Assignment 3 - Model-Based Testing

INTRODUCTION

This third assignment is about model-based testing. The objective is to test several implementations (IUTs) of elevator control software against a UPPAAL model. This part consists of 2 tasks. Each task ends with a description of what you should hand in for that task. Each of the tasks consists of several actions. Actions are labelled as follows:

▷ Read this assignment.
Each action is followed by further explanation.

Task 3.1 Setting up a test environment (3 hours)

Download and open the virtual box image (see YouLearn, "Bronnen", "Externe links").

▷

If you are not familiar with VirtualBox, you can follow detailed instructions for this action at:
http://www.cs.ru.nl/~freekver/courses/svt/vbox_help.html
Once Ubuntu has started, you should see a Mozilla Firefox window with a tab opened at URL http://cs.ru.nl/~freekver/courses/svt/ (otherwise clicking "Home" will take you there). To get a testing environment set up, you will need to have both an IUT and the testing software up and running.

▷ (IUT) Open a terminal and execute the following two commands:

```
cd /home/svv/svv/assignments
tronjava LiftRealistic.Main -M 0
```

These two commands will start an IUT that controls one elevator with 4 floors and a FIFO request handler with a queue of size 4.

▷ (TRON) Open *another* terminal and execute the following **two** commands:

```
cd /home/svv/svv/assignments
tron -P eager -I SocketAdapter
       initial_spec/single_elevator.xml -v 8 -- localhost 9999
```

This command will start TRON to test the IUT. Let it run for a while and observe the output of both terminals. On the IUTs' terminal, you see the output of the IUT. This output consists of the signals send by the IUT:

Signal	**is send when**
CloseDoor, OpenDoor	the IUT starts closing and opening the elevator doors
opened, closed	the IUT has closed/opened the elevator doors
button *n*	the IUT receives a request for floor *n*
reachedFloor	the IUT has reached some floor
moveUp, moveDown	the IUT is moving
stop	the IUT is stopping

On TRONs' terminal, you see the output of TRON. After a while (about 15 minutes) it will tell you that the IUT passed the test.

▷ Open the UPPAAL model that is the specification for this test, located at:

```
/home/svv/svv/assignments/initial_spec/single_elevator.xml
```

We have implemented 7 different IUTs. IUT 0 has just passed the test. It is indeed a correct implementation. The other 6 IUTs all very closely behave according to the UPPAAL specification, but have exactly one bug in them.

▷ Run IUT 1 by executing in the IUTs terminal:

```
tronjava LiftRealistic.Main -M 1
```

▷ In the other terminal, run TRON again to test IUT 1.

Each time you run TRON the output varies, since test cases are generated with a certain degree of randomness. For IUT 1, TRON should however show that a test failed! For example, TRON can produce the following output (the output has been cleaned up a little, on the actual screen you will see more):

```
( door.closed engine.moving floor_sensors.active
  RequestHandler._id19 MainControl.moving_up button._id7 )


...
dest_floor=2 current_floor=1 RequestHandler.list[0]=2
RequestHandler.list[1]=3 RequestHandler.list[2]=2
RequestHandler.list[3]=0 RequestHandler.len=3

  Options for input   : reach_floor()@(4;5), ...
  Options for output  : (empty)
  Options for internal: (empty)
  Options for delay   : until 5)
  Last time-window    : (3;4)
Got unacceptable output: stop()@302425us at (3;4)
Expected outputs were: (empty)
TEST FAILED: Observed unacceptable output.
```

The first part of this output describes the state reached by TRON. In that state, a test failed! MainControl was in state moving_up. The elevator was at floor 1 and the main control was instructed to go to floor 2. The queue of the elevator was $[2, 3, 2]$ and was not full yet.

Examining the UPPAAL model, you can see that in that state only one thing can happen. Since `dest_floor != current_floor`, the elevator should wait to receive a `reach_floor` signal (this is also produced by TRON in the line "Options for input"). It should not produce a `stop` signal. As a matter of fact, it should not produce any signal at all (this fact is produced by TRON in the line "Options for output"). However, the implementation outputs a `stop` signal, which is unacceptable. TRON has found a bug in IUT 1.

Looking at the screen of the IUT, in this example the following output was produced:

```
Lift started
button 2
CloseDoor
button 3
closed
MoveUp
button 2
reachedFloor
stop
OpenDoor
```

Requests for floors 2, 3 and 2 have been received. However, apparently the IUT decided to stop (i.e., it outputted a `stop` signal) at floor 1. By running TRON several times on IUT 1, different test cases are found that fail. Analysis of these failing test scenarios should give you an idea of the bug that is in IUT 1.

For this task, you do not need to hand in anything. If you encounter any issues during this task, contact Freek Verbeek.

Task 3.2 Testing the IUTs (7 hours)

Your task is: test implementations 1 up to 6 using the UPPAAL model and UPPAAL-TRON. All these IUTs have exactly 1 bug in them. For each IUT, you should deliver:

- A short description of a characteristic run that leads to the bug.
- A characterisation of the UPPAAL states in which the bug occurs.
- An informal description of the bug.

Be short and consise in your descriptions. Note that IUT 6 is very difficult: here it suffices to pinpoint which component you think causes the bug.