ISSN 0280–5316
ISRN: LUTFD2/TFRT--5478--SE

Réseaux de Petri Continus Asymptotiques

Charlotta Johnsson

Department of Automatic Control
Lund Institute of Technology
July 1993

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
Master Thesis

Date of issue
July 1993

Document Number
ISRN LUTFD2/TFRT--5478--SE

Author(s)
Charlotta Johnsson

Supervisor
Hassane Alla, LAG–ENSIEG and Karl-Erik Årzén

Sponsoring organisation

Title and subtitle
Réseaux de Petri Continus Asymptotiques (Asymptotic Continuous Petri Nets)

Abstract

This master thesis is based on the Ph D thesis by Jean Le Bail at LAG (Laboratoire d'Automatique de Grenoble) 1992 concerning Asymptotic Continuous Petri Nets (ACPN).

The major properties of ACPN are presented. Four special cases are discussed: accumulation, synchronization, divergence, and conflicts. A new and improved method of solving the conflict case has been added to an already existing algorithm for simulation of ACPN. The thesis has also evaluated some ideas for improving the behaviour in the case of saturated and non-saturated loops. Here, the new results proved worse than the results of the original algorithm.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
French

Number of pages
79

Recipient's notes

Security classification

Table Des Matières

Principales Notations

RdP	Réseau de Petri
RdPCC	Réseau de Petri Continu à Vitesses Constantes
RdPCV	Réseau de Petri Continu à Vitesses Variables
RdPCA	Réseau de Petri Continu Asymptotique
P_i	Place numéro i d'un RdP
T_j	Transition numéro j d'un RdP
M ou M_k	Marquage d'un RdP
M_0	Marquage initial d'un RdP
$m_i(t)$	Marquage de la place P_i à l'instant t
m_i^*	Marquage asymptotique de la place P_i
M^r	Marquage réservé
M^n	Marquage non réservé
$b_i(t)$	Bilan de marquage de la place P_i à l'instant t
d_k	Temporisation (durée) associée à la transition T_k
$v_j(t)$ ou v_j	Vitesse instantanée de franchissement associée à la transition T_j
U_j	Fréquence de franchissement associée à la transition T_j
$< P_1, \{T_1, T_2, \ldots\} >$	Conflit structurel
S ou S_k	Séquence de franchissement
$\underline{S}$	Vecteur caractéristique de la séquence de franchissement
$Pré$	Matrice d'incidence avant
$Post$	Matrice d'incidence arrière
W	Matrice d'incidence ($W = Post - Pré$)
${}^\circ T_j$	Ensemble des places d'entrée de T_j
T_j°	Ensemble des places de sortie de T_j
${}^\circ P_i$	Ensemble des transitions d'entrée de P_i
P_i°	Ensemble des transitions de sortie de P_i

Introduction

Les Réseaux de Petri (noté dans la suite du rapport RdP) sont un outil mathématique permettant de décrire des relations entre des conditions et des événements. Les RdP ont été définis par le mathématicien allemand *Carl Adam Petri* dans les années 1960-62, [Petri 62].

Ce modèle, initialement utilisé pour décrire le fonctionnement de systèmes informatiques, a donné lieu à de nombreuses recherches dans le monde entier. Au début le modèle était discret. Grâce à divers chercheurs il a été enrichi et permet de décrire des systèmes plus complexes à l'aide de nouveaux modèles de RdP : les réseaux de Petri temporisés, interprétés, synchronisés, stochastiques, colorés ....

Dans l'étude d'applications industrielles, la simulation des modèles peut prendre des durées qui peuvent devenir prohibitives. Pour pallier cet inconvénient une nouvelle classe de RdP a été défini : les RdP Continus [David 87]. L'idée de départ est que l'utilisation d'un modèle continu pour un système discret peut apporter une bonne approximation de ce système [Dubois 82]. De plus, les RdP Continus (RdPC) peuvent également modéliser des systèmes continus. Dans un premier temps, deux modèles continus temporisés ont été élaborés dans l'équipe Sylodi[1] du Laboratoire d'Automatique de Grenoble, les RdPC à Vitesses Constantes *(RdPCC)* et les RdPC à Vitesses Variables *(RdPCV)*.

Le modèle RdPCC a été défini le premier. Il autorise une simulation rapide mais fournit des résultats qui, selon le cas, peuvent être éloignés de ceux qu'un modèle discret aurait donnés. Par contre, le modèle RdPCV possède des caractéristiques opposées, une simulation demandant davantage de temps de calcul mais donnant des résultats plus proches de ceux fournis par une simulation discrète classique.

Un nouveau modèle continu a récemment été défini [LeBail 92], *les Réseaux de Petri Continu Asymptotique (RdPCA)*. Ce modèle réunit les avantages des modèles RdPCC et RdPCV.

Dans ce rapport je regarde ses propriétés, ses avantages et ses inconvénients ; puis j'essayerai de les améliorer.

Les trois premiers chapitres de ce rapport sont une introduction ou un rappel sur les RdP Discrets et les RdP Continus. Dans ces chapitres le fonctionnement des réseaux, leurs propriétés et aussi des exemples sont étudiés. Dans la suite du rapport c'est expliqué ce que j'ai fait pendant mon stage; les changements essayés, les résultats trouvés et les amélioration du programme initial. A la fin du rapport il y a une résumé du rapport écrit en anglais.

[1] Sylodi = SYstèmes LOgiques et DIscrets

Chapitre 1

Réseaux de Petri Discrets

Un réseau de Petri est composé de places et de transitions reliées par des arcs orientés. Une *place* est représentée par un cercle et une *transition* par un trait. Un *arc* est orienté et relie soit une place à une transition soit une transition à une place. Chaque place contient un nombre entier (réel pour le RdPC) positif ou nul de *marques* ou *jetons*.

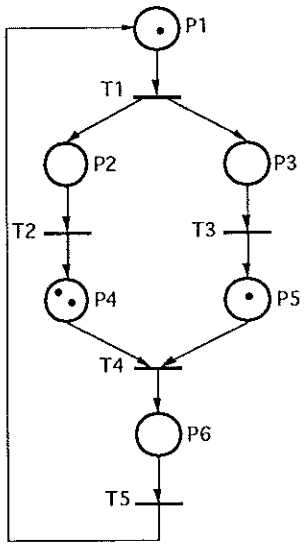


Figure 1.1: Un Réseau de Petri

Par exemple, le RdP de la figure 1.1 contient six places et cinq transitions. Le marquage initial vaut $M_0 = (1, 0, 0, 2, 1, 0)$.

Si chacune des places en amont d'une transition T_j contient au moins une marque, cette transition T_j est dite *franchissable* ou *validée* ; elle peut donc être franchie. Le franchissement d'une transition T_j consiste à retirer une marque dans chacune des places en amont (entrée) de T_j et à ajouter une marque dans chacune des places en aval (sortie).

L'évolution d'un système dynamique décrit par un RdP est représentée par l'évolution des marquages. Le nombre de marques dans un RdP peut augmenter et diminuer mais il reste toujours positif ou nul. Un RdP est dit *borné* si le nombre total de marques est fini. Les notions de RdP *vivant* (aucune transition ne peut devenir infranchissable) et aussi de RdP *sans blocage* (il y a au moins une transition franchissable) sont importantes.

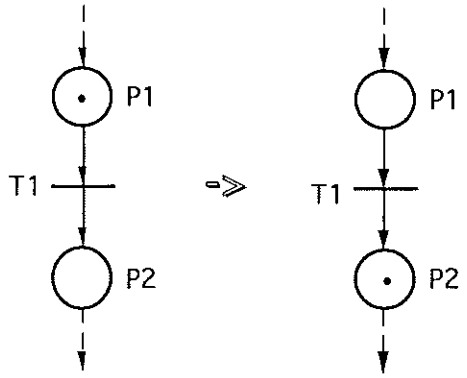


Figure 1.2: Avant et après le franchissement de la transition T_1

1.1 Définitions

Un RdP est un quintuplet $R =< P, T, Pré, Post, M_0 >$ tel que:

$P = \{P_1, P_2 \dots P_n\}$ est un ensemble fini et non vide de places
$T = \{T_1, T_2 \dots T_m\}$ est un ensemble fini et non vide de transitions
$Pré : PxT \rightarrow N$, est l'application d'incidence avant
$Post : PxT \rightarrow N$, est l'application d'incidence arrière
M_0 : marquage initial

On utilise aussi les notation suivantes:

${}^\circ T_j = \{P_i \in P \mid Pré(P_i, T_j) > 0\}$ = ensemble de places d'entrée de T_j
$T_j^\circ = \{P_i \in P \mid Post(P_i, T_j) > 0\}$ =ensemble de places de sortie de T_j
${}^\circ P_i = \{T_j \in T \mid Post(P_i, T_j) > 0\}$ = ensemble de transitions d'entrée de P_i
$P_i^\circ = \{T_j \in T \mid Pré(P_i, T_j) > 0\}$ = ensemble de transitions de sortie de P_i.

Les conditions de validation peuvent s'exprimer maintenant de la façon suivante : la transition T_j est validée pour le marquage M si et seulement si $M(P_i) \geq Pré(P_i, T_j)$ pour tout $P_i \in^\circ T_j$.

On appelle *matrice d'incidence* la matrice :

$$W = Post - Pré \tag{1.1}$$

Pour le RdP de la figure 1.1 la matrice d'incidence est la suivante :

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

$$\Rightarrow W = Post - Pré = \begin{array}{c} \begin{array}{ccccc} T_1 & T_2 & T_3 & T_4 & T_5 \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix} \end{array} \begin{array}{c} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{array}$$

1.2 Equation fondamentale

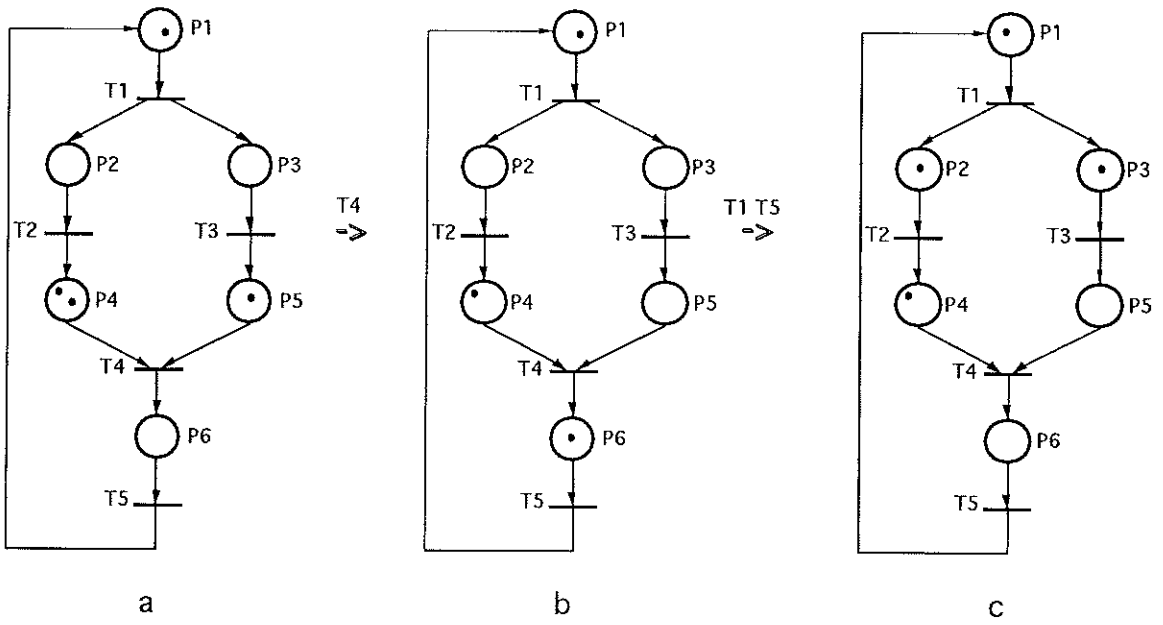


Figure 1.3: Franchissement dans un Réseau de Petri

Soit S une séquence de franchissements réalisable à partir d'un marquage M_i, ceci peut s'écrire $M_i[S>$. Par exemple, pour le marquage $M_i = M_a$ = (1,0,0,2,1,0) du RdP de la figure 1.3a, on a $M_a[T_4>$. La séquence de franchissement $S = T_4$ contient une fois la transition T_4 et zéro fois les autres transitions. Le *vecteur caractéristique* de la séquence S, noté $\underline{S}$, est le vecteur de dimension m dont la composante numéro j correspond au nombre de franchissements de la transition T_j dans la séquence S. Pour l'exemple de la figure 1.3a, $\underline{S}_a$ = (0,0,0,1,0) est la séquence qui fait passer du marquage en figure 1.3a vers celle de la figure 1.3b.

Si la séquence de franchissement S est telle que $M_i[S > M_k$ alors on a *l'équation fondamentale.*

$$M_k = M_i + W * \underline{S} \tag{1.2}$$

Dans notre exemple on a :

$$M_a + W * \underline{S}_a = M_b$$

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 2 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} -1 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix} * \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

On peut avoir des séquenses de franchissement qui contiennent plusieurs transitions. Pour notre exemple dans figure 1.3b $\underline{S_b}$ = (1,0,0,0,1) est la séquence de franchissement qui fait passer du marquage de la figure 1.3b à celle de la figure 1.3c. Cette séquence de franchissement, $\underline{S_b}$, contient une fois la transition T_1, une fois la transition T_5 et zéro fois les autres.

On aura:

$$M_b + W * \underline{S_b} = M_c$$

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} -1 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

1.3 Conflit

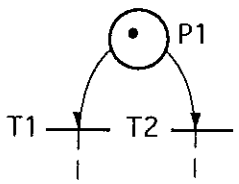


Figure 1.4: Un conflit structurel

Il y a un conflit structurel lorsqu'une même place est en amont de deux ou plusieurs transitions. Si plusieurs transitions sont validées et qu'il n'y a qu'une marque dans la place, une seule de ces transitions pourra être franchie. Il y donc un conflit effectif entre ces transitions et il faut faire un choix. Cette notion est relative à une décision à prendre.

Un conflit (ou conflit structurel) correspond donc à l'existence d'une place P1 qui a au moins deux transitions de sortie T_1,T_2 ... On notera ce conflit par le couple formé d'une place et d'un ensemble de transitions : $< P_1, \{T_1, T_2, \ldots\} >$.

1.4 RdP Généralisé

Un RdP généralisé est un RdP dans lequel les arcs ont un poids. Un poids est un nombre entier strictement positif et par défaut 1.

Lorsqu'un arc $P_i \rightarrow T_j$ a un poids p, cela signifie que la transition T_j ne sera validée que si la place P_i contient au moins p marques, c'est-à-dire au moins un lot (un lot = p marques). Lors du franchissement de cette transition, p marques seront retirées de la place P_i. Lorsqu'un arc $T_j \rightarrow P_i$ a un poids, q, cela signifie que lors du franchissement de T_j, q marques seront ajoutées à la place P_i.

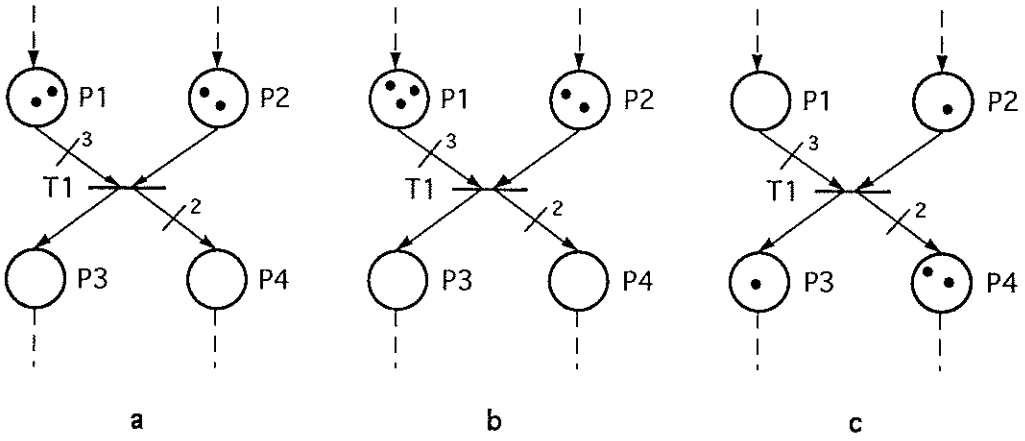


Figure 1.5: Un RdP generalisé

Dans la figure 1.5a la transitons T_1 n'est pas validée parce que P_1 contient moins de trois marques. Dans la figure 1.5b P_1 contient trois marques et P_2 en contient plus d'une. La transition T_1 peut donc être franchie. Le franchissement consiste à retirer trois marques de la place P_1 et une marque de la place P_2 et à ajouter une marque à P_3 et deux marques à P_4. Le marquage de la figure 1.5c est alors obtenu.

1.5 Ce qu'un RdP permet de visualiser

Une des caractéristiques importante des RdP est de pouvoir représenter graphiquement certaines relations, par exemple : le parallélisme, la synchronisation, le partage de ressource, la mémorisation, la lecture et la capacité limitée.
Ces exemples sont trésentés ci-desous.

Parallélisme

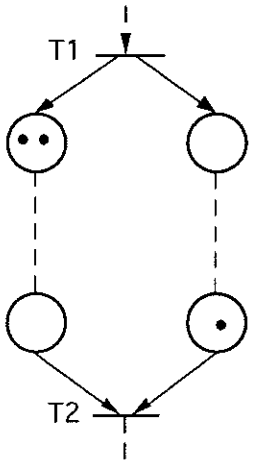


Figure 1.6: Parallélisme

On peut voir qu'après le franchissement de la transition T_1 on a des évolutions parallèles jusqu'au franchissement de la transition T_2. Dans la figure 1.6, on a deux évolutions en parallèle mais on peut également en avoir d'autres. Chacune de ces évolutions se déroule indépendamment l'une de l'autre.

