REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Khider – BISKRA

Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie

Département d'Informatique

N° d'ordre : /M2/2018

Mémoire

Présenté pour obtenir le diplôme de Master académique en

Informatique

Parcours : **Génie Logiciel et Systèmes Distribués (GLSD)**

Outil pour les réseaux de Petri stochastiques reconfigurables

---

Par :

KHARFALLAH NACEREDDINE

Soutenu le 24/06/2018, devant le jury composé de :

Bennaoui Hammadi	Président
Hmidi Zohra	Membre
Tigane Samir	Encadreur

Résumé

Notre travail, est la création d'un outil pour les réseaux de Petri stochastique reconfigurable (R-SP Ns) ; cet outil, est un objet créé en langage Javascript configurable pour être utilisé dans une page web, ont utilisant un navigateur web pour son exécution ; où la modélisation de R-SPNs se fait dans l'interface web, puis l'envoie de sa structure au serveur pour l'application de l'algorithme de transformation (dépliage) en réseaux de Petri stochastique (SPN) équivalent, la nouvelle structure de ce dernier sera renvoyé à l'outil pour être remodélisées automatiquement ; et parmi nos butes aussi sont le gain du temps d'exécution et l'espace mémoire.

Mots-clés : réseaux de Petri, stochastique, reconfigurable, transformation, dépliage.

Our job is the creation of a tool for reconfigurable stochastic Petri nets (R-SPNs); this tool, it is an object created in Javascript language configurable to be used in a web page, have using a web browser for its execution; where the modeling of R-SPNs is done in the web interface, then sends its structure to the server for the application of the transformation algorithm (unfolding) in stochastic Petri nets (SPN) equivalent, the new structure will be returned to the tool to be remodeled automatically; and among our goals also are the gain in execution time and memory space.

Keywords : Petri nets, stochastic, reconfigurable, transformation, unfolding.

مهمتنا هي إنشاء أداة لشبكات بتري العشوائية القابلة لإعادة تشكيلها (R-SPNs) ؛ هذه الأداة ، هي وسيلة تم إنشاؤه بلغة جافا سكريبت (Javascript) قابل للتكوين لاستخدامه في صفحة ويب ، باستخدام متصفح ويب لتنفيذه ؛ حيث يتم تشكيل نماذج R-SPNs في واجهة الويب ، ثم يرسل هيكله إلى الخادم (Serveur) لتطبيق خوارزمية التحويل الى شبكات بيتري العشوائية (SPN) المكافئة له ، الهيكل الجديد سيتم إعادة ارساله الى الأداة ليتم إعادة تشكيلها تلقائيًا ؛ ومن بين أهدافنا أيضًا الربح في وقت التنفيذ ومساحة الذاكرة.

الكلمات الدالة : شبكات بتري ، ستوكاستيك ، إعادة تشكيل ، تحول ، طي.

Remerciements

Avant tout, je merci ALLAH qui m'a aidé pour finir ce travail.

Je remercie toute ma famille où'ils ont été vraiment un grand soutien.

Je tiens à remercier mon encadreur Dr. Tigane Samir, pour sa méthode de travail, son orientation, ces conseils qui m'ont été d'une grande aide.

Je remercie les professeurs : Kahloul laid, Menadi Samir, Bennaoui hammadi, pour leurs soutiens.

Je remercie aussi mes amis et collègues, pour tout type d'aide.

Je remercie enfin l'ensemble des membres du jury.

Nacereddine.

Dédicaces

À ma mère, ma femme, mes enfants.

À mes sœurs.

À mon frère Ridha.

À toute ma famille.

À mes amis.

Et enfin à tous ceux que j'aime et tous ceux qui m'aiment.

Je dédie ce modeste travail.

Nacereddine.

Table des matières

Liste des figures

Introduction générale

Les réseaux de Petri (RdP) sont introduits en 1962 par Dr. Carl Adam Petri dans sa thèse de doctorat. Les réseaux de Petri sont des outils graphiques et mathématiques puissants qui sont utilisés dans les domaines informatiques, télécommunications et d'autres. D'un côté, on peut avoir une représentation graphique du comportement dynamique des systèmes qui nous permet de visualiser le changement d'état, et d'un autre côté ils fournissent des méthodes formelles pour vérifier les propriétés des systèmes modélisés. Cet outil a connu un succès remarquable dans son utilisation au fil du temps, ce qui a conduit à son développement aux réseaux de Petri de haut niveau, en citant quelques-uns : TPN (Timed Petri Nets) [21], SPN (Stochastic Petri Nets) [7], CPN (Coloured Petri Nets) [22].

Une extension reconfigurable de réseaux de Petri a été proposée dans [15] qui exploite un modèle RdP exprimant la configuration initiale du système et de règles modélisant les reconfigurations et génère par la suite un modèle équivalent de ce système via un algorithme de dépliage. Le modèle obtenu est utilisé pour vérifier les propriétés de ce système.

Les réseaux de Petri reconfigurables (R-SPN) [11], inspirés de MC-RPN, introduisent une famille de techniques de modélisation composée de réseaux de Petri avec un ensemble de règles ; d'où la topologie peut également être modifiée par l'application de ces règles pour obtenir un nouveau réseau (SPN). La vérification des propriétés des R-SPNs utilise un algorithme de dépliage qui transforme un R-SPN donné vers un SPN classique équivalent afin d'appliquer les algorithmes classiques de vérification sur le modèle obtenu pour analyser le modèle reconfigurable. Ce dépliage engendre des modèles volumineux à cause de la duplication des nœuds dans le modèle cible pour modéliser le changement de la topologie. Cela augmente considérablement la complexité temporelle et spatiale de la phase de vérification.

Le but de notre travail est de créer un outil pour la modalisation des R-SPNs et les déplier vers des SPNs classiques. Cet outil fait la modélisation par un navigateur Web qui ne demande pas un matériel performant, puis la structure associée à cette modélisation va se transmettre vers un serveur puissant (selon nos besoins) où se passe la transformation de R-SPN au SPN ; par la suite, la structure de ce dernier sera renvoyée à notre outil pour qu'elle

soit affichée comme nouveau modèle transformé. Le choix de déployer l'outil dans un serveur (une machine puissante) est motivé par le fait que les modèles équivalents utilisés dans la vérification sont volumineux, donc une simple machine va consommer beaucoup de temps dans les étapes d'analyse.

Ce mémoire est organisé comme suit.

Chapitre I : ce chapitre sera consacré aux réseaux de Petri stochastiques (SPN), en les présentant et en définissant les principales notions.

Chapitre II : dans ce chapitre, nous définissons la reconfiguration dans les réseaux de Petri.

Chapitre III : dans ce chapitre, nous allons expliquer la reconfiguration dans les réseaux de Petri stochastiques.

Finalement, Chapitre IV est consacré à la mise en œuvre de l'outil de modélisation de la transformation de réseaux de Petri stochastiques reconfigurables (R-SPN) aux réseaux de Petri stochastiques (SPN), en détaillant toutes les structures de données nécessaires ainsi que les algorithmes utilisés.

Réseaux Petri Stochastiques

I.1. Introduction

Le terme stochastique est utilisé dans de nombreux domaines différents, en particulier lorsque des processus stochastiques ou aléatoires sont utilisés pour représenter des systèmes ou des phénomènes qui semblent changer de manière aléatoire; où un processus stochastique peut être une description du mouvement d'un objet dans le temps. À chaque nouvelle unité de temps, l'objet peut prendre l'une des nombreuses positions possibles et chaque position est associée à une probabilité. Bien que nous ne puissions pas connaître le chemin exact que prendra l'objet, nous pouvons faire des inférences sur le chemin qu'il pourrait prendre en fonction de ces probabilités.

Idéalement, le développeur doit être capable de spécifier, concevoir et mettre en œuvre son système et de le tester pour corriger ses fonctionnalités et sa performance en utilisant un seul formalisme. Les réseaux de Petri, bien que de format graphique, sont quelque peu fastidieux pour spécifier de grands systèmes complexes mais, d'un autre côté, ont été développés exactement pour tester des systèmes discrets distribués pour la correction fonctionnelle. Un autre paradigme qui vise à tester l'exactitude fonctionnelle est celui de traiter des algèbres ou des calculs pour les systèmes communicants. Au cours de la dernière décennie, les chercheurs ont ajouté du temps, sous diverses formes, à des réseaux de Petri ordinaires pour créer des réseaux de Petri stochastiques et des réseaux de Petri stochastiques généralisés (GSPN) pour la vérification des performances.

I.2. Réseaux de Petri (RdP) :

I.2.1. Introduction :

Les réseaux de Petri sont un outil de modélisation graphique et mathématique applicable à de nombreux systèmes. Ils sont un outil prometteur pour décrire et étudier les systèmes de traitement de l'information qui sont caractérisés comme étant simultanés, asynchrones, distribués, parallèles, non déterministes et / ou stochastiques. En tant qu'outil graphique, les réseaux de Petri peuvent être utilisés comme une aide à la communication visuelle similaire aux organigrammes, diagrammes et réseaux. De plus, des jetons sont utilisés dans ces réseaux pour simuler les activités dynamiques et concurrentes des systèmes. En tant qu'outil mathématique, il est possible de créer des équations d'état, des équations algébriques et d'autres modèles mathématiques régissant le comportement des systèmes. Les réseaux de Petri peuvent être utilisés par les praticiens et les théoriciens. Ainsi, ils fournissent un puissant moyen de communication entre eux: les praticiens peuvent apprendre des théoriciens comment rendre leurs modèles plus méthodiques, et les théoriciens peuvent apprendre des praticiens comment rendre leurs modèles plus réalistes. Historiquement parlant, le concept de réseau de Petri trouve son origine dans la thèse de Carl Adam Petri, présentée en 1962 [1].

En général, les méthodes de l'étude de système par réseau de Petri se composent de trois étapes : premièrement on écrit le système en termes de réseau, pour obtenir un modèle en réseau ; deuxièmement on analyse le modèle obtenu, pour en déduire des propriétés comme l'absence de blocage, etc. Finalement, on fait la révision des propriétés obtenues pour montrer si le système est bon. Le résultat de cette méthode nous indique une analyse qualitative du système. Elle constitue une approche très importante pour avoir une bonne évaluation des systèmes [2].

I.2.2. Structure :

Un réseau de Petri est un 4-uplet $R = (P, T, W^-, W^+)$ où :

- P est un ensemble fini et non vide de places,
- T est un ensemble fini de transitions,
- W^- (resp. W^+) est la fonction d'incidence avant (resp. arrière) de domaine $P \times T$ et de codomaine $\mathbb{N}$.

$W^-(p, t)$ est la précondition associée à la transition t et la place p ; elle définit le nombre minimal de jetons dans p nécessaire au franchissement t et retirées de p en cas de franchissement. De même, $W^+(p, t)$ est la postcondition associée à la transition t et la place p ; elle définit le nombre de marques apportées à p par le franchissement de t. Ces deux matrices peuvent être synthétisées en une seule : la matrice d'incidence.

La matrice d'incidence d'un réseau de Petri est la matrice entière $W \in M(P, T)$ définie par : $\forall (p, t) \in P \times T$, $W(p, t) = W^+(p, t) - W^-(p, t)$. Cette matrice traduit le coût global du franchissement d'une transition pour chaque place - la différence entre ce qui est produit et ce qui est consommé -.

Remarque : De par sa définition, cette matrice masque les « boucles » du réseau et il n'est donc pas équivalent de donner la matrice d'incidence ou de donner les deux matrices pre et post.

Un élément de $\mathbb{N}^P$ est appelé marquage d'un réseau de Petri R et définit l'état du réseau R à un instant donné, c'est à dire le nombre de jetons $M(p)$ présentes dans chaque place $p \in P$ du réseau de Petri R.

Un réseau de Petri marqué est donc un 5-uplet $R = \langle P, T, W^-, W^+, M_0 \rangle$ où : $\langle P, T, W^-, W^+ \rangle$ est un réseau de Petri, et M_0 est le marquage initial de R définissant ainsi l'état initial du réseau - c'est à dire, avant tout franchissement - [3].

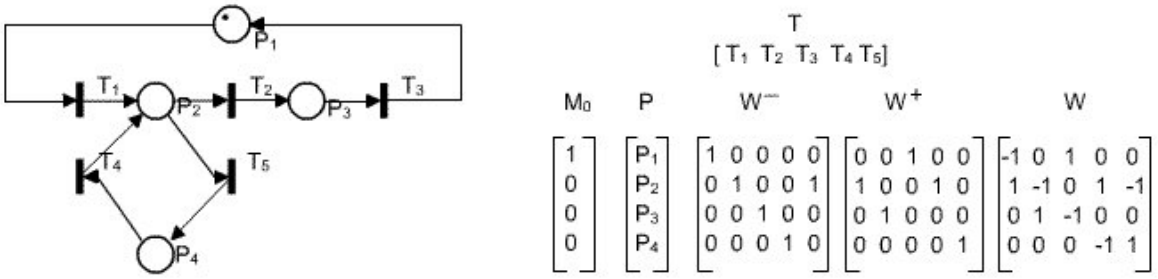


Figure 1 Marquage initial, Matrices *pre*, Matrice *post*, Matrice d'incidence.

I.2.3. Modélisation graphique : [4]

1) Places, transitions et arcs:

Le graphe d'un réseau de Petri est formé de deux types de nœuds appelés places et transitions reliés par des arcs orientés, et biparti, c'est-à-dire qu'un arc relie alternativement une place à une transition et une transition à une place.

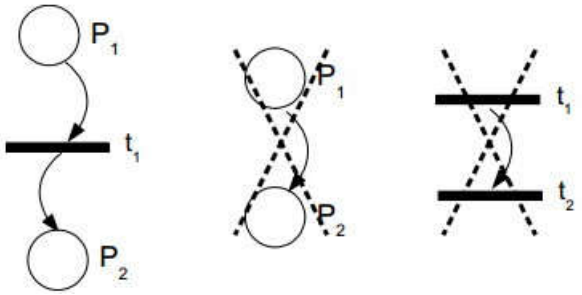


Figure 2 Places, transitions et arcs.

Lorsqu'une place est reliée à une transition par un arc : $W^-(p_i, t_j)$, on parle de place en entrée de t_j. Lorsqu'une transition est reliée à une place par un arc $W^+(p_i, t_j)$, on parle de place en sortie de t_j.

Une transition sans place en entrée est une transition source, une transition sans place en sortie est une transition puits.

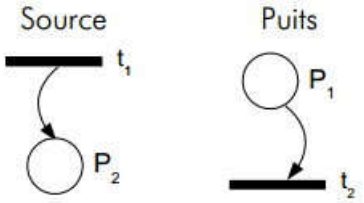


Figure 3 Transition source, transition puits.

2) Marquages :

Chaque place d'un réseau de Petri peut contenir un ou plusieurs jetons (on parle aussi de marques). La configuration complète du réseau, avec toutes les marques positionnées, forme le marquage et définit l'état du réseau (et donc l'état du système modélisé). Dans la suite on traitera principalement des réseaux marqués, et de l'évolution des marquages.

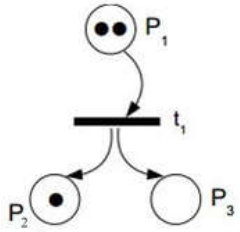


Figure 4 Marquages ($P_1 = 2$, $P_2 = 1$, $P_3 = 0$).

3) Franchissement de transitions :

Pour rendre compte de l'évolution du système modélisé, les réseaux de Petri intègrent un formalisme permettant de passer d'un marquage à un autre : est le franchissement des transitions.

Une transition est franchissable si pour chaque place p en entrée son marquage m est supérieur ou égal le poids de l'arc $W^-(p,t)$.

Pour les transitions franchissables, on définit le franchissement effectif selon les règles suivantes :

- le franchissement est une opération indivisible (atomique),
- des jetons ($W^-(p,t)$) sont consommés dans chaque place en entrée,
- des jetons ($W^+(p,t)$) sont produits dans chaque place en sortie.

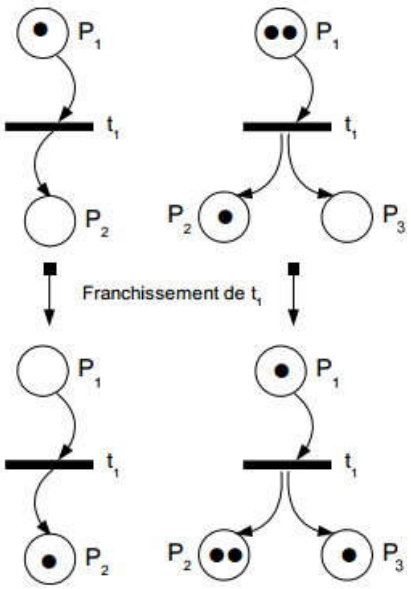


Figure 5 Franchissement.

