

ALAIN FINKEL

LAURE PETRUCCI

## **Composition/décomposition de réseaux de Pétri et de leurs graphes de couverture**

*Informatique théorique et applications*, tome 28, n° 2 (1994),  
p. 73-124

[http://www.numdam.org/item?id=ITA\\_\\_1994\\_\\_28\\_2\\_73\\_0](http://www.numdam.org/item?id=ITA__1994__28_2_73_0)

© AFCET, 1994, tous droits réservés.

L'accès aux archives de la revue « Informatique théorique et applications » implique l'accord avec les conditions générales d'utilisation (<http://www.numdam.org/conditions>). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

Article numérisé dans le cadre du programme  
Numérisation de documents anciens mathématiques

<http://www.numdam.org/>

## COMPOSITION/DÉCOMPOSITION DE RÉSEAUX DE PETRI ET DE LEURS GRAPHES DE COUVERTURE (\*)

par Alain FINKEL <sup>(1)</sup> et Laure PETRUCCI <sup>(2)</sup>

Communiqué par W. BRAUER

---

*Résumé. – Dans cet article, nous nous intéressons à la composition et à la décomposition de réseaux de Petri via un ensemble commun de places ou de transitions. Nous montrons que certaines propriétés sont conservées par ces techniques, d'autres pouvant être obtenues par composition des graphes de couverture. Nous proposons des algorithmes permettant la décomposition d'un réseau selon les propriétés que l'on désire vérifier. Nous présentons un algorithme permettant de composer des graphes de couverture minimaux, moins coûteux qu'une construction directe, aussi bien en temps qu'en espace.*

*Abstract. – In this paper, we study composition and decomposition of Petri nets via a common set of places or transitions. We prove that some properties are preserved by these techniques, and some other ones can be obtained by the composition of the covering graphs. We propose some algorithms to decompose a net according to the properties that must be verified. We present an algorithm to compose minimal covering graphs, less expensive than a straightforward construction, concerning as well time as space.*

### INTRODUCTION

Le but de cet article est d'analyser de gros réseaux de Petri sans calculer directement leurs graphes de couverture complets même lorsque le graphe de couverture est minimal [6]. Il y a deux manières d'étudier un gros réseau de Petri : la première part du principe que l'on a composé des petits réseaux de Petri (des modules) par fusion de places et/ou par fusion de transitions, la deuxième consiste à étudier un réseau de Petri sans connaître la méthode modulaire ayant permis de l'obtenir. Nous examinerons ces deux cas et nous proposerons des méthodes d'analyse modulaire pour les deux cas envisagés.

---

(\*) Reçu en mai 1990, accepté en septembre 1993.

(1) LIFAC, École Normale Supérieure de Cachan, 61, avenue du Président Wilson, F-94235 Cachan Cedex, e-mail: finkel @ ens-cachan.fr.

(2) CEDRIC-IEE, 18, allée Jean Rostand, F-91025 Évry Cedex, e-mail: petrucci @ iie.cnam.fr.

Work supported by the Esprit project DEMON (BRA 3148) and C<sup>3</sup>.

Soit le premier cas où nous construisons un gros réseau de Petri par composition de modules. Nous utilisons alors la construction modulaire pour faire de l'analyse modulaire en calculant, en parallèle, les graphes de couverture minimaux de chacun des modules. De l'ensemble de ces graphes de couverture minimaux, on peut déduire plusieurs propriétés générales sur le réseau global; cependant, il peut être nécessaire de calculer le graphe de couverture minimal du réseau complet : nous présentons, pour ce faire, un algorithme de composition des graphes de couvertures de chacun des modules qui construit le graphe de couverture minimal complet. Le calcul du graphe de couverture minimal est ainsi réduit par rapport à la construction directe du graphe de couverture minimal complet. Si les propriétés des petits réseaux ne permettent pas de déduire les propriétés souhaitées pour le réseau total, au lieu de construire le graphe de couverture de ce dernier, on peut le considérer comme un gros réseau à décomposer en modules, et donc utiliser la démarche duale.

La démarche duale consiste à partir d'un gros réseau de Petri sans indication sur la démarche modulaire ayant servi à le construire. La personne qui a travaillé un certain temps sur un gros réseau commence à avoir certaines idées sur par exemple les places bornées et celles non bornées, et elle sait que toutes ses transitions doivent être quasi-vivantes. Soit  $p$  une place qu'on suppose bornée. Nous décomposons le grand réseau en petits réseaux communiquant par transitions communes. Prouver que la place  $p$  est bornée dans l'un de ces petits réseaux est suffisant pour inférer que  $p$  est bornée dans le réseau total. Soit maintenant  $p$  une place qu'on pense non bornée. Nous décomposons le grand réseau en petits réseaux communiquant par places communes. Prouver que la place  $p$  est non bornée dans l'un de ces petits réseaux est suffisant pour inférer que  $p$  est non bornée dans le réseau total. Par une décomposition similaire, on peut étudier la quasi-vivacité des transitions.

La composition/décomposition de réseaux, telle que nous la présentons, sert non seulement à l'analyse des réseaux places/transitions, mais aussi à l'analyse de réseaux de haut niveau comme les schémas de réseaux algébriques ([2], [5], [12]). En effet, pour étudier ces derniers, on se ramène à des modèles plus simples, et en particulier aux réseaux de Petri. Les relations entre les propriétés des modèles simples et celles des réseaux de haut niveau sont de même type que celles qui existent entre des réseaux de Petri et leur composé. Nous montrons donc comment la composition/décomposition peut être étendue aux réseaux de haut niveau et quelles propriétés peuvent être vérifiées.

## 1. DÉFINITIONS DE BASE

Rappelons que  $T^*$  désigne l'ensemble des suites finies sur  $T$ ; on appellera *séquence* une telle suite de transitions. La *longueur*  $l(s)$  d'une séquence  $s = t_1 \dots t_n$  est égale à  $n$ .

Nous rappelons brièvement la définition d'un réseau de Petri ([3], [4] et [9]).

**DÉFINITION 1.1 :** Un *réseau de Petri* PN est un quintuplet  $PN = (P, T, \text{Pre}, \text{Post}, M_0)$  où  $P$  est l'ensemble fini des places,  $T$  est l'ensemble fini des transitions, Pre et Post sont deux fonctions partielles de  $P \times T$  dans  $\mathbb{N}$  et  $M_0 \in \mathbb{N}^P$  est le marquage initial.

Une transition  $t$  est *franchissable* à partir d'un marquage  $M$  si pour toute place  $p$ ,  $M(p) \geq \text{Pre}(p, t)$ . Le marquage  $M'$ , atteint en franchissant la transition  $t$ , est calculé pour toute place  $p$ , par

$$M'(p) = M(p) - \text{Pre}(p, t) + \text{Post}(p, t).$$

On notera  $M[t > M']$  le fait que  $t$  est franchissable à partir de  $M$  et que  $M'$  est le marquage obtenu.

Une séquence  $s \in T^*$  est *franchissable* à partir de  $M$  lorsque

$$M[t_1 > M_1, M_1[t_2 > M_2, \dots, M_i[t_{i+1} > M_{i+1}, \dots, M_{n-1}[t_n > M_n$$

avec  $s = t_1 \dots t_n$ .

**DÉFINITION 1.2 :** Un marquage  $M$  est *mort* quand aucune transition est franchissable à partir de  $M$ .

L'*ensemble des états accessibles* d'un réseau de Petri est l'ensemble des marquages que l'on peut atteindre à partir du marquage initial  $M_0$ .

Une place  $p$  est *bornée* s'il existe un entier  $k \geq 0$  tel que tout marquage  $M$  de l'ensemble des états accessibles vérifie  $M(p) \leq k$ .

Une transition  $t$  est *quasi-vivante* s'il existe un marquage  $M$  dans l'ensemble des états accessibles tel que  $t$  est franchissable à partir de  $M$ .

**Exemple 1.3 :** Soit  $PN_0 = (P, T, \text{Pre}, \text{Post}, M_0)$  où  $P = \{p_1, p_2, p_3, p_4, p_5\}$ ,  $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$ , Pre et Post sont les fonctions partielles de  $P \times T$

dans  $\mathbb{N}$  définies par :

$$\text{Pre}(p_1, t_1) = \text{Pre}(p_1, t_2) = \text{Pre}(p_3, t_4) = 1$$

$$\text{et } \text{Pre}(p_2, t_3) = \text{Pre}(p_4, t_5) = \text{Pre}(p_5, t_6) = 1,$$

$$\text{Post}(p_2, t_1) = \text{Post}(p_2, t_4) = \text{Post}(p_4, t_2) = \text{Post}(p_4, t_6) = 1$$

$$\text{et } \text{Post}(p_5, t_5) = \text{Post}(p_3, t_3) = 2.$$

$M_0 = (1, 0, 0, 0, 0)$  est le marquage initial.

L'ensemble des états accessibles du réseau PNO est infini. Toutes les transitions de PNO sont quasi-vivantes car  $t_1, t_2, t_3, t_4, t_5$  et  $t_6$  sont franchissables à partir d'un marquage accessible.

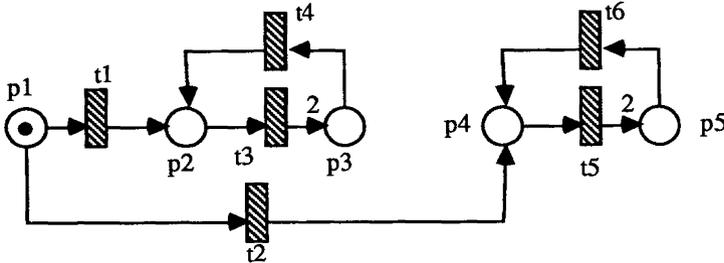


Figure 1. — Le réseau PNO

**DÉFINITION 1.4 :** L'arbre d'accessibilité d'un réseau de Petri  $PN = (P, T, \text{Pre}, \text{Post}, M_0)$  est défini de la manière suivante :

- la racine de l'arbre est étiquetée par le marquage initial  $M_0$  ;
- pour toute transition  $t$  franchissable à partir d'un nœud  $nd$  étiqueté par  $M$ , on crée un nœud  $nd'$ , étiqueté par  $M'$ , où  $M'$  est le marquage obtenu en franchissant  $t$  à partir de  $M$ . Un arc étiqueté par  $t$  relie  $nd$  et  $nd'$ .

Le graphe d'accessibilité de  $PN$  est obtenu à partir de l'arbre d'accessibilité de  $PN$  en identifiant les nœuds ayant la même étiquette.

**DÉFINITION 1.5 :** Un réseau *termine* lorsque son graphe d'accessibilité est fini et sans circuit.

## 2. COMPOSITION/DÉCOMPOSITION

La démarche d'un modélisateur consiste généralement en la conception de modules représentés par de petits réseaux, puis en l'intégration de ces derniers au sein d'un plus gros réseau représentant le système entier. Dans le cas

général, la composition de modules se fait par fusion d'un ensemble commun soit de places soit de transitions. Nous nous proposons d'examiner la conservation des propriétés par ce type de composition/décomposition.

De telles études ont déjà fait l'objet de nombreux travaux ([1], [7], [10], [11]). Nous montrerons quelles sont les propriétés d'un module qui peuvent être transmises au réseau complet, et réciproquement, nous proposerons une nouvelle façon de prouver des propriétés par extraction d'un sous-réseau. Les algorithmes de décomposition permettent la vérification d'une propriété sur le réseau complet en étudiant un module. Ces démarches évitent la construction du graphe de couverture d'un réseau important, car il suffira de vérifier les propriétés sur des réseaux plus petits.

On peut analyser les schémas de réseaux algébriques à l'aide du réseau places/transitions sous-jacent appelé squelette. Ainsi cette étude sera applicable non seulement à l'analyse des réseaux de Petri, mais aussi à celle des réseaux algébriques correspondants.

## 2.1. Vérification de propriétés par composition/décomposition

Nous nous intéressons à deux sortes de composition/décomposition de réseaux places/transitions : la fusion sur un ensemble commun de transitions et la fusion sur un ensemble commun de places. Nous commençons donc par exposer ces deux techniques.

**DÉFINITION 2.1.1 :** Soit un réseau  $R_0 = (P_0, T_0, \text{Pre}_0, \text{Post}_0, M_0)$ . La *projection* d'une séquence  $x$  de  $R_0$  sur un sous-ensemble de transitions  $T$  de  $T_0$  est la plus grande séquence extraite de  $x$  contenant uniquement des transitions de  $T$ . Cette projection sera notée  $\text{proj}_T(x)$ .

Plus formellement,  $\forall t \in T_0, \forall x \in T_0^*, \text{proj}_T(xt) = \text{proj}_T(x) \text{proj}_T(t)$ , avec  $\text{proj}_T(t) = t$  si  $t \in T$ , et  $\text{proj}_T(t) = \lambda$  sinon.

### 2.1.1. Fusion de transitions

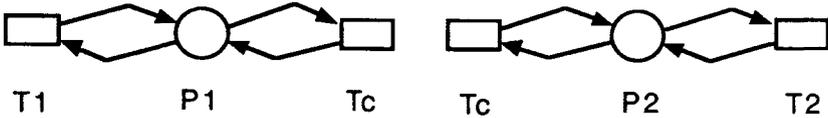
Nous commençons par définir la structure que doivent avoir deux réseaux à composer par fusion de transitions, ainsi que le résultat de cette opération.

**DÉFINITION 2.1.1.1 (fusion de transitions) :** Soient  $R_i = (P_i, T_i \cup T_c, \text{Pre}_i, \text{Post}_i, M_{0_i})$ ,  $i = 1, 2$ , deux réseaux places/transitions. L'ensemble  $T_1$  (resp.  $T_2$ ) contient les transitions propres à  $R_1$  (resp.  $R_2$ ). L'ensemble  $T_c$  contient les transitions communes aux deux réseaux.  $(P_1 \cup T_1) \cap (P_2 \cup T_2) = \emptyset$ .

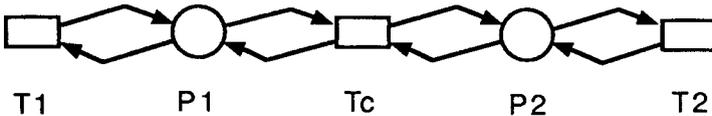
Le réseau obtenu en composant  $R_1$  et  $R_2$  par fusion de leurs transitions communes est  $R_0 = (P_1 \cup P_2, T_1 \cup T_2 \cup T_c, \text{Pre}_0, \text{Post}_0, M_0)$  tel que :

- (i)  $\forall t_c \in T_c, \forall p_1 \in P_1, \text{Pre}_0(p_1, t_c) = \text{Pre}_1(p_1, t_c), \text{Post}_0(p_1, t_c) = \text{Post}_1(p_1, t_c)$   
(dans  $R_0$ ,  $t_c$  est reliée aux places de  $P_1$  de la même manière que dans  $R_1$ );
- (ii)  $\forall t_c \in T_c, \forall p_2 \in P_2, \text{Pre}_0(p_2, t_c) = \text{Pre}_2(p_2, t_c), \text{Post}_0(p_2, t_c) = \text{Post}_2(p_2, t_c)$   
(dans  $R_0$ ,  $t_c$  est reliée aux places de  $P_2$  de la même manière que dans  $R_2$ );
- (iii)  $\forall t_1 \in T_1, \forall p_1 \in P_1, \forall p_2 \in P_2, \text{Pre}_0(p_1, t_1) = \text{Pre}_1(p_1, t_1)$  et  $\text{Pre}_0(p_2, t_1) = 0$ ,  
 $\text{Post}_0(p_1, t_1) = \text{Post}_1(p_1, t_1)$  et  $\text{Post}_0(p_2, t_1) = 0$   
(les transitions de  $T_1$  ne sont reliées qu'aux places de  $P_1$ , de la même manière que dans  $R_1$ );
- (iv)  $\forall t_2 \in T_2, \forall p_1 \in P_1, \forall p_2 \in P_2, \text{Pre}_0(p_2, t_2) = \text{Pre}_2(p_2, t_2)$  et  $\text{Pre}_0(p_1, t_2) = 0$ ,  
 $\text{Post}_0(p_2, t_2) = \text{Post}_2(p_2, t_2)$  et  $\text{Post}_0(p_1, t_2) = 0$   
(les transitions de  $T_2$  ne sont reliées qu'aux places de  $P_2$ , de la même manière que dans  $R_2$ );
- (v)  $\forall p_1 \in P_1, M_0(p_1) = M_{0_1}(p_1)$   
(le marquage initial des places de  $P_1$  est inchangé);
- (vi)  $\forall p_2 \in P_2, M_0(p_2) = M_{0_2}(p_2)$   
(le marquage initial des places de  $P_2$  est inchangé).

Nous considérons donc des réseaux  $R_1$  et  $R_2$  tels que :

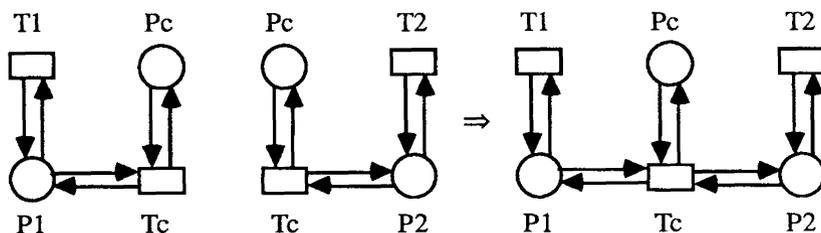


Nous obtenons, par fusion des transitions communes à  $R_1$  et  $R_2$ , le réseau  $R_0$  suivant :

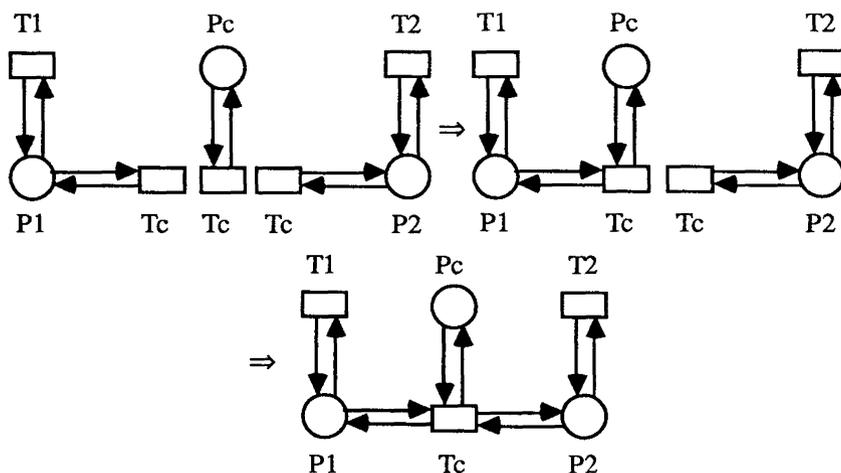


Cette approche correspond à celle de [11].

