System modelling and design
Session 1, 2013

12 May 2013

COMP2111 Assignment 2

Vincent Tran z3415372 vtra143

1 Requirements

REQ1 - The system must have 1 set of 2-way traffic lights.
REQ2 - The system must have 4 sets of 4-way traffic lights.
REQ3 - The system must support a 3 light system for both pedestrians and cars.
REQ4 - The system must ensure that no conflicting lights are green at the same time.
REQ5 - The system must account for all combinations of lights at a traffic intersection.
REQ6 - The system must allow for pedestrians to cross the street at traffic intersection.
REQ7 - The system must have push sensors for pedestrians to reqeust a change of lights.
REQ8 - The system must have induction coils for cars to request a change of lights.
REQ9 - The sytsem must give ample time for a car or pedestrian to cross the intersection.
REQ10 - The system must not allow a light to turn green twice before all conflicting directions have turned green once.
REQ11 - The system must fairly distribute the time that a light is green with the rest of the intersection.
REQ12 - The system must allow for the fairness of the lights to be overridden by an emergency signal.

2 Abstract model

The abstract model (Traffic Light Intersection) was mostly based off the traffic light intersection in Ken Robinson's Event-B draftbook. I modified it slightly to be able to handle multiple traffic light intersections. Because we didn't have to model turning cars, there are only two directions in which traffic can run; NorthSouth and EastWest. As such, the pedestrian lights and the car traffic lights should always go green at the same time, so they have been modelled as the same thing. Note that the light colour *Amber* has two different meanings. For cars, it is the standard Amber light indicating drivers to slow down. For pedestrians, Amber is equivalent to the red light flashing before the light stays red.
The INITISIALISATION of the the traffic lights looks rather awful and unnecessary, but it was significantly easier to prove than:

$$lights :\mid \forall\, i, d \cdot i \in INTERSECTIONS \wedge d \in DIRECTION \Rightarrow lights'(i \mapsto d) = Red \tag{1}$$

Which would've initialised all the traffic lights in all directions to Red. Unfortunately I couldn't discharge the FIS and WD POs, so I decided to go with the super long call which would be inflexible for more intersections.

There's a few theorems thrown in there which serve no functional purpose other than to reinforce an understanding of how the machine should be operating and also make it easier to prove other undischarged POs. I believe only *inv2* is necessary for proving *inv4* (you can use proof by contrapositive), I put *inv3* in to make proving *inv4* easier and also enforce more saftey invariants.

3 Adding sensors

Similar to how I initialised the traffic lights, I initialised the state of button presses retardedly. So I simply store whether or not the button has been pushed or a car arrives to trigger the induction coils, and then allow the light to turn Amber if someone in the other direction has arrived. I modelled this similar to night time traffic lights were the lights for one direction stay green until someone from a conflicting direction arrives at the intersection. Note that at Intersection1 (which I have deemed the one in front of the hotel which used to be a zebra crossing), I've restricted the buttons and induction coils to only a single direction using:

$$inters = Intersection1 \Rightarrow dir \neq NorthSouth \tag{2}$$

$$inters = Intersection1 \Rightarrow dir \neq EastWest \tag{3}$$

4 Fairness and emergencies

I decided to put fairness and emergencies in the same refinement because emergencies interefere directly with how the fairness of the lights operate. If I chose to do it at a later refinement (and I did try), I would get an EQL error trying to mess around with the fairness, which involves learning how to use witnesses. And I have no idea how to use witnesses.
To simulate the concept for fairness, I just stored whether or not a certain direction at an intersection was just recently green. Obviously I don't want that light to turn green again, so there's another guard for the ToGreen event. When the emergency signal is switched on, there's a direction in which the emergency response team wants to go. The switch will force the intersections to think that the direction the emergency crews want to go weren't recently green and the other directions were. Since there are no turns in the system, we only need to prioritise one direction, because they cant go EastWest and then NorthSouth. With the emergency signal, I still didn't want to break saftey, so the lights still have to turn red before a light can turn green again.
I try and force the lights to turn Green as soon as possible by setting wasGreen to true for the other direction at all intersections. This does kind of have a few side effects, such as at Intersection1, the pedestrian lights will stay green throughout the emergency if the emergency direction is NorthSouth, and pedestrians suddenly can't cross the road during an emergency if the emergency runs EastWest (although it's probably a good idea for safety reasons).

5 Flow

Probably the dodgiest section of my Event-B model. I experimented using a variant to create a delay of some sort to satisfy REQ6. The "delay" is just a placeholder for an time to pass while not changing the lights. The idea was:

1. The light changes to a new colour
2. delay stores which light (direction at an intersection) just changed colours
3. Delay event occurs to simulate time passing
4. The light can change colour again

I took a bit of inspiration from Ken's draftbook, but his variant looked way too complicated. Something like the delay time changing depending on what colour the light just changed to.
As part of managing emergencies, the goal is the get the direction that the emergency crew want to go green as soon as possible, and keep it green. When the emergency switch is activated, the delay is basically disabled (I stop adding things to it when lights change), so if the light is green in the other direction, it can go red as soon as possible. When it turns green, I've essentially forced the lights to stay green while the emergency signal is on by disabling the ToAmber event. I put the guard:

$$dir \neq emergencyDirection \tag{4}$$

in the ToAmber event to do this. I didn't put it in the ToRed event though because if the emergency signal gets called while the light is Amber, then we won't be able to get back to Green. Well, you probably could leave the lights Amber, that that would be kind of weird and confusing for the drivers.