Software Architecture and Calculi - Assignment 1

Design and analysis of a cyber-physical system

a85700 Pedro Costa
a85954 Luís Ribeiro

Mestrado em Engenharia Informática
Universidade do Minho

Contents

1 | Introduction

In this project, its described a scenario composed by a major road and a minor road, where each road have traffic lights in it. In the first part of this assignment, we assume that only the minor road has a sensor providing information about the traffic. We also assume that the major road has priority over the minor road, implying that traffic in the major road is higher. But such an assumption is often too strong, therefore the second part of this assignment aims to find a fair approach to this problem.

2 | First Part

In the first part of this assignment it was requested to *design* and *analyse* a system that ensures the correct functioning of traffic lights at a T-junction. The latter connects a “major” and a “minor” road.
There were some properties that had to be ensured about the functioning of the traffic lights that shall be proved on subsection Background properties.

2.1 Design choices

Since both traffic lights on major road have a "mirrored" behaviour we decided to model both of them as one entity alone. Meaning that the concept of the traffic light can be abstracted by its road, and both traffic lights in the major road act as one, since they don't have a sensor attached to it.
Our approach was very straightforward, consisting of three templates:

1. Treats the major road lights.
2. Acts as the minor road light.
3. Sensor, which purpose is to synchronize the opposing lights.

Both templates that represent the roads are similar, where we added the locations corresponding to its representing lights colour, and some other to respect and represent the delay between switching one light off and the other on. These switches between lights must be synchronized between roads.
Here, the sensor dictates how the system evolves, since the major road will remain on green unless the sensor provides information about the traffic in the minor road. As stated, after providing information, the system will allow traffic to leave the minor road and the sensor will disable itself until the minor-road lights are on red again (30 seconds given to the minor road). So, the sensor will be defined by 2 locations, one being the waiting state and the other being the disabled state. The clock declared in this template represents the time passed between the state where the sensor triggers the minor road to go green and the state where the minor road returns to red.