Notons que la composition de réseaux via un medium de communication ([10]) est entièrement prise en compte dans nos travaux. L'étude de Souissi concerne la composition de deux réseaux  $R_1$  et  $R_2$  ayant chacun des ensembles de places propres ( $P_1$  et  $P_2$ ), de transitions propres ( $T_1$  et  $T_2$ ), et contenant un sous réseau formé d'un ensemble de transitions communes  $T_c$  et d'un ensemble de place communes  $P_c$  ne pouvant communiquer qu'avec les transitions de  $T_c$ , selon le schéma suivant :



Dans notre cas, les places communes n'appartiennent qu'à un seul des sous-réseaux. Nous pouvons toujours nous ramener à notre schéma de composition, pour étudier la composition via un medium de communication :



Certaines propriétés du réseau composé  $R_0$  peuvent être déduites directement des propriétés des réseaux  $R_1$  et  $R_2$ , sans construction du graphe de couverture du réseau complet.

*Notation* : Tout marquage  $M$  de  $R_0$  peut être décomposé en  $M = (M_1, M_2)$  où  $M_1$  est le marquage des places de  $P_1$ , et  $M_2$  celui des places de  $P_2$ .

La proposition suivante est signalée dans [1], mais nous présentons ici une preuve détaillée.

PROPOSITION 2.1.1.2 : Soit  $x$  une séquence de franchissements de  $R_0$ . La projection de  $x$  sur  $T_1 \cup T_c$  est une séquence de franchissements de  $R_1$ . La projection de  $x$  sur  $T_2 \cup T_c$  est une séquence de franchissements de  $R_2$ .

*Preuve* : Soit  $x$  une séquence de franchissements de  $R_0$ . On décompose  $x$  en  $x = x_1 x'_1 t_1 \dots x_n x'_n t_n$  où les  $x_i$  sont des séquences de franchissements de  $T_1$ , les  $x'_i$  des séquences de franchissements de  $T_2$  et les  $t_i$  des transitions de  $T_c$ . Les réseaux  $R_1$  et  $R_2$  jouant des rôles symétriques, nous prouvons la proposition pour le réseau  $R_1$  uniquement. Nous effectuons une preuve par induction sur la longueur des séquences de franchissement. Soit dans  $R_0 : (M_1, M'_1)[x_1 x'_1 t_1 > (M_2, M'_2)]$ . Dans  $R_1$ ,  $x_1$  est franchissable car le franchissement de  $x_1$  dépend uniquement des places de  $P_1$ . Seul le marquage de ces places est modifié. On a donc  $(M_1, M'_1)[x_1 > (m_1, M'_1)]$ . De même pour  $x'_1$  qui ne fait appel qu'aux places de  $P_2$ . Donc  $(m_1, M'_1)[x'_1 > (m_1, m'_1)]$ . Par conséquent, la projection de  $x_1 x'_1$  sur  $T_1 \cup T_c$  est  $x_1$  et  $M_1[\text{proj}_{T_1 \cup T_c}(x_1 x'_1) > m_1]$ . Dans  $R_0$ , le marquage  $(m_1, m'_1)$  permet le franchissement de  $t_1$ . Donc les places de  $P_1$  en particulier ont un marquage suffisamment grand pour ce faire. C'est pourquoi  $t_c$  est franchissable à partir de  $m_1$  dans  $R_1 : (m_1, m'_1)[t_1 > (M_2, M'_2)]$  dans  $R_0$  et  $m_1[t_1 > M_2]$  dans  $R_1$ . ■

Nous pouvons déduire des propriétés du réseau composé à partir de celles des sous-réseaux :

PROPOSITION 2.1.1.3 : (i) une transition  $t$  est non quasi-vivante dans  $R_1$  ou dans  $R_2 \Rightarrow t$  est non quasi-vivante dans  $R_0$  ;

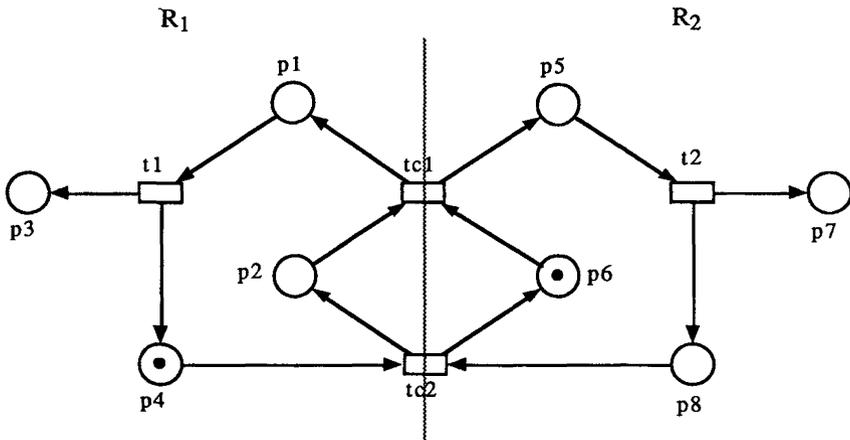
(ii) une place  $p$  est  $k$ -bornée dans  $R_1$  ou dans  $R_2 \Rightarrow p$  est  $k$ -bornée dans  $R_0$  ;

(iii)  $M_1$  est un marquage mort dans  $R_1$  et  $M_2$  est un marquage mort dans  $R_2 \Rightarrow (M_1, M_2)$  est un marquage mort dans  $R_0$  ;

(iv)  $R_1$  et  $R_2$  terminent  $\Rightarrow R_0$  termine.

*Preuve* : Immédiat d'après la proposition 2.1.1.2. ■

Nous présentons un contre-exemple aux réciproques des propositions précédentes :



Le marquage initial de  $R_0$  est mort alors que les réseaux  $R_1$  et  $R_2$  sont vivants (réciproques de (iii) et (iv)). Donc toutes les transitions sont non quasi-vivantes dans  $R_0$ , et elles sont toutes quasi-vivantes dans  $R_1$  et  $R_2$  (réciproque de (i)). Toutes les places sont bornées dans  $R_0$ , mais  $p_3$  n'est pas bornée dans  $R_1$  et  $p_7$  n'est pas bornée dans  $R_2$  (réciproque de (ii)).

PROPOSITION 2.1.1.4 : On suppose que  $T_c = \{t_c\}$  ( $T_c$  est un singleton).

- (i)  $R_0$  est vivant  $\Leftrightarrow R_1$  et  $R_2$  sont vivants ;
- (ii)  $(M_1, M_2)$  est un marquage mort dans  $R_0 \Rightarrow M_1$  est un marquage mort dans  $R_1$  ou  $M_2$  est un marquage mort dans  $R_2$  ;
- (iii)  $R_0$  termine  $\Rightarrow R_1$  termine ou  $R_2$  termine.

*Preuve:* (ii) : soit  $(M_1, M_2)$  un marquage mort dans  $R_0$ . Soit  $t_1$  une transition de  $T_1$ . Les entrées de  $t_1$  étant uniquement des places de  $P_1$ ,  $M_1$  ne permet pas de franchir  $t_1$  dans  $R_1$ . De même, soit  $t_2$  une transition de  $T_2$ . Les entrées de  $t_2$  étant uniquement des places de  $P_2$ ,  $M_2$  ne permet pas de franchir  $t_2$  dans  $R_2$ . Reste à examiner le cas de la transition  $t_c$ . Supposons que  $t_c$  soit franchissable dans  $R_1$  et dans  $R_2$ . Alors le marquage  $(M_1, M_2)$  permet de franchir  $t_c$  dans  $R_0$ . Ceci est en contradiction avec l'hypothèse  $(M_1, M_2)$  marquage mort dans  $R_0$ . On en déduit que le marquage  $M_1$  dans  $R_1$  ou le marquage  $M_2$  dans  $R_2$  ne permet pas de franchir  $t_c$ . Dans tous les cas possibles, les transitions ne peuvent être franchies, donc  $M_1$  est mort dans  $R_1$  ou  $M_2$  est mort dans  $R_2$ .

(iii) : supposons que  $R_1$  et  $R_2$  ne terminent pas. Nous nous intéressons à la transition  $t_c$  qui est la seule à pouvoir amener à un marquage mort dans  $R_0$  mais pas dans  $R_1$  ni dans  $R_2$ . Deux cas se présentent : soit il existe  $k$  tel

que  $t_c$  est  $k$ -bornée dans l'un des deux sous-réseaux, soit elle n'est bornée dans aucun des deux sous-réseaux.

Supposons, pour l'instant, que  $t_c$  est  $k$ -bornée dans un des sous-réseaux, par exemple  $R_1$  (elle peut être franchie au maximum  $k$  fois), et que dans  $R_2$  elle n'est pas  $(k-1)$ -bornée (on peut la franchir au moins  $k$  fois). Les réseaux  $R_1$  et  $R_2$  jouant des rôles symétriques, la restriction à ce cas suffit pour prouver la proposition sur toutes les possibilités de « $t_c$  est bornée dans l'un des deux sous-réseaux».  $R_1$  ne terminant pas, il existe donc une séquence infinie de franchissements de  $R_1$  qui s'écrit sous la forme  $M_{0_1}[s > M_1[y^* >$  où  $s$  contient au plus  $k$  occurrences de  $t_c$  et  $y$  est une séquence répétitive de  $T_1$ . On peut écrire  $s = x_1 t_c \dots t_c x_n t_c x_{n+1}$  où  $n \leq k$ . Dans  $R_2$ ,  $t_c$  n'étant pas  $k$ -bornée, il existe des  $x'_i$ , séquences de franchissements de  $T_2$  telles que:  $M_{0_2}[x'_1 t_c \dots t_c x'_n t_c > M_2$ . Ainsi, on obtient dans  $R_0$  la séquence de franchissements:  $(M_{0_1}, M_{0_2})[x_1 x'_1 t_c \dots t_c x_n x'_n t_c > (M_1, M_2)$ ; de plus, on a  $(M_1, M_2)[y^* >$ . Donc  $(M_{0_1}, M_{0_2})[s >$ , d'où  $R_0$  ne termine pas.

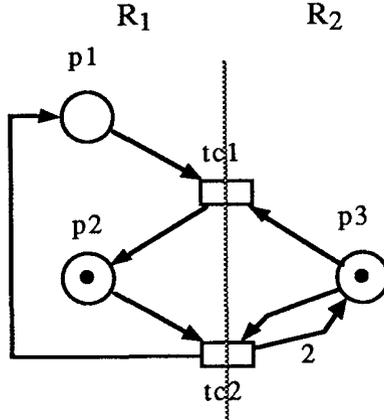
Supposons que  $t_c$  n'est bornée dans aucun des deux sous-réseaux. Autrement dit, il existe une séquence infinie contenant  $t_c$  dans  $R_1$ ,  $M_{0_1}[x_1 t_c x_2 t_c \dots >$ , et une séquence infinie dans  $R_2$ ,  $M_{0_2}[x'_1 t_c x'_2 t_c \dots >$ . La séquence  $M_0[x_1 x'_1 t_c x_2 x'_2 t_c \dots >$  est une séquence infinie de  $R_0$ . Nous avons donc montré que si  $R_1$  et  $R_2$  ne terminent pas, alors  $R_0$  ne termine pas.

(i),  $\Leftarrow$  : la démonstration utilise le même principe que celle de (iii).

(i),  $\Rightarrow$  : supposons que  $R_0$  soit vivant. Nous allons effectuer une preuve par induction sur les séquences de franchissement. Nous montrons que si  $x_1$  est une séquence de franchissements dans  $R_1$ , alors il existe une séquence de franchissements  $x$  dans  $R_0$  telle que  $\text{proj}_{T_1 \cup T_c}(x) = x_1$ . Soit  $x_1$  une séquence de franchissements de  $R_1$  ne contenant pas  $t_c$ . La séquence  $x_1$  est franchissable dans  $R_0$ . Soit  $x'_1$  une séquence de franchissements de  $R_1$  contenant  $t_c$ . La transition  $t_c$  étant vivante dans  $R_0$ , il existe une séquence de franchissements  $x_2$  de  $R_2$  telle que  $M_{0_2}[x_2 t_c >$  et dans  $R_0$ ,  $M_0[x'_1 x_2 t_c >$ . On en déduit que toute séquence de franchissements de  $R_1$  est la projection d'une séquence de franchissements de  $R_0$ . De même, toute séquence de franchissements de  $R_2$  est la projection d'une séquence de franchissements de  $R_0$ . Donc  $R_1$  et  $R_2$  sont vivants. ■

Le contre-exemple précédent permet de montrer que (ii), (iii) et la réciproque de (i) sont fausses, dans le cas où  $T_c$  n'est pas un singleton. On a  $T_c = \{tc1, tc2\}$ , le réseau  $R_0$  est mort alors que les réseaux  $R_1$  et  $R_2$  sont vivants.

Nous présentons un second contre-exemple qui permet de montrer que si  $T_c$  n'est pas un singleton, l'implication de (i) est fausse.



Les réseaux  $R_0$  et  $R_1$  sont vivants, mais pas  $R_2$ .

Le graphe de couverture d'un réseau places/transitions permet de décider de ces propriétés, ou de leur négation. Puisque l'on ne peut pas toujours déduire les propriétés du réseau composé de celles des réseaux de départ, nous nous intéresserons plus loin à la construction du graphe de couverture de  $R_0$  par composition des graphes de couverture de  $R_1$  et  $R_2$ .

### 2.1.2. Fusion de places

Dans ce paragraphe, nous nous intéressons à la fusion de deux réseaux ayant en commun un ensemble de places. La définition de la composition par fusion de places communes est similaire à celle de la fusion de transitions.

DÉFINITION 2.1.2.1 (fusion de places) : Soient

$$R_i = (P_i \cup P_c, T_i, \text{Pre}_i, \text{Post}_i, M_{0_i}), \quad i = 1, 2,$$

deux réseaux places/transitions tels que  $\forall p_c \in P_c, M_{0_1}(p_c) = M_{0_2}(p_c)$  (les places de  $P_c$  ont le même marquage initial dans  $R_1$  et dans  $R_2$ ).

L'ensemble  $P_1$  (resp.  $P_2$ ) contient les places propres à  $R_1$  (resp.  $R_2$ ). L'ensemble  $P_c$  contient les places communes aux deux réseaux.  $(P_1 \cup T_1) \cap (P_2 \cup T_2) = \emptyset$ .

Le réseau obtenu en composant  $R_1$  et  $R_2$  par fusion de leurs places communes est  $R_0 = (P_1 \cup P_2 \cup P_c, T_1 \cup T_2, \text{Pre}_0, \text{Post}_0, M_0)$  tel que :

$$(i) \quad \forall p_c \in P_c, \forall t_1 \in T_1, \text{Pre}_0(p_c, t_1) = \text{Pre}_1(p_c, t_1), \text{Post}_0(p_c, t_1) = \text{Post}_1(p_c, t_1)$$

(dans  $R_0$ ,  $p_c$  est reliée aux transitions de  $T_1$  de la même manière que dans  $R_1$ );

$$(ii) \quad \forall p_c \in P_c, \forall t_2 \in T_2, \text{Pre}_0(p_c, t_2) = \text{Pre}_2(p_c, t_2), \text{Post}_0(p_c, t_2) = \text{Post}_2(p_c, t_2)$$

(dans  $R_0$ ,  $p_c$  est reliée aux transitions de  $T_2$  de la même manière que dans  $R_2$ );

$$(iii) \forall p_1 \in P_1, \forall t_1 \in T_1, \forall t_2 \in T_2, \text{Pre}_0(p_1, t_1) = \text{Pre}_1(p_1, t_1) \text{ et } \text{Pre}_0(p_1, t_2) = 0, \\ \text{Post}_0(p_1, t_1) = \text{Post}_1(p_1, t_1) \text{ et } \text{Post}_0(p_1, t_2) = 0$$

(les places de  $P_1$  ne sont reliées qu'aux transitions de  $T_1$ , de la même manière que dans  $R_1$ );

$$(iv) \forall p_2 \in P_2, \forall t_1 \in T_1, \forall t_2 \in T_2, \text{Pre}_0(p_2, t_2) = \text{Pre}_2(p_2, t_2) \text{ et } \text{Pre}_0(p_2, t_1) = 0, \\ \text{Post}_0(p_2, t_2) = \text{Post}_2(p_2, t_2) \text{ et } \text{Post}_0(p_2, t_1) = 0$$

(les places de  $P_2$  ne sont reliées qu'aux transitions de  $T_2$ , de la même manière que dans  $R_2$ );

$$(v) \forall p_c \in P_c, M_0(p_c) = M_{0_1}(p_c) = M_{0_2}(p_c)$$

(le marquage initial des places de  $P_c$  est inchangé);

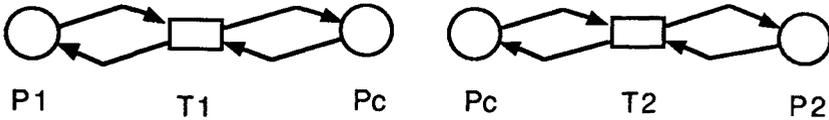
$$(vi) \forall p_1 \in P_1, M_0(p_1) = M_{0_1}(p_1)$$

(le marquage initial des places de  $P_1$  est inchangé);

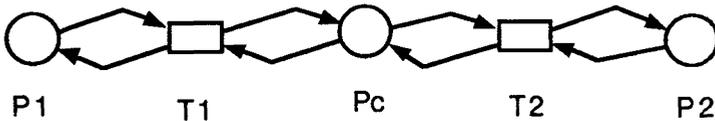
$$(vii) \forall p_2 \in P_2, M_0(p_2) = M_{0_2}(p_2)$$

(le marquage initial des places de  $P_2$  est inchangé).

