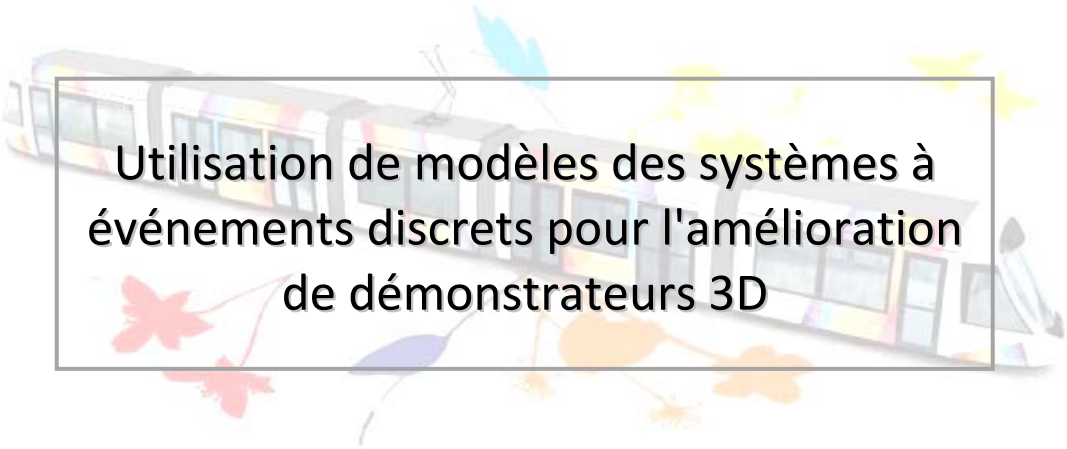
Rapport de stage du Master 2 Recherche



Utilisation de modèles des systèmes à événements discrets pour l'amélioration de démonstrateurs 3D

Encadrant universitaire : **Sébastien LAHAYE**
sebastien.lahaye@istia.univ-angers.fr

Stage réalisé par : **Tony DOAT**
tonydoat@gmail.com

Etudiant en **M**aster **2 IAIE** et **M**aster **2 SDS**

Ingénierie **A**utomatique et **I**nformatique d'**E**ntreprise

Systèmes **D**ynamiques et **S**ignaux

Remerciements

Tout d'abord, je tiens à transmettre mes remerciements à **M. Jean-Louis Boimond** et **M. Jean-Louis Ferrier**, respectivement *nouveau et ancien Directeur du Laboratoire LISA*, mais aussi *professeurs* de mes formations, pour m'avoir accueilli au sein du Laboratoire.

Ensuite, je tiens à remercier **M. Bertrand Vigouroux**, *professeur et responsable de la formation Master2 Systèmes Dynamiques et Signaux*, pour m'avoir accepté dans la formation et ainsi m'avoir permis de découvrir le monde de la Recherche ainsi que le domaine du traitement d'images.

Puis, je tiens à remercier grandement **M. Sébastien Lahaye** *responsable du stage et maître de conférences,* pour m'avoir proposé ce sujet liant théorique et pratique. Je le remercie également pour m'avoir accordé autant de temps et d'attention durant mon travail.

Mais aussi, **M. Erwan Mahe**, *Directeur Technique de la société Artefacto* pour avoir proposé ce stage, pour sa disponibilité ainsi que pour m'accueil au sein de l'entreprise afin d'y réaliser la seconde partie de ce stage.

Merci également à **Mme. Trichet**, *Ingénieur projet attaché au tramway de la ville d'Angers*, pour m'avoir reçu et m'avoir transmis les informations techniques nécessaires pour faire une simulation plus réaliste du tramway.

Enfin, je remercie Doctorants et Etudiants attachés au LISA pour l'ambiance et le temps qu'ils m'ont accordé.

Sommaire

Tables

I. Introduction

Afin de valider mon année de double inscription en Master 2 Recherche Système Dynamique et Signaux (M2R SDS) et Master 2 Professionnel Ingénierie Automatique et Informatique d'Entreprise (M2P IAIE), j'effectue un stage d'une durée de six mois (du 17 mars 2008 au 17 septembre 2008).

Dans le cadre de cette double inscription, mon stage se compose de deux parties:

- La première partie est relative à mon année de M2R SDS. Elle concerne la partie théorique de mon sujet de stage. Dans l'optique de me sensibiliser au domaine de la Recherche, la partie modélisation et simulation d'un système est réalisée en laboratoire.
- La seconde partie est relative à mon année de M2P IAIE. Elle concerne le développement d'un outil informatique. Cette partie, plus "professionnalisante", est réalisée au sein d'une entreprise.

Ce stage est co-encadré par M. Sébastien Lahaye[1] et M. Paul Richard[2] au sein du Laboratoire d'Ingénierie des Systèmes Automatisés[3] (LISA). Il s'effectue en partenariat avec l'entreprise Artefacto[4] située à Rennes. Cette entreprise est spécialisée dans la réalisation de maquettes 3D temps-réel, d'animations 3D, etc.

Lors d'une réunion entre M. S. Lahaye et M. P. Richard (pour le LISA) et M. Erwan MAHE[5] (responsable au sein de l'entreprise Artefacto), ce dernier a présenté les démonstrateurs développés par Artefacto pour les infrastructures de transport (Tramway, métro, etc.).

Dans ces applications, les véhicules virtuels se déplacent à vitesse constante sans prise en compte d'interactions entre eux et les autres éléments de l'environnement virtuel

[1] sebastien.lahaye@istia.univ-angers.fr
[2] paul.richard@univ-angers.fr
[3] http://www.istia.univ-angers.fr/LISA/
[4] http://www.artefacto.fr/
[5] e.mahe@artefacto.fr

(voitures, piétons, etc.). M. Erwan MAHE a fait part de l'intérêt d'Artefacto pour enrichir leurs démonstrateurs afin qu'ils simulent de façon réaliste les mouvements des véhicules.

Le LISA a depuis une quinzaine d'années une expertise reconnue dans la modélisation, l'analyse et la commande des systèmes à événements discrets. Une action sur l'analyse des systèmes de transport au sein de l'équipe "Modèles et systèmes dynamiques"[6] (MSD) a en particulier été développée depuis 2002.

A l'issue, de la réunion, il a été convenu de chercher à développer un partenariat en vue d'étudier les apports potentiels des modèles étudiés au LISA pour l'enrichissement des solutions développées par Artefacto. Il a notamment été choisi de prendre comme sujet d'étude le futur tramway d'Angers dont le démonstrateur est en cours de développement chez Artefacto.

Le stage a pour but de synthétiser un/des modèle(s) représentant de façon réaliste la dynamique des tramways sur la future ligne d'Angers. Il s'agit en outre de développer un simulateur s'interfaçant avec le démonstrateur 3D proposé par la société Artefacto.

La modélisation doit prendre en compte les diverses situations et acteurs pouvant influencer la dynamique du tramway (voitures, piétons, etc.). Au travers de ce modèle (et de la simulation), on cherche à augmenter, autant que possible, le réalisme des déplacements des différents acteurs.

Le programme de simulation devra générer un fichier au format XML contenant toutes les informations nécessaires à intégrer au sein du démonstrateur de la société Artefacto.

[6] http://www.istia.univ-angers.fr/LISA/orgfr.html/

Le travail réalisé durant ce stage se décompose selon les grandes lignes suivantes :

- Etude des démonstrateurs développés par Artefacto en C/C++, librairie OpenGL. Il s'agit, en particulier, d'identifier les interfaces (et leurs champs d'action) disponibles pour "piloter" les véhicules dans l'environnement virtuel.

- Etude des différents modèles pour les systèmes dynamiques à événements discrets (Réseaux de Petri temporisés, algèbre max-plus, automates à états finis temporisés, etc.) et leurs outils informatiques de simulation. Il s'agit en particulier de choisir le formalisme le mieux adapté pour répondre au cahier des charges et pour lequel l'outil de simulation peut s'interfacer avec les démonstrateurs.

- Synthèse d'un modèle (dans le formalisme choisi) à même de représenter de façon réaliste les déplacements des véhicules au sein de l'infrastructure (conformément à un tableau de marche réaliste), en prenant en compte les interactions entre les véhicules virtuels et les autres composants de l'environnement virtuel (voitures, piétons, etc.). Implémentation d'un simulateur et interfaçage avec le démonstrateur d'Artefacto.

I. Modèle de représentation

Afin de créer une simulation réaliste, il est essentiel sélectionner le modèle théorique le plus adapté pour notre application. Le choix de ce modèle nécessite de prendre en compte les divers comportements et les divers phénomènes induits du milieu urbain.

A. Première analyse

Précisons tout d'abord que dans cette démarche de choix du formalisme de modélisation (en vue de réaliser une simulation), nous avons élargi la classe de systèmes considérés. En effet, l'objectif à plus long terme est de disposer d'un outil permettant de simuler non seulement le fonctionnement d'une ligne de tramway, mais aussi le comportement de lignes de bus et de métro, l'évolution des voitures, des piétons et des cyclistes en milieu urbain, etc. Artéfacto développe effectivement des démonstrateurs pour ces divers types d'infrastructure, et l'idée est donc de disposer d'un outil suffisamment générique pour simuler tous ces systèmes.

En première approche, nous nous sommes intéressés aux situations que nous vivons chaque jour. En effet, en tant que piéton ou conducteur, nous passons une partie de nos journées à évoluer dans un environnement urbain. De ce fait nous pouvons lister intuitivement une série (non exhaustive) de situations à prendre en compte lorsque nous souhaitons réaliser la simulation de divers acteurs évoluant dans un environnement urbain :

- Croisement entre une route et un passage piéton
- Carrefour avec priorité à droite
- Carrefour à feux tricolores
- Deux entités ne peuvent se trouver exactement au même endroit au même moment (comme deux voitures sur une même voie, elles se suivent)
- Une voiture de tramway ne peut contenir une infinité de passagers.

B. Phénomènes considérés

A partir des situations précédemment citées et en se conformant à la définition classiquement utilisée pour les systèmes à événements discrets (SED) (voir par exemple [GC+95] §1.2), nous pouvons mettre en avant une série de phénomènes mis en jeu dans les systèmes considérés:

- *Concurrence* : Certains événements excluent l'apparition simultanée d'autres événements. Dans notre cas, citons par exemple les situations suivantes :
  - *Concurrence* entre un piéton et une voiture autour d'un passage clouté mais aussi entre une voiture et un tramway au niveau d'un passage à niveau (Partage de ressource).
  - *Concurrence* entre deux routes lorsque le piéton arrive à un croisement. Il a alors le choix de la direction à prendre.
- *Synchronisation* : L'accomplissement de certains événements nécessite la disponibilité simultanée de plusieurs ressources ou la vérification simultanée de plusieurs conditions. Par exemple, nous pouvons mentionner :
  - *Synchronisation* entre les feux tricolores d'un même carrefour.

- *Parallélisme* : Des événements peuvent se dérouler simultanément et indépendamment dans diverses parties du système.

C. Choix du modèle de représentation

Nous avons choisi de voir le système urbain comme un système à événements discrets (SED). D'une part, les phénomènes à prendre en compte dans sa modélisation ont été listés au §I-B (p4). D'autre part, en vue d'une simulation au sein d'un démonstrateur, nous devons prendre en compte le caractère temporisé du système, c'est-à-dire envisager un modèle temporisé pour celui-ci.

Assez naturellement, nous nous sommes intéressés au formalisme des réseaux de Petri (RdP) temporisés, qui est bien connu des chercheurs s'intéressant aux SED.

Ce choix est justifiable par plusieurs arguments :

- Les RdP constituent un outil graphique suffisamment intuitif pour aborder la modélisation des SED et exposer les résultats. Leur support mathématique permet, en outre, d'établir des résultats analytiques.
- Ce formalisme est suffisamment générique pour prendre en compte tous les phénomènes listés précédemment.
- La définition de sous classes de réseau de Petri permet de "catégoriser" facilement les SED.

