Universidade do Minho

Departamento de Informática

Programação Ciber-Física

Modelação e análise de um sistema ciber-físico

Trabalho Prático 1

Carlos Ferreira PG47087
Henrique Neto PG47238

16 de maio de 2022

1 Introdução

No âmbito da disciplina de Programação Ciber-Física, foi proposto um projeto para modelar e analisar um sistema que simule o funcionamento de um par de semáforos de trânsito. Nesse sistema, exemplificado pela figura 1.1, existem duas estradas, uma que representa uma estrada principal (ou coletor) que devido as suas características apresenta um ritmo constante de trânsito, enquanto que a outra representa uma estrada secundária (ou local) que apresenta um fluxo baixo e intermitente de trânsito. O objetivo é usar a ferramenta *UPPAAL* para modelar um sistema ciber-físico que permita gerir os semáforos na intercessão das estradas com recurso a um sensor de trânsito presente na estrada secundária.

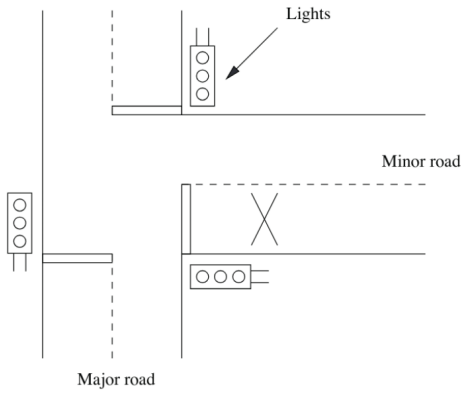


Figura 1.1: Conjunção das Vias

O projeto em si é divido em duas partes sendo que a primeira foca-se na construção de um sistema rudimentar que consiga garantir o funcionamento correto do entrocamento. Para tal, os semáforos devem garantir várias regras de segurança salientado o facto de que os semáforos permitirão apenas tráfego alternado entre as estradas, em conjunto com regras de *liveness*, ou seja o sistema permitirá que qualquer carro consiga eventualmente atravessar o cruzamento. Para comprovar estas propriedades serão formuladas *queries* usando uma linguagem da ferramenta *UPPAAL* que se baseia em *CTL* (*Computation Tree Logic*).

A segunda parte consiste em modelar um cruzamento que envolva antes dois coletores. Nesta fase ambas as estradas vão conter sensores que apresentam um comportamento alternativo ao sensor origem. Este sensor é definido de forma livre e fornece uma componente mais exploratória ao problema em questão. Neste caso, consideramos que os sensores são consideravelmente avançados e desta forma são capazes de quantizar número de automóveis que se encontram a tentar cruzar o cruzamento. Tal como a fase anterior, esta implementação terá de garantir vários princípios de segurança e de *liveness* para garantir o seu funcionamento correto do sistema.

2 Primeira Parte

2.1 Escolhas de Design

Como foi referido na introdução, o objetivo do primeiro modelo é garantir o funcionamento correto da via, ou seja, um sistema que cumpra regras de segurança e vivacidade de forma a que não haja nem acidentes nem embaraço do trânsito. Desta forma, o modelo foi especificado tendo em conta os seguintes requisitos.

- O semáforo da estrada principal deve permanecer verde, por pelo menos 30 segundos,

após os quais se fechara só se for detetado trânsito na via secundária.

- Os semáforos de ambas as vias irão então alterar-se de forma a permitir que o trânsito saia da estrada secundária, sendo que após um intervalo razoável, as luzes serão revertidas ao estado inicial onde a via principal permanece outra vez indefinitivamente aberta.
- O sensor da estrada secundária permanece desativado desde o momento em que deteta um carro até as luzes da sua via ficarem novamente vermelhas, sendo assim mais econômico.
- As luzes amarelas devem permanecer ligadas por um tempo definido, existindo também um atraso de 1 segundo entre transições de luzes em cada semáforo.