Nous considérons donc des réseaux à composer  $R_1$  et  $R_2$  tels que :



Nous obtenons, par fusion des places communes de  $R_1$  et  $R_2$ , le réseau  $R_0$  suivant :



*Notation:* Tout marquage  $M$  de  $R_0$  peut être décomposé en  $M = (M_1, M_c, M_2)$  où  $M_1$  est le marquage des places de  $P_1$ , et  $M_2$  celui des places de  $P_2$  et  $M_c$  celui des places de  $P_c$ .

**PROPOSITION 2.1.2.2 :** *Toute séquence de franchissements de  $R_1$  est une séquence de franchissements de  $R_0$ . Toute séquence de franchissements de  $R_2$  est une séquence de franchissements de  $R_0$ .*

*Preuve:*  $R_1$  et  $R_2$  jouant des rôles symétriques, nous prouvons la proposition seulement pour  $R_1$ . Soit  $x_1$  une séquence de franchissements de  $R_1$  telle

que  $(M_{0_1}, M_{0_c})[x_1 > (M_1, M_c)$ . Dans  $R_0$ , on a le marquage initial  $(M_{0_1}, M_{0_c}, M_{0_2})$  où  $M_{0_1}$  est le marquage des places de  $P_1$ ,  $M_{0_2}$  celui des places de  $P_2$  et  $M_{0_c}$  le marquage des places de  $P_c$ . Les transitions de  $T_1$  n'étant pas reliées aux places de  $P_2$ , elles peuvent être franchies quel que soit le marquage  $M_{0_2}$ . D'où, dans  $R_0$ :  $(M_{0_1}, M_{0_c}, M_{0_2})[x_1 > (M_1, M_c, M_{0_2})$ . ■

Les propriétés que l'on peut déduire directement, sans construction du graphe de couverture du réseau composé sont les suivantes :

PROPOSITION 2.1.2.3 : (i) une place  $p$  est non bornée dans  $R_1$  ou dans  $R_2 \Rightarrow p$  est non bornée dans  $R_0$  ;

(ii) une transition  $t$  est quasi-vivante dans  $R_1$  ou dans  $R_2 \Rightarrow t$  est quasi-vivante dans  $R_0$  ;

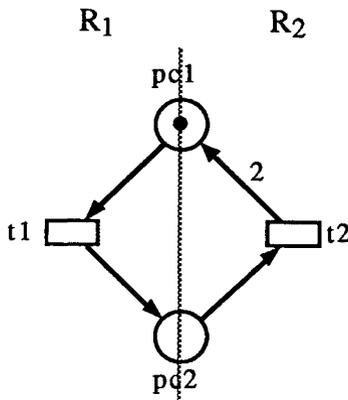
(iii)  $(M_1, M_c)$  est un marquage non mort dans  $R_1$  ou  $(M_2, M_c)$  est un marquage non mort dans  $R_2 \Leftrightarrow (M_1, M_c, M_2)$  est un marquage non mort dans  $R_0$  ;

(iv)  $R_1$  ou  $R_2$  ne termine pas  $\Rightarrow R_0$  ne termine pas.

Preuve : (i), (ii), (iv) et (iii)  $\Rightarrow$  : immédiat d'après la proposition 2.1.2.2.

(iii)  $\Leftarrow$  : soit  $(M_1, M_c, M_2)$  un marquage non mort de  $R_0$ . Alors, il existe une transition  $t_1 \in T_1$  ou  $t_2 \in T_2$  franchissable dans  $R_0$  à partir de  $(M_1, M_c, M_2)$ . La transition  $t_1$  a uniquement pour entrées des places de  $P_1$  et de  $P_c$  ; la transition  $t_2$  a uniquement pour entrées des places de  $P_2$  et de  $P_c$ . Donc  $(M_1, M_c)$  est un marquage non mort dans  $R_1$  ou  $(M_2, M_c)$  est un marquage non mort dans  $R_2$ . ■

Les réciproques des propositions (i), (ii) et (iv) sont fausses, comme on peut le constater en étudiant le contre-exemple suivant :



Les places  $pc_1$  et  $pc_2$  sont non bornées dans  $R_0$ , mais elles sont bornées dans  $R_1$  et  $R_2$  (réciproque de (i)).  $t_2$  est quasi-vivante dans  $R_0$ , mais pas

dans  $R_2$  (réciproque de (ii)). Le réseau  $R_0$  ne termine pas, mais  $R_1$  et  $R_2$  terminent (réciproque de (iv)).

De même que pour la fusion sur un ensemble commun de transitions, puisque l'on ne peut pas toujours déduire les propriétés du réseau composé de celles des réseaux de départ, nous nous intéresserons plus loin à la construction du graphe de couverture de  $R_0$  par composition des graphes de couverture de  $R_1$  et  $R_2$ .

### 2.1.3. Algorithmes de vérification de propriétés par décomposition

Les deux paragraphes précédents établissaient que la composition de réseaux conserve un certain nombre de propriétés. Dans ce paragraphe, nous nous intéressons à une démarche inverse qui consiste à extraire d'un réseau un sous-réseau conservant les propriétés du réseau initial. L'intérêt de cette démarche est évident : plus un réseau est gros, plus son analyse est complexe. Il est donc judicieux d'analyser un réseau réduit.

L'extraction d'un sous-réseau conduit à une perte d'informations. Le sous-réseau extrait ne peut donc pas être représentatif de toutes les propriétés du réseau initial. Le sous-réseau qui nous intéresse devra donc dépendre de la propriété à vérifier. Dans notre étude, nous nous sommes intéressés à trois propriétés d'un réseau : les places bornées, les places non bornées et les transitions quasi-vivantes. Pour chacune de ces trois propriétés, nous avons conçu un algorithme qui extrait d'un réseau donné un sous-réseau vérifiant cette propriété. La construction effectuée par les algorithmes de vérification du caractère borné ou non borné des places ne dépend pas du marquage initial. Il est alors nécessaire, lorsque le sous-réseau est construit, de l'analyser à l'aide d'une méthode classique, pour pouvoir valider la propriété.

#### 2.1.3.1. Vérification du caractère borné d'une place

Nous avons vu, dans la proposition 2.1.1.4, que la composition par fusion de transitions conserve le caractère borné des places. On part d'un réseau  $R_0$  dans lequel on désire montrer qu'une place  $p_0$  est bornée. Nous allons extraire un réseau  $R_1$  contenant  $p_0$  et tel que  $p_0$  soit bornée dans  $R_1$ . Le complémentaire de  $R_1$  dans  $R_0$  est  $R_2$  tel que la composition des réseaux  $R_1$  et  $R_2$  par fusion de transitions soit  $R_0$ . Si la construction échoue, c'est que l'on est pas arrivé à déterminer un sous-réseau  $R$  dans lequel la place  $p_0$  est bornée. L'algorithme que nous présentons permet de déterminer un tel sous-réseau.

Le sous-réseau  $R_1$  cherché est entièrement caractérisé par son ensemble de places : il ne communique avec le reste qu'au travers de transitions, donc

toutes les transitions connectées aux places dans le réseau complet sont des transitions de  $R_1$ .

Soit

$$R_0 = (P_0, T_0, \text{Pre}_0, \text{Post}_0, M_0).$$

Le réseau  $R_1 = (P_1, T_1, \text{Pre}_1, \text{Post}_1, M_{0_1})$  est défini par son ensemble de places  $P_1$  et par  $T_1 = \{t \in T_0 \mid \exists p \in P_1 \ tq \text{Pre}_0(p, t) \neq 0 \text{ ou } \text{Post}_0(p, t) \neq 0\}$ ,  $\forall p \in P_1, \quad \forall t \in T_1 : \quad M_{0_1}(p) = M_0(p), \quad \text{Pre}_1(p, t) = \text{Pre}_0(p, t) \text{ et } \text{Post}_1(p, t) = \text{Post}_0(p, t)$ . Le sous-réseau complémentaire  $R_2$  est alors entièrement défini :

$$R_2 = (P_2, T_2, \text{Pre}_2, \text{Post}_2, M_{0_2}) \quad \text{où } P_2 = P_0 \setminus P_1,$$

$T_2 = \{t \in T_0 \mid \exists p \in P_2 \ tq \text{Pre}_0(p, t) \neq 0 \text{ ou } \text{Post}_0(p, t) \neq 0\}$ ,  $\forall p \in P_2, \quad \forall t \in T_2 : \quad M_{0_2}(p) = M_0(p), \text{Pre}_2(p, t) = \text{Pre}_0(p, t) \text{ et } \text{Post}_2(p, t) = \text{Post}_0(p, t)$ .

On cherche s'il existe des transitions susceptibles de mettre un nombre arbitrairement grand de jetons dans la place dont on souhaite prouver le caractère borné. Une telle transition appartient à une séquence répétitive croissante. L'algorithme consiste à vérifier que les sous-réseaux situés en amont de la place étudiée sont bornés. Pour chaque transition en entrée de la place, on cherche à vérifier si la transition a elle-même une place bornée en entrée. On note, dans l'algorithme,  $P_R$  l'ensemble des places et  $T_R$  l'ensemble des transitions en cours d'examen.  $P_I$  dénote l'ensemble des places déjà examinées et pouvant faire partie d'un sous-réseau borné (résultat intermédiaire). Pour chaque sous-réseau ainsi isolé, on examine le caractère borné de la place à étudier en utilisant des méthodes classiques. La construction d'un sous-réseau s'arrête lorsque l'on a un cycle ou lorsqu'il n'y a plus de transition en amont. Il est ainsi aisé de vérifier le caractère borné d'une place.

fonction place\_bornée( $P_R, T_R, pb, p, \text{var} : P_I$ ) : booléen;

*/\* à l'appel initial, les paramètres sont :  $\emptyset, \emptyset, p_0, p_0, \emptyset^*$  /*

début

entrées :=  $\{t \notin T_R \text{ telle que } \text{Post}(p, t) > \text{Pre}(p, t)\}$ ; */\* transitions ajoutant des jetons à  $p^*$  /*

$P_R := P_R + \{p\}$ ; */\* ensemble des places utilisées dans les appels non dépilés \*/*

$T_R := T_R + \{t \text{ telle que } \text{Pre}(p, t) \neq 0 \text{ et } \text{Pre}(p, t) \geq \text{Post}(p, t)\}$ ;

*/\* on ajoute à  $T_R$  les transitions ayant  $p$  comme entrée et ne créant pas de jeton dans  $p^*$  /*

si entrées =  $\emptyset$

alors */\* on a défini un sous-réseau  $(P_R + P_I, T_R)^*$  /*

si  $pb$  est bornée dans le sous-réseau  $(P_R + P_I, T_R)$  */\* la place à laquelle l'on s'intéresse \*/*

alors */\* on a un ensemble de places potentiellement bornées \*/*

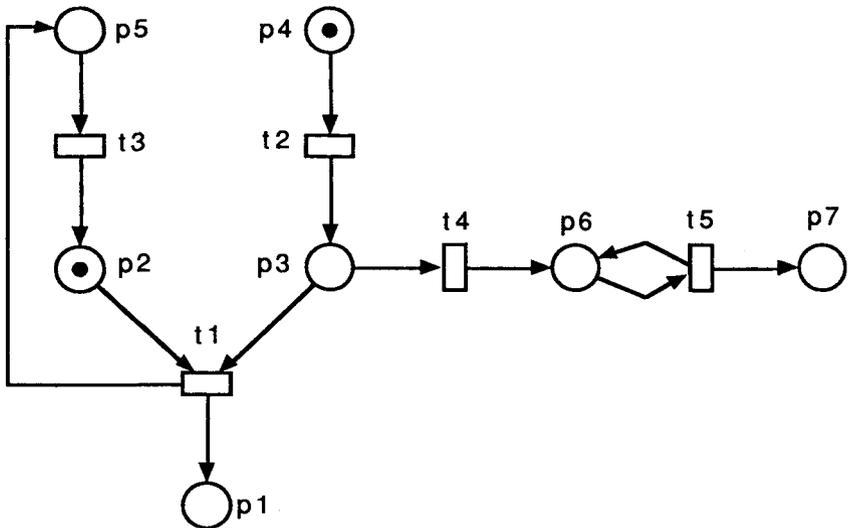
$P_I := P_R + P_I$ ;

retour (vrai);

```

    sinon (retour(faux);          /*les places choisies ne conviennent pas*/
    fsi
    sinon  $P'_I := \emptyset$ ;
    pour toute transition  $t \in \text{entrées}$  faire      /*toutes les transitions ajoutant des jetons
    à  $p^*$ /          résultat := faux;
    pour toute place  $p'$  telle que  $\text{Pre}(p', t) \neq 0$  faire /*on choisit une place en entrée de
     $t^*$ /
    résultat := place_bornée( $P_R, T_R + \{t\}, pb, p', P'_I$ ); /*la vérification est
    récursive*/
    si résultat
    alors sortie pour;          /*on sort de la boucle, car on a trouvé une place qui
    convient*/
    fsi
    fpour
    si non résultat
    alors retour(faux);        /*toutes les places en entrée de la transition sont non
    bornées*/
    fsi
    fpour
     $P_I := P'_I$ ;
    retour(vrai);              /*on a pu bloquer toutes les transitions en entrée de  $p^*$ /
    fsi
    fin
    
```

Exemple 2.1.3.1 : Nous présentons un exemple afin de détailler le fonctionnement de l’algorithme. Nous désirons vérifier que la place  $p_1$  est bornée. Pour cela, nous appelons la fonction `place_bornée`, avec comme paramètres  $P_R = \emptyset, T_R = \emptyset, pb = p_1, p = p_1, P_I = \emptyset$ .



Nous explicitons les appels de procédures et la modification des variables :

appel de place bornée( $\emptyset, \emptyset, p1, p1, \emptyset$ )

entrées =  $\{t1\}$

$P_R = \{p1\}$

$T_R = \emptyset$

$P'_i = \emptyset$

choix de  $t$  dans entrées  $t = t1$

résultat = faux

choix de  $p'$   $p' = p2$

appel de place bornée( $\{p1\}, \{t1\}, p1, p2, \emptyset$ )

entrées =  $\{t3\}$

$P_R = \{p1, p2\}$

$T_R = \{t1\}$

$P'_i = \emptyset$

choix de  $t$  dans entrées  $t = t3$

résultat = faux

choix de  $p'$   $p' = p5$

appel de place bornée( $\{p1, p2\}, \{t1, t3\}, p1, p5, \emptyset$ )

entrées =  $\emptyset$

$P_R = \{p1, p2, p5\}$

$T_R = \{t1, t3\}$

$p1$  n'est pas bornée dans  $(P_R + P_i, T_R)$

retour résultat = faux,  $P'_i = \emptyset$

finchoix

finchoix

retour résultat = faux,  $P'_i = \emptyset$

choix de  $p'$   $p' = p3$

appel de place bornée( $\{p1\}, \{t1\}, p1, p3, \emptyset$ )

entrées =  $\{t2\}$

$P_R = \{p1, p3\}$

$T_R = \{t1, t4\}$

$P'_i = \emptyset$

choix de  $t$  dans entrées  $t = t2$

résultat = faux

choix de  $p'$   $p' = p4$

appel de place bornée( $\{p1, p3\}, \{t1, t4, t2\}, p1, p4, \emptyset$ )

entrées =  $\emptyset$

$P_R = \{p1, p3, p4\}$

$T_R = \{t1, t4, t2\}$

$p1$  est bornée dans  $(P_R + P_i, T_R)$

$P_i = \{p1, p3, p4\}$

$$\begin{array}{c}
 \text{retour résultat} = \text{vrai}, P'_I = \{p1, p3, p4\} \\
 \text{sortie pour} \\
 \text{finchoix} \\
 P_I = \{p1, p3, p4\} \\
 \text{retour résultat} = \text{vrai}, P'_I = \{p1, p3, p4\} \\
 \text{sortie pour} \\
 \text{finchoix} \\
 P_I = \{p1, p3, p4\} \\
 \text{retour résultat} = \text{vrai}, P'_I = \{p1, p3, p4\}
 \end{array}$$

On a donc trouvé un sous-réseau convenable : il est composé des places  $p1$ ,  $p3$  et  $p4$  et des transitions qui leur sont connectées, c'est-à-dire  $t1$ ,  $t2$  et  $t4$ .

### 2.1.3.2. Vérification du caractère non borné d'une place

La démarche est duale de la précédente (propriété 2.1.2.3). On part d'un réseau  $R_0$  dans lequel on désire montrer qu'une place  $p_0$  est non bornée. Nous allons extraire un réseau  $R_1$  contenant  $p_0$  et tel que  $p_0$  soit non bornée dans  $R_1$ . Le complémentaire de  $R_1$  dans  $R_0$  est  $R_2$  tel que la composition des réseaux  $R_1$  et  $R_2$  par fusion de places soit  $R_0$ . Si la construction échoue, c'est que l'on n'est pas arrivé à déterminer un sous-réseau  $R$  dans lequel la place  $p_0$  est non bornée.

Le sous-réseau  $R_1$  cherché est entièrement caractérisé par son ensemble de transitions : il ne communique avec le reste qu'au travers de places, donc toutes les places connectées aux transitions dans le réseau complet sont des places de  $R_1$ .

