جامعة هواري بومدين
للعلوم والتكنولوجيا
USTHB

Rapport de Recherche

Approche de décomposition multicouche de réseaux de Petri stochastiques et spécifications algébriques

Nabila SALMI, Malika IOUALALEN

LSI-TR-1604 **Octobre 2004**

Laboratoire LSI

FACULTE ELECTRONIQUE & INFORMATIQUE
Département informatique
El Alia BP n°32 Bab Ezzouar 16111 Alger.
Tél / Fax : 213 (0) 21 24 79 17 - 24 76 07

Approche de décomposition multicouche de réseaux de Petri stochastiques et spécifications algébriques

Nabila SALMI[1] Malika IOUALALEN[2]

LSI-Département Informatique, Faculté d'Informatique & Electronique, USTHB
El Alia BP n°32, Bab Ezzouar, Alger, Algérie.

[1] Email: salmi@lsi-usthb.dz
[2] Email: ioualalen@lsi-usthb.dz

Résumé. L'analyse des performances des systèmes parallèles et complexes par réseaux de Petri stochastiques (RDPS) s'avère difficile à effectuer, à cause de l'explosion de l'espace d'états à analyser. A fin d'éviter ce problème, nous proposons une nouvelle approche de décomposition qui décompose le système en un ensemble de sous-systèmes, tout en exprimant le système global en une expression algébrique reliant ses propres composants. Cette approche tente de retrouver les sous réseaux composant le réseau global, et déduire les relations liant ces sous réseaux. Le résultat fourni par cette décomposition consistera alors en une expression algébrique représentant le RDPS, composée des sous réseaux identifiés, liés par des opérateurs à définir.
Mots-clé. Réseaux de Petri Stochastiques, Evaluation des performances, Explosion d'états, décomposition, expression algébrique, composition synchrone, composition asynchrone.

1. Introduction

Les réseaux de Petri Stochastiques (RDPS) ont été largement utilises pour l'évaluation des performances des systèmes informatiques, des réseaux de télécommunication et des systèmes de production. Ils sont considérés comme un modèle puissant qui permet d'exprimer les caractéristiques essentielles d'un système, et permet, grâce à sa nature probabiliste, d'effectuer une analyse quantitative des performances. En effet, le modèle RDPS a prouvé son efficacité grâce aux résultats analytiques inspirés des processus stochastiques, en particulier des chaînes de Markov. Toutefois, l'analyse d'un RDPS entraîne souvent l'apparition du problème d'explosion combinatoire dans l'espace d'états à analyser. Ce problème rend l'analyse difficile, voire impossible dans certains cas.

Afin de pallier ou éviter ce problème, des travaux ont été proposés dans la littérature. Ils peuvent être classés comme suit :

1-Méthodes qui se basent sur la transformation de modèle, composition ou décomposition de modèle, et aussi méthodes d'agrégation d'états [9, 10,15, 13, 16, 17].

2-Techniques for the computation of product form solution, inspired from queuing system analysis [7, 11, 12].

3-Techniques computing bounds for performance indicators [20, 21].

Ces méthodes ont permis de réduire dans certains cas l'impact de l'explosion combinatoire. Toutefois, la taille des systèmes d'aujourd'hui est en train d'augmenter de plus en plus, avec l'avènement de nouvelles applications notamment multimédia et l'utilisation de toute forme de réseau de communication (LAN, WAN, Wireless, adhoc, …). D'où, le besoin en de nouvelles approches est toujours ressenti.

Parmi ces techniques, plusieurs tentatives ont essayé d'identifier les sous-réseaux pouvant composer un RDPS, dans l'objectif d'analyser les sous-réseaux et résoudre le processus stochastique associé au réseau d'origine, sans pour autant générer la chaîne de Markov sous-jacente [7, 19, 2, 3, 5, 14]. Cette approche est intéressante, cependant l'identification des sous-réseaux est effectuée manuellement par l'utilisateur, en recherchant potentiellement deux types de dépendances entre les composants :

- Les composants peuvent communiquer entre eux via des places, en s'échangeant des jetons : cette communication entre des sous-systèmes est dite asynchrone.
- Les composants peuvent communiquer entre eux via des transitions (activités) qui se synchronisent : ceci représente des communications synchrones entre sous-systèmes.

Pour résoudre le RDPS décomposé en plusieurs sous-réseaux, ces travaux exploitent l'algèbre de Kronecker [6] pour exprimer la solution globale en fonction des solutions associées aux sous-réseaux, sans avoir besoin de stocker le générateur infinitésimal de la chaîne de Markov complète.

En étendant ce concept, nous proposons dans ce rapport une nouvelle approche de décomposition des RDP/RDPS, qui tire profit des travaux liés à la composition synchrone et asynchrone de sous-réseaux. L'objectif de cette approche est d'identifier les sous-réseaux composant le réseau à analyser, et déduire les relations liant ces sous-réseaux. Le résultat fourni par cette décomposition consistera alors en une expression algébrique représentant le RDPS, composée des sous-réseaux identifiés, liés par des opérateurs à définir.

Cette approche de décomposition va nous permettre de simplifier l'évaluation des performances du réseau à analyser par l'évaluation numérique de ses composants (sous-réseaux), puis un calcul de la solution globale associée au RDPS initial, en se basant sur l'expression algébrique déduite et l'algèbre de Kronecker. Afin de calculer la solution globale, il sera nécessaire d'étudier l'analyse numérique des performances du résultat de tout type d'expression algébrique reliant deux sous-réseaux ou plus par un des opérateurs ayant été définis auparavant. Comme notre souci ici est de présenter la technique de décomposition, la méthode de calcul de la solution globale d'un RDPS sera clarifiée dans un autre papier.

Ce rapport est organisé comme suit : nous rappelons d'abord les concepts inhérents aux RDPS et certaines définitions de la théorie des graphes. Nous introduisons après des sous-réseaux particuliers qui vont aider Durant l'identification des sous-réseaux d'un RDPS. Ensuite, nous définissons une algèbre de RDP/RDPS, et donc l'ensemble des opérateurs décrivant le lien entre deux sous-réseaux. Ces opérateurs sont utilisés dans la construction de l'expression algébrique d'un RDPS. Enfin, nous donnons l'approche de décomposition proposée, à partir de laquelle nous obtenons les sous-réseaux identifiés et l'expression algébrique correspondante. Un exemple illustre l'utilisation de l'algorithme proposé.

2. Concepts préliminaires

Dans ce qui ce suit, nous présentons le modèle de base utilisé ici : le modèle Réseau de Petri Stochastique RDPS. Nous rappelons également la définition de ce modèle au sens de la théorie des graphes, ainsi que d'autres concepts inhérents.

Définition 1: Un Réseau de Petri Stochastique (RDPS) est un couple S=<N,W> où:
- N est un réseau de Petri marqué, N = (P, T, Pre, Post, Inh, M_0) :
  - P, T sont deux ensembles finis disjoints, appelés respectivement places et transitions.
  - Pre, Post et Inh : PxT $\rightarrow$ Bag(P) sont des fonctions d'arcs :
    - Pre, Post sont respectivement les fonctions d'incidence arrière et avant,
    - Inh est la fonction d'inhibition, et
    - Bag(P) est l'ensemble des multiensembles sur P : Bag(P)={x=$(x(p))_{p\in P}$ / $\forall p\in P$, $x(p)\in IN$}.
  - M_0 est une fonction définie de P à N appelée le marquage initial.
- W = (λ_1,.....,λ_n) est le vecteur des taux de transition.

L'ensemble des places d'entrée (respectivement places de sortie) d'une transition est :
$^{\cdot}t = \{p\in P / Pre(p, t) >0\}$, respectivement $t^{\cdot} = \{p\in P / Post(p, t)>0\}$.

Un RDPS peut être défini au sens de la théorie des graphes comme suit :

Définition 2 : Un Réseau de Petri Stochastique S = <N, W> est un graphe biparti orienté G_N=(X,E), où :
- X : est un ensemble non vide de de sommets de G : X= P $\cup$ T.
- E : est un ensemble d'arcs reliant les sommets de G. Un arc ne relie jamais deux sommets de même type.

On définit également deux applications I et T / I, T : E $\rightarrow$ X appelées respectivement **extrémité Initiale** et **extrémité Terminale** de G.

Définition 3 : Degré d'un sommet
On appelle **degré** d'un sommet x, noté **d(x)**, le nombre d'arcs entrants et sortants (dits incidents) de x.
- On appelle **degré sortant** (**output degree**) $\mathbf{d^+(x)}$ le nombre d'arcs sortants (dits incidents extérieurs) de x : $d^+(x) = |\{e\in E / I(e)=x\}|$

- On appelle **degré entrant** (**input degree**) $\mathbf{d^-(x)}$ le nombre d'arcs entrants (dits incidents intérieurs) de x : $d^-(x) = |\{e \in E / T(e)=x\}|$

Remarque : Les degrés entrant et sortant d'une transition peuvent être aussi donnés par :
$d^+(x) = |{}^\bullet t| = |\{ p \in P / Pre(p,t) >0 \}|$,
resp. $d^-(x) = |t^\bullet| = |\{ p \in P / Post(p,t) >0 \}|$.

Définition 4 : Successeur et prédécesseur
Soit un RDP G=(X, E), X= P ∪ T.
Un sommet y (place ou transition) est dit **successeur** d'un autre sommet x (respectivement x est prédécesseur de y) si : ∋ e=(x, y) un arc ∈E.
On note $X^\bullet = \{y \in X / e=(x, y) \in E\}$ l'ensemble des successeurs de X.
$^\bullet X = \{y \in X / e=(y, x) \in E\}$ l'ensemble des prédécesseurs de X.

Définition 5: Source et Puits
Soit un RDP G=(X, E), X= P ∪ T.
Un sommet x (place ou transition) est dit **Source** (respectivement **Puits**) ssi :
$d^-(x) =0$ (respectivement $d^+(x) =0$).

