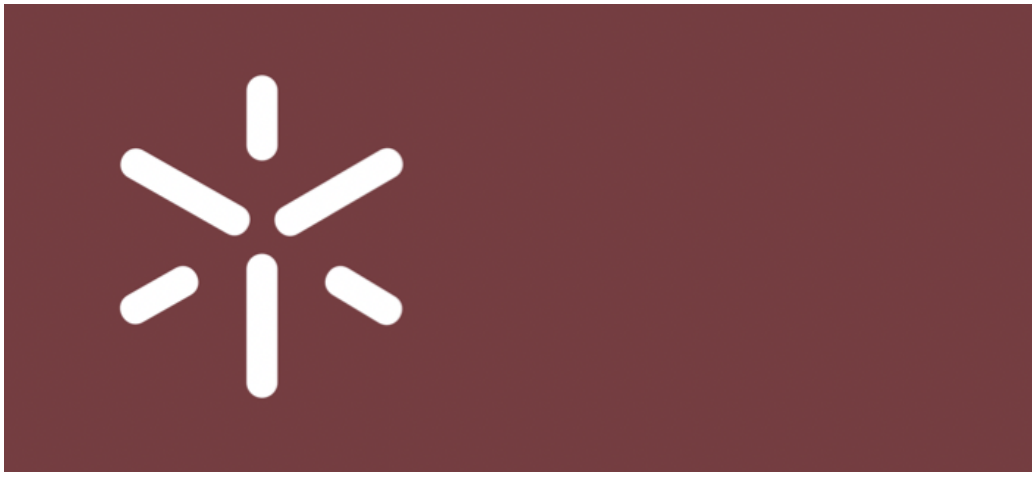

UNIVERSIDADE DO MINHO

Arquitetura e Cálculo

TP1 - Design and analysis of a cyber-physical system

Rafael Lourenço
(A86266)

Rodolfo Silva
(A81716)

Introdução

Os semáforos habitualmente implementados são um meio de **poluição** indireta, visto que, enquanto um veiculo está parado num semáforo, o combustível continua a ser queimado produzindo assim vários gases poluentes para a atmosfera. Um estudo realizado nos USA , concluiu que mais de **10 mil milhões** de litros de combustível são queimados nestas circunstancias.[1]
No âmbito da UC de Arquitetura e Cálculo do perfil de mestrado MFES da Universidade do Minho, foi nos proposto um projeto que consiste na modelação de um possível semáforo inteligente. Para isso, este relatório é constituído por uma versão inicial da modelação de um semáforo comum, para introdução do conceito, contendo posteriormente a modelação do semáforo inteligente, e por fim a especificação das propriedades para garantir o funcionamento dos dois sistemas.

1ª Parte

Esta parte do projeto consiste em modelar um sistema de semáforos num entroncamento, ou seja, na interseção de 2 estradas onde uma delas é a principal (major road), e a outra a secundária (minor road). Nesta primeira parte, assume-se que na rua principal irá existir maior tráfego. Neste sistema existem 4 entidades principais :

- **Carros**

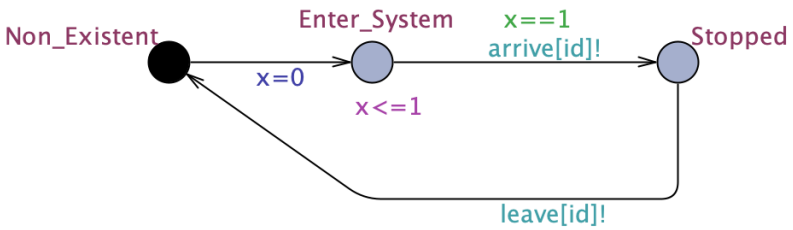

Figura 1: Modelação da chegada/saída de carros ao sistema

Os carros são representados por um **Id**, podendo permanecer em três estados possíveis, sendo estes o *Non_Existent* , *Enter_System* ou *Stopped*. O estado inicial é o *Non_Existent* e de forma a simular a chegada de um carro, colocou-se um estado intermédio *Enter_System*,para este demorar 1 segundo a entrar no sistema, pois, caso não houvesse este *delay* eles poderiam entrar e sair do sistema instantaneamente. Após um segundo, este é registado no sistema através da operação *arrive*, sendo que, permanece nesse estado até o semáforo virar verde. Posteriormente, quando isso se verificar, o carro pode sair do semáforo realizando a operação *leave*, passando assim novamente para o estado *Non_Existent*.