4) Réseaux particuliers :

Le graphe associé à un réseau de Petri peut être très complexe. Un certain nombre de situations présentent un intérêt particulier :

1. les graphes d'états, dans ce cas chaque transition ne dispose que d'une place en entrée et une place en sortie.
2. les réseaux sans conflits dans lesquels chaque place n'a qu'une transition en sortie.
3. les réseaux dits simples Réseaux avec conflits mais dans lequel chaque transition n'intervient au plus que dans une situation de conflit.
4. les réseaux purs, dans cette situation aucune place n'est à la fois en entrée et en sortie de la même transition.

Le tableau suivant illustre les définitions précédentes :

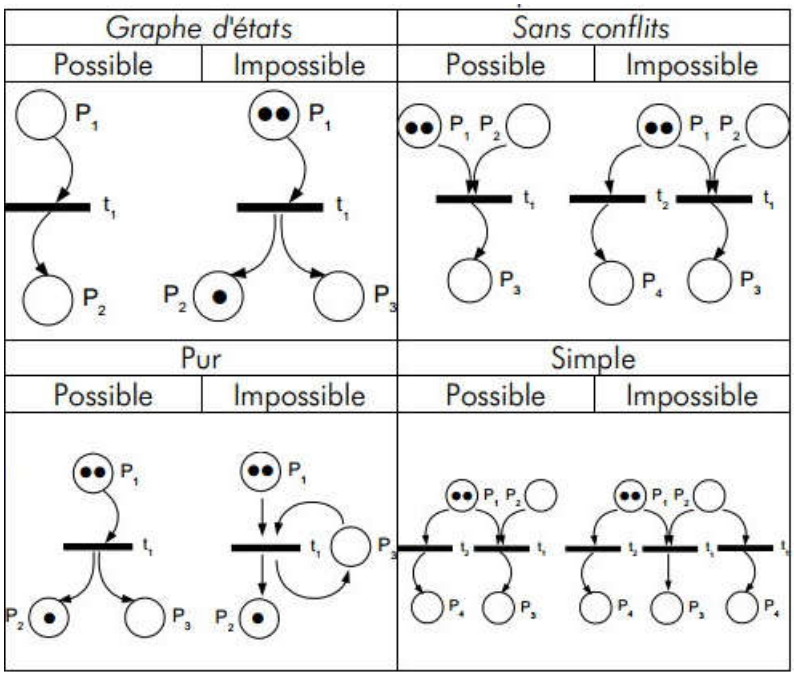


Figure 6 Graphes d'états, les réseaux sans conflits, simples, purs.

I.2.4. Propriétés des réseaux de Petri :

Dans la suite on appellera M_0 le marquage initial, *tt* $*M_0$ l'ensemble des marquages accessibles à partir du marquage M_0 initial.

On notera la franchissabilité d'une transition t_j à partir d'un marquage M_i comme suit: $M_i[t_j>$.

Si le marquage résultant est M'_i alors on notera : $M_i[t_j>M'_i$.

Cette notation est extensible aux séquences de transitions. On note dans ce cas la franchissabilité de la séquence comme suit : $M_i[t_j t_k>$.

Par ailleurs on définit une notion d'ordre sur les marquages en définissant la notion de couverture. Un marquage M' couvre un marquage M (on dit aussi : M' est supérieur à M) si le nombre de marques dans chaque place du réseau $M'(p_i)$ pour M' est supérieur au nombre de jetons pour chaque place $M(p_i)$ dans M. Soit :

$$M' \geq M \Leftrightarrow \forall\ p_i M'(p_i) \geq M(p_i)$$

1) Réseaux bornés et/ou binaires :

Cette propriété, comme les suivantes, est définie pour un marquage M_0 donné. Sa validité pour un autre marquage n'est en rien garantie.

Une place est dite bornée pour un marquage initial M_0 s'il existe un entier k tel que pour tous les marquages accessibles depuis M_0, le nombre de jetons dans cette place est inférieur à k. Ainsi, la place est dite k-bornée.

Un réseau dont toutes les places sont bornées est lui même borné.

Enfin un réseau 1-borné (chaque place contient au maximum un jeton) est dit sauf ou binaire.

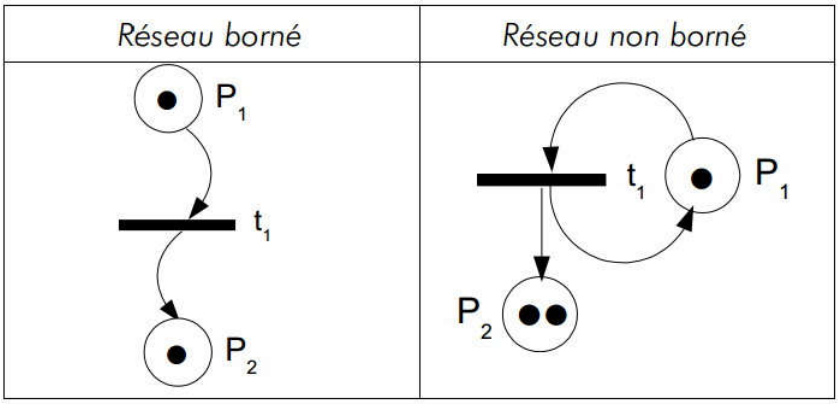


Figure 7 Réseau bornés, non bornés.

2) Vivacité :

La vivacité porte sur les transitions. Une transition t_j est dite vivante pour un marquage initial M_0 si depuis tout marquage accessible il est possible de trouver une séquence de transitions amenant à franchir t_j. Dans un tel réseau il sera toujours possible de refranchir t_j, peu importe les transitions déjà franchies. La vivacité est donc une propriété très forte.

Un réseau dont toutes les transitions sont vivantes est dit vivant.

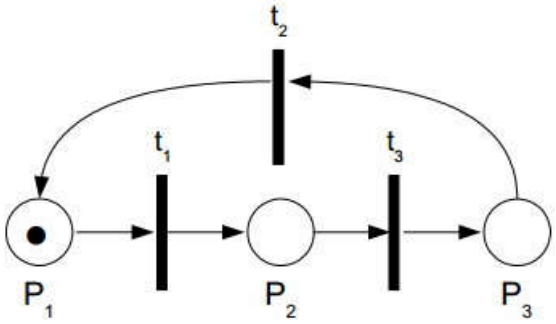


Figure 8 Réseau vivant.

3) Quasi-vivacité :

La quasi-vivacité va définir une propriété moins contraignante que la vivacité. Là ou la vivacité exige que la transition soit franchissable à partir de tout marquage, la quasi-vivacité impose juste l'existence d'une séquence de transition permettant de franchir tj, depuis le seul marquage initial.

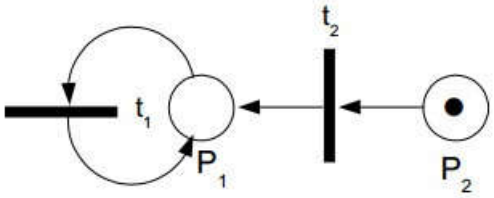


Figure 9 t_2 est quasi vivante.

Un réseau dont toutes les transitions sont quasi-vivantes est dit quasi-vivant.

On peut donc dire de manière simple qu'un tel réseau ne comporte pas de branches mortes pour le marquage initial, il existe toujours au moins un moyen de franchir chaque transition en partant de M0.

4) Blocage :

Un blocage correspond à un marquage du réseau de Petri pour lequel plus aucune transition n'est franchissable.

Un réseau est dit sans blocage si aucun marquage de l'ensemble des marquages accessibles *M0 n'est un blocage.

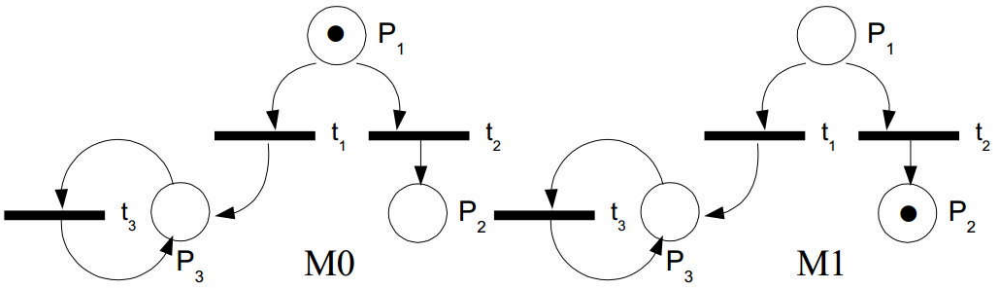


Figure 10 *M1* est un blocage.

5) État d'accueil et réseaux réinitialisable :

Un état d'accueil pour un réseau et son marquage initial est un marquage particulier Ma tel qu'il existe un chemin (une séquence de transitions franchissables) menant à Ma pour tous les marquages accessibles.

Il est donc toujours possible, quelques soient les transitions déjà franchies, de revenir à l'état d'accueil en franchissant de nouvelles transitions.

Si M0 est un état d'accueil, alors le réseau est dit réinitialisable.

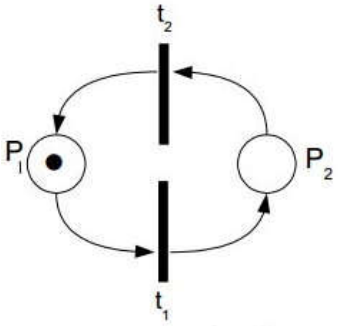


Figure 11 Réseau réinitialisable.

I.2.5. Graphe des marquages et de couverture :

1. Arbre et graphe des marquages accessibles :

Première méthode : construire de manière exhaustive le graphe des marquages accessibles. Cette méthode est extrêmement simple.

Les nœuds du graphe sont formés des différents marquages de $*M_0$. On démarre avec le seul marquage initial, et l'on va construire les différents arcs et nœuds progressivement.

Pour chaque nouveau marquage on détermine l'ensemble des transitions franchissables et pour chaque transition de cet ensemble on ajoute un arc vers le nouveau marquage.

Si un marquage est déjà présent dans le graphe, on se contente de tracer l'arc.

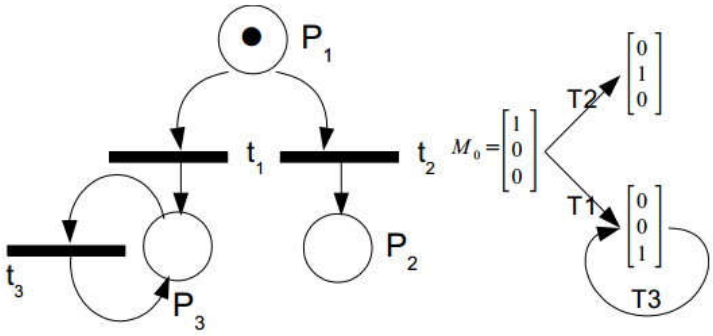


Figure 12 Arbre de marquage.

2. Arbre et graphe de couverture :

L'inconvénient de la méthode précédente est rédhibitoire : tous les réseaux n'ont pas un graphe des marquages accessibles fini.

Donc une méthode alternative a été proposée. Le mécanisme de construction est le même que pour le graphe des marquages accessibles. A ceci prêt que pour chaque nouveau marquage (nœud du graphe) ajouté, on vérifie s'il n'est pas supérieur à un marquage déjà présent sur au moins une séquence entre M_0 et le nouveau marquage.

Si tel est le cas tous les marquages de place supérieurs sont remplacés par ω. Ce symbole matérialise le fait que la place en question peut contenir autant de jetons que souhaité (elle est donc non bornée).

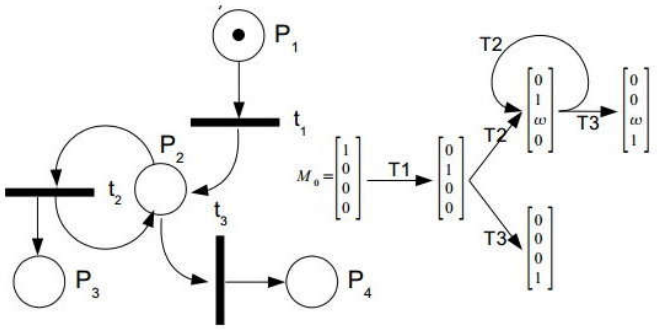


Figure 13 Arbre de couverture.

Dans la suite, les places non bornées le demeurent naturellement et ceci quelles que soient les transitions franchies, ainsi le symbole ω ne disparaît jamais.

Cette méthode produit le graphe de couverture, un graphe fini dans tous les cas.

Comme pour le graphe des marquages accessibles, on va pouvoir déduire de l'observation du graphe de couverture un certain nombre de propriétés pour le réseau de Petri.

I.2.6. Méthodes d'analyse :

Les méthodes d'analyse pour les réseaux de Petri peuvent être classées dans les trois groupes suivants:

- Méthode de l'arbre d'accessibilité.
- L'approche matrice-équation.
- Techniques de réduction ou de décomposition.

La première méthode implique essentiellement le dénombrement de toutes les marques atteignables ou de leurs marques recouvrables. Il devrait pouvoir s'appliquer à toutes les classes de réseaux, mais il est limité aux «petits» réseaux en raison de la complexité de l'explosion de l'espace d'état. D'un autre côté, les équations matricielles et les techniques de réduction sont puissantes mais, dans de nombreux cas, elles ne s'appliquent qu'à des sous-classes spéciales de réseaux de Petri ou à des situations spéciales. [1]

I.3. Variable aléatoire : [5]

I.3.1. La probabilité :

On considère l'ensemble E des éventualités possibles résultant d'une épreuve (expérience, observation ou simulation), chacune de ces éventualités étant appelée événement élémentaire. Un événement quelconque est défini comme un sous-ensemble A de E contenant tous les événements élémentaires de E composant l'événement A. La probabilité attachée à un événement A est un nombre $P(A)$ compris entre 0 et 1, obéissant à certaines règles axiomatiques, en particulier :

- L'événement de l'ensemble vide a une probabilité nulle.
- L'événement E a une probabilité égale à 1.
- $\forall A \subseteq E$, on a $0 \leq P(A) \leq 1$.
- $\forall A, B \subseteq E$, on a $P(A \cup B) = P(A) + P(B)$ si $A \cap B = \varnothing$.

I.3.2. Variables aléatoires continue :

Une variable aléatoire continue X est une fonction à valeurs réelles définie sur un ensemble Ω (ensemble des événements possibles, par exemple : Ω = {pile , face} dans le cas du lancé d'une pièce de monnaie) telle que l'ensemble des valeurs prises par X, noté $X(\Omega)$, est un intervalle fini ou infini.

I.3.3. Variables aléatoires discrètes :

Une variable aléatoire est discrète si elle ne peut prendre qu'un nombre fini de valeurs. Pour chaque valeur x_i, on associe la probabilité $P(x_i)$ d'apparition de cette valeur.

Pour N valeurs, l'ensemble des probabilités associées est tel que :

$$\sum_{i=1}^{N} \mathrm{p}(x_i) = 1$$

Si N couvre l'ensemble des valeurs.

I.4. Chaînes de Markov : [6]

I.4.1. Définition formelle :

Un processus stochastique $\{X(t), t \geq 0\}$ forme une chaîne de Markov à temps continu si pour tous les entiers n, et pour n'importe quelle séquence $t_0, t_1, ..., t_n, t_{n+1}$ telle que $t_0 < t_1 < ... < t_n < t_{n+1}$, on a

$$Prob\ \{X(t_{n+1}) = x_n + 1 \mid X(t_0) = x_0, X(t_1) = x_1, ..., X(t_n) = x_n\}$$

$$= Prob\ \{X(t_{n+1} = x_{n+1} \mid X(t_n) = x_n\}$$

1. Probabilités de transition :

À partir du processus stochastique $Prob\{X(t_{n+1} = x_{n+1} | X(t_n) = x_n\}$ on peut écrire les probabilités de transition d'une chaîne de Markov à temps continu et non-homogène par : $p_{ij}(s, t) = Prob\{X(t) = j | X(s) = i\}$

Où $X(t)$ est l'état de la chaîne de Markov dans le temps $t \geq s$

2. Pourquoi étudier ce modèle ?

- Formalisme classique ;
- Avantage : simplicité et facilité d'utilisation ;
- Problème : quasiment impossible d'envisager une modélisation directe pour de grands systèmes.
- Les propriétés mathématiques facilitent l'analyse des performances.

3. Modélisation par des chaînes de Markov :

- Représentation du système en termes d'états/transitions ;
- La dynamique du système est représentée par une matrice de transition ;
- Échelles de temps :
    - Temps continu, distribution exponentielle, taux de transition;
    - Temps discret, distribution géométrique, probabilités de transition.

4. Définition :

Une chaîne de Markov est une suite de variables aléatoires (X_n, $n \in \mathbb{N}$) qui permet de modéliser l'évolution discrète et dynamique d'un système aléatoire : X_n représente l'état du système à l'instant n.

5. Propriété de Markov :

La propriété fondamentale des chaînes de Markov, dite propriété de Markov, est que son évolution future ne dépende du passé que au travers de sa valeur actuelle.

Autrement dit, (X_o, ..., X_n) et (X_{n+k}, $k \in \mathbb{N}$) sont indépendants en relation à X_n.