Synchronisation

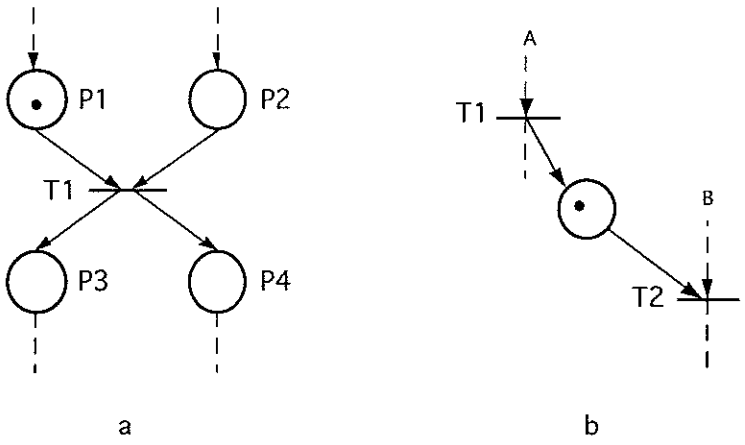


Figure 1.7: Synchronisation

La figure 1.7a représente la synchronisation réciproque de deux évolutions. Si seule une des deux places P_1, P_2 est marquée elle doit attendre que l'autre place soit marquée pour poursuivre l'évolution. Dès que les deux places sont marquées, on peut franchir la transition T_1 et l'évolution des deux parties continue de manière indépendante. C'est-à-dire que la partie la plus rapide doit attendre pour se synchroniser avec la partie la plus lente.

Le cas de la figure 1.7b est différent. L'évolution de la partie A est indépendante de celle de la partie B (l'inverse n'est pas vraie). Le franchissement de la transition T_2 doit se faire après celui de la transition T_1.

Partage de ressource

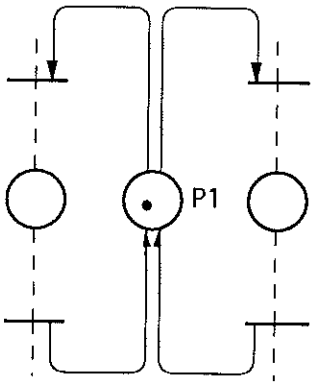


Figure 1.8: Partage de ressource

La ressource, dans la figure 1.8 modélisée par la marque dans P_1, peut être utilisée par la partie gauche du graphe ou par la droite mais pas par les deux en même temps.

Mémorisation

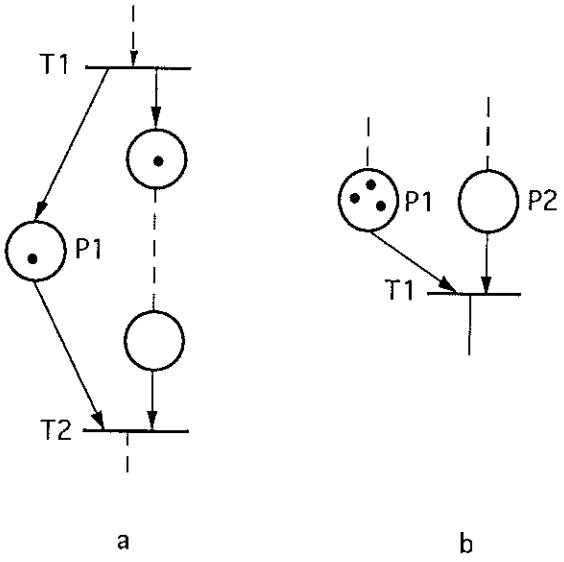


Figure 1.9: Mémorisation

La place $P1$ de la figure 1.9 a mémorise le fait que la transition T_1 a été franchie. On peut aussi mémoriser un nombre par la présence de plusieurs marques dans une place, voir la figure 1.9b.

Lecture

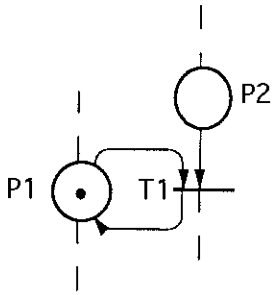


Figure 1.10: Lecture

Le franchissement de la transition T_1, dans la figure 1.10, est conditionnée par le marquage de P_1. On dit qu'on fait une lecture de la place P_1.

Capacité limitée

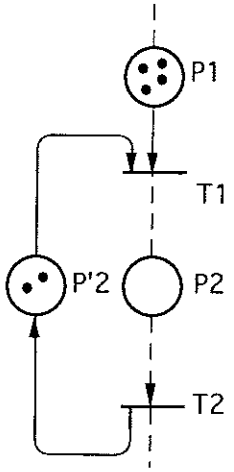


Figure 1.11: Capacité limitée

Dans la figure 1.11, T_1 ne peut pas être franchie plus de deux fois avant que T_2 ne soit franchie. La place P_2' limite le nombre de franchissement de T_1. C'est-à-dire qu'on ne peut jamais avoir plus de deux marques dans la place P_2, elle est donc limitée. On peut remarquer ici un invariant de marquage : $m_2 + m_2' = 2$.

1.6 Réseaux de Petri Temporisés

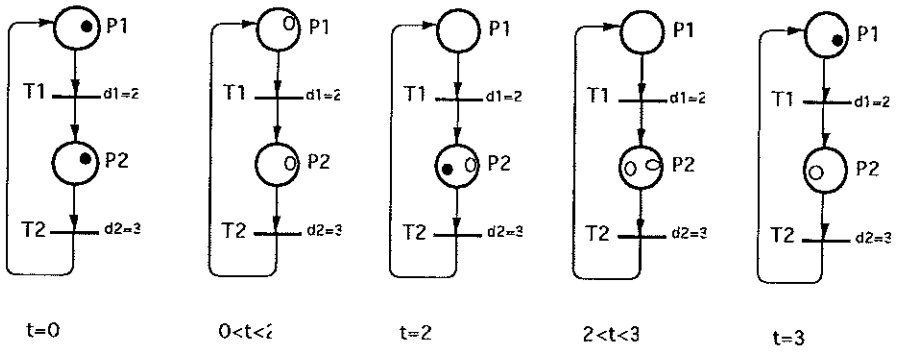


Figure 1.12: Franchissement d'un Réseau de Petri T-temporisé

Dans les RdP Discrets, une marque correspond à un entier, positif ou nul. Une transition est validée si, chacune des places en amont de la transition contient au moins une marque. Un RdP est dit autonome si le fonctionnement d'un RdP ne dépend pas du temps et non autonome quand les instants de franchissement sont connus et indiqués. Un RdP est par défaut discret. Il est pratique de considérer que le franchissement d'une transition a une durée nulle.

Un RdP temporisé permet de décrire un système dont le fonctionnement dépend du temps. Par exemple, il peut s'écouler une certaine durée entre le début d'une opération et la fin de cette opération. Si une marque dans une certaine place indique qu'il y a une opération en cours, un RdP temporisé permettra de rendre compte de cette durée.

Il y a deux façons de modéliser la temporisation : soit les temporisations sont associées aux places (on dira alors que l'on a un RdP P-temporisé), soit elles sont associées aux transitions (on a alors un RdP T-temporisé). On peut facilement passer d'un modèle à l'autre par une transformation particulière. Dans la suite du rapport le modèle T-temporisé sera considéré.

A chaque transition T_j on peut associer une temporisation d_j (nombre rationnel positif ou nul). Un RdP T-temporisé est un couple $< R, Tempo >$ tel que:

R est un RdP
Tempo est une application de l'ensemble T des transitions vers l'ensemble des nombres rationnels positifs ou nuls.
Tempo $(T_j) = d_j$.

Une marque peut avoir deux états : elle peut être réservée pour le franchissement d'un transition ou elle peut être non réservée. A l'instant initial toutes les marques sont non réservées. La transition T_j est validée si les marques non réservées sont en nombre suffisant pour la valider. On peut alors réserver, à l'instant t, les marques nécessaires à son franchissement. Les marques restent réservées pendant l'intervalle $[t, t + d_j[$ et le franchissement est effectué à l'instant $t = t + d_j$.

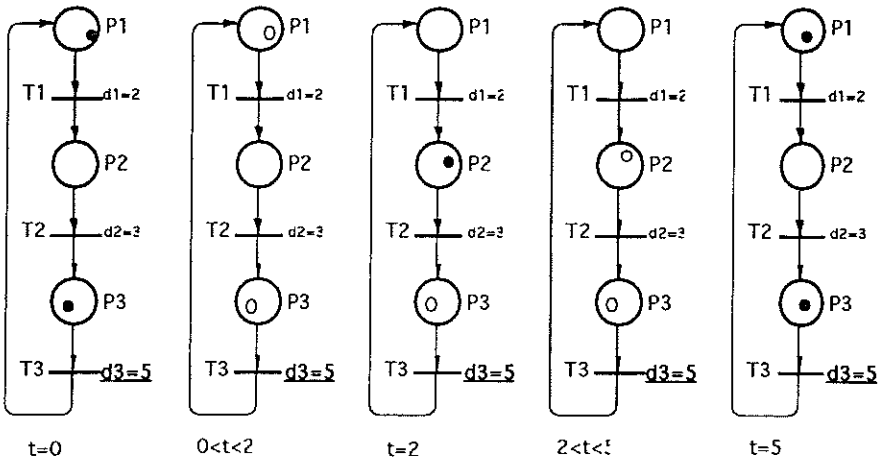


Figure 1.16: Un Réseau de Petri saturé

Dans la figure, 1.16, les franchissements sont tous limités à une marque à la fois, (mono-serveur). Pour alléger le graphis cette limitation n'est pas dessinée
La machine la plus lente correspond à la transition T_3, car à cette transition est associée la durée la plus longue, d_3. Le stock alimentant cette machine correspond à la place P_3. On voit que cette place n'est jamais vide, c'est-à- dire que la machine la plus lente n'attend jamais de pièces. Le réseau est dit *saturée*, figures 1.16 et 1.17.

Le nombre des pièces dans la place P_3 en fonction du temps est illustré dans la figure 1.17:

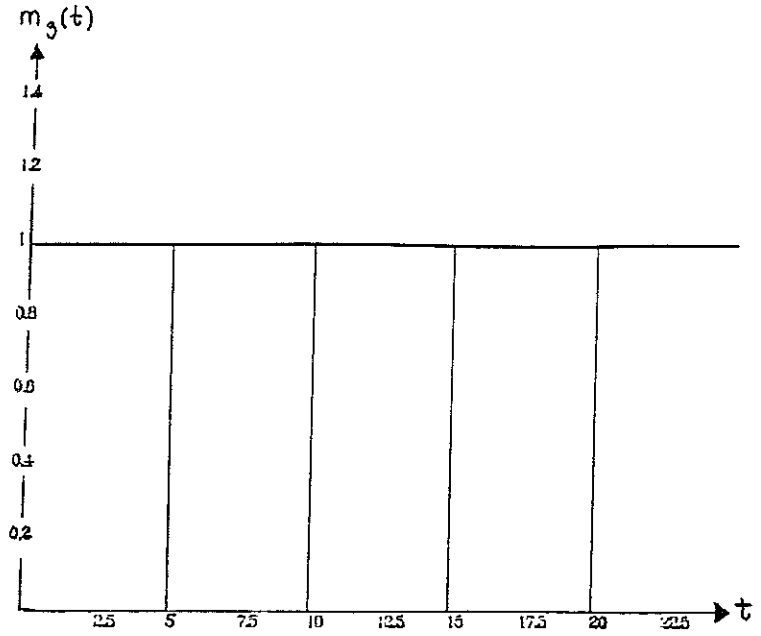


Figure 1.17:

Considérons à nouveau la même ligne de production dans laquelle la durée de la machine 3, d_3, a été changée.

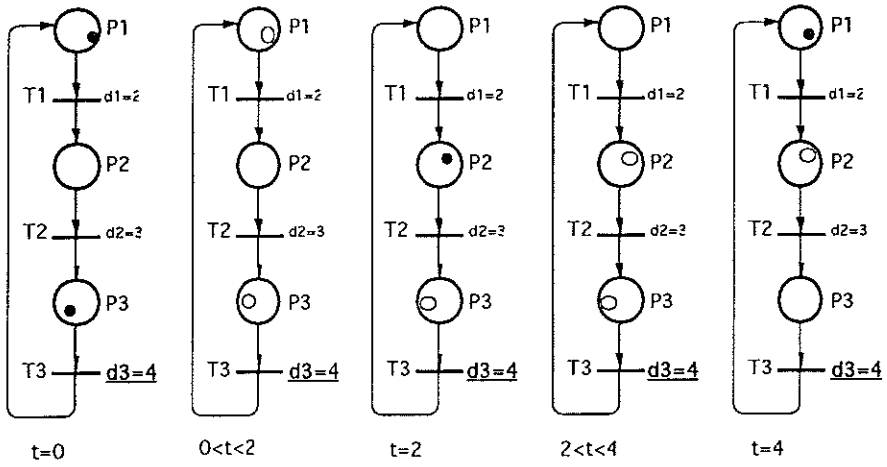


Figure 1.18: Un Réseau de Petri non saturé

La machine la plus lente correspond encore à la transition T_3 et le stock l'alimentant à la place P_3. Au cours de la simulation cette place, P_3, est de temps en temps vide, c'est-à-dire que la machine la plus lente doit attendre des pieces. Le réseau est dit, dans ce cas, *non saturée* figures 1.18 et 1.19.

Le nombre des pièces dans la place P_3 en fonction du temps et à nouveau illustré, voir la figure 1.19:

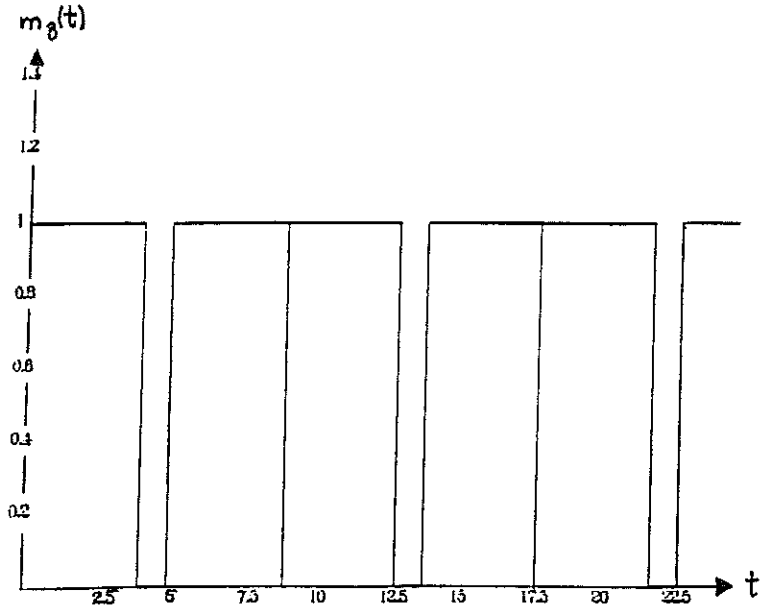


Figure 1.19:

Chapitre 2

Réseaux de Petri Continus

Pour les RdP Continus (RdPC) le nombre de marques contenues dans une place n'est plus un entier mais un nombre réel. Il a été montré que le comportement d'un RdPC est un cas limite de celui d'un RdP discret, obtenu lorsque chacune des marques du réseau est divisée en k jetons et que k tend vers l'infini [David 89] [David 90].

Dans un RdPC, les places sont représentées par un double cercle et les transitions par un rectangle. Cette représentation est utile pour les RdP Hybrides qui contiennent des parties discrètes et des parties continues.

Le franchissement d'une transition consiste à retirer une quantité x des places en amont et à l'ajouter dans les places en aval.

Le pendant d'un RdP T-temporisé est un RdPC ayant des fréquences de franchissement associées aux transitions, $U_j = 1/d_j$.

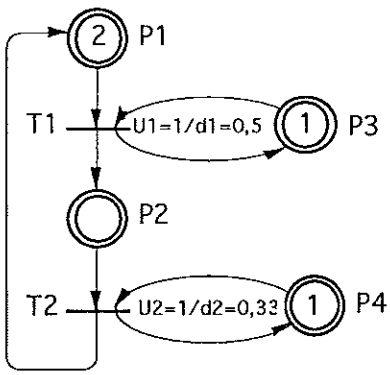


Figure 2.1: Un RdPC temporisé

Deux modèles RdPC temporisé ont été définis : *les RdPC à vitesses Constantes (RdPCC)* et *les RdPC à vitesses Variables (RdPCV).*

2.1 Réseaux de Petri à Vitesses Constantes

Les RdPCC se caractérisent par des intervalles de temps durant lesquels les vitesses de franchissement sont constantes, ces intervalles sont appelés phases de fonctionnement.

Dans ce modèle, U_1 et U_2 sont appelées les vitesses maximales de franchissement. Entre les deux instants t et $t + dt$ la quantité $U_1 * dt$ est retirée de P_1 et ajoutée à P_2.

A chaque transition une vitesse instantanée est associée. Dans la figure 2.2 on a deux transitions et donc deux vitesses instantanées, v_1 et v_2. Elles sont associées à T_1 et T_2 respectivement. La place P1 n'est pas vide, elle est alors franchie à une vitesse instantanée égale à sa vitesse maximale, $v_1 = U_1$. Par contre, la place P2 est vide mais comme il y a un flot, v_1, qui traverse la transition T_2 cette transition peut aussi être franchie. Comme $U_2 < v_1$ la vitesse instantanée v_2 est égale à sa vitesse maximale, $v_2 = U_2$.

Les équations du marquage dans les deux places deviennent:

$$m_1 = m_1(0) - U_1 * t + U_2 * t \tag{2.1}$$

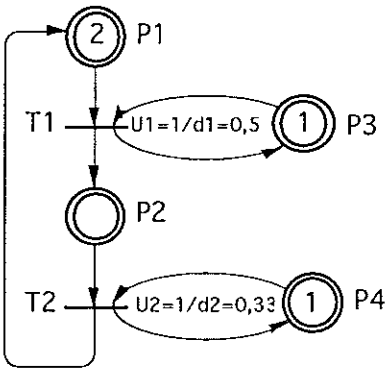
$$m_2 = m_2(0) + U_1 * t - U_2 * t \tag{2.2}$$

A partir d'un certain temps t_1, P_1 est vide et T_1 ne peut plus être franchie à sa vitesse maximale. A partir de ce moment la transition T_2 est dite fortement validée car aucune place d'entrée de T_2 n'est vide et la transition T_1 est dite faiblement validée car sa place d'entrée P_1 est vide et alimentée à la vitesse U_2. T_1 est alors franchie à une vitesse inférieure à sa vitesse maximale U_1: $v_1 = U_2 < U_1$,$v_2 = U_2$.

L'évolution peut comporter plusieurs phases. Chacune des phases de fonctionnement se caractérise par un vecteur des vitesses de franchissement constant. Le passage d'une phase à la suivante se produit lorsque le marquage d'une place devient nul. Le temps de calcul nécessaire à la simulation d'un RdPCC dépend directement du nombre de phases de fonctionnement.

