Sécurité & Fiabilité

Projet vérification (Réseaux de Petri)

Master Informatique parcours SAR
Université Pierre et Marie Curie

UE SF - 5I454

10/12/2017

Participants :
Florian REYNIER - 3506673
Alexandre Lavigne - 3502363

1 Rappel du sujet

Le but de ce projet est de réaliser un modèle sous forme d'un réseau de Petri du système suivant :

— **VAA :** qui représente un véhicule automatisé de type A (qui circule dans un sens donné sur le pont),
— **VAB :** qui représente un véhicule automatisé de type B (qui circule dans un sens donné sur le pont),
— **CTRLP :** qui représente le système de contrôle qui autorisera lesVAA et les VAB à entrer sur le pont,
— **P :** qui représente les propriétés intrinsèques du pont.

1.1 Controleur

Le controller est la pour assurer que seul les voitures d'un certains type passe à la fois. Parmis ces voitures il doit limiter le nombre de voiture consécutivfes qui passent pour laisser aux voiture de l'autre type l'occasion de paser.

1.1.1 Vérification du type de voitures

Pour ne laisser qu'un seul de type de voiture passer nous avons définis une classe `Type` qui varie dans l'enssemble : `[a, b]`. le modèl contient une place `LastType` qui contient initialment le jeton `Type.a`.

Comme cité précédement la voiture est en fait un type composé. Ce type est composé d'un numéro de voiture et d'un type de voiture. Pour qu'une voiture puisse avancer, elle doit tirer ce jeton, et ce dernier doit être égale au type de la voiture.

1.1.2 Vérification capacité du pont

Le pont ne peut accepter que `Nalt` voitures à la fois. Pour modéliser cette contrainte nous avons utilisé une classe de type `Compteur` avec la plage de valeurs suivante : [0 ;Nalt]. Dans nos exemples nous utilisons `Nalt = 2`.

Nous avons une place `Position` qui est initialisé avec le jeton `Counter.0` et qui produit le jeton qu'elle contient, puis récupére le jeton + 1. ainsi la guarde ne laisse passer une voiture que si la place `Position` ne dépasse pas une valeur maximum.

Voici le modèle du controleur Figure :1.1.3 :
Le coeur du controlleur est représenté en rouge.

La transition Dem (de couleur bleu sur la gauche) recoit le jeton de la voiture, si ce jeton satisfait la garde, alors la voiture peut aller sur le pont.

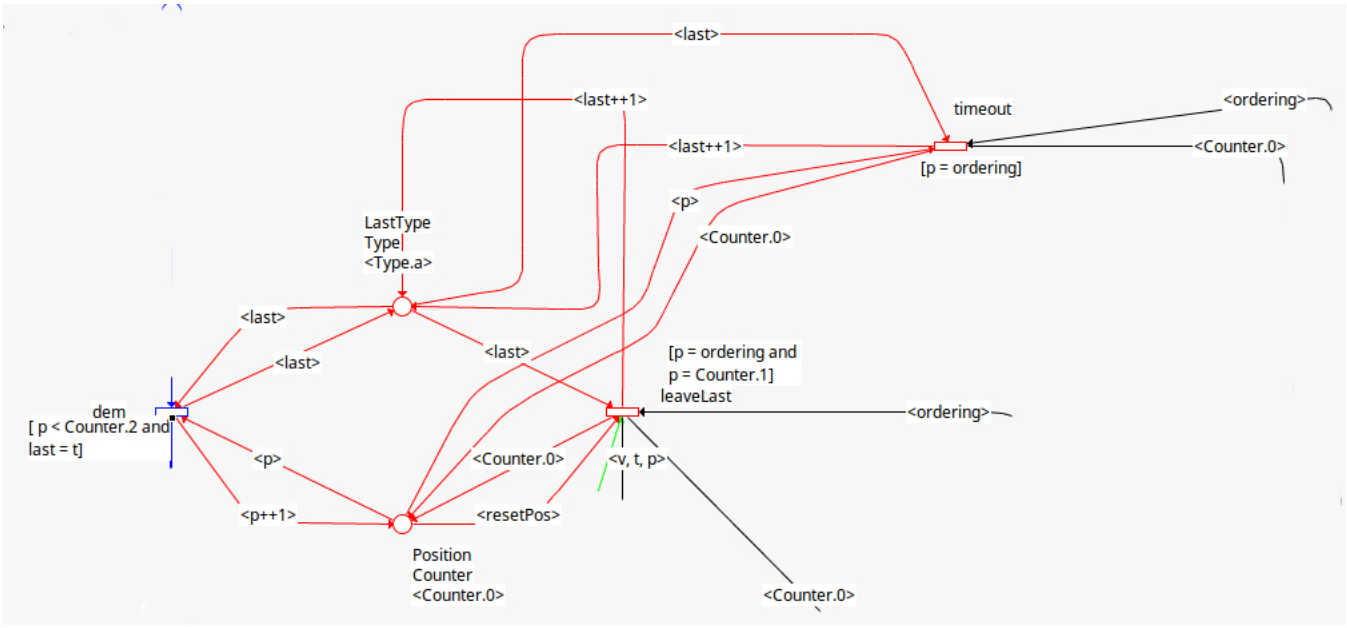
La transition `leaveLAst` est lié au Pont et au Controleur car elle permet à la dernière voiture de quitter le Pont mais c'est elle qui permet aussi le changement de sens du pont. Cette transition (`leaveLast`) change la place `LastType` avec l'autre type, remet à zéro le compteur en entré et en sortie du Pont.