A forma como os requisitos foram cumpridos será explicada detalhadamente no próximo capitulo, no entanto, a arquitetura definida tem algumas especificações importantes:

- Os dois semáforos utilizam canais de informação dedicados à gestão do seu funcionamento, para que seja assim possível garantir que um semáforo só inicia luz verde após o oposto ligar a luz vermelha e vice-versa.
- O sensor comunica com o semáforo principal para este iniciar o processo de transição de luz verde para vermelha.
- A via secundária comunica com o sensor para este se ligar e iniciar o processo de detecção de trânsito.
- Utiliza-se relógios internos em ambos os semáforos para garantir os tempos que cada luz deve satisfazer.
- Os tempos que restringem o comportamento de cada luz, foram definidos como variáveis globais para que possam ser facilmente ajustadas.

Como já foi mencionado, o modelo desta fase é bastante rudimentar. Esta simplicidade deve-se ao problema em questão, visto que pelo facto que o trânsito na via secundária é bastante inferior ao da via principal, as condições anteriores já são suficientes para garantir um funcionamento eficiente do entrocamento.

2.2 Modelos

2.2.1 Declarações globais

Para representar os componentes do sistema foram modulados três *templates* diferentes nos quais dois deles representam os semáforos e que desta forma são extremamente semelhantes. Estes componentes fazem recurso a duas constantes globais, onde a constante *sw_dur* (*switch duration*) representa o tempo que o semáforo demora a trocar de luz e a constante *yel_dur* (*yellow duration*) representa o tempo da luz amarela. Adicionalmente, o sistema apresenta quatro canais de comunicação cujas funcionalidades são descritas na tabela 2.1.

Canal	Funcionalidade
sensor	Permite uma transmissão entre o sensor e o semáforo principal que desencadeia o processo de rotação dos semáforos
on	Permite ligar o sensor se este estiver desligado
red	Permite ao semáforo principal indicar que ficou vermelho
green	Permite indicar ao semáforo principal para ficar verde

Tabela 2.1: Descrição dos Canais

2.2.2 Template *Sensor*

O sensor é o modelo mais simples do sistema. Contem três estados ligados em anel como é possível ver na figura 2.1

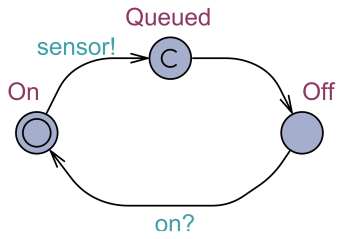


Figura 2.1: Modelo Sensor

O sensor inicializa-se no estado *On* no qual após reconhecer um carro fará uma transmissão pelo canal sensor que será recebida pelo semáforo *Main*. Logo a seguir à transmissão este atravessa um estado *commited* denominado *Queued* e desliga-se no estado *Off*. O estado *Queue* está presente neste modelo para ser possível reconhecer o momento em que ocorreu a transmissão na especificação das *queries*. Por fim, assim que componente receber uma mensagem do canal *on* por parte do semáforo *Side* irá ligar-se e voltará ao estado inicial *On*.

2.2.3 Template *Main* e *Side*

Ambos estes componentes possuem três estados que representam a cor da luz atual e pelo menos três estados que representam as transações entre as diferentes luzes. Adicionalmente as operações e os argumentos de ambos os *templates* apresentam um correlação evidente. Cada uma das *templates* possui um relógio privado designado de *z*.

Main

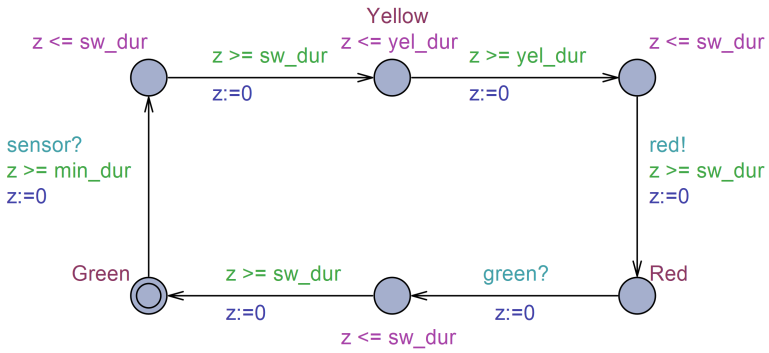


Figura 2.2: Modelo *Main*

Este *template* recebe como argumento o inteiro constante *min_dur* (*minimum duration*) que representa o tempo mínimo que este tem que ficar no estado verde.

