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 : GLSD10/M2/2022

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

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

Transformation des RdPSGs à structure dynamique en RdPSGs à structure statique

---

Par :

BEN NADJI DJIHAD

Soutenu le 26/06/2022 devant le jury composé de :

Ouaar Hanane	MCB	Président
Tigane Samir	MCB	Rapporteur
Djaber Khaled	MAA	Examinateur

Année universitaire 2021-2022

REMERCIEMENT

Je remercie tout particulièrement mon superviseur, Dr Tigane Samir, pour ses encouragements, ses conseils et sa patience infinie.

Je tiens à exprimer mes remerciements les plus sincères aux membres du jury pour l'évaluation de mon mémoire.

Je n'oublie jamais de remercier tous mes professeurs et mes collègues du département d'informatique.

Enfin, ma gratitude est profondément payée à ma mère et mon père, mes sœurs et mon frère, et à tous mes amies et membres de ma famille.

Résumé

Dans ce mémoire, nous implémentons un outil pour un formalisme, appelé les réseaux de Petri stochastiques généralisés dynamiques (RdPSGDs), qui peut modéliser des structures dynamiques (les places et les transitions sont dynamiques), et nous allons transformer les RdPSGDs en RdPSGs (Les réseaux de Petri stochastiques généralisés) équivalents. Cette transformation de graphes peut se produire lorsque le modèle originel a un nombre fini de configurations.

Mots-clés : Réseaux de Petri stochastiques généralisés ; structures dynamiques ; Modélisation et vérification formelles ; transformation de graphes ; structure dynamiques.

Table des matières

Liste des figures

Liste des tableaux

Introduction Générale

Dans cette introduction, nous présentons d'abord le contexte de ce travail, à savoir la modélisation et la vérification formelle basée sur des réseaux de Petri stochastiques dynamiquement structurés. Ensuite, nous nous concentrons sur les motivations de ce travail et précisons la problématique et les objectifs.

Arrière-plan

La modélisation, l'évaluation et l'analyse des performances des systèmes à événements discrets, en particulier les systèmes de stockage, les systèmes de production et surtout les chaînes d'approvisionnement, reste une préoccupation majeure des différentes communautés scientifiques, et en particulier de l'ingénierie de production. Les progrès techniques et économiques, et donc la complexité croissante de ces systèmes, présentent toujours de nouveaux défis et de nouveaux problèmes qui nécessitent l'utilisation de méthodes bien adaptées capables de guider le concepteur dans les choix à faire dans les phases de conception.

Ce problème est reconnu depuis la fin des années 1960, qui a marqué la naissance du génie logiciel. L'un des principaux objectifs de ce dernier est de permettre aux développeurs d'implémenter des systèmes complexes qui fonctionnent bien. Sur ce sujet, différentes approches ont émergé pour répondre à cette exigence critique à différentes étapes du cycle de vie du logiciel.

Ainsi, de nombreuses approches sont apparues, comme les réseaux de Petri et la communication séquentielle de processus (CSP)…etc. Comme des approches dans laquelle la manipulation du langage de spécification était clairement et explicitement définie en mathématiques, et on l'a appelées approches formelles.

Les approches formelles permettent de prouver l'existence de propriétés spécifiques et de vérifier le comportement de l'ensemble du système.il est également possible, grâce aux méthodes formelles, d'écrire des spécifications formelles pour un système sur lequel de nombreuses caractéristiques différentes ont déjà été prouvées.

Cependant, l'utilisation de méthodes formelles ne garantit pas a priori l'exactitude des systèmes développés. En effet, leur utilisation améliore notre compréhension d'un système en construction tout en révélant ses lacunes, ses incohérences et ses ambiguïtés qui pourraient passer inaperçues **(1)** .

Avec la large et remarquable diffusion de nombreux formalismes, on trouve les réseaux de Petri, qui se caractérisent par trois avantages principaux, que nous mentionnons comme suit **(2)**:

1. **Niveau de modélisation :** Ils ont une base mathématique puissante, ainsi que, une représentation graphique intuitive. La représentation graphique donne une vue à plat aux modèles RdP, ce qui permet d'avoir des modèles simples et très explicites. De plus, leur modélisation graphique permet de visualiser facilement des systèmes complexes.
2. **Niveau de vérification :** Leur base mathématique est à l'origine de toute l'analyse technique proposée pour vérifier les systèmes modélisés. En effet, ils disposent d'une panoplie d'analyse qualitative/quantitative bien développée.
3. **Modélisation et vérification du couplage :** Ils offrent un équilibre délicat entre la modélisation et le pouvoir de décision. En fait, les réseaux de Petri ont été utilisés dans la modélisation d'une grande variété de systèmes. Quant à leur pouvoir de décision, le problème d'accessibilité est déterminable dans les filets Petri (notez que la plupart des problèmes peuvent être convertis en problèmes d'accessibilité).

Le concept de réseau de Petri a été développé pour la première fois par Carl Adam Petri, un mathématicien Allemand. L'auteur a défini un outil graphique et

mathématique permettant de décrire les relations existant entre des conditions et des événements. Ils permettent de modéliser le comportement de systèmes à événements discrets et de capturer divers phénomènes qui les caractérisent à savoir le parallélisme, la synchronisation, le partage de ressources, la concurrence, etc. Il est à l'origine du Grafcet (ce dernier étant spécialisé dans la description de la commande de systèmes automatisés) **(3)** .

Nous recommandons à ceux qui ne sont pas familiers avec les réseaux de Petri de consulter le livre de David et Alla **(3)** et l'article de synthèse de Murata **(4)** qui couvrent les bases du formalisme des réseaux de Petri et diverses techniques d'analyse associées **(5)**. Les extensions les encore notables des réseaux de Petri se répartissent en quatre catégories principales :

- **Réseaux de Petri temporisés :** Les réseaux de Petri temporisés introduisent la notion de temps dans le parcours du réseau qui permettent de décrire un système à événements discrets dont le fonctionnement dépend du temps **(6)**. Le temps introduit dans les RdPs permet de mettre des contraintes explicites sur la dynamique du modèle qui reflètent les contraintes temporelles réelles imposées au système **(7)** .

- **Réseaux de Petri colorés :** Les réseaux de Petri colorés sont des réseaux de Petri dans lesquels les jetons portent des couleurs. Une couleur est une information attachée à un jeton. Cette information permet de distinguer des jetons entre eux et peut être de type quelconque **(8)**, **(9)**. Chaque jeton devient plutôt une valeur distincte de l'autre jeton. Les poids sur les arcs ne sont plus des constantes, mais plutôt des fonctions mathématiques qui peuvent être compliquées. Ce modèle permet d'avoir des modèles de taille raisonnable pour des systèmes compliqués **(10)**.

- **Réseaux de Petri stochastiques (RdPS) :** Les réseaux de Petri stochastiques sont une réponse à un autre aspect réaliste manquant dans les modèles précédents, qui est l'aspect du danger, de l'aléa et des événements non négligeables. Les événements aléatoires de leurs arrivées seront explicitement considérés et modélisés, permettant au modèle de se rapprocher du système réel et ainsi d'avoir une bonne représentation du système étudié **(11)**.

- **Réseaux de Petri reconfigurables :** Cette catégorie inclut les formalismes qui permettent de modéliser la flexibilité de la structure, Les concepts de base ainsi que les propriétés principales se retrouvent dans plusieurs œuvres **(12) (13)** .

Cette dernière variante a pour objet de fournir un modèle formel des systèmes structuraux dynamiques, par exemple les systèmes de fabrication reconfigurables (RMS) **(14)** , les systèmes de fabrication flexibles (FMS) **(15)**…, etc.

Longuement systèmes d'événements discrets sont de mieux en mieux complexes, dynamiques sur le plan structural et interconnectés de manière variable. Ces systèmes sont conçus de telle sorte qu'ils permettent de modifier leur structure et/ou leur topologie, en même temps que l'exécution, en ajoutant/supprimant des interconnexions, et des sous-systèmes, pour s'adapter aux nouvelles conditions. Dans ce type de systèmes, les études portent sur sa propriété fondamentale, qui est la reconfigurabilité, qui doit être au moment de l'exécution (c.-à-d., reconfigurabilité dynamique).

L'utilisation des réseaux de Petri (RdPs) dans l'étude de ces systèmes suscite l'intérêt d'un grand nombre de chercheurs **(16), (17), (18)**. Ils offrent plusieurs classes de RdPs pour l'application dans la spécification/vérification de systèmes reconfigurables. À travers cela on peut distinguer trois catégories de travail, la première est basée sur les RdPs de base, la seconde est basée sur l'utilisation de

RdPS reconfigurables, et la dernière applique de RdPs temporelles ou stochastiques.

Afin de modéliser et d'évaluer les systèmes stochastiques, on utilise des réseaux de Petri stochastiques (RdPSs) et des réseaux Petri stochastiques généralisés (RdPSGs), qui sont des extensions des réseaux de Petri. En effet, les formalismes des RDPSs et RdPSGs permettent l'analyse de mesures de performance telle que la consommation d'énergie, la productivité, etc.

E. Simon et d'autres **(19)** se sont concentrés sur la démonstration que RdPSs et RDPSGs sont parmi les outils de conception les plus polyvalents qui s'adaptent bien au comportement DES à toutes les étapes du développement.

Bien que les RdPs (bas / haut) soient un outil puissant et expressif, ils sont incapables de spécifier/vérifier naturellement, les systèmes avancés de structure dynamique **(20)** car ils sont très complexes. Pour surmonter ce problème, les chercheurs introduisent des structures dynamiques dans les RdPs, élargissant ainsi le formalisme standard **(21)**.

D'autre part, les transformations graphiques basées sur des règles **(22)** offrent un cadre graphique mathématique pour modéliser les reconfigurations dans les structures RdP. Néanmoins, augmenter le pouvoir de modélisation d'un formalisme diminue son pouvoir de décision. Par conséquent, les extensions proposées dans la littérature introduisant la reconfigurabilité des RdPs tentent de trouver un compromis entre la modélisation et les niveaux de vérification **(2)** .

Motivation et objectifs

Parce que les systèmes à événements discrets sont très complexes, ils sont représentés par les RdPSGDs en raison de leurs propriétés, afin que nous puissions modéliser ces systèmes de manière élaborée. On sait que pour les RdPSGDs , il existe de nombreux outils à travers lesquels nous pouvons modéliser ce dernier, alors que le manques d'outils de vérifications et d'analyse pour cela

,nous avons trouvé approprié de la transformation d'un RdPSGD en un RdPSG équivalent, en raison de la disponibilité de plusieurs de ses outils de vérification et d'analyse à réutiliser plus tard, et c'est le principal motif de le faire ce travail. L'objectif de ce travail est de développer un outil permettant de transformer les RdPSGs dynamique en ceux statique. L'objectif de ce travail est de développer un outil permettant de transformer les RdPSGs dynamique en ceux statique.

Organisation du mémoire

Ce chapitre a donné un aperçu du sujet de mémoire. Le reste de ce mémoire est organisé comme suit :

Chapitre 1: Présente le contexte des réseaux de Petri ainsi que la chaîne de Markov, modèle checking, les réseaux de Petri stochastique et les réseaux de Petri stochastique généralisé. Nous décrivons les réseaux Petri sous leur forme de base, ainsi que les définitions de base du modèle checking et la chaînes de Markov. Enfin, l'extension des réseaux de Petri aux les réseaux de Petri stochastique généralisé est présentée.

Chapitre 2: Nous introduisons le domaine de transformation de graphes et son utilisation dans le contexte des RdPs. Initialement, les systèmes de transformation de graphes sont décrits. Ensuite, nous nous concentrons sur les applications de transformation de graphes des RdPs dans la littérature.

Chapitre 3: Ce chapitre est le dernier. Il présente les outils et techniques utilisés pour développer ce projet et l'outil réalisé en précisant son interface et ses fonctionnalités.

Le mémoire se termine par une conclusion générale qui évalue le résultat et discute certaines perspectives de ce travail.

Chapitre 1 : Etat de l'art

1 Introduction

Les réseaux de Petri constituent un outil très approprié pour étudier les systèmes à événements discrets en raison de la puissance de modélisation et de leurs propriétés mathématiques **(23)**. Principalement, les RdPs ont été utilisés pour modéliser des systèmes dans lesquels certains événements peuvent se produire en même temps que certaines contraintes sur la concordance, la priorité ou la fréquence de leurs événements **(24)** et en effet , Les réseaux de Petri (RdPs), possèdent un intérêt fondamental indéniable puisqu'ils permettent de modéliser et de maîtriser les comportements des systèmes parallèles et distribués, synchronisés et communicants **(15)**.

Les réseaux de Petri sont analysés à l'aide de nombreuses méthodes, telles que l'utilisation de différentes méthodes mathématiques comme les équations matricielles... ou de différentes manières, par exemple le calcul de tous les états accessibles. Nous soulignons également que les RdPs ont de nombreuses propriétés qui détectent les blocages et les situations irréversibles...etc. Les performances peuvent également être évaluées grâce à des réseaux de RdP temporisé et stochastiques.

Les RdPs présentent de nombreux avantages, notamment qu'ils ont un haut niveau d'expérience dans le domaine de la modélisation en raison de leur utilisation fréquente dans des groupes de domaines d'application en expansion. Nous signalons également qu'ils fournissent un grand nombre mécanismes intégrés qui nous permettent de concevoir correctement des systèmes complexes. Il faut aussi noter que les RdPs ont une base mathématique solide ainsi qu'une représentation graphique sophistiquée qui permet la visualisation de systèmes complexes d'une manière simple et explicite.

En dépit de tout cela, les réseaux Petri ont certains inconvénients, en particulier l'explosion de l'espace d'État (comme les systèmes deviennent de plus

en plus complexes, leurs espaces d'État augmentent de plus en plus, ce qui peut conduire à un problème d'explosion de l'espace d'État), et le délicat couplage de

modélisation et de vérification (les sous-classes de RdP augmentent la puissance de décision, cependant, un grand nombre de systèmes ne peuvent plus être modélisés. D'autre part, les extensions de RdP peuvent augmenter la puissance de modélisation, mais au détriment de la décision de propriété) **(25)** .

1.1 Modélisation avec Les réseaux de Petri

Les systèmes d'événements distincts nécessitent un contrôle et une coordination pour assurer le déroulement ordonné des événements. En tant que systèmes dynamiques contrôlés (ou potentiellement contrôlables), les systèmes d'événements discrets sont qualifiés pour être un sujet approprié pour la théorie du contrôle **(26)**. Au cours de la phase de modélisation de ces systèmes, l'accent est mis sur les éléments de base, qui sont les événements et les conditions, ainsi que sur les relations entre eux, et comme mentionné précédemment, nous pouvons modéliser ces systèmes à l'aide de réseaux de Petri.

Les RdPs sont constitués de deux types de nouds : les places qui modélisent les conditions, et les transitions qui modélisent les événements. Graphiquement les places sont représentées par des cercles et les transitions par des traits, et nous trouverons que chaque place est composée de nombre entier (≥ 0) de jetons qui est conçu sous la forme de points noire représentant les vraies valeurs des conditions, ces jetons peuvent se déplacer d'une place à une autre grâce à ce qu'on appelle " franchissement d'une transition". Nous constatons également que les places et les transitions sont reliées par des arcs. L'état d'un système modélisé par un réseau de Petri qui représenté par un marquage (est un vecteur indiquant la répartition des jetons aux les places de réseau). L'évolution de ce dernier

correspond à l'évolution de l'état du réseau de Petri, et donc du système qu'il représente.

