

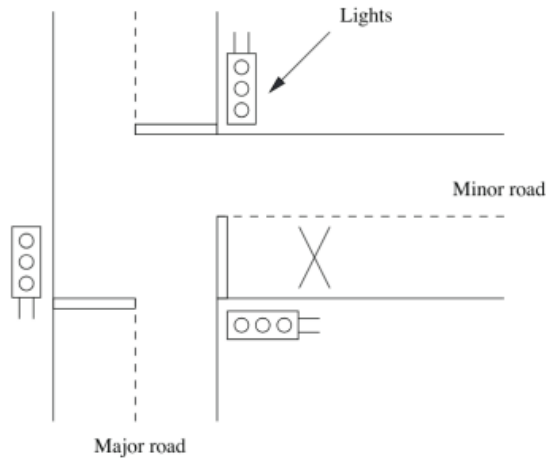
# Computação Ciberfísica - TP1 - Modelling and analysis of a cyber-physical system

Catarina Madaleno - pg45927 e Ricardo Correia - pg47607

16 de maio de 2022

# 1 First part

The first goal of the assignment is to model 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 and is depicted below (together with the respective traffic lights):



In this scenario vehicles drive on the left side of the road and the cross in the picture represents a sensor that tells whether a car is waiting in the minor road or not.

In order to guarantee a reasonable traffic flow, the system has the following constraints:

1. 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.
2. In the latter case, the lights will switch in the standard manner and allow traffic to leave the minor road. After a suitable time interval (30s), the lights will revert to their default position so that traffic can flow on the major road again.
3. Finally, as soon as a vehicle is detected by the sensor the latter is disabled until the minor-road lights are on red again.

The system also respects the following temporal constraints:

1. Interim lights stay on for 5s.
2. There exists 1s delay between switching one light off and the other on.
3. The major-road light must stay on green for at least 30s in each polling cycle, but must respond to the sensor immediately after that.

**The first part of the students' assignment:**

1. Model in UPPAAL the system of traffic lights described previously;
2. Express in CTL the following reachability properties and test them in UPPAAL: (1) the minor-road light can go green; (2) the major-road light can go red.
3. Express in CTL the following safety properties and test them in UPPAAL: (3) the system never enters in a deadlock state; (4) the minor-road and major-road lights cannot be green at the same time.
4. Express in CTL the following liveness property and test it in UPPAAL: (5) if there are cars waiting they will eventually have green light.
5. Can you think of other desirable properties? If so please register them and check whether they hold or not.

## 1.1 Model in UPPAAL the system of traffic lights described previously

Para modelarmos o sistema descrito, utilizámos 7 autómatos: semaforo\_P, semaforo\_S, road\_P, road\_S, Sensor\_S, generate\_car\_P e generate\_car\_S.

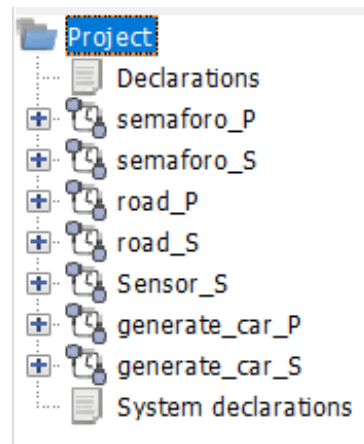


Figura 1: Autómatos elaborados

Na qual as declarações que utilizamos foram as seguintes:

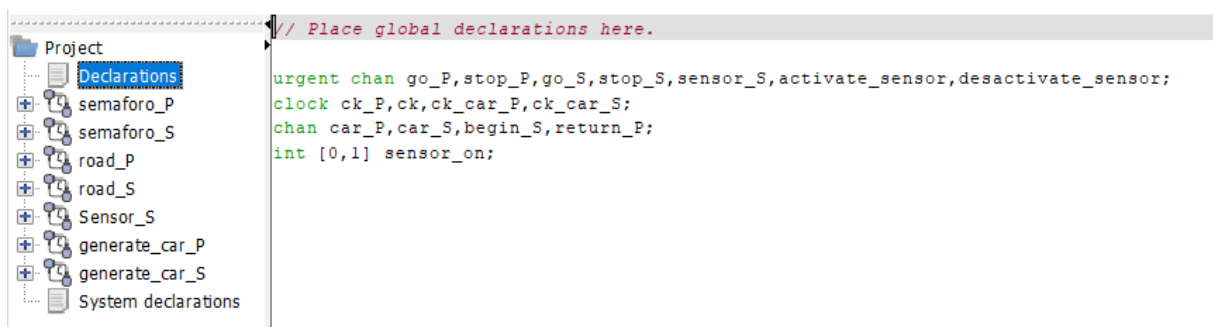


Figura 2: Declarações

E criou-se as seguintes declarações do sistema:

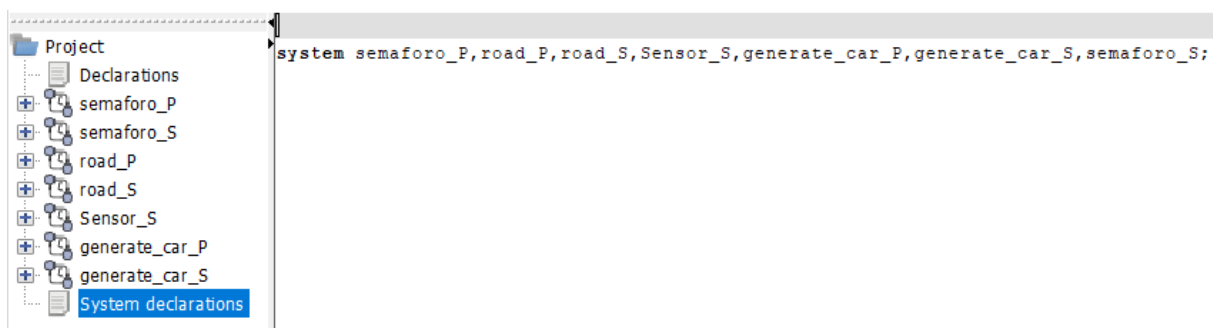


Figura 3: Declarações do sistema

### 1.1.1 generate\_car\_P e generate\_car\_S :

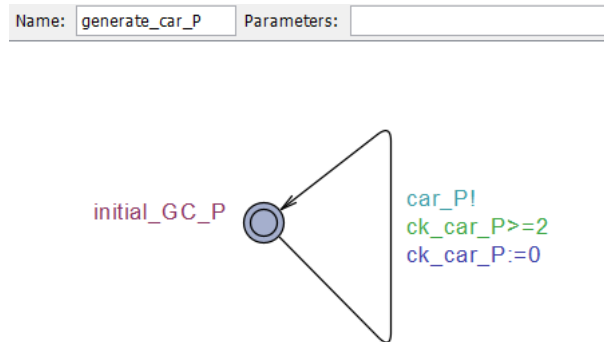


Figura 4: *generate\_car\_P*

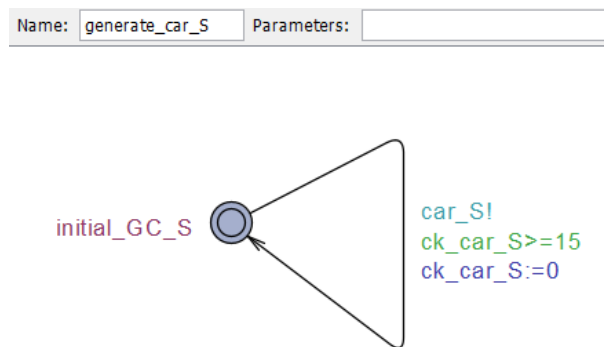
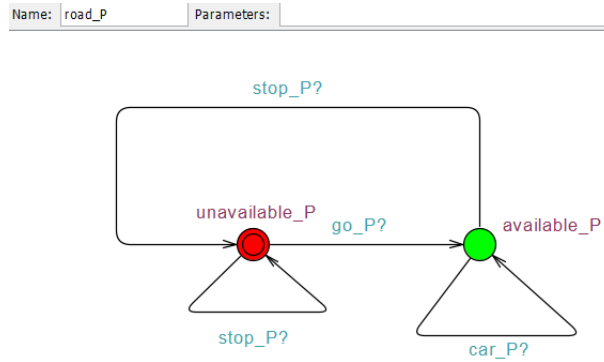


Figura 5: *generate\_car\_S*

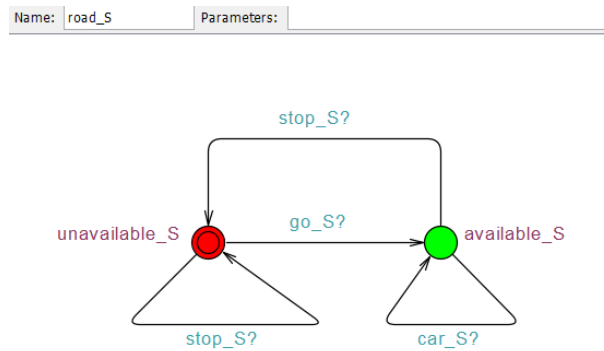
Estes dois autómatos têm a função de gerar um carro, seja na estrada principal (P) ou na estrada secundária (S). O seu aspeto e funcionamento é bastante semelhante: simplesmente geram um carro, enviando uma sincronização ( $car\_P!$  e  $car\_S!$ ). A diferença entre eles é que a estrada principal gera um carro no mínimo de 2 em 2 segundos ( $ck\_car\_P \geq 2$ ) e a estrada secundária gera um carro no mínimo de 15 em 15 segundos ( $ck\_car\_S \geq 15$ ). Isto pretende simbolizar a diferença de afluência de carros em cada estrada apresentada no enunciado: passam carros mais frequentemente na estrada principal que na estrada secundária.