Le marquage du RdPCC diffère souvent du marquage moyen obtenu du RdP discret correspondant. Cette remarque est générale et l'erreur relative entre le RdPCC et le modèle discret est d'autant plus faible que le marquage dans le réseau est important. Le modèle RdPCC est bien adapté à la simulation de systèmes mettant en jeu un grand nombre d'entités.

Exemple - RdPCC



$$0 < t < 12 :$$

$$m_1(t) = m_1(0) - U_1 * t + U_2 * t = 2 - \frac{t}{6}$$

$$m_2(t) = m_2(0) - U_2 * t + U_1 * t = \frac{t}{6}$$

$$t > 12 :$$

$$m_1(t) = 0$$

$$m_2(t) = 2$$

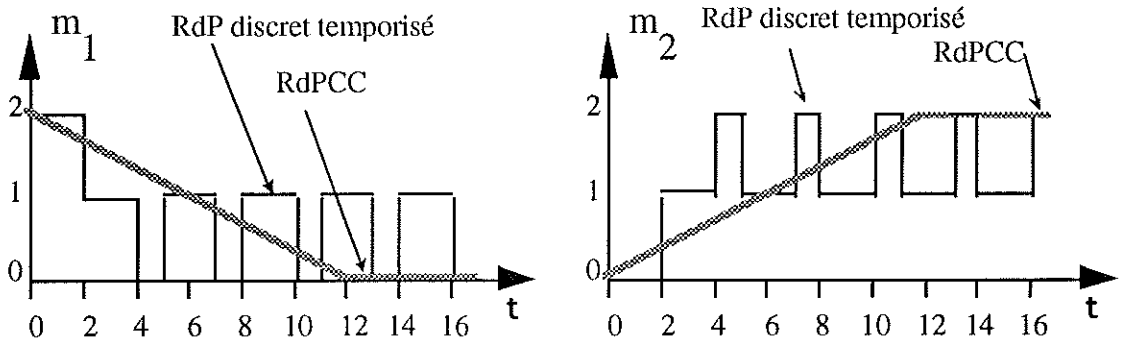


Figure 2.2: L'évolution des marquages dans les deux places P1 et P2 avec le modèle RdPCC

2.2 Réseaux de Petri à Vitesses Variables

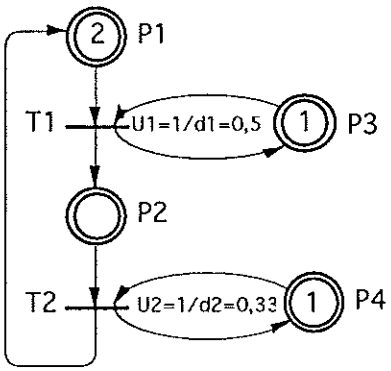
Pour le modèle RdPCV, la vitesse de franchissement d'une transition dépend à chaque instant du nombre de marques contenues dans les places d'entrée de la transition considérée. La vitesse instantanée de franchissement est donnée par la relation suivante :

$$v_j = U_j * Min_i(m_i(t)) \qquad avec \qquad P_i \in^\circ T_j \tag{2.3}$$

Ce système d'équations reste vrai tant que l'expression de chacune des vitesses reste la même, c'est-à-dire tant que les places d'entrée les moins marquées de chacune des transitions restent les moins marquées.

Il est possible de définir des phases de fonctionnement du modèle RdPCV, une phase étant définie cette fois par un système d'équations différentielles donné. Il a été montré que les RdPCV permettent d'obtenir une très bonne approximation des RdP Discrets. Malheureusement les marquages et les vitesses de franchissement s'obtiennent par intégration d'un système d'équations différentielles et par conséquent, le temps de calcul nécessaire à la simulation peut être assez long.

Exemple - RdPCV



$$0 < t < 3 * ln3 :$$

$$v_1(t) = U_1 * min(m_1(t), m_3(t)) = U_1 * m_3(t) = U_1$$
$$v_2(t) = U_2 * min(m_2(t), m_4(t)) = U_2 * m_2(t)$$

$$\frac{dm_1(t)}{dt} = U_2 * m_2(t) - U_1$$
$$\frac{dm_2(t)}{dt} = U_1 - U_2 * m_2(t)$$
$$m_1(t) = \frac{1}{2} + \frac{3}{2}e^{-\frac{1}{3}t}$$
$$m_2(t) = \frac{3}{2} - \frac{3}{2}e^{-\frac{1}{3}t}$$

$$t > 3 * ln3 :$$
$$m_1(t) = \frac{2}{3} + \frac{1}{3}e^{-\frac{1}{2}t}$$
$$m_2(t) = \frac{4}{3} - \frac{1}{3}e^{-\frac{1}{2}t}$$

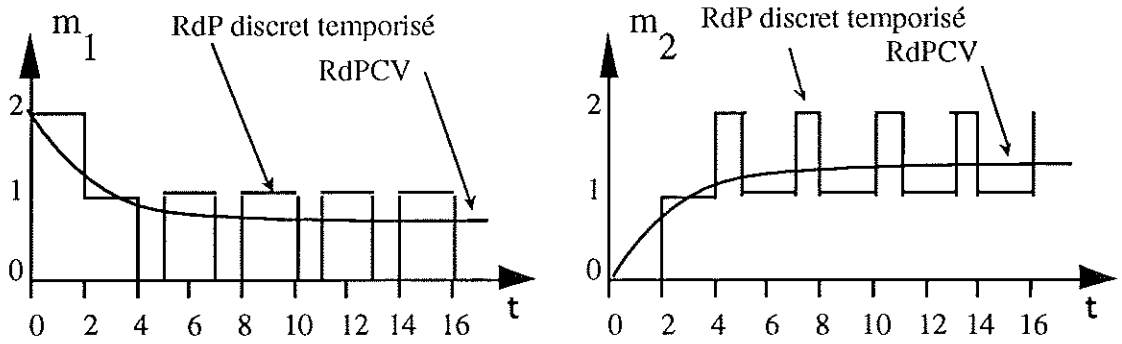


Figure 2.3: L'évolution des marquages dans les deux places P1 et P2 avec le modèle RdPCV

2.3 Conclusion

Le RdPCC permet de simuler rapidement le comportement de systèmes dans lesquels un grand nombre d'événement doit être pris en compte à cause des nombreuses entités y circulant. Cependant cette rapidité de simulation s'opère au détriment de la qualité de l'approximation.

Le RdPCV réalise une bonne approximation du comportement discret mais les calculs nécessaires à la simulation font qu'en pratique il n'est pas utilisable pour simuler le comportement de systèmes de taille importante.

Chapitre 3

Réseaux de Petri Continus Asymptotiques

Le RdPCA est un nouveau modèle continu permettant à la fois de réaliser des simulations rapides (comme pour le RdPCC) et de fournir une bonne approximation du comportement discret (comme le RdPCV).

L'idée de départ est d'obtenir une approximation du modèle RdPCV au moyen d'un modèle continu pour lequel les vitesses franchissement des transitions sont constantes par morceaux tout comme c'est le cas pour le RdPCC. Cette condition garantit un modèle facile à simuler car l'intégration des systèmes d'équations differentielles est immédiate.

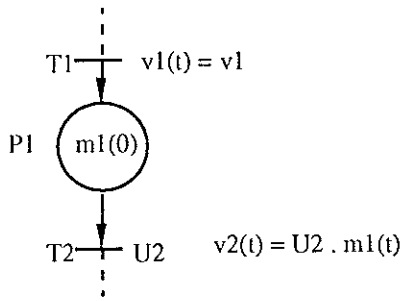
Pour faciliter la représentation graphique du RdPCA on représentera par des traits simples les places et les transitions.

3.1 Présentation intuitive

Nous allons introduire progressivement les différentes notions qui caractèrisent l'évolotion du modèe RdPCA.

3.1.1 Marquage Asymptotique

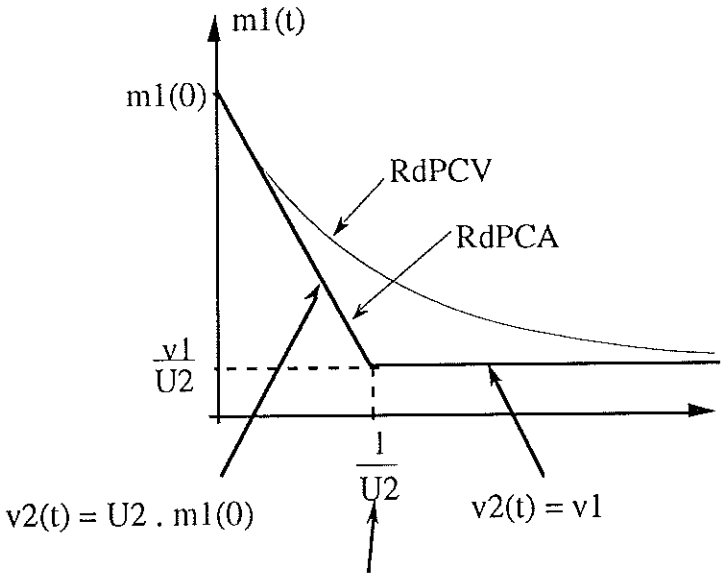
Pour le RdPCV on a:



Si v_1 est constant alors m_1 tends vers une valeur asymptotique

$$m_1 = \frac{v_1}{U_2} \text{ telle que } v_2(t) = v_1$$

Pour avoir une bonne aproximation du modèle RdPCV, le RdPCA a un comportement lineaire ainsi que l'indique la figure ci-dessous. En partiqulir, la demi-droite horizontale est l'asymptote au modele RdPCV.



Cette valeur de l'asymptotie est appellée *Marquage asymptotique.*

Pour un RdPCA, tout comme pour un RdPCC, un intervalle de temps durant lequel le vecteur des vitesses de franchissement instantanées reste constant est appellé *phase de fonctionnement.* Pour un RdPCC, le passage d'une phase à la suivante se produit lorsque le marquage d'une place devient nul. Pour le RdPCA, ce passage se produit lorsque le marquage d'une phase atteint sa *valeur asymptotique.* Cela signifie que le bilan dynamique de cette place vaut zéro. Le *bilan dynamique* représente la différence entre la quantité de marques entrant dans la place par unité de temps (c'est-a-dire $v_{entrée}(t)$) et la quantité de marques sortant de cette place par unité de temps (c'est-a-dire $v_{sortie}(t)$). Le bilan dynamique vaut alors zéro lorsque $v_{entrée}(t) - v_{sortie}(t) = 0$. Si on a seulement un transition de sortie ($P_i^\circ = \{T_j\}$) on a :$v_{sortie}(t) = m * U_{sortie}$, et la valeur asymptotique est égale à :

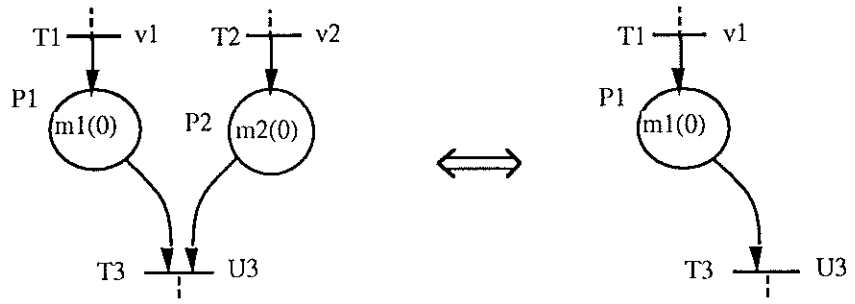
$$m^* = \frac{v_{entrée}}{U_{sortie}} \tag{3.1}$$

3.1.2 Vitesse instantanée

Pour un RdPCA, les vitesses instantanées sont calculées de la même manière que celles du RdPCV, voir équation 2.3. C'est-à-dire :

$$v_j = U_j * Min_i(m_i(t)) \qquad avec \qquad P_i \in^{\circ} T_j \tag{3.2}$$

3.1.3 Place critique



Pour le RdPCV on a:

$$v_3 = U_3 * min[m_1(t), m_2(t)] = U_3 * m_1(t)$$

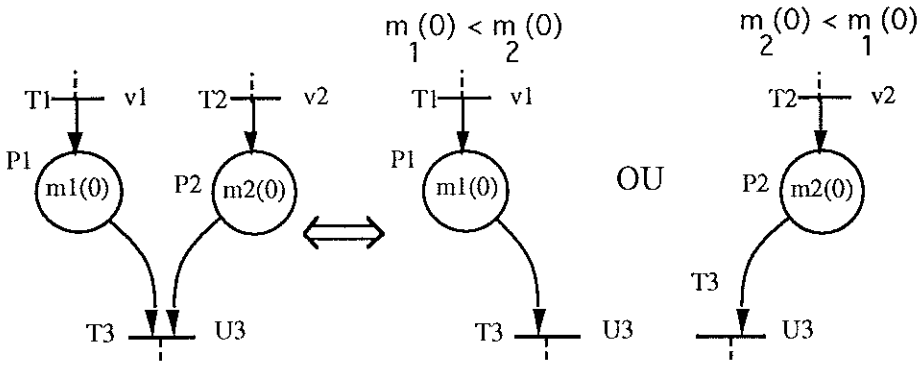
La vitesse instantanée dépend du nombre de marques dans la place critique à chaque instant.

Pour le RdPCA, la vitesse est constante par morceaux (comme pour le RdPCC). On approxime l'exponentielle qu'on avait en RdPCV par des droites. La vitesse dépend du nombre de marques dans la place critique au début de la phase et non pas à chaque instant.

$$v_3 = U_3 * min[m_1(t_0), m_2(t_0)] = U_3 * m_1(t_0) \qquad t_0 = \text{date de debut de la phase}$$

Pour le comportement du réseau c'est alors seulement la place critique qui est important.

Choix de place critique:



3.2 Définitions et propriétés

Toutes les équations ci-dessous sont données pour des réseaux non généralisés. Leur transformation pour les réseaux généralisés ne pose aucun problème.

Definition 3.1

Pour un RdPCA quelconque, le marquage asymptotique d'une place d'entrée P_i, noté m_i^*, d'une transition T_j est défini par le rapport du bilan de cette place, sans tenir compte de la quantité sortant de P_i à travers T_j par la fréquence de franchissement de la transition T_j.
Ceci peut s'écrire :

$$m_i^* = \frac{Post_{i,j} * v_j + \sum w_{i,k} * v_k}{U_j}$$

Le marquage asymptotique d'une place dépend du vecteur vitesse, il peut donc changer lors d'un changement de phase fonctionnement.

Propriété 3.1

Si les trois conditions suivantes sont remplies:

1. P_i est la seule place d'entrée d'une transition T_j, c'est-à-dire $\{P_i\} =^\circ T_j$.
2. P_i n'est pas une place de sortie de T_j, c'est-àdire $P_i \in T_j^\circ$.
3. le bilan de P_i est nul à l'instant considéré.

alors le marquage de P_i est égal à son marquage asymptotique m_i^*.
Ceci peut s'écrire :

si $(P_i =^\circ T_j$ et $P_i \in T^\circ j)$
alors $(m_i' = 0 \quad \Rightarrow \quad m_i(t) = m_i^*)$

Propriété 3.2

Soit P_i une place d'entrée de T_j. Si d'une phase de fonctionnement débutant à l'instant t_0 et de durée supérieure ou égale à $\frac{1}{U_j}$, on a : $v_j = U_j * m_i(t_0)$, alors à la date $t_0 + \frac{1}{U_j}$, le marquage $m_i(t_0 + \frac{1}{U_j})$ devient égal au marquage asymptotique, m_i^*.

La propriété 3.2 indique alors que lorsque la vitesse d'une transition T_j est calculée, on sait que $\frac{1}{U_j}$ unités de temps après, le marquage asymptotique de la place d'entrée de la transition aura atteint sa valeur asymptotique.

Si la première des trois conditions données en propriété 3.1 n'est pas remplie (transition T_j a plusieurs places en amont) il faut savoir quelle place a le marquage le plus faible.

Definition 3.2

Pour un marquage donné d'un RdP quelconque, la *place critique* associée à une transition T_j est la place d'entrée de T_j ayant le marquage le plus faible à cet instant.
En cas d'égalité de marquage, la place critique est choisie arbitairement parmi celles ayant le marquage le plus faible.

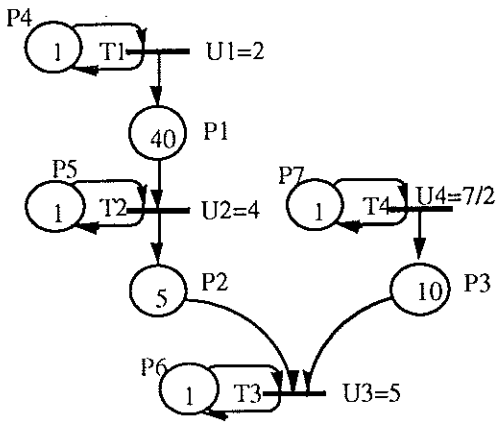
A chaque transition est associée, à chaque instant, une et une seule place critique.

Definition 3.3

Nous appellons *fonction critique*, à un instant donné, la fonction qui associe à chaque transition sa place critique. Nous notons X cette fonction.

$$X(j) = i \Rightarrow P_i \in^{\circ} T_j \qquad et \qquad m_i = Min_k(m_k)$$

Exemple



Dans la figure 3.2, la fonction critique est la suivante:

$$(X(1), X(2), X(3), X(4)) = (4, 5, 6, 7)$$

3.3 Etats

L'évolution d'un RdPCA se caractérise par des intervalles de temps durant lesquels certaines grandeurs restent constantes. Deux grandeurs sont à distinguer :

1. le vecteur des vitesses instantanées de franchissement des transitions
2. la fonction critique

Le couple, composé des ces deux éléments, définit l'état comportemental. Une phase comportementale est un intervalle de temps durant lequel les deux grandeurs spécifiées restent constantes.

3.4 Evénements

Il existe deux types d'événements susceptibles de modifier un état comportemental:

Type1 : Le marquage de la place critique $P_{X(j)}$ d'une transition T_j atteint sa valeur asymptotique.

Type2 : Le marquage d'une place d'entrée non critique P_i de T_j devient égal au marquage de la place critique $P_{X(j)}$ de cette transition.

