Projet SF

RAPPORT

Modèlisation et Vérification du comportement de véhicules automatiques sur un pont à voie unique

Réalisé par ***DOAN Cao Sang***

7 décembre 2015

Chapitre 1

Propriétés

Le programme doit satisfaire les propriétés suivantes :

1. *Il n'y a pas de collision (i.e. deux véhicules circulants en sens inverse) sur le pont.*
2. *Un véhicule qui arrive est certain de passer sur le pont à l'issue d'une durée bornée.*

Chapitre 2

Explication

Je me constate que les jetons colorés ne sont pas bien exporter en format CAMI, par exemple, dans le modèle *Controlleur(CTRLP)* la place **Budget**, domain **TimeOut**, j'ai initialisé avec le jeton *<TimeOut.3>* mais quand j'exporte en format CAMI et ré-importe, cette place est initialisée avec un jeton *<3>*. Du coup, les formules CTL ne marche pas car le coloane ne connaît pas le jeton *<3>*. Donc, j'ai mis à côté des fichiers CAMI, des fichiers du type **.model**, grâce à ces fichiers vous pouvez importer directement le modèle (dans le coloane, **Import** → **File system**). En plus, j'ai mis aussi mon répertoire entier dans lequel je travaille pour vous faciliter votre travail (sous Mac). Merci de votre attention.

2.1 Question 1.1

2.1.1 Vaa

VerificationA est une `place` de sortie, qui lie le composant *Vaa* avec le contrôlleur *CTRLP*, lorsque un véhicule A veut traverser le pont, d'abord il communique avec le contrôlleur *CTRLP* pour avoir son autorisation.

OKControlleurA est une `place` d'entrée qui lie entre *Vaa* et *CTRLP*, lorsque le *Vaa* peut traverser sur le pont, le *CTRLP* met son jeton pour que *Vaa* puisse passer.

OKPont est une `place` entrée qui lie le *P* avec le *Vaa*, le *P* marque cette place lors qu'il reçoit l'autorisation de contrôlleur *CTRLP*.

AuRevoirControlleurA est une `place` de sortie, le *Vaa* marque cette place dès qu'il sort du *P*.

2.1.2 Vab

VerificationB est une `place` de sortie, qui lie le composant *Vab* avec le contrôlleur *CTRLP*, lorsque un véhicule B veut traverser le pont, d'abord il communique avec le contrôlleur *CTRLP* pour avoir son autorisation.

OKControlleurB est une `place` d'entrée qui lie entre *Vab* et *CTRLP*, lorsque le *Vab* peut traverser sur le pont, le *CTRLP* met son jeton pour que *Vab* puisse passer.

OKPont est une `place` d'entrée qui lie le *P* avec le *Vab*, le *P* marque cette place lors qu'il reçoit l'autorisation de contrôlleur *CTRLP*.

AuRevoirControlleurB est une `place` de sortie, le *Vab* marque cette place dès qu'il sort du *P*.

2.1.3 Pont

CapaciteController est une `place` de sortie, qui lie le composant *P* avec le contrôlleur *CTRLP* pour lui signaler sa capacité restante.

OKPont est une `place` de sortie qui lie le *P* avec les 2 composants *Vaa* et *Vab*, le *P* marque cette place lorsque sa capacité reste suffisante.

2.1.4 CTRLP

VerificationA est une `place` d'entrée, qui lie le composant *Vaa* avec le contrôlleur *CTRLP*, lorsque un véhicule A veut traverser le pont, le *CTRLP* doit d'abord, recevoir sa demande.

VerificationB est une `place` d'entrée, qui lie le composant *Vab* avec le contrôlleur *CTRLP*, lorsque un véhicule B veut traverser le pont, le *CTRLP* doit d'abord, recevoir sa demande.

OKControlleurA est une `place` de sortie qui lie entre *Vaa* et *CTRLP*, lorsque le *Vaa* peut traverser sur le pont, le *CTRLP* met son jeton pour que *Vaa* puisse passer.

OKControlleurB est une `place` de sortie qui lie entre *Vab* et *CTRLP*, lorsque le *Vab* peut traverser sur le pont, le *CTRLP* met son jeton pour que *Vab* puisse passer.

CapaciteController est une `place` d'entrée, qui lie le composant *P* avec le contrôlleur *CTRLP* pour savoir la capacité restante du *P*.