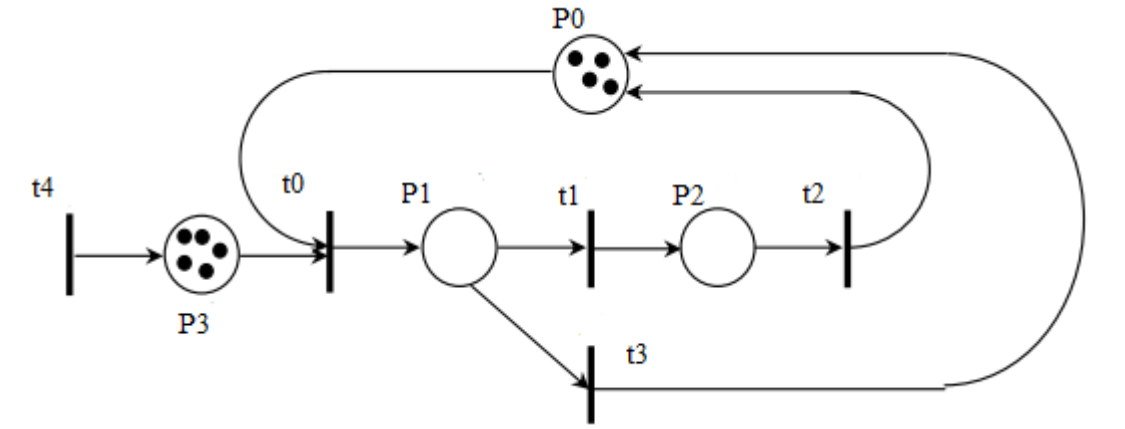

Figure 1. Modèle RdP d'un système de production géré avec la méthode Kanban

Pour illustrer le fonctionnement du réseau de Petri, nous avons pris la figure 1, ci-dessus, qui exprime un modèle de système de production contrôlé par Kanban[1]. La transition t4 représente l'entrée du stock alors les jetons de la place P3 sont représentés le stock d'entrée du système, d'autre part en la place P0, on trouve un certain nombre de jetons représenté les Kanban libres. Lorsqu'il y a au moins un kanban libre dans la place P0 et il y a au moins un produit dans le stock d'entrée en la places P3, il y aura un franchissement pour la transition t0. Et puisque la transition t1 est franchie, un jeton sera lancé des deux places P0 et P3, pour ajouter un jeton à la place P1. Ce qui signifie que la Kanban libre et le produit dans le stock d'entrée sont liés, et la fabrication de ce dernier va commencer. La qualité du produit est vérifiée, et si elle n'est pas satisfaisante, le franchissement de t3 a lieu, elle est donc rejetée, et si la qualité est bonne, il arrive qu'un jeton apparaisse à la place P2, après le franchissement de la transition t1, qui représente la fabrication du produit. Le produit quitte le

[1] L'approche Kanban : est une méthode de gestion du stock qui permet de produire à la demande des clients. L'objectif premier de cette méthode est d'arriver à un équilibre entre production et demande. Le Kanban vient de l'approche Lean, qui se traduit par l'amélioration continue des processus de production afin d'arriver à une production sans gaspillage.

système et le kanban attaché est retourné en P1 et ainsi, le franchissement de la transition t2.

1.2 L'aspect structurel

1.2.1 Définition d'un réseau de Petri (4)

Un réseau de Petri est un quadruple $N = \{P, T, F, M_o\}$ où :

- P est un ensemble fini et non vide de places.
- T est un ensemble fini et non vide de transitions disjointes de P.
- $F : (P \times T) \cup (T \times P) \rightarrow N$ est une relation de flux pour un ensemble d'arcs.
- $M_o : P \rightarrow N$ est un marquage initial.

Certains auteurs se réfèrent au réseau non marqué $N = \{P, T, F\}$ comme structure du RDP et se réfèrent au réseau marqué $N' = \{P, T, F, M_o\}$ comme un système.

Exemple:

Considérez le réseau de Petri H simple dans la figure 2. Sa définition formelle est donnée par $H = \{P, T, F, M_o\}$ où :

- $P = \{p1, p2, p3, p4\}$.
- $T = \{t1, t2, t3\}$.
- $F = (p1, t1) = 2$; $F = (t1, p2) = 2$; $F = (t1, p3) = 1$ … *etc.*
- $M_o = (p1, p2, p3, p4) = (2, 0, 0, 0)$.

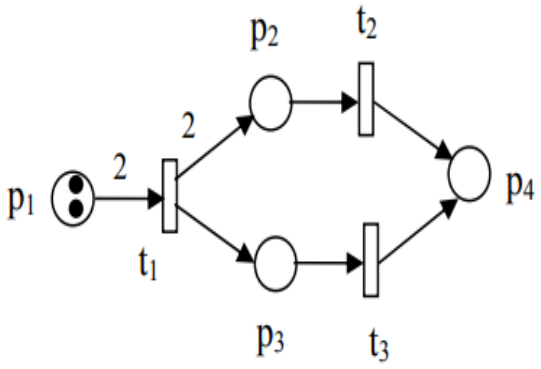

Figure 2.Modèle de réseau de Petri simple

1.2.2 Définition : Preset & Postset

Les entrées et les sorties (inputs et outputs en anglais) ou ce qu'on appelle aussi preset et postset respectivement pour les places et les transitions. Pour illustrer, nous prenons le preset et le postset des transitions. Le preset d'une transition t ∈ T correspond au multi-ensemble de places pointant vers t mais le postset d'une transition, correspond au multi-ensemble de places pointées par t. Pour les places et les transitions peuvent être définis formellement comme suit :

- Le preset de la place *p*, indiqué par •*p*, est donné par $\bullet p = \{t \in T \mid F(t, p) > 0\}$.
- Le postset de la place *p*, indiqué par *p*•, est donné par $p\bullet = \{t \in T \mid F(p, t) > 0\}$.
- Le preset de la transition *t*, indiqué par •*t*, est donné par $\bullet t = \{p \in P \mid F(p, t) > 0\}$.
- Le postset de la transition *t*, indiqué par *t*•, est donné par $t\bullet = \{p \in P \mid F(t, p) > 0\}$.

Exemple:

En considérant le RdP H montrée à la Fig. 2, nous avons, par exemple :

1. $\bullet p3 = \{t1\}$ et $p3\bullet = \{t4\}$.
2. $\bullet t1 = \{p1\}$ et $t1\bullet = \{p2, p3\}$.

1.3 Comportement dynamique des RdPs

1.3.1 Franchissement d'une transition

Le franchissement d'une transition ou ce qui appelé dans certaines littératures par "le tir d'une transition", consiste à enlever un jeton de tous les places d'entrée pour cette transition et l'ajouter à ses places des sortie. Nous introduisons maintenant "La règle d'activation" et "La règle de franchissement", qui régissent le flux de jetons :

1) ***Règle d'activation*** : Une transition t est dite activée si chaque lieu d'entrée *p* de *t* contient au moins le nombre de jetons égal au poids de l'arc dirigé reliant *p* à *t*, et en d'autres termes, la transition t est activée au marquage *M*, indiqué par $M\,[t\rangle$. Si *:*

$$M(p) \geq F\,(p, t),\ \forall p \in {\bullet}t.$$

2) ***règle de franchissement***: Seule la transition activée peut franchir. Le franchissement d'une transition *t* activée enlève de chaque place d'entrée *p* le nombre de jetons égal au poids de l'arc dirigé reliant *p* à *t*. Il dépose également dans chaque place de sortie *p* le nombre de jetons égal au poids de l'arc dirigé reliant *t* à *p*. Le franchissement d'une transition *t* activé au marquage *M* donne un nouveau marquage *M',* désigné par $M\,[t\rangle\,M'$, de telle sorte que *:*

$$M'\,(p) = M(p) + F(t, p) - F\,(p, t),\ \forall p \in P$$

Lorsqu'une transition est franchie, le nombre de jetons à chaque place reste toujours non négatif, de sorte que la transition ne peut jamais essayer de supprimer le jeton qui n'existe pas.

Une transition sans place d'entrée est appelée une transition source, mais la transition qui sans place de sortie est appelée une transition puits. Notez qu'une

transition source est inconditionnellement activé, et que le franchissement d'une transition puits consomme des jetons, mais ne produit pas de jetons.

Exemple: Considérez le RdP montré dans la figure 2. Sous le marquage initial $M_o = (2, 0, 0, 0)$.

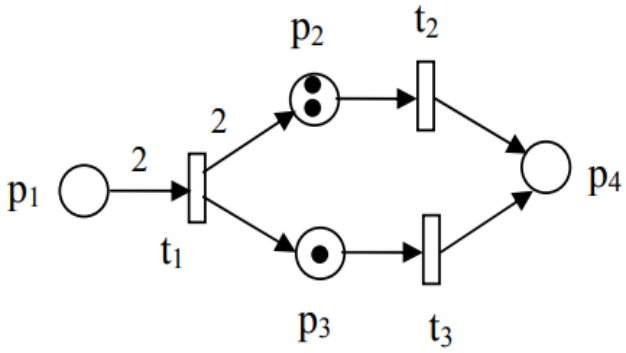

Figure 3.Franchissement de la transition t1

La transition *t1* est le seul est activé. Le franchissement de t1 entraîne un nouveau marquage M_1 .Il découle de "Firing Rule" que $M_1 = (0\ 2\ 1\ 0)$.

La nouvelle distribution de jeton de ce réseau de Petri est montrée dans la figure 3. Encore une fois, en marquant M_1, les deux transitions de t2 et t3 sont activés. Si *t3* est franchie, le nouveau marquage M_2, est : $M_2 = (0\ 2\ 0\ 1)$. Si *t2* est franchie, le nouveau marquage M_3 est : $M_3 = (0\ 1\ 1\ 1)$.

1.3.2 Séquence de franchissement

Une séquence de franchissements est franchissement successif de séquence des transitions $\sigma = t_1, t_2, \ldots, t_n$ dans un ordre donné à partir d'un marquage M_1 si il existe une séquence de marquages $M_1, M_2, \ldots, M_n$ tel que $M_i\ [t_i\rangle,\ \forall i \in \{1, 2, \ldots, n\}$.

$M_1\ [\sigma\rangle M_{n+1}$ désigne le franchissement de séquence de transition σ à partir de M_1 et M_{n+1} est accessible à partir de M_1 par le franchissement σ.

1.3.3 Ensemble d'accessibilité

L'ensemble d'accessibilité d'un RdP ayant le marquage initial M_o, désigné par $RS(M_o)$, il est défini comme suit : $RS(M_o) = \{M | M_o [\sigma\rangle M_n \wedge \sigma$ est une séquence de franchissements.

Notez que (i) σ peut être vide, c'est-à-dire que $M_o \in RS(M_o)$, (ii) $RS(M_o)$ peut être infini, et (iii) un marquage initial doit être complètement fourni pour calculer l'ensemble d'accessibilité, c'est-à-dire que ce calcul n'est pas possible ni pour la structure de RDP ni pour la famille RdP **(23)**.

1.3.4 Graphe d'accessibilité

Le *Graphe d'accessibilité* d'un RdP $N = \{P, T, F, M_o\}$, désigné par $G(M_o) = \{V, E\}$, est un graphe dirigé étiqueté, où :

- $V = RS(M_o)$.
- $E = \{(M_i, t, M_j) | M_i \in RS(M_o).) \wedge M_j \in RS(M_o) \wedge t \in T\}$.

Exemple: Considérez le RdP *H* montré dans la figure 4.

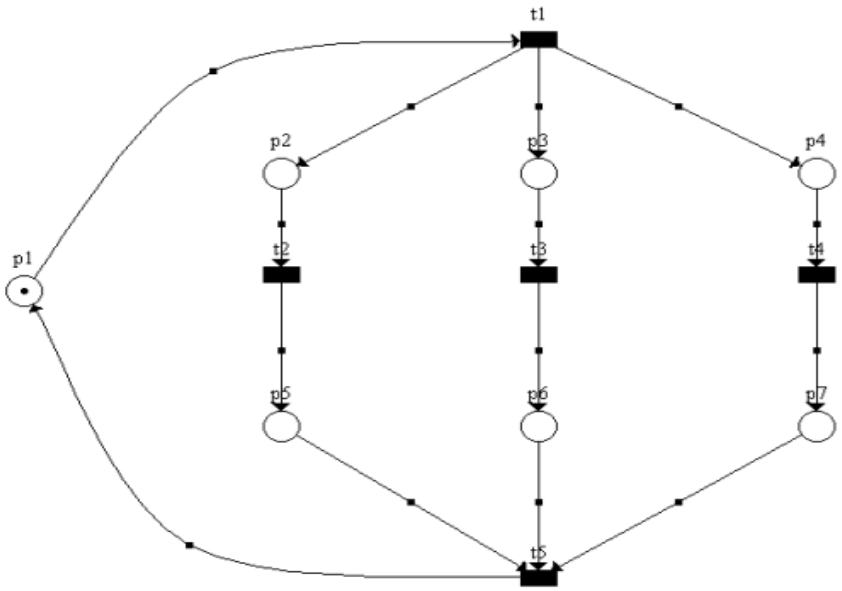

Figure 4.Un modèle simple d'un RDP

Son graphe d'accessibilité est illustré à la figure 5, $G(M_o)= \{V, E\}$ où :

1. *V = {s0, s1, s2, s3, s4, s5, s6, s7,s8},*
2. *E = {(s0, t1, s1), (s1, t2, s2), (s1, t3, s3), (s1, t4, s4)…}.*

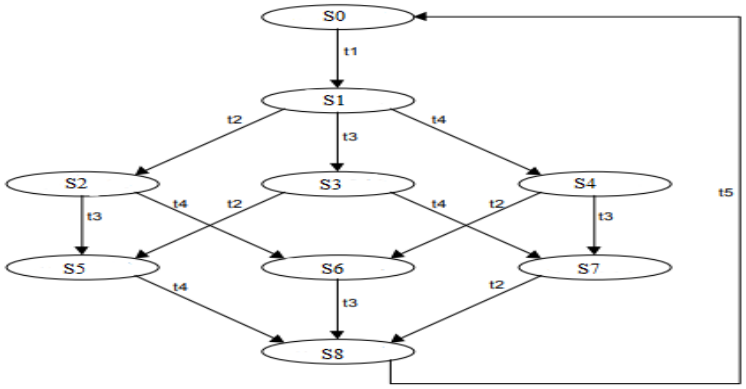

Figure 5. Graphe d'accessibilité de H

	P1	*P2*	*P3*	*P4*	*P5*	*P6*	*P7*
S0	1	0	0	0	0	0	0
S1	0	1	1	1	0	0	0
S2	0	0	1	1	1	0	0
S3	0	1	0	1	0	1	0
S4	0	1	1	0	0	0	1
S5	0	0	0	1	1	1	0
S6	0	0	1	0	1	0	1
S7	0	1	0	0	0	1	1
S8	*0*	*0*	*0*	*0*	*1*	*1*	*1*

Table 1.Les marquages accessibles de H

1.4 Analyse des réseaux de Petri

Dans les sections précédentes, nous avons présenté la puissance de modélisations des réseaux de Petri, mais la modélisation elle-même est de peu d'utilité. Il est nécessaire d'analyser le système modélisé. Nous espérons que cette

analyse se traduira par une meilleure compréhension du comportement du système modélisé.

En fait, les réseaux de Petri prennent en charge l'analyse d'une gamme de caractéristiques et de problèmes associés aux systèmes concurrents. L'analyse de ces caractéristiques repose sur l'utilisation de nombreuses méthodes, soit par des caractéristiques comportementales qui appelé le graphe accessible, ou par les caractéristiques structurelles. Dans cette section, nous discutons des propriétés importantes ainsi que des problèmes de leur analyse.

1.4.1 Les propriétés des réseaux de Petri

❖ L'accessibilité:

Le problème d'accessibilité pour les RdPs consiste à décider si un marquage M est accessible à partir du marquage initial .Le problème d'accessibilité est décidable.

❖ Bornitude

Un réseau de Petri est borné si son ensemble de marquages sensibilisés est fini. Karp et Miller ont prouvé en **(27)** que le bornitude est décidable. Ce résultat résulte de la caractérisation suivante des RdPs non bornés qui n'est pas difficile à démontrer. Un réseau Petri est non borné s'il existe un marquage M accessible et une séquence de transition σ de telle sorte que $M\ [\sigma\rangle\ M + L$, où L est un marquage non nulle **(28)**.

Un réseau Petri est k-borné si, pour tout marquage accessible, il y a tout au plus des jetons k en tout place. Un réseau Petri est sûr (safe) s'il est 1-borné.

❖ Vivacité

Hack a montré en **(29)** que le problème de vivacité est récursivement équivalent au problème d'accessibilité et donc décidable. Un réseau de Petri est

dit être vivant si chaque transition peut toujours tirer à nouveau dans le futur à tout marquage accessible. Ce genre de vivacité est considéré comme une caractéristique forte du RDP. Ainsi, il est détendu à différents niveaux de vivacité **(4) (24)** . Une transition dans un RdP ayant le marquage initial M_o est :

- **N0-vivant (ou mort) :** si elle ne peut jamais être activée lors d'un marquage accessible.
- **N1-vivant :** s'il peut être activé au moins une fois dans une certaine séquence de franchissement à partir de M_o.
- **N2-vivant :** s'il peut faire le franchissement, au moins, k fois dans certaines séquences de franchissement à partir de M_o, où k est un nombre entier positif.
- **N3-vivant** : s'il apparaît infiniment souvent dans certaines séquences de franchissement à partir de M_o.
- **N4-vivant (ou vivant) :** s'il est N1-vivant pour tout marquage accessible à partir de M_o.
- **N*k*-vivant :** si toutes les transitions sont N*k*-vivant, de sorte que $k \in 0, 1, 2, 3, 4$.

