

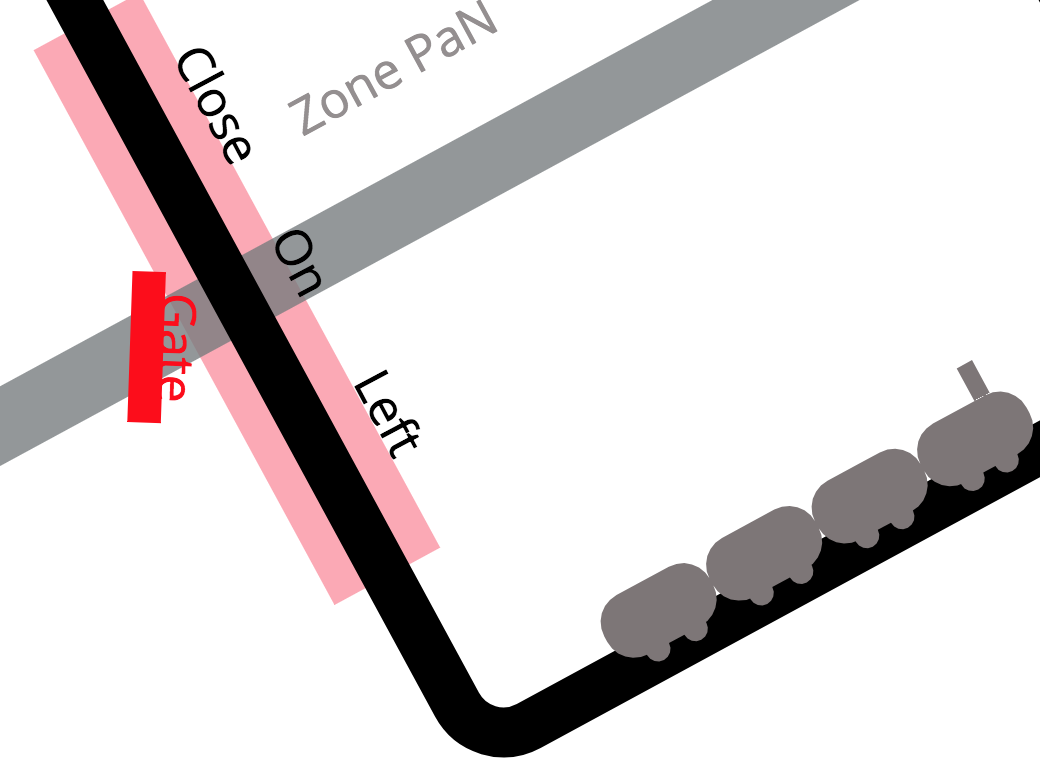
Automates et Vérification Formelle

Séance pratique 3

Ciprian Teodorov et Philippe Dhaussy

# Modèles temporisés : exemple du « *Passage à Niveau »*

Le but de cet exercice est de créer un «**Contrôleur de passage à niveau**» permettant de gérer une barrière pour arrêter la circulation routière lors du passage d’un train. La figure suivante illustre l’architecture physique du passage à niveau.



Pour détecter l’arrivée d’un train le passage de niveau est équipé d’une « Zone de détection » mise en œuvre avec un **circuit de voie**. Celle-ci permet également de détecter la libération de la zone. Afin d’assurer la sécurité et la fluidité du trafic, une fois détecté dans la zone, les trains doivent arriver en maximum 50 secondes au niveau du passage (franchir la partie « close » de la zone). Une fois sur le passage (la partie « on » de la zone), les trains doivent le quitter en maximum 40 secondes. Toutefois leur vitesse est limitée, ainsi ils sont obligés de rester dans la Zone PaN au moins 20 secondes.

La barrière, installée physiquement au croisement de la route avec la voie ferrée, dispose d’un moteur électrique permettant son ouverture et fermeture. Le moteur est commandé par un circuit électronique à deux entrées (open ; close). Lors de la réception d’une commande la barrière mettra 10 à 20 secondes pour s’ouvrir ou fermer. La séquence de relevage de la barrière peut être interrompue à tout moment par l’arrivé d’un train sur le passage à niveau.