- **Sensor**

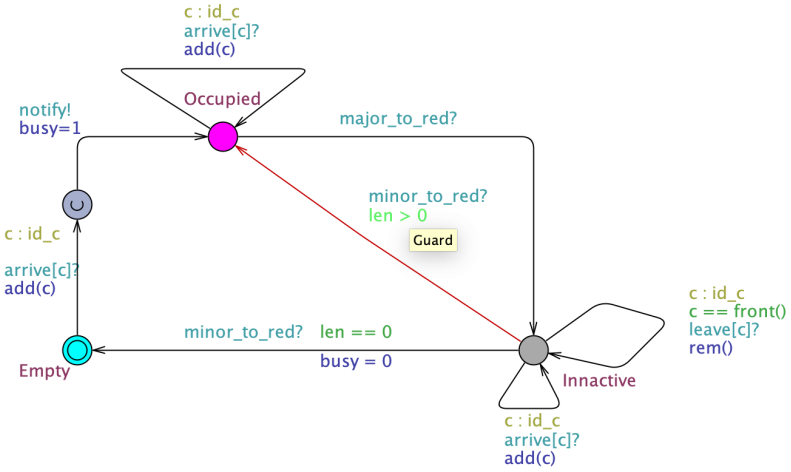

Figura 2: Modelação do sensor da rua secundária

O sensor é usado para detetar a chegada dos carros à rua secundária(*minor road*). Além disso, é o que permite a comunicação entre semáforos e os carros, pois sempre que um carro chega à rua secundária*(arrive)*, este envia uma notificação que será sincronizada com uma ação no semáforo da rua principal*(notify)*, colocando assim o sensor no estado ***Occupied***. Quando o semáforo da rua principal ficar vermelho, este executa a operação ***major_to_red*** que irá estar sincronizada com o sensor e com a **rua secundária**.Assim este avança para o estado ***innactive*** e o semáforo da rua secundária fica verde. Neste momento, os carros podem sair por ordem de chegada, sendo que, para implementar esse comportamento foi utilizado um *array*. Esta operação sincroniza-se com a **leave** do sistema dos carros. Caso todos os carros consigam arrancar antes do semáforo*(minor road)* ficar vermelho, o sensor avançará para o estado ***Empty***, caso contrário o sensor irá para o estado ***Occupied*** novamente.

- **Major Road**

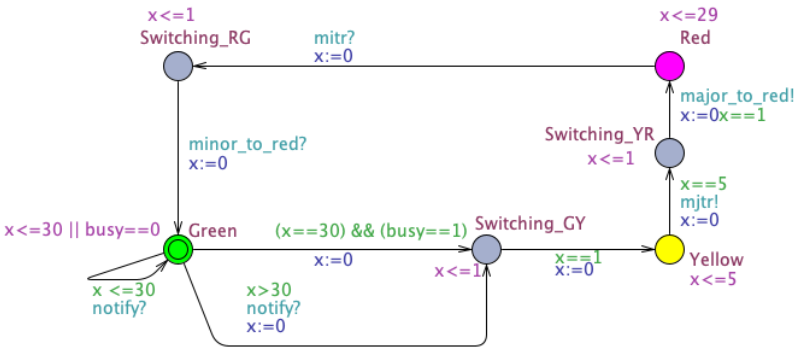

Figura 3: Modelação do semáforo da rua principal

O estado inicial deste semáforo é verde, sendo que têm de permanecer nesse estado pelo menos 30 segundos. Após o sensor receber um carro, este pode ficar amarelo caso já tenha passado esse tempo,

caso contrário o carro tem de esperar que o relógio chegue aos 30 segundos. Assim, quando essas restrições forem satisfeitas, este pode mudar para amarelo(***yellow***), passando por um estado intermédio(*Switching_GY*) que é usado para simular o delay da transição. Após passar 5 segundos no amarelo, este irá para o vermelho, simulando mais um segundo de *delay* com o estado *Switching_YR*. Quando este passa a vermelho, é sincronizada as operações ***major_to_red*** e **mjtr** com a rua secundária, o que permite a este passar novamente para verde.

