02224 Modelling and Analysis of Real-Time Systems Week 3 2021–02–17
DTU Compute Hans Henrik Løvengreen

Lecture

Topics:

- Discussion of Week 2 Exercises
- Introduction to Computation Tree Logic (CTL)
- Introduction to Timed Computation Tree Logic (TCTL)
- The query language of UPPAAL

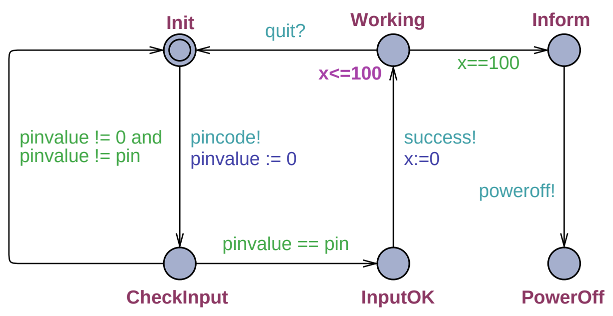
Readings

1. Read sections 3.1, 3.2, 3.3, 3.4 (pages 148– 172) in *Logic in Computer Science: Modelling and reasoning about systems*, by M. Huth and M. Ryan,Cambridge University Press, 2000.
2. Read sections 2.2 and 2.3 in [UPPAAL]. A fast reading of 2.3 suffices for now. We will address committed locations later.

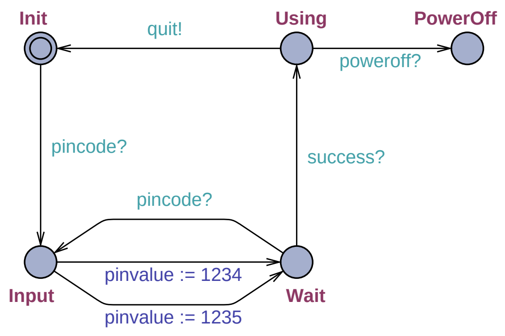
Both of these texts can be found on DTU Learn.

Exercise 1: Simple Access Control

This exercise looks at a simple access control system for mobile phones consisting of an AccessControl process and a User process:



(a) AccessControl



(b) User

The interactions between User and AccessControl are modelled by means of synchronization channels and integer variables. Download the file `mobile.xml` from Learn, and solve the following tasks using UPPAAL:

1. Simulate the system and find out how it works.
2. Verify that it always holds that if the user is using the phone then she has entered the correct pin-code.
3. Verify that the user only receives a poweroff if she has used the phone for 100 or more time units.
4. Verify that if the AccessControl is Working, then the user is Using the phone.
5. The system has a deadlock. Verify this by using Uppaal's special property `deadlock`. Modify the two models such that the locations PowerOff and Init in both processes can be combined, i.e., the location Init corresponds to that the power is off. Verify that the modified system is not deadlocking.
6. Modify the models such that when the clock x reaches 100 time-units, the Access-Control process gives the user 5 time-units warning time before it takes the action poweroff.
7. Modify further the models such that when the user has failed for three times to input the correct pin code, the AccessControl process will take the action poweroff and return to the Init location.

Exercise 2: Simple railroad gate controller

Consider a simple system consisting of three components: a train, a gate, and a controller.

The train starts in location l_0, and communicates with the controller over two channels, *approach* and *leave*. In location l_1 the train is approaching the gate, and in l_2 the train is inside the Gate. In location l_3 the train is leaving the gate, and will emit a *leave* signal before going to the initial location again. The train must emit an *approach* signal at least 3 minutes before entering the crossing. Furthermore, the time between the *approach* and *leave* signals is less than or equal to 5 minutes.

The gate is open in location l_0 and closed in location l_2. In location l_1 it is closing, and in l_3 it is opening. The Gate communicates with the controller via two channels, *lower* and *raise*. When the gate is open it reponds to a *lower* signal by closing within 1 minute. When the gate is closed it responds to a *raise* signal by opening within $]1;2]$ minutes.

The controller is idle in location l_0. Whenever it receives an *approach* signal from the train, it responds by emitting a *lower* signal to the gate, and goes to location l_2. The time between receiving the *approach* signal and emitting the *lower* signal is exactly 1 minute. Then, whenever it receives a *leave* signal, it responds by emitting a *raise* signal to the gate within 1 minute, and is then idle again.

Solve the following tasks in UPPAAL:

1. Model the train, gate and controller as three timed automata in UPPAAL.
2. Check that the system is safe; that is, when the train is inside the gate, the gate should be closed.
3. The system has a deadlock because the gate opens too slowly (check this). Run the simulator with 'Diagnostic Trace' set to 'Shortest' in the Options menu, and try to figure out why. Fix the deadlock by allowing the Gate to open faster (i.e., change the lower bound of the time between the gate receiving a *raise*? event and the gate being open).
4. Check that the train can approach, be in, and leave the gate, respectively.
5. Check that the gate can be lowered and raised, respectively.
6. Check that the controller can lower and raise the gate, respectively.
7. Check that whenever the train approaches the gate, it will inevitably cross it.
8. Check that whenever the gate is lowering, it will inevitably be raising again.
9. Check that the gate is never closed (or lowering/raising) for more than 10 minutes at a time.
10. What is the minimum/maximum time the gate will be closed (or lowering/raising) at a time?