### 1.1.2 road\_P e road\_S:

Estes dois autómatos são idênticos, sendo que cada um representa o estado de uma estrada. Road\_P simboliza o estado da estrada principal, road\_S simboliza o estado da estrada secundária. Estas estradas podem estar no estado "available" ou "unavailable", sendo o "unavailable" o estado inicial. O estado "available" significa que podem circular carros, ou seja o semáforo está verde, e o estado unavailable significa que não podem circular carros, estando o semáforo dessa estrada vermelho. Partindo do estado "unavailable", se é recebida uma sincronização "stop" (que significa esse semáforo passar a estar vermelho), esse estado mantém-se. Por outro lado, se recebe uma sincronização



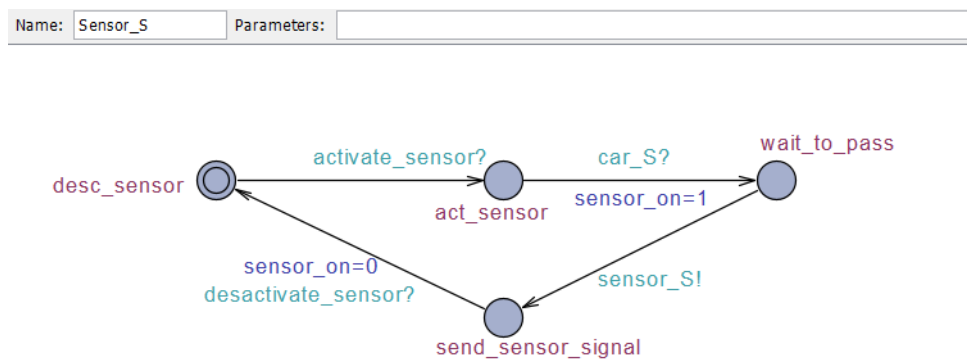
**Figura 6:** *road\_P*



**Figura 7:** *road\_S*

"go"(que significa que esse semáforo passa a estar verde), a estrada passa a estar no estado "available". No estado available esta pode receber a sincronização "car"as vezes que quiser, que significa que estão carros a passar na estrada. Ficarão carros a passar na estrada até ser recebida uma sincronização "stop", que como já foi referido significa que o semáforo passa a ficar vermelho, e assim se passa de novo ao estado "unavailable".

### 1.1.3 Sensor\_S



**Figura 8:** *Sensor\_S*

Este autômato simboliza o sensor presente na estrada secundária, que informa se há carros à espera nessa estrada. Este sensor tem como estado inicial "desc\_sensor", que significa que o sensor está em descanso (desativo), à espera de receber a sincronização "activate\_sensor", vinda do semaforo\_S. Uma vez recebida essa sincronização, o sensor passa para o estado "act\_sensor", o que significa que fica ligado, à espera de receber a sincronização "car\_S", ou seja, à espera que um carro acione o sensor da estrada S. Quando um carro passar na estrada, a constante "sensor\_on", que simboliza se existem carros à espera (ou seja, se o sensor detetou um carro), passa a 1, e o estado do sensor passa para "wait\_to\_pass" ou seja os carros ficam à espera para passar. De seguida, é enviada a sincronização "sensor\_S", que permite dizer ao semáforo P que há carros à espera para passar na estrada secundária (veremos isto mais à frente). Passa-se assim para o estado "send\_sensor\_signal", e ficará neste estado até receber a sincronização "deactivate\_sensor", vinda do semaforo\_P, que vai dizer ao sensor para desligar, vai fazer ao mesmo tempo a constante "sensor\_on" voltar a 0 e volta assim ao estado inicial, "desc\_sensor", até ser ativado novamente.

#### 1.1.4 Semaforo\_P

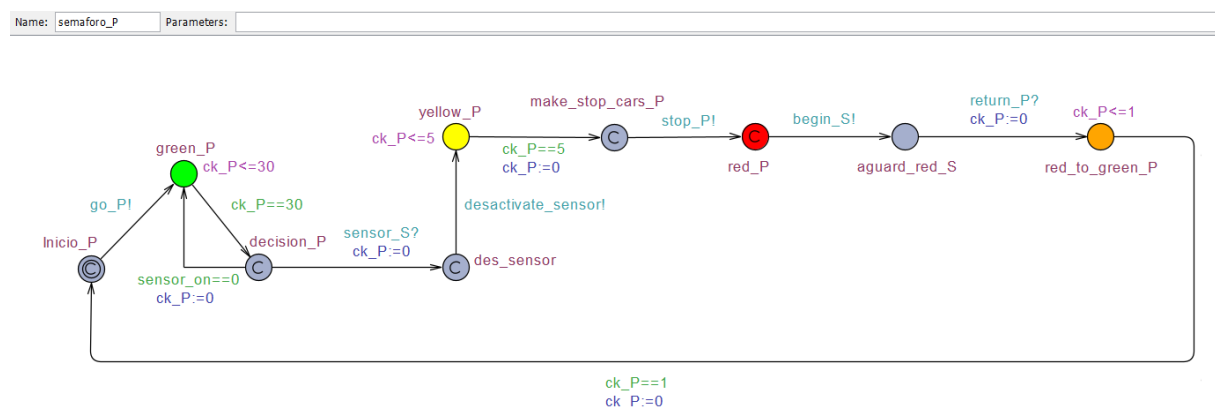


Figura 9: Semaforo\_P

Pode dizer-se que este autômato, que representa o semáforo da estrada principal, é o autômato central do nosso modelo. Tem como estado inicial "Inicio\_P", que é um estado committed ou seja não passa lá tempo nenhum, enviando logo a sincronização "go\_P!" (recebida por road\_P) e segue para o estado green\_P, que simboliza que o semáforo\_P está verde, como é ilustrado pela cor do estado no autômato. O semáforo entra agora num polling cycle. Fica neste estado durante 30 segundos, e passado 30 segundos vai para o estado committed "decision\_P", onde vai verificar se o sensor\_S foi ativado ou não. Ou seja, se o sensor não tiver sido ativado nestes 30 segundos (se não estiverem carros à espera na road\_S) ele segue o caminho "sensor\_on==0", o clock P é zerado, e o semáforo\_P volta a ficar 30 segundos verde até ser tomada nova decisão. Caso o sensor\_S tenha sido ativado, ou seja, se houverem carros à espera na road\_S durante os 30 segundos em que o semaforo\_P esteve verde, é recebida a sincronização "sensor\_S" (vinda do Sensor\_S), passa para o estado committed "des\_sensor", o sensor é de seguida desativado ao ser enviada a sincronização "desactivate\_sensor" (recebida pelo

Sensor\_S), e o semáforo P passa a amarelo, no estado "yellow\_P". Fica neste estado durante 5 segundos, após esses 5 segundos vai para o estado committed "make\_stop\_cars", a partir do qual envia a sincronização "stop\_P", que é recebida pela road\_P, e chega finalmente ao estado red\_P, no qual o semáforo P passa a vermelho. É enviada de seguida a sincronização "begin\_S"(recebida, como iremos ver, pelo semáforo\_S, que serve para inicializar a sua troca de cores), e passa-se para o estado aguard\_red\_S. O semáforo P fica neste estado, ou seja, a vermelho e à espera da sincronização "return\_P"vinda do Semáforo S - que significará que o semáforo S terminou a sua execução e o semáforo P poderá continuar a sua. Recebida esta sincronização e zerado o clock P, passa-se para o estado "red\_to\_green\_P", que tal como requerido no enunciado, obriga a que haja um delay de 1 segundo entre desligar uma luz e ligar a outra. Quando passa 1 segundo, volta-se ao estado inicial.

