

UNIVERSIDADE DO MINHO

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA

ARQUITECTURA & CÁLCULO

Design and Analysis of a Cyber-Physical System

Etienne Costa (a76089)

Rui Azevedo (a80789)

18 de maio de 2021

Conteúdo

1	Introdução	3
2	Primeira Parte	4
2.1	Modelo	4
2.2	Verificação	5
2.2.1	Reachability	6
2.2.2	Safety	6
2.2.3	Liveness	6
3	Segunda Parte	7
3.1	Modelo	7
3.1.1	Primeira abordagem	7
3.1.2	Segunda abordagem	9
3.1.2.1	Major Road & Minor Road	9
3.1.2.2	Sensor	10
3.1.2.3	Controller	10
3.2	Verificação	11
3.2.1	Eficiência do Sistema	12
3.3	Conclusão	14

Lista de Figuras

2.1	Sistema de tráfego com um sensor	4
2.2	Verificação em UPPAAL	6
3.1	Versão 1 do sistema com controlo de fluxo de tráfego	8
3.2	Versão 2 do sistema com controlo de fluxo de tráfego	9
3.3	Critério de troca de sinais	11
3.4	Verificação das propriedades anteriormente definidas	11
3.5	Verificação das propriedades de eficiência do sistema	13

1. Introdução

O presente relatório é referente ao trabalho prático da unidade curricular **Arquitetura e Cálculo**, do perfil de mestrado **Métodos Formais na Engenharia de Software**, da **Universidade do Minho**.

É pretendido, com este trabalho prático, explorar alguns conceitos de sistemas **ciberfísicos** através da ferramenta de modelação, verificação e validação de sistemas reais, **UPPAAL**. Esta ferramenta utiliza uma rede de *timed automatas* como estrutura de modelação e *Computation Tree Logic* (CTL) como lógica de verificação de sistemas.

A exploração da ferramenta vai ser feita tendo como base um caso de estudo de sistemas luminosos de tráfego, mais especificamente, a análise de uma *T-junction* que conecta uma rua principal e uma secundária.

Na primeira parte do trabalho prático vai ser feito o desenho e análise do sistema assumindo que existe um sensor na rua secundária que é activado quando um carro chega a essa mesma rua.

Na segunda parte do trabalho, o objetivo é desenhar e analisar o mesmo sistema, mas assumindo agora que cada semáforo tem sensores que medem o fluxo de trânsito da respetiva rua.

2. Primeira Parte

A primeira parte do trabalho tem como objetivo desenhar e analisar um sistema luminoso de tráfego. A rua principal contém dois semáforos, um em cada lado da rua, e a rua secundária, por sua vez, contém também um semáforo. A rua secundária, para além do semáforo, contém um sensor que deverá ser ativado sempre que um carro é detetado.