Notons qu'en début de stage une attention a été portée à la sous-classe des RdP correspondant aux graphes d'événements temporisés (GET). Ces derniers ont été intensivement étudiés au LISA en tant que systèmes (max,+) linéaires[7] (cf. [LH+04]) avec des applications aux systèmes de transports (cf. [LH+06]). Cette classe, ne permettant pas de prendre en compte les phénomènes de *concurrence*, elle nous a rapidement paru inadaptée en vue de la modélisation et de la simulation du système urbain. Nous garderons toutefois en tête, qu'une restriction du modèle RdP du système urbain à un GET (en négligeant certains phénomènes) devrait permettre l'utilisation "d'outils max-plus" en vue d'analyses, voire de commandes du système urbain.

II. Introduction aux réseaux de Petri

Ce chapitre est une introduction succincte à la modélisation par réseaux de Petri. Son rôle est de fournir au lecteur les rappels utiles pour la suite du mémoire. Le lecteur désirant connaître plus en détail les réseaux de Petri pourra se reporter à l'ouvrage [RDHA+92].

A. Introduction au modèle

Carl Adam Petri est un mathématicien allemand contemporain qui a défini un outil mathématique très général permettant de décrire des relations existant entre des conditions et des événements, de modéliser le comportement de systèmes dynamiques à événements discrets. Ce travail a été effectué dans les années 1960-62.

Dés lors, ces réseaux ont donné lieu à de nombreuses recherches. Les réseaux de Petri ont été enrichis par divers chercheurs pour permettre des descriptions plus condensées et/ou

[7] http://www.istia.univ-angers.fr/LISA/FICHES/plaquette_LISA_dioide.pdf

faisant intervenir le facteur temps : réseaux de Petri temporisés, interprétés, stochastiques, colorés, continus et hybrides, etc.

Les réseaux de Petri présentent deux caractéristiques intéressantes. Premièrement ils permettent de modéliser (et de visualiser) des comportements mettant en jeu du parallélisme, de la synchronisation et de la concurrence. Deuxièmement, les résultats théoriques qui les concernent sont abondants ; les propriétés de ces réseaux ont été et sont encore très largement étudiés. Les principales propriétés seront abordées et présentées de façon assez intuitive.

Le réseau de Petri constitue un outil pour modéliser graphiquement la dynamique d'un système à événements discrets.

B. Notions de bases

1. Places, transitions et arcs:

Un réseau de Petri est d'abord un graphe "bi-parti". Car il est composé de deux sortes de nœuds, les *places* (représentées par des ronds) et les *transitions* (représentées par des barres ou des rectangles) reliés par des *arcs* (représentés par des flèches) (voir Figure 2, p6). Le nombre de places est *fini*, et non *nul*. Le nombre de transitions est également *fini* et non *nul*. Un arc est orienté. Il relie soit une place à une transition soit une transition à une place. Ainsi un réseau de Petri est composé par une alternance de places et de transitions sur un chemin formé d'arcs consécutifs. Et tout arc doit obligatoirement avoir un nœud à chacune des ses extrémités.

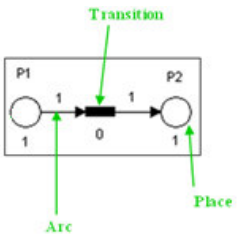


Figure 2: Place, transition, arc

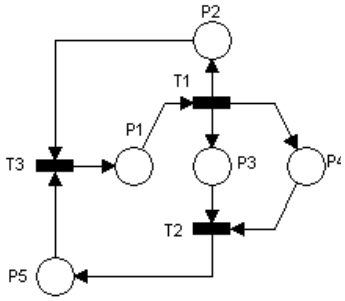


Figure 1: Réseau de Petri

La Figure 2 présente un réseau de Pétri comportant 5 places, 3 transitions et 10 arcs orientés. L'ensemble des places d'un réseau de Petri est noté P. Pour l'exemple ci-dessus nous avons donc : $P=\{P_1,P_2,P_3,P_4,P_5\}$. L'ensemble des transitions est noté T. Nous avons donc $T=\{T_1,T_2,T_3\}$. On dira que P_2 est en amont (ou est une entrée) de T_3 car l'arc est orienté de P_2 vers T_3. On dira que P_2 est en aval (ou est une sortie) de T_1 car l'arc est orienté de T_1 vers P_2.

Une transition sans place en entrée sera appelée *transition source*. Une transition sans place en sortie sera appelée *transition puits*.

2. Marquage

Une place peut être dans deux états : *marquée* ou *non marquée*. Le marquage, se fait par la présence ou non de "*jeton(s)*" (chacun représenté par un point noir). Une place possède un nombre entier de jeton(s) (positif ou nul). Le nombre de jeton(s) contenu(s) dans une place est noté $M(P_i)$ ou M_i.

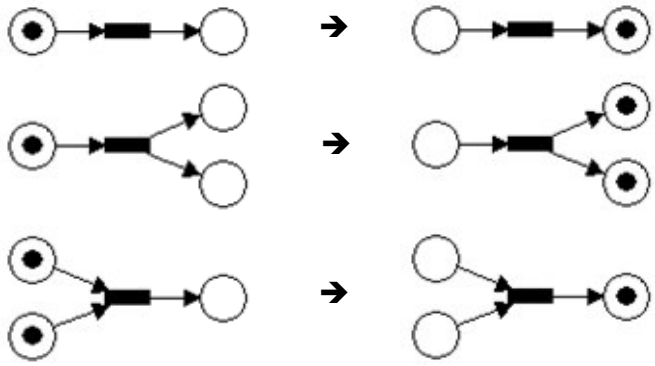
Chaque jeton circule dans le réseau de Petri selon des règles préétablies (définies ci-dessous). Le déplacement des jetons, et donc le changement du marquage des places,

symbolise l'évolution dynamique du système. Le marquage des places à l'instant initial est appelé *marquage initial*.

3. Règles d'évolution

Pour qu'une transition soit franchie, il faut que toutes les places situées en amont de la transition contiennent au moins un jeton (Figure 1). Dans ce cas la transition est dite franchissable (ou validée).

Si une transition est franchissable, elle est alors immédiatement franchie (ou tirée). Un franchissement consiste à enlever un jeton dans chacune des places amont et d'en rajouter un dans chacune des places aval (Figure 3-1 et Figure 3-2). Le tir d'une transition est invisible.



- 1 : Avant franchissement

- 2 : Après franchissement

Figure 3: Franchissement d'une transition

C. Classes de réseaux de Petri

Nous abordons seulement les classes de réseaux Petri utilisées dans le cadre de notre simulation d'environnement urbain.

1. Réseau de Petri généralisé

Un réseau de Petri généralisé est un RdP dans lequel des poids (nombre entier strictement positif) sont associés aux arcs.

La figure 4 représente un RdP dans lequel l'arc $P_1 \rightarrow T_1$ a un poids de 2 et l'arc $T_1 \rightarrow P_2$ a un poids de 4. Cela signifie que la transition T_1 sera franchissable si P_1 contient au moins deux jetons et le franchissement de T_1 provoquera l'ajout de quatre jetons dans P_2.

Figure 4: Réseau de Petri généralisé

2. Réseau de Petri à capacité

Il s'agit de RdP dans lequel des capacités (nombre entier strictement positif) sont associées aux places. Le franchissement d'une transition T_j en amont d'une place P_i dont la capacité est $Cap(P_i)$ n'est possible que si le franchissement de T_j entraîne un marquage de P_i, inférieur ou égal à $Cap(P_i)$.

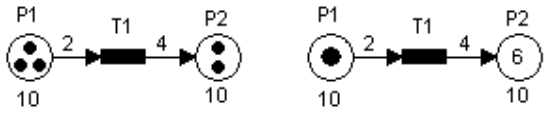


Figure 5: Réseau de Petri à capacité

La Figure 5 représente un RdP dans lequel les places P_1 et P_2 ont une capacité de dix jetons. Cela signifie que le franchissement de la transition T_1 est possible si P_2 peut recevoir au moins quatre jetons (son marquage courant doit être inférieur ou égal à 6).

3. Réseau de Petri à arc inhibiteur

Un arc inhibiteur est un arc orienté qui part d'une place P_i pour aboutir à une transition T_j. Son extrémité est marquée par un petit cercle (Figure 6). L'arc inhibiteur entre P_i et T_j signifie que la transition T_j n'est validée que si la place P_i ne contient aucune marque. Dans le cas d'un réseau de pétri généralisé, T_j n'est validée que si la place P_i contient un nombre de jetons inférieur au poids de l'arc inhibiteur.

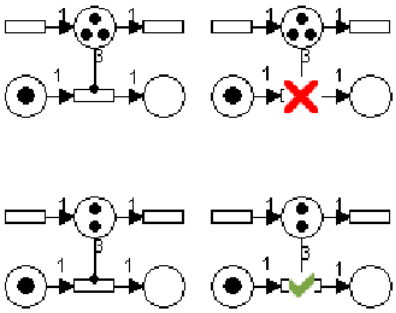


Figure 6: Réseau de Petri à arc inhibiteur

4. Réseau de Petri temporisé

Un réseau de Petri est dit temporisé lorsque des durées sont associées aux places (réseau *P-temporisé*) ou aux transitions (réseau *T-temporisé*).

Si une durée d est associée à une place P, tout jeton arrivant dans cette place devient indisponible pendant la durée d.

Si une durée d est associée à une transition T, lorsque la transition est validée, les jetons concernés sont réservés pendant une durée d et la transition T est tirée à l'issue de ce délai.

Nota : Les approches RdP P-temporisée et T-temporisées sont équivalentes.

Pour notre application, nous avons choisi d'utiliser les réseaux de Petri T-temporisé (Figure 7)

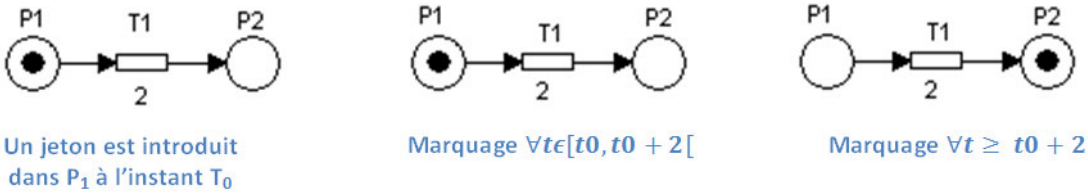


Figure 7: Réseau de Petri T-temporisé

D. Modélisation de phénomènes.

1. Concurrence

- 1 : Fourniture — - 2 Consommation

Figure 8: Concurrence

Certains réseaux de Petri contiennent un ou plusieurs cas de concurrence (le *ou* logique). Ce phénomène est présent dans deux cas:

- Plusieurs transitions sont franchissables pour une même place aval (concurrence à la "fourniture" de jetons dans une place) (Figure 8-1).
- Plusieurs transitions sont franchissables pour une même place amont (concurrence à la "consommation" de jetons dans une place) (Figure 8-2).

En particulier, le partage d'une ressource peut ainsi être modélisé.

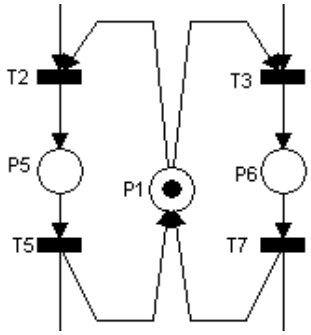
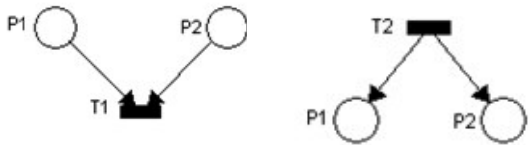


Figure 9: Partage de ressources

La place P_1 de la Figure 9 modélise la disponibilité d'une ressource qui peut être utilisée par la partie gauche à partir du franchissement de T_2 jusqu'au franchissement de T_5 ou, similairement, par la partie droite, mais pas par les deux simultanément.

Notons que par la suite (cf. §III.A, p11), les phénomènes de concurrence seront parfois modélisés de façon plus condensée à l'aide d'arcs inhibiteurs.

