Form Validation

Generated by Doxygen 1.8.3.1

Sat May 18 2013 16:49:59

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 FA::Component Class Reference

Class to represent a component (field) of a form.

`#include <Component.h>`

Public Member Functions

- Component (std::string type)
  Constructor.
- bool regex (std::string value)
  add eNFA described by regex to component
- bool db (std::string file)
  add database to regex
- bool db (std::string file, bool corrector)
  add database to regex
- bool ENFA (std::string file)
  add eNFA described by file to component
- std::string DBcorrector (std::string value)
  generates corrected version of input string
- bool **process** (std::string)
- std::string getType ()
  returns the type of the component

2.1.1 Detailed Description

Class to represent a component (field) of a form.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 FA::Component::Component (std::string *type*)

Constructor.

Parameters

type	type of the component

2.1.3 Member Function Documentation

2.1.3.1 bool FA::Component::db (std::string *file*)

add database to regex

Parameters

file	Name of the database file

Precondition

No database should be present

2.1.3.2 bool FA::Component::db (std::string *file*, bool *corrector*)

add database to regex

Parameters

file	Name of the database file
corrector	Indicates where input correction should be applied

Precondition

No database should be present

2.1.3.3 std::string FA::Component::DBcorrector (std::string *value*)

generates corrected version of input string

Parameters

value	the string to be corrected

Returns

the corrected string

2.1.3.4 bool FA::Component::ENFA (std::string *file*)

add eNFA described by file to component

Parameters

file	The filename

Precondition

No database should be present

2.1.3.5 std::string FA::Component::getType ( )

returns the type of the component

Returns

the type

2.1.3.6 bool FA::Component::regex (std::string *value*)

add eNFA described by regex to component

Parameters

value	the regex

Precondition

No database should be present

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

- /home/jakob/Dropbox/UA/workspace/FormValidation/src/Component.h
- /home/jakob/Dropbox/UA/workspace/FormValidation/src/Component.cpp

2.2 FA::DFA Class Reference

Class representing a DFA.

`#include <DFA.h>`

Public Member Functions

- DFA (alphabet, DFAstates, int)
  constructor
- DFAstates & getQ ()
  getter for Q (the states)
- DFAstate * getQ0 ()
  getter for q0 (start state)
- alphabet & getSigma ()
  getter for sigma (the alphabet)
- bool process (std::string)
  Check if string is part of the DFA.

Friends

- std::ostream & operator<< (std::ostream &, DFA &)
  << overloader for DFA

2.2.1 Detailed Description

Class representing a DFA.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 FA::DFA::DFA (alphabet *alphabet_*, DFAstates *states_*, int *start_*)

constructor

Parameters

*alphabet_*	The alphabet of the DFA
*states_*	The states of the DFA
*start_*	Number of start state

2.2.3 Member Function Documentation

2.2.3.1 DFAstates& FA::DFA::getQ ( ) [inline]

getter for Q (the states)

Returns

the states

2.2.3.2 DFAstate* FA::DFA::getQ0 ( ) [inline]

getter for q0 (start state)

Returns

the start state

2.2.3.3 alphabet& FA::DFA::getSigma ( ) [inline]

getter for sigma (the alphabet)

Returns

the alphabet

2.2.3.4 bool FA::DFA::process (std::string *str*)

Check if string is part of the DFA.

Parameters

str	The input string

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

- /home/jakob/Dropbox/UA/workspace/FormValidation/src/DFA.h

- /home/jakob/Dropbox/UA/workspace/FormValidation/src/DFA.cpp

2.3 FA::DFAstate Struct Reference

struct to represent state of DFA

`#include <DFA.h>`

Public Member Functions

- bool transition (char symbol, DFAstate *state)

  Checks if there is a transition from the state of a certain symbol.
- DFAstate * go (char symbol)

  returns pointer to target
- stateset makeTransitions (char symbol, transitions delta, eNFA automata)

  Finds target states for transition with certain symbol.
- bool corresponds (stateset checkSet)

  check whether the states in the stateset are also in this DFAstate
- bool isState (state *checkState)

  check whether a state is part of the DFAstate
- DFAstate (std::string name, bool accepting)

  constructor
- DFAstate (std::string name)

  constructor
- DFAstate (stateset states)

  Constructor.

Public Attributes

- std::string **label**
- std::map< char, DFAstate * > **DFAtransitions**
- bool **isAccepting**
- stateset **multiStates**

2.3.1 Detailed Description

struct to represent state of DFA

2.3.2 Constructor & Destructor Documentation

2.3.2.1 FA::DFAstate::DFAstate (std::string *name,* bool *accepting*) [inline]

constructor

Parameters

name	Name of the state
accepting	true if state is accepting state

2.3.2.2 FA::DFAstate::DFAstate (std::string *name*) [inline]

constructor

Parameters

name	Name of the state

2.3.2.3 FA::DFAstate::DFAstate (stateset *states*) [inline]

Constructor.

