**RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE**

**MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE**

**Université Mohamed Khider –BISKRA**

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

**Département d'Informatique**

N° d'ordre :…….

Serié :……..

Projet

Présenté pour le diplôme de

Master en informatique

Option: Génie logiciel et systèmes distribués

**Titre du projet :**

# Une approche de transformation de graphe pour l'édition et la simulation des RDPs

**Présenté le** : \ \ \

**par :** sakri radhia

**Composition du Jury :**

**M.** Guerrouf Fayçal **Superviseur**

**M** **Examinateur**

**M** **Examinateur BISKRA**

# *Dédicace*

*À mes parents…*

*Et toute ma famille*

# Remerciements

Mes remerciements à M. Guerrouf Fayçal qui a dirigé ce travail avec un très grand talent de professionnel. Je le remercie aussi pour sa patience et sa bienveillance durant toute cette année de mémoire.

Au terme de ce travail, Nous tenons à exprimer nos profonde gratitude ainsi que mes sincères remerciements aux membres de jury d'avoir accepté de juger notre travail.

# Résumé

Un réseau de pétri est un outil graphique pour la description formelle des systèmes dont la dynamique est caractérisée par la concurrence, la synchronisation, l'exclusion mutuelle et le conflit des choix multiples. La facilité d'adaptation des réseaux de pétri élargit leur champ de pratique jusqu'aux protocoles de communication, architectures des ordinateurs, etc. Leur aspect mathématique leur permet l'analyse des propriétés comportementales et corrigé les défauts de conception

Dans ce travail, nous avons présenté une conception et une implémentation d'un outil d'édition et de simulation des Rdps, l'idée de notre projet est basée sur une approche de transformation de graphe à l'aide de l'outil Atom3. Pour cela nous avons proposé : i) un méta-modèle de réseau de Petri ordinaire pour représenter la partie statique; ii) un ensemble de règles pour représenter la partie dynamique .

# Abstract

A petri net is a graphical tool for the formal description of systems whose dynamics are characterized by competition, synchronization, mutual exclusion and multiple choice conflict. The ease of adaptation of petri nets broadens their field of practice to include communication protocols, computer architectures, etc. Their mathematical aspect allows them to analyze behavioral properties and correct design flaws

In this work, we presented a design and implementation of a tool for editing and simulation of Rdps, the idea of our project is based on a graph transformation approach using the Atom3 tool. For this we have proposed: i) an ordinary Petri net meta-model to represent the static part; ii) a set of rules to represent the dynamic part.

# ملـــــــــــــــخص

شبكة بتري هي أداة رسومية للوصف الرسمي للأنظمة التي تتميز ديناميكياتها بالمنافسة والتزامن والاستبعاد المتبادل وتعارض الخيارات المتعددة .إن سهولة تكييف شبكات البتري توسع مجال ممارستهم ليشمل بروتوكولات الاتصال وهياكل الكمبيوتر وما إلى ذلك .يسمح لهم جانبهم الرياضي بتحليل الخصائص السلوكية وتصحيح عيوب التصميم.

في هذا العمل ، قدمنا تصميمًا وتنفيذًا لأداة لتحرير ومحاكاة شبكة بتري ، وتستند فكرة مشروعنا على نهج تحويل الرسم البياني باستخدام أداة .Atom3 لهذا اقترحنا: 1) نموذج méta-modèle لشبكة بيتري العادي لتمثيل الجزء الثابت

ب) مجموعة من القواعد لتمثيل الجزء الديناميكي.

# Table des Matières

# Table de Figures

## Chapitre 01 Les Réseaux de petri



## Chapitre 02 Transformation de graphes et ATOM³ .

**Chapitre 03 Conception et L'implémentation.**

# Introduction

Les systèmes logiciels sont présents, dans tous les domaines de l'activité humaine (industrie, transports, communications, construction, etc.). Avec le temps, ces systèmes sont devenus de plus en plus complexes. Aujourd'hui, le but de chaque domaine de la science de l'ingénieur est d'utiliser le langage de modélisation formel pour décrire, analyser ou diagnostiquer les systèmes, afin de permettre la fiabilité des systèmes.

Dans la discipline d'ingénierie de systèmes, cette complexité augmente à un rythme rapide, et devient de plus en plus incontrôlable. En raison de contraintes de temps et d'efforts, il n'est plus possible de suivre le cycle de conception qui consiste à faire des essais et de corriger les erreurs sur des prototypes et par conséquent, leur analyse et leur vérification représentent un enjeu capital . Au lieu de cela, la tendance des modèles d'aujourd'hui s'oriente plus vers la simulation, et dans certains cas vers une spécification formelle de conception, de sorte que les défauts de conception peuvent être détectés avant même qu'un prototype soit construit. C'est ici que les réseaux de Petri trouvent tout leur intérêt et sont de plus en plus utilisés

ils représenté un outil mathématique puisant dans le domaine de la modélisation et de la vérification des systèmes .en plus de leur force d'analyse il offre une représentation graphique simple qui aide à la modélisation des systèmes complexe , Donc notre approche est basée sur la transformation de graphes (l'approche algébrique **DPO**) à l'aide de l'outil AToM3.

Le but de notre projet est de fournir un outil d'étudier et de simulation avec les réseaux Ordinaire.

Notre document est organisé comme suit:
Le premier chapitre est décrit et donne les concepts de base des réseaux de Petri , et le deuxième chapitre nous présentons des définitions de modèle et un méta-modèle et bien sur un langage de modélisation et la relation entre eux, ensuite nous avons voir la transformation de modèles et la

classification des ses approches. Le troisième chapitre combine entre la conception de notre démarche et aussi illustre et explique notre démarche de transformation. Enfin, ce document se termine par une conclusion générale qui est une collection de l'idée principale de ce document.

# Chapitre 1

# Les réseaux de pétri

# Chapitre 1

# Les réseaux de pétri

## 1. Introduction :

Les Réseau de Petri représenté un outil mathématique puisant dans le domaine de la modélisation et de la vérification des systèmes .en plus de leur force d'analyse il offre une représentation graphique simple qui aide à la modélisation des systèmes complexe

Donc Le modèle des réseaux de Petri est un outil graphique de modélisation et d'analyse des systèmes parfaitement adapté à l'étude des structures de contrôle. Il permet notamment de maîtriser et d'assurer la sûreté de fonctionnement de logiciels complexes (aéronautique, transports, industrie...).

Le formalisme formel des Réseau de Petri (RDP), adapté à la prise en compte des problèmes de concurrence, de synchronisme et de parallélisme, constitue un excellent outil de spécification fonctionnelle d'un problème et de mise en évidence des contraintes. Les propriétés mathématiques qui découlent de l'analyse des RDPs permettent une étude comportementale et structurelle essentielle à la validation d'une spécification. Les possibilités de simulation offertes par les outils informatiques supportant le formalisme contribuent également à cette validation.

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

## 2. Qu'est-ce que les réseaux de Petri :

Les réseaux de Petri sont définis comme étant un formalisme proposées pour spécifier, analyser et vérifier du comportement des systèmes discrets, particulièrement les systèmes concurrents, parallèles, introduit par Carl Adem Petri dans sa thèse "Communication avec des Automates" à l'université Darmstadt en Allemagne en 1962. Les définitions concernant les réseaux de Petri portées sur deux aspects : [1]

- **Un aspect structurel** : Quelles sont les actions, quels sont les sites, quelles sont les conditions pour qu'une action soit possible et quelles sont les conséquences d'une action?
- **Un aspect comportemental** : Comment représenter le fonctionnement d'un réseau de Petri? c.-à-d. ce qui se passe quand une action ou plusieurs actions sont exécutées.

## 2.1 L'aspect structurel :

### 2.1.1 Définition d'un réseau de Petri :

Un réseau de Petri **(R)** est un triple **R= (P, T, W)** où **P** est l'ensemble des places (les places représentent les sites) et **T** l'ensemble des transitions (les transitions représentent les actions) tel que $\mathbf{P}\cap\mathbf{T}=\phi$ et **W** est la fonction définissant le poids porté par les arcs tel que $\mathbf{W}: \mathbf{((P\times T)\cup(T\times P))} \rightarrow \mathbf{N}= \{0, 1, 2,...\}$.

Le réseau **R** est fini si l'ensemble des places et des transitions est fini c.-à-d. $|\ \mathbf{P}\cup\mathbf{T}\ |\in\mathbf{N}$.

Un réseau **R= (P, T, W)** est ordinaire si pour toute $(x,y)\in ((\mathbf{P}\times\mathbf{T})\cup(\mathbf{T}\times\mathbf{P}))$: $\mathbf{W}(x,y) \leq \mathbf{1}$ .

Dans un réseau ordinaire la fonction **W** est remplacée par **F** où :

$\mathbf{F}\subseteq((\mathbf{P}\times\mathbf{T})\cup(\mathbf{T}\times\mathbf{P}))$ tel que $(x,y)\in F\Leftrightarrow \mathbf{W}(x,y) \neq 0$ [2].

Pour chaque $x\in P\cup T$:

***x** représente l'ensemble des entrées de x:

$*x=\{y\in\mathbf{P}\cup\mathbf{T}/\ \mathbf{W}(y,x)\neq 0\}$

[1] Christophe Reutenauer.Aspects mathématiques des réseaux de Petri, volume 36.Masson Paris, 1989

[2] Murata ,Petri nets: « Properties, analysis and applications », Murata, Tadao,Proceedings of the IEEE,1989

**x*** représente l'ensemble des sorties de x:

$$x^* = \{ y \in \mathbf{P} \cup \mathbf{T} / \mathbf{W}(x,y) \neq 0 \}$$

**Remarque**: si ***x**=ϕ, x est dite source, si **x***=ϕ, x est dite puits.

### 2.1.2 Représentation d'un réseau de Petri :

**a. Représentation graphique :** L'un des aspects les plus agréables des réseaux de Petri est qu'il est extrêmement aisé de les visualiser; c.-à-d., donner une interprétation graphique à sa structure qui peut être représentée à travers un **graphe** bipartie fait de deux types de sommets: les **places** et les **transitions** reliées alternativement par des **arcs orientés** qui portent des poids entier positifs, si un poids n'est pas porté alors il est égal à **1** ( **RdP ordinaire**). Généralement, les places sont représentées par des cercles et les transitions par des rectangles, le marquage d'un **RdP** est représenté par la distribution de jetons dans l'ensemble de ses places telle que chaque place peut contenir un ou plusieurs jetons représentés par des points dans le cercle représentant la place . [3].

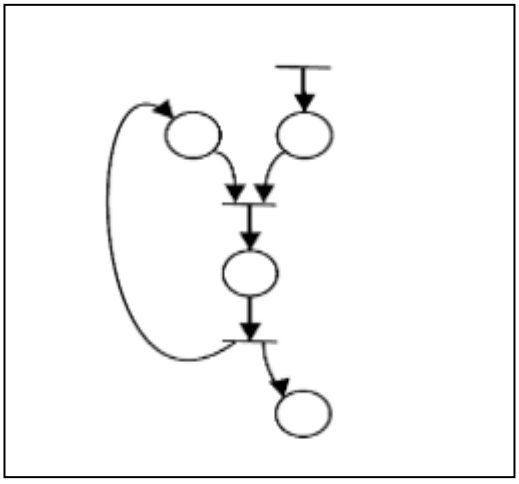


Figure 1.1 : Réseau de Petri simple

[3] Même référence [2]

**b. Représentation matricielle:** Une représentation matricielle d'un RdP est offerte afin de simplifier les Tâches d'analyse et de vérification effectuée sur un modèle RdP. Agir sur une représentation graphique d'un modèle RdP est une Tâche délicate en comparant avec une représentation matricielle.

Il est possible de représenter la fonction W (fonction de poids) par des matrices.

**Définition :**

Soit Un réseau de Petri **R=(P, T, W)** avec **P={p1, p2, ...,pm}** et **T={t1, t2, ...,tn}**, on appelle matrice des pré conditions ***pré*** la matrice **m ×n** à coefficients dans **N** tel que ***pré*****(i,j)= W($p_i$ , $t_j$),** elle indique le nombre de marque que doit contenir la place $p_i$ pour que la transition $t_j$ devienne franchissable, de la même manière on définit la matrice des post conditions ***post*** la matrice **n ×m** tel que ***post*****(i,j)= W($t_j$, $p_i$)** contient le nombre de marques déposées dans $p_i$ lors du franchissement de la transition $t_j$. La matrice **C= *post* –*pré*** est appelée matrice d'incidence du réseau (**m** représente le nombre de places d'un réseau de Petri et **n** le nombre de transitions). [4] .

La représentation Graphique d'un marquage dans un **RdP** marquée et présenté par des marques dans la place ces marques sont dites jetons. Le marquage d'un réseau de Petri est représenté par un vecteur de dimension **m** à coefficients dans **N**. La règle de franchissement d'un réseau de Petri est définie par **M' (p) =M (p) + C (p, t).**

[4] Charles C Poirier and Stephen E Reiter.La Supply Chain: Optimiser la chaînelogistique et le réseau interentreprises. Dunod, 2001.

**Exemple :**

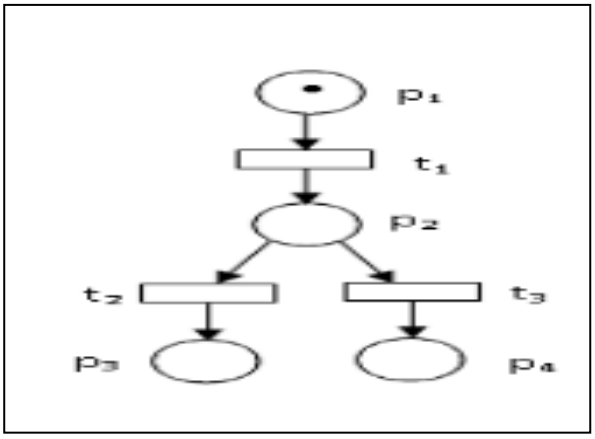



**Figure 1.2 : Réseau de Petri marqué**

$P = \{p_1, p_2, p_3, p_4\}$ $T = \{t_1, t_2, t_3\}$

$$Pré = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Post = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La matrice d'incidence est :

$$C = \begin{bmatrix} -1 & 0 & 0 \\ 1 & -1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Le vecteur de marquage $M$ est :

$$M = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

**Figure 1.3: Représentation matricielle d'un Réseau de petri**

On appelle marquage initial, noté $M_0$ , le marquage à l' instant initial (**t** = 0).

## 2.2 L'aspect comportemental :

Le comportement d'un réseau de Petri est déterminé par sa structure et par son état. Pour exprimer l'état d'un réseau de Petri, les places peuvent contenir des jetons qui ne sont que de simples marqueurs.

**2.2.1 L'état dans un réseau de Petri :** Dans la théorie des réseaux de Petri, l'état d'un réseau est souvent appelé marquage du réseau qui est définit par la distribution des jetons sur les places. Le marquage d'un réseau de Petri **R=(P, T,W)** est défini par la fonction de marquage **M** : **P→N**. Un réseau de Petri marqué est dénoté par **∑=(P, T, W,$M_0$)** où **$M_0$** est le marquage initial. Le comportement d'un réseau de Petri marqué est déterminé par ce qu'on appelle règle de franchissement.[5]

### 2.2.2 Franchissement d'une transition :

Une règle de franchissement est une simple relation de transition qui définit le changement d'état dans un réseau marqué lors de l'exécution d'une action. Afin de définir une règle de franchissement, il est nécessaire de formaliser quand le réseau peut exécuter une action: on dit qu'une transition peut être **franchie** à partir d'un marquage **M** (qui représente l'état du system à un instant donné) si et seulement si chaque place d'entrée **p∈*t** de la transition **t** contient au moins un nombre de jetons qui est supérieur ou égal au poids de l'arc reliant cette place d'entrée **p** avec la transition **t** tel que: **M(p) ≥W(p, t) ∀p ∈P**. [6]

### 2.2.3. L'exécution d'un réseau de Petri :

#### a) Exécution séquentielle :

- **Séquence de franchissement :** Une séquence de franchissement **« s »** est une suite de transitions **(t1, t2, ..., tn)** qui permet, à partir d'un marquage « **M** », de passer au marquage « **M'** » par le franchissement successif des transitions définissant la séquence

[5] R Bourdais, B Trouillet, and P Yim. Étude de noeuds ferroviaires à l'aide des réseauxde petri temporisés stochastiques. InProc. Conference Internationale Francophoned'Automatique, CIFA, 2004.
[6] R Valk. Object petri nets: Using the nets-within-nets paradigm, advanced course onpetri nets 2003 (j. desel, w. reisig, g. rozenberg, eds.), 3098.Appendix A: Proof ofTheorem, 3, 2003.

- **Marquage accessible :** Le marquage d'un Réseau de Petri à un instant donné est une vectrice colonne dont la valeur de la **I** ème composante est le nombre de marques dans la place $\mathbf{P_i}$ à cet instant. Le franchissement d'une transition conduit à un changement du marquage. Le passage du marquage $\mathbf{M_k}$ au marquage $\mathbf{M_l}$ par franchissement de la transition $\mathbf{T_j}$ est noté : $\mathbf{M_k(T_j> M_l}$. Le nombre de marques dans la place $\mathbf{P_i}$ pour le marquage $\mathbf{M_k}$ est noté $\mathbf{M_k(P_i)}$. A partir d'un Même marquage, il peut être possible de franchir plusieurs transitions, menant ainsi à des marquages différents. L'ensemble des marquages accessibles à partir du marquage $\mathbf{M_0}$ est l'ensemble des marquages obtenus à partir de $\mathbf{M_0}$ par franchissements successifs d'une ou plusieurs transition(**s**). Cet ensemble est noté $\boldsymbol{A}\mathbf{(R ; M_0)}$.[7]
- **Graphe de marquage :** On peut représenter l'ensemble de marquage accessible par un graphe si ce dernier et fini. Le graphe de marquage a comme sommet l'ensemble de marquage accessible $\boldsymbol{A}\mathbf{(R,M_0)}$ Un arc orienté relie deux sommet $\mathbf{M_i}$ et $\mathbf{M_j}$ s'il existe une transition **t** franchissable permettant de passer d'un marquage à un autre $\mathbf{M_i[t\rangle M_j}$ Les arcs du graphe sont étiqueté par les transitions correspondantes.

