Trafic Light

G.Fernandes Junior Spinelli Aurélien

25 mai 2018

Table des matières

Première partie

Tests Trafic Light Simple

1 Parties Aurélien Spinelli

Deuxième partie

Tests Trafic Light Extension

2 Tests des anciennes propriétés du Trafic Light Simple sur le model Trafic Light Extension

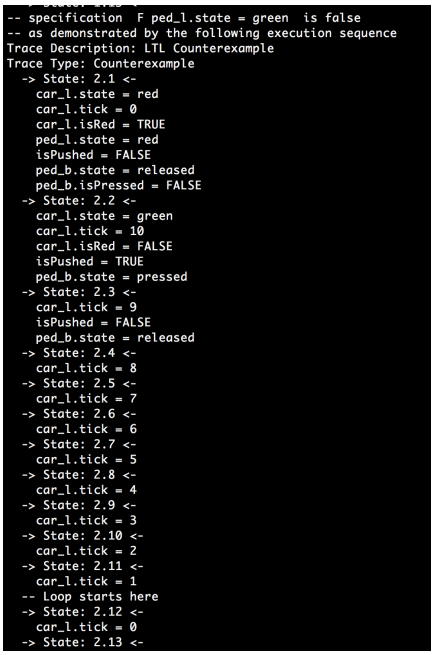
2.1 Test Liveness des piétons

Tout d'abord, le fichier test se nomme "ExtTest1.smv". Le fichier contient le même model que "traffic-light-ex.smv" et les deux propriétés (CTL et LTL) de Liveness pour les piétons, concernant le model simple du trafic light. Nous allons ici tester si la propriété de Liveness (P2) du model Trafic Light Simple est toujours vérifiée dans le model Trafic Light Extension. Étant donné que dans le deuxième model, le piéton traverse uniquement lorsqu'il en fait la demande, en appuyant sur le bouton, la propriété du premier model ne devrait pas être vérifiée dans le second model. Notre propriété CTL est la suivante : SPEC AF (ped_l.state = green). Cette propriété stipule que l'état du feu de piéton à vert est atteignable par tous les futurs chemins. Bien évidemment, étant donné que le piéton n'appuie pas à chaque fois, il y aura forcément un contre exemple :

```
-- specification AF ped_l.state = green  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    car_l.state = red
    car_l.tick = 0
    car_l.isRed = TRUE
    ped_l.state = red
    isPushed = FALSE
    ped_b.state = released
    ped_b.isPressed = FALSE
  -> State: 1.2 <-
    car_l.state = green
    car_l.tick = 10
    car_l.isRed = FALSE
    isPushed = TRUE
    ped_b.state = pressed
  -> State: 1.3 <-
    car_l.tick = 9
    isPushed = FALSE
    ped_b.state = released
  -> State: 1.4 <-
    car_l.tick = 8
  -> State: 1.5 <-
    car_l.tick = 7
  -> State: 1.6 <-
    car_l.tick = 6
  -> State: 1.7 <-
    car_l.tick = 5
  -> State: 1.8 <-
    car_l.tick = 4
  -> State: 1.9 <-
    car_l.tick = 3
  -> State: 1.10 <-
    car_l.tick = 2
  -> State: 1.11 <-
    car_l.tick = 1
  -- Loop starts here
  -> State: 1.12 <-
    car_l.tick = 0
  -> State: 1.13 <-
```

FIGURE 1 – Test numéro un

Notre specification LTL pour la deuxième propriété du premier model est la suivante : LTLSPEC F (ped_l.state = green) ; Comme convenu, elle n'est pas vérifiée pour les mêmes raisons :



```
-- specification  F ped_l.state = green  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -> State: 2.1 <-
    car_l.state = red
    car_l.tick = 0
    car_l.isRed = TRUE
    ped_l.state = red
    isPushed = FALSE
    ped_b.state = released
    ped_b.isPressed = FALSE
  -> State: 2.2 <-
    car_l.state = green
    car_l.tick = 10
    car_l.isRed = FALSE
    isPushed = TRUE
    ped_b.state = pressed
  -> State: 2.3 <-
    car_l.tick = 9
    isPushed = FALSE
    ped_b.state = released
  -> State: 2.4 <-
    car_l.tick = 8
  -> State: 2.5 <-
    car_l.tick = 7
  -> State: 2.6 <-
    car_l.tick = 6
  -> State: 2.7 <-
    car_l.tick = 5
  -> State: 2.8 <-
    car_l.tick = 4
  -> State: 2.9 <-
    car_l.tick = 3
  -> State: 2.10 <-
    car_l.tick = 2
  -> State: 2.11 <-
    car_l.tick = 1
  -- Loop starts here
  -> State: 2.12 <-
    car_l.tick = 0
  -> State: 2.13 <-
```

FIGURE 2 – Test numéro deux

2.2 Test Liveness des voitures

Nous allons entreprendre les mêmes tests pour les voitures dans le fichier "ExtTest2.smv". Dans ce fichier, le model sera aussi le même que celui du fichier "traffic-light-ex.smv" concernant le model Trafic Light Extension. Nous y avons ajouté les anciennes propriétés (pour P2) du model simple qui sont :

— en CTL : SPEC AF (car_l.state = green)

— en LTL : LTLSPEC F (car_l.state = green) ;

Dans ce cas, ces deux propriétés devraient être vérifiées car le feu de voiture reste au vert tant que le piéton n'appuie pas sur le bouton pour demander à passer. Nous aurons donc, pour chaque chemin, un état ou le feu de voiture sera vert :

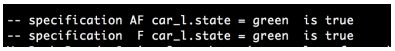


FIGURE 3 – Test numéro deux

3 Test des propriétés du Trafic Light Extension

Nous avons, pour la propriété P2, définit quatre spécifications, c'est à dire deux propriétés LTL et deux propriétés CTL. Les deux premières spécifications LTL et CTL seront vues dans la partie "Test de la demande de passage du piéton" puis les deux autres spécifications seront vues dans la partie nommée "Test du feu de voiture".