Como podemos observar a *template* começa inicialmente com o semáforo na cor verde (*Green*). A partir deste ponto o sistema tem a possibilidade de transitar para o estado de troca de luzes entre a cor verde e amarela assim que tenha passado o tempo mínimo definido para estado verde (*min_dur*). Depois de transitar para este estado intermédio este é obrigado a transitar exatamente quando o relógio atingir o valor do tempo de troca de luzes (*sw_dur*) graças ao invariante presente no estado e à condição na aresta. Seguidamente o semáforo permanece no estado amarelo durante o tempo definido globalmente por *yel_dur* acabando por avançar para um novo estado de transação, desta vez entre as luzes amarela e vermelha. Por fim ao chegar ao estado vermelho o modelo transmite uma mensagem pelo canal *red* indicando que acabou de ficar vermelho e fica a aguardar uma mensagem no canal *green*. Ao receber a mensagem, o sistema transita para o estado entre a cor vermelha e a cor verde, voltando eventualmente ao estado inicial *Green*.

Side

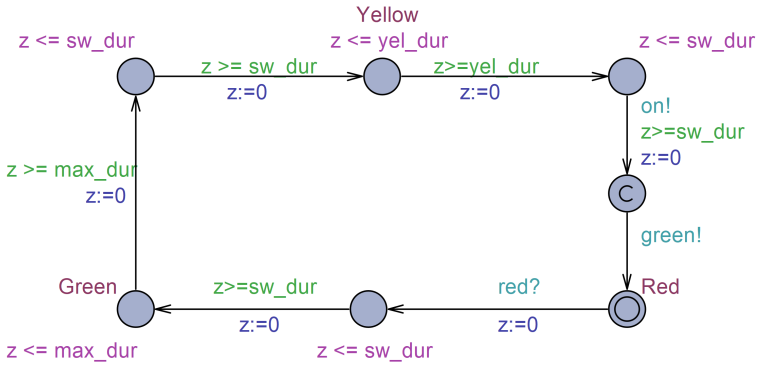


Figura 2.3: Modelo *Side*

Quando comparado com o template *Main* o modelo do *Side* altera-se apenas três aspetos. Estas diferenças estão presentes no estado inicial, nas ações antes e após ter ficado verde e vermelho, e no facto do seu argumento ser antes o tempo máximo que pode ficar na cor verde. Adicionalmente o modelo passa a executar duas operações ao ficar vermelho, sendo a operação adicional correspondente voltar a ligar o sensor da estrada (*on!*).

Desta forma, o estado inicial do sistem é *Red* na qual este fica a aguardar uma mensagem no canal *red* que significa que o semáforo da rua principal ficou vermelho. Após ter recebido a mensagem este eventualmente transitará para o estado *Green* onde ficara no máximo o tempo definido pelo argumento fornecido. Após esse período, este irá transitar pelos estados intermédios associados à cor amarela da mesma forma como a *template* anterior. Por fim este termina com duas operações antes de voltar ao seu estado inicial *Red*. A primeira operação consiste em ligar o sensor através de uma mensagem pelo canal *on* enquanto que a segunda consiste por sinalizar o semáforo da estrada principal para ficar verde através de uma mensagem no canal *green*.

2.3 Expressões *CTL*

Foram estabelecidas sete *queries* com linguagem *CTL* que o *UPPAL* disponibiliza, nas quais a ferramenta validou e comprovou no sistema implementado. Três das *queries* provam propriedades de segurança, duas delas provam a acessibilidade do sistema a certos estados e as restantes duas provam propriedades de *liveness* do sistema.

2.3.1 Acessibilidade

Estes *queries* servem simplesmente para provar que um dado estado ou acontecimento é possível.

A esta principal pode ficar Vermelha Para especificar isto temos apenas que dizer que existe um caminho em que com pelo menos um estado em que a estada principal está vermelha, ou seja E<> mainStreet.Red.

A esta secundária pode ficar Verde Semelhante á *querie* anterior queremos provar que existe um caminho em que existe pelo menos um estado no qual a estrada secundária está verde, ou seja E<> sideStreet.Green.

