Quantitative Verification Session 3

November 7, 2017

UPPAAL Modeling

Exercise 1. The pseudocode of Fishcher's mutual exclusion protocol is shown in Figure 1. *For reading more about the protocol, refer [1].*

```
while true do
begin
  noncritical section;
  L: if id ≠ 0 then goto L;
  1: id := i;
  2: pause(delay);
  3: if id ≠ i then goto L;
  critical section;
  id := 0;
end
```

pause(delay) makes the process wait for the amount of time specified by the constant 'delay'.

Figure 1: Fischer's Mutual Exclusion Protocol

Model the protocol in Uppaal (use the model shown in the slides). For a system of 10 processes following this protocol, verify the following properties.

1. Mutual exclusion (no two processes are in the critical section at the same time)
2. The system is deadlock free
3. Whenever P(6) requests access to the critical section it will eventually enter the wait state

Exercise 2. In the previous tutorial, we saw an infinite capacity elevator moving between two floors. Get used to the Uppaal tool by modeling this elevator system and visualizing a few traces. If you encounter some unusual behaviour, say while simulating, try to rectify it by altering your model. Can you try to limit the capacity of the elevator? (*Hint:* Uppaal *models can have variables other than clocks*)

Exercise 3. Construct a system modeling trains from multiple tracks crossing a bridge with a single track. The expected behaviour of the train is elaborated below.

- When the Train approaches a bridge, it sends a signal to the controller.
- If the bridge is occupied, the controller sends a stop signal to the train within 10 time units.
- Otherwise, if the train doesn't receive a stop signal within 10 time units, it starts to cross the bridge within 20 time units. It takes the train 3 to 5 time units to leave the bridge.
- If the train receives a stop signal within 10 time units, it comes to a stop. When it receives a go signal from the controller, it starts moving within 15 time units and it takes atleast 7 time units to enter the bridge.

Design a controller which uses an FCFS strategy to process requests. If the bridge is free and a train requests to use it, add it to a queue. When the train leaves, remove it from the queue. If the bridge is being used, always add the train to the queue and ask it to wait until its turn.

Verify the following properties

- Gate can receive (and store in queue) msg's from approaching trains.
- Train 1 can reach crossing.
- Train 0 can be crossing bridge while Train 1 is waiting to cross.
- Train 0 can cross bridge while the other trains are waiting to cross.
- There is never more than one train crossing the bridge (at any time instance).
- There can never be N elements in the queue (thus the array will not overflow).
- Whenever a train approaches the bridge, it will eventually cross.
- The system is deadlock-free.

Source: UPPAAL Demos

References

[1] `https://www2.informatik.hu-berlin.de/~hs/Aktivitaeten/2011_Vino/Talks/Morning/11_Tom-Davies_Fischers-Algorithm.pdf`