2.1 Modelo

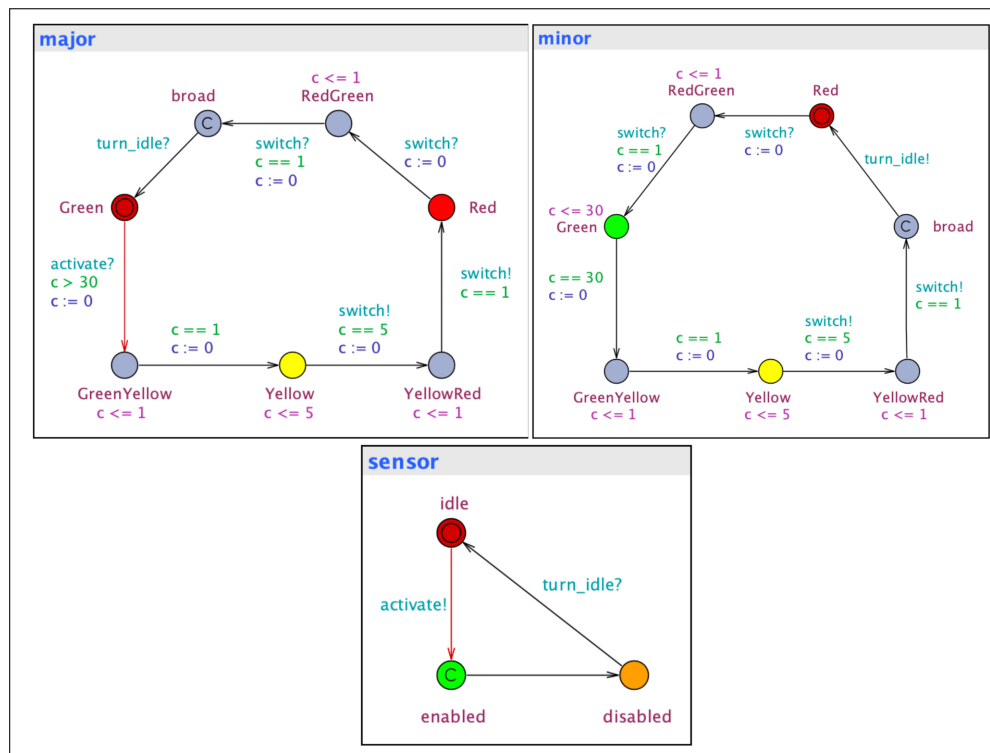


Figura 2.1: Sistema de tráfego com um sensor

Para representar este sistema foram desenvolvidos três *timed automatas*. Um que representa o comportamento dos semáforos da rua principal, outro o comportamento da rua secundária e, por fim, um autómato que representa o sensor da rua secundária. É de notar que, apesar da rua principal ser composta por dois semáforos, apenas existe um autómato para modelar a rua principal uma vez que os dois devem ter exatamente o mesmo comportamento. Dado isto, assume-se então que esse autómato representa o facto da rua principal ter a passagem bloqueada ou não. Cada semáforo contém um relógio local de forma a controlar as restrições temporais impostas pelo enunciado.

O comportamento deste sistema é completamente determinístico no que conta às ações a tomar. Inicialmente, o sensor está na localização *idle*, enquanto o sensor não for ativado pela entrada de um carro. Quando essa ação acontece, o sensor sincroniza com o semáforo da rua principal (que é o que está a verde inicialmente). Ao sincronizarem estas ações, o sensor passa para a localização *enabled* e dá-se início à troca de sinais. O estado *enabled* foi definido como *committed* para que a próxima ação seja desativar o sensor para que o mesmo fique *disabled* até a rua secundária voltar a ter o sinal vermelho.

Os sistemas relativos aos semáforos, são compostos pelas localizações que representam a cor das luzes do semáforo e por localizações que representam os estados de transição entre cores. Estas localizações foram criadas para que seja possível explicitar a existência de limitações temporais entre a troca de luzes dos semáforos, nomeadamente, a troca de cores demora 1s e a cor amarela mantém-se durante 5s. Isto é feito através de invariantes nas localizações e condições nas guardas das transições. Para além destas localizações existe ainda uma outra, *broad*, à qual sai uma transição do tipo *broadcast*, *turn_idle*. Isto é usado quando a rua secundária vai voltar a ter um sinal vermelho e, então, envia um sinal de *broadcast* ao semáforo da rua principal e ao sensor, para que o primeiro fique com o sinal verde e o segundo volte ao estado *idle*, pronto para voltar a detetar entrada de carros. Outras restrições temporais impostas dizem respeito ao facto da rua principal ter que ter o sinal verde no mínimo 30s e, na rua secundária, o sinal verde tem que durar 30s até à troca. Mais uma vez, isto foi resolvido definindo invariantes nas localizações e adicionando guardas nas transições.

2.2 Verificação

Uma vez modelado o sistema de tráfego, foram criadas algumas propriedades em CTL para que sejam verificadas pelo *UPPAAL*, nomeadamente, propriedades de *safety*, *liveness* e *reachability*.