Chacun de ces deux événements a une conséquence sur l'état comportemental du réseau. Chaque événement peut changer soit le vecteur vitesse, soit la fonction critique, soit les deux simultanément. Ces deux types d'évévenement sont à la base de l'algorithme donné au paragraphe 3.5.

Propriété 3.3

On peut remarquer qu'il est nécessaire de recalculer certaines vitesses seulement dans deux cas:

1. lorsqu'il y a un changement de place critique (événement type 2) et que l'ancienne place critique n'a pas encore atteint son marquage asymptotique.

2. lorsque le marquage d'une place critique atteint sa valeur asymptotique (événement type 1).

3.5 Algorithme d'interprétation

Pas1 : Initialisation
1. t=0 (instant initial).
2. L'ensemble $T_C$des transitions dont les vitesses doivent être calculées est initialisé avec l'ensemble de toutes les transitions du réseau: $T_C = T$.
3. La fonction critique X est initialisée.

Pas2 : 1. Pour toute transition T_j de T_C:
Calculer la vitesse instantanée de T_j: $v_j(t) = m_{X(j)}(t) * U_j$.
2. Pour chaque place P_i, calculer le bilan b_i: $b_i = \frac{dm_i(t)}{dt}$

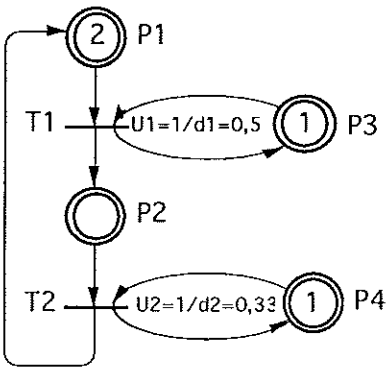
Pas3 : Pas 3.1. Evénements possibles
Pour chaque transition du réseau,
1. pour chaque place d'entrée P_i de T_j dont le bilan est strictement inférieur au bilan de la place critique $P_{X(j)}$, il se produit un événement de type 2 lorsque les marquages de ces deux places deviennent égaux.
2. si la place critique de T_j n'a pas un bilan nul, il se produit un événement de type 1 dès que le marquage de $P_{X(j)}$ devient égal à son marquage asymptotique.
Pas 3.2. Prochain événement
Si aucun événement n'est détecté au Pas 3.1, alors FIN.
Sinon soit t_e la date du premier de tous les événements obtenus au Pas 3.1 et soient T_{je} et P_{ie} soit la transition et la place correspondante à cet événement.

Pas4 : Calcul du marquage des places à la date t_e.
Pour chaque place P_i: $m_i(t_e) = m_i(t) + (t_e - t) * b_i$.
Puis faire $t = t_e$ (la date actuelle de la simulation devient celle de l'événement).

Pas5 : Soit $k = X(j_e)$. Si l'événement est de type2:
1. La fonction critique est modifiée ainsi: $X(j_e) = i_e$.
2. Si le bilan de P_k est nul alors aller au Pas 3 (Il n'y a pas de vitesse à calculer).

Pas6 : Déterminer l'ensemble T_C des transitions dont il faut recalculer la vitesses grâce à l'algorithme de Gauss-Seidel.
Puis aller au pas 2.

Exemple - RdPCA



Avec l'algorithme donné à la page précédente on obtiendra:

Pas1 : 1. t=0.
2.$T_C = T_1, T_2$
3. $(X(1), X(2)) = (3, 2)$.
$M(0) = (2, 0, 1, 1)$.

Pas2 : 1. vitesses instantanées: $(v_1, v_2) = (1/2, 0)$.
2. bilan $(b_1, b_2, b_3, b_4) = (-1/2, 1/2, 0, 0)$.

Pas3 : 1. Evénements possibles:
T_1: Type 2 à la date 2.
T_2: Type 2 à la date 2.
2. Prochain événement:
$t_e = 2$
$T_{je} = 1$ et 2
$P_{ie} = 1$ et 2.

Pas4 : marquage la date $t_e = 2$.
$M(2) = (1, 1, 1, 1)$
$t = t_e = 2$

Pas5 : fonction critique $(X(1), X(2)) = (1, 2)$.

Pas6 : $T_C = 1, 2$.

Pas2 : 1. vitesses instantanées: $(v_1, v_2) = (1/2, 1/3)$.
2. bilan $(b_1, b_2, b_3, b_4) = (-1/6, 1/6, 0, 0)$.

Pas3 : 1. Evénements possibles:
T_1: Type 1 à la date 2+2=4.
T_2: pas d'événement possible.
2. Prochain événement:
$t_e = 4$
$T_{je} = 1$
$P_{ie} = 1$

Pas4 : marquage la date $t_e = 4$.
$M(4) = (4/6, 8/6, 1, 1)$
$t = t_e = 4$

Pas5 : L'événement n'est pas du type 2.

Pas2 : 1. vitesses instantanées: $(v_1, v_2) = (1/3, 1/3)$.
2. bilan $(b_1, b_2, b_3, b_4) = (0, 0, 0, 0)$.

Plus d'événements possibles
Marquage final obtenu:
$M(4) = (2/3, 4/3, 1, 1)$

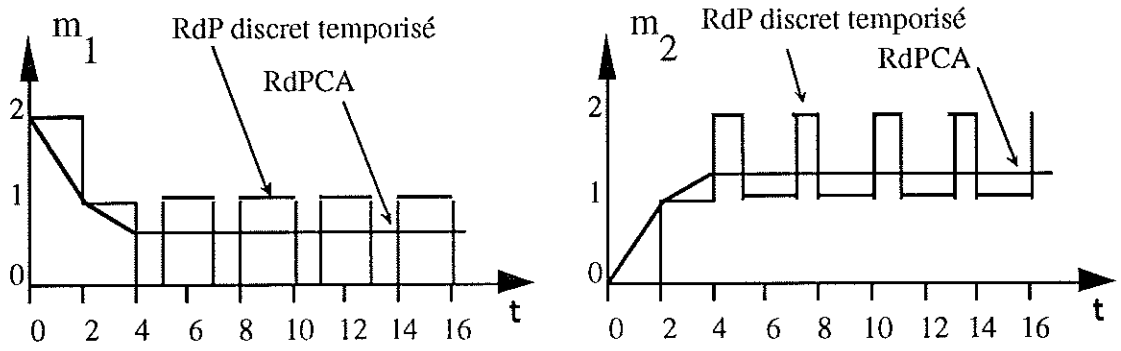


Figure 3.1: L'évolution des marquages dans les deux places P1 et P2 avec le modèle RdPCA

On constante que l'évolution du modèl RdPCA approxime du façon bien satisfaisante le modèle discret. En particulier la valeur finale constante du RdPCA est égale à la valeur moyenne du modèle discret en régime périodique.

Chapitre 4

Le Programme

4.1 Description du programme

La structure du programme du RdPCA existait déjà et avait été implantée par Jean Le Bail. Ce programme est composé de six parties principales et d'une partie réalisant l'initialisation des variables. Ce chapitre contient une brève explication du rôle de chaque partie .

Initialisation

Cette tâche est uniquement effectuée une fois au début de l'exécution du programme. Toutes les variables sont initialisées à cet instant.

Partie 1 - Place critique

Dans cette partie on détermine pour chaque transition sa place critique.

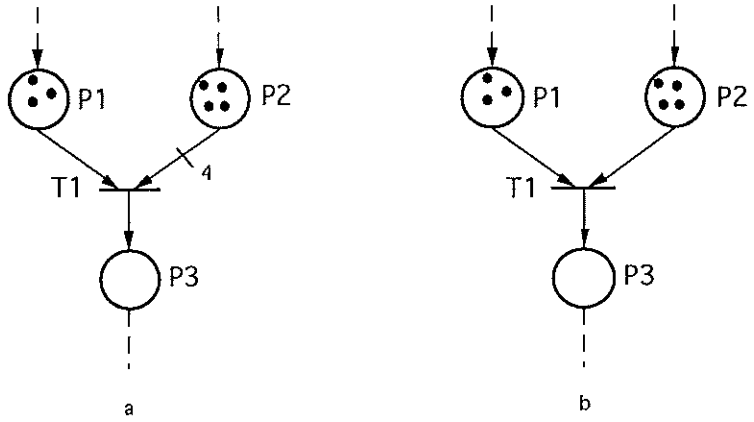


Figure 4.1: Réseaux avec et sans poids sur les arcs

Pour trouver la place critique d'une transition T_j, le programme calcule pour chaque

place P_i, en amont de T_j, le nombre de lots qu'elle contient.

$$\text{nombre de lots} = \frac{\text{nombre de marques dans la place}}{\text{poids de l'arc sortant de la place}} \tag{4.1}$$

La place contenant le nombre de lots le plus faible est donc la place critique de cette transition.

Dans la figure 4.1a, la place P2 est la place critique, en figure 4.1b c'est la place P1.

Si le réseau n'est pas généralisé un lot est alors égal à une marque car le poids de l'arc est égal à 1, voir equation 4.1

Ce calcul est à nouveau effectué pour les transitions indiquées en partie 6 du programme. Cependant, la première fois il est fait pour toutes les transitions du réseau.

Partie 2 - Vitesse instantanée et date de l'événement du type 1

La vitesse instantanée est donnée par:

$$\text{vitesse instantanée} = \frac{\text{nombre de lots de la place critique}}{\text{temporisation d}} \tag{4.2}$$

Un changement de place critique nécessite le calcul des vitesses instantanées pour les transitions concernées.

La date du prochain événement du type1 (marquage asymptotique atteint) est aussi calculée pour les mêmes transitions comme les vitesses. Cette date est donnée par propriété 3.2:

$$date + \frac{1}{U_j} \tag{4.3}$$

Partie 3 - Bilan

Le bilan est calculé pour chaque place du réseau. Le bilan correspond au flot de la place, c'est-à-dire à la différence entre ce qui entre et ce qui sort.

$$bilan = \sum V_{entree} - \sum V_{sortie} \tag{4.4}$$

Partie 4 - Recherche de l'événement suivant

Pour chaque transition, l'instant où la place critique atteint son marquage asymptotique (type1), est déjà calculé, (équation 4.3). La date du prochain changement de place critique est aussi calculée. Parallèlement, une comparaison des différentes dates calculées est fait. La date la plus petite est gardée en mémoire. Cette date est appelée *date*.

Non seulement la date du prochain événement est gardée en mémoire mais aussi son type et l'indice de la transition associée à cet événement.

Si l'événement est de type1 (marquage asymptotique atteint) on indique à l'aide d'une variable, *calcul*, s'il est nécessaire de recalculer certaines vitesses (pour savoir lesquelles voir Partie 5). Si l'événement n'est pas du type1, mais du type2, un changement de place critique est d'abord fait. Le variable *calcul* est mis à 1, seulement si l'ancienne place

critique n'a pas déjà atteint son marquage asymptotique, voir propriété 3.3, ceci indique qu'il y a des transitions où les vitesses sont à recalculer. Un événement du type0 veut dire qu'il n'y a plus d'événement possible. C'est alors une indication de fin de simulation.

Partie 5 - Nouveau marquage et nouveau temps

Pour chaque place dans le réseau, le marquage à l'instant du prochain événement est calculé:

$$m_i(date + temps) = date * bilan_i + m_i(temps) \tag{4.5}$$

le temps est aussi augmonté:

$$temps = date + temps \tag{4.6}$$

Partie 6 - Détermination des transitions à calculer

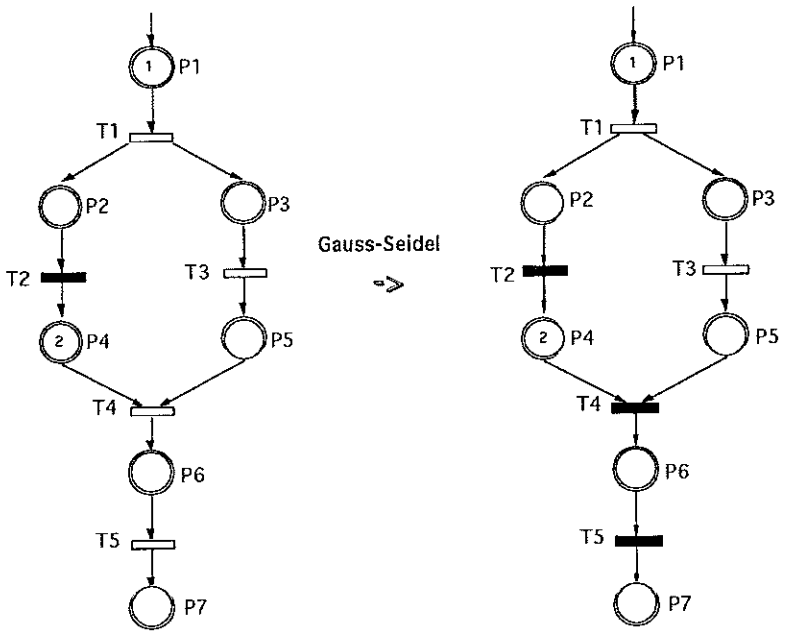


Figure 4.2: Transitions a recalculer avec l'algorithme de Gauss-Seidel

La variable *calcul*, introduite dans la parite 4, indique s'il faut recalculer des vitesses ou non. Si ce n'est pas nécessaire on va directement à la partie 4 et on recommence. S'il faut les recalculer il faut d'abord chercher quelles sont les transitions concernées.

L'algorithme de Gauss-Seidel est utilisé à cette fin. Grâce à cet algorithme, toutes les transitions qui sont directement ou indirectement en aval de la transition, source de l'événement, doivent être recalculées.

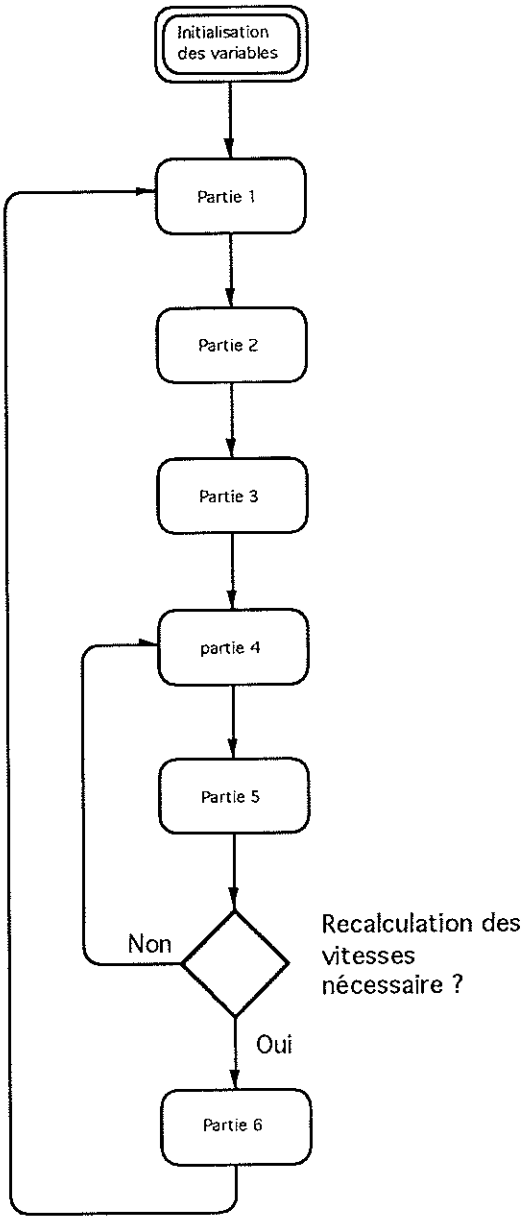
Exemple

Supposons que, dans la figure 4.2a, la transition T2 soit à l'origine de l'événement nécessitant le calcul de certaines vitesses.

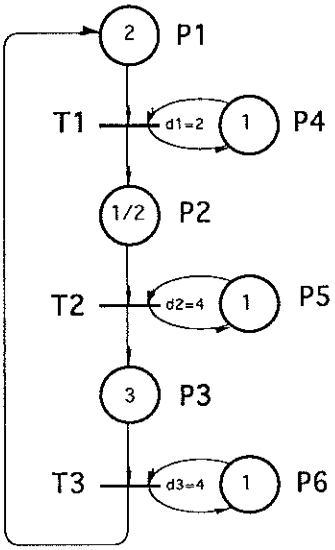
Avec l'algoritme de Gauss-Seidel, on trouve qu'il faut recalculer les vitesses instantanées des transitions T_2, T_4, et T_5, voir figure 4.2b.

Structure du programme

Ceci la structure du programme:



4.2 Exemple d'utilisation du programme



Les fréquences U, associées aux transitions sont égales aux inverses des temporisations d.

$t = 0$
$M(0) = (2, 1/2, 3, 1, 1)$
fonction critique $= (X(1), X(2), X(3)) = (4, 2, 6)$
vitesse instantanée $= (v_1, v_2, v_3) =$*(1/2, 1/8, 1/4)*
bilan $= (b_1, b_2, b_3, b_4, b_5, b_6) = (-1/4, 3/8, -1/8, 0, 0, 0)$
événements possibles:
T_1 : changement de place critique, $t = 4$
T_2 : changement de place critique, $t = 4/3$
T_3 : changement de place critique, $t = 16$

$t = 4/3$
$M(4/3) = (5/3, 1, 17/6, 1, 1)$
fonction critique $= (X(1), X(2), X(3)) = (4, 5, 6)$
vitesse instantanée $= (v_1, v_2, v_3) =$*(1/2, 1/4, 1/4)*
bilan $= (b_1, b_2, b_3, b_4, b_5, b_6) = (-1/4, 1/4, 0, 0, 0, 0)$
événements possibles:
T_1 : changement de place critique, $t = 4/3 + 8/3$
T_2 : pas d'événement possible.
T_3 : pas d'événement possible.

$t = 4$
$M(4) = (1, 5/3, 17/6, 1, 1)$

fonction critique = $(X(1), X(2), X(3)) = (1, 5, 6)$
vitesse instantanée = (v_1, v_2, v_3) =*(1/2, 1/4, 1/4)*
bilan = $(b_1, b_2, b_3, b_4, b_5, b_6) = (-1/4, 1/4, 0, 0, 0, 0)$
événements possibles:
T_1 : marquage asymptotique, *t=4+2*
T_2 : pas d'événement possible.
T_3 : pas d'événement possible.