6. Représentation graphique :

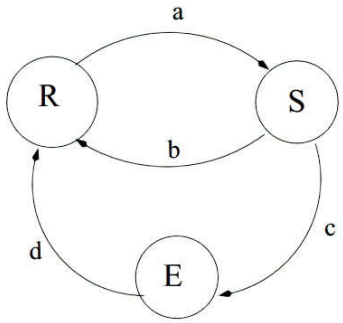


Figure 14 Représentation graphique de chaînes de Markov.

- R (réveillé), S (sommeil) et E (endormi) sont des états ;
- a, b, c et d sont des taux ou probabilités de transition.

I.4.2. Chaînes de Markov à temps discret – DTMC :

1. Présentation :

Dans une chaîne de Markov à temps discret, on observe l'état du système dans un ensemble discret du temps. Autrement dit, l'intervalle de temps entre chaque observation est constant.

2. Intervalle d'observation :

Un important paramètre dans le DTMC est le choix de l'intervalle de temps entre les observations :

- Un intervalle trop petit ne permet pas d'observer des changements d'état.
- Un intervalle trop grand permet de multiples changements d'état entre chaque observation.

3. Matrice de transition :

P est une matrice stochastique si la somme de chaque ligne est égale 1.

I.4.3. Chaînes de Markov à temps continu – CTMC :

1. Présentation :

Dans une chaîne de Markov à temps continu, le changement d'état peut se produire dans n'importe quel point dans le temps.

Ces points sont aléatoires et pas nécessairement entiers.

2. Intervalle de temps :

L'intervalle de temps entre chaque changement d'état suit une variable exponentielle dont le taux dépend uniquement de l'état courant du système.

3. Lois exponentielles :

X variable aléatoire à valeur dans $\mathbb{R}_{\geq 0}$

1) Distribution sur $\mathbb{R}_{\geq 0}$:

- Densité de probabilité : $f : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ *telle que* $\int_{R \geq 0} f(t)dt = 1$
- Fonction de répartition : $F(x) = P(X \leq x) = \int_{t=0}^{X} f(t)dt = 1$

2) Propriété "sans mémoire" :

On cherche une distribution de probabilité vérifiant la propriété :

$P(X > s + t \mid X > t) = P(X > s)$

Solutions de cette équation : les lois exponentielles

3) Les lois exponentielles :

Paramètre $\lambda \in \mathbb{R}_{>0}$

- Densité de probabilité : $f(t) = \lambda e^{-\lambda t}$
- Fonction de répartition : $F(x) = P(X \leq x) = 1 - e^{-\lambda x}$

4) Propriétés :

Loi sans mémoire Espérance (valeur moyenne) = $1/\lambda$

I.4.4. Propriétés :

1. Irréductibilité :

- état accessible : Un état j est dit accessible à partir d'un état i ($i \rightarrow j$) si à partir de l'état i on peut arriver à l'état j avec une probabilité non-nulle.
- état communicant : Un état i est dit communicant avec l'état j ($i \leftrightarrow j$) si $i \rightarrow j$ et $j \rightarrow i$. Un ensemble d'états C est une classe communicante si tous les états dans C communiquent avec tous les autres et aucun état dans C ne communique avec un état en dehors de C.

2. Chaîne de Markov irréductible:

Une chaîne de Markov est dite irréductible si son espace d'états est formé par une seule classe communicante. Autrement dit, s'il est possible d'atteindre n'importe quel état à partir d'un état quelconque.

3. Périodicité :

- état périodique : Un état i a une période k si tout retour à l'état i doit avoir lieu dans un nombre de pas multiple de k ($k > 1$).
- état apériodique : Si $k = 1$, alors l'état est dit apériodique.

4. Chaîne de Markov apériodique:

Une chaîne de Markov est apériodique si tous les états sont aperiodiques. Il suffit d'un seul état apériodique pour que tous les états d'une chaîne de Markov irréductible soient aperiodiques.

5. Récurrence:

- état transitoire : un état i est dit transitoire si, étant donné qu'on commence par l'état i, il y a une probabilité non-nulle de ne plus jamais retourner à l'état i.
- état récurrent : un état i est récurrent s'il n'est pas un état transitoire.
- état absorbant : un état i est dit absorbant s'il est impossible de sortir de cet état.

6. Ergodicité:

- état ergodique : un état i est dit ergodique s'il est apériodique et récurrent.

7. Chaîne de Markov ergodique:

Une chaîne de Markov est ergodique si elle est apériodique, et si tous ses états sont ergodiques.

I.4.5. Analyse de performance :

1. Type d'analyse:

- Transitoire : évolution du comportement du système (dépendant du temps et de l'état initial) ;
- Stationnaire : comportement moyen du système (indépendant du temps et de l'état initial).

L'analyse transitoire nous permet d'observer l'évolution d'un système dans le temps à partir d'un état initial. Ces systèmes ont tendance à aller vers un comportement stationnaire, indépendant de l'état initial et du temps.

2. Évolution du système :

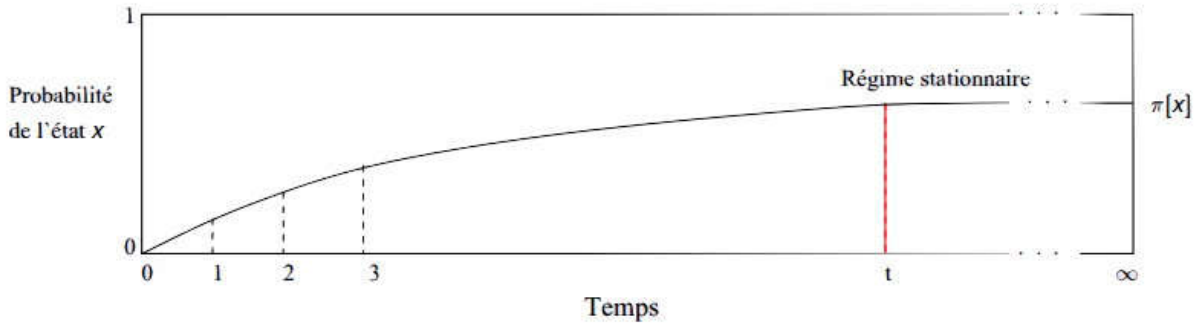


Figure 15 Évolution du système.

3. Cas ergodique :

On considère des chaînes de Markov ergodiques.

1) Chaînes de Markov à temps discret :

On note Q la matrice de transition. Le comportement en régime stationnaire est caractérisé par une distribution de probabilités π, vecteur de taille n (nb d'états de la chaîne de Markov), vérifiant :

- $Pi\ \pi i = 1$
- $\pi.Q = \pi$

2) Chaînes de Markov à temps continu :

On note Q la matrice de transition.

Le comportement en régime stationnaire est caractérisé par une distribution de probabilités π, vecteur de taille n (nb d'états de la chaîne de Markov), vérifiant :

- $Pi\ \pi i = 1$
- $\pi.Q = 0$

I.5. Réseaux de Petri stochastique (SPN) : [7]

Les réseaux de Petri temporisés dans lesquels les temps de franchissement sont spécifiés par des variables aléatoires sont des modèles probabilistes.

L'exécution d'un modèle réseaux de Petri temporisé correspond à la réalisation d'un processus ponctuel stochastique.

L'utilisation de distributions exponentielles pour la définition des spécifications temporelles est particulièrement attrayante car le réseau de Petri temporisé dans lequel tous les délais de transition sont exponentiellement distribués peut être mappé sur des chaînes de Markov à temps continu (CTMC).

Dans ce cas, la propriété sans mémoire de la distribution exponentielle rend inutile la distinction entre la distribution du retard lui-même et la distribution du retard restant après un changement d'état.

Les réseaux de Petri stochastiques sont temporisés (transition) avec tir atomique et dans lesquels les temps de franchissement de transition sont des variables aléatoires distribuées exponentiellement: chaque transition t_i est associée à un temps de franchissement aléatoire dont la fonction de densité de probabilité est exponentielle négative avec le taux (rate) w_i.

Les SPN ont été définis à l'origine dans [37,53]. Formellement, un modèle SPN est un 6-tuple
SPN = (P, T, I (.), O (.), W (.), M_0) (12).

P, T, I (.), O (.) Et M_0 ont les significations usuelles pour le modèle réseaux de Petri sous-jacent constitue la composante structurelle d'un modèle SPN.

La fonction W permet la définition de la composante stochastique d'un mappage de modèle des transitions SPN en fonctions réelles positives du marquage SPN. Ainsi, pour toute transition t, il est nécessaire de spécifier une fonction $W(t,m)$.

Dans le cas de l'indépendance de marquage, la notation plus simple w_k est normalement utilisée pour indiquer $W(t_k)$, pour toute transition $t_k \in T$.

La quantité $W(t_k, m)$ (ou w_k) est appelée le "taux" (rate) de transition t_k dans le marquage m.

I.5.1. Réseaux de Petri stochastiques généralisés (GSPN) : [8]

Cette classe est introduite par Ajmone Marsan [12], [13]. Le réseau se compose de transitions avec une temporisation nulle dites transitions immédiates et de transitions avec une temporisation aléatoire distribuée exponentiellement dites transitions stochastiques. Le processus stochastique sous-jacent au graphe des marquages est un processus de Markov à temps continu.

Reconfiguration dans les réseaux de Petri

II.1. Introduction :

La caractéristique des réseaux de Petri reconfigurables, consistant en un réseau de Petri et un ensemble de règles qui peuvent le modifier, est la possibilité de distinguer les différents niveaux de changement.

Ils fournissent des formalismes puissants et intuitifs pour modéliser des systèmes logiciels ou matériels dynamiques qui sont exécutés dans des infrastructures dynamiques.

Ces infrastructures sont dynamiques car elles sont également sujettes à changement et supportent diverses applications pouvant partager certaines ressources.

De tels systèmes logiciels ou matériels dynamiques sont devenus de plus en plus courants mais sont difficiles à gérer.

La modélisation et la simulation de systèmes à structures dynamiques nécessitent à la fois la représentation de leurs processus et des changements de système au sein d'un même modèle.

Comme le type sous-jacent de réseau de Petri peut varier (par exemple, réseaux de transition / lieu, réseaux d'objets, réseaux temporisés et / ou stochastiques ou réseaux de haut niveau), cette approche peut être considérée comme une famille de techniques de modélisation formelle.

Les réseaux de Petri reconfigurables sont une instanciation de systèmes de transformation abstraits formulés dans la théorie des catégories.

L'idée fondamentale est de caractériser les catégories qui permettent des transformations en double-poussée: seules les descriptions schématiques sont donc nécessaires. Ceci a l'avantage d'une théorie approfondie qui donne une grande quantité de résultats concernant la partie transformation. [9]

II.2. Les réseaux de Petri reconfigurables :

Les réseaux de Petri reconfigurables sont proposés dans [14] pour prendre en charge la reconfiguration dans PN. Les reconfigurations sont exprimées via un ensemble de règles de réécriture qui modifient uniquement les relations de flux (c'est-à-dire que les places et les transitions sont toujours inchangées après l'application d'une règle). Pour contrôler l'application des règles, les réseaux de Petri reconfigurables sont étendus aux réseaux de Petri reconfigurables contrôlés [15], où l'activation d'une règle de réécriture dépend du marquage de réseau. Un MC-RPN [15] est une structure $N = <P, T, R, \gamma 0>$ où:

(i) $P = \{p_0, \ldots, p_n\}$ est un ensemble non vide et fini de places,
(ii) $T = \{t_0, \ldots, t_m\}$ est un ensemble non vide et fini de transitions,
(iii) $P \cap T = \emptyset$,
(iv) $R = \{r_0, \ldots, r_k\}$ est un ensemble fini de règles de réécriture,
(v) γ_0 est la configuration initiale du N.

Une règle $r \in R$ est une structure $r = <D, {}^\bullet r, r^\bullet, \mathbb{C}, \mathbb{M}>$, telle que:

(i) $D \subseteq P$,
(ii) ${}^\bullet r$: $(D \times T) \cup (T \times D) \rightarrow \mathbb{N}$ décrit les relations d'écoulement des places dans D à $\mathbb{N}$ avant d'appliquer r,
(iii) $r^\bullet$: $(D \times T) \cup (T \times D) \rightarrow \mathbb{N}$ décrit les relations d'écoulement des places dans D à $\mathbb{N}$ après application de r,
(iv) $\mathbb{C} \subseteq D$, est l'ensemble des places de contrôle, et $\mathbb{M}$ est le marquage minimum requis des places de $\mathbb{C}$ afin que la règle puisse être appliquée.

Reconfiguration dans les réseaux de Petri Stochastique

III.1. Introduction :

Les réseaux de Petri stochastiques (SPN) sont des réseaux de Petri de haut niveau qui ont été proposés pour modéliser des systèmes aléatoires et stochastiques. Les variantes les plus utilisées sont les SPNs [16]. Ils sont une extension des réseaux de Petri de transition temporisés où la durée n'est plus déterministe mais stochastique avec une loi probabiliste prédéfinie. Les réseaux de Petri stochastiques généralisés (GSPNs) [17] sont une extension des SPNs où les transitions peuvent être de deux sortes: stochastique ou immédiate. L'utilisation des réseaux GSPNs permet aux concepteurs d'évaluer les performances des systèmes modélisés et d'étudier plusieurs paramètres quantitatifs. L'introduction de la reconfiguration dans les réseaux GSPNs est une question ambitieuse qui donnera lieu à un nouveau formalisme de réseau de Petri stochastique reconfigurable. Une première approche a été développée dans certains travaux récents [18,19]. Les GSPNs ont été étendus aux GSPNs reconfigurables en utilisant les systèmes de réécriture de réseau améliorés [20] pour fournir un nouveau formalisme appelé INRS-GSPN. Cette approche a été utilisée pour concevoir des RMS stochastiques et pour permettre l'évaluation des performances des configurations. Sur la base de l'approche de l'INRS, la reconfiguration du réseau GSPN devrait préserver les qualités spécifiques requises: vivacité, bornitude et réversibilité [9].

Afin d'étudier la reconfiguration dans les SPNs, un nouveau formalisme appelé R-SPNs a été proposé dans [11] qui prend en charge la reconfigurabilité et l'analyse structurelle / comportementale proposée pour les SPN.

1) Définition formelle :

Un R-SPN est défini comme un tuple $N = <P, T, R, \gamma_0>$, où:

(i) P, T et R sont définis comme dans MC-RPN,
(ii) γ_0 est la configuration initiale de N modélisée par SPN.

Une règle $r \in R$ est un tuple $r = <D, {}^\bullet r, r^\bullet, \mathbb{C}, \mathbb{M}, V>$, tels que:

(i) D, ${}^\bullet r$, $r^\bullet$, $\mathbb{C}$, $\mathbb{M}$ sont définis comme dans MC-RPN,
(ii) V est un taux (rate) d'une distribution exponentielle négative spécifiant le délai d'application de la règle de reconfiguration r.

En effet, le formalisme SPN est équipé d'un ensemble d'algorithmes et de méthodes utilisés pour vérifier ses propriétés qualitatives / quantitatives en fonction de sa structure (analyse structurelle) ou de son graphe de marquages accessibles (analyse comportementale). Pour cela, l'analyse d'un R-SPN N se fasse à travers son dépliage dans un SPN G_e. L'algorithme de dépliage proposé dans MC-RPN est adopté pour vérifier les R-SPNs.

En fait, un R-SPN N peut être reconfiguré en ajoutant et / ou en supprimant des arcs. Ces transformations ont lieu pour activer, désactiver et / ou modifier les conditions / effets (resp. pré / postconditions) de certaines actions (resp. transitions) dans le système (resp. le modèle). Par conséquent, le réseau équivalent G_e doit être capable de modéliser ces transformations et leurs effets. De plus, la commutation entre les configurations doit être modélisée. Dans ce cadre, nous réutilisons le même algorithme proposé dans MC-RPN [15] et l'adaptons pour considérer le cas discuté sur la Figure 17. En effet, dans le SPN G_e obtenu comme un dépliement de N, nous supprimerons toutes les transitions t satisfaisant $\bullet t = t\bullet = \emptyset$. Les étapes de l'algorithme sont données comme suit.

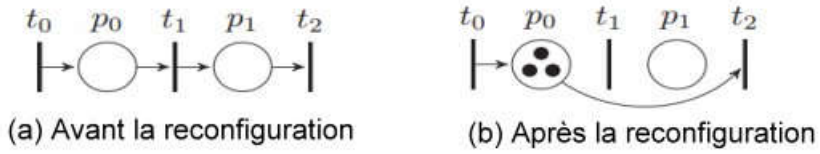


Figure 16 La reconfiguration après application r.

Figure 17 Contre-exemple.