2.2.1 Reachability

A rua principal pode ficar vermelha:

$E\Diamond \text{major.Red}$

A rua secundária pode ficar verde:

$E\Diamond \text{minor.Green}$

2.2.2 Safety

O sistema é *deadlock-free*:

$A\Box \neg \text{deadlock}$

As duas ruas não podem ter ambas o sinal verde:

$A\Box \neg (\text{major.Green} \wedge \text{minor.Green})$

O sinal amarelo nunca fica ligado mais do que 5 segundos:

$A\Box \neg (\text{major.Yellow} \wedge \text{major.c} > 5)$

$A\Box \neg (\text{minor.Yellow} \wedge \text{minor.c} > 5)$

O sinal verde na rua secundária nunca pode estar mais do que 30s ativo:

$A\Box \neg (\text{minor.Green} \wedge \text{minor.c} > 30)$

2.2.3 Liveness

Se existem carros à espera então eventualmente vão ter luz verde:

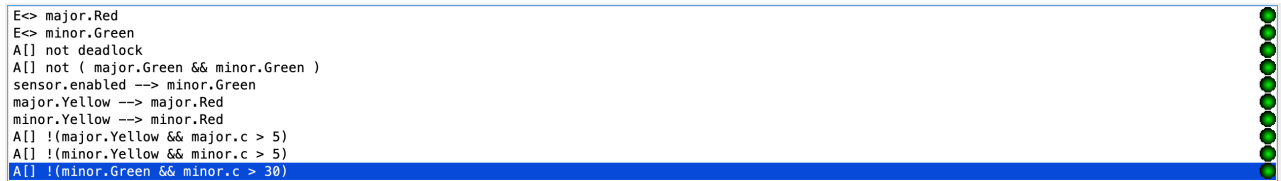
$\text{sensor.enabled} \rightsquigarrow \text{minor.Green}$

Se o sinal de uma rua está a amarelo então eventualmente vai ficar verde:

$\text{major.Yellow} \rightsquigarrow \text{major.Red}$

$\text{minor.Yellow} \rightsquigarrow \text{minor.Red}$

Na figura abaixo pode-se observar as propriedades definidas em *UPPAAL* e o resultado do mesmo. Todas as propriedades definidas são válidas.



```
E<> major.Red
E<> minor.Green
A[] not deadlock
A[] not ( major.Green && minor.Green )
sensor.enabled --> minor.Green
major.Yellow --> major.Red
minor.Yellow --> minor.Red
A[] !(major.Yellow && major.c > 5)
A[] !(minor.Yellow && minor.c > 5)
A[] !(minor.Green && minor.c > 30)
```

Figura 2.2: Verificação em UPPAAL

3. Segunda Parte

A primeira parte do trabalho prático partia da assumption de que existe uma rua (rua secundária) tem muito menos tráfego que a outra. No entanto, existem casos em que o tráfego das duas ruas variam drasticamente, pelo que a primeira solução não é a mais eficiente para este tipo de sistemas. Dado isto, a segunda parte do trabalho tem como objectivo modelar a mesma *T-junction* mas assumindo agora que cada rua tem um sensor capaz de verificar a intensidade do tráfego. O objectivo é que, com base na intensidade do tráfego, os semáforos consigam decidir qual deles fica a verde ou não.

3.1 Modelo

3.1.1 Primeira abordagem

Numa primeira tentativa de criar um modelo para este caso de uso, definiram-se dois *templates*, para além dos que definem os semáforos de cada rua, nomeadamente, um sensor do qual existem duas instâncias do mesmo (um para cada rua) e um controlador, capaz de fazer a decisão de trocar ou não os sinais de trânsito. O sensor definido contém três localizações, uma para cada fluxo de tráfego (*None*, *Low*, *High*). O processo controlador é responsável por fazer leituras num determinado intervalo de tempo e, com base no fluxo de tráfego das ruas, tomar uma decisão se troca ou não de sinais.

Apesar do modelo definido representar o caso de uso, o grupo achou que era um modelo demasiado abstrato pois não se tem controlo sobre a variação de fluxo de tráfego. Este modelo assume que os sensores simplesmente definem o fluxo de tráfego estando implícito o critério da variação do mesmo. Para além disso, era complicado definir o critério de diminuição de fluxo numa determinada estrada quando os sinais da mesma ficassem a verde. Dado isto, decidiu-se criar um modelo menos abstrato capaz de representar ainda melhor o caso de uso.

