Projet NuSMV
Feux de circulation

TRAN Thanh Long
GAUTHIER Thomas

M1 IFI
2017-2018

I.Introduction

Le but de ce projet est de réaliser une simulation de feux de circulation grâce à NuSMV. Ces feux seront composés d'un feu pour les voitures et d'un autre pour les piétons.

Deux types de feux seront implémentés :

- Un feu simple : Le feu des voitures sera initialement rouge, puis passera au vert et ensuite au jaune, et ainsi de suite. Le feu piéton sera rouge et passera au vert uniquement si le feu des voitures est rouge et la voie sécurisée.
- Un feu avec un signal : Le feu des voitures reste au vert tant qu'un piéton n'a pas appuyé sur un bouton pour traverser. Lorsque cet événement se produit, le feu des voitures passera au jaune puis au rouge et le feu piéton au vert.

Pour vérifier que nos systèmes sont correctement fonctionnels, nous mettrons en place des propriétés LTL et CTL. Nous utiliserons des modèles volontairement incorrects, dans lesquels nous ne changerons qu'un seul paramètre par rapport au véritable modèle, afin d'être certains de leur validité. Les traces de ces tests se trouvent dans la partie IV.

II.Feu simple

1. Fonctionnement

Initialement, les deux feux sont rouges. Le feu des voitures fonctionne en suivant un cycle :

- Il reste rouge pendant 5 ticks,
- Il passe au vert pendant 10 ticks,
- Il reste ensuite jaune pendant 2 ticks,
- Et ainsi de suite.

Le programme est composé de 3 modules :

- *carLightState*, qui simule le fonctionnement du feu voiture,
- *pedestrianLight*, qui simule le comportement du feu piéton,
- *main*, qui permet de mettre les deux précédents modules en relation.

Pour simuler la réalité, nous avons fait en sorte que pendant le tick où le feu voiture passe au rouge, le feu piéton reste lui aussi rouge. Il passera au vert au tick suivant, et on le repasse en rouge au tick précédent le passage au vert du feu voiture. En résumé, si le feu voiture ne reste rouge que pendant 5 ticks, le feu piéton ne sera vert que pendant 3 ticks. Le feu piéton sait s'il doit changer de couleur en fonction du booléen *safeCross* du module *carLightState*.

2. Propriétés

a. Propriétés de vivacité

Propriété CTL :

AF (pedestrianLightState.pedestrianLight = red & EX (pedestrianLightState.pedestrianLight = green)) & AF (carLightState.carLight = red & EX (carLightState.carLight = green));

→ Dans tous les cycles, le feu piéton sera rouge à un moment et finira par passer au vert. Même chose pour le feu voiture.

Propriété LTL :

F pedestrianLightState.pedestrianLight = green & F carLightState.carLight = green;

→ Le feu voiture et le feu piétons finiront forcément par passer au vert.

Preuve : (*fichiers simple_liveness.smv*)

Pour montrer que ces propriétés étaient correctes, nous avons mis en place un modèle où l'on ne signale jamais au feu piéton que la voie est sécurisée. On trouve bien que ces deux propriétés y sont fausses étant donné que le feu piéton ne passe jamais au vert.

b. Propriétés de sécurité

Propriété CTL :

!EG (pedestrianLightState.pedestrianLight = green & carLightState.carLight != red);

→ Il n'existe pas d'état dans lequel le feu piéton est vert et le feu voiture d'une couleur différente du rouge.

Propriété LTL :

G (carLightState.carLight = yellow | carLightState.carLight = green -> pedestrianLightState.pedestrianLight = red);

→ Si le feu piéton est vert, alors le feu voiture est forcément rouge.

Preuve : (*fichier simple_safety.smv)*

Nous avons mis en place un modèle dans lequel le feu piéton passe aléatoirement de vert à rouge. Bien évidemment, la propriété y est fausse puisqu'aucune aucun contrôle n'est effectué sur ce changement.

III. Feu avec signal

1. Fonctionnement

Ce feu fonctionne sur le même principe que le précédent, cependant pour celui-ci nous avons simulé le fait qu'un piéton puisse appuyer sur un bouton afin que le feu piéton puisse passer au vert.

Tant qu'aucun signal n'a été émis, le feu voiture reste vert (tick 13). Une fois qu'un piéton appuie sur le bouton, le feu reste vert encore 1 tick puis passe au jaune pour suivre le même cycle défini dans le système des feux simples. Le feu voiture ne peut pas rester vert moins de 10 ticks. Si un piéton appuie sur le bouton en cours de cycle, alors ce signal est mémorisé et le feu passera automatiquement à l'orange à la suite des 10 prochains ticks verts du feu voiture. Si le cycle est terminé, donc que le feu est "en attente" d'un signal, et qu'un piéton appuie sur le bouton, il devra attendre **4 ticks complets** avant que le feu piéton ne passe au vert (1 tick où le feu voiture reste vert, 2 ticks où le feu voiture passe au jaune, 1 pendant lequel les deux feux sont au rouge, puis seulement le feu piéton passe au vert). Initialement, les deux feux sont rouges.

