KF6009 – Model based Design and Verification — Building Larger Systems

Dr Alun Moon

Lab 06

Traffic Lights for a junction

The traffic light system controls traffic at a junction (cross-roads) between two roads. The lights allow traffic to flow North-South, and East-West alternately.

The system will be augmented in later stages with sensors to detect waiting card, and a pedestrian crossing.

Traffic Lights

Traffic lights have three lights Red, Amber, Green shown in the sequence

$$Red \rightarrow Red, Amber \rightarrow Green \rightarrow Amber \rightarrow Red$$

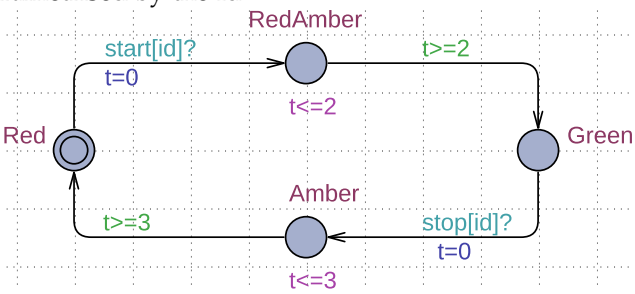
Red,*Amber* is shown for 2s, *Amber* for 3s

- Suggests 4 states
- local clock needed on 2 states
- What triggers for exiting states *Red*, *Green*?

I have two identical systems, can I parametrise them is some way?

We can create a traffic light model, By using Template parameters (`TLid id`) can can use one model and create several instances parametrised by the Id

`TF1.xmf`



In the global declarations. We can use arrays of channels, each template has it's own unique id used as an index into the array.

```
typedef int[1,2] TLset;

const int North = 1;
const int East = 2;

chan   start[TLset];
chan    stop[TLset];
```

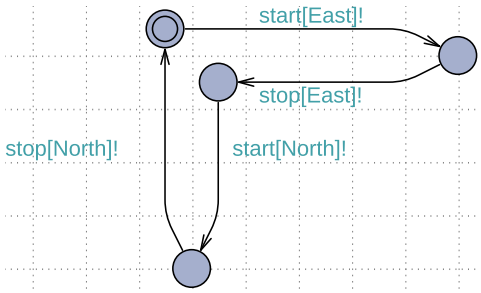
System Declarations In the System Declarations we can create two instances of the traffic-light template.

```
L1 = TrafficLight(Left);
L2 = TrafficLight(Right);

system L1, L2;
```

Controller

The controller is needed to coordinate the behaviour of the subsys-



tems.

Simulator/Concrete simulator

We can examine the model in the simulator and concrete simulator.

We can populate the Gannt chart as follows;

```
gantt {
NorthSouth: L1.Red -> 0, L1.Green->1, L1.RedAmber->14, Amber->6;
EastWest  : L2.Red -> 0, L2.Green->1, L2.RedAmber->14, Amber->6;
}
```

Examining the behaviour in the Concrete Simulator shows a problem with the sequence, shown in figure 1. It is possible to start the sequence turning the North-South lights to green, before the East-West lights have turned Red.

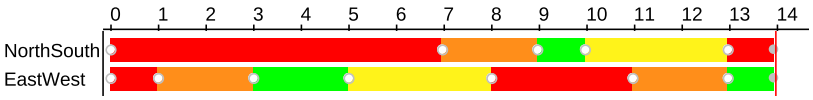


Figure 1: Sequencing problem

We need to modify our system to inhibit this behaviour. The controller needs to know when it can start the next sequence turning to green. We could add the timing to the controller. As we already have the timing in the lights, the better approach is to add some additional signalling.

Revised Lights

`TF2.xml`

The revised lights with the additional signalling is shown in figure 2 and the controller in figure 3.