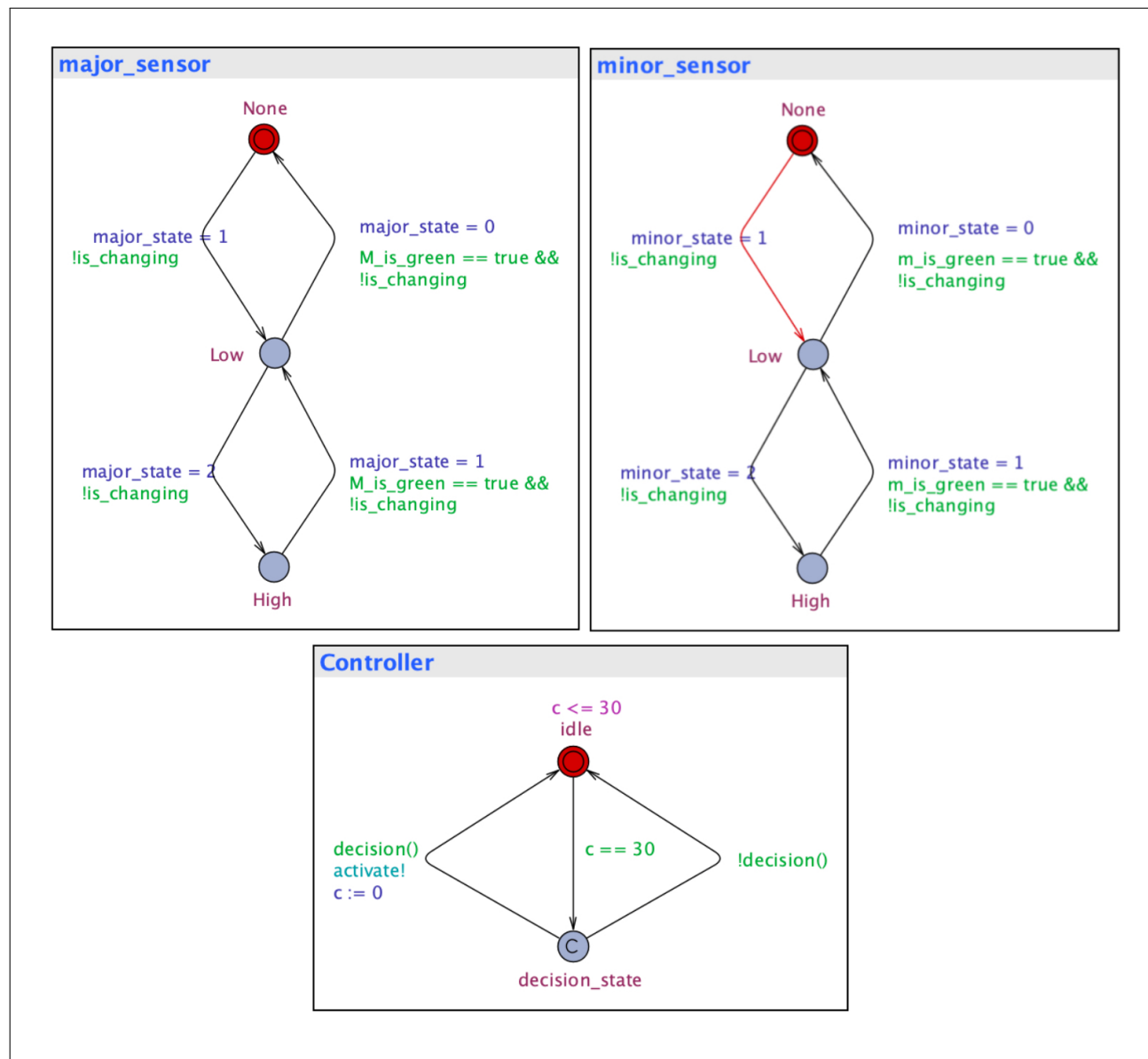


Figura 3.1: Versão 1 do sistema com controlo de fluxo de tráfego

3.1.2 Segunda abordagem

A segunda versão do modelo do sistema desenvolvida é menos abstrata que a anteriormente apresentada uma vez que se sabe à partida o momento em que o fluxo de trânsito muda.

