Projet Système de feux tricolores

Adrien Garandel Ran Bao Franck Boncler
Jérémy Bardon

6 novembre 2015

Table des matières

1 Partie 1 - Deux feux synchronisés

Dans cette partie, on s'intéresse à la synchronisation de deux feux tricolores. On utilise pour cela des réseaux de Pétri et le logiciel Roméo.

1.1 Questions 1 et 2

Feux non synchronisés et non temporises

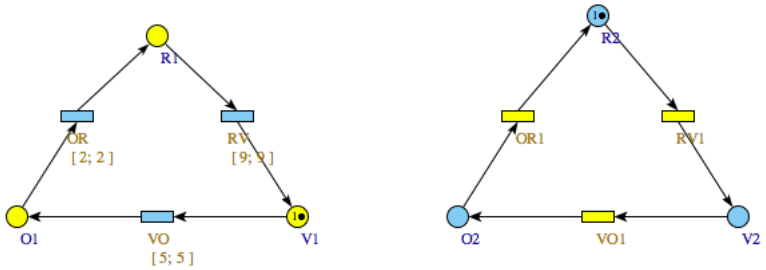

FIGURE 1 – Réseau de Pétri, feux non synchronisés

Chaque feu est identique et dispose de trois places suivant les trois couleurs qu'un feu de circulation tricolore peut prendre : Rouge, Orange et Vert.

1.2 Question 3

Feux synchronisés et non temporises

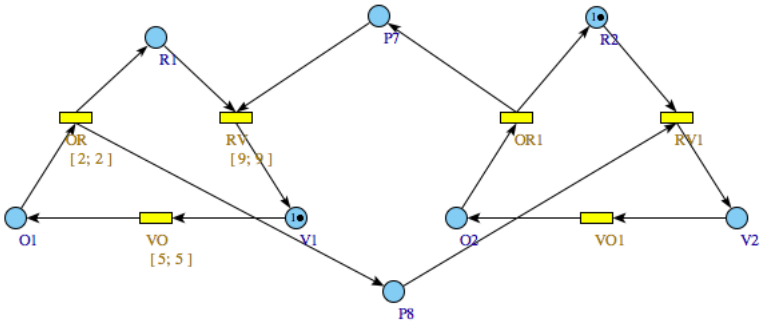

FIGURE 2 – Réseau de Pétri, feux synchronisés

Dans le cas réel, les feux de circulation sont synchronisés. En effet, lorsqu'un premier feu est Vert, le second est Rouge. Puis le premier feu passe alternativement au Orange puis au Rouge tandis qu'ensuite le second feu devient Vert.

Pour respecter ce cahier des charges, nous avons ajouté deux places entre les deux feux. Alors, lorsqu'un feu devient Vert, il empêche l'autre de devenir Vert puis le libère une fois qu'il est passé à l'Orange puis au Rouge.

1.3 Question 4

Un graphe de marquage permet de montrer les différentes executions possibles de notre système de réseaux de Pétri.

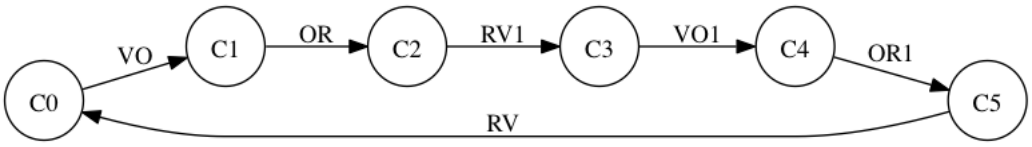

FIGURE 3 – Graphe de marquage des feux synchronisés

On constate que l'execution de notre système est linéaire. Cela signifie qu'il existe une seule execution, ainsi prouver que celle-ci est correcte revient à prouver que le système entier est correct.

1.4 Question 5 et 6

Vérification des propriétés de sureté et de vivacité du système de feux synchronisés.