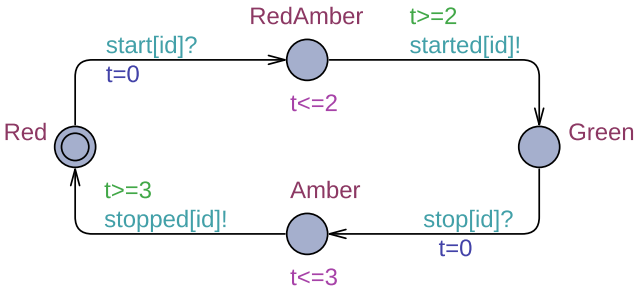


Figure 2: Revised Traffic-Lights

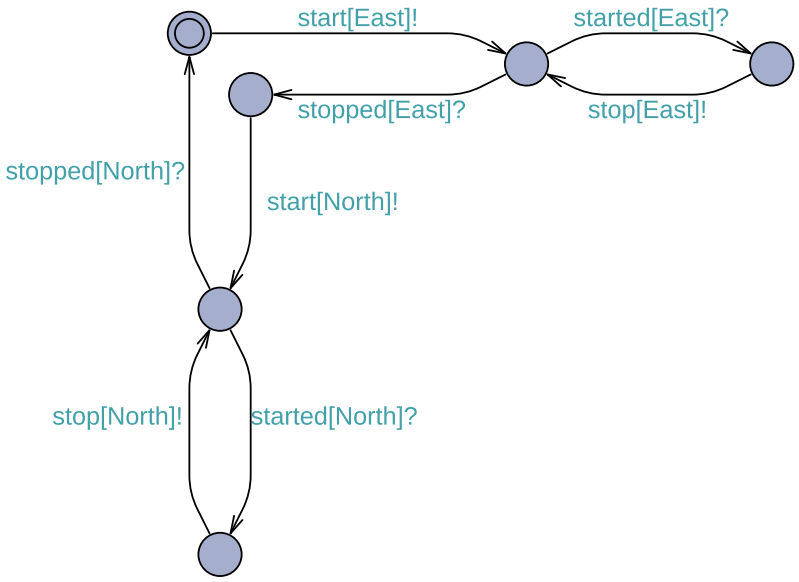


Figure 3: Revised Controller

The added states in the controller (a *'Changing"*) state restrict the availability of the *"start"* edge until the other lights have finished changing.

The needs two extra channels in the global declarations. The revised lines from the global declarations are shown below.

```
chan   start[TLset], started[TLset];
chan    stop[TLset], stopped[TLset];
```

Verification

We want to be able to verify some properties of the system.

Lights will be Green eventually

```
E<> L1.Green
E<> L2.Green
```

Both sides will never be green always

```
A[] not( L1.Green and L2.Green)
```

Both sides will not be changing at the same time, always

```
A[] not(L1.Amber and L2.Amber)
A[] not(L1.RedAmber and L2.RedAmber)
A[] not(L1.Amber and L2.RedAmber)
A[] not(L1.RedAmber and L2.Amber)
```

ALL THESE PROPERTIES ARE TRUE. In the first version the invariant property that both lights are not green passes, the invariants that both lights are not changing at the same time fail.

Adding Pedestrians

`TF3.xml`

Now we have a working set of traffic lights, we can add a pedestrian crossing.

Pedestrian Lights

The pedestrian lights are a simple *Red–Green* sequence. The Green light is on for between 4s and 10s, figure 4