**Exmple :**

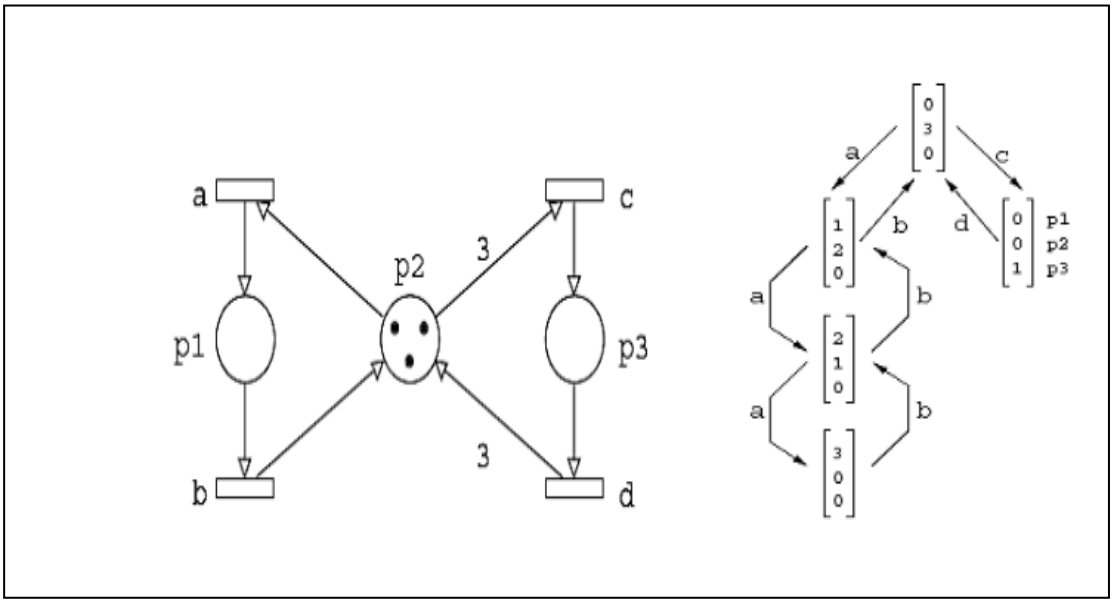



**Figure 1.4 : Graphe de Marquage**

[7] Même référence [2]

- **L'exécution séquentielle d'un réseau de Petri :**

L'exécution séquentielle d'un réseau de Petri est définie en termes d'un ensemble de séquences d'occurrence. Une séquence d'occurrence est une séquence de transitions franchissables dénotée par $\sigma$=**$M_0$ t1 $M_1$ t2**...tel que **Mi-1[ti⟩Mi**. Une séquence **t1 t2**... est une séquence de transitions (commencée par le marquage **M**) si et seulement si il existe une séquence d'occurrence **$M_0$ t1 $M_1$**... avec **M=$M_0$**. Si la séquence finie **t1 t2...tn** conduit à un nouveau marquage **M'** à partir du marquage **M**, on écrit **M[t1 t2...tn⟩M'**ou simplement **M[t1t2...tn⟩** si on ne veut pas spécifier le marquage résultat. [8].

## b) Exécution concurrente :

Une exécution concurrente d'un réseau de Petri est une exécution dans laquelle plusieurs transitions peuvent se franchir au même temps, elle est souvent déterminée par la notion de processus. Ceci permet de donner une interprétation de la concurrence dans un réseau de Petri selon la sémantique basée sur la vraie concurrence (sémantique d'ordre partiel) qui est interprétée dans la théorie des réseaux de Petri par un type spécial de réseaux appelés réseaux d'occurrences.

# 3. Propriétés des RdP :

**3.1 Caractère borné :** Cette propriété définit et caractérise la possibilité pour une place d'accumuler une quantité bornée ou pas de jetons au cours de l'évolution d'un réseau.

## Place k-bornée, non bornée :

Pour un réseau **R** et un marquage ***Mo*** , une place « **p** » du réseau marqué **(R,*Mo* )** est k-bornée si pour tout marquage **M** accessible depuis ***Mo*** , **M(p) ≤ k**. Dans le cas contraire

[8] Même référence [2]

la place « **p** » est dite **non-bornée**. Autrement dit : **p** est **k-bornée** $\Leftrightarrow \forall \mathbf{M} \in \mathbf{A(R,}Mo\mathbf{)}$, $\mathbf{M(p)} \leq \mathbf{k}$

## Réseau borné :

Un réseau marqué est **borné** si toutes ses places sont bornées. Les réseaux **1-bornés** sont appelés des réseaux **saufs**.

**Exemple** :

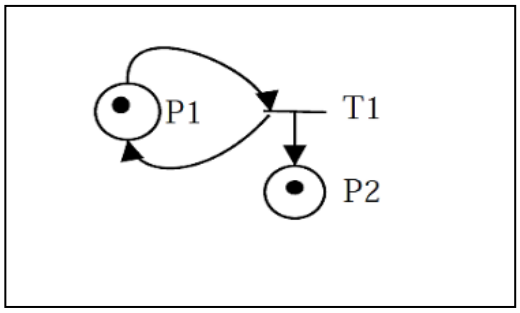



**Figure 1.5 : RdP non borné**

## 3.2 Activité d'un réseau :

La notion d'activité d'un réseau recouvre deux classes de définitions. La première concerne l'activité individuelle des transitions, la seconde concerne l'activité globale d'un réseau (indépendamment de transitions particulières).

## a. Pseudo-vivacité

Un réseau de Petri **(R,*Mo*)** est dit **pseudo-vivant** si pour tout marquage accessible depuis le marquage initial, il existe toujours une transition « **t** » qui puisse être franchie. Autrement dit : $\forall \mathbf{M} \in \mathbf{A(R,}Mo\mathbf{)}$, $\exists \mathbf{t} \in \mathbf{T}$ tel que $\mathbf{m[t\rangle}$.

## b. Quasi-vivacité :

### Quasi-vivacité d'une transition :

La quasi-vivacité d'une transition signifie que depuis le marquage initial, cette transition peut être franchie au moins une fois. Autrement dit, pour un réseau marqué

**(R,*Mo* ) t ∈T** est **quasi-vivante** ⇔ ∃**M∈ A(R,*Mo* )** tel que **M[t⟩**

Conséquemment, une transition qui n'est pas quasi-vivante est inutile!

### Quasi-vivacité d'un réseau

Un réseau est quasi-vivant si toutes ses transitions le sont.

## C. Vivacité :

### Vivacité d'une transition

La vivacité d'une transition exprime le fait que quelque soit l'évolution du réseau à partir du marquage initial, le franchissement à terme de cette transition est toujours possible.Autrement dit, pour un réseau marqué **(R,M o )**

**t ∈T** est **vivante** ⇔ : ∀**M∈A(R,*Mo* )**, **t** est **quasi-vivante** pour **M**.

### Vivacité d'un réseau

Un réseau est vivant si toutes ses transitions le sont. Autrement dit, le réseau de Petri **(R,Mo)** est **vivant** si, pour tout marquage **M ∈ A(R,Mo)**, le réseau **(R,M)** est **quasi-vivant**, c'est-à-dire :

∀**M∈A(R,*Mo* ),** ∀**t∈T**, ∃ **M'∈A(R,M)** tel que **M'[t⟩.**

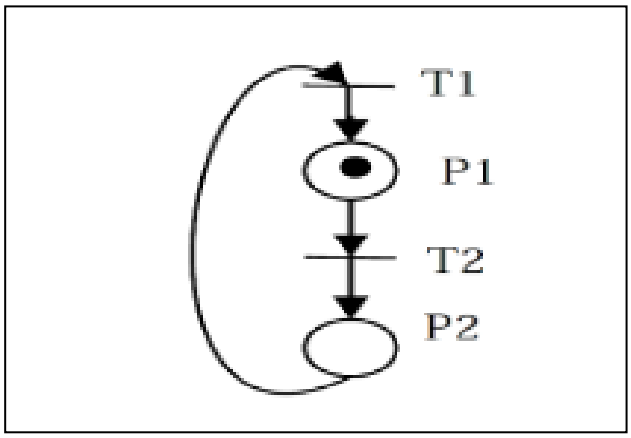



Figure 1.6 : RdP vivant

## d. État d'accueil et réversibilité :

### État d'accueil

Un RdP possède un état d'accueil $\mathbf{M_a}$ pour un marquage initial **Mo** si pour tout marquage accessible $\mathbf{M_i}$ il existe une séquence « **s** » telle que $\mathbf{M_i}[s\rangle\mathbf{M_a}$ . Autrement dit :

$\forall \mathbf{M} \in \mathbf{A(R,}\mathit{Mo}\mathbf{)}, \exists s \in \mathbf{T}^*$ tel que $\mathbf{M}[s\rangle\mathbf{M_a}$

### RdP réinitialisable (ou réversible)

Un RdP est réinitialisable (ou réversible) pour un marquage initial **Mo** si **Mo** est un **état d'accueil**.

## e. Absence de blocage :

Cette propriété est plus faible que celle de vivacité. Elle implique seulement que le réseau a toujours la possibilité d'évoluer.

## 4. Méthodes d'analyse des réseaux de Petri :

La modélisation d'un système n'est utile que si elle permet d'analyser ses propriétés. La théorie des réseaux de Petri offre des techniques d'analyse puissantes pour analyser les propriétés d'un réseau de Petri. Parmi ces techniques nous pouvons citer :

### 4.1 Analyse par graphes des marquages :

Pour commencer l' analyse les propriétés d'un réseau de Petri il doit premièrement construire son graphe des marquages accessibles. Dans ce graphe, chaque sommet représente un marquage accessible et chaque arc représente franchissement d'une transition permettant de passer d'un marquage à un autre. Nous pouvons distinguer entre deux situations possibles :

#### 4.1.1. Le graphe est fini :

C'est la situation la plus favorable car dans ce cas toutes les propriétés peuvent être déduites simplement par inspection de celui-ci.

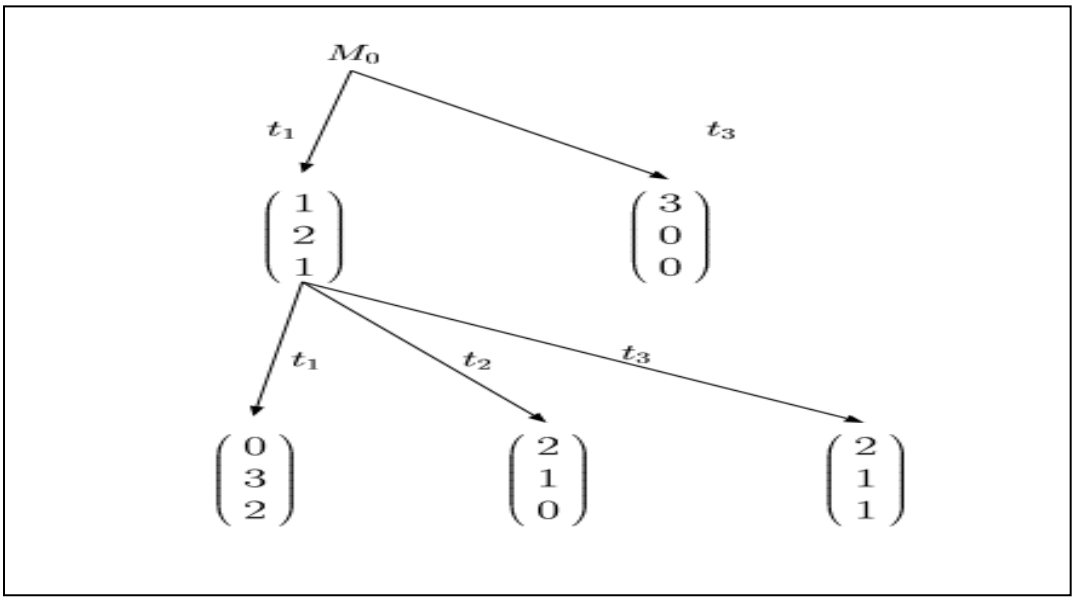



Figure 1.7 : Arbre des marquages atteignables

### 4.1.2. Le graphe est infini :

Dans ce cas le réseau n'est pas borné, on construit un autre graphe appelé "graphe de couverture" permettant de déduire certaines propriétés.

Dans l'arbre infini, le nombre de jetons dans les places pouvant avoir une quantité grande de jetons peut être considéré comme une valeur, noté par ω. Il a la propriété que pour chaque entier n,

ω + n = ω

ω - n = ω

n<= ω

ω >= ω

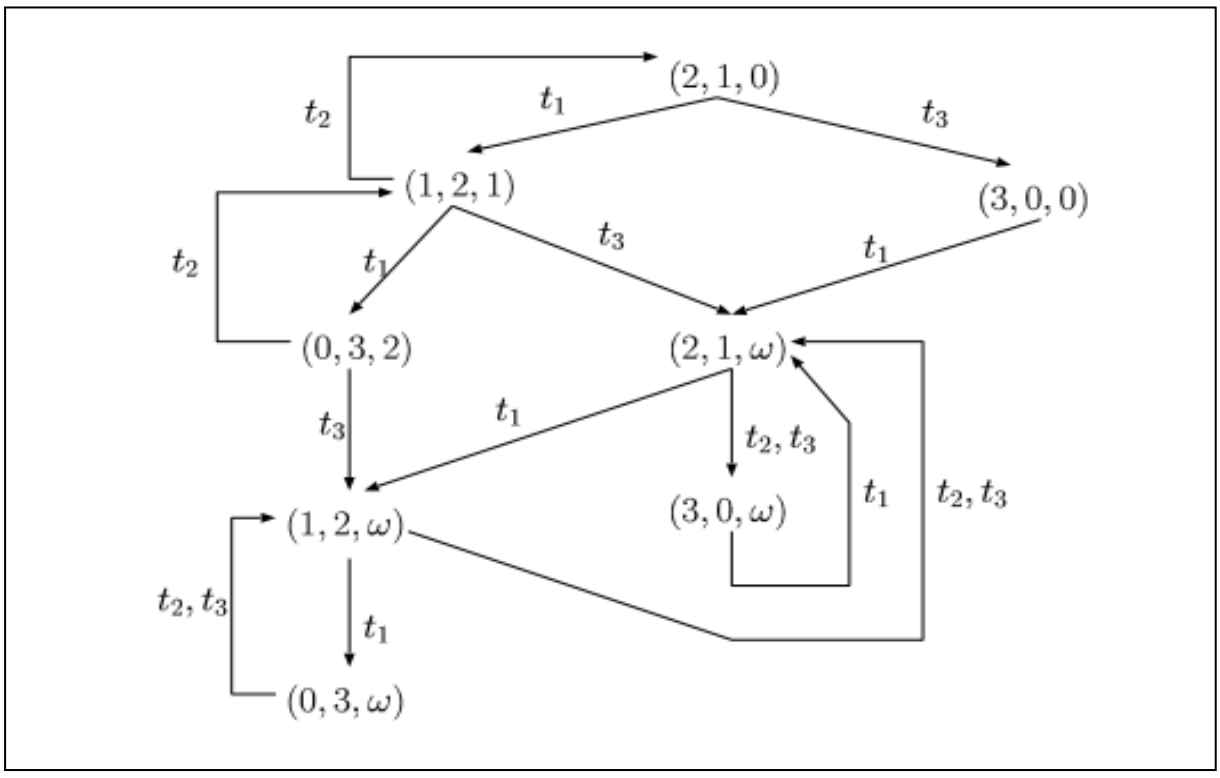



Figure 1.8 : Arbre des marquages atteignables

**4.2 Analyse par algèbre linéaire :** Par cette méthode nous pouvons étudier les propriétés (caractère borné, vivacité) d'un réseau de Petri indépendamment du marquage initial. Nous parlons alors de propriétés structurelles du réseau. Cette approche nous donne un outil pour vérifier l'accessibilité. Si un marquage M ' est accessible à partir d'un marquage M, donc il existe une séquence **σ** du franchissement de transition qui se produit à M ' , et ça interprété par l'équation d'état : M ' = M + x.D

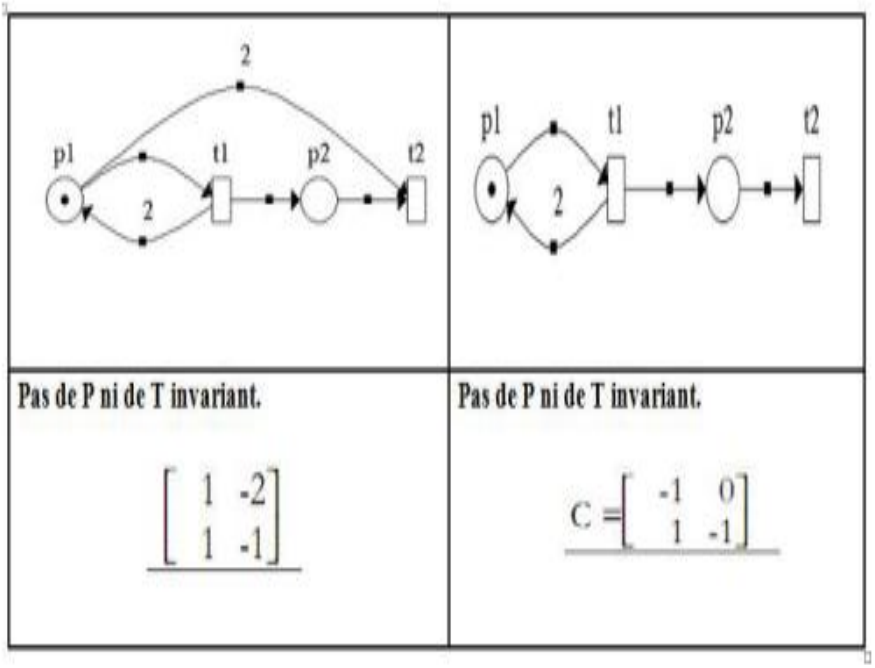



Figure 1.9 : .exemple Analyse par algèbre linéaire

## 4.3 Analyse par réduction :

Dans un Rdp de taille grande/ importante , la vérification de ses propriétés peut être devient difficile par l'utilisation du graphe de marquages [9]. Pour ce cas on a présenté des règles de réduction permettant de transférer d'un RdP marqué, un RdP marqué plus simple, par la réduction de nombre de places et le nombre de transitions .on peut cites les 4 règles importantes :

[9] Gérard Scorletti and G Binet. Réseaux de pétri.cours EL401T2, master 1A mention,2006

## R1 (Place substituable) :

Cas d'une place n'ayant q'une transition d'entrée et une de sortie.