AuRevoirControlleurA est une `place` d'entrée, cette place est marqué par le *Vaa*.

AuRevoirControlleurB est une `place` d'entrée, cette place est marqué par le *Vab*.

2.2 Question 1.2

Cette composant contient 8 places, dont 4 ont pour le rôle d'interface, et 3 transitions.

La place **VehiculeA** modélise le véhicule A lors qu'il arrive devant le pont. S'il veut passer sur le pont, d'abord, il communique avec le contrôlleur par la transition *demanderEntrerA*, cette transition met un jeton dans l'interface **VerificationA** qui lie avec le contrôlleur *CTRLP* et passe à l'état **AttenteAutorisationA**. Lorsqu'il reçoit l'autorisation du contrôlleur *CTRLP* via l'interface **OKControlleurA** et celle du pont *P* via l'interface **OKPont**, il peut traverser le pont, en passant à l'état **DansPontA**. Ensuite, quand il sort, il marque la place d'interface **AuRevoirControlleurA** et aussi passe à l'état **FiniA** par la transition *sortPontA*.

La sécurité est garantie car, pour passer à l'état **AttenteAutorisationA** le véhicule doit d'abord communiquer avec le contrôlleur *CTRLP*. Pour entrer, il doit avoir l'autorisation du contrôlleur *CTRLP* et consommer le jeton dans la place d'interface **OKPont**, cette place modélise la consommation de la capacité du pont *P*. Enfin, il signale sa sortie du pont au contrôlleur *CTRLP* en mettant un jeton dans la place **AuRevoirControlleurA**.

