Technische Universität München
Institut für Informatik I07
Theoretical Computer Science

Quantitative Verification 2

Ex 1: Modelling[1]

Consider an autonomous elevator which operates between two floors. The requested behaviour of the elevator is as follows:

1. The elevator can stop either at the ground floor or the first floor. When the elevator arrives at a certain floor, its door automatically opens. It takes at least 2 seconds from its arrival before the door opens but the door must definitely open within 5 seconds.
2. Passengers can enter if and only if the elevators doors are open. They enter one by one and we (optimistically) assume that the elevator has a sufficient capacity to accommodate any number of passengers waiting outside.
3. The door can close only 4 seconds after the last passenger entered. After the door closes, the elevator waits at least 2 seconds and then travels up or down to the other floor.

Your task is to

1. model the elevator in `UPPAAL`,
2. think about which of the above properties you can formalize in `UPPAAL`, and
3. check your model by testing said properties.

How does your model change when additional floors are added? Can you model passengers on different floors or a limited capacity of the elevator? *Note: `UPPAAL` also supports non-clock variables.*

Ex 2: Modelling Continued

Algorithm 1 Fischer's Mutual Exclusion Protocol.

Require: id: Global, atomic variable, initialized to 0. *delay*: waiting time parameter
  while true do
    if $id \neq 0$ **then continue**
    $id \leftarrow i$
    pause(*delay*)
    if $id \neq i$ **then continue** L
    `// critical section`
    $id \leftarrow 0$

Model Fischer's mutual exclusion protocol (shown in Algorithm 1) in `UPPAAL`. For a system of 10 processes following this protocol, verify the listed 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.

[1] Based on `http://www.ru.is/kennarar/luca/GSSI/TUTORIALS/tutorial-ta1.pdf`

Hw 1: Modelling a Train

Note: This is a (voluntary) homework to further train modelling in `UPPAAL`*.*

Modelling

Construct a system modelling trains from multiple tracks crossing a bridge with a single track. The expected behaviour of the train is as follows.

- 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 at least 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.

Verification

For a system with three trains, verify the following properties.

- Train 1 can reach the other side.
- Train 0 can be crossing the bridge while Train 1 is waiting to cross.
- Train 0 can cross the bridge while other trains are waiting to cross.
- There is never more than one train crossing the bridge.
- There can never be three elements in the queue.
- Whenever a train approaches the bridge, it will eventually cross.
- The system is deadlock-free.

Note: You can also try to verify the timing constraints by equipping the model with additional clocks.