Formal Methods - Course Project Specification and Proofs

Abhi Jagdev

Daniel McIlvaney

Design Decisions

For ease of later implementation a design based on a logical timer was used, a timer from 5 to -1. Five was chosen as the maximum value to leave room for flexibility when adding additional behaviours. The timer can be considered to map 1:1 to real time seconds, however there should be no loss of generality if other mappings are used.

All formal specifications were developed using Pluscal.

Part 1

Pluscal

Part 1 is a very simple loop in a single process: if the logical timer is greater than 0 it is decremented during each iteration. If the timer is equal to 1 the currently green lights are switched to yellow, and then to red when the timer is 0. The timer is then reset back to 5.
No explicit labels were required to translate the Pluscal into TLA.

Spec and Correctness

Model checking is sufficient to prove all properties of the models since there are no parameters used to define it. The logical timer was chosen to have more values than there are steps in the execution of the program so will never have to change, and there are only two valid choices for the lights at initialization, and they are exact opposites of each other.
Two properties were used to verify the behaviour of the model:

- NoCollisions checks that the lights always cycle such that the EW lights are red when the NS lights are green or yellow, and vice versa
- Cycle checks that the lights always cycle in the correct order, ie green, to yellow, to red, to green.
- The built in deadlock checker was used to verify the liveness of the model.

Part 2

Pluscal

The addition of a non-deterministic behaviour, specifically the pedestrian buttons, required the addition of a second process. This second process sets two flags at random, EW_ped_button and NS_ped_button. The process is not fair since it is completely valid to have no pedestrians press any buttons.

Since the pedestrian lights are expected to switch to yellow earlier than the street lights they trigger when the logical timer is equal to 2.
To more accurately model the real world behaviour of some street lights the model allows for the buttons to be pressed even when the pedestrian lights are already green. This will trigger the pedestrian lights a second time after the intersection has cycled. (This behaviour is distinct from the sensor behaviour)
Unlike part 1, part 2 required several labels be added to translate. These labels are also included in the python code.

Spec, Refinement, and Correctness

Again, given the small size of the model theory proofs were not required to prove the properties.
The second model is a fairly straight forward refinement of part 1, simply mapping the behaviour of the street lights to themselves. The logical clock should not affect the refinement.
Several additional properties need to be verified in part 2.

- Press checks that the pedestrian buttons trigger the corresponding pedestrian crossing
- LongerYellow checks that the pedestrian signals always switch to yellow before the street lights
- NoPedCollisions checks that the pedestrian signals are always red if the opposite traffic is green or yellow
- PedCycle checks the cycle of the pedestrian lights similarly to Cycle from part 1.
- PedOnRed verifies that the pedestrian signals only change when both street lights are red.
- All properties from part 1 are also verified to show the refinement is valid.

Part 3

Pluscal

Part 3 follows the same design as part 2, with two processes. The first process was expanded to also trigger the vehicle sensors. The second process was modified such that the logical timer is allowed to reach -1, a special state signifying no change should be made. The timer is only restarted if there is input from a button or sensor.
The decision was made to not cause a change in signaling if a sensor triggers and the corresponding light is already green. Unlike the pedestrian buttons vehicles will trigger the sensors as they drive over regardless of the state of the light. The sensors will only trigger the lights if they are currently red.

Spec, Refinement, and Correctness

As before model checking was used to prove the correctness of the program.
The refinement from part 3 to part 2 maps the vehicle and pedestrian lights, and the pedestrian buttons.
Only a single additional property was required:

- Sensor verifies that eventually the corresponding light will be green.
- All properties of both part 1 and part 2 were verified.

Results

All of the above properties were successfully verified using the model checker, showing the specification is valid and there is no chance of deadlock.