Chapitre 2 :

Le formalisme des réseaux de Petri

1.1. 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.

Nous introduirons dans ce chapitre la notion des réseaux de Petri, nous présenterons par la suite quelques une de leur propriétés et extensions. Nous finirons par une courte description de l'outil *CPNTools*.

1.2. Qualités et faiblesses des réseaux de Petri

En tant que formalisme de spécification, les réseaux de Petri présentent un certain nombre de qualités très importantes **[Bas00]** :

- Ils disposent d'une définition formelle : ce caractère formel permet de produire des spécifications exemptes d'ambiguïté ; chaque construction des modèles possède une sémantique parfaitement définie.

- Ils présentent un grand pouvoir d'expression : les RDP sont notamment très bien adaptés à décrire des comportements complexes, réactifs ou concurrents.

- Ils sont exécutables : les modèles peuvent être interprétés par un programme construit à partir de la définition formelle de la notation, ce qui permet de simuler le fonctionnement du système en cours de spécification. Le modélisateur profite ainsi d'une vision dynamique du système qu'il spécifie pour approfondir la compréhension qu'il a de son comportement.

- Ils disposent de nombreuses techniques de vérification automatique des propriétés des modèles. Il est possible de rechercher des propriétés génériques telles que le caractère borné, vivant ou réinitialisable, ou des propriétés spécifiques telles que l'existence d'invariants.

- Ils disposent d'une représentation graphique attrayante, qui accroît la lisibilité et facilite la compréhension des modèles. Cette représentation graphique est également très utile lors de l'exécution interactive des modèles, servant alors de « débogueur » graphique.

Toutefois, malgré ces qualités, il semble que les réseaux de Petri souffrent de certains reproches

- Leur manque de structuration : l'utilisation des RDPs produirait des modèles dont la taille croît rapidement avec la complexité du système, et qui deviennent rapidement impossibles à comprendre et à gérer.

- Leur difficulté à prendre en compte l'aspect « structure de données » : A l' origine, les RDPs étaient essentiellement destinés à décrire la structure de contrôle d'un système et son évolution dynamique. Les réseaux Place/Transition échouent en effet à décrire la

structure des données manipulées par le système, leur fermeture à la modélisation de l'environnement extérieur et la manière dont ces données peuvent influencer le comportement dynamique.

Ces reproches ont certainement été légitimes à une époque, mais ils ont été entendus par la communauté RDPs, qui a développé des techniques de structuration des réseaux, et qui a traité le problème de la prise en compte des données en développant la théorie des réseaux de Petri de haut niveau.

1.3. Qu'est-ce que les réseaux de Petri

Les réseaux de Petri sont définis comme étant un formalisme qui permet la description et l'analyse du comportement des systèmes concurrents, introduit par *Carl Adem Petri* en 1962. Les définitions concernant les **réseaux de Petri** portées sur deux aspects **[ScBi05]**:

- **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.

1.3.1. L'aspect structurel

a. 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 $\boldsymbol{P} \cap \boldsymbol{T} = \phi$ et ***W*** est la fonction définissant le poids porté par les arcs tel que $\boldsymbol{W}$: $((\boldsymbol{P}\times\boldsymbol{T}) \cup (\boldsymbol{T}\times\boldsymbol{P})) \rightarrow \mathbf{N}= \{0, 1, 2,\ldots\}$.

Le réseau *R* est fini si l'ensemble des places et des transitions est fini c.-à-d. $|P \cup T| \in \mathrm{N}$. Un réseau *R* = (*P, T, W*) est ordinaire si pour toute $(x,y) \in ((P\times T)\cup(T\times P))$: $W(x,y) \leq 1$ .Dans un réseau ordinaire la fonction *W* est remplacée par *F* où : $F \subseteq ((P\times T)\cup(T\times P))$ tel que $(x,y) \in F \Leftrightarrow W(x,y) \neq 0$ **[Mur89]**.

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

${}^{*}x$ représente l'ensemble des entrées de x : ${}^{*}x = \{y \in P \cup T | W(y,x) \neq 0\}$

x^{*} représente l'ensemble des sorties de x : $x^{*} = \{y \in P \cup T | W(x,y) \neq 0\}$

Remarque : si ${}^{*}x = \phi$, x est dite source, si $x^{*} = \phi$, x est dite puits.

b. Représentation d'un réseau de Petri

b.1. 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 **[ScBi05]**.

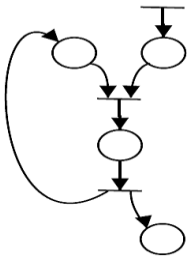

Figure 0.1 : Réseau de Petri simple

b.2. 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 $\boldsymbol{W}$(fonction de poids) par des matrices.

Définition

Soit Un réseau de Petri ***R*= (*P, T, W*)** avec ***P*= {$\mathbf{p_1}$, $\mathbf{p_2}$, …, $\mathbf{p_m}$} et *T*= {$\mathbf{t_1}$, $\mathbf{t_2}$, …,$\mathbf{t_n}$}**, on appelle matrice des pré conditions *pré* la matrice m × n à coefficients dans **N** tel que ***pré*(i,j)= *W*($\mathbf{p_i}$, $\mathbf{t_j}$)**, elle indique le nombre de marque que doit contenir la place $\mathbf{p_i}$ pour que la transition $\mathbf{t_j}$ devienne franchissable, de la même manière on définit la matrice des post conditions ***post*** la matrice **n × m** tel que ***post*(i,j)= *W*($\mathbf{t_j}$, $\mathbf{p_i}$)** contient le nombre de marques déposées dans $\mathbf{p_i}$ lors du franchissement de la transition $\mathbf{t_j}$. 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.)**[Poi01]**

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

$$M'(p) = M(p) + C(p, t).$$

c. Représentation d'un RdP marqué

Un réseau de Petri marqué est le couple $N = <R, M>$où :

- *R* est un réseau de Petri
- *M* est une application de marquage
- $M : P \rightarrow N$

M(p) est le nombre de marques (jetons) contenus dans la place p. Le marquage d'un réseau de Petri est une opération qui consiste à assigner des jetons dans les places.