Soit $G = \{C_0, C_1, ..., C_n\}$ est l'ensemble des configurations d'un R-SPN N, tel que $C_0 = \gamma_0$ est la configuration initiale et $C_1, ..., C_n$ sont les configurations obtenues en appliquant des règles de reconfiguration sur C_0. Soit $P^\circ = \{p^\circ_0, p^\circ_1, ..., p^\circ_n\}$ est un ensemble de places associés à $C_0, C_1, ..., C_n$, respectivement. Soit $M^{\circ 0} = (1, 0, ..., 0)$ le marquage initial de $^\circ p_0, ^\circ p_1, ..., ^\circ p_n$, respectivement. Soit $G_e = (P_e, T_e, F_e, m_{e0}, \Lambda_e)$ le SPN équivalent du R-SPN N. Nous considérons les notations suivantes:

- X_Y: désigne la concaténation des deux vecteurs X et Y,
- $M_e = M\_M^{\circ}_i$: désigne un marquage accessible de G_e, où M est le marquage des places de P et M°_i est le marquage des places $\{^{\circ}p_i\}$ associés aux configurations $\{C_i\}$, respectivement, de telle sorte que: $\Sigma_i M_e(^{\circ}p_i) = 1$,
- $t^{\cdot i}$: désigne la représentation de la transition $t \in T$ dans l'équivalent SPN G_e avec les mêmes connexions (*pre* and *post*) dans C_i,
- $F_{Ci}(n, \cdot)$: désigne les post-conditions d'un nœud n en C_i,
- $F_{Ci}(\cdot, n)$: dénote les pré-conditions de n dans C_i,
- $F_e(P, t)$: dénote les pré-sous-conditions de t avec des nœuds dans P dans le SPN G_e équivalent,
- $F_e(t, P)$: désigne les sous-post-conditions de t avec des nœuds dans P dans le SPN Ge équivalent,
- ${}^{\bullet}t_C$: indique le pré-ensemble de t dans la configuration C,
- $t^{\bullet}_C$: désigne le post-ensemble de t dans la configuration C,
- $[M_0 >^0 t_0 [M_1 >^0 t_1 \ldots [M_x > r^i_0 [M_y >^i t_q \ldots [M_z >^j t_p$ désigne une séquence de tir de transitions et de règles dans R-SPN, où $[M_0 >^i t_0 [M_1 >^i t_1 \ldots [M_x$ désigne une séquence de transitions franchissables dans la configuration C_i, et $[M_x > r^i_k [M_j$ désigne une règle applicable r depuis une configuration C_k dans le marquage M_x à une configuration C_i.

Pour créer le SPN G_e équivalent, d'abord nous clonons tous les places dans P avec leur marquage initial dans l'ensemble des places P_e dans le SPN G_e. Deuxièmement, les places P° associées aux configurations sont ajoutées à P_e. Ainsi, l'endroit $^{\circ}p_{i,\ i=0,\ n}$ dans P° est associé à la configuration $C_{i,\ i=0,\ n,}$ respectivement. Un jeton dans p°_i signifie que le système reconfigurable est dans la configuration C_i, de sorte que $\Sigma^n_{i=0} M_e(p^{\circ}_i) = 1$, puisque le système ne peut pas être dans deux configurations à la fois.

Compte tenu le dépliage des transitions, deux cas doivent être considérés. Tout d'abord, pour tout couple "transition t et configuration C_i", *si* ${}^{\bullet}t_{Ci} \neq \emptyset \vee t^{\bullet}_{Ci} \neq \emptyset$, c'est-à-dire t n'est pas neutralisée, (une transition t est dite neutralisée lorsque ${}^{\bullet}t_{Ci} = \emptyset \wedge t^{\bullet}_{Ci} = \emptyset$), on ajoute la transition $t^{\cdot i}$ représentant la transition t à l'ensemble T_e ainsi que la préservation des ses connexions avec $p \in P$ dans C_i. Il est indispensable de connecter $t^{\cdot i}$ à p°_i avec une auto-boucle, afin de ne rendre cette activation que si le système reconfigurable est dans la configuration C_i. Le but de la duplication est de modéliser le changement de ses conditions pré / post par des

reconfigurations. Deuxièmement, si ${}^{\bullet}t_{Ci} = t^{\bullet}_{Ci} = \emptyset$ (c'est-à-dire, que t est neutralisée en C_i) alors on omet t, ce qui signifie qu'il n'y a pas de représentation de $t \in T_{Ci}$ dans G_e.

Enfin, compte tenu des règles de reconfiguration, chaque règle $r = <D, {}^{\bullet}r, r^{\bullet}, \mathbb{C}, \mathbb{M}, V>$ dans le R-SPN $N = <P, T, R, \gamma_0>$ sera représentée par une transition r^j_i, où C_i et C_j sont les configurations source et cible, respectivement. Dans le R-SPN N, une règle de reconfiguration r peut être appliquée lorsque les relations de flux dans la configuration courante C_i des places dans D sont identiques à ${}^{\bullet}r$, *et* $M(p) \geq M(p)$, $\forall p \in C$ (ici, M est le marquage courant de C_i). En conséquence, les pré-conditions de la transition r^j_i dans $G_e = (P_e, T_e, F_e, M_{e0}, \Lambda_e)$ sont: $M_e(p) \geq M(p)$, $\forall p \in C$ *et* $M_e({}^{\circ}p_i) = 1$, où M_e est le marquage actuel de G^e. Par conséquent, le dépliage de la règle r nécessite ce qui suit:

- Ajouter une auto-boucle entre la transition r^j_i et chaque place $p \in C$, telle que $F_e(p, r^j_i) = F_e(r^j_i, p) = M(p)$,
- Ajouter un arc *de* ${}^{\circ}p_i$ à r^j_i et un arc de r^j_i à ${}^{\circ}p_j$ (c'est-à-dire, $F_e({}^{\circ}p_i, r^j_i) = F_e(r^j_i, {}^{\circ}p_j) = 1$) pour modéliser le passage de la configuration C_i à la configuration C_j.

Un récapitulatif des étapes de transformation d'un modèle R-SPN N vers un modèle SPN G_e équivalent est donné dans l'algorithme 1.

Algorithm 1 From R-SPN to an equivalent SPN

Input: $N = \langle P, T, \mathcal{R}, C_0 \rangle$

Output: Equivalent SPN $G_e = \langle P_e, T_e, F_e, M_{e0}, \Lambda_e \rangle$

Set G_e to empty graph
Create the set $\mathring{P}$
Create the marking $\mathring{M}^0$
Create the set of configurations $\mathcal{G}$
$P_e \longleftarrow P \cup \mathring{P}$
$M_{e0} \longleftarrow M_0 \_ \mathring{M}^0$
for all $t \in T$ **do**
  for all $C_i \in \mathcal{G}$ **do**
    if ${}^\bullet t_{C_i} \neq \emptyset \vee t^\bullet_{C_i} \neq \emptyset$ **then**
      Let $\dot{t}^i$ be the representation of t at G_e
      Let $\mathring{p}_i$ be the associated place to C_i
      $T_e \longleftarrow T_e \cup \{\dot{t}^i\}$
      $F_e(\dot{t}^i, P) \longleftarrow F_{C_i}(t, \cdot)$
      $F_e(P, \dot{t}^i) \longleftarrow F_{C_i}(\cdot, t)$
      $F_e(\mathring{p}_i, \dot{t}^i) \longleftarrow 1$
      $F_e(\dot{t}^i, \mathring{p}_i) \longleftarrow 1$
    end if
  end for
end for
for all $r = \langle D, {}^\bullet r, r^\bullet, \mathbb{C}, \mathbb{M}, V \rangle \in \mathcal{R}$ **do**
  Let r_i^j be the representation of reconfiguration rule from C_i to C_j
  $T_e \longleftarrow T_e \cup \{r_i^j\}$
  $\Lambda_e(r_i^j) \longleftarrow V$
  Let $\mathring{p}_i$ be the associated place to C_i
  Let $\mathring{p}_j$ be the associated place to C_j
  $F_e(\mathring{p}_i, r_i^j) \longleftarrow 1$
  $F_e(r_i^j, \mathring{p}_j) \longleftarrow 1$
  for all $p \in \mathbb{C}$ **do**
    $F_e(p, r_i^j) \longleftarrow \mathbb{M}(p)$
    $F_e(r_i^j, p) \longleftarrow \mathbb{M}(p)$
  end for
end for

Algorithme 1 Transformation d'un modèle R-SPN *N* vers un modèle SPN équivalent

2) Preuves:

Le but de cette sous-section est de prouver l'équivalence entre le R-SPN *N* et le SPN G_e obtenu à partir de l'application de l'algorithme décrit ci-dessus. Pour prouver l'équivalence, nous allons considérer le théorème suivant.

Théorème 1.

Toute séquence franchissable de transitions (qui sont neutralisées) et des règles (applicables) dans N est aussi une séquence franchissable dans son réseau équivalent G_e obtenu par l'algorithme de dépliage et vice versa.

Preuve. Dans la preuve, nous considérons les lemmes suivants.

Lemme 1. Si la configuration actuelle est C_i (c.-à-d. M_e (p°_i) = 1) et qu'une transition t dans C_i est franchissable à M (dans le réseau N), alors $t^{\cdot}_i$ est franchissable à $M\_M^{\alpha i}$ dans le réseau équivalent G_e.

Lemme 2. Si la configuration courante est C_i (*c.-à-d,* M_e (p°_i) = 1) et qu'une règle de reconfiguration r est applicable à partir de C_i à M (dans le réseau N) alors la transition t_i est franchissable à $M\_M^{\alpha i}$, où t_i est la transition correspondante de r (dans le réseau équivalent G_e).

Premièrement, nous démontrons le lemme 1. Soit T^i l'ensemble des transitions $t^i_j \in T_e$ qui sont le dépliage des transitions $t_j \in T_{Ci}$.

Si la configuration courante est C_i (c.-à-d, M_e (p°_i) = 1) alors $\forall t \in T^i$, t est désactivé, car elle est connectée par une auto-boucle avec $p^{\circ}_{j\neq i}$ dont son connexion M_e (p°_j) = 0. Aussi, si $t^{\cdot}_i$ est une représentation de $t \in T_{Ci}$ alors, on a par définition F_e ($t^{\cdot i}$, P) = F_{Ci} (t, ·) ∧ F_e (P, $t^{\cdot i}$) = F_{Ci} (·, t) et F_e ($p^{\circ}_{j,\, j\neq i}$, $t^{\cdot i}$) = 0 ∧ F_e (p°_i, $t^{\cdot i}$) = 1, ce qui signifie que si t dans C_i est franchissable à M, alors $t^{\cdot i}$ est franchissable à $M\_M^{\alpha i}$. Autrement dit, si:

- $t^{\cdot i}$ la représentation de $t \in T_{Ci}$ qui a la condition pre/post comme t, est connectée par une auto-boucle avec p°_i, et M_e (p°_i) = 1, et
- toutes les autres transitions t, qui n'appartiennent pas au dépliage de toutes transitions $t \in T_{Ci}$, est connectée par une auto-boucle avec $^{\circ}p_{k,\, k\neq i}$, et
- la configuration actuelle est C_i (c.-à-d, Me (p°_i) = 1 ∧ M_e ($^{\circ}p_{k,\, k\neq i}$) = 0), et
- t dans T_{Ci} est franchissable à M alors $t^{\cdot i}$ est également franchissable à $M\_M^{\alpha i}$.

Considérant le lemme 2, si la configuration courante est C_i alors $\forall t \in T_e$ qui n'est pas une représentation d'une transition dans C_i et ne modélise pas une règle de reconfiguration à partir de C_i alors t doit être désactivée, puisqu'elle est connectée par une auto-boucle avec ${}^{\circ}p_{k,\, j\neq i}$ que son connexion $M_e\,({p}^{\circ}_j) = 0$. Aussi, si ${r^*}_i$ modélise une règle de reconfiguration r à partir de la configuration C_i au marquage M alors nous avons par définition les pré-conditions de ${r^*}_i$ sont $\forall p \in C_i,\ M_e\,(p) \geq M\,(p)$ *et* $M_e\,({p}^{\circ}_i) = 1$, ce qui signifie que si r est applicable à partir de C_i à M, alors ${r^*}_i$ est applicable à $M\_{M}^{\circ}_i$, où $\forall p \in C,\ M\,(p) \geq M\,(p)$.

De la preuve précédente, il est évident de déduire que si $[M_0 >^0 t_0\ [M_1 >^0 t_1 \ldots [M_x > {r^i}_0\ [M_y >^i t_q \ldots [M_z >^j t_p$ est une séquence franchissable dans le R-SPN N alors $[M_0\_M^{\circ 0} > {t^0}_0\ [M_1\_M^{\circ 0} > {t^0}_1 \ldots [M_x\_M^{\circ 0} > {r^i}_0\ [M_y\_M^{\circ i} > {t^0}_q \ldots [M_z\_M^{\circ j} > {t^j}_p$ est également une séquence franchissable dans G_e, où ${t^i}_j$ est la représentation de la transition $t_j \in T_{Ci}$ dans G_e, et vice versa.

III.2. Exemple illustratif :

Dans cette section, nous démontrons l'application du formalisme proposé par une étude de cas système sur les systèmes reconfigurables. Premièrement, nous décrivons la structure et le comportement du système, puis nous appliquons l'algorithme pour obtenir le SPN équivalent.

Considérons un système composé de deux machines M_1, M_2, deux tampon buf_1 et buf_2 avec une capacité de cinq espaces pour chacun d'eux, et une zone de sortie *exz* d'une capacité de cinq espaces. La machine M_1 (resp. M_2) peut : charger une matière première arrivée du tampon buf_1 (resp. un produit P_1 déjà traité par M_1 de buf_2) ; une fois qu'elle est libre, traiter la matière première pour produire le produit P_1 (resp. décharger le produit dans le tampon buf_2 (resp. à la zone de sortie *exz*).

Le système a trois configurations, C_0, C_1 *et* C_2. En C_0, la machine M_1 produit un produit P_1 qui sera traité par M_2 plus tard pour produire P_2. Alors que M_1 est en train de produire, elle peut échouer, si c'est le cas, le processus de maintenance de M_1 est déclenché et le système doit être reconfiguré en C_1 où la réception de la matière première est arrêtée. Si la machine M_1 est toujours en cours de maintenance et que M_2 a fini de traiter tous les produits P_1 dans le tampon buf_2, le système passe à la configuration C_2, de sorte que la réception de la matière première est redémarrée et la machine M_2 produit un produit P'_2. Dans un premier temps, le système est dans la configuration C_0 qui est représentée sur la

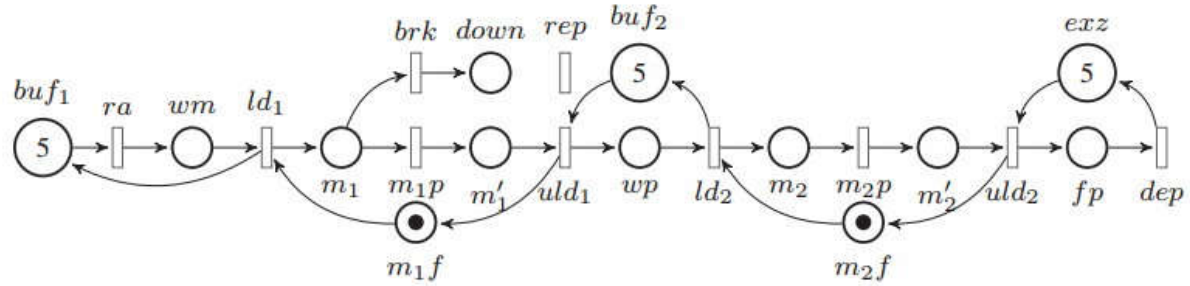
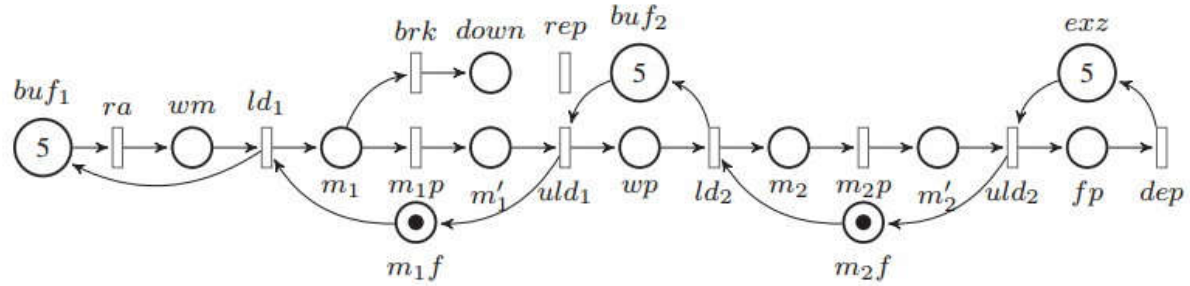


Figure 18.



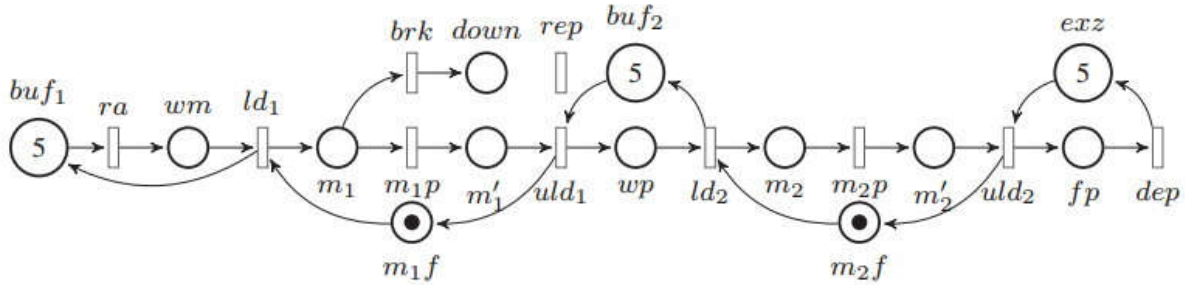


Figure 18 Le modèle SPN pour le système à sa première configuration.