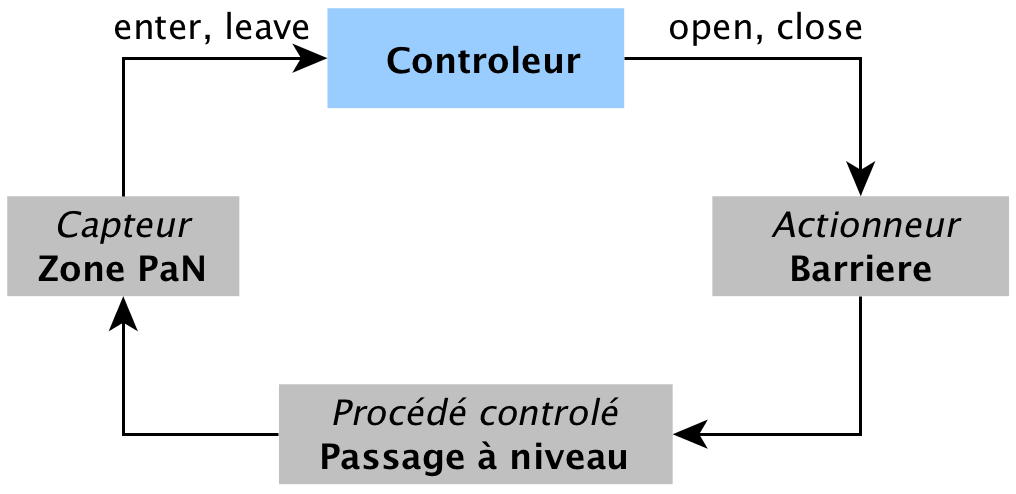
Pour réduire les coûts de développement et d’entretien, le module de contrôle du PaN doit être générique afin de permettre son déploiement sur des PaN avec une ou plusieurs voies de circulation.

**Hypothèses simplificatrices :**

1. Les réglés de circulation sont respectées par les chauffeurs.
2. Les infrastructures physiques sont réputées fiables (pas de panne).
3. Les trains ne s’arrêtent jamais dans la « Zone de détection » du passage à niveau.
4. On ne peut pas avoir plusieurs trains sur la même voie. Un train ne peut pas en cacher un autre.
5. Les entrées et les sorties du contrôleur sont des signaux électriques qui se propagent instantanément.
6. Le temps d’exécution du contrôleur est négligeable.

## Modélisation du système pour la vérification formelle par « model checking »

Le contrôleur de passage à niveau, qu’on doit développer, est un exemple de système de contrôle. Les entrées du contrôleur sont fournies par le capteur de Zone, en fonction des entrées, le contrôleur envoie des commandes vers un actionneur (la Barriere). Cette boucle de contrôle est fermée par le procédé physique à contrôler : la circulation sur un passage à niveau. La figure suivante illustre ces éléments et leurs interactions.

****

Avant de pouvoir commencer le développement du contrôleur, il faudrait donc identifier et modéliser son environnement de fonctionnement. Le modèle d’environnement doit encoder l’ensemble des lois et d’hypothèses qui sont imposés sur notre système.