$t = 6$
$M(6) = (1/2, 13/6, 17/6, 1, 1)$
fonction critique = $(X(1), X(2), X(3)) = (1, 5, 6)$
vitesse instantanée = (v_1, v_2, v_3) =*(1/4, 1/4, 1/4)*
bilan = $(b_1, b_2, b_3, b_4, b_5, b_6)$ =*(0, 0, 0, 0, 0, 0)*
$M(final) = (1/2, 13/6, 17/6, 1, 1, 1)$

Dans cet exemple, les vitesses instantanées sont recalculées chaque fois.

Remarque1: On peut vérifier que la propriété 3.3 est correcte. Quand on a un changement de place critique et que l'ancienne a déjà atteint son marquage asymptotique il n'est pas nécessaire de recalculer les vitesses. Voir cet exemple, $t = 4/3$ et $t = 4$.

Remarque2: On peut vérifier que la propriété 3.2 est correcte. Le marquage asymptotique d'un place critique X(j) est atteint d_j unité de temps après le dernier calcul de sa vitesse v_j.

Remarque3: Quand le bilan est égal à zéro pour toutes les transitions la simulation est finie.

4.3 Conclusion

Le programme original, qui était implanté par Jean Le Bail, a été vérifié par des simulations de réseaux très simples. Des comparaisons entre les résultats obtenus et ceux de simulations discrètes ont été faites. En plus ils ont été vérifiés par des calculs à la main. Pour ces réseaux le programme n'avait pas des défauts. Après cette étude préliminaire du fonctionnement du programme, d'autres structures plus compliquées pouvaient être étudiées.

Chapitre 5

Structures particulières

Il est intéressant d'étudier le comportement de l'algorithme vis-à-vis des quatre structures suivantes:

Cas 1 - Cumul

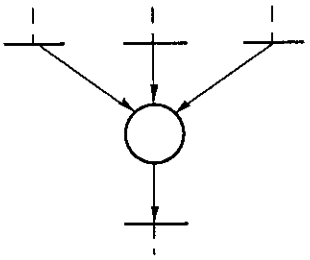


Figure 5.1: cumul

Cas 2 - Synchronisation

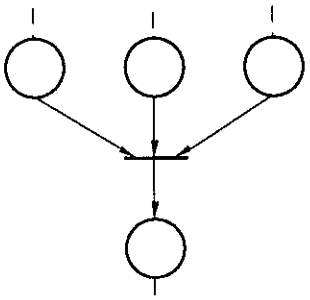


Figure 5.2: synchronisation

Cas 3 - Divergence

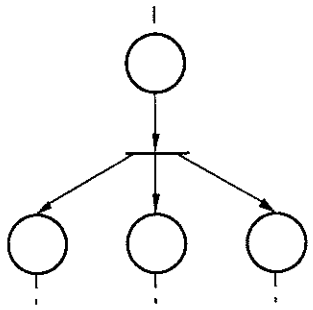


Figure 5.3: divergence

Cas 4 - Conflit

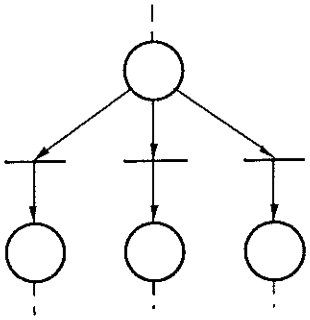


Figure 5.4: conflit

Pour les trois premiers cas les résultats se sont avérés satisfaisants, c'est-à-dire que les résultats obtenus avec le programme du RdPCA sont de bonnes approximations du modèle discret équivalent.
Par contre, pour le conflit (cas 4), le résultat obtenu n'était pas satisfaisant.

Les conflits sont alors étudiés en détail. En prenant un exemple trivial, on peut comprendre l'origine du problème afin de tenter de le résoudre.

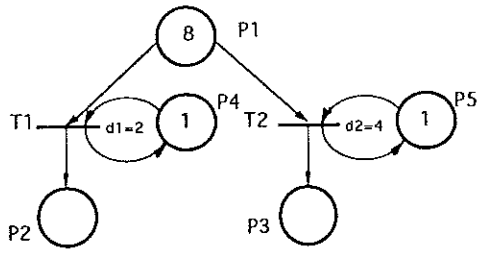


Figure 5.5: réseau avec un conflit

Dans notre exemple la transition T1 a priorité sur la transition T2.
Pour chaque place deux variables, *marquage* et *marquage disponible* sont définies. Elles ne doivent pas forcément contenir le même nombre de marques. Cependant le marquage disponible ne peut jamais être plus grand que le marquage.

Dans la place P1, de notre exemple, le nombre de marques est 8. Le marquage disponible de P1 vu de la transition T1 est aussi 8 , mais vu de la transition T2 et pour le même instant, il est égal à 7. C'est-à-dire que le marquage disponible de la place P1 n'est pas le même pour les différentes transitions en aval.

5.1 Marquage disponible

Le marquage disponible d'une place P_i vu d'une transition T_j est noté $md_{i,j}$. Pour le calculer il faut connaître l'indice de la place critique de la transition T_j. Le marquage disponible dans la place critique,$md_{placecritique,j}$ correspond au maximum de quantité qui peut franchir la transition T_j. Cette quantité,x, est réservée dans toutes les places en aval de la transition T_j. Si une de ces places à plusieurs transitions de sortie moins prioritaire que T_j, elles peuvent seulement voir:
marquage disponible dans P_i vu des transitions moins prioritaires =
marquage disponible dans P_i – quantité réservée, x

Dans notre exemple on a:

T1: marquage disponible dans P1: 8
place critique de T1: P4
marquage disponible dans P4: 1
quantité à réserver: 1

T2: marquage disponible dans P1: 8-1=7
place critique de T1: P5
marquage disponible dans P5: 1
quantité à réserver: 1

Si il y avait encore une transition après T2 le nombre de marques disponible dans cette place aurait été, 7-1=6 (en gardant le même ordre de priorité).

NOTE: pour la transition la plus prioritaire, les variables marquage et marquage disponible contiennent le même nombre des marques.

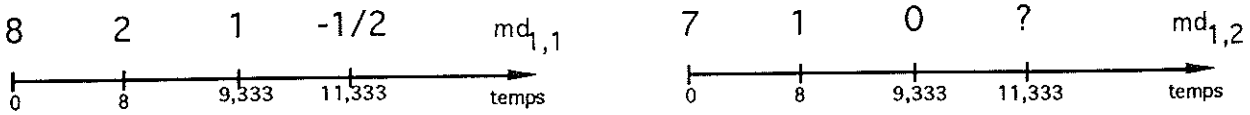
5.2 Avant changement

Le marquage disponible de la place P1 vu de la transition T1 est, selon chapitre 5.1, noté $md_{1,1}$.
Le marquage disponible de la place P1 vu de la transition T2 est noté $md_{1,2}$.
Pour la figure 5.5 on obtiendra les marquages suivants:

t = 0:
prochain événement: changement de place critique de T2 (événement du type 2).
t = 0 + 8
l'ancienne place critique a déjà atteint son marquage asymptotique
$\Longrightarrow$ recalcul des vitesses non nécessaire, voir propriété 3.3.

t = 8:
prochain événement: changement de place critique de T1 (événement du type 2).
t = 8 + 4/3
l'ancienne place critique a déjà atteint son marquage asymptotique
$\Longrightarrow$ recalcul des vitesses non nécessaire.

t = 9.3333:
prochain événement: marquage asymptotique de P1 vu de T1 (événement du type 1).
t = 9.3333 + 2
$\Longrightarrow$ recalcul des vitesses nécessaire, voir propriété 3.3.



t = 11.3333:
prochain événement: marquage asymptotique de P1 vu de T2 (événement du type 1).
t = 8 + 4
$\Longrightarrow$ recalcul des vitesses nécessaire.

t = 12: plus d'événement possible (événement du type 0).

Le nombre des marques dans la place P1, figure 5.5 est simulé est montré dans la figure 5.6:

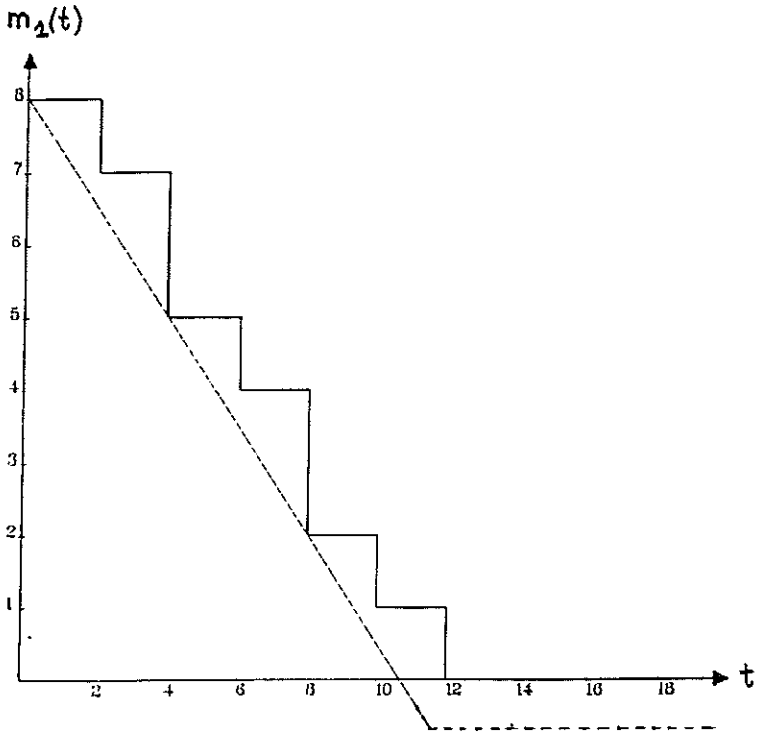


Figure 5.6:

On peut remarquer que le marquage final de la place P1 est négatif.
Ceci est inacceptable !!

Intuitivement on sait que le marquage final de cette place est nul, car elle n'est jamais alimentée. Cette faute vient du pas 2, $t = 8$, où on a calculé le prochain événement pour la transition T2. Dans le calcul on utilise la propriété 3.2, c'est-à-dire que le marquage asymptotique est atteint à la date:
$8 + d_2 = 8 + 4 = 12$.

Quand on a un conflit, cette définition n'est plus valable. Si on regarde le pas 3, on voit que déjà ici - à la date 9.3333 - quand P1 vue de la transition T1 a un changement de place critique, la place P1 vue de la transition T2 atteint son marquage asymptotique. D'après la propriété 3.3 il est donc nécessaire de recalculer des vitesses. Le programme ne calcule pas correctement cette date, il trouve seulement l'événement du type 2 et on ne recalcule donc pas de vitesses.

5.3 Changements effectués

5.3.1 Premier changement

On s'est dit que si on pouvait recalculer les vitesses au pas 3, peut-être pourrait-on avoir de meilleurs résultats. On peut vérifier cette hypothèse en forçant un nouveau calcul des vitesses pour chaque itération de l'algorithme. Cela signifie qu'on ne prend plus en compte le propriété 3.3. Pour la figure 5.5 on obtiendra:

t = 0:
prochain événement: changement de place critique de T2 (événement du type 2).
t = 0 + 8
l'ancienne place critique a déjà atteint sa marquage asymptotique
$\Longrightarrow$ recalcul des vitesses.

t = 8:
prochain événement: changement de place critique de T1 (événement du type 2).
t = 8 + 4/3
l'ancienne place critique a déjà atteint sa marquage asymptotique
$\Longrightarrow$ recalcul des vitesses.

t = 9.3333:
prochain événement: marquage asymptotique de P1 vue de T1 (événement du type 1).
t = 9.3333 + 2

$\Longrightarrow$ recalcul des vitesses.

t = 11.3333:
plus d'événements possibles (événement du type 0).

Le nombre des pièces dans la place P1, figure 5.5, est à nouveau simulé et montré dans la figure 5.7:

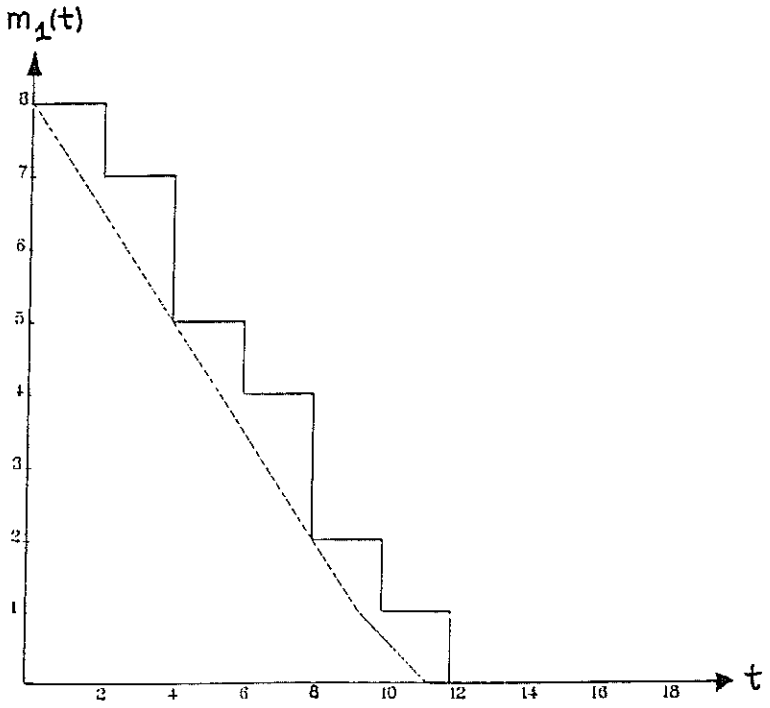


Figure 5.7:

Le marquage de la place P1 devient - comme on s'y attendait - nul. Avec cette modification du programme, on obtient de bons résultats. Mais on est obligé de faire des calculs même quand ce n'est pas nécessaire, (par exemple dans pas 2, $t = 8$, on recalcule les vitesses pour rien).

Si on pouvait trouver la bonne date où la place P1 vue de T2 atteint son marquage asymptotique on éviterait de faire des calculs inutiles.

5.3.2 Deuxième changement

Pour pouvoir calculer correctement la date de l'événement du type 1 de la place P1 vue de la transition T2, il faut connaître son marquage asymptotique,$ma_{1,2}$.

Pour une place quelconque on a:

$$ma_{i,j} = \frac{\sum V_{entree} - \sum V_{sortie} + v_j}{U_j} \tag{5.1}$$

Quand le marquage asymptotique est connu, la date peut être calculée:

$$d = \frac{ma_{i,j} - md_{i,j}}{bilan_i} \tag{5.2}$$

Avant la modification, le programme utilisait la date, qui était donnée par la propriété 3.2, puis le marquage asymptotique était calculé.

Après la modification, le programme calcule d'abord le marquage asymptotique, puis à partir du marquage trouvé il détermine la date.

Il faut bien vérifier que les formules 5.1 et 5.2 donnent des bons résultats même quand il n'y a pas un conflit.
L'équation 5.1 devient:

$$ma_{i,j} = \frac{\sum v_{entree}}{U_j} \tag{5.3}$$

et l'équation 5.2 devient alors:

$$d = \frac{\frac{\sum v_{entree}}{U_j} - md_{i,j}}{\sum v_{entree} - U_j * md_i}$$

$$= \frac{\sum v_{entree} - U_j * md_{i,j}}{U_j * (\sum v_{entree} - U_j * md_i)}$$

$$\Longrightarrow \frac{1}{U_j} \tag{5.4}$$

Le résultat obtenu par l'équation 5.4 correspond bien à celui de le propriété 3.2 utilisée dans le programme avant la modification.

Dans notre exemple, figure 5.5, on obtiendra:

t = 0:
prochain événement: changement de place critique de T2 (événement du type 2).
t = 0 + 8
l'ancienne place critique a déjà atteint son marquage asymptotique
$\Longrightarrow$ recalcul des vitesses non nécessaire.

t = 8:
prochain événement: changement de place critique de T1 (événement du type 2).
t = 8 + 4/3
l'ancienne place critique a déjà atteint son marquage asymptotique
$\Longrightarrow$ recalcul des vitesses non nécessaire.

t = 8:
prochain événement: marquage asymptotique de P1 vue de T2 (événement du type 1).
t = 8 + 4/3
$\Longrightarrow$ recalcul des vitesses nécessaire.

t = 9.3333:
prochain événement: marquage asymptotique de P1 vue de T1 (événement du type 1).
t = 9.3333 + 2
$\Longrightarrow$ recalcul des vitesses nécessaire.

t = 11.3333:
plus d'événement possible (événement du type 0).

Exemple

Prenons maintenant un exemple plus compliqué:

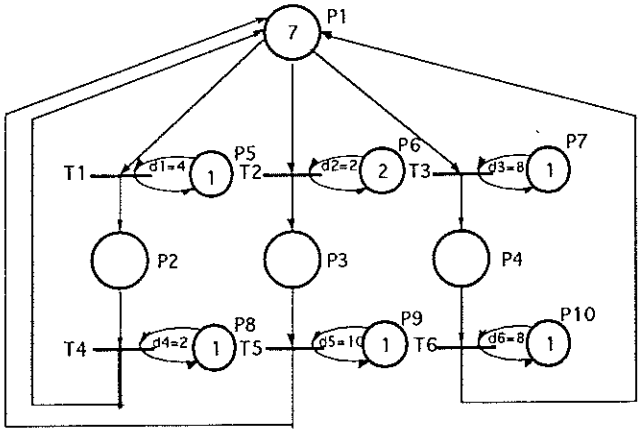


Figure 5.8:

L'évolution de la place P4 est simulé et montré dans la figure 5.9:

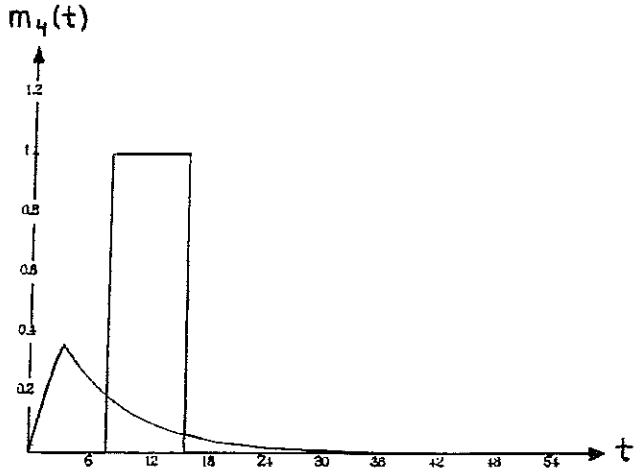


Figure 5.9:

On voit que l'algotithme de RdPCA donne des résultats qui sont des bonnes approximations du discret au sens de marquage final.