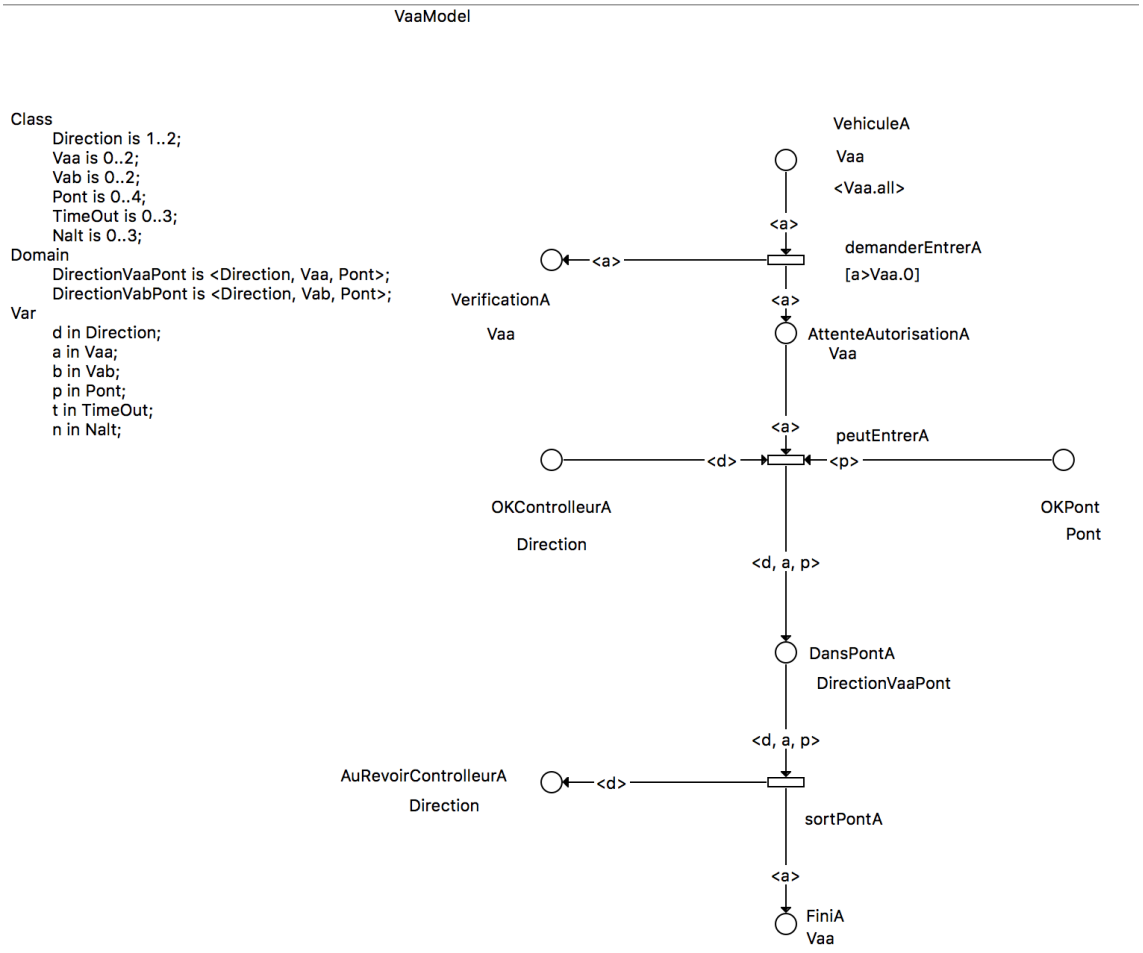

FIGURE 2.1 – Le composant modélise le véhicule automatisée A.

2.3 Question 1.3

Cette composant est identique avec celle de véhicule A, contient 8 places, dont 4 ont pour le rôle d'interface, et 3 transitions.

La place **VehiculeB** modélise le véhicule B lors qu'il arrive devant le pont. S'il veut passer sur le pont, d'abord, il communique avec le contrôlleur par la transition *demanderEntrerB*, cette transition met un jeton dans l'interface **VerificationB** qui lie avec le contrôlleur *CTRLP* et passe à l'état **AttenteAutorisationB**. Lorsqu'il reçoit l'autorisation du contrôlleur *CTRLP* via l'interface **OKControlleurB** et celle du pont *P* via l'interface **OKPont**, il peut traverser le pont, en passant à l'état **DansPontB**. Ensuite, quand il sort, il marque la place d'interface **AuRevoirControlleurB** et aussi passe à l'état **FiniB** par la transition *sortPontB*.

La sécurité est garantie car, pour passer à l'état **AttenteAutorisationB** le véhicule doit d'abord communiquer avec le contrôlleur *CTRLP*. Pour entrer, il doit avoir l'autorisation du contrôlleur *CTRLP* et consommer le jeton dans la place d'interface **OKPont**, cette place modélise la consommation de la capacité du pont *P*. Enfin, il signale sa sortie du pont au contrôlleur *CTRLP* en mettant un jeton dans la place **AuRevoirControlleurB**.