### 1.1.5 Semaforo\_S

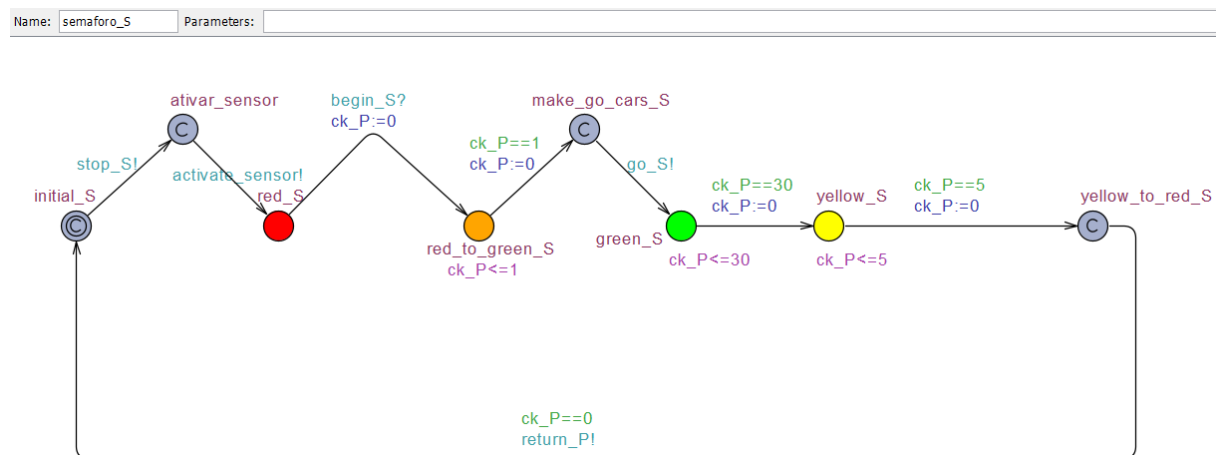


Figura 10: Semaforo\_S

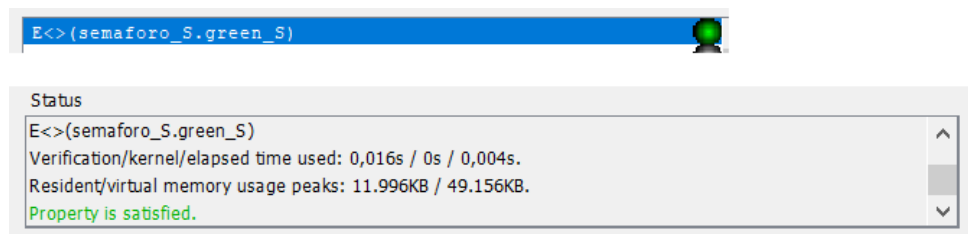
Este autômato pretende simular as mudanças de cores do semáforo da estrada secundária. O seu estado inicial é "initial\_S", que é um estado committed ou seja não passa tempo lá, e é enviada de imediato a sincronização "stop\_S"(recebida, como nos podemos recordar, pelo autômato road\_S), e passa-se para o estado "ativar\_sensor", que também é committed ou seja envia sem perder tempo a sincronização "activate\_sensor"(recebida por Sensor\_S), e passa para o estado "red\_S", que tal como demonstrado pelas cores do autômato, significa que o semáforo S vai ficar vermelho. De seguida, ele fica à espera de receber a sincronização "begin\_S", vinda do semáforo P, para poder continuar a sua execução - relembrando que isto só acontece quando o semáforo P recebe a informação que há carros à espera na road\_S, ou seja, só depois disto é que o semáforo S poderá seguir para verde. Uma vez recebida a sincronização, passa-se para o estado red\_to\_green\_S, no qual fica 1 segundo pelas razões anteriormente explicadas, e passado esse segundo vai para o estado committed make\_go\_cars\_S, que envia sem esperar a sincronização "go\_S", recebida pelo autômato road\_S, e fica finalmente verde, chegando ao estado "green\_S", tal como ilustrado pela cor. Fica neste estado durante 30 segundos, ou seja o semáforo S fica verde durante 30 segundos, e de seguida passa a



amarelo, ou seja para o estado "yellow\_S", onde fica 5 segundos, passando para o estado committed "yellow\_to\_red\_S", que envia imediatamente a sincronização "return\_P" ao autómato semaforo\_P para ele recomeçar a sua execução, e volta assim ao estado inicial.

## 1.2 Express in CTL the following reachability properties and test them in UPPAAL: (1) the minor-road light can go green; (2) the major-road light can go red.

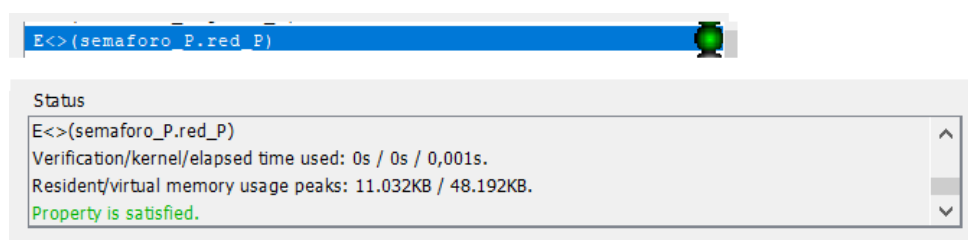
### 1.2.1 The minor-road light can go green



**Figura 11:** Propriedade CTL: o semáforo da estrada secundária pode ficar verde

Pode verificar-se que a seguinte propriedade é satisfeita: O semáforo da estrada secundária pode ficar verde. Verificámos isto ao utilizar o E<>, ou seja "existe um caminho" e um estado, no qual o semáforo secundário fica verde ("semaforo\_S.green\_S"). Uma vez que quando damos check com o Verifier a luz fica verde, significa que a propriedade é satisfeita.

### 1.2.2 The major-road light can go red

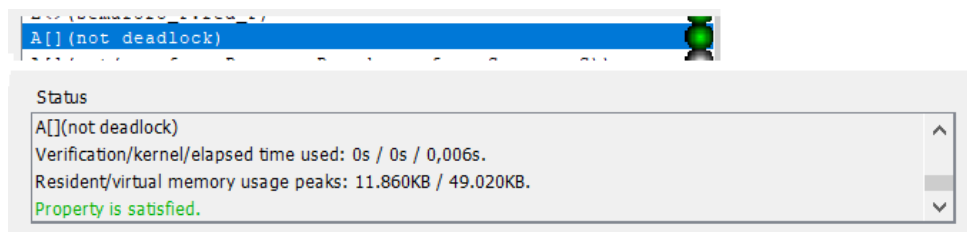


**Figura 12:** Propriedade CTL: o semáforo da estrada principal pode ficar vermelho

Pode verificar-se que a seguinte propriedade é satisfeita: O semáforo da estrada principal pode ficar vermelho. Verificámos isto ao utilizar o E<>, ou seja "existe um caminho" e um estado no qual o semáforo principal fica vermelho ("semaforo\_P.red\_P"). Uma vez que quando damos check com o Verifier a luz fica verde, significa que a propriedade é satisfeita.

### 1.3 Express in CTL the following safety properties and test them in UPPAAL: (3) the system never enters in a deadlock state; (4) the minor-road and major-road lights cannot be green at the same time.

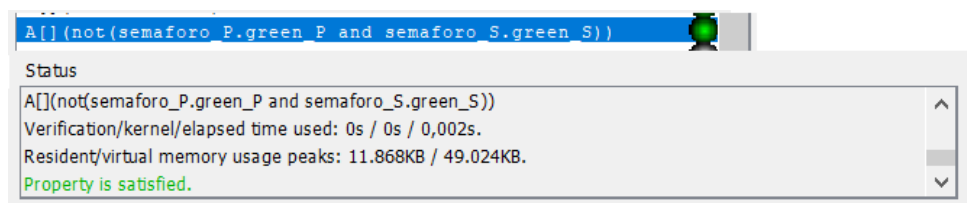
#### 1.3.1 The system never enters in a deadlock state



**Figura 13:** Propriedade CTL: o sistema nunca entra em deadlock

Pode verificar-se que a seguinte propriedade é satisfeita: o sistema nunca entra em deadlock. Verificámos isto ao utilizar o `A[]`, ou seja "em todos os caminhos" e em todos os estados, o sistema não entra em deadlock, "not deadlock". Uma vez que quando damos check com o Verifier a luz fica verde, significa que a propriedade é satisfeita.

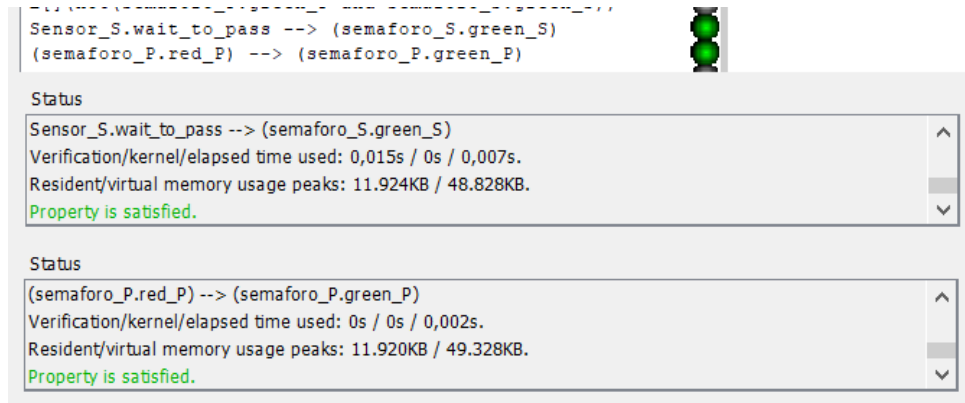
#### 1.3.2 The minor-road and major-road lights cannot be green at the same time



**Figura 14:** Propriedade CTL: os semáforos não podem estar verdes ao mesmo tempo

Pode verificar-se que a seguinte propriedade é satisfeita: o semáforo da estrada principal e o semáforo da estrada secundária nunca ficam verdes ao mesmo tempo. Verificámos isto ao utilizar o `A[]`, ou seja "em todos os caminhos" e em todos os estados, o semáforo principal e o semáforo secundário não podem ficar verdes simultaneamente ("`not (semaforo_P.green_P and semaforo_S.green_S)`"). Uma vez que quando damos check com o Verifier a luz fica verde, significa que a propriedade é satisfeita.

