Analyse et simulation en langage APL de systèmes de commande décrits par des réseaux de Petri

Jean Renalier

N° d'ordre : 1992

Année 1977

THESE

présentée

DEVANT L'UNIVERSITE PAUL SABATIER DE TOULOUSE (SCIENCES)

en vue de l'obtention

du TITRE de DOCTEUR de SPECIALITE E.E.A.

Option Automatique

par

Jean RENALIER

Maître es Sciences

ANALYSE ET SIMULATION EN LANGAGE APL DE SYSTEMES DE COMMANDE DECRITS PAR DES RESEAUX DE PETRI

Soutenue le 17 juin 1977, devant la commission d'examen :

MM.	R.	BEAUFILS	Président
	A.	COSTES	Examinateurs
	M.	COURVOISIER	
	M.	DIAZ	
	C.	MICHEL	
	J.C.	RAULT	

AVANT - PROPOS

Ce travail de recherche a été effectué au sein de l'Equipe "Représentation, Analyse et Sécurité des Systèmes de Commande" du Laboratoire d'Automatique et d'Analyse des Systèmes du C.N.R.S.

Je tiens, tout d'abord, à exprimer ma gratitude à Monsieur le Professeur J. LAGASSE pour la confiance qu'il m'a témoignée en m'accueillant au Laboratoire.

Je suis très reconnaissant à :

- *Monsieur R. BEAUFILS, Professeur à l'Université Paul Sabatier de Toulouse,*
- *Monsieur A. COSTES, Maître de Conférences à l'Institut National Polytechnique de Toulouse,*
- *Monsieur M. COURVOISIER, Maître-Assistant à l'Université Paul Sabatier de Toulouse,*
- *Monsieur M. DIAZ, Chargé de Recherche au C.N.R.S.,*
- *Monsieur C. MICHEL, Ingénieur à la Division "Systèmes" à la S.I.N.T.R.A.,*
- *Monsieur J.C. RAULT, Chef du Service Recherche et Développement des Applications Scientifiques, Direction de l'Informatique du Groupe THOMSON-C.S.F.,*

qui me font l'honneur de participer à la Commission d'Examen.

Je remercie en particulier Monsieur M. COURVOISIER qui a dirigé notre travail, Monsieur M. DIAZ, Responsable de l'Equipe R.A.S.S.C. du L.A.A.S., et Monsieur A. COSTES, Responsable de la Division "Structures des Systèmes de Commande Automatique".

Monsieur R. VALETTE a été mon premier lecteur. Ses remarques constructives m'ont permis d'améliorer la présentation et la qualité de ce mémoire ; ses conseils et ses encouragements m'ont grandement aidé. Qu'il trouve ici l'expression de ma reconnaissance.

Je tiens à remercier les membres du Service Informatique et Simulation, en particulier Monsieur J.E. DOUCET, pour son aide précieuse, et Monsieur B. MEUNIER, pour son soutien amical.

Ma reconnaissance va également à Monsieur P. AZEMA et à tous les membres de l'Equipe R.A.S.S.C., spécialement Monsieur A. RUIZ DE OLANO qui fut mon compagnon de travail.

Enfin, je remercie Madame J. PENAVAYRE et tous ceux qui, à titres divers, ont permis d'assurer la réalisation matérielle de ce mémoire.

INTRODUCTION

La transcription claire et précise du cahier des charges d'un système logique a toujours été un problème délicat, les seules solutions satisfaisantes ayant consisté à définir puis à utiliser un outil de représentation formel : les machines à états.

Cependant, la complexité actuelle de certains systèmes fait qu'il est nécessaire de les réaliser en utilisant plusieurs sous-systèmes fonctionnant simultanément. Il est alors encore plus difficile d'assurer une définition claire du fonctionnement, et on retrouve un problème de description, car les machines à états sont un modèle insuffisant pour représenter des évolutions simultanées.

C.A. PETRI est un des premiers (1962) à avoir appréhendé le problème de la coordination entre automates, ce qui nécessite l'élaboration d'un modèle adapté à la prise en compte d'évolutions simultanées (ou parallèles) et de procédures de synchronisation.

Plus tard, d'autres modèles ont été proposés : KARP et MILLER [KAR] ont introduit les schémas de programmes parallèles ; les travaux de MARTIN et ESTRIN ont conduit aux graphes UCLA [CER] , HOLT et COMMONER ont repris les travaux de PETRI et ont donné aux réseaux de PETRI leur forme actuelle.

Parmi ces modèles, il a été montré que les réseaux de PETRI sont bien adaptés à la représentation des systèmes de commande automatique [PAT - BLA - COU - VAL.1].

Pour prendre en compte d'une part les événements issus du monde extérieur et agissant sur un système de commande, d'autre part les opérations déclenchées à l'intérieur même du système de commande afin d'élaborer l'information restituée au monde extérieur, on est amené à associer au réseau de PETRI qui représente le séquencement des opérations, un graphe qui montre la façon dont les opérations s'exécutent : le graphe de données.

L'association réseau de PETRI -graphe de données, a été nommée "Schéma à réseau de PETRI" [VAL.1] et la description complète d'un système est obtenue en ajoutant une interprétation qui spécifie la nature des opérations.

D'autres modèles de ce type existent, tel LOGOS [ROS] mais ils n'ont pas été prévus pour une prise en compte aussi aisée des événements extérieurs.

D'autre part, et c'est un deuxième problème très important qu'il faut aborder, les études théoriques sur les réseaux de PETRI ont amené un ensemble de propriétés qui permettent d'entreprendre une analyse d'un système décrit à l'aide d'un réseau. Il est en effet essentiel de pouvoir avoir certaines assurances sur la qualité de la description qui a été faite d'un système et donc de pouvoir détecter les erreurs de fonctionnement qui sont toujours nombreuses au début de la conception.

L'extension de ces propriétés aux schémas à réseaux de PETRI [VAL.1] a amené des procédures permettant de faire une analyse d'un système à plusieurs niveaux.

Il est également souhaitable, avant la phase de réalisation, de vérifier le comportement d'un système en fonction de nombreuses séquences d'événements, c'est-à-dire de faire une simulation. Ceci permet d'accroître considérablement le degré de confiance que l'on pourra avoir dans la réalisation ultérieure.

Le traitement manuel de la cohérence de la description d'un système, et surtout de son analyse et de sa simulation, est absolument impraticable dès que l'on s'intéresse à des systèmes autres qu'académiques. C'est pour cette raison que nous proposons dans ce mémoire l'étude et la mise en oeuvre d'un logiciel d'aide à la conception.

Cependant, dans ce type d'études, il faut éviter, car cela est souvent arrivé, de définir un logiciel trop volumineux et d'accès difficile. Dans cette optique, il était nécessaire de choisir un langage dont la puissance soit reconnue, interactif, et tourné vers la mise au point. Notre choix s'est porté sur le langage APL [IVE] , qui est particulièrement intéressant pour décrire les spécifications d'un système.

En effet, le type des variables, en APL, est défini par le contexte, c'est-à-dire d'une part par la nature des opérateurs qui mettent en jeu ces variables, d'autre part par les assignements que reçoivent les identificateurs. Si les vérifications de compatibilité de types sont nécessaires et effectuées par la machine, l'utilisateur est déchargé de toute déclaration. De plus, il dispose de facilités pour s'enquérir du type d'un identificateur ou d'opérateurs permettant le changement de type, comme par exemple la transformation d'un vecteur en matrice et inversement. Un autre détail intéressant, en langage APL, concerne la portée des variables. La notion de variable commune est la règle par défaut, un identificateur n'étant local que s'il est explicitement défini comme tel.[RAU].

Nous allons maintenant indiquer les différentes parties de ce mémoire.

Dans le premier chapitre, nous donnerons, après avoir présenté les réseaux de PETRI de façon informelle, toutes les définitions concernant ces graphes et leur fonctionnement. Nous verrons ensuite comment a été défini le schéma à réseau de PETRI, le réseau de PETRI seul ne suffisant pas pour représenter tout algorithme.

Dans le deuxième chapitre, nous étudierons les procédures d'analyse de ce modèle. La première partie ne concernera que le graphe de commande, mais la deuxième partie fera aussi intervenir le graphe de données.

Dans le troisième chapitre, nous expliquerons de façon détaillée les programmes que nous avons écrits pour l'étude des schémas à réseaux de PETRI. Les deux premières fonctions réaliseront la description du schéma. Ensuite, utilisant les résultats du deuxième chapitre, la fonction d'analyse donnera automatiquement toutes les propriétés du schéma. Le dernier programme permettra d'effectuer une simulation du système décrit.

Enfin, dans le but d'illustrer les procédures proposées au troisième chapitre, quelques exemples d'application seront étudiés dans le quatrième.

CHAPITRE I

LES RÉSEAUX DE PETRI

-

I.1. INTRODUCTION

Avant d'aborder l'étude des propriétés des réseaux de PETRI, il nous semble nécessaire d'exposer en détail au lecteur ce mode de représentation des systèmes de commande.

L'objet du premier paragraphe sera donc de montrer progressivement les notions de transition, de place et de jeton, c'est-à-dire d'expliquer le rôle des éléments composant un réseau de PETRI.

Ainsi, dans le deuxième paragraphe, nous pourrons définir de façon formelle ces réseaux et expliquer leur fonctionnement.

Pour pouvoir étudier de façon méthodique les propriétés d'un réseau et son fonctionnement, nous indiquerons des moyens mathématiques de représentation.

Nous verrons ensuite que selon les conventions adoptées, il existe plusieurs catégories de réseaux de PETRI ; certaines auront l'avantage de permettre une représentation simple et concise, d'autres celui d'être analysées facilement.

Enfin, nous préciserons comment un réseau de PETRI peut représenter un système de commande ; en associant à ce réseau de PETRI un graphe de données et une interprétation, nous obtiendrons le "Schéma à Réseau de PETRI".

I.2. PRESENTATION INFORMELLE

I.2.1. L'automate à états fini

Pour l'étude et la représentation des automatismes logiques utilisés dans certains systèmes de commande, un des modèles généralement employés est la machine à états finie. Pour décrire un tel système, on fait correspondre un cercle à chacun des états. Ces cercles sont reliés entre-eux par des flèches pour former un graphe. Chacune de ces flèches correspond à un événement qui modifie l'état dans lequel se trouve le système.

Considérons par exemple l'automate à états fini de la figure I.1. Supposons le système à l'état 1 : l'arrivée de l'événement A fait passer le système à l'état 2, mais celle des événements B ou C ne change rien. De même, l'état 2 est insensible aux événements A ou C, alors que l'événement B amène le système à l'état 3, et ainsi de suite selon l'arrivée des événements associés aux flèches.

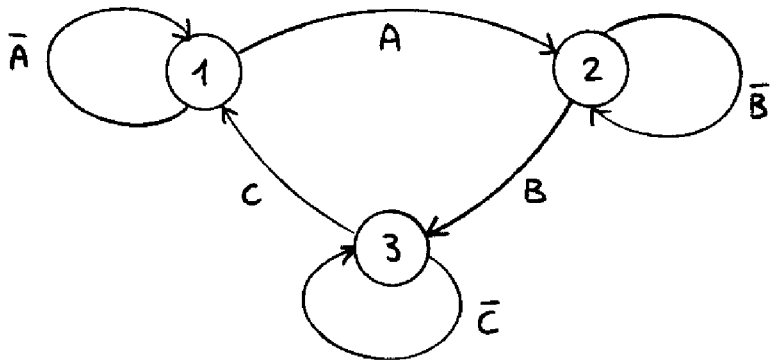


FIGURE I.1.

I.2.2. Transitions

Ce système peut également être représenté par un réseau de PETRI de la façon suivante (figure I.2.).

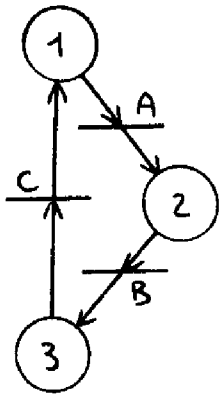


FIGURE I.2.

Le passage de la première à la seconde représentation se fait en plaçant une barre sur les flèches. Cette barre souligne la "transition" d'un état à un autre.

L'évolution du système se fait de la même façon que précédemment ; mais ici, la représentation est plus condensée, car, par convention, seuls sont indiqués les événements auxquels le système est réceptif, tandis que dans le cas des machines à états, il est convenu de représenter explicitement les maintiens.

I.2.3. Evénements simultanés

Dans cet exemple, nous n'avons considéré que des événements successifs. Mais il est possible que, dans un certain état, un système soit réceptif à plusieurs événements susceptibles d'arriver de façon totalement indépendante, en parallèle.

Ainsi, considérons l'automate de la figure I.3.a. et le réseau de PETRI obtenu en faisant la transposition comme précédemment, figure I.3.b.

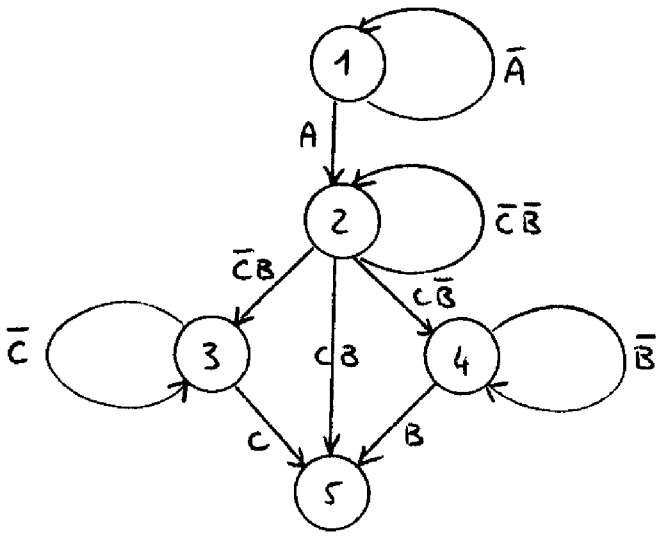


FIGURE I.3.a.
Automate à états fini.

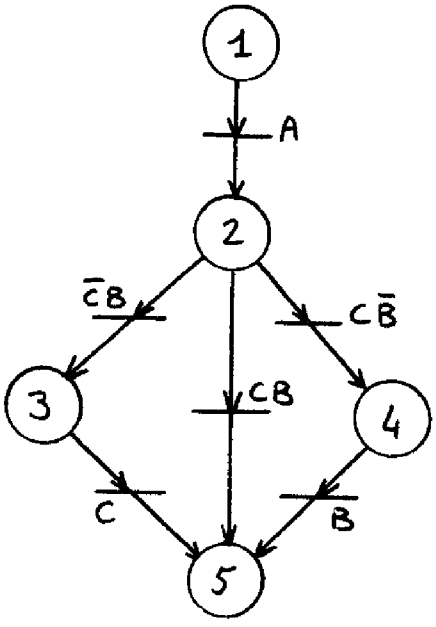


FIGURE I.3.b.
Réseau de PETRI

Examinons simultanément ces deux représentations du même système. L'événement A amène le système dans un état où il est sensible à la fois à l'arrivée de B et à celle de C : ces deux événements font passer le système à l'état 3 ou 4, puis à l'état 5 quand ils arrivent successivement ou bien directement à l'état 5 lorsqu'ils arrivent simultanément.

Il est possible de donner une représentation différente de ce même système par un autre réseau de PETRI plus simple, figure I.4., qui traduira cependant le même fonctionnement.

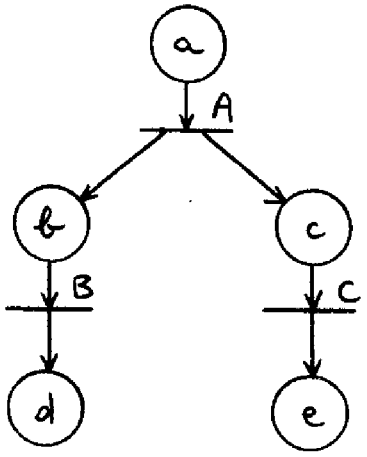


FIGURE I.4.

Si le degré de parallélisme augmente, le nombre d'états nécessaires pour représenter l'automate augmente lui aussi, et ceci d'une manière exponentielle. Au contraire, dans la représentation par réseau de PETRI, un événement en parallèle supplémentaire se traduit par une seule branche supplémentaire pour le graphe : la croissance du graphe est donc linéaire.

La figure I.5. souligne l'intérêt des réseaux de PETRI pour la représentation du parallélisme. Il apparaît nettement qu'à partir de trois événements en parallèle (ici les événements B, C et D), la représentation sous forme de machine à états finie devient illisible, alors que la description sous forme de réseau de PETRI reste claire et concise.

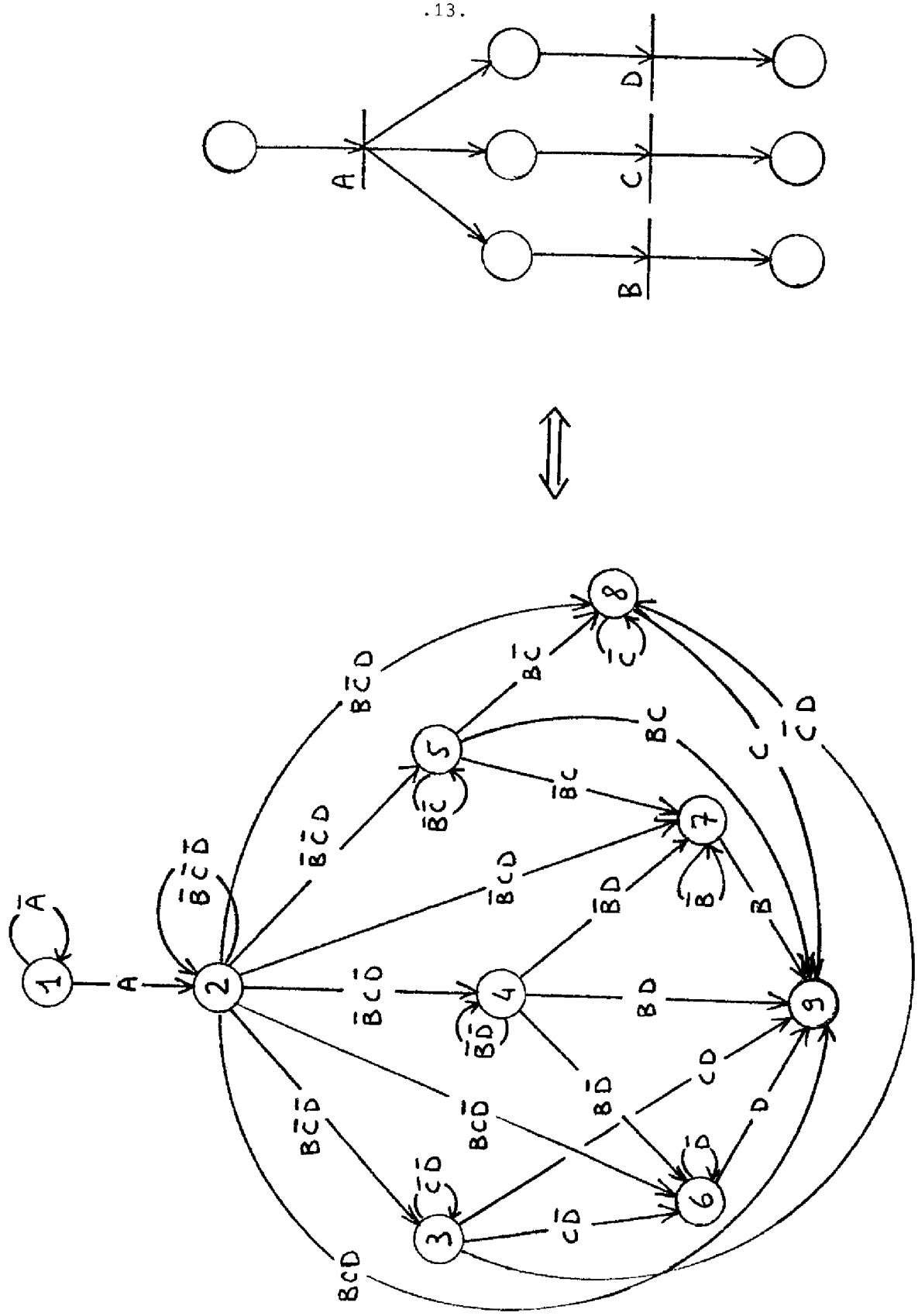

A
D
C
B
1
2
3
4
5
6
7
8
9
$\bar{A}$
A
$\bar{B}\bar{C}\bar{D}$
$\bar{B}\bar{C}D$
$\bar{B}C\bar{D}$
$\bar{B}CD$
$B\bar{C}\bar{D}$
$B\bar{C}D$
$BC\bar{D}$
BCD
$\bar{C}\bar{D}$
$C\bar{D}$
$\bar{B}$
$\bar{C}$
$\bar{D}$
B
C
D

FIGURE I.5.

I.2.4. Places et jetons

Lorsque le fonctionnement du système est décrit par un automate à états fini, chaque sommet du graphe représente un état du système ; c'est-à-dire qu'à partir de ce sommet, il est possible de prévoir l'évolution future du système, en fonction des événements.

Par contre, dans le cas de la représentation du système à l'aide du réseau de PETRI de la figure I.4., plusieurs sommets du graphe peuvent être activés à la fois. Les sommets du réseau ne représentent que des états partiels, des conditions locales pour que le système soit réceptif à certains événements. Le fait que le sommet b soit actif signifie que le système est réceptif à l'événement B et ceci indépendamment du fait que l'événement C ait été pris en compte ou non.

Pour souligner cette nature différente, les sommets d'un réseau de PETRI représentés par des cercles sont appelés des "places". Les places activées seront marquées par un signe distinctif : un "jeton".

Reprenons le fonctionnement du système sur le réseau de PETRI de la figure I.4.

Initialement, un jeton se trouve dans la place a. L'arrivée de A amène le système en deux états partiels à la fois : b et c, réceptifs aux événements B et C ; A constitue la transition entre la place a et les places b et c ; le sommet a n'est plus actif, mais b et c le deviennent : on enlève donc le jeton de a et on en met un dans chacune des places b et c.

De même, l'événement B constitue la transition entre les places b et d : le jeton de b passe dans d ; et l'arrivée de C fait passer le jeton de c dans e. Dans le cas où B et C sont deux événements simultanés, les jetons de b et c passent simultanément dans d et e.

Le tableau suivant nous donne la correspondance, à chaque instant, de l'évolution du système, entre les états de l'automate et les sommets activés dans le réseau de PETRI.

AUTOMATE	1	2	3	4	5
RESEAU DE PETRI	a	b et c	d et c	b et e	d et e

I.2.5. Cas de plusieurs jetons dans une place

La position des jetons dans le réseau de PETRI indique l'état du système et évolue donc en même temps que lui.

Un problème se pose au sujet de cette évolution et de la signification des jetons. Considérons par exemple le cas de la figure I.6.

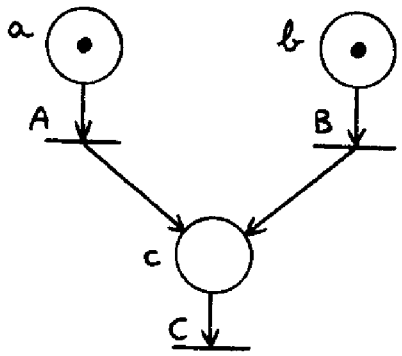


FIGURE I.6.

Deux places a et b, marquées d'un jeton, sont suivies chacune d'une transition, correspondant respectivement aux événements A et B et ces deux transitions précèdent la même place c. Supposons qu'arrivent successivement A, puis B : le sommet c serait activé deux fois, il y aurait deux jetons dans cette place.

Il est possible d'accepter ou non cette situation et il existe plusieurs façons de l'interpréter :

a) une première solution consiste à dire qu'une place est marquée ou ne l'est pas : dans le cas où le sommet est actif, le nombre de jetons n'a pas d'importance. Si une place représente une condition

logique, ceci revient à dire qu'une condition remplie deux fois est tout simplement une condition remplie. Donc même après le marquage de la place c consécutif à l'arrivée de l'événement A, le système reste réceptif à l'événement B et l'occurrence de C rendra le sommet c inactif de la même façon que s'il n'avait été activé qu'une seule fois. La figure I.7. montre l'évolution des jetons.

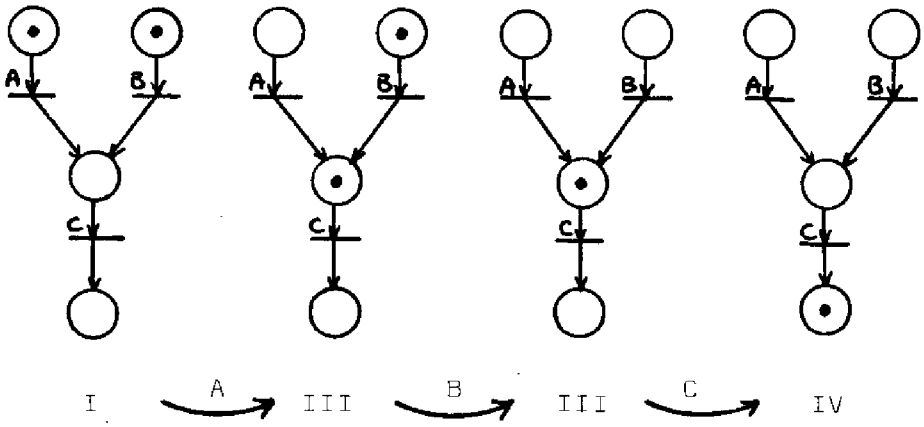


FIGURE I.7.

Cette solution a été adoptée dans certains modèles : les RCPP [MOA] et l'Organiphase [PRU].

b) Une deuxième solution consiste à refuser qu'un sommet soit activé deux fois. Pour cela, il faut s'imposer une nouvelle condition : une transition ne pourra être tirée si un des sommets suivant cette transition est actif à l'instant considéré. Dans notre exemple, si la place c est marquée d'un jeton consécutivement à l'arrivée de l'événement A ou B cette place ne pourra être active une deuxième fois qu'après l'arrivée de l'événement C qui aura enlevé le premier jeton (figure I.8.).

C'est cette solution qui a été retenue à l'origine par PETRI lui-même [PET] ; elle a aussi été adoptée pour le modèle LOGOS [ROS].

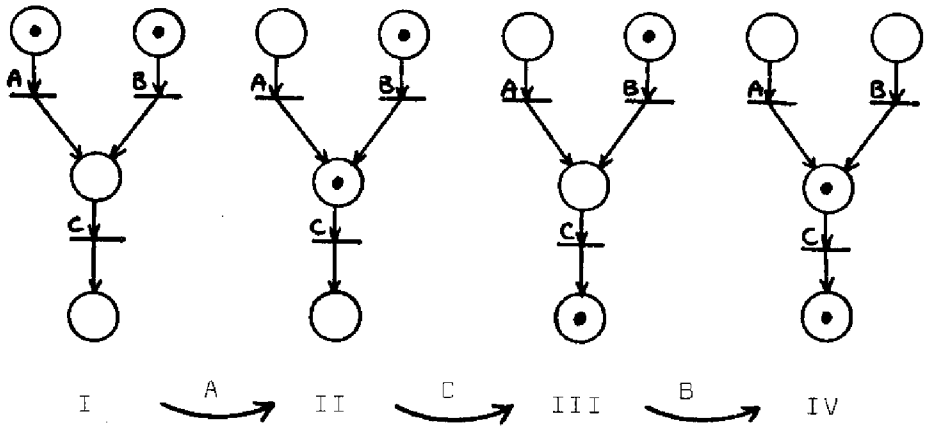


FIGURE I.8.

c) On peut aussi considérer un marquage multiple d'une troisième façon : chaque jeton active le sommet une fois. Les jetons sont accumulés, ce qui permet de garder en mémoire le nombre d'activations d'un sommet, et l'arrivée de l'événement associé à la transition suivant un sommet marqué de plusieurs jetons, n'enlèvera qu'un seul jeton à cette place. Ainsi, si nous reprenons notre exemple, la place c possède deux jetons après l'arrivée de A et de B, puis l'arrivée de l'événement C fait passer l'un d'eux sur le sommet suivant (figure I.9.).

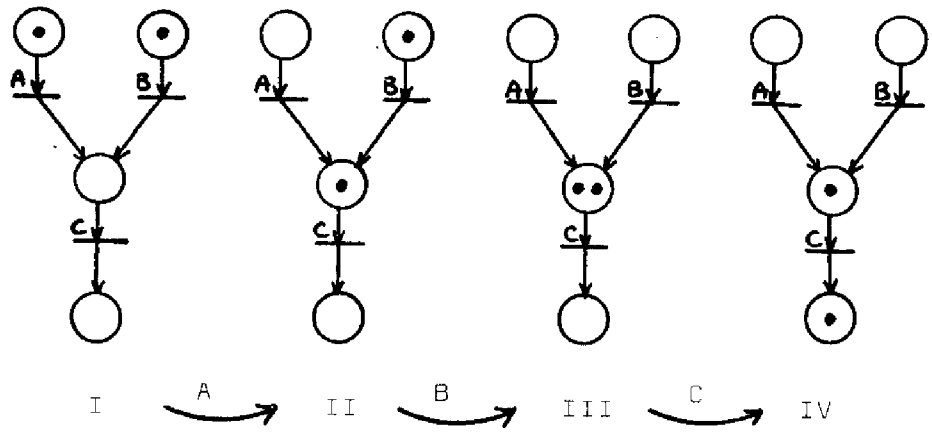


FIGURE I.9.

d) Enfin, une quatrième solution consiste à considérer une place marquée deux fois dans un réseau de PETRI comme une erreur de conception. Si un opérateur est associé à cette place, l'arrivée d'un jeton correspond au début de la tâche effectuée par cet opérateur. Il est donc nécessaire que cette tâche soit terminée, ce qui se traduit par le départ du jeton, pour que l'opérateur puisse être sollicité à nouveau.

CONCLUSION

La première solution présente un inconvénient : la disparition de certains jetons fait perdre une partie de l'information. La deuxième, elle, est trop contraignante, puisqu'il faut tenir compte, pour le tir d'une transition, des places qui suivent celle-ci ; d'un point de vue matériel, elle est plus compliquée à réaliser ; de plus, elle est trompeuse car elle peut amener à corriger d'elle-même des erreurs de conception.

C'est la troisième solution qui a été retenue par HACK [HAC] et d'autres chercheurs ayant développé les réseaux de PETRI aux Etats-Unis. C'est aussi cette solution que nous avons adoptée dans la suite de ce chapitre. Mais nous verrons qu'en général, tant pour la description, que pour la réalisation, il est préférable de considérer qu'une place possède un jeton ou aucun ; aussi, c'est la quatrième solution que nous retiendrons pour les chapitres suivants.

I.3. DEFINITIONS

Nous pouvons maintenant présenter les réseaux de PETRI de façon formelle.

I.3.1. Définition I.1. : réseau de PETRI

Un réseau de PETRI $\mathcal{P}$ est un triplet $\{P,T,A\}$, où :

- P est un ensemble fini de places $\{p_1,p_2,\ldots,p_n\}$,
- T est un ensemble fini de transitions $\{t_1,t_2,\ldots,t_m\}$,
- A est une relation qui correspond à un ensemble d'arcs où chaque arc lie, soit une place à une transition, soit une transition à une place. A est inclus dans l'ensemble $(P \times T) \cup (T \times P)$.

$\mathcal{P}$ est donc un graphe qui comprend deux types de sommets : des places et des transitions. La figure I.10. montre un exemple de réseau de PETRI.

I.3.2. *Définition I.2. : marquage d'un réseau*

On appelle marquage M une fonction de l'ensemble des places P sur l'ensemble des entiers naturels N :

$$M : P \longrightarrow N$$

C'est une distribution de jetons : à chaque place, on en associe un certain nombre.

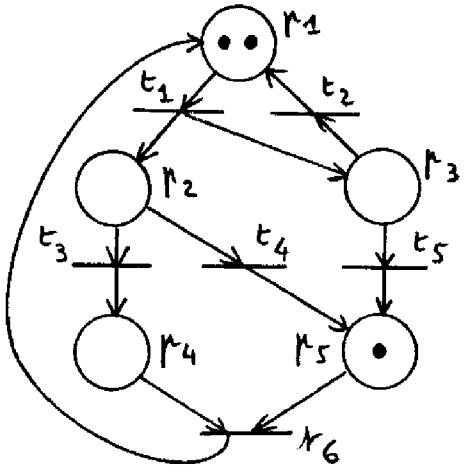
Le marquage d'un réseau de PETRI est généralement donné par la liste des nombres de jetons contenus dans chaque place ; ainsi le marquage du réseau de la figure I.10. est donné par :

$M = (2,0,0,0,1)$.

Il est aussi possible de donner le nom des places en répétant celles marquées plusieurs fois :

$M = (p_1,p_1,p_5)$ ou encore (p_1^2, p_5).

Le marquage initial d'un réseau de PETRI (distribution des jetons avant l'analyse ou avant une simulation) est généralement noté M_o.



$$P = \{p_1,p_2,p_3,p_4,p_5\}$$

$$T = \{t_1,t_2,t_3,t_4,t_5,t_6\}$$

$$A = \{(p_1,t_1),(t_1,p_2),\ldots,(t_6,p_1)\}$$

FIGURE I.10.

I.3.3. Définition I.3. : transition sensibilisée

On dit qu'une transition t_i est sensibilisée si toutes les places précédentes de t_i possèdent au moins un jeton.

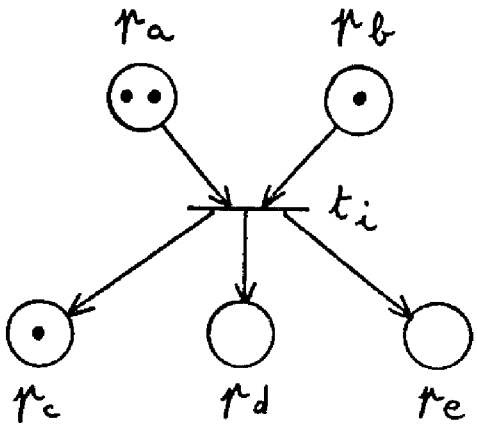
Nous dirons aussi, principalement au cours de la simulation des réseaux de PETRI, qu'une telle transition est "possible".

I.3.4. Définition I.4. : tir d'une transition

Une transition sensibilisée peut être tirée. Un jeton est alors enlevé à chacune de ses places précédentes et chacune de ses places suivantes reçoit un jeton. Le tir d'une transition est supposé instantané.

La figure I.11. montre la position des jetons avant et après le tir de la transition (nous avons adopté dans cet exemple la solution consistant à admettre plusieurs jetons dans une même place).

AVANT



APRES

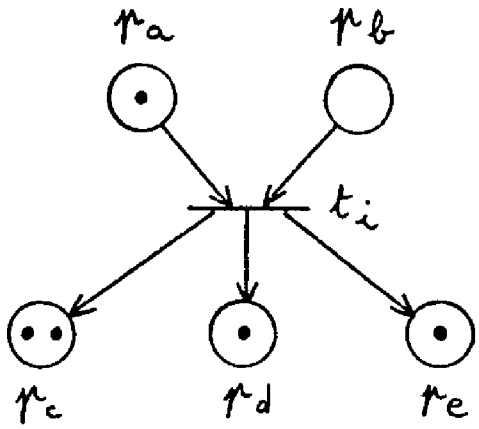


FIGURE I.11.

