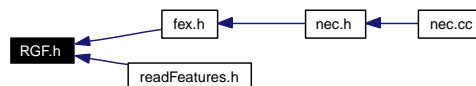
## 9.56 RGF.h File Reference

```
#include <string>
#include <vector>
#include <set>
#include "language.h"
#include "sensor.h"
```

Include dependency graph for RGF.h:



This graph shows which files directly or indirectly include this file:



### Classes

- struct **RGF**

### Typedefs

- typedef `vector< RGF >` **SubRGF**

### Enumerations

- enum **ExtractMode** {  
**EXTRACT\_LABEL**, **EXTRACT\_CONJ**, **EXTRACT\_DISJ**, **EXTRACT\_COLOC**,

- EXTRACT\_SCOLOC,        EXTRACT\_LINK,        EXTRACT\_SENSOR,  
EXTRACT\_NOT,  
EXTRACT\_CONJUNCT }
- enum TargetConstants { TARGET\_NULL = -2, TARGET\_ALL = -1 }

## Variables

- const int RANGE\_ALL = 1000000

### 9.56.1 Typedef Documentation

#### 9.56.1.1 typedef vector<RGF> SubRGF

### 9.56.2 Enumeration Type Documentation

#### 9.56.2.1 enum ExtractMode

Enumerator:

*EXTRACT\_LABEL*  
*EXTRACT\_CONJ*  
*EXTRACT\_DISJ*  
*EXTRACT\_COLOC*  
*EXTRACT\_SCOLOC*  
*EXTRACT\_LINK*  
*EXTRACT\_SENSOR*  
*EXTRACT\_NOT*  
*EXTRACT\_CONJUNCT*

#### 9.56.2.2 enum TargetConstants

Enumerator:

*TARGET\_NULL*  
*TARGET\_ALL*

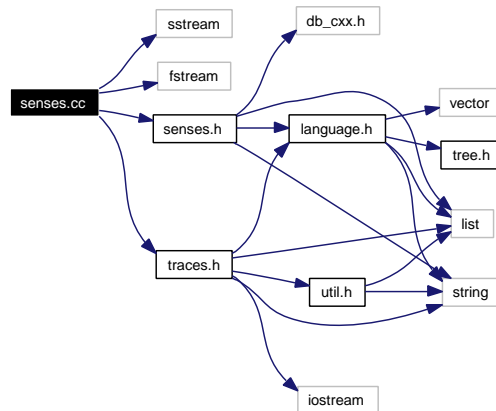
### 9.56.3 Variable Documentation

#### 9.56.3.1 const int RANGE\_ALL = 1000000

## 9.57 senses.cc File Reference

```
#include <sstream>
#include <fstream>
#include "senses.h"
#include "traces.h"
```

Include dependency graph for senses.cc:



### Defines

- `#define MOD_TRACENAME "SENSES"`
- `#define MOD_TRACECODE SENSES_TRACE`

### 9.57.1 Define Documentation

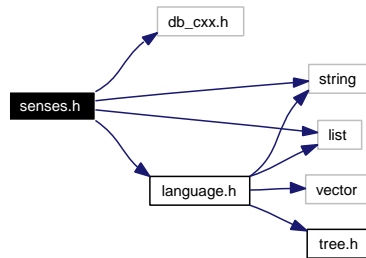
9.57.1.1 `#define MOD_TRACECODE SENSES_TRACE`

9.57.1.2 `#define MOD_TRACENAME "SENSES"`

## 9.58 senses.h File Reference

```
#include <db_cxx.h>
#include <string>
#include <list>
#include "language.h"
```

Include dependency graph for senses.h:



This graph shows which files directly or indirectly include this file:



### Classes

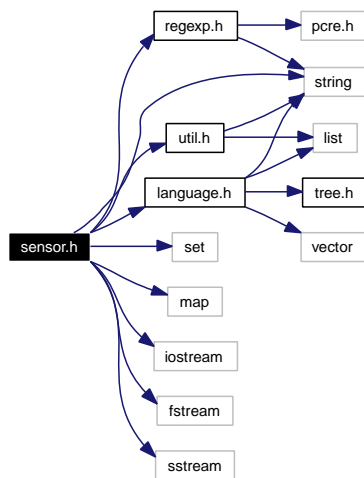
- class **senses**

*Class senses implements a sense annotator.*

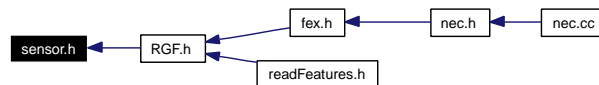
## 9.59 sensor.h File Reference

```
#include "language.h"
#include "regex.h"
#include "util.h"
#include <string>
#include <set>
#include <map>
#include <iostream>
#include <fstream>
#include <sstream>
```