Définition 6 : Chemin
Un chemin dans un graphe orienté G=(X, E) est une suite alternée de sommets et d'arcs :
$\gamma=x_0e_1x_1e_2 \ldots x_{k-1}e_kx_k$ tel que pour $i\in[1, k]$, $e_i=(x_{i-1}, x_i)$.
On dit que γ est un chemin de **longueur k** de x_0 à x_k .
Le chemin γ est **élémentaire** si tous les sommets du chemin sont distincts.

Définition 7 : Circuit
On appelle **Circuit** tout chemin fermé (k>0 et $x_0=x_k$).

Définition 8 : Interface d'un sous-réseau
Soit un RDP G=(X, E), X= P ∪ T. Soit un sous-réseau G' de G tel que G'=(X', E').
On appelle **Interface** de G l'ensemble des sommets de G', noté **Intf(G')**, relié à l'extérieur de G' et défini comme suit :
$Intf(G')=\{x' \in X'/ x \in X-X', (x', x) \in E$ ou bien $(x, x') \in E\}$
L'interface entrante (**Input Interface**), notée $\mathbf{Intf^-(G')}$, l'ensemble des sommets de G' reliés à l'extérieur de G' par des arcs entrants à G'.

L'interface sortante ou Output Interface, notée $\mathbf{Intf^+(G')}$, l'ensemble des sommets de G' reliés à l'extérieur de G' par des arcs sortants de G'.

Nous passons maintenant à la définition des différents sous-réseaux types sur lesquels on se base dans notre approche de décomposition.

3. Définition des Sous-réseaux Types

Définition 9 : Réseau sans choix ni synchronisation NCSN (None Choice/Synchronization Net)
Un RDP N = (P, T, Pre, Post,Inh, M0) est dit **Réseau sans choix ni synchronisation (NCSN)** ssi :
$\forall x \in X, d^+(x)= d^-(x) =1$
Exemple :

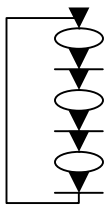


Fig. 1. NCSN

Définition 10 : Réseau de Petri SN_PnT
Un RDP $N = (P, T, Pre, Post, Inh, M_0)$ est **SN_PnT** ssi :
$|Intf^{+}(N)| = 1$ où l'interface entrante est constituée d'une seule place et,
$|Intf^{-}(N)| = n$ où l'interface sortante est constituée de n transitions.

Exemple : SN_P2T

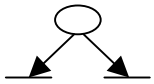


Fig. 2. SN_P2T

Définition 11 : Réseau de Petri SN_nPT
Un RDP $N = (P, T, Pre, Post, Inh, M_0)$ est **SN_nPT** ssi :
$|Intf^{+}(N)| = n$ où l'interface entrante est constituée de n places et,
$|Intf^{-}(N)| = 1$ où l'interface sortante est constituée de n transitions.

Exemple : SN_2PT

Fig. 3. SN_2PT

Définition 12 : Réseau de Petri SN_nTP
Un RDP $N = (P, T, Pre, Post, Inh, M_0)$ est **SN_nTP** ssi :
$|Intf^{+}(N)| = n$ où l'interface entrante est constituée de n transitions et,
$|Intf^{-}(N)| = 1$ où l'interface sortante est constituée d'une seule place.

Exemple : SN_2TP

Fig. 4. SN_2TP

Définition 13 : Réseau de Petri SN_TnP
Un RDP $N = (P, T, Pre, Post, Inh, M_0)$ est **SN_TnP** ssi :
$|Intf^{+}(N)| = 1$ où l'interface entrante est constituée d'une transition et,
$|Intf^{-}(N)| = n$ où l'interface sortante est constituée de n places.

Exemple : SN_T2P

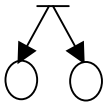


Fig. 5. SN_T2P

Définition 14 : Réseau de Petri SN_nPTmP
Un RDP $N = (P, T, Pre, Post, Inh, M_0)$ est **SN_nPTmP** ssi :
$|Intf^{+}(N)| = n$ où l'interface entrante est constituée de n places et,
$|Intf^{-}(N)| = m$ où l'interface sortante est constituée de m places.

Exemple : SN_2PT2P

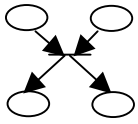


Fig. 6. SN_2PT2P

Définition 15 : Réseau de Petri SN_nTPmT
Un RDP $N = (P, T, Pre, Post, Inh, M_0)$ est **SN_nTPmT** ssi :
$|Intf^{+}(N)| = n$ où l'interface entrante est constituée de n transitions et,
$|Intf^{-}(N)| = m$ où l'interface sortante est constituée de m transitions.

Exemple : SN_2TP2T

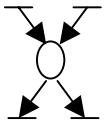


Fig.7. SN_2TP2T

4. Algèbre des réseaux de Petri stochastiques

Nous définissons dans la suite une algèbre qui va nous permettre de composer des RDPS via la manipulation d'opérateurs.

4.1. Définition de l'algèbre de RDP/RDPS

Définition 16: Algèbre de RDPS
Une **algèbre** de RDPS est un triplet <S, ∑, E> où :
- S : est un ensemble de RDPS $N=(P, T, Pre, Post, Inh, Pri, M_0, W)$,
- ∑ : est un ensemble d'opérateurs avec leur arité correspondante dans S, et
- E : est un ensemble d'axiomes ou équations entre termes composés de fonctions et de variables libres.

4.2. Opérateurs de l'algèbre

L'ensemble ∑ des opérateurs est défini comme suit :

4.2.1. Opérateur de choix ou sélection

Le choix entre deux sous-réseaux de Petri modélise deux comportements possibles dont l'un seulement doit être choisi, sur la base du choix d'une activité parmi deux, laquelle activité initie le comportement du réseau. Dans ce cas, structurellement, chacun des deux sous-réseaux est relié à une transition. Les deux transitions associées aux sous-réseaux sont en **conflit structurel** défini comme suit :

Définition 17 : Conflit structurel [16]
Une transition t est en **conflit structurel** avec une transition t' (noté t SC t') ssi :
$({}^{\bullet}t \cap {}^{\bullet}t' \neq \varnothing)$ ou $({}^{\bullet}t \cap t' \neq \varnothing)$ avec $t'=\{p \in P / Inh(p, t) \neq 0\}$

Définition 18 : Opérateur de choix (Choice operator) +
Soient deux RDPSG $N_1 = (P_1, T_1, Pre_1, Post_1, Inh_1, Pri_1, M_{01}, \theta_1)$, $N_2= (P_2, T_2, Pre_2, Post_2, Inh_2, Pri_2, M_{02}, \theta_2)$. $N_1 + N_2$ est un RDPSG $N=(P_1 \cup P_2, T_1 \cup T_2, Pre, Post, Inh, Pri, M_0, \theta)$ où :

- Pre : $(P_1 \cup P_2, T_1 \cup T_2) \rightarrow |N$ tel que : $Pre(p,t) = pre_1(p,t)$ si $p \in P_1$ et $t \in T_1$
  $Pre(p,t) = pre_2(p,t)$ si $p \in P_2$ et $t \in T_2$
  $Pre(p,t) = 0$ sinon
- Post : $(P_1 \cup P_2, T_1 \cup T_2) \rightarrow |N$ tel que : $Post(p,t) = post_1(p,t)$ si $p \in P_1$ et $t \in T_1$
  $Post(p,t) = post_2(p,t)$ si $p \in P_2$ et $t \in T_2$
  $Post(p,t) = 0$ sinon
- Inh : $(P_1 \cup P_2, T_1 \cup T_2) \rightarrow |N$ tel que : $Inh(p,t) = Inh_1(p,t)$ si $p \in P_1$ et $t \in T_1$

$Inh(p,t) = Inh_2(p,t)$ si $p \in P_2$ et $t \in T_2$
$Inh(p,t) = 0$ sinon

- Pri : $(T_1 \cup T_2) \rightarrow |N$ tel que : $Pri(t) = pri_1(t)$ si $t \in T_1$
  $Pri(t) = pri_2(t)$ si $t \in T_2$
  $Pri(t) = 0$ sinon
- θ : $(T_1 \cup T_2) \rightarrow |N$ tel que : $\theta(t) = \theta_1(t)$ si $t \in T_1$
  $\theta(t) = \theta_2(t)$ si $t \in T_2$
  $\theta(t) = 0$ sinon

Sémantique de l'opérateur de choix

Le choix entre deux sous-réseaux de Petri est supposé s'appuyer sur deux transitions en conflit structurel ayant une place commune (ressource commune) comme préconditions.

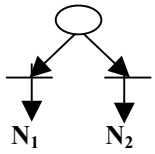


Fig.8. Choix entre deux processus

Dans ce cas, une politique de course (race policy) est utilisée pour décider du comportement à choisir lorsque des activités sont possibles (transitions correspondantes sensibilisées) simultanément. Comme à chaque transition est associée une durée de tir de distribution exponentielle, le RDPS à choisir est celui relié à la transition ayant la plus petite durée de franchissement échantillonnée.

4.2.2. Opérateur de parallélisme

Le parallélisme entre deux sous-réseaux de Petri modélise deux comportements qui peuvent être exécutés simultanément, à l'encontre de l'opérateur de choix.