Soit

$$R_0 = (P_0, T_0, \text{Pre}_0, \text{Post}_0, M_0).$$

Le réseau  $R_1 = (P_1, T_1, \text{Pre}_1, \text{Post}_1, M_{0_1})$  est défini par son ensemble de transitions  $T_1$  et par  $P_1 = \{p \in P_0 \mid \exists t \in T_1 \text{ tq } \text{Pre}_0(p, t) \neq 0 \text{ ou } \text{Post}_0(p, t) \neq 0\}$ ,  $\forall p \in P_1, \forall t \in T_1$  :

$$M_{0_1}(p) = M_0(p), \text{Pre}_1(p, t) = \text{Pre}_0(p, t) \quad \text{et} \quad \text{Post}_1(p, t) = \text{Post}_0(p, t).$$

Le sous-réseau complémentaire  $R_2$  est alors entièrement défini :

$$R_2 = (P_2, T_2, \text{Pre}_2, \text{Post}_2, M_{0_2}) \quad \text{où} \quad T_2 = T_0 \setminus T_1,$$

$$P_2 = \{p \in P_0 \mid \exists t \in T_2 \text{ tq } \text{Pre}_0(p, t) \neq 0 \text{ ou } \text{Post}_0(p, t) \neq 0\}, \quad \forall p \in P_2, \\ \forall t \in T_2 : M_{0_2}(p) = M_0(p), \text{Pre}_2(p, t) = \text{Pre}_0(p, t) \text{ et } \text{Post}_2(p, t) = \text{Post}_0(p, t).$$

L'idée de l'algorithme est la suivante : le sous-réseau  $R$  doit contenir au moins la place  $p_0$ , dont on veut montrer le caractère non borné. On regarde s'il existe une transition indéfiniment franchissable en entrée de la place dont on veut prouver le caractère non borné, puis on réapplique récursivement l'algorithme sur toutes les places en entrée de la transition choisie. Si on trouve en entrée de la place considérée une transition qui ne consomme pas les jetons de ses entrées, cette transition sera toujours franchissable et alors la place sera non bornée. Le détail de l'algorithme est similaire à celui utilisé pour vérifier le caractère borné d'une place.

Ici, nous divisons l'algorithme en deux parties : nous cherchons les propriétés d'une place en effectuant de la fusion sur les places communes. La place à laquelle nous nous intéressons appartient forcément au sous-réseau. La *fonction appelante* est donc une étape de *place-non-bornée* dans laquelle seule cette place est traitée.

fonction appelante :

début

résultat := faux ;

transitions := {  $t$  telle que  $\text{Post}(p_0, t) > \text{Pre}(p_0, t)$  } ; /\* ensemble des transitions ajoutant des jetons à  $p_0$  \*/

tant que résultat = faux et que transitions  $\neq \emptyset$  faire

    sélectionner  $t$  dans transitions ;                    /\* on choisit une transition qui crée des jetons dans  $p_0$  \*/

    transitions := transitions - {  $t$  } ;

$T := \emptyset$  ;

    résultat := place\_non\_bornée({  $p_0$  },  $\emptyset$ ,  $p_0$ ,  $t$ ,  $T$ ) ;

ftq

retour (résultat) ;                    /\* s'il existe une transition telle que résultat = vrai, la place  $p_0$  est potentiellement non bornée. Le sous-réseau cherché est formé des transitions de  $T$  et de toutes les places qui y sont connectées \*/

fin

fonction place\_non\_bornée( $P_R, T_R, pb, t, \text{var} : T_I$ ) : booléen ;

début

entrées := {  $p \notin P_R$  telle que  $\text{Pre}(p, t) \neq 0$  et  $\text{Pre}(p, t) \geq \text{Post}(p, t)$  et  $\text{Post}(p, \cdot) \neq 0$  } ;

/\* places en entrée de  $t$ , dont le contenu ne croît pas avec le franchissement de  $t$ , et qui sont sortie d'une transition \*/

$P_R := P_R + \{ p \text{ telle que } \text{Post}(p, t) > \text{Pre}(p, t) \text{ ou } (\text{Pre}(p, t) \neq 0 \text{ et } \text{Post}(p, \cdot) = 0) \}$  ;

/\* places en sortie de  $t$ , dans lesquelles  $t$  crée des jetons

ou places en entrée de  $t$ , dans lesquelles aucun jeton ne peut être créé \*/

$T_R := T_R + \{ t \}$  ;

/\* transitions utilisées dans des appels non dépilés \*/

si entrées =  $\emptyset$

alors /\* on a défini un sous-réseau ( $P_R, T_R + T_I$ ) \*/

```

    si pb est non bornée dans ( $P_R, T_R + T_I$ )      /* la place à laquelle l'on s'intéresse*/
    alors /*on a un résultat intermédiaire*/
         $T_I := T_R + T_I$ ;
        retour(vrai);
    sinon retour(faux);          /*les transitions choisies ne conviennent pas*/
    fsi
sinon  $T_I := \emptyset$ ;
    pour toute place  $p \in$ entrées faire /*toutes les places susceptibles de limiter le franchissement de  $t^*$ */
        résultat := faux;
        pour toute transition  $t'$  telle que  $\text{Post}(p, t') \neq 0$  faire/*choix d'une transition en entrée de  $p^*$ */
            résultat := place_non_bornée( $P_R + \{p\}, T_R, pb, t', T_I$ );
            si résultat
                alors sortie pour;      /*sortie de la boucle, car on a trouvé une transition qui convient*/
            fsi
        fpour
        si non résultat
            alors retour(faux);      /*la place ne peut pas devenir non bornée*/
        fsi
    fpour
 $T_I := T_I$ ;
    retour(vrai);      /*il est possible de rendre infini le marquage de toutes les places en entrée de  $t^*$ */
    fsi
fin

```

*Exemple 2.1.3.2* : Nous reprenons le réseau de l'exemple 2.1.3.1. Nous désirons vérifier que la place  $p7$  n'est pas bornée.

```

fonction appelante
    résultat = faux
    transitions = {  $t5$  }
    choix de  $t$  dans transitions  $t = t5$ 
        transitions =  $\emptyset$ 
         $T = \emptyset$ 
        appel de place non bornée( $\{p7\}, \emptyset, p7, t5, \emptyset$ )
        entrées = {  $p6$  }
         $P_R = \{p7\}$ 
         $T_R = \{t5\}$ 
         $T_I = \emptyset$ 

```

choix de  $p$  dans entrées  $p = p6$

résultat = faux

choix de  $t'$   $t' = t5$

appel de place non bornée ( $\{p7, p6\}, \{t5\}, p7, t5, \emptyset$ )

entrées =  $\emptyset$

$P_R = \{p7, p6\}$

$T_R = \{t5\}$

$p7$  est bornée dans  $(P_R, T_R + T_I)$

retour résultat = faux,  $T'_I = \emptyset$

choix de  $t'$   $t' = t4$

appel de place non bornée ( $\{p7, p6\}, \{t5\}, p7, t4, \emptyset$ )

entrées =  $\{p3\}$

$P_R = \{p7, p6\}$

$T_R = \{t5, t4\}$

$T'_I = \emptyset$

choix de  $p$  dans entrées  $p = p3$

résultat = faux

choix de  $t'$   $t' = t2$

appel de place non bornée ( $\{p7, p6, p3\}, \{t5, t4\}, p7, t2, \emptyset$ )

entrées =  $\emptyset$

$P_R = \{p7, p6, p3, p4\}$

$T_R = \{t5, t4, t2\}$

$p7$  n'est pas bornée dans  $(P_R, T_R + T_I)$

$T_I = \{t5, t4, t2\}$

retour résultat = vrai,  $T_I = \{t5, t4, t2\}$

sortie pour

finchoix

$T_I = \{t5, t4, t2\}$

retour résultat = vrai,  $T_I = \{t5, t4, t2\}$

sortie pour

finchoix

$T_I = \{t5, t4, t2\}$

retour résultat = vrai,  $T = \{t5, t4, t2\}$

finchoix

retour résultat = vrai,  $T = \{t5, t4, t2\}$

On a donc trouvé un sous-réseau convenable : il est composé des transitions  $t2, t4$  et  $t5$  et des places qui leur sont connectées, c'est-à-dire  $p3, p4, p6$  et  $p7$ .

### 2.1.3.3. Vérification de la quasi-vivacité d'une transition

Une transition non quasi-vivante peut représenter, dans les cas pratiques, une erreur de modélisation. Le problème de la quasi-vivacité d'une transition, comme celui des places non bornées, est lié à la fusion par places. Nous donnons l'algorithme suivant pour déterminer un sous-réseau dans lequel une transition donnée est quasi-vivante.

Au départ, le sous-réseau contient la transition dont on veut montrer la quasi-vivacité, et les places qui y sont connectées. Les places qui ont le nombre de jetons suffisant pour franchir la transition ne posent pas de problème. On essaie donc de voir s'il est possible de créer des jetons dans les autres places en franchissant une de leurs transitions en entrée et les places qui y sont connectées. On continue récursivement.

fonction transition\_quasi\_vivante( $P_R, T_R, tqv, t, \text{var} : T_I$ ): booléen;

/\* à l'appel les paramètres sont :  $\emptyset, \emptyset, t_0, t_0, \emptyset^*$  /

début

entrées :=  $\{p \notin P_R \text{ telle que } \text{Pre}(, t) > M_0(p)\}$ ; /\*places en entrée de  $t$ , pouvant empêcher le franchissement\*/

$P_R := P_R + \{p \text{ telle que } (\text{Pre}(p, t) \neq 0 \text{ et } \text{Pre}(p, t) \leq M_0(p)) \text{ ou } \text{Post}(p, t) \neq 0\}$ ;

/\* on ajoute à  $P_R$  les places connectées à  $t$  mais n'empêchant pas son franchissement\*/

$T_R := T_R + \{t\}$ ; /\*transitions utilisées dans les appels non dépilés\*/

si entrées =  $\emptyset$

alors /\*on a défini un sous-réseau ( $P_R, T_R + T_I$ )\*/

si  $tqv$  est quasi-vivante dans ( $P_R, T_R + T_I$ ) /\*la transition à laquelle l'on s'intéresse\*/

alors /\*on a un résultat intermédiaire\*/

$T_I := T_R + T_I$ ;

retour (vrai);

sinon retour (faux); /\*les transitions choisies ne conviennent pas\*/

fsi

sinon  $T_I := \emptyset$ ;

pour toute place  $p \in \text{entrées}$  faire /\*on examine les places ne permettant pas le franchissement de  $t$ \*/

résultat := faux;

pour toute transition  $t'$  telle que  $\text{Post}(p, t') \neq 0$  faire /\*choix d'une transition en entrée de  $p$ \*/

résultat := transition\_quasi\_vivante ( $P_R + \{p\}, T_R, tqv, t', T_I$ );

si résultat

alors sortie pour; /\*sortie de la boucle, car on a trouvé une transition qui convient\*/

fsi

```

    fpour
    si non résultat
        alors retour (faux); /*la place ne permet pas de franchir t*/
    fsi
    fpour
     $T_I := T'_I;$ 
    retour (vrai); /*la transition t est quasi-vivante*/
fsi
fin

```

L'algorithme est similaire à celui permettant de vérifier le caractère non borné d'une place. Les places pouvant poser problème sont celles dont le marquage initial ne permet pas de franchir la transition que l'on examine.

*Remarque :* Lorsque l'on veut vérifier une propriété donnée, on étudie un sous-réseau. Il peut aussi permettre de montrer des propriétés sur d'autres places ou transitions.

*Exemple 2.1.3.3 :* Nous reprenons le réseau de l'exemple 2.1.3.1. Nous désirons vérifier que la transition  $t1$  est quasi-vivante.

appel de transition quasi vivante ( $\emptyset, \emptyset, t1, t1, \emptyset$ )

entrées =  $\{p3\}$

$P_R = \{p1, p2, p5\}$

$T_R = \{t1\}$

$T'_I = \emptyset$

choix de  $p$  dans entrées  $p = p3$

résultat = faux

choix de  $t'$   $t' = t2$

appel de transition quasi vivante ( $\{p1, p2, p5, p3\}, \{t1\}, t2, \emptyset$ )

entrées =  $\emptyset$

$P_R = \{p1, p2, p5, p3, p4\}$

$T_R = \{t1, t2\}$

$t1$  est quasi-vivante dans  $(P_R, T_R + T_I)$

$T_I = \{t1, t2\}$

retour résultat = vrai,  $T'_I = \{t1, t2\}$

sortie pour

finchoix

$T_I = \{t1, t2\}$

retour résultat = vrai,  $T'_I = \{t1, t2\}$

On a donc trouvé un sous-réseau convenable : il est composé des transitions  $t1$  et  $t2$  et des places qui leur sont connectées, c'est-à-dire  $p1, p2, p3, p4$  et  $p5$ .

## 2.2. Extension des propriétés de la composition aux réseaux algébriques

Rappelons tout d'abord la définition d'un réseau algébrique.

DÉFINITION 2.2.1 [5] : Une *spécification de types abstraits algébriques* est un triplet  $\text{SPEC} = (\text{SO}, \text{OP}, \text{EQ})$  où :  $\text{SO}$  est un ensemble de noms de domaines,  $\text{OP}$  est un ensemble de noms d'opérations,  $\text{EQ}$  est un ensemble d'équations sur  $(\text{SO}, \text{OP})$ .

DÉFINITION 2.2.2 [5] : Un *schéma de réseau algébrique* est un  $n$ -uplet  $N = (\text{SPEC}, X, P, T, \text{Pre}, \text{Post}, \text{eqns}, \text{sort})$  où :

(i)  $\text{SPEC} = (\text{SO}, \text{OP}, \text{EQ})$  est une spécification de types abstraits algébriques ;

(ii)  $X$  est une famille de variables typées dans  $\text{SO}$  ;

(iii)  $P$  et  $T$  sont les ensembles de places et de transitions ;

(iv)  $\text{Pre}$  et  $\text{Post}$  sont les fonctions de pré et post-conditions, qui à chaque couple (place, transition) associent un terme de la spécification, à variables dans  $X$  ;

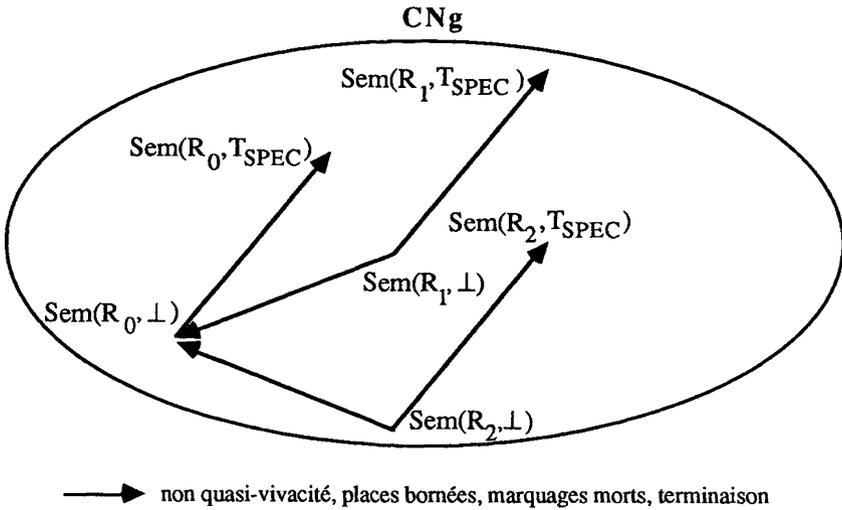
(v)  $\text{eqns}$  est une fonction qui associe à chaque transition  $t$  une équation dont les variables sont dans  $\text{Pre}(\cdot, t)$  ;

(vi)  $\text{sort} : P \rightarrow \text{SO}$  associe un type à chaque place.

DÉFINITION 2.2.3 [5] : Un *réseau algébrique* est un couple  $(N, A)$ , où  $N$  est un schéma de réseau algébrique et  $A$  une SPEC-algèbre.

Un réseau algébrique est donc un réseau de Petri dans lequel les jetons transportent une valeur (terme d'une spécification de types abstraits algébriques). Le franchissement d'une transition est conditionné par des équations sur la spécification algébrique, et modifie les valeurs des jetons en fonction de la valuation des arcs.

On peut analyser un schéma de réseaux algébriques, noté  $\text{Sem}(R, T_{\text{SPEC}})$ , à partir d'un modèle plus simple, à savoir un réseau places/transitions appelé squelette, noté  $\text{Sem}(R, \perp)$ , ([5], [12]). Les propriétés de non quasi-vivacité d'une transition, place bornée, marquage mort et terminaison, si elles sont vraies pour le squelette, le sont aussi pour le réseau algébrique. On peut donc schématiser l'héritage des propriétés comme suit, avec  $R_0$  composé de  $R_1$  et de  $R_2$  par fusion de transitions :



Lorsque l'on veut étudier une propriété d'un réseau algébrique  $R_0$ , on le décompose en sous-réseaux  $R_1, \dots, R_k$ , qui n'ont en commun que des transitions. Si le squelette d'un de ces réseaux vérifie la propriété à laquelle on s'intéresse, alors le réseau algébrique  $R_0$  la vérifie aussi. La décomposition doit être effectuée judicieusement par rapport à la propriété que l'on veut vérifier. Si l'on n'arrive pas à prouver la propriété sur les squelettes des réseaux  $R_1, \dots, R_k$ , on compose leurs graphes de couverture grâce aux algorithmes que nous exposerons dans la section 4.1, de manière à obtenir le graphe de couverture du squelette de  $R_0$ , puis on essaie de valider la propriété à l'aide de ce graphe.

### 2.3. Un exemple : le canal général

Nous nous intéressons maintenant à l'exemple du canal général, qui représente un schéma relativement complexe de producteurs/consommateurs [8]. Le réseau algébrique (*fig. 2*) peut être analysé à partir de son squelette (*fig. 3*).

Nous allons montrer, par décomposition, que les places  $p1, p2, p3, p4, p5, p6, p7, p8, p9, nbfileval, nbDepos, nbPrise$  et  $Per$  sont bornées dans le squelette et par conséquent dans le réseau algébrique, tandis que les places  $repp, repc, int, Depos, Prise$  et  $Fileval$  sont non bornées dans le squelette (on ne peut donc pas conclure quant à leur caractère borné dans le réseau algébrique).

Les figures 4, 5, 6 et 7 montrent des décompositions avec transitions communes. Elles permettent donc de prouver que certaines places sont bornées. Les figures 8, 9, 10 et 11 montrent des décompositions avec places communes, permettant de déduire le caractère non borné d'autres places. Nous avons essayé d'obtenir les sous-réseaux les plus petits possibles permettant d'établir les propriétés désirées.

Nous présentons des décompositions avec transitions communes. La figure 4, qui représente quatre sous-réseaux, montre que les places  $p_1$ ,  $p_2$ ,  $p_8$ ,  $p_9$ ,  $nbfileval$  et  $nbDepos$  sont bornées. Dans la figure 5, la place  $nbPrise$  est bornée, de même que la place  $Per$  dans la figure 6, les places  $p_3$ ,  $p_4$ ,  $p_5$ ,  $p_6$  et  $p_7$  dans la figure 7.