On appelle marquage M d'un Réseau de Petri le vecteur du nombre de marques dans chaque place : la $I^{ème}$ composante correspond au nombre de marques dans la $I^{ème}$ place. Il indique à un instant donné l´état du RdP. **[Ben01]**

Exemple

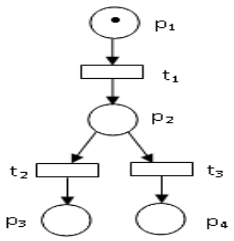

Figure 0.2 : Réseau de Petri marqué

Pour le réseau de la *Figure 2.2*

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

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

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 :

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

Le marquage du RdP présente *Figure 2.2* est donné par :

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

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

1.3.2. 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. **[ScBi05]**

a. 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 $M : P \rightarrow \mathbf{N}$.

Un réseau de Petri **marqué** est dénoté par $\Sigma = (P, T, W, M_0)$ 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. [Bou04]**

b. 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 $t \in T$ 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 $p \in {}^{*}t$ 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: $M(\mathbf{p}) \geq W(\mathbf{p}, \mathbf{t}) \quad \forall \ \mathbf{p} \in P$.

Une règle de franchissement est définie par $M'(\mathbf{p}) = M(\mathbf{p}) - W(\mathbf{p}, \mathbf{t}) + W(\mathbf{t}, \mathbf{p})$ pour tout $\mathbf{p} \in P$, ce qui veut dire que lorsque la transition **t** est franchi à partir d'un marquage M, il faut saisir $W(\mathbf{p}, \mathbf{t})$ jetons à partir de chaque place d'entrée à la transition **t** et déposer $W(t, p)$ jetons dans chaque place de sortie de la transition **t** ce qui permet de produire un nouveau marquage M'.

Le franchissement d'une transition **t** dénoté par $M[\mathbf{t}\rangle M'$ est dite l'**occurrence** de **t**. On dit que deux transitions $\mathbf{t_1}, \mathbf{t_2}$ (pas certainement distinctes) sont franchies en concurrence par un marquage M si et seulement si $\mathbf{M}(\mathbf{p}) \geq W(\mathbf{p}, \mathbf{t_1}) + W(\mathbf{p}, \mathbf{t_2})$ pour toute $\mathbf{p} \in P$.

Cette vision de l'exécution concurrente de deux transitions dans un RdP est contradictoire avec celle qui impose que deux occurrence de transition sont parallèles si et seulement si : elles sont causalement indépendantes et n'ont pas une relation de conflit entre eux. Deux occurrences sont en conflit si l'un des deux peut avoir lieu mais pas toutes les deux. **[Val07]**

c. L'exécution d'un réseau de Petri

A) Exécution séquentielle

- **Séquence de franchissement**

Une séquence de franchissement « s » est une suite de transitions (t_1, t_2, ..., t_n) qui permet, à partir d'un marquage « M », de passer au marquage « M' » par le franchissement successif des transitions définissant la séquence.

- **Marquage accessible**

Le marquage d'un Réseau de Petri à un instant donné est une vectrice colonne dont la valeur de la Ième composante est le nombre de marques dans la place P_i à cet instant.

Le franchissement d'une transition conduit à un changement du marquage. Le passage du marquage M_k au marquage M_l par franchissement de la transition T_j est noté : M_k (T_j> M_l. Le nombre de marques dans la place P_i pour le marquage M_k est noté $M_k(P_i)$. 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 M_0est l'ensemble des marquages obtenus à partir de M_0par franchissements successifs d'une ou plusieurs transition(s). Cet ensemble est noté $A(R ; M_0)$. **[Mur89]**

- **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 $A(R,M_0)$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 $\mathbf{M_i}$ $\mathbf{[t\rangle M_j}$ Les arcs du graphe sont étiqueté par les transitions correspondantes.

Exemple :

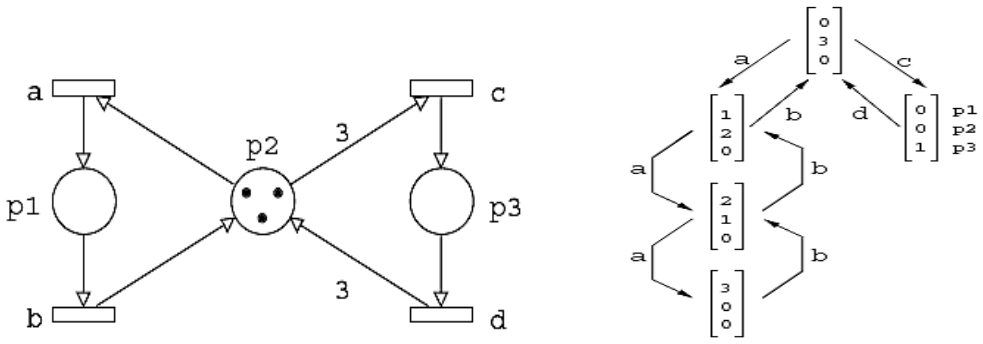

Figure 0.3 : Graphe de Marquage

- **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 $\sigma = M_0\ t_1\ M_1\ t_2 \ldots$tel que $M_{i-1}[t_i\rangle\ M_i$. Une séquence $t_1\ t_2\ldots$ est une séquence de transitions (commencée par le marquage M) si et seulement si il existe une séquence d'occurrence $M_0 t_1 M_1 \ldots$ avec $M = M_0$. Si la séquence finie $t_1\ t_2 \ldots t_n$ conduit à un nouveau marquage M'à partir du marquage M, on écrit $M[t_1\ t_2 \ldots t_n\rangle\ M'$ ou simplement $M[t_1 t_2 \ldots t_n\rangle$ si on ne veut pas spécifier le marquage résultat.**[Mur89]**

d. 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.

