UNIVERSITÉ D'ÉVRY – VAL D'ESSONNE
Département d'informatique

Processus concurrents et temporisés

2nd année de Master informatique
parcours CILS

Soutenu par

Valentin COTTE, Vianney TOUCHARD

le 18 janvier 2019

Projet M2 : Modélisation et analyse d'automates temporisés avec UPPAAL et de réseaux de Petri temporels avec TINA

Encadrant
Hanna KLAUDEL

Année universitaire 2018-2019

Table des matières

1 Introduction

L'objectif de ce projet est de modéliser un problème donné sous forme d'automates temporisés et réseaux de Petri, puis d'analyser et modifier ces modèles afin de répondre à 3 questions.

On rappelle le système à modéliser :

Alice se rend à l'université en train (cela prend entre 20 et 30 min) ou en bus, ce qui est de 15 min plus rapide, mais elle doit marcher pendant 5 min pour rejoindre l'arrêt de bus.

Bob vient à l'université en voiture (entre 30 et 40 min) ou en train (50 min). Il a besoin de 5 à 10 min pour garer sa voiture et autant pour passer de l'arrêt de tram à l'université.

Claire habite à proximité et vient à l'université à pied (entre 15 et 20 min) ou à vélo (5 min), mais dans ce cas, elle doit laisser son vélo dans un parking distant ce qui prend 5 min à 10 min pour rejoindre l'université.

On ajoute à ce système les contraintes suivantes :
Ce matin, Alice est partie de chez elle entre 7h50 et 8h. Bob est arrivé à l'université entre 8h10 et 8h20. De plus, Claire est partie 10 à 20 minutes après Alice et Bob.

Et enfin, voici les 3 questions auxquelles il faut répondre :
- 1) Les contraintes temporelles figurant dans ce scénario sont-elles consistantes ?
- 2) Est-il possible qu'Alice ait pris le bus et Claire ait utilisé le vélo ?
- 3) Dans quelle plage horaire Bob a pu partir ?

2 Automates Temporisés (UPPAAL)

La première étape est de convenir de la modélisation du système. Dans l'annexe A, les automates temporisés permettent une modélisation assez proche de l'énoncé.
Nous commençons par définir une horloge globale qui représente l'heure (en nombre de minutes écoulées depuis minuit). Chaque personnage est représenté par un automate et les places sont les différents endroits où peuvent être Alice, Bob et Claire. Les contraintes énoncées sont directement ajoutées sur les arcs et les places comme des invariants et des conditions.

2.1 Question 1

De par la modélisation choisie, les contraintes sont satisfiables si une exécution amène A., B. et C. à l'université.
Pour obtenir une réponse du vérifieur, on lui soumet le prédicat négatif exprimant « A., B. ou C. n'est pas à l'université ». Le vérifieur infirme et soumet un contre-exemple : les contraintes peuvent donc être satisfaites, elles sont consistantes.

```
E<>(Bob.universite and Alice.universite and Claire.universite)
A[] not(Bob.universite and Alice.universite and Claire.universite)
```

2.2 Question 2

Nous avons encore une fois recours à la négation de la proposition pour que UPPAAL fournisse un contre-exemple.
Le contre-exemple s'exprime « (Alice a pris le bus => Claire n'a pas utilisé le vélo) et (Claire a utilisé le vélo => Alice n'a pas pris le bus) ». Plus formellement, nous avons la proposition équivalente : non(il existe un chemin tel que A.b et C.v) = pour tout chemin, Non(A.b) ou non(C.v)
Pour exprimer correctement cette contrainte, il faut garder à l'esprit qu'on ne peut pas se contenter de vérifier l'existence d'un état où Alice.bus et Claire.velo sont actifs, car ces états pourraient être actifs à des moments différents et que la condition soit tout de même validée. Ainsi, nous avons recours à des variables drapeaux qui restent activés.
Cette précaution prise, le prouveur nous donne un contre-exemple où les contraintes sont respectées et où Alice utilise le bus et Claire le vélo.

```
E<>(Bob.universite and Claire.universite and Alice.universite) and Alice.bus_flag==true and Claire.velo_flag==true
A[] not((Bob.universite and Claire.universite and Alice.universite) and Alice.bus_flag==true and Claire.velo_flag==true)
```