1.1.3 Timeout

Le timeout est géré par le `Controlleur`. Nous avons choisi de le représenter sous la forme d'une transition.

Cette transition, pour être tirée doit prendre le jeton de la place `LastType`, `Position`, `Orderer`. en prenant ces jetons, elle renvoie dans chaqcune des place, un compteur = 0 pour les 2 compteurs, et le type suivant pour la palce `LastType`. Ainsi les voitures venant dans l'autre sens (Aka : l'autre type de voiture) peuvent passer.

Pour assurer la propriété `P1` cette transition de `timeout` ne peux être tiré que s'il y a autant de voiture qui sont entrées que de voitures qui sont sorties. ainsi si la transition est tiré c'est qu'il n'y a plus de voitures sur le Pont.

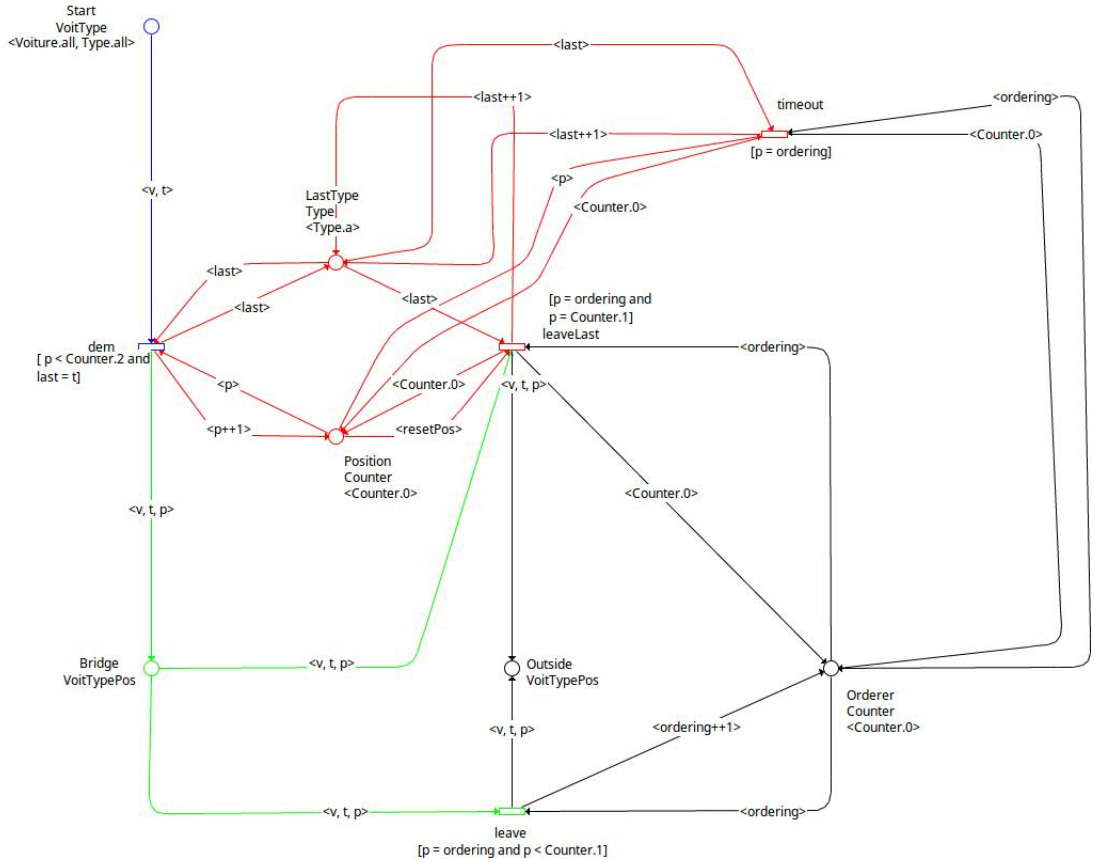


1.2 Modèle complet

Après assemblage des différent composant, on arrive donc au modèle complet suivant Figure 1.2 :

Il pemet donc `Nalt` voitures de passer de la place `Start` à la place `Bridge` en tirant la transition `dem`. Une fois sur le pont toutes les voitures sauf la dernière vont dans la place `Outsite` en tirant la tansition `leave`. La dernière voiture à être entrée sur le pont va aussi dans la place `Outside` mais cette fois ci en tirant la tansition `leaveLast`, cette dernière va remettre à zéro les compteurs et laisser les voitures de l'autre type passer.

Si par hasard on a : `Nalt` > nbVoitures, alors même si toutes les voiture d'un type passent, les voitures de l'autre type vont attendre que d'autres voiture arrivent. Du coup c'est à ce moment que le timeout peut ce déclencher (si autant de voitures sont entrée que sorties) et du coup laisser la possibilité au voiture de l'aute type de passer.




```
Class Voiture is 1..2;
Type is [a,b];
Counter is 0..2;

Domain VoitType is <Voiture, Type>;
VoitTypePos is <Voiture, Type, Counter>;

var v in Voiture;
t, last in Type;
p, ordering, resetPos in Counter;
```