Exemple : Dans la figure 6, nous présentons différents niveaux de vivacité pour les réseaux de Petri, dans la figure 6.a, nous constatons que la transition t1 est N0-vivant (mort), d'autre part les transitions t0 ; t3 ; t2; t4 de la figure 6.b sont N1; N2 ; N3 et N4-vivant respectivement.

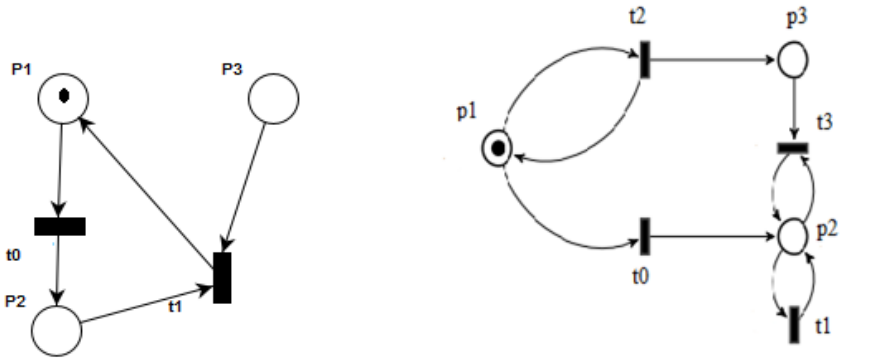

Figure 6.a. Figure 6.b.

Figure 6. Différents niveaux de vivacité

❖ Blocage

Un blocage correspond à un marquage du réseau Petri pour lequel il n'y a pas de transition sensibilisée.

Un RdP est considérée comme exempte de blocage si aucun marquage de tous les marquages accessibles ne constitue un blocage.

Exemple : Dans la figure 7, nous avons suggéré deux modèles de réseaux de Petri M1 et M2, où l'on note qu'Ilya un blocage dans M2 contrairement au modèle M1.

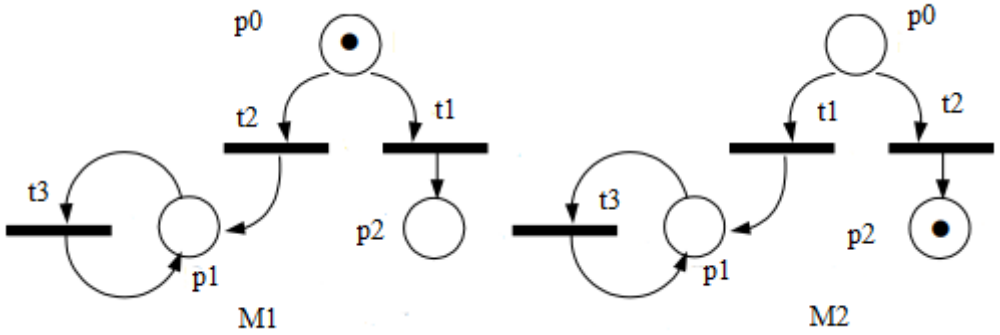

Figure 7.Exemple Illustratif pour le blocage dans les RdPs

❖ **État d'accueil et réseaux réinitialisables**

Un réseau de Petri possède un état d'accueil M_a pour un marquage initial M_o. Si pour tout marquage accessible $M \epsilon R(M)$, il existe une séquence de franchissement σ tel que : $M [\sigma\rangle M_a$ **(30)** .

Si le marquage initial M_o est un état d'accueil, alors le réseau de Petri est réinitialisable **(31)**.

1.4.2 Méthodes d'analyse des RdPs

La modélisation du système n'est utile que si ses propriétés sont analysées. Nous retrouvons dans de nombreuses littératures que la théorie des réseaux de Petri offre de nombreuses techniques analytiques, nous citons ce qui suit:

❖ **Analyse par graphes des marquages**

Afin d'analyser les propriétés du réseau de Petri, il faut d'abord construire son graphe de marquages accessibles, dans ce graphe chaque marquage accessible représenté par un sommet et chaque franchissement d'une transition représenté par un arc qui permettent de passer d'un marquage à un autre, et il y a deux situations possibles pour ce graphe, fini ou infini.

❖ **Analyse par algèbre linéaire :**

Par cette méthode, nous pouvons étudier les propriétés structurales du réseau de Petri (Bornitude, vivacité) indépendamment des marquages initiaux. Pour de plus amples renseignements sur ces techniques d'analyse, le lecteur est prié de consulter **(4), (32)**.

1.5 Les réseaux de Petri stochastiques

Le principal inconvénient de RdPs est que les analyses quantitatives ne sont pas prises en compte, le développeur qui cherche à connaître ces caractéristiques

dans son système doit créer un modèle différent du système , qui ne peut fournir aucune garantie de cohérence entre différents modèles, et donc les informaticiens au cours de la dernière décennie, ajoutent du temps aux réseaux da Petri pour créer également des réseaux de Petri stochastiques (RdPSs) et des réseaux de Petri stochastiques généralisés (RdPSGs) pour modéliser des phénomènes où le temps joue un rôle majeur (systèmes à temps réel industriel à titre comme un exemple) et l'obtenir une modélisation des performances optimales.

Nous considérons donc les RdPSs comme une élaboration des RdPs originaux cela est dû à l'avantage de la forme graphique de la conception du système et de ses spécifications, ainsi que de la manière naturelle dont le temps qui peut être ajoutée pour déterminer les caractéristiques quantitatives du système à étudier.

Des RdPs augmentés dans le temps sont introduits. En général, il y a deux façons possibles de le faire **(33)** :

i. *Réseaux de Pétri P- temporisés :* les jetons tirés sur une place sont indisponibles à toutes ses transitions de sortie pendant un certain temps. Une fois ce temps écoulé, les jetons deviennent disponibles.
ii. *Réseaux de Pétri T- temporisés :* Lorsque la transition est activée, le franchissement ne se fait pas immédiatement, mais plutôt après un certain temps.

Les RdPs augmentés dans le temps sont classés en fonction de la nature temporelle. Si le temps est déterministe, ils sont appelés "les réseaux de Petri temporisé", et si le temps de franchissement est variable et aléatoire, ils sont appelés "les RdPs stochastique", qui font l'objet une certaine loi de distribution qui contrôle leur classification. Les RdPSs dépendent également d'autres caractéristiques de franchissement, à savoir :

I. **La politique de mémoire :** Lorsqu'une transition temporisée est activée, les valeurs de temps commencent à diminuer jusqu'à ce qu'ils sont nuls, ce qui conduit au franchissement. Il y a trois politiques de mémoire qui portent

sur la façon dont le congédiement d'une transition devrait influencer les périodes des autres transitions :

a) ***Resampling (interruption)*** : Suppose que chaque transition temporisée est associée à une minuterie pour définir ses temps correspondants pour chaque transition temporisée. pour chaque franchissement, les minuteries de toutes les transitions temporisées sont éliminées.

b) ***Enabling memory (poursuite conditionnelle) :*** Pour chaque franchissement, les minuteries de toutes les transitions temporisées désactivées sont redémarrées, et le temps passé est conservé si la transition reste franchissable (mécanisme continue).

c) ***Age memory (poursuite inconditionnelle)*** : Pour chaque tir, les minuteurs de toutes les transitions temporisées conservent leurs valeurs actuelles.

II. **Politique de service :** Compte tenu d'une transition permise à un certain marquage auquel il peut tirer n fois, pour le nombre de fois où la transition doit être sensibilisée lorsque son temps de franchir est écoulé on a trois politiques :

a) ***Single server (Serveur-unique) :*** Dans cette politique, un seul tir est effectué. ce qui signifie que la transition ne peut offrir qu'un seul service à la fois.

b) ***Infinité-servers (Serveurs-infinis) :*** Peut garantir déplacement de n'importe quel nombre de services simultanés. il peut faire le franchissement k fois.

c) ***Multiple-server :*** Les tirs Min (k, deg(t)) sont effectués. Cela signifie que la transition t peut assurer deg(t) des services simultanés au maximum.

III. **politique de choix :** il y a deux politiques concernant la façon dont les jetons sont réservés jusqu'au tir de transition :

a) ***Politique de présélection*** : La transition activée conserve tous les jetons dont elle a besoin pour que ces jetons ne soient pas disponibles pour les autres, il attend donc la fin du temps de franchissement spécifié pour qu'ils tirent immédiatement.

b) ***la politique de course (race policy):*** une transition activée attend que son intervalle de temps de franchissement soit écoulé, il tire immédiatement selon qu'il soit encore activé à ce moment-là, alors tous les jetons requis ne sont pas déjà consommés par une autre transition.

1.5.1 Processus stochastique

Un processus stochastique(ou processus aléatoire) est un modèle mathématique utile pour la description de phénomènes de nature probabiliste en fonction d'un paramètre qui a habituellement la signification du temps. Les processus stochastiques sont définis en énumérant les états accessibles possibles par les systèmes modélisés et les probabilités au moment des transitions entre ces états.

Mathématiquement, un processus stochastique est une famille de variables aléatoires *X(t)* définie sur les mêmes valeurs de prise d'espace de probabilité dans un ensemble S, où le paramètre t indique le temps et utilisé pour indexer chaque variable aléatoire et *S* est l'espace d'état du processus. Les processus stochastiques peuvent être classés selon l'espace étatique ou la nature du temps. Si l'espace d'état est discret alors le processus est appelé processus d'état discret ou chaîne, alors que si l'espace d'état est continu, alors il est appelé processus d'espace continu. Analogiquement, si le temps est continu, alors le processus est appelé processus en temps continu, sinon, il est appelé processus en temps discret **(2)** .

1.5.2 Processus de Markov

Un processus de Markov est un processus stochastique dans lequel l'avenir est indépendant du passé, son évolution ne dépend que du présent. Ceci est connu sous le nom de propriété Markov. Le processus Markov a les propriétés suivantes :

(a) Le nombre de résultats ou d'états possibles est limité.

(b) Le résultat à n'importe quelle étape dépend uniquement du résultat de l'étape précédente.

(c) Les probabilités sont constantes dans le temps.

Ainsi, les processus de Markov sont les analogues stochastiques naturels des processus déterministes décrits par les équations différentielles et différentielles. Ils forment l'une des classes les plus importantes de processus aléatoires.

On a $P[X\ (t) = x]$ indique la probabilité que le processus stochastique aura valeur x au temps t et $P[X\ (t) = x|\ X\ (t_n) = x_n]$ indique la probabilité que le processus stochastique aura valeur x au temps t telle qu'il aura valeur x_n au temps t_n. Formellement, un processus de Markov est un processus stochastique $\{X\ (t)\}$ dont la fonction de densité de probabilité conditionnelle est telle que :

$$P[X\ (t) = x|\ X\ (t_n) = x_n, X\ (t_n - 1) = x_n - 1, \ldots, X\ (t_0) = x_0] = P[X\ (t) = x|\ X\ (tn) = x_n]. \quad (1.1)$$

Où $t > t_n > t_{n-1} > \ldots > t_0$.

Un processus markovien s'appelle chaîne de Markov de temps discrète(CMTD), lorsque l'intervalle T d'observation est un ensemble discret, et il s'appelle chaîne de Markov de temps continue (CMTC), lorsque l'intervalle d'observation T est un ensemble continu. Considérons un ensemble discret et fini d'états d'un système (ensemble d'observations successives) **(5)**.

Si l'évolution future d'un processus de Markov est indépendante de t_n instantané particulier en Eq. 1.1, qui déterminé en connaissance de l'état actuel, puis le processus de Markov est dit être temps homogène, pour ce dernier la condition suivante s'applique :

$$P[X(t+s) = x | X(t_n + s) = x_n] = P[X(t) = x | X(t_n) = x_n]. \quad (1.2)$$

Dans les CMTC, les probabilités de transitions entre les états sont données en fonction du temps. Le temps pendant lequel le processus reste dans un état avant de passer à un autre est appelé temps de séjour. D'autre part la distribution exponentielle est la seule distribution continue qui satisfait la propriété du processus de Markov.

1.5.3 RdPS ayant une loi exponentielle

Un réseau de Petri stochastique ayant la loi exponentielle $\mathcal{N}$ est un réseau de Petri augmenté d'une fonction Λ qui assigne à chaque transition t_i en $\mathcal{N}$ un retard de tir λ_i qui est réparti exponentiellement. La distribution de la variable aléatoire X_i du délai de tir de transition t_i est donnée par

$$F_{Xi(x)} = 1 - e^{-\lambda_i x} \quad (1.3)$$

1.5.4 Définition un réseau de Petri stochastique (RdPS)

Un RdPS se définissent formellement comme étant un quintuplet telle que

$\mathcal{N} = (P, T, F, M_0, \Lambda_i)$, où :

- P est un ensemble fini et non vide de places.
- T is a finite and non-empty set of transitions disjoint from P.
- $F : (P \times T) \cup (T \times P) \longrightarrow N$ est une relation de flux pour un ensemble d'arcs.
- $M_o : P \longrightarrow N$ est un marquage initial.

- $\Lambda : T \longrightarrow R^+$ est une fonction qui associe à chaque transition $t_i \in T$ un temps de tir réparti de façon exponentielle λ_i.

Exemple : Considère le RdPS $\mathcal{N}$ dans la Fig. 8. La transition *t1* est activée au marquage initial M_0= [P0, P1, P2, P3, P4] = [1, 0, 0, 0, 0]. Avant de tirer, la transition *t1* attend qu'un certain temps se soit écoulé. Ce temps est réparti de façon exponentielle avec le taux λ1,

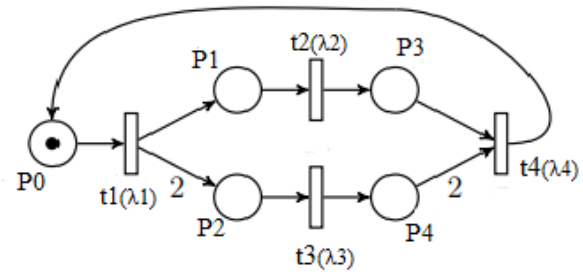

Figure 8.Modèle de RdPS

Les règles d'habilitation et de franchissement des RdPs de base sont préservées dans les RdPSs, par ce que la nature des RdPSs ne modifie pas le comportement de base du modèle non temporel sous-jacent. En tenant compte des aspects probabilistes des RdPSs, le marquage suivant dépend de quelle transition se franchir en premier. Étant donné *n* transitions activées *t1, t2, . . . , tn* ayant des taux λ1, λ2, . . ., λn au marquage *M*, la probabilité que t_i tire en premier est donnée par :

$$\boldsymbol{P[t_i\ tire\ d'abord\ \text{à}\ M]} = \frac{\boldsymbol{\lambda i}}{\boldsymbol{\lambda 1 + \lambda 2 + \cdots + \lambda n}} \qquad (1.4)$$

L'analyse quantitative des RdPSs peut être effectuée en analysant le CMTC correspondant, où les RdPSs décrivent un CMTC qui est isomorphe à son graphe d'accessibilité.

Nous pouvons obtenir l'analyse quantitative des RdPSs directement via le graphe d'accessibilité d'un RdPS de la même manière utilisée avec les RdPs de base où le taux de transition entre deux marquages est le taux de la transition de franchissement correspondante dans le RdPS.

Exemple : Considère l'exemple du CTMC illustré à la figure 9.

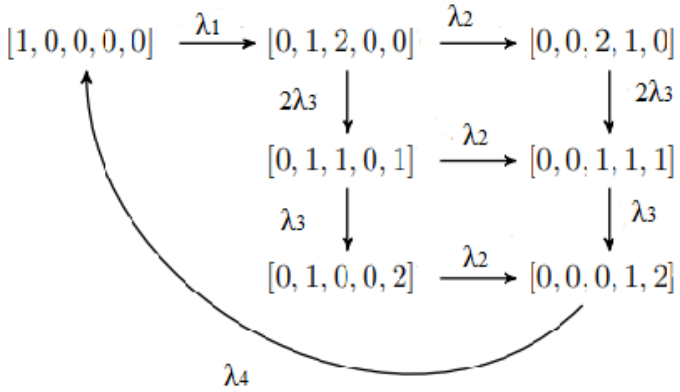

Figure 9.CTMC correspondant du RDPSs N représenté à la Fig.8

On peut obtenu de CMTC en calculant le graphe d'accessibilité de RdPS N représenté à la Fig.8 .et aussi en appliquant ce qui a été mentionné dans le paragraphe précédent.