Définition 18 : Opérateur de parallélisme (parallelism operator) ||

Soient deux RDPSG $N_1 = (P_1, T_1, Pre_1, Post_1, Inh_1, Pri_1, M_{01}, \theta_1)$, $N_2 = (P_2, T_2 Pre_2, Post_2, Inh_2, Pri_2, M_{02}, \theta_2)$. $N_1 || N_2$ est un RDPSG $N = (P_1 \cup P_2, T_1 \cup T_2, Pre, Post, Inh, Pri, M_0, \theta)$ où :

- Pre : $(P_1 \cup P_2, T_1 \cup T_2) \rightarrow |N$ tel que : $Pre(p,t) = pre_1(p,t)$ si $p \in P_1$ et $t \in T_1$
  $Pre(p,t) = pre_2(p,t)$ si $p \in P_2$ et $t \in T_2$
  $Pre(p,t) = 0$ sinon
- Post : $(P_1 \cup P_2, T_1 \cup T_2) \rightarrow |N$ tel que : $Post(p,t) = post_1(p,t)$ si $p \in P_1$ et $t \in T_1$
  $Post(p,t) = post_2(p,t)$ si $p \in P_2$ et $t \in T_2$
  $Post(p,t) = 0$ sinon
- Inh : $(P_1 \cup P_2, T_1 \cup T_2) \rightarrow |N$ tel que : $Inh(p,t) = Inh_1(p,t)$ si $p \in P_1$ et $t \in T_1$
  $Inh(p,t) = Inh_2(p,t)$ si $p \in P_2$ et $t \in T_2$
  $Inh(p,t) = 0$ sinon
- Pri : $(T_1 \cup T_2) \rightarrow |N$ tel que : $Pri(t) = pri_1(t)$ si $t \in T_1$
  $Pri(t) = pri_2(t)$ si $t \in T_2$
  $Pri(t) = 0$ sinon
- θ : $(T_1 \cup T_2) \rightarrow |N$ tel que : $\theta(t) = \theta_1(t)$ si $t \in T_1$
  $\theta(t) = \theta_2(t)$ si $t \in T_2$
  $\theta(t) = 0$ sinon

Sémantique de l'opérateur de parallélisme

Le parallélisme entre deux sous-réseaux de Petri modélise des processus qui peuvent s'exécuter simultanément. Donc, le comportement de chacun des sous-réseaux est indépendant de l'autre. Les sous-réseaux de Petri n'ont rien en commun et évoluent d'une manière indépendante.

4.2.3. Opérateur de composition synchrone

La composition synchrone des RDPs modélise la **synchronisation d'événements** associés aux sous-systèmes sous_jacents représentés par ces RDPs (Exemple : mécanisme du Rendez-vous). Ceci se présente sous-forme de transition commune (événement de synchronisation) qui ne peut être tiré que si les

réseaux à synchroniser disposent des ressources nécessaires (jetons nécessaires au franchissement de la transition).

Susanna Donatelli a étudié cette notion de composition synchrone mais des RDP stochastiques généralisés (RDPSG), en l'appelant **Superposition de transitions**. Initialement [17], elle transposa aux machines à états la méthode des Réseaux d'automates Stochastiques (SAN), puis, elle a étendu la méthode à la composition synchrone des RDPSG [8].

Définition 19 : Opérateur de composition synchrone (Synchronous composition operator) ▶◀

Le RDPSG $N = (P, T, Pre, Post, Inh, Pri, M_0, \theta)$ est la ***composition synchrone*** de ses K sous-réseaux $N_k = (P_k, T_k, Pre_k, Post_k, Inh_k, Pri_k, M_{0,k}, \theta_k)$ ssi :

1. $(P_k)_{k \in K}$ est une partition de P ;
2. $T = TS \cup TL$:
   a) $TS \neq \theta$, ensemble des transitions de synchronisation, ne contient que des transitions temporisées ;
   b) TL est l'ensemble des transitions locales.
3. $T_k = {}^\bullet P_k \cup P_k{}^\bullet$ et $(T_k \cap TL)_{k \in K}$ est une partition de TL ;
4. Les fonctions Pre_k, $Post_k$, Inh_k, Pri_k, θ_k sont les restrictions à P_k *et* T_k des fonctions correspondant à N ;
5. $M_0 = (M_{0,k})_{k \in K}$ est le marquage initial.

Dans chaque RDPSG N_k, nous avons ainsi une partition $T_k = TL_k \cup TS_k = TS \cap T_k$ est l'ensemble des transitions de synchronisation de N_k, et $TL_k = T_k \setminus TS_k$ est l'ensemble des transitions locales de N_k.

Tout marquage M de S est un tuple $(M_1, \ldots M_k)$ de marquage de S_k. On notera $r_k = |TRS_k|$.

Notation : Dans la suite, on notera $▶◀_{Ts}(N_1, N_2, \ldots, N_K)$ ou $\mathbf{Synch}_{Ts}(N_1, N_2, \ldots, N_K)$ la ***composition synchrone*** de K réseaux $N_1, N_2, \ldots, N_K$ par un ensemble de transitions Ts.

Exemple : Composition synchrone de RDPS

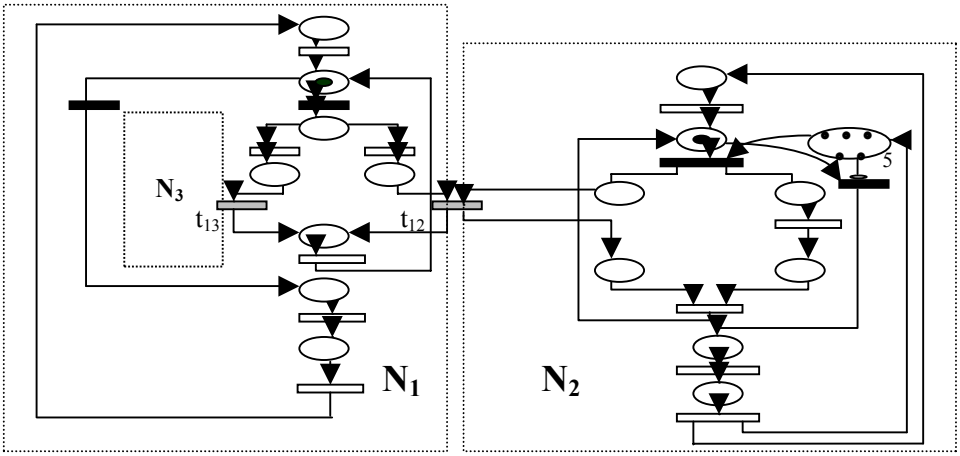


Fig.9. Composition synchrone de RDPS

4.2.4. Opérateur de composition asynchrone

La composition asynchrone des RDPs modélise des sous-systèmes communicants via des entités (tâches, ressources, …). Les sous-systèmes communiquent via des places communes, par échange de jetons passant d'un sous-système à un autre : on parle de *fusion de places*. Le passage de jetons se fait par le franchissement de transitions qui tirent dans les places communes. Ces transitions sont appelées ***transitions de sortie***.

P.Buchholz a étudié la composition asynchrone des RDPSG et d'autres types de RDP dans plusieurs papiers [1, 2, 3, 4, 5]. Dans ces travaux, il a proposé plusieurs modèles orientés vers une description hiérarchique de la composition.

Définition 20 : Opérateur de composition asynchrone (Asynchronous composition operator) ▼

Le RDPSG $N = (P, T, Pre, Post, Inh, Pri, M_0, \theta)$ est la ***composition asynchrone*** de K sous-réseaux $N_k = (P_k, T_k, Pre_k, Post_k, Inh_k, Pri_k, M_{0,k}, \theta_k)$ ssi :

1. $T = \cup_{k \in K} T_k$ (partition des transitions) ;

2. $P = \cup_{k\in K} P_k$ (partition des places) ;
3. $\forall k\in K, \forall t\in T_k \ {}^\bullet t \cup t^\bullet \subseteq P_k$

Dans la suite, nous notons pour tout marquage M de N, M = M(P_k), donc M=$(M_k)_{k\in K}$ et $TO_k=\{t\in T_k \mid t^\bullet \cap (P\backslash P_k) \neq \phi\}$ l'ensemble des transitions de sortie de N_k.

Notation : Dans la suite, on notera $\blacktriangledown_{Ta}(N_1, N_2, \ldots, N_K)$ ou **Asynch**$_{Ta}(N_1, N_2, \ldots, N_K)$ la ***composition asynchrone*** de K réseaux $N_1, N_2, \ldots, N_K$ via un ensemble de transitions Ta.

Exemple : Composition asynchrone de RDPS

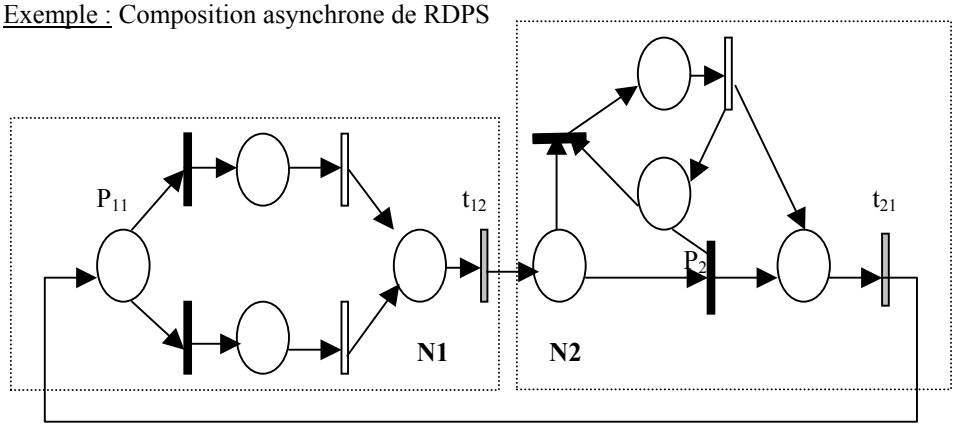


Fig.10. Composition asynchrone de RDPS

4.2.5. Opérateur d'encapsulation

Il est parfois intéressant de cacher certaines actions, et d'en faire abstraction. Pour cela, nous introduisons l'opérateur d'encapsulation qui **cache** un sous-réseau de Petri dans un autre.

