Vérification formelle

Formation Systèmes à Evénements Discrets

1ère édition
Janvier 2024

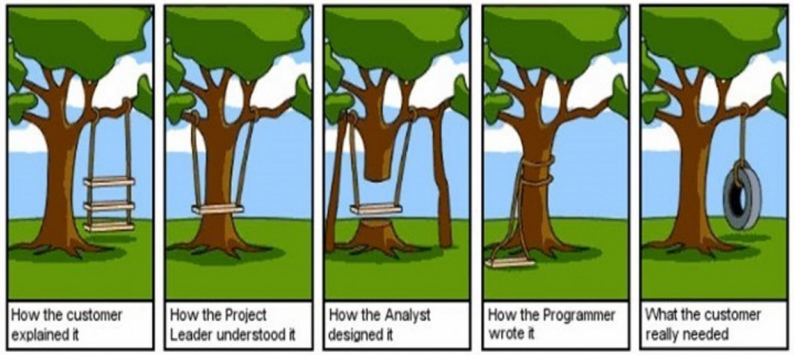
Société d'Automatique,
de Génie Industriel & de Productique

Motivations pour faire de la vérification formelle

Motivations pour faire de la vérification formelle

- Pour tous les secteurs industriels, même dans le cas de systèmes à très haute sûreté de fonctionnement, les erreurs sont découvertes tard dans le processus de développement, voire en exploitation
- Exemples.

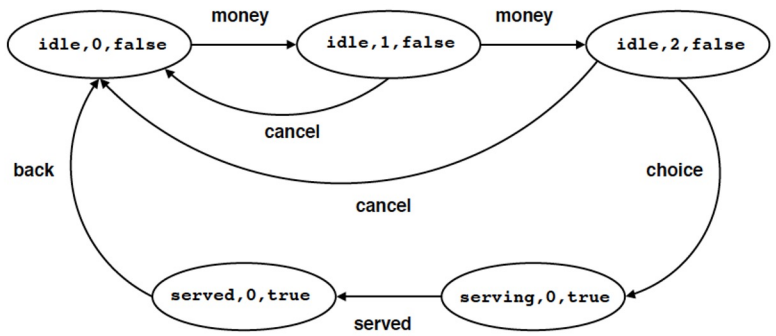
Analyser la compréhension d'un besoin client



Analyser le système de contrôle commande d'un avion

Motivations pour faire de la vérification formelle

- Il est possible de modéliser le comportement d'un système



- Sur ce comportement, il est possible d'analyser les propriétés suivantes
  - Accessibilité : Une certaine situation peut être atteinte (exemple : x peut valoir 5)
  - Invariance : Chaque état respecte une bonne propriété (exemple : x n'est jamais négatif)
  - Sureté : Quelque chose de mauvais n'arrive jamais (exemple : j'accède à un fichier que si j'ai les droits)
  - Vivacité : Quelque chose de bon finit par arriver (exemple : le code se termine)
  - Equité : Quelque chose de bon se répète infiniment souvent (exemple : si le processus demande toujours la main, il l'aura infiniment)
  - Equivalence comportementale : Deux systèmes ont le même comportement (exemple : équivalence entre un programme de base et celui optimisé)

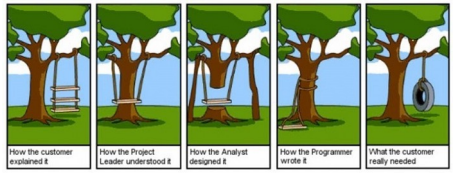
Motivations pour faire de la vérification formelle