Si M_i est le marquage sensibilisant la transition t_i, et M_j le marquage résultant du tir de t_i, nous écrirons :

$$M_i \xrightarrow{t_i} M_j$$

Dans l'exemple précédent :

$$(2,1,1,0,0) \xrightarrow{t_i} (1,0,2,1,1)$$

ou encore : $(p_a,p_a,p_b,p_c) \xrightarrow{t_i} (p_a,p_c,p_c,p_d,p_e)$

Dans le réseau de PETRI de la figure I.10., le tir des transitions t_1 et t_2 s'écrit :

$$(2,0,0,0,1) \xrightarrow{t_1} (1,1,1,0,1)$$

$$(1,1,1,0,1) \xrightarrow{t_2} (2,1,0,0,1)$$

I.3.5. Définition I.5. : séquence de tir de transitions

Soit S une séquence finie de transitions, t_i, $t_{i+1},\ldots,t_{i+k}$, de l'ensemble T ; on dit que S est une séquence de tir tirable à partir de M_i si et seulement si il existe des marquages M_{i+1}, $M_{i+2},\ldots,M_{i+k+1}$, du réseau de PETRI, tels que $M_i \xrightarrow{t_i} M_{i+1}$, $M_{i+1} \xrightarrow{t_{i+1}} M_{i+2},\ldots$ $M_{i+k} \xrightarrow{t_{i+k}} M_{i+k+1}$.

Nous écrirons ceci : $M_i \xrightarrow{S} M_{i+k+1}$

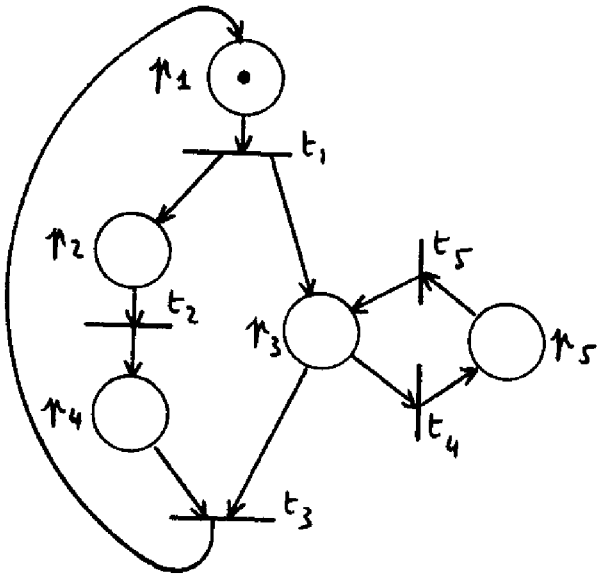


FIGURE I.12.

Pour le réseau de PETRI de la figure I.12., nous pouvons écrire :

$$M_o \xrightarrow{t_1} M_1$$

avec $M_o = (p_1)$ et $M_1 = (p_2,p_3)$,

puis $M_1 \xrightarrow{t_2} M_2$ avec $M_2 = (p_3,p_4)$,

puis $M_2 \xrightarrow{t_4} M_3$ avec $M_3 = (p_4,p_5)$.

Cette séquence de tir de trois transitions peut s'écrire plus simplement :

$$M_o \xrightarrow{t_1\ t_2\ t_4} M_3$$

I.3.6. Définition I.6. : classe des marquages conséquents

La classe des marquages conséquents $\overrightarrow{M_o}$ est l'ensemble des marquages M_i accessibles, c'est-à-dire tels qu'il existe au moins une séquence de tir S, tirable à partir de M_o et produisant M_i :

$$M_i \in \overrightarrow{M_o} \iff \exists\ S \text{ telle que } M_o \xrightarrow{S} M_i$$

Pour l'exemple de la figure I.12., la classe des marquages conséquents, pour le marquage initial représenté $M_o = (p_1)$, est :

$$\overrightarrow{M_o} = \left\{ (p_1) ; (p_2,p_3) ; (p_3,p_4) ; (p_2,p_5) ; (p_4,p_5) \right\}$$

Pour un autre marquage initial, par exemple $M'_o = (p_3,p_5)$, nous obtenons :

$$\overrightarrow{M'_o} = \left\{ (p_3,p_5) ; (p_3,p_3) ; (p_5,p_5) \right\} .$$

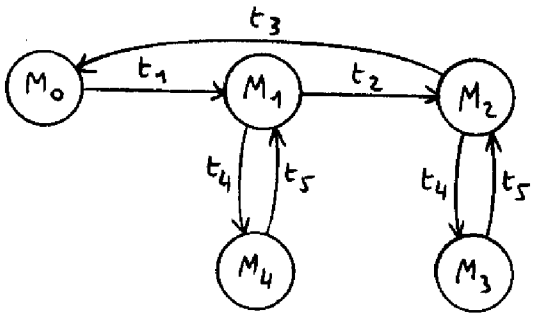
I.3.7. *Définition I.7. : graphe des marquages conséquents*

Soit un réseau de PETRI $\mathcal{P}$ et un marquage initial M_o tels que la classe des marquages conséquents $\overrightarrow{M_o}$ soit finie. Le graphe des marquages conséquents $G(\overrightarrow{M_o})$ est le graphe orienté (X,U) où $X = \overrightarrow{M_o}$ est l'ensemble des sommets et où U est l'ensemble des arcs (M_i,M_j) tels que :

1. $M_i \in \overrightarrow{M_o}$ et $M_j \in \overrightarrow{M_o}$
2. Il existe une transition t_i de $\mathcal{P}$ telle que $M_i \xrightarrow{t_i} M_j$.

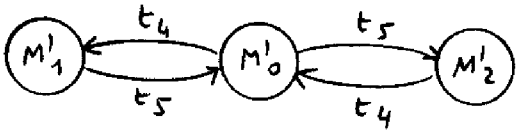
Chaque arc de $G(\overrightarrow{M_o})$ est étiqueté par la transition de $\mathcal{P}$ correspondante.

Les figures I.13., I.14. et I.15. donnent les graphes des marquages conséquents obtenus pour le réseau de la figure I.12., avec trois marquages initiaux différents.



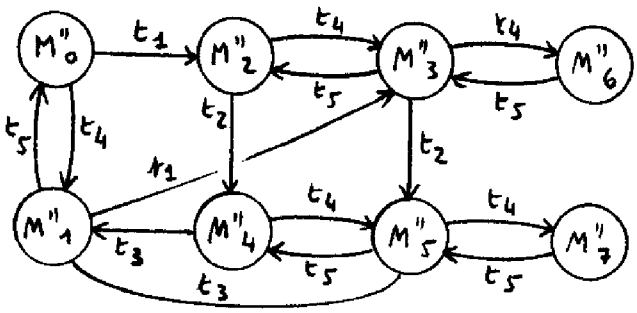
$M_o = (p_1)$
$M_1 = (p_2,p_3)$
$M_2 = (p_3,p_4)$
$M_3 = (p_4,p_5)$
$M_4 = (p_2,p_5)$

FIGURE I.13.



$M'_o = (p_3,p_5)$
$M'_1 = (p_5,p_5)$
$M'_2 = (p_3,p_3)$

FIGURE I.14.



$M"_o = (p_1,p_3)$
$M"_1 = (p_1,p_5)$
$M"_2 = (p_2,p_3,p_3)$
$M"_3 = (p_2,p_3,p_5)$
$M"_4 = (p_3,p_3,p_4)$
$M"_5 = (p_3,p_4,p_5)$
$M"_6 = (p_2,p_5,p_5)$
$M"_7 = (p_4,p_5,p_5)$

FIGURE I.15.

I.3.8. Remarque

Nous pouvons remarquer l'analogie entre le graphe des marquages conséquents obtenu à partir d'un marquage initial donné d'un réseau de PETRI et l'automate à états fini qui représente le fonctionnement d'un même système pour les mêmes conditions initiales. Chaque sommet du graphe, c'est-à-dire chaque marquage, regroupe les différents "états partiels" du système, de sorte que nous retrouvons bien "l'état global" représenté par l'automate.

Toutefois, nous l'avons déjà signalé précédemment, les événements auxquels le système n'est pas réceptif, n'interviennent pas dans la représentation par les réseaux de PETRI ; aussi, le graphe de l'automate possède des arcs qui n'ont pas leur correspondant dans le graphe des marquages conséquents.

La figure I.16. montre, par exemple, le graphe des marquages conséquents obtenu pour le réseau de PETRI de la figure I.4., muni d'un marquage initial (a). Le tableau de correspondance du paragraphe I.2.4. nous permet de voir l'analogie avec l'automate de la figure I.3., représentant le même système ; la différence est faite par les arcs correspondant à $\overline{A}, \overline{B}, \ldots$, auxquels le système n'est pas réceptif.

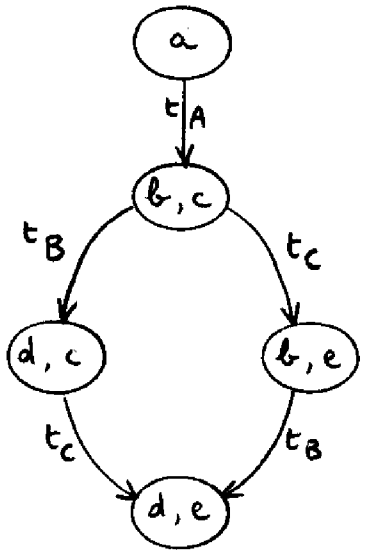


FIGURE I.16.

I.3.9. Places et transitions particulières

Des configurations particulières peuvent apparaître dans un réseau de PETRI.

Définitions I.8. : transitions source et puits

Une transition n'ayant aucune place précédente sera dite transition source.

Une transition n'ayant aucune place suivante sera dite transition puits.

La figure I.17. montre un exemple de telles transitions.

transition source puits

FIGURE I.17.

Le tir d'une transition source est toujours possible ; il conduit à une accumulation de jetons dans ses places suivantes.

Celui d'une transition puits conduit à une disparition de jetons.

Définitions I.9. : places source et puits

Une place n'ayant aucune transition précédente est appelée place source.

Une place n'ayant aucune transition suivante est appelée place puits.

La figure I.18. montre un exemple de place source et de place puits.

place source puits

FIGURE I.18.

Si une place source contient initialement des jetons, elle pourra les céder quand une des transitions suivantes sera tirée, mais aucun jeton ne pourra plus parvenir à cette place.

Inversement, une place puits ne pourra qu'accumuler les jetons qu'elle reçoit sans jamais les céder.

Quand on décrit un système de commande, il arrive que le réseau de PETRI obtenu possède une place initiale source et une place finale puits, représentant les états initial et final (figure I.19.a.)

En fait, un système de commande fonctionne de manière répétitive et il est tout à fait normal de reboucler le réseau en introduisant une transition supplémentaire permettant de remettre le système dans l'état initial (figure I.19.b.).

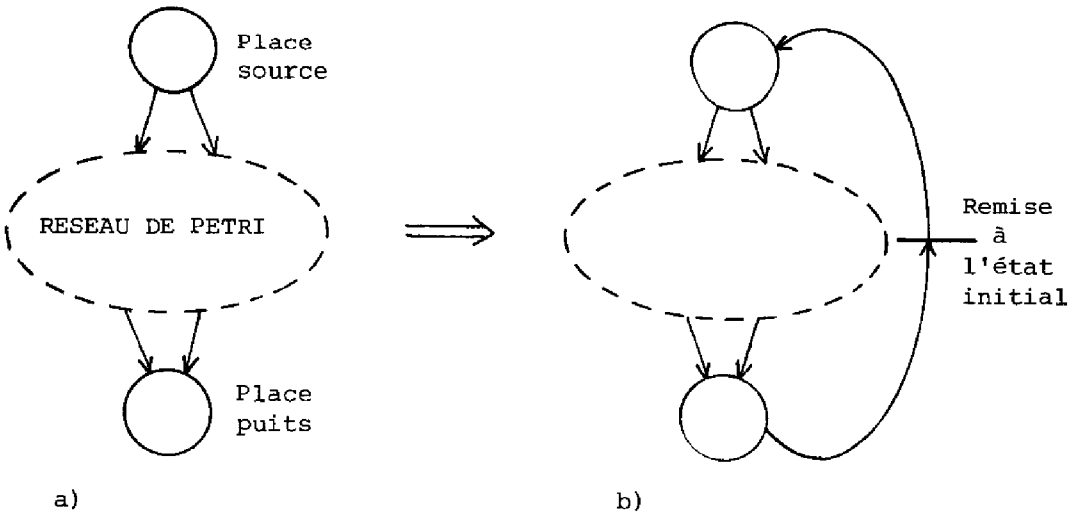


FIGURE I.19.

Cette transition fait disparaître les places sources et les places puits, ce qui, comme nous le verrons par la suite, est une condition nécessaire pour pouvoir analyser un réseau de PETRI.

Définition I.10. Boucle élémentaire

Il y a une boucle élémentaire dans un réseau de PETRI quand une transition est telle que l'ensemble de ses places précédentes n'est pas disjoint de celui de ses places suivantes.

Définition I.11. : réseau de PETRI pur

Un réseau de PETRI sans boucle élémentaire est appelé un réseau pur.

La figure I.20. montre une boucle élémentaire.

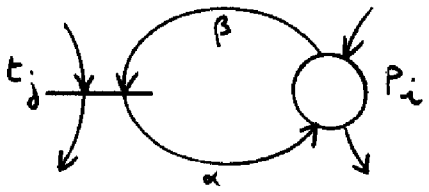


FIGURE I.20.

Nous verrons plus loin l'utilité que peut avoir une telle configuration, mais aussi ses inconvénients et la façon d'y remédier.

I.4. REPRESENTATION MATRICIELLE D'UN RESEAU DE PETRI

Un réseau de PETRI peut être facilement représenté par des outils mathématiques ; une matrice suffit pour décrire les sommets, places et transitions, et les arcs qui les relient entre-eux.

I.4.1. Matrice d'incidence

Définition I.12.

La matrice d'incidence d'un réseau de PETRI est une matrice possédant autant de lignes qu'il y a de places, autant de colonnes qu'il

y a de transitions et telle que :

si C_{ij} est l'élément de cette matrice situé à l'intersection de la ième ligne et de la jème colonne :

- $C_{ij} = + 1$ quand il existe un arc allant de la transition j vers la place i,
- $C_{ij} = - 1$ quand il existe un arc allant de la place i vers la transition j,
- $C_{ij} = 0$ dans tous les autres cas.

EXEMPLE : Reprenons le réseau de PETRI de la figure I.12. La matrice d'incidence est la suivante :

$$\begin{pmatrix} -1 & 0 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & 1 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

→ place 2

→ transition 4

REMARQUE

La présence d'une boucle élémentaire dans le réseau de PETRI rend impossible la construction de sa matrice d'incidence.

En effet, l'élément C_{ij} qui a pris la valeur +1 lors de la description de l'arc α (figure I.20.) prend la valeur -1 quand est décrit l'arc β, ce qui efface la présence du premier arc.

Nous verrons plus loin, pour notre programme de description, au paragraphe III.2.1., qu'il est facile de trouver une solution à cet inconvénient.

I.4.2. Matrices des places d'entrée et de sortie

Pour décrire la forme graphique du réseau de PETRI, il y a une autre possibilité que nous n'utiliserons que pour notre programme de

simulation ; elle consiste à indiquer, pour chaque transition, les numéros attribués aux places précédentes et aux places suivantes. On obtient ainsi deux matrices dont chaque ligne correspond à une transition.

Pour le même réseau de PETRI que précédemment, nous obtenons :

PE =				PS =	
1	0			2	3
2	0			4	0
3	4			1	0
3	0	⟵ - - - - -	t_4 - - - - ⟶	5	0
5	0			3	0

Ces deux matrices sont dites "des places d'entrée" et "des places de sortie". Il faut évidemment compléter les places vides des dernières colonnes par des zéros pour obtenir des matrices rectangulaires complètes.

Ce mode de représentation admet la présence des boucles élémentaires dans le réseau. Il présente peut-être une meilleure lisibilité que la matrice d'incidence, mais il est bien moins adapté au calcul des marquages conséquents. Selon le traitement effectué sur les données, nous utiliserons l'une ou l'autre des deux méthodes ; mais, comme nous allons le voir, la matrice d'incidence sera indispensable pour le calcul des marquages.

Cette matrice reflète de façon complète la forme graphique du réseau de PETRI ; il reste tout de même à représenter le marquage du réseau : la position des jetons et leur nombre. D'après la définition I.2. du marquage, il suffit pour cela d'écrire un vecteur ligne de dimension égale au nombre de places, chacune de ses composantes étant égale au nombre de jetons figurant dans la place correspondante.

Ainsi, pour le réseau de PETRI de la figure I.12., le vecteur marquage initial s'écrit :

$M_o = (1,0,0,0,0)$, tandis que pour celui de la figure I.10.,

$M_o = (2,0,0,0,1)$.

C'est l'ensemble constitué de ce vecteur marquage initial et de la matrice d'incidence du réseau qui nous permet de calculer tous les marquages conséquents, avec l'aide des systèmes d'addition de vecteurs.

1.4.3. *Système d'addition de vecteurs*

C'est un outil mathématique utilisé pour l'étude des réseau de PETRI, mais aussi pour celle d'autres modèles.

Définition I.13.

Un système d'addition de vecteurs de dimension n est une paire w = (d,W) où :

- d est un vecteur de $\mathbb{N}^n$ (ses composantes sont des entiers positifs ou nuls),
- W est un ensemble fini de vecteurs de $\mathbb{Z}^n$ (leurs n composantes sont des entiers relatifs).

Le système d'addition de vecteurs qui nous intéresse ici est formé :

- du vecteur marquage initial M_o : ses composantes, dont le nombre est égal à celui des places, "NP", sont des entiers positifs ou nuls,
- de l'ensemble des colonnes de la matrice d'incidence "C" : chaque colonne a NP composantes égales à -1, 0, ou +1.

Définition I.14.

L'ensemble des vecteurs accessibles à partir du système d'addition (d,W) est l'ensemble, noté R(W), de tous les vecteurs de la forme :

$$d + W_1 + W_2 + \ldots + W_i + \ldots + W_s,$$

où les vecteurs W_i sont des vecteurs de W et où les vecteurs $d + W_1 + \ldots + W_i$ ont leurs composantes positives ou nulles.

L'ensemble des vecteurs accessibles pour le système (M_o,C) est l'ensemble des marquages accessibles à partir du marquage M_o par une séquence de tir de transitions, c'est-à-dire la classe des marquages conséquents $\overrightarrow{M_o}$.

Le marquage M_j du réseau, après le tir de la transition t_k, à partir du marquage M_i, est obtenu de la façon suivante :

$$M_i \xrightarrow{t_k} M_j$$

$$M_j = M_i + C_k$$

où C_k est le kème vecteur colonne de la matrice d'incidence.

EXEMPLE : Reprenons le réseau de PETRI de la figure I.12. et la classe des marquages conséquents obtenue figure I.13. :

à partir de $M_o=(p_1)$, le tir de la transition t_1 se traduit par :

$$M_o \xrightarrow{t_1} M_1 = M_o + C_1$$

$$M_1 = (1,0,0,0,0) + (-1,1,1,0,0)$$

$$M_1 = (0,1,1,0,0)$$

puis $M_1 + C_2 = M_2$,... etc.

Théorème I.1.

La condition nécessaire et suffisante pour que l'ensemble R(w) soit fini est que toutes les composantes des vecteurs de R(w) soient bornées [KAR].

Nous reviendrons plus loin sur ce point, lors de l'analyse des réseaux de PETRI ; mais nous pouvons déjà déduire de ce théorème que la classe des marquages conséquents est finie si et seulement si les vecteurs marquages sont bornés, c'est-à-dire si le nombre de jetons dans chaque place est borné.

I.5. DIFFERENTES CLASSES DE RESEAUX DE PETRI

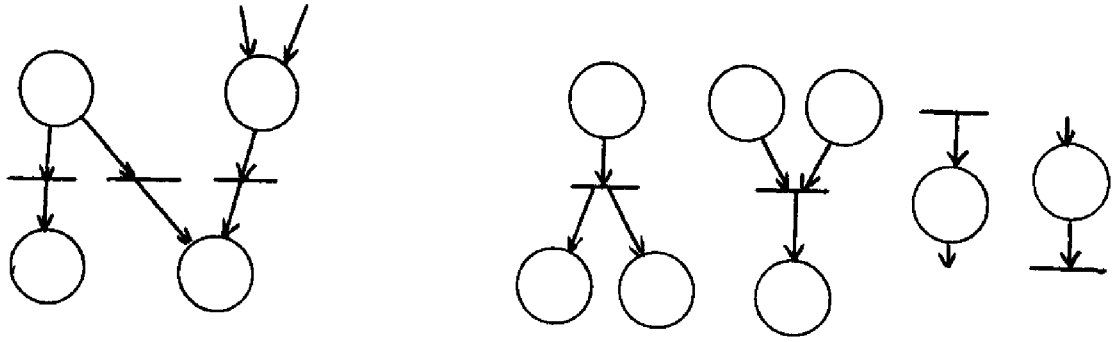
Il est possible d'établir une classification des réseaux de PETRI en prenant pour critère le nombre d'arcs reliant les places et les transitions [RAM - HAC].

I.5.1. Machine à états

Définition I.15.

Un réseau de PETRI est une machine à états si et seulement si toute transition du réseau possède une place suivante et une seule, et une place précédente et une seule.

La figure I.21. montre différentes configurations permises et interdites dans une machine à états.



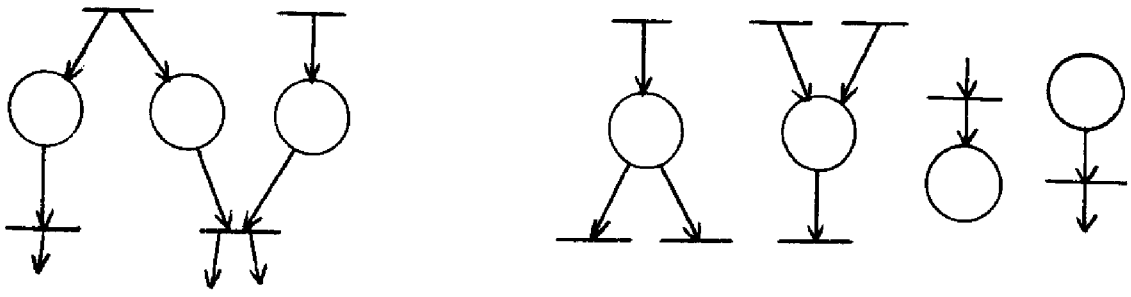
permis interdit

FIGURE I.21.

I.5.2. Graphe d'événements

Définition I.16.

Un réseau de PETRI est un graphe d'événements si et seulement si toute place du réseau possède une transition précédente et une seule, et une transition suivante et une seule.



permis interdit

FIGURE I.22.

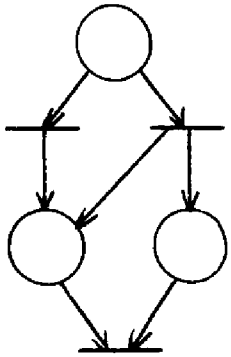
I.5.3. Réseaux libre choix

Définition I.17.

Un réseau de PETRI est dit "réseau libre choix" si et seulement si chaque arc partant d'une place vers une transition est soit l'unique sortie de cette place, soit l'unique entrée de cette transition.

Cette classe de réseaux contient les deux précédentes : une machine à états ou un graphe d'événements sont des réseaux libre choix avec chacun une restriction supplémentaire.

permis



interdit

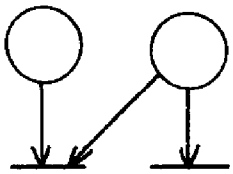


FIGURE I.23. Configurations permises et interdites dans un réseau de PETRI libre choix.

I.5.4.

Enfin, la classe la plus générale des réseaux de PETRI est celle où aucune restriction n'est faite sur le nombre de places précédentes partagées d'une transition.

Toutes les configurations possibles sont alors permises (figure I.24., par exemple).

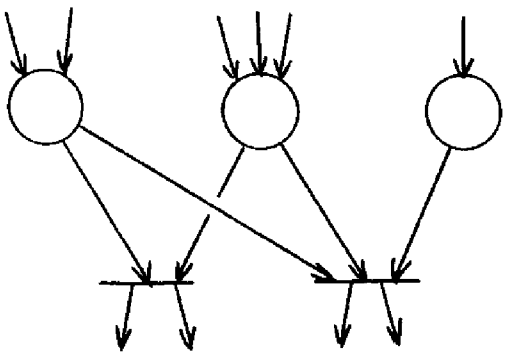


FIGURE I.24.

REMARQUE 1

Les graphes d'événements sont aussi appelés "graphes orientés marqués". Mais nous n'emploierons pas cette dénomination pour éviter toute confusion avec les "réseaux de PETRI marqués", qui sont pour nous des réseaux de PETRI pour lesquels ont été donnés des marquages initiaux.

REMARQUE 2

L'utilisation exclusive des deux premières classes de réseaux, les graphes d'événements et les machines à états, ne permet pas de représenter des systèmes comportant à la fois des choix et du parallélisme.

I.6. GENERALISATION DES RESEAUX DE PETRI

On peut considérer que les réseaux de PETRI sont eux-mêmes une certaine classe des systèmes d'addition de vecteurs (d,W), celle pour laquelle les composantes de vecteurs de l'ensemble W appartiennent à l'ensemble (-1,0,+1). On peut alors proposer une généralisation des réseaux de PETRI [HAC].

I.6.1. Réseaux de PETRI généralisés

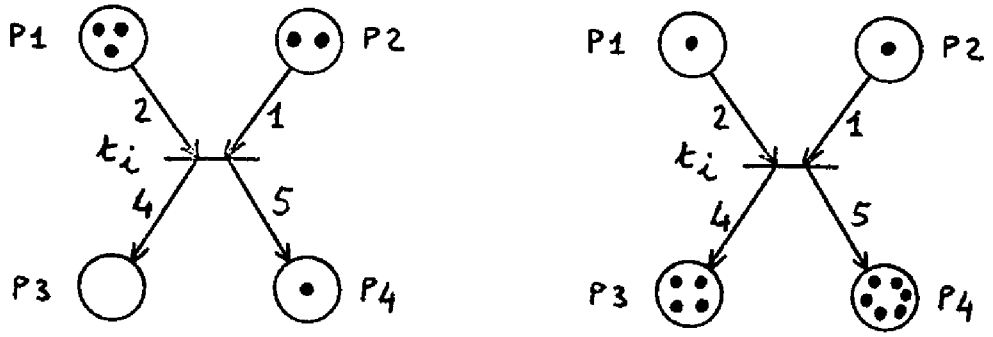
Définition I.18.

Les réseaux de PETRI généralisés sont définis comme les réseaux de PETRI, mais à chaque arc est associé un poids. Une transition ne sera sensibilisée que si chacune de ses places précédentes P_i possède au moins p_i jetons, p_i étant le poids de l'arc reliant la place à cette transition ; après le tir de la transition, les nombres de jetons enlevés aux places précédentes et ceux ajoutés aux places suivantes sont égaux aux poids des arcs correspondants.

Un exemple de tir de transition dans un réseau de PETRI généralisé est donné par la figure I.25.

Avant le tir de t_i (figure I.25.a.), le nombre des jetons dans P_1 et P_2 est supérieur au poids des arcs : t_i est donc sensibilisée.

Après le tir (figure I.25.b.), bien que P_1 et P_2 possèdent un jeton, t_i n'est plus sensibilisée.



a) b)

FIGURE I.25.

I.6.2. *Réseaux inhibiteurs*

Nous avons vu, jusqu'ici, des transitions sensibilisées par un nombre de jetons dans les places précédentes, égal à 1 (réseau de PETRI normal), ou supérieur à 1 (réseau de PETRI généralisé). Pourquoi ne pas envisager la possibilité que ce nombre soit nul ?

Cette solution a été proposée [AGE] : des arcs "inhibiteurs", étiquetés d'un zéro, et allant d'une place vers une transition, indiquent que cette place doit être vide de jetons pour que cette transition soit sensibilisée.

La figure I.26. montre des arcs inhibiteurs : la transition t_1 est sensibilisée, mais t_2 ne l'est pas car P'_2 n'est pas vide.

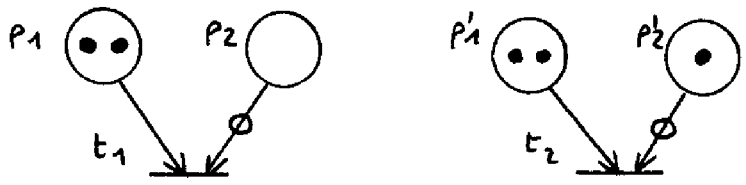


FIGURE I.26. Arcs inhibiteurs.

Définition I.19.

Les réseaux inhibiteurs sont définis comme des réseaux de PETRI généralisés possédant des arcs inhibiteurs.

Les réseaux inhibiteurs donnent la possibilité de représenter tout algorithme car il a été montré qu'ils étaient aussi généraux que les machines de TURING [HAC].

Inversement, il a été montré que les réseaux de PETRI sans arcsinhibiteursne pouvaient pas représenter tout algorithme [KOS]. En effet, il n'est pas possible de modéliser le test à zéro d'une file d'attente (ou d'un stock) dont la valeur n'est pas bornée.

I.6.3. Exemple

Dans les figures I.27. sont données différentes solutions pour la représentation de file d'attente et de test à la valeur zéro du stock.

La figure I.27.a. présente le cas où le stock est limité à une quantité de trois objets, et où le réseau de PETRI marqué est tel qu'aucune place ne contient plus d'un jeton à la fois.

Si le stock n'était pas borné, la configuration élémentaire décrite par la figure I.27.b. devrait être reproduite une infinité de fois. Le réseau obtenu ne serait plus un réseau de PETRI car le nombre de places et de transitions ne serait plus fini.

La figure I.27.c. représente le cas où le stock est limité à une quantité de trois objets, et où le réseau de PETRI utilisé est un réseau de PETRI généralisé (il comprend deux arcs de poids "3"). Si le stock n'était pas borné, le marquage de la place P_2 devrait avoir la valeur "infini", ce qui est exclu par la définition I.1., et les poids des deux arcs généralisés devraient être également "infinis" ce qui est impossible.

Dans le cas où les arcs inhibiteurs sont autorisés, on obtient la représentation de la figure I.27.d., pour laquelle la valeur du stock n'est pas bornée.

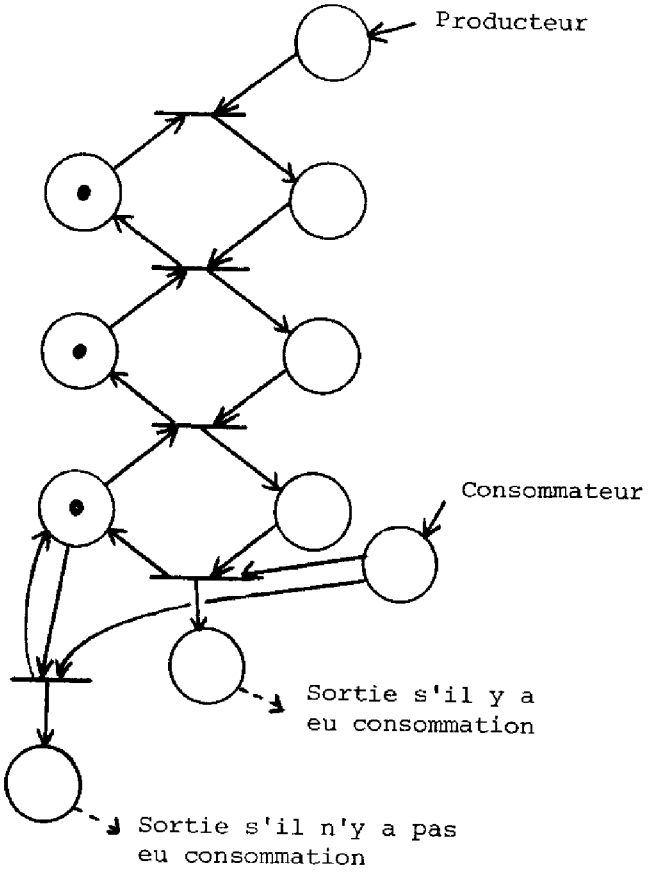


FIGURE I.27.a.

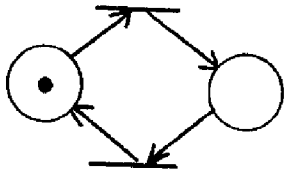


FIGURE I.27.b.

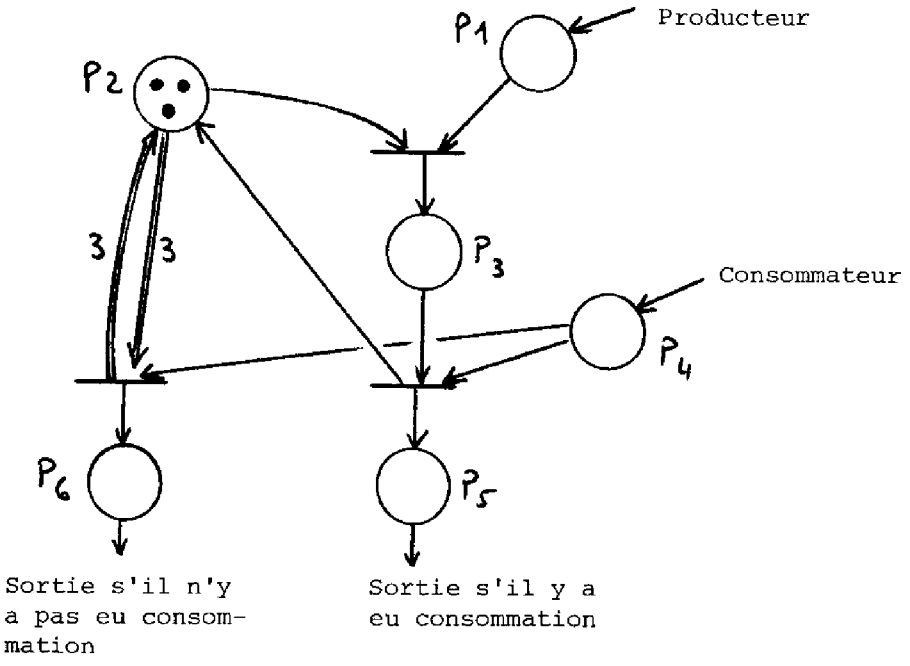


FIGURE I.27.c.

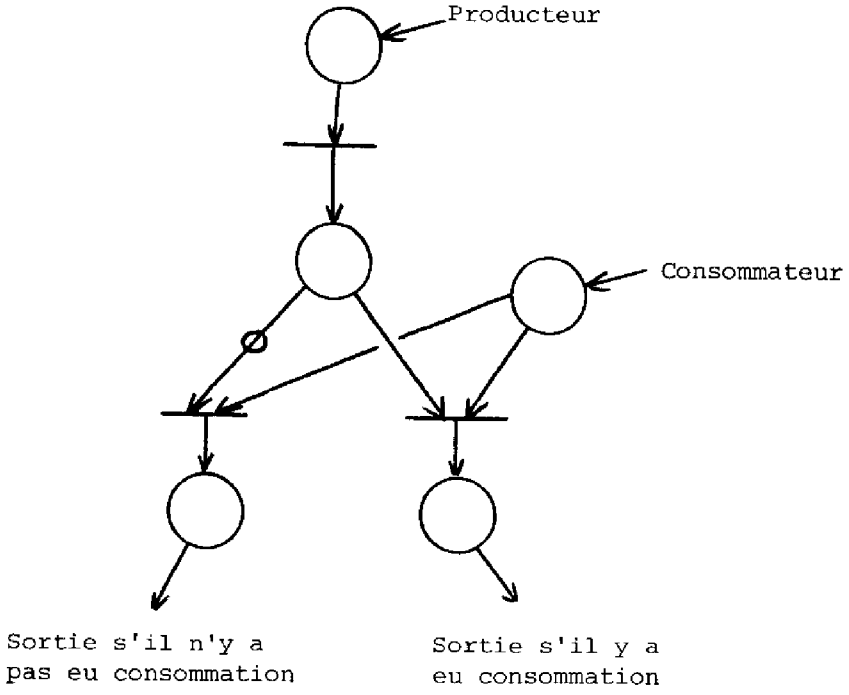


FIGURE I.27.d.