2. Synchronisation



- 2 : Consommation — - 1 : Production

Figure 10: Synchronisation

Le cas de synchronisation (le *et* logique) arrive dans deux situations :

- Lorsque plusieurs arcs convergent sur une même transition (Figure 10-1). Ce cas est appelé synchronisation dans consommation de jeton dans plusieurs places.
- Lorsque plusieurs arcs divergent à partir de la même transition (Figure 10-2). Ce cas est appelé synchronisation dans la fourniture de jetons à plusieurs places.

3. Parallélisme

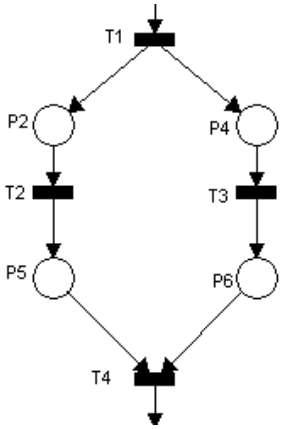


Figure 11: Parallélisme

La Figure 11 représente la modélisation d'un phénomène de parallélisme. On voit clairement qu'après le franchissement de la transition T_1, et jusqu'au franchissement de la transition T_4, on a des évolutions en parallèles, de la place P_2 à la place P_5 d'une part et de la place P_4 à la place P_6 d'autre part.

E. Conclusion

L'approche par les réseaux de Petri permet de prendre en compte les phénomènes de synchronisation, concurrence et parallélisme. Ainsi, avec ce modèle, nous pensons pouvoir formaliser toutes les situations existantes en milieu urbain.

Ceci couplé à l'aspect graphique qui en font un outil facile à comprendre, font de lui un outil théorique très intéressant pour notre application.

III. Modélisation et simulation

A présent, nous avons validé l'utilisation du formalisme des réseaux de Petri. Dans le chapitre précédent, nous avons vu qu'il permettait d'interpréter tous les phénomènes mis à jour existant dans le milieu urbain.

A. Principes de modélisation adoptés

Ce paragraphe présente la correspondance entre divers éléments d'un RdP (place, transition, jeton, etc.) avec des éléments de situation urbaine réelle (marche, arrêt, etc.).

Pour notre simulation, nous avons commencé par définir une place comme étant une zone géographique (aire) indivisible (unité de distance, tronçon). Soulignons que le choix d'utiliser des RdP à capacités permet directement de traduire les contraintes d'occupation des tronçons : la capacité associée à la place, représentant un tronçon, traduit le nombre maximum d'acteurs pouvant simultanément se trouver sur ce tronçon.

Ainsi, la zone à modéliser sera découpée en une succession d'unités de distance. La taille de cette unité sera définie dans une étape postérieure car sa taille influe sur le nombre de croisement de chemins possibles (complexité du réseau) et sur la quantité de places à simuler (temps d'exécution).

Il sera donc nécessaire de trouver un optimum entre la précision du tracé (qui sera proportionnelle au nombre de places) et le temps d'exécution du programme (croissant avec le nombre de places)[8].

La temporisation associée à une transition correspond au temps de déplacement de l'entité sur le tronçon modélisé par la place en amont. Le franchissement d'une transition symbolise le passage de l'acteur d'un tronçon à un autre.

Le jeton contenu ou non dans une place identifie la présence ou non de l'acteur dans une zone géographique.

Les convergences ou les divergences correspondent aux choix d'itinéraires d'un acteur. Par exemple, le cas de la concurrence à la *consommation* de jetons dans une place correspond au choix pour l'acteur de s'engager sur la route de gauche ou celle de droite.

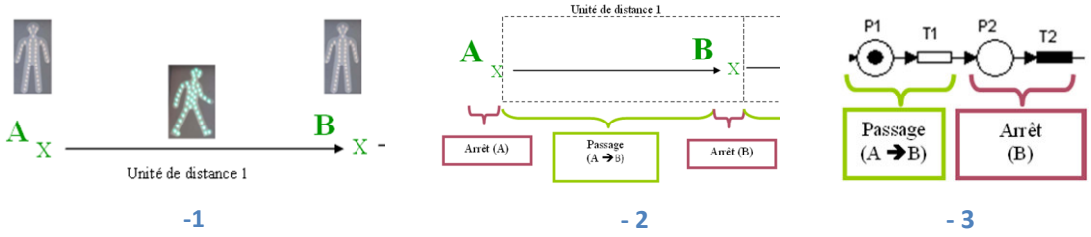
Ainsi les diverses règles d'évolution des réseaux de Petri permettent de reproduire le comportement d'acteurs dans le monde virtuel.

Voici quelques exemples illustrés pour des situations de bases :

Les cas présentés ci-dessous sont simplifiées à un seul sens de déplacement afin de rester sur des schémas facilement compréhensibles. L'extension de ces modèles aux voies à deux sens est une superposition du même modèle pivoté de 180°. Les nouvelles liaisons entre les réseaux seraient alors des arcs inhibiteurs permettant de coordonner le tout.

[8] Notons que dans un premier temps, ce programme de simulation sera exécuté en différé. C'est-à-dire que les données de sortie de la simulation seront intégrées au simulateur de la société Artefacto au travers d'un fichier type XML.

- Déplacement d'un piéton entre deux points



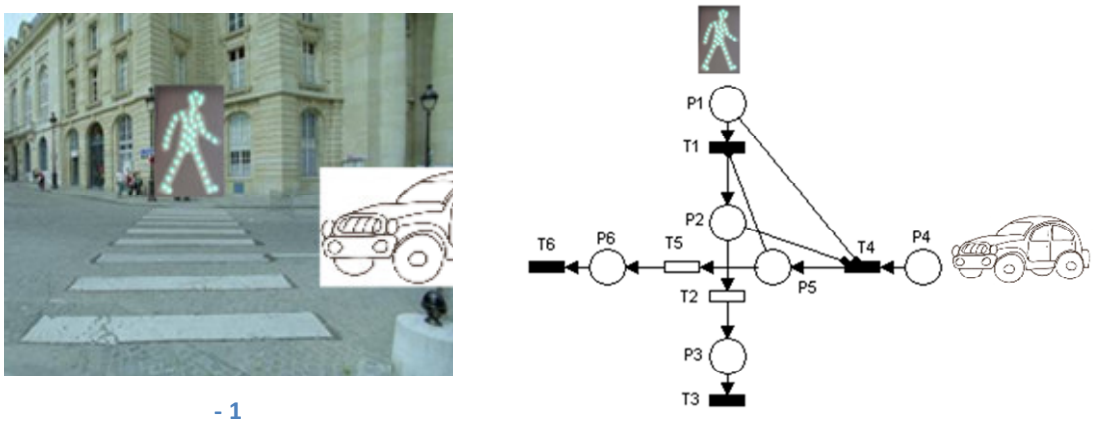
-1 - 2 - 3

Figure 12: Déplacement entre deux points

Lorsque le piéton se déplace d'un point A à un point B dans une zone géographique (Figure 12-1), la place associée devient marquée (ici P_1 Figure 12-3). Le temps de déplacement du piéton dans la zone est représenté par la temporisation associée à la transition T_1 (Figure 12-3). Ainsi la place P1 restera marquée jusqu'à ce que le temps d'attente soit écoulé, c'est-à-dire jusqu'à ce que le piéton arrive au point B. Une fois arrivé, le piéton s'arrête au point B (Figure 12-2). La durée de cet arrêt est soit nulle (T_2 est une transition instantanée), c'est-à-dire que le piéton continue sa route dans une autre zone, soit il s'arrête en attendant que la zone suivante se libère (le tir de T_2 peut être conditionné par les capacités des places en aval).

La marche d'un piéton sur un trottoir peut donc être modélisée en juxtaposant une série de modèles identiques à celui de la Figure 12-3.

- Passage piéton



- 1 - 2

Figure 13: Passage piéton (1 sens)

Lors de l'étude des passages piétons (Figure 13-1), nous avons mis à jour le phénomène de concurrence. En effet, deux acteurs (voitures, piétons) se partagent une même ressource (route). Afin de représenter ce cas en réseaux de Petri, tout en restant sur une forme très proche de la réalité, nous avons choisi d'utiliser la classe particulière des RdP à arcs inhibiteur (cf. §II.C.3, p8).

La Figure 13-2 présente la formalisation du cas courant. Un jeton dans les places P_1 et P_4 représente respectivement la présence d'un piéton et d'une voiture. L'arc inhibiteur orienté

de P_1 vers T_4 donne la priorité au piéton. En effet, si un jeton est présent en P_1, T_4 ne pourra pas être tirée.

L'arc inhibiteur P2 → T4 permet d'empêcher qu'une voiture ne puisse passer pendant que le piéton traverse la route.

L'arc inhibiteur P5 → T1 permet d'empêcher qu'un piéton ne puisse passer pendant qu'une voiture passe sur le passage piéton.

Nota : Nous sommes ici dans le cas d'une ressource partagée, de ce fait, ces deux arcs inhibiteurs pourraient être remplacés par une structure comparable à la Figure 9.

- Priorité à droite

Le cas de la priorité à droite est identique au cas du passage piéton. Le schéma formalisé est donc exactement le même. Le piéton est alors remplacé par la voiture prioritaire (Figure 13-2). Le cas de la priorité à droite peut représenter deux voitures dans un carrefour à priorité à droite et la priorité d'un tramway par rapport aux voitures et aux piétons.

- Carrefour à feux

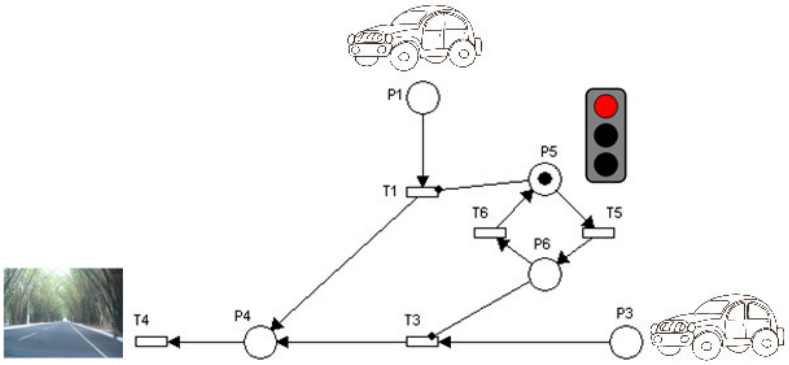


Figure 14: Carrefour à feux simplifié

La Figure 14 présente le carrefour à feux formalisé par un réseau de Petri. Les places P_1 et P_3 représentent le point d'arrivé des voitures et P_4 représente le point de sortie de carrefour.

Les feux tricolores (ici deux), sont représentés par les place P_5, P_6 et par les transitions T_5, T_6. La présence d'un jeton dans la place P_5 correspond au feu rouge pour la voiture en P_1 et au feu vert pour la voiture située en P_3. Inversement, un jeton situé dans la place P_3 correspond au feu vert pour la voiture en P_1 et au feu rouge pour la voiture située en P_3. Nous remarquerons que le chemin $P_6 \rightarrow T_6 \rightarrow P_5 \rightarrow T_5$ est un circuit, ce qui traduit que les cycles des feux se déroulent "librement" (ils ne sont notamment pas conditionnés par la présence de véhicules).

B. Etude des solutions logicielles existantes pour la simulation

1. Critères

La formalisation des situations urbaines confirme que l'approche par réseaux de Petri est convaincante. A présent il est nécessaire de trouver un outil logiciel simple et efficace permettant de créer les divers schémas formalisés en vue de pouvoir facilement les appliquer.

Il existe un très grand nombre de situations urbaines. Les formaliser toutes n'étant pas imaginable, il est dont important que la solution définie pour le projet actuel permette d'enrichir le nombre de situations formalisées. Un critère important du logiciel est donc la facilité à l'utiliser afin qu'il soit accessible à des personnes non expertes du domaine de l'informatique.

Le programme devra également permettre de créer les réseaux de Petri dérivés tels que les RdP à arcs inhibiteurs, à capacités, temporisés.

Un dernier aspect très important est la possibilité de récupérer le résultat de la simulation du réseau. En effet, rappelons que le but de générer un fichier de type XML contenant le marquage des places à chaque instant.