2.2 UPPAAL Models

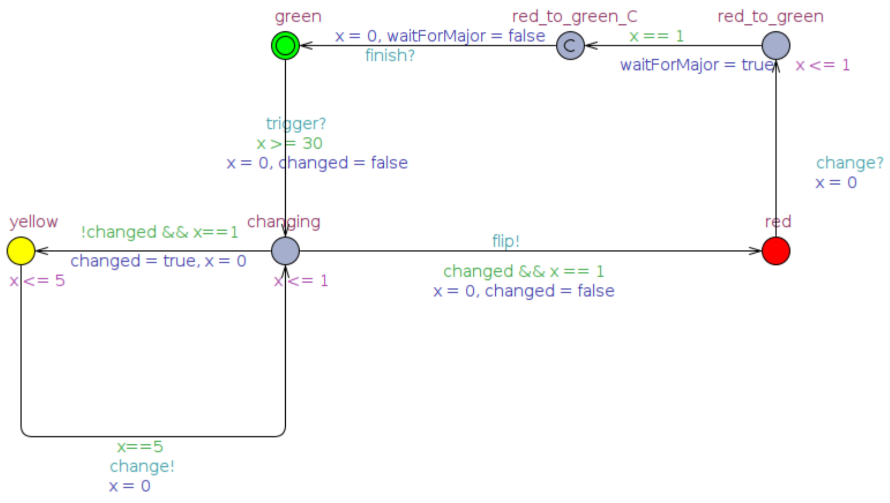

Figure 2.1: Template 1 - Major road

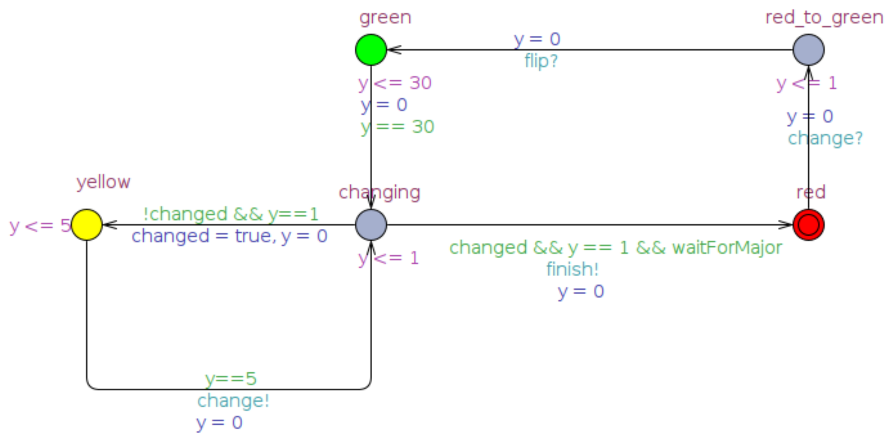

Figure 2.2: Template 2 - Minor road

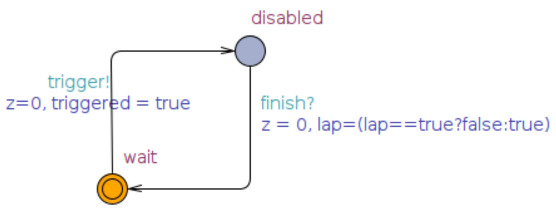

Figure 2.3: Template 3 - Sensor

2.3 Properties

2.3.1 Background properties

As mentioned in the introduction of this chapter, there were some behavioural and temporal constraints that were required to hold as true regarding the system in development. We confirmed them using UPPAAL query system, obtaining the following results:

- The lights on the major road will be always set on green and red on the minor road unless a vehicle is detected by the sensor.

```
        A[] !triggered imply MajorRoad.green and MinorRoad.red
```

- After a suitable time interval (30 seconds), the lights will revert to their default position so that traffic can flow on the major road again (we use 37 since we also need to consider the transition from yellow).

```
        A[] !(MajorRoad.red and MajorRoad.x > 37)
```

- Finally, as soon as a vehicle is detected by the sensor the latter is disabled until the minor-road lights are on red again.

```
        A[] MinorRoad.green imply Sensor.disabled
```

- Interim lights stay on for 5s.

```
        A[] (MajorRoad.yellow imply MajorRoad.x <= 5) and (MinorRoad.yellow
            imply MinorRoad.y <= 5)
```

- There exists 1s delay between switching one light off and the other on.

```
        A[] !(MajorRoad.changing and MajorRoad.x > 1) and !(MajorRoad.
            red_to_green and MajorRoad.x > 1)
        A[] !(MinorRoad.changing and MinorRoad.x > 1) and !(MinorRoad.
            red_to_green and MinorRoad.x > 1)
```

- The major-road light must stay on green for at least 30 seconds in each polling cycle, but must respond to the sensor immediately after that - We ensured this condition is fulfilled by checking that the action **trigger** requires *Sensor.z >= 30.*

2.3.2 Reachibility

The minor-road light can go green:

```
    E<> MinorRoad.green
```

The major-road light can go red.

```
    E<> MajorRoad.red
```

2.3.3 Safety

The system never enters in a deadlock state.

```
A[] !deadlock
```

The minor-road and major-road green lights can not be on at the same time.

```
A[] !(MajorRoad.green and MinorRoad.green)
```

2.3.4 Liveness

If there are cars waiting they will *eventually* have green light.

```
MajorRoad.red --> MajorRoad.green
Sensor.disabled and MinorRoad.red --> MinorRoad.green
```

2.3.5 Extra

The sensor can only be in a reading state while the traffic light on minor-road is red.

```
A[] !(MinorRoad.green and Sensor.wait)
```

The sensor can only be in a post-reading state while the traffic light on major-road is red.

```
A[] !(MajorRoad.green and Sensor.disabled)
```

Disabled sensor only comes back to disabled after being in a wait state.

```
Sensor.disabled and lap --> (!lap)?(Sensor.disabled):(true)
Sensor.disabled and !lap --> (lap)?(Sensor.disabled):(true)
```

The amount of time that passes between green lights on major-road is 44 seconds.

```
MajorRoad.changing --> MajorRoad.red_to_green_C and Sensor.z == 44
```

3 | Second Part

The second part of this assignment aims to address precisely this problem which is well-known to have significant impact in the economy and the environment. To this effect, we can now assume that each traffic light has a smart sensor attached to it. The sensor informs if the traffic near the light is `high`, `low`, or simply `non-existent`.

3.1 Design choices

In this second part we decided that abstracting the idea of every traffic light as the road it belongs to, like in the first part of the assignment, couldn't be possible since each traffic light is linked to a smart sensor and the major road is composed by 2 different lights, where traffic may differ. However, the template of the minor road remained the same since there's just one traffic light in it. Considering that there's different levels of traffic, we decided to declare each level by a corresponding integer, `0` being non-existent, `1` being low and `2` being high. Consequently, were also created variables for each road denoting the related traffic level.