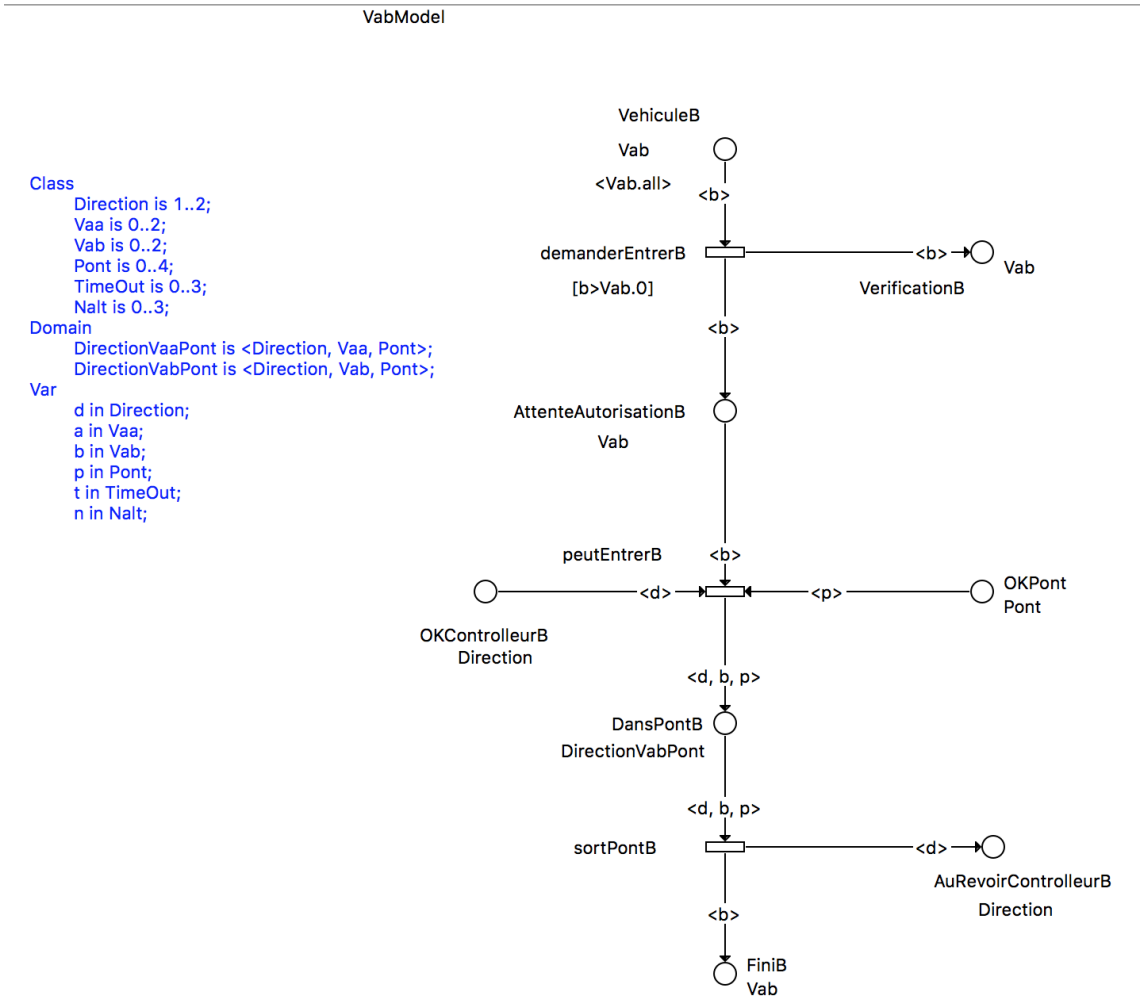

FIGURE 2.2 – Le composant modélise le véhicule automatisée B.

2.4 Question 1.4

Le comportement d'un pont est simple, il notifie sa capacité au contrôlleur *CTRLP* et diminue si consommé par un véhicule. Donc, il contient 3 places, dont 2 interfaces et une transition.

La transition *notifieCapacite* marque 2 interfaces **CapaciteControlleur** et **OKPont** par 2 jetons, 1 pour chaque.

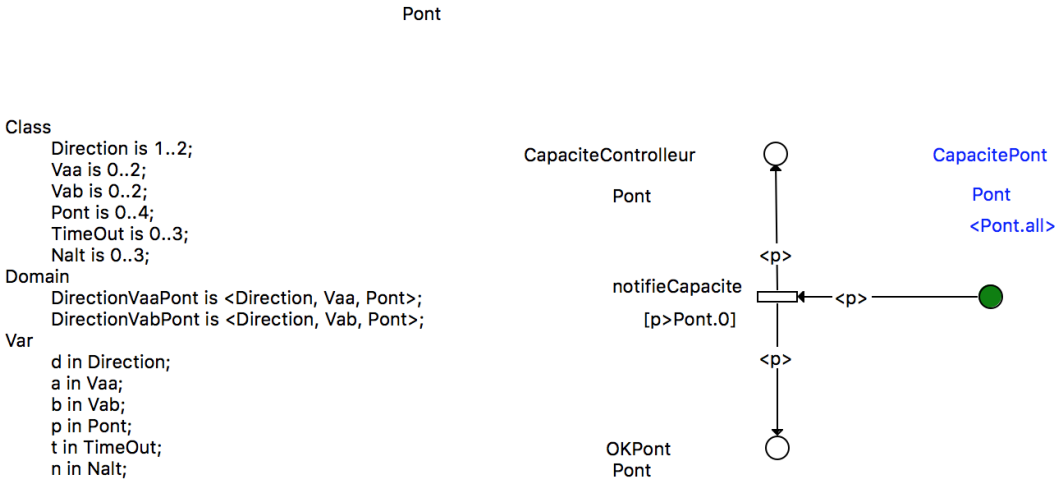

FIGURE 2.3 – Le composant modélise le pont.

2.5 Question 1.5

Ce composant contient 12 places dont 7 interfaces, et 6 transitions.

Lorsque le *CTRLP* reçoit une demande d'entrer d'une véhicule *Vaa* (idem, *Vab*) via l'interface **VerificationA** (idem. **VerificationB**), il vérifie le sens actuel (initilisé à 1, direction de A vers B) est correspondant à celui de *Vaa* (idem, *Vab*), le timeout restant, le nombre *Nalt* restant et la capacité *CapaP* restant du pont. Si ces conditions sont satisfaites, alors le contrôlleur *CTRLP* donne l'autorisation au véhicule *Vaa* (idem, *Vab*) en attente. Quand le véhicule sort du pont, le contrôlleur reçoit son signal.