2. Solutions logicielles

La recherche des solutions logicielles a été réalisée par Internet. Il existe un nombre relativement important de sites dédiés aux réseaux de Petri (contenant des cours, logiciels, nouvelles, etc.). Il existe également un site tenu par "la communauté internationale des réseaux de Petri[9] ". Ce site contient une grande source d'informations et de services en ligne avec notamment des bibliographies, des informations sur les conférences internationales et une base de données contenant des logiciels de simulation de réseaux de Petri[10].

A partir de toutes les solutions trouvées, voici l'analyse des plus intéressantes de ces solutions:

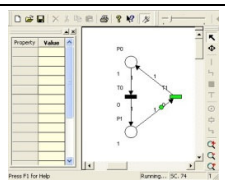
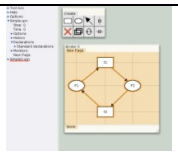
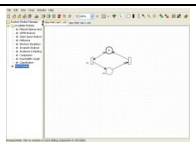
Nom	Aperçu	Avantages	Désavantages
hpsim1_1:		Très simple Animation des jetons Possibilité d'utilisation des arcs inhibiteurs Génération du graphe des marquages	Le résultat de la simulation est parfois erroné
CPNtools:		utilisé par plus de 750 organisations (~200 entreprises) Possibilité d'utilisation des arcs inhibiteurs	très difficile à prendre en main
pipe25_rc5		Possibilité d'utilisation des arcs inhibiteurs Exportation du réseau sous format XLM Multi-plate-forme (Java)	Pas d'exportation du résultat de la simulation

Tableau 1: Solution logicielle de simulation de Rdp les plus intéressantes

Voici, à titre d'informations, les solutions logicielles retenues après un premier filtrage. Bien qu'elles ne correspondent pas à nos besoins, nous les avons citées ici pour les lecteurs à la recherche de ce type de solutions. De plus, dans l'avenir, ces logiciels seront probablement améliorés.

[9] http://www.informatik.uni-hamburg.de/TGI/PetriNets/ (en anglais)
[10] http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/search.html (en anglais)

Voici une série de logiciels de simulation des RdP :

DaNAMiCS	pipe25	tina-2.9.0-i386
jarp-1.1.16	Pnk2.2	VisualPetri
PetriLLD	SIPN-Editor_10_03	
PetriParC	Sirphyco	

Tableau 2: Solutions logicielles de simulation de Rdp

3. Vers le développement de l'algorithme

Aucune des solutions logicielles ne répondaient parfaitement à notre problématique. La plus proche est "hpsim", logiciel avec lequel ont été créées toutes les figures de ce document. Cependant bien que graphiquement ce programme ait un grand intérêt, pour ce qui est de sa simulation, la performance est plus limitée. En effet, parfois, un décalage apparait entre la valeur du temps et le marquage des places (pour ce même temps).

Etant donné que la solution envisagée va être utilisée par une société (Artefacto), il peut y avoir des problèmes de copyright en utilisant une solution existante. De plus, dans les perspectives d'évolution du projet, il a été émis l'idée d'introduire le module de simulation des réseaux de Petri directement dans le simulateur de la société. Ainsi l'environnement urbain du simulateur pourra être modifié (position et nombre d'acteurs) et les modifications seront prises en compte en temps réel.

A la vue de ces points importants, le choix de l'outil a convergé vers le développement de l'algorithme de simulation des réseaux de Petri.

C. Développement d'une solution de simulation des RdP

La solution de simulation de réseaux de Petri présentée ci-dessous a été développée dans l'environnement Scilab (voir annexe 2). Dans cet environnement, nous avons pu facilement mettre au point et valider l'algorithme de simulation. Un portage en C++ est envisagé dans la suite du stage pour une intégration au sein d'un démonstrateur.

Cet algorithme est comparable à celui présenté dans [ELM+02] . La principale différence réside dans le fait que l'algorithme dans [ELM+02] s'appuie sur une historisation des dates de tir des transitions alors que notre solution utilise une mémorisation des temps de séjour de chaque jeton dans les places.

4. Les données

Données d'entrée du programme :

Vocable utilisé :
P_{places} fait référence au nombre de places du réseau
$T_{transitions}$ fait référence au nombre de transitions du réseau
Une matrice (P_{places}, $T_{transitions}$) est une matrice dont le nombre de lignes est égal au nombre de places et le nombre de colonne est égal au nombre de transitions.
Minh (P_i, T_j) : fait référence à la valeur située à l'intersection de la ligne numéro P_i et de la colonne numéro T_j dans la matrice Minh.

Avant de pouvoir simuler les RdP, le programme a besoin de données d'entrée. Voici la liste et le rôle de ces données d'entrées :

- Matrice d'incidence (W)

W est une matrice (P_{places}, $T_{transitions}$) qui définit complètement le RdP. Elle est indépendante du marquage des places. Cette matrice est liée aux deux matrice W+ et W- sous la relation suivante:

$W = W^+ - W^-$

Avec:

W^+ (Wplus): indique combien de jetons sont ajoutés dans les places suite aux tirs de transitions. W^+ (P_i, T_j) correspond au poids de l'arc fléché de la transition T_i à la place P_j.

W^- (Wmoins): indique combien de jetons sont consommés dans les places suite aux tirs de transitions. W^- (P_i, T_j) correspond au poids de l'arc fléché de la place P_i à la transition T_j.

- Matrice inhibiteur (Minh)

Minh est une matrice (P_{places}, $T_{transitions}$) qui contient la configuration des arcs inhibiteurs. Par exemple, Minh(1,2) signifie que le réseau contient arc inhibiteur allant de la place P_1 à la transition T_2.

Minh (P_i, T_j) est égale au poids de l'arc inhibiteur orienté de la place P_i à la transition T_j.

- Matrice de capacité (McapacitéP)

McapacitéP est une matrice colonne (P_{places}) qui contient les capacités de chaque place.

- Matrice de marquage initial (MposJetonP)

MposJetonP est une matrice colonne (P_{places}) qui contient le marquage initial.

- *Matrice de temporisation* (Mtps)

Mtps est une matrice ligne ($T_{transitions}$) qui contient la valeur de la temporisation associée à chaque transition.

- Temps total (TpsTotCycle)

Contient la durée de la simulation.

- Nombre total de cycle (NbTotCycle)

Contient le nombre d'itérations pour la simulation.

Données internes du programme:

- Matrice des transitions franchissables (TabXFranch)

TabXFranch est une matrice ligne ($T_{transitions}$) qui contient l'information des transitions franchissables sous forme binaire pour l'itération en cours itération.

- Matrice des temps d'arrivées des jetons (MTpsSejourJeton)

MTpsSejourJeton est une matrice (P_{places}, N) avec N = max(Mtps). MTpsSejourJeton sert notamment à connaître le temps de présence de chaque jeton dans les places. Cette matrice est notamment utilisée pour savoir si un jeton est resté suffisamment longtemps dans une place en vue du franchissement de transition(s) en aval.

Données de sortie du programme:

- Matrice de franchissement des transitions (TabSortieX)

TabSortieX est une matrice de dimension (NbTotCycle , $T_{transitions}$) qui contient le nombre de franchissement d'une transition à chaque itération.

- Matrice de marquage des places (TabSortieP)

est une matrice est une matrice de dimension (P_{places} , NbTotCycle) qui contient le marquage (nombre de jetons contenus) de toutes les places à chaque itération.

5. Description de l'algorithme

A partir des données d'entrée, l'algorithme de simulation détermine le marquage des places à tout instant en réalisant successivement les étapes suivantes :

La Figure 15 présente l'ordre d'exécution des diverses fonctions du programme. Nous allons présenter les points clés de l'algorithme.

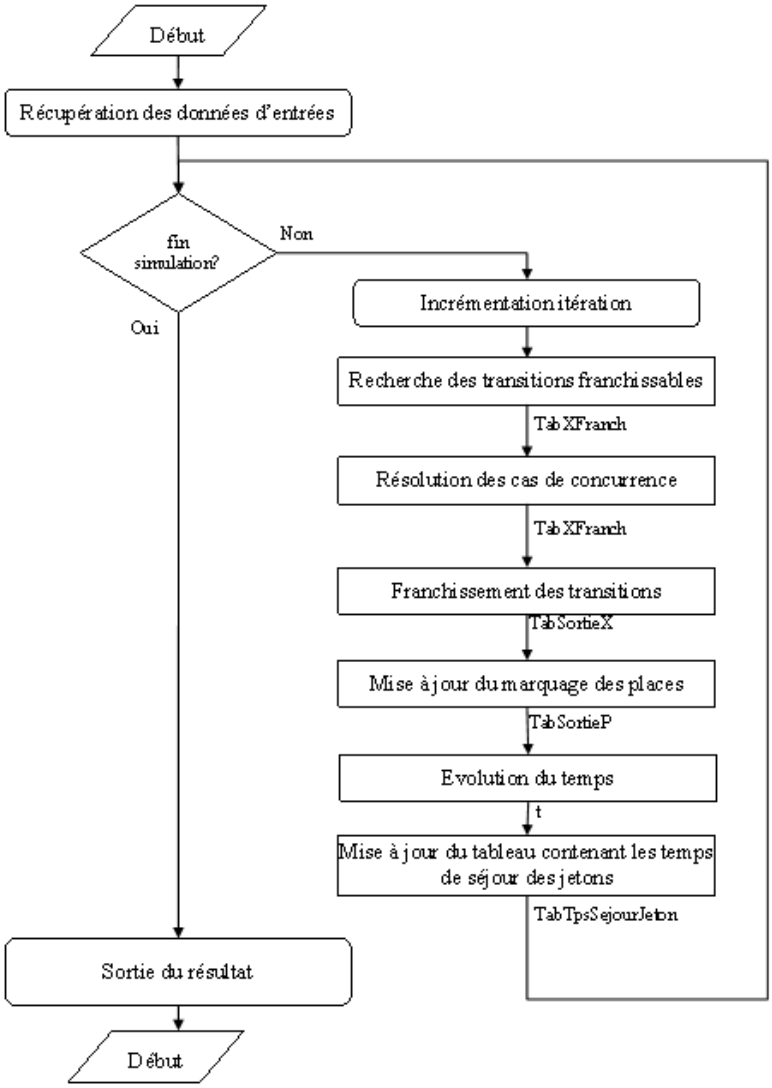


Figure 15: Arbre de programmation

Fonction recherche des transitions franchissables

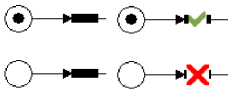
Cette fonction permet au programme de déterminer quelles sont les transitions franchissables. Pour ce faire il utilise les données d'entrée suivantes : McapacitéP, TabSortieP, Mtps,Wplus, Wmoins, Minh. A partir de ces données, le programme stocke l'information des transitions franchissables dans TabXFranch.

Une série de contraintes sont vérifiées pour déterminer si une transition est franchissable ou non. Afin de réduire le nombre de ligne du programme, nous avons choisi d'utiliser la logique inverse. C'est-à-dire que nous commençons par considérer que la transition est franchissable puis nous regardons si les règles d'évolutions sont validées. Dans le cas contraire la transition est déclarée comme non franchissable.

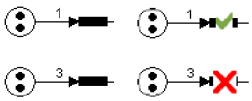
Voici les conditions de franchissement vérifiées :

Prédécesseur

Pour qu'une transition soit franchissable, il faut tout d'abord que chaque place prédécesseur contienne un nombre de jetons supérieur ou égal au poids de l'arc les reliant (Dessin 1, Dessin 2).



Dessin 1: Condition sur les prédécesseurs (cas simple)



Dessin 2: Condition sur les prédécesseurs (cas complexe)

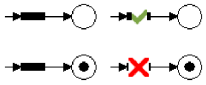
```
//Condition prédécesseurs
Pr tout p€P,
{
   If( Wmoins(:,t)>0) then
   {
         if  TabSortieP(p,n-1)< Wmoins(p,t)  then
         {
            %BoolFranchissable=false;
         }
   }
}
```

Illustration 1: Code condition sur les prédécesseurs

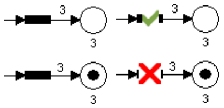
Successeurs

Il est pris pour convention que les places du réseau de Petri ont une capacité limitée (elles ne peuvent contenir plus de jetons que ce qui est défini dans McapacitéP).