Include dependency graph for sensor.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class **Sensor**
- class **SensorData**
- class **SensorMap**
- class **SensorSet**
- class **SensorSetPart**
- class **SensorMatchRE**
- class **SensorCheckMwRE**
- class **SensorCheckRE**

## Enumerations

- enum `SensorType` { `ST_WORD`, `ST_PHRASE` }

### 9.59.1 Enumeration Type Documentation

#### 9.59.1.1 enum `SensorType`

Enumerator:

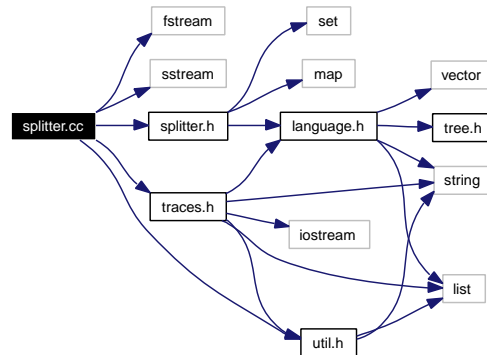
*`ST_WORD`*

*`ST_PHRASE`*

## 9.60 splitter.cc File Reference

```
#include <fstream>
#include <sstream>
#include "splitter.h"
#include "util.h"
#include "traces.h"
```

Include dependency graph for splitter.cc:



### Defines

- `#define MOD_TRACENAME "SPLITTER"`
- `#define MOD_TRACECODE SPLIT_TRACE`
- `#define SAME 1000`

#### 9.60.1 Define Documentation

9.60.1.1 `#define MOD_TRACECODE SPLIT_TRACE`

9.60.1.2 `#define MOD_TRACENAME "SPLITTER"`

9.60.1.3 `#define SAME 1000`



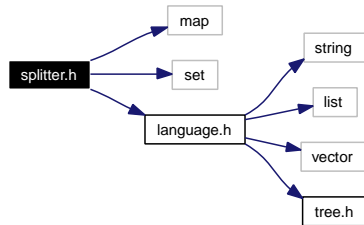
## 9.61 splitter.h File Reference

```
#include <map>
```

```
#include <set>
```

```
#include "language.h"
```

Include dependency graph for splitter.h:



This graph shows which files directly or indirectly include this file:



### Classes

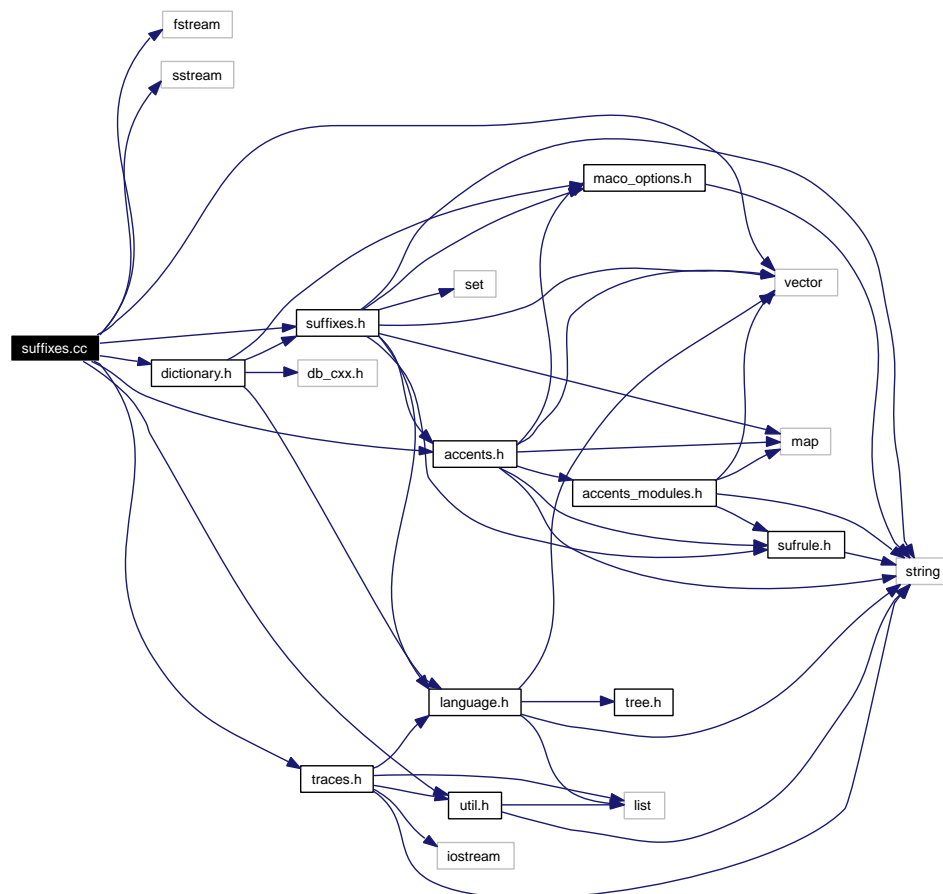
- class **splitter**

*Class `splitter` implements a sentence splitter, which accumulates lists of words until a sentence is completed, and then returns a list of sentence objects.*

## 9.62 suffixes.cc File Reference

```
#include <fstream>
#include <sstream>
#include <vector>
#include "suffixes.h"
#include "accents.h"
#include "dictionary.h"
#include "util.h"
#include "traces.h"
```

Include dependency graph for suffixes.cc:



### Defines

- `#define MOD_TRACENAME "SUFFIXES"`
- `#define MOD_TRACECODE SUFF_TRACE`

## 9.62.1 Define Documentation

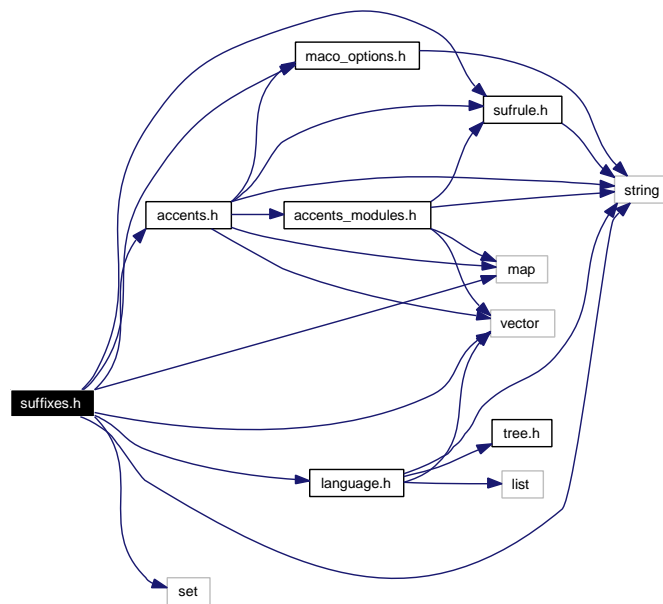
9.62.1.1 `#define MOD_TRACECODE SUFF_TRACE`

9.62.1.2 `#define MOD_TRACENAME "SUFFIXES"`

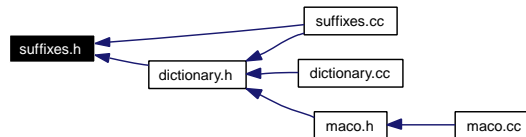
## 9.63 suffixes.h File Reference

```
#include <string>
#include <set>
#include <map>
#include <vector>
#include "maco_options.h"
#include "language.h"
#include "sufrule.h"
#include "accents.h"
#include "accents_modules.h"
```

Include dependency graph for suffixes.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class **suffixes**

*Class `suffixes` implements suffixation rules and dictionary search for suffixed word forms.*

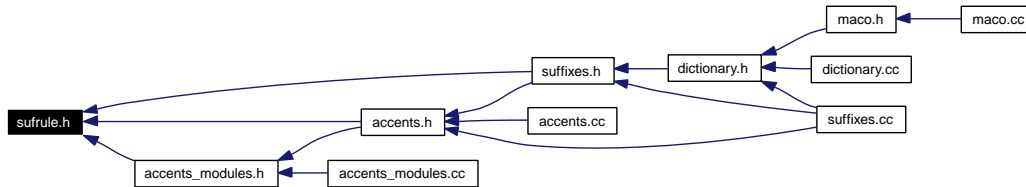
## 9.64 sufrule.h File Reference

```
#include <string>
```