L'interprétation des places et des transitions de C_0 est la suivante:

(i) Places:

a) buf_1 (resp. buf_2): Le nombre de jetons, à l'intérieur de cette place, modélise le nombre d'espaces tampon disponibles dans le tampon buf_1 (resp. buf_2).

b) *wm* (resp. *wp*): Le nombre de jetons dans wm (resp. *wp*) modélise le nombre de matières premières en attente dans le tampon buf_1 (resp. les produits en attente P_1 dans le tampon buf_2).

c) m_1 (resp. m'_1): Un jeton dans m_1 (resp. m'_1) signifie que la machine M_1 a commencé (ou fini) à traiter un produit P_1.

d) m_2 (resp. m'_2): Un jeton en m_2 (resp. m'_2) signifie que la machine M_2 a commencé (ou fini) à traiter un produit P_2.

e) m_1f (resp. m_2f): Un jeton dans m_1f (resp. m_2f) signifie que la machine M_1 (resp. M_2) est inactive.

f) *down*: un jeton dans *down* signifie que la machine M_1 est tombée en panne.

g) *exz*: Le nombre de jetons, à l'intérieur de cette place, modélise le nombre d'espaces disponibles dans la zone de sortie *exz*.

h) *fp*: Le nombre de jetons dans *fp* modélise le nombre de produits finis en attente dans la zone de sortie *exz*.

(ii) Transitions :

a) *ra* (*rate* = λ_{ra})*:* La matière première est chargée dans le tampon buf_1.

b) ld_1 (*rate* = λ_{ld1}) (resp. uld_1 (*rate* = λ_{uld1})): M_1 charge (ou décharge) un élément du tampon buf_1 (resp. du tampon buf_2).

c) ld_2 (*rate* = λ_{ld2}) (resp. uld_2 (*rate* = λ_{uld2})): M_2 charge (ou décharge) un élément du tampon buf_2 (resp. pour quitter la zone *exz*).

d) m_1p (*rate* = λ_{m1p}) (resp. m_2p (*rate* = λ_{m2p})): La machine M_1 (resp. M_2) traite un produit P_1 (resp. P_2).

e) *dep* (*rate* = λ_{dep}): Un produit fini quitte la zone de sortie.

f) *brk* (*rate* = λ_{brk}): La machine M_1 échoue.

g) *rep* (*rate* = λ_{rep}): Réparation de la machine M_1.

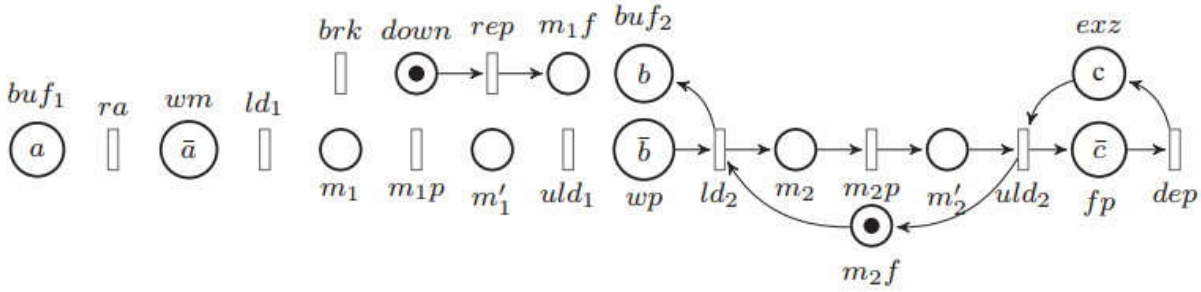


Figure 19 Le modèle SPN pour la configuration du système C_1 lorsque la machine M_1 tombe en panne

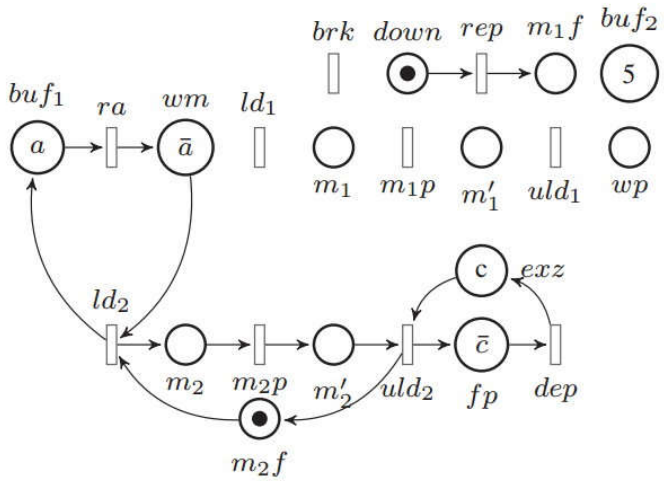


Figure 20 Le modèle SPN pour la configuration du système C_2 où la machine M_2 produit P'_2.

Lorsque la machine M_1 est en panne (c.-à-d., M (*down*) = 1), le système doit passer à C_1 représentée sur la

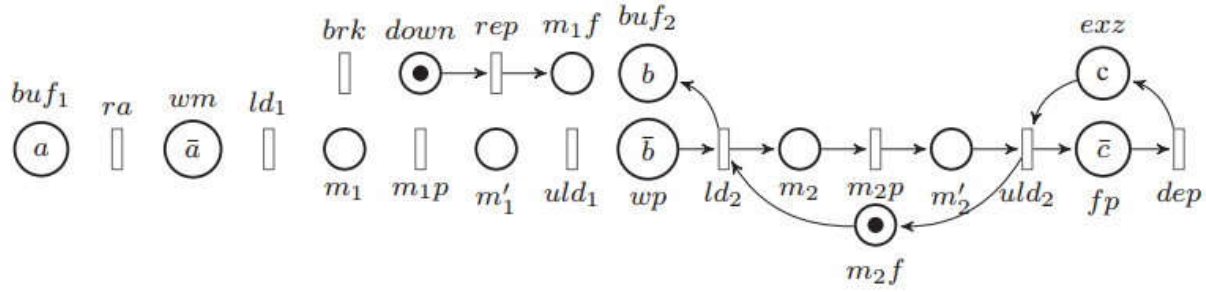


Figure 19, ainsi la réception de matière première doit s'arrêter et M_1 doit être réparée. C_1 est obtenue en appliquant le $r_0 = <D_0, \bullet r_0, r\bullet_0, \mathbb{C}_0, \mathbb{M}_0, V_0>$ sur C_0 où:

(i) $D_0 = \{buf_1, wm, m_1, m_1f, down, m'_1, buf_2, wp\}$,

(ii) $\bullet r_0$: $buf_1\ (ld_1 - ra) + wm\ (ra - ld_1) + m_1\ (ld_1 - m_1p - brk) + m_1f\ (uld_1 - ld_1) + down\ (brk) + m'_1\ (m_1p - uld_1) + buf_2\ (ld_2 - uld_1) + wp\ (uld_1 - ld_2)$,

(iii) $r\bullet_0$: $buf_1\ (\emptyset) + wm\ (\emptyset) + m_1\ (\emptyset) + m_1f\ (rep) + down\ (-rep) + m'_1\ (\emptyset) + buf_2\ (ld_2) + wp\ (-ld_2)$,

(iv) $C_0 = \{down\}$,

(v) $M_0\ (down) = 1$,

(vi) V_0 le délai d'activation de r_0.

Une fois la machine M_1 est réparée, la production de P_1 peut être redémarrée. C_1 est reconfigurée en C_0 en appliquant $r_1 = <D_1, \bullet r_1, r_1\bullet, \mathbb{C}_1, \mathbb{M}_1, V_1>$ où

(i) $D_1 = \{buf_1, wm, m_1, m_1f, down, m'_1, buf_2, wp\}$,

(ii) $\bullet r_1$: $buf_1\ (\emptyset) + wm\ (\emptyset) + m_1\ (\emptyset) + m_1f\ (rep) + down\ (-rep) + m'_1\ (\emptyset) + buf_2\ (ld_2) + wp\ (-ld_2)$,

(iii) $r_1\bullet$: $buf_1\ (ld_1 - ra) + wm\ (ra - ld_1) + m_1\ (ld_1 - m_1p - brk) + m_1f\ (uld_1 - ld_1) + down\ (brk) + m'_1\ (m_1p - uld_1) + buf_2\ (ld_2 - uld_1) + wp\ (uld_1 - ld_2)$,

(iv) $C_1 = \{m_1f\}$,

(v) $M_1\ (m_1f) = 1$,

(vi) V_1 le délai d'activation de r_1.

Lorsque la machine M_1 est en panne et que la machine M_2 a traité toutes les pièces en attente dans buf_2, le système passe à C_2 représentée sur la Figure 20, d'où le M_2 peut produire P'_2. C_2 est obtenue en appliquant le $r_2 = <D_2, \bullet r_2, r_2 \bullet, \mathbb{C}_2, \mathbb{M}_2, V_2>$ sur C_1 où:

(i) $D_2 = \{buf_1, wm, buf_2, wp\}$,

(ii) $\bullet r_2$: buf_1 $(\emptyset)$ + wm $(\emptyset)$ + buf_2 (ld_2) + wp $(-ld_2)$,

(iii) $r_2 \bullet$: buf_1 $(ld_2 - ra)$ + wm $(ra - ld_2)$ + buf_2 $(\emptyset)$ + wp $(\emptyset)$,

(iv) $C_2 = \{down, buf_2\}$,

(v) M_2 $(down) = 1$, M_2 $(buf_2) = 5$,

(vi) V_2 le délai d'activation de r_2.

C_2 est reconfigurée en C_0 une fois la machine M_1 est réparée, en appliquant $r_3 = <D_3, \bullet r_3, r_3 \bullet, \mathbb{C}_3, \mathbb{M}_3, V_3>$ où

(i) $D_3 = \{buf_1, wm, m_1, m_1f, down, m'_1, buf_2, wp\}$,

(ii) $\bullet r_3$: buf_1 $(ld_2 - ra)$ + wm $(ra - ld_2)$ + m_1 $(\emptyset)$ + m_1f (rep) + $down$ $(-rep)$ + m'_1 $(\emptyset)$ + buf_2 $(\emptyset)$ + wp $(\emptyset)$,

(iii) $r_3 \bullet$: buf_1 $(ld_1 - ra)$ + wm $(ra - ld_1)$ + m_1 $(ld_1 - m_1p - brk)$ + m_1f $(uld_1 - ld_1)$ + $down$ (brk) + m'_1 $(m_1p - uld_1)$ + buf_2 $(ld_2 - uld_1)$ + wp $(uld_1 - ld_2)$,

(iv) $C_3 = \{m_1f\}$,

(v) M_3 $(m_1f) = 1$,

(vi) V_3 le délai d'activation de r_3.

Maintenant, nous considérons l'aspect de la vérification. Pour analyser les propriétés du système, nous créons d'abord le SPN G_e équivalent du R-SPN $N = <P, T, R, C_0>$ décrit ci-dessus, où:

(i) $P = \{buf_1, wm, m_1, m'_1, m_1f, buf_2, wp, m_2, m'_2, m_2f, down, fp, exz\}$,

(ii) $T = \{ra, ld_1, m_1p, uld_1, ld_2, m_2p, uld_2, dep, brk, rep\}$,

(iii)C_0 est la configuration initiale représentée sur la

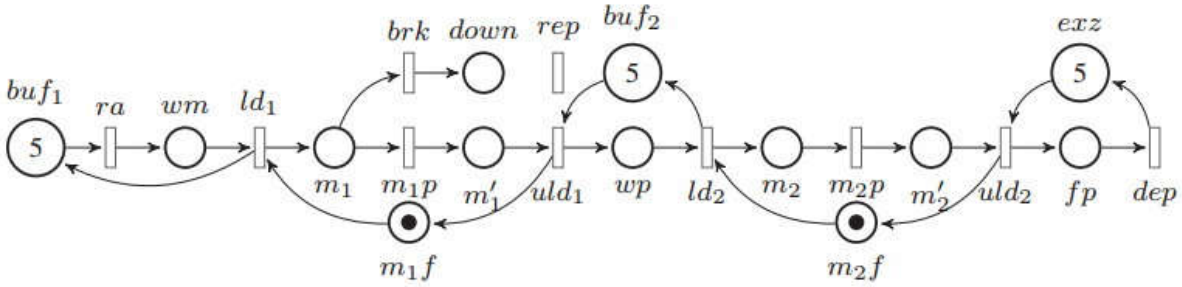


Figure 18,

(iv)$R = \{r_0, r_1, r_2, r_3\}$ est l'ensemble des règles de reconfiguration données ci-dessus.

L'application de règles dans R sur N donne $G = \{C_0, C_1, C_2\}$ l'ensemble des configurations finies, où C_1 est représentée sur la

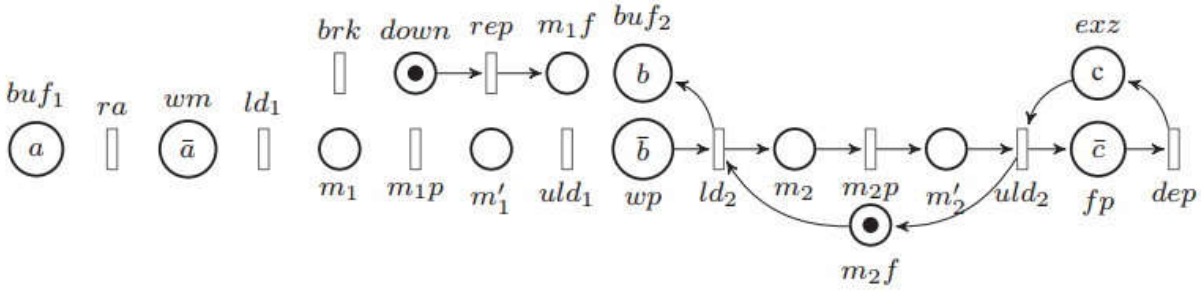


Figure 19 et C_2 est représentée sur la Figure 20.

Le SPN équivalent $G_e = (P_e, T_e, F_e, M_{e0}, \Lambda_e)$ peut être construit en utilisant l'algorithme décrit précédemment. Pour obtenir $P_e = P \cup P^\circ$, on crée $P^\circ = \{^\circ p_0, {}^\circ p_1, {}^\circ p_2\}$ où $^\circ p_0$, $^\circ p_1$ et $^\circ p_2$ sont associés à C_0, C_1 et C_2, respectivement. Le marquage initial des places dans P est M°_{e0} $(^\circ p_0, {}^\circ p_1, {}^\circ p_2) = (1, 0,0)$.

Pour ajouter et connecter les transitions dans T, nous donnons trois exemples à propos de ra, ld_2 et ld_1, les autres transitions peuvent être ajoutées et connectées de la même manière. Nous pouvons observer que ld_1 n'est connecté par aucun arc en C_1 et C_2 ce qui signifie que ld_1 est neutralisée dans les deux configurations. Par conséquent, ld_1 aura une représentation unique dans G_e ($T_e \leftarrow T_e \cup \{ld^0_1\}$) et les connexions entre ld^0_1 et $p \in P$ sont les mêmes que dans C_0, donc F_e $(ld^0_1, m_1) = F_e$ $(ld^0_1, buf_1) = F_e$ $(wm, ld^0_1) = F_e$ $(m_1f, ld^0_1) = 1$, sinon vaut 0. Quant aux connexions entre ld^0_1 et $p \in P^\circ$, on a F_e $(ld^0_1, {}^\circ p_0) = F_e$ $(^\circ p_0, ld^0_1) = 1$ et, F_e $(ld^0_1, {}^\circ p_1) = F_e$ $(ld^0_1, {}^\circ p_2) = F_e$ $(^\circ p_1, ld^0_1) = F_e$ $(^\circ p_2, ld^0_1) = 0$, elles désactivent ld^0_1 lorsqu'il n'y a pas de jeton marquant la place $^\circ p_0$, indiquant que la configuration actuelle est C_1 ou C_2. Prenant en compte ra qui a des relations de flux dans les deux configurations C_0 et C_2 et aucune relation

de flux dans C_1, elle a donc deux représentations ra_0 et ra_2 dans G_e ($T_e \leftarrow T_e \cup \{ra^0, ra^2\}$). Les relations entre ra^0 et $p \in P_e$ sont données comme suit, F_e (ra^0, wm) = F_e (buf_1, ra^0) = F_e (ra^0, $^\circ p_0$) = F_e ($^\circ p_0$, ra^0) = 1, sinon est 0. ra^2 est relié à $p \in P_e$ par F_e (ra^2, wm) = F_e (buf_1, ra^2) = F_e (ra^2, $^\circ p_2$) = F_e ($^\circ p_2$, ra^2) = 1, sinon vaut 0. La transition ld_2 a les mêmes pré/post-conditions en C_0 et C_1 (qui ne sont pas vides) et des pré/post-conditions différentes en C_2. En conséquence, ld_2 aura trois représentations dans G_e comme suit,

(i) $ld^0{}_2$ ($T_e \leftarrow T_e \cup \{ld^0{}_2\}$) dont les relations de flux sont F_e ($ld^0{}_2$, m_2) = F_e ($ld^0{}_2$, buf_2) = F_e (wp, $ld^0{}_2$) = F_e (m_2f, $ld^0{}_2$) = F_e ($ld^0{}_2$, $^\circ p_0$) = F_e ($^\circ p_0$, $ld^0{}_2$) = 1, sinon vaut 0,

(ii) $ld^1{}_2$ ($T_e \leftarrow T_e \cup \{ld^1{}_2\}$) dont les relations de flux sont F_e ($ld^1{}_2$, m_2) = Fe ($ld^1{}_2$, buf_2) = F_e (wp, $ld^1{}_2$) = F_e (m_2f, $ld^1{}_2$) = F_e ($ld^1{}_2$, $^\circ p_1$) = F_e ($^\circ p_1$, $ld^1{}_2$) = 1, sinon vaut 0,

(iii) $ld^2{}_2$ ($T_e \leftarrow T_e \cup \{ld^2{}_2\}$) dont les relations de flux sont F_e ($ld^2{}_2$, m_2) = Fe ($ld^2{}_2$, buf_1) = F_e (wm, $ld^2{}_2$) = F_e (m_2f, $ld^2{}_2$) = F_e ($ld^2{}_2$, $^\circ p_2$) = F_e ($^\circ p_2$, $ld^2{}_2$) = 1, sinon vaut 0.