1.4. Réseaux particuliers

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

1.4.1. Graphe d'états

Dans ce cas chaque transition ne dispose que d'une place en entrée et une place en sortie.

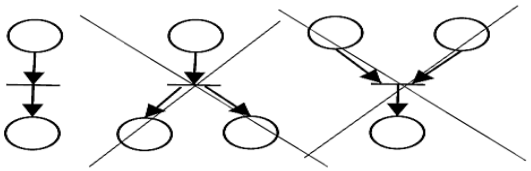

Figure 0.4 : Graphe d'états ou pas

1.4.2. Les réseaux sans conflits

Un réseau de Petri est dit sans conflit si et seulement si toute place a *au plus une* transition de sortie (voir *Figure 2.5*). Un conflit (structurel) correspond à l'existence d'une place P_i qui a *au moins deux* transitions de sortie T_j, T_k, etc... Notation $< Pi, \{T_j, T_k, \ldots\}>$. Sur le RdP de droite de la *Figure 2.5*, on a le conflit $< P1, \{T2, T3\}>$. Quand la place $P1$contient une marque, les transitions $T1$ et $T2$ sont franchissables. Seule une des deux transitions peut être franchie : il est nécessaire de prendre une décision pour savoir laquelle des deux le sera effectivement. L'absence ou la présence d'un conflit est une propriété importante d'un réseau de Petri.

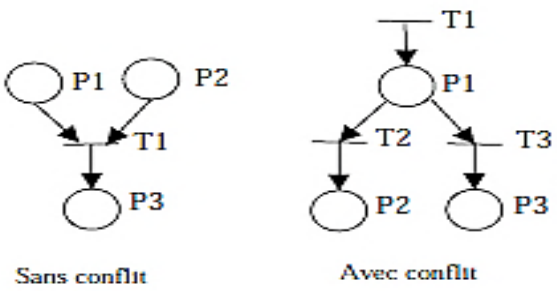

Figure 0.5 : Conflit ou pas

1.4.3. Les réseaux purs

Un réseau de Petri est dit pur si et seulement s'il n'existe pas de transition ayant une place d'entrée qui est aussi place de sortie. Le RdP représente *Figure 2.6* n'est pas pur car la place $P3$ est place d'entrée et place de sortie de la transition $T1$. On parle alors de RdP impur

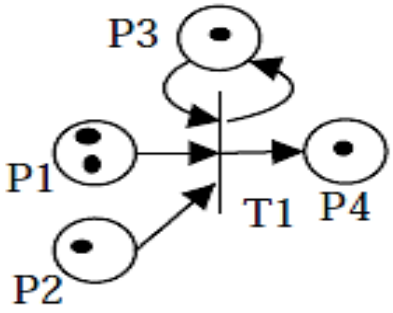

Figure 0.6 : RDP impur

1.5. Propriétés des RdP

1.5.1. Réseau K-borné

Une place Pi est bornée pour un marquage initial M0 si pour tout marquage accessible à partir de M0, le nombre de marques dans Pi reste borné. Elle est dite **k-bornée** si le nombre de marques dans Pi est toujours inférieur ou égal à k. Un RdP marqué est (k) borné si toutes ses places sont (k) bornées.

Un RdP marqué peut ne pas être borné : sur l'exemple représente *Figure 2.7*, la transition *T*1 admet la place *P*1 comme unique place d'entrée. La place *P*1 a une marque : la transition *T*1 est franchissable. Comme *P*1 est aussi place de sortie de *T*1, le franchissement de *T*1 ne change pas le marquage de *P*1. La transition *T*1 est donc franchissable en permanence et peut donc être franchie un nombre de fois infini. Chaque franchissement de *T*1 ajoutant une marque dans la place *P*2, le marquage de celle-ci peut donc tendre vers l'infini.

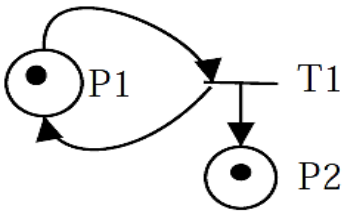

Figure 0.7 : RdP non borné

Définition : Un RdP marqué est sauf ou binaire pour un marquage initial M_0 s'il est 1-borné.

1.5.2. Réseau vivant

L'évolution du marquage d'un RdP se fait par franchissement de transitions. Lorsqu'au cours de son évolution, certaines transitions ne sont jamais franchies, cela indique que

l'évènement associé à la transition ne se produit pas et que le marquage d'une partie du RdP n''évolue pas. Cela indique que le sous-système modélisé par cette partie-là ne fonctionnera pas. Il y a donc un problème au niveau de la conception du système. L'idée est d'être capable de d´détecter systématiquement ce phénomène par l'analyse de propriétés du modèle RdP du système afin de disposer d'un outil d'aide à la conception des systèmes.

Définition : Une transition T_j est vivante pour un marquage initial M_0 si pour tout marquage accessible M_k, il existe une séquence de franchissements à partir de M_k contenant T_j :

$$\forall M_k \in {}^*M_0, \quad \exists S, \quad M_k \mid S > \; et \; S = \ldots T_j \ldots .$$

Si une transition T_j est vivante alors, à tout instant, on sait que T_j peut être franchie dans le futur. Dans le cas d'un RdP modélisant un système fonctionnant en permanence, si une transition n'est pas vivante et si une fonction du système est associée au franchissement de cette transition, cela veut dire qu'à partir d'un certain instant, cette fonction ne sera plus disponible dans le futur, ce qui peut traduire une erreur ou une panne.