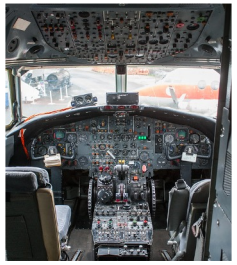
Il existe différentes méthodes

- Test sur le système réel
  - Très proche de l'utilisation nominale (+)
  - Vue très limitée des possibilités du contrôle (-)
  - Nécessite du matériel spécifique (-)
  - Certifiant (+)
  - Dénombrer tous les tests possibles (-)
- Simulation par modèle
  - Long à exécuter toutes les simulations (-)
  - Pas toujours possible de faire les cas d'utilisation possibles (-)
  - Ne nécessite pas de matériel (+)
  - Possibilité de l'utiliser tout au long du cycle de vie du système (+)
- Vérification formelle
  - Taux de couverture de 100% du comportement (+)
  - Long à exécuter (-)
  - vérification de modèles et non du système (-)
  - Possibilité de l'utiliser tout au long du cycle de vie du système (+)

Vérification formelle par model-checking

Principes du model-checking





Modèles du comportement du système $\mathcal{M}$

Modèles des spécifications φ

Vérification par model-checking

$\mathcal{M} \models \varphi$? Si non donner un contre-exemple

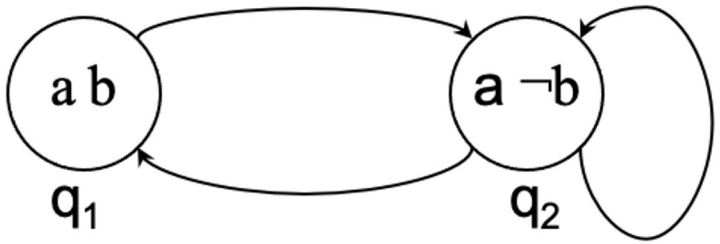
Principes du model-checking

A partir du modèle, définir une structure de Kripke où toute l'information de satisfaction des propriétés est portée par les états.

$$M = < Q, \rightarrow, AP, L, Q_0 >$$

où

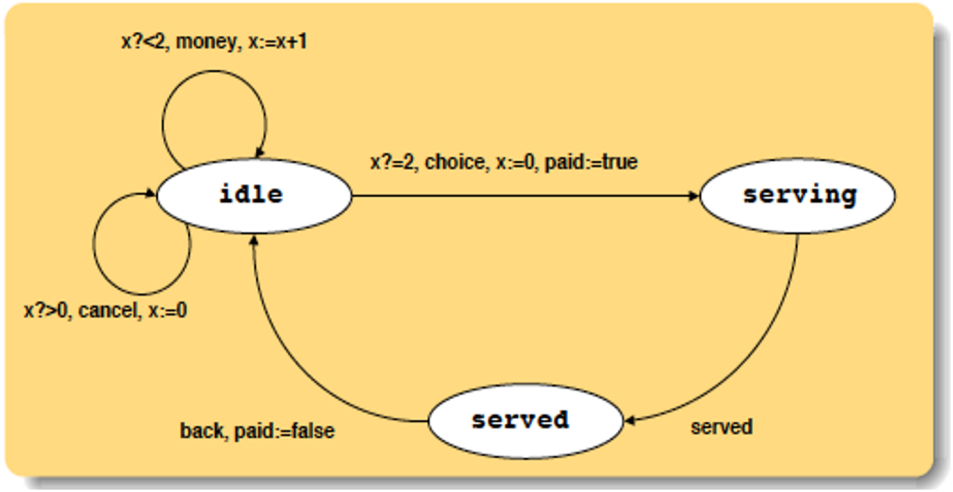
- Q : ensemble fini des états
- $\rightarrow \subseteq Q \times Q$: relation de transition (relation totale)
- AP: ensemble de propositions logiques atomiques
- $L : Q \rightarrow 2^{AP}$ fonction qui étiquette un état avec un ensemble de propositions atomiques vraies dans cet état
- Q_0 : état initial



Principes du model-checking

Exemple. Si nous considérons une machine à café et que nous voulons vérifier qu'il n'est pas possible de servir un café sans avoir payé