Include dependency graph for sufrule.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **sufrule**

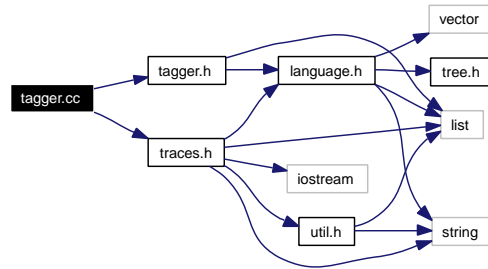
*Class **sufrule** contains a suffixation rule, and is used by class **suffixes**.*

## 9.65 tagger.cc File Reference

```
#include "tagger.h"
```

```
#include "traces.h"
```

Include dependency graph for tagger.cc:



### Defines

- `#define MOD_TRACE_NAME "TAGGER"`
- `#define MOD_TRACE_CODE TAGGER_TRACE`

#### 9.65.1 Define Documentation

9.65.1.1 `#define MOD_TRACE_CODE TAGGER_TRACE`

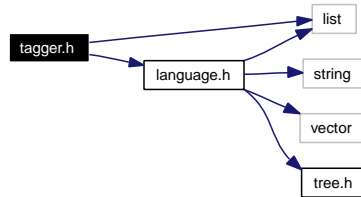
9.65.1.2 `#define MOD_TRACE_NAME "TAGGER"`

## 9.66 tagger.h File Reference

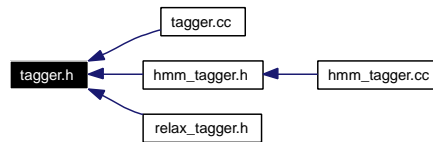
```
#include <list>
```

```
#include "language.h"
```

Include dependency graph for tagger.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class **POS\_tagger**

*The class `POS_tagger` is just an abstract class generalizing a `PoS tagger`.*

## 9.67 tok\_features.h File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- `#define SENSOR` 258
- `#define COMPRGF` 259
- `#define CONJUNCT` 260
- `#define WORD` 261
- `#define FLAG` 262
- `#define TARG` 263
- `#define INT` 264
- `#define COLON` 265
- `#define LBRACK` 266
- `#define RBRACK` 267
- `#define LPAREN` 268
- `#define RPAREN` 269
- `#define NEWLINE` 270
- `#define COMMA` 271
- `#define AMP` 272
- `#define BAR` 273
- `#define SMALLX` 274
- `#define BIGX` 275
- `#define EQUALS` 276
- `#define SEMICOLON` 277
- `#define FLOC` 278
- `#define FINC` 279
- `#define FMARK` 280
- `#define ERROR` 281
- `#define QUOTE` 282
- `#define YYSTYPE YYSTYPE`
- `#define YYSTYPE_IS_DECLARED` 1
- `#define YYSTYPE_IS_TRIVIAL` 1

### Typedefs

- `typedef int YYSTYPE`

### Enumerations

- `enum yytokentype` {  
`SENSOR` = 258, `COMPRGF` = 259, `CONJUNCT` = 260, `WORD` = 261,  
`FLAG` = 262, `TARG` = 263, `INT` = 264, `COLON` = 265,  
`LBRACK` = 266, `RBRACK` = 267, `LPAREN` = 268, `RPAREN` = 269,