Définition 21 : Opérateur d'encapsulation (Hide operator)

Soient deux RDPSG $N_1=(P_1, T_1, Pre_1, Post_1, Inh_1, Pri_1, M_{01}, \theta_1)$, $N_2=(P_2, T_2, Pre_2, Post_2, Inh_2, Pri_2, M_{02}, \theta_2)$. On dit que N_1 **encapsule** ou **cache (Hide)** N_2 ssi :

- $P_2 \subset P_1$
- $T_2 \subset T_1$
- $Pre_2(p,t) = pre_1(p,t)$ si $p\in P_2$ et $t\in T_2$
- $Post_2(p,t) = post_1(p,t)$ si $p\in P_2$ et $t\in T_2$
- $Inh_2(p,t) = Inh_1(p,t)$ si $p\in P_2$ et $t\in T_2$
- $Pri_2(t) = pri_1(t)$ si $t\in T_2$
- $\theta_2(t) = \theta_1(t)$ si $t\in T_2$

4.2.6. Opérateur d'Union

L'union entre deux sous-réseaux de Petri représente la concaténation des deux sous-réseaux, par l'ajout de liens (arcs) entre l'interface de sortie de l'un et l'interface d'entrée de l'autre.

Définition 22 : Opérateur d'union (Union operator)

Soient deux RDPSG $N_1 = (P_1, T_1, Pre_1, Post_1, Inh_1, Pri_1, M_{01}, \theta_1)$, $N_2= (P_2, T_2, Pre_2, Post_2, Inh_2, Pri_2, M_{02}, \theta_2)$. $N_1 \cup N_2$ est un RDPSG $N=(P_1\cup P_2, T_1\cup T_2, Pre, Post, Inh, Pri, M_0, \theta)$ où :

- Pre : $(P_1\cup P_2, T_1\cup T_2) \rightarrow |N$ tel que : $Pre(p,t) = pre_1(p,t)$ si $p\in P_1$ et $t\in T_1$
  $Pre(p,t) = pre_2(p,t)$ si $p\in P_2$ et $t\in T_2$
  $Pre(p,t) = 0$ sinon
- Post : $(P_1\cup P_2, T_1\cup T_2) \rightarrow |N$ tel que : $Post(p,t) = post_1(p,t)$ si $p\in P_1$ et $t\in T_1$
  $Post(p,t) = post_2(p,t)$ si $p\in P_2$ et $t\in T_2$
  $Post(p,t) = 0$ sinon
- Inh : $(P_1\cup P_2, T_1\cup T_2) \rightarrow |N$ tel que : $Inh(p,t) = Inh_1(p,t)$ si $p\in P_1$ et $t\in T_1$
  $Inh(p,t) = Inh_2(p,t)$ si $p\in P_2$ et $t\in T_2$

Inh(p,t) =0 sinon

- Pri : $(T_1 \cup T_2) \rightarrow |N$ tel que : Pri(t) =pri_1(t) si t$\in T_1$
  Pri(t) =pri_2(t) si t$\in T_2$
  Pri(t) =0 sinon
- θ : $(T_1 \cup T_2) \rightarrow |N$ tel que : θ(t) =θ_1(t) si t$\in T_1$
  θ(t) =θ_2(t) si t$\in T_2$
  θ(t) =0 sinon

Une fois notre algèbre définie, nous présentons ci-après notre approche de décomposition.

5. Approche de décomposition

L'approche de décomposition proposée s'effectue en deux étapes :

- La première consiste à effectuer une décomposition du RDPS à analyser **en couches**, chaque couche représentant les actions possibles qui peuvent être exécutées à la $n^{ième}$ étape de la vie du système.
- La deuxième étape se fait sur la base de la vue multicouche du réseau, construite durant la première étape. Elle consiste à faire une recherche des sous-réseaux types, ayant été définis dans la section 3, identifie les sous-réseaux cibles qui composeront le réseau global, ainsi que le type d'opérateur les liant, déduisant ainsi l'expression algébrique qui sous-tend le réseau analyser.

5.1. Décomposition multicouches d'un RDPS

L'idée est de structurer le réseau initial en plusieurs **couches** ou **niveaux**: une couche i représente la $i^{ème}$ étape de la vie du système modélisé ; elle est donnée par un ensemble d'actions possibles à cette étape, munies des ressources nécessaires à l'exécution de ces actions. En termes de réseau de Petri, une couche regroupera les places marquées à un instant t, et leur transitions de sortie (successeurs immédiats). L'exécution de la couche est caractérisée par les franchissements possibles des transitions sensibilisées en même temps. Une fois cette structuration faite, les couches sont parcourues pour rechercher éventuellement des sous-réseaux liés par les opérateurs définis dans notre algèbre.

On note que l'approche proposée pour la structuration en couches du réseau de Petri est définie pour des réseaux de Petri ordinaires ou autres, en l'occurrence pour des RDPS. Dans ce qui suit, nous introduisons cette approche.

5.1.1. Structuration en couches d'un RDP

Définition 23 : Niveau d'un sommet

On appelle **niveau** d'un sommet x, noté **v(x)**, la longueur maximum d'un chemin élémentaire se terminant en x.

Définition 24 : Ensemble de sensibilisation et préconditions de sensibilisation

Soit un RDP *N = (P, T, Pre, Post,Inh, M_0).* On appelle **Ensemble de sensibilisation** de T pour M_0, noté **Tsens(T, M_0)**, l'ensemble défini comme suit :

{ t$\in$T / $\forall$p$\in$P telle que Pre(p, t) $\neq$ 0, on a M(p) >=Pre (p, t) }.

A un ensemble de sensibilisation T'=Tsens(T, M_0), on associe l'ensemble des **Préconditions de sensibilisation**, noté **Psens(T', M_0)**, défini comme l'ensemble suivant des places :

{ p$\in$P / $\forall$t$\in$T, p est une précondition de t (càd Pre(p, t) $\neq$ 0) et M(p) >=Pre (p, t) }.

Proposition 1 : Soit un graphe G=(X, E) un graphe sans circuit. Alors, son ensemble de sommets X admet une partition {$C_0 \cup C_1 \cup \ldots \cup C_p$ }telle que : x$\in C_i \Leftrightarrow$ v(x)=i. Posons :

C_0 ={ Sources de G}

C_1 ={Sources de G-C_0 }

…

C_p ={Sources de G- $\cup_{j=0}^{i-1} C_j$ }

Se basant sur cette proposition, il est possible de partitionner un RDP en couches comme suit :

Proposition 2 : Soit un RDP *N = (P, T, Pre, Post, Inh, M_0)* auquel correspond le graphe G=(X,E). Alors, N admet une structure en couches $\{C_0 \cup C_1 \cup \ldots \cup C_p\}$ telle que :

$x \in C_i \Leftrightarrow$ (x=p et v(p)=2i) ou (x=t et v(t)=2i+1).

De plus, xi G est un graphe avec ou sans circuit, la structure en couches est une partition de couches telle que: $\exists$ i, j, $C_i \cap C_j \neq \emptyset$ à cause de la constitution des couches.

N peut être décomposé en couches selon l'algorithme suivant :

Algorithme Construct_layers

BEGIN

1. $C_0 = (X_0, E_0)$ / $X_0 = P_0 \cup T_0$ où T_0=Tsens(T, M_0) et P_0=Psens(T_0, M_0), et $E_0=\{e \in E / \exists x \in X_0, \exists y \in X_0, e=(x, y)\}$.
2. $E = E - E_0$
3. i=1
4. Répéter
   - Soit M_i est le marquage obtenu de M_{i-1}lorsque toutes les transitions de sont franchies.
   - $C_i=(X_i, E_i)$ / $X_i=P_i \cup T_i$, T_i=Tsens(T, M_i), P_i=Psens(T_i, M_i), et $E_i=\{e \in E / \exists x \in X_i, \exists y \in X_i, e=(x, y)\}$.
   - Soit L_i l'ensemble des arcs liant C_{i-1} à C_i. Nous l'appelons Lien entre C_{i-1} et C_i, dénoté $L(C_{i-1},C_i)$:
     $L_i=L(C_{i-1}, C_i)=\{e \in E / \exists x \in X_{i-1}, \exists y \in X_i, e=(x, y)\}$.
   - $E = E - E_i$, $E = E - L_1$.
   - i=i+1.

   Jusqu'à E=$\emptyset$

END

5.1.2. Arbre de Liens d'une structure en couches d'un RDP

Nous définissons ci_après un arbre représentant les liens entre les différentes couches identifiées.

Définition 25 : Arbre de liens

Soit p le nombre de couches obtenues d'un RDP N. On appelle **Arbre de liens** d'un RDP N le graphe A=(C, L) construit par l'algorithme suivant :

Algorithme Link_Tree

BEGIN

C= { C_0, C_1, …, C_p}

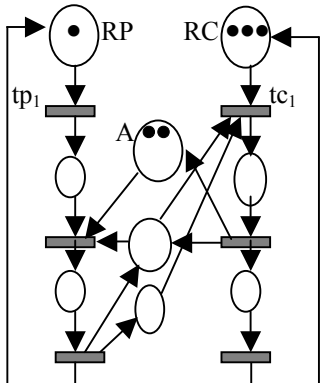
L= { L_1, …, L_p}$\cup$ L' / L' est l'ensemble des arcs reliant des couches non adjacentes et construit comme suit :

- L'=$\emptyset$.
- Pour i=0 à p Faire
  Pour j=i+2 à p Faire
  Si $P_i \cap P_j \neq \emptyset$
  Alors L'=L'$\cup$\{ e'=(C_j, C_i) / $\exists e \in E$, e=(x,y), $x \in C_j$, $y \in C_i$ \}
  Fsi
  Fait
  Fait

END

Exemple : Producteurs et consommateurs

Soit le réseau de Petri suivant modélisant le modèle des producteurs et consommateurs.