Maintenant, nous examinons des sous-réseaux obtenus par décompositions avec des places communes. La figure 8 permet d'établir que les places  $repp$ ,  $int$  et  $Depos$  ne sont pas bornées. La figure 9 nous montre en plus que la place  $Prise$  n'est pas bornée, de même que la place  $Fileval$  dans la figure 10 et la place  $repc$  dans la figure 11.

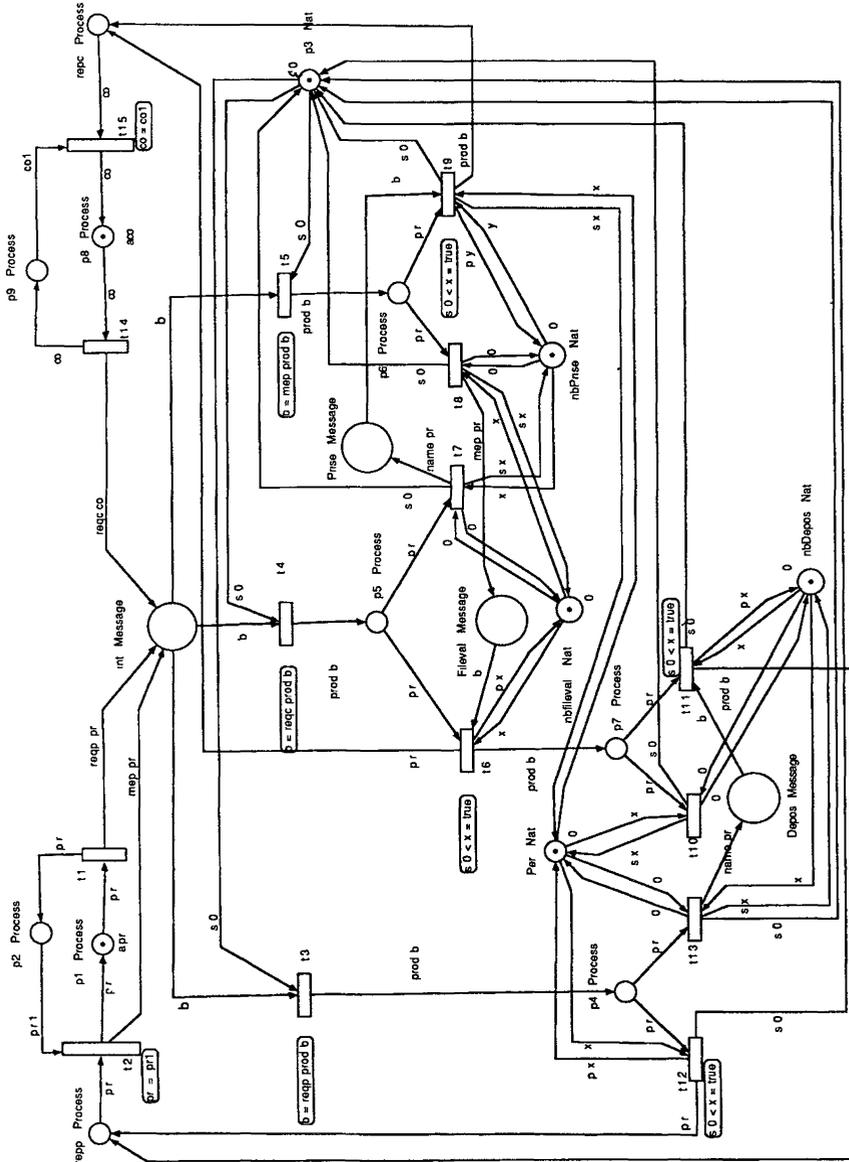


Figure 2.

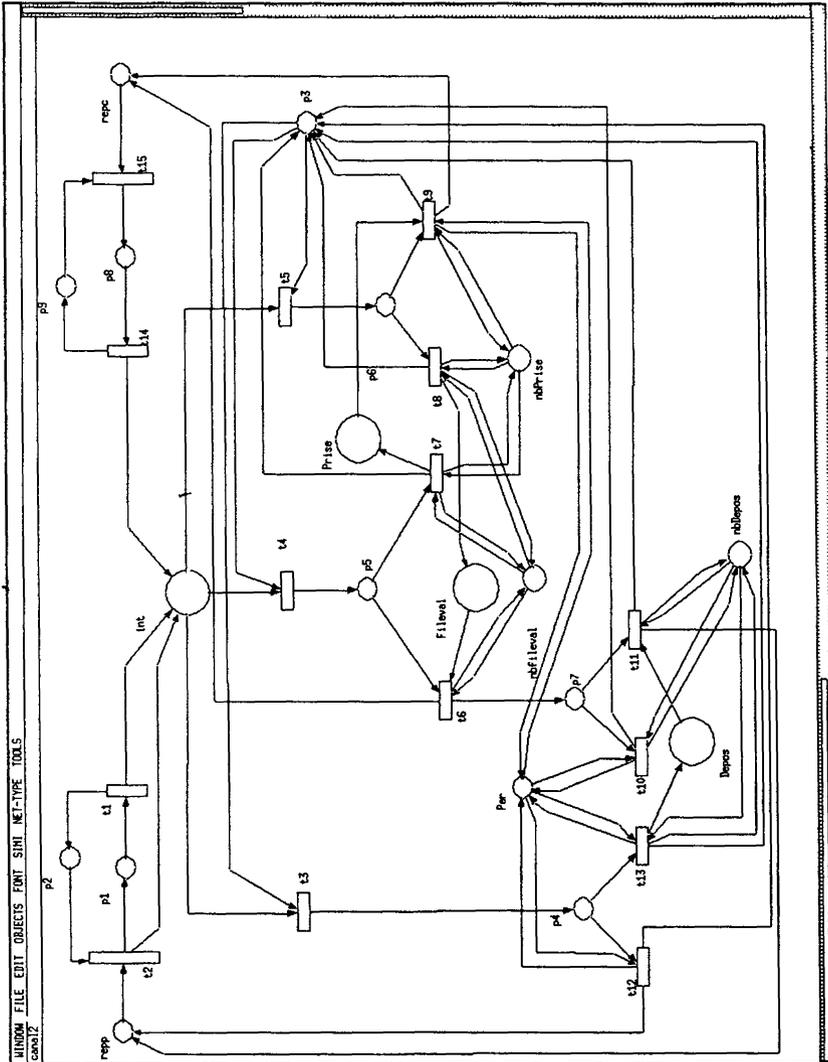


Figure 3.

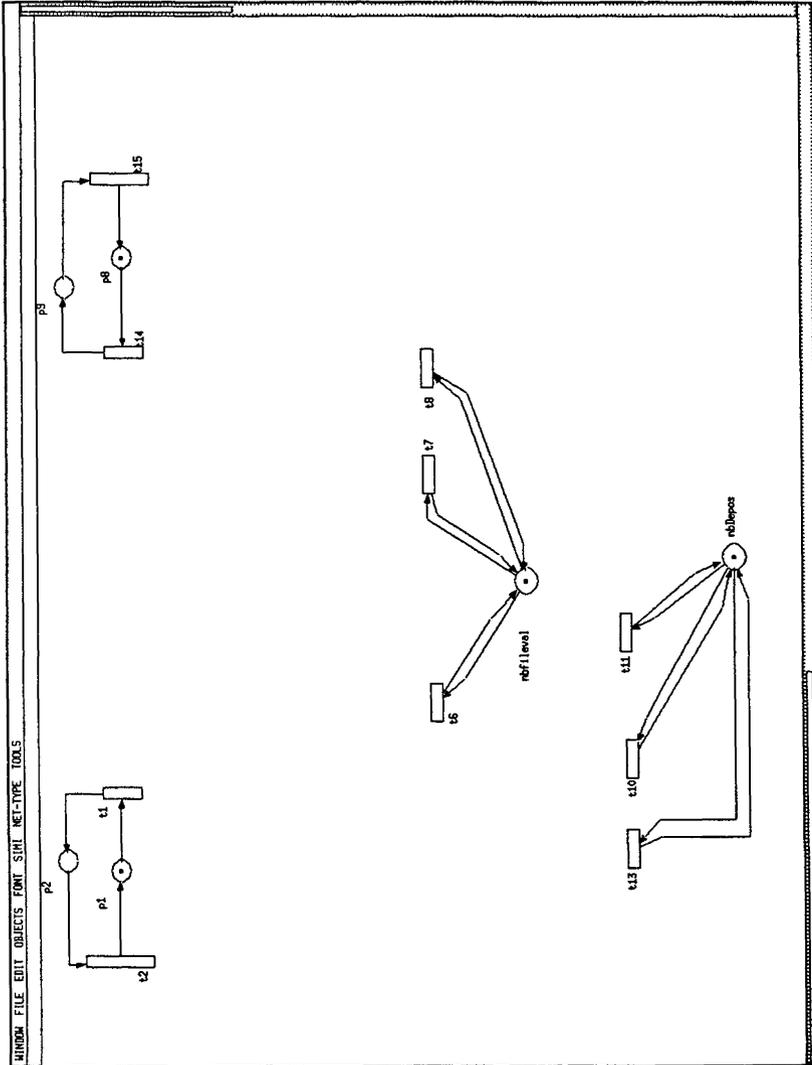


Figure 4.

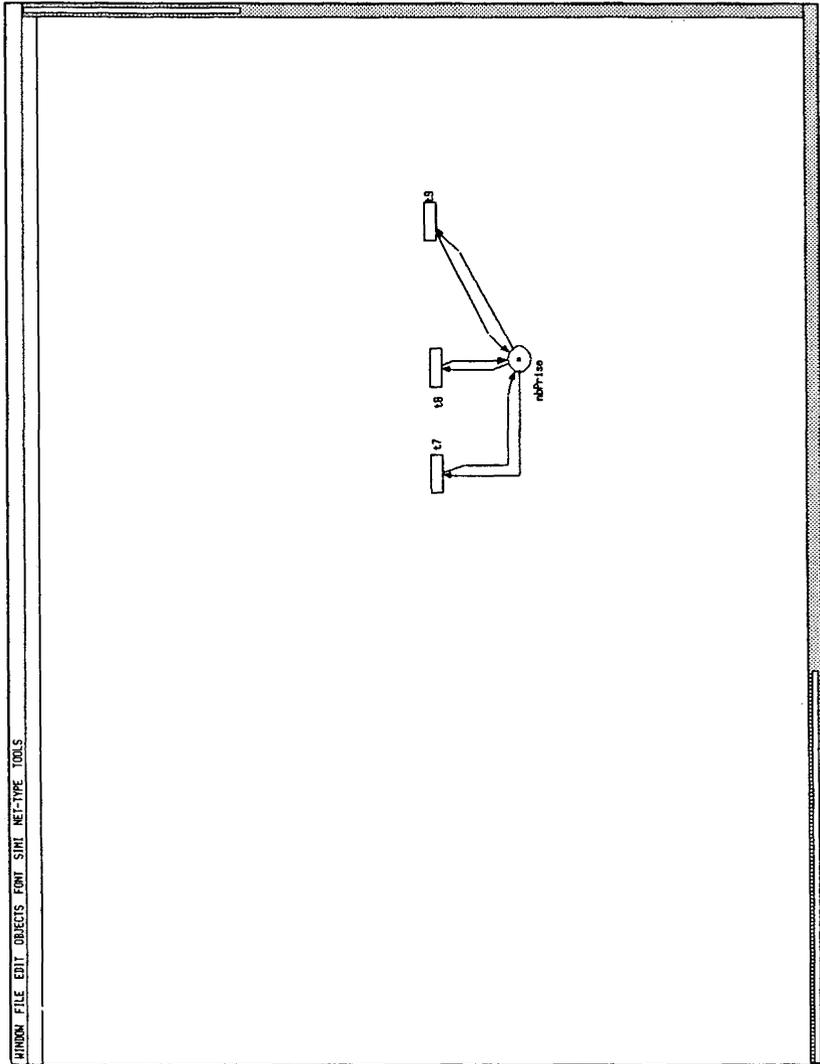


Figure 5.

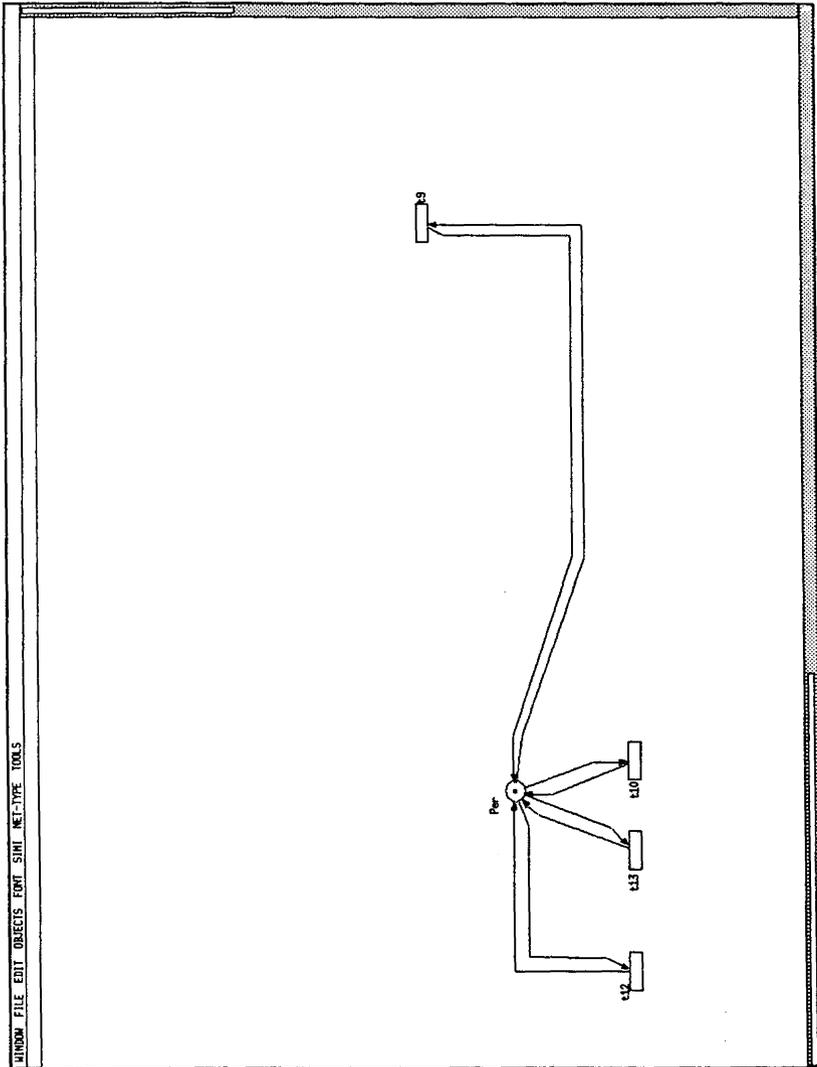


Figure 6.

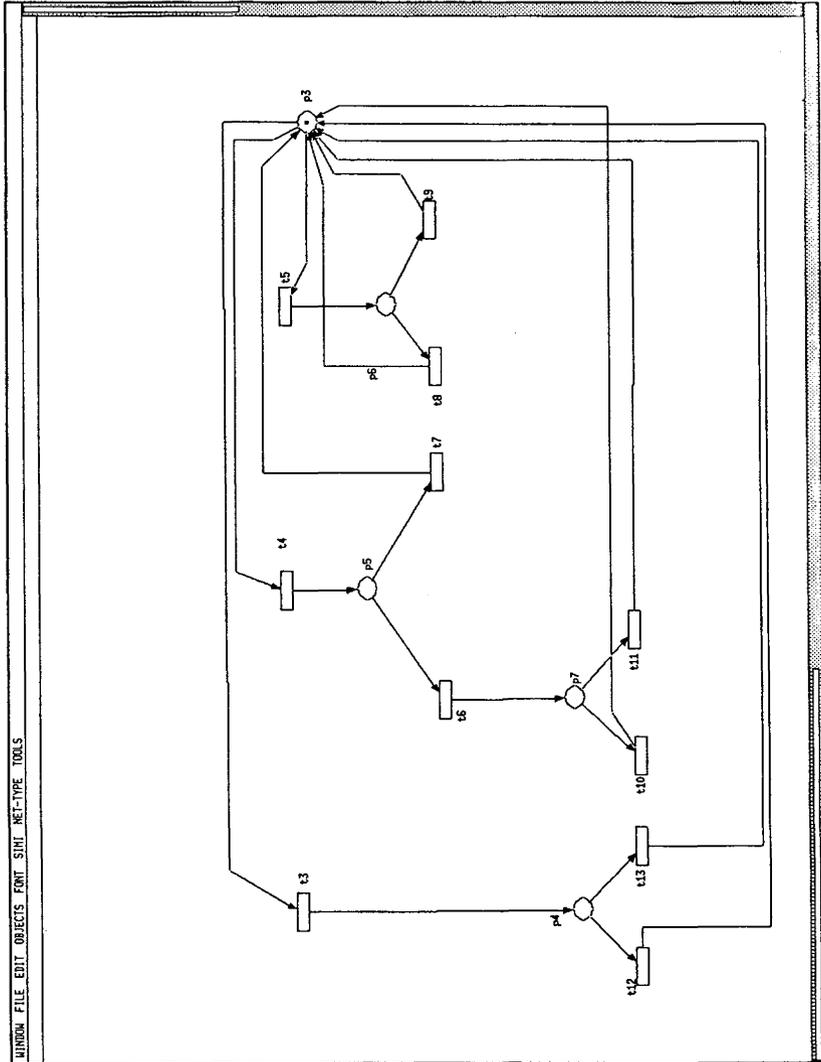


Figure 7.