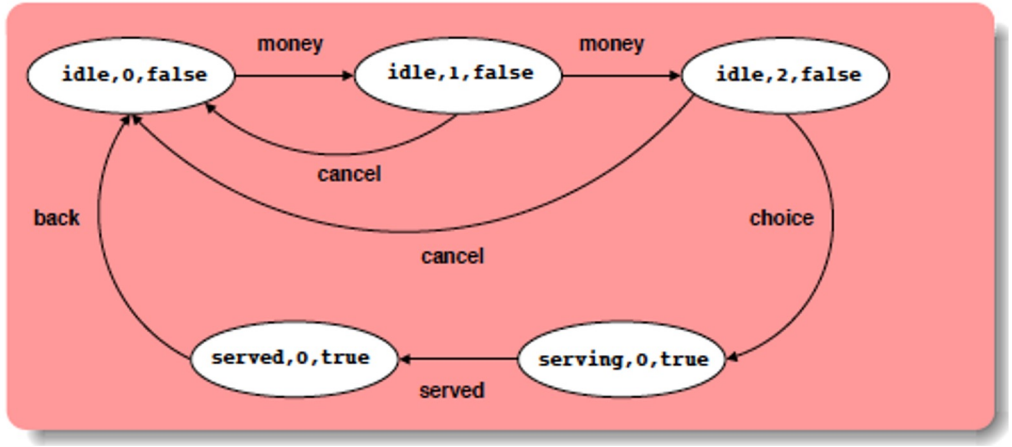
1. Définir un modèle du comportement de la machine



Principes du model-checking

Exemple. Si nous considérons une machine à café et que nous voulons vérifier qu'il n'est pas possible de servir un café sans avoir payé

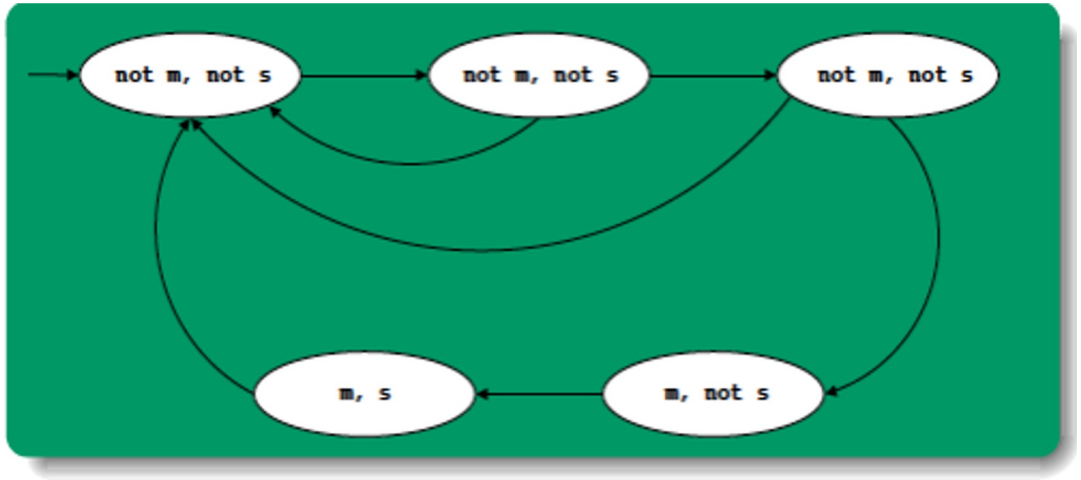
1. Définir un modèle du comportement de la machine
2. Définir les propositions atomiques



Principes du model-checking

Exemple. Si nous considérons une machine à café et que nous voulons vérifier qu'il n'est pas possible de servir un café sans avoir payé