2.3 Question 3

Pour cette question, nous nous trouvons obligés de procéder par tâtonnements en cherchant les valeurs (supérieure et inférieure) de la plage horaire cherchée.
Cette plage horaire correspond à celle que nous avons trouvée à la main : Bob doit partir entre 7h10 et 7h45 pour que l'énoncé soit respecté.
Cela fait, on valide les horaires vérifiant la formule «Bob est parti en dehors de la plage horaire => les contraintes de l'énoncé ne seront pas satisfaites»

```
((Bob.tram or Bob.voiture) and Bob.x==0 and h<431 )-->(Bob.parking or Bob.marche) and Bob.x==10 and h<490
((Bob.tram or Bob.voiture) and Bob.x==0 and h<430 )-->(Bob.parking or Bob.marche) and Bob.x==10 and h<490
((Bob.tram or Bob.voiture) and Bob.x==0 and h>464 )-->not Bob.universite and h>500
((Bob.tram or Bob.voiture) and Bob.x==0 and h>465 )-->not Bob.universite and h>500
((Bob.tram or Bob.voiture) and Bob.x==0 and (h<430 or h>465) )-->(not Bob.universite and h>500) or ((Bob.parking or Bob.marche) and Bob.x==10 and h<490)
```

3 Réseaux de Petri (TINA)

Pour cette seconde partie, nous avons une modélisation du système tout aussi proche de l'énoncé. Cependant, contrairement aux automates temporisés, les réseaux de Petri en annexe B sont indépendants des contraintes de l'énoncé. C'est à l'aide de réseaux «observateurs» que nous allons faire respecter les contraintes dans des requêtes LTL.
Les différents agents sont donc Alice, Bob et Claire qui vont parcourir chacun un réseau différent simulant leur trajet jusqu'à l'université, ainsi que deux observateurs : un pour renseigner l'heure actuelle et un autre pour chronométrer depuis combien de temps Bob et Alice sont partis de chez eux (ce dernier est uniquement utile pour la contrainte du départ de Claire).

3.1 Question 1

Tout comme l'on fait nos prédécesseurs dans l'article [BV06], nous avons fait calculé par TINA un graphe de classe du réseau de l'annexe B afin de vérifier la satisfiabilité de formules LTL.
Pour montrer la consistance des contraintes, il faut prouver l'existence d'une exécution dans laquelle Alice, Bob et Claire vont à l'université tout en respectant toutes les contraintes de l'énoncé.
Il s'agit de vérifier la satisfiabilité de la formule signifiant qu'il n'existe pas de telle exécution :

$\neg(\ \Diamond\ (depart\_alice\ \wedge\ 7h50\_8h00)$
$\wedge\ \Diamond\ (arrivee\_bob\ \wedge\ 8h10\_8h20)$
$\wedge\ \Diamond\ (depart\_claire\ \wedge\ 10\_20)$
$\wedge\ \Diamond\ (univ\_alice\ \wedge\ univ\_bob\ \wedge\ univ\_claire))$

Avec le model checker *selt*, nous trouvons que cette formule est fausse. Pour le prouver, il nous renvoie une trace contre-exemple, soit une trace qui respecte toutes les contraintes et donc prouve qu'elles sont consistantes.