Exemples : Les transitions *T*1 et *T*2 du RdP marqué *Figure 2.8* sont vivantes

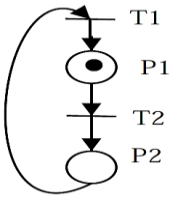

Figure 0.8 : RdP vivant

1.5.3. Réseau réinitialisable

Soit un réseau de Petri marqué < R;M_0 > et soit A(R;M_0) l'ensemble de ses marquages accessibles. Ce réseau marqué est réinitialisable si et seulement si :

$$\forall M \in \mathcal{A}(\mathcal{R}; M_0)\,, M \neq M_0\,, \quad \exists s \;\text{ tel que }\; M \xrightarrow{s} M_0$$

On dit également que $\boldsymbol{M_0}$ est un état d'accueil (en anglais "*home state*") pour le réseau de Petri marqué.

Soit un réseau de Petri marqué <$\boldsymbol{R}$;$\boldsymbol{M_0}$>, s'il est réinitialisable et que toute ses transitions sont **quasi-vivantes** ou **vivantes**, alors il est vivant

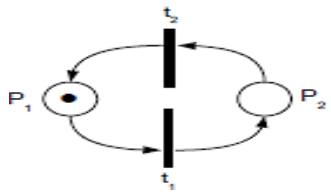

Figure 0.9 : Réseau réinitialisable

1.6. Les méthodes d'analyse des RdP

Les méthodes d'analyse des RdP peuvent être classées en trois groupes:

- Méthode d'arbre de couverture
- Approche d'équations matricielles
- Technique de réduction et de décomposition

1.6.1. Méthode d'arbre de couverture

L'idée la plus naturelle pour étudier les propriétés d'un RdP est de construire le graphe de tous les marquages accessibles. Le graphe des marquages accessibles est un graphe dont chaque sommet correspond à un marquage accessible et dont chaque arc correspond au franchissement d'une transition permettant de passer d'un marquage à l'autre.

Pour un RdP marqué (**R**, **M0**), à partir d'un marquage initial M_0 on peut obtenir autant de nouveau marquage qu'il existe de transition franchissable. A partir de chaque nouveau marquage on peut accéder à d'autres nouveaux marquages.

Le résultat de ce processus est un arbre dont chaque nœud est un marquage accessible et chaque arc est une transition franchissable qui transforme un marquage à un autre

L'arbre de couverture est une méthode alternative. Le mécanisme de construction est le même que pour le graphe des marquages accessibles. A ceci prêt que pour chaque nouveau marquage (nœud du graphe) ajouté, on vérifie s'il n'est pas supérieur à un marquage déjà présent sur au moins une séquence entre M0 et le nouveau marquage. Si tel est le cas tous les marquages de place supérieurs sont remplacés par ω. Ce symbole matérialise le fait que la place en question peut contenir autant de jetons que souhaité (elle est donc non bornée).

Dans la suite, les places non bornées le demeurent naturellement et ceci quelles que soient les transitions franchies, ainsi le symbole ω ne disparaît jamais. Cette méthode produit le graphe de couverture, un graphe fini dans tous les cas. **[Ben01]**

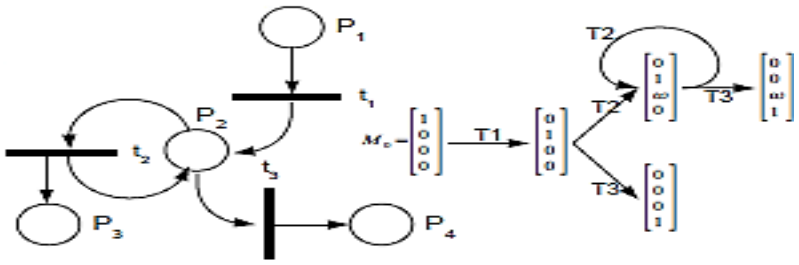

Figure 0.10 : RDP et leur d'arbre de couverture

1.6.2. Approche d'équations matricielles

L'analyse par équations matriciel (algèbre linéaire) permet d'étudier des propriétés d'un réseau (caractère borné, vivacité) indépendamment d'un marquage initial. De ce fait, on parlera de propriétés structurelles du réseau. Par exemple, on pourra dire qu'un réseau est structurellement borné s'il est borné pour tout marquage initial fini. De la même façon, si pour tout marquage initial, le réseau est vivant, on dira que le réseau est structurellement vivant.

1.6.3. Technique de réduction et de décomposition

L'analyse des RDPs par réduction permet d'obtenir à partir d'un RdP marqué, un RdP marqué plus simple, c'est-à-dire avec un nombre réduit de places et un nombre réduit de transitions ceci en appliquant un ensemble de règles dites règles de réduction.

1.7. Réseaux de Petri de haut niveau

Dans la pratique, on est amené à modéliser toute sorte de système, à savoir les protocoles de communication, les systèmes de production, les systèmes réactifs, les systèmes temps réel … Cette variété de systèmes a poussé les chercheurs à étendre la théorie des RdPs en introduisant beaucoup de concepts relatifs à chaque domaine d'application. Ces efforts ont donné naissance aux RDPs de haut niveau tels que les RDPs, colorés, hiérarchiques, temporisés et autres. Une liste plus complète des extensions RDP est détaillée dans **[Val07].**

1.7.1. Les réseaux de Petri coloré

a. Définitions formelles pour les réseaux de Petri colorés

La modélisation d'un système réel peut mener à des réseaux de Petri de taille trop importante rendant leur manipulation et/ou leur analyse difficile. La question est alors de modifier (étendre) la modélisation par RdP de façon à obtenir des modèles RdP de plus petite taille. Les RDPs colorés. Ils sont importants pour la modélisation de systèmes de production (au sens large).