Parameters

states	States that belong to the DFAstate

2.3.3 Member Function Documentation

2.3.3.1 bool FA::DFAstate::corresponds (stateset *checkSet*) [inline]

check whether the states in the stateset are also in this DFAstate

Parameters

checkSet	the stateset to be checked

Returns

true if all states are in the DFAstate

2.3.3.2 bool FA::DFAstate::isState (state * *checkState*) [inline]

check whether a state is part of the DFAstate

Parameters

checkState	the state to be checked

Returns

true if state is part of DFAstate

2.3.3.3 stateset FA::DFAstate::makeTransitions (char *symbol,* transitions *delta,* eNFA *automata*) [inline]

Finds target states for transition with certain symbol.

Parameters

symbol	The input symbol
delta	The transitions
automata	The automaton of which the state is a part

Returns

Set containing all the target states

2.3.3.4 bool FA::DFAstate::transition (char *symbol*, DFAstate * *state*) [inline]

Checks if there is a transition from the state of a certain symbol.

Parameters

symbol	The input symbol
state	Pointer to the state

Returns

True if there is a transition

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

- /home/jakob/Dropbox/UA/workspace/FormValidation/src/DFA.h

2.4 FA::eNFA Class Reference

Class representing the eNFA.

`#include <eNFA.h>`

Public Member Functions

- eNFA ()

  constructor for empty eNFA
- eNFA (alphabet, states, transitions, state *, acceptingStates)

  constructor
- const transitions & getDelta () const

  getter for delta (the transitions)
- const acceptingStates & getF () const

  getter for F (the accepting states)
- const states & getQ () const

  getter for Q (the states)
- state * getQ0 () const

  getter for Q0 (the start state)
- const alphabet & getSigma () const

  getter for sigma (the alphabet)
- bool process (std::string) const

  checks if string is part of language defined by eNFA
- void toFile (std::string)

  generates File version of eNFA (can be read again)
- stateset eclose (state *)

  generate eclose of state

Friends

- std::ostream & operator<< (std::ostream &, const eNFA &)

  << *overloader for eNFA*

2.4.1 Detailed Description

Class representing the eNFA.

2.4.2 Constructor & Destructor Documentation

2.4.2.1 FA::eNFA::eNFA (alphabet *alphabet_*, states *states_*, transitions *transitions_*, state * *start_*, acceptingStates *accepting_*)

constructor

Parameters

*alphabet_*	the alphabet
*states_*	the states
*transitions_*	the transitions
*start_*	pointer to start state
acceptingStates	the accepting states

2.4.3 Member Function Documentation

2.4.3.1 stateset FA::eNFA::eclose (state * *workingState*)

generate eclose of state

Parameters

workingState	pointer to state to generate eclose of

Returns

the eclose

2.4.3.2 const transitions& FA::eNFA::getDelta ( ) const [inline]

getter for delta (the transitions)

Returns

the transitions

2.4.3.3 const acceptingStates& FA::eNFA::getF ( ) const [inline]

getter for F (the accepting states)

Returns

the accepting states

2.4.3.4 const states& FA::eNFA::getQ ( ) const [inline]

getter for Q (the states)

Returns

the states

2.4.3.5 state* FA::eNFA::getQ0 ( ) const [inline]

getter for Q0 (the start state)

Returns

the start state (pointer)

2.4.3.6 const alphabet& FA::eNFA::getSigma ( ) const [inline]

getter for sigma (the alphabet)

Returns

the alphabet

2.4.3.7 bool FA::eNFA::process (std::string *str*) const

checks if string is part of language defined by eNFA

Parameters

str	the string to be processed

Returns

true if string belongs to eNFA

2.4.3.8 void FA::eNFA::toFile (std::string *filename*)

generates File version of eNFA (can be read again)

Parameters

filename	name of the file

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

- /home/jakob/Dropbox/UA/workspace/FormValidation/src/eNFA.h
- /home/jakob/Dropbox/UA/workspace/FormValidation/src/eNFA.cpp

2.5 FA::Field Class Reference

Class representing a field of a form.

`#include <Field.h>`

Public Member Functions

- Field (Component *type, std::string name)
  constructor
- Field (Component *type, std::string name, unsigned int length)
  constructor
- Field (Component *type, std::string name, unsigned int length, bool required)
  constructor
- void required ()
  sets field to required
- void notRequired ()
  sets field to not required
- void length (unsigned int value)
  set minimal length of input
- void defaultValue (std::string value)
  set default value for field
- std::string makeLabel ()
  generates a label for the field
- bool process (std::string value)
  check if input should be accepted into the field and if so: set the field's value to input
- bool check (std::string value)
  check if input should be accepted into the field
- bool isAccepted ()
  Checks if field is filled in (or empty if not required)
- bool isFilledIn ()
  Checks if field is filled in.
- bool isRequired ()
  Checks if field is required.
- Component * getType ()
  Gets the type of field (the component)
- unsigned int getLength ()
  gets the minimal length of the input
- std::string getName ()
  gets the name of the field
- std::string getValue ()
  gets the value filled in

2.5.1 Detailed Description

Class representing a field of a form.