Pour qu'une transition soit franchissable, il faut alors que chaque place successeur contienne un nombre de jetons inférieur à la capacité moins le poids de l'arc. Autrement dit, il faut que le nombre de jetons que peut accueillir la place soit au moins égal au poids de l'arc.



Dessin 3: Condition sur les successeurs (cas simple)



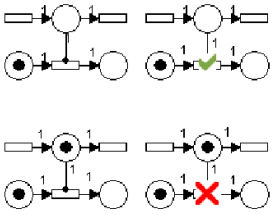
Dessin 4: Condition sur les successeurs (cas complexe)

```
//Condition successeurs
Pr tout p€P,
{
   If( Wplus(:,t)>0) then
   {
         if  TabSortieP(p,n-1)+ Wplus(p,t)> McapacitéP(p) then
         {
            %BoolFranchissable=false;
         }
   }
}
```

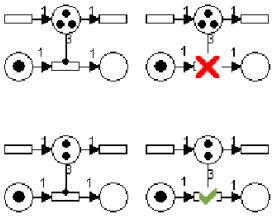
Illustration 2: Code: condition sur les successeurs

Inhibiteur

Pour qu'une transition reliée à un arc inhibiteur puisse être franchissable, il est nécessaire que la place en amont de cet arc inhibiteur contienne un nombre de jetons inférieur au poids de l'arc inhibiteur.



Dessin 5: Condition sur les inhibiteurs (cas simple)



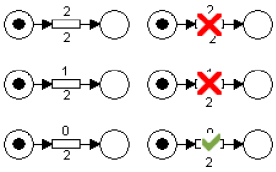
Dessin 6: Condition sur les inhibiteurs (cas complexe)

```
//Condition inhibiteur
Pr tout p€P,
{
   If( Minhibiteurs(p,t)>0) then
   {
        if  TabSortieP(p,n-1)= Minhibiteurs (p,t) then
        {
          %BoolFranchissable=false;
        }
   }
}
```

Illustration 3: Code: condition sur les arcs inhibiteurs

Condition sur les temps d'attentes

La transition T_j est franchissable si la condition sur les prédécesseurs est respectée (voir plus haut : *prédécesseurs*) et si les jetons concernés sont présent depuis une durée supérieure ou égale à $Mtps(T_i)$ (Dessin 7).



Dessin 7: Condition sur les temps d'attentes

```
//Condition temps d'attente
Pr tout p€P,
{
        If ( Wmoins(:,i)>0) then
        {
                //recherche du temps associé à la transition courante
                TpsX= TabTpsX(t) ;
                //on cherche le nombre de jetons qui ont suffisamment attendus
                NbJetons= MTpsSejourJeton (p,TpsX +1)+ MTpsSejourJeton (p,TpsX +2)
                +… + MTpsSejourJeton (p,TpsX +n) ;    Avec n=nombre de colonnes de
                MTpsSejourJeton ()

                if NbJetons<Wmoins(p,t) then
                {
                        %BoolFranchissable=false;
                }
        }
}
```

Illustration 4: Code: condition sur les temps d'attentes

- Fonction de franchissement des transitions

Cette fonction met à jour le nombre de tir de chaque transition. Les transitions franchies durant l'itération courante sont connues grâce à la matrice TabXFranch. Exemple de l'équation pour la transition T_i :

$$TabSortieX' = TabSortieX + TabXFranch;$$

- Fonction mise à jour du contenue des places

Une fois le tirage des transitions effectué, le programme appelle cette fonction afin de mettre à jour le vecteur de marquage. Exemple de l'équation pour la transition P_i :

$$TabSortieP' = TabSortieP + (Wplus - Wmoins) * TabXFranch;$$

- Evolution du temps

Cette fonction est appelée pour faire évoluer ou non la variable associée au temps. En effet, dans le cas des transitions immédiates, leur évolution n'est pas directement liée au temps. L'incrémentation du temps se fait si aucune transition n'a été franchie pendant l'itération en cours.

IV. Application du modèle

En accord avec la société Artefacto, l'objectif principal est d'appliquer notre modèle et ses outils sur la simulation du tramway d'Angers. Cette application doit prendre en compte certaines entités pouvant influencer le comportement du tramway. Les entités retenues sont les voitures et les piétons.

A la vue de la taille de la ligne et de la répétition de situations relativement identiques, nous nous sommes limités à l'extrémité nord de la ligne contenant une extrémité de ligne, trois stations, un carrefour. Pour limiter un réseau fermé, nous avons rajouté une extrémité de ligne au sud de la troisième station.

A. Ligne du tramway d'Angers

1. Contexte

Dans le cadre du développement urbain, Angers Loire Métropole a lancé un nouveau projet de création d'une ligne de tramway urbain. Un tel moyen de transport avait déjà parcouru la ville dans le passé. Ses travaux avaient commencés en 1895. Sa dernière année de mise en service est 1949 [ALM] [11].

Les travaux de ce nouveau tramway ont commencé au début du mois d'avril 2008. Ces travaux vont durer deux ans et ainsi permettre le lancement de la ligne prévu au début de l'année 2010. Une seconde ligne de tramway est prévue. Elle devrait voir le jour en 2015.

Pour l'application de notre modèle, nous nous concentrons sur la ligne actuellement en construction (la ligne A). La société Artefacto possède déjà le rendu 3D simulant, de façon simplifié, le fonctionnement du tramway A.

Longue de 12 kilomètres, la ligne A du tramway reliera en 25 stations, et en moins de 37 minutes, Avrillé au nord à Angers – La Roseraie au sud, en passant par les centres-villes d'Avrillé et d'Angers. [ALM][12]. Un plan est disponible en annexe 1.

2. Récolte des données

Le réalisme de la simulation finale du tramway est majoritairement induit de la finesse des paramètres utilisés pour modéliser la statique et la dynamique de toues les éléments de l'environnement. Ainsi, nous avons besoin de données spécifiques au tramway ainsi qu'au tracé de la ligne A.

[11] http://tramway.angersloiremetropole.fr/pourquoi-le-tram/tram-d-hier/les-grandes-dates/index.html
[12] http://tramway.angersloiremetropole.fr/trace-et-amenagements/index.html

Pour obtenir ces informations, nous nous sommes adressés à "la maison du tramway"[13]. Ainsi nous avons été mis en relation avec Mme Marie-Pierre Trichet[14], Ingénieur projet. Mme Trichet nous a fourni toutes les informations les plus importantes dont nous avons besoins. Ces informations peuvent être classées en deux catégories :

- Tracé de la ligne avec, par exemple : tracé technique de la ligne avec, entre autre, les distances entres les stations mais également la structure des carrefours (nombre de vois, sens, feux)
- Dynamique du tramway avec sa vitesse par zones, sa vitesse d'entrée dans les stations, sa vitesse maximale, etc.

B. Résultats

1. Découpage de la ligne par blocs

La première approche fut de considérer le tracé de la ligne dans son ensemble. C'est-à-dire réaliser un RdP contenant la formalisation des stations, des voies, des carrefours, etc. Puis, ensuite de simuler le tout à partir des différents paramètres choisis comme, par exemple, le paramétrage de la vitesse du tramway sur une voie en associant un temps d'attente dans une place (cas identique au déplacement d'un piéton entre deux points cf. Figure 12, p12).

Cependant cette solution implique la manipulation de très grandes matrices (ou tableaux) ainsi qu'une maintenabilité moins bonne (difficulté à extraire une partie du réseau noyée dans son modèle global). Une approche "modulaire" s'est avérée plus judicieuse. C'est vers cette nouvelle idée que nous avons concentré la suite de noter travail.

L'idée de la solution consiste à découper la ligne en sous parties (Figure 16). Du point de vue de l'infrastructure, ces sous parties correspondent à des tronçons de la ligne. Du point de vue informatique, ces tronçons sont vus comme des boîtes ou des blocs.

Nota : Par convention, une boîte est un tronçon non défini et un bloc est un tronçon défini (bloc voie, bloc station, etc.).

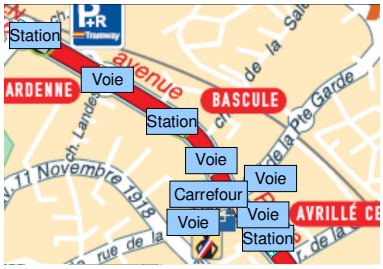


Figure 16: Représentation par "bloc" d'une partie de la ligne

Dans le but de faciliter le développement et l'utilisation de ces boîtes, il a été nécessaire de définir un standard de conception. Chaque boîte a pour entrée(s) une ou plusieurs place(s) et pour sortie(s) une ou plusieurs transition(s) immédiate(s).

[13] Centre d'informations concernant le futur tramway (12 place Imbach ; 49000 Angers)
[14] Marie-Pierre.Trichet@angersloiremetropole.fr

Dans un premier temps, il a fallu préciser les différents tronçons possibles sur la ligne (correspondant aux différents blocs) et les modéliser dans un réseau de Petri.

Dans un second temps, ces divers blocs ont étés traduits en code Scilab. La liste est la traduction de ces bloc sont accessibles dans l'annexe 1.

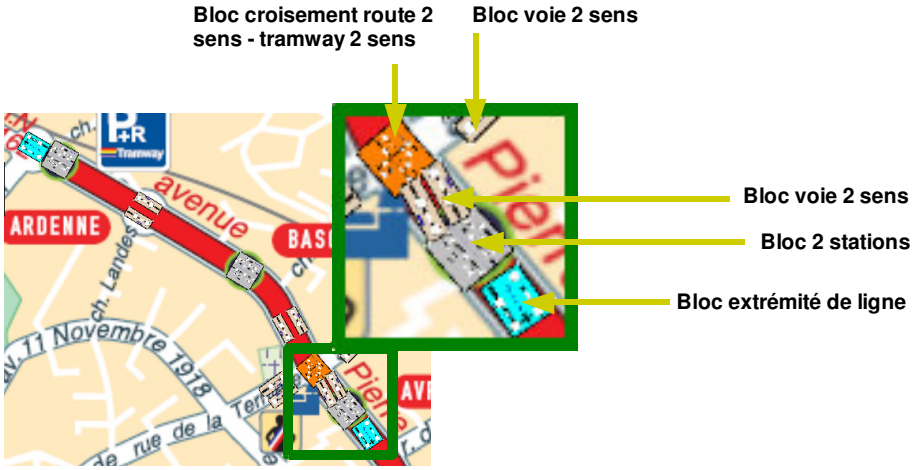


Figure 17: Représentation détaillé des "blocs" d'une partie de la ligne

Ainsi une première version du tracé du début de la ligne est maintenant traduite dans le formalisme des Rdp (Figure 18).

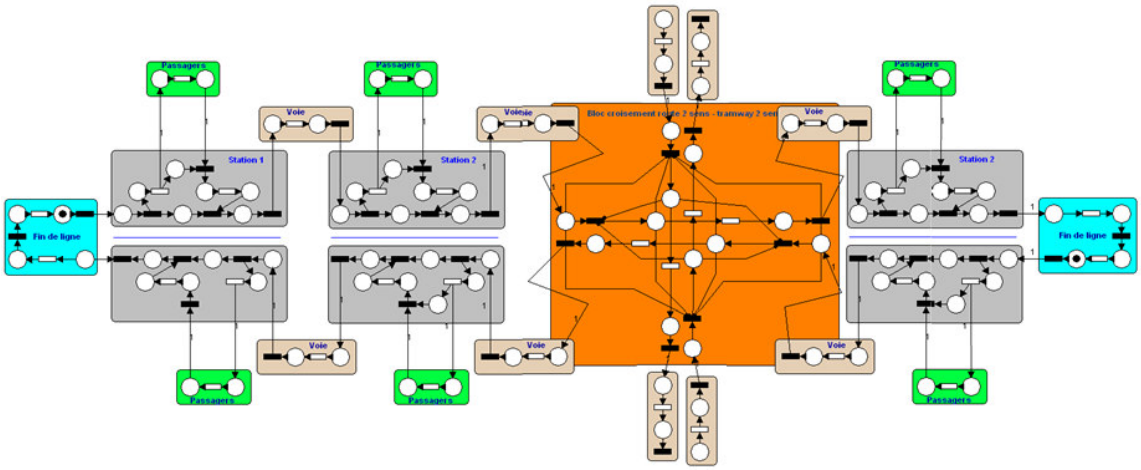


Figure 18: formalisation du début de la ligne en RdP

C. Conclusion

L'utilisation du formalisme des réseaux de Petri ainsi que l'approche modulaire ont permis d'aboutir au schéma du début de la ligne (Figure 18). L'application de la méthode à toute la ligne à est une extension du schéma ci-dessus, c'est-à dire qu'une partie du schéma sera répété un grand nombre de fois (stations, voies, etc.).La différence réside dans le tracé des intersections pouvant influencer le comportement du tramway (carrefour). Chacun de ces carrefours à un tracé propre (carrefour à sens unique, carrefour à double sens, carrefour à feux tricolores, etc.). De nouvelles boites vont donc être crées pour reproduire le comportement spécifique des carrefours.