I.6.4. Conclusion

Nous voyons donc que plus le réseau de PETRI est généralisé, plus la représentation peut être concise. Malheureusement, alors que l'analyse des graphes d'événements et des machines à états est relativement simple, elle devient nettement plus complexe pour les réseaux libre choix et pour les réseaux quelconques et très difficile pour les réseaux inhibiteurs, d'où la difficulté de trouver un juste milieu entre un modèle suffisamment simple pour être analysable, et un modèle donnant une représentation assez concise. C'est ce que nous allons proposer maintenant.

I.7. LE SCHEMA A RESEAU DE PETRI

I.7.1. Représentation d'un système de commande

Un système de commande peut lire, à tout instant, des données fournies par le monde extérieur : des mesures provenant de capteurs, par exemple, ou des ordres donnés par la personne qui surveille le système. A partir de ces données, il effectue certaines opérations, qui peuvent être successives ou simultanées, en tenant compte de conditions, tant internes qu'extérieures.

Ensuite, il renvoie des informations au monde extérieur : soit directement au système commandé (actionneurs), soit des signaux d'alarmes, de visualisation, des comptes-rendus (figure I.28).

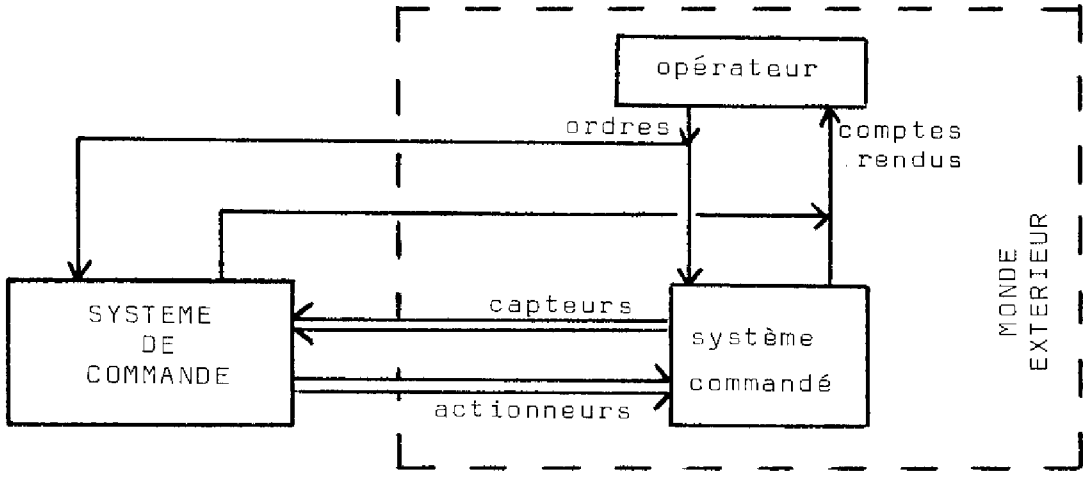


FIGURE I.28.

On peut décrire le fonctionnement d'un système de commande par un réseau de PETRI. Les sommets du réseau doivent représenter les opérations d'une part, les conditions d'autre part.

On associe à chaque transition du réseau une expression logique. Cette expression, appelée prédicat, s'exécute en permanence : elle prend la valeur "vrai" ou "faux" selon les données fournies par le monde extérieur ou évaluées de façon interne.

Définition I.20. : transition validée

Lorsque la condition logique associée à une transition a la valeur "vrai", nous disons que cette transition est validée.

Pour que la transition puisse être tirée, il faut qu'elle soit à la fois sensibilisée et validée.

Pour représenter les actions ou opérations, deux conventions différentes sont utilisées :

1. La première consiste à associer à chaque transition une opération. Les transitions sont ainsi munies d'étiquettes constituées chacune d'un prédicat et d'une opération.

Le tir de la transition équivaut à l'exécution de l'action, puis à la mise d'un jeton dans toutes les places suivantes.

2. Avec la deuxième convention, on associe une opération à chaque place [DAC].

Le tir d'une transition consiste alors à marquer d'un jeton toutes les places suivantes, puis à exécuter les actions affectées à ces places.

Le comportement du réseau est identique dans les deux cas, mais selon les systèmes étudiés, il peut être plus intéressant d'utiliser une méthode plutôt que l'autre. Il y a une certaine analogie de fonctionnement entre ces deux conventions d'une part et les machines de MEALY et de MOORE d'autre part.

Les programmes que nous avons écrits, et que nous verrons au troisième chapitre, réalisant la description, l'analyse et la simulation des réseaux, prévoient les deux possibilités.

Il est généralement très facile de passer d'une représentation à l'autre. En effet, une transition à laquelle est associée une action peut se représenter par une place entourée de deux transitions, marquant le début et la fin de l'action. La figure I.29. montre cette équivalence.

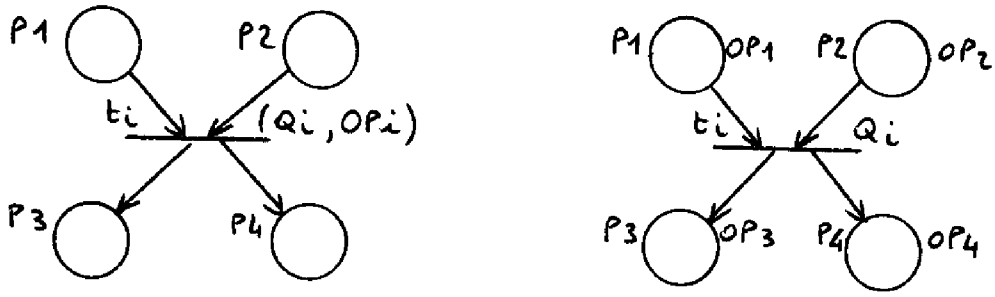


FIGURE I.29.a. Deux conventions de représentation.

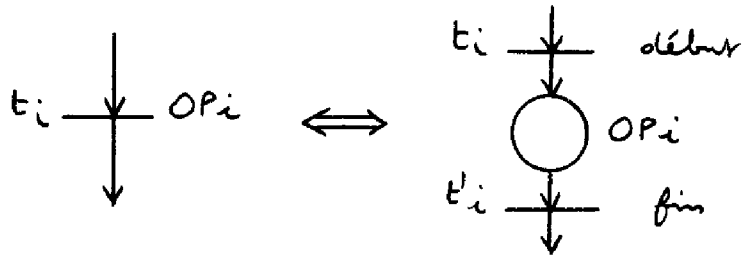


FIGURE I.29.b.

REMARQUES

Il est possible qu'une condition logique Q_i associée à une transition t_i ait toujours la valeur "vrai". L'expression associée s'écrira "1". La transition sera alors tirée dès qu'elle sera sensibilisée.

Il est aussi possible qu'aucune action ne soit associée à une place ou à une transition. L'action sera alors dite "vide".

I.7.2. Opérateurs et cellules-mémoires

Chacune des actions ou opérations met en jeu un opérateur et des cellules mémoires, dans lesquelles sont stockées les valeurs servant au déroulement de l'opération.

Un opérateur O_i lit un ensemble de mémoires $E(O_i)$, mémoires d'entrée de l'opérateur, et écrit dans d'autres mémoires, $S(O_i)$, mémoires de sortie de cet opérateur. On peut représenter par un graphe les différentes relations entre les opérateurs et les mémoires. Ce graphe représente le matériel de façon abstraite et il décrit le cheminement des informations à travers ce matériel. Nous l'appellerons "graphe de données".

Définition I.21. : graphe de données

Le graphe de données est composé des éléments suivants [VAL.1] :

- les opérateurs O_i représentés par des cercles,
- les cellules mémoires m_j représentées par des rectangles,
- les prédicats Q_k représentés par des losanges ; ces opérateurs particuliers s'exécutant en permanence, leur valeur de sortie, booléenne, est disponible à tout instant pour le graphe de commande;
- les cellules d'entrée-sortie du système représentées par des rectangles avec une flèche entrante ou/et sortante. Ce sont les mémoires particulières contenant les données fournies (capteurs) ou lues (actionneurs) par le monde extérieur.

Le graphe de données possède trois types d'arcs :

- (m_j, O_i) est un arc si et seulement si $m_j \in E(O_i)$,
- (O_i, m_k) est un arc si et seulement si $m_k \in S(O_i)$,
- (m_i, Q_k) est un arc si et seulement si $m_i \in E(Q_k)$.

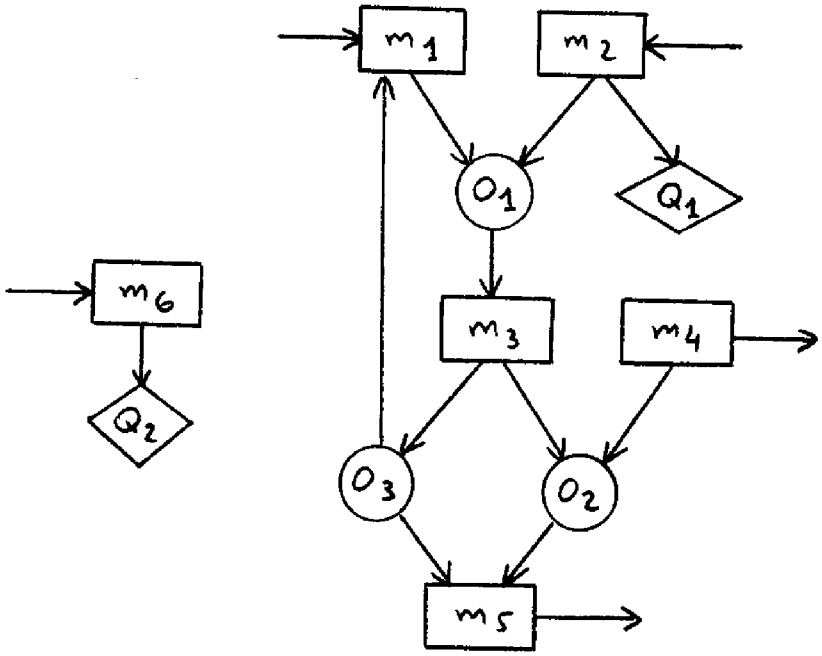


FIGURE I.30. Exemple de graphe de données.

I.7.3. Schéma à réseau de PETRI

Nous disposons finalement d'un réseau de PETRI étiqueté et muni d'un marquage initial, ainsi que d'un graphe de données.

Pour représenter complètement un système bien particulier, il faut donner une interprétation, un sens physique à chaque élément du graphe de données, c'est-à-dire préciser l'opération exacte réalisée par chaque opérateur, et les valeurs initiales des contenus des mémoires. Il faut aussi préciser la fonction booléenne attachée à chaque transition.

Le réseau de PETRI étiqueté constitue un "graphe de commande".

C'est l'ensemble formé par le graphe de commande et le graphe de données qui constitue le schéma à réseau de PETRI. Ce schéma est accompagné d'une interprétation.

I.7.4. Exemple

Sur la figure I.31., nous avons représenté le problème du test à la valeur zéro d'un stock non borné, problème que nous avons vu en détail, au paragraphe I.6.3., pour illustrer divers niveaux de généralisation des réseaux de PETRI.

Ici, le stock est décrit par la mémoire S du graphe de données. Nous voyons qu'il est alors possible de représenter le problème par un schéma à réseau de PETRI dont le graphe de commande est un réseau de PETRI non généralisé et sans arcs inhibiteurs.

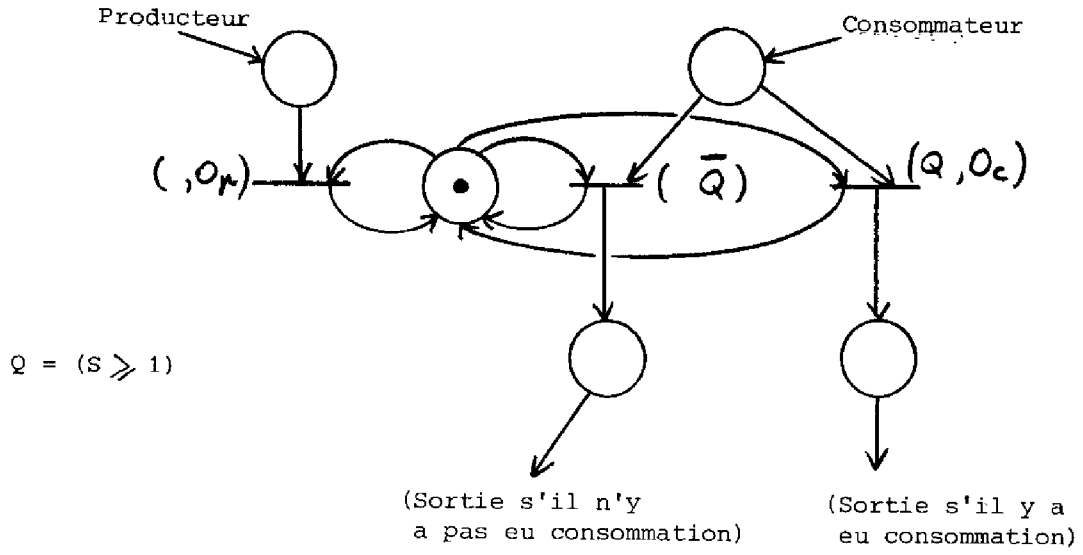
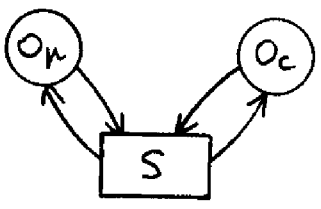


FIGURE I.31.a. Graphe de commande.



O_p : S ← S + 1

O_c : S ← S - 1

FIGURE I.31.b. Graphe de données.

I.8. CONCLUSION

Dans ce premier chapitre, nous venons d'étudier le schéma à réseau de PETRI. Ce modèle de représentation des systèmes de commande est constitué de deux graphes : un graphe de commande et un graphe de données, plus une interprétation.

Le graphe de commande est un réseau de PETRI muni d'un marquage initial. On peut considérer les réseaux de PETRI comme une extension des automates à états finis, avec la différence cependant que les sommets du graphe de l'automate représentent un état précis du système, tandis que les places du réseau de PETRI sont seulement des états partiels où le système peut être réceptif à certains événements.

Nous avons donné les éléments définissant le réseau et ceux nécessaires à son étude, en les illustrant de quelques exemples. Nous avons expliqué la signification et l'évolution de son marquage, décrivant le fonctionnement du système représenté par le réseau. Aux sommets du graphe sont associées les opérations effectuées pendant le fonctionnement de ce système.

Selon les conventions choisies et le niveau de généralisation de ce réseau de PETRI, la puissance de la représentation est plus ou moins grande. Certaines catégories de réseaux permettent une modélisation plus concise, mais cet intérêt est souvent perdu lorsqu'il s'agit d'effectuer leur analyse.

C'est pour cette raison qu'il a été choisi de structurer un système en deux parties : partie "commande" et partie "données" [VAL.1].

La constitution du schéma à réseau de PETRI ainsi obtenue nous permettra d'analyser le comportement d'un système.

CHAPITRE II

ANALYSE DES

SCHÉMAS À RÉSEAUX DE PETRI

-

II.1. INTRODUCTION

Lors de la conception d'un système de commande, l'analyse de sa structure est une partie importante de la vérification du système.

La modélisation a permis au concepteur d'exprimer son problème et de représenter une solution de celui-ci ; elle doit aussi servir à vérifier que le comportement de cette solution est correct. Le fonctionnement du système est-il bien celui que l'on attendait ? Le système ne se bloquera-t-il pas ? La réponse à ces questions sera déduite en partie des propriétés du schéma.

Nous allons donc voir, dans ce deuxième chapitre, comment procéder à l'analyse de ces schémas à réseaux de PETRI.

Nous étudierons tout d'abord les propriétés concernant seulement le réseau de PETRI qui constitue le graphe de commande. Il sera relativement aisé de savoir si le réseau est sauf, ou borné, mais déterminer s'il est vivant posera plus de problèmes : nous rechercherons s'il est propre avant de conclure qu'il est ou n'est pas vivant.

Puis, nous ferons intervenir le graphe de données pour rechercher si le schéma est déterminé et déterministe.

Nous définirons chacun des concepts, nous préciserons sa signification pour le comportement du système représenté ; puis nous indiquerons les différents algorithmes d'étude de ces propriétés.

II.2. PROPRIETES D'UN RESEAU DE PETRI

II.2.1. Réseau borné pour un marquage initial donné

Définition II.1.

Si $\mathcal{P}$ est un réseau de PETRI et M_o le marquage initial, $\mathcal{P}$ est dit borné pour M_o si et seulement si pour tout marquage M_i de la classe des marquages conséquents $\overrightarrow{M_o}$, aucune place du réseau ne contient plus de n jetons, n étant un entier positif donné.

EXEMPLE : La figure II.1. montre un réseau de PETRI non borné : on peut constater que la séquence de transitions (t_1, t_2) est tirable une infinité de fois successives : la place P_3 peut ainsi recevoir une infinité de jetons.

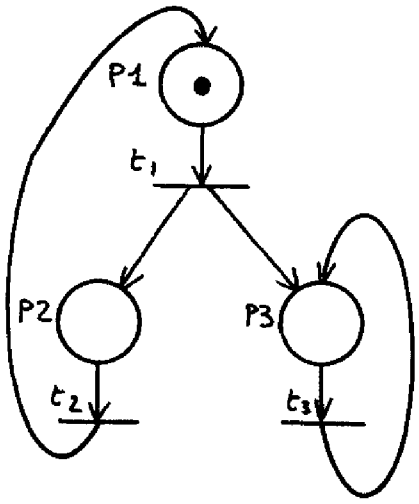


FIGURE II.1.

THEOREME II.1.

Un réseau de PETRI $\mathcal{P}$ est borné pour un marquage initial M_o si et seulement si la classe des marquages conséquents $\overrightarrow{M_o}$ est de dimension finie.

Ce théorème est une conséquence directe du théorème I.1. concernant l'ensemble des vecteurs accessibles à partir d'un système d'addition de vecteurs.

II.2.2. Réseau sauf pour un marquage initial

Définition II.2.

Si $\mathcal{P}$ est un réseau de PETRI et M_o son marquage initial, $\mathcal{P}$ est dit sauf pour M_o si et seulement si pour tout marquage M_i appartenant à la classe $\overrightarrow{M_o}$, chaque place de $\mathcal{P}$ contient au plus un jeton.

Ceci correspond à un réseau borné avec n = 1.

EXEMPLE : Reprenons, à la figure II.2., le réseau de PETRI de la figure I.12., ainsi que les trois classes des marquages conséquents obtenues pour différents marquages initiaux (figures I.13., 14 et 15, page 23). Ce réseau est sauf pour le marquage initial $M_o=(1,0,0,0,0)$, mais il ne l'est plus ni pour $M'_o=(0,0,1,0,1)$: il y a deux jetons dans les places p_3 et p_5 après le tir des transitions t_5 ou t_4, ni pour le marquage $M''_o=(1,0,1,0,0)$.

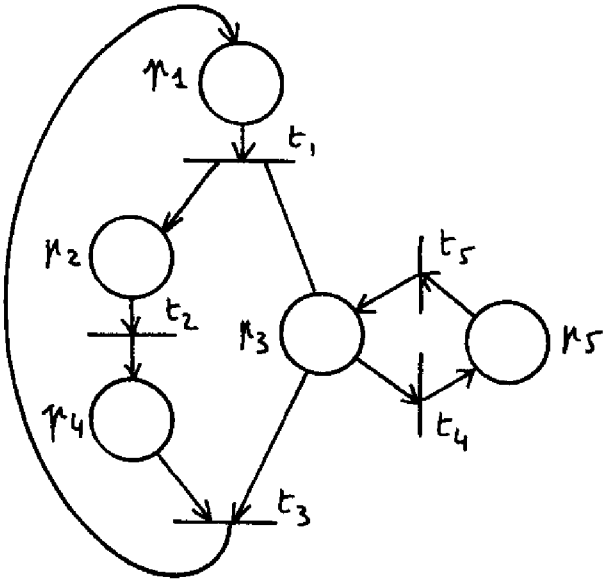


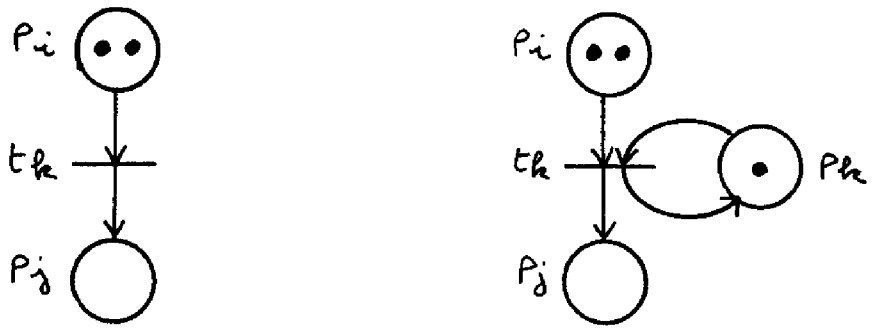
FIGURE II.2.

Lors de l'analyse systématique des schémas à réseaux de PETRI à l'aide de programmes informatiques, nous n'étudierons pas si le graphe de commande est un réseau borné. Nous nous limiterons à vérifier que ce graphe est un réseau de PETRI sauf.

Dans le cas où les opérateurs sont associés aux places, l'arrivée d'un deuxième jeton dans une place équivaut à solliciter un opérateur alors qu'il est déjà activé. C'est pratiquement la même chose dans le cas où l'opérateur est associé à une transition : si le réseau n'est pas sauf, cette transition pourra rester sensibilisée alors qu'elle est en train d'être tirée ; donc ici aussi un opérateur activé pourra être à nouveau sollicité.

Le fait que le réseau soit sauf implique que la description ne comportera pas ces ambiguïtés. Néanmoins, il faut remarquer qu'il existe d'autres possibilités pour empêcher qu'une transition soit à la fois sensibilisée et en train d'être tirée : par exemple, on peut rajouter une boucle élémentaire.

La transition de la figure II.3.a. est deux fois sensibilisée ; après l'introduction de la place P_k, figure II.3.b., le fonctionnement du réseau cesse d'être ambigu et t_k sera tirée deux fois successivement.



a) b)

FIGURE II.3.

Imposer au réseau de PETRI d'être sauf est en fait, plutôt, une position de principe. Les places représentent des conditions internes à la partie commande,qui sont des conditions logiques soit vraies, soit fausses. Tout compteur doit être rejeté dans la partie données. Dans ce cas, il est clair que le fait que le graphe de commande soit un réseau de PETRI non sauf provient d'une erreur de conception et révèle une description non cohérente où une condition logique déjà satisfaite serait vérifiée à nouveau.

Enfin, un autre intérêt de représenter un système par un réseau de PETRI sauf réside dans le fait que pour une réalisation pratique, il sera facile, par exemple, de matérialiser chaque place par une bascule : les valeurs 0 et 1 correspondant alors respectivement à l'absence et à la présence d'un jeton.

Pour ces diverses raisons, de nombreux travaux préconisent l'usage exclusif de réseaux de PETRI saufs [PAT - MIS].

II.2.3. Réseau vivant pour un marquage initial

Définition II.3.

Un réseau de PETRI $\mathcal{P}$ est dit vivant pour un marquage initial M_o si et seulement si pour toute transition t de $\mathcal{P}$, et pour tout marquage M_i appartenant à $\overrightarrow{M_o}$, il existe une séquence de tir comprenant t et tirable à partir de M_i.

EXEMPLE : Examinons à nouveau le réseau de la figure II.2. Ce réseau est vivant pour les marquages initiaux (1,0,0,0,0) ou (1,0,1,0,0). Mais il ne l'est plus pour M_o=(0,0,1,0,1). En effet, dans ce dernier cas, il n'existe pas de séquence de tir comprenant t_1, ni t_2, ni t_3. De même, pour le marquage initial M_o=(1,0,0,0,0), le réseau ne serait plus vivant si l'on supprimait l'arc allant de la transition t_3 à la place p_1 : la transition t_1 ne serait plus tirable à partir d'un marquage autre que M_o.

Le fait que le graphe de commande soit un réseau vivant est intéressant pour l'étude d'un schéma car cela correspond à une absence de blocage, c'est-à-dire de situation telle qu'aucune transition n'est tirable. D'autre part, si le réseau est vivant, on est sûr que toute partie de la commande est accessible.

Si un réseau de PETRI est borné pour un marquage initial, il existe un algorithme permettant de savoir si ce réseau est vivant pour ce marquage [RAM]. Mais, pour simplifier l'analyse, et surtout pour réduire le temps de calcul, nous n'étudierons pas cette propriété seule : nous chercherons à savoir si le réseau du graphe de commande est "vivant et propre".

II.2.4. Réseau propre pour un marquage initial

Définition II.4.

Un réseau est dit propre pour un marquage initial M_o si et seulement si pour tout marquage M_i appartenant à $\overrightarrow{M_o}$, il existe une séquence

S de tirs de transitions de $\mathcal{P}$ telle que $M_i \xrightarrow{S} M_o$, c'est-à-dire ramenant au marquage initial à partir de M_i.

EXEMPLE : Le réseau de PETRI de la figure II.2. est propre pour les trois marquages initiaux différents que nous avons étudiés (d'après les graphes des marquages conséquents, figures I.13., 14 et 15, page 23).

Pour l'analyse du schéma à réseau de PETRI, nous considèrerons que le graphe de commande doit être propre ; s'il ne l'est pas, cela traduit une erreur de conception. En effet, un système de commande doit pouvoir fonctionner de manière répétitive : lorsqu'il a terminé la tâche pour laquelle il a été conçu, il doit être prêt à la recommencer. Le marquage initial représentant l'état initial de la commande, il faut donc qu'à partir de tout état, on puisse revenir à ce marquage initial : autrement dit, il est nécessaire que le réseau soit propre.

II.2.5. Remarques

a) Réseau de PETRI vivant et non propre

Un réseau de PETRI peut être vivant pour un marquage initial donné, bien qu'il ne soit pas propre pour ce même marquage.

C'est le cas par exemple du réseau de la figure II.4.a., dont le graphe des marquages conséquents est donné par la figure II.4.b. On peut remarquer, en effet, que le marquage initial M_o n'est plus accessible à partir d'un quelconque de ses successeurs. Pourtant, la boucle formée par M_1, M_2 et M_3 dans le graphe montre que chacune des trois transitions peut être tirée à partir de chacun des marquages atteints, donc que le réseau est vivant.

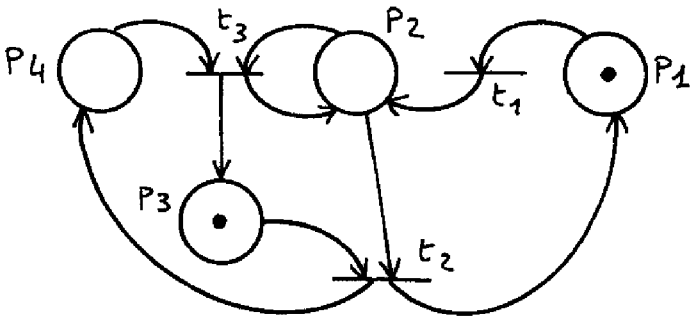
b) Réseau de PETRI propre et non vivant

Inversement, un réseau de PETRI peut être propre mais non vivant pour un marquage initial donné. C'est le cas du réseau de la figure II.5.a. où la transition t_5 n'est jamais sensibilisée. Le graphe des marquages conséquents de la figure II.5.b. montre que le réseau est propre.

Réseau vivant

et non propre

a)



M_o = (P1,P3)
M_1 = (P2,P3)
M_2 = (P1,P4)
M_3 = (P2,P4)

b)

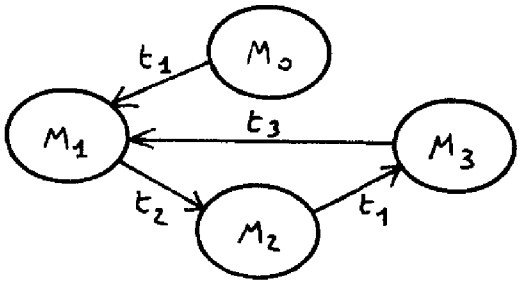
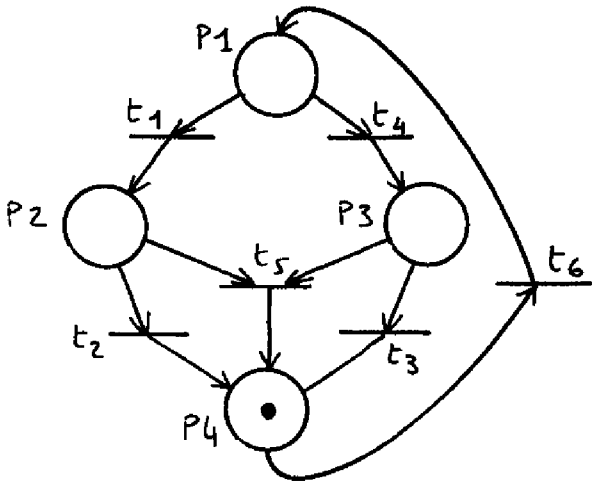


FIGURE II.4.

Réseau propre

et non vivant

a)



M_o = (P4)
M_1 = (P1)
M_2 = (P2)
M_3 = (P3)

b)

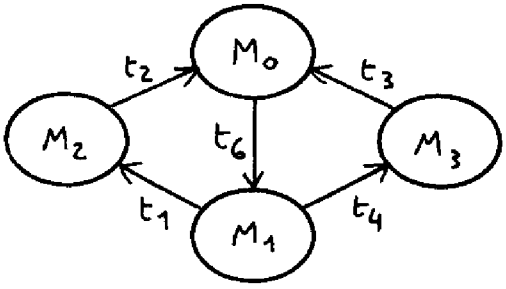


FIGURE II.5.

II.3. ANALYSE DU GRAPHE DE COMMANDE

Nous allons ici préciser les procédures d'analyse utilisées pour les programmes en langage APL. Ces programmes seront détaillés dans le prochain chapitre.

II.3.1. Réseau de PETRI sauf

L'algorithme déterminant si un réseau de PETRI marqué, c'est-à-dire muni d'un marquage initial, est borné, consiste à rechercher si le graphe des marquages conséquents est fini ou non (d'après le théorème II.1.).

La solution consiste donc à construire l'arbre des marquages obtenus à partir du marquage initial, et, pendant cette construction, à observer si un marquage est "supérieur ou égal" à l'un de ses prédécesseurs. Nous appelons un marquage M' "supérieur" à un autre marquage M, si chaque place du réseau contient pour M' au moins autant de jetons que pour M, l'une au moins en contenant plus pour M' que pour M : par exemple, le marquage (2,1,0,1,0) est supérieur à (2,0,0,1,0), mais n'est pas supérieur à (1,2,0,1,0).

Lorsqu'un marquage supérieur à l'un de ses prédécesseurs est rencontré, on peut arrêter la construction de l'arbre, car l'on est sûr que le réseau ne peut être borné. Si l'on rencontre un marquage égal à un de ses "pères", il est inutile de continuer cette branche de l'arbre, et l'on s'arrête aussi. On continue lorsque le marquage est "inférieur". Comme le nombre des marquages inférieurs à un marquage initial donné est un nombre fini, la procédure se terminera toujours.

Si, initialement, chaque place contient au plus un jeton, et si l'on cherche à savoir si le réseau est sauf, nous pouvons arrêter la procédure dès qu'une place contient deux jetons, et conclure immédiatement. C'est précisément ceci que réalisera la procédure d'analyse automatique que nous avons mise au point.

II.3.2. Réseau de PETRI vivant

Il n'a pas pu être montré si le problème suivant était soluble ou non :

"Etant donné un réseau de PETRI quelconque, est-il vivant ou non pour un marquage initial donné ? ".

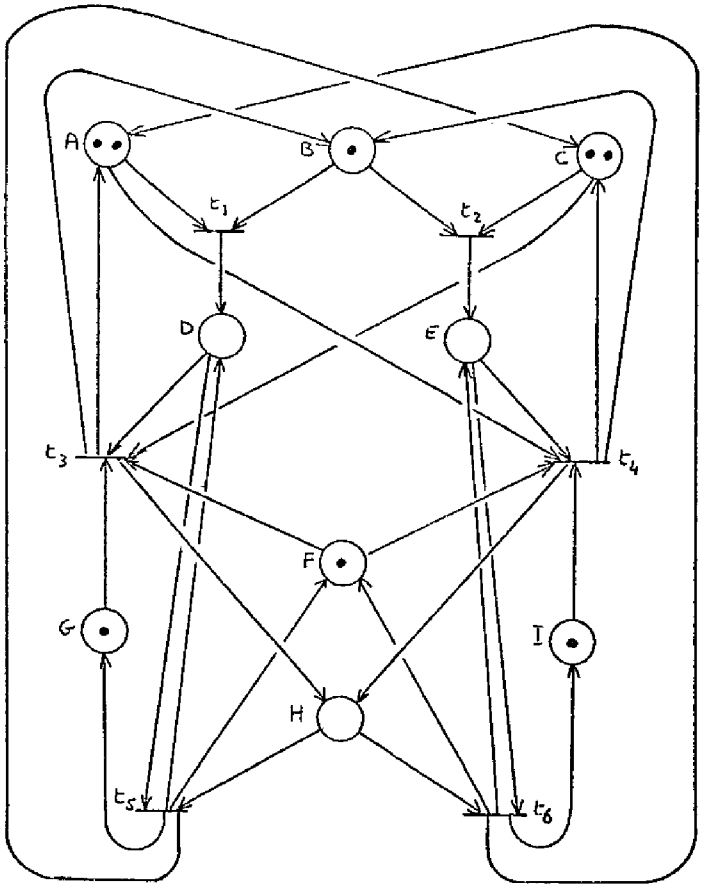
Cela signifie qu'il n'existe aucun algorithme permettant de répondre à cette question dans le cas général. Par contre, si le réseau de PETRI est borné pour le marquage initial considéré, alors un tel algorithme existe [RAM - BER].

Il consiste essentiellement à rechercher les différentes composantes fortement connexes du graphe des marquages conséquents et à vérifier que toute transition du réseau étiquette au moins un arc de chacune de ces composantes fortement connexes.

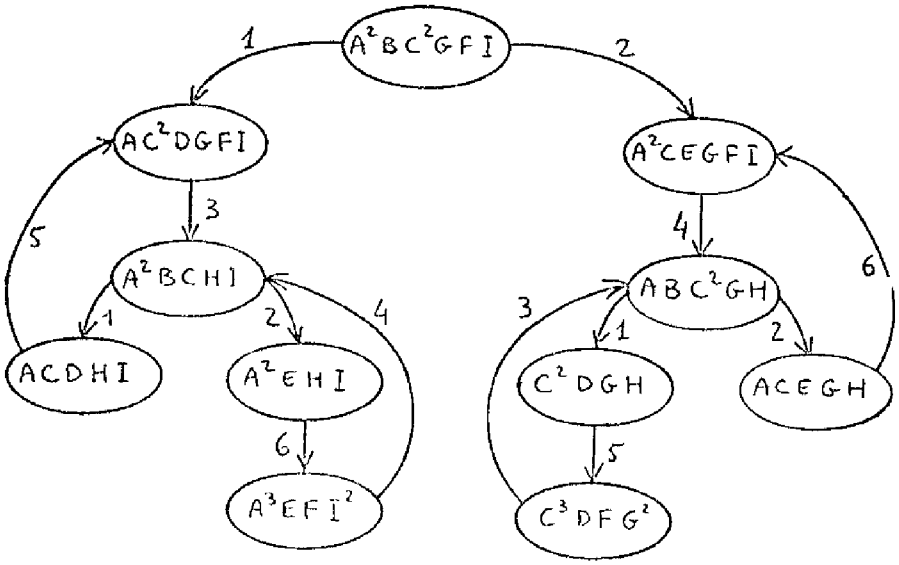
Dans le cas général d'un réseau de PETRI borné, cet algorithme peut être très lourd, car il est tout à fait possible, même dans le cas d'un réseau relativement simple, d'avoir un réseau de PETRI vivant pour un marquage initial et dont le graphe des marquages conséquents comporte plusieurs composantes fortement connexes.