2.3.2 Segurança

O propósito destas regras é certificar a segurança do sistema de forma a que nunca ocorram estados errados que levem a consequências más, como por exemplo acidentes.

Sem Deadlocks O propósito desta *querie* é certificar que o sistema nunca pára e que desta forma, o sistema consegue evoluir sempre independentemente do seu estado atual. A formula provada pela ferramente foi A[] **not** deadlock, ou seja para todos os estados de todos os caminhos não existem impasses (*deadlocks*).

Ambos semáforos não podem estar Verdes simultaneamente Esta propriedade serve para provar que não pode ocorrer acidentes por causa de ambas as estradas estarem verdes. Para provarmos isto, especificaremos uma *querie* que indica que para todos os estados de todos os caminhos, se uma estrada estiver verde então a outra não pode estar verde e vice-versa. Desta forma, resulta a expressão:

```
A[] (sideStreet.Green imply not mainStreet.Green)
    and (mainStreet.Green imply not sideStreet.Green)
```

Pelo menos um dos semáforos tem de estar vermelho em qualquer momento Embora a propriedade anterior permita impedir uma das situações de acidente, não permite certificar que o sistema impede todas as situações de acidente visto que segundo o código da estrada, um condutor pode atravessar uma luz amarela se a sua travagem puser em risco outros utentes da via pública. Tendo em conta que cada um dos templates dos nossos semáforos apresentam pelo menos três estados de transação situados entre as luzes que também tem de ser contabilizados nestas situações, para provarmos deliberadamente que nenhuma situação de acidente ocorre teríamos de especificar exaustivamente as várias combinações de estados entre a estrada principal e secundária. Invés disto, vamos antes especificar uma propriedade que implica estas, ou seja a qualquer momento do sistema pelo menos um dos semáforo tem que estar vermelho. Esta propriedade descreve assim que em qualquer estado de qualquer caminho, se o semáforo *Main* não estiver vermelho então o semáforo *Side* estará e se o semáforo *Side* não estiver vermelho então o semáforo *Main* estará. A expressão resultante é:

```
A[] (not sideStreet.Red imply mainStreet.Red)
    and (not mainStreet.Red imply sideStreet.Red)
```

2.3.3 *Liveness*

Estas propriedades certificam que o sistema de facto cumpre o problema que está a ser efetuado, e que desta forma este evoluirá de forma útil.

Se há carros à espera então eventualmente o semáforo ficará verde A formula para especificar esta propriedade é sen.Queued −−> sideStreet.Green, que indica que se o sensor ativar (ou seja se atravessar o estado *Queued*) então o semáforo da estrada secundária eventualmente ficará verde.

A estrada principal nunca fica vermelha permanentemente A formula para especificar esta propriedade é mainStreet.Red −−> mainStreet.Green, que indica que se o semáforo da estrada principal estiver vermelho, então este eventualmente ficará verde.

3 Segunda Parte

3.1 Escolhas de Design

O segundo sistema consiste na modelação de um cruzamento entre duas estradas principais onde cada uma possui um sensor. Adicionalmente esta fase é de desenvolvimento livre, sendo possível assumir vários contextos visíveis na vida real. O objetivo do modelo criado é modelar um sistema destinado a estradas com trânsito constante sujeitas a fluxos de trânsito variáveis, resultantes de por exemplo horas de ponta. Para tal, foram considerados que os sensores nas estradas são avançados o suficiente para quantizar o número de carros que estão a tentar atravessar o cruzamento em cada uma das estradas.

De forma a simplificar o modelo, o processamento das transações foram feitos por um gestor centralizado. Adicionalmente foi necessário especificar um modelo que simulasse concretamente a chegada dos automóveis. Desta forma foi considerado que os automóveis podem chegar espontaneamente ao sistema a um ritmo máximo que é estabelecido com um tempo mínimo de espera entre as chegada. Adicionalmente foi considerado que quando os semáforo estiverem ligados, os automóveis saíram a um ritmo constante que é considerado ser média do ritmo real da estrada ao longo de um perídio de tempo. Esta assunção introduz algum erro no sistema visto na realidade este valor varia bastante conforme as circunstancias do sistema e do veiculo. Exemplificando, um veículo longo em principio demora mais tempo a sair do sistema que um veiculo ligeiro, e um veículo que esteja inicialmente parado demora mais tempo que um que já se encontra em movimento. Porém devido à escala a que o modelo se destina, consideramos este erro possa vir a ser desprezável e desta forma não o exploramos.