**1.4 Express in CTL the following liveness property and test it in UPPAAL: (5) if there are cars waiting they will eventually have green light.**



**Figura 15:** Propriedade CTL: se existem carros à espera, eventualmente terão uma luz verde

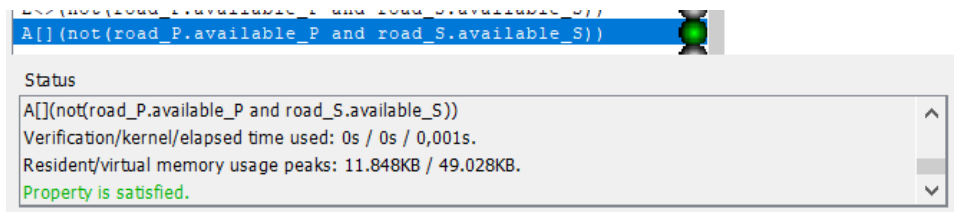
Pode verificar-se que a seguinte propriedade é satisfeita: e existem carros à espera, eventualmente terão uma luz verde. Dividimos o raciocínio em 2 casos: na estrada secundária e na estrada principal. Para verificar esta propriedade, em ambos os casos, usámos `-->`, que significa "dentro de determinadas condições algo inevitavelmente vai acontecer".

Para verificar esta propriedade para a estrada secundária, dissemos que, se o sensor se encontra no estado "wait\_to\_pass", ou seja, quando os carros da estrada secundária se encontram à espera para passar, eventualmente o semáforo S ficará verde:  
`Sensor_S.wait_to_pass --> (semaforo_S.green_S)`

Para verificar esta propriedade para a estrada principal, dissemos que, se o semáforo da estrada principal se encontra vermelho (ou seja, seria a única opção desta estrada ter carros à espera), eventualmente o semáforo P vai ficar verde:  
`(semaforo_P.red_P) --> (semaforo_P.green_P)`

## 1.5 Can you think of other desirable properties? If so please register them and check whether they hold or not.

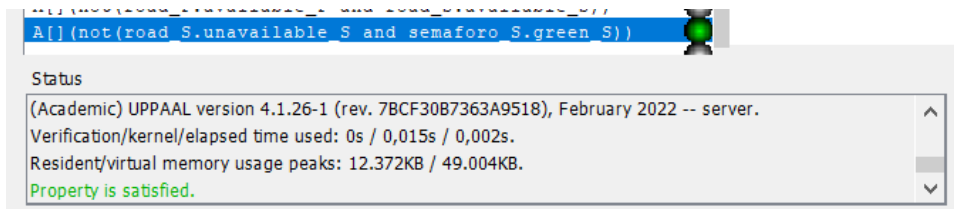
### 1.5.1 Não há nenhum caminho em que as duas ruas estejam disponíveis ao mesmo tempo



**Figura 16:** Propriedade CTL: Não há nenhum caminho em que as duas ruas estejam disponíveis ao mesmo tempo

Pode verificar-se que a seguinte propriedade é satisfeita: Não há nenhum caminho em que as duas ruas estejam disponíveis ao mesmo tempo. Verificámos isto ao utilizar o `A[]`, ou seja "em todos os caminhos" e todos os estados, a rua principal e a rua secundária não podem ficar disponíveis simultaneamente `"(not(road_P.available_P and road_S.available_S))"`. Uma vez que quando damos check com o Verifier a luz fica verde, significa que a propriedade é satisfeita.

### 1.5.2 Não há nenhum caminho em que a rua secundária esteja indisponível e que o seu semáforo esteja verde



**Figura 17:** Propriedade CTL: Não há nenhum caminho em que a rua secundária esteja indisponível e que o seu semáforo esteja verde

Pode verificar-se que a seguinte propriedade é satisfeita: Não há nenhum caminho em que a rua secundária esteja indisponível e que o seu semáforo esteja verde. Verificámos isto ao utilizar o `A[]`, ou seja "em todos os caminhos" e todos os estados, a rua secundária não pode estar indisponível ao mesmo tempo que o seu semáforo está verde `"not(road_S.unavailable_S and semaforo_S.green_S)"`. Uma vez que quando damos check com o Verifier a luz fica verde, significa que a propriedade é satisfeita.

## **1.6 Justificação das opções tomadas e conclusões obtidas**

Para criarmos esta montagem, tentámos no fundo imitar o funcionamento de um sensor real, com ativação, deteção, envio do sinal e desativação. Tentámos colocar-nos no lugar dos condutores e imaginar como faria sentido cada decisão que tomámos, tentando sempre adaptar as nossas escolhas a uma situação real.

Por exemplo, a criação das `road_S` e `road_P` teve por base a tentativa de dar uma ideia de os carros poderem ou não passar, não dando indicações de para onde, uma vez que os semáforos não necessitam dessa informação.

Além disso, tentámos montar os semáforos também como se estivéssemos nos cruzamentos a imaginar como seria a vida real, o que tornou intuitiva a montagem.

No fundo tentámos incorporar todos os elementos que imaginávamos numa situação real: o aparecimento de carros, uma estrada com carros a passar ou não, os semáforos como controladores do sistema geral, todos estes elementos reais. Isto tornou muito intuitiva a criação e montagem de todo o nosso sistema.

Podemos concluir então que esta estratégia permitiu que obtivéssemos êxito em cumprir todos os requisitos da primeira parte do trabalho e ainda nos permitiu sermos rápidos na elaboração da mesma.

## 2 Second part

The previous system of traffic lights works reasonably well under the assumption that one of the roads has more traffic than the other. But such an assumption is often too strong: it may be the case that both roads have the same amount of traffic, or even that their traffic flow varies drastically throughout the day. The second part of this assignment (more exploratory) aims to address precisely this problem which is well-known to have significant impact in the economy and the environment <sup>1</sup>. To this effect, we can now assume that each traffic light has a smart sensor attached to it. The sensor informs whether the traffic near the light is **high**, **low**, or simply **non-existent**.

### The second part of the assignment:

1. Adapt your previous UPPAAL model to take into account the information provided by the sensors. One expects, for example, that if the rightmost sensor outputs **high** and the other sensors output **no** then the rightmost traffic light should be on green at least until the sensors provide new information.
2. Verify that all the properties mentioned in the first part of the assignment still hold
3. Note that the second part of the assignment is of a more exploratory nature, and thus we give freedom to adjust sensor parameters as seen fit in order to promote different and creative solutions. We will value properties expressed in CTL that say something about the efficiency of the system developed by the students. Such a property can be for example, “If the rightmost sensor always detects high traffic and the others detect no traffic at all, then we will observe at most one change in the traffic lights”.

## 2.1 Adapt your previous UPPAAL model to take into account the information provided by the sensors

Para modelarmos o sistema descrito, utilizámos 5 autómatos: Sensor\_S, Sensor\_P, Casos, Semaforo\_P, Semaforo\_S.

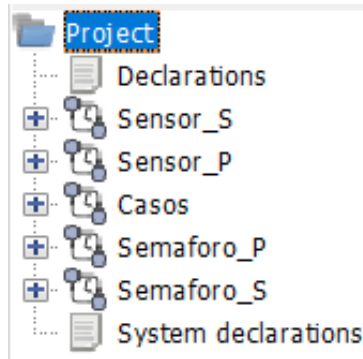


Figura 18: Autómatos elaborados

Na qual as declarações que utilizamos foram as seguintes:

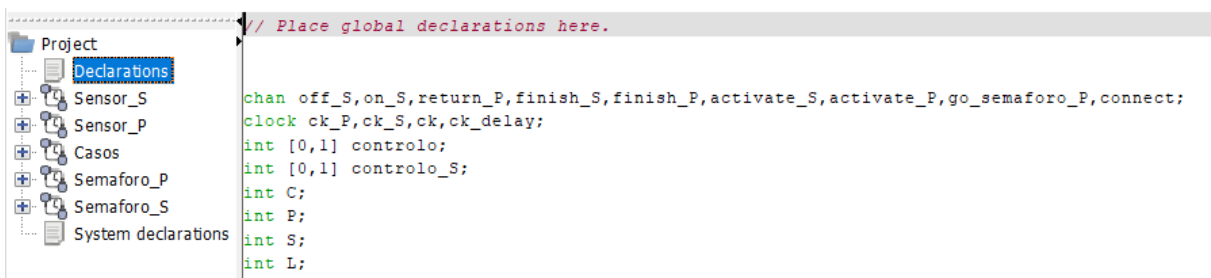


Figura 19: Declarações

E criou-se as seguintes declarações do sistema:

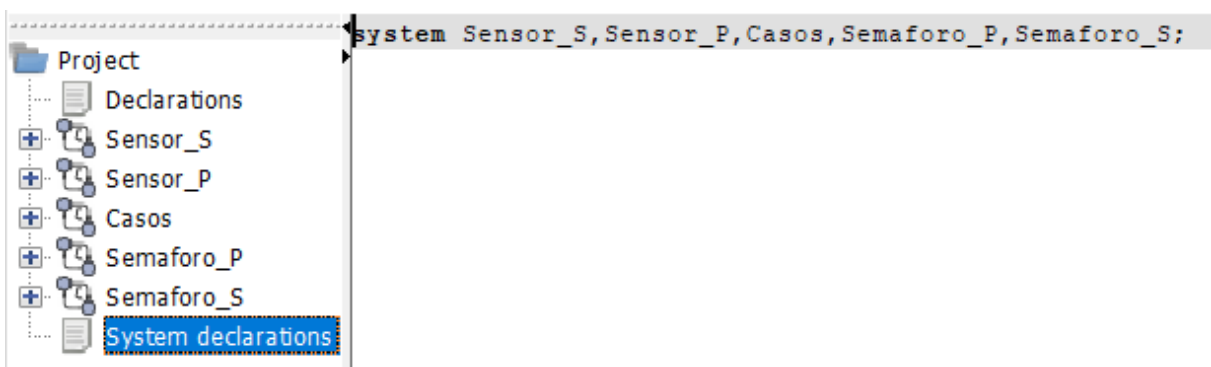


Figura 20: Declarações do sistema

### 2.1.1 Sensor\_P e Sensor\_S :

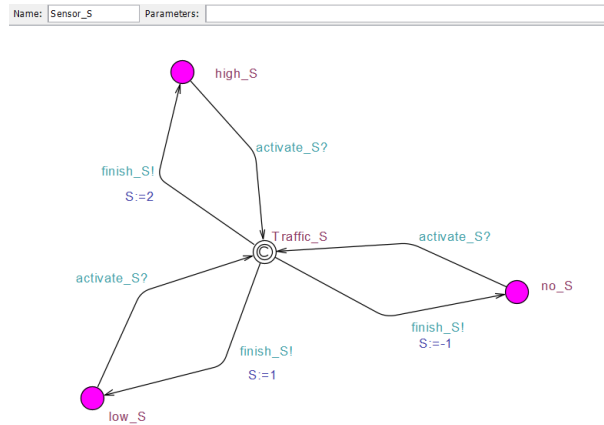


Figura 21: *Sensor\_S*

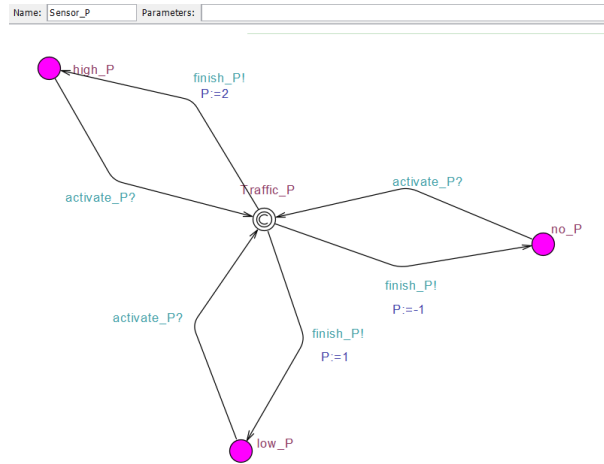


Figura 22: *Sensor\_P*

Estes dois autómatos estão construídos essencialmente da mesma maneira. Estes pretendem simular a afluência de tráfego na estrada principal (Sensor\_P) e na estrada secundária (Sensor\_S). Num cômputo geral, estes sensores pretendem guardar a informação, que é gerada de maneira aleatória, se o tráfego na respetiva estrada é high (alto), low (pouco), ou no (inexistente). Guardam esta informação numa variável: A variável S guarda o valor de tráfego na estrada secundária e a variável P guarda o valor de tráfego na estrada principal (2 se for high, 1 se for low, -1 se for no). Para se transicionar do estado inicial "Traffic" para estes estados tem de se enviar a sincronização "finish", que vai ser recebida mais tarde no autómato Casos para informar que já transicionou para o estado do tráfego, e vai ser enviada ao mesmo tempo que se guarda a informação da quantidade de tráfego nas variáveis previamente mencionadas. Volta-se ao autómato inicial quando se receber a sincronização "activate", que vem do autómato Semaforo\_P, no final da sua execução.



## 2.1.2 Casos

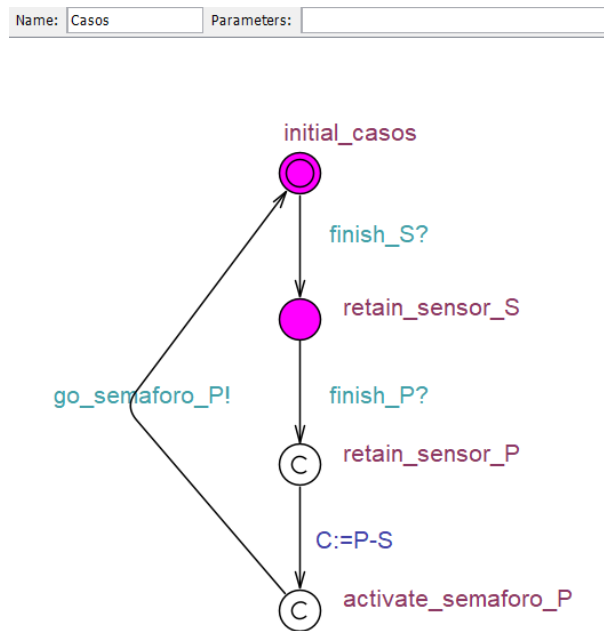
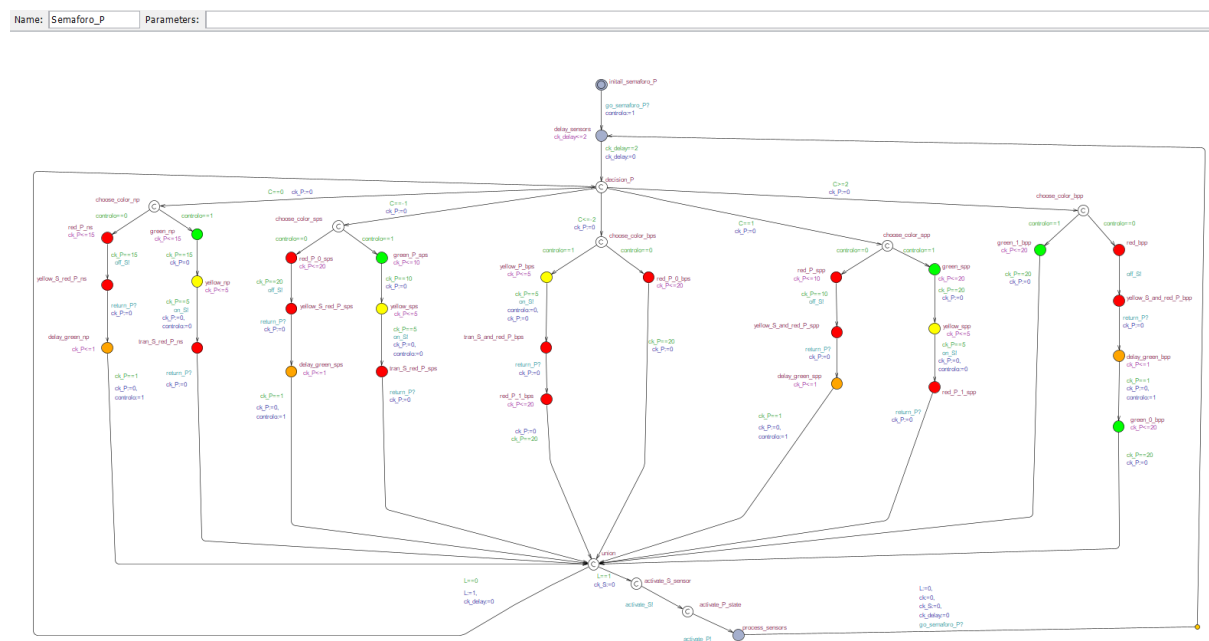


Figura 23: Casos

Este autômato tem como estado inicial `initial_casos`. Para sair deste estado, precisa de receber a sincronização "finish\_S", que é enviada vinda do autômato Sensor\_S para informar que já transicionou para o estado atual do tráfego da estrada secundária. De seguida, entra no estado `retain_sensor_S`, e para transicionar para o próximo estado tem de receber a sincronização "finish\_P", que é enviada vinda do autômato Sensor\_P para informar que já transicionou para o estado atual do tráfego da estrada principal. Transiciona assim para o estado `retain_sensor_P`, que é committed portanto transiciona sem perder tempo para o estado seguinte, `activate_semaforo_P`, ao mesmo tempo que guarda na variável C a diferença entre as variáveis P e S. Uma vez que os valores destas variáveis dependem da intensidade do tráfego em cada estrada, e estão definidos como  $C=0$  (sem preferência),  $C=1/C=-1$  (baixa preferência para a estrada primária/secundária) e  $C \geq 2/C \leq -2$  (alta preferência para a estrada primária/secundária), a variável C, correspondendo à conta P-S, poderá corresponder aos valores -3, -2, -1, 0, 1, 2 e 3. Ao se transicionar para o estado inicial, é enviada a sincronização "go\_semaforo\_P", que será recebida pelo autômato Semaforo\_P para inicializar a sua execução.