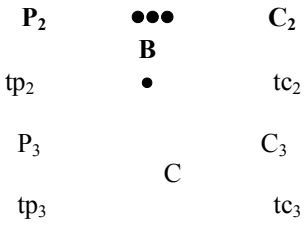
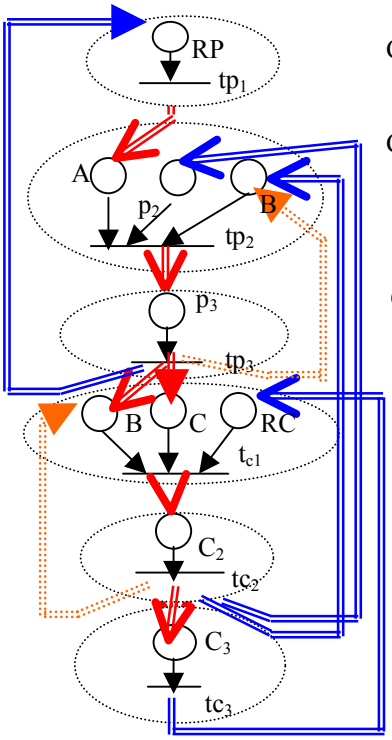


Fig.11. RDP des producteurs/consommateurs

La construction des couches résulte en ce qui suit :



$C_0=(P_0 \cup T_0, E_0)$ / $P_0=\{RP\}$, $T_0=\{tp_1\}$, $E_0=\{(RP, tp_1)\}$

$C_1=(P_1 \cup T_1, E_1)$/ $P_1=\{p_2,A,B\}$, $T_1=\{tp_2\}$, $E_1=\{(p_2,tp_2),(A,tp_2),(B,tp_2)\}$

$C_2=(P_2 \cup T_2,E_2)/P_2=\{p_3\}$, $T_3=\{tp_3\}$, $E_2=\{(p_3, tp_3)\}$

$C_3=(P_3 \cup T_3, E_3)$/ $P_3=\{B,C,RC\}$,$T_3=\{t_{c1}\}$, $E_3=\{(B,t_{c1}),(C,t_{c1}),(RC, t_{c1})\}$

$C_4=(P_4 \cup T_4,E_4)$ /$P_4=\{C_2\}$, $T_4=\{tp_2\}$, $E_4=\{(C_2,tp_2)\}$

$C_5=(P_5 \cup T_5,E_5)/P_5=\{C_3\}$, $T_5=\{tc_3\}$, $E_5=\{(C_3, tc_3)\}$

Fig.12. Structure en couches du RDP

L'arbre de liens correspond est donné ci_après :

$L_1 = L(C_0, C_1)=\{(tp_1, p_2)\}$
$L_2 = L(C_1, C_2)=\{(tp_3, B)\}$
$L_3 = L(C_2, C_3)=\{(tp_1, p_2)\}$
$L_4 = L(C_3, C_4)=\{(tc_1, C_2)\}$
$L'_4 = L(C_4, C_2)=\{ (tc_2, A), (tc_2, B)\}$
$L_5 = L(C_4, C_5)=\{(tc_2, C_3)\}$,
$L'_5 = L(C_5, C_3)=\{(tc_3, RC)\}$

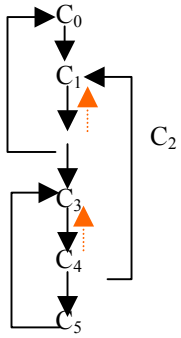


Fig.13. Arbre de liens du RDP

5.2. Recherche de l'expression algébrique associée à un RDPS

Soit un réseau de Petri $N = (P, T, Pre, Post, Inh, M_0)$ auquel correspond le Graphe G=(X,E). Soit np le nombre de couches obtenues de N et A=(C, L) le graphe de liens correspondant où :

- $C= \{ C_0, C_1, \ldots, C_{np}\}$ / $C_i =(X_i, E_i)$, $X_i= P_i \cup T_i$ et,

- $L = \{ L_1, \ldots, L_{np}\} \cup L'$

Afin de rechercher l'expression algébrique correspondant à un RDPS, nous introduisons les termes suivants :

Définition 26 : Réseau dérivé avant et après d'un sommet

Soit un réseau de Petri marqué *N = (P, T, Pre, Post,Inh, M_0)* auquel correspond le Graphe G=(X,E). On appelle **Réseau dérivé après** d'un sommet $x \in X$ (respectivement **Réseau dérivé avant** de x), noté **NetAfter(x)** (resp. **NetBefore(x)**), l'ensemble des successeurs (resp. prédécesseurs) de x et les successeurs (resp. prédécesseurs) de chaque élément de cet ensemble, construits d'une manière récursive :

1. NetAfter(x)= (X_a, E_a) /

Calcul de X_a : $X_a = \{y \in x^\bullet\}$

Répéter $S = X_a$. Soit $y \in S$, $X_a = X_a \cup \{z \in y^\bullet\}$, $S = S - \{y\}$ **Jusqu'à** $S = \varnothing$

Calcul de E_a : $E_a = \{e \in E / \exists x \in X_a, \exists y \in X_a, e=(x, y)\}$

2. NetBefore(x)= (X_b, E_b) /

Calcul de X_b : $X_b = \{y \in {}^\bullet x\}$

Répéter $S = X_b$, Soit $y \in S$, $X_b = X_b \cup \{z \in {}^\bullet y\}$, $S = S - \{y\}$ **Jusqu'à** $S = \varnothing$

Calcul de E_b : $E_b = \{e \in E / \exists x \in X_b, \exists y \in X_b, e=(x, y)\}$

Une fois tous les outils nécessaires définis, il est possible d'identifier les sous-réseaux composant un RDPS, et trouver l'expression algébrique correspondante grâce à l'algorithme suivant basé sur les couches obtenues à partir de N, :

Algorithme Decompose ()

BEGIN

1. Identifier dans chaque couche l'ensemble des sous-réseaux qui sont totalement indépendants, càd : Pour une couche C_1, chercher les sous-réseaux $SN_{jl}=(P_{jl}, T_{jl}, Pre_{jl}, Post_{jl}, Inh_{jl}, Pri_{jl}, M_{0jl}, \theta_{jl})$ tels que : $\cap_{j1=1}^{k1} P_{jl} = \varnothing$, et $\cap_{j1=1}^{k1} T_{jl} = \varnothing$.

On peut déduire une première expression algébrique liant les sous-réseaux des couches, et ce, comme suit :

Soit $C_0 = \{SN_{j0} / j_0=1,\ldots, k_0\}$,
$C_1 = \{SN_{j1} / j_1=1,\ldots, k_1\}$, …,
$C_{np} = \{SN_{jnp} / j_{np}=1,\ldots, knp\}$. So,

$N=(||_{j0=1}^{k0} SN_{j0}) \blacktriangledown_{Ta0} (||_{j1=1}^{k1} SN_{j1}) \blacktriangledown_{Ta1} \ldots \blacktriangledown_{Tanp} (||_{jnp=1}^{knp} SN_{jnp})$ ....**(1)**

Où T_{ai}, i=1 à np : est l"ensemble de transitions liant les sous-réseaux $SN_{j(i-1)}$ and SN_{ji} par une composition asynchrone.

2. a) Chercher les places dupliquées dans plus d'une couche : Comme la construction des couches se base sur l'activité (transition) pour décrire une étape d'exécution du système étudié, on peut se retrouver avec une place servant à plusieurs transitions ou activités de couches différentes.
Pour cela, il faut d'abord détecter les places dupliquées, puis regrouper les sous-réseaux contenant la même place dupliquée (en utilisant l'opérateur de l'union) afin d'en faire un seul sous-réseau.
b) Recalculer les liens L entre les couches, et déduire l'arbre des liens correspondant.
c) Pour chaque lien l_i entre deux couches C_k et C_j reliant deux sous-réseaux SN_{ku} et SN_{ju} par une transition t, on a : $SN_{ku} \blacktriangledown_{\{t\}} SN_{ju}$

3. Chercher à l'intérieur de chaque couche C_1 si des **sous-réseaux de type NCSN** existent, en explorant pour chaque sous-réseau des couches leurs réseaux successeurs appartenant aux couches suivantes.

4. Vérifier l'existence de **sous-réseaux de type SN_nPT,** càd rechercher les transitions t ayant $d(t)>1$, soit $d^-(t)=k$.
- Retrouver $t^\bullet$ et ${}^\bullet t$.
- Pour chaque place p_u de $t^\bullet$, retrouver $NetBefore(p_u)= (X_{SNu}, E_{SNu})$/ $X_{SNu} = P_{SNu} \cup T_{SNu}$.