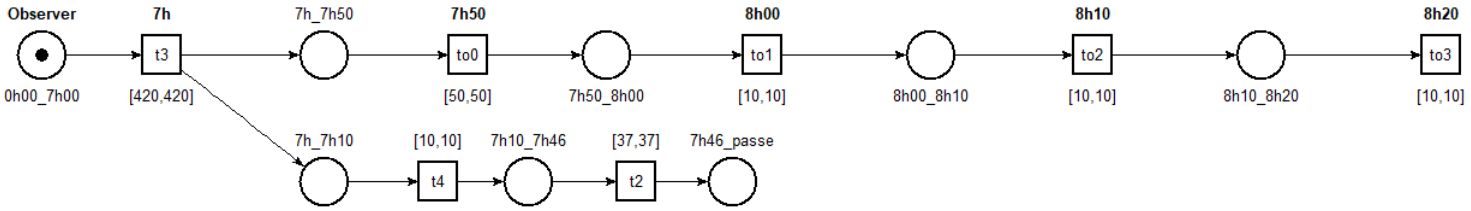
3.2 Question 2

Ici, contrairement à la situation pour les automates temporisés, pas besoin de variables : le chemin complet est considéré. Nous rajoutons aux contraintes précédentes la proposition voulue, et un contre-exemple est trouvé.

$\neg(\ \Diamond\ (univ\_bob\ \wedge\ univ\_alice\ \wedge\ univ\_claire)$
$\wedge\ \Diamond\ (depart\_alice\ \wedge\ 7h50\_8h00)$
$\wedge\ \Diamond\ (arrivee\_bob\ \wedge\ 8h10\_8h20)$
$\wedge\ \Diamond\ (depart\_claire\ \wedge\ 10\_20)$
$\wedge\ \Diamond\ alice\_bus\ \wedge\ \Diamond\ claire\_velo)$

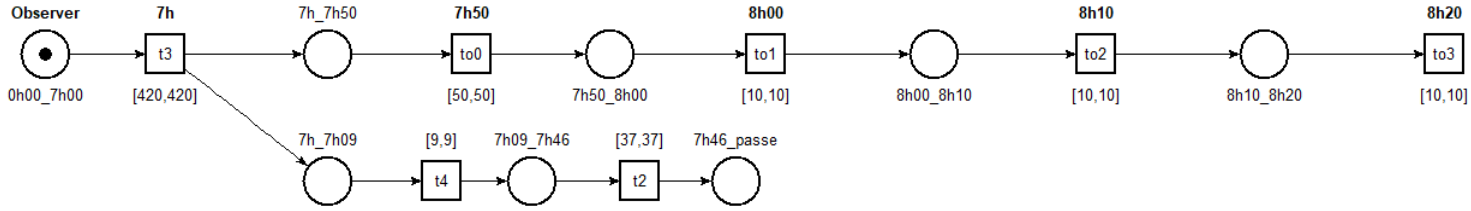
3.3 Question 3

Pour cette question, on rajoute une branche à l'observateur qui nous permet de vérifier la présence dans une plage précise. Encore une fois, nous approchons les valeurs par le haut et par le bas. Une fois la plage trouvée, on peut vérifier qu'un départ en dehors de celle-ci ne permet pas aux contraintes d'être satisfaites.
Pour cela, on a donc fabriqué 3 réseaux identiques à l'annexe B mais avec un observateur différent. Avec chacun d'eux, nous avons testé une série de formules LTL pour trouver la plage horaire du départ de Bob.



