RegEX Generator

Generated by Doxygen 1.8.1.1

Tue May 31 2016 22:53:18

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 EX_UTIL__ Namespace Reference

Functions

- string removeSubstrs__ (string &)
- bool removeSubstrs (string &, const string &)
- void ReplaceStringInPlace (string &subject, const string &search, const string &replace)
- string trim (const string &str, const string &whitespace=" ")
- string brak (string str)
- void exception (string message, bool ext=false)

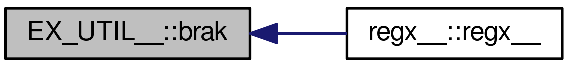
4.1.1 Detailed Description

simple exception handling.

4.1.2 Function Documentation

4.1.2.1 string EX_UTIL__::brak (string *str*)

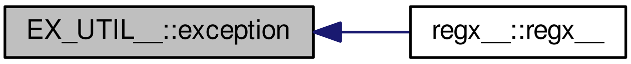
Here is the caller graph for this function:



4.1.2.2 void EX_UTIL__::exception (string *message*, bool *ext* = `false`)

setting ext to **true** exits program. Default value (**false**) only issues console message

Here is the caller graph for this function:



4.1.2.3 bool EX_UTIL__::removeSubstrs (string & *s,* const string & *p*)

Utility function to remove substring `const string` from a given `string`

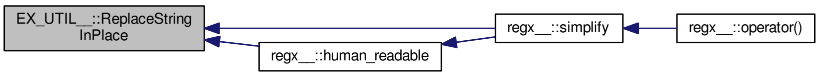
4.1.2.4 string EX_UTIL__::removeSubstrs__ (string & *s*)

Utility function to remove substring `const string` from a given `string` returns string

4.1.2.5 void EX_UTIL__::ReplaceStringInPlace (string & *subject,* const string & *search,* const string & *replace*)

Utility function to replace all occurrences of substring `const string` **search** with `const string` **replace** from a given `string` subject

Here is the caller graph for this function:



4.1.2.6 string EX_UTIL__::trim (const string & *str,* const string & *whitespace* = " ")

Utility function to remove leading and trailing whitespace substring `consist string` **whitespace** from a given `string`. The white space can be given to be any string, default is **whitespace**

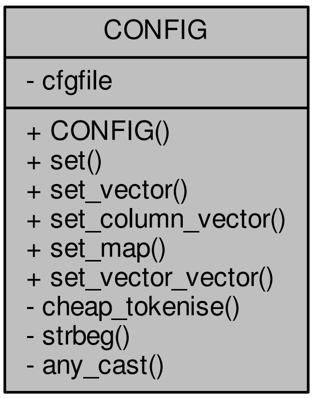
Chapter 5

Class Documentation

5.1 CONFIG Class Reference

`#include <config.h>`

Collaboration diagram for CONFIG:



Public Member Functions

- CONFIG (string)
- template<class T >
  bool set (T &Varname, string Fname)
- template<class T >
  bool set_vector (vector< T > &Varname, string Fname)
- template<class T >
  bool set_column_vector (vector< T > &Varname, string Fname)
- template<class Ti , class Tj , class T >
  bool set_map (map< Ti, map< Tj, T > > &Varname, string Fname)
- template<class Tj >
  bool set_vector_vector (vector< vector< Tj > > &Varname, string Fname)

Private Member Functions

- vector< string > cheap_tokenise (string const &)
- bool strbeg (string, string)
- template<class T >
  bool any_cast (T &t, const std::string &s)

Private Attributes

- string cfgfile

5.1.1 Constructor & Destructor Documentation

5.1.1.1 CONFIG::CONFIG (string *s*)

5.1.2 Member Function Documentation

5.1.2.1 template< class T > bool CONFIG::any_cast (T & *t*, const std::string & *s*) `[inline]`, `[private]`

5.1.2.2 vector< string > CONFIG::cheap_tokenise (string const & *input*) `[private]`

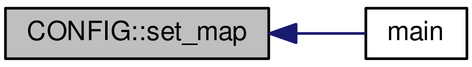
5.1.2.3 template< class T > bool CONFIG::set (T & *Varname*, string *Fname*) `[inline]`

The following map is platform and compiler dependent There is apparently no portable way of getting unmangled names from typeid