**NEWLINE** = 270, **COMMA** = 271, **AMP** = 272, **BAR** = 273,

**SMALLX** = 274, **BIGX** = 275, **EQUALS** = 276, **SEMICOLON** = 277,

**FLOC** = 278, **FINC** = 279, **FMARK** = 280, **ERROR** = 281,

**QUOTE** = 282 }

## Variables

- **YYSTYPE** yylval

## 9.67.1 Define Documentation

9.67.1.1 `#define AMP` 272

9.67.1.2 `#define BAR` 273

9.67.1.3 `#define BIGX` 275

9.67.1.4 `#define COLON` 265

9.67.1.5 `#define COMMA` 271

9.67.1.6 `#define COMPRGF` 259

9.67.1.7 `#define CONJUNCT` 260

9.67.1.8 `#define EQUALS` 276

9.67.1.9 `#define ERROR` 281

9.67.1.10 `#define FINC` 279

9.67.1.11 `#define FLAG` 262

9.67.1.12 `#define FLOC` 278

9.67.1.13 `#define FMARK` 280

9.67.1.14 `#define INT` 264

9.67.1.15 `#define LBRACK` 266

9.67.1.16 `#define LPAREN` 268

9.67.1.17 `#define NEWLINE` 270

9.67.1.18 `#define QUOTE` 282

9.67.1.19 `#define RBRACK` 267

9.67.1.20 `#define RPAREN` 269

9.67.1.21 `#define SEMICOLON` 277

9.67.1.22 `#define SENSOR` 258

9.67.1.23 `#define SMALLX` 274

9.67.1.24 `#define TARG` 263

9.67.1.25 `#define WORD` 261

9.67.1.26 `#define yystype YYSTYPE`

9.67.1.27 `#define YYSTYPE_IS_DECLARED` 1

---

Generated on Wed Apr 26 12:55:30 2006 for FreeLing by Doxygen

9.67.1.28 `#define YYSTYPE_IS_TRIVIAL` 1

## 9.67.2 Typedef Documentation

9.67.2.1 `typedef int YYSTYPE`

*COMPRGF*  
*CONJUNCT*  
*WORD*  
*FLAG*  
*TARG*  
*INT*  
*COLON*  
*LBRACK*  
*RBRACK*  
*LPAREN*  
*RPAREN*  
*NEWLINE*  
*COMMA*  
*AMP*  
*BAR*  
*SMALLX*  
*BIGX*  
*EQUALS*  
*SEMICOLON*  
*FLOC*  
*FINC*  
*FMARK*  
*ERROR*  
*QUOTE*

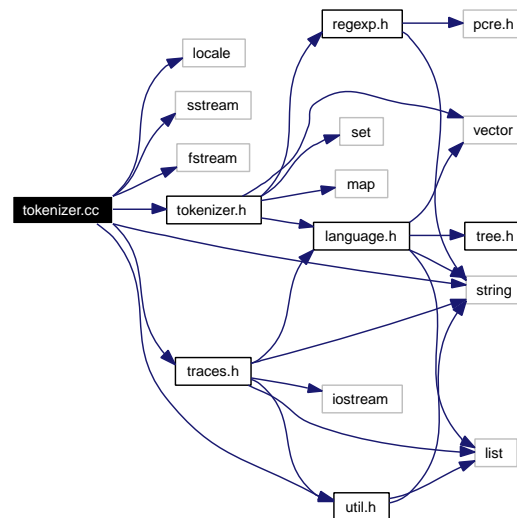
## 9.67.4 Variable Documentation

### 9.67.4.1 YYSTYPE yylval

## 9.68 tokenizer.cc File Reference

```
#include <locale>
#include <sstream>
#include <fstream>
#include <string>
#include "tokenizer.h"
#include "util.h"
#include "traces.h"
```

Include dependency graph for tokenizer.cc:



### Defines

- `#define MOD_TRACENAME "TOKENIZER"`
- `#define MOD_TRACECODE TOKEN_TRACE`

#### 9.68.1 Define Documentation

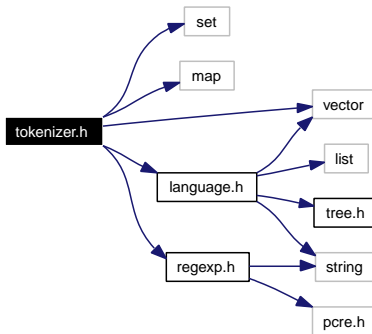
9.68.1.1 `#define MOD_TRACECODE TOKEN_TRACE`

9.68.1.2 `#define MOD_TRACENAME "TOKENIZER"`

## 9.69 tokenizer.h File Reference

```
#include <set>
#include <map>
#include <vector>
#include "language.h"
#include "regexp.h"
```

Include dependency graph for tokenizer.h:



This graph shows which files directly or indirectly include this file:



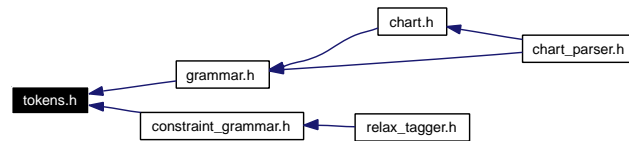
## Classes

- class **tokenizer**

*Class tokenizer implements a token splitter, which converts a string into a sequence of word objects, according to a set of tokenization rules read from a configuration file.*

## 9.70 tokens.h File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- `#define CATEGORY 1`  
*Token definition for flex scanners for parser grammar files and constraint grammar files.*
- `#define FORM 2`
- `#define LEMMA 3`
- `#define COMMENT 4`
- `#define ERROR 5`
- `#define ARROW 6`
- `#define BAR 7`
- `#define COMMA 8`
- `#define DOT 9`
- `#define FLAT 10`
- `#define HIDDEN 11`
- `#define NOTOP 12`
- `#define ONLYTOP 13`
- `#define PRIOR 14`
- `#define START 15`
- `#define FILENAME 16`
- `#define HEAD 17`
- `#define BARRIER 18`
- `#define CPAR 19`
- `#define FLOAT 20`
- `#define NOT 21`
- `#define OR 22`
- `#define OPAR 23`
- `#define OUT 24`
- `#define POSITION 25`
- `#define SEMICOLON 26`

### 9.70.1 Define Documentation

**9.70.1.1** `#define ARROW 6`

**9.70.1.2** `#define BAR 7`

**9.70.1.3** `#define BARRIER 18`

**9.70.1.4** `#define CATEGORY 1`

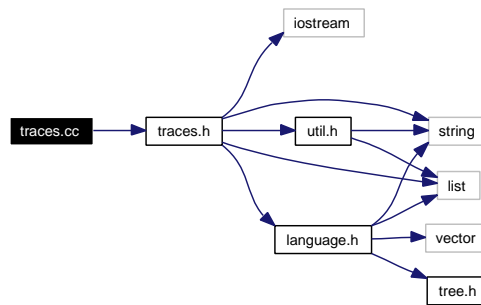
Token definition for flex scanners for parser grammar files and constraint grammar files.

- 9.70.1.5 `#define COMMA 8`
- 9.70.1.6 `#define COMMENT 4`
- 9.70.1.7 `#define CPAR 19`
- 9.70.1.8 `#define DOT 9`
- 9.70.1.9 `#define ERROR 5`
- 9.70.1.10 `#define FILENAME 16`
- 9.70.1.11 `#define FLAT 10`
- 9.70.1.12 `#define FLOAT 20`
- 9.70.1.13 `#define FORM 2`
- 9.70.1.14 `#define HEAD 17`
- 9.70.1.15 `#define HIDDEN 11`
- 9.70.1.16 `#define LEMMA 3`
- 9.70.1.17 `#define NOT 21`
- 9.70.1.18 `#define NOTOP 12`
- 9.70.1.19 `#define ONLYTOP 13`
- 9.70.1.20 `#define OPAR 23`
- 9.70.1.21 `#define OR 22`
- 9.70.1.22 `#define OUT 24`
- 9.70.1.23 `#define POSITION 25`
- 9.70.1.24 `#define PRIOR 14`
- 9.70.1.25 `#define SEMICOLON 26`
- 9.70.1.26 `#define START 15`

## 9.71 traces.cc File Reference

```
#include "traces.h"
```

Include dependency graph for traces.cc:

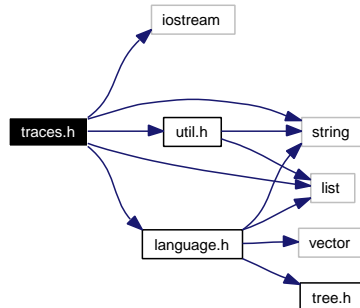




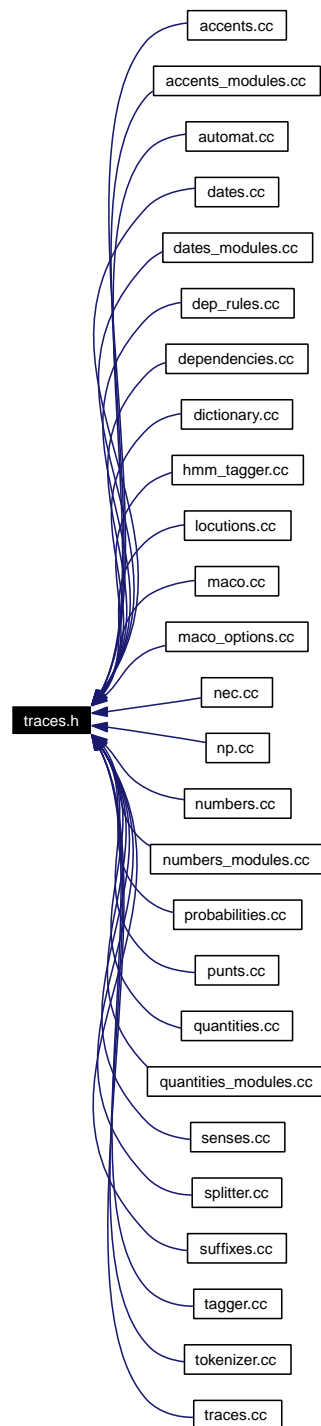
## 9.72 traces.h File Reference

```
#include <iostream>
#include <string>
#include <list>
#include "util.h"
#include "language.h"
```

Include dependency graph for traces.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class **traces**

*Class traces implements trace and error handling utilities.*

## Defines

- `#define SPLIT_TRACE 0x00000001`  
*possible values for MOD\_TRACECODE*
- `#define TOKEN_TRACE 0x00000002`
- `#define MACO_TRACE 0x00000004`
- `#define OPTIONS_TRACE 0x00000008`
- `#define NUMBERS_TRACE 0x00000010`
- `#define DATES_TRACE 0x00000020`
- `#define PUNCT_TRACE 0x00000040`
- `#define DICT_TRACE 0x00000080`
- `#define SUFF_TRACE 0x00000100`
- `#define LOCUT_TRACE 0x00000200`
- `#define NP_TRACE 0x00000400`
- `#define PROB_TRACE 0x00000800`
- `#define QUANT_TRACE 0x00001000`
- `#define NEC_TRACE 0x00002000`
- `#define AUTOMAT_TRACE 0x00004000`
- `#define TAGGER_TRACE 0x00008000`
- `#define HMM_TRACE 0x00010000`
- `#define RELAX_TRACE 0x00020000`
- `#define RELAX_TAGGER_TRACE 0x00040000`
- `#define CONST_GRAMMAR_TRACE 0x00080000`
- `#define SENSES_TRACE 0x00100000`
- `#define CHART_TRACE 0x00200000`
- `#define GRAMMAR_TRACE 0x00400000`
- `#define DEP_TRACE 0x00800000`
- `#define UTIL_TRACE 0x01000000`
- `#define ERROR_CRASH(msg) traces::error_crash(msg,MOD_TRACENAME,MOD_TRACECODE)`  
*Macros that must be used to put traces in the code.*
- `#define WARNING(msg) traces::warning(msg,MOD_TRACENAME,MOD_TRACECODE)`
- `#define TRACE(x, y)`  
*ifndef VERBOSE -> No messages displayed. Faster code.*
- `#define TRACE_WORD_LIST(x, y)`
- `#define TRACE_SENTENCE(x, y)`
- `#define TRACE_SENTENCE_LIST(x, y)`

## 9.72.1 Define Documentation

9.72.1.1 `#define AUTOMAT_TRACE 0x00004000`

9.72.1.2 `#define CHART_TRACE 0x00200000`

9.72.1.3 `#define CONST_GRAMMAR_TRACE 0x00080000`

9.72.1.4 `#define DATES_TRACE 0x00000020`

9.72.1.5 `#define DEP_TRACE 0x00800000`

9.72.1.6 `#define DICT_TRACE 0x00000080`

9.72.1.7 `#define ERROR_CRASH(msg) traces::error_crash(msg,MOD_-  
TRACENAME,MOD_TRACECODE)`

Macros that must be used to put traces in the code.

They may be either defined or null, depending on -DVERBOSE compilation flag.