1) **Association de couleur**

Dans le but de différencier les jetons, on leur associe des couleurs ou des entiers ou encore un ensemble d'étiquette. On associe à chaque place l'ensemble des couleurs des jetons qui peuvent y séjourner et à chaque transition l'ensemble de couleurs pour laquelle elle est franchissable **[Tvd05].**

2) Association de fonctions aux arcs

Dans un RdP coloré les arcs ne seront plus étiquetés par des entiers mais par des fonctions, qui désigneront les actions à effectuer à chaque franchissement. A chaque arc correspond une fonction qui à chaque couleur de transition associe un vecteur ligne d'entiers positifs qui décrit le nombre de jetons de chaque couleur qui sont retirés ou produits à chaque franchissement **[Lks98].**

3) Définition informelle

- Chaque place p est caractérisée par un domaine de couleur C(p),
- Un jeton d'une place p est un élément de C(p),
- Chaque transition t est caractérisée par un domaine de couleur C(t),
- Le domaine de couleur d'une transition caractérise la signature de cette transition,
- Les fonctions de couleur sur les arcs déterminent les instances de jetons nécessaires, consommées et produites lors du franchissement d'une transition.

4) Définition formelle

D'abord on définit des multi-ensembles.

Définition :

Un multi-ensemble **m**, sur un ensemble non vide **S**, est une fonction $\mathbf{m}\epsilon[S \rightarrow N]$, représenté comme somme formelle **[Tvd05]** :

$$\sum_{s \in S} m(s)`s$$

On dénote **SMS** l'ensemble de tous les multi-ensembles sur S. Les nombres entiers non négatifs **{m(s) : s ϵ S}** sont les coefficients du multi-ensemble. **sϵm** si et seulement si $\mathbf{m}(\mathbf{s}) \neq \mathbf{0}$.

Pour donner une définition abstraite du réseau de Petri coloré, on suppose que les concepts suivantes soient claires et on ne redéfinit pas, on donne seulement des descriptions en bref :

- Les éléments d'un type, *T*. L'ensemble de tous les éléments dans *T*est dénoté par le nom *T* lui-même de type ;
- Le type d'une variable, *v* - dénoté par *Type(v)* ;
- Le type d'une expression, *expr* - dénoté par *Type(expr)* ;
- L'ensemble des variables dans une expression, *expr* - dénoté par *Var(expr)* ;
- Une association d'un ensemble de variables, V - assignant chaque variable *v*ϵ V à un élément *b(v) ϵ Type(v)* ;
- La valeur obtenue en évaluant une expression, *expr*, dans une association, *b* - dénoté par *expr* < *b* >.*Var(expr)* est appliqué d'abord pour obtenir un sous-ensemble des variables de *b*, puis l'évaluation est exécutée en remplaçant chaque variable *v ϵVar(expr)* à la valeur *b(v) ϵ Type(v)* qui est déterminée par la association.
- Une expression sans variables s'appelle une expression fermée. Elle peut être évaluée dans toutes les associations, et toutes ces évaluations donnent la même valeur.

Définition [Lks98] :

Un réseau coloré est un tuple CPN = (∑, P, T, A, N, C, G, E, I) où :

- ∑ est un ensemble fini de types non-vides, appelé des ensembles de couleur ;
- **P** est un ensemble fini de places ;
- **T** est un ensemble fini de transitions ;
- **A** est un ensemble fini d'arcs tel que :

$$P \cap T = P \cap A = T \cap A = \emptyset$$

- **N** est une fonction de nœud. Il est défini de A dans P x T ∪ T x P ;
- **C** est une fonction de couleur. Il est défini de P dans∑;
- **G** est une fonction de garde. Il est défini de T dans les expressions tel que :

$$\forall t \in T : [Type(G(t)) = Boolean \wedge Type(Var(G(t))) \subseteq \Sigma]$$

- **E** est une fonction d'expression d'arc. Il est défini de A dans les expressions tel que :

$$\forall a \in A : [Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$$

Où *p(a)* est la place de *N(a)* ;

- **I** est une fonction d'initialisation. Elle est définie de P dans les expressions fermées tels que :

$$\forall p \in P : [Type(I(p)) = C(p)_{MS}].$$

Définition :

Une association d'une transition *t* est une fonction *b* définie sur *Var(t)*, tel que **[Lks98]** :

- $\forall v \in Var(t) : b(v) \in Type(v)$
- G(t) < b >= vrai

On dénote B(t) l'ensemble de tous les associations pour t.

On écrit une association de transition t sous la forme <*v1 = c1, v2 =c2,……,vn = cn*>où

Var(t) = {v₁, v₂,…, vₙ} etc. $i \in Var(v_i)$. Et un élément de l'association est une paire *(t; b)* où $t \epsilon T$ et $b \epsilon B(t)$.

Définition :

Un pas est un multi-ensemble des éléments de l'association. Un pas Y est accessible dans un marquage M si et seulement si la propriété suivante est satisfait :

$$\forall p \in P \sum_{(t,b)\in Y} E(p,t) < b > \leqslant M(p)$$

Quand un pas Y est accessible dans le marquage M1, s'il se produit, M1change à un marquage M2, défini par :

$$\forall p \in P : M_2(p) = (M_1(p) - \sum_{(t,b)\in Y} E(p,t) < b >) + \sum_{(t,b)\in Y} E(t,p) < b >$$

La première somme s'appelle le jeton supprimé tandis que la seconde s'appelle le jeton ajouté. D'ailleurs on dit que le M2 est directement accessible de M1 par l'occurrence du pas Y, qu'on dénote également : M1[Y > M2. Une séquence d'occurrence est une séquence des marquages et des pas :

$$M_1[Y_1 > M_2[Y_2 > M_3 \cdots M_n[Y_n > M_{n+1}$$

Tel que $\forall i \in \{1, \cdots, n\} : M_i[Y_i > M_{i+1}$.On utilise [M > pour dénoter l'ensemble de tous les marquages qui sont accessibles de M.