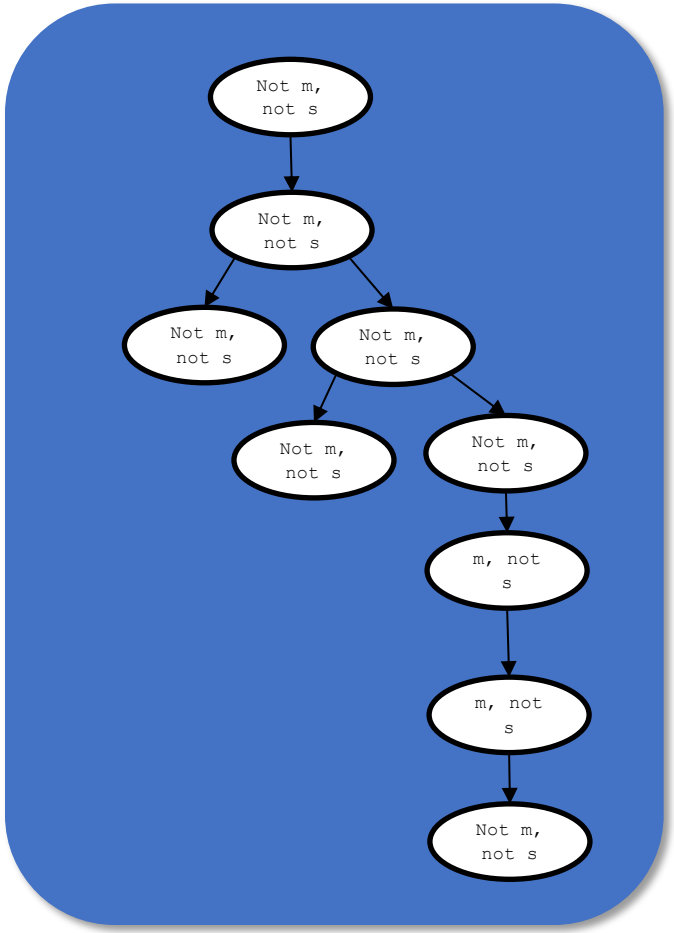
1. Définir un modèle du comportement de la machine
2. Définir les propositions atomiques
3. Étiqueter chaque état par la satisfaction de la proposition



Principes du model-checking

Exemple. Si nous considérons une machine à café et que nous voulons vérifier qu'il n'est pas possible de servir un café sans avoir payé

1. Définir un modèle du comportement de la machine
2. Définir les propositions atomiques
3. Étiqueter chaque état par la satisfaction de la proposition
4. Déplier la structure de Kripke



Modélisation des propriétés

Formalisation des propriétés

Les propriétés sont modélisées en logique temporelle qui se décompose en général en deux parties

- Une proposition pour décrire les propriétés du système à un instant donné

Exemple. « la place P contient n jetons », « aucune transition ne peut être franchie »

- Un ajout de modalités temporelles pour exprimer des propriétés temporelles

Exemple. « il est possible que. . . », « il est certain que. . . », « . . . toujours. . . »

Formalisation des propriétés

Les propriétés sont modélisées en logique temporelle qui se décompose en général en deux parties

- Une proposition pour décrire les propriétés du système à un instant donné

Exemple. « la place P contient n jetons », « aucune transition ne peut être franchie »

- Un ajout de modalités temporelles pour exprimer des propriétés temporelles

Exemple. « il est possible que. . . », « il est certain que. . . », « . . . toujours. . . »

Les modalités temporelles se décomposent en logique classique booléenne et des opérateurs dédiés au temps

- Connecteurs temporels : Suite d'événements attendus le long d'un seul chemin partant de l'état initial

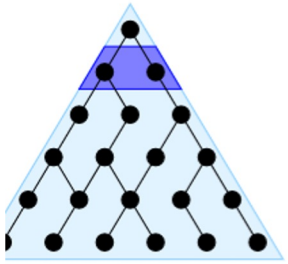
  X (suivant), F (un jour), G (toujours), U (jusqu'à), ⇒ (implique)

- Quantification des chemins sur un dépliage de la structure de Kripke : Quantifie les chemins partant d'un état qui doivent vérifier la propriété

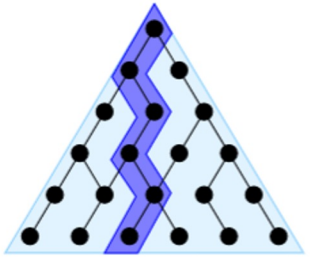
  A (pour tous les chemins), E (il existe un chemin)