Perspectives

V. Vers la création d'une solution logicielle

Au travers des parties précédentes, nous avons présenté une méthode de modélisation pour le fonctionnement de la ligne du tramway de la ville d'Angers. Nous avons également explicité le formalisme sur lequel elle se base ainsi que détaillé l'outil de simulation développé. Ensuite nous avons appliqué la méthode sur le début du tracé de la ligne A.

Etendre ce travail à l'ensemble du tracé de la ligne de façon manuelle est long et fastidieux. Développer un logiciel permettant de faciliter la création de la simulation est nécessaire. Son rôle sera de simplifier le paramétrage et l'utilisation des boîtes. Son atout sera de générer automatiquement les diverses matrices transmises en entrée de l'algorithme de simulation des RdP (matrice d'incidence, matrice du marquage initial, vecteur contenant la valeur des temporisations associées aux transitions, matrice des arcs inhibiteurs, etc.).

La première analyse sur le rôle du futur logiciel a permis de mettre à jour un certain nombre de fonctions à intégrer. Voici la liste de ces fonctions :

- Création d'une base de données contenant la structure des boîtes avec un paramétrage par défaut
- Une fonction permettant d'explorer les boites existantes dans la base de données
- Une fenêtre permettant à l'utilisateur d'insérer des boites, de les paramétrer et de les lier entre elle.
- Une fonction permettant de générer les diverses données relatives au tracé (matrices, vecteurs, liaison entre les boites, etc.)
- Une fonction permettant de faire une simulation boîte par boîte (approche *modulaire*)
- Une fonction permettant d'exporter le résultat de la simulation dans un fichier XML.

Schémas d'illustration du programme prévu:

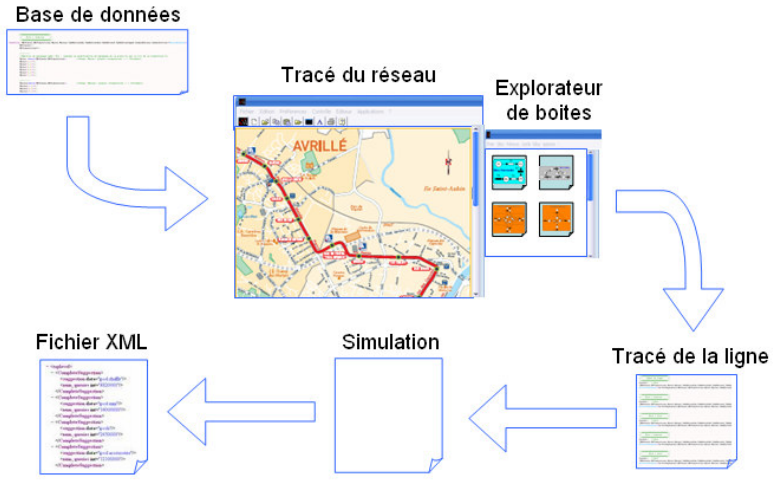


Figure 19: Prévision de l'architecture du logiciel

La portée de ce programme sera de pouvoir créer une simulation réaliste de n'importe quel tramway et, au prix du développement de nouvelles boîtes, étendre la solution à la simulation d'un quartier d'une ville, voire de la ville entière. La simulation pourra être un bon outil de communication et, pourquoi pas à terme, un outil de simulation de flux par exemple, tester la charge maximale en terme de flux que peut accepter une ville. Le développement de ce logiciel sera réalisé durant la seconde partie de mon stage

Conclusion

Lors de ce stage de M2 SDS, nous avons commencé par sélectionner le modèle théorique le plus adapté pour représenter notre travail relatif aux systèmes urbains. Ayant choisi de voir ce-dernier comme un système à événements discrets (SED), nous nous sommes intéressés aux réseaux de Petri (RdP).

Grâce à ce formalisme, nous avons montré qu'il est possible de modéliser des situations urbaines telles que le passage piéton ou encore le carrefour à feux tricolores. Mais aussi de simuler l'évolution de divers acteurs tels que piétons, voitures, tramway, etc.

Dans un second temps nous avons implémenté dans l'environnement Scilab un algorithme capable de simuler des RdP temporisés.

Ainsi, à partir du formalisme choisi et de l'outil développé, nous pouvons simuler le fonctionnement de la ligne du futur tramway de la ville d'Angers. Pour ce faire, nous avons opté pour une approche modulaire en considérant la ligne comme une série de tronçon, chacun correspondant à une boîte dans notre simulateur. Ainsi le réseau est découpé en sous boîtes reliées entre elles. La simulation se fait dés lors boîtes par boîtes. Cette approche nous a permis de réduire considérablement la taille du réseau et ainsi faciliter la maintenance et l'évolution du programme.

En résultat de cette première partie du stage, nous avons pu réaliser le tracé du début de la ligne A. la suite du travail concernera le développement du programme aboutissant à la création finale de la simulation du tramway d'Angers.

Pour mon bilan personnel, durant ce stage j'ai essentiellement appris sur trois plans :

- Tout d'abord, cette première partie du stage m'aura permis de découvrir plus en profondeur le domaine de la Recherche relative aux SED. Au travers de l'approche bibliographique j'ai appris à réutiliser les résultats de diverses sources et m'en servir de base pour le développement d'une simulation du tramway.
- Ensuite, au travers de la rédaction de ce rapport, j'ai pu constater la difficulté à transmettre le message de manière tant simple, claire, précise que concise. Je remercie grandement M. S. Lahaye pour m'avoir aidé et appris dans ce domaine lors de la rédaction de ce rapport.
- Enfin, durant ce stage j'ai également appris à démarcher un organisme en vue d'obtenir des informations importantes pour mon stage. Au travers du rendez-vous avec Mme Trichet, j'ai été mis en situation durant laquelle j'ai dû savoir me présenter, présenter le laboratoire ainsi que mon sujet de stage. L'obtention des informations m'a permis de gagner en confiance en moi.

I. Références

[GC+95]
G. Cohen
Cours : Théorie algébrique des systèmes à événements discrets
Centre Automatique et Systèmes
Ecole des Mines de Paris, Fontainebleau & INRIA, Rocquencourt
1995

[LH+04]
Laurent Hardouin
Cours : Sur la Commande des Systèmes (max,+) Linéaires.
DEA Automatique et Informatique Appliquée - Angers
Décembre 2004

[RDHA+92]
René David et Hassane Alla
Livre : Du grafcet aux réseaux de Petri2ième édition
Traité des nouvelles Technologies, série Automatique
Hermes, 1992

[LH+06]
Laurent Houssin
Thèse : Contribution à la commande des systèmes (max,+)-linéaires. Application aux réseaux de transport.
Laboratoire d'ingénierie des Systèmes Automatisés.
Décembre 2006

[ELM+02]
Ernesto López-Mellado
Short communication : Analysis of discret event systems by simulation on timed Petri net models
Mathematics and computers in simulation
Mai 2002

[ALM]
Angers Loire Métropole
Communauté d'agglomération qui regroupe 31 communes représentant 270 331 habitants, dont la Ville d'Angers.
http://tramway.angersloiremetropole.fr/

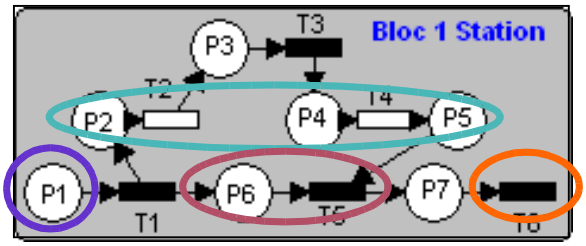
I. Annexes

Annexe 1

Description des "boîtes"

Tracé du futur tramway:

Boites Petri



Entrée (A)

Arrêt station

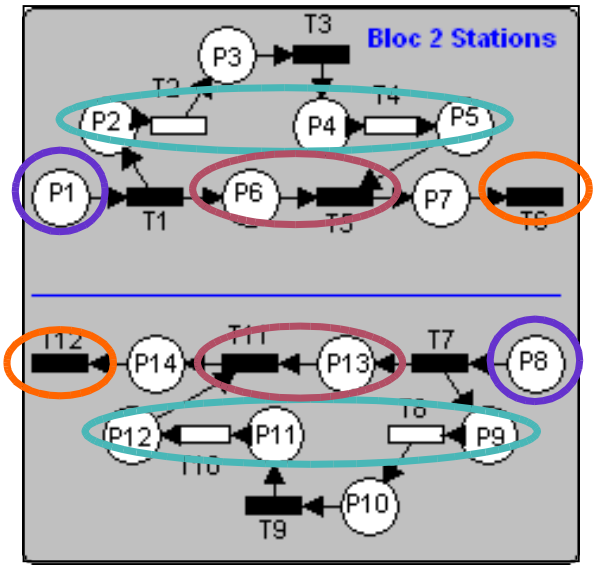
Ouverture / Fermeture portes

Arrêt /Sortie (B)

$$W+ = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad W- = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad Inh = (0)$$

Code Scilab associé:

Paramètres	**W+**	**W-:**	**Inh:**	**Tps:**
Nb Places: 7	(2,1)	(1,1)		(1,2)
Nb tranisitions: 6	(3,2)	(2,2)		(1,4)
	(4,3)	(3,3)		
	(5,4)	(4,4)		
	(6,1)	(5,5)		
	(7,5)	(6,5)		
		(7,6)		



Entrée (A)

Arrêt station

Ouverture / Fermeture portes

Arrêt /Sortie (B)

$$W+ = \begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0
\end{pmatrix}$$

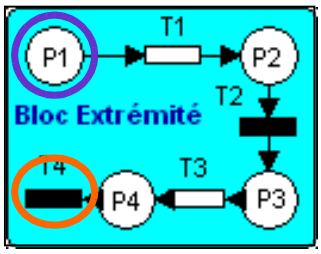
$$W- = \begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&1
\end{pmatrix}$$

$$Inh = (0)$$

Code Scilab associé:

Paramètres
Nb Places: 14
Nb transitions: 12

W+	W-:	Inh:	Tps:
(2,1)	(1,1)		(1,2)
(3,2)	(2,2)		(1,4)
(4,3)	(3,3)		(1,8)
(5,4)	(4,4)		(1,10)
(6,1)	(5,5)		
(7,5)	(6,5)		
(9,7)	(7,6)		
(10,8)	(8,7)		
(11,9)	(9,8)		
(12,10)	(10,9)		
(13,7)	(11,10)		
(14,11)	(12,11)		
	(13,11)		
	(14,12)		



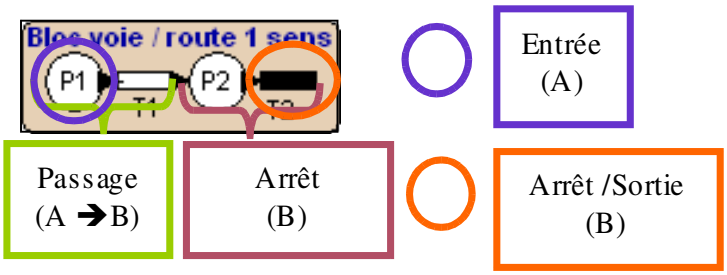
Entrée
(A)