Enfin, nous ajoutons quatre transitions $r_0{}^1$, $r_1{}^0$ pour modéliser la reconfiguration de C_0 à C_1 et de C_1 à C_0, et $r_1{}^2$, $r_0{}^2$ pour modéliser la reconfiguration de C_1 à C_2 et de C_2 à C_0. A titre d'exemple, nous considérons $r_0{}^1$, elle modélise le passage à C_1 lorsque la machine M_1 est en panne et la configuration actuelle est C_0, donc F_e ($r_0{}^1$, $down$) = F_e ($down$, $r_0{}^1$) = 1 et F_e ($^\circ p_0$, $r_0{}^1$) = F_e ($r_0{}^1$, $^\circ p_1$) = 1. Le SPN équivalent G_e obtenu est illustré sur la Figure 21.

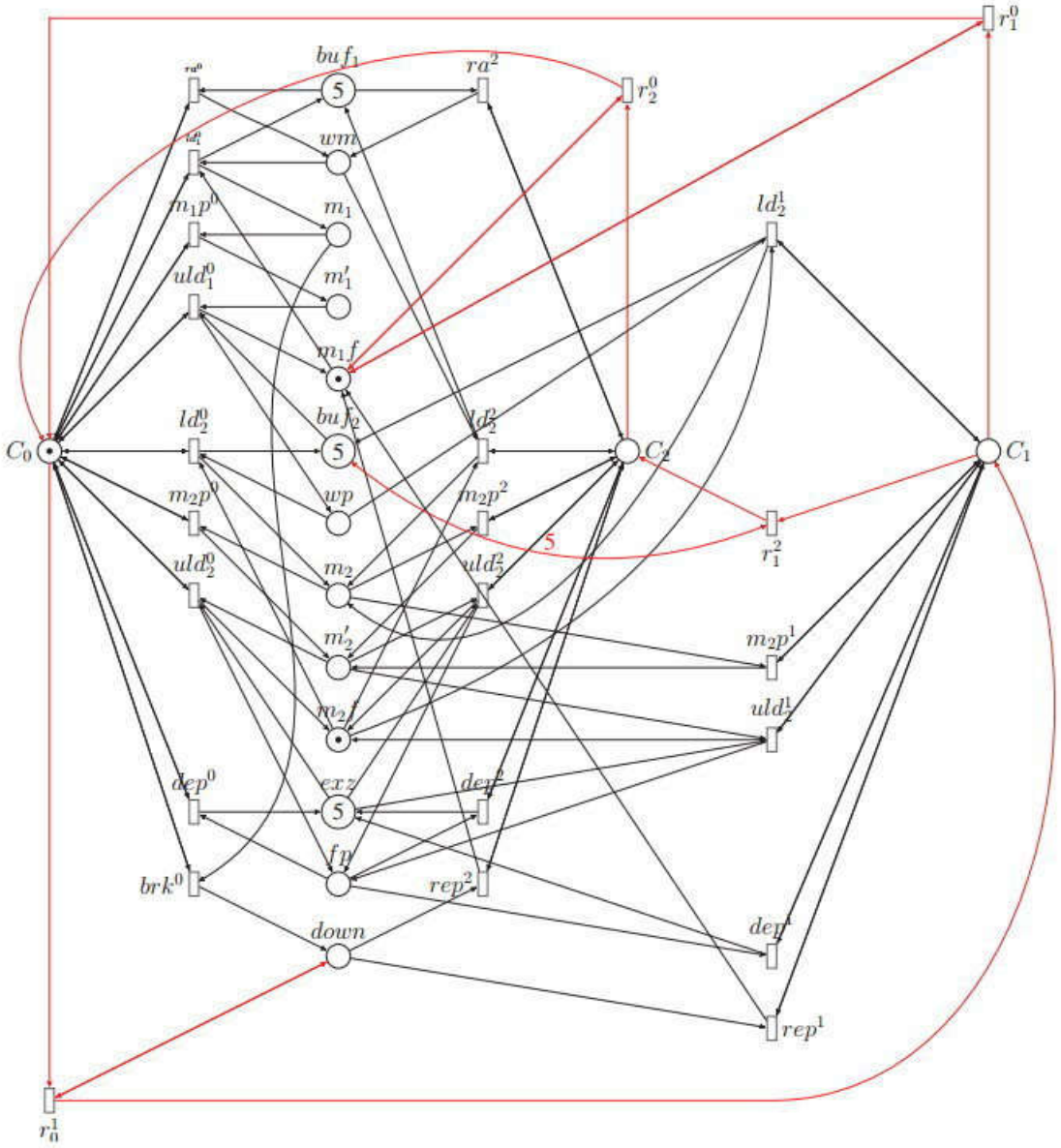


Figure 21 SPN équivalent.

Mise en œuvre

IV.1. Introduction

Dans le chapitre précédent, on a présenté la reconfiguration dans les réseaux de Petri stochastique d'où on a vue la transformation d'un modèle R-SPN N vers un modèle SPN G_e équivalent, d'autre côté on a remarqué le manque des outils qui aides a modalisé ses transformations, de plus l'algorithme utilisé applique le dépliage qui nécessite une capacité de mémoire et vitesse de processeur puissante (matériel puissant) ; On a trouver aussi que la majorité des outils existants qui modalisent des RdP s'exécutent sur des monoposte ; pour ce là on propose de réaliser ou de construire l'outil désiré dans une architecture qui nous aide a aboutir nous besoins.

IV.2. Présentation de l'outil:

Pour les raisons demandées, on a choisi le Web comme une plateforme pour le développement de cet outil d'où chacun peut en profiter sur son poste personnel qui nécessite simplement une connexion Internet et un exploiteur de Web ; de plus l'outil est ouvert pour les développeurs qui peuvent profiter a l'utiliser pour leurs propre besoins.

IV.2.1. Schéma de cas d'utilisation :

L'outil est composer en deux parties, la première partie est pour la modalisation de R-SPN qu'utilise l'exploiteur de Web qui ne demande pas de puissance de matériel, par contre la deuxième partie nécessite cette puissance de matériel pour l'exécution de la transformation; qui sera exécutée dans un serveur puissant. Le résultat sera envoyé a l'utilisateur sur son exploiteur pour le nouveau SPN équivalant comme démontre la

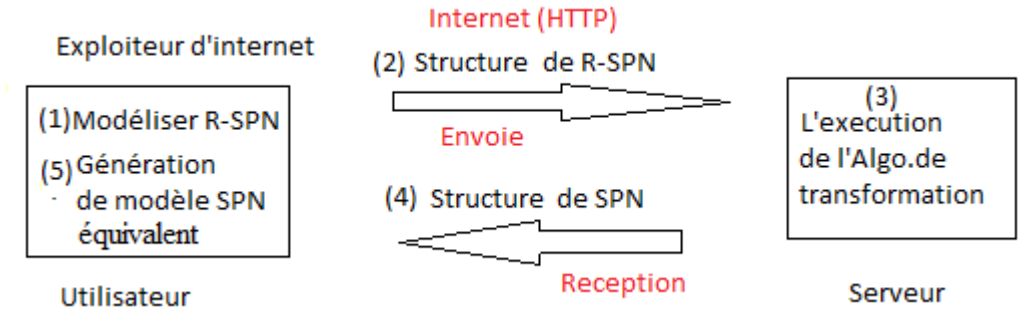


Figure 22.

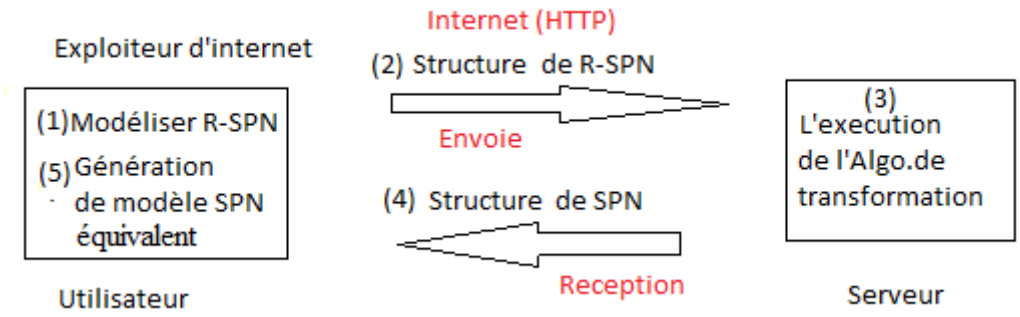


Figure 22 Le schéma des étapes de transformation d'un modèle R-SPN vers un modèle SPN équivalent.

IV.2.2. Architecture global :

1. Côté Utilisateur :

L'utilisateur a une interface Web pour modéliser son R-SPN, cette interface est composée en des composants HTML (buttons, options de sélection) pour sélectionner les actions à faire (Ajout des places, transitions ou des arcs ; suppression, sélection, exécution,.. etc.) et une zone graphique (Canvas) pour la modélisation de R-SPN selon les actions choisies. Lors de la modélisation, une structure est associer au modèle qui se met à jour avec les actions choisies. A fin de terminer sa modélisation, il envoie le modèle au serveur pour l'exécution de transformation en cliquant sur un buttons d'envoie, où la structure est envoyée au serveur. Après la transformation, le serveur renvoie une autre structure de modèle SPN équivalant qui sera adaptée avec la structure des objets Web (places, transitions, arcs) pour être affichés au utilisateur d'où il peut modifier ou choisir d'autres tâches.

L'utilisateur peut être un développeur, où il a la main d'ajouter des sous structures à la structure principale ou bien il ajoute des propriétés aux objets (places, transitions, arcs) qui peuvent être envoyés au serveur choisi. Comme il peut aussi ajouter des composants Web ou bien il change totalement le design (l'interface de l'outil) en associant les composants Web choisis aux propriétés demandées de notre objet. Cet outil est développé de sorte qu'il soit extensible pour d'autres fonctionnalités.

2. Côté serveur :

Après la modélisation de R-SPN par l'utilisateur, le serveur reçoit une structure (structure de base + structure extensible) qui sera manipulée selon le langage de

programmation choisi. Cette structure est reçue par le module où on va exécuter la transformation de R-SPN vers un SPN. Après la transformation, une nouvelle structure est générée et renvoyée à l'utilisateur pour être affiché sur son outil. Pour un développeur, il peut créer un autre module pour la réception de la structure et après les traitements il renvoie n'importe quelle structure qu'il a besoin d'utilisé plu-tard.

IV.2.3. Architecture détaillé :

L'outil développé comme on a présenté avant est déployé sur une plateforme Web. L'architecture de l'outil (côté utilisateur) est composés d'une interface Web où on utilise le langage HTML avec l'aide des bibliothèques (Jquery, Bootstrap) et des fiches de styles CSS, où les composants Web nécessaires sont reliés à notre objet comme propriétés qui sont initialisés lors du chargement de la page Web. Cet objet qui est développé sous le langage Javascript, contient quatre parties, une partie pour l'initialisation de la structure (places, transitions, arcs) ainsi la configuration par défauts de style (couleur arrière plan, couleur de bordure, taille de rayon, langueur et largeur, … etc.) des places , transitions et des arcs, aussi les composants HTML reliés à notre objet (Canvas, bouton ajout, suppression, sélection, etc.) comme montre la Figure 23. Autre partie contient les procédures de design graphique sur le Canvas (dessin de cercle, rectangle, arcs, etc.) qui eux même sont appelés par des procédures pour dessiner les places, transitions et les arcs (exemple pour une place : dessiner une cercle qui contiens le nom et le marquage de la place ; troisième partie contiens les procédures et les événements pour les traitements ; lorsqu'on clique sur un bouton, sur la souris ; envoie de la structure, etc.) ; la dernière partie contient les procédures qui génèrent les composants HTML à partir de notre structure pour la sélection ou la modification des propriétés. D'autres optionalités ont été ajoutées à cet outil (Zoomer, exporter et importer la structure, exporter le modèle comme image).

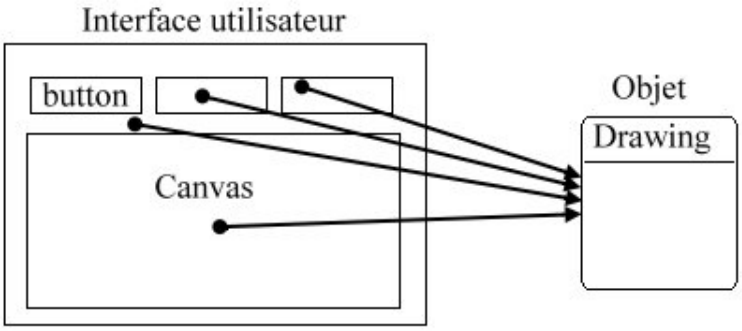


Figure 23 Relier les composants HTML de l'interface a notre objet.

On choisit le langage PHP pour l'exécution de la transformation de côté serveur, est un langage puissant et simple à manipuler. D'où on reçoit la structure de côté utilisateur sous forme de texte de format JSON (JavaScript Object Notation) qui est utilisé ou manipulé par plusieurs langages de programmation. Aussi, le développeur peut choisir d'autres langages de programmation selon son besoin (Exemple Python), il suffi de convertir cette structure de format JSON à une autre structure. Après la transformation, le résultat de transformation de R-SPN au SPN est une nouvelle structure qui sera renvoyé à notre outil (côté utilisateur) qui est elle-même de format texte JSON. Dans le côté utilisateur, une extension (développée sous Javascript) reçoit la nouvelle structure et la réaffiche (réinitialisation de notre objet par la nouvelle structure) comme nouvel modèle (SPN) d'où on peut appliquer d'autres opérations sur le modèle résultant.

IV.2.4. Structure :

1. Interface principale:

L'interface principale contient : les modules « Jquery-3.2.1.js, Jquey-ui.js» qui nous facilite la manipulation des composants HTML (buttons, selects, etc.), le module «bootstrap-3.3.7» pour simplifier la tâche de design avec des styles CSS. On initialise notre objet « drawing » qui se trouve dans le module « drawing-v1.23.js » lors du chargement de la page comme il est présenté ci-dessous, ainsi le module extension « ext_reconfig .js» qui associe la structure des règles de transformations (rules) au structure de base pour les envoyer au serveur qui recevra lui-même la nouvelle structure de SPN qui remplace l'ancienne structure.

Initialisation de l'objet « drawing » et le module « ext_reconfig.js »:

```
var draw = new drawing({
```

```
drawing_zone: $("#canvas"),
html_objs: {
   places: $("#HTML_places"),
   transitions: $("#HTML_transitions"),
   arcs: $("#HTML_arcs")
},
html_detail: $("#HTML_detail"),
html_buttons: {
   state: $("#state"),
   insert_place: $("#bt-insert-place"),
   insert_transition: $("#bt-insert-transition"),
   insert_arc: $("#bt-insert-arc"),
   select: $("#bt-select"),
   select_all: $("#bt-select-all"),
   deselect_all: $("#bt-deselect-all"),
   activate_region: $("#bt-region"),
   drag: $("#bt-drag"),
   remove: $("#remove"),
   reset: $("#reset"),
   export: $("#export"),
   send: $("#send"),
   functions_list: $("#functions-list"),
   zoom_in: $("#zoom_in"),
   zoom_out: $("#zoom_out"),
   width_input: $("#width"),
   height_input: $("#height")
```

```
    },
    grid_size: 25
});
//-----------------------------------------
ext_reconfig(draw);
```

2. L'objet « Drawing » :

L'objet «drawing» est le cœur de l'outil qui peut être utilisé par n'importe quel développeur, juste on fait l'instancier à partir du module « drawing-v1.23.js » avec l'initialisation de ses propriétés ; l'objet contient les structures des places, transitions, arcs, comme il est présenté ci-dessous ; ainsi les procédures et les événements de manipulations des objets créés (places, transitions, arcs, régions, etc.) ; où il associe les structures aux objets dessinés et les composants HTML. La création des places, transitions, arcs et les régions sont faites par des propriétés par défauts qui peuvent être modifiés ou manipulés lors de l'utilisation.