5.4 Résultats

Le fait de calculer le marquage asymptotique pour pouvoir déterminer la date du prochain événement permet à l'algorithme d'offrir des possibilités accrues de modélisation. Avant le programme n'était pas capable de calculer correctement le marquage asymptotique pour des réseaux comportant des conflits, ce problème est maintenant résolu. Ce qui aété implémenté ne modifie en aucune façon le comportement de la simulation lorsqu'une transition n'est pas élément d'un conflit.

La politique des résolution de conflits réels a toujours été de donner un ordre de priorité aux transitions. Dans la programmation les conflits sont résolus en attribuant une priorité aux transitions dont les indices sont les plus faibles.

Chapitre 6

Réseaux comportant des boucles

Pour connaître le comportement pour des boucles saturées et non saturées, des essais avec logiciel en prenant quelques exemples judicieux ont été fait. Voir chapitre 1.6.1 pour une explication de la saturation.

6.1 Boucles saturées

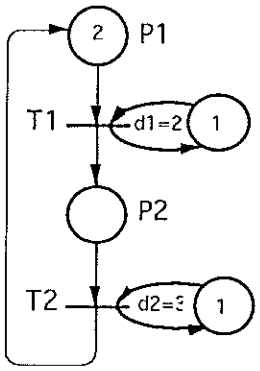


Figure 6.1: Boucle saturée

Considérons le RdPCA de la figure 6.1 et étudions son fonctionnement au cours du temps. Nous obtenons le marquage suivant:

$M(0) = (2, 0, 1, 1)$
$M(2) = (1, 1, 1, 1)$
$M(4) = (2/3, 4/3, 1, 1)$
$M(final) = (2/3, 4/3, 1, 1)$

Cette boucle est effectivement saturée car le marquage asymptotique de la place P_2 (la place associée à la transition de la durée la plus longue, $d_2 > d_1$), est supérieur à 1.

Pour les boucles saturées le programme marche bien. Avec le modèle RdPCA les marquages des places convergent vers la valeur moyenne de celui du modèle discret.

La simulation représentée par la figure 6.2 permet de montrer les evolutions des marques des places P_1 et P_2:

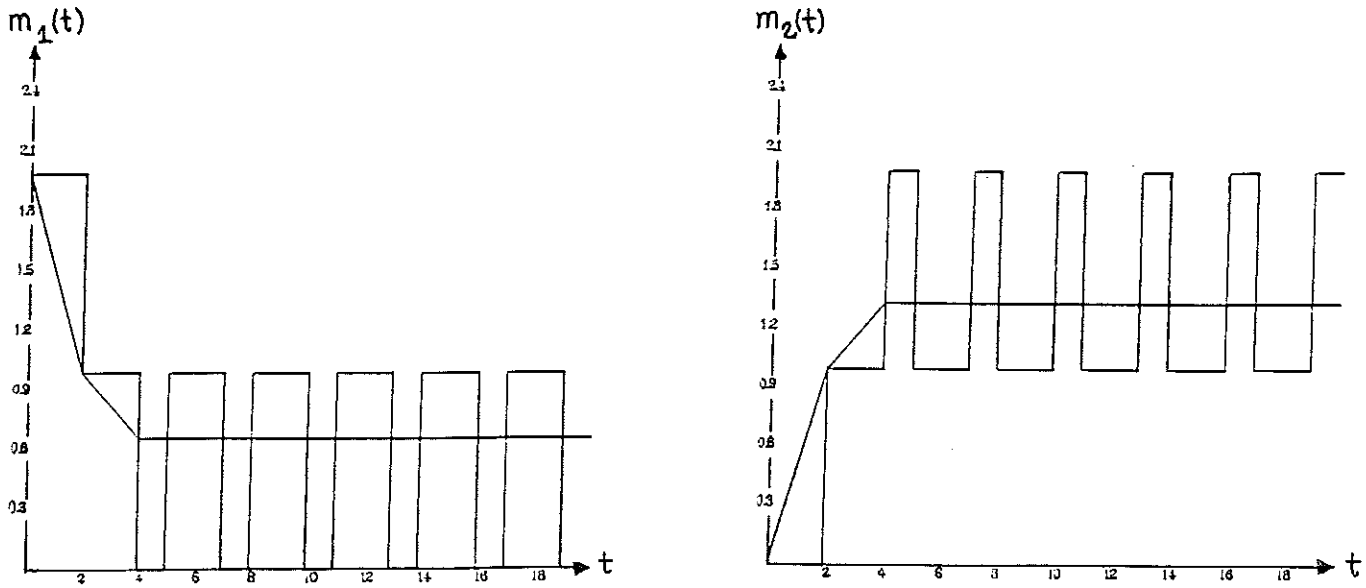


Figure 6.2:

6.2 Boucles non saturées

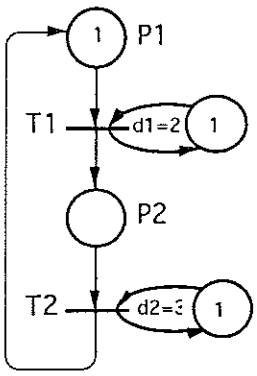


Figure 6.3: Boucle non saturée

Considérons le RdPCA de la figure 6.3 et étudions son comportement au cours du temps. Nous obtenons les marquages suivants:

$M(0) = (1, 0, 1, 1)$
$M(2) = (0, 1, 1, 1)$
$M(4) = (2/3, 1/3, 1, 1)$

$M(6) = (2/9, 7/9, 1, 1)$
$M(8) = (14/27, 13/27, 1, 1)$
$M(10) = (26/81, 55/81, 1, 1)$ ...

Cette suite de marquage converge vers les valeurs:
Place 1:

$$\underbrace{\underbrace{1+1}_{0}+2/3}_{2/3} - 4/9 + 8/27 - \ldots.$$

$2/9$

$14/27$

$\Rightarrow 1 - \sum_{n=0}^{\infty}(-2/3)^n = 1 - \frac{1}{1+\frac{2}{3}} = \frac{2}{5}$
$m_1(\infty) = \frac{2}{5}$

Place 2:
$m_1(t) + m_2(t) = 1$
$m_2(\infty) = \frac{3}{5}$

Les deux marquages asymptotiques des places P_1 et P_2
$m_1(\infty) = \frac{2}{5}$ et $m_2(\infty) = \frac{3}{5}$
correspondent bien aux marquages moyens du modèle discret.
Le comportement des marquages des places P_1 et P_2 est illustré par la figure 6.4:

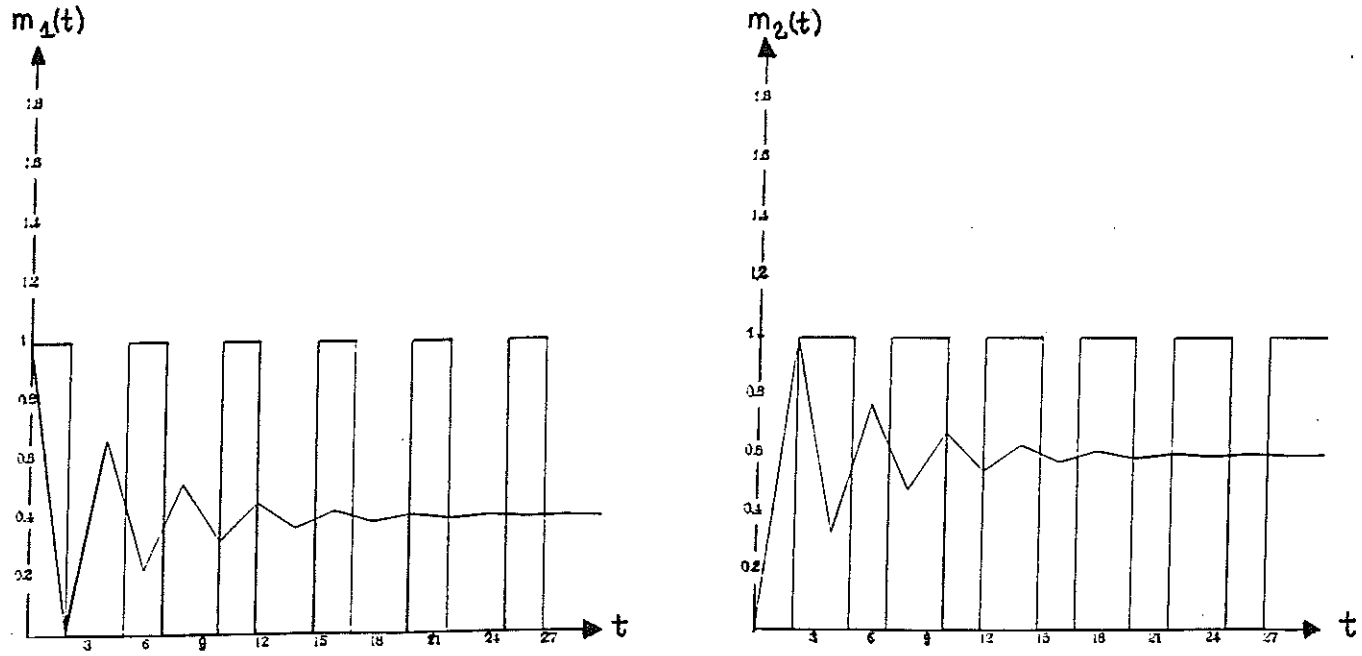


Figure 6.4:

Cette boucle est non saturée car le marquage asymptotique de la place P_2 est inférieur à 1.
Pour les boucles non saturées deux inconvénients du programme ont été constatés:

1. Il converge vers la bonne valeur moyenne du modèle discret mais en temps infini.
2. Il n'a pas une période qui correspond à celle du modèle discret.

Un comportement plus souhaitable serait le suivant, figure 6.5:

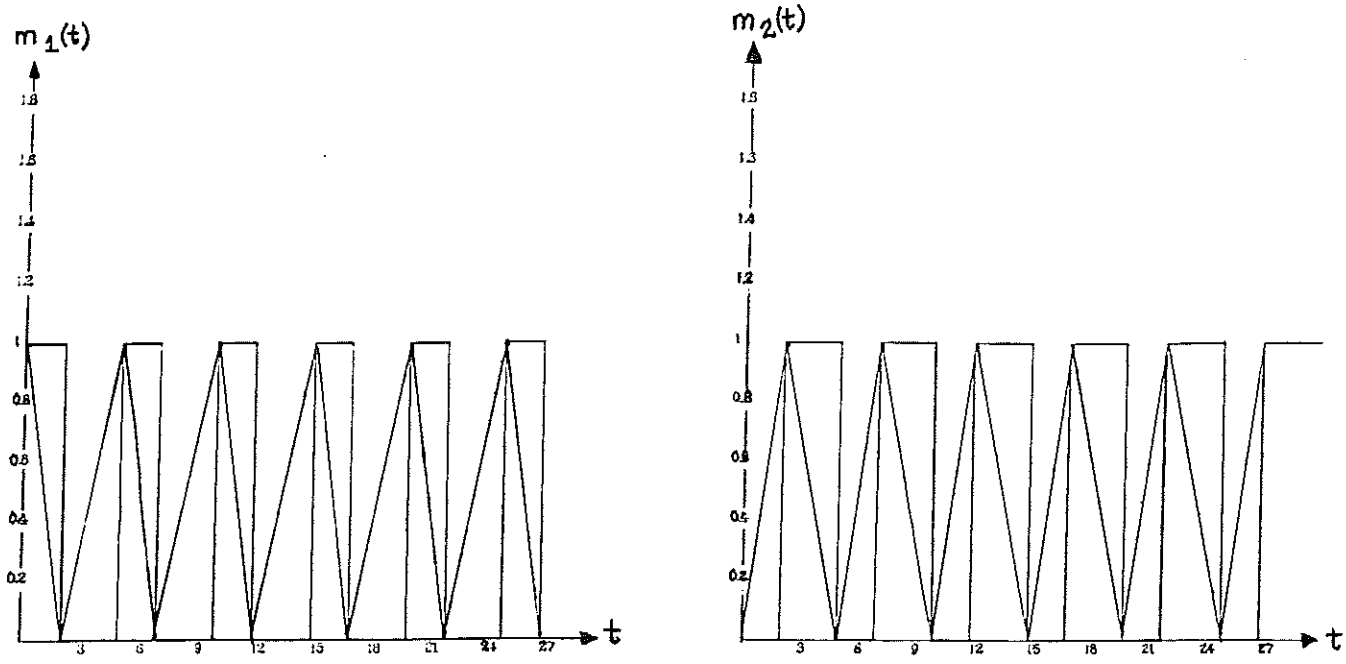


Figure 6.5:

6.3 Changements

Avec le comportement donné par la figure ci-dessus, figure 6.5, les marquages ne convergent plus vers une valeur fixe. En revanche, la période correspond bien à celle du modèle discret. Pour obtenir le comportement souhaitable présenté ci-dessus, il est nécessaire d'opérer des modifications sur la partie "Determination des transitions a calculer" (Partie 6 du programme).

6.3.1 Avant changement

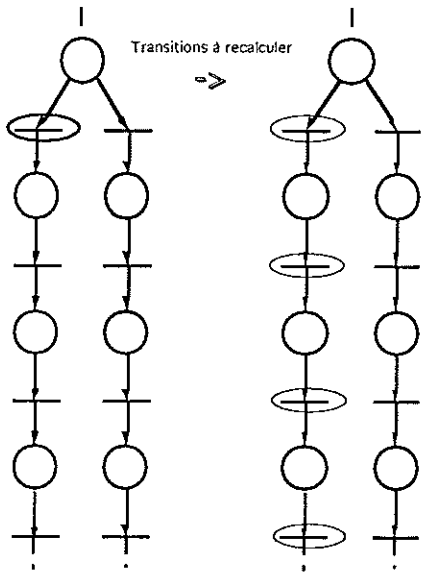


Figure 6.6: Transitions à recalculer avec l'algorithme de Gauss-Seidel

Avant modification, le programme utilisait l'algorithme de Gauss- Seidel pour trouver quelles étaient les transitions à calculer. Cette algorithme est détaillé dans la partie 6 du programme dans chapitre 4.

6.3.2 Après changement

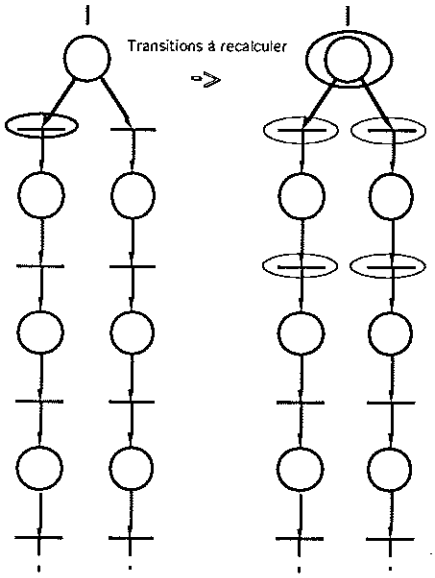


Figure 6.7: Transitions à recalculer avec le nouvel algorithme

Après les modifications le programme utilise une autre algorithme pour trouver les transitions à calculer. Cette algorithme ne part plus de la transition concernée mais de la

place. A partir de cette place seulement les transitions dans les deux pas suivants sont à calculer. Avec ce changement on retarde la propagation du flot et on espère avoir un fonctionnement oscillatoire qui suit bien le comportement du modèle discret.

Remarque:

Pour un réseau avec seulement deux transitions les deux algorithmes sont equivalents mais dès qu'il y a plus de deux transitions on a un changement de comportement.

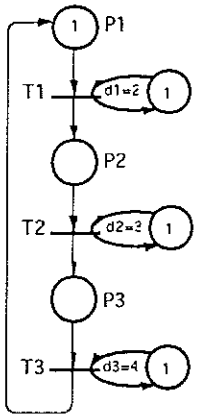


Figure 6.8: Boucle non saturée

Le comportement de la place P_1 de la figure 6.8 avant et après modifications est illustré par la figure 6.9:

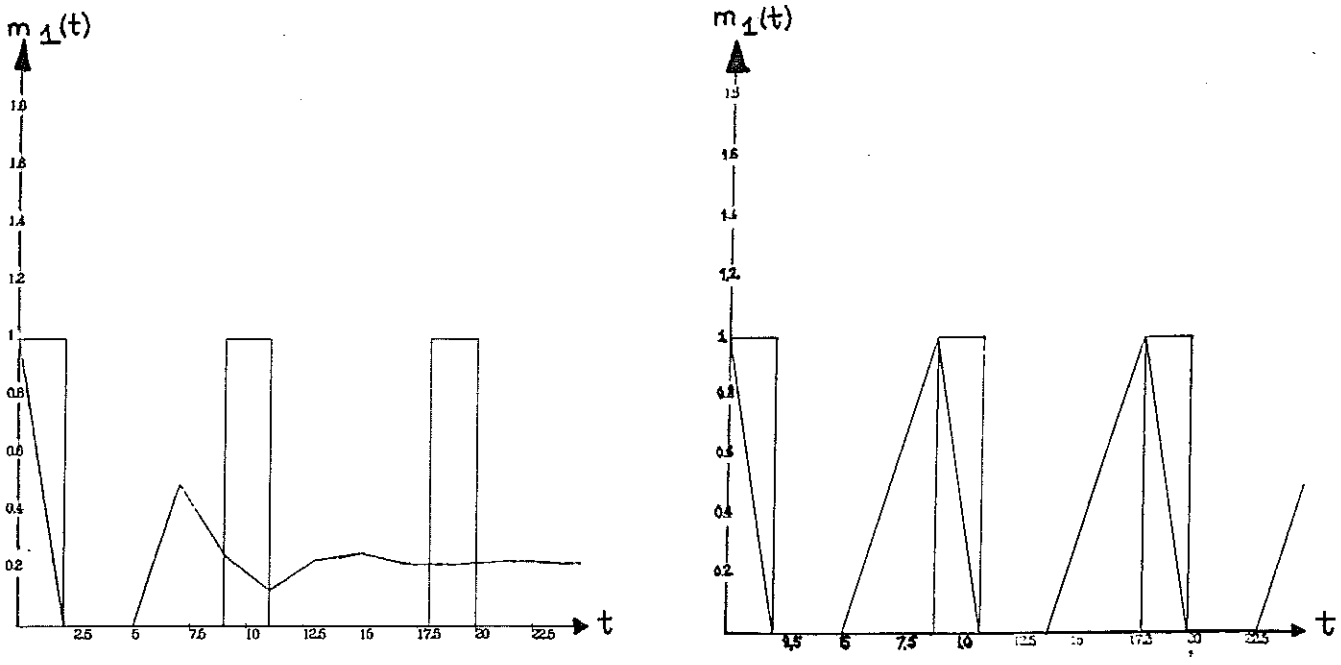


Figure 6.9: a. Avant modification b. Après modification

6.4 Vérification

Est-ce-que ce changement marche dans tous les cas? Obtient-on toujours une bonne approximation RdPCA-Discret?
Pour pouvoir répondre à ces questions divers d'essais avec des réseaux différents ont été fait. Des avantages mais aussi des inconvénients ont été trouvé.