5.1.2.4 template< class T > bool CONFIG::set_column_vector (vector< T > & *Varname*, string *Fname*) `[inline]`

5.1.2.5 template< class Ti , class Tj , class T > bool CONFIG::set_map (map< Ti, map< Tj, T > > & *Varname*, string *Fname*) `[inline]`

Here is the caller graph for this function:



5.1.2.6 template< class T > bool CONFIG::set_vector (vector< T > & *Varname*, string *Fname*) `[inline]`

5.1.2.7 template< class Tj > bool CONFIG::set_vector_vector (vector< vector< Tj > > & *Varname*, string *Fname*) `[inline]`

5.1.2.8 bool CONFIG::strbeg (string *s*, string *frag*) `[private]`

5.1.3 Member Data Documentation

5.1.3.1 string CONFIG::cfgfile [private]

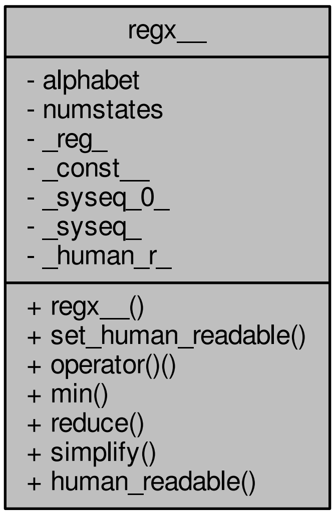
The documentation for this class was generated from the following files:

- config.h
- config.cc

5.2 regx__ Class Reference

```
#include <regx.h>
```

Collaboration diagram for regx__:



Public Member Functions

- regx__ (connx aut)
- void set_human_readable (vector< string >)
- string & operator() (state q)
- string & min (state q)
- void reduce (state)
- string & simplify (string &)
- string & human_readable (string &str)

Private Attributes

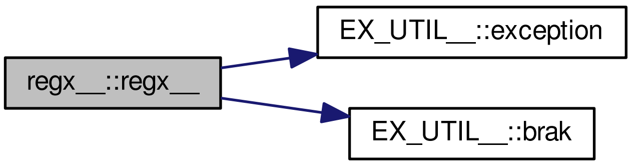
- unsigned int alphabet
- unsigned int numstates
- map< state, string > _reg_
- map_str _const__
- matrix_str _syseq_0_

- matrix_str _syseq_
- map< string, string > _human_r_

5.2.1 Constructor & Destructor Documentation

5.2.1.1 regx__::regx__ (connx *aut*)

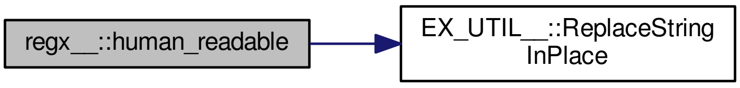
Here is the call graph for this function:



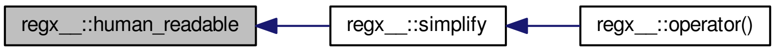
5.2.2 Member Function Documentation

5.2.2.1 string & regx__::human_readable (string & *str*)

Here is the call graph for this function:



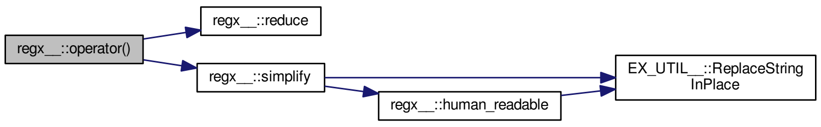
Here is the caller graph for this function:



5.2.2.2 string& regx__::min (state *q*)

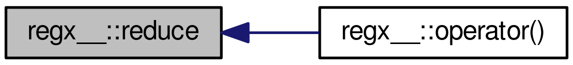
5.2.2.3 string & regx__::operator() (state *q*)

Here is the call graph for this function:



5.2.2.4 void regx__::reduce (state *q*)

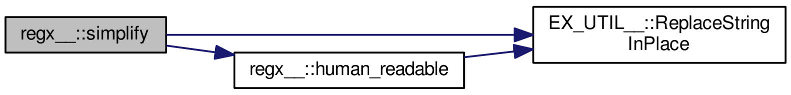
Here is the caller graph for this function:



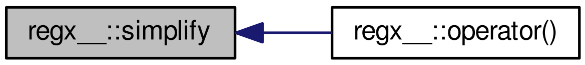
5.2.2.5 void regx__::set_human_readable (vector< string > *vecstr*)

5.2.2.6 string & regx__::simplify (string & *str_*)

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3 Member Data Documentation

5.2.3.1 map_str regx__::_const__ [private]

5.2.3.2 map<string,string> regx__::_human_r_ [private]

5.2.3.3 map<state,string> regx__::_reg_ [private]

5.2.3.4 matrix_str regx__::_syseq_ [private]

5.2.3.5 matrix_str regx__::_syseq_0_ [private]

5.2.3.6 unsigned int regx__::alphabet [private]

5.2.3.7 unsigned int regx__::numstates [private]

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

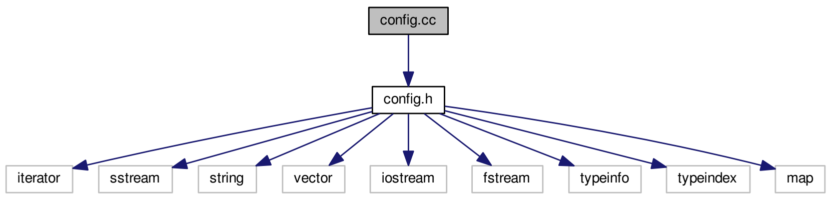
- regx.h
- regx.cc

Chapter 6

File Documentation

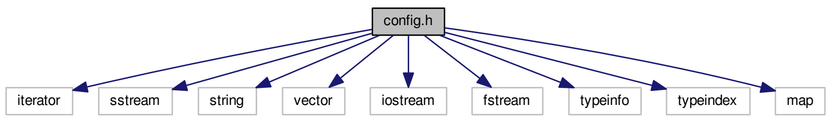
6.1 config.cc File Reference

```
#include "config.h"
```
Include dependency graph for config.cc:

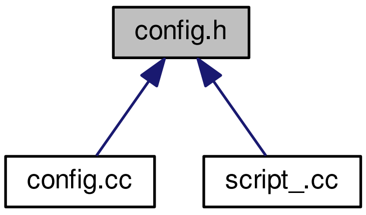


6.2 config.h File Reference

```
#include <iterator>
#include <sstream>
#include <string>
#include <vector>
#include <iostream>
#include <fstream>
#include <typeinfo>
#include <typeindex>
#include <map>
```
Include dependency graph for config.h:



This graph shows which files directly or indirectly include this file:



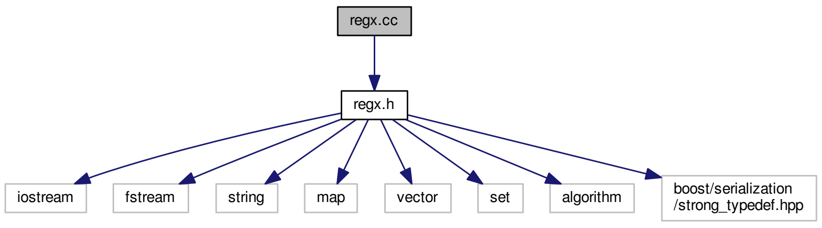
Classes

- class CONFIG

6.3 regx.cc File Reference

```
#include "regx.h"
```

Include dependency graph for regx.cc:



Functions

- ostream & operator<< (ostream &out, map_str &s)
- ostream & operator<< (ostream &out, matrix_str &s)

Variables

- const string _EMPTY_ = "_LAMBDA_"

6.3.1 Function Documentation

6.3.1.1 ostream& operator<< (ostream & *out*, map_str & *s*)

6.3.1.2 ostream& operator<< (ostream & *out*, matrix_str & *s*)

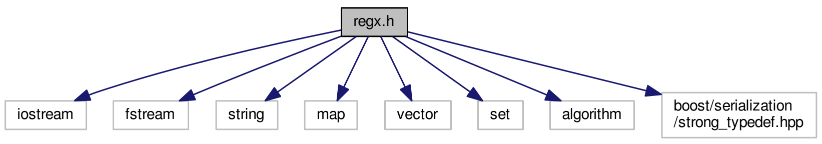
6.3.2 Variable Documentation

6.3.2.1 const string _EMPTY_ = "_LAMBDA_"