2. Propriétés

a. Propriété de vivacité

Propriété CTL :

SPEC EF (carLightState.tick = 13 & (carLightState.signal = TRUE | carLightState.memorisedSignal = TRUE) & AX(AX(AX (AX (pedestrianLightState.pedestrianLight = red & AX (pedestrianLightState.pedestrianLight = green) & EF (carLightState.carLight = green & pedestrianLightState.pedestrianLight = red))))));

→ Si le feu est en attente d'un signal, et qu'un a déjà été mémorisé ou qu'il en reçoit un, alors au bout de 4 ticks le feu piéton va passer du rouge au vert, et plus tard le feu piéton sera à nouveau au rouge tandis que le feu voiture sera vert.

Propriété LTL :

LTLSPEC F(carLightState.signal = TRUE -> (F carLightState.carLight = red U carLightState.carLight = green));

→ Si un signal est reçu, le feu voiture passera un moment au rouge jusqu'à ce qu'il repasse au vert.

Preuve : (*fichier extended_liveness_1.smv et extended_liveness_2.smv)*

Pour prouver la propriété CTL, nous avons prouvé qu'elle était fausse dans un modèle où il faut un tick de plus avant que le feu piéton ne passe au vert.

Pour prouver la propriété LTL, nous avons prouvé qu'elle était fausse dans un modèle où le feu voiture ne passe jamais au vert.

b. Propriété de sécurité

Etant donné que les conditions de sécurité restent inchangées par rapport au modèle de feu simple, il n'est pas nécessaire de modifier les propriétés de sécurité pour ce modèle plus complexe.

Pour vérifier qu'elles étaient toujours valables, nous avons comme précédemment fait en sorte que le feu de piéton change aléatoirement de couleur, et nous avons obtenu les mêmes résultats.

IV. Annexes

1. Propriétés feu simple

a. simple.smv

```
-- specification (AF (pedestrianLightState.pedestrianLight = red & EX pedestrianLightState.pedestrianLight = green) & AF (carLightState.carLight = red & EX carLightState.carLight = green))  is true

-- specification !(EF (pedestrianLightState.pedestrianLight = green & carLightState.carLight != red))  is true
-- specification ( F pedestrianLightState.pedestrianLight = green &  F carLightState.carLight = green)  is true
-- specification  G (pedestrianLightState.pedestrianLight = green -> carLightState.carLight = red)  is true
```

b. simple_liveness.smv

- CTL :

```
-- specification (AF (pedestrianLightState.pedestrianLight = red & EX pedestrianLightState.pedestrianLight = green) & AF (carLightState.carLight = red & EX carLightState.carLight = green))  is fals
e
```

- LTL :

```
-- specification ( F pedestrianLightState.pedestrianLight = green &  F carLightState.carLight = green)  is false
```

c. simple_safety.smv

- CTL :

```
-- specification !(EF (pedestrianLightState.pedestrianLight = green & carLightState.carLight != red))  is false
```

- LTL :

```
-- specification  G (pedestrianLightState.pedestrianLight = green -> carLightState.carLight = red)  is false
```

2. Propriétés feu avec signal

a. extended.smv

```
-- specification EF ((carLightState.tick = 13 & (carLightState.signal = TRUE | carLightState.memorisedSignal = TRUE)) & AX (AX (AX (AX ((pedestrianLightState.pedestrianLight = red & AX pedestrianLi
ghtState.pedestrianLight = green) & EF (carLightState.carLight = green & pedestrianLightState.pedestrianLight = red))))))  is true
-- specification !(EF (pedestrianLightState.pedestrianLight = green & carLightState.carLight != red))  is true
-- specification  F (carLightState.signal = TRUE -> (( F carLightState.carLight = red) U carLightState.carLight = green))  is true
-- specification  G ((carLightState.carLight = yellow | carLightState.carLight = green) -> pedestrianLightState.pedestrianLight = red)  is true
```

b. extended_liveness

- CTL : (extended_liveness_2.smv)

```
-- specification EF ((carLightState.tick = 13 & (carLightState.signal = TRUE | carLightState.memorisedSignal = TRUE)) & AX (AX (AX (AX ((pedestrianLightState.pedestrianLight = red & AX pedestrianLi
ghtState.pedestrianLight = green) & EF (carLightState.carLight = green & pedestrianLightState.pedestrianLight = red))))))  is false
```

- LTL : (extended_liveness_2.smv)

```
-- specification  F (carLightState.signal = TRUE -> (( F carLightState.carLight = red) U carLightState.carLight = green))  is false
```

c. extended_safety.smv

- CTL :

```
-- specification !(EF (pedestrianLightState.pedestrianLight = green & carLightState.carLight != red))  is false
```

- LTL :

```
-- specification  G ((carLightState.carLight = yellow | carLightState.carLight = green) -> pedestrianLightState.pedestrianLight = red)  is false
```

V. Lien Github

https://github.com/ThomasGauthierr/NuSMV