Il existe également une matrice dite une matrice de taux de transition, qu'une étape d'analyse quantitative des RdPSs, qui est un tableau de nombres décrivant la vitesse instantanée à laquelle une chaîne de Markov de temps continu passe entre les états. Dans cette matrice $Q = (q_{ij})$, un élément q_{ij} $(i \neq j)$ indique le taux de départ de i et d'arrivée dans l'état j. L'élément q_{ii} est calculé de telle sorte que, $\sum_{n}^{j=1} \boldsymbol{qij} = \mathbf{0}$, c'est-à-dire:

$$\text{qii} = -\sum_{n}^{j=1} \text{qij}, \text{i} = \text{j} \qquad (1.5)$$

Nous demandons à ceux qui ne connaissent pas les matrices de taux de transition à consulter l'article de Mark .A et des autres **(34)** Donner les bases des matrices de taux de transition. Quant à deuxième étape qui suit la matrices de taux de transition c'est calculer la distribution à l'état stable $\Pi = [\pi_1, \pi_2, \ldots, \pi_n]$, Pour ce faire, on résout le système d'équation suivant :

$$\Pi \times \text{Q} = 0, \text{telle que } \sum_{\boldsymbol{n}}^{\boldsymbol{i=1}} (\boldsymbol{\pi i}) = \mathbf{1} \qquad (1.6)$$

1.5.5 Quelques propriétés qualitatives des RdPSs

Les RdPSs héritent de la plupart des propriétés qualitatives d'intérêt des réseaux Petri de base, ce qui permet d'utiliser un riche ensemble de méthodes pour leur analyse qualitative **(4)**. Dans cette section, nous rappelons quelques-unes des propriétés qualitatives des RdPSs.

1.5.5.1 La vivacité

On dit qu'un RdPS est vivant si pour chaque marquage accessible $M \in RS$ et chaque transition $t_i \in T$, il existe une séquence de transition qui, lorsqu'elle est franchie du marquage M, entraîne un marquage dans lequel la transition t_i est

activée. Tandis que la probabilité de transition de franchir t_i calculée par l'équation suivante :

$$r = \sum_{s_i \in EN_t} \pi_i \left(\frac{\lambda_t}{-q_{ii}} \right) = \sum_{s_i \in EN_t} \pi_i P[t \text{ fires first at } s_i]. \qquad (1.7)$$

Remarque : EN_t désigne le sous-ensemble de *RS(N)* dans lequel une transition donnée *t* est possible et la probabilité *r* qu'un observateur qui regarde au hasard dans le RdPS.

1.5.5.2 S-invariant

L'ensemble de places pour lesquels les éléments correspondants d'un S-invariant ***U*** ne sont pas nuls est appelé le support de S-invariant ***U.*** Si un invariant ***U :***

1. Il a un support minimal (en ce sens qu'il n'y a pas d'un S-invariant dont le support est un sous-ensemble approprié du support de ***U***).

2. ***U*** est un vecteur minimal parmi les S-invariants (en ce sens qu'il n'y a pas un S-invariant $\boldsymbol{U}' \leq \boldsymbol{U}$ et un index k pour lequel $U'_k < U_k$) il est appelé un S-invariant de support minimal.

1.5.5.3 Probabilité d'être dans un sous-ensemble de marquages

Pour **"** Probabilité d'être dans un sous-ensemble de marquages**",** il y a $B \subseteq$ *RS(RdPS),* la probabilité d'être dans un état du sous-ensemble correspondant est donnée par

$$\mathbf{P}[\boldsymbol{B}] = \sum_{\mathbf{si} \in \mathbf{B}} \boldsymbol{\pi}\mathbf{i}. \qquad (1.8)$$

1.6 Réseaux de Petri stochastique généralisé

Comme précédemment, nous avons mentionné que les RdPSs sont un formalisme décrivant les systèmes dynamiques d'événements discrets, qui peuvent être représentés par le comportement dynamique des chaînes de Markov homogènes en temps continu. Afin de rendre la puissance de modélisation des RdPSs plus puissante et d'élargir sa portée, des réseaux de Petri stochastiques généralisés a été proposé. Dans **(35)** , Les RdPSGs comprennent deux classes de transitions : les transitions temporisées (il représenté par une barre non remplies) distribuées de façon exponentielle, qui sont utilisées pour modéliser les retards aléatoires associés à l'exécution d'activités, et les transitions immédiates (il représenté par une barre remplies), qui sont consacrées à la représentation d'actions logiques qui ne consomment pas de temps. Lorsque des transitions temporisées et immédiates sont activées dans le même marquage, les transitions immédiates franchirent toujours en premier. La sélection parmi les transitions immédiates possiblement conflictuelles activées se fait par des probabilités de franchissements formant les commutateurs dits aléatoires.

Les RdPSGs ont été appliqués avec succès à l'analyse de la performance de divers systèmes dont les principales caractéristiques incluent la concurrence et la synchronisation. Parmi les domaines qui peuvent être appliqués là où les RdPSGs succès, nous trouvons des systèmes des fabrications flexibles **(36)**, ainsi que des systèmes distribués **(37)**. Néanmoins, l'acceptation de RdPSG comme outil de modélisation n'a pas été aussi répandue que la puissance descriptive et d'analyse de l'outil le mérite. Cela était dû à deux raisons : la difficulté dans la construction des modèles et la complexité de calcul dans la solution de modèle.

Considérez RdPSGs G avec le marquage M représenté à la Fig.10. Les transitions enter1 et enter2 sont des transitions immédiates, et les autres transitions sont temporisées. Au marquage M, les transitions t2 et t3 sont activées (ce qui désactive toutes les transitions temporisées).Le franchissement de t2 où t3 consomme un jeton à

partir de la place p1 ; et produit un jeton dans les places p3 et p4. Après le franchissement de t2 et t3, l'autre transition devient activée.

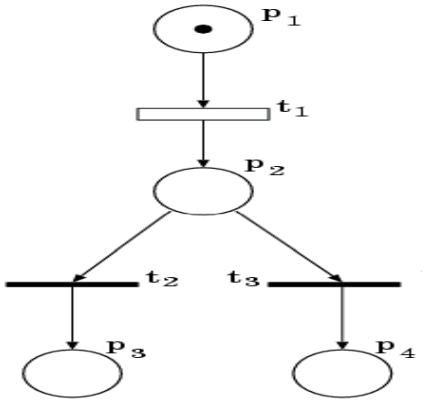

Figure 10.Modèles de RdPSG

1.6.1 Définition de Réseaux de Petri stochastique généralisé

Un réseau stochastique généralisé de Petri est 7-tuple $\mathcal{G} = \langle P, T, F, M_0, T_1, T_2, \Lambda \rangle$ où :

- P est un ensemble fini et non vide de places.
- T est un ensemble fini et non vide de transitions disjointes de P.
- $F : (P \times T) \cup (T \times P) \longrightarrow N$ est une relation de flux pour un ensemble d'arcs.
- $M_o : P \longrightarrow N$ est un marquage initial.
- $T_1 \subseteq T$ est un ensemble de transitions temporisées.
- $T_2 \subseteq T$ est un ensemble de transitions immédiates, telles que $T_1 = \emptyset$, $T_1 \cap T_2 = \emptyset$, $T_1 \cup T_2 = T$.
- $\Lambda : T \longrightarrow R^+$, où $\Lambda(t_i)$ est un taux/poids de t_i .

Dans les RdPSGs, le franchissement d'une transition immédiate t à un marquage M_V donnant un marquage M_{0V} est effectué en temps nul, donc, si un observateur qui regarde le marquage net ne verra pas M_V puisque ce dernier disparaîtra instantanément.

En d'autres termes, lorsque le réseau atteint le marquage de M_V à un instant *d*, la transition *t* sensibilisait immédiatement et le nouveau marquage de réseau devient M_{0V} au même instant *d*, donc le temps de séjour dans le marquage de M_V est nul. Ces marquages sont appelées marquages qui disparaissent. Une marque qui disparaît est une marque qui permet une transition immédiate. D'autre part, un marquage tangible est un marquage qui permet soit une transition temporelle, soit un marquage sans issue. Le processus stochastique séjourne dans de tels marquages sont répartis exponentiellement. Par conséquent, ces marquages ne sont pas laissés immédiatement **(2)**. Et parmi les problèmes que nous rencontrons dans le RdPSG, les cycles de transitions immédiates sont un problème complexe, leur tir circulaire est souvent appelé un piège intemporel, où aucune transition temporisée ne peut être tirée à l'avenir.

Considérons le processus stochastique illustré à la Fig.11, où le marquage *M2* est en train de disparaître, par ce qu'il permet des transitions immédiates.

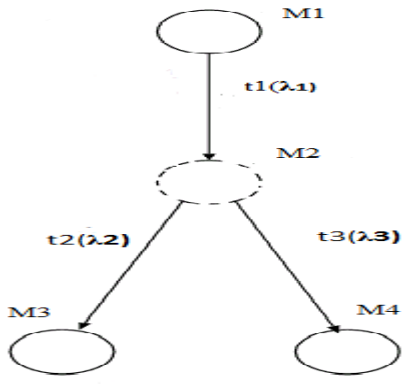

Figure 11.Processus stochastique correspondant du RdPSG G représenté à la Fig.10

1.6.2 Analyse quantitative de RdPSGs

Cette partie d'analyse consiste à calculer les probabilités stationnaires et les indices de performance. Elle est basée sur la chaîne de Markov associée au

RdPSG. Cette chaîne peut être construite à partir du graphe des marquages accessibles de la manière suivante **(38)** :

Théorème 1. Le graphe de marquage d'un RdPSG est isomorphe à une chaîne de Markov à temps continue.

Théorème 2. Un RdPSG bornée et tel que son graphe des marquages accessibles est fortement connexe est ergodique.

Théorème 3. Un RdPSG bornée et ergodique s'il admet le marquage initial comme état d'accueil.

Remarque : Le processus stochastique engendré par un RdPSG borné avec le marquage initial comme état d'accueil, peut être classé comme un processus semi- markovien à temps continu, à espace d'états fini, stationnaire et irréductible.

1.6.3 Evaluation des indices de performances

On a la distribution de probabilité pour les RdPSGs telle que $\pi = (\pi 1, \pi 2, ..., \pi n)$ à l'état stationnaire sur les marquages tangibles peut alors être obtenue par la résolution de système d'équation linéaire suivant **(39)** :

$$\begin{cases} \boldsymbol{\pi Q} = \mathbf{0}; \\ \sum_{i=1}^{r} \boldsymbol{\pi} = \mathbf{1} \end{cases} \qquad (1.9)$$

En utilisant la distribution des probabilités stationnaires, on peut calculer les indices de performances parmi lesquels on peut citer :

- **Fréquence moyenne de franchissement d'une transition :** Et ça s'appelle aussi (débit moyen) de tirs d'une transition, lequel est le nombre moyen de franchissement de t_i en une unité du temps. Elle est calculée par :

$$\boldsymbol{\lambda(tj) = \sum_{Mj \in E(ti)} \lambda i(Mj)\pi j} \qquad (\mathbf{1.10})$$

Remarque : $E(t_i)$ est l'ensemble des marquages telle que la transition ti est sensibilisé, et $\lambda(M_j)$ est le taux de franchissement de ti en M_j .

- **Nombre moyen des marques dans une place :** il est donné par l'équation suivante:

$$\boldsymbol{n(p) = \sum_{i:Mi \in E} Mi(p)\pi i} \qquad (\mathbf{1.11})$$

Remarque : Pour le marquage M_i, $M_i(p)$ est le nombre de jetons dans la place p, et E est l'ensemble des marquages accessibles.

1.6.4 La chaîne de Markov embarquée

Nous n'utilisons pas le temps de séjour pour analyser les RdPSGs, car il est nul dans le marquage de disparition, alors qu'il est réparti de façon exponentielle dans les marquages tangibles. Au lieu de cela, nous considérons la probabilité de passer d'un marquage M à un marquage M'. Ces probabilités sont données par :

$$\boldsymbol{P[M \rightarrow M0] = \frac{\sum_{ti \in \{M[ti\rangle M'\}} \Lambda(ti)}{\sum_{tj \in EN(M)} \Lambda(ti)}} \qquad (1.12)$$

Remarque : EN(M) est l'ensemble des transitions activées au marquage M.

Nous analysons la chaîne de Markov embarquée du processus stochastique correspondant des RdPSGs, parce que la probabilité de passer d'un marquage M (soit en voie de disparition ou tangible) à un marquage M' est indépendante du temps de séjour dans le marquage M. En utilisant la chaîne de Markov embarquée, la matrice de probabilité de transition comme indiqué ci-dessous. **(33)**. Où le côté inférieur droit décrit les probabilités de transition entre les états tangibles, le côté inférieur gauche décrit les probabilités de transition entre les états tangibles et les états disparus, le côté supérieur droit décrit les probabilités de transition entre les états disparus et les états tangibles, et enfin, le côté supérieur gauche décrit les probabilités de transition entre les états en voie de disparition , nous avons également $\hat{Y}$ désigne un ensemble d'états tangibles et $\hat{H}$ est une ensemble d'états disparition , m = | $\hat{H}$ |, et n = | $\hat{Y}$ |.

$$P = \left[\begin{array}{ccc|ccc} c_{11} & \cdots & c_{1m} & d_{11} & \cdots & d_{1n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{mm} & d_{m1} & \cdots & d_{mn} \\ \hline e_{11} & \cdots & e_{1m} & f_{11} & \cdots & f_{1n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ e_{n1} & \cdots & e_{nm} & f_{n1} & \cdots & f_{nn} \end{array}\right] \qquad (1.13)$$

Et on a :

$C_{ij} = P\,[M_i \rightarrow M_j \,|M_i \in \hat{H} \wedge M_j \in \hat{H}]$, $d_{ij} = P\,[M_i \rightarrow M_j \,|M_i \in \hat{H} \wedge M_j \in \hat{Y}]$,

$E_{ij} = P\,[M_i \rightarrow M_j \,|M_i \in \hat{Y} \wedge M_j \in \hat{H}]$, $f_{ij} = P\,[M_i \rightarrow M_j \,|M_i \in v \wedge M_j \in \hat{Y}]$.

1.7 Conclusion

Dans ce chapitre, nous avons présenté des réseaux de Petri qui représentent la forme de base d'autres classes de RdP. Leurs aspects intuitifs de la modélisation ont été discutés. De plus, la modélisation formelle et la vérification basée sur le RdP sont présentées réseaux Petri stochastiques qui fournissent une extension commune des réseaux Petri combinant la vérification qualitative et quantitative sont introduits. Enfin, plusieurs de leurs concepts sous-jacents tels que les chaînes de Markov,....etc. sont illustrés par certains exemples, nous avons également fourni une explication simplifiée sur RdPSGs. Malgré leur modélisation et leur pouvoir de décision, les réseaux Petri sont confrontés à plusieurs lacunes dans la conception et l'analyse des systèmes de structures dynamiques. Par conséquent, il est nécessaire d'introduire des structures dynamiques dans les réseaux de Petri afin de modéliser/vérifier ces systèmes de manière naturelle. Dans le chapitre suivant, nous nous intéressons à l'extension des réseaux Petri aux formalismes de structure dynamique.

Chapitre 2 : Les Réseaux de Petri stochastiques généralisés dynamique

2 Introduction

Aujourd'hui, plusieurs systèmes d'événements discrets se complexifient, deviennent structurellement dynamiques et interconnecté dans une façon variable. Ces systèmes sont conçus pour pouvoir modifier leur structure pendant le temps d'exécution en ajoutant ou en supprimant des parties, c'est pour les rendre capables de s'adapter aux nouvelles conditions et exigences auxquelles ils sont confrontés. Où de nombreuses études en cours sur ce type de systèmes se concentrent sur son principal avantage, qui est la reconfigurabilité.

Comme nous l'avons mentionné plus tôt qu'il avait utilisé des réseaux de Petri pour étudier ce type de systèmes, puis élargi pour inclure de nombreuses catégories qui sont considérées comme une extension d'il comme RdPSs et RdPSGs qui cadrent bien avec le comportement SED à différents stades de développement.