- Il y a toujours au minimum un feu rouge (sureté)
  `AG[0, inf](M(R1) + M(R2) >= 1)`
- Le deuxième feu passe au moins une fois au vert (non bloqué)
  `AG[0, inf](M(V2) = 1)`
- Les feux ne se bloquent pas entre eux
  `AG[0, inf](M(R1) + M(P7) = 2) # P7 = place intermédiaire`

2 Partie 2 - Deux feux temporises

La partie précédente expose un système de feux synchronisés mais non temporises. En effet, dans la réalité les feux restent un temps donné dans leurs états. Nous proposons alors un système de feux temporises sous la forme d'automates à états finis modélisés avec le logiciel Uppaal.

2.1 Question 7

Feux temporisés sans synchronisation

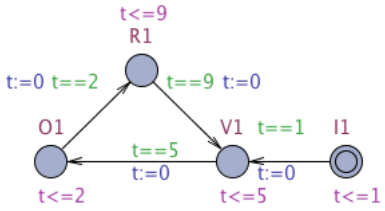

FIGURE 4 – Automate du feu commençant en Vert

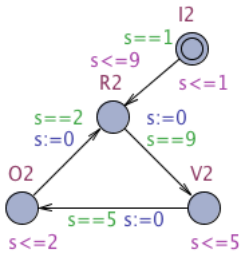

FIGURE 5 – Automate du feu commençant en Rouge

Par défaut, les automates ne démarrent pas en même temps et on observe une désynchronisation des feux. Une solution est d'ajouter un état avant l'état réel pour chacun des feux.

Validation des propriétés de sureté de vivacité du système de feux temporises.

- Toujours au moins un feu rouge
  `A[](feu1.R1 or feu2.R2 or feu1.I1 or feu2.I2)`
- Les deux feux atteignent leur état le plus éloigné, ils ne sont pas bloqués par le temps
  `E<>(feu1.R1 and feu2.02)`
- Pas de blocage
  `E<> deadlock`

2.2 Question 8

Afin de controller au mieux le système, cette solution propose l'utilisation d'un contrôleur.

Cet élément est chargé de faire évoluer les états des feux de manière correcte. Cela signifie par exemple qu'il lui est impossible de mettre les deux feux Verts en même temps.

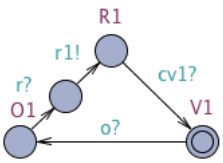

FIGURE 6 – Automate du feu commençant en Vert

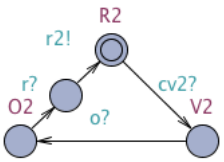

FIGURE 7 – Automate du feu commençant en Rouge

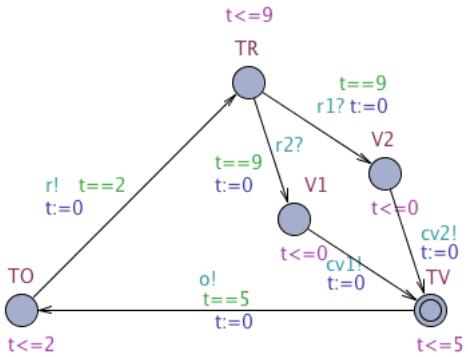

FIGURE 8 – Contrôleur des feux

Les feux ne sont pas temporises mais il y a toujours un feu Rouge et un feu Vert au démarrage du système. En effet, c'est le rôle du contrôleur qui temporise et commande les feux à travers une synchronisation par signaux.

Le contrôleur allume alors alternativement le premier et le second feu tout en prenant en compte le temps pendant lequel chaque feu doit rester dans un état donné.

Vérification des propriétés de sureté de vivacité du système de feux synchronisés

- Toujours au moins un feu rouge
  `A[](feu1.R1 or feu2.R2)`
- Les deux feux atteignent leur état le plus éloigné, ils ne sont pas bloqués par le temps
  `E<>(feu1.R1 and feu2.O2)`
- Pas de deadlock
  `E<> deadlock`

3 Partie 3 - Carrefour en T

Après avoir étudié des systèmes représentant la synchronisation de deux feux de circulation sur une route, nous allons nous intéresser à un carrefour en T.

Comme dans le premier cas, il y a une route principale avec deux feux tricolores. On ajoute ici une route secondaire sur le côté. Dans un cas normal, le feux de la route secondaire est Rouge mais lorsque des voitures sont captés il passe au Vert.

3.1 Question 9

Dans cette première version de notre système de feux, il n'y a pas de contraintes de temps.