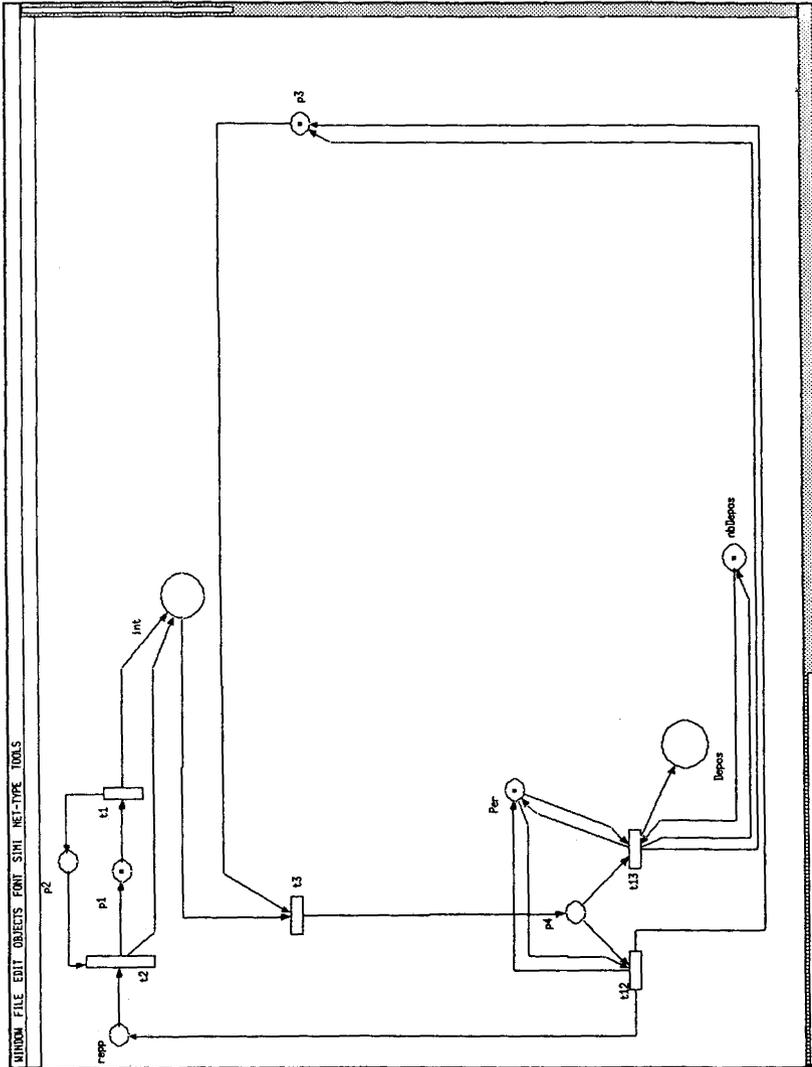


Figure 8.

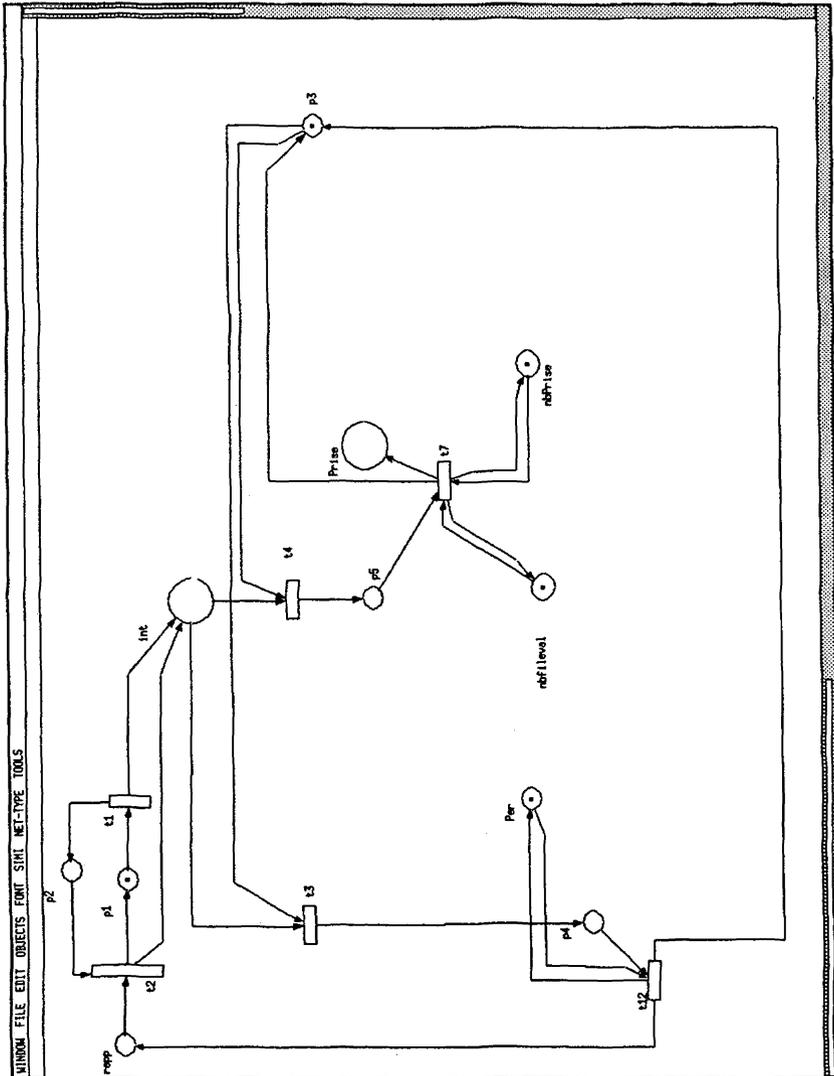


Figure 9.

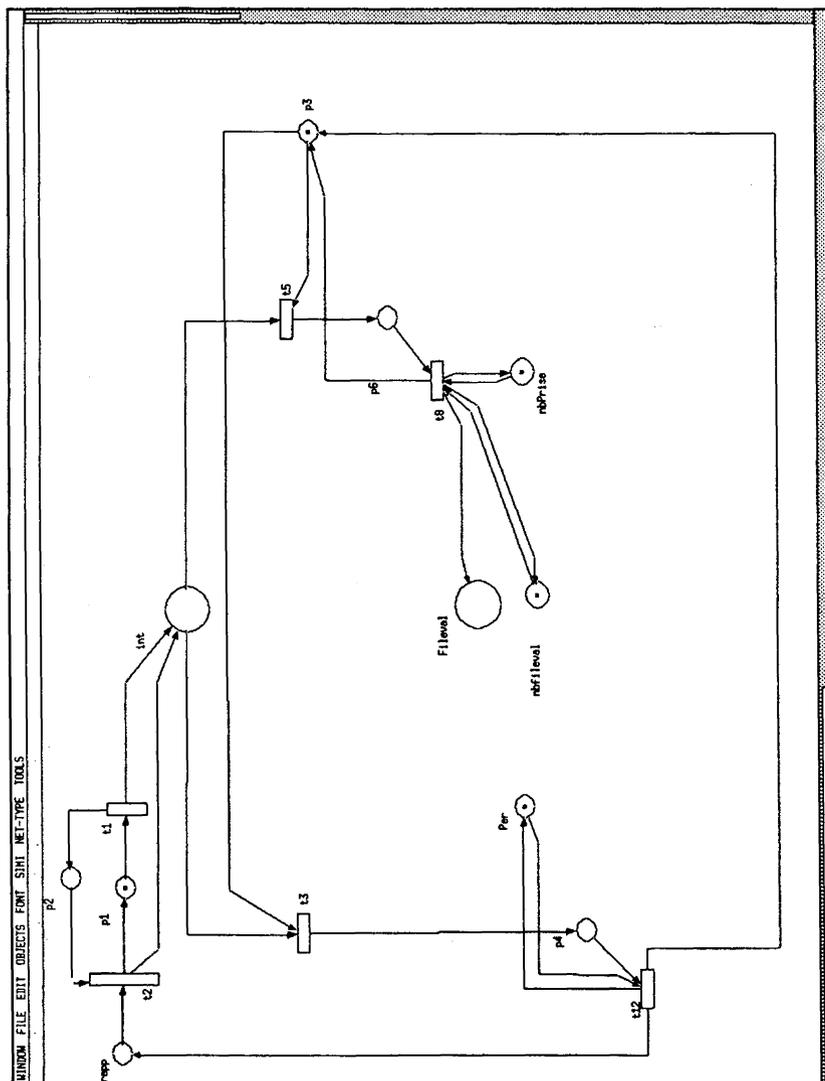


Figure 10.

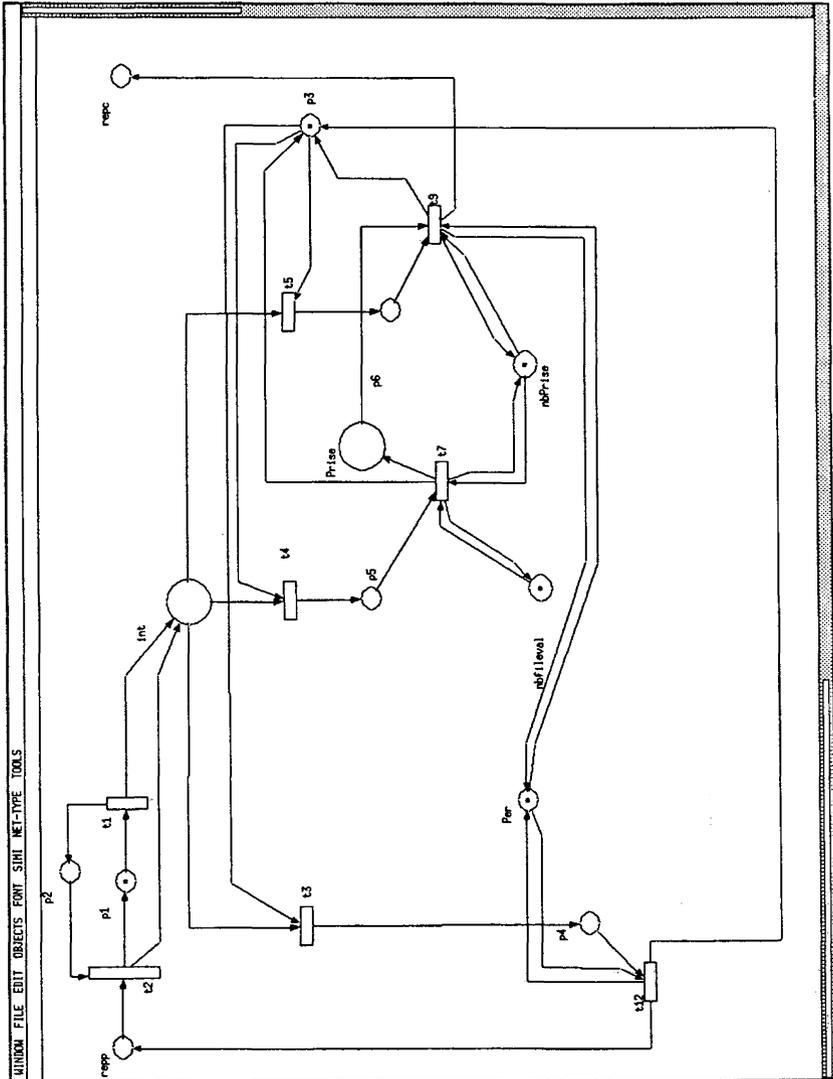


Figure 11.

## 3. LE GRAPHE DE COUVERTURE MINIMAL

Il est possible de réduire la taille du graphe d'accessibilité (et aussi de le rendre fini) en construisant le graphe de couverture minimal, selon la procédure suivante :

```

nœuds_non_traités := { créer_nœud( $r$ ,  $M_0$ ) };           /*  $M_0$  est le marquage de la racine  $r$  */
nœuds_traités :=  $\emptyset$ ;
tant que nœuds_non_traités  $\neq \emptyset$ ;
  sélectionner un nœud  $n \in$  nœuds_non_traités;
  nœuds_non_traités := nœuds_non_traités - {  $n$  };
  /*  $m$  est le marquage de  $n$  et  $m'$  celui de  $n'$  */
  cas  $n$  : [ 1 ., 4 ] dans
    1 : il existe un nœud  $n' \in$  nœuds_traités tel que  $m = m'$  :
      /* un nœud de même marquage existe déjà */
      nœuds_traités := nœuds_traités + {  $n$  };
      sortie cas
    2 : il existe un nœud  $n' \in$  nœuds_traités tel que  $m < m'$  :
      /* il existe un nœud de marquage plus grand */
      supprimer_nœud( $n$ );                               /* on supprime le nœud  $n$  */
      sortie cas
    3 : il existe un nœud  $n' \in$  nœuds_traités tel que  $m > m'$  :
      /* il existe un nœud  $n'$  de marquage plus petit */
       $m' := m$ ;
      pour tout  $n'$  ancêtre de  $n$  tel que  $m > m'$  faire
        pour toutes les places  $p$  telles que  $m(p) > m'(p)$  faire
           $m''(p) := \omega$ ;                               /* mise à  $\omega$  des marquages plus petits */
        fpour
      fpour
      nœuds_traités := nœuds_traités + {  $n$  };
      si  $n'$  est un ancêtre de  $n$ 
        alors  $n' :=$  premier nœud traité, sur le chemin de la racine à  $n$  tel que  $m > m'$ ;
           $m' := m''$ ;
          supprimer_arbre( $n'$ );
          /* le nœud n'ayant changé de marquage, il faut le traiter à nouveau */
          nœuds_non_traités := nœuds_non_traités + {  $n$  };
          nœuds_traités := nœuds_traités - {  $n'$  };
        fsi
      pour tout  $n' \in$  nœuds_traités tel que  $m'' > m'$  faire
        supprimer_arbre( $n'$ );
        supprimer_nœud( $n'$ );
        /* on supprime le nœud de marquage plus petit */
      fpour

```

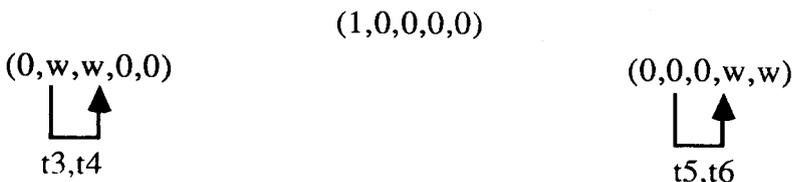
```

4 : sinon :                               /*on calcule les fils du nœud n de marquage m*/
    pour toute transition  $t \in T$  telle que  $m[t > m'$  faire
        créer_nœud( $n', m'$ );
        nœuds_non_traités := nœuds_non_traités + {  $n'$  };
        créer_arc( $n, n', t$ );                /*créer un arc de n à n' étiqueté par t*/
    fpour
        nœuds_traités := nœuds_traités + {  $n$  };
    fcas
ftq
identifier_nœuds_avec_même_étiquette;
pour tout arc( $n, n', t$ ), tel que  $n \neq n'$  faire
    si non  $m[t > m'$ 
        alors supprimer arc( $n, n', t$ );
    fsi
fpour

```

L'algorithme fonctionne de la manière suivante : tant qu'il y a des nœuds non traités, un nœud  $n$  est sélectionné pour être traité. Le traitement d'un nœud  $n$  commence par un test sur  $n$  : s'il existe un nœud  $n'$  de  $n$  tel que  $m' = m$  (cas 1) ou si  $m$  est un marquage mort, le nœud  $n$  est une feuille et l'on s'arrête. S'il existe un nœud  $n'$  tel que  $m' > m$  (cas 2), alors, on enlève le nœud  $n$  et l'arc y aboutissant. S'il existe un nœud  $n'$  tel que  $m' < m$  (cas 3), alors on enlève le sous-arbre de  $n'$ , y compris la racine. On construit le marquage  $m''$  tel que  $m'' = m(p)$  si  $m(p) = m'(p)$  et  $m''(p) = \omega$  si  $m(p) > m'(p)$ . Si, de plus,  $n'$  était un ancêtre de  $n$ , alors on garde  $n'$  que l'on étiquette avec  $m''$ . Sinon (cas 4), pour toutes les transitions  $t$  franchissables à partir de  $m$ , on construit le marquage  $m'$  obtenu à partir de  $m$  en franchissant  $t$ .

*Exemple 3.1* : Le graphe de couverture minimal de PNO (exemple 1.3) est le suivant :



On remarque sur cet exemple que le graphe de couverture minimal n'est pas forcément connexe.

Nous utiliserons, dans la suite, les pré-graphes de couverture minimaux, qui sont construits de la même manière que les graphes de couverture

minimaux, mais dans lesquels la dernière étape consistant à supprimer des arcs inutiles n'est pas appliquée.

Les fonctions Pré et Post sont naturellement étendues de  $P_x T^+$  vers  $\mathbb{N}$  de la manière suivante : pour tout  $x \in T^*$ ,  $t \in T$ , on pose

$$\text{Pré}(p, xt) = \text{Pré}(p, x) + \text{Pré}(p, t) \text{ et } \text{Post}(p, xt) = \text{Post}(p, x) + \text{Post}(p, t).$$

**THÉORÈME 3.2 [6] :** *Soit PN un réseau de Petri et GCM(PN) son graphe de couverture minimal.*

- (i) *GCM(PN) est calculable, fini et unique ;*
- (ii) *une place p est non bornée si et seulement s'il existe un marquage  $M \in \text{GCM}(PN)$  tel que  $M(p) = \omega$  ;*
- (iii) *une transition t est quasi-vivante si et seulement s'il existe un marquage  $M \in \text{GCM}(PN)$  tel que, pour toute place p,  $M(p) \geq \text{Pré}(p, t)$  ;*
- (iv) *l'arbre d'accessibilité de PN est infini si et seulement s'il existe un circuit dans GCM(PN) ;*
- (v) *l'ensemble des états accessibles de PN est infini si et seulement s'il existe au moins un symbole  $\omega$  dans GCM(PN).*

Le point à remarquer est que l'ensemble des circuits élémentaires de  $\text{GCM}(PN)$  coïncide avec l'ensemble des circuits élémentaires du graphe de couverture de Karp et Miller.