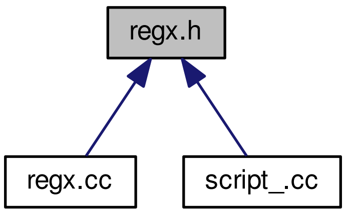
6.4 regx.h File Reference

```
#include <iostream>
#include <fstream>
#include <string>
#include <map>
#include <vector>
#include <set>
#include <algorithm>
#include <boost/serialization/strong_typedef.hpp>
```

Include dependency graph for regx.h:



This graph shows which files directly or indirectly include this file:



Classes

- class regx__

Namespaces

- namespace EX_UTIL__

Macros

- #define DEBUG_ 0

Typedefs

- typedef vector< symbol > symbol_list_
- typedef int state
- typedef std::map< state,
  vector< double > > pitilde
- typedef std::map< symbol, state > map_sym_state
- typedef std::map< state,
  map_sym_state > connx
- typedef std::map< unsigned int,
  map< unsigned int, double > > matrix_dbl
- typedef map< state, string > map_str
- typedef map< state, map_str > matrix_str

Functions

- BOOST_STRONG_TYPEDEF (unsigned int, symbol)
- string EX_UTIL__::removeSubstrs__ (string &)
- bool EX_UTIL__::removeSubstrs (string &, const string &)
- void EX_UTIL__::ReplaceStringInPlace (string &subject, const string &search, const string &replace)
- string EX_UTIL__::trim (const string &str, const string &whitespace=" ")
- string EX_UTIL__::brak (string str)
- void EX_UTIL__::exception (string message, bool ext=false)

6.4.1 Macro Definition Documentation

6.4.1.1 #define DEBUG_ 0

6.4.2 Typedef Documentation

6.4.2.1 typedef std::map< **state, map_sym_state** > **connx**

connx is the data-type for representing the underlying graph of a probabilistic automata. It is implemented as a `map` between `state` and `map_sym_state`. Each map element represents a row of the corresponding connection matrix, such that `connx var`[i][j] is the new `state` after symbol *j* is fired from `state` *i*.

6.4.2.2 typedef map<**state**,string> **map_str**

6.4.2.3 typedef std::map< symbol, **state** > **map_sym_state**

`map` between `symbol` and `state`. It represents a row of the connection matrix defining the graph of the probabilistic automata.

6.4.2.4 typedef std::map< unsigned int, map < unsigned int, double > > **matrix_dbl**

uiuidbl is the data-type for representing matrices with `double` entries. It is implemented as a `map` between `unsigned int` and a `map` between `unsigned int` and a `double`.

6.4.2.5 typedef map< state,map_str > matrix_str

6.4.2.6 typedef std::map< state, vector < double > > pitilde

$\widetilde{\Pi}$ is the morph matrix of dimension $|Q| \times |\Sigma|$, such that $\widetilde{\Pi}|_{ij}$ is the probability of generating symbol j from state i. It is implemented as a `map` between `state` and `vector` <double>. Each `vector` represents a row of the corresponding **stochastic** matrix, and hence must **sum to unity**.

6.4.2.7 typedef int state

6.4.2.8 typedef vector< symbol > symbol_list_

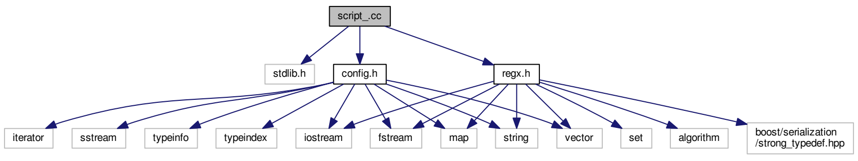
6.4.3 Function Documentation

6.4.3.1 BOOST_STRONG_TYPEDEF (unsigned *int*, symbol)

6.5 script_.cc File Reference

```
#include <stdlib.h>
#include "config.h"
#include "regx.h"
```

Include dependency graph for script_.cc:



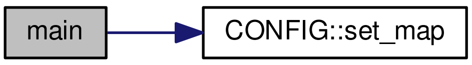
Functions

- int main (int argc, char *argv[])

6.5.1 Function Documentation

6.5.1.1 int main (int *argc*, char * *argv[]*)

Here is the call graph for this function:


main
CONFIG::set_map