9.72.1.8 `#define GRAMMAR_TRACE 0x00400000`

9.72.1.9 `#define HMM_TRACE 0x00010000`

9.72.1.10 `#define LOCUT_TRACE 0x00000200`

9.72.1.11 `#define MACO_TRACE 0x00000004`

9.72.1.12 `#define NEC_TRACE 0x00002000`

9.72.1.13 `#define NP_TRACE 0x00000400`

9.72.1.14 `#define NUMBERS_TRACE 0x00000010`

9.72.1.15 `#define OPTIONS_TRACE 0x00000008`

9.72.1.16 `#define PROB_TRACE 0x00000800`

9.72.1.17 `#define PUNCT_TRACE 0x00000040`

9.72.1.18 `#define QUANT_TRACE 0x00001000`

9.72.1.19 `#define RELAX_TAGGER_TRACE 0x00040000`

9.72.1.20 `#define RELAX_TRACE 0x00020000`

9.72.1.21 `#define SENSES_TRACE 0x00100000`

9.72.1.22 `#define SPLIT_TRACE 0x00000001`

possible values for MOD\_TRACECODE

**9.72.1.23** `#define SUFF_TRACE 0x00000100`

**9.72.1.24** `#define TAGGER_TRACE 0x00008000`

**9.72.1.25** `#define TOKEN_TRACE 0x00000002`

**9.72.1.26** `#define TRACE(x, y)`

`ifndef VERBOSE` -> No messages displayed. Faster code.

Compile `*without* -D VERBOSE` to get a faster, non-traceable, exploitation version

**9.72.1.27** `#define TRACE_SENTENCE(x, y)`

**9.72.1.28** `#define TRACE_SENTENCE_LIST(x, y)`

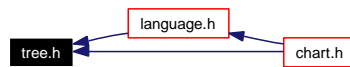
**9.72.1.29** `#define TRACE_WORD_LIST(x, y)`

**9.72.1.30** `#define UTIL_TRACE 0x01000000`

**9.72.1.31** `#define WARNING(msg) traces::warning(msg,MOD_ -  
TRACENAME,MOD_TRACECODE)`

## 9.73 tree.h File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- class **tree**< **T** >
- class **tree**< **T** >::**generic\_iterator**
- class **tree**< **T** >::**preorder\_iterator**  
*traverse the tree in preorder (parent first, then children)*
- class **tree**< **T** >::**sibling\_iterator**  
*traverse all children of the same node*

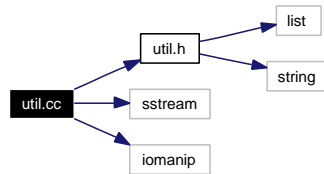
## 9.74 util.cc File Reference

```
#include "util.h"
```

```
#include <sstream>
```

```
#include <iomanip>
```

Include dependency graph for util.cc:

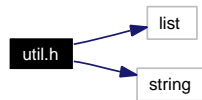


## 9.75 util.h File Reference

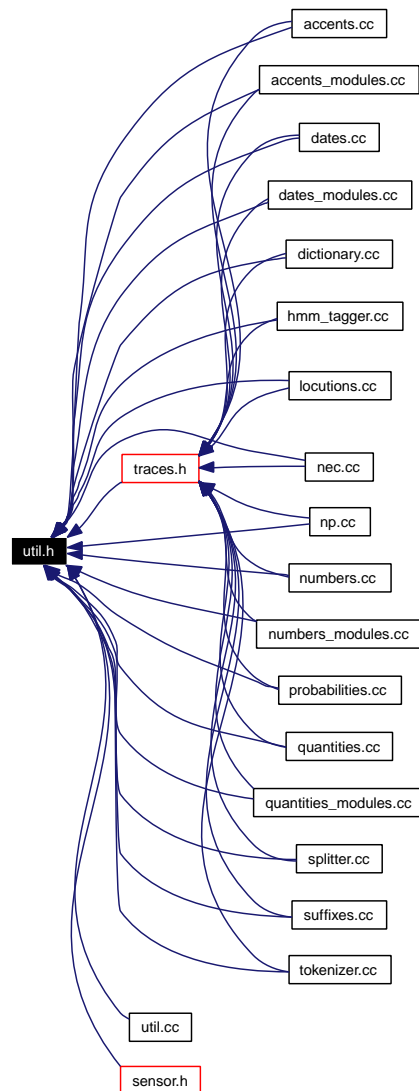
```
#include <list>
```

```
#include <string>
```

Include dependency graph for util.h:



This graph shows which files directly or indirectly include this file:





## Classes

- class `util`

*Class util implements some utilities for NLP analyzers: "tolower" for latin alphabets, parole tags manipulation, string2number and viceversa conversions, etc.*