### 2.1.3 Semaforo\_P



**Figura 24:** *Semaforo\_P*

O autômato Semaforo\_P é o mais complexo do nosso modelo, por ser o autômato que trata dos 5 casos diferentes, correspondentes aos 7 valores diferentes da variável C previamente mencionados. Importante será mencionar, antes da explicação do funcionamento do autômato, que a variável "controle" corresponderá à cor do semáforo principal: se estiver verde a variável controle será 1, se estiver vermelho será 0. O valor desta variável vai decidir qual de 2 caminhos é escolhido no autômato dentro de cada um dos 5 casos, de modo a simplificarmos o comportamento dos nossos semáforos.

Este autômato tem como estado inicial `initial_semaforo_P`, e quando recebe a sincronização `go_semaforo_P` pela primeira vez, provinda do autômato `Casos`, vai dar à variável `controle` o valor 1, para começar a correr. Passará então ao estado `delay_sensors`, onde vai esperar 2 segundos devido à guarda `ck_delay`, e passados 2 segundos transicionará para o estado `decision_P`, zerando o `ck_delay` na transição (este estado `delay_sensores` representa o delay da ativação, detecção do congestionamento e envio do sinal por parte dos sensores). O estado `decision_P` é de onde vão derivar os 5 casos mencionados previamente. O caminho seguinte, de entre os 5, vai ser escolhido de acordo com o valor da variável `C`, que virá do autômato `Casos`. Os 5 casos serão explicados individualmente a seguir. Depois de dar uma "volta", ou seja, de entrar para um dos 5 ramos da variável `C`, e para um dos dois ramos da variável `controle`, seguirá para o estado `committed "union"`. Nesta primeira volta, a variável `L` estará a 0, e assim sendo, o autômato vai dar outra volta: volta para o estado `"decision_P"` e vai passar exatamente no mesmo dos 5 ramos da variável `C` que passou na primeira volta (`C` não muda), mas vai passar agora no outro ramo da variável de `controle` (se tinha passado no 0, passa agora no 1 e vice-versa), uma vez que a variável `controle` muda de valor no fim de cada ramo (exceto em alguns casos, como quando o semáforo `P` já está a verde e os sensores

indicaram que tem uma alta preferência nessa estrada, fazendo com que passe duas vezes pelo mesmo caminho). Decidimos fazer a estratégia das duas voltas de modo a simplificar cada ramo, e de modo a alternar mais as luzes verdes e vermelhas. Tínhamos feito de outra maneira antes, que fazia com que os tempos a verde/vermelho dos semáforos se acumulasse, o que não era suposto, e isto contornou esse problema, tornando a nossa abordagem mais realista. Ora, disto resulta que, por exemplo, em  $C=0$  (tráfego igual em ambas as estradas) na primeira volta o semáforo da estrada principal passe por exemplo 15 segundos a verde, e na segunda volta 15 segundos a vermelho. Por exemplo, em  $C=1$  (ligeiramente mais afluência na estrada principal), passa 20 segundos a verde e 10 a vermelho, em  $C=3$ , passa os 30 segundos a verde. Ora, depois desta segunda volta, voltamos ao estado "union" e agora a nossa variável  $L$  já é 1, então passamos para o estado "activate\_S\_sensor", que de seguida envia a sincronização "activate\_S" ao autómato Sensor\_S e vai para o estado "activate\_P\_sensor", que envia a sincronização "activate\_P" ao autómato Sensor\_P (de modo a estes autómatos poderem fazer uma nova leitura do estado das estradas) - e acaba aqui uma volta completa dada por este autómato, depois de zerar todas as suas variáveis, e ficar à espera de receber novo da sincronização go\_semaforo\_P para reiniciar a sua atividade com o novo estado da estrada, a partir do estado delay\_sensors.

**$C==0$  : Tráfego igual na estrada principal e secundária** O primeiro estado será choose\_color\_np. A partir deste estado, se o controlo estiver a 0 (semáforo vermelho), vai para um estado que corresponde a isso mesmo "red\_P\_ns", e fica a vermelho durante 15 segundos. Após estes 15 segundos, envia a sincronização off\_S para o Semáforo\_S, que manda o semáforo da estrada secundária ficar a vermelho, e passa para o estado yellow\_S\_red\_P\_ns. Fica à espera de receber a sincronização return\_P, que é enviada no final da execução do autómato Semáforo\_S, e transiciona para o estado delay\_green\_np, onde fica durante 1 segundo, de modo a respeitar o segundo que o semáforo demora a mudar de cor, mencionado na parte 1. Por fim, muda de cor para verde, colocando a variável controlo a 1. Tal como explicado anteriormente, dará uma segunda volta ainda com  $C==0$ , mas desta vez com controlo==1, e entrará para o outro ramo, começando pelo estado green\_np. O semáforo principal fica neste estado, ou seja, a verde, durante 15 segundos, e depois passa para amarelo, para o estado "yellow\_np", onde fica 5 segundos (que é o tempo que o semáforo tem de ficar amarelo). Passados estes 5 segundos, transiciona para o estado tran\_S\_red\_P\_ns, colocando a variável controlo a 0 (ou seja, semáforo principal a vermelho) e enviando a sincronização "on\_S" para o Semáforo\_S, que manda o semáforo da estrada secundária ficar a verde. Fica mais uma vez à espera de receber a sincronização return\_P, que é enviada no final da execução do autómato Semáforo\_S, e termina a execução deste caso. De notar que caso o controlo estivesse a 1 no início da execução, seriam dadas as duas voltas na mesma, só que pela ordem contrária (primeiro ficava 15 segundos a verde, depois a vermelho).

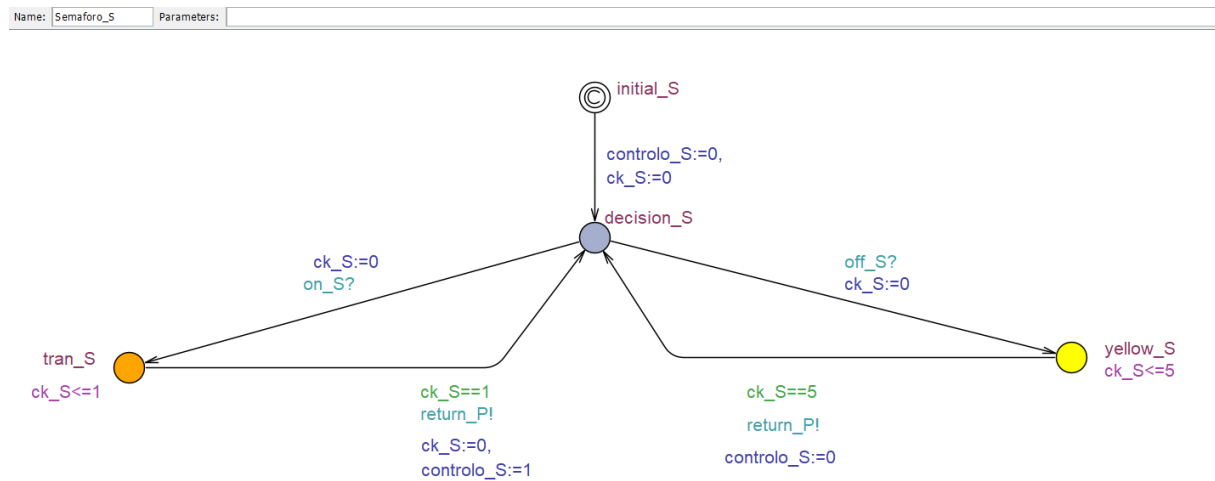
**$C==1$  : Tráfego ligeiramente maior na estrada principal e  $C==-1$  : Tráfego ligeiramente maior na estrada secundária** O funcionamento destes dois casos será muito semelhante ao do caso  $C==0$ , porém, em vez de o semáforo principal ficar 15 segundos em cada cor, ficará 20 segundos a verde e 10 segundos a vermelho no caso  $C==1$ , e ficará 20 segundos a vermelho e 10 segundos a verde no caso  $C==-1$ . Serão mais

uma vez dadas duas voltas por cada C, para correr os dois ramos possíveis da variável "controlo".