En fait, la modélisation de systèmes reconfigurables avec des RdPs de base (non configurables) rend les tâches du concepteur encore plus compliquées et, par conséquent, les modèles résultants seront souvent très grands, compliqués et difficiles à assimiler. L'analyse de ces modèles ne peut être que plus compliquée. En effet, les RdPs de base sont caractérisés par leur structure rigide qui empêche la modélisation, la vérification, la simulation et la visualisation de la structure dynamique de cette classe de systèmes. Pour surmonter ce problème, les chercheurs introduisent des structures dynamiques dans les RdPs, élargissant ainsi le formalisme standard **(21)** .

L'augmentation du pouvoir de modification du formalisme réduit son pouvoir de décision, bien que les transformations graphiques basées sur des règles offrent un cadre graphique basé sur la modification de la reconfiguration des RdPs. Par conséquent, nous constatons qu'il y a un grand effort pour trouver un compromis entre les niveaux de modification et de vérification.

L'idée derrière les RdPs reconfigurables est de se rapprocher de systèmes dynamiques réels et d'offrir des modèles réalistes reflétant les aspects inhérents de ces systèmes. Cependant, ces gains dans la modélisation sont au détriment du niveau d'analyse de sorte que certains, voire tous, les propriétés deviennent indécidables. Cette dernière lacune n'a pas empêché le développement de cette catégorie de formalismes et des recherches considérables sont menées au niveau de l'analyse **(2)** .

2.1 Systèmes de transformation des graphes

Les graphes sont très utiles pour décrire les SEDs et les structures complexes de manière directe et intuitive. Les systèmes de transformation des graphes (STG) **(40)** ajoutent à la description statique donnée par les graphes une dimension supplémentaire qui modélise l'évolution graphique par l'application de règles. STG a été reconnu pour avoir des applications fructueuses dans divers domaines de l'informatique **(41)**, et en particulier dans la modélisation et la spécification de systèmes concurrents et distribués, par ce que la transformation de graphes permet de modéliser facilement la dynamique des systèmes par une évolution de *la* structure graphique.

Les STGs consistent en un système *G* d'un graphe de départ G_0 qui modélise une structure initiale et un ensemble de règles de réécriture *RR* qui exprime la reconfiguration du système. Chaque règle de transformation se compose d'un *L* de gauche et d'un *R* de droite.

Gardez à l'esprit que le graphe *G* est une configuration de $\mathcal{G}$. lorsque l'occurrence de *L* est présente en *G*, la règle *RR* est applicable, son application supprimant l'occurrence de *L* et ajoutant l'occurrence de *R* à *G*. Un graphique d'interface est parfois fourni dans STG, afin d'identifier l'association de R avec le reste du graphique reconfiguré et aussi décrire certaines des pièces qui doivent être sauvegardées.

2.1.1 Définition des Systèmes de transformation des graphes

Un système de transformation graphe (STG) $\mathcal{G} = (G_0, RR)$, se compose d'un graphe de départ G_0 et d'un ensemble de règles de réécriture *RR.* Un graphe *G* est généré par $\mathcal{G}$ si *G* est obtenu en appliquant un ensemble de règles dans *RR* à G_0. Le STG a plusieurs approches, nous allons étudier l'approche à Double-Pushout de manière simple.

2.2 Approche à Double-Pushout pour les RdPs

L'approche DPO utilise des morphismes graphiques qui mappent les transitions aux transitions et les places aux places, de sorte que si place p est mappé pour placer p', alors un mappage entre leur préréglage et postset doit exister, ceci est similaire aux transitions.

2.2.1 Morphismes sur les RdPs

Les morphismes sont donnés comme une paire de mappages pour les transitions et les places préservant la structure et le marquage. Donné deux réseaux de Petri :

H_1 (P1, T1, F1, M1) et H_2 (P2, T2, F2, M2), Le morphisme $\mathcal{F}$ entre les deux réseaux H_1 et H_2 est une fonction $\mathcal{F}$: $\mathcal{F}$: ($H_1 \rightarrow H_2$), Nous avons : $\mathcal{F} = (\mathcal{F}_p, \mathcal{F}_T)$, de sorte que : $\mathcal{F}_T$ (T1 T2), et $\mathcal{F}_p$ (P1 P2) sont deux morphismes qui :

Cartographier les transitions en transitions et les places en places, respectivement. $\mathcal{F}_p$ et $\mathcal{F}_T$ satisfaire :

$$\forall p'_1 \in {}^{o}t' \Rightarrow \exists p_1 \in {}^{o}t : \mathcal{F}_p\ (p_1) = p'_1 \text{ et } \forall p'_1 \in t'^{o} \Rightarrow \exists p_1 \in t^{o} : \mathcal{F}_p(p_1) = p'_1 \qquad (1)$$

$$\mathcal{F}_p\ (M_1(p)) \leq M_2(\mathcal{F}_p\ (p)) \qquad (2)$$

Exemple : Dans l'exemple ci-dessous, nous avons deux réseaux de Petri H1 et H2, t1' est l'image de t1, et la source de t1' est p0' qui est en H2, l'image de p0 la source de t1 en H1, aussi la cible de t1' est p1' en H2 qui est l'image de p1 la cible de t1 en H0.

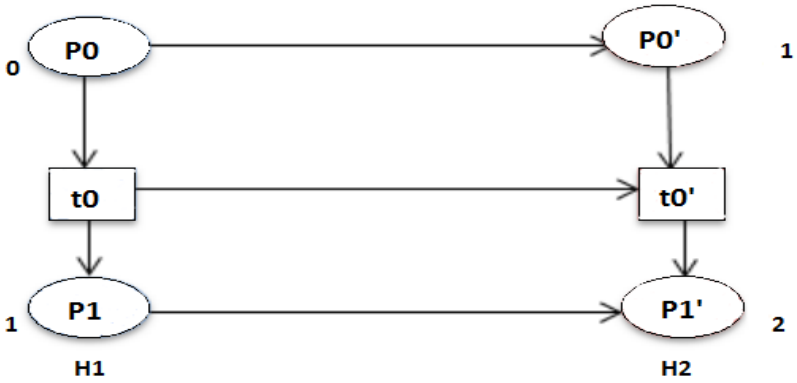

Figure 12.Exemple de Morphisme

2.2.2 Union de RdP comme "pushout "

Sur la base des morphismes sur les RdPs, il est possible de définir une construction spécifique qui est pushout (ou l'union) de deux RdPs (42). Que N_1 et N_2 soient deux RdPs, avec les deux morphismes, $\mathcal{F}: I \rightarrow N_1$ *et* $g : I \rightarrow N_2$, telle que Le réseau I est dit une interface commune entre N_1 et N_2. L'union de N_1 et N_2 est le RdP N défini en utilisant les deux morphismes :

$\mathcal{F}' : N_1 \rightarrow N$ *et* $g' : N_2 \rightarrow N$. On écrit $N = N_1 + I\ N_1$. L'opérateur +I est appelé le cons -truction de pushout ou l'opérateur de l'union (42). Nolte a résumé ceci dans le suivant (43):

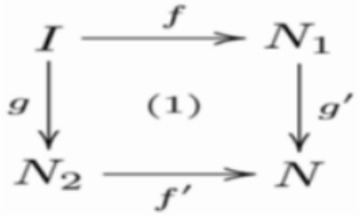

Figure 13.Schéma de pushout

2.3 RdPSG dynamiques

2.3.1 Introduction

Il est intéressant de noter que les réseaux Petri ont de nombreux algorithmes de vérification améliorés, pour profiter de ce dernier, des extensions ont été proposées pour enrichir les réseaux Petri avec reconfigurabilité.

Cependant, à notre connaissance, les formalismes proposés dans la littérature ne permettent que la topologie dynamique, c'est-à-dire que les ensembles de places et de transitions ne peuvent pas être modifiés. Cette dernière restriction a été motivée par la nécessité de permettre la vérification des réseaux à structure dynamique par leur codage ou leur transformation en RdP de base. Toutefois, cette restriction limite fortement la puissance de modélisation de ces formalismes **(2)**.

Les réseaux de Petri stochastiques généralisés dynamiques (RdPSGDs) permettent de modéliser des ensembles dynamiques de places et de transitions et /ou des arcs. En outre, nous fournirons une explication sur la possibilité de convertir RdPSGD en RdPSG afin de les vérifier, car ce dernier conserve les comportements stochastiques des dynamiques, nous permettant d'utiliser de nombreuses façons et des outils de vérification proposés pour les RdPSGs dans l'analyse des RdPSGDs. Tout d'abord, nous présentons la définition formelle du formalisme des RdPSGDs. Par la suite, nous présentons l'algorithme qui transforme les RdPSGDs en RdPSGs, puis nous décrivons sa vérification qualitative/quantitative.

2.3.2 Définition formelle

La structure d'un RdPSGD consiste en un ensemble de RdPSG dont chacun décrit une configuration. Cet ensemble peut être obtenu en transformant une configuration initiale RdPSG G0 d'un RdPSGD via un ensemble de règles de transformation.

Les RdPSG dynamiques bénéficient de structures reconfigurables qui peuvent être modifiées à l'exécution. Un changement structurel est modélisé par une règle de reconfiguration dont l'application peut supprimer/ajouter des places, des transitions et/ou des arcs donnant une nouvelle configuration. Par souci de simplicité, les règles de reconfiguration sont définies comme une structure composée d'une configuration source G^s, pré-conditions M et une configuration cible G^t .On écrit $w = \langle G^s, M, G^t \rangle$, C'est-à-dire, si une règle $w = \langle G^s, M, G^t \rangle$ est appliquée à un RdPSGD D à la configuration G^s et au marquage M, alors D change sa configuration vers G^t **(44)**.

2.3.3 Définition RdPSG dynamique

On définit un RdPSGD **(2)** comme un quadruple $\mathcal{D} = \langle G_0, \mathcal{G}, \mathcal{R}, \Omega \rangle$ telles que:

- G_0 est une configuration initiale de $\mathcal{D}$,
- $G = \{G_0, G_1, \ldots, Gn\}$ est un ensemble fini de configurations,
- $\mathcal{R} = \{W_0, \ldots, W_n\}$ est un ensemble fini de règles non vides,
- $\Omega : R \longrightarrow R^+$ associe un délai d'application (réparti de façon exponentielle) à chaque règle de reconfiguration.

Un RdPSGD peut être défini d'une autre manière comme une paire $\mathcal{D} = \langle G_0, \mathcal{R} \rangle$, où :

• G_0 est une modélisation RdPSG d'une configuration initiale,

• $\mathcal{R} = \{w_0, \ldots, w_n\}$ est un ensemble de règles.

2.3.4 Règle de transformation

Une règle w est écrite comme $w = \langle G^s, M, G^t \rangle$, telle que :

- G^s est une source RdPSG,
- M est un sous-marquage de G^s auquel w est applicable,
- G^t est un RdPSG cible.

Nous Supposons que D est un RdPSGD et la règle de reconfiguration w est appliquée à D. Afin réaliser faire cette application, il faut que la configuration actuelle de D est G^s, et le marquage M est contenu dans le marquage actuel de G^s. Le lecteur peut en savoir plus sur ce sujet dans de nombreux publications notamment **(2)** , **(45)**.

Exemple : Considérons un RdPSGD D0 = ⟨N0, $\mathcal{G}$, $\mathcal{R}$, Ω ⟩, tel que :

- N0 est représenté à la figure 14,
- $\mathcal{G}$ = {N0, N1}, telle que N1 est représenté à la figure 15,
- $\mathcal{R}$ = $\{w_1, w_2\}$, telle que w_1 = ⟨N0, M_1, N1⟩ où M_1 (p_0) = 3 (la configuration actuelle de D0 est N0 et le marquage actuel de la place p_0 est trois, D0 passe à la configuration N1), et w_2 = ⟨N1, M_2, N0 ⟩ où M_2 (p_1, p_3) = (3, 2) (la configuration actuelle de D0 est N1 et les marquages actuels de p_1 et p_3 sont trois et deux, respectivement, D0 passe à la configuration N0),
- Ω (w_1) = α et Ω (w_2) = β.

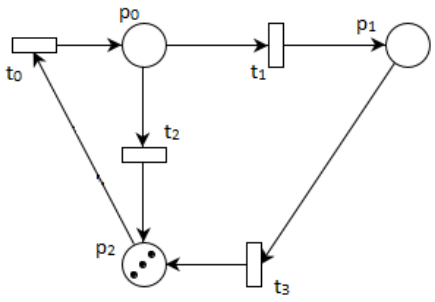

Figure 14.Configuration N0

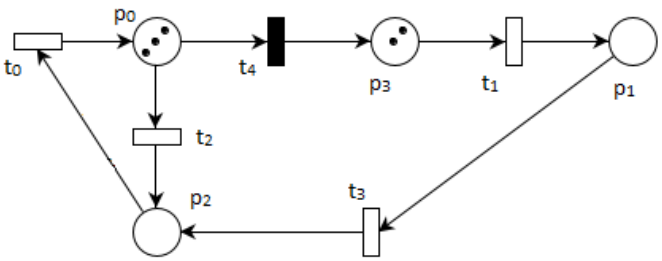

Figure 15.Configuration N1

2.3.5 Transformation des RdPSGDs en RdPSGs

Comme nous l'avons mentionné précédemment, lors de la conversion de RdPSGD en RdPSGD équivalent, nous pouvons utiliser les méthodes et les outils d'analyse de ce dernier pour analyser RdPSGD. Il existe un algorithme qui calcule un RdPSG N équivalent pour tout RdPSGD D donné ayant un nombre fini de configurations. Où, pour un RdPSGD D, nous avons ce qui suit :

- $\mathcal{P}$ désigne un ensemble de toutes les places de toutes les configurations de *G*, formellement $\mathcal{P} = \bigcup_{Gi \in G} PGi$.
- $\mathcal{T}$ désigne un ensemble de toutes les transitions de toutes les configurations de G, formellement $\mathcal{T} = \bigcup_{Gi \in G} TGi$.

Et pour une configuration donnée G_i, nous trouvons ce qui suit :

- P_{Gi} désigne son ensemble de places.
- T_{Gi} indique son ensemble de transitions.
- F_{Gi} indique sa fonction de flux.
- M_0 Gi indique son marquage initial.
- Λ_{Gi} désigne une fonction associant des délais/poids de franchissement aux transitions de G_i.

Afin de préserver les comportements d'un RdPSGD D donné, sa transformation vers N = $\langle P, T, F, M_0, T_1, T_2, \Lambda \rangle$ utilise un ensemble de morphismes $\{\mathcal{F}_0, . ., \mathcal{F}_n\}$, qui mappent les configurations de $\mathcal{D}$ en N. Un morphisme $\mathcal{F}_i$: $P_{Gi} \cup T_{Gi} \longrightarrow P \cup T$ mappe les places et les transitions de G_i en places et transitions de N, telle que la condition suivante s'applique : pour toute paire de nœuds $(x, y) \in (P_{Gi} \times T_{Gi}) \cup (T_{Gi} \times P_{Gi})$, $\mathcal{F}_{Gi}(x, y) = \mathcal{F}(\mathcal{F}_i(x), \mathcal{F}_i(y)))$. Ainsi, chaque $\mathcal{F}_i$ conserve la fonction d'écoulement $\mathcal{F}_{Gi}$ en N **(44)**. On peut créer un équivalent RdPSGD N à un RdPSGD D par algorithme **(2)** qui est implémenté comme suit :

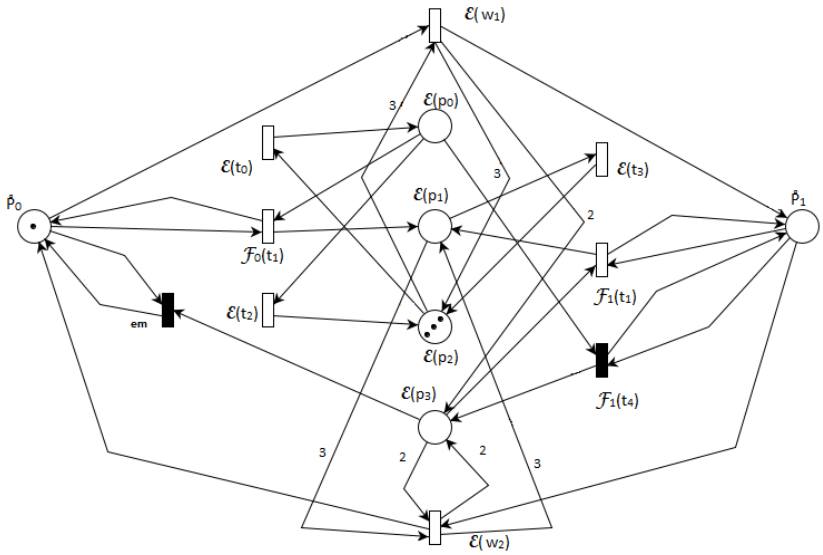

Figure 16.Équivalent RdPSG N0 à RdPSGD D0

Etape1 (Ajout de places équivalentes) : Pour chaque place $p \in \mathcal{P}$, insérer une place équivalent, noté par $\mathcal{E}(p)$, en $\mathcal{P}$ (initialement vide), telle que si $p \in P_{G0}$, puis $M^0(E(p)) = M^0_{G0}(p)$; sinon $M^0(E(p)) = 0$. Nous considérons le RdPSGD D_0, on a $\mathcal{P} = \{ p_0, p_1 , p_2, p_3\}$, ensemble des places équivalentes, désignent par $\mathcal{E}(\mathcal{P})$, devient $\mathcal{E}(\mathcal{P}) = \{ \mathcal{E}(p_0), \mathcal{E}(p_1), \mathcal{E}(p_2), \mathcal{E}(p_3)\}$, et le marquage initial des places équivalentes est M^0 ($\mathcal{E}(p_0)$, $\mathcal{E}(p_1)$, $\mathcal{E}(p_2)$, $\mathcal{E}(p_3)$), = (0, 0, 3, 0).