In order to distinguish the major road's lights, we declared an `id` that is given as a parameter of the template that models the idea of both. Although they represent two different lights, they must be synchronized in the way that both should be on the same location (Green, Yellow or Red) at the same time. We also established that if one of the lights gets permission to go green, it must warn the other one to go green as well, resulting in adding more locations to this template. As said above, every road has a matching level of traffic, so we must compute which of the sensors-lights of this road outputs the higher level of traffic.

At first, our approach to the sensors template was a non-synchronized one, meaning that sensors could provide new information about the traffic near them independently. The only restriction given was that if the traffic light near the sensor became red, the sensor couldn't read less traffic from that moment on, unless the light turned green again. Even though this approach seemed promising, it gave us a lot of problems becoming complex and unreadable, as we had to add restrictions so the system didn't stall. At some point, we had to ensure that neither sensor could read while the lights were changing, and they could change back instantly if the other sensor read more than the latest. Although we got this approach working, we decided to go for a synchronized one where every sensor provide information simultaneously.

As mentioned above, our final approach was to synchronize every sensor. For that reason, we created another template, `SynchronizeS`, for synchronization purposes, simplifying the sensor's templates, where this last only reads the traffic into this new template. Meaning there's a location that gets information of every sensor before taking a transaction. This location also computes how the system evolves, whether allows light switching between roads or not, depending on the traffic measured by each sensor and other fairness properties that should be considered. These properties mentioned will be explained further on. After this process of determining how to manage the roads, this template will evolve to a timed location, before reading the sensor's information once again, giving allowance to one of the roads to stay on green for a period. For the most part, this time given is 30 seconds, however it can be lower as one of the roads may give permission to allow traffic flow on the other, even if it has higher traffic. There was also added a clock that records the time passed between traffic reads, representing the period given to stay on green plus the lights switching delay, if a switch between roads occurs (`clock transiction_time`).
After introducing the design of this template, we will proceed to explain some details regarding the evolution of the system, upon reading the traffic levels.