O sistema implementado pretende permitir atribuir de forma dinâmica um intervalo de tempo a cada semáforo com base na sua ocupação em relação ao sistema como um todo. Para tal, é estabelecido previamente um período de tempo corresponde a um ciclo total nos semáforos. Desta forma se o sistema detetar trânsito na via oposta, após ter acabado o intervalo do semáforo atual irá trocar de semáforo e atribuir-lhe um tempo de atividade com base na distribuição de carros no sistema no instante de troca.

Esta implementação é bastante eficiente no caso em questão visto que, em princípio, ambas as estradas estão constantemente sujeita a trânsito. Se implementássemos este sistema para uma estrada com trânsito intermitente sem as devidas configurações sobre o período máximo dos semáforos, então poderia levar a situações onde o alocamento dinâmico implementado poderia agravar os tempos de espera, pois uma estrada poderia ser submetida a um tempo desnecessariamente longo, enquanto que a outra estrada amontoava uma fila dentro desse período.

3.2 Modelos

O modelo envolveu naturalmente a criação mais *templates* sendo estes a especificação do sensor da via principal, a simulação dos automóveis, e finalmente, o gestor (*manager*) que apresenta a maior complexidade devido á sua funcionalidade.

3.2.1 Declarações globais

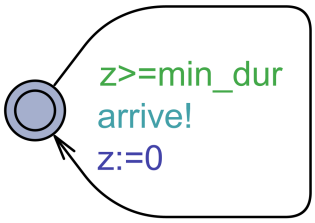
Graças ao aumento da escala do sistema surgiu também a necessidade de introduzir mais variáveis globais e canais que são visíveis na tabela 3.1.

Nome	Tipo	Descrição	Nome	Tipo	Descrição
openNormal	chan	Ordena abertura da via normal.	openIverted	chan	Ordena abertura de via secundária.
closeNormal	chan	Ordena desativação da via normal.	closeInverted	chan	Ordena desativação da via secundária.
normalClosed	chan	Indica estado da via normal.	invertedClosed	chan	Indica estado da via secundária.
trafficNormal	int	Mede trânsito da via normal.	trafficInverted	int	Mede trânsito da via secundária
normalOpen	bool	Confirma abertura da via normal.	InvertedOpen	bool	Confirma abertura da via secundária.
total_period	int	Tempo distribuído entre semaféros.	min_dur	int	Tempo de duração mínimo de um ciclo
cycle_dur	int	Tempo de duração dinâmico do ciclo atual.			

Tabela 3.1: Declarações globais

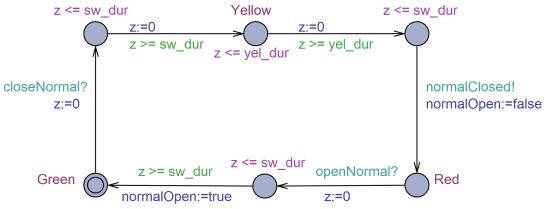
3.2.2 *Automobile*

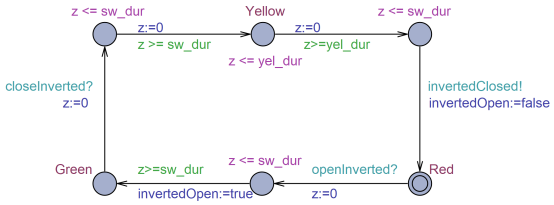
O automóvel corresponde a um simples sistema que pode sincronizar com os sensores pelo canal *arrive* de forma a comunicar que chegou um carro ao sistema. Como é quase impossível termos múltiplos carros a chegar no mesmo instante, na mesma via, consideramos que existe um tempo mínimo entre cada uma das chegadas definido por *min_dur* que é cumprido graças à condição associada ao relógio existente.