Etape2 (Ajout des places pour émuler les configurations) : Créer $\mathring{P}$ un ensemble de places **{**$\mathring{P}_0, \ldots, \mathring{P}_n\}$ associées à un ensemble de configurations $\mathcal{G} = \{G_0, \ldots G_n\}$. Par conséquent, chaque place $\mathring{P}_i \in \mathring{P}$ est associée à configuration G_i**,** respectivement. Un jeton à $\mathring{P}_i$ signifie que la configuration actuelle de D est G_i, ainsi $\sum_{i=0}^{n} \boldsymbol{M}(\mathring{\mathbf{P}}\mathbf{i}) = \mathbf{1}$, puisque D ne peut pas être en même temps dans deux configurations. Considérant à nouveau D**,** il a deux configurations possibles N0 et N1 (illustrée à la fig. 16), qui sont associés aux endroits $\mathring{P}_0$et $\mathring{P}_1$, respectivement. Selon $\mathring{P}$ = $\{\mathring{P}_0, \mathring{P}_1\}$, M^0 ($\mathring{P}_0, \mathring{P}_1$) = (1, 0) et $\mathcal{P} = \{\mathcal{E}(p_0), \mathcal{E}(p_1), \mathcal{E}(p_2), \mathcal{E}(p_3)\} \cup \mathring{P}$.

Etape3 (Ajout de transitions équivalentes): Nous considérons deux cas :

Cas 1 : Pour chaque transition *t* qui ne change pas ses paramètres (preset, postset, taux et /ou type) dans n'importe quelle configuration, nous insérons une transition équivalente $\mathcal{E}(t)$ de *t* en *T* ayant un taux/type indenté. De plus, nous préservons son preset et son postset, de sorte que pour tout la place *p* de G_0, s'il existe un arc de *t* à *p* (de *p* à *t*), ensuite nous ajoutons un arc de $\mathcal{E}(t)$ à $\mathcal{E}(p)$ (de $\mathcal{E}(p)$ à $\mathcal{E}(t)$). Formellement, $F(\mathcal{E}(t), \mathcal{E}(p)) = F_{G0}(t, p)$ et $F(\mathcal{E}(p), \mathcal{E}(t)) = F_{G0}(p, t)$, pour tout $p \in P_{G0}$. Laissez $\mathcal{E}(t) = F_i(t_1)$, $\forall\, i \in \{0, \ldots, n\}$. Considérer la transition t_0 et t_3 en Fig. 14, elle ne change pas ses paramètres en N0 ou N1, donc on insère une transition $\mathcal{E}(t_0)$ en *T*, telle que $F(\mathcal{E}(t_0), \mathcal{E}(p_0)) = F_{N0}(t_0, p_0)$ et $F(\mathcal{E}(p_2), \mathcal{E}(t_0)) = F_{N0}(p_2,t_0)$, et insère une transition $\mathcal{E}(t_3)$ en *T*, de sorte que $F(\mathcal{E}(t_3), \mathcal{E}(p_2)) = F_{N0}(t_3, p_2)$ et $F(\mathcal{E}(p_1), \mathcal{E}(t_3)) = F_{N0}(p_1,t_3)$.

Cas2 : Afin de modéliser l'évolution de la structure de RdPSGD D par rapport aux transitions, nous dupliquons chaque transition aussi souvent qu'elle apparaît dans les configurations de D. Cette duplication des transitions signifie modéliser ses différents paramètres (type, taux, presets et/ou postsets).

Pour chaque transition t_j de chaque configuration G_i, nous insérons une transition équivalente $\mathcal{F}_i(t_j)$ de t_j en T ayant un taux/type identique. Aussi, nous préservons son preset et postset, de sorte que pour toute la place p de G_i, s'il existe un arc de t_j à p (de p à t_j), puis on ajoute un arc de $\mathcal{F}_i(t_j)$ à $\mathcal{E}(p)$ (de $\mathcal{E}(p)$ à $\mathcal{F}\mathcal{F}_i(t_j)$. Formellement, $F(\mathcal{F}_i(t_j), \mathcal{E}(p)) = F_{Gi}(t_j, p)$ et $F(\mathcal{E}(p), \mathcal{F}_i(t_j)) = F_{Gi}(p, t_j)$, pour tout $p \in P_{Gi}$. Enfin, nous connectons $\mathcal{F}_i(t_j)$ avec p_i par une boucle automatique, donc $\mathcal{F}_i(t_j)$ est désactivé si la configuration actuelle n'est pas G_i. Considère la transition t_1 de N0 (Fig. 14), nous insérons une transition $\mathcal{F}_0(t_1)$ dans T, telle que $F(\mathcal{F}_0(t_1), \mathcal{E}(p_1)) = F_{N0}(t_1, p_1)$, $F(\mathcal{E}(p_0), \mathcal{F}_0(t_1)) = F_{N0}(p_0, t_1)$, et $F(\mathcal{F}_0(t_1), \overset{\circ}{P}_0) = F(\overset{\circ}{P}_0, \mathcal{F}_0(t_1),) = 1$.

Etape 4 (ajout de transitions émulant les règles) : Chaque règle de reconfiguration $w = (G_i, w, G_j) \in \mathcal{R}$ est modélisé par une transition désignée par : $\mathcal{E}(w)$, telle que :

1. Insérer la transition $\mathcal{E}(W)$ dans T. Formellement, $T \leftarrow T \cup \{\mathcal{E}(w)\}$,
2. $F(\overset{\circ}{P}_i, \mathcal{E}(w)) = F(\mathcal{E}(w), \overset{\circ}{P}_j) = 1$. Autrement dit, le franchissement $\mathcal{E}(w)$ supprime un jeton de $\overset{\circ}{P}_i$ (associé à la configuration G_i) et ajoute un jeton à $\overset{\circ}{P}_j$ (associé à la configuration G_j), dont les modèles passent de G_i à G_j.
3. Pour chaque place p impliqué dans M, soit $F(\mathcal{E}(p), \mathcal{E}(w)) = F(\mathcal{E}(w), \mathcal{E}(p)) = (p)$, qui modélise les conditions préalables à l'application de la règle w,
4. Pour chaque place p, soit $F(\mathcal{E}(w), \mathcal{E}(p)) = M^0{}_{Gj}(p)$. C'est-à-dire, tirer $\mathcal{E}(w)$ initialise $\mathcal{E}(p)$, qui émule l'ajout de p, pour chaque place d'interface p,

5. Pour chaque place obsolète p, ajouter une transition immédiate *em* qui vide $\mathcal{E}(p)$ des jetons pour émuler la suppression de *p* comme suit : (1) ajouter un arc de $\mathcal{E}(p)$ à *em*, c'est-à-dire $F(\mathcal{E}(p), em) = 1$, et (2) relier *em* à $\overset{o}{P}_j$ par une auto-boucle, ce qui permet à *em* de commencer à vider $\mathcal{E}(p)$ lorsque la configuration actuelle devient Gj qui ne contient pas de place *p*,

6. $\Lambda(\mathcal{E}(w)) = \Omega(w)$.

En raison de la nature de RdPSG, nous constatons que si une transition immédiate vidant une place est activée, Notez que, toute transition temporisée émulant une application de règle est désactivée. Par conséquent, toute émulation de reconfiguration n'est pas autorisée tant que chaque transition qui vide une image de place obsolète n'est pas activée. En fait, ce dernier garantit que le réseau équivalent n'émule jamais l'ajout d'une place avec des jetons supplémentaires que spécifié par la règle *w*.

Ensuite, l'équivalent net N doit retirer tous les jetons de la place *p*, avant d'émuler une reconfiguration de G_j vers G_k (sinon, si *p* contient toujours des jetons, alors son marquage ne sera pas égal à n). Ce comportement est garanti par l'étape (4).

Par exemple, la règle w_2 de RdPSGD D_0 est modélisée par la transition $\mathcal{E}(w_2)$, représentée à la fig. 16, telle que :

(1) $F(\overset{o}{P}_1, \mathcal{E}(w_2)) = F(\mathcal{E}(w_2), \overset{o}{P}_0) = 1$, pour modéliser le passage de N1 à N0,

(2) Quant au place p_1, on a : $F(\mathcal{E}(p_1), \mathcal{E}(w_2)) = M_2(p_1) = 3$ et $F(\mathcal{E}(w_2), \mathcal{E}(p_1)) = M_2(p_1) = 3$,

(3) Quant au place obsolète p_2, nous avons : $F(\mathcal{E}(p_3), em) = 1$ pour modéliser sa suppression, de sorte que *em* est une transition immédiate et qu'il est relié à la place $\overset{o}{P}_0$ (correspondant à la configuration cible N0) par une auto-boucle.

Comme pour la règle w_1, il est modélisé par la transition $\mathcal{E}(w_1)$ montrée à la Fig. 16, telle que :

(1) F ($\overset{\circ}{P}_0$, $\mathcal{E}$ (w_1))= F ($\mathcal{E}(w_1)$, $\overset{\circ}{P}_1$) = 1, pour modéliser le passage de N0 à N1,

(2) Comme pour la place frais p_3, $F(\mathcal{E}(w_1)$,E (p_3)) = M^0_{N1} (p_3) = 2,

(3) Pour la place p_0, on a : F ($\mathcal{E}$ (p_2), $\mathcal{E}$ (w_1)) = M_1 (p_2) = 3 et F ($\mathcal{E}$ (w_1), $\mathcal{E}$ (p_2)) = M_1 (p_2) = 3.

2.3.6 Analyse qualitative/quantitative de RdPSGD

En analysant RdPSGD N équivalent à RdPSGD D, nous pouvons dériver et réaliser une analyse quantitative et qualitative de ce dernier. Nous considérons certaines propriétés importantes, nous avons : **(44)**

- Une place p de $\mathcal{D}$ est k-borné, si $\mathcal{E}$ (p) est k-borné.
- $\mathcal{D}$ est k-borné (respectivement), si N est k-borné (respectivement).
- Une transition t de $\mathcal{D}$ est vivante, s'il existe une transition vivante équivalente.
- Une règle w ϵ $\mathcal{R}$ est vivante, s'il existe une transition $\mathcal{E}(w)$ vivant est sont la modélisation de w.
- **Probabilité de franchissement des transitions :** La probabilité pr_t, que la transition t de D tirs suivante est donnée par :

$$\boldsymbol{prt} = \sum_{\boldsymbol{t} \,\epsilon\, \mathbf{Gi}} \boldsymbol{pr}(\boldsymbol{\mathcal{F}}\mathbf{i}(\mathbf{t})$$

Où pr ($\mathcal{F}_i$(t)) est une probabilité que la transition $\mathcal{F}_i$(t) se déclenche ensuite dans N.

- **Nombre moyen de jetons :** Nombre moyen de jetons à place *p* égale le nombre moyen de jetons à sa place $\mathcal{E}$(p).
- **Probabilité d'avoir n jetons à une place** : La probabilité d'avoir *n* jetons à la place *p* de $\mathcal{D}$ est égale à la probabilité d'avoir *n* jetons à sa place équivalente $\mathcal{E}$(p) de N.

2.4 Conclusion

Dans ce chapitre, présenté les réseaux de Petri stochastique généralisée dynamique (RdPSGD) qui est une extension de réseau de Petri stochastique généralisée (RdPSG). On a décrit la transformation des réseaux de Petri stochastiques généralisés dynamiques en réseaux de Petri stochastiques généralisés (la transformation proposée vers les RdPSGs ne peut avoir place que lorsque la configuration obtenue définie par des transformations graphiques est finie), de telle sorte que les propriétés qualitatives et quantitatives soient encore décidables en utilisant les méthodes d'analyse proposées pour les RdPSGs , où RdPSGDs permet des formes de reconfiguration sans restriction tout en préservant le pouvoir de décision RdPSGs.

Chapitre 3 : Implémentation

3 Introduction

Après les étapes d'analyse et de conception qui sont mentionnées dans le chapitre précédent, nous devons passer aux prochaines étapes du projet, qui sont le codage et le test. Ces phases visent à mettre en œuvre un outil qui traite des réseaux de Petri stochastiques généralisés reconfigurable. La première section présente brièvement les outils de développement et les langages que nous avons appris et les exploiter dans la réalisation de notre projet. La deuxième section présente les principaux résultats de la mise en œuvre de notre application finale.

3.1 Outils et langages de développement

Dans cette section, nous présentons différents outils et langages, qui nous aident lors de la réalisation de notre projet dans les deux niveaux (niveau de programmation, et niveau théorique).

3.1.1 Langage de programmation JAVA

Java est un langage de programmation informatique orienté objet que nous avons utilisé dans l'implémentation de notre application. Java est open source et a une communauté mondiale investie pour guider son développement et sa croissance continus. Java a la particularité principale que les logiciels écrits avec ce dernier sont très facilement portables sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS ou GNU/Linux avec peu ou pas de modifications.

3.1.2 Eclipse

Eclipse est un environnement de développement intégré (IDE) open source, pour la programmation java. C'est un assistant de codage puissant. C'est un éditeur approprié pour écrire et tester de nombreuses lignes de code et de classes, car il offre une vue de projet structurelle, et une navigation rapide des fichiers.

3.1.3 PIPE

Platform Independent Petri net Editor est un outil open source, indépendant de la plate-forme pour créer et analyser les réseaux Petri, y compris les RdPSGs. Il est entièrement implémenté en Java pour sécuriser l'indépendance de la plateforme et fournit une interface utilisateur graphique élégante et facile à utiliser qui permet de créer, enregistrer et charger des réseaux Petri conformes au format d'échange PNML. PIPE offre également une suite complète de modules d'analyse pour vérifier les propriétés comportementales, produire des statistiques de performance, et certaines fonctionnalités moins communes telles que la comparaison et la classification des RdPs.

3.1.4 XML

XML (Extensible Markup Languge) est une extension de fichier pour un format de fichier XML utilisé pour créer des formats d'information communs et partager à la fois le format et les données sur le World Wide Web, intranets, et ailleurs en utilisant le texte ASCII standard. Nous avons utilisé ce langage pour stocker nos fichiers (RdPs, règles de transformation) à partir de notre outil.

3.1.5 PNML

PNML est une proposition de format d'échange XML pour les réseaux Petri. A l'origine, le PNML était destiné à servir de format de fichier pour la version Java du Petri Net. Mais, il s'est avéré qu'actuellement plusieurs autres groupes développent également un format d'échange basé sur XML. Ainsi, le PNML n'est qu'une contribution à la discussion en cours et aux efforts de normalisation d'un format basé sur XML.

3.2 Implémentation

De plus, nous avons utilisé un ensemble de logiciels et de matériel qui sont résumés dans le tableau 4.1 ci-dessous.

Logiciel / materiel	**Version**
OS	Microsoft Windows 10 Home, 64bits
CPU	Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz
RAM	4.00Go
Eclipse	IDE for Java Developers - 2022-03
PIPE	4.2.1

Table 2.Versions logicielles et matérielles

3.3 Page d'accueil de l'outil

Cet outil permet de modéliser et de convertir RdPSGDs en RdPSGs en appliquant l'algorithme susmentionné. Il permet également l'édition des règles de transformation qui sont appliquées entre l'ensemble de configurations, ainsi que les résultats sont exportés vers des fichiers XML avec une grammaire spécifique que nous avons proposé à quelles extensions ont été données (PML) en

fonction de la grammaire dans PIPE, les résultats que nous avons obtenus sont exportés vers cette dernière pour être affichés dans les graphiques de formulaire.