3.2 UPPAAL Models

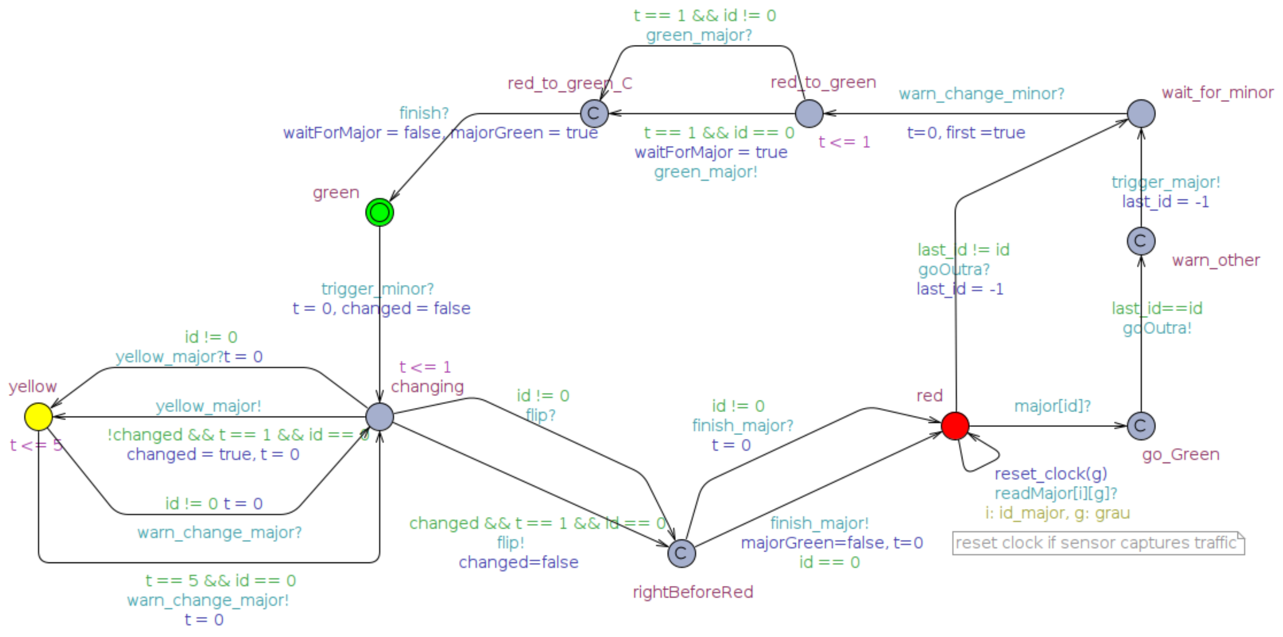

Figure 3.1: Template 1 - Major road

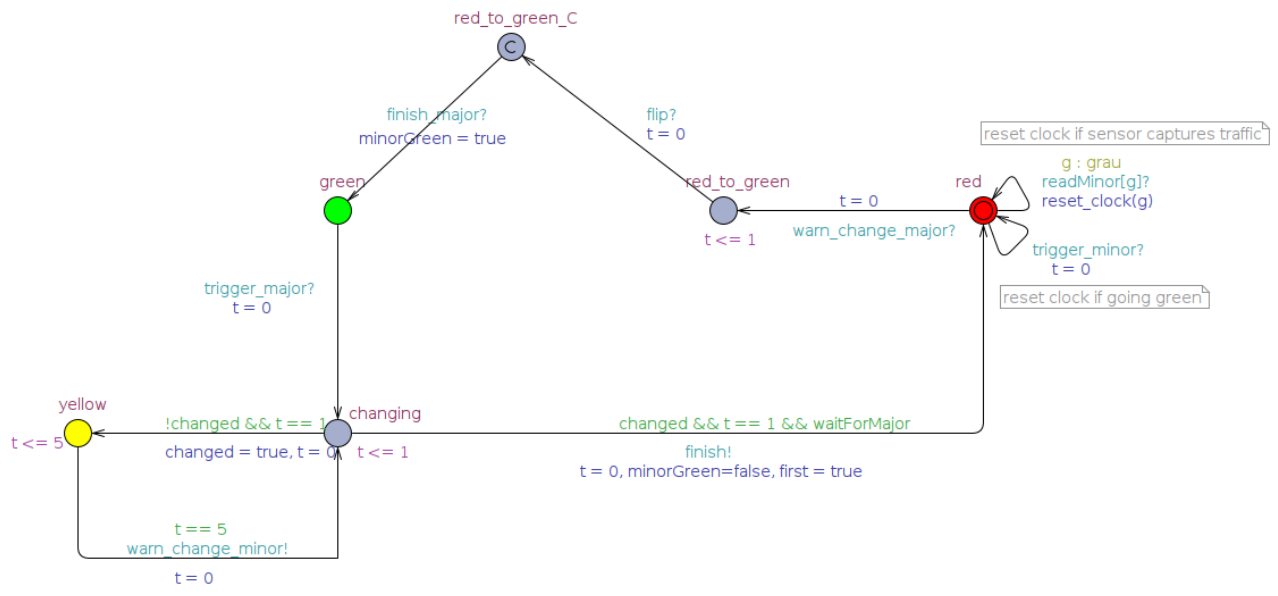

Figure 3.2: Template 2 - Minor road

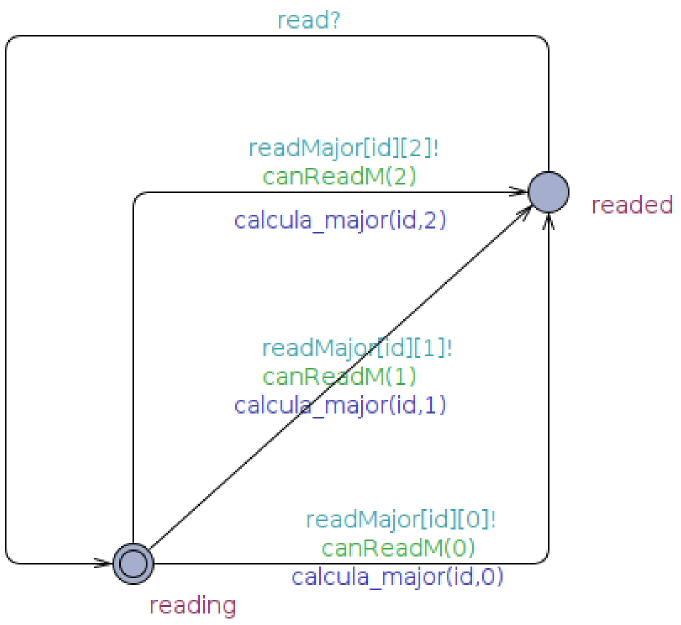

Figure 3.3: Template 3 - Sensor Major

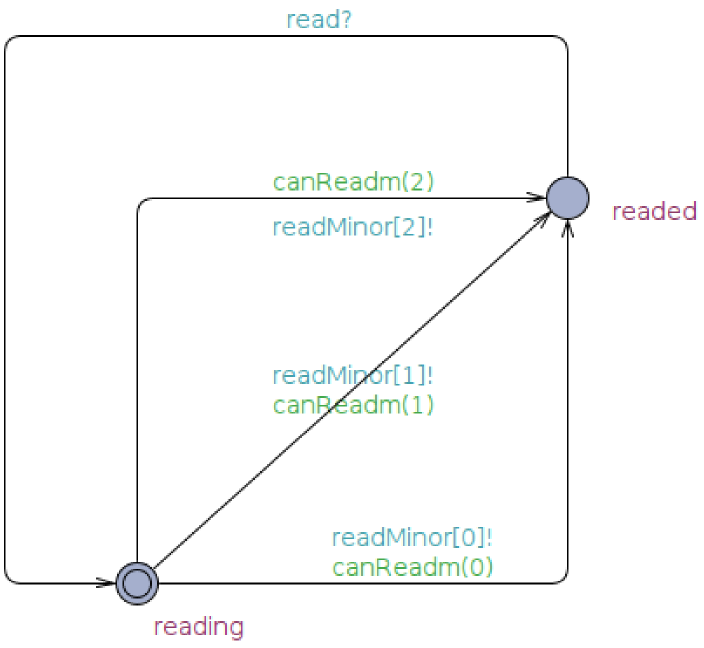

Figure 3.4: Template 4 - Sensor Minor