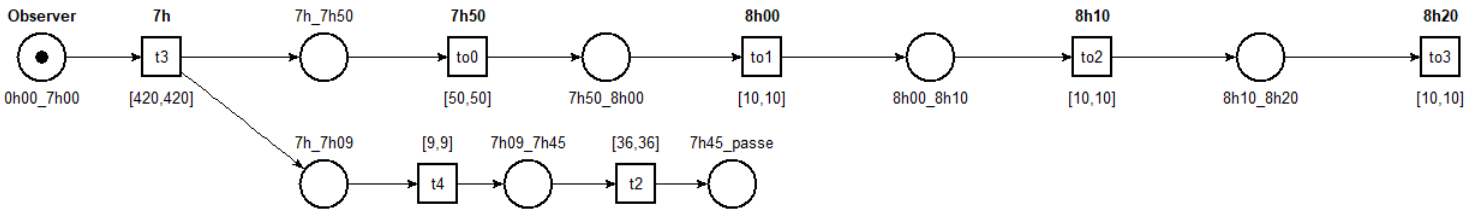
$\neg(\Diamond\,(univ\_bob \wedge univ\_alice \wedge univ\_claire)$
$\wedge\ \Diamond\,(depart\_alice \wedge 7h50\_8h00)$
$\wedge\ \Diamond\,(arrivee\_bob \wedge 8h10\_8h20)$
$\wedge\ \Diamond\,(depart\_claire \wedge 10\_20)$
$\wedge\ \Diamond\,(depart\_bob \wedge (0h00\_7h00 \vee 7h\_7h10)))$

La formule renvoie un contre-exemple, ce qui nous donne un départ déjà possible à 7h10.



$\neg(\Diamond\,(univ\_bob \wedge univ\_alice \wedge univ\_claire)$
$\wedge\ \Diamond\,(depart\_alice \wedge 7h50\_8h00)$
$\wedge\ \Diamond\,(arrivee\_bob \wedge 8h10\_8h20)$
$\wedge\ \Diamond\,(depart\_claire \wedge 10\_20)$
$\wedge\ \Diamond\,(depart\_bob \wedge (7h45\_passe)))$

La formule renvoie un contre-exemple, ce qui nous donne un départ encore possible à 7h45.



$\neg(\Diamond\,(univ\_bob \wedge univ\_alice \wedge univ\_claire)$
$\wedge\ \Diamond\,(depart\_alice \wedge 7h50\_8h00)$
$\wedge\ \Diamond\,(arrivee\_bob \wedge 8h10\_8h20)$
$\wedge\ \Diamond\,(depart\_claire \wedge 10\_20)$
$\wedge\ \Diamond\,(depart\_bob \wedge (0h00\_7h00 \vee 7h\_7h09)))$

La proposition est vraie, ce qui donne un départ impossible avant 7h10.

$\neg(\ \Diamond\ (univ\_bob\ \wedge\ univ\_alice\ \wedge\ univ\_claire)$
$\wedge\ \Diamond\ (depart\_alice\ \wedge\ 7h50\_8h00)$
$\wedge\ \Diamond\ (arrivee\_bob\ \wedge\ 8h10\_8h20)$
$\wedge\ \Diamond\ (depart\_claire\ \wedge\ 10\_20)$
$\wedge\ \Diamond\ (depart\_bob\ \wedge\ (7h46\_passe)))$

La proposition est vraie, ce qui donne un départ impossible après 7h46.

On peut donc confirmer l'impossibilité de partir hors de cette tranche horaire avec :
$\neg(\ \Diamond\ (univ\_bob\ \wedge\ univ\_alice\ \wedge\ univ\_claire)$
$\wedge\ \Diamond\ (depart\_alice\ \wedge\ 7h50\_8h00)$
$\wedge\ \Diamond\ (arrivee\_bob\ \wedge\ 8h10\_8h20)$
$\wedge\ \Diamond\ (depart\_claire\ \wedge\ 10\_20)$
$\wedge\ \Diamond\ (depart\_bob\ \wedge\ \neg 7h09\_7h46))$

La formule est évaluée à vrai, donc la plage horaire est de 7h10 à 7h45.
Une façon plus élégante aurait été de définir cette plage comme un intervalle ouvert plutôt qu'en rajoutant une minute.

Références

[BV06] Bernard Berthomieu and François Vernadat. Réseaux de petri temporels : méthodes d'analyse et vérification avec tina. *Traité IC2*, page 101, 2006.