L'exemple nous en est donné par le réseau de PETRI de la figure II.6.a. Le graphe des marquages conséquents (figure II.6.b.) fait apparaître deux composantes fortement connexes pendantes et pourtant le réseau est bien vivant pour son marquage initial puisque chacune des six transitions du réseau apparaît une fois dans chacune des composantes fortement connexes.

Il vient tout naturellement à l'esprit de simplifier cet algorithme en cherchant à restreindre son domaine d'application. Nous avons vu au paragraphe II.2.4. que dans le cas particulier où le système modélisé était un système de commande, le fait que le réseau de PETRI ne soit pas propre pour son marquage initial impliquait une erreur de conception. Compte tenu également du paragraphe II.2.2., l'algorithme d'analyse que nous utiliserons vérifie directement que le réseau de PETRI utilisé comme graphe de commande est un réseau à la fois sauf, propre et vivant pour un marquage initial donné.



a)



b)

FIGURE II.6.

II.3.3. Procédure d'analyse

La procédure d'analyse du graphe de commande d'un schéma à réseau de PETRI sera basée sur deux théorèmes.

THEOREME II.2.

Un réseau de PETRI borné pour un marquage initial M_o est propre pour ce marquage M_o si et seulement si le graphe des marquages conséquents $G(\overrightarrow{M_o})$ est fortement connexe.

On peut à ce moment-là utiliser un deuxième résultat, permettant de savoir si un réseau borné pour un marquage est vivant pour ce marquage.

THEOREME II.3.

Un réseau de PETRI borné et propre pour un marquage initial M_o est également vivant pour M_o si et seulement si chaque transition du réseau est au moins une fois étiquette d'un arc du graphe des marquages conséquents $G(\overrightarrow{M_o})$.

L'analyse commence par la recherche de la classe des marquages conséquents. Le passage d'un marquage à un autre correspond au tir d'une transition ; autrement dit, chaque arc de l'arbre des marquages reçoit comme étiquette une transition.

Pendant la construction de cet arbre, il est vérifié que le réseau est sauf pour son marquage initial. Ensuite, nous nous assurons que toute transition du réseau est étiquette au moins une fois d'un arc du graphe des marquages.

Dans le cas d'une réponse négative, on peut directement conclure et affirmer que le réseau de PETRI n'est pas vivant. Mais si le résultat est positif, nous ne pouvons rien en déduire, ne sachant pas si le réseau est propre. C'est ce qu'il faut examiner dans un deuxième temps.

Le programme d'analyse étudie si le graphe des marquages conséquents est fortement connexe. Si ce deuxième résultat est aussi positif, nous pouvons alors appliquer successivement les deux théorèmes II.2. et II.3. et en conclure que le graphe de commande est un réseau de PETRI vivant et propre.

Nous pouvons résumer l'analyse du graphe de commande par la figure II.7.

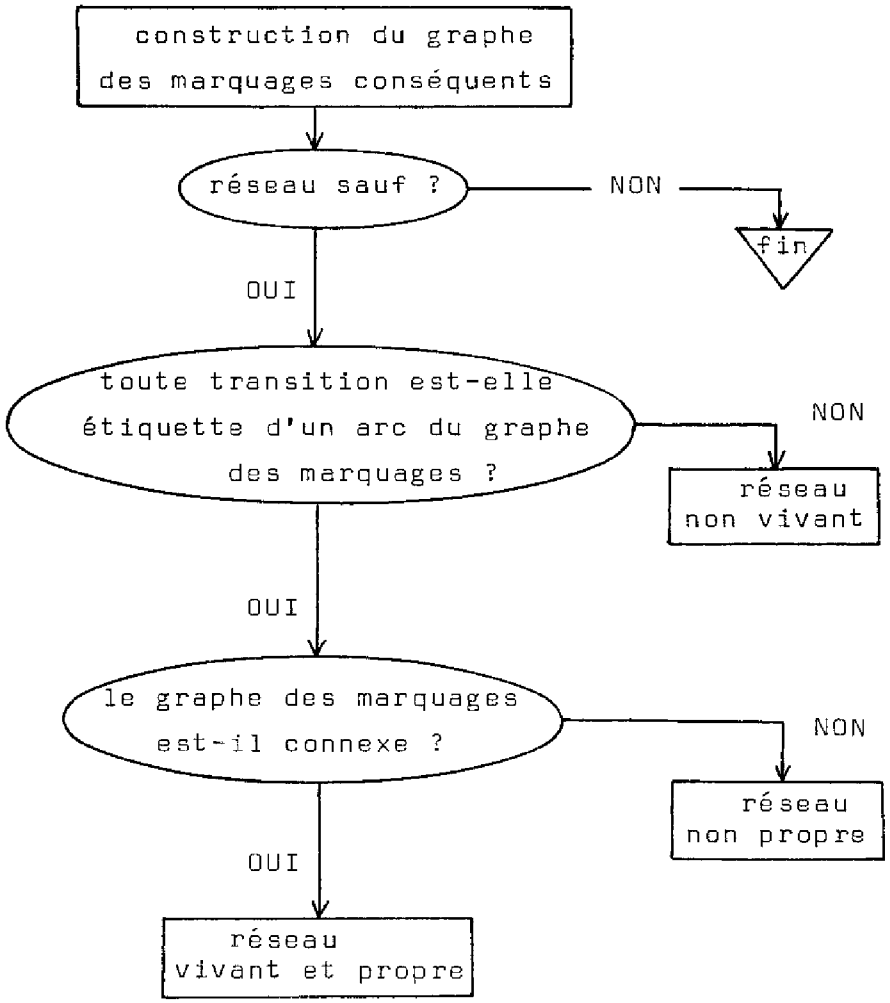


FIGURE II.7.

II.4. PROPRIETES ET ANALYSE DU SCHEMA A RESEAU DE PETRI

Nous venons de définir et d'étudier les propriétés intrinsèques du réseau de PETRI constituant le graphe de commande. Maintenant, si nous faisons intervenir le graphe de données du schéma, deux nouveaux concepts apparaissent : schéma déterminé et schéma déterministe. Ces deux notions nécessitent quelques nouvelles définitions préliminaires.

II.4.1. Schéma à réseau de PETRI déterministe

Définissons tout d'abord ce qu'est un conflit dans un réseau de PETRI.

Définition II.5.

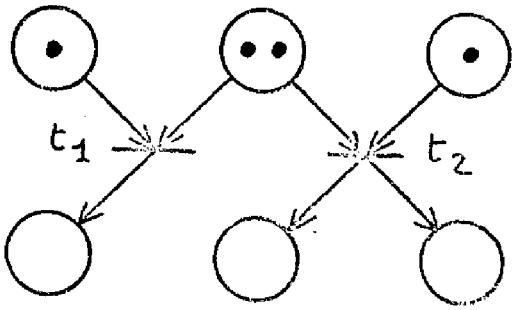
Soit $\mathcal{P}$ un réseau de PETRI muni d'un marquage initial M_o, et M_k un marquage appartenant à la classe des marquages conséquents $\overrightarrow{M_o}$.

Deux transitions de $\mathcal{P}$, t_i et t_j, sont dites en conflit pour M_k si et seulement si les deux conditions suivantes sont vérifiées :

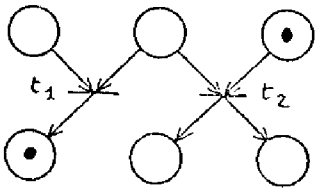
- 1 - t_i et t_j sont sensibilisées par ce marquage M_k,
- 2 - le marquage obtenu à partir de M_k en enlevant un jeton à toutes les places précédentes de t_i (respectivement de t_j) est tel que la transition t_j (respectivement t_i) n'est plus sensibilisée.

Pour que deux transitions soient en conflit, il faut donc qu'elles aient au moins une place précédente commune et que cette place ne contienne pas assez de jetons pour permettre les tirs simultanés de t_i et de t_j. La figure II.8. montre des exemples de conflits. Mais les transitions des figures II.9. ne sont pas en conflit.

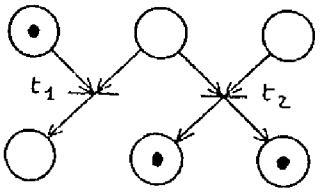
t_1 et t_2 sont en conflit :



Après le tir de t_1, t_2 n'est plus sensibilisée :

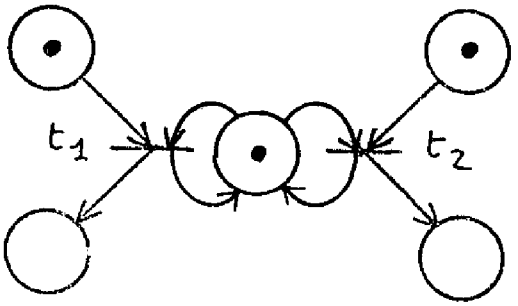


Inversement, après le tir de t_2, t_1 n'est plus sensibilisée :



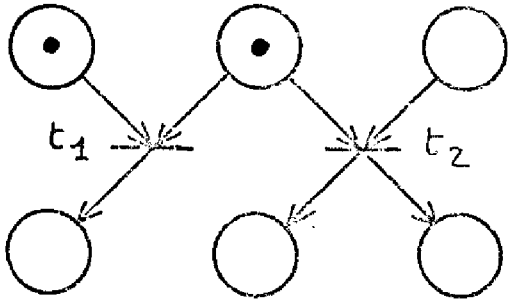
a)

Ici, t_1 et t_2 sont en conflit, mais il est possible de tirer successivement les deux transitions.



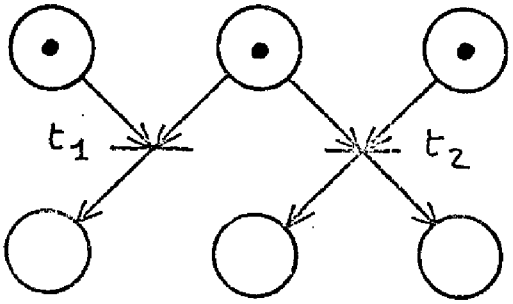
b)

FIGURE II.8.



Ici, t_2 n'est pas sensibilisée.
Pas de conflit.

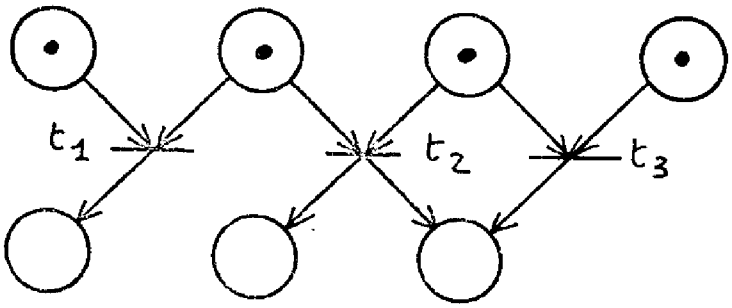
a)



Pas de conflit, car t_1 et t_2 peuvent être tirées simultanément.

b)

FIGURE II.9.



Il y a conflit entre t_1 et t_2, et entre t_2 et t_3, mais non entre t_1 et t_3.

FIGURE II.10.

Plus de deux transitions peuvent être en conflit. Mais si deux sont en conflit avec une même troisième, elles ne sont pas obligatoirement en conflit entre elles : c'est le cas de la figure II.10. La notion de conflit n'est donc pas transitive.

Supposons que deux transitions soient en conflit dans un réseau de PETRI non étiqueté. Rien n'indique de tirer une transition plutôt que l'autre. Un choix est alors fait de manière arbitraire et le réseau de PETRI ne sera pas déterministe, c'est-à-dire que les séquences de tirs de transitions ne seront pas déterminées uniquement par le marquage de départ.

Le fait qu'un réseau de PETRI ne soit pas déterministe, indique la présence de décisions à prendre.

Mais lorsque le réseau est étiqueté, les fonctions logiques associées aux transitions peuvent être telles qu'une seule des deux transitions en conflit soit tirable. On dit alors que le conflit est résolu. Pour cela, les fonctions logiques doivent être exclusives : à tout moment, une seule aura la valeur "vraie".

Un exemple très simple de conflit résolu est le cas où l'unedes fonctions logiques est la négation de l'autre : figure II.11.

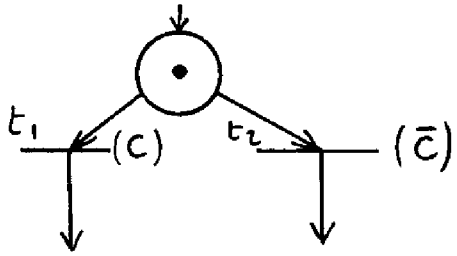


FIGURE II.11.

REMARQUE

Nous avons étudié au premier chapitre les différentes classes de réseaux de PETRI : nous pouvons noter que la présence d'un conflit est impossible dans un "graphe d'événements" ; il est, en effet, exclu que deux arcs partent d'une même place.

Nous pouvons maintenant donner la définition précise d'un schéma à réseau de PETRI déterministe.

Définition II.6.

Un schéma à réseau de PETRI est déterministe si et seulement si tous les conflits, pouvant apparaître dans le réseau de PETRI utilisé comme graphe de commande et muni d'un marquage initial M_o, sont résolus.

Pour tout marquage M_i de la classe $\overrightarrow{M_o}$, les fonctions logiques associées aux transitions en conflit doivent être exclusives.

Lorsque nous construisons l'arbre des marquages conséquents, pour analyser le graphe de commande, nous ne tenons pas compte des fonctions logiques associées aux transitions. Il ne nous est donc pas possible, à ce moment, d'étudier l'aspect déterministe du schéma à réseau de PETRI. Par contre, lorsque nous ferons une simulation du réseau, chaque transition aura reçu pour étiquette une fonction logique dont la valeur devra être vraie pour que cette transition soit tirable. C'est à ce moment là que nous pourrons savoir si les conflits sont résolus ou non.

Nous reviendrons sur ce point dans le prochain chapitre, lors de l'étude du programme de simulation.

II.4.2. Schéma à réseau de PETRI déterminé

a) Donnons tout d'abord une définition préliminaire portant sur le graphe de commande.

Définition II.7.

Soient deux transitions t_i et t_j d'un réseau de PETRI muni d'un marquage initial M_o. Ces deux transitions sont dites "sensibilisables en parallèle" s'il existe au moins un marquage M_k de la classe $\overrightarrow{M_o}$ tel que t_i et t_j sont toutes les deux sensibilisées et ne sont pas en conflit.

Nous pouvons remarquer que cette définition n'exclut pas que t_i et t_j puissent être la même transition : une transition t_i peut être sensibilisable en parallèle avec elle-même. Il faut évidemment, dans ce cas, que le réseau de PETRI ne soit pas sauf. Mais nous avons vu au paragraphe II.2.2., comment empêcher qu'une tâche soit effectuée deux fois simultanément : en rajoutant une boucle élémentaire sur une transition, comme sur la figure II.3.b., on empêche cette transition d'être sensibilisable en parallèle avec elle-même.

Considérons la figure II.12. : cette partie d'un réseau de PETRI nous montre des transitions sensibilisables en parallèle : il s'agit des transitions t_2 et t_3, qui pourront être tirées simultanément à partir du marquage (P_2,P_3), si les fonctions logiques qui leur sont associées sont vraies en même temps. Il en est de même pour les transitions t_3 et t_4 pour le marquage (P_2,P_4).

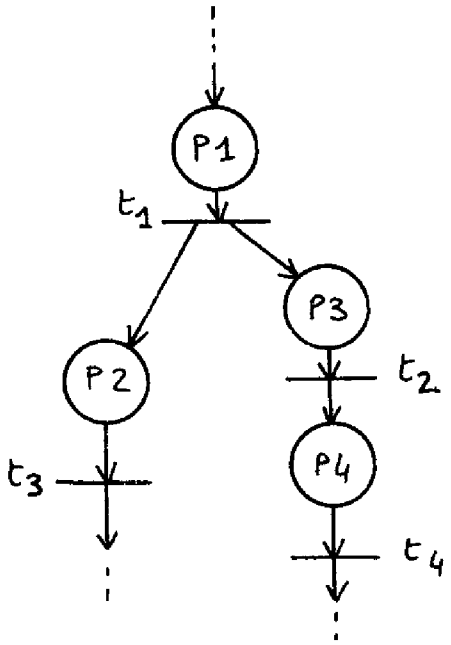


FIGURE II.12.

b) Schéma déterminé

Nous avons vu au premier chapitre, lors de la définition du graphe de données, que des actions sont exécutées par des opérateurs

travaillant sur des mémoires. Chaque opération op agit sur un ensemble de mémoires "d'entrée" : E(op), mémoires lues lors de l'opération, et sur un ensemble de mémoires de "sortie" : S(op), mémoires où sont écrits les résultats de cette opération.

Lorsque deux actions sont exécutées simultanément, il peut exister des incompatibilités : il est évident, en effet, qu'une opération op1 ne doit pas écrire un résultat dans une mémoire m, au moment même où une deuxième opération op2 lit la valeur du contenu de cette mémoire.

Egalement, deux opérations ne doivent pas ranger leurs résultats simultanément dans la même mémoire. Sinon, le résultat dépendrait de la durée de chaque opération.

RELATION ρ

Pour mettre en évidence les opérations qui ne peuvent s'exécuter en parallèle, une relation " ρ " d'incompatibilité est définie sur OP x OP,(OP étant l'ensemble des opérateurs), de la manière suivante :

Soit op1 et op2 deux éléments de OP ; le couple (op1, op2) vérifie la relation ρ si et seulement si l'une des trois conditions suivantes est vérifiée :

- 1 - $S(op1) \cap E(op2) \neq \emptyset$
- 2 - $S(op2) \cap E(op1) \neq \emptyset$
- 3 - $S(op1) \cap S(op2) \neq \emptyset$ et $op1 \neq op2$

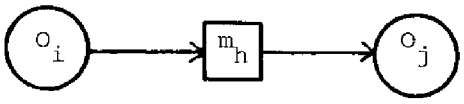
Cette relation est symétrique. Nous appellerons aussi ρ l'ensemble des couples d'opérations qui vérifient cette relation.

On peut considérer que chaque opération à exécuter pour représenter le fonctionnement du système active un et un seul opérateur. Le graphe de données du schéma à réseau de PETRI nous donne alors directement l'ensemble ρ, que l'on peut maintenant définir de la façon suivante :

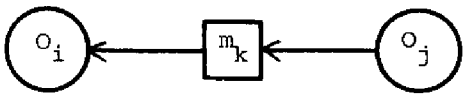
Définition II.8.

La relation ρ est l'ensemble des couples d'opérateurs (O_i,O_j) vérifiant l'une des trois conditions suivantes :

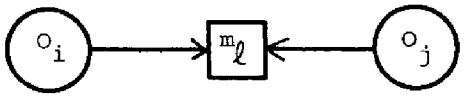
- 1. Il existe une mémoire m_h telle que les deux arcs (O_i,m_h) et (m_h,O_j) appartiennent au graphe de données.
(L'opérateur O_j lit dans la mémoire m_h où O_i écrit son résultat).



- 2. Il existe une mémoire m_k telle que (m_k,O_i) et (O_j,m_k) soient des arcs du graphe de données (O_i lit dans m_k où O_j écrit).



- 3. Il existe une mémoire m_ℓ telle que les arcs (O_i,m_ℓ) et (O_j,m_ℓ) soient des arcs du graphe de données (les deux opérateurs O_i et O_j écrivent dans la même mémoire m_ℓ).



Si i est la relation d'identité, nous appellerons $\tilde{\rho}$ la relation telle que $\tilde{\rho} = \rho \cup i$.

Nous pouvons maintenant donner la définition suivante :

Définition II.9.

Un schéma à réseau de PETRI est déterminé si et seulement si aucun des couples d'opérateurs pouvant être exécutés en parallèle n'appartient à l'ensemble $\tilde{\rho}$.

L'ensemble des opérateurs du graphe de données étant fini, le nombre des couples de l'ensemble $\tilde{\rho}$ sera obligatoirement fini lui aussi.

c) Recherche des opérateurs simultanément activés

Le problème est donc maintenant de rechercher quels sont ces opérateurs simultanément activés. La question sera résolue de deux façons légèrement différentes selon l'interprétation que l'on donne au système de commande.

Nous avons dit, en effet, dans le premier chapitre, que les actions à exécuter pouvaient être associées soit aux places du réseau, soit à ses transitions.

1. Considérons le premier cas : lorsqu'une place reçoit un jeton, l'opérateur associé est activé et l'opération exécutée. Pour un marquage donné, l'ensemble des opérateurs activés simultanément se déduit immédiatement de l'ensemble des places marquées.

L'algorithme à suivre pour rechercher si le schéma est déterminé consiste donc à étudier chacun des marquages de la classe des marquages conséquents. En prenant deux à deux les places marquées, il faut vérifier que leurs opérateurs associés n'ont pas une même mémoire de sortie, et qu'aucune mémoire d'entrée de l'un n'est mémoire de sortie de l'autre, autrement dit que le couple de ces deux opérateurs n'appartient pas à l'ensemble $\tilde{\rho}$.

Si le schéma à réseau de PETRI est tel que le graphe de commande est un réseau de PETRI borné, l'ensemble des marquages conséquents est fini ; le nombre de places du réseau l'est également. L'algorithme donne donc rapidement la solution à notre problème.

C'est cette procédure qui sera utilisée par notre programme d'analyse automatique.

2. Dans le deuxième cas, les opérateurs pouvant être activés simultanément sont ceux qui sont associés aux transitions sensibilisables en parallèle. Ici aussi, c'est la construction du graphe des marquages conséquents qui nous apporte la solution.

En effet, à partir de chaque marquage, la méthode de construction de l'arbre recherche toutes les transitions sensibilisées, et en établit une liste. L'ensemble des opérateurs activés simultanément est donné par toutes les transitions de cette liste.

Il s'agit donc d'étudier, à chaque niveau de construction du graphe des marquages, cette liste de transitions sensibilisées, en prenant deux à deux ces transitions, de la même façon que les places marquées dans le cas précédent.

Toutefois, s'il y a un conflit, résolu ou non, entre deux (ou plusieurs) transitions, il faut étudier successivement deux (ou plus) listes, chacune comportant une seule de ces transitions en conflit, plus toutes les autres. En effet, deux opérateurs associés à deux transitions en conflit ne peuvent être activés simultanément.

La construction du graphe des marquages conséquents comporte un nombre fini d'étapes ; le nombre des transitions étant également fini, cette procédure donne ici aussi rapidement la réponse à notre question.

En résumé, nous avons recherché les opérateurs activés simultanément, puis nous avons étudié si ces opérateurs vérifiaient la relation ρ . On peut remarquer qu'il est également possible d'utiliser la méthode inverse, c'est-à-dire de rechercher systématiquement tous les couples d'opérateurs de la relation ρ , par exemple en étudiant successivement chaque cellule-mémoire du graphe de données, puis de mettre en mémoire cet ensemble, et ensuite de comparer ce dernier avec l'ensemble des opérateurs activés simultanément, calculé comme précédemment.

La première méthode, celle que nous avons utilisée, présente un inconvénient : il se peut qu'un même groupe d'opérateurs simultanément activables soit rencontré plusieurs fois ; il y a alors une perte de temps à étudier plusieurs fois s'ils vérifient la relation ρ . Mais cette méthode présente l'avantage, par rapport à la seconde, de ne pas avoir à rechercher tous les éléments de l'ensemble ρ , d'où un gain de temps sur ce point, et celui de ne pas avoir à mémoriser cette liste, d'où un gain de place mémoire occupée. Ce dernier avantage n'est pas négligeable en APL, où les zones de travail ne sont pas très étendues, et il semble, finalement, que la méthode que nous avons choisie soit préférable.

REMARQUE

Nous devons noter toutefois un point important. Lors de la construction du graphe des marquages conséquents, une transition est tirée dès qu'elle est sensibilisée, car il n'est pas tenu compte, à ce niveau, de la condition booléenne associée à la transition. D'autre part, en fonction de la durée des actions, certains marquages qui apparaissent dans le graphe peuvent ne pas exister en réalité. En conséquence, dans chacun des deux cas envisagés précédemment, il est possible d'obtenir des ensembles d'opérateurs contenant plus d'éléments que ceux que l'on recherchait.

Une solution de ce problème apparaît immédiatement : elle consiste, pour la recherche des opérateurs activables simultanément, et lorsque les opérateurs sont associés aux transitions, à considérer les transitions qui sont non seulement sensibilisables en parallèle, mais aussi validées simultanément. Nous allons montrer que cette solution, qui peut paraître bonne a priori, doit être rejetée car elle conduit à "oublier" de possibles simultanéités d'opérations ; ceci est dû au fait que dans la réalité, les opérations ne sont pas instantanées.

Reprenons le réseau de PETRI de la figure II.12., mais cette fois en mettant une étiquette sur chaque transition : nous obtenons la figure II.13. Précisons que $Q_4 = Q_3$, et supposons qu'au départ, Q_2 et Q_3 aient la valeur "vrai". Les opérations op2 et op3 vont donc se dérouler simultanément. Mais nous pouvons aussi supposer que op2 va se terminer la première, et que cette opération a modifié la valeur de Q_4 : la transition t_4 est maintenant validée, et également sensibilisée. L'opération op4 va donc débuter, alors que op3 n'est pas terminée. Il faut donc compter les opérateurs exécutant op3 et op4 parmi les couples d'opérateurs activables simultanément, alors que l'exclusivité des fonctions logiques Q_3 et Q_4 aurait pu permettre d'exclure cette possibilité.

C'est donc bien les transitions sensibilisables en parallèle comme nous les avons définies en II.4.2.a., qu'il faudra considérer.

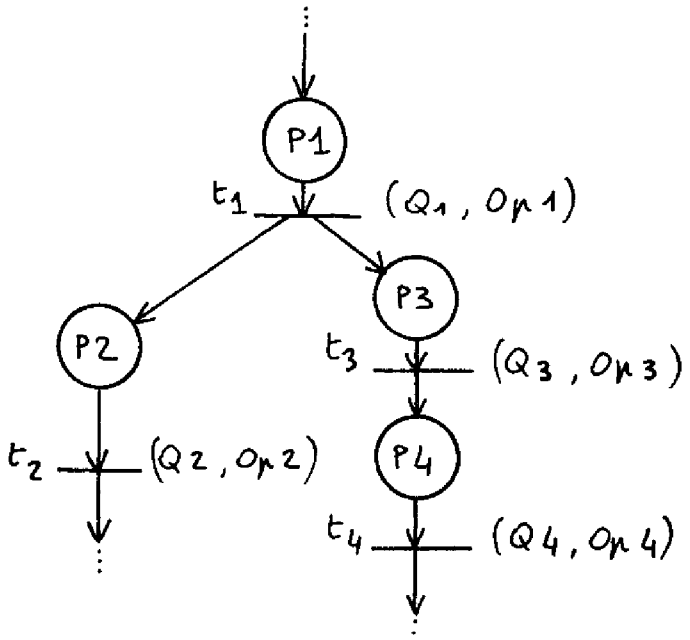


FIGURE II.13.

En conclusion, si l'analyse par cet algorithme donne pour résultat un schéma non déterminé, il sera préférable de vérifier la provenance de ce résultat, c'est-à-dire de rechercher précisément quel est le couple d'opérateurs activés simultanément qui appartient à l'ensemble $\tilde{\rho}$; pour confirmer ce résultat, une étude plus approfondie du système en ce point particulier du fonctionnement, faisant intervenir les durées des opérations, pourra être nécessaire.

II.5. CONCLUSION

Nous avons étudié dans ce chapitre l'analyse des schémas à réseaux de PETRI. Pour la partie concernant le graphe de commande, nous pouvons nous résumer en trois étapes :

1. On recherche la classe des marquages conséquents, à partir du marquage initial. Ceci permet de savoir si le réseau de PETRI est sauf pour ce marquage.

2. S'il est sauf, on peut utiliser un premier théorème pour montrer qu'il est propre.

3. Enfin, s'il est propre, on peut utiliser un deuxième théorème pour montrer qu'il est vivant.

Ensuite, le graphe de données et la recherche des opérateurs activés simultanément nous permet d'étudier l'aspect déterminé du schéma complet.

Mais si le réseau est complexe, c'est-à-dire s'il est de grande taille et très parallèle, cette analyse devient lourde à effectuer, car le graphe des marquages conséquents peut être très étendu. Heureusement, il a été montré [VAL.1] que dans certains cas, il était possible de remplacer une transition du réseau par un bloc (réseau de PETRI comportant une transition initiale et une transition finale), ce remplacement conservant les trois propriétés : sauf, vivant et propre.

Dans le cas de systèmes complexes, il est alors préférable de procéder par affinements successifs, en utilisant une méthode de conception descendante. Le système est d'abord décrit d'une manière grossière par un schéma à réseau de PETRI. Ce schéma est analysé et vérifié afin de détecter toute erreur de conception introduite à ce niveau. Puis chaque opération est décrite en détail à l'aide de schémas à réseaux de PETRI, et ces schémas sont à leur tour analysés et validés. Si le réseau de départ était sauf, propre et vivant pour son marquage initial, et si chaque opération est représentée par un réseau également sauf, propre et vivant, alors le réseau total obtenu en substituant chaque transition, étiquetée par une opération, par le réseau correspondant, sera également sauf, propre et vivant pour son marquage initial.

En utilisant cette méthodologie, il est ainsi possible de traiter de gros systèmes à l'aide du logiciel que nous avons écrit.

CHAPITRE III

PROGRAMMES DE DESCRIPTION,

D'ANALYSE ET DE SIMULATION

DES SCHÉMAS À RÉSEAUX DE PETRI

-

III.1. INTRODUCTION

Après avoir défini les Schémas à Réseaux de PETRI et leurs diverses propriétés, nous allons maintenant présenter un ensemble de fonctions destinées à les étudier.

Nous avons déjà exposé les raisons pour lesquelles nous avons choisi le langage APL : c'est un langage conversationnel, qui permet une mise au point rapide et aisée des programmes. Ses instructions présentent une grande concision. Aussi, pour conserver une bonne lisibilité, les programmes ont été structurés en petits blocs, ce qui les rend facilement vérifiables. Ces blocs exécutent chacun une fonction bien précise. Dans le déroulement normal de l'étude d'un schéma, ils s'enchaînent successivement, mais grâce à une grande souplesse d'utilisation, il est possible de faire appel à certains séparément pour étudier un point particulier. Enfin, de brefs commentaires, indiquant à l'utilisateur le mode d'emploi de chaque fonction, rendent ces programmes mieux transmissibles.

Nous considèrerons deux niveaux de fonctions : les fonctions permanentes, toujours présentes, concernant les algorithmes de description, d'analyse et de simulation, et les fonctions spécifiques, associées à chaque système particulier étudié.

Le premier programme que nous utiliserons est celui qui permet de décrire le graphe de commande. Pour faire l'analyse, une fois que cette description est correcte, nous appellerons ensuite le programme de déclaration du graphe de données. Nous pourrons alors effectuer l'analyse du Schéma ou réaliser directement une simulation.

EXEMPLE : Afin que le lecteur puisse suivre facilement la présentation de ces programmes, nous allons donner un exemple simple de schéma à réseau de PETRI. Ce schéma est essentiellement didactique et ne décrit pas un système réel ; nous l'utiliserons seulement, tout au long de ce chapitre, pour illustrer les diverses étapes de l'étude d'un schéma. D'autres exemples, qui cette fois représenteront des systèmes réels, seront étudiés au dernier chapitre.

Les figures III.1.a. et III.1.b. donnent respectivement le graphe de commande et le graphe de données de ce schéma ; l'interprétation est donnée dans le tableau III.1.c. Le nom du schéma est "EXEMP".

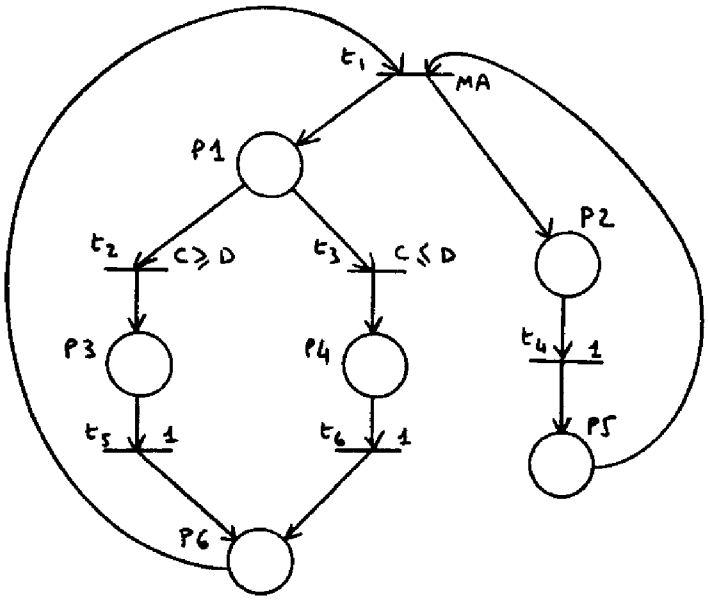


FIGURE III.1.a.

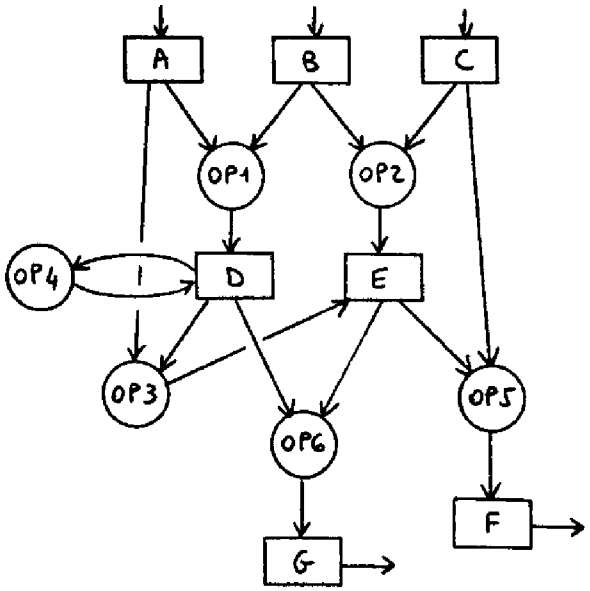


FIGURE III.1.b.

Interprétation du schéma

OP1 : D ← A + B

OP2 : E ← B + C

OP3 : E ← D - A

OP4 : D ← D + 2

OP5 : F ← E + C

OP6 : G ← 2 x D + E

"Entrées" du système :

A , B , C

"Sorties" du système :

F , G

TABLEAU III.1.c.

III.2. LE PROGRAMME DE DESCRIPTION

Avant de décrire le schéma écrit par un utilisateur à partir du cahier des charges d'un système, nous devons nous assurer que le graphe de commande ne présente a priori aucune incompatibilité avec nos fonctions d'étude.