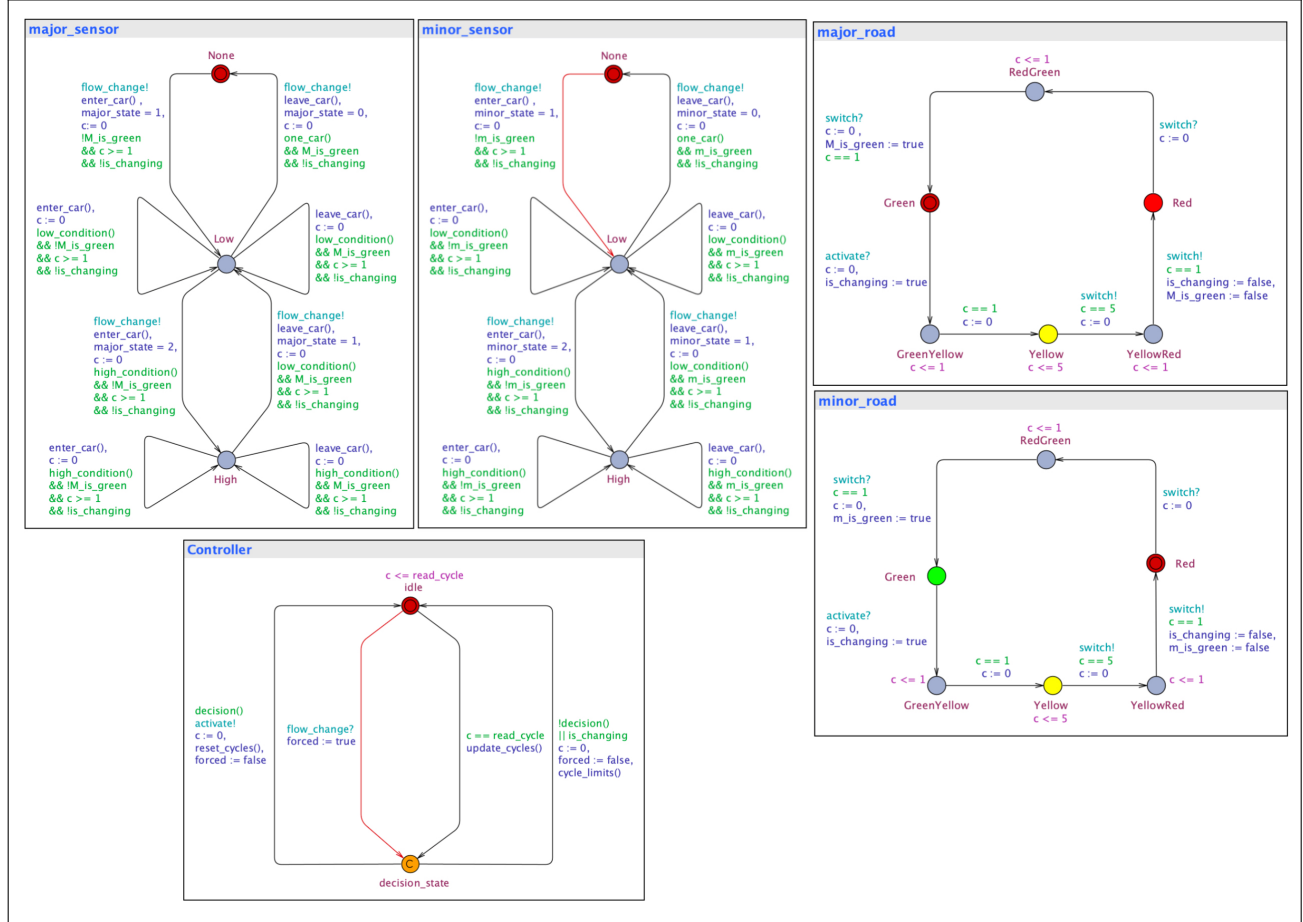


Figura 3.2: Versão 2 do sistema com controlo de fluxo de tráfego

3.1.2.1 Major Road & Minor Road

Em relação aos processos dos semáforos, pode-se observar que são praticamente iguais aos da primeira parte do trabalho, apenas se removeram as transições com canais *broadcast* pois já não faziam sentido neste caso de uso, e a restrição temporal relativa à permanência do sinal verde na rua principal por 30s. Fora isto, os processos dos semáforos estão iguais e apenas representam o comportamento de troca de luzes.

