Form Validation

Generated by Doxygen 1.8.3.1

Thu Jun 6 2013 12:58:15

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::Arc Class Reference

Class representing an arc.

`#include <DFA.h>`

Public Member Functions

- Arc (State ∗destination)

  Constructor.
- bool addSymbol (char symbol)

  Add a symbol to the arc.
- bool addSymbols (std::vector< char > symbols)

  Add symbols to the arc.
- bool checkAlphabet (std::vector< char > &alphabet)

  Check if the symbols in the arc are legitimate by the alphabet.
- State ∗ process (char symbol)

  Process a symbol.
- State ∗ getDestination ()

  Get the state this arc is going to.

Friends

- std::ostream & **operator**<< (std::ostream &out, Arc &arc)

2.1.1 Detailed Description

Class representing an arc.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 FA::Arc::Arc (State ∗ *destination*)

Constructor.

Parameters

State∗	The State this arc is going to

2.1.3 Member Function Documentation

2.1.3.1 bool FA::Arc::addSymbol (char *symbol*)

Add a symbol to the arc.

Parameters

symbol	The symbol

Returns

bool True if succes

2.1.3.2 bool FA::Arc::addSymbols (std::vector< char > *symbols*)

Add symbols to the arc.

Parameters

vector	The symbols

Returns

bool True if succes

2.1.3.3 bool FA::Arc::checkAlphabet (std::vector< char > & *alphabet*)

Check if the symbols in the arc are legitimate by the alphabet.

Parameters

alphabet	The alphabet

Returns

bool True if the symbols are legitimate

2.1.3.4 State ∗ FA::Arc::getDestination ( )

Get the state this arc is going to.

Returns

state The state

2.1.3.5 State ∗ FA::Arc::process (char *symbol*)

Process a symbol.

Parameters

symbol	The symbol

Returns

state Or if there is no such symbol in the arc NULL

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

2.2 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)

  check if string is accepted by component
- std::string getType ()

  returns the type of the component

2.2.1 Detailed Description

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

2.2.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

type	type of the component

2.2.3 Member Function Documentation

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

add database to regex

Parameters

file	Name of the database file

Precondition

No database should be present

2.2.3.2 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

2.2.3.3 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

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

add eNFA described by file to component

Parameters

file	The filename

Precondition

No database should be present

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

returns the type of the component

Returns

the type

2.2.3.6 bool FA::Component::process (std::string *value*)

check if string is accepted by component

Returns

true if accepted

2.2.3.7 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

2.3 FA::DFA Class Reference

Class representing a DFA.

`#include <DFA.h>`

Public Member Functions

- DFA ()

  Constructor.
- bool process (std::string string)

  Process a string.
- State ∗ process (char symbol)

  Process a symbol.
- State ∗ process (char symbol, State ∗currentState)

Process a symbol at a specified state.
- bool addState (State state)

  Add a state to the DFA.
- bool addStates (std::vector< State > states)

  Add states to the DFA.
- bool addAlphabet (char symbol)

  Add a symbol to the DFA's alphabet.
- bool addAlphabet (std::vector< char > symbols)

  Add symbols to the DFA's alphabet.
- bool isInAlphabet (char symbol)

  Check if symbol is in the Dfa's alphabet.
- bool hasStartState ()

  Check if a DFA has a start state.
- std::vector< State > getStates ()

  Get the states in the DFA.
- std::vector< char > getAlphabet ()

  Get the alphabet in the DFA.
- void clearStates ()

  Removes all the states in the DFA.
- void clear ()

  Clears the states, startstate, transitions in the DFA.

Public Attributes

- std::vector< State > fStates

  Read's a DFA file and makes one based upon this file.

Friends

- std::ostream & **operator**<< (std::ostream &out, DFA &dfa)

2.3.1 Detailed Description

Class representing a DFA.

2.3.2 Member Function Documentation

2.3.2.1 bool FA::DFA::addAlphabet (char *symbol*)

Add a symbol to the DFA's alphabet.

Parameters

char	The symbol

Returns

bool True if success

2.3.2.2 bool FA::DFA::addAlphabet (std::vector< char > *symbols*)

Add symbols to the DFA's alphabet.

Parameters

vector	The symbols

Returns

bool True if success

2.3.2.3 bool FA::DFA::addState (State *state*)

Add a state to the DFA.

Parameters

state	The state

Returns

bool True if success

2.3.2.4 bool FA::DFA::addStates (std::vector< State > *states*)

Add states to the DFA.

Parameters

vector	The states

Returns

bool True if success

2.3.2.5 std::vector< char > FA::DFA::getAlphabet ( )

Get the alphabet in the DFA.

Returns

vector The states

2.3.2.6 std::vector< State > FA::DFA::getStates ( )

Get the states in the DFA.

Returns

vector The states

2.3.2.7 bool FA::DFA::hasStartState ( )

Check if a DFA has a start state.

Returns

bool True if there is a start state

2.3.2.8 bool FA::DFA::isInAlphabet (char *symbol*)

Check if symbol is in the Dfa's alphabet.

Parameters

char	The symbol

Returns

bool True if symbol is in the alphabet

2.3.2.9 bool FA::DFA::process (std::string *string*)

Process a string.

Parameters

string	The string

Returns

bool True if string is accepted

2.3.2.10 State ∗ FA::DFA::process (char *symbol*)

Process a symbol.

Parameters

symbol	The symbol

Returns

bool True if symbol is accepted

2.3.2.11 State ∗ FA::DFA::process (char *symbol,* State ∗ *currentState*)

Process a symbol at a specified state.