L'outil a été conçu pour être facile à utiliser avec une interface simple (voir la figure 17). Il se compose de trois zones : Source, Target et Marking, il se compose également de quatre boutons comme suit : Add, Delete, Transform et Save, nous apprendrons sur le travail de chacun d'eux plus tard.

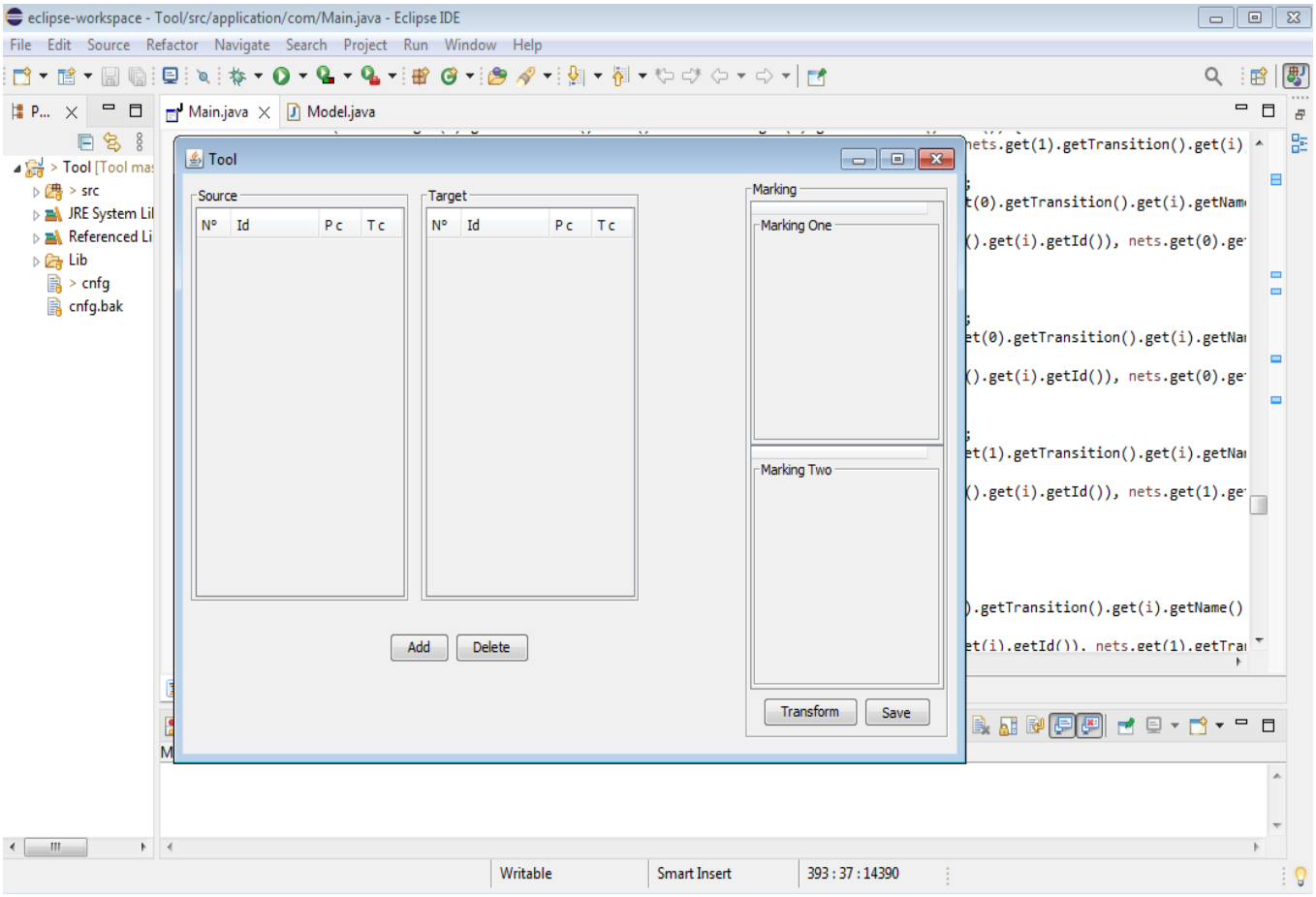

Figure 17. Page d'accueil de l'outil

3.3.1 Les listes Source et Target

Nous avons développé des listes source et Target pour faciliter l'utilisation de l'application. Par le bouton "Add", nous pouvons ajouter un nombre spécifique de

configurations (il a déjà été écrit comme un fichier XML dans l'application PIPE)pour obtenir une liste exhaustive des réseaux de Petri apparaissant dans les listes Source et Target .Lors de l'ajout de toute configuration, il est distingué par de nombreuses caractéristiques comme suit : nombre des réseau dans la liste (N), l'identification(Id), nombre des places (Pc) et nombre des transitions (Tc). Et par le bouton "Delete", nous pouvons supprimer toute configuration identifiée dans Source ou Target.

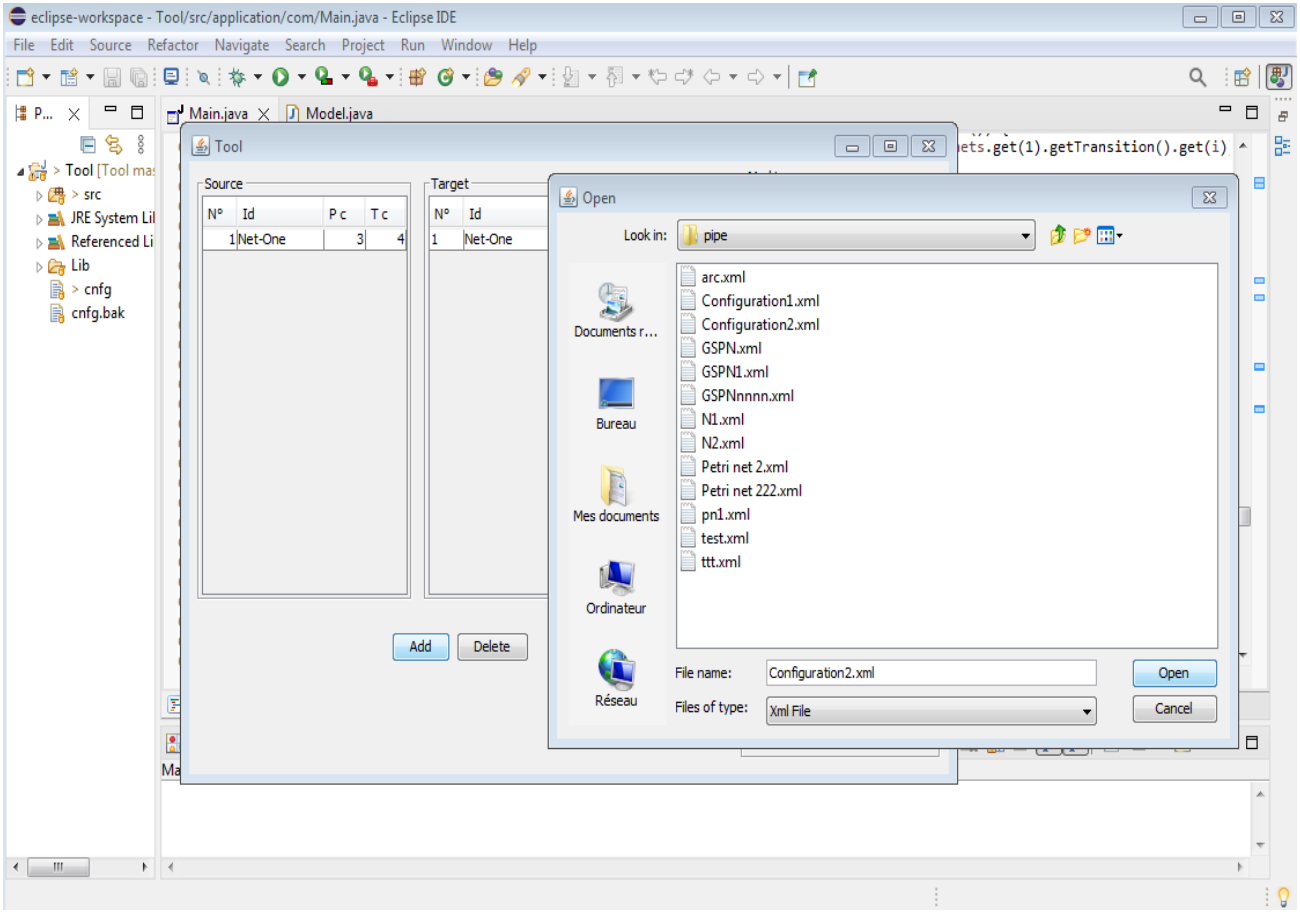

Figure 18. Ajouter une configuration

3.3.2 L'espace Marking

Marking contient deux listes: Marking One et Marking Two. Lors de la sélection d'une configuration dans la liste "Source", une liste apparaît dans

Marking One afin que son nombre d'éléments soit déterminé en fonction du nombre de places dans la configuration précédemment sélectionnée, afin d'ajouter le marquage (l'utilisateur va l'ajouter) par lequel la reconfiguration se fera de la configuration sélectionnée dans la table "Source"(nous allons choisir les configurations dans la figure 14) à la seconde configuration que nous choisirons dans la table "Target"(la configuration dans la figure 15). Et la même chose quand a sélection d'une configuration dans la liste "Target", une liste apparaît dans Marking Two afin que son nombre d'éléments soit déterminé en fonction du nombre de places dans la configuration qui sélectionnée, pour ajouter le marquage par lequel la reconfiguration est effectuée à partir de la configuration choisie dans la table Target à la configuration que nous avons initialement sélectionnée dans la table.

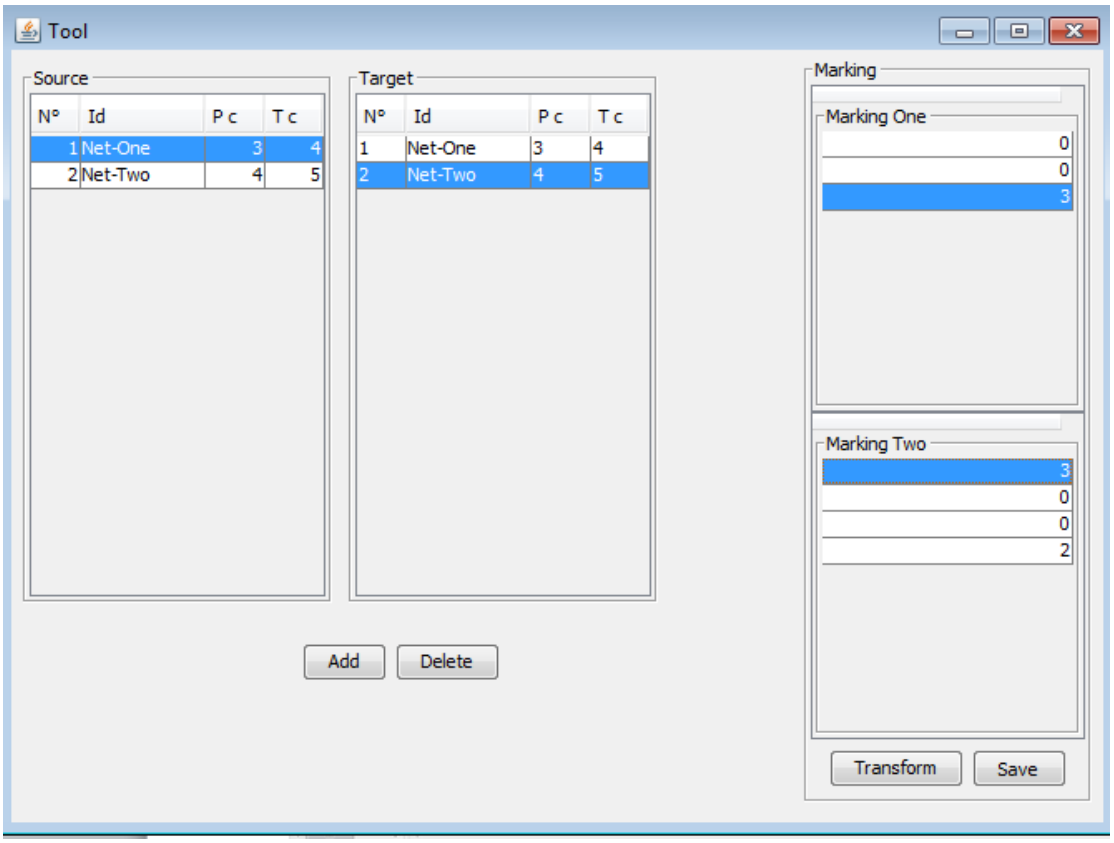

Figure 19.Ajouter les marquages de la reconfiguration

Lorsque le bouton Transform est enfoncé, l'algorithme que nous avons étudié précédemment est appliqué pour faire la transformation de RdPSGD (l'ensemble de configurations que nous avons sélectionné) en RdPSG, ainsi un nouveau fichier est créé au format PNML pour RdPSG résultant de la conversion.

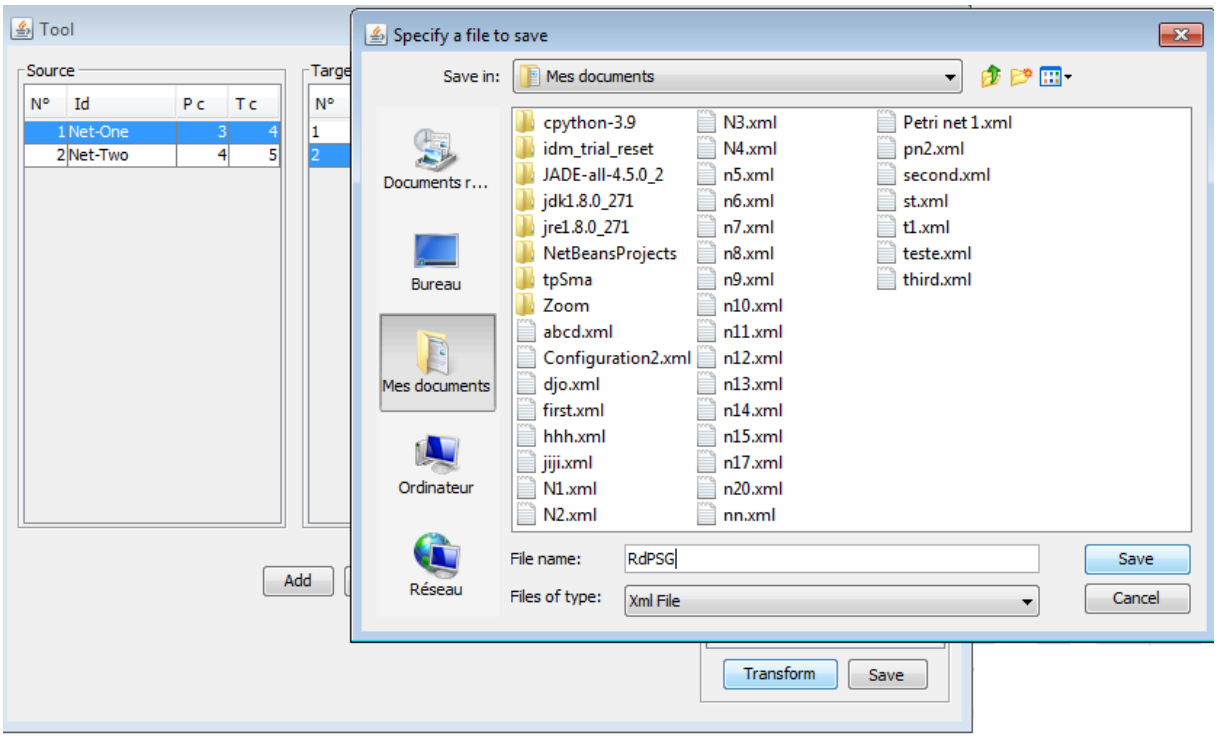

Figure 20 .L'application de l'algorithme

Afin de valider ces résultats, nous allons ouvrir ce fichier dans PIPE et le résultat apparaîtra sous forme de graphique comme le montre la figure suivante