A Annexe

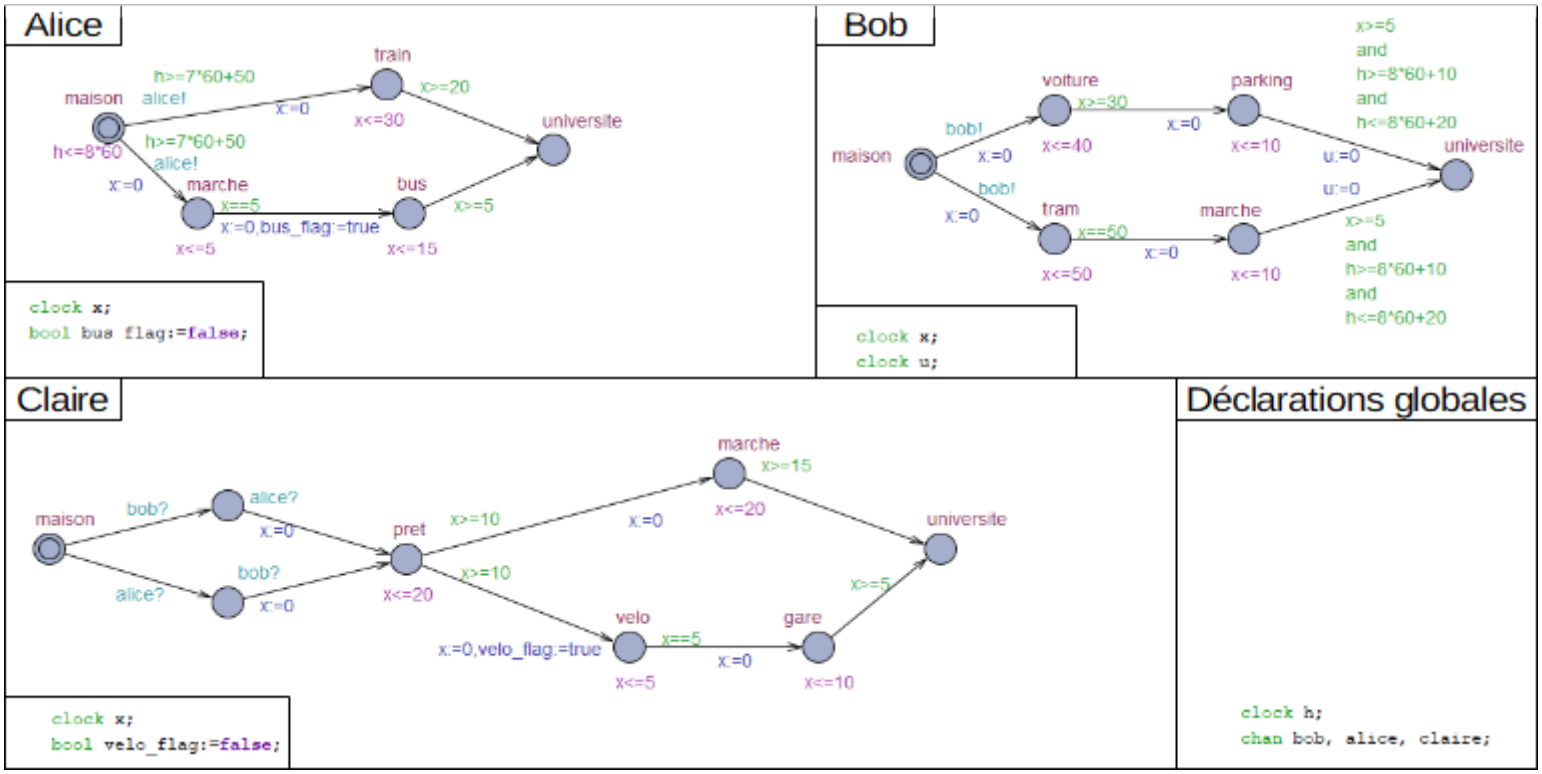


FIGURE 1 – Modélisation du système en automates temporisés avec les contraintes