6.4.1 Avantages

Pour des boucles saturées on peut seulement constater que ce changement d'algorithme ne peut qu'améliorer le comportement. Les marquages de places du modèle RdPCA convergent toujours vers la bonne moyenne mais le programme fait moins de calculs pour les trouver.

Dans la figure 6.10, le comportement de la figure 5.8 place P4 avant et aprés le changement l'algorithme, est illustré:

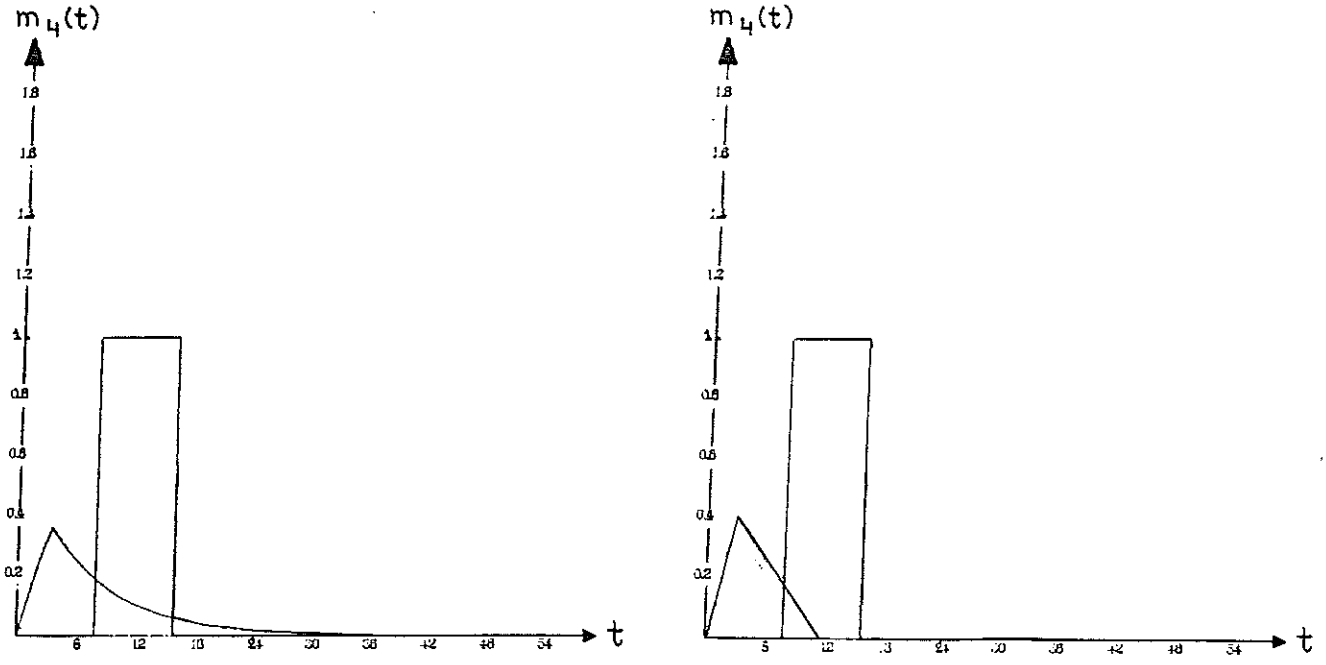
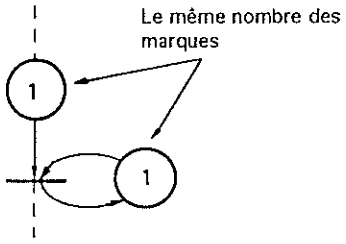


Figure 6.10: a. Avant modification b. Après modification

6.4.2 Inconvénients

Pour les boucles non saturées plusieurs cas où l'algorithme ne correspond plus bien au comportement discret ont été trouvé. Dans tous les cas où le marquage d'une place n'est pas égal à celui de sa place limitée on n'obtiendra pas un bon comportement. C'est-à-dire qu'il marche seulement dans un cas très particulier donné par la figure suivante:



Dans tous les autres cas ça ne marche pas. Un comportement qui ne correspond pas au comportement souhaité est ontenu. Pour quelques essais (réseaux contenant des grand boucles), le comportement ne correspond ni à la valeur moyenne ni à la péroide.

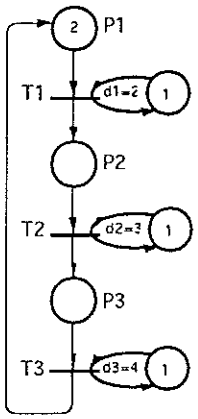


Figure 6.11: Boucle non saturée

Dans la figure 6.12, la comportement de la figure 6.11 place P_1 avant et aprés le changement l'algorithme, est illustré:

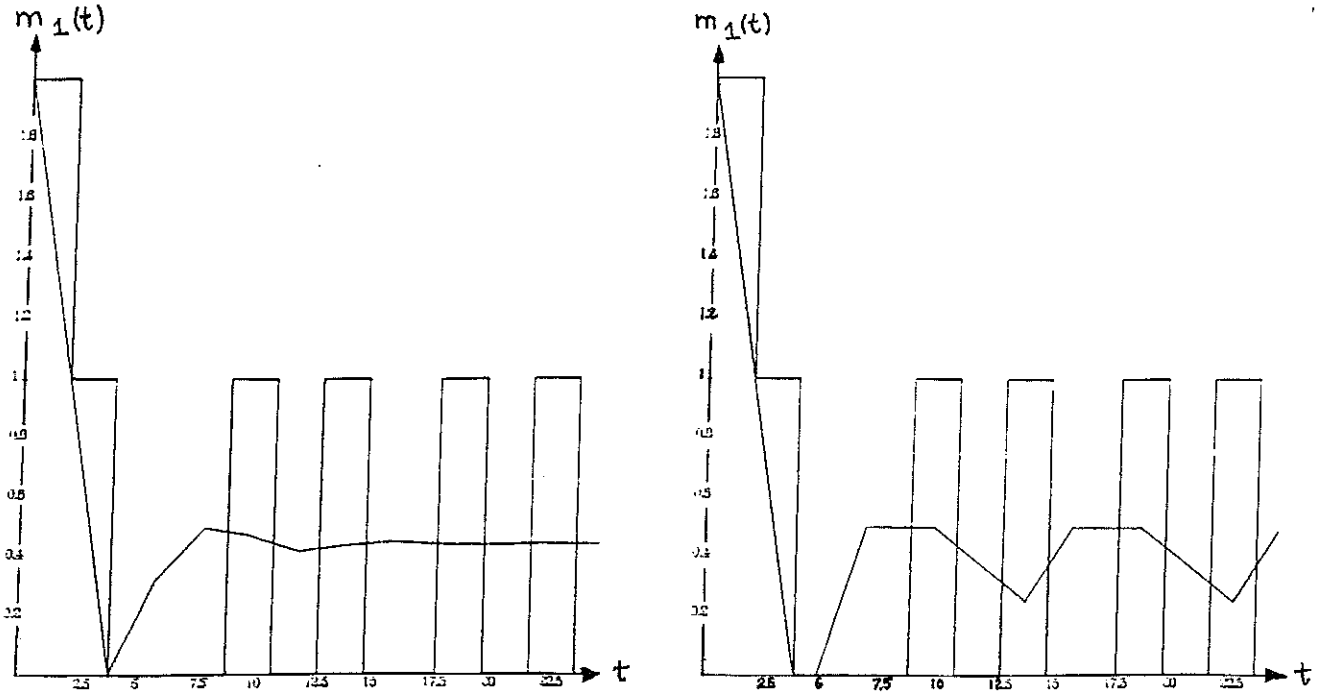


Figure 6.12: a. Avant modification b. Après modification

6.5 Conclusion

L'idée était de trouver une manière de modifier le programme pour les boucles non saturées. On a eu une idée et avec celle-ci beaucoup d'essais ont été fait. Ce changement l'améliore dans un seul cas non saturé et dans tous les cas saturés. Dans les cas non saturés où l'algorithme n'améliore pas, le nouveau comportement ne correspond plus à celui souhaité et dans certain cas il est pire que celui qu'on avait avant de modifier le programme.
L'inconvénient du nouvel algorithme est plus important que ses avantages et pour l'instant, il est nécessaire de revenir à l'algorithme du programme original (Gauss-Seidel). Cela étant, ce problème reste ouvert et des recherches futures pourront être menées pour se résolution.

Chapitre 7

Conclusion

Dans ce mémoire, le modèle réseau de Petri continu asymptotique a été étudié. Ce modèle constitue la troisième approche pour le calcul des vitesses instantanées de franchissement des transitions. Il constitue une très bonne approximation des systèmes à événements discrets (même dans le cas de faible marquage) et il permet des simulations très rapides. Ce modèle garde la simplicité du RdPCC et la précision du RdPCV. Dans le rapport j'ai regardé ses propriétés, ses avantages et ses inconvénients; puis j'ai essayé de les améliorer.

Un programme du RdPCA existait déjà et avait éte impanté par Jean Le Bail. Mon travail a commencé par vérifier celui-ci. Beacoup de simulations des réseaux simples ont été effectuées et les résultats comparés avec ceux donnés par le modèle discret correspondant et aussi par des calculs à la main. Pour ces réseaux le programme n'avait pas des défauts. Après cette étude préliminaire du fonctionnement du programme, d'autres structures plus compliquées ont été étudiées.

D'abord, quelques structures particulières ont été etudiées: cumul, synchronisation, divergence et conflit. Les comportements pour les trois premières structures correspondaient bien à celles du modèle discret. Pour le conflit on a eu des résultats qui n'étaient pas acceptables. Le marquage final devenait négatif, ce qui n'était pas tolérable pour un RdP. Les modifications apportées ont permis à l'algorithme de trouver de bons résultats. Avant il utilisait la date donnée en propriété 3.2 pour pouvoir calculer le marquage asymptotique. Maintenant après les changements il cherche d'abord le marquage asymptotique et puis il l'utilise pour pouvoir calculer la date de cet événement, equations 5.1 et 5.2. Le programme a été changé des cette manière et offre maintenant la possibilité d'introduire au sein du RdPCA des conflits. Les changements implémentés ne modifient en aucun façon le comportement lorsqu'une transition n'est pas élément d'un conflit. Les conflit sont tous résolus par priorité.

Ensuite, j'ai étudié les réseaux comportant des boucles. Pour les boucles saturéés l'algorithme a un comportement correct. Pour les boucles non saturées le comportement n'était pas satisfaisant, il comportait plusieurs inconvénients. Une idée d'un comportement souhaitable été proposée. Le fait de changer l'algorithme des recalcul des vitesses réalisait dans un premier temps, ce comportement souhaitable et en plus il améliorait le comportement pour les boucles saturées. Dès que plusieurs essais ont été faits, plusieurs cas où le nouveau comportement ne correspondait pas au celui souhaité ont été trouvés. Pour certaines boucles, le nouveau comportement était pire que l'ancien, et il a alors été

nécessaire de reprendre l'algorithme comme il était dans le programme origine.
Voir le programme original avec des modifications dans l'appendice.

Chapitre 8

Resumé

Petri Nets (PNs) are a well known tool for modelling and analyzing discrete event systems. When solving problems in the industry the user is often confronted with a large number of states, and any small expansion of the process may result in an large increase of states. To avoid this inconvenience, continous Petri nets were introduced [David 87].

To start with, two timed continuous Petri nets were presented by the group Sylodi[1] at the Laboratoire d'Automatique de Grenoble; *Constant speeds Continuous Petri Nets (CCPN)* and *Variable speeds Continuous Petri Nets (VCPN).* CCPN have the advantage of a short simulation time but the results sometimes fail to follow those given by the discrete model.On the other hand, VCPN have opposed caracteristics ; the results are closer to those given by the discrete model but the simulation time can sometimes be long.

Recently a new model has been developed by the same group (Sylodi) ; *Asymtotic Continuous Petri Nets (ACPN)* [LeBail 92]. This model combines the simplicity of the CCPN and the precision of the VCPN.

In this report the major properties of the ACPN are presented. The original program written by Jean LeBail is discused, some of its deficiencies looked upon and improvements added. All simulations were based on the corrected program and the results compared with corresponding simulations done on the discrete model and handcalculations. After the initial investigation into the functionallity of the program, more complex structures were studied.

First four special structures were studied: Accumulation, Syncronisation, Divergence and Conflicts. The first three structures proved to be good approximations of the corresponding discrete models. However, the results for the conflict situations were not acceptable. The final marking became negative, something that is imposible for PNs. Through changes in the original program, i.e. first finding the asymptotic marking and then using it to calculate the date of the event, instead of using the date given by a certain property of the ACPN (not counting for the ability to handle conflicts) to find the asymptotic marking, the program reaches the correct values even in conflict situations. These changes have been added to the program which now offers the possibility to introduce conflicts to the ACPN. The changes does not effect, in any way, the transitions that are not part of a conflict. The conflicts are all solved by priority.

[1]Sylodi = SYstèmes LOgiques et DIscrets

Secondly, loop structures have been studied.Two types are defined: saturated and none saturated loops. Saturated loop structures were handled correctly by the original program. For non saturated loops, the behavior was not satisfying and ideas of how to improve this was suggested. By changing the speed recalculation algorithm, it first seemed as if the desired behavior was reached. More simulations on different cases showed that this was not always true. The behavior of the new algorithm was sometimes worse than that of the old one. Therefore, though not perfect, the original algorithm is still implemented and used in the final program.

Bibliographie

[Petri 62] C.A. PETRI,*Kommunikation mit Automaten, Schriften des Rheinisch,* Westfalischen Institutes fur Instrumentalle Mathematik and der Universitat Bonn, 1962, traduit par C.F. Greene, Applied Data Research Inc., Suppl., 1 to tech. Report RADC-TR-65-337, N.Y.,1965.

[David 87] R.DAVID, H.ALLA,*Continuous Petri Nets*, 8ème Conférence Européenne sur l'application et la Théorie de réseaux de Petri, Sagarosse(E), Juin 1987, pp.275-294.

[David 89] R.DAVID, H.ALLA,*Du Grafcet aux réseaux de Petri*, Edition Hermès, Paris, 1989.

[David 90] R.DAVID, H.ALLA,*Autonomous and Timed Continuous Petri Nets*,11ème Conférence Internationale sur l'application et la Théorie de réseaux de Petri, Paris Juin 1990, pp.367-386.

[Dubois 82] D.DUBOIS, J.-P.FORESTIER,*Productivité et en-cours moyens d'un ensemble de deux machines séparées par un stock*, Revue RAIRO Automatique, vol.16, nr2, 1982, pp.105-132.

[LeBail 92] J.LEBAIL, H.ALLA,R.DAVID,*Asymptotic Continuous Petri Nets : An efficient Approximation of Discret Event Systems*, IEEE International Conference on Robotics and Automation, Nice, Mai 1992, pp.1050-1056.