Formalisation des propriétés

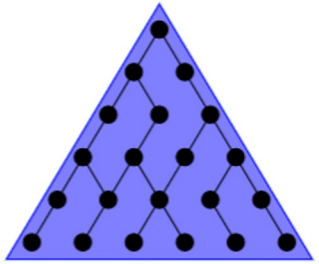
Exemple. Sur une structure de Kripke, différentes propriétés :



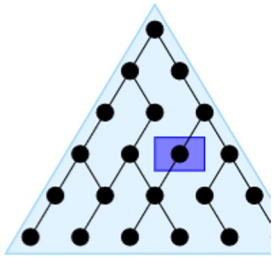
AX(p)
AF(p)



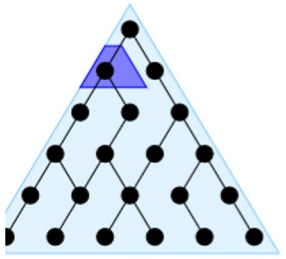
EG(p)



AG(p)

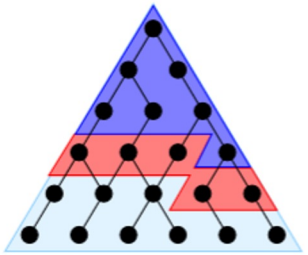


EF(p)

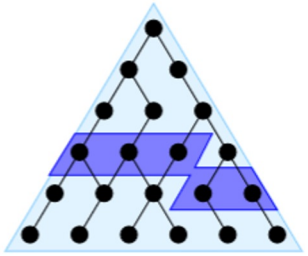


EX(p)

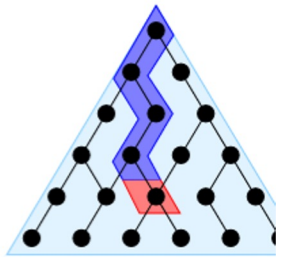
EF(p)



A(pUq))



AF(p)



E(pUq))

Formalisation des propriétés

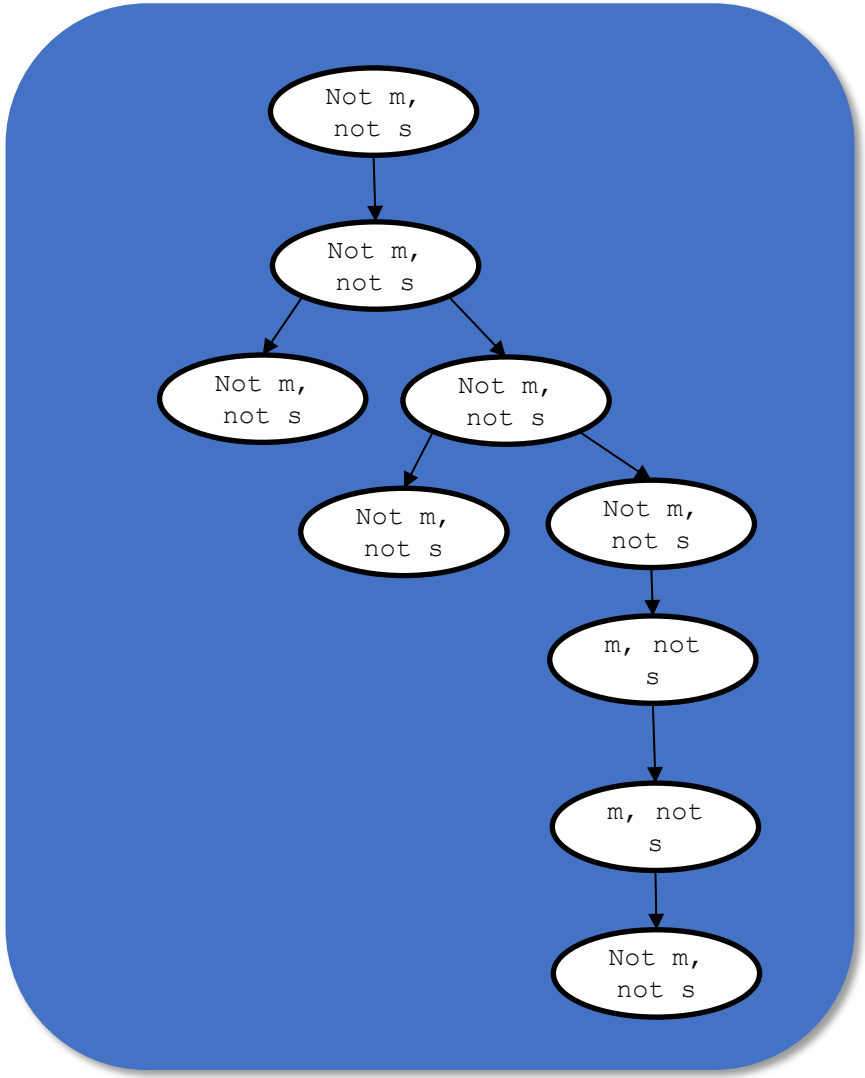
Sur une structure de Kripke, il est donc possible de vérifier :