Arrêt /Sortie
(B)

$$W+ = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad W- = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad Inh = (0)$$

Code Scilab associé:

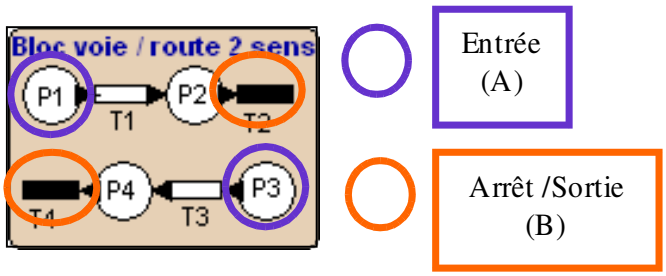
Paramètres	**W+**	**W-:**	**Inh:**	**Tps:**
Nb Places: 4	(2,1)	(1,1)		(1,1)
Nb transitions: 4	(3,2)	(2,2)		(1,3)
	(4,3)	(3,3)		
		(4,4)		



$$W+=\begin{pmatrix}0 & 0\\ 1 & 0\end{pmatrix} \qquad W-=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \qquad Inh=(0)$$

Code Scilab associé:

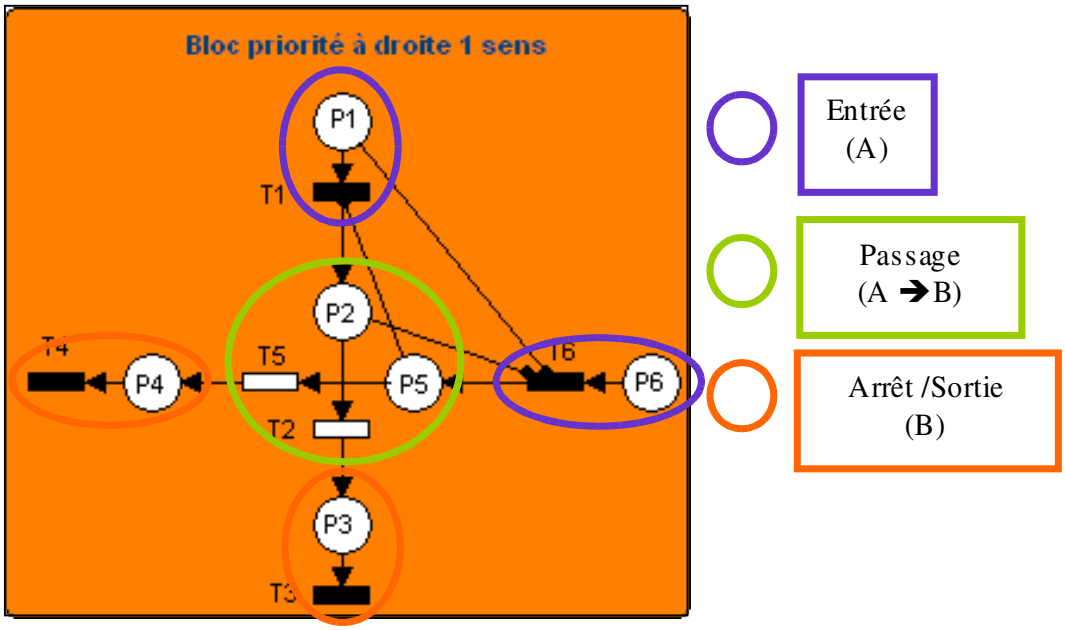
Paramètres	**W+**	**W-:**	**Inh:**	**Tps:**
Nb Places: 2	(2,1)	(1,1)		(1,1)
Nb transitions: 2		(2,2)		



$$W+=\begin{pmatrix}0 & 0 & 0 & 0\\ 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{pmatrix} \qquad W-=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \qquad Inh=(0)$$

Code Scilab associé:

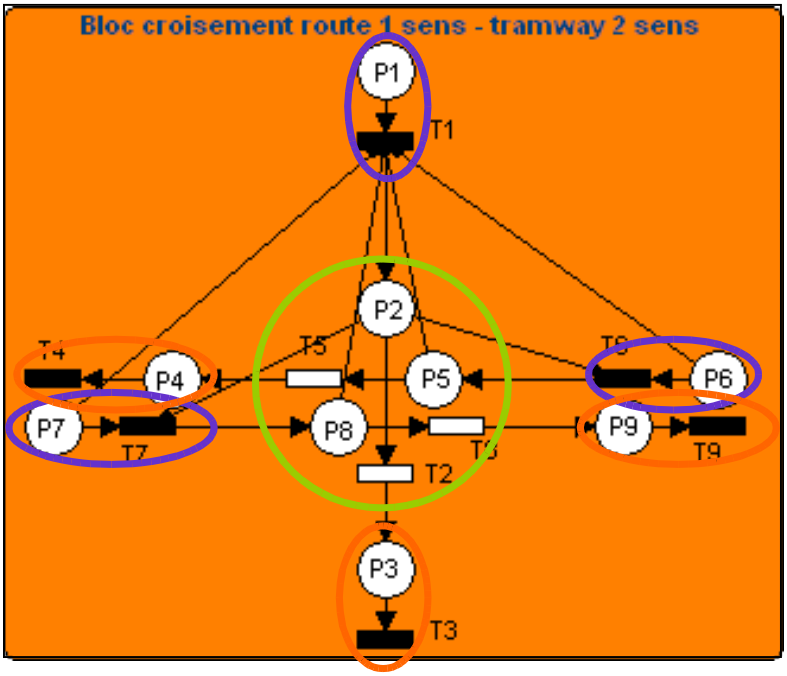
Paramètres	**W+**	**W-:**	**Inh:**	**Tps:**
Nb Places: 4	(2,1)	(1,1)		(1,1)
Nb transitions: 4	(4,3)	(2,2)		(1,3)
		(3,3)		
		(4,4)		



$$W+ = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad W- = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad Inh = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Code Scilab associé:

Paramètres	**W+**	**W-:**	**Inh:**	**Tps:**
Nb Places: 6	(2,1)	(1,1)	(1,6)	(1,2)
Nb transitions: 6	(3,2)	(2,2)	(2,6)	(1,5)
	(4,5)	(3,3)	(5,1)	
	(5,6)	(4,4)		
		(5,5)		
		(6,6)		



Entrée (A)

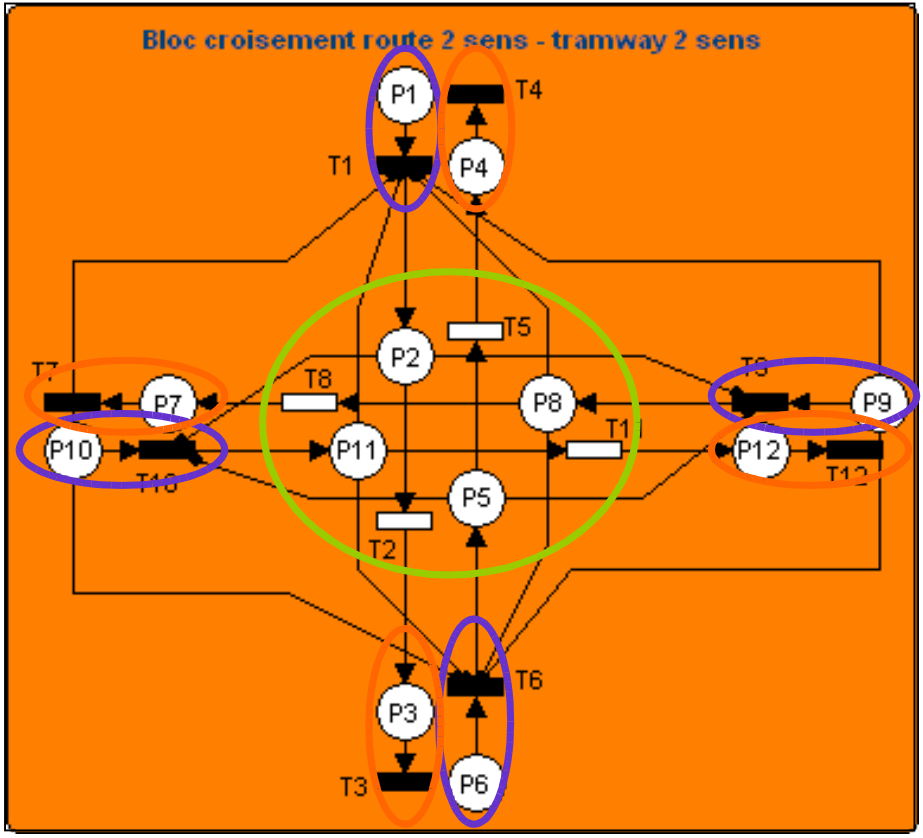
Passage (A ➔B)

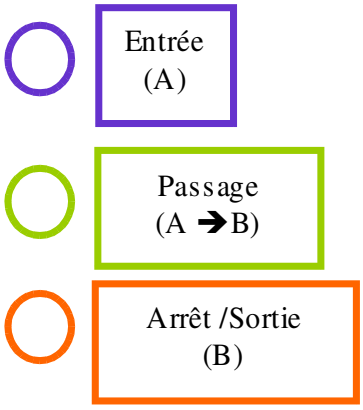
Arrêt /Sortie (B)

$$W+ = \begin{pmatrix} 0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1&0 \end{pmatrix} \quad W- = \begin{pmatrix} 1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&1 \end{pmatrix} \quad Inh = \begin{pmatrix} 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \end{pmatrix}$$

Code Scilab associé:

Paramètres	**W+**	**W-:**	**Inh:**	**Tps:**
Nb Places: 9	(2,1)	(1,1)	(2,6)	(1,2)
Nb transitions: 9	(3,2)	(2,2)	(2,7)	(1,5)
	(4,5)	(3,3)	(5,1)	
	(5,6)	(4,4)	(6,1)	
	(8,7)	(5,5)	(7,1)	
	(9,8)	(6,6)	(8,1)	
		(7,7)		
		(8,8)		
		(9,9)		





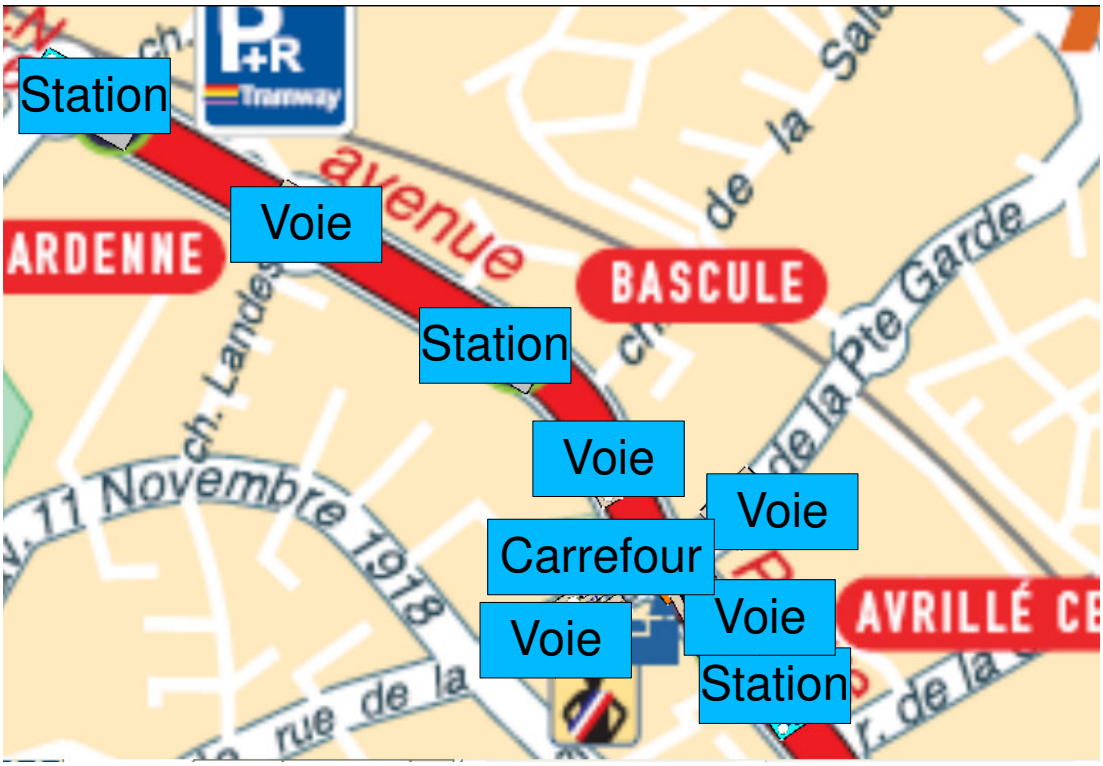
$$W+ = \begin{pmatrix} 0&0&0&0&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&0 \end{pmatrix}$$

