Warsaw University of Technology
Faculty of Mathematics and Computer Science

Testing of ReCon

an application for building finite-state machine that is equivalent to a given regular expression and for simulating machine's evaluation of a given word

Author:
Mateusz Bysiek

Supervisor:
dr Lucjan Stapp

Warsaw, 6 Jun 2013

Document metric

Project	Testing of ReCon	**Company**	WUT
Document name	test report		
Document topics	conformance tests, testing scenarios, detected anomailes, general recommendations		
Author	Mateusz Bysiek		
File	bysiekm-testing.pdf		
Version no.	1.0 **Status** final	**Opening date**	1 Jun 2013
Summary	Author provides definitions of test scenarios that include expected application behavior, and performs them, writnig down all results, and he compares the contents of arfifacts from previous stages of the project to the actual application experience.		
Authorized by	dr Lucjan Stapp	**Last modification date**	6 Jun 2013

History of changes

Version	Date	Author	Description
1.0	1 Jun 2013	Mateusz Bysiek	creation of template for the document
1.0	2 Jun 2013	Mateusz Bysiek	added basic tests
1.0	6 Jun 2013	Mateusz Bysiek	added regexp to automaton conversion tests
1.0	6 Jun 2013	Mateusz Bysiek	added word evaluation tests
1.0	6 Jun 2013	Mateusz Bysiek	added abstract

Contents

Abstract

This is a acceptance test report for an application for building finite-state machine that is equivalent to a given regular expression
and for simulating machine's evaluation of a given word. Application's name is ReCon.

Author of the document lists test scenarios that are relevant for working application that is meant to solve a well defined language-theory related problem, he does it from end-user perspective, i.e. provides definitions of test scenarios that include expected application behavior, and performs them, writnig down all results.

He also compares the contents of arfifacts from previous stages of the project to the actual application experience, he does it from end-user perspective, adversary perspective, and technical advisor perspective.

1 Release package

Release package contains application executable, together with business analysis and technical analysis. The application executable is accompanied by several supporting files.

1.1 Anomalies

There are some obsolete files in the release package:

- *.pdb
- RegexFiniteAutomaton.vshost.*

The presence of additional executable may confuse some users. Presence of debugging symbols may help the adversary to reverse engineer the application, which may be undesired.

Otherwise, release package is well prepared - it contains all required files and the application can be launched without the need for any initial environment configuration.

2 Conformance with the documentation

2.1 Business analysis

The application fulfills most of the requirements defined in the business analysis. However there are some parts of the application that are in contradiction with the business analysis, and there was no indication in technical analysis that the requirements with regard these parts have changed.

2.2 Business analysis - nonconforming parts

2.2.1 Section 2.1.2

Per "Upper index characters should be preceded by a "^" sign." the language for regular expressions used in the application is partially wrong. It is important to remark that in most part in conforms with the business analysis.

2.2.2 Section 2.2.3

Per "user of the application should be able to observe computation for a given regular expression on the generated automata" and "The transition is made when the user clicks Next step button" the application is missing step-by-step machine construction.

2.3 Technical analysis

When main features are concerned, the application realizes the main purposes described in the documentation, however there are some details that require attention.

2.4 Technical analysis - nonconforming parts

2.4.1 Section "General Requirements"

Per "All states are marked as black ellipses, all transitions as directed nodes (also black)." the coloring scheme used in the application is wrong.

2.4.2 Section "GUI"

Per "in the input section there are two radio button's groups. One for entering the input and the second one for selecting computation mode." the application has two radio buttton groups, but one of these groups consists of only one radio option.

3 Startup

There are no general preconditions for these tests.

3.1 Testing scenarios

Nr	Scenario	Input	Expected Result
1	starting app	double-click on app executable	app opens

3.2 Anomalies

None detected.

4 Regular expression handling

Before theses thests: launch application, and wait for 2 seconds.

4.1 Testing scenarios

Nr	Scenario	Input	Expected Result
1	empty text box	leave text box empty, click convert	error indication is seen
2	empty word	enter "$" in the text box, click convert	computation ends normally, automaton is seen
3	letter "a"	enter "a" in the text box, click convert	computation ends normally, automaton is seen
4	nonsense input	enter "$a^*(^*$" in the text box, click convert	error indication is seen
5	advanced test 1	enter "$(a+b)(a+b)(a+b)(a+b)$" in the text box, click convert	computation ends normally, automaton is seen
6	advanced test 2	enter "$(a+b)^*$" in the text box, click convert	computation ends normally, automaton is seen
7	kleene star	enter "a^*" in the text box, click convert	computation ends normally, automaton is seen
8	kleene plus	enter "$a\hat{}$" in the text box, click convert	computation ends normally, automaton is seen
9	special characters	enter "#", ".", "_" or "-" in the text box, click convert	error indication is seen

4.2 Anomalies

Detected in the following scenarios:

- 1: **Critical**, application stops working and exits.
- 2: **Critical**, application stops working and exits.
- 4: **Critical**, application stops working and exits.
- 5, 6 and 7: Minor, resulting automatons are not minimal.
- 9: **Critical**, application stops working and exits.

4.3 Comments

Major, I have not found any method of launching conversion of regular expression into finite-state machine in step-by-step mode - the only choice is to do immediate computation.

Major, there is an issue of missing step-by-stem finite-state machine construction feature. Either the module is missing, or a user control that enables the user to choose construction mode was forgotten.

5 Word handling

Before these tests: launch application, enter "a^*" in the regexp text box, click convert.

5.1 Testing scenarios

Nr	Scenario	Input	Expected Result
1	accepted word	select normal mode, enter "aaa", click observe	computation ends normally, information that word was accepted is seen
2	rejected word	select normal mode, enter "aab", click observe	computation ends normally, information that word was rejected is seen
3	accepted word, stepping	select step-by-step mode, enter "aaa", click observe	computation ends normally, information that word was accepted is seen
4	rejected word, stepping	select step-by-step mode, enter "aab", click observe	computation ends normally, information that word was rejected is seen
5	accepted empty word	select normal mode, enter nothing, click observe	computation ends normally, information that word was accepted is seen
6	accepted empty word, stepping	select step-by-step mode, enter nothing, click observe	computation ends normally, information that word was accepted is seen

5.2 Anomalies

Detected in the following scenarios:

- 2 and 4: **Critical**, application stops working and exits.

6 Closing

Before these tests: launch application.

6.1 Testing scenarios

Nr	Scenario	Input	Expected Result
1	exiting while app idle	wait for 2 seconds, click on X in app window	application shuts down
2	exiting while app computing	enter "a^{128}" (128 "a"s) in the text box, click convert, immediately click on X in app window	application shuts down

6.2 Anomalies

Detected in the following scenarios:

- 2: **Major**, the X button is not responding until the application finishes computation

7 Summary

Application performs well for a very specific input data. It handles typical regular expressions, and produces not always minimal, but correct finite-state machines. However, application is unable to handle any unexpected content, therefore in my opinion error handling is either not present or it is not properly done.

In most part, application conforms with requirements defined in the previous stages of the project, but there are some features that are either required but not implemented, or implemented but in a way that is not correct from the formal requirements point of view.

There is also an issue of UI blocking that is an application wide issue - whenever application is occupied, user cannot perform any other tasks, but there is no indication of this, the application window simply hangs until the application becomes idle.

In conclusion, the ReCon application is conditionally accepted as it is. There is one condition of acceptance: it is that error handling must be done properly, so that no unexpected application shutdown happens when the tests defined above are performed.