III.2.1. Précautions préliminaires

Il suffit, pour qu'il soit accepté par le programme, que le réseau à décrire ne comporte ni boucle élémentaire, ni place source ou puits. Nous allons voir que ces deux points ne constituent en aucun cas des restrictions.

Nous avons déjà dit, en effet, au paragraphe I.4.1., que la présence d'une boucle élémentaire dans un réseau de PETRI empêchait la construction correcte de sa matrice d'incidence. Il est heureusement facile de corriger ce défaut, en remplaçant la transition composant la boucle élémentaire par un bloc formé d'une place comprise entre deux transitions. C'est ce qui est réalisé dans la figure III.2. Il s'agit d'utiliser l'équivalence que nous avions montrée à la figure I.29.

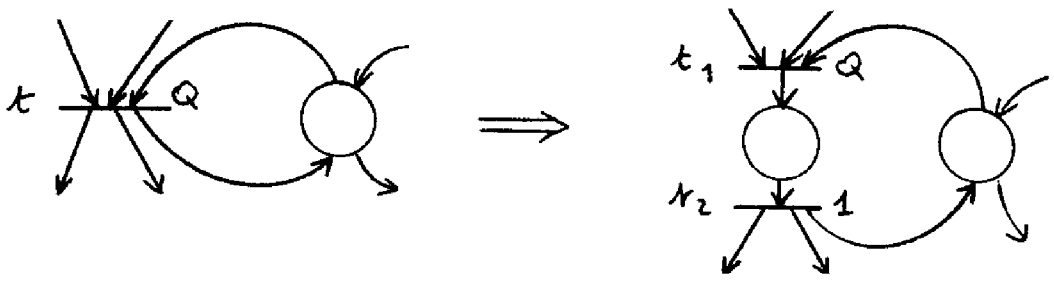


FIGURE III.2. Suppression d'une boucle élémentaire.

La première transition du bloc, t_1, reçoit les places antécédentes de l'ancienne transition t, ainsi que la même fonction logique réalisant la validation : Q ; la deuxième, t_2, reçoit les places suivantes de t, et l'expression logique associée aura toujours la valeur 1. Ainsi, le fonctionnement du réseau n'est en rien modifié. De plus, il a été montré que cette modification du réseau de PETRI ne changeait pas les propriétés : sauf, vivant et propre.

En outre, lorsque le graphe de commande possède une place initiale source et une place finale puits, il est facile de reboucler le réseau en rajoutant une transition entre ces deux places extrêmes. Comme nous l'avons dit au paragraphe I.3.9., cette transition supplémentaire permettra au système de fonctionner de manière répétitive. Elle pourra être validée par un signal de réinitialisation.

Mais si le réseau comporte plusieurs places sources et places puits, il est nécessaire de savoir ce qu'a voulu exprimer le concepteur avant de faire d'éventuels rebouclages.

III.2.2. Principe de la description

La méthode utilisée pour décrire le graphe de commande consiste à donner successivement l'origine et l'extrémité de chaque arc.

Pour limiter les risques d'erreur, chaque arc est décrit de deux façons différentes : pour chaque place, les transitions d'entrée et de sortie sont indiquées ; puis, pour chaque transition, sont données les places d'entrée et celles de sortie. Cette redondance dans la description permet d'en vérifier la cohérence.

L'utilisateur donne tout d'abord les trois premiers paramètres du réseau : le nom du système étudié, le nombre de places et le nombre des transitions.

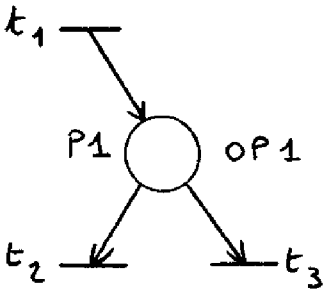
Puis la description s'effectue de manière interactive : au cours d'un dialogue avec l'ordinateur, l'utilisateur doit compléter une phrasetype pour chaque place ou pour chaque transition.

III.2.3. Description par rapport aux places

Dans la première partie, une phrase type correspond à chaque place : l'utilisateur donne le nom de cette place, ses transitions précédentes, ses transitions suivantes, et éventuellement l'action associée à cette place (dans le cas où la convention choisie est d'affecter les opérations aux places).

Les transitions sont indiquées par un numéro, précédé ou non d'une lettre. Ceci suppose que toutes les transitions du réseau de PETRI ont été numérotées ; de même, toutes les places ont reçu un nom. Le langage APL permet de traiter aussi facilement des nombres ou des caractères. Si une place a plusieurs transitions d'entrée ou de sortie, les numéros de celles-ci seront indiqués séparés par une virgule ou un séparateur quelconque. Généralement, l'action associée est donnée sous la forme du nom de l'opérateur correspondant, mais il est possible de donner directement une quelconque instruction en langage APL.

La figure III.3.a. montre la description d'une place ; les mots encadrés sont les réponses de l'utilisateur.




```
SI                    : | T1
NOM DE LA PLACE       : | P1
FAIRE                 : | OP1
ALLER A               : | T2,T3
```

FIGURE III.3.a.

Avant d'effectuer la mise en mémoire des données de chaque phrase, le programme effectue certaines vérifications. La comparaison des numéros des transitions d'entrée et de sortie d'une même place permet de détecter une boucle élémentaire. L'absence de transition d'entrée (ou de sortie) indique une place source (ou puits). Dans chaque cas, l'utilisateur en sera averti par un message.

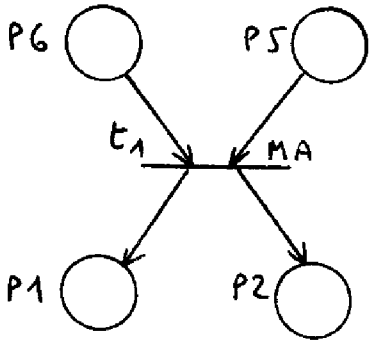
La phrase type est répétée automatiquement un nombre de fois égal au nombre de places déclaré initialement. Mais, une erreur de frappe étant toujours possible, l'utilisateur peut demander la correction d'une phrase autant de fois qu'il le désire : la phrase fausse est frappée à nouveau, et le programme efface automatiquement l'ancienne phrase déclarée sous le même nom de place, puis lui substitue celle qui vient d'être introduite. Le tableau III.3.b. montre la procédure de correction.

III.2.4. Description par rapport aux transitions

Dans la deuxième partie, la phrase type décrit une transition : l'utilisateur donne son numéro, les noms de ses places d'entrée, les noms de ses places de sortie, la condition associée, et éventuellement l'action associée à cette transition.

Les noms des places sont obligatoirement séparés par des virgules. La condition validant la transition doit être exprimée sous forme d'une expression en langage APL prenant la valeur logique "vrai" ou "faux".

La figure III.3.c. montre la description d'une transition (dans notreexemple, il n'y a pas d'action sur la transition).

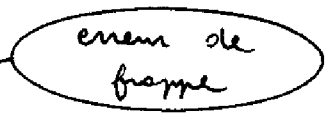


DEPUIS	:	*P6,P5*
SI	:	*MA*
TIRER	:	*T1*
ET ALLER A	:	*P1,P2*

FIGURE III.3.c.

Le programme n'accepte la phrase que si tous les noms de places ont déjà été donnés dans la première partie de la description.

```
DESCRIPTION_PAR_RAPPORT_AUX_PLACES_:
      .
      .
      .
      .
      .
      .
      .
SI    :                    T3
NOM DE LA PLACE   :        P4
FAIRE     :                OP4
ALLER A     :              T5
      .
      .
      .
      .
      .
SI    :                    P5,T6
NOM DE LA PLACE   :        P6
FAIRE     :                OP6
ALLER A     :              T1
```




```
VOULEZ-VOUS CORRIGER LA DESCRIPTION D'UNE PLACE ?
OUI

SI    :                    T3
NOM DE LA PLACE   :        P4
FAIRE     :                OP4
ALLER A     :              T6
```


```
VOULEZ-VOUS CORRIGER LA DESCRIPTION D'UNE PLACE ?
NON

DESCRIPTION_PAR_RAPPORT_AUX_TRANSITIONS_:

DEPUIS     :           P6,P5
SI    :                MA
TIRER     :              T1
ET ALLER A     :         P1,P2
      .
      .
      .
      .
      .
      .
      .
      .
      .
```

(Les parties encadrées sont celles écrites par l'utilisateur).

TABLEAU III.3.b.

Comme précédemment, la phrase type est répétée automatiquement un nombre de fois égal au nombre de transitions, puis l'utilisateur peut corriger s'il le désire, comme pour la description par rapport aux places.

III.2.5. Contrôle de la description

Ces deux descriptions ont permis de construire, séparément, deux matrices d'incidence du même réseau de PETRI, qui, bien entendu, doivent être identiques. Le programme calcule leur différence. Si la matrice obtenue n'est pas nulle, ceci indique une erreur de description : celle-ci est facilement localisée, les indices de ligne et de colonne de l'élément non nul correspondant à la place et à la transition en cause.

En cas de plusieurs erreurs, seule la première est indiquée, mais il est facile de détecter les autres en faisant écrire la matrice différence.

L'utilisateur doit corriger la place ou la transition mal décrite en appelant une fonction particulière ; après cela, le réseau est à nouveau contrôlé.

Lorsque cette description a été acceptée, il reste à indiquer le marquage initial : les noms des places marquées sont énumérés. L'utilisateur doit enfin donner les numéros des "transitions-utilisateurs" ; ces transitions permettront au monde extérieur d'intervenir au cours de la simulation;nous examinerons plus tard leur rôle particulier.

III.2.6. Eléments caractéristiques du réseau

A ce niveau de la description, l'utilisateur dispose de tous les éléments caractéristiques du réseau de PETRI constituant le graphe de commande du schéma. Ces éléments sont gardés en mémoire sous un nom formé par la concaténation d'un préfixe et du nom du réseau.

Il s'agit des objets suivants :

INC nom : matrice d'incidence, ayant autant de lignes qu'il y a de places (NP) et autant de colonnes qu'il y a de transitions dans le réseau (NT).

PE nom et PS nom sont deux matrices de caractères de NT lignes, chacune contenant les noms des places précédentes et suivantes, (d'entrée et de sortie), d'une transition.

N nom est la matrice des noms des places (NP lignes).

C nom (NT lignes) contient les conditions booléennes associées aux transitions.

A nom contient les actions associées aux places ou aux transitions (NP ou NT lignes) selon la convention choisie.

M nom est un vecteur binaire de dimension NP, les bits 1 correspondant aux places marquées pour le marquage initial.

TU nom est un vecteur binaire de dimension NT, les bits 1 correspondant aux transitions-utilisateurs.

Q nom est un indicateur, égal à 1 ou 0, selon que les actions ont été associées aux places ou aux transitions ; il sera utilisé lors de l'analyse pour le choix de certains algorithmes.

INCEXEMP

1	-1	-1	0	0	0
1	0	0	-1	0	0
0	1	0	0	-1	0
0	0	1	0	0	-1
-1	0	0	1	0	0
-1	0	0	0	1	1

PREXEMP

5	6
1	0
1	0
2	0
3	0
4	0

PSEXEMP

1	2
3	0
4	0
5	0
6	0
6	0

NEXEMP

*P*1
*P*2
*P*3
*P*4
*P*5
*P*6

CEXEMP
MA
C≥*D*
C≤*D*
1
1
1

AEXEMP

*OP*1
*OP*2
*OP*3
*OP*4
*OP*5
*OP*6

MEXEMP
0 0 0 0 1 1

TUEXEMP
1 0 0 0 0 0

QEXEMP
1

Cette façon de garder en mémoire les éléments caractéristiques d'un réseau rend très rapide et très facile le chargement de ces données dans la zone de travail, lorsque l'utilisateur appelle un des programmes d'étude. En effet, le langage APL possède une instruction très intéressante, l'ordre ⍎ , ("exécute") ; l'une des nombreuses possibilités de cette instruction est de réaliser automatiquement des concaténations de mots.

EXEMPLE : Prenons un exemple précis : pour la simulation du système appelé "EXEMP", décrit par un schéma à réseau de PETRI, la fonction doit rechercher en mémoire la matrice des noms des places, appelée "NEXEMP" lors de la description du schéma.

Au début de la simulation, il suffit que l'utilisateur donne le nom du système qu'il veut étudier : ici "EXEMP", ce qui a pour effet de donner à une variable, appelée SYS, la valeur alphanumérique "EXEMP". L'instruction N ⟵ ⍎ 'N', SYS chargera alors le contenu de NEXEMP dans la variable locale N. Et il en est de même pour les autres éléments.

REMARQUES

1. En langage APL, l'utilisateur est déchargé de toute déclaration. Cependant, la construction progressive des diverses matrices, au cours de la description, se fait par chargement de lignes successives, qui doivent avoir la même longueur, c'est-à-dire la longueur de la plus grande. Il est donc nécessaire de réserver une place volontairement trop grande. Pour limiter la place mémoire occupée au maximum nécessaire, il faut donc soumettre ces matrices à un sous-programme de réduction. Ainsi, les colonnes ne comportant que des zéros, dans le cas des matrices numériques, ou uniquement des blancs, pour les matrices de caractères, sont supprimées.

2. En dehors du programme de description, l'utilisateur a la possibilité d'effectuer certaines modifications sur les éléments caractéristiques du réseau : changement d'une action, d'une condition logique,

d'un nom de place, et même rajout ou suppression d'un arc. Ceci permet de faciliter la mise au point. Mais dans le cas où une correction entraîne une modification du nombre des places et des transitions, il est préférable, pour l'utilisateur, de procéder à une nouvelle description complète du graphe de commande, ceci pour deux raisons : d'une part, il est long et peu facile de changer les dimensions de tous les éléments caractéristiques, d'autre part, il n'y aurait pas de vérification après toutes ces modifications.

III.3. DECLARATION DU GRAPHE DE DONNEES

III.3.1. Introduction

Nous avons vu au premier chapitre que l'on associait un graphe de données à un réseau de PETRI, et, dans le deuxième, qu'il était nécessaire de connaître ce graphe pour effectuer l'analyse du schéma à réseau de PETRI ainsi formé. Nous avons donc écrit un programme permettant de déclarer tous les opérateurs et toutes les mémoires qui interviennent dans le fonctionnement du système.

Le graphe de données, tel qu'il a été défini en I.7.2. d'après [VAL.1] , contient les prédicats. Mais nous ne déclarerons pas ces prédicats ici. En effet, leur expression développée a déjà été donnée lors de la description du graphe de commande (paragraphe III.2.4.) et le langage APL permet, à nouveau grâce à l'instruction "exécute" , de calculer leurs valeurs sans avoir recours à un opérateur.

La description du graphe de données se fait obligatoirement après celle du graphe de commande, certains éléments mis en mémoire par le premier programme intervenant dans le second.

Comme précédemment, nous allons utiliser une certaine redondance : avant de décrire les arcs du graphe de données, nous énumèrerons ses sommets, c'est-à-dire les mémoires et les opérateurs.

Le tableau III.4. montre la déclaration du graphe de données de l'exemple présenté en III.1. C'est ce résultat que nous allons commenter.

```
DECLARE

NOM DU RESEAU  : EXEMP

DONNER LA LISTE DES MEMOIRES
(NOMS SEPARES PAR DES VIRGULES,
 6 CARACTERES MAXIMUM )        .
ENSUITE TAPER : //
A,B,C,D,E,F,G,MA
//

DONNER L'OPERATEUR ASSOCIE A CHAQUE
PLACE :
P1     :OP1
P2     :OP2
P3     :OP3
P4     :OP4
P5     :OP5
P6     :OP6

DEFINIR CES OPERATEURS ,
PUIS TAPER : FINI .

     VOP1
[1]  D←A+B  ∇
     VOP2
[1]  E←B+C  ∇
     VOP3
[1]  E←D-A  ∇
     VOP5
[1]  F←E+C  ∇

FINI
DEFINIR LES 2 FONCTIONS APL :
OP4        OP6
PUIS TAPER : FINI .

     VOP4
[1]  D←D+2  ∇
     VOP6
[1]  G←2×D+2  ∇
FINI
VOULEZ-VOUS LISTER CES 6 OPERATEURS ?
NON

INDIQUER LES MEMOIRES D'ENTREE ET SORTIE
POUR CHAQUE OPERATEUR :
EXEMPLE : "OPE:A1,A2;A3"
PUIS TAPER : //

OP1:A,B;D
OP2:B,C;E
OP3:A,D;E
OP4:D;D
OP5:C,E;F
OP6:D,E;G
//
```

FONCTION	MEMOIRES D'ENTREE		MEMOIRES DE SORTIE
OP1	**A	B	**D
OP2	**B	C	**E
OP3	**A	D	**E
OP4	**D		**D
OP5	**C	E	**F
OP6	**D	E	**G

TABLEAU III.4.

III.3.2. Listes des identificateurs

Après avoir indiqué le nom du système, l'utilisateur doit donner la liste des noms des cellules mémoires : ces noms sont écrits dans un ordre quelconque, simplement séparés par des virgules. Le programme déchiffre la liste, détecte les sépérateurs, et range les noms en colonne dans une matrice de caractères.

Le programme demande ensuite la liste des noms des opérateurs. Pour chaque transition, ou pour chaque place indiquée, selon la convention choisie pour la représentation des actions, l'utilisateur doit répondre le nom de l'opérateur associé.

Nous devons noter que cette réponse n'est pas toujours identique à l'action associée indiquée lors de la description du réseau de commande ; en effet, l'action complète pouvait comporter, outre le nom de l'opérateur, les noms de deux ou trois arguments (comme le permet la règle de définition d'une fonction en langage APL) ; au contraire, maintenant, c'est l'opérateur seul qui nous intéresse pour étudier l'aspect déterminé du schéma.

Il faut ensuite définir ces opérateurs. Quand l'utilisateur tape "FINI", le programme vérifie que tous ont été définis sous forme de fonctions APL. Dans le cas contraire, il donne la liste de ceux qui manquent en bibliothèque : le clavier est alors placé en position d'attente pour laisser la main à l'utilisateur.

Enfin, quand toutes ces fonctions sont définies, le programme passe à la description du graphe de données proprement dit. Auparavant, il est possible de faire lister les opérateurs, pour les contrôler.

III.3.3. Mémoires d'entrée et de sortie d'un opérateur

Pour chaque opérateur, l'utilisateur doit écrire une phrase contenant, avec des séparateurs bien précis, les noms de ses mémoires d'entrée et ceux de ses mémoires de sortie. Cette phrase est construite de la façon suivante :

```
OPERATEUR : ME1, ME2,..., MEN ; MS1,..., MSK
```

Aussitôt frappée, la phrase est examinée : le programme contrôle la présence de chaque opérateur, et vérifie que tous les noms de mémoires donnés font partie de la liste établie précédemment. Un message est imprimé en cas d'erreur : "mauvaise forme de l'expression", ou "opérateur inconnu", ou "variable inconnue".

A la fin, les données enregistrées apparaissent sous forme de tableaux.

III.3.4. Eléments en mémoire

Les variables d'entrée et de sortie de chaque opérateur sont gardées en mémoire sous la forme de deux matrices alphanumériques : VAE nom et VAS nom (tableau III.5.). Chaque ligne de ces matrices correspond à un opérateur. Pour plus de commodité lors de l'analyse du schéma, le programme construit aussi, parallèlement, deux matrices numériques contenant les numéros des variables d'entrée et de sortie, ces numéros étant leurs indices dans la liste établie en III.3.2. : NVE nom et NVS nom (tableau III.5.).

Pour le schéma à réseau de PETRI "EXEMP" :

	VAE EXEMP	NVE EXEMP	VAS EXEMP	NVS EXEMP
OP1	A B	1 2	D	4
OP2	B C	2 3	E	5
OP3	A D	1 4	E	5
OP4	D	4 0	D	4
OP5	C E	3 5	F	6
OP6	D E	4 5	G	7

TABLEAU III.5.

En résumé, pour l'étude ultérieure, nous disposerons des éléments suivants, gardés en mémoire de la même façon que ceux du graphe de commande :

- OP nom : matrice des noms des opérateurs
- VAE nom et VAS nom : mémoires d'entrée et de sortie
- NVE nom et NVS nom : numéros correspondant aux mémoires des deux matrices précédentes.

Ces cinq matrices ont un nombre de lignes égal à celui des opérateurs, c'est-à-dire soit le nombre des places, soit celui des transitions.

Si les cellules mémoires sont nombreuses, l'utilisateur peut aussi, éventuellement, mettre en mémoire sous le nom "INIT nom" la liste de celles qui devront être mises à une valeur initiale avant de commencer une simulation.

III.4. PROGRAMME D'ANALYSE

III.4.1. Appel du programme

Ce programme ne peut être appelé qu'après la description du graphe de commande et la déclaration du graphe de données. Dès que l'utilisateur a indiqué le nom du système, la présence en mémoire de tous les éléments nécessaires à l'analyse est vérifiée, et le programme signale éventuellement leur absence. La connaissance du graphe de données est indispensable pour analyser l'aspect déterminé du schéma.

Toutefois, il est possible d'effectuer l'analyse du graphe de commande seul, lorsque le réseau de PETRI n'a pas été décrit par la procédure exposée jusqu'ici, en donnant uniquement son marquage initial et sa matrice d'incidence. Mais pour des réseaux complexes, l'expérience a montré, et il est intéressant de le signaler, que la procédure de description normale pouvait être plus rapide et surtout plus facile que la donnée d'une matrice d'incidence de grandes dimensions.

Nous l'avons vu au deuxième chapitre, la base de l'analyse est la recherche du graphe des marquages conséquents.

III.4.2. Graphe des marquages conséquents

A partir du marquage initial donné, la fonction GRAMAR calcule les marquages résultant du tir des différentes transitions possibles. Rappelons qu'ici pour qu'une transition soit tirée, il suffit qu'elle soit sensibilisée. Après avoir détecté ces transitions, la fonction procède par addition de vecteurs, comme nous l'avons vu au paragraphe I.4.3., le marquage résultant du tir d'une transition étant calculé en ajoutant au marquage antérieur la colonne de la matrice d'incidence correspondant à cette transition.

Chacun des marquages obtenus reçoit un numéro : le marquage initial est numéroté "1", ses successeurs "2","3","4",... dans l'ordre des entiers naturels. Puis la même procédure recommence à partir de chaque nouveau marquage : d'autres transitions sont sensibilisées, ce qui donne d'autres marquages successeurs. On construit ainsi un arbre, par couches successives, comme le montrent les figures III.6. et 7.

Marquages obtenus pendant la construction du graphe :

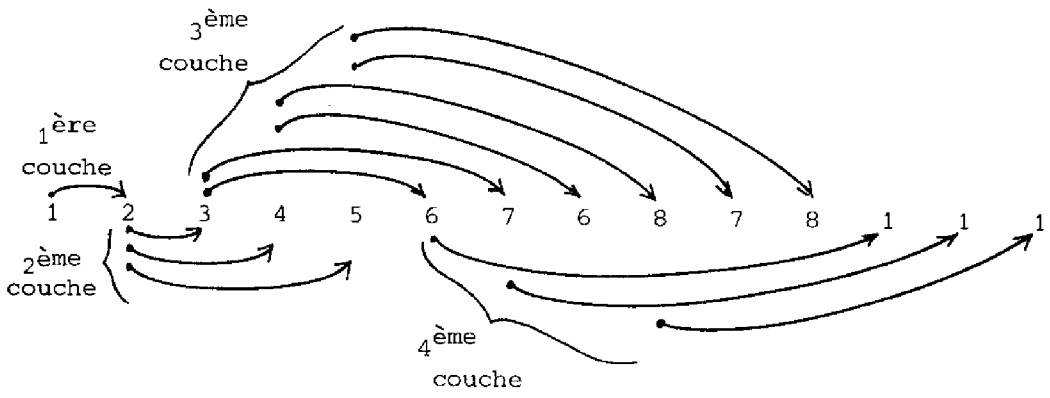


FIGURE III.6.

Le test d'arrêt de cette construction est donné par l'une des trois circonstances suivantes :

1. Le marquage créé a déjà été examiné :
   Avant de calculer les successeurs d'un marquage, la fonction compare celui-ci à tous les précédents : s'il est identique à l'un deux, elle passe directement à l'étude d'un marquage non encore examiné.

2. Le marquage conséquent conduit à deux jetons dans une même place : Dans ce cas, le réseau de PETRI n'est pas sauf, et compte tenu de ce que nous avons dit au paragraphe II.3.3., il est inutile, pour l'instant, de poursuivre plus loin l'analyse.

3. Aucune transition n'est sensibilisée, pour chacun des marquages non encore examinés : il y a alors blocage.

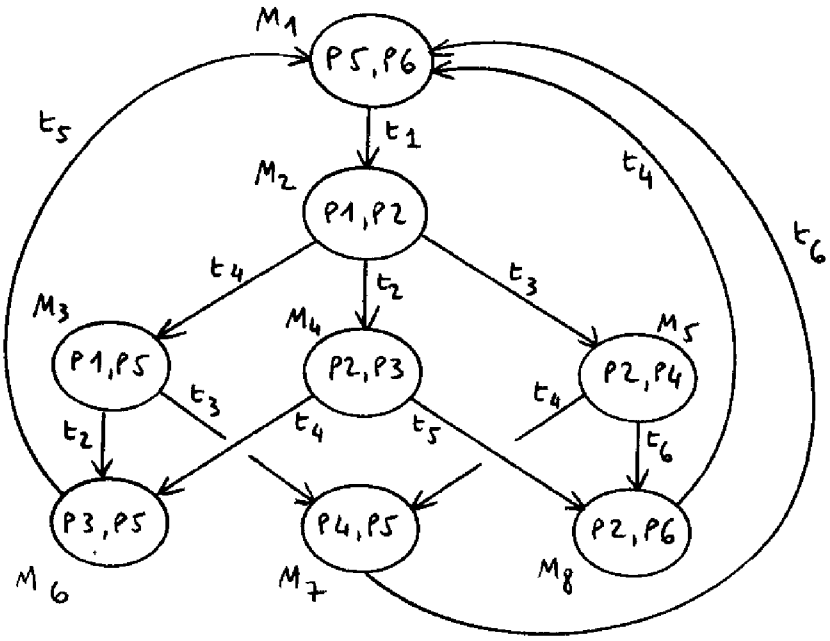


FIGURE III.7.

L'algorithme de construction du graphe des marquages conséquents est donné par la figure III.8.

A la fin de cela, nous disposons d'une suite de marquages numérotés (et aussi de leur nombre), sous forme d'un vecteur ligne ; un autre vecteur, de même dimension nous donne les marquages antérieurs ou "pères" de chacun : tableau III.9.

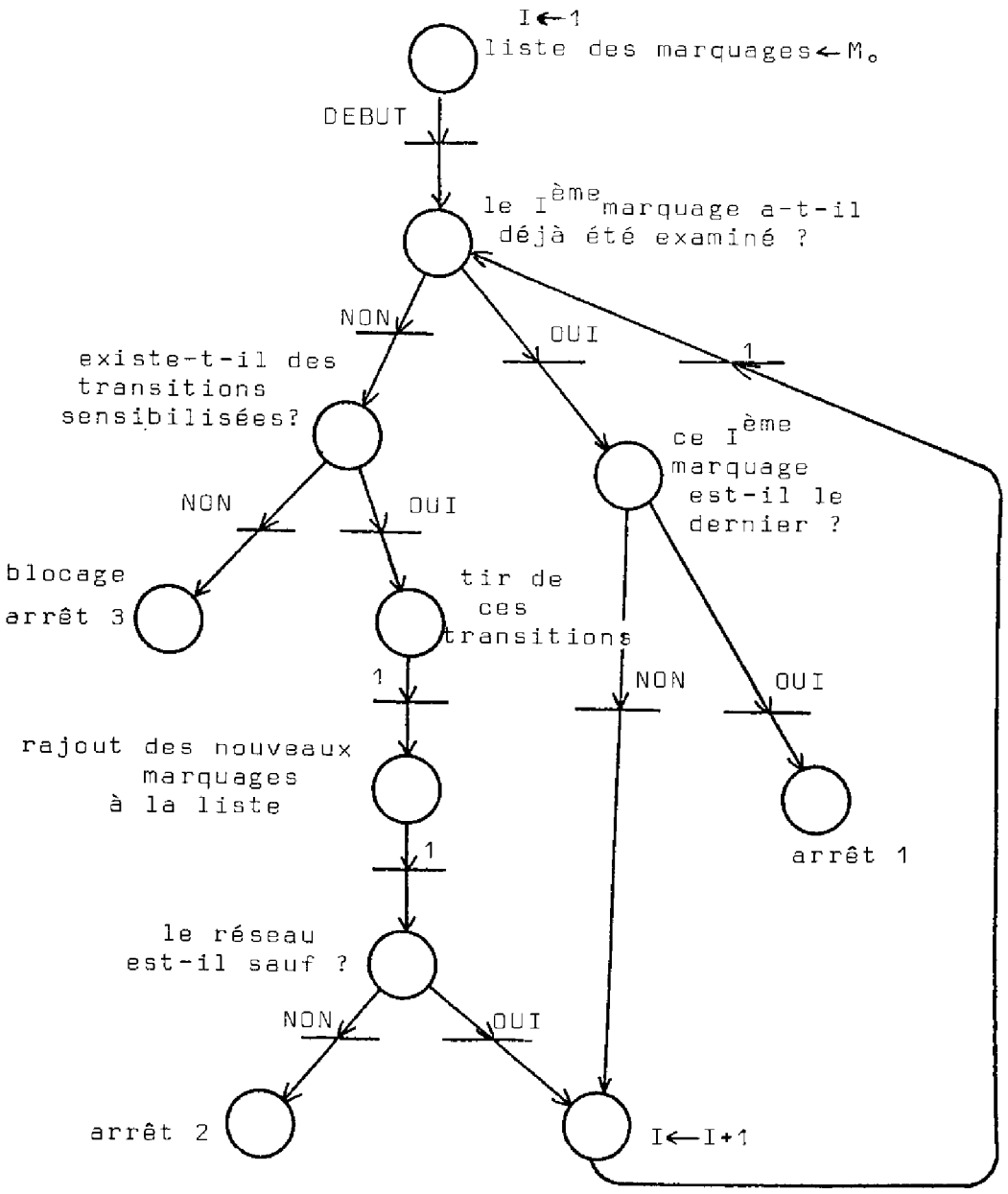


FIGURE III.8. Algorithme de construction du graphe des marquages conséquents.

Père de chaque marquage :

marquage :	1	2	3	4	5	6	7	6	8	7	8	1	1	1
père :		1	2	2	2	3	3	4	4	5	5	6	7	8

TABLEAU III.9.

Une matrice des marquages indique les places marquées : ses lignes correspondent aux places, ses colonnes aux marquages successifs (figure III.10.).

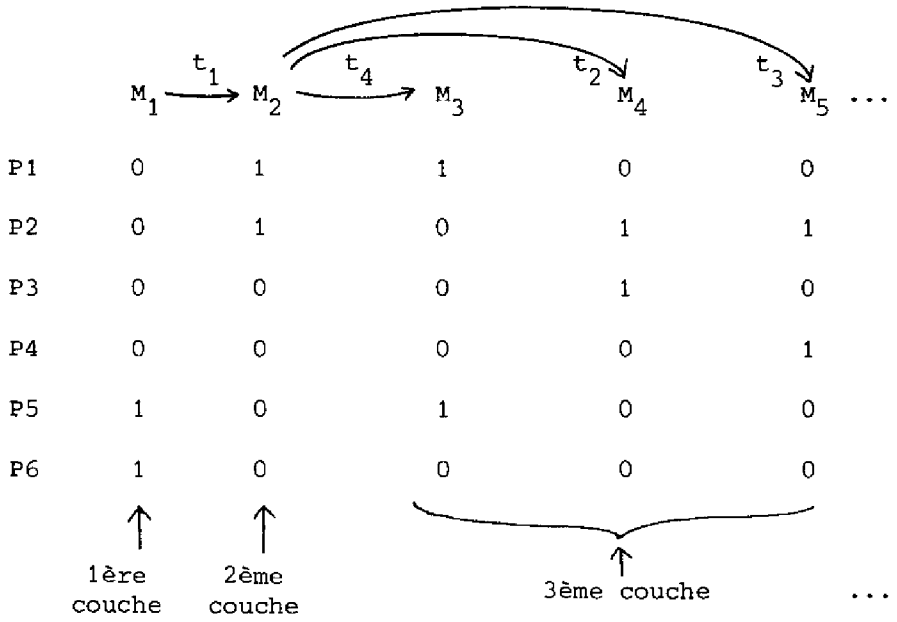
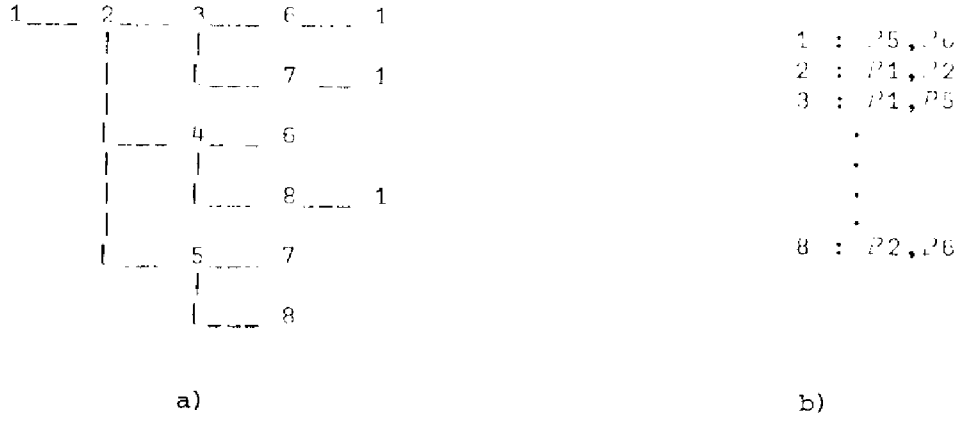


FIGURE III.10.

Le programme trace ensuite l'arbre des marquages conséquents obtenu. Chaque sommet est représenté par un numéro ; ces numéros sont liés entre-eux par des branches. La figure III.11.a. montre l'arbre des marquages pour le schéma EXEMP. Les boucles sont mises en évidence par le fait que tous les marquages identiques portent le même numéro.

Pour chaque marquage, c'est-à-dire pour chaque numéro, les noms des places marquées sont ensuite énumérés (figure III.11.b.).



a) b)

FIGURE III.11.

REMARQUE

Il suffit d'apporter une légère modification aux instructions de la fonction APL réalisant la construction du graphe, pour supprimer le deuxième test d'arrêt (réseau non sauf). Ceci rendra possible l'étude des marquages conséquents d'un réseau borné non sauf, ce qui peut avoir un intérêt dans certains cas particuliers.

A partir de ce graphe des marquages conséquents, nous allons pouvoir déduire les propriétés du réseau de PETRI du graphe de commande.

III.4.3. Résultats de l'analyse du réseau

a) Réseau sauf

La première propriété, réseau sauf, est déduite directement de la circonstance qui a arrêté la construction du graphe des marquages. Si le réseau n'est pas sauf, le nom de la place (ou des places) marquée de plusieurs jetons est indiqué .

b) Réseau vivant et propre

Ensuite, la fonction d'analyse suit la procédure que nous avons donnée au paragraphe II.3.3.

Au cours de l'élaboration du graphe des marquages, la liste des transitions qui ont été tirées a été mise en mémoire. Il est donc facile de rechercher si toutes les transitions du réseau sont inscrites au moins une fois sur cette liste.

Si ce n'est pas le cas, le programme indique immédiatement que le réseau n'est pas vivant. Il précise quelles sont les transitions qui n'ont jamais été tirées, ainsi que les places qui n'ont pas été marquées.