3.1.2.2 Sensor

No processo dos sensores, o fluxo de trânsito é calculado com base na entrada e saída de carros. Para este efeito, foi criada uma variável, *n_cars*, que conta os carros numa determinada rua. As funções *enter_car()* e *leave_car()* são responsáveis por incrementar e decrementar esta variável, respetivamente. A mudança de fluxo de uma determinada rua acontece então quando o número de carros sai de um determinado intervalo de valores. Os predicados *one_car()*, *low_condition()* e *high_condition()* definem o intervalo de valores entre o qual o fluxo de trânsito deve ou não variar, tendo como base o número de carros. Foi definido ainda um canal de sincronização, *flow_change*, que deve sincronizar com o processo *Controller* sempre que existe uma variação de fluxo de maneira a que o controlador consiga tratar a mudança de imediato. É de notar ainda que a função *leave_car()* só é executada quando o sinal da rua está a verde e a função *enter_car()* só acontece quando o sinal da rua está a vermelho e representa o momento em que o sensor volta a contar quantos carros estão à espera.

3.1.2.3 Controller

O processo Controlador encontra-se recorrentemente na localização *idle* e muda de localização de duas maneiras diferentes. Uma delas, como foi referido anteriormente, é quando existe uma mudança de fluxo de uma das ruas e tem que haver uma decisão imediata sobre se os semáforos mudam ou não. A outra maneira é ao fazer ciclos recorrentes de leitura do estado do sistema. Quando passa um ciclo de leitura, o controlador verifica o estado do sistema e, com base no fluxo de trânsito toma uma decisão. Quando o Controlador passa para a localização *decision_state*, pode voltar ao estado *idle* de duas maneiras, ou decide que os sinais têm que trocar e faz a sincronização da ação *activate*, ou decide que os sinais permanecem como estão e simplesmente volta a *idle*.

O critério de decisão sobre se os sinais mudam ou não é definido na função *decision()* e pode-se observar na Figura 3.3 esse mesmo critério.

Os casos com decisão direta são aqueles em que pelo menos uma das ruas não tem nenhum tráfego. Nesse caso, se numa rua está com o sinal verde e sem tráfego e a outra rua contém carros à espera, então os sinais vão mudar. Os outros casos requerem um critério mais forte, baseado no número de ciclos de leitura. No caso em que as duas ruas estão com o fluxo de tráfego *Low*, o *Controller* passa 2 ciclos de leitura sem mudar os sinais. Quando o valor máximo de ciclos de leitura são ultrapassados, o Controlador troca os sinais da rua. As outras abordagens são semelhantes variando apenas no número de ciclos que se deixa passar até trocar de sinais. O caso em que se deixa passar mais ciclos de leitura sem tomar alguma ação é quando a rua que tem o sinal a verde tem o fluxo *High* e a outra tem o fluxo *Low* pois, desta maneira, é possível fazer sair muitos mais carros da rua que está mais congestionada, sendo então a rua com maior prioridade.





Major Road	Minor Road	Major Road	Minor Road
			
None	False	None	None
None	True	None	Low
None	True	None	High
Low	False	Low	None
Low	2cycles	Low	Low
Low	True	Low	High
High	False	High	None
High	4cycles	High	Low
High	3cycles	High	High

Figura 3.3: Critério de troca de sinais

3.2 Verificação

Uma vez feito o modelo para o sistema com um sensor em cada semáforo e controlo de tráfego, voltou-se a verificar as propriedades definidas na primeira parte do trabalho. As propriedades tiveram que ser ligeiramente alteradas para serem adaptadas ao novo modelo.

```

E<= major_road.Red
E<= minor_road.Green
A[] not deadlock
A[] not ( major_road.Green && minor_road.Green )
major_sensor.n_cars > 0 && major_road.Red ==> major_road.Green
minor_sensor.n_cars > 0 && minor_road.Red ==> minor_road.Green
A[] !(major_road.Yellow && major_road.c > 5)
A[] !(minor_road.Yellow && minor_road.c > 5)
major_road.Yellow ==> major_road.Red
minor_road.Yellow ==> minor_road.Red

```

Figura 3.4: Verificação das propriedades anteriormente definidas

3.2.1 Eficiência do Sistema

Nesta secção vão ser apresentadas algumas propriedades em CTL capazes de verificar a eficiência do sistema desenvolvido.