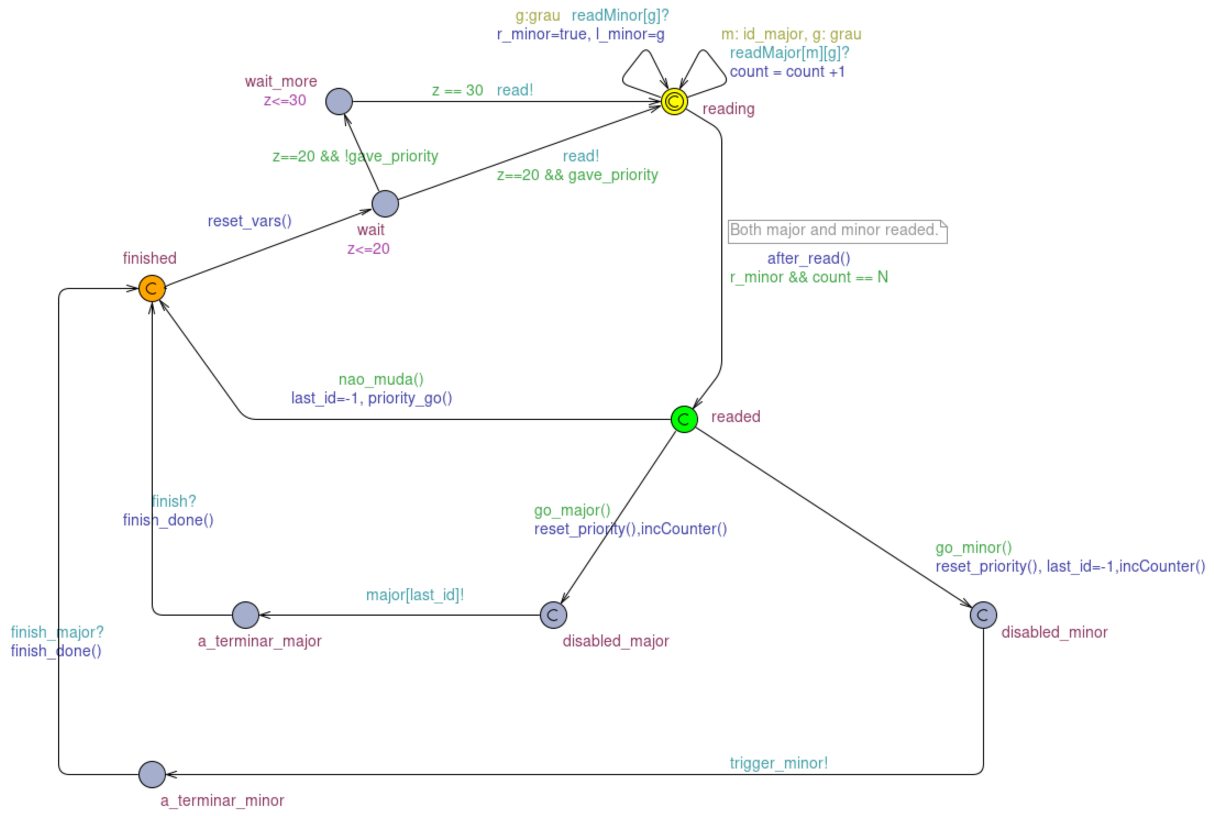

Figure 3.5: Template 5 - SynchronizeS

As mentioned before, we still want to explain a bit better the decision process behind SynchronizeS. It is used to decide what happens provided the information regarding traffic intensity levels in each road and, therefore, it breaks possible situations into one of three scenarios.