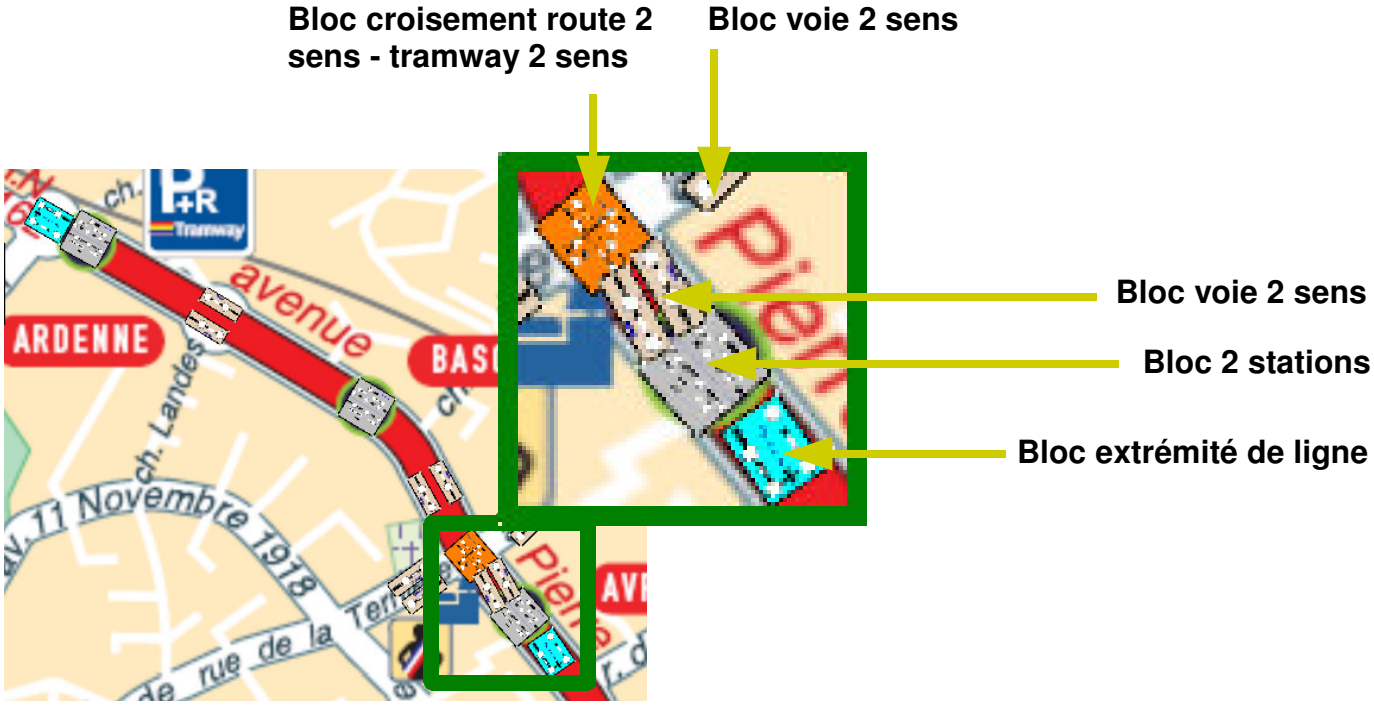
$$W- = \begin{pmatrix} 1&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&0&1 \end{pmatrix}$$

$$Inh = \begin{pmatrix} 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&1&0&0&0&0&0&0\\ 1&0&0&0&0&1&0&0&0&0&0&0\\ 1&0&0&0&0&1&0&0&0&0&0&0\\ 1&0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0 \end{pmatrix}$$

Code Scilab associé:

Paramètres	**W+**	**W-:**	**Inh:**	**Tps:**
Nb Places: 12	(2,1)	(1,1)	(2,9)	(1,2)
Nb transitions: 12	(3,2)	(2,2)	(2,10)	(1,5)
	(4,5)	(3,3)	(5,9)	(1,8)
	(5,6)	(4,4)	(5,10)	(1,11)
	(7,8)	(5,5)	(8,1)	
	(8,9)	(6,6)	(8,6)	
	(11,10)	(7,7)	(9,1)	
	(12,11)	(8,8)	(9,6)	
		(9,9)	(10,1)	
		(10,10)	(10,6)	
		(11,11)	(11,1)	
		(12,12)	(11,6)	


Station
Voie
Station
Voie
Voie
Carrefour
Voie
Voie
Station
ARDENNE
BASCULE
AVRILLÉ
avenue
ch. Landes
11 Novembre 1918
rue de la
de la Pte Garde


Bloc croisement route 2 sens - tramway 2 sens
Bloc voie 2 sens
Bloc voie 2 sens
Bloc 2 stations
Bloc extrémité de ligne
ARDENNE
avenue
ch. Landes
11 Novembre 1918
rue de la

Annexe 2

Algorithme de résolution des réseaux de Petri

Input data:

Wplus,Wmoins, Minh, McapacitéP, MposJetonP, MtpsX, TpsTotCycle, NbTotCycle

TabXFranch = t = n =0;

Pr tout p€P, MTpsSejourJeton (p,1)= MposJetonP(p) ;

```
PasFin=true
While (PasFin) do
{
        n=n+1

        //Transitions franchissables
        Pr tout t€T,
        {
                %BoolFranchissable=true

                //Condition prédécesseurs
                Pr tout p€P,
                {
                        If ( Wmoins(p,t)>0) then
                        {
                                if TabSortieP(p,n-1)<Wmoins(p,t) then
                                {
                                        %BoolFranchissable=false;
                                }
                        }
                }

                //Condition successeurs
                Pr tout p€P,
                {
                        If ( Wplus(p,t)>0) then
                        {
                                if TabSortieP(p,n-1)+Wplus(p,t) > McapacitéP(p) then
                                {
                                        %BoolFranchissable=false;
                                }
                        }
                }

                //Condition inhibiteur
                Pr tout p€P,
                {
                        If ( Minh(p,t)>0) then
                        {
                                if TabSortieP(p,n-1)>=Minh (p,t) then
                                {
                                        %BoolFranchissable=false;
                                }
                        }
                }
```

```
        //Condition temps d'attente
        Pr tout p€P,
        {
                If ( Wmoins(p,t)>0) then
                {
                        //recherche du temps associé à la transition courante
                        TpsX= TabTpsX(t) ;
                        //on cherche le nombre de jetons qui ont suffisamment attendus
                        NbJetons= MTpsSejourJeton (p,TpsX +1)+ MTpsSejourJeton (p,TpsX +2)
                        +… + MTpsSejourJeton (p,TpsX +n) ;     Avec n=nombre de colonnes de
                        MTpsSejourJeton ()

                        if NbJetons<Wmoins(p,t) then
                        {
                                %BoolFranchissable=false;
                        }
                }
        }

        //Affectation du résultat
        TabXFranch(t)=%boolXFranch;
}

//Concurrence
Pr tout p€P,
{
        Pr tout t€T,
        {
                if Wplus(i,j)>0 then NbPredecesseur=NbPredecesseur+1
                if Wmoins(i,j)>0 then NbSuccesseur=NbSuccesseur+1
        }
}

if NbPredecesseur>1 or NbSuccesseur>1 then
{
        //Résolution du problème de concurrence
        TabXFranch =PBCONCURENCE();
}


//Franchissement des tranisitions
TabSortieX(n,:)=TabSortieX(n-1,:)+TabXFranch;

//MAJ des places
TabSortieP(:,m)=TabSortieP(:,m-1)+(Wplus-Wmoins)*TabXFranch';

//EvolutionTemps
Pr tout t€T, if TabXFranch(i)~=0 then
{
        t=t+1;
        %BoolEvolutionTemps=true;
}
```

```
        //MAJ tableau des temps de séjour des jetons
        Pr tout t€T
        {
                //On supprime les jetons qui sont partis de la place amont
                If TabXFranch(1,t)=1
                {
                        //Cette transition a été franchie, on recherche les places amont
                        Pr tout p€P, if ( Wmoins(p,t)>0) then
                        {
                                //un nombre de jeton égal au poids de l'arc est à supprimer de
                                //TabTpsSejourJeton (on retire en premier le jeton ayant le plus
                                //attendu)
                                SupprJetons(TabTpsSejourJeton, Wmoins(p,t) ) ;
                        }
                }
        }

        Pr tout t€T
        {
                if %EvolutionTemps then
                {
                        //On décale le temps de séjour
                        //On additionne la dernière et l'avant dernière colonne
                        //On décale colonne par colonne sauf pour la dernière
                        MoveCol (TabTpsSejourJeton) ;
                }
                else
                {
                        //Une unité de temps ne s'est pas écoulée, on ne fait pas le décallage
                }
        }

        Pr tout t€T
        {
                //On ajoute les jetons qui sont arrivé dans la place aval
                If TabXFranch(1,t)=1
                {
                        //on recherche les places aval
                        Pr tout p€P, if ( Wplus(p,t)>0) then
                        {
                                //un nombre de jeton égal au poids de l'arc est à ajouter dans
                                //TabTpsSejourJeton (dans la première colonne)
                                AjoutJetons(TabTpsSejourJeton, Wplus(p,t) ) ;
                        }
                }
        }

        //Affichage du résultat
        AFFICHAGE();

        if n>=NbTotCycle or t>= TpsTotCycle then PasFin=false, end
}
Return(TabSortieX,TabSortieP)
```

Doat Tony

LISA – Université d'Angers
Année 2007 - 2008

ABSTRACT

In order to validate my both inscription in Master degree (Research and Professional), I carry out my training period, for six month, under the supervision of Mr. Lahaye, assistant professor. This training period is in association with the company Artefacto, based on Rennes, which develops demonstrators of public transports (trams, bus, etc.). Training period is composed by two parts: the first one is made in LISA Laboratory and the second one is made in Artefacto.

The aim of the training period is to make a realist simulation of tram in the Angers future line and to integrate the result in Artefacto's 3D demonstrator.

This report deals with the first part of the training period which consists of seeing the urban system as a discrete events system. With Petri Net (PN) formalisation, we have presented a method to model the tram line. We also developed a PN simulation tool which will be used in the second part of the training period.

Key words: Public transports, tram simulation, discrete event systems, Petri Net.

RESUME

Afin de valider mon année de double inscription en Master 2 (Recherche et Professionnel), j'effectue un stage d'une durée de six mois encadré par M. Lahaye, maître de conférences. Ce stage est en collaboration avec la société Artefacto, basée sur Rennes, qui développe des démonstrateurs de transports urbains (tramways, bus, etc.). Le stage se compose en deux parties : une première effectuée au sein du Laboratoire LISA et d'une seconde réalisée au sein de l'entreprise Artefacto.

Le but du stage est de faire une simulation réaliste du tramway sur la future ligne d'Angers et d'intégrer le résultat au sein du démonstrateur de la société.

Ce rapport présente la première partie du stage qui consiste à voir le système urbain comme un Système à Evénements Discrets. En utilisant la formalisation par réseaux de Petri (RdP), nous avons présenté une méthode pour modéliser la ligne du tramway. Nous avons également développé un outil de simulation des RdP temporisés qui sera utilisé dans la seconde partie de ce stage.

Mots-clefs : Transports urbains, simulation tramway, Systèmes à Evénements Discrets, réseaux de Petri.