- **Minor Road**

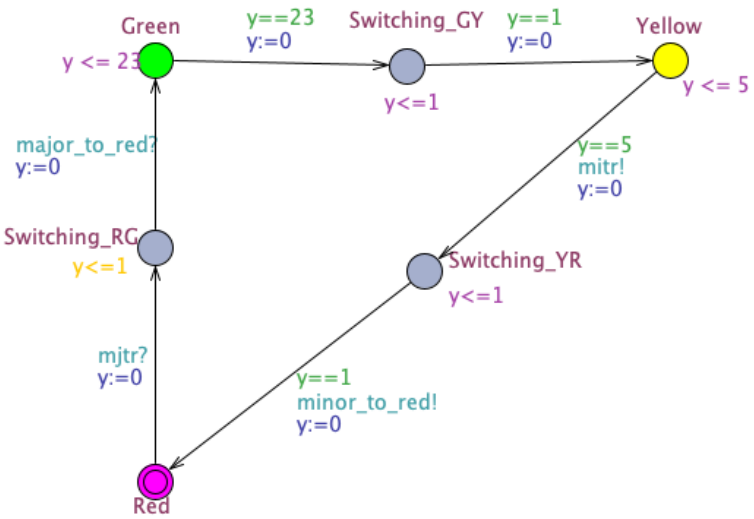

Figura 4: Modelação do semáforo da rua secundária

O estado inicial deste semáforo é vermelho, visto que, só poderá virar verde quando um carro é detectado pelo sensor. Após o sensor comunicar com a rua principal a chegada de um veículo, caso já tenham passado 30 segundos explicados anteriormente, a rua secundária irá esperar que a rua principal mude de amarelo para vermelho. Nesse instante são executadas as duas operações que simulam o *delay* da transição, sendo que após este, o semáforo da rua secundária vira verde e o semáforo da rua principal vira vermelho.
Desta forma, o semáforo desta rua pode permanecer apenas 23 segundos no verde, uma vez que, o objectivo é que o semáforo após 30 segundos volte a ficar vermelho (cada ciclo 30 segundos). Assim após 23 segundos é feita a mudança de estado para amarelo onde irá permanecer 5 segundos,sendo que posteriormente volta a vermelho. Dado que cada transição demora 1 segundo, completa-se assim um ciclo de 30 segundos.

2ª Parte

Para a segunda fase do projeto tomou-se algumas liberdades em como construir o mesmo. Considerou-se um numero de carros igual a 5, por questões de performance. Para a rua secundária esta tem três estados de ocupação, vazia, quando o seu sensor deteta 0 carros, ocupação baixa, quando deteta 1 carro e por fim, ocupação alta, quando deteta 2 ou mais carros. No caso da rua principal esta também vai possuir 3 estados de ocupação mas com valores diferentes, quando deteta 2 carros esta tem ocupação baixa, mas ao detetar 3 ou mais carros a ocupação desta já passa a ser alta. Foi feito desta maneira para dar ênfase ao facto de

a rua principal tem 2 semáforos enquanto a secundária tem 1. Outra consideração que foi tomada, foi a junção de ambos os sensores e semáforos da rua principal, num só, de forma a facilitar o desenvolvimento do modelo.

- **Carros**

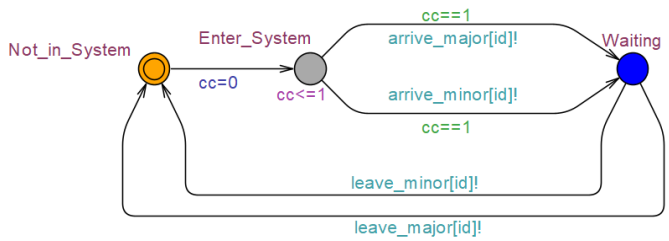

Figura 5: Modelação da chegada/saída de carros ao sistema

De seguida têm-se a modelação dos carros, estes necessitam de 1 segundo para conseguirem entrar no sistema, a seguir do qual, vão poder chegar a um semáforo da rua principal ou ao semáforo da rua secundaria. Depois disso eles possuem o *urgent chan leave_minor* ou *leave_major* de modo a, garantir que mal o seu semáforo vira verde, eles saem do sistema.