Soit $t_e$ la transition d'entré a **p** et $t_s$ la transition de sortie de **p** On doit avoir :

1) Post(**p**,$t_e$ )=Pre(**p**,$t_s$ ) (Post(**p**,$t_e$ ) ≠0)
2) ∀**p'**∈**P** si **p'** ≠**p** alors Pre(**p'**,$t_s$ )=0 et (Pre(**p**,$t_s$ )≠0)

On fusionne $t_e$ avec $t_s$ pour obtenir $t_{es}$ et on efface **p** avec :

∀**p'**∈ **P** Pre(**p'**,$t_{es}$ )=Pre(**P'**,$t_e$ )

Post(**p'**,$t_{es}$ )=Post(**P'**,$t_e$ )+Post(**p'**,$t_s$ )

**Exemple** :

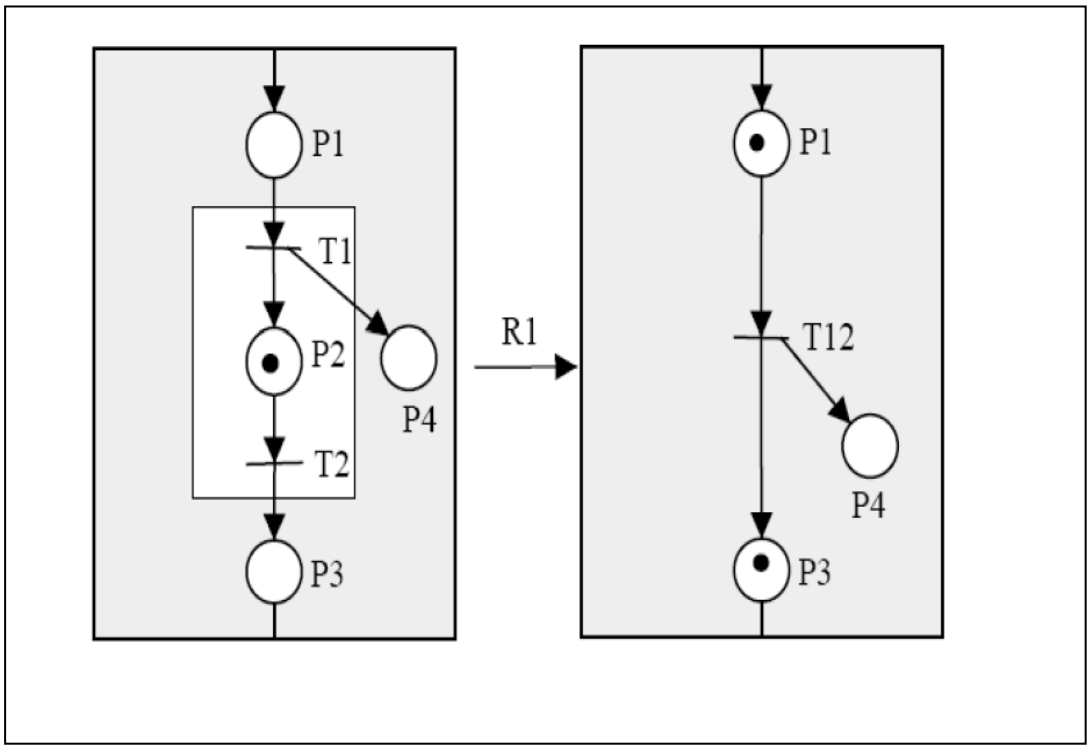



**Figure 1.10 : place avec une seule transition de sortie et une d'entré**

## R2(Place implicite) :

**Qu'est-ce que une place implicite :** Une place implicite est une place qui ne sert a rien, elle n'ajoute aucune condition supplémentaire de franchissement.

La réduction **R2** consiste à supprimer la place implicite avec ses arcs en entrée et en sortie

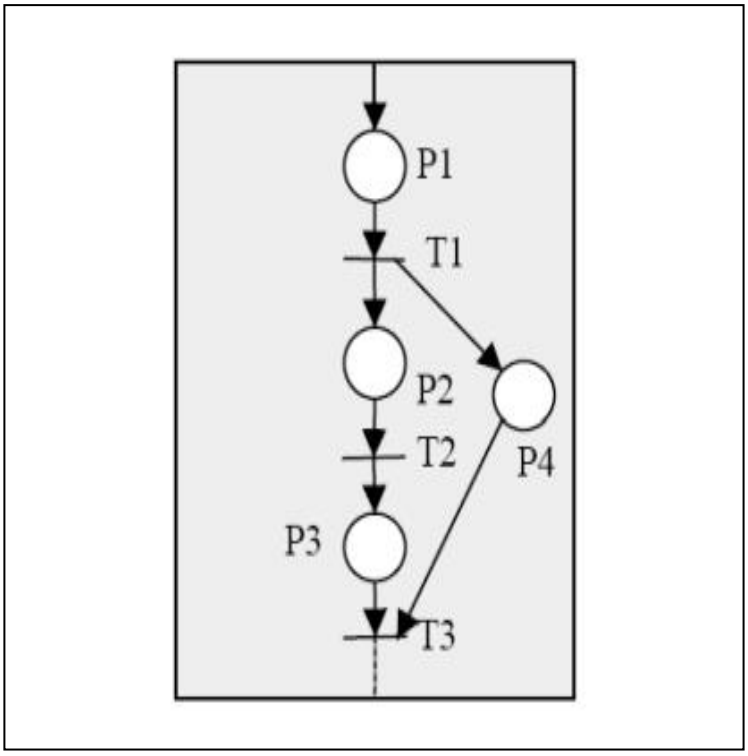



Figure 1.11 : exemple de place implicite (p4).

## R3(Transition neutre) :

Transition neutre : Une transition est neutre si l'ensemble de ces places d'entrées est égal à l'ensemble de ses places de sorties.

**Pre(.,t)=Post(.,t)**

**R3** admet à supprimer la transition neutre **Tj** avec l'ensemble de ses arcs en entrée et en sortie. Elle ne sera effectuée que s'il existe une transition **Ti ≠ Tj** telle que son franchissement mette suffisamment de jeton dans les places d'entrée de **Tj** pour la rendre franchissable.

**Exemple**

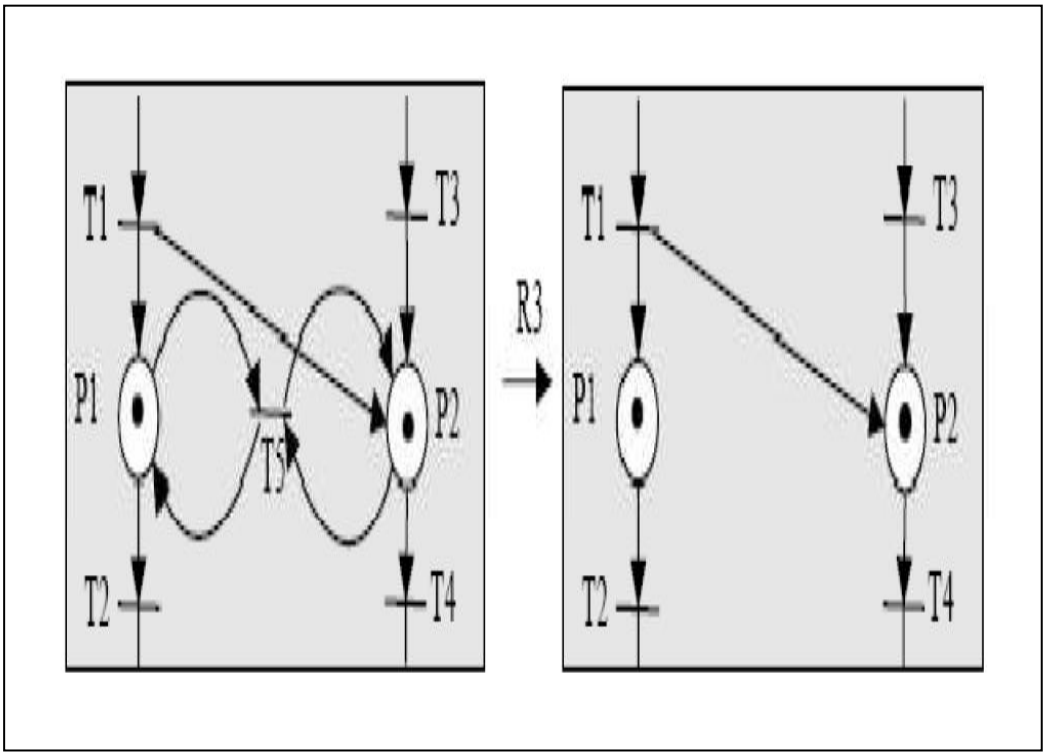



**Figure 1.12 : Réduction R3**

## R4(Transitions identiques) :

Deux transitions T1 et T2 sont identiques ssi

$$Pre(.,t\ 1)=pre(.,t\ 2)$$

$$post(.,t\ 1)=post(.,t\ 2)$$

**R4** supprime l'une des transitions identiques avec l'ensemble de ses entrées et sorties arcs en entrée et en sortie.

**Exemple**

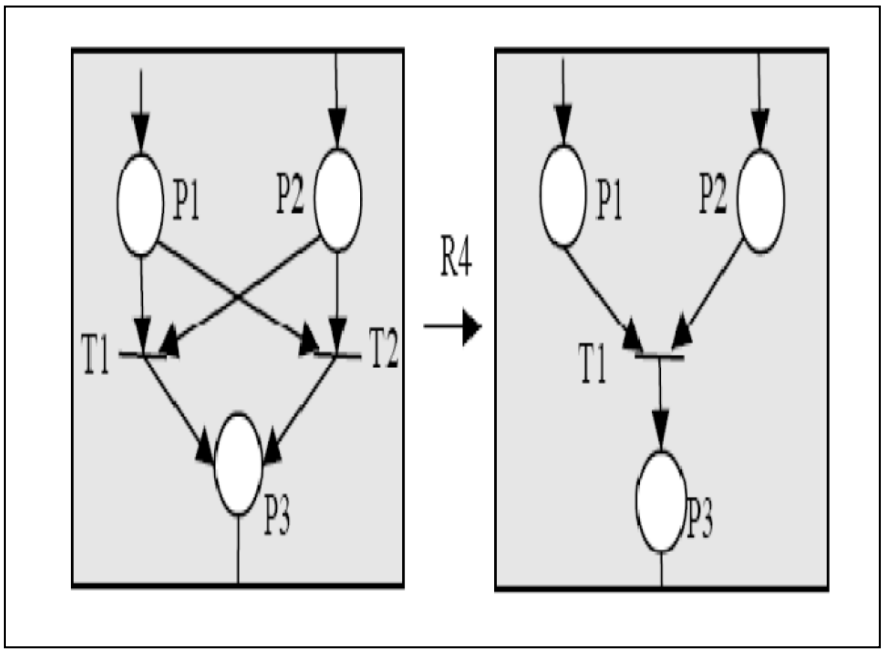



**Figure 1.13 : Réduction R4.**

Toutes ces règles doit être assurer :

1.équivalent au RdP marqué de départ (pour les propriétés étudiées).

2. suffisamment simple pour que l'analyse de ses propriétés soit simple.

## 5. Conclusion :

Dans ce chapitre nous avons donné les définitions de base d'un réseau de Petri. Bien que simples, ces définitions se compliquent lorsque l'on va au bout des choses.les Rdps sont les représentations mathématiques permettant la modélisation d'un système. Et l'analyse d'un réseau de Petri peut révéler des caractéristiques importantes du système concernant à sa structure et son comportement dynamique.

Les réseaux de Petri ( Place/Transition) ont une théorie basée sur une base mathématique solide, les résultats sont profonds. Les systèmes étant modélisé par les réseaux de Petri (Place/Transition) peuvent être évalué facilement grâce à ces résultats. Mais avec les applications industrielles, c'est très difficile à être modélisé par ces réseaux à cause des règles de

Franchissement trop simple, donc c'est trop compliqué, on doit peut-être utiliser des millier de places et transitions pour une petite application. Dans ce cas, les RdP haut niveau sont plus raisonnables.

Nous expliquerons dans le prochain chapitre la transformation des graphes et comment on l'utilise cette notion pour réaliser un simulateur des Rdps

# Chapitre 2

# Transformation de graphes et ATOM$^3$

# Chapitre 2

# Transformation de graphes et ATOM³

## 1. Introduction :

Au cours des dernières décennies, le développement de grands projets d'ingénierie, toujours plus complexes, a mis en évidence la nécessité de disposer d'outils, de méthodes et de processus permettant d'en assurer la maîtrise tout au long de leur cycle de vie.

L'Ingénierie Dirigée par les Modèles (IDM), ou Model Driven Engineering (MDE) est une discipline récente du génie logiciel qui met l'accent sur les modèles au sein du processus de développement logiciel. L'IDM est donc le domaine de l'informatique mettant à disposition des outils, concepts et langages pour créer et transformer des modèles. [10]

La transformation des modèles est une tâche complexe qui nécessite la disposition d'outils flexibles permettant la gestion des modèles et des langages durant la transformation et la manipulation des modèles. La sémantique de ces derniers doit être spécifiée, et les langages utilisés doivent être décrit de manière précise. [11]

Dans ce chapitre, nous donnons un rappel sur la transformation de modèle et la méta-modélisation. La transformation de graphe est présentée comme outil de transformation de modèles, et ATOM³ comme outil de transformation de graphe.

[10] WIKIPEDIA, l'encyclopédie libre, Home page: http://wikipedia.fr

[11] Samba Diaw, Redouane Lbath, and Bernard Coulette. État de l'art sur le développe-ment logiciel basé sur les transformations de modèles.Technique et Science Infor-matiques, 29(4-5):505–536, 2010

## 2. Modèle, Langage de modélisation et Méta-modèle :

**2.1 Un modèle** :est une abstraction et une simplification d'un système qu'il représente. Il offre donc une vision schématique d'un certains nombre d'éléments que l'on décrire sous la forme d'un ensemble de faits. Un modèle doit pouvoir être utilisé pour répondre à des questions que l'on se pose sur lui, exactement de la même façon que le système aurait répondu lui-même.

Le modèle doit se substituer au système pour permettre d'analyser de manière plus abstraite certaines de ses propriétés [12] .

**2.2 Un langage de modélisation** : est défini par une syntaxe abstraite, une syntaxe concrète et une sémantique. La syntaxe abstraite définit les concepts de base du langage. La syntaxe concrète définit le type de notation qui sera utilisé pour chaque concept abstrait qui peut être graphique, textuelle ou mixte. Enfin, la sémantique définit comment les concepts du langage doivent être interprétés par les concepteurs mais surtout par les machines.

**2.3 Un méta-modèle** : est un modèle qui définit le langage d'expression d'un modèle. Autrement dit, le méta-modèle représente (modélise) les entités d'un langage, leurs relations ainsi que leurs contraintes, c'est-à-dire une spécification de la syntaxe du langage

Le Méta-modèle à son tour est exprimé dans un langage de méta-modélisation spécifié par le Méta-Méta-modèle. Le langage utilisé au niveau du méta-méta-modèle doit être suffisamment puissant pour spécifier sa propre syntaxe abstraite

Un modèle est dit **conforme** à un méta-modèle et constitue une représentation d'un système existant ou imaginaire. Cette relation est illustrée dans la Figure suivante

[12] ML Minsky. Matter, mind and models: Semantic information processing, ed. marvinminsky, 1968

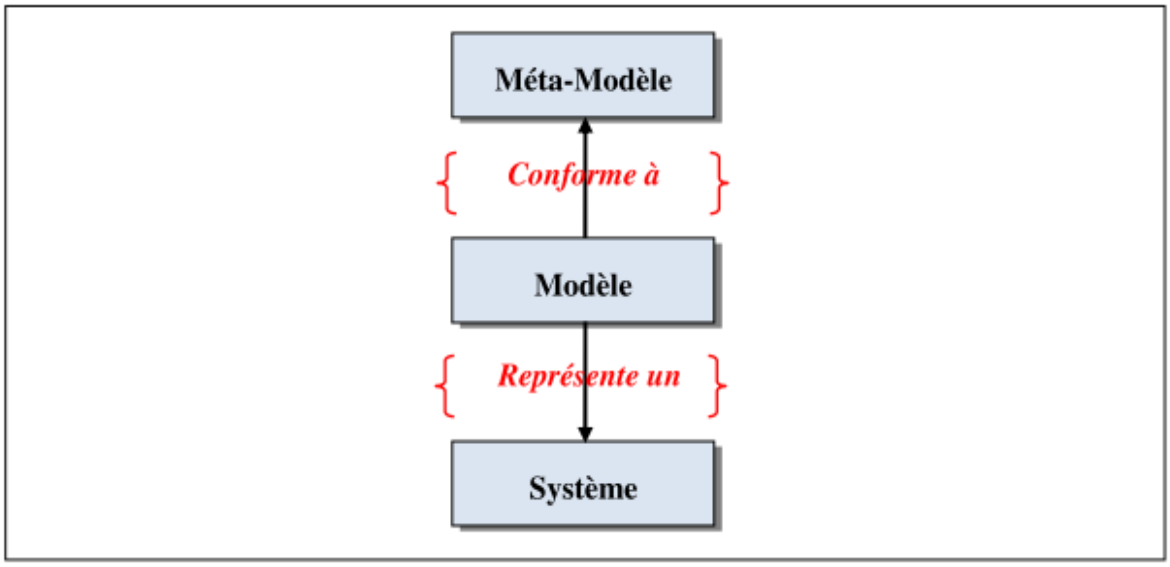



**Figure 2.1 : Relation entre système, modèle et méta-modèle**

## 3. Transformation de Modèles :

La notion de transformation de modèles constitue l'élément central de la démarche IDM. En effet, cette notion porte sur l'automatisation de l'opération de transformation pendant le cycle de développement qui peut avoir des sémantiques différentes en fonction des utilisations: raffinement, optimisation, génération de code, etc. [13]

La transformation de modèles est une opération qui consiste à générer un ou plusieurs modèles cibles conformément à leur méta-modèle à partir d'un ou de plusieurs modèles sources conformément à leur méta-modèle. Elle est qualifiée d'endogène si les modèles sources et cibles

Le développement logiciel basé sur des modèles (MDD) a été utilisé avec succès au cours des deux dernières décennies pour la génération de systèmes logiciels. La transformation de modèle consiste à définir des transformations entre des modèles (différents). Il joue un rôle central dans MDD et plusieurs autres applications. Les transformations de modèles dans MDD sont

[13] Krzysztof Czarnecki and Simon Helsen. Feature-based survey of model transformation approaches.IBM systems journal, 45(3):621–645, 2006

particulièrement utilisées pour refactoriser les modèles, pour les traduire en modèles intermédiaires et pour générer du code.[14]

## 3.1 Classification des approches de transformation:

selon Czarnek [15] , il existe deux grandes classes de transformation de modèles: les transformations de type Modèle vers code qui sont aujourd'hui relativement matures et les transformations de type modèle vers modèle qui sont moins maîtrisées. La deuxième classe de transformation fait l'objet de plusieurs recherches au cours de ces dernières années et plus particulièrement depuis l'apparition du MDA (Model Driven Architecture).

Le MDA se résume à la pyramide suivante avec 4 niveaux d'abstraction.

