Generation of Test Cases

Programming a tool for the generation of test cases from finite automata

Stefan Hildenbrand
Supervisor: Diana Senn
ETH Zürich

Topics

- Task and Basics
- Resources
- Examples
- Concluding Comments

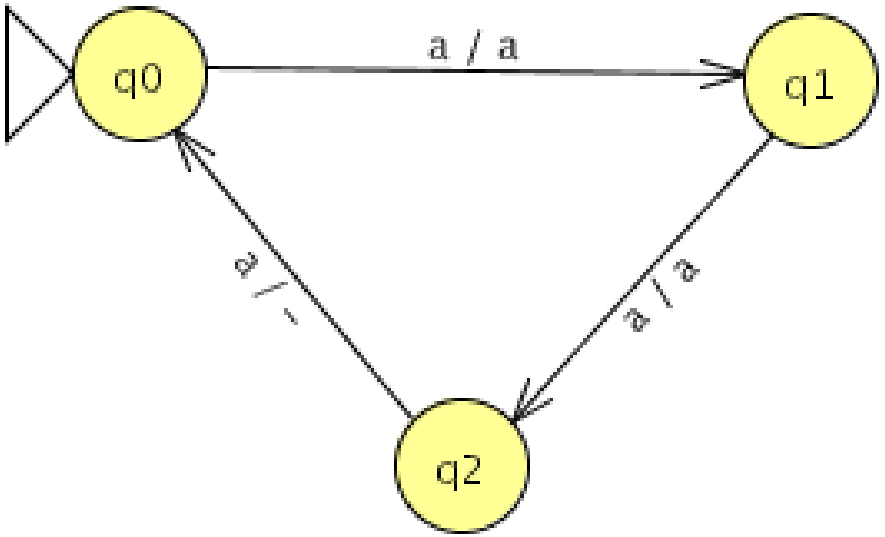
Task and Basics

Task

- implement a tool for
  - graphical specification of Mealy Machines
  - generating abstract test cases
- use another tool for graphical specification of automata as foundation (if possible)
- algorithm for test case generation given by the supervisor

Mealy Machines

When I talk about Finite State Machines, I mean so called *Mealy Machines*. This type of FSM fits best for this application. A Mealy Machine can be represented graphically:



Resources

Foundation for the Tool

- Tool is based on *JFLAP - Java Formal Languages and Automata Package* from Susan Rodger, Duke University, Durham.
- JFLAP offers functions for graphical manipulation of automata and can save the automata as XML-files.
- I extended JFLAP for Mealy Machines and integrated the Test Case Generation Algorithms.

From the Automata

The main components of which a test suite is built are

- *state cover set*, a set of input sequences containing for each state S_i an input sequence p_i such that $S_0 \stackrel{p_i}{\rightarrow} S_i$.
- *transition cover set*, a set of input sequences containing for each transition $S_i \stackrel{x/y}{\rightarrow} S_j$ an input sequence p_i such that $S_0 \stackrel{p_i}{\rightarrow} S_i$ and $S_0 \stackrel{p_i x}{\rightarrow} S_j$.
- *identification set*, a set of input sequences which can be used to distinguish the state from all other states in the automaton

Methods

partial W-method (Wp-method) from Fujiwara et al.

method to generate test cases using a two phase approach:

1. check if all states are present in the implementation
2. check if all transitions reach the correct state

state identification with Gill

method to find the identification sets in two basic steps:

1. find input sequences to distinguish every pair of states
2. use these sequences to divide the set of states in singletons

Examples

Example A: TCP Handling

This example is about TCP-Protocol handling.