- Atteignabilité : EF(x = 0)
- Invariance : AG (x = 0)
- Sûreté : AG (not (x<0)
- Vivacité : AG(p → Fq)

Principes du model-checking

Exemple. Si nous considérons une machine à café et que nous voulons vérifier qu'il n'est pas possible de servir un café sans avoir payé

La propriété à vérifier EF(not m and s) pour laquelle le model-checker doit retourner FALSE



Mise en application

Application : Barrière train

Soit un passage à niveau composé d'une barrière, d'un train et d'un système de pilotage, nous souhaitons vérifier que :

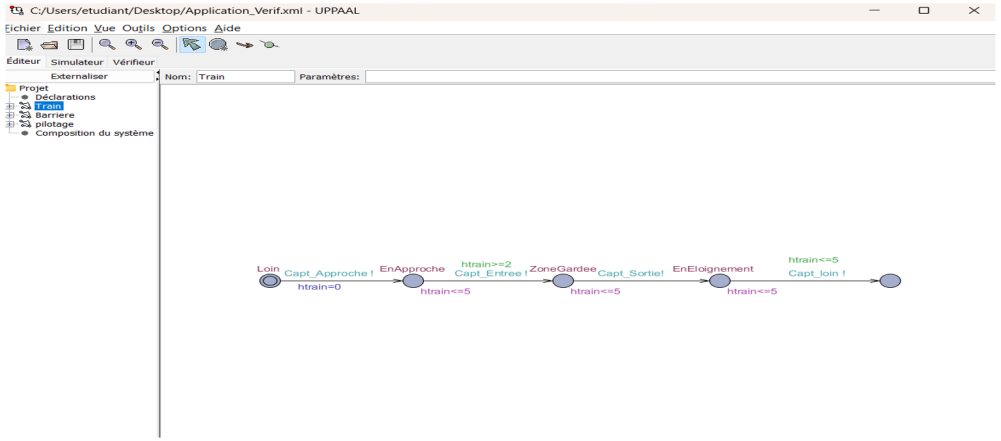
1. Quand le train est dans la section gardée, la barrière est toujours fermée
2. La barrière ne reste pas fermée plus de 5 minutes