Mais si la réponse est positive, il nous reste à nous assurer que le graphe des marquages est fortement connexe. Pour cela, il serait possible d'appliquer une procédure classique de recherche de connexité d'un graphe. Mais avec les résultats dont nous disposons, nous pouvons utiliser une méthode plus simple et plus efficace pour le temps de calcul : il est évident, par définition, que tous les marquages de $\overrightarrow{M_o}$ sont accessibles à partir de M_o ; il suffit donc de rechercher si M_o est accessible à partir de tous les autres marquages. Nous connaissons le marquage "père" de chacun : il est possible alors de rechercher tous les "ancêtres" de M_o, (tableau III.12.).

Recherche des ancêtres du marquage initial :

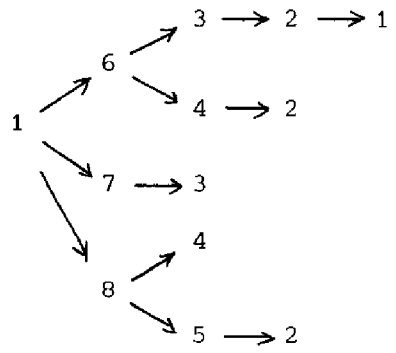


TABLEAU III.12.

Si la liste de ceux-ci contient tous les marquages de $\overrightarrow{M_o}$, le graphe des marquages est fortement connexe et le réseau est vivant et propre. Sinon, le réseau n'est pas propre.

La figure III.13. représente l'algorithme correspondant, sous forme d'un réseau de PETRI.

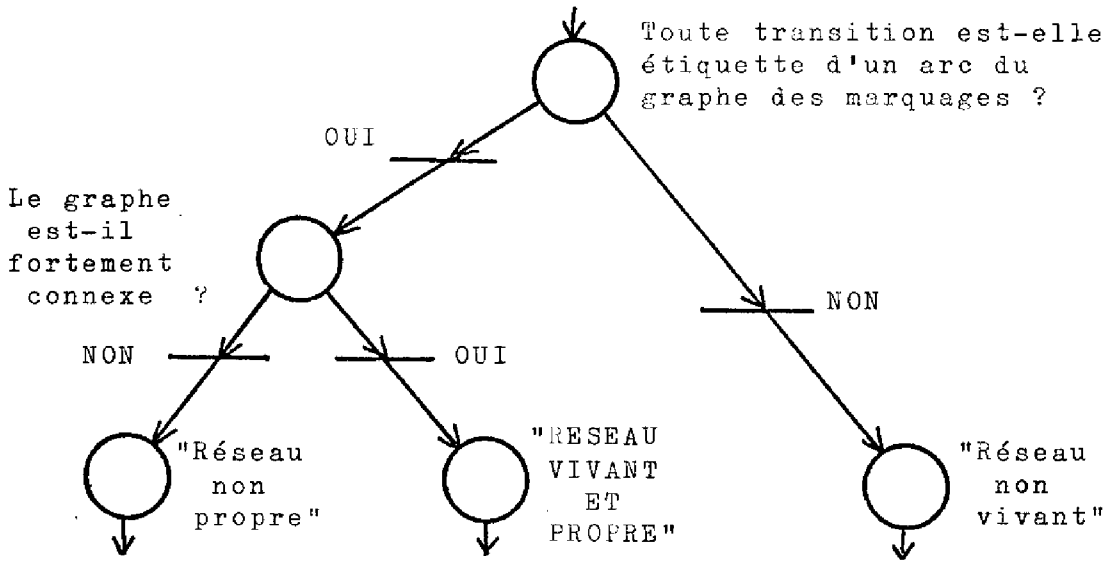


FIGURE III.13. Test de réseau vivant et propre.

III.4.4. Schéma déterminé

En dernier lieu, le programme d'analyse recherche si le schéma à réseau de PETRI est déterminé. Il est fait appel à deux sous-programmes distincts, selon que les actions ont été associées aux places ou aux transitions. La procédure a été donnée au paragraphe II.4.2.c., pour rechercher les opérateurs activés simultanément.

1. Cas où les actions sont associées aux places.

Chaque colonne de la matrice des marquages donne immédiatement les places marquées, donc les opérateurs activés simultanément pour chaque marquage.

2. Cas où les actions sont associées aux transitions.

Les listes des transitions sensibilisées par un marquage, à chaque étape de la construction du graphe des marquages, ont été gardées en mémoire.

Pour chacune de ces listes de transitions, il faut rechercher si certaines sont en conflit : ce sera le cas si des transitions de cette liste ont des places d'entrée communes. Le programme compare les lignes de la matrice des places d'entrée correspondant à ces transitions. Si deux transitions sont en conflit, il établit deux listes, de façon à séparer les transitions en conflit.

EXEMPLE : Pour le schéma "EXEMP", c'est aux places que nous avons associé les actions. Nous sommes donc dans le cas 1. Les listes d'opérateurs activés simultanément sont données par la liste des places marquées pour chaque marquage, de la figure III.11.b. :

"OP5,OP6" "OP1,OP2" "OP1,OP5" etc.

Sans se soucier du système physique qu'il pourraît représenter, on peut imaginer le même réseau de PETRI (de la figure III.1.a.) avec les opérateurs OP1,...,OP6 associés respectivement aux transitions $t_1,\ldots,t_6$. Nous serions alors dans le cas 2, et les listes d'opérateurs activés simultanément seraient obtenues de la façon suivante :

MARQUAGE	TRANSITIONS SENSIBILISEES	LISTES D'OPERATEURS
M1	t_1	OP1
M2	t_2 , t_3 (en conflit) , t_4 $\Rightarrow$ { t_2 , t_4 ; t_3 , t_4 }	OP2 , OP4 ; OP3 , OP4
M3	t_2 , t_3 $\Rightarrow$ { t_2 ; t_3 }	OP2 ; OP3
M4	t_4 , t_5	OP4 , OP5
M5	t_4 , t_6	OP4 , OP6
...		...

Dans les deux cas, en prenant deux à deux les éléments de ces listes d'opérateurs, comme indiqué au paragraphe II.4.2.c., le programme peut alors rechercher si le schéma est déterminé, en utilisant les matrices NVE nom et NVS nom.

EXEMPLE : Pour le marquage M4 = (P2,P3) du réseau EXEMP, les opérateurs activés simultanément sont OP2 et OP3. Or NVS EXEMP indique que la mémoire numérotée "5" est mémoire de sortie de ces deux opérateurs : OP2 et OP3 rangent tous deux leurs résultats dans "E" (tableau III.5.). Notre schéma n'est donc pas déterminé.

De même, si les opérateurs sont associés aux transitions : pour le marquage M2, OP3 et OP4 sont activables simultanément. Or la comparaison de la ligne 3 de NVE EXEMP avec la ligne 4 de NVS EXEMP indique que la mémoire numérotée "4" est à la fois mémoire d'entrée de OP3 et mémoire de sortie de OP4. Ce schéma est également non déterminé.

Savoir si le schéma est déterminé se ramène donc à un problème de comparaison de composantes de vecteurs, ce qui est facilement résolu grâce aux instructions du langage APL.

La figure III.14. résume, par un réseau de PETRI, l'étude de l'aspect déterminé du schéma.

Dans le cas où le schéma n'est pas déterminé, un message indique quels sont les opérateurs activés simultanément qui sollicitent une même mémoire et quelle est cette cellule mémoire.

III.5. PROGRAMME DE SIMULATION

Le programme de simulation nécessite la connaissance de tous les éléments caractéristiques du graphe de commande : matrice d'incidence, actions, conditions,... Il doit donc être précédé de la description de ce réseau de PETRI. Le graphe de données n'est pas indispensable, mais nous avons besoin des fonctions APL exécutant les opérations pour que la simulation reproduise le fonctionnement du système réel.

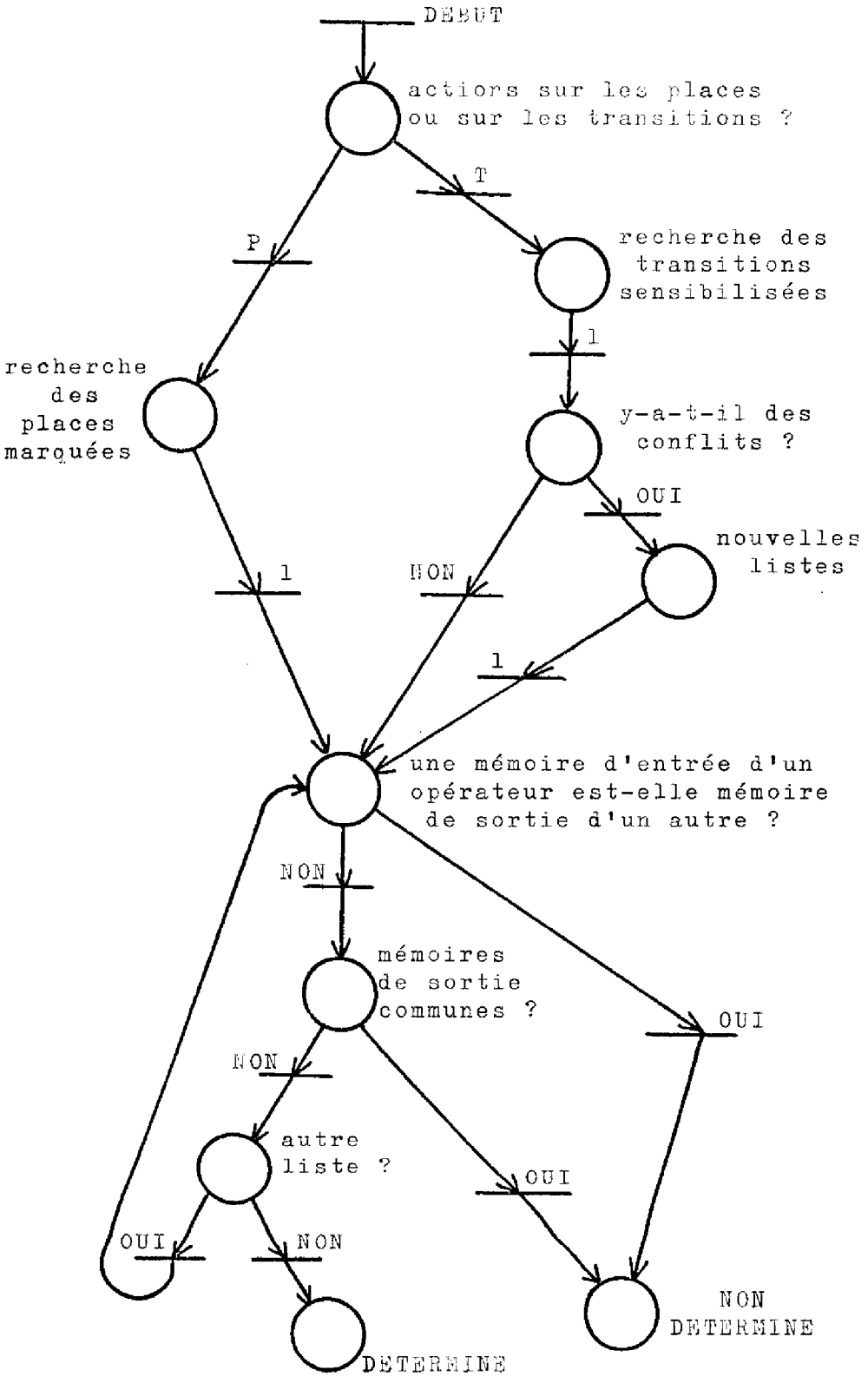


FIGURE III.14. Test de schéma déterminé .

III.5.1. Préparation de la simulation

Lorsque l'utilisateur a frappé l'instruction "SIMUL", le programme commence par demander le nom du schéma, et il vérifie que celui-ci a été décrit et que tous les éléments caractéristiques sont disponibles. Puis, l'utilisateur indique s'il désire donner une seule séquence d'entrée, ou plusieurs.

Dans le premier cas, on donne les valeurs initiales de toutes les mémoires, en particulier celles des mémoires d'entrée du système : ceci correspond à lire une seule fois les valeurs fournies par les capteurs. Puis la simulation est déclenchée en tapant l'instruction 'TIR nom'. Lorsque le réseau sera bloqué, la simulation sera terminée.

Mais dans le deuxième cas, la possibilité est laissée au système simulé de lire à plusieurs reprises les valeurs données par le monde extérieur. Ces différentes séquences d'entrée sont mises en mémoires, avant la simulation. Lorsque celle-ci sera arrêtée, une nouvelle séquence d'entrée sera lue et la simulation reprendra au point où elle s'est arrêtée.

Pour cela, l'utilisateur donne une liste de fonctions APL, dans l'ordre désiré, et définit ces fonctions qui réaliseront automatiquement le chargement des mémoires d'entrée.

Le tableau III.15. nous en montre un exemple.

Ces différentes séquences d'entrée permettent de prendre en compte l'évolution du monde extérieur. Mais certains événements extérieurs au système de commande, tels que sa mise en marche ou son arrêt, doivent intervenir d'une façon différente au cours de la simulation.

Dans ce but, nous avons défini les "transitions-utilisateurs". Ces transitions doivent être déclarées comme telles avant la simulation. L'expression booléenne qui leur est associée est fonction d'une variable atteinte par le monde extérieur. Lorsqu'une de ces transitions sera possible mais non tirable, c'est-à-dire sensibilisée sans être validée, l'utilisateur aura la possibilité, s'il le désire, de modifier la valeur de cette variable et ainsi de valider la transition. En outre, on pourra con-

```
      SIMUL
NOM DU RESEAU : EXEMP
Y-A-T-IL PLUSIEURS SEQUENCES D'ENTREE ?
OUI
DONNER LA LISTE DES FONCTIONS APL
REALISANT CES SEQUENCES D'ENTREE :(PUIS //)
ENT1,ENT2,ENT3
//
DEFINIR CES 3 FONCTIONS APL,
PUIS APPELER 'TIRS'.
      VENT1
[1]   A←B←C←3 ∇
      VENT2
[1]   A←2×B←4
[2]   C←¯4.5 ∇
      VENT3
[1]   A←-B←3×C←2 ∇

      TIRS
```

TABLEAU III.15.

sidérer que cette modification est soit une impulsion, soit un changement de valeur durable.

Prenons l'exemple du réseau de la figure III.1. Lors du marquage M1=(P5,P6), la transition t_1 est sensibilisée. Si le bouton "marche-arrêt" du système est en position "arrêt", MA est égal à 0 et la simulation est bloquée. Il est alors possible, à condition que t_1 ait été déclarée "transition-utilisateur", d'appuyer sur le bouton "marche".

Si MA reçoit une simple impulsion, le système se mettra en marche, calculera D,E,F,G à partir des données A,B,C, puis s'arrêtera en l'état correspondant au marquage (P5,P6).

Mais si MA reste au niveau 1, le système évoluera indéfiniment.

Examinons maintenant comment s'effectue la simulation.

III.5.2. Recherche des transitions tirables

La première étape consiste à rechercher la liste des transitions sensibilisées par le marquage. Les numéros de ces transitions forment le vecteur T.P. (transitions possibles). Les valeurs des expressions logiques associées à ces transitions sont calculées, et seules sont retenues les transitions validées ; les numéros de ces transitions tirables forment le vecteur T.T.

Ici interviennent les "transitions-utilisateurs".

Lorsqu'une transition (ou plusieurs) est possible, non tirable, et a été déclarée "transition-utilisateur", le programme écrit la liste des transitions possibles, celle des transitions tirables, et celle des conditions logiques non satisfaites. L'utilisateur peut alors effectuer facilement les modifications éventuelles et un nouveau vecteur T.T. est établi, tenant compte des nouvelles conditions.

Avant que ces transitions soient tirées, le programme étudie la présence de conflit, par comparaison des places précédentes de ces transitions. Un conflit est simplement signalé ; la place et les transitions concernées sont précisées ; mais la simulation ne s'arrête pas : une des transitions est choisie arbitrairement ; nous utilisons ici l'instruction APL "?n" qui tire aléatoirement une valeur comprise entre 1 et n, (ici n étant le nombre de transitions en conflit). Si cette procédure de recherche conduit à un vecteur T.T. vide, la simulation est arrêtée.

III.5.3. Tir des transitions

Le programme effectue ensuite le tir, et calcule le nouveau marquage par addition de vecteurs. Les actions associées, soit à la transition tirée, soit aux places nouvellement marquées, sont exécutées.

Dans le cas où une place reçoit un jeton alors qu'elle en contenait déjà un, le programme annonce par un message que le réseau n'est pas sauf et indique la place marquée deux fois. Ce détail est surtout intéressant lorsque l'analyse du réseau n'a pas été faite auparavant. Mais

contrairement à la construction du graphe des marquages conséquents, qui s'arrête quand le réseau n'est plus sauf, la simulation se poursuit, avec la convention énoncée en I.2.5.c. et concernant l'évolution des jetons.

Après chaque tir, pour permettre à l'utilisateur de suivre facilement le déroulement de la simulation, le programme indique quelles transitions ont été tirées et quelles sont les places marquées. Ensuite, il recherche la nouvelle liste des transitions possibles et le cycle recommence.

La simulation est arrêtée soit sur intervention de l'utilisateur, lorsqu'il a le contrôle grâce aux transitions-utilisateurs, soit par blocage, lorsqu'aucune transition n'est tirable.

Elle peut être relancée avec la même initialisation, s'il s'agit de faire une mise au point, ou avec de nouvelles valeurs initiales.

Le schéma de la figure III.16. montre le système de la simulation ; le réseau de PETRI de la figure III.17. en résume les différentes étapes.

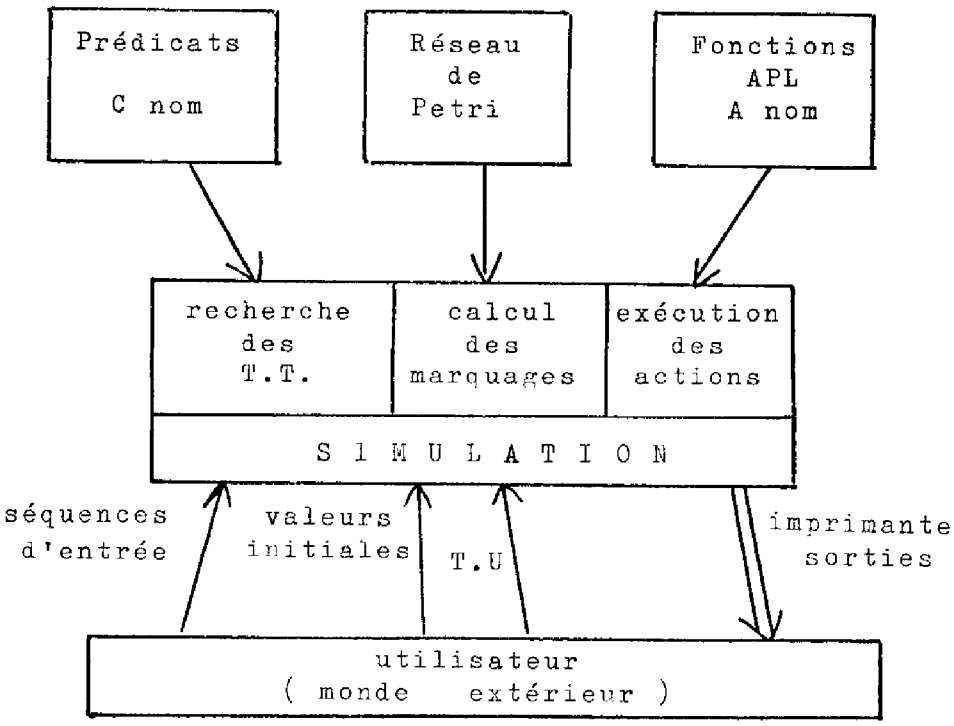


FIGURE III.16.

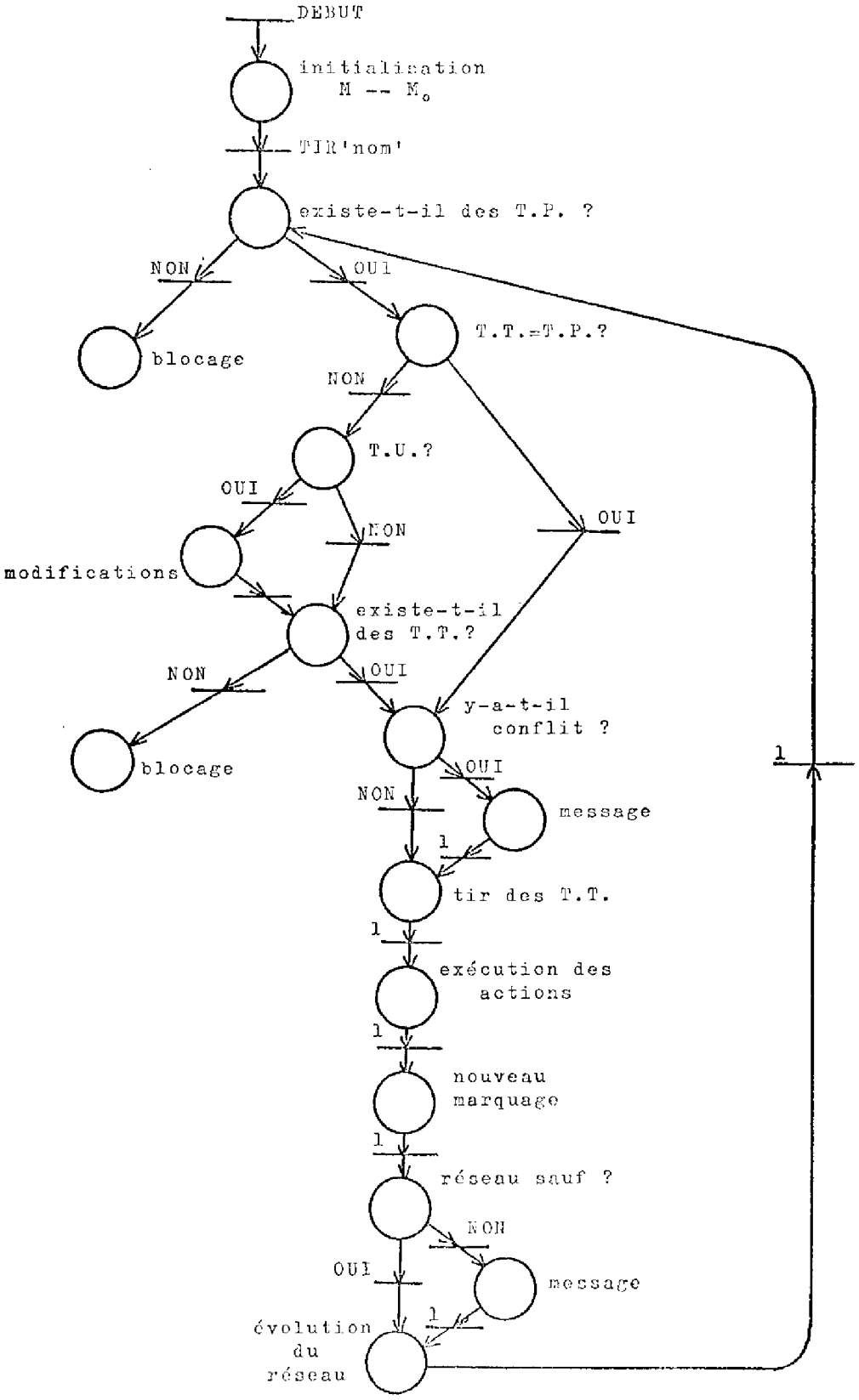


FIGURE III.17.

REMARQUE

Pour étudier le comportement de systèmes interconnectés, il est possible d'écrire un réseau de PETRI dont chaque place aura pour action associée la simulation d'un réseau de PETRI représentant un de ces systèmes,et d'effectuer la simulation de ce "réseau de réseaux" [AZE].

III.6. ENCOMBREMENT MEMOIRES ET TEMPS DE CALCUL

Nous avons travaillé sur un terminal APL connecté au Centre de Calcul I.B.M. d'Orléans, équipé d'un ordinateur I.B.M. 360-145.

La version du langage APL, utilisée pour l'écriture de ces programmes, est la version "APL-S.V." qui a l'avantage de posséder des instructions très puissantes, telles l'ordre "exécute" [IBM].

III.6.1. Place mémoire

Les zones de travail dont dispose un utilisateur APL ont une dimension de l'ordre de 120.000 octets. L'ensemble des fonctions permanentes représente un encombrement mémoire d'environ 30.000 octets.

Après l'étude d'un certain nombre de réseaux de PETRI, on peut chiffrer à moins de 8.000 octets supplémentaires la place occupée par les éléments caractéristiques et les fonctions spécifiques d'un réseau de dix places et dix transitions.

Il faut bien noter à ce sujet que ces fonctions sont un outil d'aide à la mise au point ; on aura donc intérêt à se limiter à des systèmes de taille réduite, de l'ordre de vingt places et vingt transitions, de façon à garder des graphes facilement lisibles. Pour des systèmes plus complexes, il sera préférable de travailler par affinements [VAL.1], comme nous l'avons indiqué au paragraphe II.5.

III.6.2. Temps de calcul

La durée d'une séance de travail, en mode conversationnel, en langage APL, est relativement courte. Pour les phases de description du graphe de commande et de déclaration du graphe de données, ainsi que

pour la simulation, le temps de connexion est pratiquement limité aux temps d'écriture de l'imprimante d'une part, de l'utilisateur d'autre part.

Pour un schéma à réseau de PETRI dont le graphe de commande est un réseau à dix places et dix transitions environ, la durée de l'étude complète sera de l'ordre de vingt minutes. Quant au temps de calcul "machine", pour ce même schéma, il est inférieur à dix secondes, dont la majeure partie pour l'analyse, plus exactement pour la construction du graphe des marquages conséquents. (Ce dernier résultat dépend évidemment de l'ordinateur utilisé).

III.7. CONCLUSION

Pour résumer la procédure à utiliser pour l'étude d'un système représenté par un schéma à réseau de PETRI, nous pouvons en donner une représentation par le réseau de PETRI de la figure III.18.

Ces programmes ont été écrits de façon qu'un utilisateur, voulant faire l'étude d'un schéma à réseau de PETRI, n'ait qu'un minimum de choses à connaître concernant les algorithmes utilisés. Les instructions nécessaires sont données au moment utile. La plupart des entrées de données sont contrôlées, les erreurs ou oublis signalés, chaque sous-programme vérifiant, avant de commencer, que tous les éléments nécessaires sont acquis en mémoire.

Dans le prochain chapitre, nous nous proposons de donner quelques exemples soulignant l'intérêt de l'outil que nous avons mis au point. Cet outil doit permettre au concepteur de représenter un système, de rechercher d'éventuelles erreurs de conception grâce à une analyse du système, puis de le simuler afin de valider son projet.

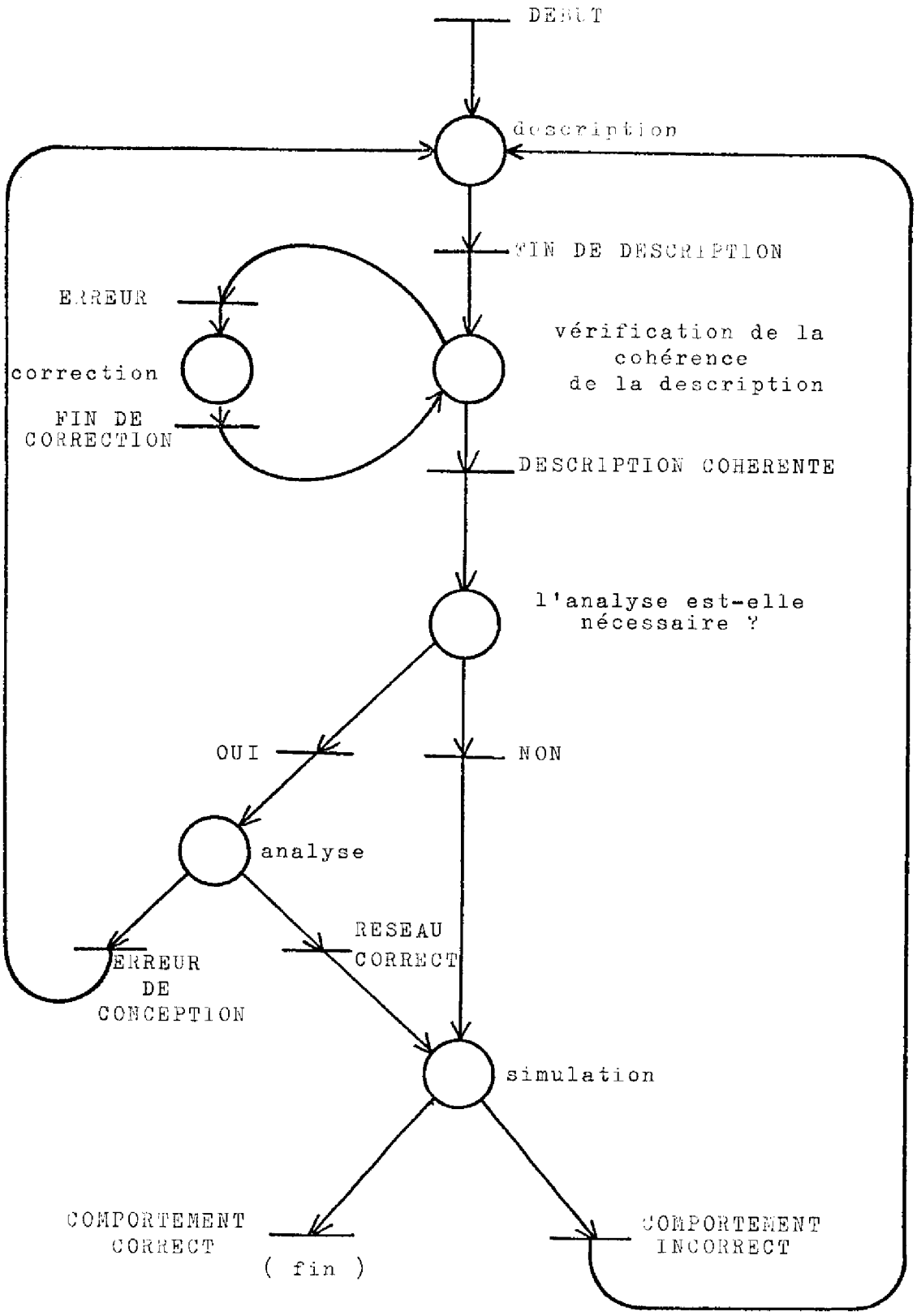


FIGURE III.18.

CHAPITRE IV

EXEMPLES D'APPLICATION

-

IV.1. INTRODUCTION

Nous nous proposons, dans ce dernier chapitre, de donner trois exemples d'application, dans le but d'illustrer les méthodes exposées. Nous avons choisi des exemples relativement simples pour qu'ils demeurent présentables et facilement compréhensibles, mais suffisamment variés pour qu'ils fassent intervenir la majorité des notions introduites dans ce mémoire.

Le premier système étudié, le plus simple, est la partie émettrice d'une cellule d'émission et de réception universelle asynchrone (U.A.R.T.) ; cet exemple permettra d'étudier un parallélisme.

Puis nous aborderons le problème très classique du "producteur-consommateur". Nous avons étudié, de manière générale, le fonctionnement d'un tel système au paragraphe I.6.3. ; ici la représentation par schéma à réseau de PETRI sera assez différente, car basée sur un cahier des charges bien précis.

Le dernier exemple, plus complexe, représentera le fonctionnement d'un bus servant d'interface entre plusieurs modules.

IV.2. PREMIER EXEMPLE : L'U.A.R.T.

Considérons un système qui reçoit des mots du monde extérieur, par exemple des informations données par un capteur numérique ou un microcalculateur ; le système met un mot dans un registre tampon, puis l'émet sur une ligne de transmission ; l'émission se fait en série.

IV.2.1. Graphe de commande

Enumérons les différentes opérations successives à représenter pour décrire la réception et l'émission d'un mot, lorsque le système est actif :

- réception du mot et validation de son contenu,
- chargement de ce mot dans le registre tampon,

- transfert dans un registre d'émission,
- émission du mot sur la ligne.

Nous pouvons en déduire un premier graphe ; la figure IV.1. montre l'ordre de ces opérations.

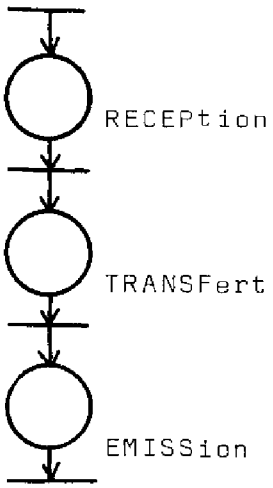


FIGURE IV.1.

A partir de cette description sommaire, il faut considérer que plusieurs mots peuvent arriver successivement. Il y aurait une perte de temps inutile si l'on attendait la fin de l'émission du premier mot pour commencer la réception du suivant : le matériel permet de charger un deuxième mot dans le registre tampon dès que le contenu de ce dernier a été transféré dans le registre d'émission. Après cette opération de transfert, il y a un choix à faire, selon la présence d'un nouveau mot, entre la réception de celui-ci et la désactivation du système (figure IV.2.).

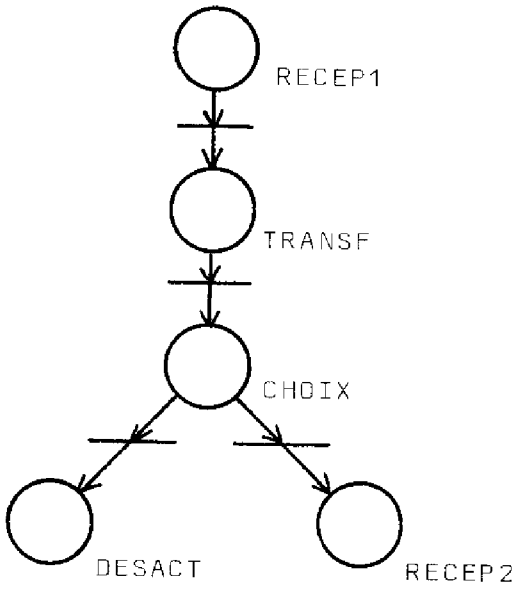


FIGURE IV.2.

Ce choix se fait parallèlement à l'émission du premier mot. La désactivation du système ne doit se faire qu'après l'émission du dernier mot. S'il y a réception d'un nouveau mot, il faut naturellement recommencer la séquence transfert-émission.

Compte tenu de toutes ces conditions, nous pouvons représenter le parallélisme par la figure IV.3.

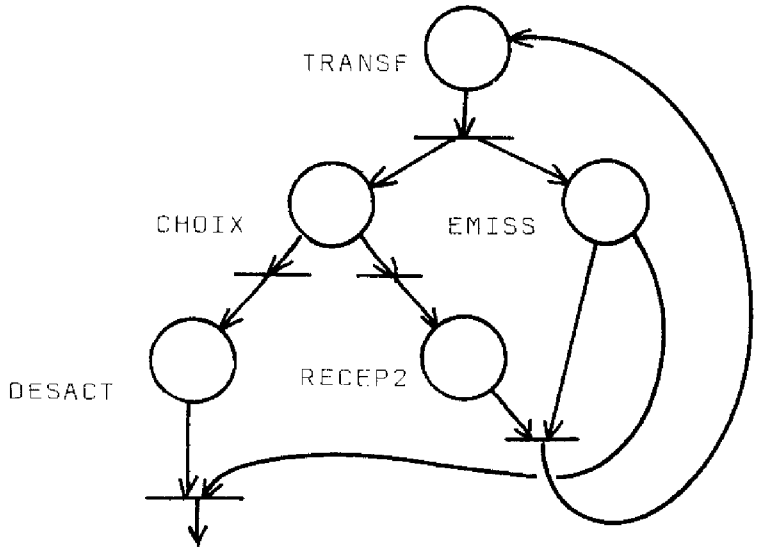


FIGURE IV.3.