- **Major Road**

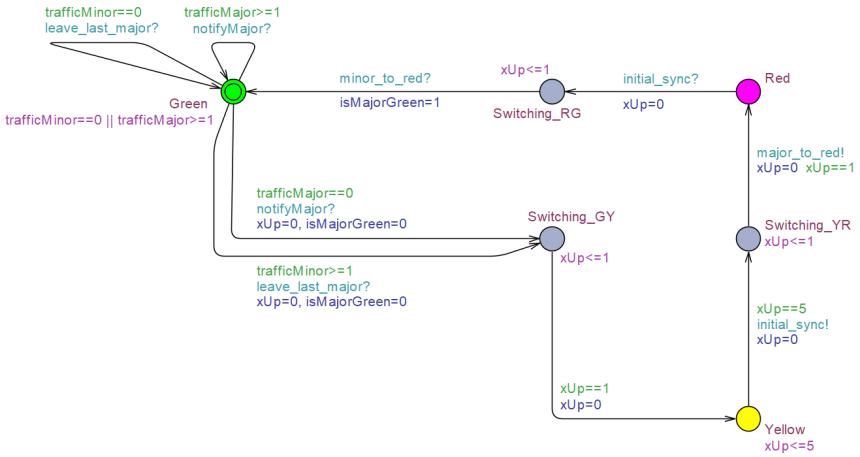

Figura 6: Modelação do semáforo da rua principal do sistema

Tal como anteriormente, o estado inicial do semáforo da rua principal vai ter luz verde.Este pode receber e deixar passar carros enquanto está verde.Todavia, no instante que sejam detetados carros em espera na rua **secundária**, este vai mudar para amarelo mal os carros detetados pelo sensor da rua **principal** saiam todos do sistema. Vai demorar 1 segundo entre cada transição e passar 5s com as luzes amarelas até eventualmente chegar a vermelho.Deste modo, estando em vermelho, este vai esperar pela sincronização com o semáforo da rua secundária para poder passar novamente a verde.

- **Minor Road**

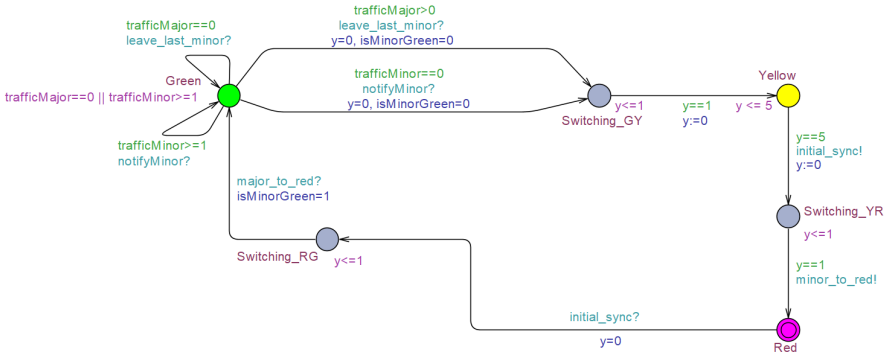

Figura 7: Modelação do semáforo da rua secundária do sistema

O semáforo da rua secundária tem como estado inicial a luz vermelha. No inicio, se este receber carros, vai esperar pela sincronização por parte do semáforo da rua principal para poder mudar para verde. Depois, estando em verde, vai permitir que os carros que estavam a espera na rua secundária saiam do sistema. Caso hajam carros à espera na rua principal, este vai permanecer verde até que todos se retirem.Após ser notificado que existem carros à espera na outra rua, caso os seus sensores não detetem nenhum carro, este vai mudar a luz para amarelo e subsequentemente para vermelho.