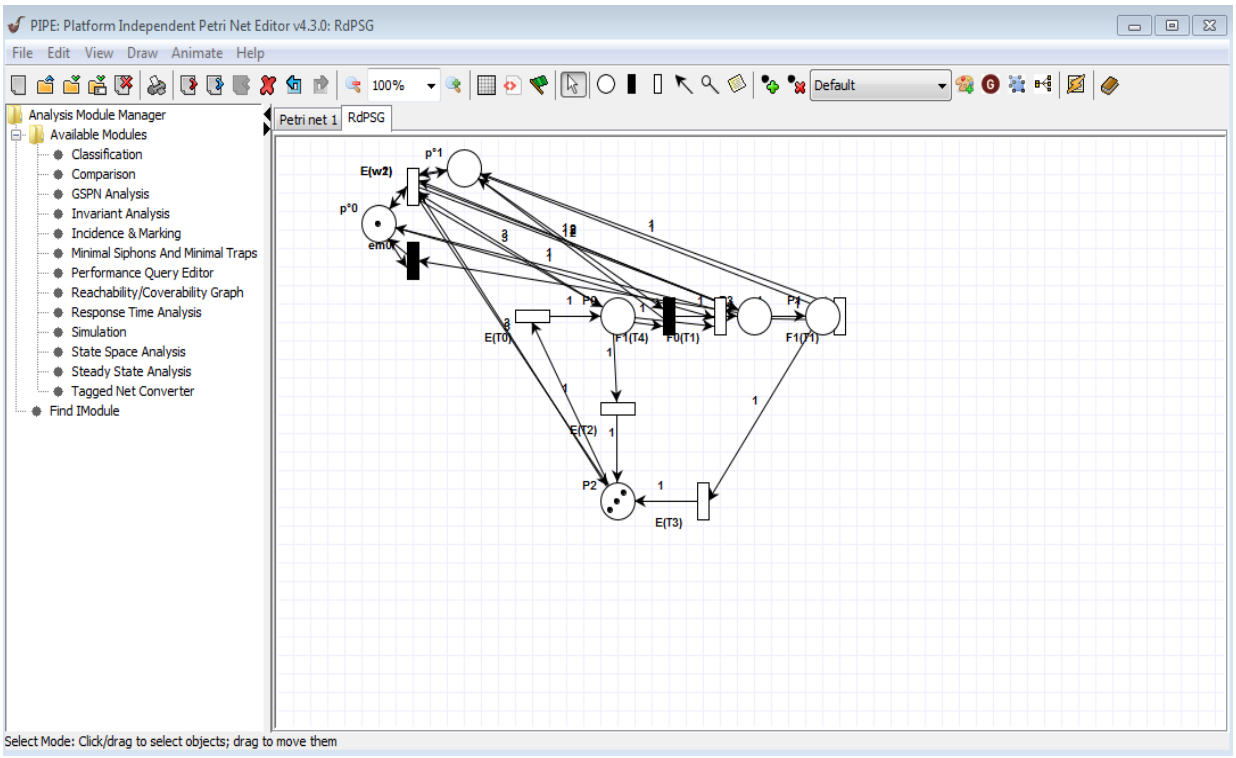

Figure 21. Le résultat de notre outil par PIPE

Dans notre travail, et dans le graphisme en particulier, nous n'avons pas pris en compte les positions des transitions ainsi que les places du réseau à former, parce que c'est un autre domaine et différent de notre étude, qui fait apparaître le graphe de cette façon.

Mais d'autre part, si nous organisons les positions des places et des transitions manuellement (de cette façon nous pouvons vérifier que nos résultats sont corrects), nous obtenons le résultat suivant :

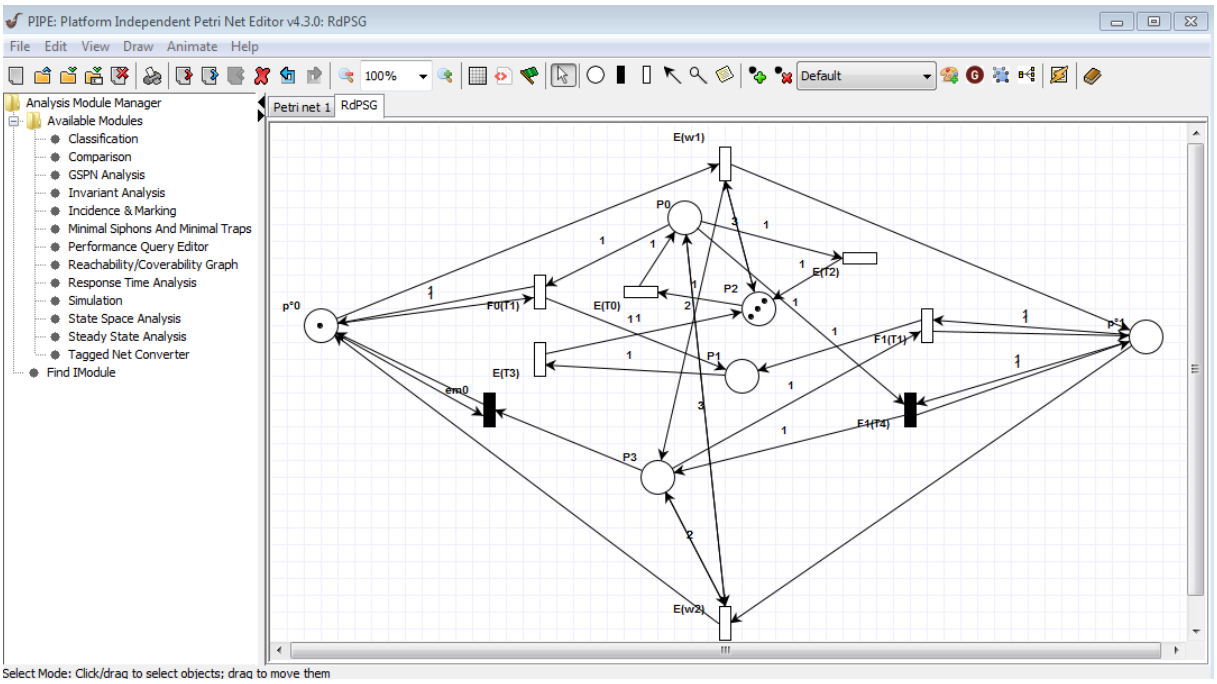

Figure 22.Proposition des positions des places et des transitions pour notre résultat

D'autre part, on constate que lorsqu'on appuie sur le bouton Save, un fichier est enregistré au format PNML, on retrouve dans ce fichier les deux configurations sélectionnées et on retrouve la règle de transformation entre elles.

3.4 Description

Ici, nous donnons une description de base du fichier PNML sur lequel notre travail est axé.

3.4.1 Réseau de Petri

```
<net id="Net-One" type="P/T net">
<! -- identifier un réseau de Petri -->
……. </net>
```

3.4.2 Place

```
<place id="P0">
<! -- identifier la place et ses propriétés--->
…… </place>
```

3.4.3 Transition

```
<transition id="T0">
<! -- identifier la place et ses propriétés--->
</transition>
```

3.4.4 Arc

```
<arc id="P0 to T1" source="P0" target="T1">
<! -- identifier l'arc et ses propriétés--->
</arc>
```

3.5 Conclusion

Dans ce chapitre, nous avons présenté les outils que nous avons utilisés pour publier notre application, puis les résultats de notre implémentation en tant qu'ensemble d'interfaces utilisateur graphiques (IUG) et les étapes de réalisation d'un outil. Cet outil permet de transformer les RdPSGDs en RdPSGs et d'éditer le fichier de règles de reconfiguration. Chaque module est créé indépendamment des autres, où la structure pour enregistrer ces modules est PNML. Ce travail peut être amélioré en ajoutant l'analyse des propriétés.2

Conclusion Générale

Au cours des dernières années, on s'est intéressé de plus en plus aux systèmes d'événements discrets et aux systèmes reconfigurables, et afin de pouvoir modéliser ce type de systèmes, on a mis au point des réseaux de Petri de faible niveau et on a connu de nombreuses nouvelles extensions à partir desquelles on trouve des RdPSs (Haut niveau), qui fournissent la capacité de concevoir ces systèmes et d'analyser leurs propriétés.

En regardant ce qui précède, nous avons résolu le problème posé, qui est de développer un outil qui nous permet de transformer un RdPSGD en un RdPSG équivalent, afin de pouvoir vérifier le RdPSGD, en vérifier son RdPSG équivalent à travers l'une des vérifications et analyses outils à sa disposition, car ce dernier n'est pas disponible pour le RdPSGD.

Au cours de la réalisation du projet, nous avons appris à connaître :

1. Réseaux de Petri de haut niveau.
2. Exploitation de ces connaissances pour en faire un outil pour les RdPSGs.
3. 3. Programmation du langage JAVA.

Dans le cadre de nos travaux futurs, nous avons l'intention de nous concentrer sur d'autres questions qui restent à résoudre, notamment:

- Compléter l'outil en résolvant le problème de localisation des transitions et des places du réseau et en les positionnant automatiquement sans intervention de l'utilisateur.

Bibliographie

1. **Wing, E. M. Clarke and J. M.** *Formal Methods: State of the Art and Future Directions.* s.l. : In: ACM Comput. Surv, 28.4 (1996). pp. 626–643.

2. **S.Tigane.** *Reconfiguration in Stochastic Petri Nets.* BISKRA : Université Mohamed Khider, 2020.

3. **H, David R. and Alla.** *Du Grafcet aux réseaux de Petri.* Paris : Editions Hermès, 1992.

4. **Murata, T.** *Petri nets: Properties, analysis and applications.* s.l. : Proceedings of the IEEE 77.4, 1989. pp. 541–580.

5. **LABADI, Karim.** *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.* s.l. : Université de Technologie de Troyes, 2005.

6. **Jiacun, Wang.** *Timed Petri Nets: Theory and Application.* s.l. : Kluwer Academic Publishers, 1998.

7. **Ramchandani, C.** *Analysis of asynchronous concurrent systems by timed Petri nets.* s.l. : PhD thesis. Massachusetts Institute of Technology, 1973.

8. **S, Haddad.** *Une catégorie régulière de réseaux de Petri de haut niveau: définition, propriétés et réduction.* s.l. : Thèse de Doctorat, Université Paris VI, 1973.

9. **K, Jensen.** *Coloured Petri nets. In: High-Level Petri Nets: Theory and Application.* Berlin : Springer-Verlag, 1991.

10. **Jensen, K.** *Coloured Petri nets: basic concepts, analysis methods and practical use.* s.l. : Vol. 1. Springer Science & Business Media, 2013.

11. **Marsan, M. A.** *Modelling with Generalized Stochastic Petri Nets.* s.l. : New York, NY, USA: John Wiley & Sons, Inc, 1994. 1st.

12. **Eric Badouel, Javier Olivier.** *Reconfigurable Nets, a Class of High Level Petri Nets Supporting Dynamic Changes Within Workflow Sysems.* IRISA : s.n., 1998. p. 23.

13. **Oliver, Marisa Llorens and Javier.** *Structural and Dynamic Changes in Concurrent Systems: Reconfigurable Petri Nets.* SEPTEMBER 2004.

14. **Marwa KANSO, Pascal BERRUET.** *RMS : une évolution des systèmes manufacturiers.* 10 juil. 2010.

15. **Jha, Nand K.** *Handbook of Flexible Manufacturing Systems.* 2012.

16. **Valette, M. Silva and R.** *Petri nets and flexible manufacturing.* Berlin : Heidelberg: Springer Berlin Heidelberg, 2004. pp. 742–788.

17. **L. Recalde, M. Silva, J. Ezpeleta, and E. Teruel.** *Petri Nets and Manufacturing Systems: An Examples-Driven Tour.* Berlin : Heidelberg: Springer Berlin Heidelberg. pp. 742–788.

18. **F. Long, P. Zeiler, and B. Bertsche.** *Modelling the flexibility of production systems in Industry 4.0 for analysing their productivity and availability with high-level Petri nets.* s.l. : In: IFAC-PapersOnLine 50.1, 2017. pp. 5680–5687. 20th IFAC World Congress.

19. **E. Simon, J. Oyekan.** *Adapting Petri nets to DES: stochastic modelling of manufacturing systems.* s.l. : International Journal of Simulation Modelling 17.1, 2018. p. 5_17.

20. **Camilli, L. Capra and M.** *Towards Evolving Petri Nets: a Symmetric Nets-based Framework.* s.l. : IFAC-PapersOnLine 51.7, 2018. pp. 480–485.

21. **Kahloul, J. Padberg and L.** *Overview of Reconfigurable Petri Nets.* 2018. pp. 201–222. Graph Transformation, Specifications, and Nets.

22. **J., H. Ehrig and.** *Padbeoncurrency and Petri Nets: Advances in Petri Nets.* Berlin, Heidelberg: Springer Berlin Heidelberg : s.n., 2004. pp. 496–536.

23. **ATLI, Maen.** *Contributions à la synthèse de commande des systèmes à événements discrets : nouvelle modélisation des états interdits et application à un atelier flexible.* s.l. : Doctorat de l'Université de Lorraine, 2012. p. 17.

24. **Peterson, J. L.** *Petri Nets.* 1977. pp. 223–252.

25. —. *Petri Net Theory and the Modeling of Systems.* Upper Saddle River, NJ, USA: Prentice Hall PTR : s.n., 1998.

26. **Alessandro Giua, Carla Seatzu.** *Petri nets for the control of discrete event systems.* 2014.

27. **Miller, R.M. Karp and R.E.** *Parallel Program Schemata.* s.l. : Journal of Computer and system Sciences 3, 1969. pp. 147-195.

28. **Nielsen, J. Esparza and M.** *Decidability issues for Petri nets.* s.l. : In: Petri nets newsletter 94, 1994. pp. 5–23.

29. **M.H.T.Hack.** *Decidability Questions for Petri Nets.* 1976. Ph. D. Thesis, M.I.T.

30. **Redouane, M. KARA.** *Matérialisation d'une commande pour un atelier à contraintes de temps de séjour.* 24 / 09 /2013.

31. **KARA, Redouane.** *Contribution à l'Analyse Quantitative et à la Commande des Réseaux de Petri continus à Arcs Valués. Application aux Systèmes de Production Manufacturière.* UMMTO 2009.

32. **Valk, C. Girault and R.** *Petri Nets for System Engineering: A Guide to Modeling, Verification, and Applications.* Berlin : Heidelberg: Springer-Verlag, 2001.

33. **Kritzinger, F. Bause and P. S.** *Stochastic Petri Nets: An Introduction to the Theory.* s.l. : Vieweg+Teubner, 2002.

34. **Mark A. Pinsky, Samuel Karlin.** *An Introduction to Stochastic Modeling (Fourth Edition).* 2011. pp. 79-163.

35. **Chiola, Giovanni.** *Galized Stochastic Petri Nets: A Definition at the Net Level and Its Implications.* 1993. pp. 89-90.

36. **G. Balbo, G. Chiola.** *Generalized stochastic Petri nets for the performance evaluation of fms.* s.l. : in Proc, 1987. pp. 1013-1018.

37. **G. Balbo, G. Chiola, S. C.** *An example of modelling and evaluation of a concurrent program using coloured stochastic Petri nets: Lamport's fast mutual exclusion algorithm.* Mar. 1992. pp. 221-240. vol. 3, no. 2.

38. **ROUBECHE, Asma.** *Modélisation et étude de performances d'une approche coopérative de gestion des ressources d'un réseau sans fil.* 2012. Mémoire de Master.

39. **IKHLEF, L.** *Evaluation des Performances de Réseaux de Files d'Attente via les GSPN.*

40. **Rozenberg, G.** *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 1: Foundations.* s.l. : World Scientific, 1997.

41. **H. Ehrig, G. Engels.** *Handbook of Graph Grammars and Computing by Graph Transformation.* October 1999.

42. **L Kahloul, A Chaoui.** *Using high level nets for the design of reconfigurable manufacturing systems.* 2014. 1161:1–19.

43. **D, Nolte.** *A Tutorial on Graph Transformation.In Graph Transformation, Specifications, and Nets.* s.l. : Springer, Cham, 2018. pp. 83–104.

44. **Samir Tigane, Souheib Baarir.** *Dynamic GSPNs: formal definition, transformation towards.* 2020. pp. 164-171.

45. **Laid Kahloul, S.Tigane.** *Technical report:A prototype for reconfigurable GSPNs.*

46. **Bourekkache S. Djouani K. Kahloul, L.** *Designing reconfigurable manufacturing systems using reconfigurable object Petri nets.* s.l. : International Journal of Computer Integrated Manufacturing, 1161:1–19, 2014.