**C $\geq$ 2: Tráfego na estrada principal e sem tráfego na estrada secundária** Neste caso, queremos que o semáforo principal fique verde 40 segundos. Assim sendo, se o controlo for 1, transiciona para o estado green\_1\_bpp e fica neste estado durante 20 segundos, indo logo para a segunda volta. Na segunda volta, como neste caso a variável controlo não muda, dá a mesma volta e fica 20 segundos no verde, somando fica 40 segundos a verde. Caso o controlo inicial seja 0, o semáforo principal está vermelho, no estado red\_bpp, e transiciona para o estado yellow\_S\_and\_red\_P\_bpp, enquanto envia a sincronização off\_S para o Semaforo\_S, que manda o semáforo da estrada secundária ficar a vermelho. De seguida, fica à espera de receber a sincronização return\_P, que é enviada no final da execução do autómato Semaforo\_S, e transiciona para o estado delay\_green\_bpp, onde fica durante 1 segundo, de modo a respeitar o segundo que o semáforo demora a mudar de cor, mencionado na parte 1. Por fim, muda de cor para verde, colocando a variável controlo a 1, e transicionando para o estado green\_0\_bpp, onde o semáforo principal fica durante 20 segundos a verde. Tal como explicado anteriormente, dará uma segunda volta ainda com C $\geq$ 2, mas desta vez com controlo=1, e entrará para o outro ramo, já explicado anteriormente, que faz com que o semáforo fique verde durante mais 20 segundos, dando um total de 40 segundos a verde.

**C $\leq$ -2: Tráfego na estrada secundária e sem tráfego na estrada principal** O funcionamento deste ramo é muito semelhante ao ramo anterior, porém invertendo as cores. Neste caso, se o controlo começa a 0 fica 20 segundos a vermelho, e na segunda volta fica de novo 20 segundos a vermelho, totalizando, como queríamos, 40 segundos a vermelho. Caso comece a verde, começa por mudar logo para o estado amarelo yellow\_P\_bps, fica 5 segundos no amarelo, e de seguida coloca o controlo a 0, ao mesmo tempo que envia a sincronização on\_S, que faz o semáforo S ficar verde. De seguida, recebe o return\_P já explicado, e fica no estado vermelho durante 20 segundos. Após estes 20 segundos, dá a volta e como o controlo está a 0 apenas fica no mesmo estado durante 20 segundos, totalizando 40 segundos a vermelho.

## 2.1.4 Semaforo\_S



**Figura 25:** *Semaforo\_S*

O semáforo secundário tem como estado inicial `initial_S`. Tal como no semáforo principal, existe aqui uma variável `control_S` que estando a 1 significa que o semáforo secundário está verde, estando a 0 significa que o semáforo secundário está a vermelho. Define-se o `control_S` inicial como 0, para contrastar com o facto de o semáforo principal começar com o seu `control_S` a 1, uma vez que `control_S` e `control_P` nunca podem estar os dois a 1. De seguida, transiciona-se para o estado `decision_S`. Deste estado, pode tomar-se dois caminhos, que já foram brevemente mencionados anteriormente. De notar que, uma vez que obrigamos o semáforo S a mudar de cor com o semáforo P, para a cor inversa, os tempos definidos no autómato `Semaforo_P` em que o semáforo principal fica a vermelho correspondem aos tempos que o semáforo secundário fica a verde, e vice versa.

**on\_S:** A sincronização enviada pelo semáforo principal recebida pode ser "on\_S", que no fundo manda o semáforo da estrada secundária ficar a verde. Portanto, transiciona-se para o estado `tran_S`, que simboliza a mudança de cores de vermelho para verde, que demora 1 segundo a ocorrer, de acordo com o enunciado, então fica 1 segundo neste estado. Depois desse segundo, transiciona de volta para o estado `decision_S`, e na transição envia a sincronização `return_P` para o `Semaforo_P`, para ele continuar a sua execução, e coloca o `control_S` a 1, ou seja, coloca o semáforo S a verde.

**off\_S:** A sincronização enviada pelo semáforo principal recebida pode ser "off\_S", que no fundo manda o semáforo da estrada secundária ficar a vermelho. Portanto, transiciona-se para o estado `yellow_S`, que significa que o semáforo S fica a amarelo, que demora 5 segundos a ocorrer, de acordo com o enunciado, então fica 5 segundos neste estado. Depois transiciona de volta para o estado `decision_S`, e na transição envia a sincronização `return_P` para o `Semaforo_P`, para ele continuar a sua execução, e coloca o `control_S` a 0, ou seja, coloca semáforo S a vermelho.

## Esquema elucidativo que demonstra alguns exemplos de estados possíveis do nosso modelo



























Linha temporal e procedimentos	Estado em P	Estado em S
Estado inicial do sistema: Semáforo P: Verde Semáforo S: Vermelho  Tempo: ck=0		
Semáforo P: Verde Semáforo S: Vermelho  <b>Preparação e detecção do sinal do sensor:</b> P: Low S: Low  <b>Classificação do trânsito:</b> Sem preferência (C=0)  <b>Delay</b> = 2 segundos  ck=2	 LOW 	 LOW 
Semáforo P: Verde Semáforo S: Vermelho  <b>Delay</b> = 15 segundos  ck=17		
Semáforo P: Amarelo Semáforo S: Vermelho  <b>Delay</b> = 5 segundos  ck=22		
Semáforo P: Vermelho Semáforo S: Vermelho  <b>Delay</b> = 1 segundo  ck=23		
Semáforo P: Vermelho Semáforo S: Verde  <b>Delay</b> = 15 segundos  ck=38		
Semáforo P: Vermelho Semáforo S: Amarelo  <b>Delay</b> = 5 segundos  ck=43		
Semáforo P: Vermelho Semáforo S: Vermelho  <b>Delay</b> = 1 segundo  ck=44 Reset(ck)		
Semáforo P: Verde Semáforo S: Vermelho  <b>Preparação e detecção do sinal do sensor:</b> P: High S: Low  <b>Classificação do trânsito:</b> Baixa preferência no P (C=1)  <b>Delay dos sensores</b> = 2 segundos  ck=2	 HIGH 	 LOW 
Semáforo P: Verde Semáforo S: Vermelho  <b>Delay</b> = 20 segundos  ck=22		
Semáforo P: Amarelo Semáforo S: Vermelho  <b>Delay</b> = 5 segundos  ck=27		

Figura 26: Parte 1 do esquema

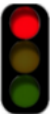
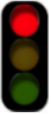


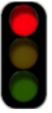
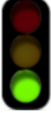
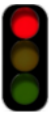

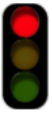

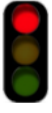

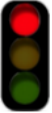











Semáforo P: Vermelho Semáforo S: Vermelho  <b>Delay= 1 segundo</b>  ck=28			Semáforo P: Amarelo Semáforo S: Vermelho  <b>Delay= 5 segundos</b>  ck=17		
Semáforo P: Vermelho Semáforo S: Verde  <b>Delay= 10 segundos</b>  ck=38			Semáforo P: Vermelho Semáforo S: Vermelho  <b>Delay= 1 segundo</b>  ck=18		
Semáforo P: Vermelho Semáforo S: Amarelo  <b>Delay= 5 segundos</b>  ck=43			Semáforo P: Vermelho Semáforo S: Verde  <b>Delay= 20 segundos</b>  ck=38		
Semáforo P: Vermelho Semáforo S: Vermelho  <b>Delay= 1 segundo</b>  ck=44 Reset(ck)			Semáforo P: Vermelho Semáforo S: Amarelo  <b>Delay= 5 segundos</b>  ck=43		
Semáforo P: Verde Semáforo S: Vermelho  <b>Preparação e detecção do sinal do sensor:</b> P: Low S: High  <b>Classificação do transito:</b> Baixa preferência no S (C=-1)  <b>Delay dos sensores = 2 segundos</b>  ck=2			Semáforo P: Vermelho Semáforo S: Vermelho  <b>Delay= 1 segundo</b>  ck=44 Reset(ck)		
Semáforo P: Verde Semáforo S: Vermelho  <b>Delay= 10 segundos</b>  ck=12			Semáforo P: Verde Semáforo S: Vermelho  <b>Preparação e detecção do sinal do sensor:</b> P: No S: low  <b>Classificação do transito:</b> Alta preferência no S (C=-2)  <b>Delay dos sensores = 2 segundos</b>  ck=2		

Figura 27: Parte 2 do esquema



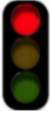



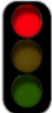

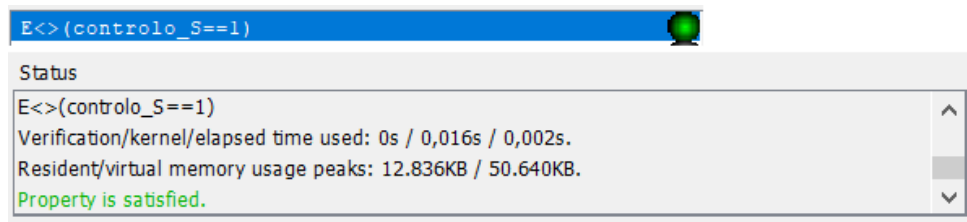
Semáforo P: Amarelo Semáforo S: Vermelho  <b>Delay= 5 segundos</b>  ck=7			Semáforo P: Vermelho Semáforo S: Verde  <b>Delay= 20 segundos</b>  ck=28		
Semáforo P: Vermelho Semáforo S: Vermelho  <b>Delay= 1 segundo</b>  ck=8			Semáforo P: Vermelho Semáforo S: Verde  <b>Delay= 20 segundos.</b>  ck=48 Reset(ck)		