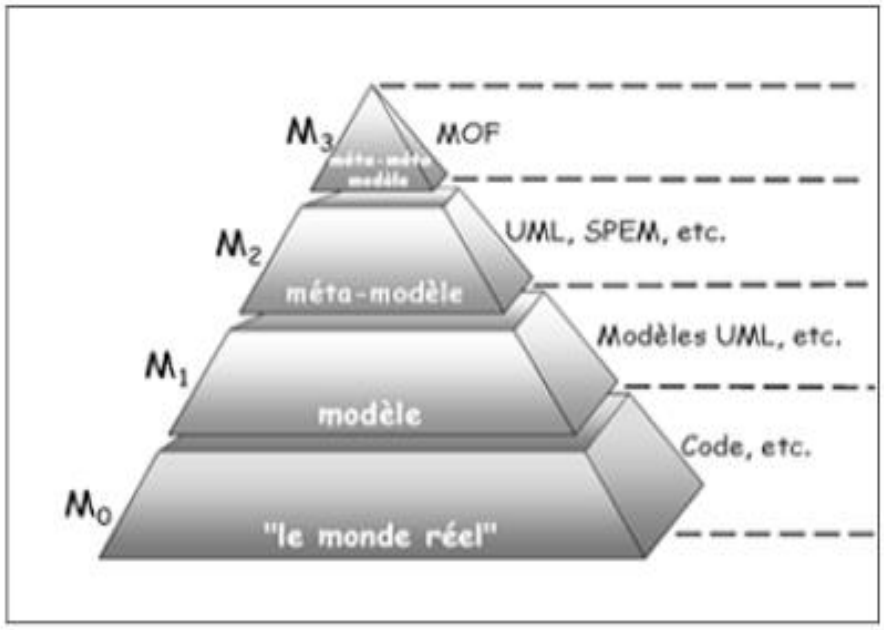



**Figure 2.2 : Pyramide de modélisation de l'OMG.( Object Modeling Group)**

[14] Hartmut Ehrig, Claudia Ermel, Ulrike Golas, and Frank Hermann. Graph and modeltransformation.Monographs in Theoretical Computer Science. Springer, 2015

[15] Krzysztof Czarnecki and Simon Helsen. Classification of model transformation ap-proaches. InProceedings of the 2nd OOPSLA Workshop on Generative Techniquesin the Context of the Model Driven Architecture, volume 45, pages 1–17. USA, 2003.

**M0** : Niveau des instances des modèles réels, il est composé des informations que l'on souhaite modéliser.

**M1** : Ce niveau représente toutes les instances d'un méta-modèle.. La définition de ces modèles est fournie explicitement au niveau M2.

**M2** Ce niveau représente toutes les instances d'un méta-méta-modèle.

**M3** : Composé d'un langage unique de définition des méta-modèles (Méta-meta modèle ou MOF).

### 3.1.1 - Transformations de type Modèle vers code :

on distingue dans ce type entre deux approches : basées sur le principe du **visiteur** (Visitor-based approach) par exemple le **framework Jamda** et celles basées sur le principe **des patrons** (Template-based approach), Actuellement, la majorité des outils **MDA** disponibles supportent cette approche

### 3.1.2 Transformations de type modèle vers modèle :

Les transformations de type modèle vers modèle consistent à transformer un modèle source en un modèle cible, ces modèles peuvent être des instances de différents méta-modèles. Elles offrent des transformations plus modulaires et faciles à maintenir

## 4. Quelques rappels sur la théorie des graphes :

1. **Définition de graphe :** On appelle graphe G = (V, E) ou :

V : l'ensemble fini dont les éléments sont appelés sommets

**E** : l'ensemble de couples sommets (v1 ,v2) sont appelés arêtes

Il existe deux types de graphes : les graphes non orientés et les graphes orientés.

## 2. Définition de graphe non orienté :

Un graphe non orienté **G** est la donnée d'un couple G =(V, E) tel que :

V : est un ensemble fini de sommets.

E : l'ensemble de couples non ordonnés de sommets (v1 ,v2) \in v2

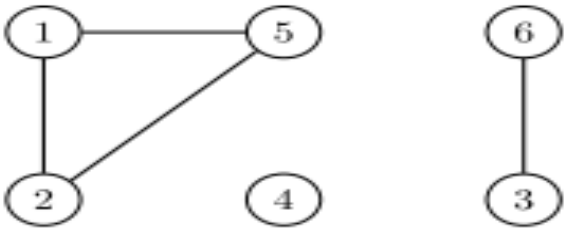



**Figure 2.3: graphe non orienté**

représente le graphe non orienté G = (V, E) avec V = {1, 2, 3, 4, 5, 6} et E = {{1, 2}, {1, 5}, {5, 2}, {3, 6}}.

## 3. Définition de graphe orienté :

Un graphe orienté **G** est la donnée d'un couple G =(V, E) tel que :

V : est un ensemble fini de sommets.

E : l'ensemble de couples ordonnés de sommets (v1 ,v2) \in v2

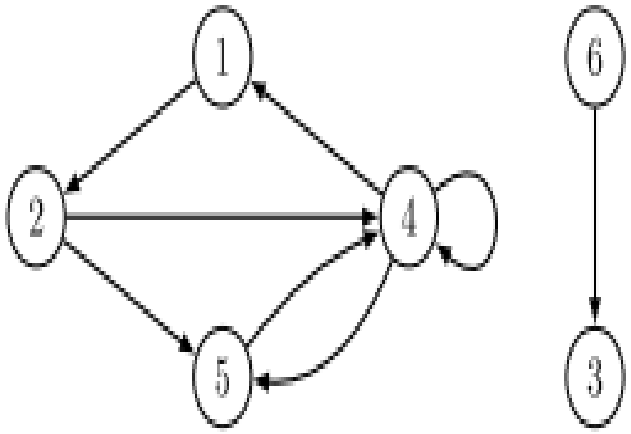



**Figure 2.4 : graphe orienté**

représente le graphe orienté G (V, E) avec V = {1, 2, 3, 4, 5, 6} et E = {(1, 2), (2, 4), (2, 5), (4, 1), (4, 4), (4, 5), (5, 4), (6, 3)}.

## 5. La transformation des graphes :

La transformation de graphes est apparue à cause du manque d'expressivité des approches classiques de ré-écriture comme les grammaires de Chomsky et la ré-écriture de termes pour gérer les structures non-linéaires. Les premières propositions apparues à la fin des années 60 et au début des années 70 [16]. Ces travaux s'occupaient de traductions de diagrammes et d'implémentations efficaces de λ-réduction basée sur des structures de graphes.

La théorie des graphes ouvre un grand champ de modélisation conduisant à des solutions efficaces pour de nombreux problèmes, et dans sa définition intuitive un graphe est un schéma constitué par un ensemble de points et par un ensemble de flèches reliant chacune deux de ceux-ci.

[16] John L Pfaltz and Azriel Rosenfeld. Web grammars. InProceedings of the 1stinternational joint conference on Artificial intelligence, pages 609–619, 1969

Une transformation de graphe [17] consiste en l'application d'une règle à un graphe, et la partie du graphe qui corresponde cette règle sera remplacée par un autre graphe. Ce processus est réitéré jusqu'à ce qu'aucune règle ne puisse être appliquée.

## 5.1 Grammaire de Graphe :

Une grammaire de Graphe [18] est généralement définie par un triplet : GG = (P, S, T) où :

**P** : ensemble de règles.

**S** : un graphe initial.

**T** : ensemble de symboles

L'idée est de combiner deux concepts importants: les graphes et les règles. La grammaire de graphes distingue les graphes non terminaux, qui sont les résultats intermédiaires sur lesquels les règles sont appliquées et les graphes terminaux dont on ne peut plus appliquer de règles. Ces derniers sont dans le langage engendré par la grammaire de graphe. Pour vérifier si un graphe. G est dans les langages engendrés par une grammaire de graphe, il doit être analysé. Le processus d'analyse va déterminer une séquence de règles dérivant G

## 5.2 Le principe de règles :

soit r une règle de transformation de graphe est définie par : r = (L, R, K, glue, emb, cond)

**L :** graphe de coté gauche.

[17] Anna Gerber, Michael Lawley, Kerry Raymond, Jim Steel, and Andrew Wood. Trans-formation: The missing link of mda. InInternational Conference on Graph Trans-formation, pages 90–105. Springer, 2002.
[18] Marc Andries, Gregor Engels, Annegret Habel, Berthold Hoffmann, Hans-Jörg Kre-owski, Sabine Kuske, Detlef Plump, Andy Schürr, and Gabriele Taentzer. Graphtransformation for specification and programming.Science of Computer program-ming, 34(1):1–54, 1999

**R :** graphe de coté droit

**K :** Un sous graphe de **L**.

**glue** : Une occurrence de K dans R qui relie le sous graphe avec le graphe de coté droit.

**emb** : Une relation d'enfoncement qui relie les sommets du graphe de coté gauche et ceux du graphe du coté droit.

**cond** : Un ensemble qui spécifie les conditions d'application de la règle.

## 5.3 Application des règles :

L'application d'une règle r = (L, R, K, glue, emb, cond) à un graphe **G** produit un graphe résultant **H**. Le graphe **H** peut être obtenu depuis le graphe d'origine **G** en passant par les cinq étapes suivantes :

1. Choisir une occurrence du graphe de coté gauche L dans G.
2. Vérifier les conditions d'application d'après cond.
3. Retirer l'occurrence de L(jusqu'à K) de G ainsi que les arcs pendillé, c'est-à-dire tout les arcs qui ont perdu leurs sources et/ou leurs destinations. Ce qui fourni le graphe de contexte D de L qui a laissé une occurrence de K.
4. Coller le graphe de contexte **D** et le graphe de coté droit **R** suivant l'occurrence de **K** dans **D** et dans **R**. C'est la construction de l'union de disjonction de D et R et, pour chaque point dans K, identifier le point correspondant dans D avec le point correspondant dans R.
5. Enfoncer le graphe de coté droit dans le graphe de contexte de L suivant la relation d'enfoncement emb.

L'application de r à un graphe G pour produire un graphe H est appelée une dérivation directe depuis G vers H à travers r, elle est dénotée par G → H.

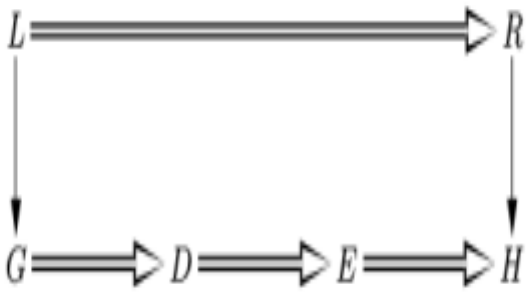



**Figure 2.5 : transformation directe de graphe**

## 5.4 Langage engendré :

Soit P de règles et un graphe G 0 , une séquence de transformations de graphe successive :G 0 → G 1 → ... → G n est une dérivation à partir de G 0 vers G n par les règles de P (à condition que toutes les règles utilisées appartiennent à P). G 0 est le graphe initial et G n est le graphe dérivé de la séquence de transformation. L'ensemble des graphes dérivés à partir d'un graphe initial S en appliquant les règles de P qui sont étiquetées par les symboles de T, est dit langage engendré par P, S et T et on écrit L(P, S, T)

## 5.5 Qu'est-ce que la transformation graphique?

Il existe de nombreuses transformations de graphe racine à définir:
• des grammaires de Chomsky sur les chaînes aux grammaires des graphes;
• de la réécriture de termes à la réécriture de graphes;
• de la description textuelle à la modélisation visuelle.

Pour donner un moyen simple de comprendre le concept de grammaire et de réécriture de graphes, nous utilisons la notion de transformation de graphes. Dans tous les cas, l'idée

principale de la transformation de graphe est la modification basée sur des règles des graphes, comme le montre la Fig


**Figure 2.6 : règle de transformation de graphe**

une règle de graphe ou une production, p = (L, R) est une paire de graphes (L, R), appelés le côté gauche L et le côté droit R. Application de la règle p = (L, R ) signifie trouver une correspondance de L dans le graphe source et remplacer L par R, conduisant au graphe cible de la transformation de graphe.
Les principaux problèmes techniques sont de savoir comment supprimer L et comment connecter R avec le contexte dans le graphe cible. En fait, il existe plusieurs solutions différentes pour gérer ces problèmes, conduisant à plusieurs approches de transformation de graphe différentes, qui sont résumées ci-dessous.

## 5.6 Approches de transformation de graphes : [19]

- Approche de remplacement d'étiquette de nœud
- Approche de remplacement Hyperedge
- Approche algébrique
    - Double Pushout (DPO) (since 1973)

[19] Grzegorz Rozenberg.Handbook of graph grammars and computing by graph transfor-mation, volume 1. World scientific, 1997

- Single Pushout (SPO) (since 984/90)
- High Level Replacement (HLR)(since 1991/2004)

- Approche logique
- Théorie des 2 structes
- Approche de remplacement de graphe programmée

### 5.6.1 L'approche algébrique :

la transformation de graphe algébrique peut prendre en charge la définition et l'analyse des transformations de modèles basées sur des règles . Surtout pour les modèles visuels, la transformation de graphes est un choix naturel pour manipuler leurs structures de graphes sous-jacentes. L'approche double pushout (DPO) peut être interprétée comme une sorte de transformation sur place, où le graphe source est transformé étape par étape en graphe cible. L'utilisation de l'approche DPO pour les graphes typés - avec des graphes de types différents pour le domaine source et cible - nous permet d'assurer la cohérence des types par construction .[20]

L'idée principale est de modéliser transformation de graphe par deux constructions de collage pour les graphes et chaque construction de collage par un pushout. En gros, une production est donnée par p = (L, K, R), où L et R sont les graphes de gauche et de droite et K est une interface commune de L et R. Étant donné une production p = (L, K, R) et un graphe contextuel D, qui comprend aussi l'interface K, le graphe source G d'une transformation graphe

G ⇒ H via p est donné par le collage de L et D via K, noté G = L + kD, et le graphe cible H par le collage de R et D via K, écrit H = R + kD. Plus précisément, nous utiliserons les morphismes de graphes K → L, K → R et K → D pour

[20] Même référence de 14

exprimer comment K est inclus dans L, R et D respectivement. Ceci permet de définir les constructions de collage G = L + kD et H = R + kD comme des constructions de poussée (1) et (2) conduisant à une double poussée dans fig10. Avant de présenter l'essentiel de l'approche algébrique de l'approche DPO

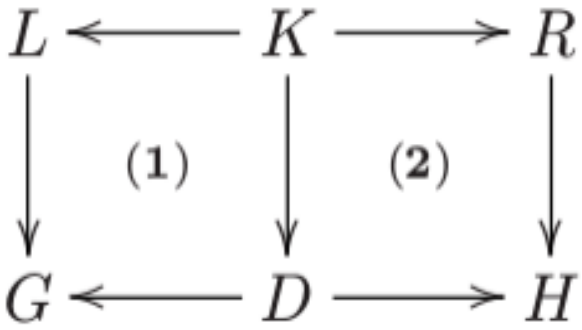



**Figure 2.7 L'approche double pushout (DPO).**

## 6. Présentation de l'outil AToM3 :

AToM3 [21] « A Tool for Multi-formalism and Meta-Modeling » est un outil visuel pour la transformation de modèles, écrit en Python et s'exécute sur de différentes plateformes (Windows, Linux, etc.). Il utilise et implémente un ensemble de concepts tel que : la modélisation suivant un multi formalisme , la méta modélisation et les grammaires de graphes. Il est utile pour la modélisation, la méta modélisation et la transformation de modèles à l'aide des grammaires de graphes. Par ailleurs, il permet par extension de manipuler la simulation ainsi que la génération du code à partir des modèles élaborés. Les modèles dans AToM3 ont une représentation graphique et une construction basée sur des règles issues d'une spécification d'un formalisme déterminé. Dans AToM3, la description graphique est attribuée aux modèles ainsi qu'aux formalismes. AToM3 permet de générer des outils de manipulation graphique des modèles décrits selon des formalisme spécifiés dans des méta-spécification. Par

[21] AToM3 home page: http://moncs.cs.mcgill.ca/MSDL/research/projects/ATOM3.html.

la suite, la transformation de modèles s'effectue en utilisant des grammaires de graphes (cf. la figure 2.7) d'un méta modèle, il offre un canevas pour la modélisation graphique des différentes entités.

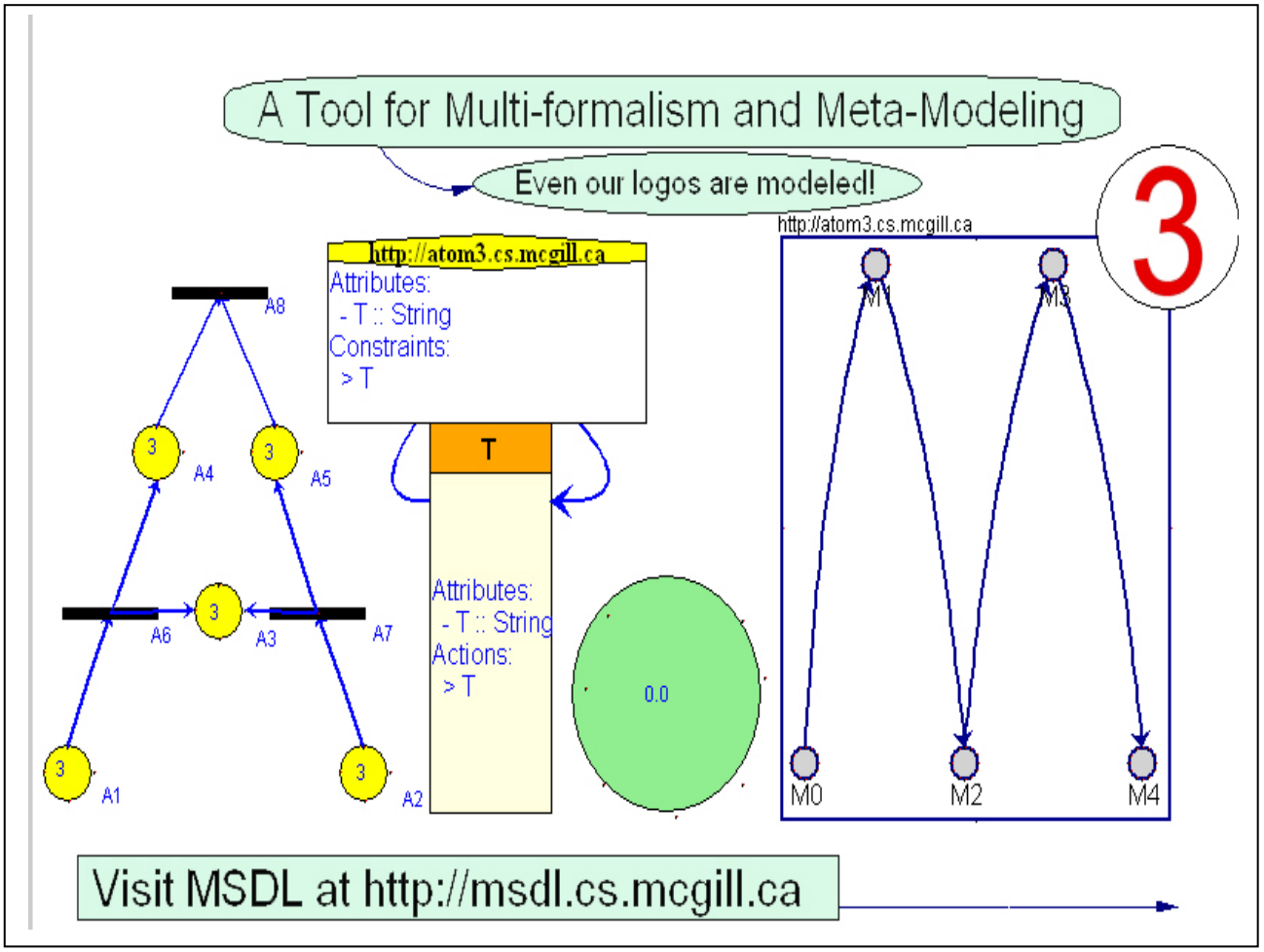



**Figure 2.8. Présentation AToM3**

La modélisation automatique du diagramme de classes avec ATOM3, doit passer par les étapes suivantes:

1. Chargement d'un méta-modèle pour les diagrammes de classes, qui se trouve dans les formalismes principaux d'ATOM3.

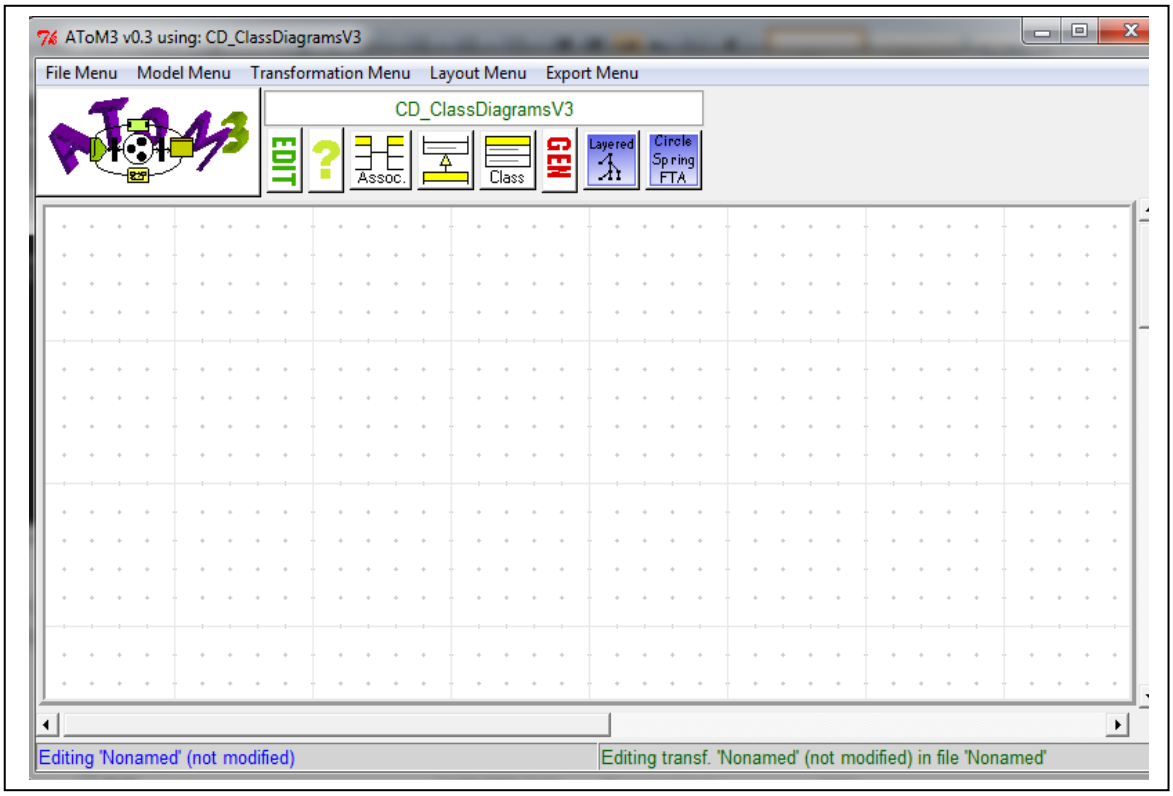



**Figure 2.9. Présentation le meta modele CD-ClassDiagramsV3**

2. Concevoir le méta-modèle du processus de production selon la structure définie dans les diagrammes de classes. Ce méta- modèle est composé d'une méta-classe présentant un composant du processus de production qui possède un nom ,les attributs et les méthodes mentionnées auparavant et une association avec la même méta-classe, avec une cardinalités égale à 0-N, comme le montre la Figure suivante :

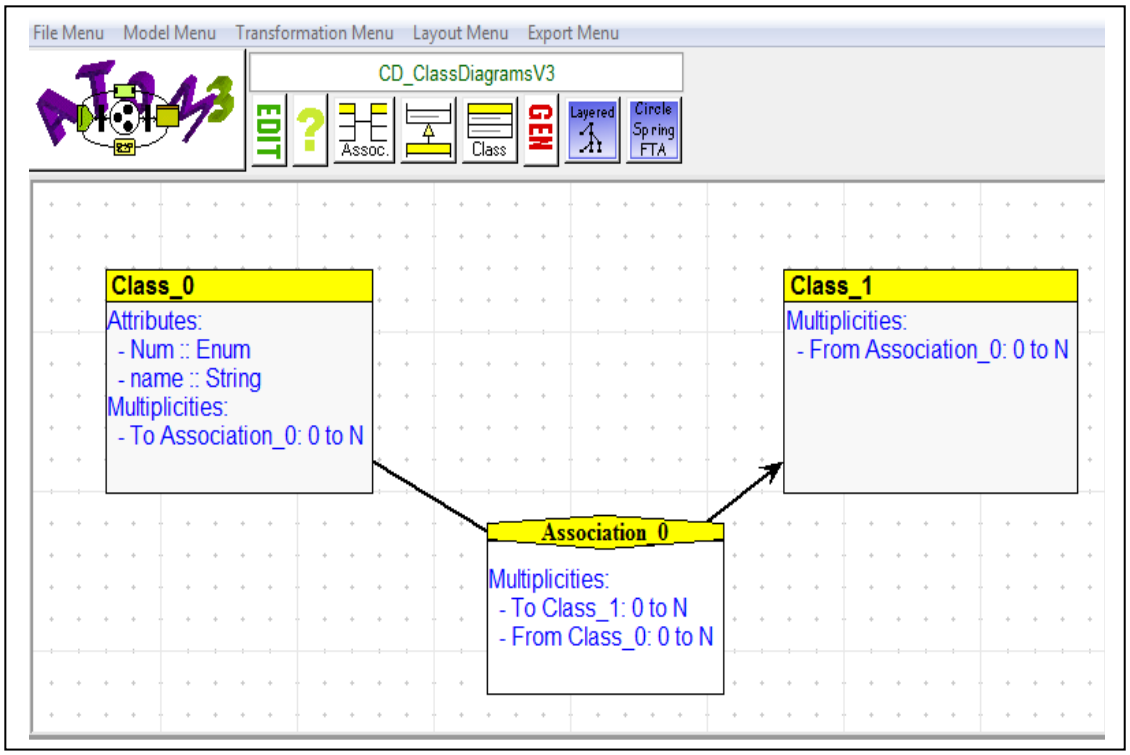



**Figure 2.10. Méta-modèle pour le diagramme de classes.**

3. A partir de ce méta-modèle, ATOM3 génère un outil de modélisation de diagramme de classes pour un processus de production distribué. Cet outil offre un ensemble de boutons de manipulation de ce diagramme. Cet outil est illustré dans la Figure suivante :

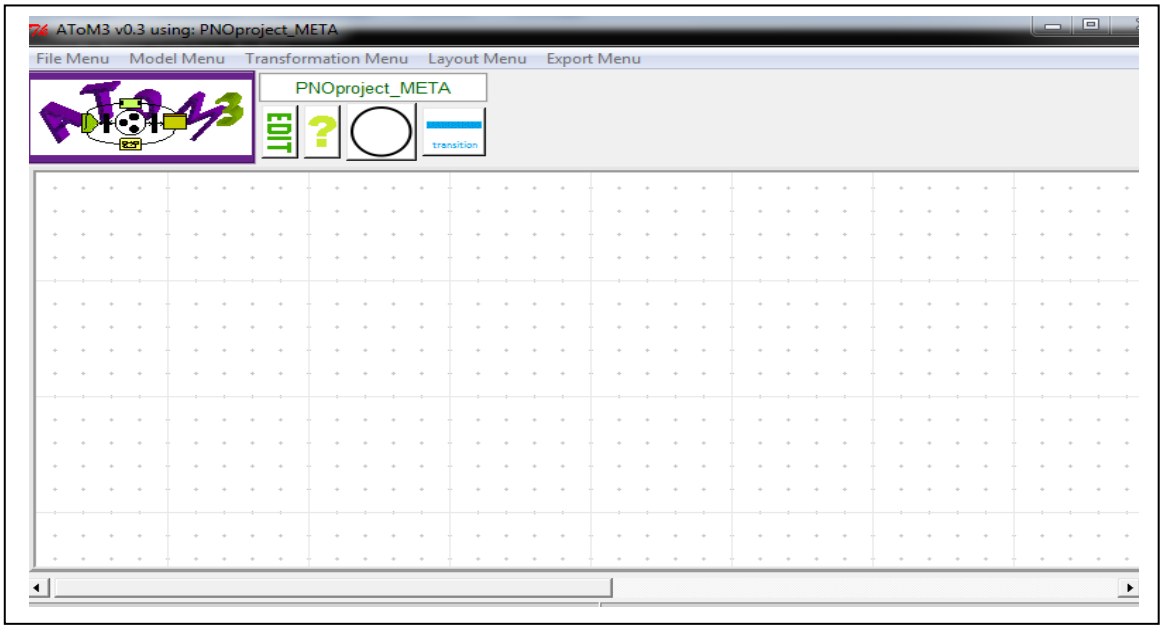



**Figure 2.11. Outil de modélisation généré par ATOM3**

## 7. Conclusion :

Dans ce chapitre nous avons présenté des définitions de modèle et un méta-modèle et bien sur un langage de modélisation et la relation entre eux, ensuite nous avons voir la transformation de modèles et la classification des ses approches , par la suite nous avons donné des détailles sur la transformation des graphe et on précise l'approche double pushout (DPO) et comme l'appliqué dans notre transformation sur les réseaux de pétri Nous avons également présenté l'outil de transformation de graphes AToM³, l'outil visuel de modélisation et de méta-modélisation multi-formalismes, utilisé dans l'implémentation de notre travail. Ce dernier sera présenté au chapitre suivant.

# Chapitre 3

# Conception et L’implémentation

# Chapitre 3

# Conception et L'implémentation

## 1. Introduction :

Dans les chapitres précédents nous cité les différents concepts les plus importants pour réaliser notre travail, nous passons maintenant à la conception et pour construire un tel système, on doit passer par plusieurs étapes, et comme il s'agit d'un outil logiciel, il est préférable de passer par un cycle de développement d'un logiciel. **La première étape** dans le cycle de développement d'un logiciel est l'analyse des besoins. Sa fonction principale est de préciser les services qui seront rendus par le logiciel ainsi que l'ensemble de contraintes sous lesquelles ce logiciel devra fonctionner. Cette phase est suivie d'une phase de conception. Une conception finale est généralement obtenue par un processus itératif à partir de conceptions préliminaires. Une bonne conception est la clé d'un développement de logiciel efficace. Un système bien conçu est facile à réaliser, à maintenir et à comprendre. La dernière phase consiste en une réalisation, lors de cette étape on réalise un ensemble d'unités de programme, écrites dans un langage de programmation exécutable

## 2. Analyse des besoins :

Les services principale offerts par notre environnement est permet avec un Rdp la modélisation du comportement des systèmes dynamiques à évènements discrets et la description des relations existantes entre des conditions et des évènements. Les RdPs permettent d'analyser et de simuler des systèmes. En étant rôle d'outil graphique, il nous aide à comprendre facilement le système modélisé, en plus il nous permet de simuler les activités dynamiques et concurrentes.

## 3. Conception :

Après avoir fixé l'objectif du projet et avant d'entrer dans le processus de développement ou de programmation, il est nécessaire de présenter les résultats d'analyse de notre sujet sous forme d'architecture abstraite. La figure 3.1 représente notre modèle abstrait.

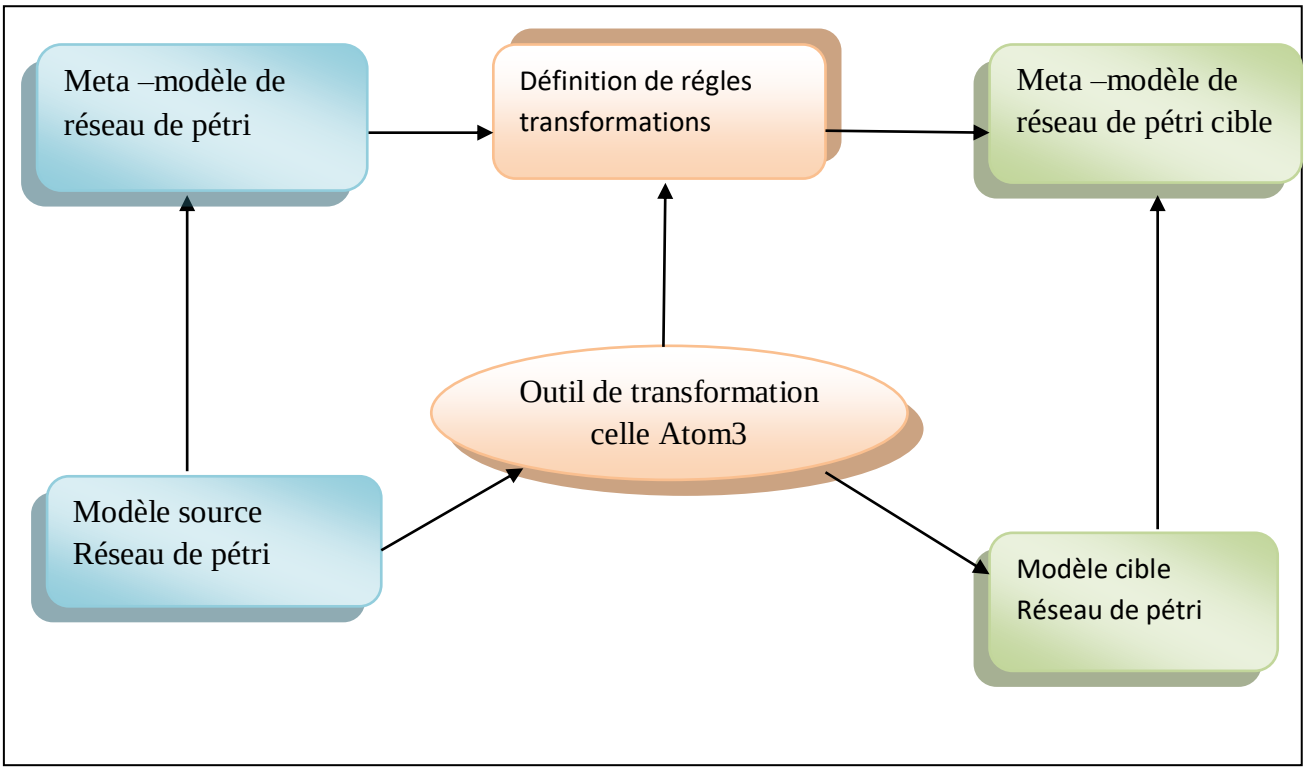



**figure 3.1 conception globale**

## 4. Méta-Modélisation de réseau de pétri :

Puisque les réseaux de Petri Ordinaire consistent en des places, des transitions et d'arcs de places vers des transitions et d'arcs de transitions vers les places, nous avons proposé pour les méta-modéliser, quatre classes comme le montre la figure suivante.

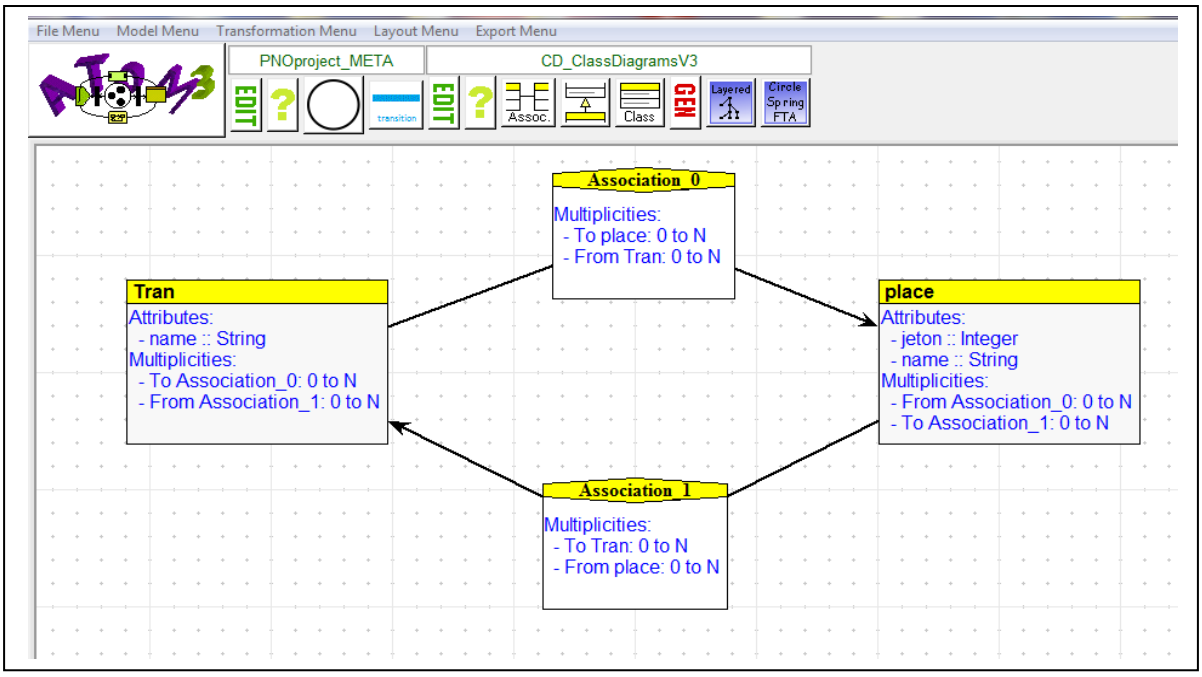



**figure 3.2 Méta-Modele de Reseau de petri**

## 5.Outil de modélisation d'un DATA :

Le méta-modèle ainsi défini est représenté dans AToM 3 (figure 3.2), va nous permettre la génération d'un outil qui contient deux boutons essentiels sur lesquels on peut dessiner un réseau de petri désiré .