- **Major Sensor**

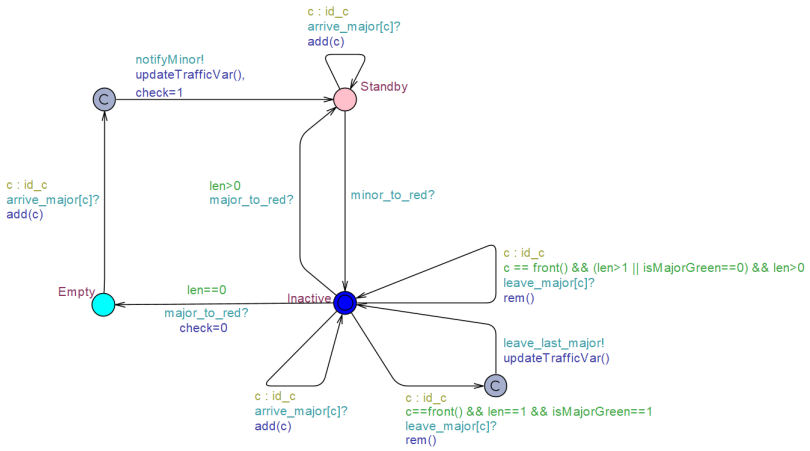

Figura 8: Modelação do sensor da rua principal

Este sensor vai monitorizar o número de carros que estão na área do semáforo da rua principal. Este, não só serve para notificar os semáforos de quando devem trocar de luz, como também para manusear as saídas dos carros do sistema e assim, atualizar as filas de espera dos carros. Este dispõe de 3

estados distintos. No caso do sensor da rua principal, este começa em modo *inactive* pois, o semáforo encontra-se verde. Os outros estados dependem da quantidade de carros na fila de espera, uma vez que, quando ele muda para vermelho, pode ir tanto para o estado *Empty*(se não houver mais carros) ou *Standby*(caso contrário).

- **Minor Sensor**

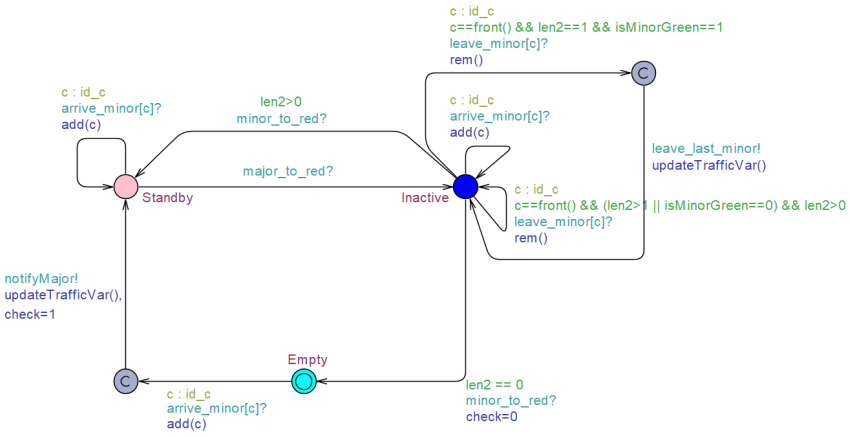

Figura 9: Modelação do sensor da rua secundária

Por fim, temos o sensor da rua secundária que, não só vai monitorizar o número de carros que estão na área do semáforo dessa rua, como também notificar os semáforos de quando devem trocar de luz e ainda consegue manusear as saídas dos carros do sistema(que estão na rua secundária). Este sensor também possuí 3 estados distintos. Neste caso, o sensor vai ser inicializado no estado *empty* pois, não possui nenhum carro em espera no estado inicial.Depois de receber um carro, este vai transitar para o estado *Standby* e no caminho,irá avisar o semáforo da rua principal que existem carros à espera, para poder assim transitar para o estado *Inactive*.Eventualmente, o que corresponde ao semáforo da rua secundária mudar para verde através de uma sincronização de *broadcast*.

Propriedades

As propriedades servem para verificar se o comportamento do sistema é adequado ao objectivos em mente, sendo que, para isso é usado um *model checker* para testar a sua veracidade. Estas dividem-se em 4 tipos:

Deadlock

Para verificar se o sistema entrava em *deadlock*, ou seja, se o sistema chegar a um estado onde é impossível a evolução do modelo para um estado sucessor, nem realizar um *delay*, nem ocorrer uma transição, este encontra-se em *deadlock*. Para isso foram feitas duas propriedades que são equivalentes:

```
A[] not deadlock                    E<> deadlock
```

Tanto no modelo da parte 1, como para o modelo da parte 2, a primeira propriedade dá verdadeiro. Já a segunda dá falso,garantindo assim que não existem deadlock's.