Le timeout est modélisé par une place nommée *TimeOut* avec une seule transition *ecoule*. Le *TimeOut* décrémente une unité de temps automatiquement et indépendamment avec le système. Lors de l'expiration de *TimeOut*, il réinitialise le *Nalt*, lui même, et change la direction actuelle. Idem pour le *Nalt*, si $Nalt = 0$, il réinitialise le *TimeOut*, *Nalt* et change la direction.

Quand la capacité du pont est à 0, alors aucun véhicule ne peut utiliser ce pont.

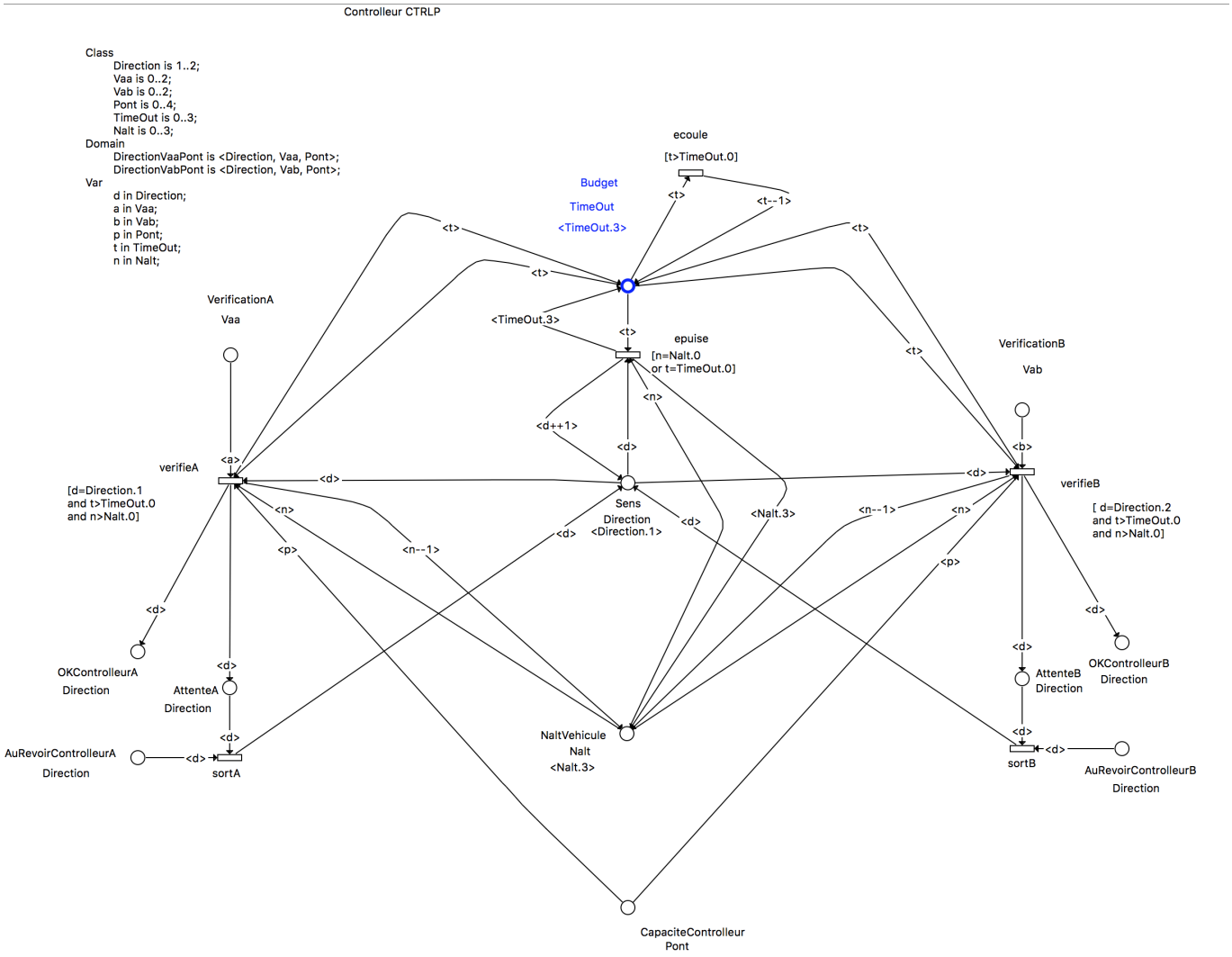

FIGURE 2.4 – Le composant modélise le pont.

