Formal methods 2020: lab exam simulation

Enrico Magnago

June 4, 2020

1 NuSMV, nuXmv: SIR model

SIR is a popular model for the spread of a disease in a population. Each person can be in one of three states: Susceptible, Infected or Removed. In our modelling we are interested in the number of people that are susceptible to the disease, currently infected and spreading the disease or removed.

- There is a population of N (pick some value $N \geq 5$) individuals.
- $\frac{N}{5}$ of them are initially infected with a disease, all the others are susceptible.
- A susceptible person becomes infected with a probability proportional to the current number of infected people over N.
- An infected person becomes removed eventually in the future (FAIRNESS).
- A ”removed” person remains immune.

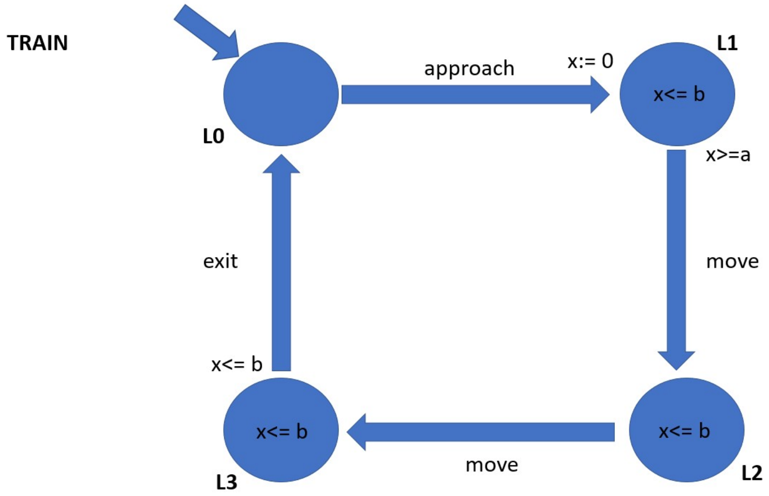
Write and verify the following properties:

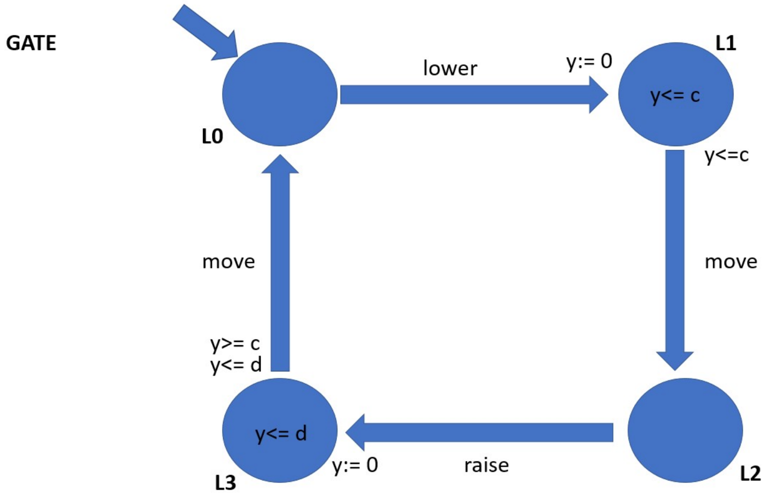
1. Verify that it is possible that the number of infected people does not increase.
2. Verify that it is possible that all the people are eventually infected.
3. Verify that if every person gets infected at some point, then eventually all the people will become immune.

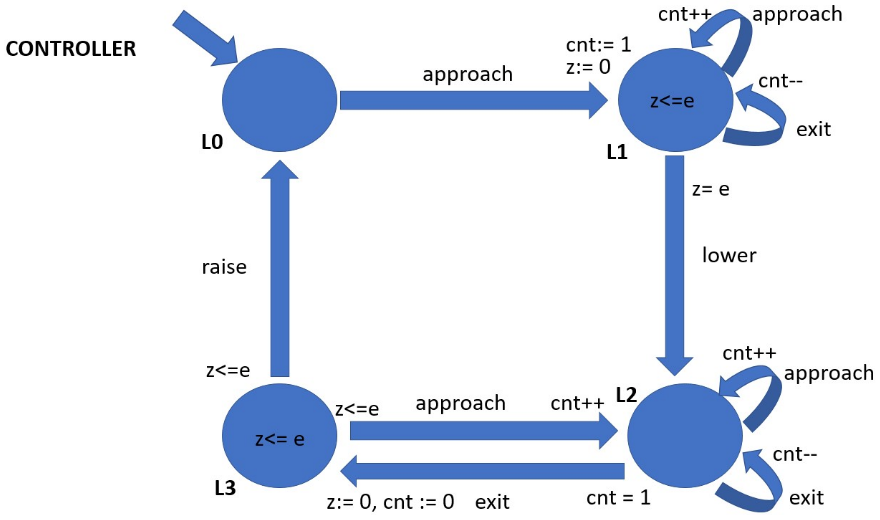
2 timed nuXmv: train-gate-controller

There are 2 trains circling around a railway. A controller communicates with the trains and tells a gate placed on a crossing when to open and close. Each train takes at least *a = 2* time units to approach the crossing and at most *b = 5* time units to approach, reach and exit the crossing. The gate takes at most c = 1 time units to lower the bars and to raise them it takes at least *c* and at most *d = 2* time units. The controller takes *e = 1* time units to communicate to the gate that a train is approaching. Write a timed SMV model representing the asynchronous composition of these components, where each component is represented with the corresponding automaton in the figure. In the controller automaton cnt is NOT a clock, but a discrete variable with domain $0..3$, for simplicity of representation its updates are shown on the transitions, when no update is specified its value remains unchanged.

The four automata (2 trains, 1 gate and 1 controller) synchronise as follows. The controller synchronises with the gate on *lower* and *rise*. The controller synchronises with a train on *approach* and *exit*, notice that it cannot synchronise with both trains on the same transition but with one train at a time.






CONTROLLER
L0
approach
cnt:= 1
z:= 0
z<=e
L1
cnt++
approach
cnt--
exit
z= e
lower
raise
z<=e
L3
z<= e
z<=e
approach
cnt++
L2
cnt++
approach
cnt--
exit
z:= 0, cnt := 0
exit
cnt = 1