- No change: Happens whenever the most intense road doesn't change and there's also no need to give green light to the other road for fairness motivations.
- Major gets green light: Happens when either the major road is on red light and has more traffic intensity than the minor road or, when major road has less intense, but some, traffic than the minor road but the latter has given priority to major due to fairness reasons.
- Minor gets green light: The exact opposite of the previous situation. Either the major has more traffic but gave in priority to minor, or the minor has more traffic and is currently on red light, or the last road to be on green was major but since the traffic level is identical, it becomes minors turn to get green.

3.3 Previous part queries

As requested in exercise 2 of this second part of the assignment, we have confirmed that all properties mentioned in the first part of the assignment still hold. There's no point in presenting them here since their presence can be checked directly on the UPPAAL queries.

3.4 Benchmarking and new queries

3.4.1 Safety

Both major-road traffic lights are synchronized (both at green at the same time).

```
A[] (MajorLight(0).green imply MajorLight(1).green)
A[] (MajorLight(1).green imply MajorLight(0).green)
```

3.4.2 Liveness

MajorRoad will be green if it has at least low traffic, even though traffic in MinorRoad is higher.

```
MajorLight(0).red && MajorLight(1).red && lvl_major < lvl_minor && lvl_major > 0
    --> MajorLight(0).green && MajorLight(1).green
```

MinorRoad will be green if it has at least low traffic, even though traffic in MajorRoad is higher.

```
MinorLight.red && lvl_minor < lvl_major && lvl_minor > 0 --> MinorLight.green
```

3.4.3 Benchmark queries

Traffic cannot diminish if the road light is red.

```
forall(id:id_major) MajorLight(id).red && lvl_major==1 --> lvl_major >= 1
forall(id:id_major) MajorLight(id).red && lvl_major==2 --> lvl_major >= 2
MinorLight.red && lvl_minor == 1 --> lvl_minor >= 1
MinorLight.red && lvl_minor == 2 --> lvl_minor >= 2
```

If the MinorRoad sensor is always detecting high traffic and the other sensors do not detect any traffic, then we observe a maximum of 1 signal exchange.

```
lvl_minor == 2 && lvl_major == 0 && flipCounter <= 1 --> lvl_minor == 2 &&
    lvl_major == 0 && flipCounter <= 1
```

If both roads have the same traffic level then the lights will alternate constantly.

```
lvl_minor == lvl_major && SynchronizeS.readed && lvl_minor > 0 && majorGreen -->
    lvl_minor == lvl_major && lvl_minor > 0 && !majorGreen
```

If a road has traffic, the waiting time before turning green is less than 60 seconds.

```
A[] forall(i:id_major) lvl_major>0 && MajorLight(i).red imply MajorLight(i).t<=60
A[] lvl_minor > 0 && MinorLight.red imply MinorLight.t <= 60
```

The time passed between two traffic reads is at most 37 (30 on green plus 7 on lights changing) seconds.

```
A[] transiction_time <= 37
```

If one road gets permission to release traffic (the other road gave permission to allow traffic flow), the transiction time is always 27 (20 on green plus 7 on lights changing) seconds, before reading again.

```
A[] (SynchronizeS.gave_priority && SynchronizeS.reading) imply transiction_time
    == 27
```

4 | Conclusion

Arriving to the end of the assignment, we consider that we have completed everything that was asked to be done. We have modelled a smart semaphore system that respects some basis rules and focus on giving a smooth experience to every driver, reducing their lost time and sense of frustration.

Modelling time-critical reactive systems as this one isn't an easy task, in the sense that debate takes a huge part in determining the start-up process. As we were starting to reason for the right solution, we had to make definitive decisions that ended up leading us to our current solution.

One of our main struggles was the rapidly increasing complexity of the templates. We believe that our inexperience plays a part in this but it still makes us wonder how would we approach a more complex example.

Overall, we consider that this assignment went fairly well and helped us comprehend the difficulties in modelling a real life situation whilst also motivating us too find better solutions to our everyday problems because they can be improved if the effort is put.