Enfin, il est utile de décrire un système en partant d'un état initial facile à repérer : la lisibilité de la description est accrue. Aussi, nous ajouterons une place supplémentaire, appelée "INIT", pour représenter un état de repos du système attendant l'ordre d'activation. Cette place permet de reboucler le graphe, ce qui traduit la propriété du système de pouvoir fonctionner de manière répétitive ; l'absence de cette place suffirait à empêcher le réseau d'être vivant et propre.

Il est alors possible de donner le réseau de PETRI complet du système de commande : c'est celui de la figure IV.4. Dans cet exemple, c'est aux places du réseau que nous associons les opérations : le marquage de chacune d'elles correspond à l'activation d'un opérateur, indiqué sous le nom de la place ; nous expliciterons plus loin ces opérations.

IV.2.2. Graphe de données

Les cellules mémoires dont nous devons disposer représentent :

- le mot reçu du monde extérieur — MOT
- le contenu du registre tampon — RT
- le contenu du registre d'émission — RE
- le mot émis — LIGNE
- l'ordre d'activation ou de désactivation — MA
- la validation du contenu de MOT — M

Voyons maintenant sur quelles mémoires et de quelle façon, agit chaque opérateur :

- place ACTIV : l'opérateur OA recherche si un mot est arrivé du monde extérieur ; si "MOT" n'est pas vide, il met la mémoire M à 1,
- place RECEP 1 : l'opérateur OR transfère le contenu de la mémoire MOT dans le registre tampon (RT ← MOT). De plus, mais ceci uniquement pour les besoins de la simulation, la mémoire MOT est vidée et M reçoit 0.

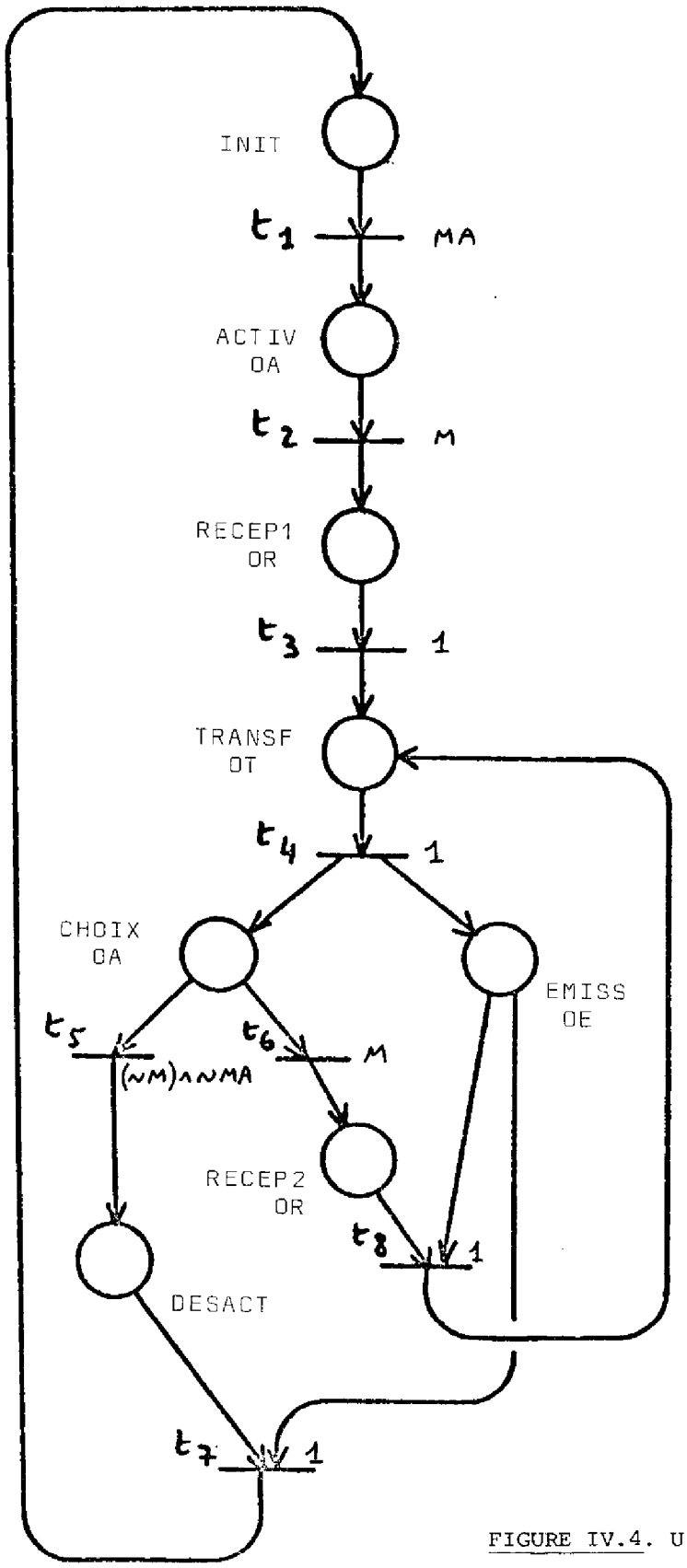
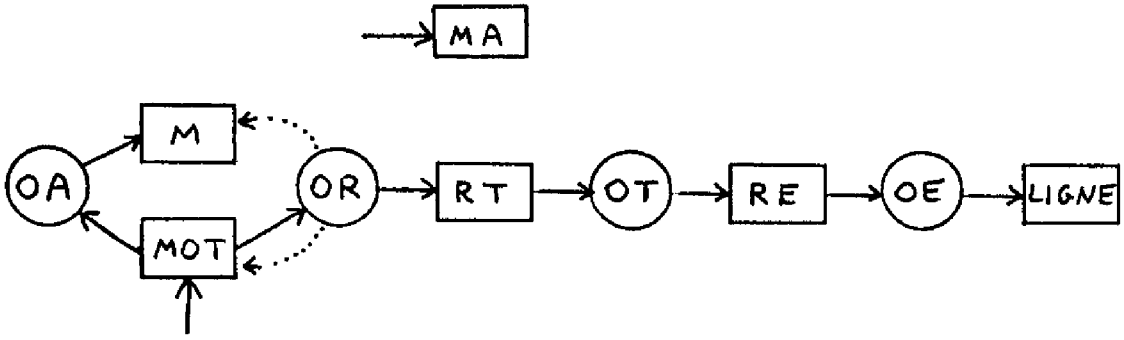


FIGURE IV.4. U.A.R.T.

Graphe de commande.

- place TRANSF : l'opérateur OT transfère le contenu de RT dans le registre RE,
- place EMISS : OE émet le contenu de RE sur la LIGNE,
- place CHOIX : c'est l'opérateur OA qui est à nouveau activé,
- place RECEP 2 : comme en RECEP 1, l'action est exécutée par l'opérateur OR,
- places INIT et DESACT : ces deux places sont des positions d'attente. Aucune action (donc aucun opérateur) ne leur est associée.

Tout ceci se traduit par le graphe de données de la figure IV.5.



U.A.R.T. Graphe de données, pour l'analyse.

```
OA : [1]  M←1 ≤ ρ MOT  ∇

OR : [1]  RT←MOT
     [2]  MOT←⍳M←0  ∇

OT : [1]  RE←RT  ∇

OE : [1]  LIGNE←RT
     [2]  'LIGNE = ',10 0 ⍕ LIGNE  ∇
```

pour la simulation (.....→)

FIGURE IV.5.

C'est le monde extérieur qui donnera l'ordre d'activation ou de désactivation du système.

Les opérateurs, écrits sous forme de fonctions APL, ne comportent ici que des instructions simples, réalisant les chargements des mémoires ou les lectures et écritures.

IV.2.3. Prédicats et transitions-utilisateurs

Il ne reste plus qu'à déterminer les conditions associées aux transitions, c'est-à-dire les expressions booléennes qui devront être vraies pour que les transitions soient validées.

La transition t_1 sera validée par un événement extérieur, la mise en marche du système. Elle sera donc déclarée transition-utilisateur et son prédicat sera "MA".

Les deux transitions t_2 et t_6 reçoivent pour condition logique "M". Ces transitions ne seront validées que si la cellule MOT n'est pas vide.

La transition t_5 sera aussi une transition-utilisateur. Son prédicat sera "$(\sim M) \wedge \sim MA$". Elle permettra à l'utilisateur de décider la désactivation totale du système, ce qui signifie "appuyer"sur le bouton "Arrêt" .

Les transitions t_5 et t_6 sont en conflit, mais ce conflit est résolu car leurs conditions logiques ne peuvent être vraies simultanément.

Enfin, les transitions t_3, t_4, t_7 et t_8 seront tirées dès qu'elles seront sensibilisées : l'expression booléenne, qui leur est associée, est donc simplement "1".

REMARQUE

Au cours de la simulation, il faut tenir compte du fait qu'un mot, après avoir été transféré dans le registre tampon, n'occupe plus la mémoire MOT. La programmation nous oblige à rajouter une instruction pour vider cette cellule ("MOT←⍳0" et "M←0"). (Les arcs correspondants du graphe de données sont en pointillés).

IV.2.4. Etude

Le schéma à réseau de PETRI est maintenant complet et nous pouvons l'analyser et le simuler.

Description

La description du graphe de commande de ce schéma ne pose aucun problème vu l'absence de boucles élémentaires, de transitions ou de places sources ou puits. Nous donnerons, pour marquage initial, un jeton dans la place INIT, et nous déclarerons t_1, t_5, t_6 et t_2 transitions-utilisateurs.

Les listings des programmes de description sont donnés par les tableaux IV.6. (graphe de données) et IV.7. (graphe de commande).

```
   DECLARE

NOM DU RESEAU  :  UART
DONNER LES NOMS DES MEMOIRES UTILISEES :
M,MA,RT,RE,MOT,LIGNE
//

DONNER L'OPERATEUR ASSOCIE A CHAQUE PLACE :
INIT   :
ACTIV  : OA
RECEP1 : OR
TRANSF : OT
EMISS  : OE
CHOIX  : OA
DESACT :
RECEP2 : OR

INDIQUER LES MEMOIRES D'ENTREE ET DE SORTIE
POUR CHAQUE OPERATEUR :

OA : ;MOT,M
OR :MOT;RT
OE :RE;LIGNE
OT :RT;RE
 //
```

FIGURE IV.6.

```
     DESCRIPTION

NOM DU RESEAU : UART1
NOMBRE DE PLACES :    8
NOMBRE DE TRANSITIONS : 8
ACTIONS SUR LES PLACES OU SUR LES TRANSITIONS? (P OU T) :  P

DECRIRE LE RESEAU PAR RAPPORT AUX PLACES :
-----------------------------------------

SI     :    T1
NOM DE LA PLACE : ACTIV
FAIRE :    OA
ET ALLER A : T2

SI     :    T2
NOM DE LA PLACE :RECEP1
FAIRE :    OR
ET ALLER A : T3
     .
     .
     .
     .
SI     :    T6
NOM DE LA PLACE :RECEP2
FAIRE :    OR
ET ALLER A : T8

VOULEZ-VOUS CORRIGER LA DESCRIPTION D'UNE PLACE ?
NON

DECRIRE LE RESEAU PAR RAPPORT AUX TRANSITIONS :
-----------------------------------------------

DEPUIS : INIT
SI   MA
TIRER :  T1
ET ALLER A : ACTIV

DEPUIS : ACTIV
SI  M
TIRER :  T2
ET ALLER A : RECEP1
     .
     .
     .
     .
DEPUIS : RECEP2,EMISS
SI  1
TIRER :  T8
ET ALLER A : TRANSF

VOULEZ-VOUS CORRIGER LA DESCRIPTION D'UNE TRANSITION ?
NON

LE RESEAU EST ACCEPTE . SA DESCRIPTION SEMBLE CORRECTE .
```

FIGURE IV.7.

Analyse

L'analyse du réseau fait apparaître sept marquages différents. Elle indique que le schéma décrit est sauf, vivant, propre et déterminé (tableau IV.8.).

Nous pouvons en conclure que le système a un fonctionnement correct.

```
        ANA

NOM DU RESEAU  :  UART1

MARQUAGES
 1 0 0 0 0 0 0 1 0
 0 1 0 0 0 0 0 0 0
 0 0 1 0 0 0 0 0 0
 0 0 0 1 0 0 0 0 1
 0 0 0 0 1 1 1 0 0
 0 0 0 0 1 0 0 0 0
 0 0 0 0 0 1 0 0 0
 0 0 0 0 0 0 1 0 0
PERES
  1  2  3  4  5  6  7  8  9
  0  1  2  3  4  5  5  6  7
```

```
GRAPHE
  1___ 2___ 3___ 4___ 5___ 6___ 1
                      |
                      L___ 7___ 4

  1 : INIT
  2 : ACTIV
  3 : RECEP1
  4 : TRANSF
  5 : EMISS  CHOIX
  6 : EMISS  DESACT
  7 : EMISS  RECEP2

LE RESEAU EST SAUF .

LE RESEAU EST VIVANT ET PROPRE .

LE SCHEMA EST DETERMINE .
```

TABLEAU IV.8.

Simulation

Le tableau IV.9. montre un exemple de simulation.

Trois séquences d'entrée ont été données au préalable : la première met le mot 1101100 dans la mémoire d'entrée, la seconde 10100111, la troisième "rien".

```
     SIMUL
NOM DU RESEAU : UART
Y-A-T-IL PLUSIEURS SEQUENCES D'ENTREE ?
OUI
DONNER LA LISTE DES FONCTIONS APL
REALISANT CES SEQUENCES D'ENTREE :(PUIS //)
MOT1,MOT2,MOT3
//
DEFINIR CES 3 FONCTIONS APL,
PUIS APPELER 'TIRS'.
     ∇MOT1
[1] MOT←1101100 ∇
     ∇MOT2
[1] MOT←10100111 ∇
     ∇MOT3
[1] MOT←10 ∇

     MA←1
     TIRS

TRANSITION TIREE : 1
PLACE : ACTIV

TRANSITION TIREE : 2
PLACE : RECEP1

TRANSITION TIREE : 3
PLACE : TRANSF

  LIGNE = 1101100
TRANSITIONS POSSIBLES : 5,6
     ''       TIRABLES  :
CONDITIONS 5,6 NON SATISFAITES .
5 : (~MA)∧(~M)
6 : M
MODIFS ,PUIS / .
/
TRANSITION TIREE : 4
PLACES : EMISS,CHOIX

TRANSITION TIREE : 6
PLACES : EMISS,RECEP2

TRANSITION TIREE : 8
PLACE : TRANSF

  LIGNE = 10100111
TRANSITIONS POSSIBLES : 5,6
     ''       TIRABLES  :
CONDITIONS 5,6 NON SATISFAITES .
5 : (~MA)∧(~M)
6 : M
MODIFS ,PUIS / .
/
TRANSITION TIREE : 4
PLACES : EMISS,CHOIX

TRANSITIONS POSSIBLES : 5,6
     ''       TIRABLES  :
CONDITIONS 5,6 NON SATISFAITES .
5 : (~MA)∧(~M)
6 : M
MODIFS ,PUIS / .
MA←0
/
TRANSITION TIREE : 5
PLACE : EMISS,DESACT

TRANSITION TIREE : 7
PLACE : INIT

TRANSITION POSSIBLE : 1
     ''      TIRABLE  :
CONDITION 1 NON SATISFAITE .
1 : MA
MODIFS ,PUIS / .
/

 *** FIN DE SIMULATION ***
```

TABLEAU IV.9.

Après la mise en marche du système (MA=1), la simulation commence : le premier mot est émis sur la ligne ; puis la simulation s'arrête, les transitions t_5 et t_6 n'étant pas validées. Si l'utilisateur ne met pas à 0 la mémoire MA, la deuxième séquence d'entrée est "mise en place" : on peut alors tirer t_6, transférer le deuxième mot dans le registre tampon, puis l'émettre. Ensuite, le troisième mot étant "vide", M ne passera pas à 1, et la simulation sera stoppée tant que l'utilisateur ne donnera pas l'ordre de désactivation du système.

Nous aurions pu réaliser l'émission de chaque mot bit par bit ; mais, l'émission se faisant en série, cela n'aurait rien changé au fonctionnement général.

L'étude détaillée de ce premier exemple nous a permis de montrer d'une part la représentation d'un parallélisme simple, d'autre part la façon d'utiliser le programme de simulation.

IV.3. PRODUCTEUR-CONSOMMATEUR

Le problème consiste à coordonner les fonctionnements d'une unité de production et d'une unité de consommation, reliées entre elles par un magasin.

IV.3.1. Fonctionnement du système

Le magasin peut contenir une quantité maximale de Nmax objets produits ; les deux unités ne peuvent pas accéder simultanément à ce magasin.

L'unité de production ne peut déposer un objet dans le magasin que si celui-ci n'est pas plein, tandis que l'unité de consommation ne peut en retirer un que s'il n'est pas vide.

Enfin, les objets sont produits, déposés, retirés et consommés un par un.

Aucun signal n'est échangé entre l'unité de production, le magasin et l'unité de consommation. C'est l'automatisme de contrôle qui reçoit et émet tous les signaux d'information (figure IV.10.).

Nous désignerons par :

P : l'ordre de production d'un objet
D : l'ordre de dépôt
C : l'ordre de consommation d'un objet
R : l'ordre de retrait
FP : le signal de fin de production
FD : le signal de fin de dépôt
FC : le signal de fin de consommation
FR : le signal de fin de retrait

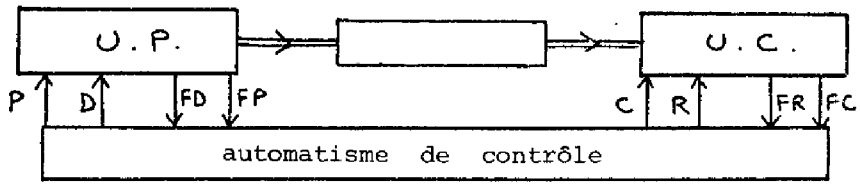


FIGURE IV.10.

IV.3.2. Graphe de commande

Du côté de l'unité de production, nous pouvons distinguer deux étapes essentielles dans le fonctionnement, auxquelles nous ajouterons une position d'attente.

Pour cette unité seule, on peut donc écrire le graphe en forme de boucle de la figure IV.11.a. Il en est de même du côté de l'unité de consommation. Le graphe bouclé de la figure IV.11.b. représente les deux étapes de retrait et de consommation, précédées d'une position d'attente.

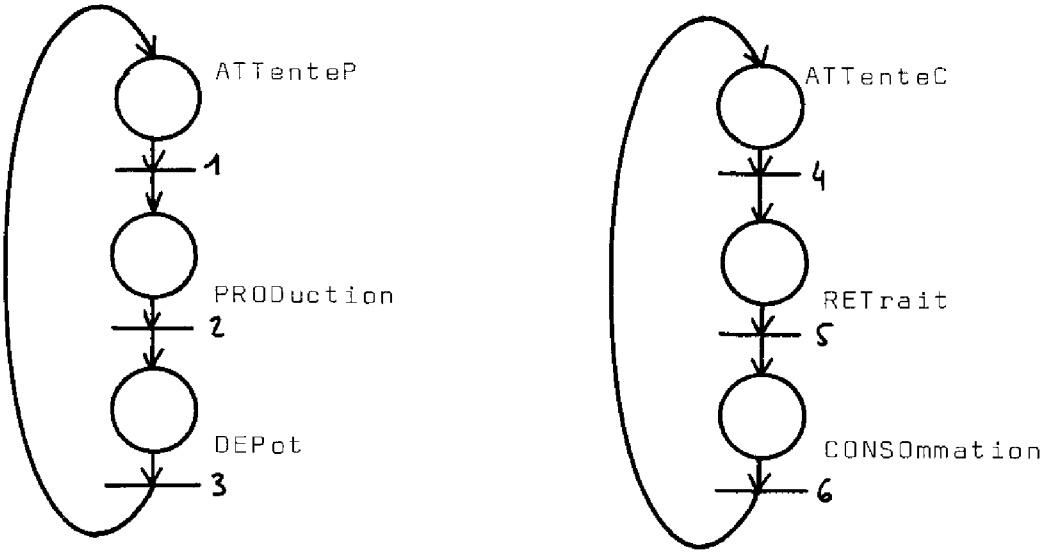


FIGURE IV.11.

Mais ces deux séries d'opérations ne sont pas indépendantes puisqu'elles sont toutes les deux conditionnées par l'état du stock contenu dans le magasin. Il faut donc réaliser la coordination des fonctionnements des deux unités. Trois contraintes essentielles doivent être respectées :

1. impossibilité de retrait et de dépôt simultanés,
2. impossibilité de retrait si le nombre N, d'objets en stock, est nul,
3. impossibilité de dépôt si N est égal à Nmax.

Comme pour le réseau de PETRI de l'U.A.R.T., nous allons associer les actions aux places du graphe de commande. Pour représenter la première contrainte énoncée, c'est-à-dire l'exclusivité des opérations de retrait et de dépôt, nous devons faire en sorte que les marquages des places correspondantes soient incompatibles. Ceci est réalisé en mettant en conflit les deux transitions t_2 et t_4 qui précèdent les opérations concernées. Cette place représente ainsi la possibilité d'accéder au magasin : si elle a un jeton, l'accès est libre ; si elle est vide, l'accès est interdit.

Le jeton est enlevé de cette place pour commencer une opération de dépôt ou de retrait ; puis il doit y être remis dès la fin de cette opération : cette place sera donc place précédente des deux transitions t_2 et t_4, et place suivante des deux transitions t_3 et t_5 marquant les fins de dépôt et de retrait.

Nous obtenons ainsi la configuration donnée par la figure IV.12 (la place en conflit est appelée STO).

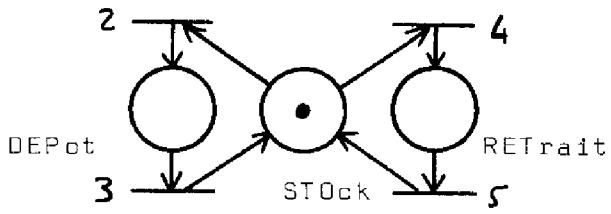


FIGURE IV.12.

Rajouter cette place a résolu le problème de l'accès au magasin. Mais elle ne rend pas compte du nombre N d'objets stockés à chaque instant. Il reste à respecter les contraintes 2 et 3 énoncées plus haut.

La méthode la plus simple consiste à faire intervenir N dans les expressions logiques validant les transitions. C'est celle que nous allons utiliser. Nous verrons plus loin qu'il est possible d'exprimer les deux contraintes à l'aide de places et de transitions supplémentaires ; mais à ce moment-là, le réseau de PETRI n'est plus sauf, il est seulement borné.

IV.3.3. Prédicats et transitions-utilisateurs

Examinons maintenant les fonctions logiques associées aux transitions. Nous laissons à l'utilisateur, c'est-à-dire au monde extérieur, la possibilité de donner les quatre ordres : P, R, C et D.

Nous aurons ainsi quatre transitions-utilisateurs : t_1, t_2, t_4 et t_5.

- La validation de t_1 est donnée par la mise à 1 de P.
- Pour tirer t_2, il faut que l'ordre de dépôt ait été donné, (D=1), que l'étape de production soit terminée, (FP=1), et enfin qu'il reste une case vide dans le magasin ($N < Nmax$) ; d'où l'expression : "$FP \wedge D \wedge N < Nmax$".
- Dès la fin du dépôt, (FD=1), on peut tirer t_3 et repasser en position d'attente.
- Outre la demande de retrait (R=1), le magasin doit contenir un objet au moins ($N > 0$) pour que t_4 soit validée ; d'où la fonction logique : "$R \wedge N > 0$".
- Un objet ne peut être consommé que s'il a été retiré du magasin (FR=1), et si l'ordre en a été donné, (C=1) ; la condition sur t_5 sera : "$C \wedge FR$".
- Enfin, le signal de fin de consommation (FC=1) validera t_6 et ramènera le système en position d'attente.

Notre système de commande est maintenant complet. Nous pouvons le représenter par le réseau de PETRI de la figure IV.13.

En position initiale, trois places sont marquées : les deux places d'attente et celle réglant l'accès au magasin.

IV.3.4. Graphe de données

Etudions en détail les opérations exécutées pour chaque place :

- Production : l'opérateur PROD remet P à zéro, puis, à la fin de sa tâche, met FP à 1.
- Dépôt : DEP remet D à 0, augmente N d'une unité, et met à 1 le signal FD.
- Retrait : RET remet R à 0, diminue N d'une unité, et met à 1 le signal FR.
- Consommation : l'opérateur CONSO met à 1 le signal FC et C à 0.
- Stock : à la place STO n'est exécutée aucune instruction ; nous avons vu le rôle particulier de cette place.
- Attente : les opérateurs ATTC et ATTP réalisent les mises à 0 des signaux FD, FP et FR, FC.

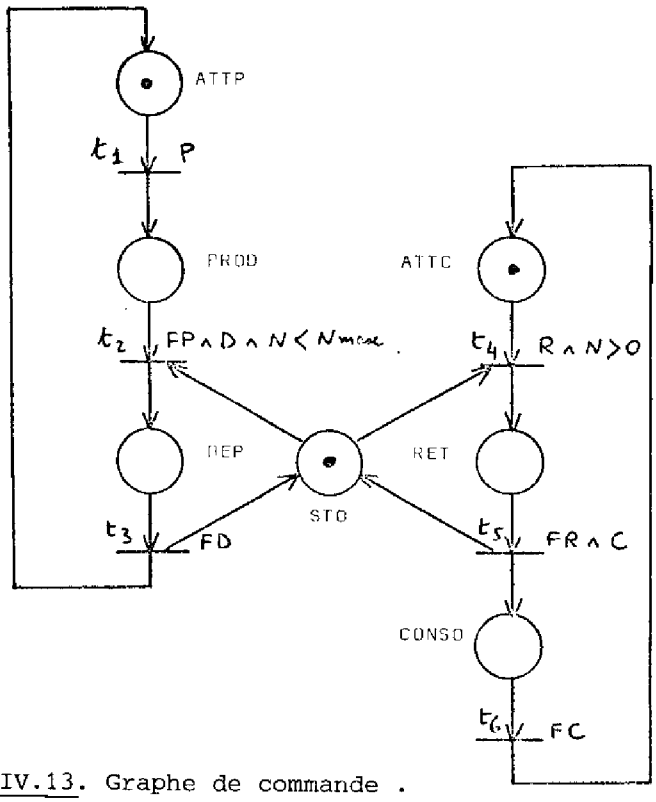
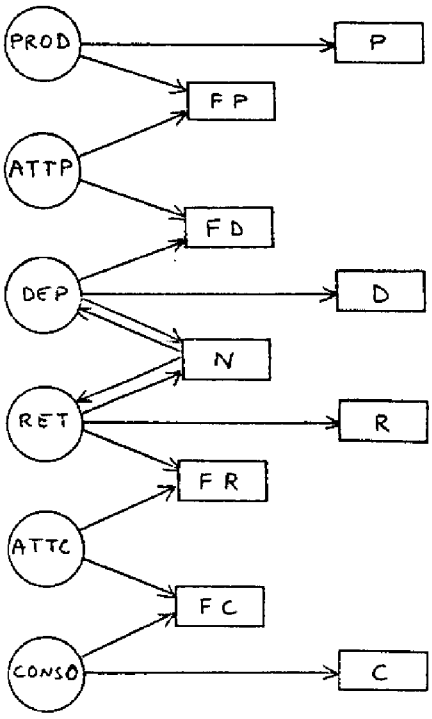


FIGURE IV.13. Graphe de commande .

Le graphe de données est représenté par la figure IV.14.



PROD : FP←NP←0

DEP : N←N+FD←ND←0

RET : N←N−FR←NR←0

CONSO : FC←NC←0

ATTP : FP←FD←0

ATTC : FR←FC←0

FIGURE IV.14. Graphe de données .

IV.3.5. Etude

Nous pouvons maintenant étudier le schéma à réseau de PETRI ainsi défini.

La description des deux graphes ne pose aucun problème particulier. Nous ne présenterons pas les résultats de ces programmes, car ceux-ci n'apporteraient rien de nouveau par rapport à ceux du premier exemple.

L'analyse du schéma réalise la construction de l'arbre des marquages conséquents représenté figure IV.15. Les résultats obtenus indiquent que le réseau est sauf, vivant et propre, et que le schéma est déterminé.

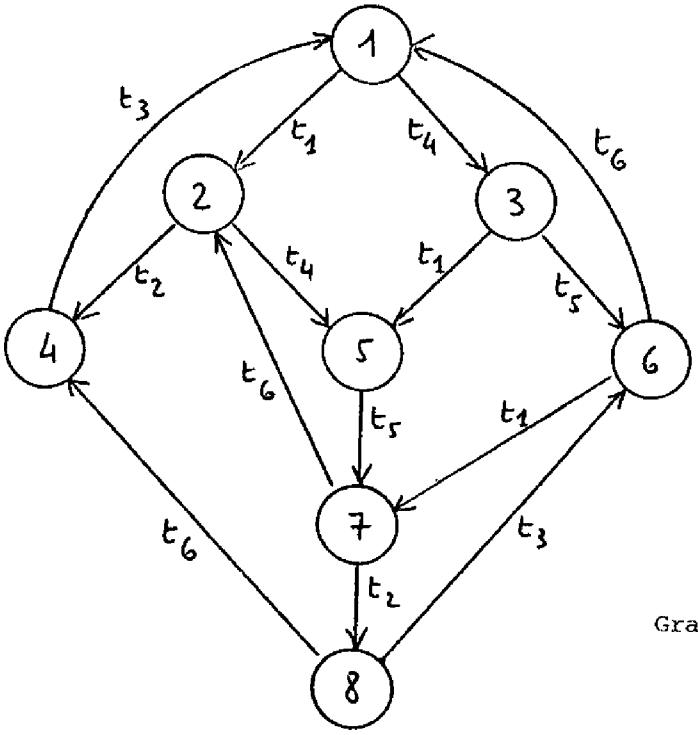
Enfin, le tableau IV.16. montre une partie d'une simulation : à partir du marquage initial (ATTP, ATTC, STO), et d'une quantité d'objets stockés non nulle, on donne l'ordre d'en retirer un, puis, simultanément, de le consommer et d'en produire un autre.

IV.3.6. Remarque

Il est possible de représenter le fonctionnement de ce même système en n'utilisant qu'un graphe de commande : un réseau de PETRI borné, mais non sauf. La solution proposée [SIF] consiste à rajouter au réseau de PETRI de la figure IV.13., deux places pouvant contenir plusieurs jetons : le marquage de l'une sera le nombre de cases vides dans le magasin, l'autre indiquera le nombre d'objets stockés, ou les cases pleines.

La figure IV.17. montre la partie du réseau de PETRI modifiée, où l'on suppose que le magasin peut contenir quatre objets.

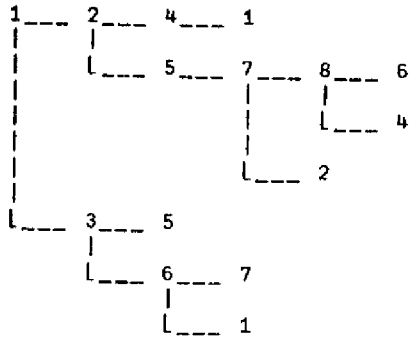
L'opération de dépôt enlève un jeton dans la place C.V. (cases vides) et en met un dans la place C.P. (cases pleines). Inversement, un retrait enlève un jeton de C.P. et en ajoute un à la place C.V. Avec cette représentation, les expressions logiques ne dépendent plus de N. Pour l'exemple, il y a trois objets dans le magasin, donc une seule case vide.



Graphe des marquages conséquents.

MAG1

GRAPHE




```
1 : ATTP  STO   ATTC
2 : PROD  STO   ATTC
3 : ATTP  RET
4 : DEP   ATTC
5 : PROD  RET
6 : ATTP  STO   CONSO
7 : PROD  STO   CONSO
8 : DEP   CONSO
```

LE RESEAU EST SAUF .
LE RESEAU EST VIVANT ET PROPRE .
LE SCHEMA EST DETERMINE .

FIGURE IV.15.

```
TIR 'MAG'
.
.
.
.
TRANSITIONS POSSIBLES : 1,4
    ''       TIRABLES  :
CONDITIONS 1,4 NON SATISFAITES .
1 : P
4 : R∧N>0
MODIFS ,PUIS / .
R←1
/
TRANSITION TIREE : 4
PLACES : ATTP,RET

TRANSITIONS POSSIBLES : 1,5
    ''       TIRABLES  :
CONDITIONS 1,5 NON SATISFAITES .
1 : P
5 : FR∧C
MODIFS ,PUIS / .
P←1
C←1
/
TRANSITIONS TIREES : 1,5
PLACES : PROD,STO ,CONSO

TRANSITIONS POSSIBLES : 2,6
    ''       TIRABLES  : 6
CONDITION 2 NON SATISFAITE .
2 : FP∧D∧N<NMAX
MODIFS ,PUIS / .
/
TRANSITION TIREE : 6
PLACES : PROD,STO ,ATTC

TRANSITIONS POSSIBLES : 22,4
    ''       TIRABLES  :
CONDITIONS 2,4 NON SATISFAITES .
2 : FP∧D∧N<NMAX
4 : R∧N>0
MODIFS ,PUIS / .
D←1
R←1
/
LES  2 TRANSITIONS 2,4
SONT EN CONFLIT .(PLACE STO ) .
LA TRANSITION 4 SERA CHOISIE ARBITRAIREMENT .
TRANSITION TIREE : 4
PLACES : PROD,RET
.
.
.
.
.
.
.
```

TABLEAU IV.16.

Cette solution supprime le compteur N dans le graphe de données, mais le graphe de commande n'est plus sauf.

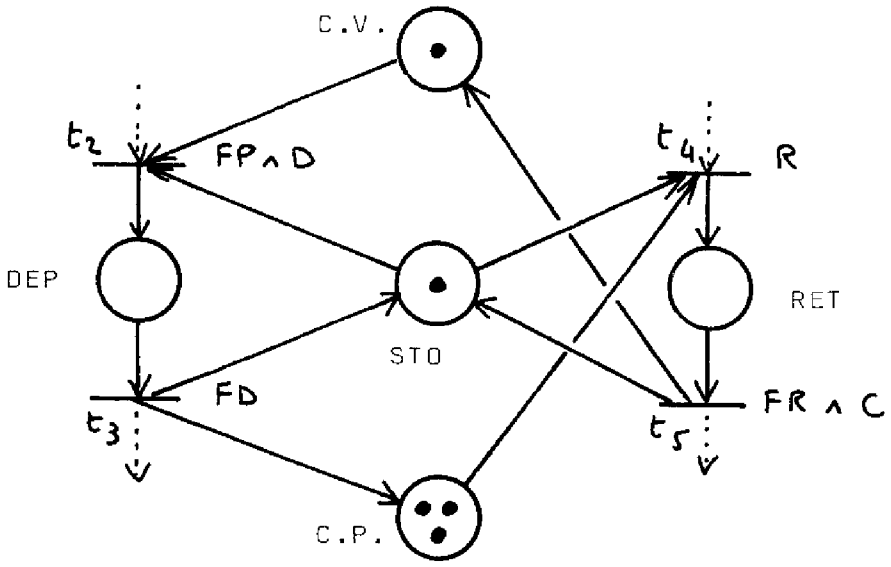


FIGURE IV.17.