Structures des places, transitions et arcs :

Places_list = Liste des objets place ;

Transitions_list = Liste des objets transition ;

Arcs_list = Liste des objets arc ;

```
Place = Objet {
    type: 'place',
    id: -1,
    name: 'P',
    marking: 0,
    br_col: default_place_br, //Couleur de bordure
    bg_col: default_place_bg, // Couleur de l'arrière plan
```

```
    br_w: default_place_br_w, // longueur de bordure
    r: default_place_r, // Rayon de cercle de place
    x: 0,
    y: 0,
    selected: false,
    connectors: [{ x: 0, y: 0 }, { x: 0, y: 0 }, { x: 0, y: 0 }, { x: 0, y: 0 }],
    config: 0
}
Transition = Objet {
    type: 'transition',
    id: -1,
    name: 'T',
    function: '',
    rate: '1',
    br_col: default_transition_br,
    bg_col: default_transition_bg,
    br_w: default_transition_br_w,
    width:  default_transition_width,
    height: default_transition_height,
    x: 0,
    y: 0,
    selected: false,
    connectors: [{ x: 0, y: 0 }, { x: 0, y: 0 }, { x: 0, y: 0 }, { x: 0, y: 0 }],
    config: 0
}
Arc = Objet {
```

```
    type: 'arc',
    id: -1,
    name: 'A',
    weight: 1,
    br_col: default_arc_br,
    br_w: default_arc_br_w,
    lines: [{ x: 0, y: 0 }, { x: 0, y: 0 }],
    from: null,
    to: null,
    selected: false,
    arc_type: arc_types.line,
    config: 0
}
```

Présentation de quelques méthodes de l'objet « Drawing »:

1) Méthodes de dessin :

```
function  draw_place (place){
    draw_circle(place.name, place.marking, place.br_col, place.bg_col, place.br_w,
place.x, place.y, place.r, place.selected);
}
function  draw_transition(transition) {
    draw_rectangle(transition.name, transition.rate, transition.br_col, transition.bg_col,
transition.br_w, transition.x, transition.y, transition.width, transition.height,
transition.selected);
}
function draw_arc(arc){
```

```
        p_from = calculate_connector(arc.from) ;

        p_to =  calculate_connector(arc.to) ;

        draw_arrow(arc_type, arc.name, arc.weight, arc.br_col, arc.br_w, arc.br, arc.br,
default_arc_arrow_s, p_from, p_to, false);

}

function   draw_region(region) {

      draw_rectangle2(region.name, region.br_col, region.bg_col, region.br_w, region.x,
region.y,   region.width, region.height, region.selected);

   }


function draw_all(){

draw_all_places() ;

draw_all_transitions() ;

draw_all_arcs() ;

}
```

2) Méthodes et événements pour la gestion de l'outil :

```
function  OnMouseDown (e) {

if (action == 'select') {

  selected_object = GetSelectedObj(vPos.x, vPos.y);

}

if (action == 'insert') {

          switch (inserting_object_type) {

            case 'place':

                var new_place = Clone(default_place);

                add_place(new_place);

            case 'transition':

                 var new_transition = Clone(default_transtion);
```

```
                    add_transition(new_transition);
                case 'arc':
                    var new_arc = Clone(default_arc);
                    add_arc(new_arc);
            }
        }
        if (action == 'drag') {
            selected_object = GetSelectedObj(vPos.x, vPos.y);
            BeginDrag = true;
        }
        }

function OnMouseUp (e) {
            if ((selected_object != undefined) && (action != 'select')) {

                BeginDrag = false;
                SetSelectedObj(false);
                selected_object = undefined;
            }
            draw_all();
}

function  WebSend (to_url, recive_to) {
        var draw_parent = this;
        $.ajax({
            url: to_url,
```

```
        type: 'POST',
        data: draw_parent.objects_list_to_send,
        success: function(result) {
            $(recive_to).html(result);
        },
        error: function(error) {
            $(recive_to).html(error);
        }
    });
}

function  HTML_init(objs) {
    switch (objs) {
        case 'place':
            HTML_init_places();
            break;
        case 'transition':
            HTML_init_transitions();
            break;
        case 'arc':
            HTML_init_arcs();
            break;
        case 'all':
            HTML_init_places();
            HTML_init_transitions();
            HTML_init_arcs();
```

```
            break;
        }
    }
```

3. L'extension « ext_reconfig » :

La fonction « ext_reconfig » déclarée dans le module « ext_reconfig.js » ajoute une structure des regèles de transformation au structure de base ; après l'envoi de structures au serveur pour le traitement, il reçoit la nouvelle structure de SPN qui réaffecte l'enceins structure pare cette dernière.

Structure de l'extension :

```
rules_list = Liste des objets rule;
configs = Liste des configurations existantes;
default_rule = {
    type: 'rule',
    id: -1,
    name: 'r',
    c_source: -1,
    c_target: -1,
    marking: [], // Liste de marquages
    v: '1',
};


function appliquer_les_regles() {
        Remplir_le_Liste(rules_lit) ;
        Remplir_le_Liste(configs) ;
        draw.objects_list_to_send['configs'] = ConvertToJSON_Format(configs);
        draw.objects_list_to_send['rules_list'] = ConvertToJSON_Format(rules_list);
}
function  reception_resultat(){
    draw.places_list = ConvertJSON_ToObject(new_ places_list);
    draw.transitions_list = ConvertJSON_ToObject(new_transitions_list);
    draw.arcs_list = ConvertJSON_ToObject(new_arcs_list);
    draw.HTML_init('all');
    draw.draw_all();
}
```

4. Module serveur :

Ce module reçoit la structure de modèle R-SPN envoyé par l'outil ; il applique la transformation d'où on aura une nouvelle structure de SPN qui sera renvoyé à l'outil.

Structure des places, transitions, arcs :

new_places = Liste des objet place;

new_transitions = Liste des objets transition;

new_arcs = Liste des objets arc;

Objet place = ["id"=>","name"=>","marking"=>","config"=>0];

Objet transition = ["id"=>","name"=>","rate"=>","config"=>0];

Objet arc = ["id"=>","name"=>","weight"=>","from"=>$obj,"to"=>$obj, "config"=>0];

Méthode :

```
function Application_Algo(){
    $places = FromJSON_ToObject(places);
    $transitions = FromJSON_ToObject (transitions);
    $arcs = FromJSON_ToObject (arcs);
    $configs = FromJSON_ToObject(configs);
    $rules = FromJSON_ToObject(rules_list);

    init_places($places);
    add_configs_to_places($configs)
    init_transitions($transitions,$arcs); // get transitions who have IN or OUT arc
    add_rules_to_transitions($rules);

    init_arcs($arcs,$places,$new_transitions);
```

```
        add_arcs_rules($rules);
        add_arcs_configs_to_transitions($configs,$new_transitions);
        $result['places'] =  $new_places;
        $result['transitions'] = $new_transitions;
        $result['arcs'] = $new_arcs;

        Renvoie (FromObjetToJSON($result));
    }
```

IV.3. Interfaces de l'outil:

L'interface principale est une page html simple, qui s'ouvre dans un navigateur web, qui contient les outils nécessaires pour la modélisation de réseau de Petri stochastique reconfigurable ; plusieurs palettes sont utilisées ; la principale palette est la palette « Utils » qui contient les bottons pour l'ajout, suppression et sélection des places, transition et arc ; elle contient aussi d'autres fonctionnalités comme le zoom, changement de taille de zone de graphe, ainsi des buttons pour l'exportation et l'importation de graphe ; la deuxième palette « objets Liste » contient les listes des places, transitions, arcs pour les sélectionner ; la troisième palette « Details » est la palette qui contient les propriétés des objets sélectionnés pour être modifiés, comme exemple (le nom, marquage d'une place) ; en fin une zone de dessin « Drawing » où on modélise notre R-SPN.

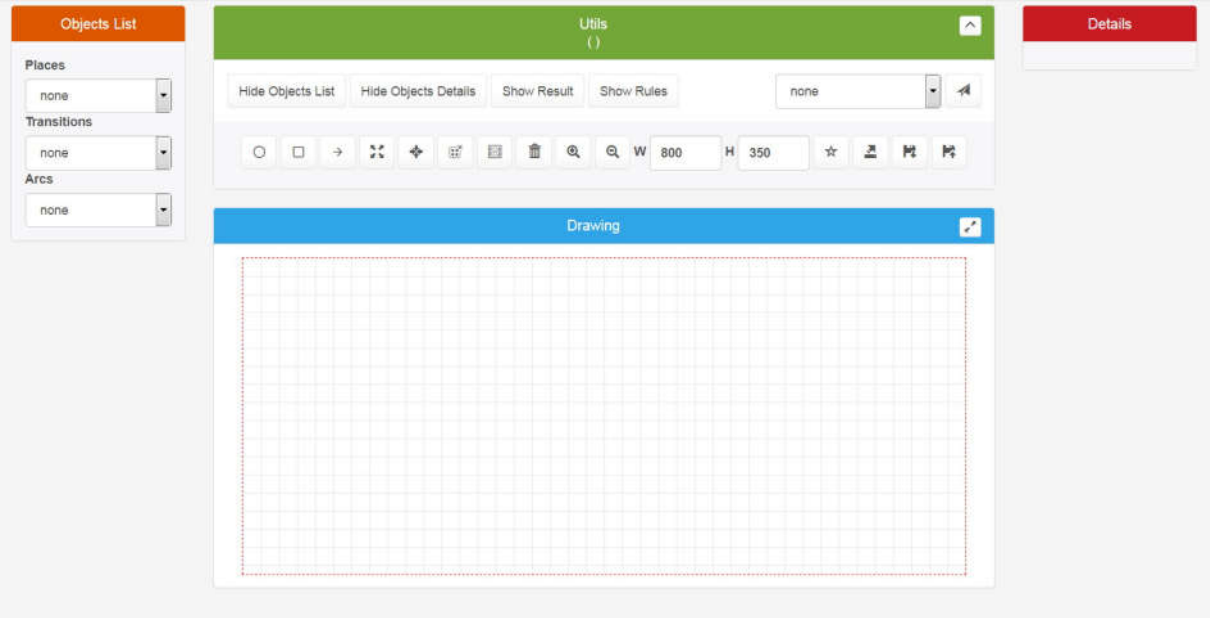
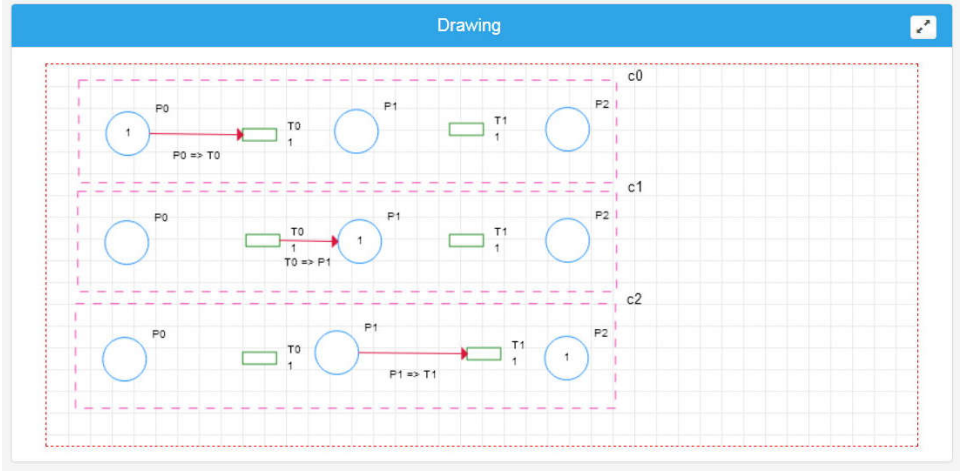


Figure 24 Interface principale de l'outil.



- La
- Figure 25 démontre un réseau de Petri Stochastique reconfigurable comme exemple ; les cercles en couleur bleu ce sont des places, les rectangles en vers ce sont les transitions, les arcs sont des lignes ou des courbes en rouge, chaque configuration est encadrée par un rectangle (région) en mauve.

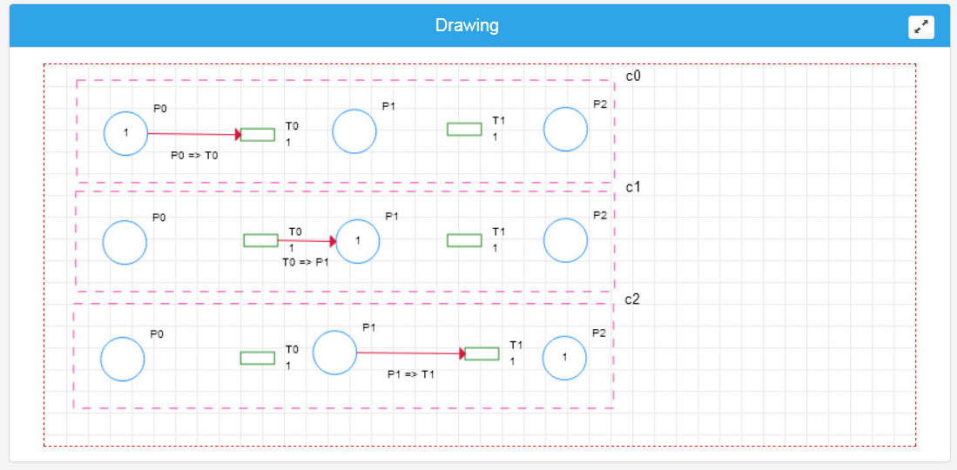


Figure 25 Exemple de réseau de Petri stochastique reconfigurable.

- Pour appliquer les règles de transformation de réseau Petri stochastique reconfigurable, on a la palette « Rules » qui contient un bouton d'ajout et autres pour l'application de ces règles, ainsi chaque règle contient des zones pour remplir les informations nécessaires pour la transformation plus un bouton de suppression d'une règle.

Figure 26 La palette « Rules » pour l'ajout des règles de R-SPN.

- Après la modélisation de notre R-SPN on envoie notre structure au serveur où la transformation sera appliquée. On a une liste de choix (composant HTML « select ») pour sélectionner notre fonction ; car on peut ajouter d'autres fonctions à la liste, comme exemple : une fonction pour avoir un graphe de marquage ; Un bouton « send » pour l'envoi de la structure au serveur.

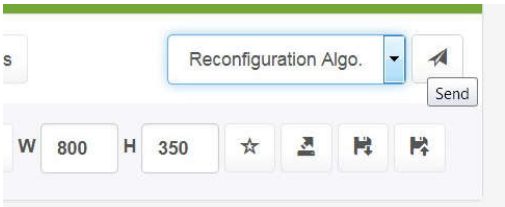


Figure 27 Envoie de structure de R-SPN au serveur.

- Le serveur contient le module occupé par la réception de la structure envoyée et l'application de l'algorithme de transformation ; à la fin il renvoie la nouvelle structure de SPN à l'outil qui sera modélisé en format brute (non organiser) dans la zone de graphe « Drawing » comme montre la figure suivante.

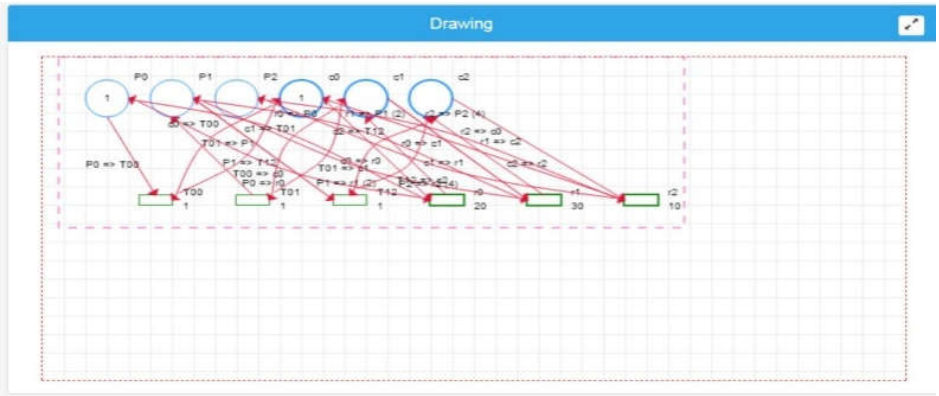


Figure 28 SPN équivalent en format brute (non organiser).