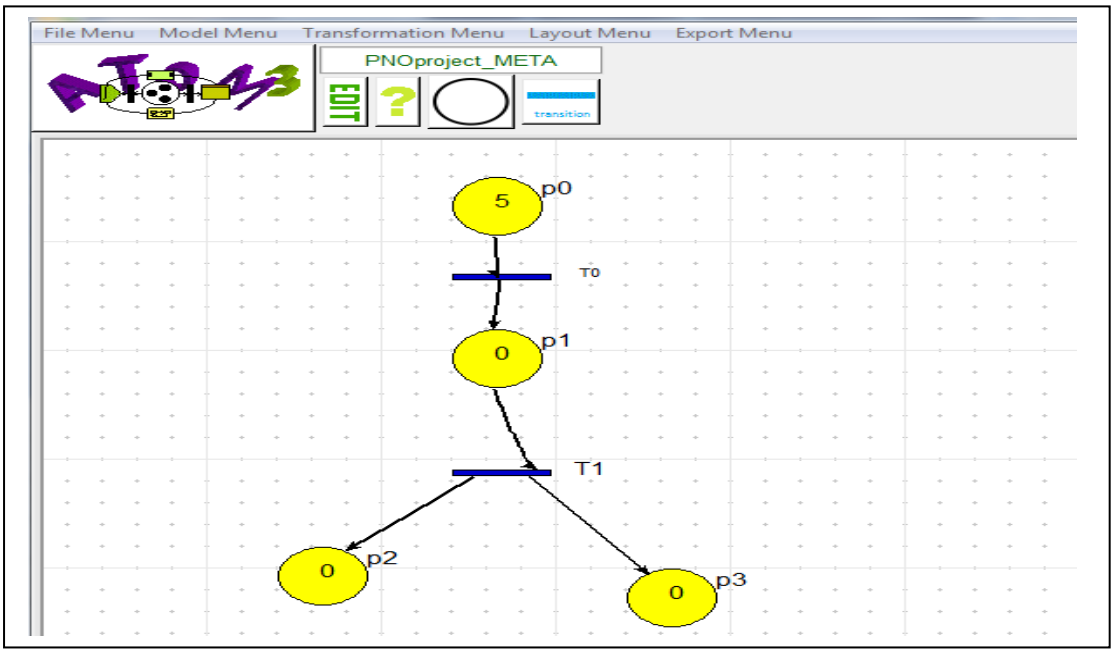
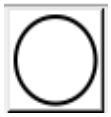



**figure 3.3 Outil de modélisation et un exemple d'un Rdp**

où :  pour representer une place .


pour répresenter une transition .

## 6. Grammaire de graphe proposée :

La grammaire proposée est composée d'ensemble des règles qui sont appliquées jusqu'à l'épuisement de toutes les règles.

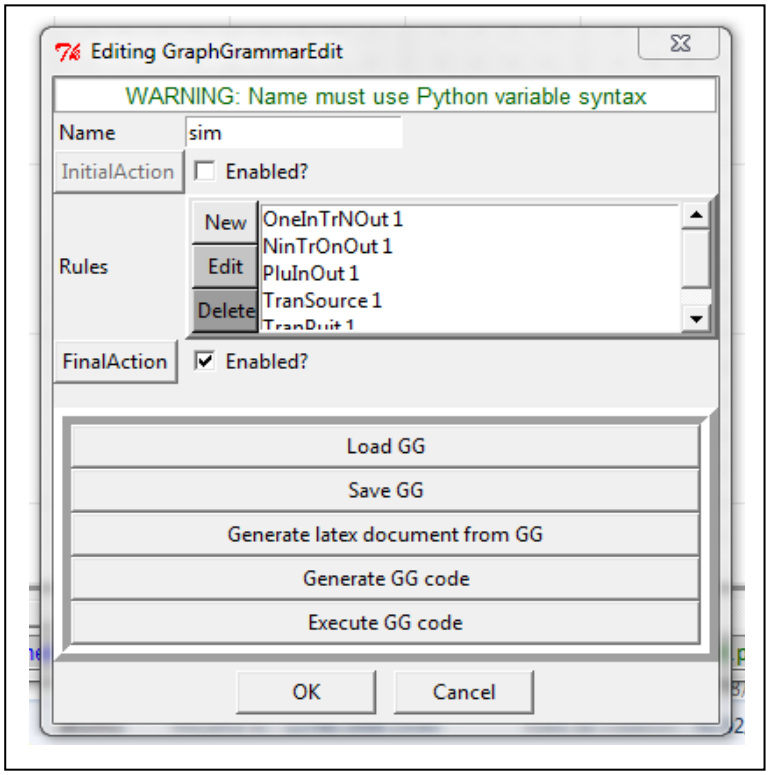



**figure 3.4 la fenètre de Grammaire proposée**

- **règle1 (OneInTrNOut) :** cette règle permet de faire le franchissement d'une transition a une seule entrée et plusieurs sorties

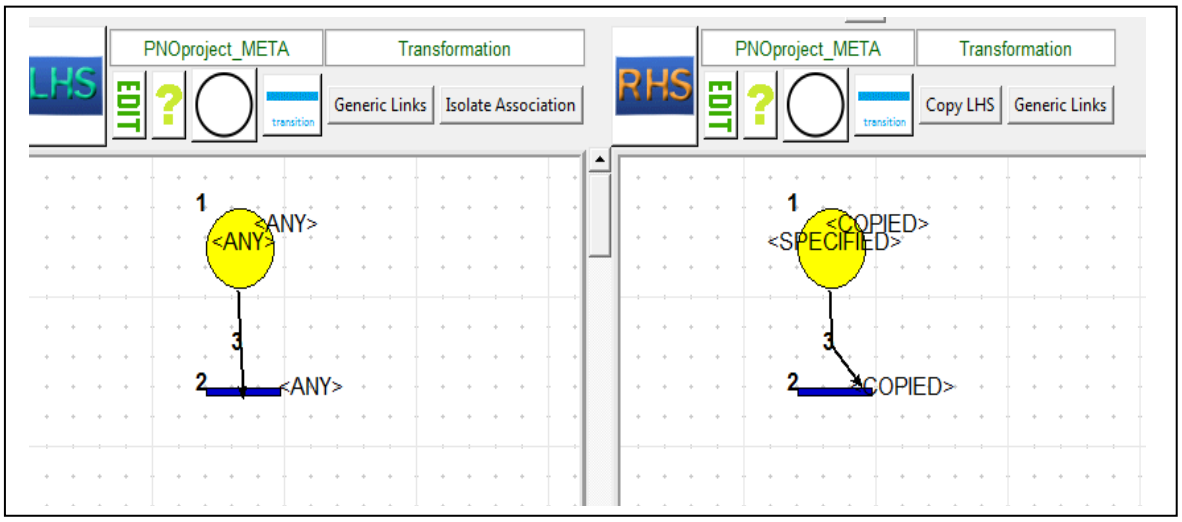



**figure 3.5 les parties gouche et droite de la règle OneInTrNOut**

a) **condition :** pour se exécuter cette regle il faut satisfaire cette condition qui est : il doit la transition avoir une seule entrée avec un nombre du jetons Supérieur ou égal à 1 et au moins une sortie ou plus

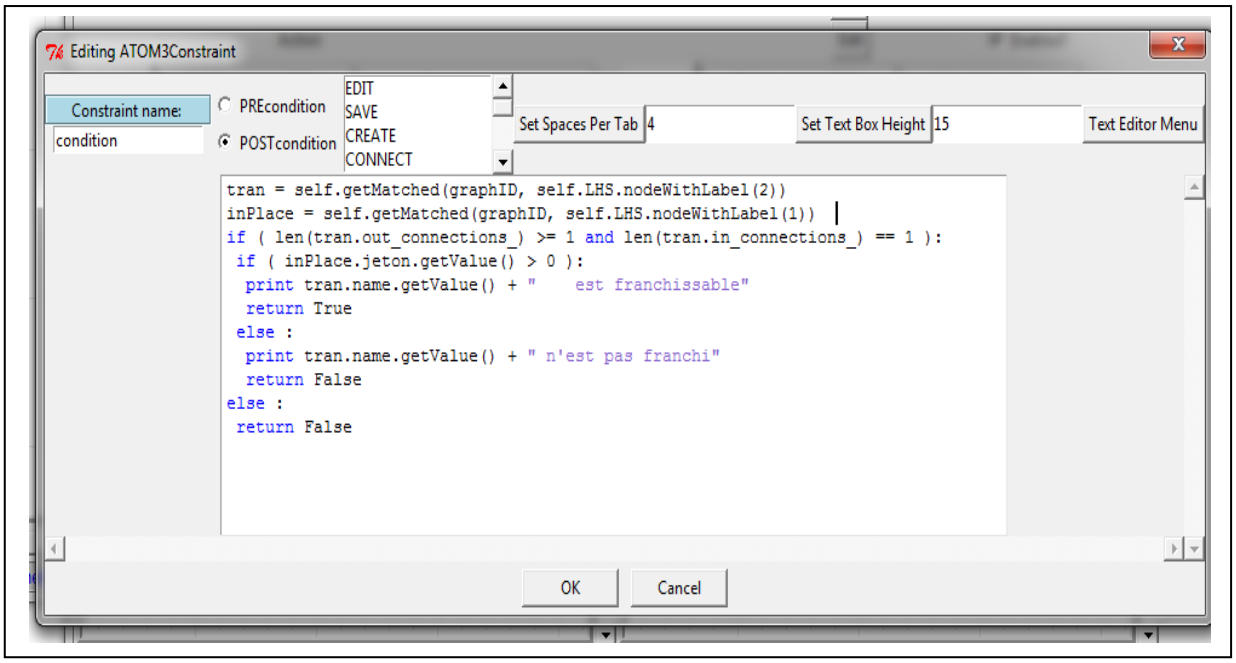



**figure 3.6 condition de la règle OneInTrNOut**

**b) action :**apres la verification de la condition on doit realiser cette action

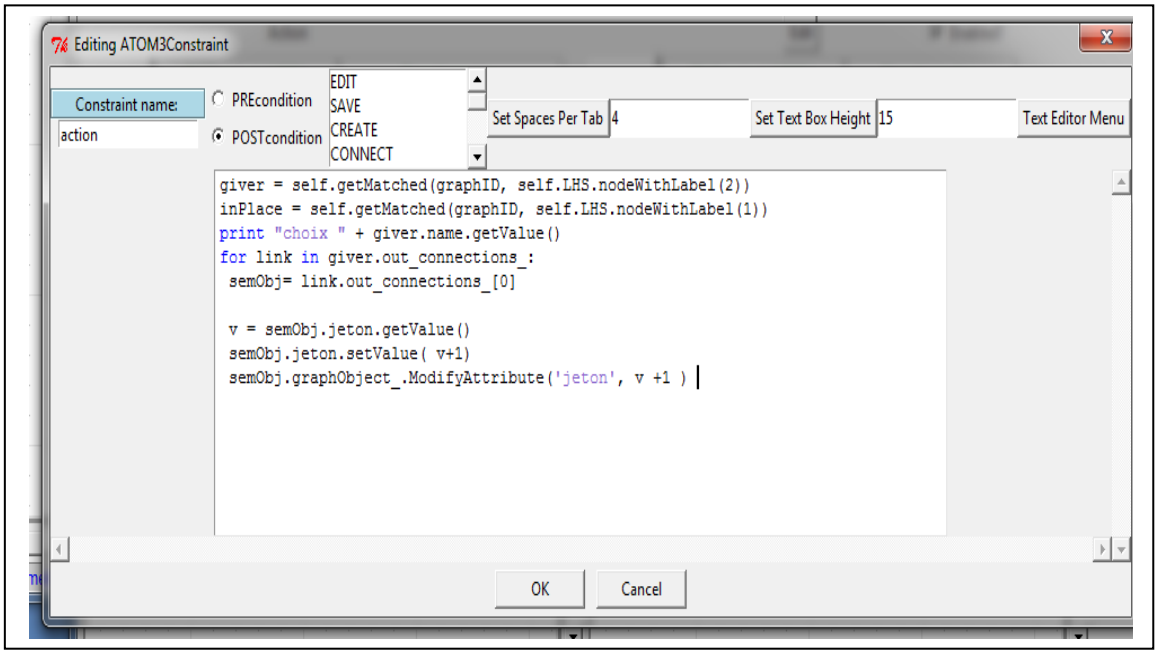



**figure 3.7 l'action de la règle OneInTrNOut**

- **règle 2 (NinTrOnOut) :** cette règle permet de faire le franchissement d'une transition a plusieurs entrées et une seule sortie

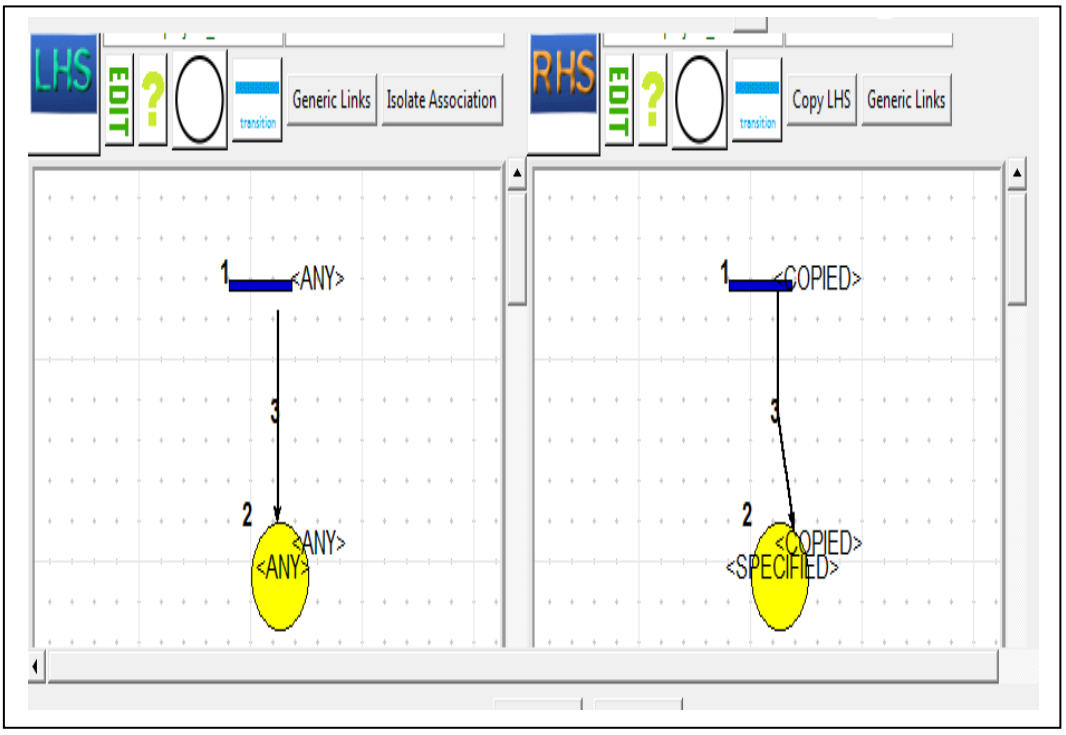



**figure 3.8 les parties gouche et droite de la règle NinTrOnOut**

**a) condition :** permit de franchir chaque transition a plusieurs entrées et une seule sortie

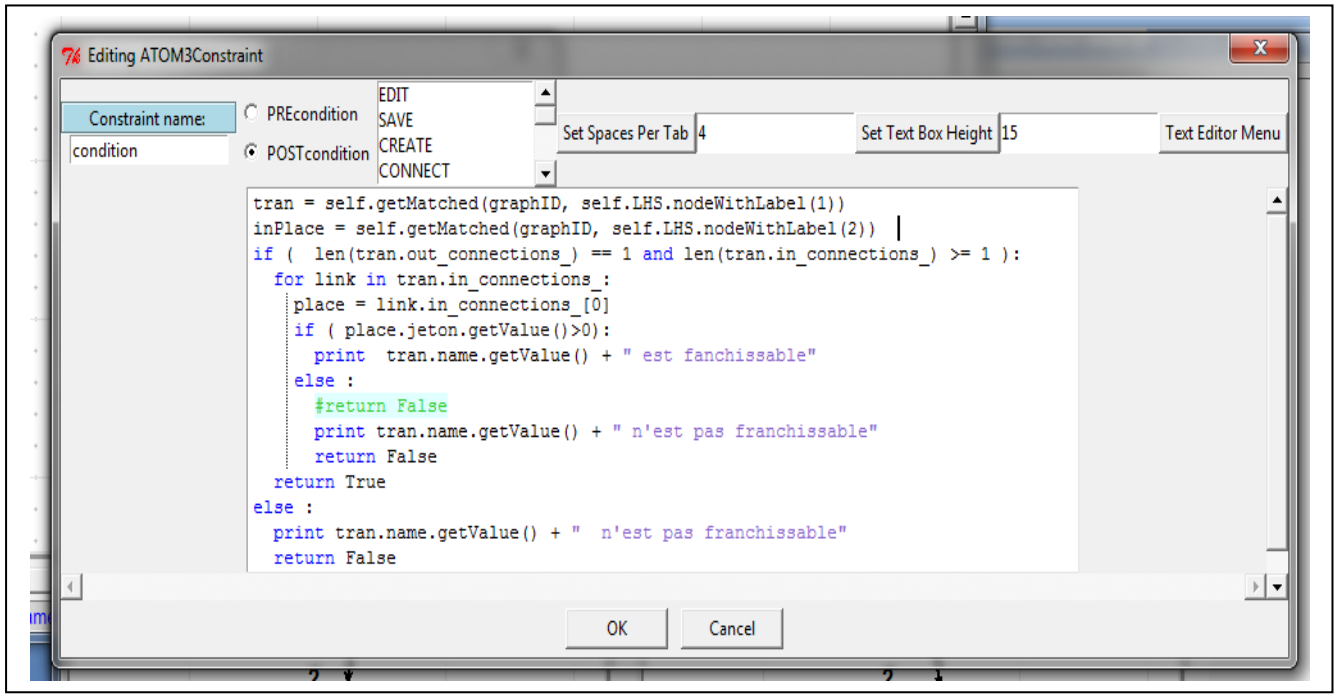



**figure 3.9 condition de la règle NinTrOnOut**

**b) action :**

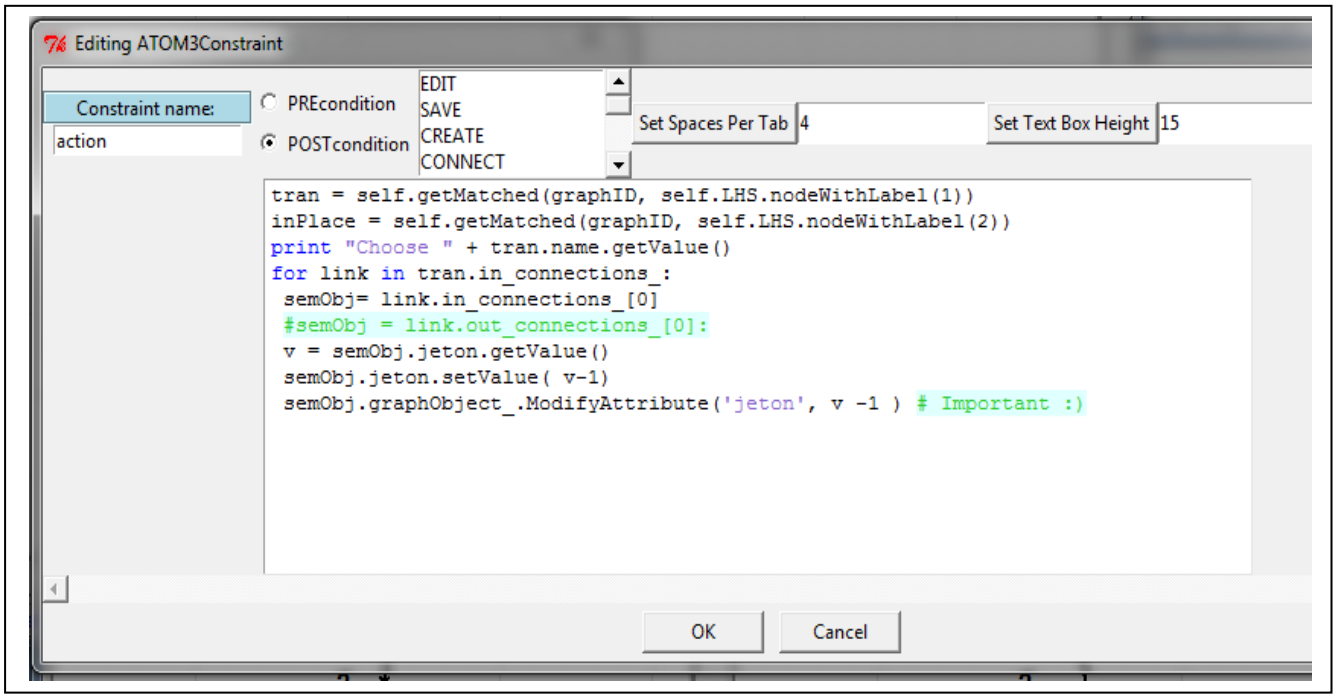



**figure 3.10 l'action de la règle NinTrOnOut**

- **règle 3 (PluInOut) :** cette règle permet de faire le franchissement d'une transition a plusieurs entrées et plusieurs sorties