- Les 2 feux de la route principale sont considérés comme un seul
- Il existe un processus qui régule l'arrivée des voitures dans la petite rue

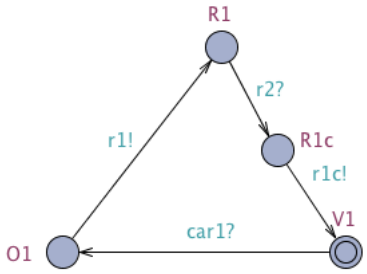

FIGURE 9 – Automate du feu de la route principale

Le feu de la route principale passe du Vert au Orange, puis au Rouge lorsqu'un véhicule arrive sur la route secondaire. Il attend que le feu de la route secondaire soit Rouge et indique que les voitures de la route secondaires sont bien passées.

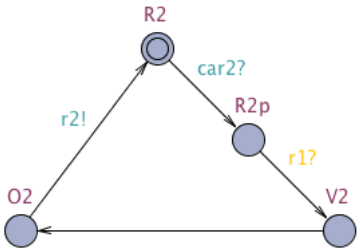

FIGURE 10 – Automate du feu de la route secondaire

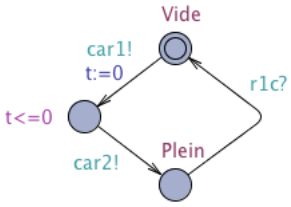

FIGURE 11 – Automate de l'arrivée des véhicules sur la route secondaire (capteur)

Par défaut Rouge, le feu de la route secondaire passe au Vert si une voiture est détectée et que les feux de la route principale sont Rouges.

L'automate qui conditionne l'arrivée des voitures sur la route secondaire est réprésenté par deux états principaux : route vide et route plein (présence d'une voiture au moins). Le signal *r1c* permet d'assurer que la voiture détectée sur la route secondaire soit bien passée pendant que le feu était Vert.

3.2 Question 10

A notre système de feux sur un carrefour en T, nous ajoutons ici des contraintes de temps notamment sur le temps pendant lequel chaque feu doit rester Vert.

- Le feu de la route secondaire reste Vert pendant 30 secondes
- Dans un cycle, le feu de la route principale doit être Vert pendant au moins 30 secondes
- Chaque feu reste orange pendant 5 secondes

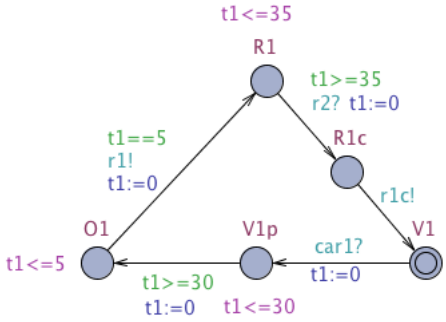

FIGURE 12 – Automate du feu de la route principale

Les signaux *r1* et *r2* permettent d'obliger qu'un des feux soit rouge à n'importe quel moment.

Le feu de la route principale attend qu'une voiture arrive sur la route secondaire pour passer à l'Orange. Attention, le feu respecte aussi le fait qu'il doit rester au Vert eu moins 30 secondes même si une voiture est détectée avant.

Avant de repasser au Vert, le feu de la route principale attend que celui de la route secondaire soit passé au Rouge.

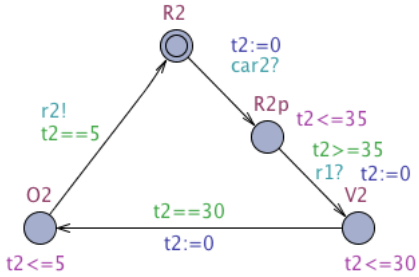

FIGURE 13 – Automate du feu de la route secondaire

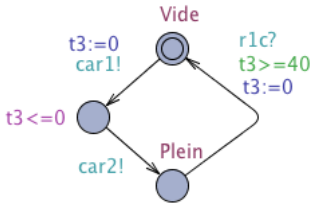

FIGURE 14 – Automate de l'arrivée des véhicules sur la route secondaire (capteur)

Pour passer au Vert, le feu de la route secondaire doit l'arrivée d'une voiture mais aussi que le feu de la route principale soit Rouge.