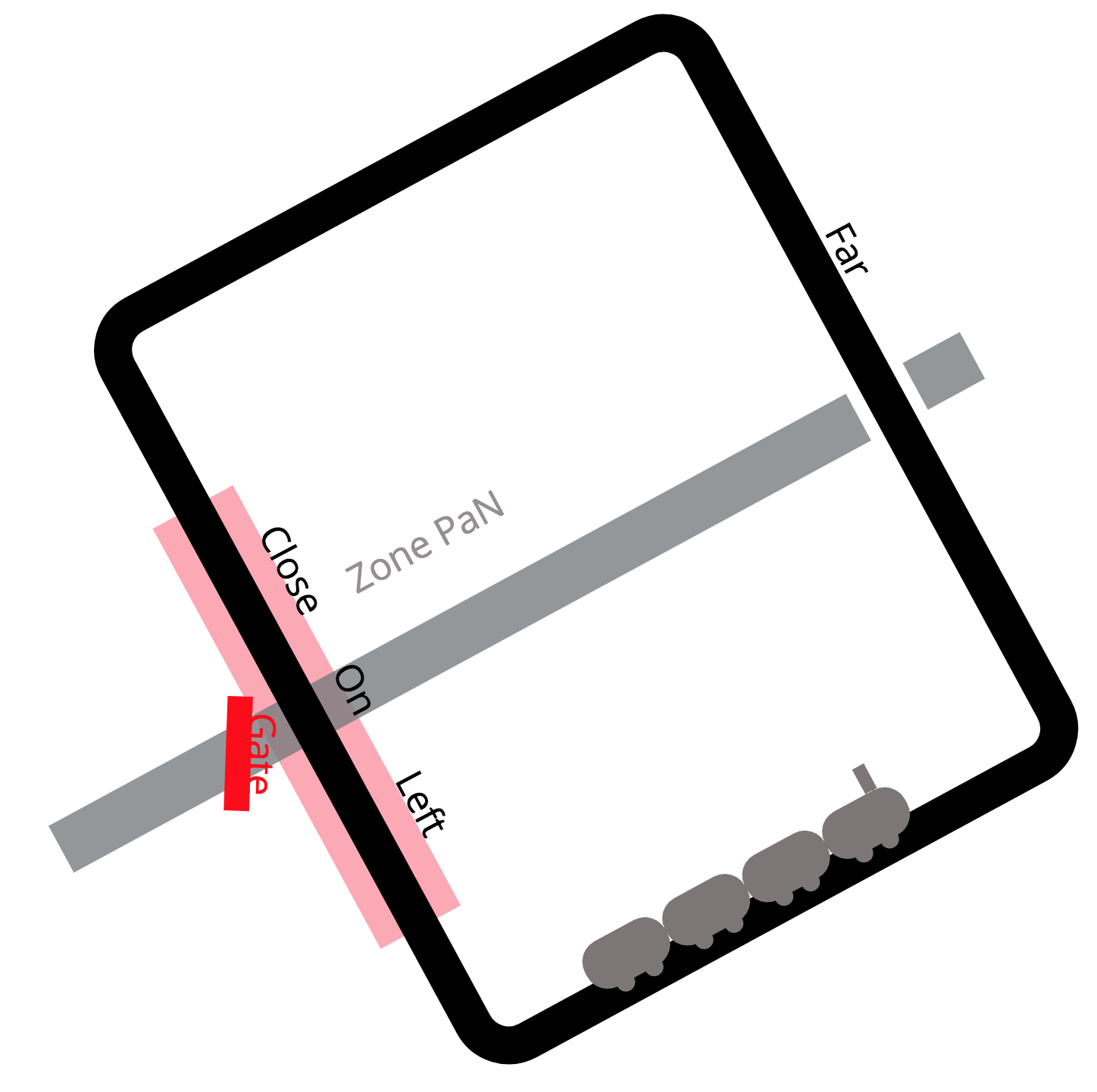
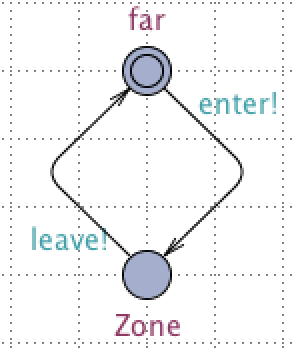
### Modèle abstrait de l’environnement ferroviaire : Voie, Zone PaN et Train

Comme indiqué précédemment le capteur de Zone émet des signaux indiquant l’arrivé d’un train et son départ.

**Exercice 7 :** Créer un automate UPPAAL capable de recevoir les signaux (enter, leave) émis par le capteur de Zone. Cet automate servira comme « Mock » d’environnement pour le développement du modèle ferroviaire.

*Note :* Utiliser des canaux de synchronisation pour représenter les signaux.

L’environnement ferroviaire considéré est composé de 3 éléments : la voie, la zone de détection et le train. Cet environnement nous permet d’abstraire le capteur et une partie du procédé physique contrôlé. La figure suivante illustre ces éléments. Dû à l’hypothèse simplificatrice **HS4** on peut considérer que la voie est un circuit fermé avec un train.

** **

D’un point de vue abstrait la voie peut être modélisé avec 2 états : un état « Far » et un état « Zone ». Le passage du train entre ces 2 zones est modélisé avec une transition de l’état « Far » à l’état « Zone » et une transition dans le sens inverse. Lors du passage de l’état far à l’état Zone un signal « enter » est émis. Le signal « leave » est émis lors du passage dans le sens inverse. L’automate précèdent illustre cette modélisation.

**Exercice 8 : Modèle abstrait de l’environnement ferroviaire**

Rappel : chaque état sans invariant permettra au système de rester dans l’état entre 0 et infini

1. Composer l’automate crée lors de l’exercice 7 avec l’automate précédent, et simuler.
2. Raffiner l’état « Zone » pour capturer les 3 parties : Close, On et Left. Nommé la *template* **« Train »**
3. Ajouter les contraintes nécessaires pour capturer le comportement temporel des trains dans la Zone de détection. Simuler le modèle à chaque pas pour s’assurer de son bon fonctionnement.
   1. Ajouter une horloge « temps » dans les déclarations de la *template*.
   2. Pour chaque contrainte de type : « le train doit arriver/quitter en maximum x secondes » ajouter un invariant « ***temps <= x*** »dans l’état qui doit être quitté pour respecter la contrainte.
   3. Pour chaque contrainte de type : « le train doit … au moins x secondes » ajouter une garde « ***temps >= x*** »sur la transition qui doit-être retardée.
   4. N’oublier pas de réarmer l’horloge systématiquement (***temps = 0***).
4. Vérifier que le modèle obtenu respecte les propriétés dans Table 1
5. Créer une propriété **PT11** permettant de vérifier que le Train ne restera plus de 40 secondes dans l’état « on ».
6. Probablement la plus importante propriété à vérifier sur ce système est **PT12** : « *le train ne doit pas arriver sur le passage à niveau plus vite que la barrière à le temps de se fermée* ». Le langage de spécification de UPPAAL ne nous permet pas d’exprimer cette propriété directement. Ainsi pour la vérifier, on va utiliser un « automate » observateur qui permettra d’observer le passage du temps et d’enregistrer toute violation de la propriété. Cet automate est illustré dans la figure suivante.