Figura 28: Parte 3 do esquema

## 2.2 Verify that all the properties mentioned in the first part of the assignment still hold

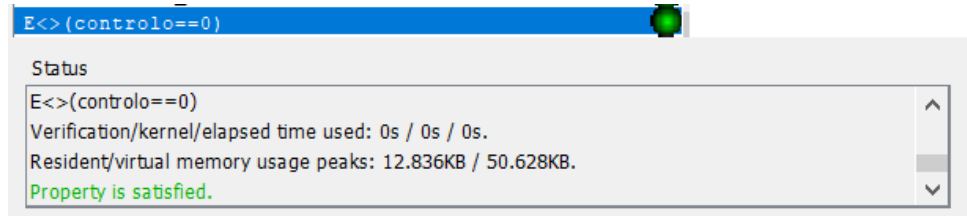
### 2.2.1 The minor-road light can go green



**Figura 29:** Propriedade CTL: o semáforo da estrada secundária pode ficar verde

Pode verificar-se que a seguinte propriedade é satisfeita: O semáforo da estrada secundária pode ficar verde. Verificámos isto ao utilizar o  $E<>$ , ou seja "existe um caminho" e um estado, no qual o semáforo secundário fica verde (" $\text{controlo\_S}==1$ "). Uma vez que quando damos check com o Verifier a luz fica verde, significa que a propriedade é satisfeita.

### 2.2.2 The major-road light can go red

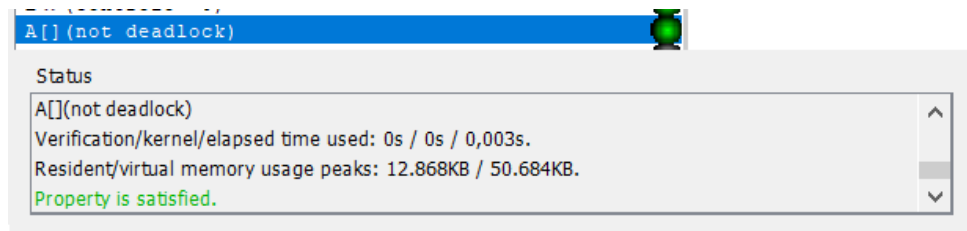


**Figura 30:** Propriedade CTL: o semáforo da estrada principal pode ficar vermelho

Pode verificar-se que a seguinte propriedade é satisfeita: O semáforo da estrada principal pode ficar vermelho. Verificámos isto ao utilizar o  $E<>$ , ou seja "existe um caminho" e um estado no qual o semáforo principal fica vermelho (" $\text{controlo}==0$ "). Uma vez que quando damos check com o Verifier a luz fica verde, significa que a propriedade é satisfeita.



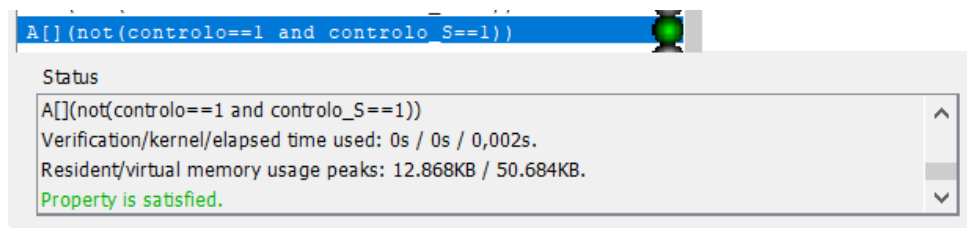
### 2.2.3 The system never enters in a deadlock state



**Figura 31:** Propriedade CTL: o sistema nunca entra em deadlock

Pode verificar-se que a seguinte propriedade é satisfeita: o sistema nunca entra em deadlock. Verificámos isto ao utilizar o  $A[]$ , ou seja "em todos os caminhos" e em todos os estados, o sistema não entra em deadlock, "not deadlock". Uma vez que quando damos check com o Verifier a luz fica verde, significa que a propriedade é satisfeita.

### 2.2.4 The minor-road and major-road lights cannot be green at the same time



**Figura 32:** Propriedade CTL: os semáforos não podem estar verdes ao mesmo tempo

Pode verificar-se que a seguinte propriedade é satisfeita: o semáforo da estrada principal e o semáforo da estrada secundária nunca ficam verdes ao mesmo tempo. Verificámos isto ao utilizar o  $A[]$ , ou seja "em todos os caminhos" e em todos os estados, o semáforo principal e o semáforo secundário não podem ficar verdes simultaneamente ("not(controlo==1 and controlo\_S==1)"). Uma vez que quando damos check com o Verifier a luz fica verde, significa que a propriedade é satisfeita.


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

E<>(S==1 and controlo\_S==1)

E<>(S==2 and controlo\_S==1)

E<>(P==1 and controlo==1)

E<>(P==2 and controlo==1)



Status

E<>(S==1 and controlo\_S==1)  
Verification/kernel/elapsed time used: 0s / 0s / 0s.  
Resident/virtual memory usage peaks: 12.872KB / 50.688KB.  
Property is satisfied.

Status

E<>(S==2 and controlo\_S==1)  
Verification/kernel/elapsed time used: 0s / 0s / 0,001s.  
Resident/virtual memory usage peaks: 12.872KB / 50.688KB.  
Property is satisfied.

Status

E<>(P==1 and controlo==1)  
Verification/kernel/elapsed time used: 0s / 0s / 0s.  
Resident/virtual memory usage peaks: 12.880KB / 50.696KB.  
Property is satisfied.

Status

E<>(P==2 and controlo==1)  
Verification/kernel/elapsed time used: 0s / 0s / 0s.  
Resident/virtual memory usage peaks: 12.880KB / 50.696KB.  
Property is satisfied.

**Figura 33:** Propriedade CTL: se existem carros à espera, eventualmente terão uma luz verde

Pode verificar-se que a seguinte propriedade é satisfeita: se existem carros à espera, eventualmente terão uma luz verde. Dividimos o raciocínio em 4 casos: dois para a estrada secundária: quando  $S==2$  e quando  $S==1$ , e dois para a estrada principal: quando  $P==1$  e quando  $P==2$ . Relembrando, só existem carros à espera quando  $S$  ou  $P$  (as variáveis que indica a afluência da estrada secundária e principal, respetivamente) são 1 (afluência baixa) ou 2 (afluência alta). Decidimos verificar se "existe algum caminho" ( $E<>$ ) em que há pouca afluência na estrada secundária e o seu semáforo fica verde ( $S==1$  and  $controlo\_S==1$ ), em que há muita afluência na estrada secundária e o seu semáforo fica verde ( $S==2$  and  $controlo\_S==1$ ), em que há pouca afluência na estrada principal e o seu semáforo fica verde ( $P==1$  and  $controlo==1$ ) e em que há muita afluência na estrada principal e o seu semáforo fica verde ( $P==2$  and  $controlo==1$ ). Uma vez que quando damos check com o Verifier a luz fica verde em todos os casos, significa que as propriedades são satisfeitas.

## 2.3 Justificação das opções tomadas e conclusões obtidas

A elaboração destes autómatos foi bastante mais trabalhosa e demorada que a elaboração dos autómatos da parte 1 deste trabalho.

A maior dificuldade foi o facto de o sensor ser inteligente e envolver situações diferentes, principalmente por termos de ter em conta que o trânsito muda em ambas as estradas.

Para resolver isto, tivemos de utilizar estratégias diferentes nesta parte do projeto, tais como:

- A utilização de constantes que referem o estado dos semáforos (facilita nas decisões);
- A utilização de 4 clocks diferentes para controlar os tempos exigidos no enunciado e os estabelecidos por nós;
- Uma boa utilização de estados committed e loops de forma a eliminar as dificuldades encontradas ao longo do trabalho.

As dificuldades encontradas foram bastantes, tais como:

- A quantidade de deadlocks de difícil debug, na qual a maioria era devido ao exagerado número de estados comitted (o que nos permitiu aumentar o conhecimento e prática na sua utilização);
- A acumulação inadequada de tempo dos semáforos a verde e vermelho quando os sensores alteravam o C;
- Conflitos nas transições, por exemplo quando estava em baixa preferência para S e o sensor detetava alta preferência para P;
- etc.

Depois de corrigir esses aspetos, com o uso correto dos estados comitted, loops, etc, obtivemos o resultado final, que se aproxima bastante do que deveria acontecer na vida real.

Quanto à escolha do número de segundos para os semáforos ficarem a verde e vermelho em cada caso, a decisão foi estabelecida conforme o que consideramos mais realista. Se não há preferência, então 15 segundos a verde e vermelho em cada estrada pareceu-nos ser o adequado; quando há baixa preferência para uma das estradas, atribui-se 20 segundos a verde e 10 segundos a vermelho para essa estrada ; se houver alta preferência numa das estradas, achámos indicado colocar o seu semáforo a verde durante 40 segundos.