**COROLLAIRE 3.3 [6] :** *Les quatre problèmes suivants sont tous décidables à l'aide du graphe de couverture minimal :*

- (i) *la finitude de l'arbre d'accessibilité ;*
- (ii) *la finitude de l'ensemble des marquages accessibles ;*
- (iii) *le caractère borné des places ;*
- (iv) *la quasi-vivacité des transitions.*

#### 4. COMPOSITION DE GRAPHES DE COUVERTURE MINIMAUX

L'héritage direct de propriétés de sous-réseaux pour l'analyse d'un réseau composé ne permet pas toujours de décider des propriétés voulues. Par contre, nous savons que le graphe de couverture du réseau total permet de le faire. Nous nous intéresserons à l'étude des graphes de couverture minimaux [6], dans lequel les nœuds sont deux à deux incomparables. La composition des graphes de couverture minimaux des réseaux  $R_1, \dots, R_n$ ,

dont la composition est un réseau complet  $R$ , permet d'obtenir le graphe de couverture minimal du réseau  $R$  à moindre coût.

Dans cette section, nous définissons la composition de graphes de couverture minimaux. Les compositions que nous considérons étant des fusions sur un ensemble commun de places ou sur un ensemble commun de transitions, nous présentons deux algorithmes de composition des graphes de couverture minimaux.

#### 4.1. Fusion de transitions

Nous construisons maintenant le graphe de couverture minimal d'un réseau composé  $R_0$  à partir des réseaux  $R_1$  et  $R_2$  par fusion de transitions.

Rappelons qu'un marquage  $M$  de  $R_0$  est composé de deux parties : un marquage  $M_1$  correspondant aux places de  $P_1$  et un marquage  $M_2$  correspondant aux places de  $P_2$ .

La construction de  $GCM0$ , graphe de couverture minimal de  $R_0$ , à partir de  $GCM1$  et  $GCM2$ , pré-graphes de couverture minimaux des réseaux  $R_1$  et  $R_2$ , s'effectue à l'aide de l'algorithme suivant :

```

GCM1 := pré-graphe de couverture minimal de  $R_1$  ;
GCM2 := pré-graphe de couverture minimal de  $R_2$  ;
/*construction de GCM0 := graphe de couverture minimal de  $R_0$ */
nœuds_non_traités := { créer_nœud ( $r, M_0$ ) };
/* $M_0 = (M_1, M_2)$  est le marquage de la racine  $r$ */
nœuds_traités :=  $\emptyset$  ;
tant que nœuds_non_traités  $\neq \emptyset$  ;
  sélectionner un nœud  $n \in$  nœuds_non_traités ;
  nœuds_non_traités := nœuds_non_traités - {  $n$  } ;
  /* $m$  est le marquage de  $n$  et  $m'$  celui de  $n'$ */
  cas  $n : [1 . . 4]$  dans
    1 : il existe un nœud  $n' \in$  nœuds_traités tel que  $m = m'$  :
          /* un nœud de même marquage existe déjà*/
          nœuds_traités := nœuds_traités + {  $n$  } ;
          sortie cas
    2 : il existe un nœud  $n' \in$  nœuds_traités tel que  $m < m'$  :
          /*il existe un nœud de marquage plus grand*/
          supprimer_nœud( $n$ ) ;          /*on supprime le nœud  $n$ */
          sortie cas
    3 : il existe un nœud  $n' \in$  nœuds_traités tel que  $m > m'$  :
          /*il existe un nœud  $n'$  de marquage plus petit*/

```

```

m'' := m;
pour tout n' ancêtre de n, n' ∈ nœuds_traités tel que m > m' faire
  pour toutes les places p telles que m(p) > m'(p) faire
    m''(p) := ω;
  fpour
  fpour
  nœuds_traités := nœuds_traités + { n };
  si n' est un ancêtre de n
    alors n' := premier nœud traité, sur le chemin de la racine à n tel que m > m';
    m' := m'';
    supprimer_arbre(n');
    /*le nœud n' ayant changé de marquage, il faut le traiter à nouveau*/
    nœuds_non_traités := nœuds_non_traités + { n' };
    nœuds_traités := nœuds_traités - { n' };
  fsi
  pour tout n' ∈ nœuds_traités tel que m'' > m' faire
    supprimer_arbre(n');
    supprimer_nœud(n');
    /*on supprime le nœud de marquage plus petit*/
  fpour
4 : sinon :                               /*on calcule les fils du nœud n de marquage m = (m1, m2)*
nc1 := le nœud de GCM1 de marquage mc1, comparable à m1;
nc2 := le nœud de GCM2 de marquage mc2, comparable à m2;
pour toute transition t ∈ T1 telle qu'il existe un arc mc1 [t > mc1 dans GCM1 et m1
  [t > m'1 faire
    créer_nœud(n', (m'1, m2));
    nœuds_non_traités := nœuds_non_traités + { n' };
    créer_arc(n, n', t);                               /*créer un arc de n à n' étiqueté par t*/
  fpour
pour toute transition t ∈ T2 telle qu'il existe un arc mc2 [t > mc2 dans GCM2 et
  m2 [t > m'2 faire
    créer_nœud(n', (m1, m'2));
    nœuds_non_traités := nœuds_non_traités + { n' };
    créer_arc(n, n', t);                               /*créer un arc de n à n' étiqueté par t*/
  fpour
pour toute transition t ∈ Tc telle qu'il existe un arc mc1 [t > mc1 dans GCM1
  et un arc mc2 [t > mc2 dans GCM2 et m1 [t > m'1 et m2 [t > m'2 faire
    créer_nœud(n', (m'1, m'2));
    nœuds_non_traités := nœuds_non_traités + { n' };
    créer_arc(n, n', t);                               /*créer un arc de n à n' étiqueté par t*/
  fpour
nœuds_traités := nœuds_traités + { n };

```

fcasftq

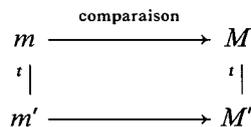
identifier\_nœuds\_avec\_même\_étiquette;

On part du marquage initial  $M_0 = (M_{0_1}, M_{0_2})$  de  $R_0$ . Soient  $\text{Acc}(N_1)$  l'ensemble d'accessibilité du réseau  $N_1$  et  $\text{Acc}_1(N)$  l'ensemble d'accessibilité de  $N$  avec des marquages restreints aux places de  $N_1$ . Comme nous utilisons la composition par fusion de transitions,  $\text{Acc}(N_1) \cong \text{Acc}_1(N)$ . Donc  $M_{0_1}$  est dans  $\text{Acc}(N_1)$ . Par définition du (pré)graphe de couverture minimal, il existe un et un seul marquage  $M_1$  comparable à  $M_{0_1}$  dans  $GCM1$ . De même, il existe un unique marquage  $M_2$  comparable à  $M_{0_2}$  dans  $GCM2$ . S'il existe un arc  $M_1[t > \dots]$  dans  $GCM1$ , avec  $t \in T_1$ , alors  $t$  peut être franchissable dans  $R_0$  et si c'est le cas, un arc similaire est construit dans  $GCM0$ . S'il existe un arc  $M_2[t > \dots]$  dans  $GCM2$ , avec  $t \in T_2$ , alors  $t$  peut être franchissable dans  $R_0$  et si c'est le cas, un arc similaire est construit dans  $GCM0$ . S'il existe un arc  $M_1[t > \dots]$  dans  $GCM1$  et un arc  $M_2[t > \dots]$  dans  $GCM2$ , avec  $t \in T_c$ , alors  $t$  peut être franchissable dans  $R_0$ , et si c'est le cas l'arc est construit.

Tous les arcs nécessaires se trouvent dans les pré-graphes de couverture des sous-réseaux. Les seuls arcs n'apparaissant pas sont ceux qui conduisent directement à un marquage plus petit. S'ils étaient utilisés, ils mèneraient à la création d'un marquage plus petit immédiatement détruit.

Ce raisonnement est valable non seulement pour le marquage initial, mais aussi pour tout marquage accessible de  $R_0$ . Pour la construction de  $GCM0$ , on sait déjà quelles transitions sont franchissables, et on connaît une partie du marquage – celle correspondant à l'ensemble des places non concernées par le franchissement de la transition, c'est-à-dire les places de  $P_2$  pour une transition de  $T_1$  et les places de  $P_1$  pour une transition de  $T_2$ .

L'algorithme que nous venons de présenter utilise donc une optimisation de la création des fils d'un nœud. Il permet d'éviter des tests de franchissabilité de transitions et des calculs de marquages, contrairement à un calcul direct. Trouver un nœud  $M'$  comparable à un nœud  $m'$  n'est pas une opération longue dans la mesure où les pères de  $M'$  et  $m'$  sont comparables et reliés à leur fils par un arc étiqueté par la même transition :



#### 4.2. Un exemple : le protocole de communication par bits alternés

Nous allons maintenant étudier un réseau algébrique (*fig. 12*) modélisant le protocole de communication par bits alternés, à l'aide de son squelette (*fig. 13*). Un émetteur  $E$  désire transmettre des messages à un récepteur  $R$ . Il possède une file de messages à envoyer (place  $P5$ ). Il envoie le premier message après l'avoir étiqueté avec un bit valant 0 (transition  $T1$ ). Quand  $R$  reçoit le message ( $T3$ ), il envoie un accusé de réception avec un bit identique ( $T4$ ). Si au bout d'un certain temps,  $E$  n'a pas reçu d'accusé de réception, il renvoie son message ( $T5$ ) (le message précédent a pu se perdre par  $T7$ ),... jusqu'à ce qu'il ait reçu l'accusé de réception ( $T2$ ). A ce moment là, il sait qu'il peut envoyer le message suivant avec un bit valant 1 (alternance par rapport au bit précédent). De même, si  $R$  ne reçoit pas de nouveau message, il peut supposer que son accusé de réception s'est perdu (par  $T8$ ) et qu'il lui faut le renvoyer ( $T9$ ). On continue jusqu'à ce qu'il n'y ait plus de message à envoyer. Les transitions  $T6$  et  $T10$  servent à enlever de la file les messages envoyés en plusieurs exemplaires et déjà reçus par leur destinataire.

On décompose le squelette en quatre parties dont le squelette est le composé par fusion de transitions : le processus émetteur (places :  $P1, P2, P5, P9$ , transitions :  $T1, T2, T5, T6$ ), le processus récepteur (places :  $P3, P4, P8$ , transitions :  $T3, T4, T9, T10$ ), le canal pour l'envoi des messages (place  $P6$ , transitions :  $T1, T3, T5, T7, T10$ ) et celui pour l'envoi des accusés de réception (place  $P7$ , transitions :  $T2, T4, T6, T8, T9$ ).

Ce découpage permet d'établir que toutes les places sont bornées aussi bien dans le squelette que dans le réseau algébrique. La construction du graphe de couverture composé (*fig. 14*) montre que dans le squelette, toutes les transitions sont quasi-vivantes et qu'il n'y a pas de marquage mort. On n'obtient pas d'autre propriété sur le réseau algébrique que le caractère borné des places.

Composition/décomposition de réseaux de Petri et de leurs graphes de couverture

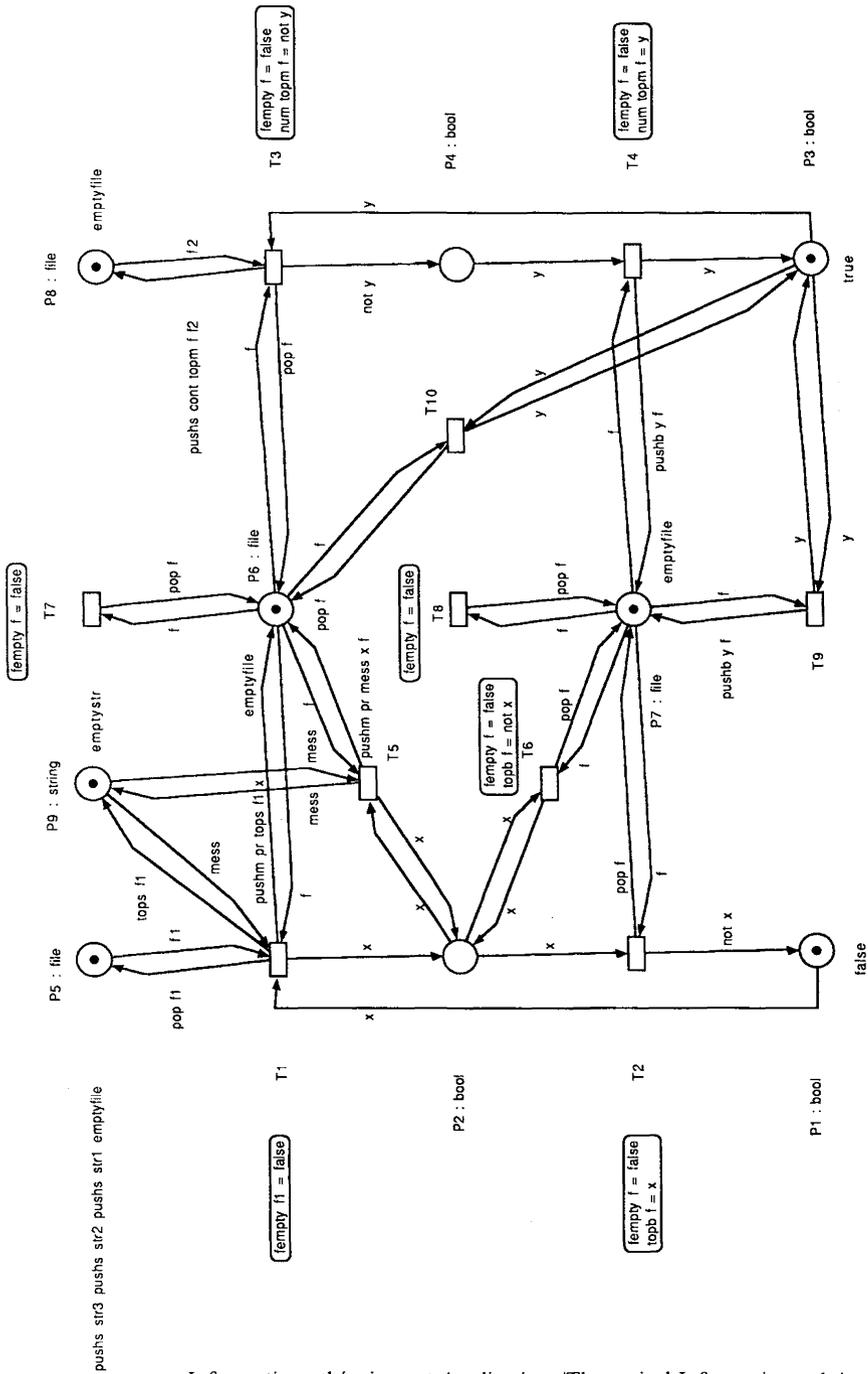


Figure 12.

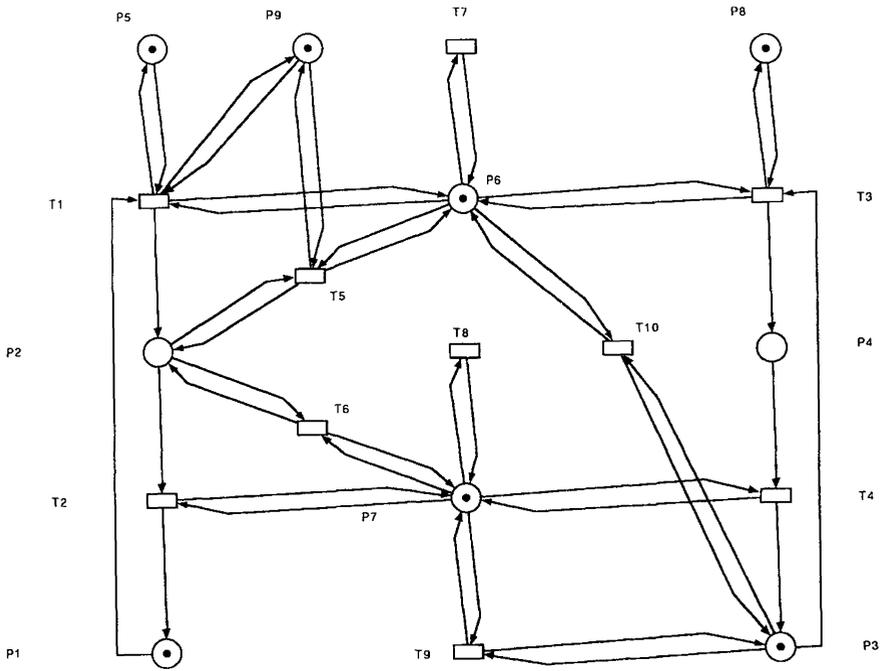
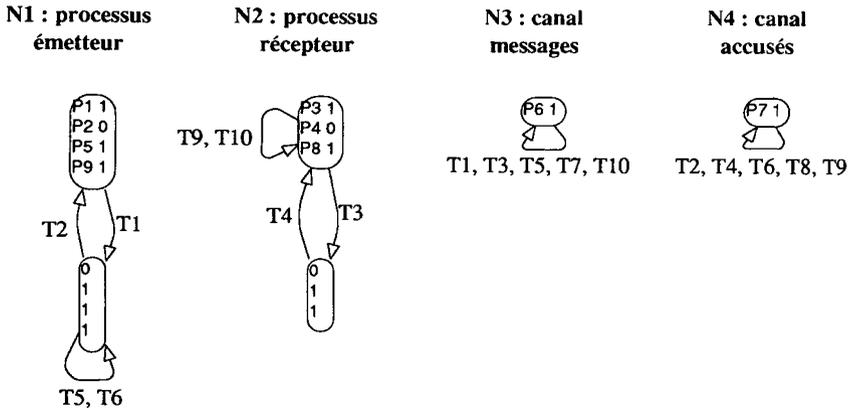


Figure 13.