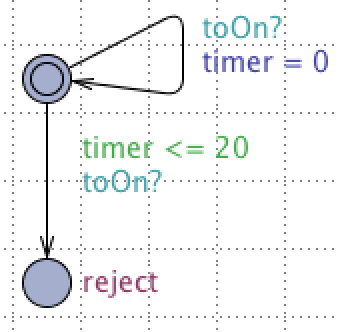
Cet automate est synchronisé avec l’automate Train à travers le canal de broadcast « toOn » et permet d’observer si la transition entre l’état « close » et l’état « on » arrive plus vite que 20 secondes (le temps nécessaire pour garantir la fermeture de la barrière). Si jamais la transition est franchie avant 20 secondes alors cet automate passe dans l’état « reject », dans tous les autres cas il restera dans son état initial. Ainsi la propriété à vérifier peut être exprime comme suit : **PT12** = **A[] not observateur.reject** Ajouter cet automate à votre modèle. Vérifiez si la propriété P12 est respectée.

Table 1 Propriétés à vérifier

|  |  |
| --- | --- |
| PT1 | A[] not deadlock |
| PT2 | E<> Train.far |
| PT3 | E<> Train.on |
| PT4 | E<> Train.close |
| PT5 | E<> Train.left |
| PT6 | Train.close --> Train.on |
| PT7 | Train.on --> Train.left |
| PT8 | Train.left --> Train.far |
| PT9 | A[] Train.close imply Train.temps >= 0 && Train.temps <= 50 |
| PT10 | A[] (Train.left imply Train.temps == 0) |

### Modèle abstrait de la barrière

**Exercice 9 :** Créer un « Mock » d’environnement pour la barrière, cet automate sera capable d’envoyer sans contraintes les commandes open/close à la barrière.

**Exercice 10 :** Réaliser un modèle de barrière qui prend en compte l’ensemble contraintes temporelles. Utiliser le simulateur de UPPAAL pour valider votre avancement.

**Exercice 11 :** Identifier les propriétés qui doivent être respecté par la barrière. Spécifier et vérifier ces propriétés avec UPPAAL.

### Modèle du « Contrôleur de passage à niveau »

Les modèles « Train » et « Barrière » représentent l’environnement de vérification nécessaire pour le développement et la vérification du contrôleur PaN.

**Exercice 12 : Modèle du « Contrôleur »**

1. Créer un automate contrôleur connecté avec le Train et la Barrière avec des canaux synchrones.
2. Simuler le système ainsi obtenu pour s’assurer de son bon fonctionnement.
3. Vérifier qu’il n’y a pas de deadlock **PC0**.
4. La principale propriété à vérifier sur ce système **PC1** : « La barrière n’est jamais ouverte si un train est sur le passage à niveau ». Spécifier et vérifier cette propriété avec UPPAAL.

Train1 = Train();

Train2 = Train();

**system** Train1, Train2, Barriere, Contrôleur;

1. Modifier la déclaration de votre système comme illustré dans l’encadré. Ceci introduit 2 voies parallèles avec 1 train chacune.
2. Simuler le nouveau système.
3. Vérifier si ce système respecte les propriétés **PC0** et **PC1.**

**Exercice 13 : « la barrière s’ouvrira finalement ».**

Un contrôleur qui ferme la barrière et la garde fermée pour toujours satisfait la propriété **PC1** d’une manière triviale. Pour rendre le modèle réaliste et utile il faudrait s’assurer que notre système respecte une propriété de vivacité qui garantit « que la barrière finira par s’ouvrir ». Celle-ci peut-être spécifié avec UPPAAL comme suit : **PC2** = **barrière.down 🡪 barrière.up**.

1. Vérifier cette propriété sur un système à un seul train ? Si elle est respecté, précisez exactement les conditions qui permettent la vérification de cette propriété. Sinon identifier clairement les raisons pour lesquels cette propriété n’est pas respectée.
2. Modifier le modèle pour prendre en compte cette contrainte. Vérifier **PC0, PC1** et **PC2** sur le nouveau modèle.
3. Vérifier PC2 sur le nouveau modèle avec 2 train. Si la propriété n’est pas vérifiée corrigé le modèle.