3.2.3 *Normal* e *Inverted*

Estes *templates* representam os semáforos das ruas principais, sendo que temos um modelo para a primeira uma rua principal e outro modelo invertido para a segunda rua principal.

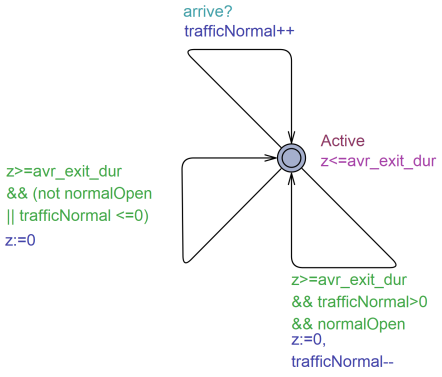


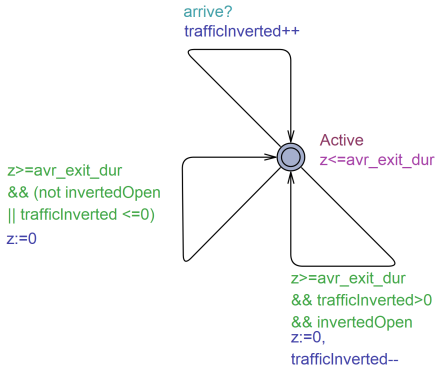


Como podemos observar ambos os modelos são iguais de um ponto de vista lógico, mudando apenas o estado inicial, sendo que um começa inicialmente verde e o outro a vermelho, e o nome dos canais envolvidos. Quando um semáforo se encontra no estado verde aguardará por uma mensagem no seu canal *close* para desta forma fechar a estrada. Quando recebe esta mensagem irá progredir pelos estados de transação e pelo estado amarelo até chegar ao estado vermelho. Ao chegar a este estado, o semáforo irá transmitir uma mensagem pelo seu canal *closed* e atualizará o estado da sua variável de confirmação. Por fim, quando o semáforo está no estado vermelho aguardará por uma mensagem no seu canda *open* para voltar a abrir e consequentemente mudar a sua variável de estado.

3.2.4 *SensorNormal* e *SensorInverted*

Semelhante aos semáforos, os dois modelos dos sensores correspondem ao mesmo modelo logico aplicado a variáveis diferentes.

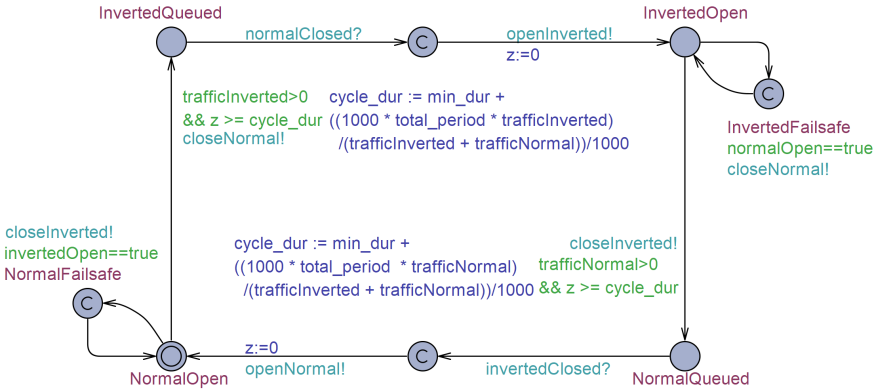




Ambos os sensores registam a entrada de carros através da arestas com os canais *arrive* e simulam a saída de automóveis do sistema se o semáforo estivesse aberto. O plano inicial

para este automota era ter o estado inicial como urgente e desta forma em conjunto com uma condição envolvendo o estado do semáforo (aberto ou fechado) e o tempo médio que um carro demora a sair certificaríamos que o carros poderiam sair com um fluxo máximo de saída. O problema é que na ferramenta *UPPALL*, como é possível ver na documentação, nem o estado urgente nem o estado *commited* obrigam a que ocorra uma transação assim que seja possível. Adicionalmente esta implementação do estado urgente obriga a que o relógio não avance o que impediria que o sistema transita-se de estado e que entra-se em *deadlock*. Desta forma, a solução para implementar esta funcionalidade consistiu em modelar um temporizador repetitivo no estado do sensor que a cada vez que este disparava, caso a estrada estivesse fechada este transitava por um estado que sem atualizações senão estando a estrada aberta este removeria um automóvel do registo do sistema.