Exemple [ScBi05] :

- Soit x1 ∈ C1, x2 ∈ C2
- t est franchissable pour (x1, x2) ssi P1 contient au moins un jeton de couleur x1 (X1(x1, x2) = x1). P2 contient au moins un jeton de couleur x2
- Si on franchit t pour (x1, x2) alors un jeton de couleur x1est retiré de P1, un jeton de couleur x2 est retiré de P2, un jeton de couleur < x1, x2> est produit dans P3: <X1, X2> (x1, x2) = < x1, x2>

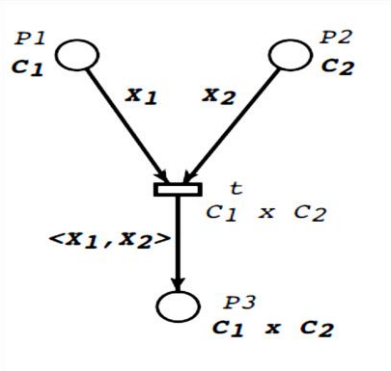

Figure 0.11 : Dynamique d'un RdPC

1.7.2. Les réseaux de Petri à Objet

Les réseaux de Petri objet (OPN) étendent le formalisme des réseaux de Petri colorés avec une intégration complète des propriétés orientées objet y compris l'héritage le polymorphisme et la liaison dynamique, l'orientation objet fournit des primitives de structuration puissant permettant la modélisation des systèmes complexes **[Val07]**

1.7.3. Les réseaux de Petri temporels

Les réseaux de Petri temporels sont obtenus depuis des réseaux de Petri en associant des dates min et max aux transitions.

Supposons qu'une transition t est devenue sensibilisée pour la dernière fois à l'instant x, alors elle ne peut l'être encore qu'après l'instant x + min(t) et avant l'instant x + max(t), sauf si le tir d'une autre transition a désensibilisé t avant que celle-ci ne soit tirée. Le tir des transitions est de durée nulle. Les réseaux de Petri temporels expriment nativement des spécifications en «délai». Ils peuvent aussi exprimer des spécifications en «durées». Leur domaine d'application est donc large. **[Ben01]**

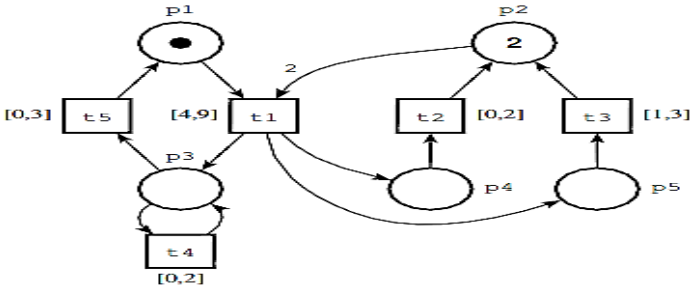

Figure 0.12 : Exemple de réseau de Petri temporel [Ben01]

1.7.4. Les réseaux de Petri hiérarchiques

Les RDPs hiérarchiques (RdPH) permettent de modéliser un système de façon modulaire comme un langage de programmation. Un RdPH se compose de plusieurs modules (pages), où chaque module représente un sous réseau de Petri. Un module dit principal référence tous les autres modules secondaires. Le référencement d'un module se fait au moyen d'une transition de substitution portant le nom du module en question. L'utilisation des RdPHs est préconisée pour la modélisation des problèmes de grande taille. **[Ben01]**

1.8. Techniques de vérification formelle

Pour avoir des systèmes (hard et soft) ables et sûrs ils doivent nécessairement être validé Les principales méthodes de validation des systèmes sont la simulation, les tests, vérification formelle basée sur les méthodes déductives et les méthodes basées sur le model-checking.

Les phases de vérification et de validation permettent de contrôler que le système satisfait les propriétés attendues. Il en résulte que tout problème lié au comportement qualitatif (Sûreté, équité, absence de blocage, etc...), comme au comportement quantitatif (perte de messages, vitesse moyenne de transmission, etc...) est détecté et corrigé très tôt dans le cycle de développement.

Pour prévenir ainsi d'éventuels dysfonctionnements, un ensemble de techniques ont été développées. Parmi celles-ci, les tests et la simulation sont les plus connus et probablement les plus largement utilisés.

Une autre technique utilisée pour assurer la validation de systèmes est la méthode formelle

1.8.1. Vérification formelle

La vérification formelle se base sur des modèles formels. Elle consiste à explorer tous les états du système. Pour pouvoir être réalisée, la vérification impose une description formelle du système, ainsi qu'une spécification formelle des propriétés.

De nombreux formalismes de spécification dédiés aux systèmes concurrents ont été proposés en littérature tels que les automates communicants, Systèmes de transition, algèbres de processus **CCS**, **CSP**, **LOTOS**, **Estelle**, **Promela**, etc... .

Parmi les méthodes de vérification formelle, nous pouvons distinguer les trois approches suivantes :

- Les vérifications basées sur les **preuves de théorèmes**.
- Les vérifications basées sur les **équivalences**.
- Les vérifications basées sur le **model-checking**.