2.6 Question 1.6

Cette assemblage contient 22 places, 13 transitions et 55 arcs.

Les propriétés attendues sont vérifiées dans chaque composant individuel, donc elles sont vérifiées lorsqu'on les rassemble.

Un véhicule quelconque veut traverser le pont, d'abord, il communique avec le contrôlleur, dès qu'il reçoit son autorisation, il peut avancer. A la sortie, il notifie le contrôlleur sa sortie. Le contrôlleur donne l'accès au pont si et seulement si le timeout, le nombre de véhicules d'un même type reste et la capacité du pont restent suffisants. Enfin de changement la direction, une des conditions doit être satifaite : soit l'expiration de timeout, soit le nombre de véhicules d'un même type sont passés sur le pont.

Une fois que la capacité du pont a été épuisée, aucun véhicule ne peut trvaverser le pont.

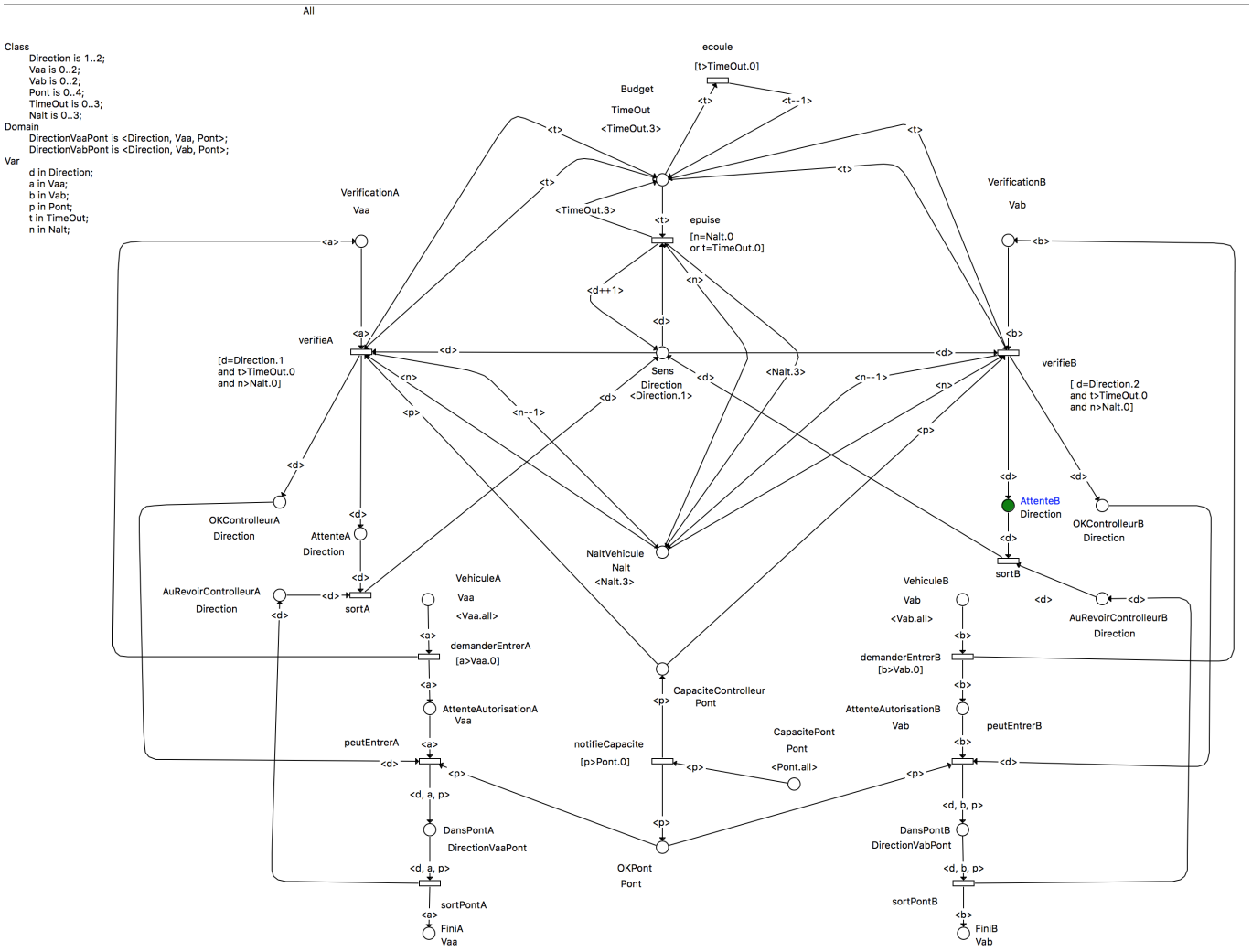

FIGURE 2.5 – Assemblage.

2.7 Question 1.7

Paramètres

Pour faciliter des vérifications car ma machine prend un temps énorme pour résoudre les formules. J'ai choisi :

— $N_{Vaa} = 2$
— $N_{Vab} = 2$
— $Capa_p = 4$
— $N_{Alt} = 3$

$Capa_p$ doit être supérieur au nombre total de véhicules, car sinon la propriété P2 ne peut être vérifiée.

Vérification par CosyVerif4PN

Pour vérifier la propriété P1, *"Il n'y a pas de collision (i.e. deux véhicules circulants en sens inverse) sur le pont."*, j'ai utilisé les formules suivantes :

1. *query node ((card(DansPontA) + card(DansPontB)) == 2)*. Le résultat est zéro chemin. (Fichier P1_1.txt)
2. *query node ((DansPontA == <.1.> and DansPontB == <.1.>)*
   or (DansPontA == <.1.> and DansPontB == <.2.>)). Le résultat est zéro chemin. (Fichier P1_2.txt)
3. *query node ((card(DansPontA) == 1) and (card(DansPontB) == 1))*. Le résultat est zéro chemin. (Fichier P1_3.txt)

Pour vérifier la propriété P2, *"Un véhicule qui arrive est certain de passer sur le pont à l'issue d'une durée bornée."*, j'ai utilisé les formules suivantes :