- Retrouver les sous-réseaux NetAfter(t)= (X_{SNv}, E_{SNv}) / X_{SNv}= P_{SNv} ∪ T_{SNv} correspondant à chaque arc e_v =(t, p_v).
Si $d^+(t)=1$
Alors
Si $\cap_u P_{SNu} = \varnothing$ et $\cap_u T_{SNu} = \varnothing$ /* Intersection des ensembles de places et transitions des sous-réseaux NetBefore() */
Alors On a l'expression :
$\bullet_{\{t\}}$**[[**$\bullet\bullet_{\{t\}}$(NetBefore(p_{u1}), NetBefore(p_{u2}), …, NetBefore(p_{uk}))], NetAfter(t)**]** **...(2)**
Sinon **R**egrouper les sous-réseaux NetBefore(p_u) qui ont une intersection non vide.
Remplacer dans l'expression (2) les sous-réseaux en question par **leur union**.
Si tous les sous-réseaux ont une intersection non vide, l'expression ne contiendra pas l'opérateur **Synch (••)**, mais l'opérateur **Asynch** (•) entre l'union entre tous les sous-réseaux NetBefore et NetAfter.
$\bullet_{\{t\}}$**(**(NetBefore(p_{u1})∪ NetBefore(p_{u2}) ∪ … ∪ NetBefore(p_{uk2})),
Fsi
Sinon Voir cas 5°/ **SN_nPTmP.**
Fsi

Une fois l'expression déduite (expression (2) ou expression apparente), pour terminer la suite de l'algorithme, il est nécessaire de faire abstraction des sous-réseaux inclus dans cette expression, afin de chercher une expression globale reliant ces sous-réseaux au reste des sous-réseaux du modèle initial. Pour ce faire, nous remplaçons l'ensemble de ces sous-réseaux par un macro sous-réseau ABS dont l'expression est donnée ci-dessus, nous recalculons le nouveau graphe des couches et l'arbre des liens correspondant, en considérant le sous-réseau ABS et non son contenu. Ensuite, nous déduisons les relations entre les sous-réseaux des nouvelles couches en utilisant l'opérateur **Asynch** (•). Cette étape est dite **étape d'Abstraction.**

5. Vérifier l'existence de **sous-réseaux de type SN_nPTmP,** càd rechercher les transitions t dont $d^+(t)>1$, soit $d^+(t)=k_1$, et $d^-(t)>1$, soit $d^-(t)=k_2$.
- Retrouver $t^\bullet$ et $^\bullet t$.
- Pour chaque place p_u de $^\bullet t$, retrouver NetBefore(p_u)= (X_{SNu}, E_{SNu})/ X_{SNu}= P_{SNu} ∪ T_{SNu}.
- Retrouver les sous-réseaux $NetAfter_v(t)$= (X_{SNv}, E_{SNv}) / X_{SNv}= P_{SNv} ∪ T_{SNv} correspondant à chaque arc e_v =(t, p_v).

Si ($\cap_u$ **P_{SNu} = ∅ et** $\cap_u$ **T_{SNu} = ∅**) **et** ($\cap_v$ **P_{SNv} = ∅ et** $\cap_v$ **T_{SNv} = ∅**)
Alors Répartir l'ensemble des sous-réseaux NetAfter sur les sous-réseaux NetBefore de telle manière à faire l'union entre un sous-réseau NetAfter et un autre NetBefore. Soit k_1 sous-réseaux $NetAfter_v(t)$, et k_2 sous-réseaux NetBefore(p_u).
Si $k_1=k_2$
Alors On a l'expression :
$\bullet\bullet_{\{t\}}$ (NetBefore(p_{u1}) ∪ $NetAfter_{v1}(t)$, NetBefore(p_{u2}) ∪ $NetAfter_{v2}(t)$, …,NetBefore(p_{uk1}) ∪ $NetAfter_{k1}(t)$) …**(3)**
Sinon Si $k_1<k_2$
Alors On a l'expression :
$\bullet\bullet_{\{t\}}$ (NetBefore(p_{u1}) ∪ $NetAfter_{v1}(t)$, …,NetBefore(p_{uk1}) ∪ $NetAfter_{k1}(t)$, …,NetBefore(p_{uk2}) ∪ $NetAfter_{k2}(t)$) … **(4)**
Sinon /* $k_1>k_2$ */ On a l'expression :
$\bullet\bullet_{\{t\}}$(NetBefore(p_{u1}) ∪ $NetAfter_{v1}(t)$, …,NetBefore(p_{uk2}) ∪ **(**$NetAfter_{k2}(t)$ || …||$NetAfter_{k1}(t)$**)**) … **(5)**
Fsi
Fsi
Sinon Si ($\cap_u P_{SNu}=\varnothing$ et $\cap_u T_{SNu}=\varnothing$) /* $\cap_v P_{SNv} \neq \varnothing$ ou $\cap_v T_{SNv} \neq \varnothing$ */

Alors - Regrouper les sous-réseaux NetAfter qui ont une intersection non vide.
- Remplacer dans l'expression (3), (4) ou (5) (selon le cas) les sous-réseaux en question par **l'union** des sous-réseaux .

Sinon Si $\cap_v P_{SNv}=\emptyset$ **et** $\cap_v T_{SNv}=\emptyset$ **/*** $\cap_u P_{SNu} \neq \emptyset$ **et** $\cap_u T_{SNu} \neq \emptyset$ ***/**
Alors
- Regrouper les sous-réseaux NetBefore(p_u) qui ont une intersection non vide.
- Remplacer dans l'expression (2) les sous-réseaux en question par **l'union** des sous-réseaux.
- Si tous les sous-réseaux ont une intersection non vide, l'expression ne contiendra pas l'opérateur Synch (••), mais plutôt l'opérateur • :

$\bullet_{\{t\}}$((NetBefore(p_{u1}) ∪ NetBefore(p_{u2}) ∪ … ∪ NetBefore(p_{uk1})), (NetAfter$_1$(t) || …||NetAfter$_{k1}$(t)).

Sinon /* ($\cap_u P_{SNu}\neq\emptyset$ ou $\cap_u T_{SNu}\neq\emptyset$) et ($\cap_v P_{SNv}\neq\emptyset$ ou $\cap_v T_{SNv}\neq\emptyset$)*/
$\bullet_{\{t\}}$((NetBefore(p_{u1})∪ NetBefore(p_{u2})∪…∪NetBefore(p_{uk2})), NetAfter$_1$(t) ∪ …∪ NetAfter$_{k1}$(t))).
Fsi

Fsi
Fsi
Une fois l'expression déduite, effectuer l'étape d'abstraction.

6. Vérifier l'existence de **sous-réseaux de type SN_PnT,** càd rechercher les places p ayant $d^+(p)>1$, soit $d^+(p)=k$ tel que : pour chaque transition t_u de $p^\bullet$, ${}^\bullet t_u = \{p\}$ exactement.
- Retrouver $p^\bullet$.
- Pour chaque transition t_u de $p^\bullet$, retrouver NetAfter(t_u)= (X_{SNu}, E_{SNu})/ $X_{SNu}= P_{SNu} \cup T_{SNu}$.

Si $\cap_u P_{SNu} = \emptyset$ **et** $\cap_u T_{SNu} = \emptyset$
Alors Soit $d^+(t)=k$ /* k est au moins égal à 2 */
On note SR le sous-réseau SN_PnT
On a l'expression :
$\bullet_{\{tu1,tu2,\ldots,tuk\}}$(SR, NetAfter($t_{u1}$)+NetAfter($t_{u2}$)+…+NetAfter($t_{uk}$)) …**(6)**
Sinon - Regrouper les sous-réseaux NetAfter(t_u) qui ont une intersection non vide.
- Remplacer dans l'expression (6) les sous-réseaux en question par **l'union** des sous-réseaux .
Fsi
Une fois l'expression déduite, effectuer l'étape d'abstraction.

7. Vérifier l'existence de **sous-réseaux de type SN_nTP,** càd rechercher les places p dont $d^-(p)>1$, soit $d^-(p)=k$.
- Retrouver $p^\bullet$.
- Retrouver les sous-réseaux NetBefore$_u$(p)=(X_{SNu}, E_{SNu})/ $X_{SNu}=P_{SNu}\cup T_{SNu}$, ayant chacun comme interface sortante $Intf^+$(NetBefore$_u$(p)) l'ensemble $\{t_u\}$constitué d'une transition t_u de $p^\bullet$.

Si $\cap_u P_{SNu} = \emptyset$ et $\cap_u T_{SNu} = \emptyset$
Alors Soit $d^-(t)=k$ /* k est au moins égal à 2 */
On note SR le sous-réseau dont l'interface entrante $Intf^-$(SR)={p}.
On a l'expression :
$\bullet_{\{tu1,tu2,\ldots,tuk\}}$(NetBefore$_{u1}$(p)||NetBefore$_{u2}$(p)||…||NetBefore$_{uk}$(p),SR..**(7)**
Sinon - Regrouper les sous-réseaux NetBefore$_u$(p) qui ont une intersection non vide.
- Remplacer dans l'expression (7) les sous-réseaux en question par **l'union** des sous-réseaux .
- Si tous les sous-réseaux ont une intersection non vide, alors l'opérateur || ne sera plus utilisé.
Fsi
Une fois l'expression déduite, effectuer l'étape d'abstraction.

8. Vérifier l'existence de **sous-réseaux de type SN_TnP,** càd rechercher les transitions t dont $d^+(t)>1$, soit $d^+(t)=k$.
- Retrouver $t^\bullet$.
- Retrouver les sous-réseaux $NetAfter_u(t)= (X_{SNu}, E_{SNu})/ X_{SNu}= P_{SNu}\cup T_{SNu}$, ayant chacun comme interface entrante $Intf^-(NetAfter_u(t))$ l'ensemble $\{p_u\}$constitué d'une place p_u de t .

Si $\cap_u P_{SNu} = \varnothing$ et $\cap_u T_{SNu} = \varnothing$
Alors Soit $d^+(t)=k$ /* k est au moins égal à 2 */
On note SR le sous-réseau dont l'interface sortante $Intf^+(SR)=\{t\}$.
On a l'expression :
$\bullet_{\{t\}}(SR, NetAfter_{u1}(p)||NetAfter_{u2}(p)||\ldots||NetAfter_{uk}(p))$ ...**(8)**
Sinon - Regrouper les sous-réseaux $NetBefore_u(p)$ qui ont une intersection non vide.
- Remplacer dans l'expression (8) les sous-réseaux en question par **l'union** des sous-réseaux .
- Si tous les sous-réseaux ont une intersection non vide, alors l'opérateur || ne sera plus utilisé.
Fsi
Une fois l'expression déduite, effectuer l'étape d'abstraction.