Reachability

Este tipo de propriedades servem para garantir que, em algum momento da execução do sistema, este alcança um certo estado. De seguida, são apresentadas as propriedades comuns aos dois sistemas:

```
E<> Minor_Road_Sem.Green #Existe algum estado onde o semáforo da minor road chega a
verde
E<> Major_Road_Sem.Red  #Existe algum estado onde o semáforo da major road chega a
vermelho
```

Ambas são verdadeiras, nos dois sistemas criados.
Posteriormente, são mostradas as propriedades específicas ao modelo 1:

```
E<> Sensor.Innactive  #Existe algum estado onde o sensor passa a inativo
E<> Sensor.Occupied   #Existe algum estado onde o sensor passa a ocupado
```

Estas também são ambas válidas. Dado isto, serão agora apresentadas as propriedades válidas no modelo 2:

```
E<> Sensor_Major.Standby #É possível chegar a um estado onde o sensor espera
permissão para mudar para verde.
E<> Sensor_Minor.Inactive #É possível chegar a um estado onde o sensor permite a
entrada dos carros
```

Ambas as propriedades são verdadeiras.

Safety

Passando agora a explicar as propriedades de *safety*,estas servem para verificar se alguma condição é verdade para todos os estados ao longo da sua execução,podendo ser exprimidas para todos os caminhos possíveis da *CTL* ou apenas só para um.Sendo assim apresenta-se as propriedades de *safety* comums aos dois modelos:

```
#As cores dos semáforos tem de ser sempre diferentes.
A[] not(Minor_Road_Sem.Green &&  Major_Road_Sem.Green)
A[] not(Minor_Road_Sem.Yellow &&  Major_Road_Sem.Yellow)
A[] not(Minor_Road_Sem.Red &&  Major_Road_Sem.Red)
```

Estas propriedades também foram verificadas com sucesso.
Assim sendo, são apresentadas as propriedades específicas ao modelo 1:

```
A[] Sensor.Innactive imply Major_Road_Sem.Red #Sempre o que sensor permite passar
carros na rua Minor o semaforo da rua Major tem de estar vermelho.
A[] Sensor.Occupied imply not (Minor_Road_Sem.Green) #Sempre o que sensor não permite
passar carros na rua Minor o semaforo da rua Major tem de estar verde.
```

Por último serão apresentadas as propriedades referentes ao modelo 2:

```
E[] Sensor_Minor.Inactive imply Major_Road_Sem.Red #Existe um caminho onde se
verifica que, quando o sensor da minor road permite a entrada de carros, o
semáforo da estrada principal está vermelho.
A[] Sensor_Minor.Standby imply  not (Minor_Road_Sem.Green) #Sempre que um sensor
está em standby implica que o semáforo da rua principal não esteja verde
A[] Major_Road_Sem.Yellow imply trafficMinor >0 #Sempre que o semaforo da rua
principal for amarelo implica que na rua secundária haja carros
A[] Minor_Road_Sem.Yellow imply trafficMajor >0 #Sempre que o semáforo da rua
secundária for amarelo implica que na rua principal haja carros
```

Todas estas propriedades são verificadas com sucesso.
De forma a mostrar que a eficiência do sistema criado, foi feita a seguinte propriedade:

```
E[] Minor_Road_Sem.Red # Existe um caminho onde em  todos os estados, o semáforo
da minor road fica vermelho
```

Assim mostra-se, que para o caso de não existir carros na rua secundária, o sistema nunca realiza trocas desnecessárias.