Chapitre 9

Annexes

LISTINGS

```
/*-------------------------------------------------------------------------------------
 -                                                                                    -
 -                              FICHIER SOURCE                                        -
 -                                                                                    -
 - Date       : 1993                                                                  -
 - fichier    : rdpca_conflit.c                                                       -
 - fonctions  : RdPCA()                                                               -
 - Description : Algorihme de simulation des RdP continus asymptotiques              -
 -               Approximation du RdPCV                                               -
 -               Avec introduction de la resolution des conflits par priorite. (Charlotta) -
 -                                                                                    -
 -                                                                                    -
 -------------------------------------------------------------------------------------*/



#include "ext_var_glo.h"



extern void init_courbe();
extern void fin_courbe();
extern void affiche_courbe();
extern void init_simu();
extern void info();
extern void erreur();
extern double pre();
extern double post();

void rend_disponible();
void reserve();



/* ------- struct de donnees des transitions pour calcul asymptotique -------*/
struct TXD
        {
         struct Arc *Ax;       /* arc critique */
         double D;             /* date de l'evt relatif a T */
         int Cal;             /* indique si la transition doit etre recalculee (1) ou non (0) */
        } *Ta[NMAXT], *pta ;
```

```
/* ------------------------------------------------------------------------
-                                Fonction rdpca()                                -
------------------------------------------------------------------------*/

RdPCA(graphe,courbe)
 int graphe;
 int courbe[NMAXCO];
 /*
  graphe : determine si on veut tracer des courbes (1) ou non (0)
  courbe : tableau comprenant les courbes (determinee par une place) a dessiner
 */

 {
  int i, j, simu, lim, calcul, ix, Ncal;
  int evt, je, ie;
  double mx, dcal, mcal, mmin, d, ma, time;
  double delta;
  FILE *f,*fopen();
  struct Marquage mr, mc, mt;
  struct Arc *pa, *pai, *Ae , *pcal;
  struct
         {
          int T;                /* indice de transition                          */
         } Tc[NMAXT];           /* tableau pour determiner les transitions a calculer */
  Gwpoint *point_courbe[2];



  /*  ouverture du fichier de compte-rendu*/
  f=fopen("res_graphe.txt","w");


  /*  initialise le graphisme, le fichier de sortie et le marquage courant */
  if (graphe)
         {
          point_courbe[0]=(Gwpoint *)malloc(NMAXCO*sizeof(Gwpoint));
          point_courbe[1]=(Gwpoint *)malloc(NMAXCO*sizeof(Gwpoint));
          if (point_courbe==NULL) exit(2);
          init_courbe(courbe,point_courbe);
         }

  /* affecte les marquages courants aux marquages initiaux */
  init_simu();


  /*      allocation de memoire pour Ta   */
  Ta[0] = (struct TXD *)malloc(NT*sizeof(struct TXD));

  if (Ta[0] != NULL)
         {
          /* l'allocation memoire a reussi ! */
```

```
        /* initialisation de Ta          */
        /* T1 est a calculer */
        Ta[0]->Cal = 1;
        /* arc critique inconnu */
        Ta[0]->Ax = NULL;

for ( j=1 ; j<NT ; j++)
        {
      Ta[j] = Ta[j-1] + 1 ;
        /* toutes les transitions sont a calculer */
      Ta[j]->Cal = 1;
        /* toutes les arcs critiques sont inconnues */
      Ta[j]->Ax = NULL;
      }

/* initialisation des variables    */

/* instant initial de la date actuelle*/
temps = 0.;

/* la simulation n'est pas finie */
simu = 1 ;

/* ATTENTION LE RDPCA EST EN EXPLICITE */
implicite = 0;
info ("Attention : le modele RdPCA utilise la notation explicite");

/* Il est necessaire de recalculer les transitions */
calcul = 1;


/*    ecrit les marquages initiaux dans le fichier  */
fprintf (f,"\n");
fprintf (f,"\n");
fprintf (f,"le marquage initial:\n");
/* pour toutes les places */
for (i=0;i<NP;i++)
        {
        /* ecrit la place et le bilan */
        fprintf(f,"m%d=%lg", i+1, (P+i)->m);
        fprintf(f,"\n");
        }
fprintf(f,"\n");
fprintf(f,"\n");


/*    Debut de la simulation    */
while (simu)
        {
      if (calcul)
                {
                /* calcul des transition a faire */

                /* marquage disponible = marquage courant */
                rend_disponible();
```

```
/* initialisation de place critique, date du prochain evenement et vitesse
   instantanee */
/* pour chaque transition */
for (j=0;j<NT;j++)
{
        /* si le calcul est necessaire */
mr.m = 1.;
if ( Ta[j]->Cal )
        {
        lim = implicite;
        mmin = 1. ;
        if (Ta[j]->Ax == NULL)
                {
                        /* place critique inconnue */
                        /* pour chaque arc d'entree de Tj */
                for ( pa=(T+j)->pre ; pa->i ; pa++ )
                        {
                        (*(pa->fct->f))(&mr,&mc,&(pa->p));
                                /* mc image par la fct */
                        mcal = (P+pa->i-1)->md / mc.m;
                        if ( (!lim) || (mmin > mcal) )
                                {
                                lim = 1;
                                mmin = mcal;
                                        /* calcule nombre de jetons/poids =
                                           nombre de lots */
                                Ta[j]->Ax = pa;
                                }
                        }
                }
        else
                {
                        /* Sinon la place critique est connue */
                pa = Ta[j]->Ax;
                (*(pa->fct->f))(&mr,&mc,&(pa->p));
                        /* nombre de lots */
                mmin = (P+pa->i-1)->md / mc.m;
                }

        /**** initialisation de la vitesse et de l'evt suivant pour Tj ****/
                /* vitesse instantanee */
        (T+j)->v = mmin/(T+j)->d;
                /* date de l'evt suivant du type 1 */
        Ta[j]->D = temps + (T+j)->d;

        mr.m = mmin ;
        if (mr.m > 0.)
                        {
                        /* reservation des marques */
                reserve(j,&mr);
                        }
        }
        }
```

```
            /*   calcul des bilans des places   */
            /* pour toutes les places on iniotialise le bilan a 0*/
            for (i=0;i<NP;i++)
           (P+i)->b=0.;
            /* pour chaque transition */
            for (j=0;j<NT;j++)
            {
            mr.m = (T+j)->v;
            if (mr.m > 0.)
                    {
                            /* la vitesse instantanee est non nulle */
                            /* pour chaque arc d'entree de Tj       */
                    for ( pa=(T+j)->pre ; pa->i ; pa++ )
                            {
                                    /* mc image de Vj par la fct */
                                    (*(pa->fct->f))(&mr,&mc,&(pa->p));
                            (P+pa->i-1)->b -= mc.m;
                            }
                             /* pour chaque arc de sortie de Tj */
                    for ( pa=(T+j)->post ; pa->i ; pa++ )
                            {
                                    /* mc = image par la fct */
                            (*(pa->fct->f))(&mr,&mc,&(pa->p));
                            (P+pa->i-1)->b += mc.m;
                            }
                    }
            }
            /* Pour toutes les places on arrondi le bilan */
            for (i=0;i<NP;i++)
          if (fabs((P+i)->b) < 1.0e-10 )
             (P+i)->b=0.;
            }



   /*   Recherche de l'evenement suivant  */
   /* evenement initialise d'aucun type *
evt = 0 ;

   /* pas de transition responsable */
je = 0 ;

   /* marquage disponible = marquage courant */
rend_disponible();

for (j=0;j<NT;j++)
            {
            mr.m = 1.;
            /* indice de la place critique */
            ix = Ta[j]->Ax->i;
            if (Ta[j]->D > 0.)
            {
                    /* marquage asymptotique pas encore atteint */
            if (fabs((P+ix-1)->b) > 1.0e-10 )
                    {
```

```
                        /* Si le bilan non nul */

                        /* evt marquage -> marquage asymptotique  */
                        /*   Recalcul le temps de l'evt du type 1 */
                ma = (pre(ix, RDPCA) - post(ix, RDPCA) + (T+j)->v) * (T+j)->d;
                        /* le m.asy.ne peut pas etre negatif */
                if (ma < 0.0)
                                ma = 0.0;
                        /* calcul la date relatif du m.asy. */
                        time = ((P+ix-1)->md - ma)/(P+ix-1)->b;
                        /* calcul de la date du marquage asymptotique */
                        if (time < 0 )
                        Ta[j]->D = temps - time;

                if ( (!evt) || (d > Ta[j]->D) )
                        {
                                /* date du prochain evenement */
                        d = Ta[j]->D;
                        Ae = Ta[j]->Ax;
                                /* indice de la transition */
                        je = j+1;
                                /* evenement de type 1 */
                        evt = 1;
                        }
                }
        else
                        {
                        /* bilan nul, pas d'evt futur pour cette transition */
                 Ta[j]->D = -1.;
                        }
        }
        (*(Ta[j]->Ax->fct->f))(&mr,&mc,&(Ta[j]->Ax->p));
        /* mx contient vitesse critique*/
        mx=mc.m ;

/* pour chaque place de .Tj sauf le p.critique regarde si le bilan < bilan de p.critique */
        /* pour chaque arc d'entree de Tj */
 for ( pa=(T+j)->pre ; pa->i ; pa++ )
        if (pa != Ta[j]->Ax )
                {
                        /* Si non arc critique */
                        /* mc = image de Vj par la fct */
                 (*(pa->fct->f))(&mr,&mc,&(pa->p));
                        /* compare les bilans normes */
                        dcal = (P+ix-1)->b*mc.m - (P+pa->i-1)->b*mx ;
                        if (dcal > 0.)
                        {
                                /* le bilan est inferieur au bilan critique */
                                /* calcule la date relative */
                        dcal = ((P+pa->i-1)->md*mx - (P+ix-1)->md*mc.m)/dcal;
                        if ( (!evt) || (d > temps+dcal ) )
                                {
                                        /* date du prochain evenement */
                                d = temps + dcal;
                                        /* arc du prochain evenement */
```

```
                                    Ae = pa;
                                        /* indice de la transition */
                                    je = j+1;
                                        /* evenement de type 2 */
                                    evt = 2;
                                    }
                                }
                                }
                mr.m = (T+j)->v*(T+j)->d;

                /* reservation des marques */
                reserve(j,&mr);
                }



        /**** ecrit des resultat obtenue dans le fichier ****/
        fprintf (f,"date de l'ancien evt=%lg\n", temps);
        fprintf (f,"date de l'evt=%lg, type d'evt=%d, transition=%d\n", d, evt, je);


        /**** ecrit les transitions et leur vitesse instantanee a l'ecran ****/
        fprintf (f,"la vitesse instantanee pour chaque transition:\n");
        for (j=0;j<NT;j++)
                {
                /* ecrit la transition et la vitesse instantanee */
                fprintf(f,"v%d=%lg, ", j+1, (T+j)->v );
                fprintf(f,"\n");
                }


        /**** ecrit les places et leur bilan dans le fichier ****/
        fprintf (f,"le bilan pour chaque place:\n");
        for (i=0;i<NP;i++)
                {
                /* ecrit la place et le bilan */
                fprintf(f,"b%d=%lg", i+1, (P+i)->b);
                fprintf(f,"\n");
                }


        /**** regarde si la simulation est finie ou non ****/
if (d > tmax || evt ==0)
                {
                simu=0;
                evt=0;
                d=tmax;
                }

        /**** nouveau temps ****/
delta = d - temps;
temps = d;
```

```
/**** Calcul du nouveau marquage a la date temps ****/
for (i=0 ; i<NP ; i++)
        (P+i)->m += delta*(P+i)->b;


   /**** ecrit les nouveaux marquages dans le fichier ****/
   fprintf (f,"le nouveau marquage:\n");
   for (i=0;i<NP;i++)
        {
         /* ecrit la place et le marquage */
         fprintf(f,"m%d=%lg", i+1, (P+i)->m);
         fprintf(f,"\n");
        }
   fprintf(f,"\n");
   fprintf(f,"\n");


/**** affichage des courbes ****/
if (graphe)  affiche_courbe(temps,courbe,point_courbe,AFFICHAGE_CONTINU);


if (evt) pta = Ta[je-1];
   if (evt == 1)
        {
        /* evenement du type 1 */
        /* evt marquage -> marquage asymptotique */
        /* necessaire de recalculer certaines vitesses */
        calcul = 1;
        /* pas d'evt futur pour cette transition */
        pta->D = -1.;
        }
   else
        if ( evt == 2 )
        {
                /* evenement du type 2 */
                /* evt marquage -> marquage de la place critique*/
                /* changement de l'arc critique */
         pta->Ax = Ae;
         if (pta->D < 0.)
                {
                        /* l'ancienne p.critique a deja atteint son m.asy. */
                        /* date du prochain evenement */
                 pta->D = temps + (T+je-1)->d;
                        /* pas necessaire de recalculer des vitesses */
                 calcul = 0;
                }
         else
                {
                        /* l'ancienne p.critique n'a pas encore atteint m.asy */
                        /* date du prochain evenement */
                 pta->D = temps + (T+je-1)->d;
                        /* necessaire de recalculer certaines vitesses */
                 calcul = 1;
                }
         }
```

```
                else
                        {
                        /* Pas de recalul necessaire */
                calcul = 0;
                        }


        /**** determination des transitions a calculer ****/
        if (calcul)
                {
                for (j=0;j<NT;j++)
                Ta[j]->Cal = 0;
                Ta[je-1]->Cal = 1 ;
                ie = (T+je-1)->pre->i;
                for (pcal=(P+ie-1)->post ; pcal->i ; pcal++)
                {
                        /* s'il y a un conflit */
                je=pcal->i;
                Ta[je-1]->Cal=1 ;
                Ncal = 1;
                Tc[Ncal].T = je-1;
                while (Ncal)
                                {
                                j = Tc[Ncal].T;
                                /* pour chaque arc de sortie de Tj */
                                for (pa=(T+j)->post ; pa->i ; pa++)
                                {
                                        /* pour chaque arc de sortie de Pj */
                                for (pai=(P+pa->i-1)->post;pai->i;pai++)
                                        {
                                        pta = Ta[pai->i-1];
                                        if (pta->Cal == 0)
                                        {
                                        Ta[pai->i-1]->Cal = 1;
                                        Tc[Ncal].T = pai->i-1;
                                        Ncal++;
                                        }
                                        }
                                }
                                Ncal--;
                                }
                }
           }
        }


/**** ecrit les marquages finals dans le fichier ****/
fprintf (f,"le marquage final:\n");
/* pour toutes les places */
for (i=0;i<NP;i++)
        {
        /* ecrit la place et le marquage */
        fprintf(f,"m%d=%lg", i+1, (P+i)->m);
        fprintf(f,"\n");
        }
```

```
  fprintf(f,"\n");
  fprintf(f,"\n");


  /**** fermeture du fichier ****/
  fclose (f);


  /**** termine le graphisme ****/
  if (graphe)
       {
       fin_courbe(courbe,point_courbe);
       free(point_courbe);
       }
  }
}
```

```
/* ----------------------------------------------------------------------
  -                        Fonction rend_disponible()                    -
  ---------------------------------------------------------------------- */
void rend_disponible()
/*
  cette fonction rend toutes les marques disponibles
*/
{
int i;

/* pour toutes les places */
for (i=0;i<NP;i++)
          {
          /* md = m */
          (P+i)->md = (P+i)->m;
          }
}
```

```
/* ----------------------------------------------------------------------
  -                        Fonction reserve()                            -
  ---------------------------------------------------------------------- */
void reserve(j, pr)
int j;
struct Marquage *pr;
/*
  cette fonction reserve de marques
*/
```

```
{
struct Marquage mc;
struct Arc *pa;

/* reservervation de pr marques pour Tj */
/* pour chaque arc d'entree de Tj */
for ( pa=(T+j)->pre ; pa->i ; pa++ )
        {
        (*(pa->fct->f))(pr,&mc,&(pa->p));
        /* mc = image par la fct */
        (P+pa->i-1)->md -= mc.m;
        }
}
```

```
/*-------------------------------------------------------------------------------------
 -                                                                                    -
 -                              FICHIER SOURCE                                        -
 -                                                                                    -
 - Date       : 27 / 05 / 93                                                          -
 - fichier    : post.c                                                                -
 - fonctions  : post()                                                                -
 - Description : Retourne le flot sortant la place continue dont l'indice est passe  -
 -                 en parametre.                                                      -
 -                                                                                    -
 -                                                                                    -
 -------------------------------------------------------------------------------------*/


#include "ext_var_glo.h"


double post( i , type_algo )
int i;
int type_algo;
/*
  i        : indice de la place dont le flot de sortie est a calculer
  type_algo : nom de l'algorithme necessitant l'apel de cette fonction
*/

{
 double flot;
 int k;
 struct Arc *pa;


 flot = 0;



/* Determination de tous les transitions en aval de Pi */
 for (pa = (P + i - 1)->post ; pa->i ; pa++)
        {
         /* k est l'indice de la transition */
         k = pa-> i;
         if ((!(T + k -1)->discret) || ALGORITHME_CONTINU)
                {
                 /* Pour le RdPCC toutes les transitions sont considerees continues */
                 flot += (T + k - 1)->v * pa->p;
                }
        }

/* flot contient le flot de sortie de pi */
return(flot);

}
```

```
/*-------------------------------------------------------------------------------------------
-                                                                                           -
-                                   FICHIER SOURCE                                          -
-                                                                                           -
- Date        : 17 / 05 / 93                                                               -
- fichier     : pre.c                                                                      -
- fonctions   : pre()                                                                      -
- Description : Retourne le flot alimentant la place continue dont l'indice est passe      -
-                en parametre.                                                              -
-                                                                                           -
-                                                                                           -
-------------------------------------------------------------------------------------------*/


#include "ext_var_glo.h"



double pre( i, type_algo )
int i;
int type_algo;
/*
 i        : indice de la place dont le flot d'entree est a calculer
 type_algo : type de l'algorithme necessitant l'appel de pre.
*/

{
 double flot;
 int k;
 struct Arc *pa;


 flot = 0;

 /* Determination de tous les transitions en amont de Pi */
 for (pa = (P + i - 1)->pre ; pa->i ; pa++)
         {
         /* k est l'indice de la transition */
         k = pa-> i;
         if ((!(T + k -1)->discret) || ALGORITHME_CONTINU)
                 {
                 flot += (T + k - 1)->v * pa->p;
                 }
         }

 /* flot contient le flot d'entree de pi */
 return(flot);
}
```

```
/*********************************************************
*                                                        *
*                    Fichier Inclusion                   *
*                                                        *
* Fichier :   Var_glo.h                                  *
* But     : Contient la declaration des structures et variables globales *
*                                                        *
*********************************************************/

#include "const.h"


/* ---------------------- DEFINITION DE STRUCTURES ---------------------- */


struct Marquage{
            double m;
            };

struct Fonction{
     int (*f)();
     char nomf[10];
     char nomg[10];
     int (*f_1)();
     };

struct Arc{
     int i;
     struct Fonction  *fct;
     Gseg seg;
     double p;       /*correspond au poids de l'arc*/
     };

struct Transition{
      struct Arc *pre;
      struct Arc *post;
      int discret; /*egal a 1 si discret, a 0 si continu */
      tempo d;
      double v;
      int cmpt;  /*compteur du nombre de franchissement de la transition*/
      Gseg seg;
     } *T;

struct Place{
      struct Arc *pre;
      struct Arc *post;
      int discret;   /*egal a 1 si discret, a 0 si continu */
      couleur nc;  /*nombre maximum de couleur */
      double m0;  /*marquage initial */
      double m;   /*marquage courant */
      double md;  /*marquage disponible */
```

```
    double mm;  /*marquage moyen */
    double b;     /*bilan */
      Gseg seg;
    } *P;


typedef union structure{
          struct Arc arc[2];
          struct Transition *tr;
          struct  Place *pl;
          }Gstruct;

typedef struct segment{
           Gint type;
           Gstruct structseg;
          }Gsegment;


struct ptarret{
    int tr;
    int se;
    } arret[1];

struct E {
    double t;
    int T;
    double m;
    } e[NMAXE];


/* Type pour la resolution des conflits */

typedef int type_tableau_conflit [NMAXP];




/* ---------- Definition de variables globales au programme ---------- */


int NP;           /* nombre de places    */
int NT;           /* nombre de transitions*/
int NF;           /* nombre de fonctions  */
int NSEG;         /* nombre de segments (nb de segments de base (5) +nb d'elements dans le RdP) */
int NSEGGRAPH;    /* nombre total de segments (avec les segment du trace de courbes) */
char *deb;        /* pointe sur le premier element de la memoire allouee */
struct Fonction fonction[NMAXF+1];  /* tableau des fonctions connues */

double temps,tmax;
int implicite;          /* designe si on travaille en implicite (1) ou non (0) */
double ABS,ORD;  /* echelles des axes du graphe */

double Pv[NMAXT];
int NTPV;         /* numero de la transition a considerer -1 */
```

```
int chxdeb[NMAXP];  /* tableau indiquant les places pour lesquelles
                       l'utilisateur desire enregistrer dans "simul.h"
                       les valeurs moyennes du marquage */
double Vv[NMAXT][NMAXTE];

type_tableau_conflit tableau_conflit;  /* Tableau de definition de resolution des conflits */
```