Parameters

symbol	The symbol
state∗	The state to process

Returns

bool True if symbol is accepted

2.3.3 Member Data Documentation

2.3.3.1 std::vector<State> FA::DFA::fStates

Read's a DFA file and makes one based upon this file.

Parameters

string	The file to be loaded

Returns

bool True if success Save the DFA in a file

Parameters

string	The filename of the file

Returns

bool True if success

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

2.4 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 ∗ getComponent ()

  Gets the component of the field.
- 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 Component ∗ FA::Field::getComponent ( )

Gets the component of the field.

Returns

pointer to the component

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

gets the minimal length of the input

Returns

minimal length of the input

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

gets the name of the field

Returns

name of the field

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, const std::vector< std::string > &=std::vector< std::string >())

  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
- void readComponents (std::string file, std::vector< std::string > &usedComps)

  get vector of components used by form

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*, const std::vector< std::string > & *usedComps* = `std::vector<std::string>()`)

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

2.6.3.10 void FA::Form::readComponents (std::string *file,* std::vector< std::string > & *usedComps*)

get vector of components used by form

Parameters

file	the file name of file containing form
usedComps	vector to contain the names of the components used by the form

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

2.7 FA::State Class Reference

Class representing a state.

```
#include <DFA.h>
```

Public Member Functions

- State (bool ending)

  Constructor.
- State (bool ending, bool starting)

  Constructor.
- std::string getLabel ()

  Get the label of the state.
- std::vector< std::string > getLabels ()

  Get the labels of the state(if there are more)
- std::string getName ()

  Get the name of the state.
- bool addLabel (std::string label)

  Add's a label.
- bool addLabels (std::vector< std::string > labels)

  Add's a labels.

- bool addArc (Arc arc)

  Add's an arc.
- bool addTransition (char symbol, State ∗destination)

  Add's an transition to another state.
- bool isEnding ()

  Check if a state is accepting.
- bool isStarting ()

  Check if a state is starting.
- bool checkAlphabet (std::vector< char > &alphabet)

  Check if a state's arcs have legitimate symbols from an alphabet.
- bool **hasLabel** (std::string label)
- void **makeEnding** ()
- State ∗ process (char symbol)

  Get the state when we process a symbol.

Friends

- std::ostream & **operator**<< (std::ostream &out, State &state)

2.7.1 Detailed Description

Class representing a state.

2.7.2 Constructor & Destructor Documentation

2.7.2.1 FA::State::State (bool *ending*)

Constructor.

Parameters

bool	is this an accepting state

2.7.2.2 FA::State::State (bool *ending,* bool *starting*)

Constructor.

Parameters

bool	is this an accepting state
bool	is this an starting state

2.7.3 Member Function Documentation

2.7.3.1 bool FA::State::addArc (Arc *arc*)

Add's an arc.

Parameters

arc	the State's arc

Returns

bool When succes

2.7.3.2 bool FA::State::addLabel (std::string *label*)

Add's a label.

Parameters

string	the State's label

Returns

bool When succes

2.7.3.3 bool FA::State::addLabels (std::vector< std::string > *labels*)

Add's a labels.

Parameters

vector	the State's labels

Returns

bool When succes

2.7.3.4 bool FA::State::addTransition (char *symbol*, State ∗ *destination*)

Add's an transition to another state.

Parameters

char	The symbol for the transition
State∗	The state this transition goes to

Returns

bool When succes

2.7.3.5 bool FA::State::checkAlphabet (std::vector< char > & *alphabet*)

Check if a state's arcs have legitimate symbols from an alphabet.

Parameters

vector	The alphabet

Returns

bool true if this state is legitimate

2.7.3.6 std::string FA::State::getLabel ( )

Get the label of the state.

Returns

the state's label

2.7.3.7 std::vector< std::string > FA::State::getLabels ( )

Get the labels of the state(if there are more)

Returns

vector The state's labels

2.7.3.8 std::string FA::State::getName ( )

Get the name of the state.

Returns

string All the labels of the state concatenated

2.7.3.9 bool FA::State::isEnding ( )

Check if a state is accepting.

Returns

bool true if this state is accepting

2.7.3.10 bool FA::State::isStarting ( )

Check if a state is starting.

Returns

bool true if this state is starting

2.7.3.11 State ∗ FA::State::process (char *symbol*)

Get the state when we process a symbol.

Parameters

char	The symbol

Returns

State Or Null if there is no such transition

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

2.8 FA::SubsetConstruction Class Reference

Public Member Functions

- SubsetConstruction (eNFA ∗automata)

  Constructor.
- DFA ∗ getDFA ()

  Get the generated DFA.

2.8.1 Constructor & Destructor Documentation

2.8.1.1 FA::SubsetConstruction::SubsetConstruction (eNFA ∗ *automata*)

Constructor.

Parameters

automata	the eNFA

2.8.2 Member Function Documentation

2.8.2.1 DFA ∗ FA::SubsetConstruction::getDFA ( )

Get the generated DFA.

Returns

DFA The generated DFA

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

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

2.9 FA::Test Class Reference

Public Member Functions

- **Test** (std::string name)
- bool **equal** (bool val1, bool val2)
- bool **equal** (int val1, int val2)
- bool **equal** (float val1, float val2)
- bool **equal** (std::string val1, std::string val2)
- bool **different** (bool val1, bool val2)
- bool **different** (int val1, int val2)
- bool **different** (float val1, float val2)
- bool **different** (std::string val1, std::string val2)
- bool **expectTrue** (bool val)
- bool **expectFalse** (bool val)

- int **runAllTests** ()

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

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