IV.4. L'UNIBUS

Le troisième exemple que nous avons choisi de présenter est un moyen de communication entre plusieurs modules [LED] , par exemple des processeurs ou encore des mémoires.

Nous n'étudierons pas ce système aussi complètement que les deux précédents. Nous construirons simplement un graphe de commande, sans préciser les éléments du graphe de données. Nous n'effectuerons que l'analyse du réseau de PETRI seul et une simulation simplifiée.

IV.4.1. Fonctionnement du système

Plusieurs modules échangent des informations entre eux par l'intermédiaire d'un bus unique. A tout instant, il ne peut y avoir qu'un seul échange à la fois.

Chaque échange d'informations entre deux modules comprend plusieurs étapes :

- le premier module fait une demande d'échange,
- il attend l'accord d'un arbitre,
- après accord, le deuxième module concerné est sélectionné,
- on attend que le bus soit libre,
- quand le bus est libre, l'échange est effectué,
- les modules redeviennent oisifs,
- le bus redevient libre.

IV.4.2. Graphe de commande

La succession des quatre étapes principales, pour un module i, forme la boucle de la figure IV.18.

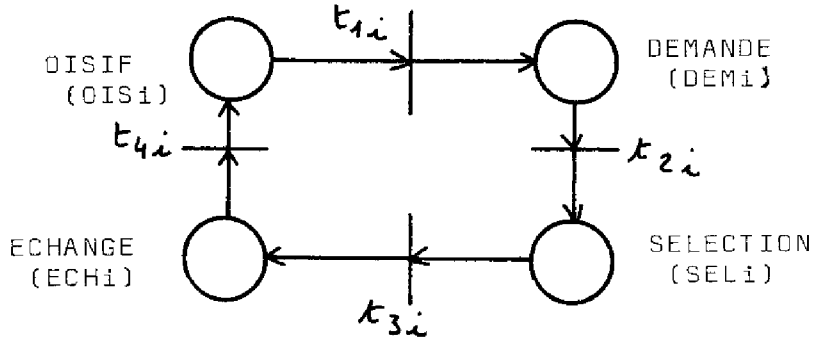


FIGURE IV.18.

Nous avons besoin d'une boucle identique pour chacun des n modules : ceci forme l'essentiel du graphe que nous allons construire peu à peu.

Demande

Lorsqu'un module i veut effectuer un échange, il émet un signal de demande personnel D_i. On réalise la fonction "OU" des n signaux D_i correspondant aux n modules : lorsque le résultat de cette fonction est "1", l'arbitre envoie son accord.

Sur notre graphe de commande, les transitions t_{1i} seront validées quand "D_i=1", et l'arbitre ne pourra donner un accord que si l'expression "$D_1 \vee D_2 \vee \ldots \vee D_n$" vaut 1 (figure IV.19.a.).

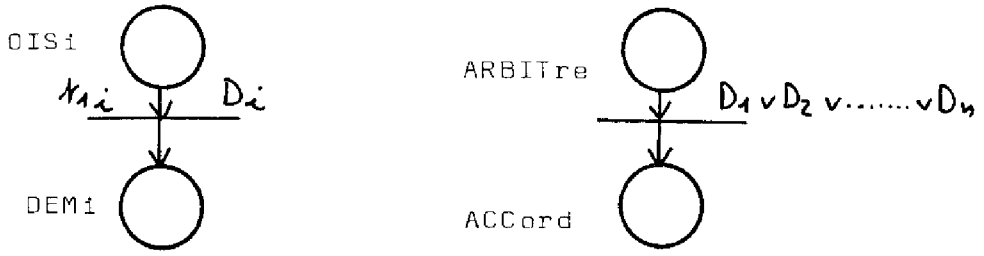


FIGURE IV.19.a.

Un module peut faire une demande à condition qu'il ne soit pas, au même instant, en train d'effectuer un échange, et qu'il n'ait pas fait une autre demande qui n'est pas encore satisfaite.

Ceci se traduit par la configuration de la figure IV.19.b. La place ND_i (non-demande) perd son jeton au moment où la demande est faite, puis le reprend quand on passe à l'étape de sélection.

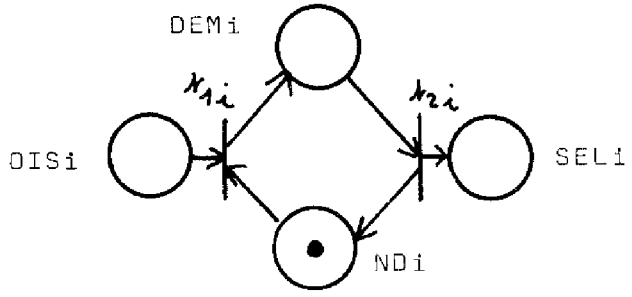


FIGURE IV.19.b.

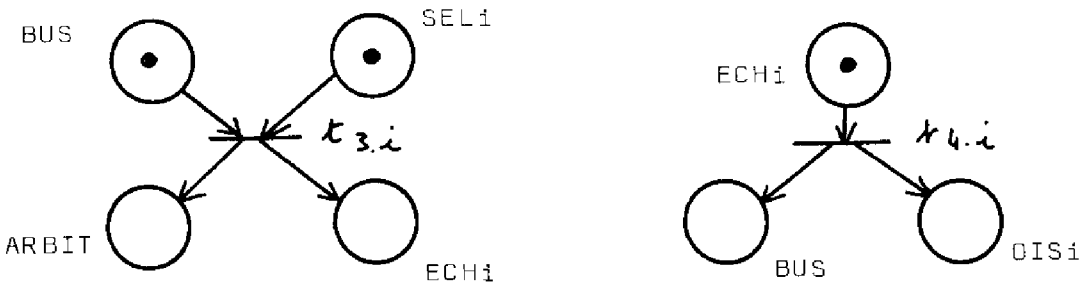
Echange

Deux échanges ne peuvent avoir lieu simultanément, puisque nous disposons d'un bus unique. Le problème d'occupation du bus est le même qu'au paragraphe IV.3.2. : de la même façon que nous avons représenté l'accès au magasin pour les opérations de dépôt et de retrait, nous mettons en conflit les n transitions t_{3i}, précédant les n places "ECHANGE", par une place appelée "BUS" . Cette place, marquée initialement d'un jeton,

le perd lorsque commence un échange, et le récupère dès que cet échange est terminé. Elle est donc place précédente de toutes les transitions t_{3i}, et place suivante de toutes les transitions t_{4i} (figure IV.22.).

Dès que commence un échange, l'arbitre est réactivé : il est à nouveau en mesure d'envoyer son accord à une autre demande (figure IV.20.a.).

Quand un échange est terminé, le module qui en avait fait la demande redevient oisif, et le bus est à nouveau libre (figure IV.20.b.).



a) b)

FIGURE IV.20.

Accord

Lorsqu'une demande a été faite de la part d'un quelconque des modules, l'arbitre envoie l'accord successivement à chaque module, de la façon suivante : c'est toujours le même, le "numéro 1" qui reçoit cet accord le premier. Si ce n'est pas lui qui a fait la demande d'échange, il le transmet au deuxième ; à son tour, celui-ci peut soit le prendre, soit le transmettre au troisième, et ainsi de suite jusqu'au dernier.

Les modules ont donc des priorités fixées (décroissantes selon leur numérotation). Lorsque deux ou plusieurs modules font simultanément une demande d'échange, c'est celui qui se trouve "le plus près" de l'arbitre qui est servi le premier.

L'acceptation et la transmission de l'accord sont représentées par la figure IV.21.a.

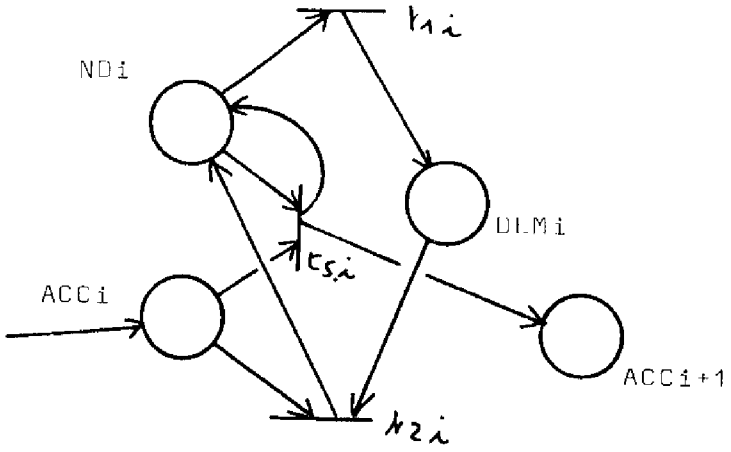


FIGURE IV.21.a.

Si le module (i) n'a fait aucune demande, la place ND_i est marquée d'un jeton ; dans ce cas, dès l'arrivée de l'accord, c'est-à-dire dès le marquage de la place ACC_i, le tir de la transition t_{5i} transmet l'accord au module (i+1). Mais si le module (i) a fait la demande d'échange, on passe, par le tir de la transition t_{2i}, à l'opération de sélection, après avoir pris l'accord.

REMARQUE

Il peut arriver qu'un signal parasite déclenche le départ du jeton de la place "ARBIT", alors qu'aucun module n'a fait une demande. Dans ce cas, l'accord ne va pas être accepté et attendra une demande du module n. Pour que le système ne soit pas bloqué, il faut que l'arbitre soit réactivé lorsque le dernier module ne prend pas l'accord (figure IV.21.b.).

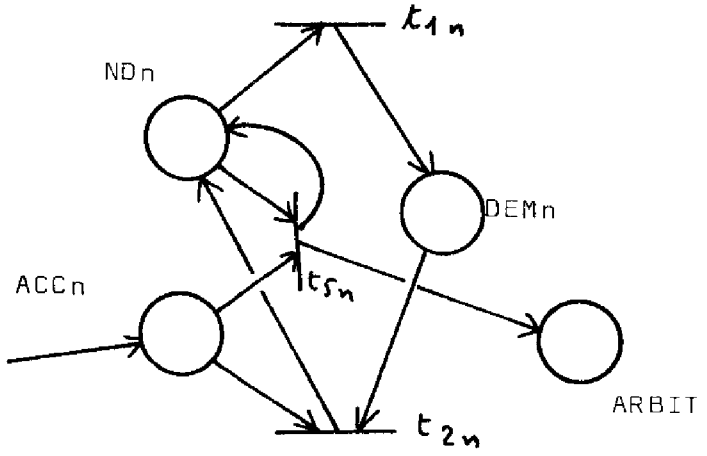


FIGURE IV.21.b.

IV.4.3. Description du réseau de PETRI

En rassemblant toutes les parties du réseau que nous venons d'expliquer, nous obtenons le graphe de commande complet du système.

Cependant, pour pouvoir utiliser sans problème les programmes de description et d'analyse en langage APL, nous devons supprimer la boucle élémentaire de la figure IV.21.a. Il suffit pour cela d'appliquer la méthode donnée en III.2.1. Après cette modification, nous obtenons le réseau de PETRI de la figure IV.22.

Pour effectuer l'étude de ce schéma, il nous faut évidemment avoir un nombre fini de modules : nous avons examiné les deux cas où ce nombre est égal à 2 et 3, mais nous ne présenterons que le plus simple : n=2, réseau de la figure IV.23.

Pour n modules, le graphe de commande possède (6n+1) transitions et (7n+2) places.

Les places marquées, pour le marquage initial, sont ARBIT, BUS, les places ND_i et les places OIS_i.

Les transitions t_{1i} sont déclarées transitions-utilisateurs.

IV.4.4. Analyse du graphe de commande

Nous avons effectué l'analyse du réseau de PETRI de la figure IV.23. Le graphe des marquages conséquents obtenu comporte 36 marquages et 70 arcs (tableau IV.24.a. et b.).

Le programme d'analyse indique que le réseau de PETRI est sauf, vivant et propre.

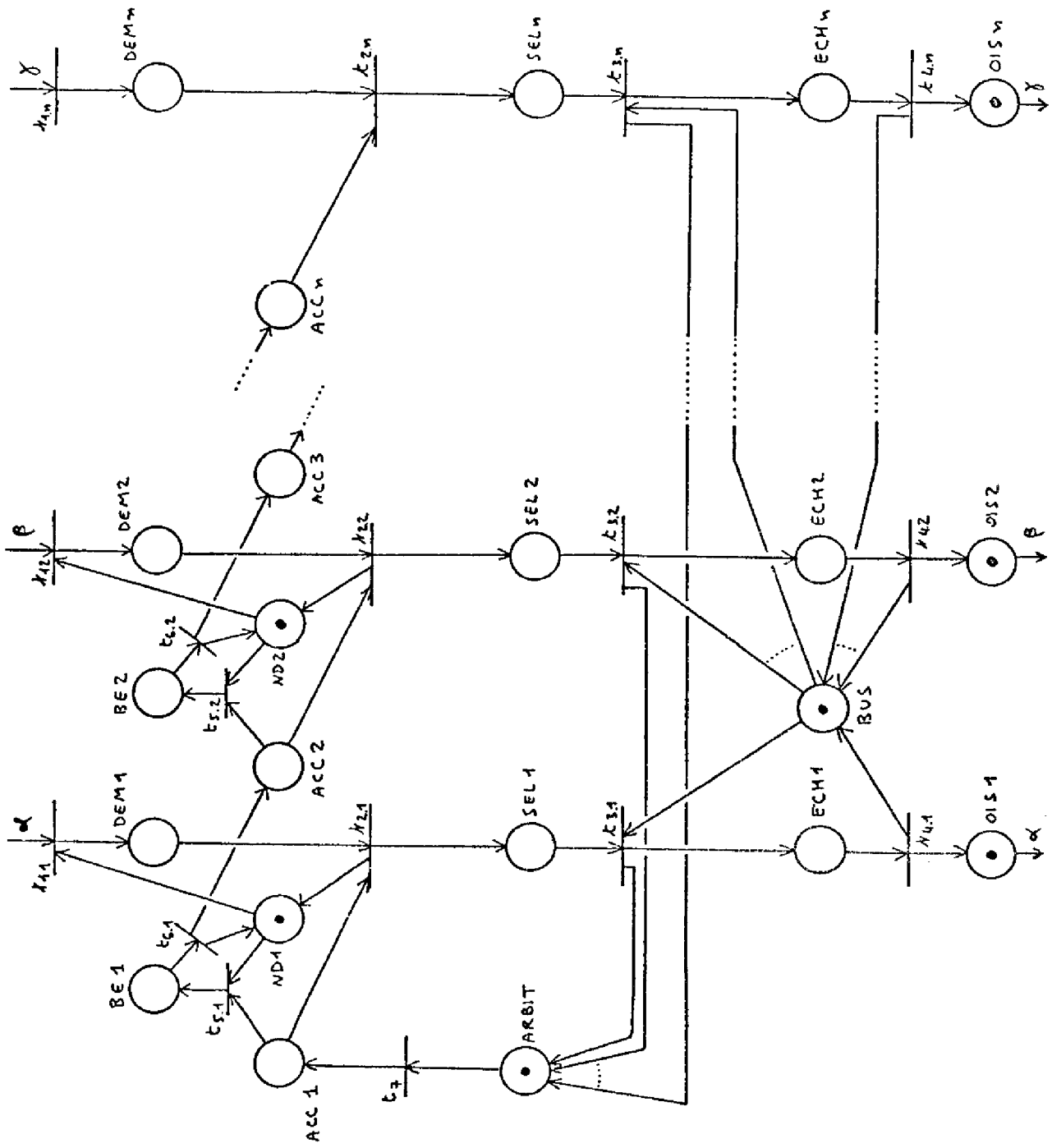

DEMn
SELn
ECHn
OISn
ACCn
DEM2
SEL2
ECH2
OIS2
ACC3
BE2
ND2
ACC2
DEM1
SEL1
ECH1
OIS1
BE1
ND1
ACC1
ARBIT
BUS

FIGURE IV.22.

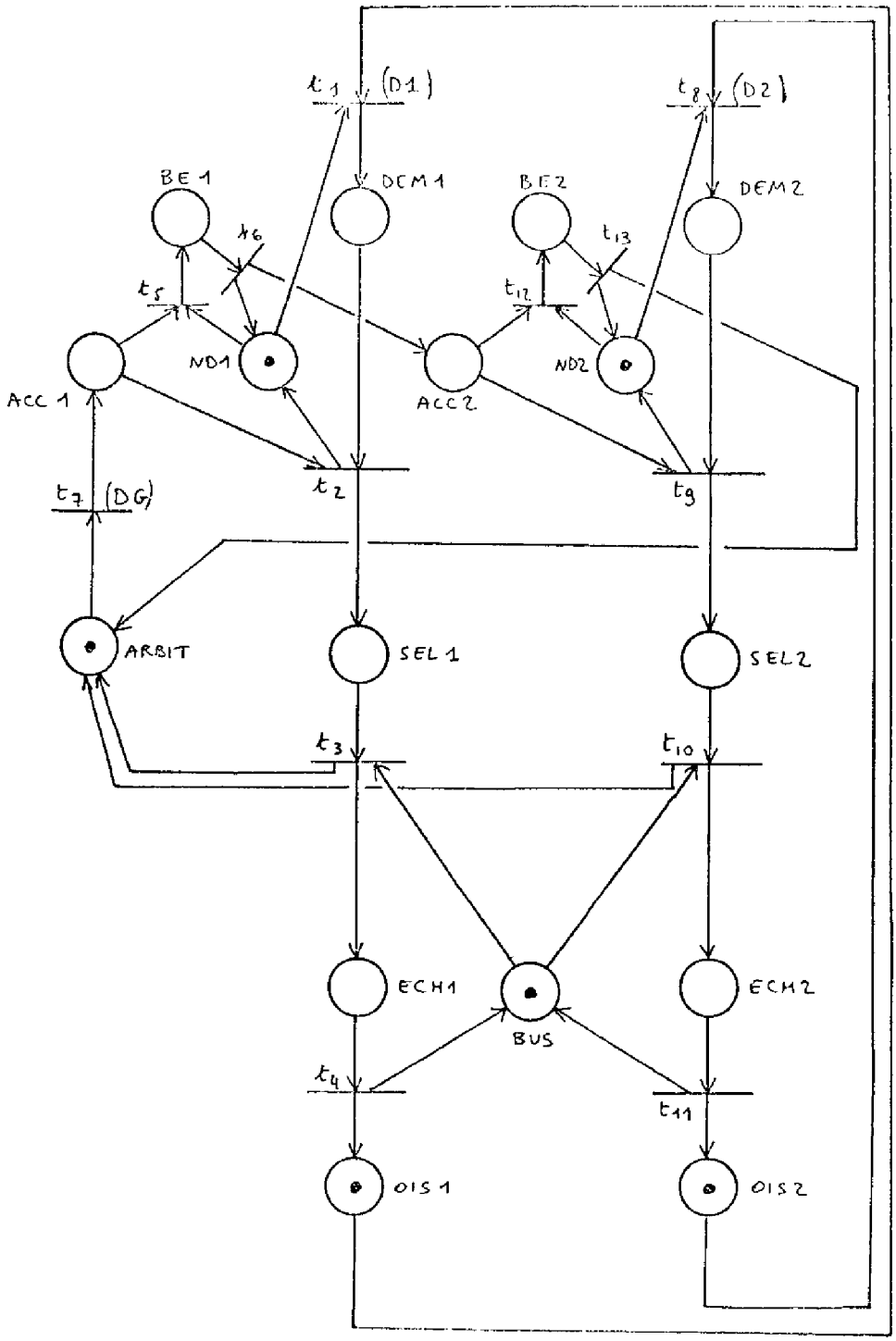


FIGURE IV.23. 2 modules.

GRAPHE

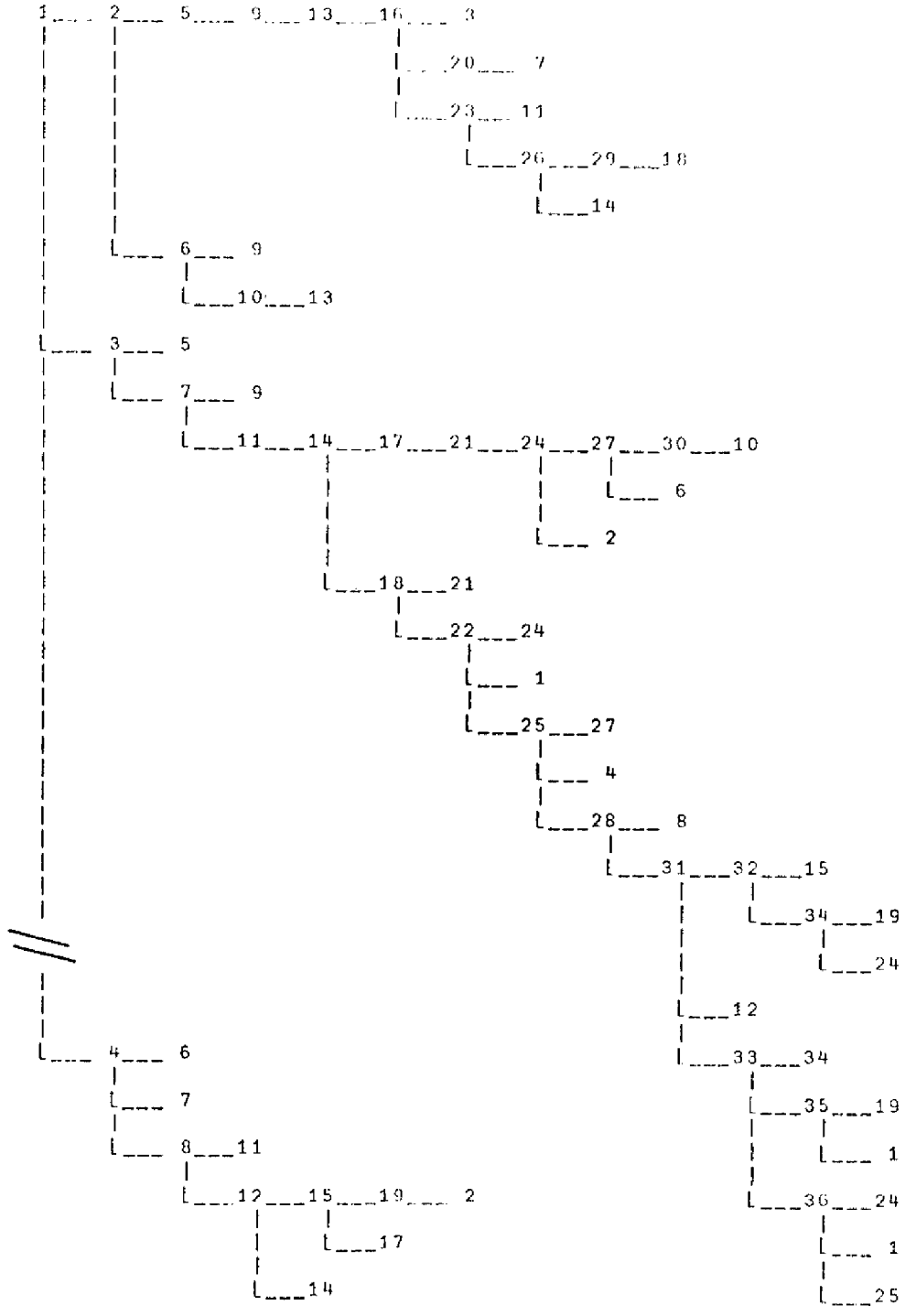


FIGURE IV.24.a.

1	: ND1	ND2	ARBIT	BUS	OIS1	OIS2
2	: DEM1	ND2	ARBIT	BUS	OIS2	
3	: DEM2	ND1	ARBIT	BUS	OIS1	
4	: ACC1	ND1	ND2	BUS	OIS1	OIS2
5	: DEM1	DEM2	ARBIT	BUS		
6	: DEM1	ACC1	ND2	BUS	OIS2	
7	: DEM2	ACC1	ND1	BUS	OIS1	
8	: BE1	ND2	BUS	OIS1	OIS2	
9	: DEM1	DEM2	ACC1	BUS		
10	: ND1	ND2	SEL1	BUS	OIS2	
11	: BE1	DEM2	BUS	OIS1		
12	: ND1	ACC2	ND2	BUS	OIS1	OIS2
13	: DEM2	ND1	SEL1	BUS		
14	: DEM2	ND1	ACC2	BUS	OIS1	
15	: DEM1	ACC2	ND2	BUS	OIS2	
16	: DEM2	ND1	ARBIT	ECH1		
17	: DEM1	DEM2	ACC2	BUS		
18	: ND1	ND2	SEL2	BUS	OIS1	
19	: DEM1	BE2	BUS	OIS2		
20	: DEM2	ACC1	ND1	ECH1		
21	: DEM1	ND2	SEL2	BUS		
22	: ND1	ND2	ARBIT	ECH2	OIS1	
23	: BE1	DEM2	ECH1			
24	: DEM1	ND2	ARBIT	ECH2		
25	: ACC1	ND1	ND2	ECH2	OIS1	
26	: DEM2	ND1	ACC2	ECH1		
27	: DEM1	ACC1	ND2	ECH2		
28	: BE1	ND2	ECH2	OIS1		
29	: ND1	ND2	SEL2	ECH1		
30	: ND1	ND2	SEL1	ECH2		
31	: ND1	ACC2	ND2	ECH2	OIS1	
32	: DEM1	ACC2	ND2	ECH2		
33	: BE2	ND1	ECH2	OIS1		
34	: DEM1	BE2	ECH2			
35	: BE2	ND1	BUS	OIS1	OIS2	
36	: ND1	ND2	ARBIT	ECH2	OIS1	

TABLEAU IV.24.b.

IV.4.5. Simulation

Nous n'avons pas étudié le graphe de données de ce système. Les opérations exécutées pendant le fonctionnement du système ne sont donc pas simulées.

La simulation nous permet tout de même d'observer le parallélisme entre les modules.

De plus, elle met en évidence un conflit. Supposons que le module 2 ait fait une demande ; l'arbitre envoie l'accord au module 1 qui le refuse et doit donc le transmettre au deuxième. Mais, au même instant, le premier module peut alors demander un échange (tableau IV.25.).

Ce conflit pourra être résolu en associant un prédicat à chaque transition t_{5i} : l'expression logique "$\sim D_i$" validera ces transitions.

```
     TIR 'BUS2'

TRANSITIONS POSSIBLES : 1,7,8
    ''       TIRABLES  :
CONDITIONS 1,7,8 NON SATISFAITES .
1 : D1
7 : D1vD2
8 : D2
MODIFS ,PUIS / .
D1←1
/

TRANSITIONS TIREES : 1,7
PLACES : DEM1,ACC1,ND2,BUS,OIS2

TRANSITIONS POSSIBLES : 2,8
    ''       TIRABLES  : 2
CONDITION 8 NON SATISFAITE .
8 : D2
MODIFS ,PUIS / .
/

TRANSITION TIREE : 2
PLACES : ND1,ND2,SEL1,BUS,OIS2

TRANSITIONS POSSIBLES : 3,8
    ''       TIRABLES  : 3
CONDITION 8 NON SATISFAITE .
8 : D2
MODIFS ,PUIS / .
D2←1
/

TRANSITIONS TIREES : 3,8
PLACES : DEM2,ACC1,ND1,BUS,OIS1

TRANSITIONS POSSIBLES : 1,5
    ''       TIRABLES  : 5
CONDITION 1 NON SATISFAITE .
1 : D1
MODIFS ,PUIS /.
D1←1
/

LES  2 TRANSITIONS 1,5
SONT EN CONFLIT .(PLACE ND1) .
LA TRANSITION 5 SERA CHOISIE ARBITRAIREMENT .

TRANSITION TIREE : 5
PLACES : BE1,DEM2,BUS,OIS1

TRANSITION TIREE : 6
PLACES : DEM2,ND1,ACC2,BUS,OIS1

TRANSITIONS POSSIBLES : 1,9
    ''       TIRABLES  : 9
CONDITION 1 NON SATISFAITE .
1 : D1
MODIFS ,PUIS /.
D1←1
/

TRANSITIONS TIREES : 9,1
PLACES : DEM1,ND2,SEL2,BUS

TRANSITION TIREE : 10
PLACES : DEM1,ND2,ARBIT,ECH2

TRANSITIONS TIREES : 7,11
PLACES : DEM1,ACC1,ND2,BUS,OIS2

      .
      .
      .
```

TABLEAU IV.25.

CONCLUSION

Dans ce mémoire, nous avons tout d'abord présenté un modèle de description des systèmes parallèles : les réseaux de PETRI. Ce modèle est bien adapté pour représenter la communication entre un système et le monde extérieur, c'est pourquoi nous pensons qu'il est tout indiqué pour décrire la partie commande du système considéré, la partie données étant représentée soit sous la forme d'un graphe (pour l'analyse), soit sous la forme d'instructions APL (pour la simulation).

Dans le deuxième chapitre, nous avons proposé des procédures d'analyse, aussi bien au niveau du réseau de PETRI seul qu'au niveau du schéma associant le graphe de données au réseau de PETRI. Cette analyse permet de détecter les erreurs de conception introduisant des incohérences au niveau de la structure du système considéré.

Cette analyse étant relativement lourde, nous avons donné, dans le chapitre III, les algorithmes permettant l'établissement d'un logiciel en langage APL. Grâce à ce logiciel, il est possible, en mode conversationnel, de décrire et d'analyser automatiquement un système.

Il nous a semblé bon de compléter cette analyse par une simulation du système, afin de vérifier que le comportement du système est bien le comportement désiré. Le logiciel de simulation a été rendu particulièrement simple grâce à l'utilisation de la fonction "exécute" qui existe dans le langage APL.

Quelques exemples sont enfin présentés dans le dernier chapitre afin de montrer la procédure à utiliser dans des cas pratiques.

Cet ensemble de programmes peut déjà apporter une aide importante au concepteur d'un système. Cependant, à partir de cette base, il faut penser à une extension intéressante, consistant à prendre en compte le temps : il s'agit de faire intervenir la durée de chaque opération, tant pour l'analyse de l'aspect déterminé du schéma, que pour la simulation du système.

La méthode consiste à construire un échéancier, chaque action exécutée se traduisant par l'écoulement d'un certain nombre d'unités de temps. Le problème est de tenir constamment à jour cet échéancier.

Cette prise en compte du temps permettra certainement une analyse plus exacte : le graphe des marquages conséquents sera moins étendu que lorsque les opérations ont des durées quelconques, et la recherche de l'aspect déterminé sera plus précise. Quant à la simulation, elle pourra reproduire avec plus de fidélité les comportements réels des systèmes décrits.

Il peut être également intéressant d'utiliser une console graphique comme outil d'entrée et de sortie du calculateur. Le résultat de la simulation pourrait, en effet, être appréhendé immédiatement d'une manière visuelle et globale.

BIBLIOGRAPHIE

[AGE] AGERWALA T.
Towards a theory for the analysis and synthesis of systems exhibiting concurrency.
Ph.D. The John Hopkins University, 1975.

[APL] ABRAMS P.S., FOSTER G.H., HAEGI H.R., TRUMPY S.
APL 75
Congrès de Pise, Italie, Juin 1975.

[AZE] AZEMA P., DIAZ M., VALETTE R.
Vérification fonctionnelle et matérielle basée sur les réseaux de PETRI.
Journée "Prévention des pannes", I.R.I.A., Décembre 1975,
Publication LAAS n°1358.

[BER] BERTHELOT G.
Une méthode de vérification des réseaux de PETRI.
Journée A.F.C.E.T., Paris, Mars 1977.

[BLA] BLANCHARD M., CAVARROC J.C., GILLON J., MARCHAND J., GUIDEZ G., THUILLIER G.
Automatismes à séquences.
Rapport du contrat D.G.R.S.T., n°71.72.912.01, 1976.

[CER] CERF V.G.
Multiprocessors, semaphores and a graph model of computation.
Ph.D. Dissertation, University of California, Los Angeles, U.S.A., 1972.

[COU] COURVOISIER M.
Etude des systèmes logiques de commande asynchrones à évolutions simultanées.
Thèse de Doctorat ès Sciences, Toulouse, 1974.

[DAC] DACLIN E., BLANCHARD M.
Synthèse des systèmes logiques.
Collection "Sup.Aéro", Cepadues Edition, 1976.

[HAC] HACK M.H.T.
Analysis of production schemata by PETRI nets.
Master of Science, Massachussets Institute of Technology, 1972.

[IBM] APL-S.V.
Manuel de référence IBM, France, 1974.

[IVE] IVERSON K.E., FALKOFF A.D.
APL : manuel de référence.
Traduction française de PINTA C. et LE BORGNE Y.
IBM, France, 1972.

[JAC] JACK L.A., HEIMERDINGER W.L., JOHNSON M.D.
Theory of fault tolerance.
1974-5. Annual report. Honeywell.Systems and Research Center.

[KAR] KARP R.M., MILLER R.E.
Parallel program schemata.
Journal of computer and system sciences, vol.3, Mai 1969.

[KEL] KELLER R.M.
Parallel program schemata and maximal parallelism.
Journal of the Association for Computing Machinery, July 1973.

[KOS] KOSARAJU S.R.
Limitations of Dijkstra's semaphore primitives and PETRI nets.
Hopkins Computer, Research Report 25, The John Hopkins University, 1973.

[LED] LE DANOIS P.L., AYACHE J.M.
Synthesis of logical systems with PLA's.
Journées d'Electronique 1977, Lausanne, Suisse, Mars 1977.

[MIS] MISUNAS D.
PETRI nets and speed independent design.
Massachussets Institute of Technology, 1973.

[MOA] MOALLA M., SIFAKIS J., ZACHARIADES M.
Un langage d'aide à la conception et à la simulation de systèmes complexes.
E.N.S.I.M.A.G., Grenoble, RR n°16, Octobre 1975.

[PAT] PATIL S.S., DENNIS J.B.
The description and realization of digital systems.
RAIRO, J1, Février 1973.

[PET] PETRI C.A.
Kommunikation mit automaten.
Université de Bonn, Allemagne, 1962.

[PRU] PRUNET F., DUMAS J.M.
Introduction à la modélisation naturelle des systèmes de commande : l'organiphase.
RAIRO, J2, Juillet 1974.

[RAM] RAMCHANDANI C.
Analysis of asynchronous concurrent systems by PETRI nets.
Ph.D., Massachussets Institute of Technology, 1973.

[RAU] RAULT J.C., DEMARS G., RUGGIU G.
Le langage et les systèmes APL.
Masson et Cie, 1974.

[REN] RENALIER J., AZEMA P., VALETTE R.
Programme de simulation et d'analyse des schémas à réseaux de PETRI, en langage APL.
Journées A.F.C.E.T., Paris, Mars 1977.

[ROS] ROSE C.W., ALBARRAN M.
Modeling and design description of hierarchical hardware/software systems.
12th. Design Automation Conference Proceedings, June 1975, Boston.

[SIF] SIFAKIS J.
Comportement permanent des réseaux de PETRI temporisés.
Journées A.F.C.E.T., Paris, Mars 1977.

[STA] STANDARD MICROSYSTEMS
COM 2502, Universal asynchronous receiver/transmitter.
Preliminary specifications, Août 1972.

[VAL.1] VALETTE R.
Sur la description, l'analyse et la validation des systèmes de commande parallèles.
Thèse de Doctorat ès Sciences, Toulouse, 1976.

[VAL.2] VALETTE R., RENALIER J.
Introduction aux réseaux de PETRI.
Note Interne LAAS-RASSC, Mars 1977.

[VAL.3] VALETTE R., COURVOISIER M.
Modèle de représentation pour l'analyse et la vérification des automatismes logiques.
Colloque A.F.C.E.T./A.D.E.P.A., Paris, Décembre 1976.

TABLE DES MATIERES