Les signaux *car1* et *car2* permettent d'indiquer au feu de la route principale et au feu de la route secondaire qu'une voiture est arrivée sur la route secondaire. Le feu de la route principale notifie aussi l'automate d'arrivée des véhicules pour indiquer que la voiture est passée.

3.3 Question 11

3.3.1 Validations question 9

- Toujours au moins un feu rouge
  `A[](Major.R1 or Major.R1c or Minor.R2 or Minor.R2p)`
- Lorsque le feu de la route secondaire passe au vert, il doit y avoir au moins une voiture qui passe sur la route secondaire.
  `E<>(Car.Vide and Minor.V2)`
- Les deux feux atteignent leur état le plus éloigné, ils ne sont pas bloqués
  `E<>(Major.R1 and Minor.O2)`

- Pas de deadlock
  `E<> deadlock`

3.3.2 Validations question 10

- Toujours au moins un feu rouge
  `A[](Major.R1 or Major.R1c or Minor.R2 or Minor.R2p)`
- Lorsque le feu de la route secondaire passe au vert, il doit y avoir au moins une voiture qui passe sur la route secondaire.
  `E<>(Car.Vide and Minor.V2)`
- Les deux feux atteignent leur état le plus éloigné, ils ne sont pas bloqués par le temps
  `E<>(Major.R1 and Minor.O2)`
- Pas de deadlock
  `E<> deadlock`

4 Implémentation

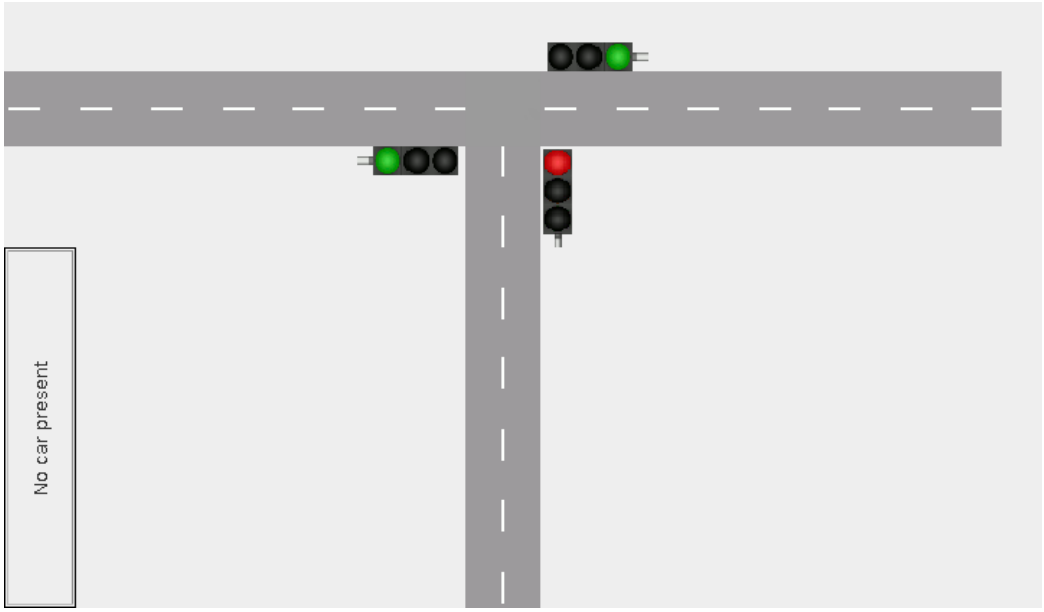

FIGURE 15 – Interface du carrefour en T

Notre implémentation simule un carrefour en T présenté par les figures 12 et 13, synchronisé par la figure 14.

On a utilisé un Thread par feu (class Road) pour l'autonomie et l'indépendance. On a une classe (Tjonction) qui sert de contrôleur pour synchroniser les feux.

Chaque route a deux procédures pour passer au rouge ou pour passer au vert. Le contrôleur permet de faire passer le signal au principal que la route secondaire à besoin de passer au vert, mais aussi permettre de temporiser les feux pour avoir une équité entre les deux routes.

Dans notre interface, nous simulons la présence de voiture par le bouton à deux états. Un état pour indiquer quand une voiture est présent sur la route secondaire, un deuxième état pour indiquer qu'il n'y a aucun voiture.