- À l'aide de la palette des outils « Utils », on réorganise notre nouveau modèle SPN, pour qu'il soit bien visualisé, comme montre la figure suivante ; d'où aussi, on peut appliquer d'autres fonctionnalités sur d'autres serveurs.

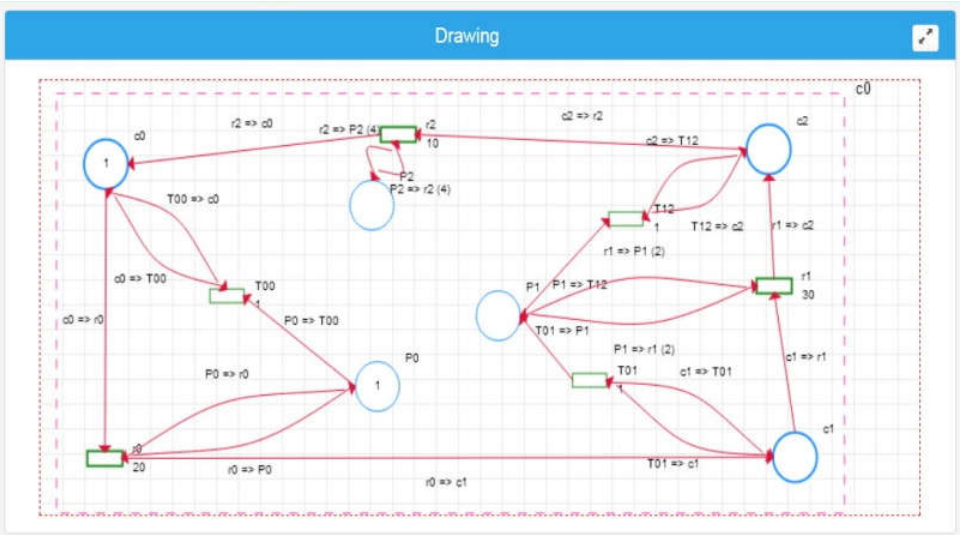


Figure 29 SPN équivalent réorganisé.

Conclusion générale

Durant ce mémoire, nous avons exploré le domaine de la vérification formelle des systèmes reconfigurables et plus spécifiquement les réseaux de Petri reconfigurables. Ces derniers fournissent des mécanismes assez puissants pour la modélisation des systèmes reconfigurables. Néanmoins, la complexité de la vérification augmente considérablement (en anglais *: the modeling power decreases the decision power*).

Les réseaux de Petri reconfigurables (R-SPN) offre un cadre pour la modélisation du changement dans la topologie, et pour la vérification des propriétés des R-SPNs en utilisant un algorithme de dépliage qui transforme un R-SPN donné vers un SPN classique équivalent Ce dépliage engendre des modèles volumineux à cause de la duplication des nœuds dans le modèle cible pour modéliser le changement de la topologie. Cela augmente considérablement la complexité temporelle et spatiale de la phase de vérification. Pour surpasser ce problème, on a opté de déployer notre outil dans un serveur (machine puissant).

Dans ce cadre, notre étude a été effectuée, principalement, en deux étapes :

Une étude théorique concernant les réseaux de Petri stochastiques (SPN), réseaux de Petri stochastiques reconfigurables (R-SPN), ainsi la technique de transformation (dépliage) de R-SPN vers SPN.

La mise en œuvre d'un outil de modélisation de transformation qui donne la solution de la demande de performance du matériel, qui a été le but principal de notre travail.

On a pu atteindre les objectifs tracés dans ce projet, néanmoins il reste quelques travaux à faire pour l'enrichir, à titre d'exemple :

Etendre la configuration à l'ensemble de places et de transitions du modèle.

Minimiser la taille des modèles équivalents obtenus en proposant un autre algorithme de dépilage.

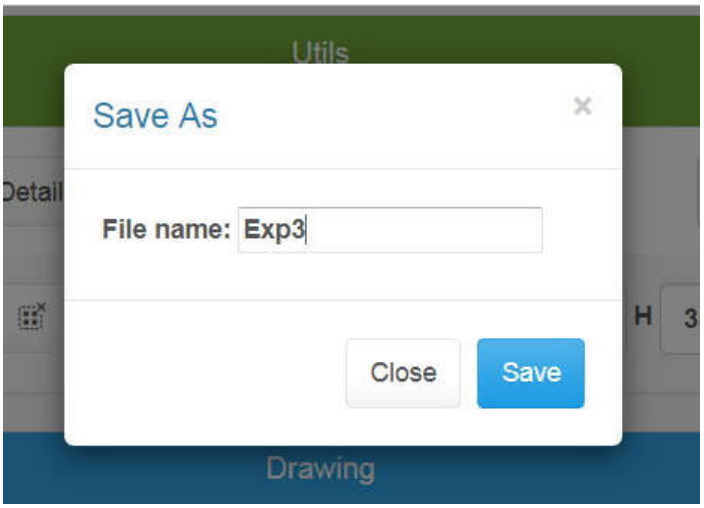
Considérer la reconfiguration dans les réseaux de Petri stochastiques généralisés et éventuellement d'autres extensions de réseaux de Petri.

Bibliographie

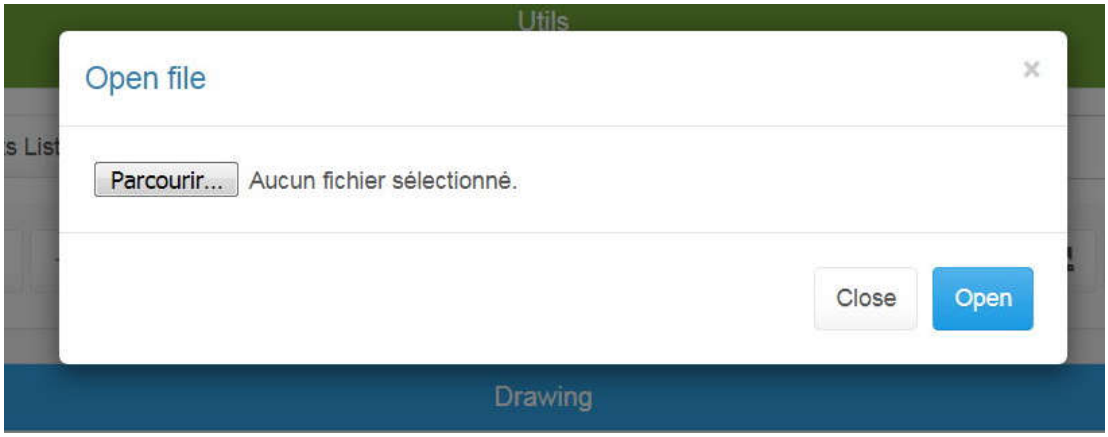
1. TADAO MURATA, FELLOW, IEEE (541) : Petri Nets: Properties, Analysis and Applications. http://www.di.univaq.it/adimarco/teaching/bioinfo15/paper.pdf.
2. Fekair Mohammed El Amine : Le formalisme des réseaux de Petri (chapitre 2). http://tadic.eb2a.com/chapitre%202.pdf?i=1.
3. Jean-François,Pradat-Peyre ,Année 2010-2011 : Introduction à la vérification structurelle des réseaux de Petri et des réseaux de haut-niveau Master SAR, Module VFSR. http://docplayer.fr/20396628-Introduction-a-la-verification-structurelle-des-reseaux-de-petri-et-des-reseaux-de-haut-niveau-master-sar-module-vfsr.html
4. Stéphane MARIEL : VADE - MECUM DE L'ÉTUDIANT EN RÉSEAUX DE PETRI. http://dept-info.labri.fr/~griffaul/Enseignement/FDS/vademecum-petri.pdf.
5. Jean-Louis Boimond : Simulation systèmes de production réseaux de Petri Siman-Arena. http://blog.univ-angers.fr/boimond/files/2012/12/Cours_simulation.pdf.
6. Leonardo Brenner : Modèles à Événements Discrets (Réseaux de Petri Stochastiques). http://www.lsis.org/brennerl/enseignement/SINAU10J/CM_SPN.pdf.
7. Gianfranco Balbo : Introduction to Stochastic Petri Nets. https://pdfs.semanticscholar.org/d2c3/48c60c30ea77dffe77e8fa72a771549bd0d7.pdf.
8. Karim Labadi : Contribution à la modélisation et à l'évaluation de performances des systèmes logistiques à l'aide d'un nouveau modèle de réseaux de Petri stochastiques. https://tel.archives-ouvertes.fr/tel-00389432/document
9. Julia Padberg and Laid Kahloul : Overview of Reconfigurable Petri Nets. https://link.springer.com/content/pdf/10.1007/978-3-319-75396-6_11.pdf
10. Eric Badouel, Javier Oliver : Reconfigurable Nets, a Class of High Level Petri Nets Supporting Dynamic Changes within Workflow Systems [Research Report] RR-3339, INRIA. 1998. https://hal.inria.fr/inria-00073350/document
11. Samir Tigane, Laid Kahloul, Samir Bourekkache : Reconfigurable Stochastic Petri Nets: A New Formalism for Reconfigurable Discrete Event Systems. https://link.springer.com/chapter/10.1007/978-3-319-51100-9_34

12. Ajmone Marsan M., Balbo G., and Conte G., “A Class of Generalized Stochastic Petri Nets for the Performance Analysis of Multiprocessor Systems”, ACM Transaction Computer, Systems, 2(2), pp. 93-122, 1984.
13. Ajmone Marsan M., Balbo G., Conte G., Donatelli, S., and Franceschinis G., « Modelling with Generalized Stochastic Petri Nets » John Wiley and Sons, 1995.
14. M. Llorens and J. Oliver, “Structural and dynamic changes in concurrent systems: reconfigurable petri nets,” IEEE Transactions on Computers, vol. 53, no. 9, pp. 1147–1158, 2004.
15. M. Llorens and J. Oliver, Introducing Structural Dynamic Changes in Petri Nets: Marked-Controlled Reconfigurable Nets, pp. 310–323. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.
16. Marsan, M.A.: Stochastic Petri nets: an elementary introduction. In: Rozenberg,G. (ed.) APN 1988. LNCS, vol. 424, pp. 1–29. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-52494-0 23.
17. Li, J., Dai, X., Meng, Z.: Automatic reconfiguration of Petri net controllers for reconfigurable manufacturing systems with an improved net rewriting systembased approach. IEEE Trans. Autom. Sci. Eng. 6(1), 156–167 (2009).
18. Tigane, S., Kahloul, L., Bourekkache, L.: Net rewriting system for GSPN: A RMS case study. In: 2016 International Conference on Advanced Aspects of Software Engineering (ICAASE), pp. 38–45. IEEE (2016).
19. Tigane, S., Kahloul, L., Bourekkache, S.: Reconfigurable stochastic Petri nets for reconfigurable manufacturing systems. In: Borangiu, T., Trentesaux, D., Thomas, A., Leit˜ao, P., Barata Oliveira, J. (eds.) Service Orientation in Holonic and MultiAgent Manufacturing. SCI, vol. 694, pp. 383–391. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-51100-9 34.
20. Bause, F., Kritzinger, P.S.: Stochastic Petri Nets: An Introduction to the Theory. Vieweg+Teubner Verlag, Cape Town (2002).
21. Serge Haddad: Time and Timed Petri Nets. http://www.lsv.fr/~haddad/disc11-part1.pdf.
22. Kurt Jensen : Coloured Petri Nets. http://www.fit.vutbr.cz/study/courses/PES/public/Pomucky/CPNSlides96.pdf.

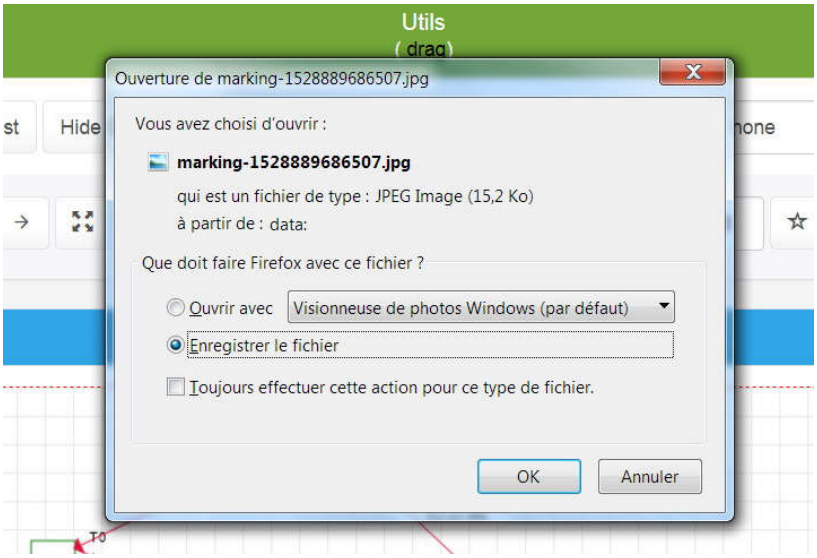
Annexes



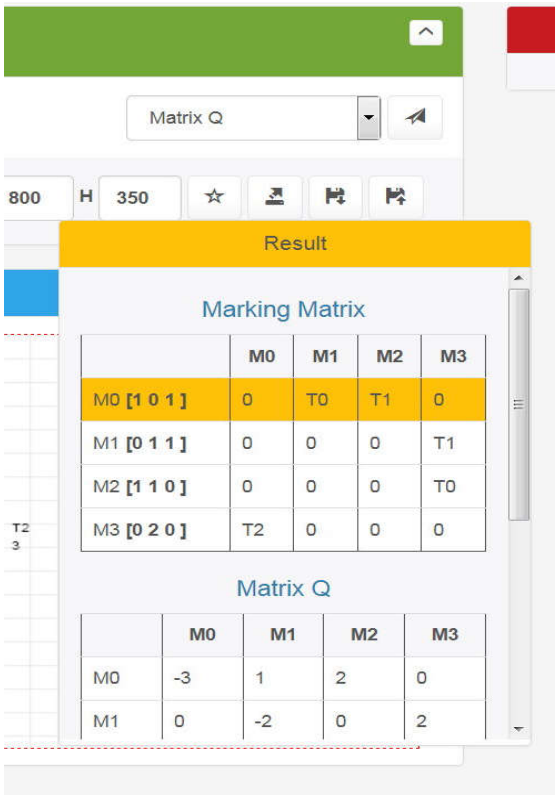
Un dialogue pour sauvegarder le modèle.



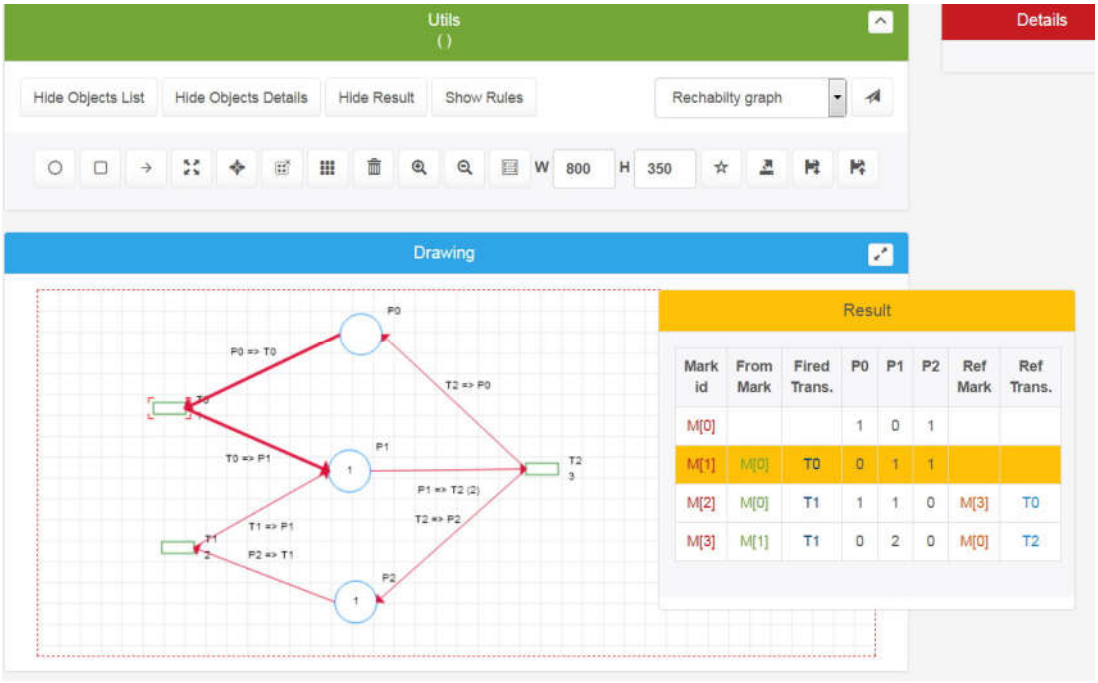
Un dialogue pour ouvrir un modèle sauvegardé.



Dialogue pour exporter le modèle comme image.



Exemple d'extension de l'outil pour faire des calculs de la matrice Q, vecteur π.



Exemple d'extension de l'outil pour faire une simulation d'exécution de graphe des marquages accessibles.