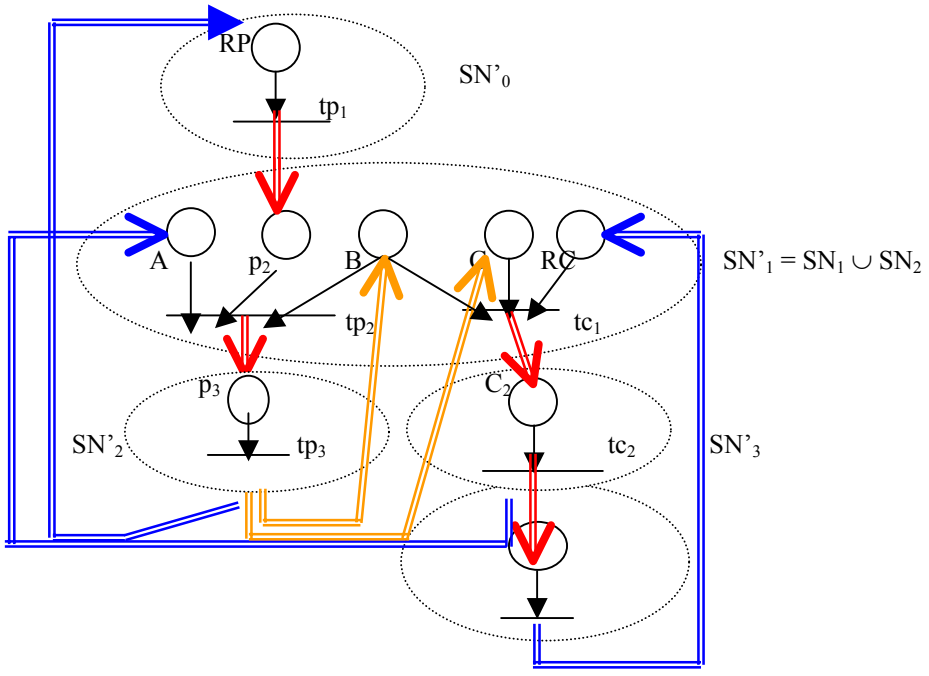
9. Vérifier l'existence de **sous-réseaux de type SN_nTPmT,** càd rechercher les places p dont $d^+(p)>1$, soit $d^+(p)=k_1$, et $d^-(p)>1$, soit $d^-(p)=k_2$.
Ceci revient à étudier le cas 6°/ **SN_PnT**, puis l'expression déduite donnera lieu à un macro sous-réseau qui fait abstraction des sous-réseaux inclus dans l'expression, enfin étudier le cas **7°/ SN_nTP.**
10. Si tous les sous-réseaux types ont été vérifiés dans toutes les couches, on utilise les expressions trouvées avec les relations déduites du dernier arbre de liens pour déduire l'expression globale de N.

END

Exemple précédent : Producteurs et consommateurs
Appliquons l'algorithme de décomposition ::

1. 1ère expression intuitive
$N = SN_0 \blacktriangledown_{\{tp1\}} SN_1 \blacktriangledown_{\{tp2\}} SN_2 \blacktriangledown_{\{tp3\}} SN_3 \blacktriangledown_{\{tc1\}} SN_4 \blacktriangledown_{\{tc2\}} SN_5$ (▼ : Opérateur **Asynch).**

2. Chercher les places dupliquées dans plus d'une couche et regrouper les sous-réseaux correspondants : Dans l'exemple, B est dupliquée dans C_1 et C_3. Donc, nous groupons ces deux couches comme suit :



C_3 SN'_4

tc_3

Fig.14. Vue en couches du RDP et ses sous-réseaux

- **Arbre de Liens correspondant**

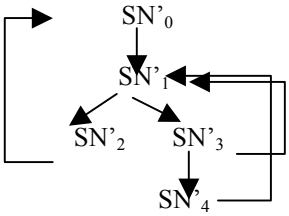


Fig.15. Arbre de liens du RDP

- **Relations déduites entre les sous-réseaux :**

$SN'_0 \blacktriangledown_{\{tp1\}} SN'_1$
$SN'_1 \blacktriangledown_{\{tp2\}} SN'_2$
$SN'_2 \blacktriangledown_{\{tp3\}} SN'_0$
$SN'_1 \blacktriangledown_{\{tp3\}} SN'_3$
$SN'_3 \blacktriangledown_{\{tc1\}} SN'_4$
$SN'_3 \blacktriangledown_{\{tc2\}} SN'_1$
$SN'_4 \blacktriangledown_{\{tc3\}} SN'_1$

3. Chercher à l'intérieur de chaque couche C_1 si des **sous-réseaux de type NCSN** existent : Néant, les sous-réseaux restent tels qu'ils sont.

4. Vérifier l'existence de **sous-réseaux de type SN_nPT,** càd rechercher les transitions t ayant $d^-(t)>1$, soit $d^-(t)=k$.

Deux sous-réseaux de type **SN_nPT** existent : SN_1, SN_2 composant SN'_1.

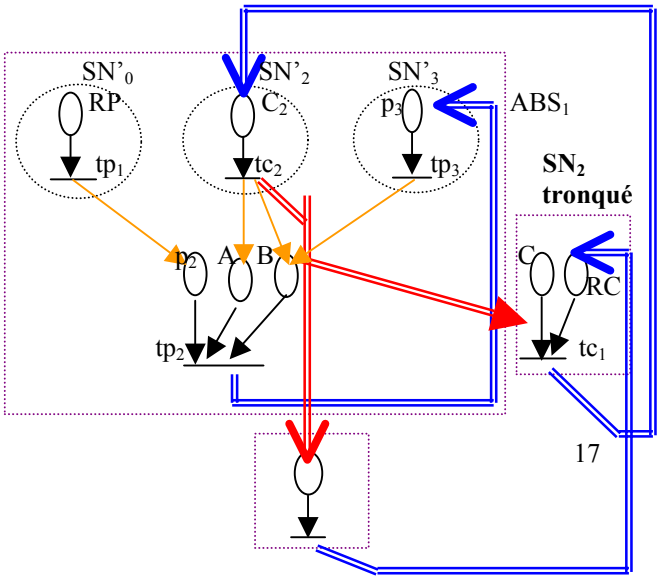
a) Pour SN_1 : on cherche NetBefore $(p_2) = SN'_0$
NetBefore(A) $= SN'_3$
NetBefore(B) $= SN'_2$, SN'_3
NetAfter(tp_2) $= SN'_2$

D'où, on peut déduire l'expression suivante :

$\blacktriangledown_{\{tp2\}}[\blacktriangleright \blacktriangleleft_{\{tp2\}}(SN'_0 \cup S_1 \ , SN'_3 \cup S_2, SN'_2 \cup S_3), SN'_2]$

Où : $S_1= (P_{s1} \cup T_{s1}, E_{s1}) / P_{s1}=\{p_2\}, T_{s1} =\{tp_2\}, \ E_{s1}=\{(p_2, tp_2)\}$.
$S_2= (P_{s2} \cup T_{s2}, E_{s2}) / P_{s2}=\{A, B\}, T_{s2} =\{tp_2\}, \ E_{s2}=\{(A, tp_2), (B, tp_2)\}$.
$S_3= (P_{s3} \cup T_{s3}, E_{s3}) / P_{s3}=\{B\}, T_{s3} =\{tp_2\}, \ E_{s3}=\{(B, tp_2)\}$.

Abstraction des sous-réseaux impliqués dans la nouvelle expression et remplacement par un macro sous-réseau :



C_3 SN'_4

tc_3

Fig.16. Application de l'étape 4 de l'algorithme Decompose()

- **Arbre de Liens correspondant**

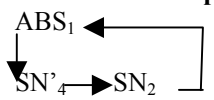


Fig.17. Nouvel Arbre de liens du RDP

- **Relations déduites entre les sous-réseaux**

$ABS_1 \blacktriangledown_{\{tc2\}} SN'_4$
$SN'_4 \blacktriangledown_{\{tc3\}} SN_2$
$SN_2 \blacktriangledown_{\{tc1\}} ABS_1$

b) Pour SN_2 : on cherche NetBefore (B) = ABS_1
NetBefore(C) = ABS_1
NetBefore(RC) = SN'_4
NetAfter(tc_1) = ABS_1

D'où, on peut déduire l'expression suivante :

$\blacktriangledown_{\{tc1\}} [\blacktriangleright\blacktriangleleft_{\{tc1\}} (ABS_1 \cup S_4 , SN'_4 \cup S_5), ABS_1]$

Où : $S_4 = (P_{s4} \cup T_{s4}, E_{s4}) / P_{s4}=\{C\}, T_{s4}=\{tc_1\}, E_{s4}=\{(C, tc_1)\}$.

$S_5 = (P_{s5} \cup T_{s5}, E_{s4}) / P_{s4}=\{RC\}, T_{s5}=\{tc_1\}, E_{s5}=\{(RC, tc_1)\}$.

Expression globale déduite :

$\blacktriangledown_{\{tc1\}} [\blacktriangleright\blacktriangleleft_{\{tc1\}} [[\blacktriangledown_{\{tp2\}} [\blacktriangleright\blacktriangleleft_{\{tp2\}} (SN'_0 \cup S_1 , SN'_3 \cup S_2, SN'_2 \cup S_3), SN'_2] \cup S_4 , SN'_4 \cup S_5] ,$
$[\blacktriangledown_{\{tp2\}} [\blacktriangleright\blacktriangleleft_{\{tp2\}} (SN'_0 \cup S_1 , SN'_3 \cup S_2, SN'_2 \cup S_3), SN'_2]]$

Abstraction des sous-réseaux impliqués dans la nouvelle expression et remplacement par un macro sous-réseau :

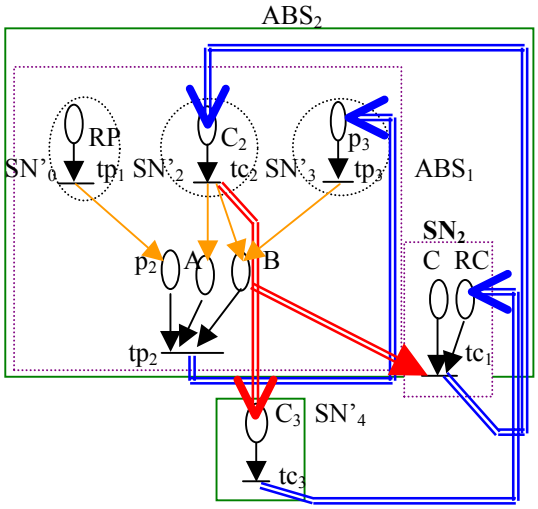


Fig.18. Application de l'étape 4 de la suite de l'algorithme Decompose().