3.2.5 *Manager*



O *Manager* é responsável por gerir o estado dos semáforos. O funcionamento base do modelo é, sempre que o sistema for alternar os semáforos e desta forma entrar em um dos estados *Queued*, o sistema irá calcular o templo de duração mínimo que o semáforo terá de estar ligado neste ciclo. Para tal, o sistema considera que o semáforo tem de estar ligado durante um período mínimo de tempo (min_dur) em conjunto com um período dinâmico que é calculado a partir da taxa de ocupação ($\frac{\#Carros\ em\ fila}{\#Carros\ no\ sistema} * periodo\ dos\ semaforos$). Ora devido ás limitações da ferramenta em questão, apenas podemos fazer divisões inteiras, e desta forma para conseguirmos calcular a taxa necessária, o parâmetros são escalados com o valor 1000 antes da divisão no qual se prossegue assim para o calculo da taxa. Por fim a escala é revertida com uma divisão 1000, permitindo assim ter precisão suficiente na componente dinâmica do ciclo. Depois do cálculo do intervalo, o procedimento é relativamente simples, o gestor transmite uma mensagem ao semáforo oposto e após ter recebido uma confirmação de resposta envia uma mensagem ao semáforo em questão para abrir. Quando o tempo alocado para o semáforo acabar, se existirem carros à espera na outra estrada este irá recomeçar este processo para o outro semáforo, senão permanecerá no estado atual até que este fator mude.

Adicionalmente estão modelados dois estados chamados *Failsafe*. Estes estados são inacessíveis ao sistema num ambiente normal e servem para o sistema conseguir ter alguma resposta caso algum meio terceiro ao sistema interfira com o ambiente e abra o semáforo oposto ao atual.

3.3 Expressões CTL

Com exceção das expressões de *liveness*, todas as *queries* formuladas anteriormente ainda são aplicáveis neste modelo. Desta forma, a exploração do funcionamento do sistema envolveu apenas modificar as queires invalidas e estipular uma nova *querie* para os novos estados envolvente.

3.3.1 Mudanças

Visto a gestão do sistema encontra-se definida no *manager*, a *querie* agora reflete antes a propriedades nesse *template*, sendo possível reduzir as expressões nas fórmulas:

```
gestor.InvertedQueued —> sideStreet.Green
gestor.NormalQueued —> mainStreet.Green
```

Que indicam que se para cada uma das estradas, se o semáforo dessa estrada tiver fila tiver fila então eventualmente a estrada correspondente irá ficar verde.

3.3.2 *Querie* Nova

A formula nova serve para provar que nenhum dos estados *Failsafe* são alcançáveis, resultando assim na *querie* A[] (**not** gestor.NormalFailsafe) **and** (**not** gestor.InvertedFailsafe).

4 Conclusões

A realização deste projeto deu a entender as diversas complexidades e limitações envolventes na criação e verificação de modelos eficientes, seguros e versáteis, quando estes estão comprometidos por fatores físicos como o tempo, mesmo quando inicialmente não aparentam ter grande dificuldade.

Um sistema que seja capaz de simular o funcionamento de um único par de semáforos já se demonstrou bastante trabalhoso com as inúmeras nuances presentes. Além disso, o sistema desenvolvido na componente exploratória garante também novas características vantajosas, como assegurar tempos de espera máximos, recuperar falhas físicas e simular trânsito realista, no entanto, esta implementação ainda contêm inconveniências que se revelaram na dificuldade em encontrar um equilíbrio entre todas as vantagens desejadas e certificar adaptabilidade a situações de trânsito particulares.

Quanto a ferramenta utilizada *UPPAAL*, esta afirmou-se ser uma ferramenta muito útil, mas de muitas formas limitada pelo seu poder expressivo e também pela sua *performance*, sendo só resolvidas se utilizadas outras ferramentas, realçando assim o potencial que o campo de ciência pode atingir no futuro.