1. *query node (AG(implies(AttenteAutorisationA==<.1.>, AF(DansPontA :field[1]==<.1.>))))*. Le résultat est plusieurs chemins. (Fichier P2_1.txt)

Quelques formules pour les "autres propriétés triviales attendues" :

1. *query node ((card(VehiculeA) + card(AttenteAutorisationA) + card(DansPontA) + card(FiniA)) != 3)* : Le nombre de véhicules *Vaa* reste constant. (Fichier P3_1.txt et pour *Vab* P3_1Bis.txt)
2. *query node (card(CapacitePont) > 5)* : La capacité du pont $Capa_p$ est constante.(Fichier P3_2.txt)
3. *query node (card(Budget) > 1)* : Il y a au plus 1 jeton dans la place **Budget** à la fois.(Fichier P3_3.txt)
4. *query node (card(Sens) > 1)* : Il y a au plus 1 jeton dans la place **Sens** à la fois.(Fichier P3_4.txt)
5. *query node (card(NaltVehicule) > 1)* : Il y a au plus 1 jeton dans la place **NaltVehicule** à la fois.(Fichier P3_5.txt)

2.8 Question 1.8

Ce problème ressemble comme l'exclusion mutuelle. Le contrôlleur est comme un serveur central, c'est lui qui gère les accès au pont. Il faut spécifier un protocol de communication entre le contrôlleur et les véhicules. Ensuite, il faut avoir des capteurs sur le pont pour calculer sa capacité restante. En plus, il faut assurer que les véhicules ne passent pas sur le pont sans autorisation du contrôlleur.

En terme d'algorithmique répartie, ce modèle est comme le système client-serveur. Le timeout et le nombre de véhicules d'un même type sont bornés, le serveur peut gérer 2 files FIFO de demandes, 1 pour côté A et 1 pour côté B. La direction est changée régulièrement en un temps borné grâce au timeout et au nombre de véhicules d'un même type. Chaque véhicule qui sorte du pont doit signaler le contrôlleur. Donc, les 2 propriétés fondamentales du système sont vérifiées. Lorsque la capacité du pont atteint, le contrôlleur bloque tous les véhicules quelque soit leurs types.