Para todos os estados onde a rua que está a verde apresenta o fluxo *High* e a outra o fluxo *Low*, o número de ciclos de leitura não é ultrapassado, significando que eventualmente a rua com menos tráfego também fica com o sinal verde:

$$\begin{aligned} A\Box & (M_is_green \wedge major_sensor.High \wedge minor_sensor.Low \\ & \vee \neg M_is_green \wedge minor_sensor.High \wedge major_sensor.Low) \\ & \rightarrow Controller.low_high_cycles \leq Controller.low_high_cycle_bound \end{aligned}$$

Para todos os estados onde as duas ruas estão com alto tráfego, o número de ciclos de leitura nunca são ultrapassados significando que eventualmente a rua que tem o sinal vermelho fica com o sinal verde:

$$\begin{aligned} A\Box & major_sensor.High \wedge minor_sensor.High \\ & \rightarrow Controller.high_cycles \leq Controller.high_cycle_bound \end{aligned}$$

Para todos os estados onde as duas ruas têm tráfego *Low*, o número de ciclos de leitura nunca é ultrapassado, significando que a rua que está com o sinal vermelho eventualmente fica com o sinal verde:

$$\begin{aligned} A\Box & major_sensor.Low \wedge minor_sensor.Low \\ & \rightarrow Controller.low_cycles \leq Controller.low_cycle_bound \end{aligned}$$

O tráfego de uma determinada rua nunca pode baixar se o sinal estiver vermelho:

$$\begin{aligned} M_is_green \wedge minor_sensor.High & \rightsquigarrow \neg minor_sensor.Low \\ M_is_green \wedge minor_sensor.Low & \rightsquigarrow \neg minor_sensor.None \\ \neg M_is_green \wedge major_sensor.High & \rightsquigarrow \neg major_sensor.Low \\ \neg M_is_green \wedge major_sensor.Low & \rightsquigarrow \neg major_sensor.None \end{aligned}$$

Se uma rua não tem tráfego e a outro tem algum tráfego, então, eventualmente, o sinal da rua que tem tráfego fica a verde

$$\begin{aligned} &major_sensor.None \wedge \neg minor_sensor.None \rightsquigarrow \neg M_is_green \\ &minor_sensor.None \wedge \neg major_sensor.None \rightsquigarrow M_is_green \end{aligned}$$

Não existe um estado em que os semáforos de ambas as ruas estão simultaneamente a vermelho

$$A \Box \neg (major_road.Red \wedge minor_road.Red)$$

Se o tráfego da rua secundária continuamente *High* e o da principal *None*, então observa-se no máximo uma mudança dos sinais

$$\begin{aligned} &(minor_sensor.High \wedge major_sensor.None \wedge \\ &M_is_green \wedge Controller.switch_lights < 1) \\ &\rightsquigarrow (minor_sensor.High \wedge major_sensor.None \wedge \\ &\neg M_is_green \wedge Controller.switch_lights = 1) \end{aligned}$$

```
A[] Controller.low_cycles <= Controller.low_cycle_bound && Controller.high_cycles <= Controller.high_cycle_bound && Controller.low_high_cycles <= Co...
A[] (M_is_green && major_sensor.High && minor_sensor.Low || !M_is_green && minor_sensor.High && major_sensor.Low ) imply Controller.low_high_cycles <...
A[] major_sensor.High && minor_sensor.High imply Controller.high_cycles <= Controller.high_cycle_bound
A[] major_sensor.Low && minor_sensor.Low imply Controller.low_cycles <= Controller.low_cycle_bound
minor_sensor.High && major_sensor.None && M_is_green && Controller.switch_lights < 1 --> minor_sensor.High && major_sensor.None && !M_is_green && Con...
M_is_green && minor_sensor.High --> !minor_sensor.Low
M_is_green && minor_sensor.Low --> !minor_sensor.None
!M_is_green && major_sensor.High --> !major_sensor.Low
!M_is_green && major_sensor.Low --> !major_sensor.None
major_sensor.None && !minor_sensor.None --> !M_is_green
!major_sensor.None && minor_sensor.None --> M_is_green
A[] !(major_road.Red && minor_road.Red)
```

Figura 3.5: Verificação das propriedades de eficiência do sistema

3.3 Conclusão

A primeira parte do projecto apresentava um caso de uso relativamente simples de modelar pelo que o tempo despendido para a sua realização foi bastante mais curto em comparação com a segunda parte do trabalho. As propriedades de *reachability*, *safety* e *liveness* impostas no enunciado foram definidas para este modelo e todas conseguiram ser verificadas.