2.5.2 Constructor & Destructor Documentation

2.5.2.1 FA::Field::Field (Component * *type*, std::string *name*)

constructor

Parameters

type	Pointer to the component
name	Name of the field

2.5.2.2 FA::Field::Field (Component * *type,* std::string *name,* unsigned int *length*)

constructor

Parameters

type	Pointer to the component
name	Name of the field
length	Minimal length of input

2.5.2.3 FA::Field::Field (Component * *type,* std::string *name,* unsigned int *length,* bool *required*)

constructor

Parameters

type	Pointer to the component
name	Name of the field
length	Minimal length of input
required	True if field has to be filled in

2.5.3 Member Function Documentation

2.5.3.1 bool FA::Field::check (std::string *value*)

check if input should be accepted into the field

Parameters

value	value to be checked

Returns

true if accepted

2.5.3.2 void FA::Field::defaultValue (std::string *value*)

set default value for field

Parameters

value	the default value

2.5.3.3 unsigned int FA::Field::getLength ( )

gets the minimal length of the input

Returns

minimal length of the input

2.5.3.4 std::string FA::Field::getName ( )

gets the name of the field

Returns

name of the field

2.5.3.5 Component * FA::Field::getType ( )

Gets the type of field (the component)

Returns

pointer to the component

2.5.3.6 std::string FA::Field::getValue ( )

gets the value filled in

Returns

the filled in value

2.5.3.7 bool FA::Field::isAccepted ( )

Checks if field is filled in (or empty if not required)

Returns

true if field doesn't have to be filled in anymore

2.5.3.8 bool FA::Field::isFilledIn ( )

Checks if field is filled in.

Returns

true if field is filled in

2.5.3.9 bool FA::Field::isRequired ( )

Checks if field is required.

Returns

true if required

2.5.3.10 void FA::Field::length (unsigned int *value*)

set minimal length of input

Parameters

value	the minimal length

2.5.3.11 std::string FA::Field::makeLabel ( )

generates a label for the field

Returns

the label

2.5.3.12 bool FA::Field::process (std::string *value*)

check if input should be accepted into the field and if so: set the field's value to input

Parameters

value	the input to be checked

Returns

true if accepted and set

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

- /home/jakob/Dropbox/UA/workspace/FormValidation/src/Field.h
- /home/jakob/Dropbox/UA/workspace/FormValidation/src/Field.cpp

2.6 FA::Form Class Reference

Class representing a form.

`#include <Form.h>`

Public Member Functions

- Form (std::string name)
  constructor
- bool add (std::string name, std::string type)
  add non-required field without minimal length to form
- bool add (std::string name, std::string type, bool required)
  add field without minimum length to form
- bool add (std::string name, std::string type, unsigned int length)
  add non-required field to form
- bool add (std::string name, std::string type, bool required, unsigned int length)
  add field to form
- bool add (std::string name, std::string type, unsigned int length, bool required)

add field to form

- bool addComponents (std::string file)

  adds all of the possible components to the form
- void build ()

  run the form
- bool ok ()

  check if form is complete
- void process ()

  run the form (without printing name of form)
- std::map< std::string, std::string > getData ()

  Get the filled in data from the form.
- bool load (std::string file)

  load form from file

2.6.1 Detailed Description

Class representing a form.

2.6.2 Constructor & Destructor Documentation

2.6.2.1 FA::Form::Form (std::string *name*)

constructor

Parameters

name	Name of the form

2.6.3 Member Function Documentation

2.6.3.1 bool FA::Form::add (std::string *name*, std::string *type*)

add non-required field without minimal length to form

Parameters

name	name of the field
type	of the field (its component)

2.6.3.2 bool FA::Form::add (std::string *name*, std::string *type*, bool *required*)

add field without minimum length to form

Parameters

name	name of the field
type	of the field (its component)
required	true if required

2.6.3.3 bool FA::Form::add (std::string *name,* std::string *type,* unsigned int *length*)

add non-required field to form

Parameters

name	name of the field
type	of the field (its component)
length	minimal length of input

2.6.3.4 bool FA::Form::add (std::string *name,* std::string *type,* bool *required,* unsigned int *length*)

add field to form

Parameters

name	name of the field
type	of the field (its component)
required	true if required
length	minimal length of input

2.6.3.5 bool FA::Form::add (std::string *name,* std::string *type,* unsigned int *length,* bool *required*)

add field to form

Parameters

name	name of the field
type	of the field (its component)
length	minimal length of input
required	true if required

2.6.3.6 bool FA::Form::addComponents (std::string *file*)

adds all of the possible components to the form

Parameters

file	file name of textfile containing components

Returns

true if success

2.6.3.7 std::map< std::string, std::string > FA::Form::getData ( )

Get the filled in data from the form.

Returns

the data

2.6.3.8 bool FA::Form::load (std::string *file*)

load form from file

Parameters

file	the file name of file containing form

2.6.3.9 bool FA::Form::ok ( )

check if form is complete

Returns

true if complete

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

- /home/jakob/Dropbox/UA/workspace/FormValidation/src/Form.h
- /home/jakob/Dropbox/UA/workspace/FormValidation/src/Form.cpp