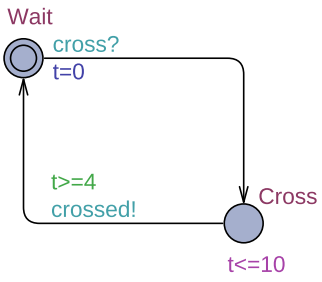


Figure 4: Pedestrian Lights

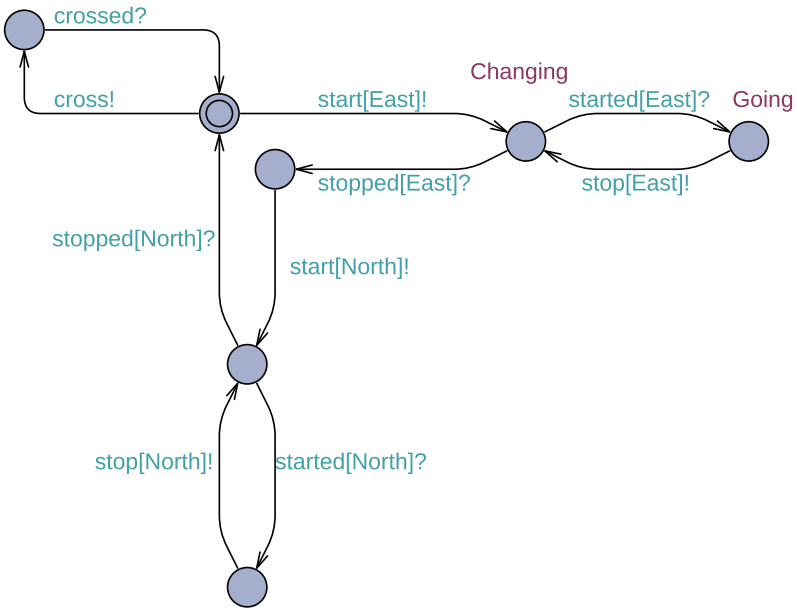


Figure 5: Crossing control with pedestrians

Controller

The controller need additional edges to a crossing state for the pedestrians. The structure shown in figure 5 only allows pedestrians to cross after both sets of traffic has been allowed.

This can be resolved by adding a second path controlling the pedestrians from both stopped states.

Deciding to cross? There is nothing in the controller to decide between starting the traffic and allowing pedestrians to cross. We can add a flag and guard conditions to allow pedestrians to cross if any are waiting.

`TF4.xml`

The Waiting Flag is set by a simple model of the button that the pedestrians push to indicate they want to cross.

Figure 7 shows the model of the button and figure 8 shows the revised pedestrian crossing that clears the waiting flag.

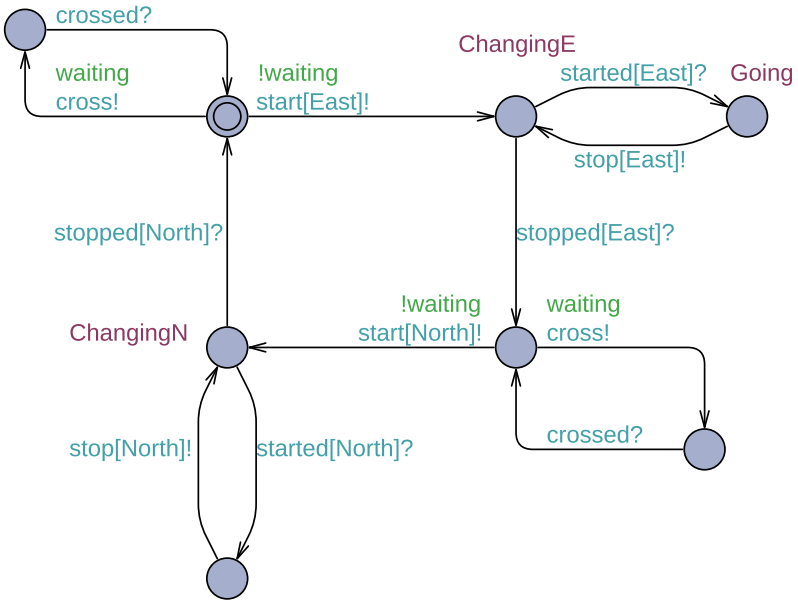


Figure 6: Controller with Crossing on demand

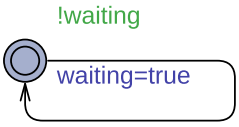


Figure 7: Button model

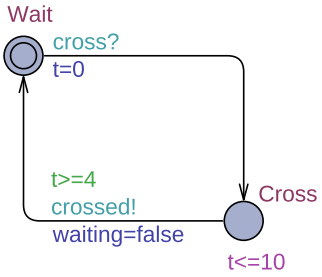


Figure 8: Pedestrians

What is not modelled

The model at this point is quite comprehensive, it is still missing some features

1. There is nothing to trigger a change in the lights from Green. We can wait infinitely long in the state where one set of traffic is going.
2. There isn't a model of the movement of traffic and people through the crossing. Is this needed?

References

Gerd Behrmann, Alexandre David, and Kim G Larsen. A tutorial on uppaal 4.0. *Department of computer science, Aalborg university*, 2006. URL `https://www.it.uu.se/research/group/darts/papers/texts/new-tutorial.pdf`.

A David, T Amnel, M Stigge, and P Ekberg. Uppaal 4.0: Small tutorial, 2011. URL `https://www.it.uu.se/research/group/darts/uppaal/small_tutorial.pdf`.

Alexandre David, Kim G Larsen, Axel Legay, Marius Mikučionis, and Danny Bøgsted Poulsen. Uppaal smc tutorial. *International Journal on Software Tools for Technology Transfer*, 17(4):397–415, 2015. URL `https://www.it.uu.se/research/group/darts/papers/texts/uppaal-smc-tutorial.pdf`.