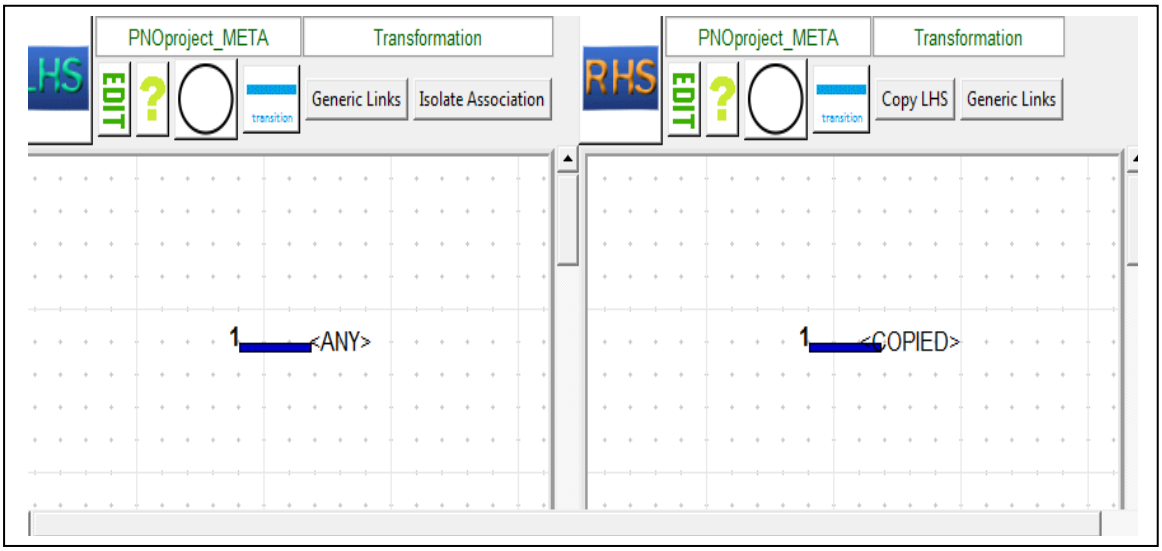



**figure 3.11 les parties gouche et droite de la règle PluInOut**

a) **condition :** permit de franchir chaque transition a plusieurs entrées et plusieurs sorties

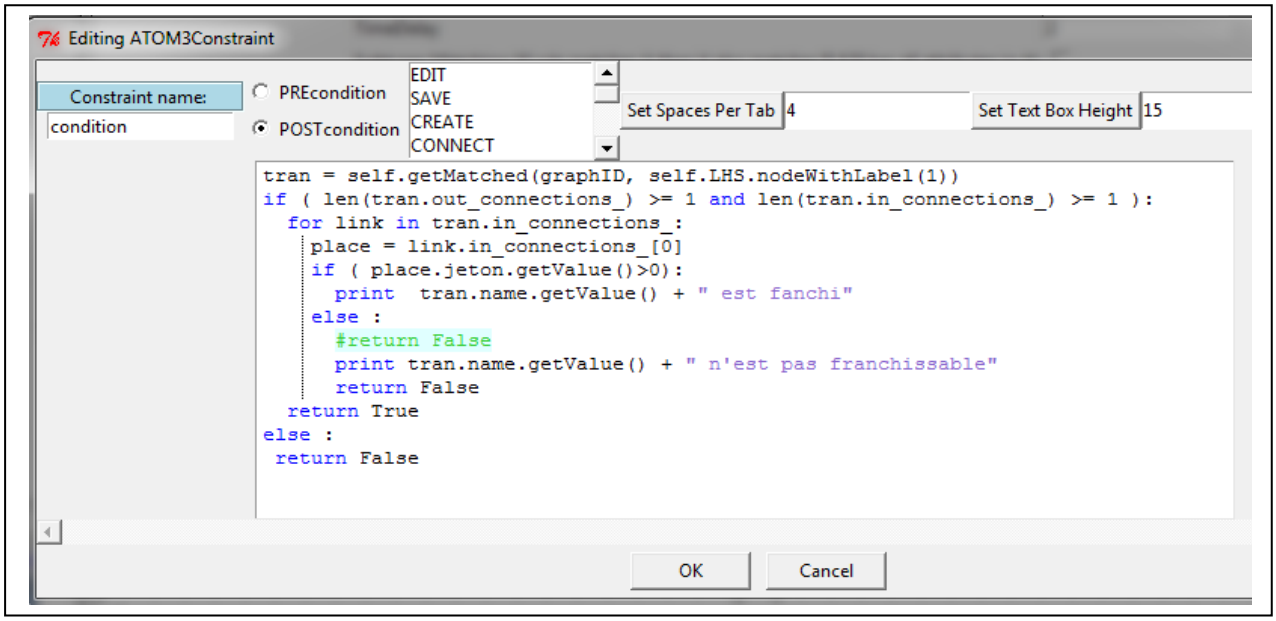



**figure 3.12 condition de la règle PluInOut**

### b) action :

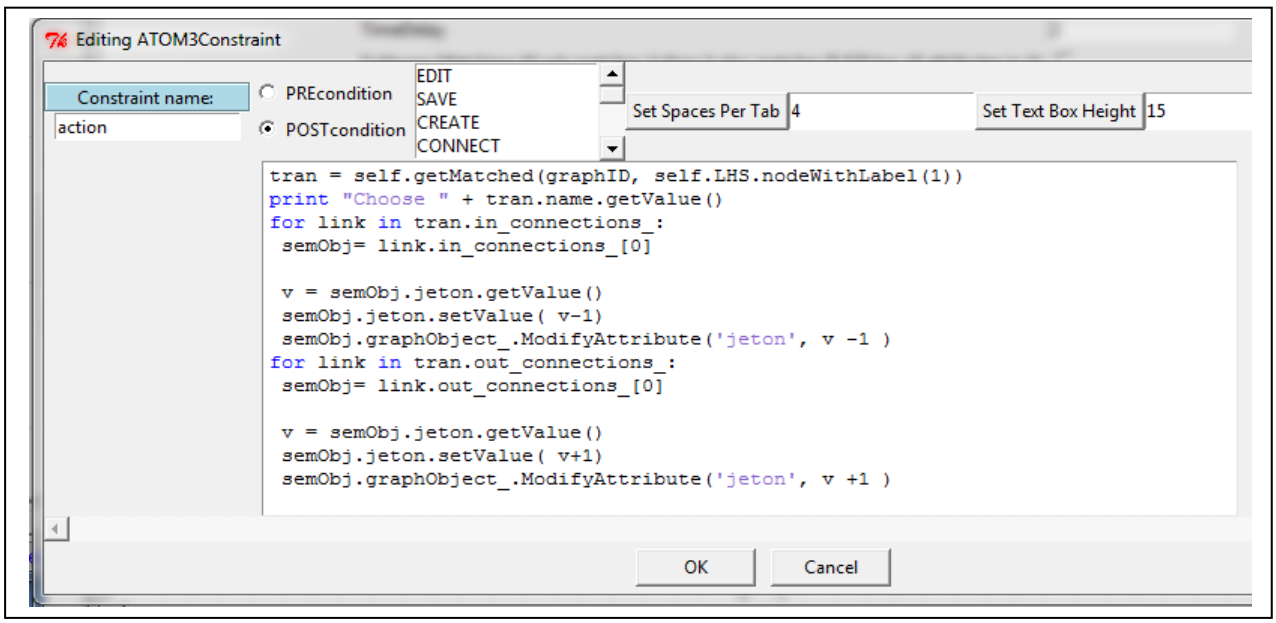



**figure 3.13 l'action de la règle PluInOut**

- **règle 4 (TranSource) :** représente le franchissement d'une transition source

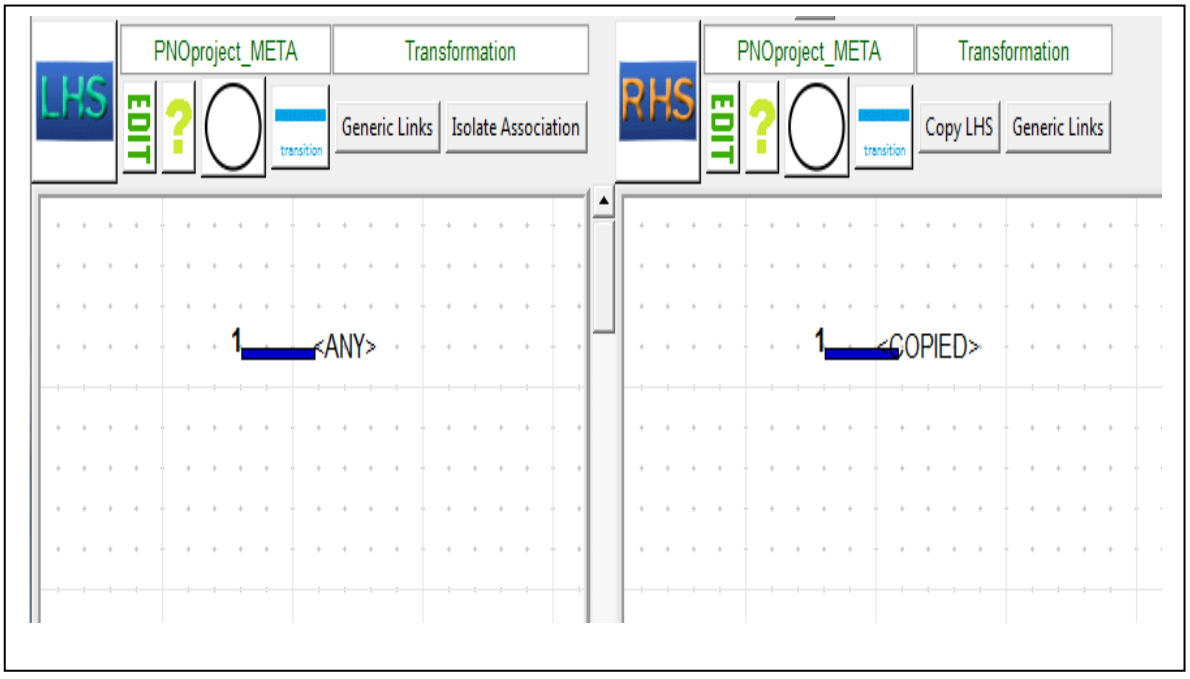



**figure 3.14 les parties gouche et droite de la règle TranSource**

a) **condition :** soit vérifier une transition a 1 ou plus sorties et aucune entrée

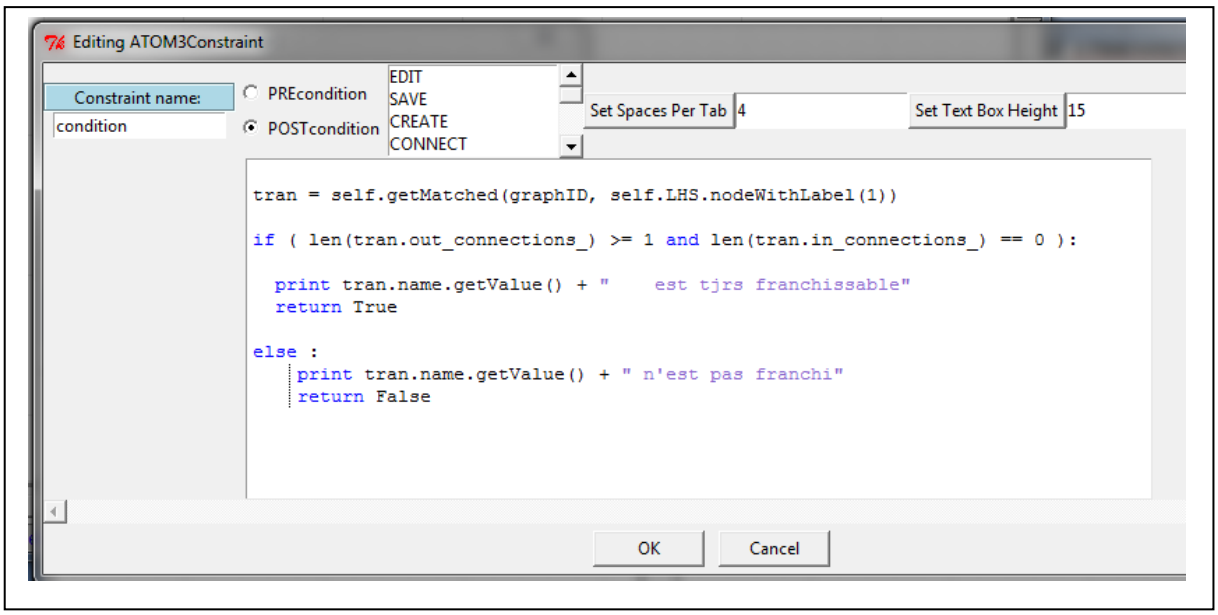



**figure 3.15 condition de la règle TranSource**

## b) Action :

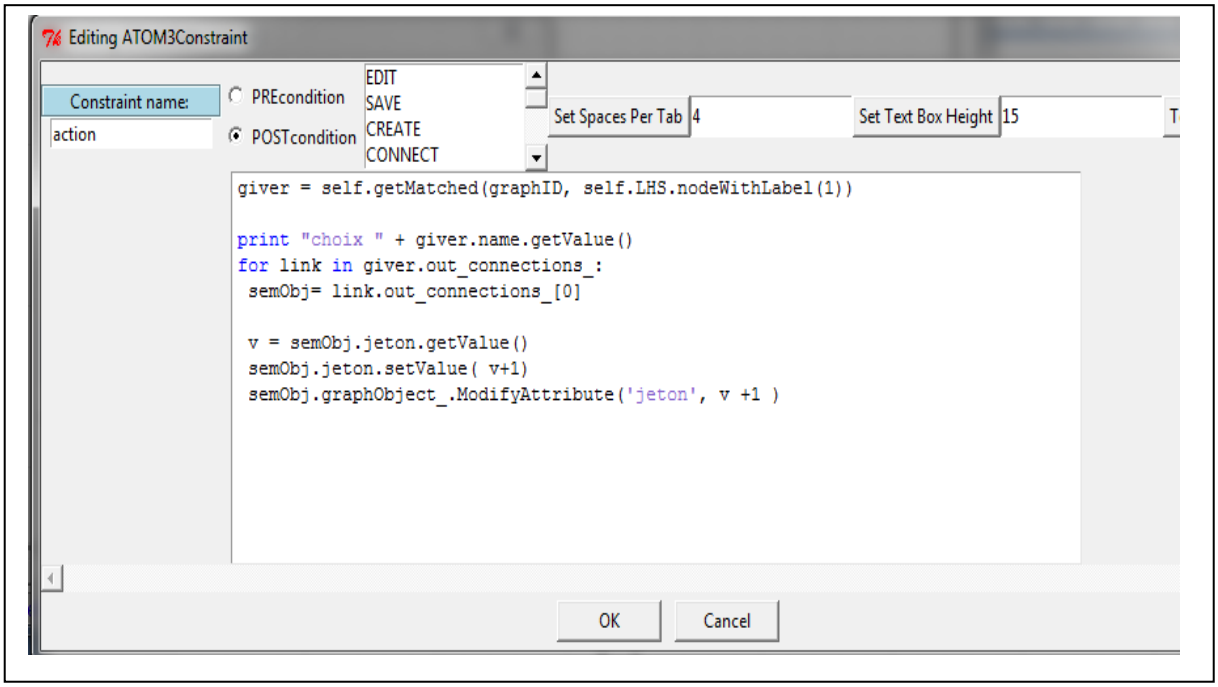



**figure 3.16 l'action de la règle TranSource**

- **règle 5 (TranPuit) :** représente le franchissement d'une transition Puits

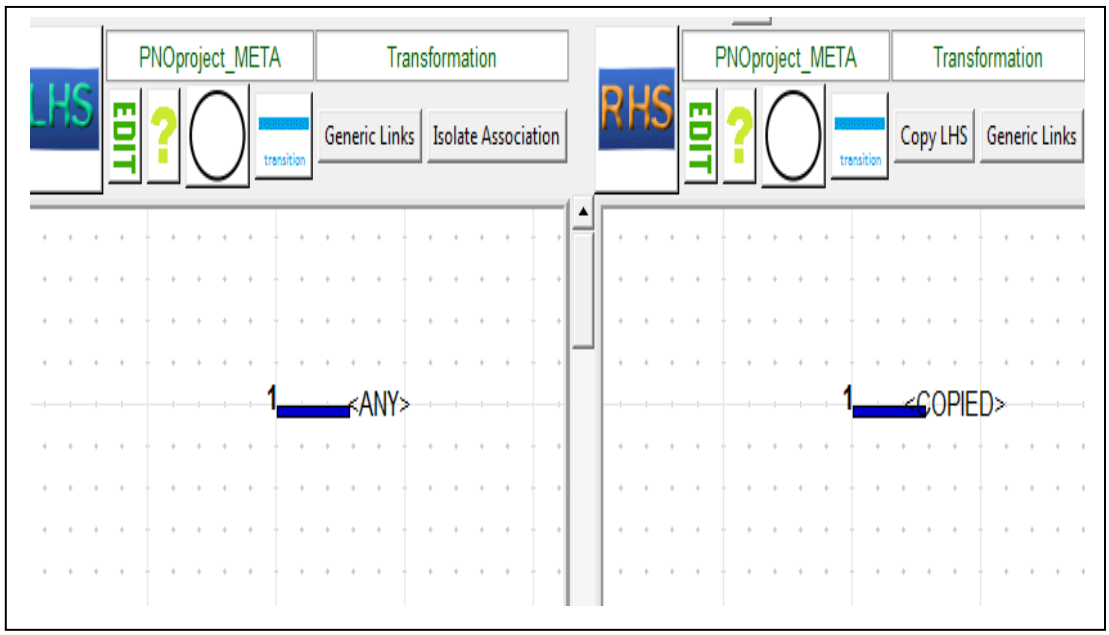



**figure 3.17 les parties gouche et droite de la règle TranPuit**

a) **condition :** soit vérifier une transition a 1 ou plus entrées et aucun sortie