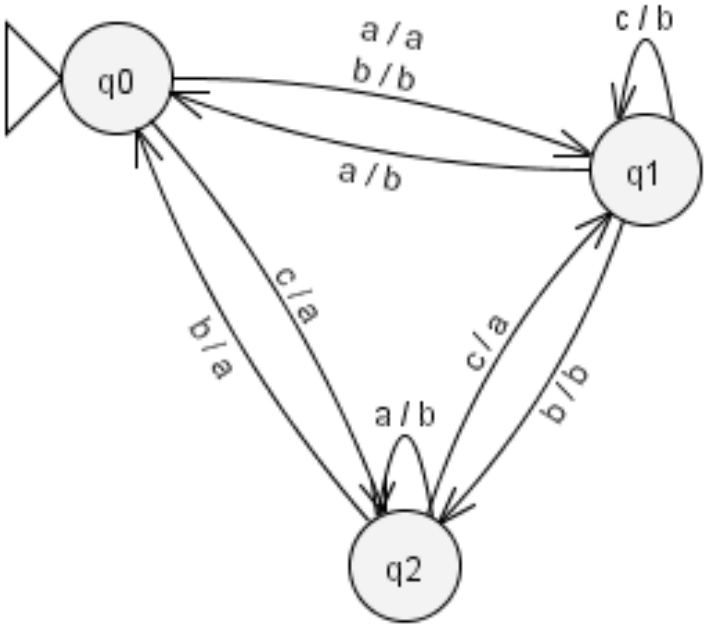
The TCP-Header consists of several fields. For correct connection handling, we need to know

- the source and destination address
- the flags for connection handling

We will see how to set up complex automata with this tool.

Example B: Simple But Detailed

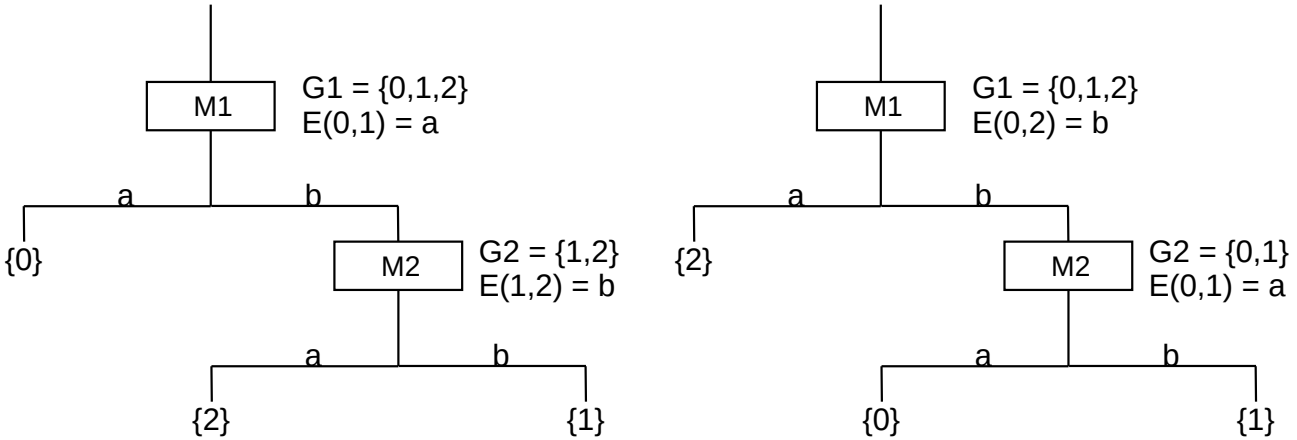
This example is about a simple automaton,
simple enough to look at the details



We will see how the algorithms work behind the scene.

Example B: behind the scene

- *cover sets:* straight forward
- *identification sets:* expected $\{\{a\}, \{a, b\}, \{b\}\}$, but got $\{\{a, b\}, \{a, b\}, \{b\}\}$



- *results from the Wp-method:* expected 10 test cases, but got 12.

Concluding Comments

Summary

- Test case generation can be done several ways
- implemented one method, focusing on genericity, traded genericity for optimality
- added features useful for network protocol testing, such as possibility to define the input and output alphabet

Conclusions

- take the right foundation: started with Exorciser, had to switch to JFLAP
- tradeoff: genericity for optimality
- references in papers can lead into really long chains: started in 1991, ended up with literature from 1962

Future Work

- add other test case generation methods
- implement the actual method more efficiently (use functional language?)
- separate input and output alphabet
- implement special symbols like *forward* (>), *drop* (-) or *other*
- *instantiate the abstract test cases*

Thanks for your attention and goodbye.