B Annexe

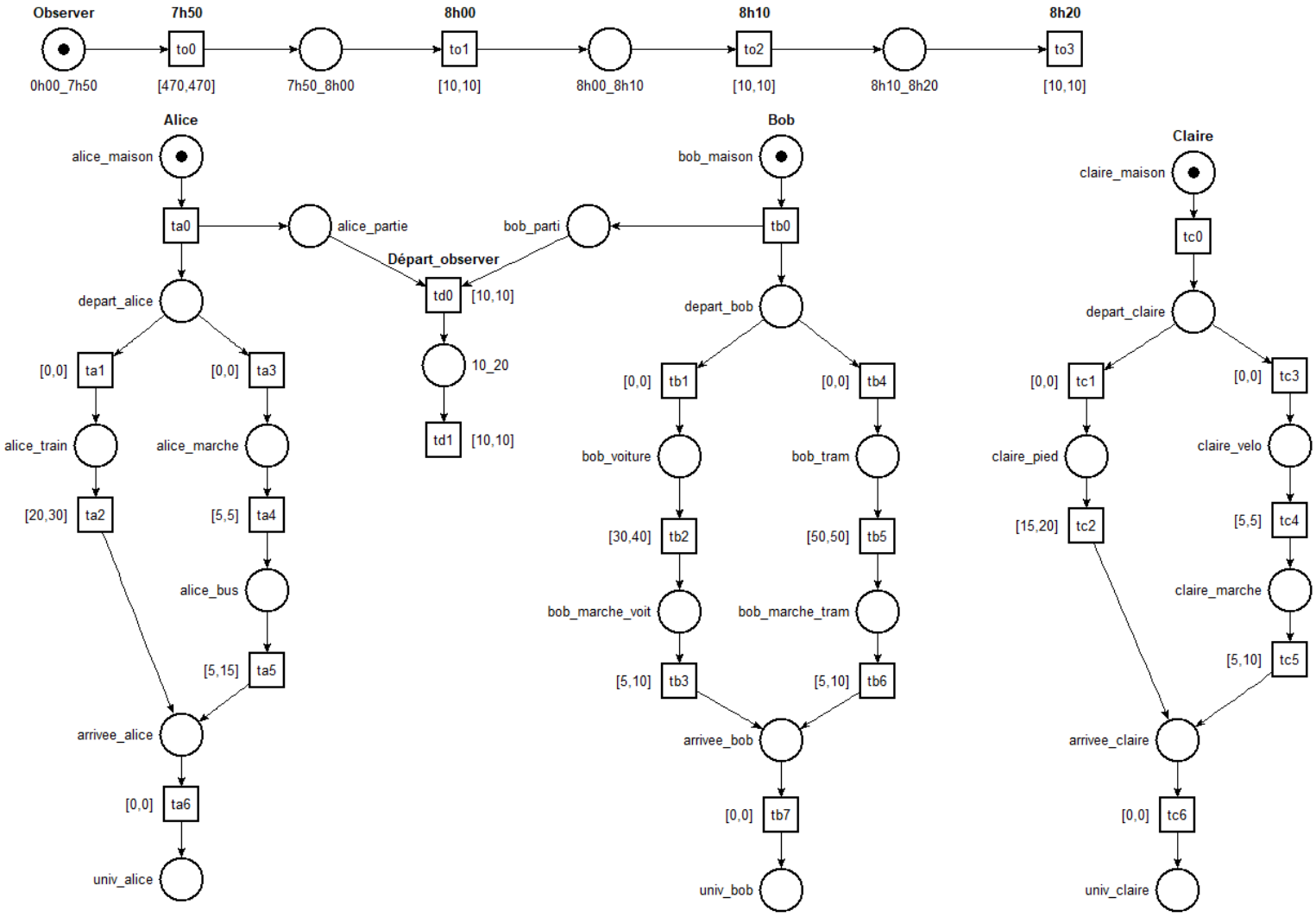


FIGURE 2 – Modélisation du système en réseaux de Petri temporels avec les observateurs