ASE 389 and CS 395 T (Spring 2022) - Assignment 3

Exercise 1: Consider an elevator system that serves 4 floors numbered 0 through 3. There is an elevator door at each floor with a call-button and an indicator light that signals whether or not the elevator has been called.

Consider that there can be at most two active requests (i.e., the elevator is called to serve a floor) requests at any time.

If, at an execution step, there is no unserved request, the elevator stays put.

In each execution step, the elevator can stay put or move to one of the adjacent floors.

The doors are "safe," i.e., a floor door is never open if the elevator is not present at the given floor. If a floor is to be served and the elevator is or arrives at that floor at an execution step, the door can open within the same execution step.

Part 1.1: Consider the floor the elevator is at and whether the door at each floor is open or not as controlled variables. The requests cannot be controlled by the system. Treat them as an environment. Design a controller such that the following two specifications (along with all other constraints listed above) will be satisfied. Use nuSMV and model check the resulting "closed-loop" system (does the resulting controller really satisfy the specifications no matter what the environment does?).

- A requested floor will be served some time.
- Again and again the elevator returns to floor 0.

Part 1.2: Now, include the following specification (in addition to those already existing.

- When the top floor is requested, the elevator serves it immediately and does not stop on the way there.

Use nuSMV and model check the resulting "closed-loop" system with the new specification. Does your controller work for the new specification?

If no, "fix it!"

What does "fixing" mean in this case? Does there exist **any** controller that will ensure the satisfaction of the specifications? For example, what if the elevator is requested from the top floor at every execution step? Fixing should involve a combination of figuring out *reasonable* restrictions on the behavior of the environment variables and modifications to your controller. For the former you may considering restricting the frequency with which the elevator can be requested to the top floor. Try to keep such restrictions as loose as possible so that your controller will be useful for a broader set of environment behaviors.

The counterexamples from model checking may give quite useful guidance in figuring out what kind of restrictions on the environment and/or modifications on your controller are needed.

Explain the controllers you design and the results you obtain in each part. Return your nuSMV implementation.

Exercise 2: Exercise 6.32 in "Principles of Model Checking" by Baier & Katoen. So that there is no confusion due to potentially different exercise numbers in different versions of the book, this is the exercise on Lloyd's puzzle.