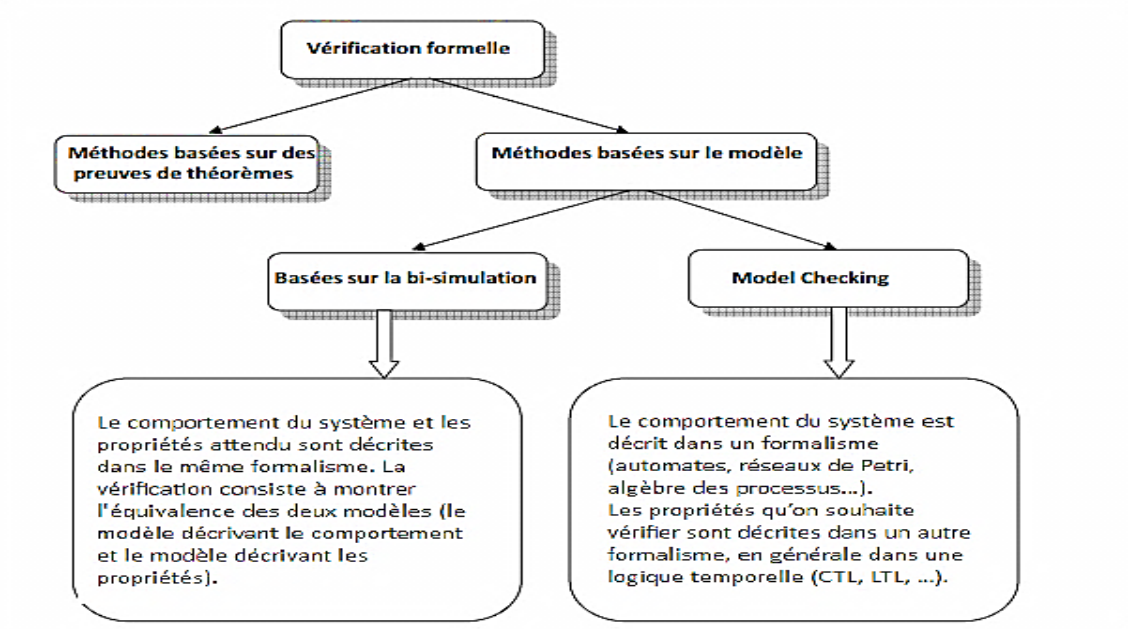

Figure 0.13 : Diagramme de Vérification Formelle
[ChKa]

a. Preuves de théorèmes

Poser un ensemble d'axiomes, souvent donnés par le concepteur, puis à prouver un ensemble d'assertions déterminant ainsi la conformité du système. Cette preuve est faite plus ou moins manuellement.

L'aide apportée par les outils tels que les démonstrateurs de théorèmes n'évite pas totalement la nécessité de l'intervention humaine. Ce type de vérification est rarement employé, il est utilisé essentiellement dans le domaine des spécifications algébriques et logiques.

b. Équivalences

La deuxième approche de vérification formelle consiste à vérifier l'équivalence (le plus souvent par rapport à une relation de bisimulation) entre le modèle de description de l'implémentation et une spécification qui décrit ce que l'on attend de cette implémentation, les deux utilisant le même formalisme de description. Si les deux modèles sont équivalents cela prouve que l'implémentation est conforme à la conception.

c. model-checking

Il est basées sur les modèles, permet d'une vérification simple et efficace et elle est complètement automatisables. La vérification basée sur les modèles ou Model-checking est surtout applicable pour les systèmes ayant un espace d'états fini.

Les algorithmes de vérification dans cette méthode utilisent l'ensemble des états que le système peut atteindre pour prouver la satisfaction ou la non-satisfaction des propriétés.

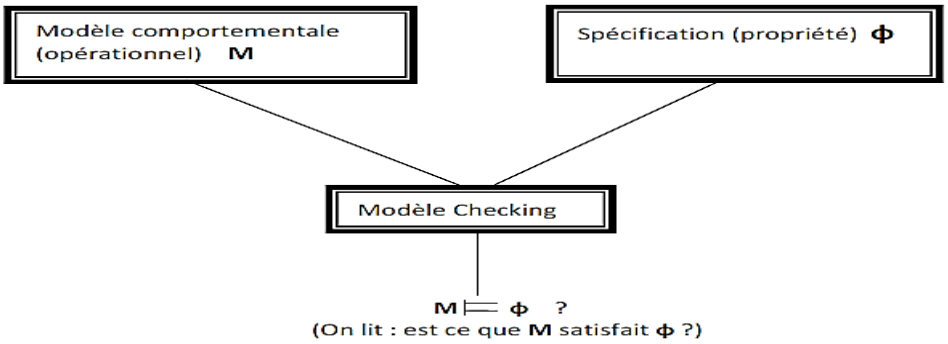

Figure 0.14: Diagramme de modèle Checking
[ChKa]

1.9. Outils de modélisation des RDPS

L'aspect formel des RdPs a encouragé les développeurs à mettre au point une multitude d'outils de simulation et de vérification des RdPs selon la technique de vérification de modèle. On propose de faire un résumé sur les fonctionnalités qu'offre chacun d'eux Le tableau ci-dessous les classe par rapport aux points suivants **[Ben01]** :

- ✓ Présence d'un environnement graphique d'édition des RDPs.
- ✓ Possibilité de Simulation du RDP.
- ✓ Possibilité d'analyse des propriétés génériques du RDP.

- ✓ Possibilité de vérification des contraintes CTL (logique temporelle arborescente)
- ✓ et LTL (logique temporelle linéaire).
- ✓ Possibilité de supporter le format d'échange XML.
- ✓ Possibilité de supporter les RDP hiérarchiques.

Outil	Interface Graphique	Simulation	Analyse	CTL	LTL	XML	RDP hiérarchique	TOTAL (oui)
CPNTools	Oui	Oui	Oui	oui	non	Oui	oui	6
CPNAMI	Oui	Oui	Oui	Oui	Oui	Non	non	5
PROD	Non	Non	Oui	Oui	Oui	Non	non	3
JARP	Oui	Non	Oui	Non	Non	Oui	non	3
Maria	Non	Non	Oui	Oui	Oui	Non	non	3
LoLa	Non	Non	Oui	Oui	Non	Non	non	2
Petri Net Kernel	Oui	Non	Non	Non	Non	Oui	non	2
Great SPN	Oui	Non	Oui	Non	Non	Non	Non	2
INA	Non	Oui	Oui	Oui	Non	Non	Non	2
OPMSE	oui	Oui	Non	Non	No	Non	non	2

Table 0.1 : Les outils de modélisation des réseaux de Petri [Ben01]

Selon le tableau comparatif des outils de vérification des RDPs, *CPNTools* pressente de meilleures caractéristiques techniques (on va voir plus loin).