3.1 Test de la demande de passage du piéton

Dans le fichier nommé "ExtTestP2Pieton.smv", nous créons un model simpliste où le bouton du feu de piéton est pressé volontairement pour vérifier si son feu passe au vert une fois la demande effectuée. Cependant, ici, nous avons fait en sorte que le feu de piéton reste rouge quoi qu'il advienne. Nos propriétés CTL et LTL stipulent le fait qu'à chaque fois que la demande de passage est effectuée cela implique que le feu de piéton sera vert à chaque fois suite à la demande de passage. Vous l'aurez comprit, en forçant le feu de piéton à rouge, les propriétés ne devraient pas être vérifiées. Ces propriétés sont d'ailleurs celles de Liveness (P2) pour le model Trafic Light Extension. Voici donc les propriétés :

— en CTL : SPEC AG (ped_b.isPressed → AF ped_l.state = green) ;

— en LTL : LTLSPEC G (ped_b.isPressed → F ped_l.state = green) ;

Nous avons un contre exemple nous démontrant que la propriété, vérifiant que le feu passe au vert après une demande de passage, a correctement été écrite car, ici, le test échoue en nous disant que le feu n'est jamais passé au vert après une demande de piéton. Voici, le premier résultat concernant la propriété CTL :

```
-- specification AG (ped_b.isPressed -> AF ped_l.state = green)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    car_l.state = red
    car_l.tick = 0
    car_l.isRed = FALSE
    ped_l.state = red
    ped_b.state = released
    ped_b.isPressed = FALSE
  -- Loop starts here
  -> State: 1.2 <-
    ped_b.state = pressed
    ped_b.isPressed = TRUE
  -> State: 1.3 <-
```

FIGURE 4 – Test propriété CTL

Respectivement, voici le résultat du contre exemple pour la propriété LTL :

```
-- specification  G (ped_b.isPressed ->  F ped_l.state = green)  is false
-- as demonstrated by the following execution sequence
Trace Description: LTL Counterexample
Trace Type: Counterexample
  -> State: 2.1 <-
    car_l.state = red
    car_l.tick = 0
    car_l.isRed = FALSE
    ped_l.state = red
    ped_b.state = released
    ped_b.isPressed = FALSE
  -- Loop starts here
  -> State: 2.2 <-
    ped_b.state = pressed
    ped_b.isPressed = TRUE
  -- Loop starts here
  -> State: 2.3 <-
  -> State: 2.4 <-
```

FIGURE 5 – Test propriété LTL

3.2 Test du feu de voiture

Nous allons ici tester les deux dernières spécifications pour la propriété P2 du Trafic Light Extension. Les deux spécifications sont les suivantes :

— en CTL : SPEC AG ((ped_b.isPressed = FALSE & AG ped_b.isPressed = FALSE) → AG car_l.state = green) ;

— en LTL : LTLSPEC G ((ped_b.isPressed = FALSE & G ped_b.isPressed = FALSE) → F (G car_l.state = green)) ;

La première specification vérifie que si aucun piéton n'appuie sur le bouton de passage maintenant ou dans le futur cela impliquerait que le feu de voiture reste vert maintenant et dans le futur globalement et pour tous les chemins possibles. La deuxième spécification vérifie que si aucun piéton n'appuie sur le bouton de passage maintenant ou dans le futur cela impliquerait que, finalement, dans le futur, le feu de voiture sera vert. Dans le fichier nommé "ExtTestP2Voiture.smv", nous faisons en sorte de mettre le feu de voiture initialement à vert, puis dans le prochain état nous le passons à rouge. Nous supposons que le piéton n'appuie jamais sur le bouton pour demander à passer. Nos spécifications vérifient que si personne n'appuie (à présent et dans le futur) le feu reste vert. En ayant mis le feu de voiture à rouge volontairement, les deux propriétés devraient échouer car le feu devait rester vert. Le contre exemple de la spécification CTL est la suivante :

```
-- specification AG ((ped_b.isPressed = FALSE & AG ped_b.isPressed = FALSE) -> AG car_l.state = green)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    isPushed = FALSE
    car_l.state = green
    car_l.tick = 0
    car_l.isRed = FALSE
    ped_l.state = red
    ped_b.state = released
    ped_b.isPressed = FALSE
  -> State: 1.2 <-
    car_l.state = red
    car_l.isRed = TRUE
    ped_l.state = green
```

FIGURE 6 – Test propriété CTL

Respectivement, voici le contre exemple de la spécification LTL :

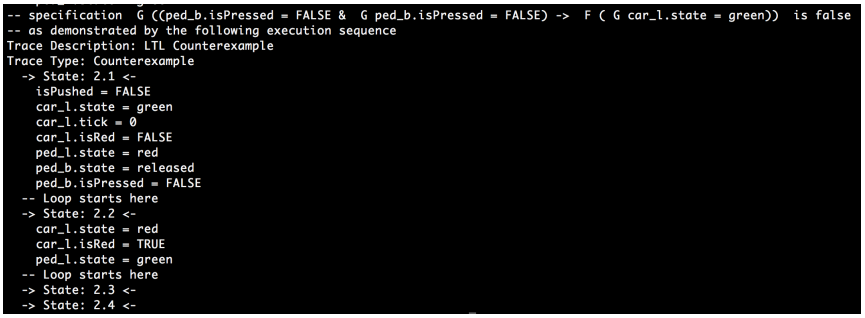


FIGURE 7 – Test propriété LTL