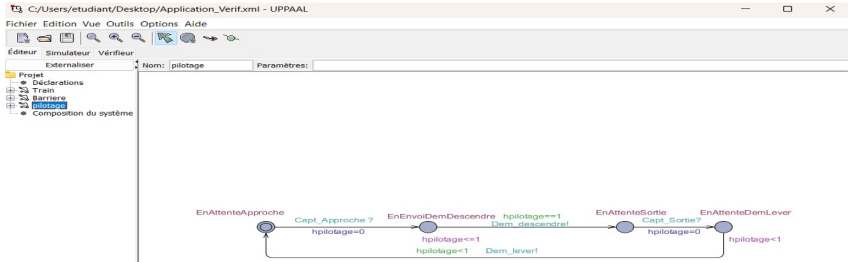
Application : Barrière train

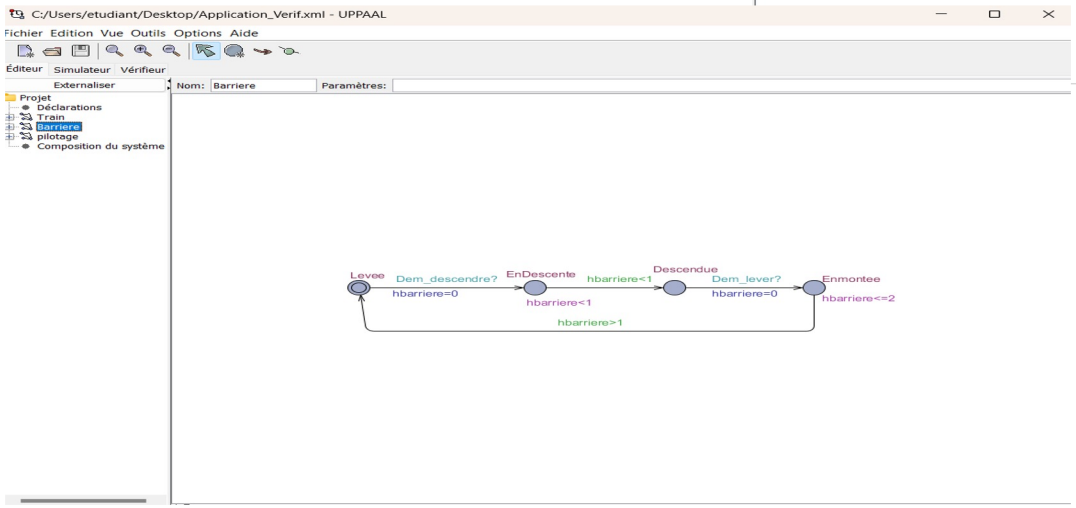
Soit un passage à niveau composé d'une barrière, d'un train et d'un système de pilotage, nous souhaitons vérifier que :

1. Quand le train est dans la section gardée, la barrière est toujours fermée
2. La barrière ne reste pas fermée plus de 5 minutes

Modèles du système :







Application : Barrière train

Soit un passage à niveau composé d'une barrière, d'un train et d'un système de pilotage, nous souhaitons vérifier que :

1. Quand le train est dans la section gardée, la barrière est toujours fermée
2. La barrière ne reste pas fermée plus de 5 minutes

Modèles des propriétés :

1. Quand le train est dans la section gardée, la barrière est toujours fermée :
   EF (train_dans_section_gardée and not barriere_fermee)
2. La barrière ne reste pas fermée plus de 5 minutes :
   EF(barriere fermee and hbarriere>5)

Application : Barrière train

Soit un passage à niveau composé d'une barrière, d'un train et d'un système de pilotage, nous souhaitons vérifier que :

1. Quand le train est dans la section gardée, la barrière est toujours fermée
2. La barrière ne reste pas fermée plus de 5 minutes

Modèles des propriétés :

1. Quand le train est dans la section gardée, la barrière est toujours fermée
2. La barrière ne reste pas fermée plus de 5 minutes

Références bibliographiques

Références bibliographiques

- Clarke E. et al (1994). Model checking and abstraction. ACM-TOPLAS, vol. 16, no 5.
- G. Behrmann, J. Bengtsson, A. David, K. Larsen, P. Petterson et Wang Yi. (2002) UPPAAL Implementation Secrets . Proc. FTRTFT 2002, Springer LNCS 2469, pp. 3-22

Equipe pédagogique

Equipe pédagogique

Auteur.rice.s : Pascale Marangé, Alexandre Philippot et Olivier H. Roux

Intervenante : Pascale Marangé