A segunda parte do trabalho prático apresentava mais complexidade ao sistema visto que cada semáforo tinha um sensor que media o fluxo de tráfego de cada rua. As maiores dificuldades sentidas nesta parte do trabalho foram como modelar o controlo de fluxo e a definição do critério de decisão sobre quando os semáforos mudavam de sinal. Em relação ao controlo de fluxo, decidiu-se que a primeira abordagem ao problema modelava de uma forma demasiado abstracta a mudança de fluxo de uma rua, o que tornava difícil definir o caso em que uma rua está a verde e, por isso, o número de carros nessa mesma rua diminui. Já a segunda abordagem permite ter controlo sobre o número de carros que existe numa rua, o que torna o sistema mais próximo da realidade. Em relação ao critério de decisão sobre se os sinais mudavam ou não, decidiu-se fazer recorrentemente leituras ao estado para que o momento de decisão da troca de sinais não dependesse só da variação de fluxo. Isto faz com que, por exemplo, se as duas ruas estiverem com o tráfego *Low*, uma das ruas fique infinitamente com o sinal verde. Os ciclos de leitura permitem então dar uma distribuição justa de sinais verdes às ruas.

Em relação à ferramenta UPPAAL, usada para fazer a modelação do sistema de tráfego, o grupo achou que é uma ferramenta bastante intuitiva para modelação. É de notar que o facto de se ter usado o *mCRL2* em trabalhos anteriores tornou o processo de aprendizagem do UPPAAL bastante mais fácil pois ambas ferramentas são baseadas em autómatos. As ferramentas dispostas pelo UPPAAL foram bastante úteis no processo de modelação e verificação do sistema. A parte de definição do modelo é bastante acessível e um ponto bastante positivo é o facto de se poder modelar o sistema de uma forma visual, com uma *interface drag and drop*. Isto permite à pessoa que está a desenvolver o modelo concentrar-se na sua concepção e não perder o foco com pormenores não relevantes, algo que poderia acontecer se a maneira de definir o modelo fosse algo mais programático. Outro facto interessante na ferramenta de edição do modelo é o facto de se poderem definir funções e variáveis. As funções criadas fizeram com que o modelo ficasse muito mais legível e menos denso ao encapsular os predicados lógicos. As ferramentas de simulação do UPPAAL permitem visualizar o comportamento dos autómatos, algo bastante útil para detectar o porquê de algumas situações anómalas e, por ventura, corrigir as mesmas. A ferramenta de verificação permitiu testar as propriedades desejáveis do sistema e, com base nisso, fazer alguns refinamentos no sistema. Para além da verificação, através da funcionalidade *Get Trace*, foi possível arranjar caminhos específicos do sistema. Isto foi bastante útil no sentido que permitiu agilizar o processo de atingir um determinado estado, por exemplo, havia casos em que se precisava de atingir o estado (*Low*, *High*)

para, a partir desse estado, verificar que o comportamento no simulador era o correto. Para isso, bastou definir-se uma propriedade $E \Diamond major_sensor.Low \wedge minor_sensor.High$ e fazer *Get Trace*. Isto faz com que o simulador seja carregado com um conjunto de traços em que o último estado é $(Low, High)$ e, a partir deste momento, é possível navegar nos autómatos a partir deste estado.

Por fim, o grupo achou que o trabalho foi bastante interessante e desafiante e permitiu aprender uma maneira de modelar bastante diferente do que a que estava habituado. Modelar sistemas com autómatos, e especialmente com o UPPAAL, faz com que haja uma ideia mais clara do sistema em questão pelo facto de haver uma representação gráfica bastante poderosa.