Liveness

O objectivo neste tipo de propriedades é garantir que o sistema **progrida**, isto é, uma certa condição tem de ser inevitavelmente verdade. Assim sendo, apresenta-se ,de seguida, as seguintes propriedades realizadas para o modelo 1:

```
Sensor.Occupied --> Sensor.Innactive # Se existirem carros à espera, estes irão
ter, inevitavelmente, um sinal verde
Sensor.Innactive --> Major_Road_Sem.Green #Se o sensor permitir a passagem de
carros, inevitavelmente irá voltar a aparece a major road vai voltar a verde.
```

Ambas são verificadas com sucesso. No entanto, é de realçar que a segunda propriedade seria inválida caso não existisse *delay* na entrada de carros nos semáforos, uma vez que, eles podiam estar sempre a entrar e a sair do sistema em ciclo.
Por fim , apresenta-se as propriedades do modelo 2:

```
Sensor_Minor.Standby --> Minor_Road_Sem.Green #Se estão carros à espera no
semáforo da minor road  estes inevitavelmente vão ter uma luz verde.
Sensor_Major.Standby --> Major_Road_Sem.Green #Se estão carros à espera no
semáforo da major road  estes inevitavelmente vão ter uma luz verde.
(trafficMajor<trafficMinor && Major_Road_Sem.Green)    -->  (Major_Road_Sem.Red)
#Se a minor road tiver menor tráfico que na major road , e o sinal da major road
estiver vermelho, implica que inevitavelmente o sinal da major road vai ser
verde.
(trafficMinor < trafficMajor && Minor_Road_Sem.Green)  -->  (Minor_Road_Sem.Red)
#Se a major road tiver menor tráfico que na minor road , e o sinal da minor road
estiver vermelho, implica que inevitavelmente o sinal da minor road vai ser
verde.
```

Todas estas propriedades foram verificadas com sucesso, no entanto, a prova da veracidade das mesmas encontra-se nas duas figuras que se encontram nos anexos, no final do relatório.

Conclusão

Após o desenvolvimento dos dois modelos do enunciado, consegue-se obter um amplo conhecimento do funcionamento da ferramenta UPPAAL e do conhecimento geral de *timed automata*. O projeto considera-se bem sucedido, apesar de que se podia ter feito algumas melhorias, a nível da eficiência computacional. Uma tal melhoria seria possivelmente o não simular a chegada e saída de carros, apenas simular uma existência arbitrária de tráfego que se classifica como alto ou baixo, isto possivelmente melhoraria a performance do sistema e do seu tempo de execução ao testar as *queries*, apesar de, ser menos complexo e não tanto alusivo à realidade.

Bibliografia

[1] https://ourworld.unu.edu/en/green-idea-self-organizing-traffic-signals

[2] https://www.seas.upenn.edu/ lee/09cis480/lec-part-4-uppaal-input.pdf

[3] http://ppedreiras.av.it.pt/resources/empse0809/slides/TheUppaalModelChecker-Julian.pdf

Anexos

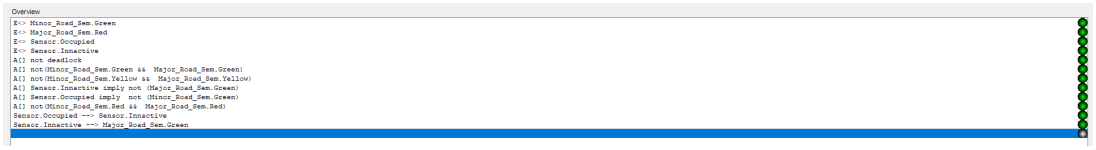

Figura 10: Prova de as *queries* serem verdadeiras no modelo da fase1.

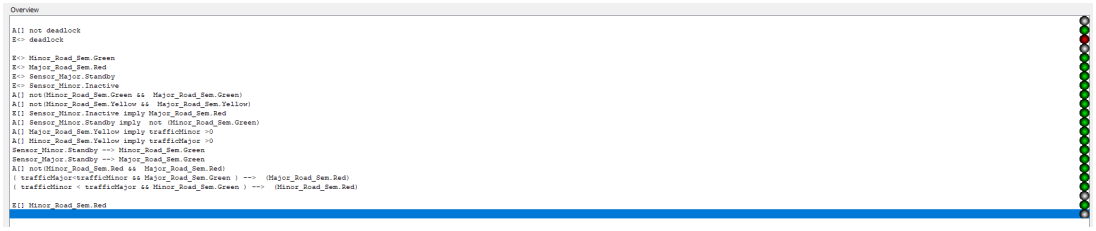

Figura 11: Prova de as *queries* serem verdadeiras no modelo da fase2.