L'outil CPNTOOLS

Le développement des réseaux colorés a été conduit par le désir de développer un langage fort de modélisation industrielle. La section 2.7.1 nous indique les définitions trop formelles. Elles conviennent aux mathématiciens, mais pas aux praticiens. Donc pour les applications industrielles, il faut avoir un outil efficace, qui nous aider à éviter les symboles formels. Les idées sont même que dans les langages, on peut programmer mais on n'a pas besoin de connaitre la syntaxe (la grammaire) formelles compliques de langage.

CPNTools est conçu pour ce but. Il se compose : CPN editor nous permet d'éditer le réseau coloré et vérifier ses syntaxes ; CPN simulator nous permet de simuler le comportement de réseau (les règles de franchissement, les transitions franchissables, etc.), CPN state space tool supporte la vérification (L'espace d'état).

Dans *CPNTools* on combine la capacité des réseaux de pétri colorés et la capacité d'un langage (le langage choisi est le langage fonctionnel Standard ML). Le RdPC fournissent les primitifs pour décrire les processus concourants, tandis que le langage fournit les primitifs pour définir des types de données (ensemble de couleur) et des manipulations des données (expression d'arcs, gardes, etc.).**[Kje]**

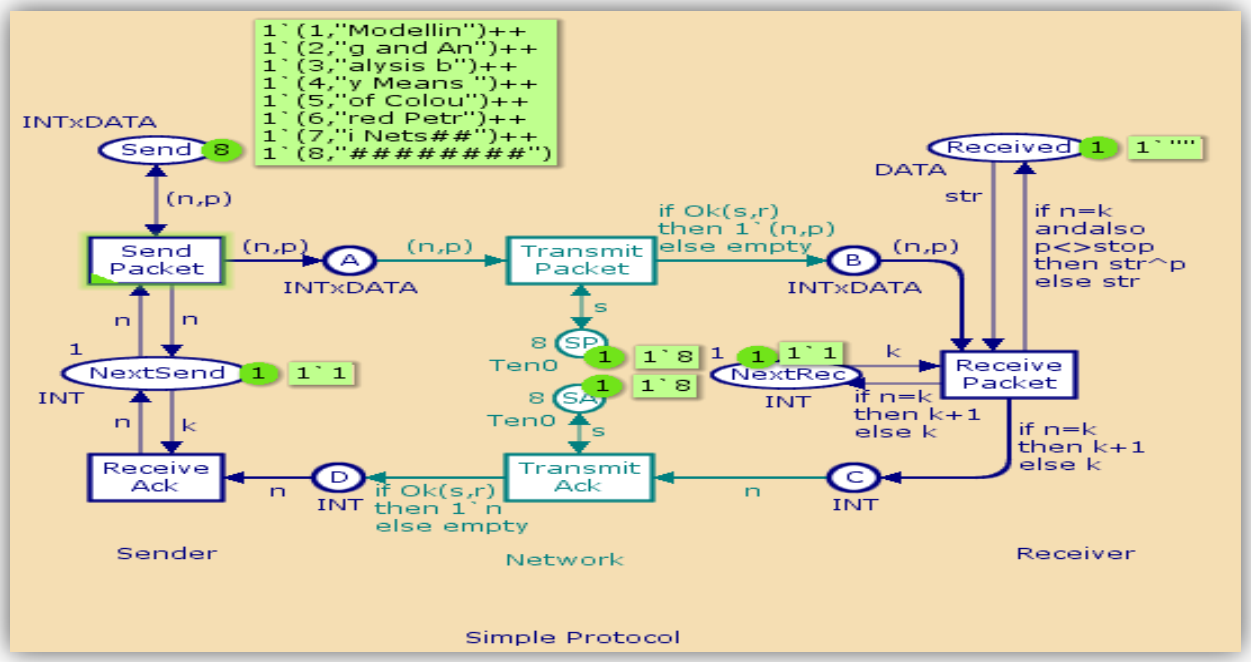

Figure 0.15 : Le réseau de Petri modélise le Protocol simple(CPNTools)

1.10. 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.

On a représenté brièvement 2 types de réseau : réseau de Place/Transition et réseau coloré. Ils 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 Colorés* sont plus raisonnables.

Les réseaux colorés ont une définition très compliquée, c'est très difficile pour les praticiens de maitriser la syntaxe du réseau. Mais heureusement, il existe des outils pour éditer les réseaux colorés, vérifier sa syntaxe, vérifier automatiquement des propriétés comportementales du réseau, etc. Donc, les praticiens peuvent utiliser le réseau coloré mais ils ne doivent pas apprendre par cœur leurs syntaxes. Il est similaire dans le cas du langage, on peut écrire des programmes mais on ne doit pas connaitre toutes les grammaires du langage.

La relation entre le réseau Place/Transition et le réseau coloré peut être considéré comme relation entre le langage à bas niveau (le langage de machine, le langage Assembly, etc.) et le langage à haut niveau (Java, C, Pascal, etc.). Ils ont même capacité de la programmation mais pour les grandes applications, qui conviennent plus aux langages de haut niveau que ceux de bas niveau.

Nous expliquerons dans le prochain chapitre la transformation de modèle et la relation entre le formalisme des réseaux Petri coloré et celui des diagrammes d'activités.