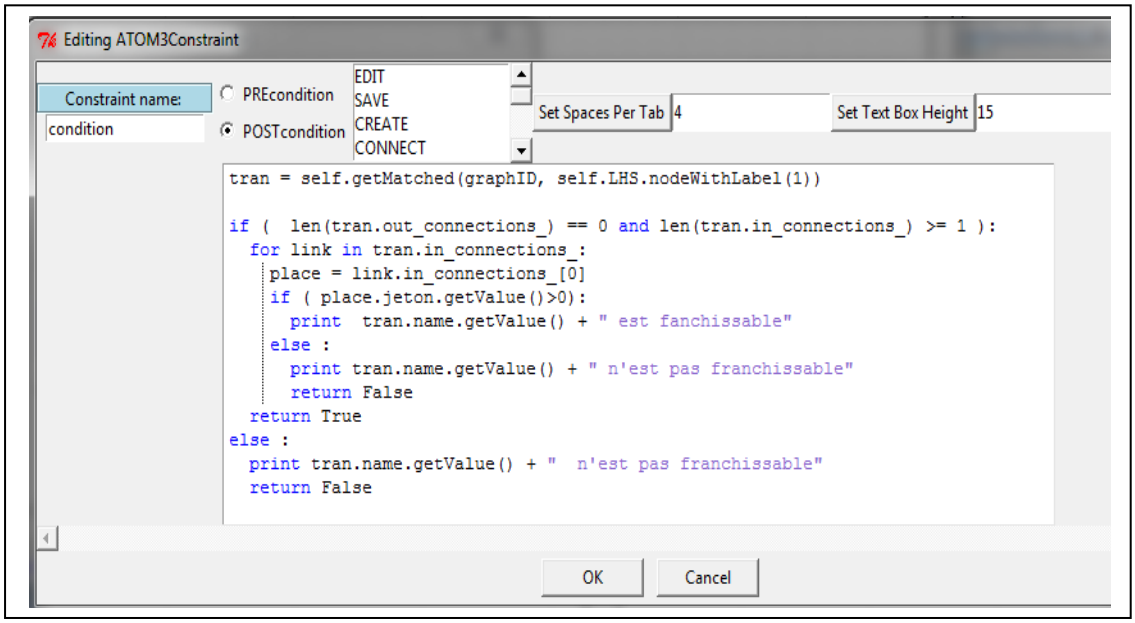



**figure 3.18 condition de la règle TranPuit**

### b) action :

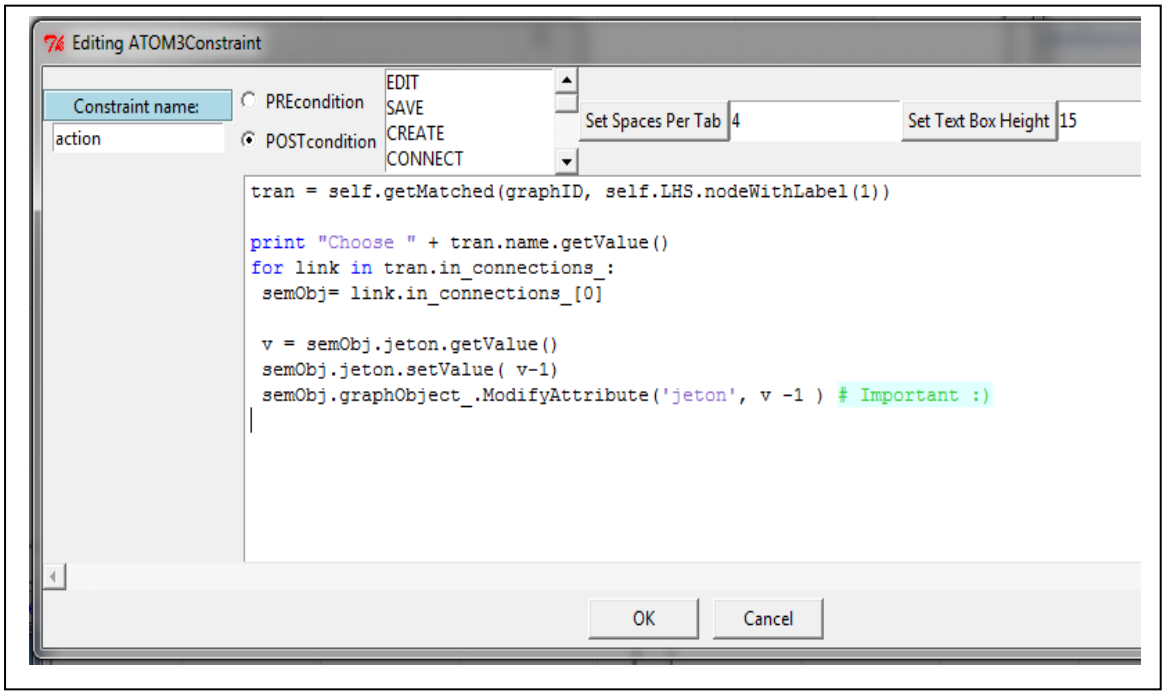



**figure 3.19 l'action de la règle TranPuit**

7. **Exemple d'exécution :** nous allons choisi un simple exemple comme ce suit :

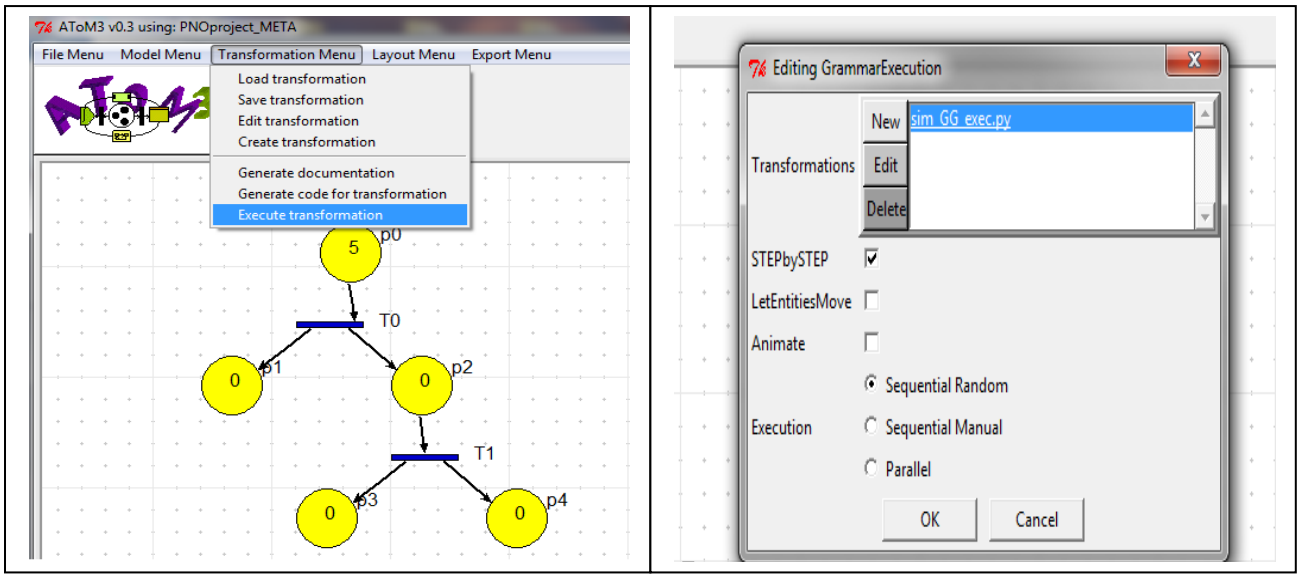



**figure 3.20 exemple à executer**

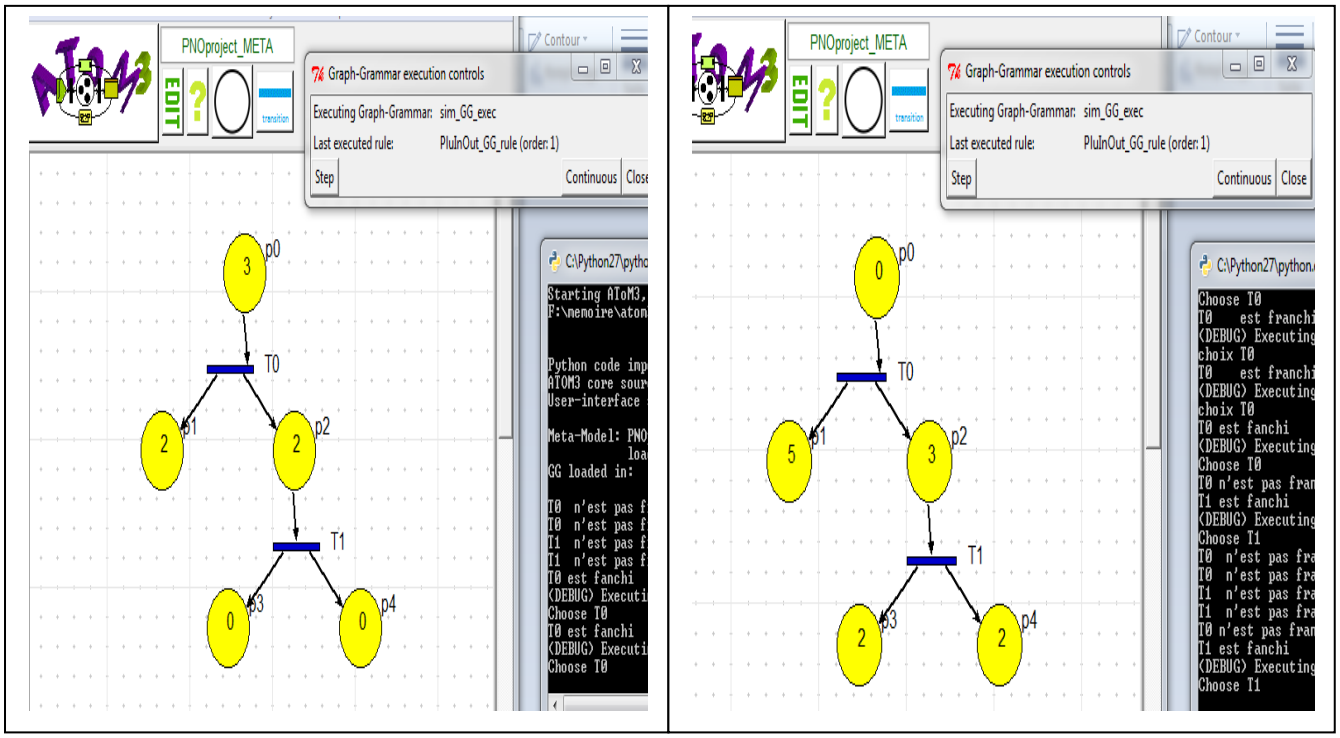



figure 3.21 franchissement de T0 et T1

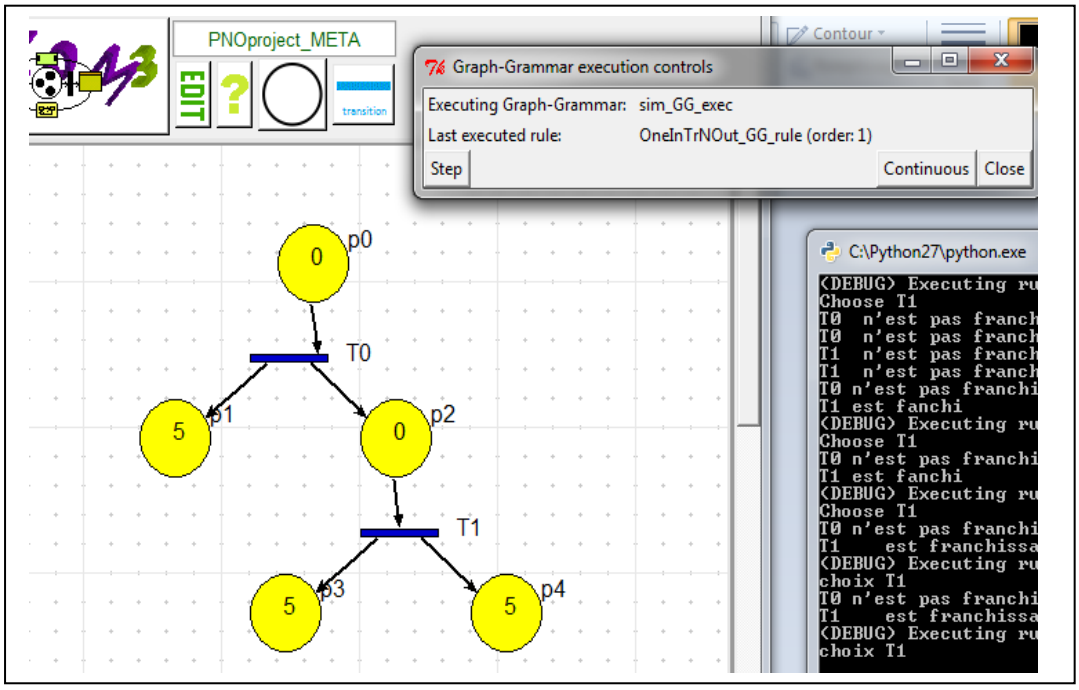



figure 3.22 exemple executé

## 8. Conclusion :

Nous avons présenté dans ce chapitre notre approche des transformations afin de faire une simulation d'un RDP, puis les résultats de notre implémentation sous la forme d'un ensemble d'interfaces utilisateur graphiques (GUI).

# Conclusion Générale

A travers notre projet, nous avons expliqué le but de notre projet qui est de fournir un outil de modélisation et de simulation avec les réseaux Ordinaire.

Nous avons introduit dans le premier chapitre, nous avons donné les définitions de base d'un réseau de Petri et ces Propriétés ainsi leurs méthodes d'analyse Le deuxième chapitre nous avons donné des détailles sur la transformation des graphe et on précise l'approche double pushout (DPO) et comme l'appliqué dans notre transformation sur les réseaux de pétri Nous avons également présenté l'outil de transformation de graphes AToM³ qui est l'outil qu'on l'utilise pour réaliser notre travail et le troisième chapitre c'est une explication détaillé de notre travail .

# Bibliographie

**[1]** Christophe Reutenauer.Aspects mathématiques des réseaux de Petri, volume 36.Masson Paris, 1989

**[2]** Murata ,Petri nets: « Properties, analysis and applications », Murata, Tadao,Proceedings of the IEEE,1989

**[4]** Charles C Poirier and Stephen E Reiter.La Supply Chain: Optimiser la chaînelogistique et le réseau interentreprises. Dunod, 2001

**[5]** R Bourdais, B Trouillet, and P Yim. Étude de noeuds ferroviaires à l'aide des réseaux de petri temporisés stochastiques. InProc. Conférence Internationale Francophoned'Automatique, CIFA, 2004.

**[6]** R Valk. Object petri nets: Using the nets-within-nets paradigm, advanced course on petri nets 2003 (j. desel, w. reisig, g. rozenberg, eds.), 3098.Appendix A: Proof of Theorem, 3, 2003.

**[9]** Gérard Scorletti and G Binet. Réseaux de pétri.cours EL401T2, master 1A mention,2006

**[10] WIKIPEDIA, l'encyclopédie libre, Home page: http://wikipedia.fr**

**[11]** Samba Diaw, Redouane Lbath, and Bernard Coulette. État de l'art sur le développement logiciel basé sur les transformations de modèles. Technique et Science Informatiques, 29(4-5):505–536, 2010

**[12]** ML Minsky. Matter, mind and models: Semantic information processing, ed. marvinminsky, 1968.

**[13]** Krzysztof Czarnecki and Simon Helsen. Feature-based survey of model transformation approaches.IBM systems journal, 45(3):621–645, 2006.

**[14]** Hartmut Ehrig, Claudia Ermel, Ulrike Golas, and Frank Hermann. Graph and model transformation. Monographs in Theoretical Computer Science. Springer, 2015

**[15]** Krzysztof Czarnecki and Simon Helsen. Classification of model transformation ap-proaches. InProceedings of the 2nd OOPSLA Workshop on Generative Techniquesin the Context of the Model Driven Architecture, volume 45, pages 1–17. USA, 2003.

**[16]** John L Pfaltz and Azriel Rosenfeld. Web grammars. InProceedings of the 1stinternational joint conference on Artificial intelligence, pages 609–619, 1969

**[17]** Anna Gerber, Michael Lawley, Kerry Raymond, Jim Steel, and Andrew Wood. Trans-formation: The missing link of mda. InInternational Conference on Graph Trans-formation, pages 90–105. Springer, 2002.

**[18]** Marc Andries, Gregor Engels, Annegret Habel, Berthold Hoffmann, Hans-Jörg Kre-owski, Sabine Kuske, Detlef Plump, Andy Schürr, and Gabriele Taentzer. Graphtransformation for specification and programming.Science of Computer program-ming, 34(1):1–54, 1999

**[19]** Grzegorz Rozenberg.Handbook of graph grammars and computing by graph transformation, volume 1. World scientific, 1997

**[21]** AToM3 home page: http://moncs.cs.mcgill.ca/MSDL/research/projects/ATOM3.html.