Graphe de couverture du bit alterné

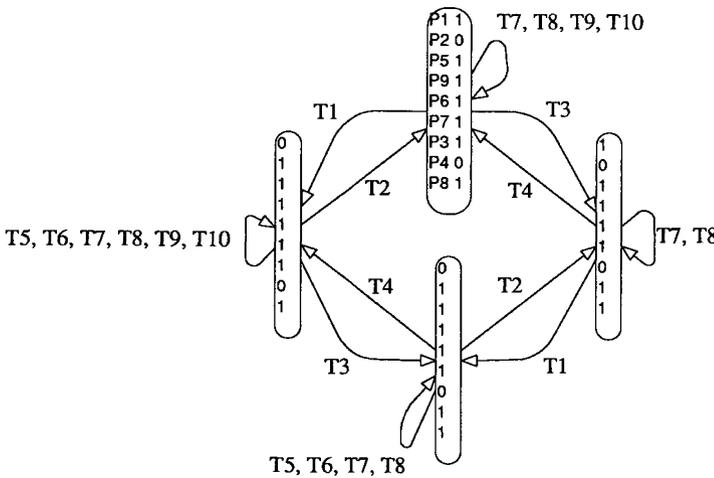


Figure 14.

### 4.3. Fusion de places

Comme pour la composition par fusion de transitions, nous étudions les possibilités de composition des graphes de couverture minimaux lorsqu'il s'agit de réseaux à fusionner par places.

Rappelons qu'un marquage  $M$  de  $R_0$  est composé de trois parties : un marquage  $M_1$  correspondant aux places de  $P_1$ , un marquage  $M_c$  correspondant aux places de  $P_c$  et un marquage  $M_2$  correspondant aux places de  $P_2$ .

Nous proposons l'algorithme suivant, pour calculer le graphe de couverture minimal du réseau complet à partir de ceux des sous-réseaux :

```

fini := faux ;
∀ i, kci := M0(pci) ; /*nombre de jetons dans la place commune pci*/
tant que non fini faire
  GCM1 = graphe de couverture de R1, avec M01(pci) = kci ;
  GCM2 = graphe de couverture de R2, avec M02(pci) = kci ;
  ∀ i, kci1 := maxGCM1 M(pci) ; /*le plus grand nombre de jetons présents dans pci dans GCM1*/
  ∀ i, kci2 := maxGCM2 M(pci) ; /*le plus grand nombre de jetons présents dans pci dans GCM2*/
  si ∀ i, kci1 = kci2
    alors fini := vrai ; /*les bornes des places pci sont les mêmes dans les deux réseaux*/
  fsi
  ∀ i, kci := max(kci1, kci2) ;
  si il existe une séquence s et i, j, i ≠ j, tels que M[s > M', M(pci) > M'(pci) et M'(pci) < M(pci)
    dans GCM1 ou GCM2
      alors fini := vrai ; /*si Pc est un singleton, ceci n'est pas appliqué*/
    fsi
  ftq
  /*construction de GCM0 := graphe de couverture minimal de R0*/
  nœuds_non_traités := { créer_nœud(r, M0) } ;
  /*M0 = (M1, M2) est le marquage de la racine r*/
  nœuds_traités := ∅ ;
  tant que nœuds_non_traités ≠ ∅ ;
    sélectionner un nœud n ∈ nœuds_non_traités ;
    nœuds_non_traités := nœuds_non_traités - { n } ;
    /*m est le marquage de n et m' celui de n'*/
    cas n : [1 . . 4] dans
      1 : il existe un nœud n' ∈ nœuds_traités tel que m = m' :
          /*un nœud de même marquage existe déjà*/
          nœuds_traités := nœuds_traités + { n } ;
          sortie cas
      2 : il existe un nœud n' ∈ nœuds_traités tel que m < m' :
          /*il existe un nœud de marquage plus grand*/
          supprimer_nœud(n) ; /*on supprime le nœud n*/
          sortie cas
      3 : il existe un nœud n' ∈ nœuds_traités tel que m > m' :
          /*il existe un nœud n' de marquage plus petit*/
          m'' := m ;
          pour tout n' ancêtre de n, n' ∈ nœuds_traités tel que m > m' faire
            pour toutes les places p telles que m(p) > m'(p) faire

```

```

     $m''(p) := \omega;$ 
  fpour
fpour
  nœuds_traités := nœuds_traités + {  $n$  };
  si  $n'$  est un ancêtre de  $n$ 
    alors  $n'$  := premier nœud traité, sur le chemin de la racine à  $n$  tel que  $m > m'$ ;
     $m' := m'';$ 
    supprimer_arbre( $n'$ );
    /*le nœud  $n'$  ayant changé de marquage, il faut le traiter à nouveau*/
    nœuds_non_traités := nœuds_non_traités + {  $n'$  };
    nœuds_traités := nœuds_traités - {  $n'$  };
  fsi
  pour tout  $n' \in$  nœuds_traités tel que  $m'' > m'$  faire
    supprimer_arbre( $n'$ );
    supprimer_nœud ( $n'$ );          /*on supprime le nœud de marquage plus petit*/
  fpour
4 : si non :          /*on calcule les fils du nœud  $n$  de marquage  $m = (m_1, m_c, m_2)$ */
   $nc_1$  := le nœud de l'un des GCM1 calculés, de marquage comparable à  $(m_1, m_2)$ ;
  /*renvoie 0 si non trouvé*/
   $nc_2$  := le nœud de l'un des GCM2 calculés, de marquage comparable à  $(m_2, m_c)$ ;
  /*renvoie 0 si non trouvé*/
  si  $nc_1 = 0$ 
    alors pour toute  $t \in T_1$  telle que  $(m_1, m_c) [t > (m'_1, m'_c)]$  faire
      créer_nœud( $n'$ ,  $(m'_1, m'_c, m_2)$ );
      nœuds_non_traités := nœuds_non_traités + {  $n'$  };
      créer_arc( $n, n', t$ );          /*créer un arc de  $n$  à  $n'$  étiqueté par  $t$ */
    fpour
    si non pour toute  $t \in T_1$  telle qu'il existe un arc  $(mc_1, mc_c)[t > (m'_1, m'_c)]$  dans l'un des
    GCM1 calculés
      et  $(m_1, m_c) [t > (m'_1, m'_c)]$  faire
        créer_nœud ( $n'$ ,  $(m'_1, m'_c, m_2)$ );
        nœuds_non_traités := nœuds_non_traités + {  $n'$  };
        créer_arc( $n, n', t$ );          /*créer un arc de  $n$  à  $n'$  étiqueté par  $t$ */
      fpour
    fsi
  si  $nc_2 = 0$  /*les deux « si » sont similaires, mais ils sont dupliqués pour plus de clarté*/
    alors pour toute  $t \in T_2$  telle que  $(m_2, m_c) [t > (m'_2, m'_c)]$  faire
      créer_nœud( $n'$ ,  $(m_1, m'_c, m'_2)$ );
      nœuds_non_traités := nœuds_non_traités + {  $n'$  };
      créer_arc ( $n, n', t$ );          /*créer un arc de  $n$  à  $n'$  étiqueté par  $t$ */
    fpour

```

```

sinon pour toute  $t \in T_2$  telle qu'il existe un arc  $(mc_2, mc_c) [t > (m'_2, m'_c)]$  dans l'un des
    GCM2 calculés et  $(m_2, m_c) [t > (m'_2, m'_c)]$  faire
        créer_nœud( $n', (m_1, m'_c, m'_2)$ );
        nœuds_non_traités := nœuds_non_traités + {  $n'$  };
        créer_arc( $n, n', t$ );                               /*créer un arc de  $n$  à  $n'$  étiqueté par  $t^*$ */
    fpour
fsi
nœuds_traités := nœuds_traités + {  $n$  };
fcas
fiq
identifier_nœuds_avec_même_étiquette;

```

L'union des graphes de couverture est définie comme suit. On part de l'état initial  $M_0$  du réseau  $R_0$ . Il se décompose en trois parties:  $M_0 = (M_1, M_c, M_2)$ , où  $M_1$  (resp.  $M_2$ ) est le marquage des places de  $P_1$  (resp.  $P_2$ ) et  $M_c$  le marquage des places de  $P_c$ , communes aux deux réseaux. Supposons que dans l'un des  $GCM1$  (resp.  $GCM2$ ) calculés, il existe un marquage  $(M'_1, M'_c)$  (resp.  $(M'_2, M'_c)$ ) comparable à  $(M_1, M_c)$  (resp. à  $(M_2, M_c)$ ). Soit  $t$  une transition de  $T_1$  telle que  $(M'_1, M'_c)[t >$  dans  $GCM1$  (pour  $GCM2$ , l'opération est similaire). Si  $t$  est franchissable dans le réseau total, alors, on obtiendra, dans  $GCM0$  un arc  $(M_1, M_c, M_2)[t > (M'_1, M'_c, M_2)$ . On réitère l'opération pour les marquages ainsi obtenus.

Si l'on a trouvé un marquage  $M'$  comparable au marquage  $M$  auquel on s'intéresse, dans l'un des graphes  $GCM1$ , alors les seules transitions de  $T_1$  franchissables à partir de  $M$  font partie de celles franchissables à partir de  $M'$ . Sinon, il faut toutes les examiner. De même pour les  $GCM2$ .

*Preuve de terminaison de l'algorithme.*

Nous allons montrer que l'algorithme présenté termine. Pour cela, il suffit de montrer que sa première partie termine.

Soit  $s$  le terme générique utilisé pour désigner une séquence de franchissements telle que  $M[s > M']$ . Deux cas peuvent se présenter :

- $(\forall s \in T_1^*) ((\forall p \in P_c M'(p) \leq M(p)) \text{ ou } (\forall p \in P_c M(p)' \geq M(p)))$ .

Considérons le premier sous-cas :  $\forall p \in P_c M'(p) \leq M(p)$ . La borne des places est conservée d'où l'arrêt de l'algorithme.

Considérons maintenant le second sous-cas :  $\forall p \in P_c M'(p) \geq M(p)$ . Par définition du graphe de couverture minimal, les nœuds sont deux à deux incomparables. Donc, il existe une place  $p'$  n'appartenant pas à  $P_c$  telle que

$M(p') > M'(p')$ , c'est-à-dire que le marquage de  $p'$  décroît par franchissement de  $s$ . D'où l'arrêt de l'algorithme.

•  $\exists s \in T_i^* tq ((\exists p \in P_c tq M'(p) > M(p)) \text{ et } (\exists p' \in P_c tq M'(p') < M(p')))$ : dans ce cas, on force l'arrêt de l'algorithme en mettant la variable fini à vrai ■.

**4.4. Suite de l'exemple du protocole de communication par bits alternés**

On reprend l'exemple du protocole de communication par bits alternés, dont on décompose le squelette en deux parties. Le squelette est alors

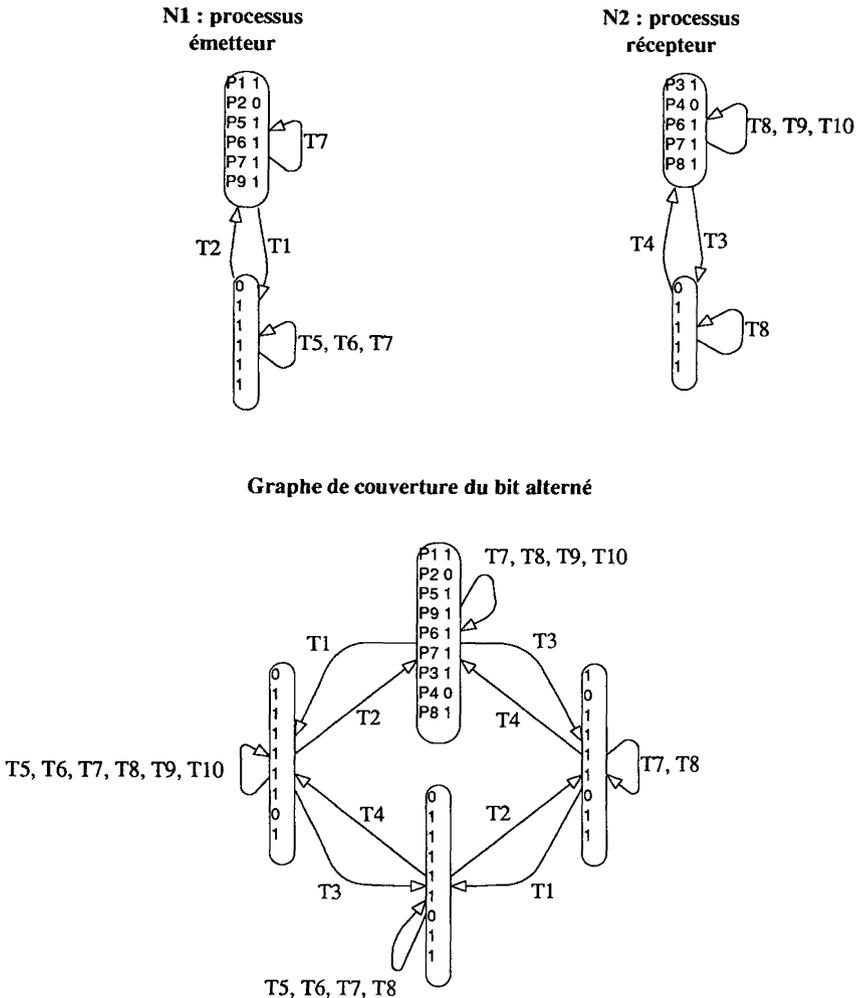


Figure 15.

le composé par fusion de places du processus émetteur (places: P1, P2, P5, P6, P7, P9, transitions: T1, T2, T5, T6, T7) et du processus récepteur (places: P3, P4, P6, P7, P8, transitions: T3, T4, T8, T8, T10). Chacun des deux processus prend en compte les canaux d'envoi de messages et d'accusés de réception. On peut déduire directement de cette décomposition que toutes les transitions sont quasi-vivantes. La figure 15 montre la composition des graphes de couverture.

## 5. CONCLUSION

Nous avons montré, dans cet article, comment exploiter la décomposition en modules d'un réseau. La composition de plusieurs sous-réseaux soit par fusion de places, soit par fusion de transitions, préserve certaines propriétés qui ne sont donc pas à prouver dans le réseau complet une fois qu'elles sont validées sur ses sous-réseaux, voire sur seulement l'un d'eux.

Nous avons exposé des algorithmes permettant la construction de sous-réseaux d'un réseau donné pour la vérification d'une propriété, à savoir le caractère borné ou non borné des places et la quasi-vivacité d'une transition. Ces algorithmes cherchent un sous-réseau convenable qui vérifie la propriété. On pourrait les optimiser en cherchant le plus petit sous-réseau.

Dans la mesure où l'on ne peut pas toujours décider de propriétés par simple analyse de sous-réseaux, il peut s'avérer utile de construire le graphe de couverture minimal du réseau complet. Pour cela, nous avons présenté des algorithmes – l'un pour la fusion de transitions et l'autre pour la fusion de places – permettant d'utiliser les graphes de couverture minimaux des sous-réseaux pour construire à moindre coût le graphe de couverture minimal du réseau complet et ensuite l'analyser.

Les méthodes proposées peuvent être utilisées en calculant les graphes de couverture de chaque module en parallèle. Une implémentation de ces algorithmes est en cours.

Revenons à l'algorithme de composition de graphes de couverture minimaux dans le cas de la fusion sur un ensemble de places communes. On peut constater que l'on force l'algorithme à s'arrêter dans le cas où le graphe de couverture de l'un des sous-réseaux contient une séquence de franchissements  $s$  telle que le marquage d'une place commune augmente par franchissement de  $s$ , et celui d'une autre place commune décroît. Cette condition d'arrêt n'est pas totalement satisfaisante car certains marquages ne seront pas calculés localement. Il faudra donc les calculer, de même que leurs fils,

lors de la construction du graphe de couverture du réseau complet ce qui prendra plus de temps et d'espace. Nous conjecturons que l'algorithme de calcul du graphe de couverture minimal du réseau complet à partir des deux graphes de couverture minimaux de ses sous-réseaux peut être optimisé en supprimant la condition d'arrêt et en itérant  $n$  fois la boucle principale où  $n$  est le nombre de places communes.

La deuxième technique importante pour l'analyse des réseaux de Petri est le calcul d'invariant. L'étude des relations entre la composition/décomposition de réseaux et les invariants serait une suite logique à ce travail.

**Remerciements.** — Les auteurs expriment leurs remerciements aux referees qui ont fourni des commentaires précis et suggéré des améliorations.

#### BIBLIOGRAPHIE

1. A. BOURGUET, *Étude de la concordance de comportement de deux réseaux de Petri. Application à la validation des protocoles : détection automatique des erreurs de conception.* Thèse de l'Université Pierre-et-Marie-Curie, septembre 1990.
2. G. BERTHELOT, L. PETRUCCI, *Putting Algebraic Nets Into Practice*, Rapport interne CEDRIC-III, janvier 1989.
3. GW. BRAMS, *Réseaux de Petri : théorie et pratique*, Masson, 1983.
4. W. BRAUER, W. REISIG, G. ROSENBERG, *Petri Nets: Central Models and Their Properties*, Advances in Petri Nets 1986, Part 1, Proceedings of an Advanced Course at Bad Honnef, in LNCS No. 254, Springer Verlag, 1986.
5. C. DIMITROVICI, U. HUMMERT, L. PETRUCCI, *The Properties of Algebraic Nets Schemes in Some Semantics*, Proceedings of the 11th International Conference on Application and Theory of Petri Nets, Paris, juin 1990.
6. A. FINKEL, *The Minimal Coverability Graph for Petri Nets*. Advances in Petri Nets 1993, LNCS 674, pp. 210-243, Springer Verlag, 1993.
7. M. HACK, *Decidability Questions for Petri Nets*, Ph. D. Thesis, Technical Report 161, MIT, Laboratory for Computer Science, juin 1976.
8. G. MEMMI, J. VAUTHERIN, *Analysing Nets by the Invariant Method*, Advances in Petri Nets 1986, LNCS 255, pp. 300-337, Springer Verlag, 1987.
9. W. REISIG, *Petri Nets*, Springer Verlag, 1985.
10. Y. SOUSSI, *Une étude de la préservation de propriétés par composition de réseaux de Petri*, Thèse de l'Université Pierre-et-Marie-Curie, février 1990.
11. A. VALMARI, *Compositional State Space Generation*, Proceedings of the 11th International Conference on Application and Theory of Petri Nets, Paris, juin 1990.
12. J. VAUTHERIN, *Un modèle algébrique, basé sur les réseaux de Petri, pour l'étude des systèmes parallèles*, Thèse de doctorat d'ingénieur, Université de Paris-Sud, Centre d'Orsay, juin 1985.