- **Arbre de Liens correspondant**

Fig.19. Nouvel Arbre de liens du RDP

- **Relations déduites entre les sous-réseaux :**
  $ABS_2 \blacktriangledown_{\{tc2\}} SN'_4$
  $SN'_4 \blacktriangledown_{\{tc3\}} ABS_2$

5. Vérifier l'existence de **sous-réseaux de type SN_nPTmP,** càd rechercher les transitions t dont $d^+(t)>1$, soit $d^+(t)=k_1$, et $d^-(t)>1$, soit $d^-(t)=k_2$: Dans le nouveau réseau abstrait, pas de sous-réseaux de type **SN_nPTmP.**

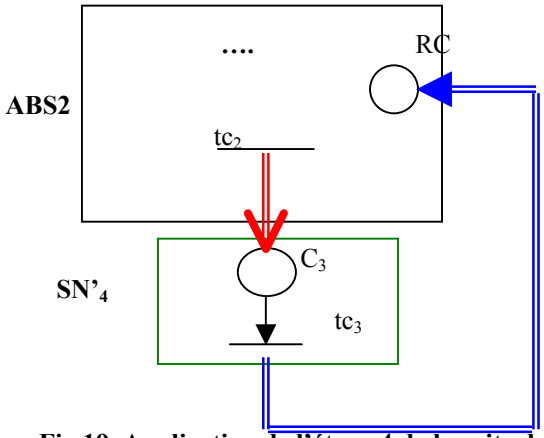


Fig.19. Application de l'étape 4 de la suite de l'algorithme Decompose().

6. Vérifier l'existence de **sous-réseaux de type SN_PnT,** càd rechercher les places p ayant $d^+(p)>1$, soit $d^+(p)=k$: Dans le nouveau réseau abstrait, pas de sous-réseaux de type **SN_PnT.**

7. Vérifier l'existence de **sous-réseaux de type SN_nTP,** càd rechercher les places p dont $d^-(p)>1$, soit $d^-(p)=k$. Dans le nouveau réseau abstrait, pas de sous-réseaux de type **SN_nTP.**

8. Vérifier l'existence de **sous-réseaux de type SN_TnP,** càd rechercher les transitions t dont $d^+(t)>1$, soit $d^+(t)=k$. Dans le nouveau réseau abstrait, pas de sous-réseaux de type **SN_TnP.**

9. Fin de l'application de l'algorithme.

Expression déduite du réseau N

En utilisant l'expression trouvée et les relations déduites du dernier arbre de liens, on déduit deux expressions globales de N, déterminant deux comportements possibles du système, et ce, comme suit :

1ere expression : $\blacktriangledown_{\{tc2\}} \{ \blacktriangledown_{\{tc1\}} [\blacktriangleright\blacktriangleleft_{\{tc1\}} [[\blacktriangledown_{\{tp2\}} [\blacktriangleright\blacktriangleleft_{\{tp2\}} (SN'_0 \cup S_1 , SN'_3 \cup S_2, SN'_2 \cup S_3), SN'_2] \cup S_4 , SN'_4 \cup S_5] , [\blacktriangledown_{\{tp2\}} [\blacktriangleright\blacktriangleleft_{\{tp2\}} (SN'_0 \cup S_1, SN'_3 \cup S_2, SN'_2 \cup S_3), SN'_2]], SN'_4 \}$

2nde expression : $\blacktriangledown_{\{tc3\}} \{ SN'_4 , \blacktriangledown_{\{tc1\}} [\blacktriangleright\blacktriangleleft_{\{tc1\}} [[\blacktriangledown_{\{tp2\}} [\blacktriangleright\blacktriangleleft_{\{tp2\}} (SN'_0 \cup S_1 , SN'_3 \cup S_2, SN'_2 \cup S_3), SN'_2] \cup S_4 , SN'_4 \cup S_5] , [\blacktriangledown_{\{tp2\}} [\blacktriangleright\blacktriangleleft_{\{tp2\}} (SN'_0 \cup S_1, SN'_3 \cup S_2, SN'_2 \cup S_3), SN'_2]] \}$

6. Conclusion

Dans ce rapport, une nouvelle technique de décomposition de RDPS a été proposée. L'objectif principal de cette approche est de réduire l'impact de l'explosion de l'espace d'états, qui peut survenir lors de l'analyse des performances d'un système. Grâce à cette méthode, il est possible de représenter le comportement d'un système par une expression algébrique, qui lie les composants du système (sous-réseaux). Cette expression peut être utilisée pour faciliter le calcul de la solution globale d'un RDPS, qui sera en fonction des solutions de composants, évitant ainsi l'analyse du système global.

Comme notre objectif est de simplifier l'analyse des performances, le calcul de la solution globale à l'aide de l'expression algébrique fera l'objet d'une étude future. Les propriétés qualitatives du système

global peuvent également être étudiées pour déduire des propriétés à partir de celles de sous-réseaux et de l'expression algébrique liant les sous-réseaux.

Bibliographie

[1] Buchholz, P. 1992a. "Hierarchical markovian models- symmetries and reduction". In *Proceedings of the 6th International Conference on Modeling Techniques and Tools for Computer Performance evaluation,* (Edinburgh, Scotland, UK, Sept.15-18). 234-246.

[2] Buchholz, P. 1992b. "A hierarchical view of GCSPNs and its impact on qualitative and quantitative analysis". *Journal of Parallel and Distributed Computing*, 15(2). 207-224.

[3] Buchholz, P. 1993a. "Hierarchies in colored GSPNs". In *Proceedings Of the 14th International Conference on Application and Theory of Petri nets*. N°691 in LNCS. (Chicago, Illinois, USA, Jun.1993). Springer-Verlag, 106-125.

[4] Buchholz, P. 1993b. "Aggregation and reduction techniques for hierarchical GCSPN". In *Proceedings Of the 5th International Workshop on Petri Nets and Performance Models*. (Toulouse, France, Oct.19-22). IEEE Computer Society Press, 216-225.

[5] Buchholz, P. 1997. "Hierarchical structuring of superposed GSPNs". IEEE, 81-90.

[6] Davio, M. 1981. "Kronecker products and shuffle algebra ". 'Feb.1981). IEEE Trans. on Computers. C-30(2).

[7] Donatelli, S. and Sereno, M. 1991. "On the product form solution for Stochastic Petri Nets". IEEE, 154-172.

[8] Donatelli, S. 1994. "Superposed generalized Stochastic Petri nets : definition and efficient solution". In Proceedings Of the 15th International Conference on Application and Theory of Petri Nets. (Zaragoza, Spain, Jun.20-24). Springer-Verlag, Robert Valette Editor, N°815 in LNCS, 258-277.

[9] Dutheillet, C. and Haddad, S. 1989. "Aggregation of states in colored Petri nets : Application to a multiprocessor architecture". (PARIS, 1989). IEEE, 40-49.

[10] Dutheillet, C. 1992. "Symétries dans les réseaux de Petri colores: Définitions, analyse et application à l'évaluation des performances". Doctorat thesis. (Paris 6 University, March 1

[11] Florin, G. and Natkin, S. 1988. "Solutions en forme produit matriciel pour les réseaux de files d'attente synchronisés fermés monovalués". Rapport de recherche CEDRIC , N° 88.

[12] Florin, G. and Natkin, S. 1989. " Matrix product form solution for closed synchronised queuing networks". IEEE, 29-37.

[13] Haddad, S. Ilié, J.M. Taghelit, M. and Zouari, B. 1995. "Symbolic reachability graph and partial symmetries". Technical Report of Masi lab. N^0 95.30, P&M.Curie university . (Paris, Mars 95).

[14] Haddad, S. and Moreaux, P. 1997. "Aggregation and decomposition for performance evaluation of asynchronous product of high level Petri Nets". Rapport de recherche N°102, Lamsade. Université de Paris Dauphine. Mai 1997.

[15] Henderson, W. and Taylor, P.G. 1989. "Aggregation methods in exact performance analysis of Stochastic Petri Nets". IEEE. 12-17.

[16] Ioualalen, M. and Aissani, A. 1996. "Symétries dans les Réseaux de Petri Stochastiques. Construction du graphe symbolique". 6ème Atelier de l'évaluation des performances. (Versailles, Nov.96).

[17] Ioualalen, M. Boukala, M.C. and Aissani, A. 1998. "Réductions dans les Réseaux de Petri Stochastiques. Construction du graphe symbolique. Application à l'évaluation des performances". 4ème Colloque Africain sur la recherche en Informatique. (Dakar, Sénégal. Oct.12-15).

[18] Moreaux.P. 1996. "Structuration des chaînes de Markov des Réseaux de Petri Stochastiques". Thèse de doctorat. (Université Paris Dauphine, Dec.1996).

[19] Plateau, B. and Fourneau, J.M. 1991. "A methodology for solving Markov models of parallel systems". Journal of Parallel and Distributed Computing, 12. 370-387.

[20] Truffet, L. 1995. " Méthodes de calcul de bornes stochastiques sur des modèles de systèmes et de réseaux ". Doctorat thesis, Paris 6 University, Nov. 1995.

[21] Zuberek,W.M. 1989. "Performance evaluation using unbounded Timed Petri Nets". (Canada, 1989). IEEE. 180-186