Université de NANCY I UER de Mathématiques

PETRI-S : UN OUTIL D'AIDE AU PROTOTYPAGE RAPIDE DES AUTOMATISMES INDUSTRIELS DECRITS PAR DES RESEAUX DE PETRI INTERPRETES

==========

THESE

Présentée à l'Université de Nancy I
pour l'obtention du grade de

DOCTEUR DE L'UNIVERSITÉ DE NANCY I
(Mention : Informatique)

Par

Dalila BOUDEBOUS

==========

Thèse soutenue publiquement le 5 Janvier 1990 devant
la commission d'examen composée de

Président	M.	**J.P. FINANCE** Université de NANCY I
Rapporteurs	MM.	**G.R. PERRIN** Université de BESANCON
		R. VALETTE CNRS LAAS TOULOUSE
Examinateurs	MM.	**J.F. AUBRY** ENSEM de NANCY
		J.C. DERNIAME Université de NANCY I
		J. TANKOANO IAI GABON

A mes parents,
en témoignage de mon amour.
A tous les miens.

Je remercie Monsieur J.P. Finance, Directeur du CRIN, qui me fait l'honneur de présider la commission d'examen.

Je remercie Monsieur J.C. Derniame, pour m'avoir accueillie dans son équipe et accepté de diriger cette recherche. Qu'il soit assuré de toute ma gratitude pour la patience dont il a fait preuve à mon égard et pour ses multiples encouragements.

Je remercie très vivement :

Monsieur G.R. Perrin spécialiste des systèmes parallèles, qui a accepté d'être rapporteur de cette thèse.

Monsieur R. Valette, éminent spécialiste de l'utilisation des réseaux de Petri dans la conception des systèmes de commande qui a accepté d'être rapporteur de ce travail. Sa lecture critique a contribué à son amélioration.

Monsieur J.F. Aubry qui a accepté de faire partie de ce jury.

Monsieur J. Tankoano pour sa collaboration et pour toutes les suggestions et les remarques constructives qu'il m'a formulées et qui m'ont permis d'améliorer mon travail.

Je remercie Messieurs F. Moitessier, N. Boudjlida, M. Djoudi pour leur collaboration dans la relecture de ce document.

Enfin, que tous les membres de l'équipe GENIE LOGICIEL et du laboratoire que j'ai côtoyés, trouvent ici le témoignage de mon amitié.

2.2.1.2 Utilisation pour l'évaluation des performances 51
2.2.1.3 Commentaires 52
2.2.2 Les RdP stochastiques 53
2.2.2.1 Commentaires 54
2.3 Conclusion 54
3 La simulation discrète 54
3.1 Les progiciels généraux de simulation 56
3.2 Les simulateurs de RdP 57
3.2.1 PSI : a Petri net based SImulator 57
3.2.1.1 Commentaires 59
3.2.2 Un simulateur de RdP colorés 60
4 Conclusion 63

CONCLUSION 64

PARTIE II
INTRODUCTION 65

CHAPITRE 4 : DESCRIPTION DES MODELES DE PETRI-S
1 Introduction 66
2 Modèle de description du comportement du système de commande. 67
2.1 Présentation du modèle RdPI 68
2.1.1 Définition 68
2.1.2 Représentation graphique 69
2.1.3 Eléments du modèle 69
2.1.3.1 La notion de réceptivité 69
2.1.3.2 Les opérations 70
2.1.3.3 Prise en compte du temps 70
2.1.4 Les règles d'évolution d'un RdPI 71
2.2 Les extensions apportées au modèle RdPI 74
2.2.1 La prise en compte des occurrences d'événements externes 74
2.2.2 La communication inter-modulaire 75
2.2.2.1 Les communications synchrones 75
2.2.2.2 Les communications asynchrones 76
2.2.2.3 L'utilisation combinée des communications synchrones et asynchrones 77
2.3 Exemples 78

3 Modèle de description de l'environnement 82
3.1 Principe de modélisation 82
3.2 Les types génériques de générateurs d'événements 84
3.2.1 Les générateurs d'événements exogènes binaires fugitifs .. 84
3.2.2 Les générateurs d'événements exogènes binaires non fugitifs 84
3.2.3 Les générateurs d'événements exogènes continus 84
3.2.4 Les générateurs d'événements endogènes binaires fugitifs 85
3.2.5 Les générateurs d'événements endogènes binaires non fugitifs 85
3.2.6 Les générateurs d'événements endogènes continus 86
3.2.7 Les générateurs de purges 86
3.3 Exemple 87
3.4 Comparaison avec l'approche préconisée dans SICLOP 88
4 Modèle de représentation du flux d'activité 89
4.1 Les éléments de base du modèle retenu 90
4.1.1 Les marques 90
4.1.2 Les places 90
4.1.3 Les transitions 91
4.1.4 Les arcs 91
4.2 Exemple 92
5 Conclusion 97

CHAPITRE 5 : DESCRIPTION DU SYSTEME PETRI-S

1 Introduction 98
2 Objectifs et principes de base 99
2.1 Expression des besoins 99
2.2 La validation 103
2.2.1 L'approche de validation retenue 103
2.2.2 Les propriétés 104
2.2.2.1 Les propriétés générales 104
2.2.2.2 Les propriétés spécifiques 111
2.3 L'évaluation des performances 114
3 Environnement général de PETRI-S 116
3.1 Les outils d'édition évolués 116
3.2 L'outil de décomposition 117
3.3 L'outil de simulation 118
4 Le langage de description des spécifications 118

4.1 Eléments lexicaux 119
4.1.1 Jeu de caractères 119
4.1.2 Les délimiteurs 119
4.1.3 Les mots-clés 119
4.1.4 Les identificateurs 120
4.1.5 Les commentaires 120
4.2 Structure d'une spécification 120
4.3 Description des différentes parties de la spécification 122
4.3.1 Partie description du réseau 122
4.3.2 Partie déclaration de l'environnement 124
4.3.2.1 Déclaration des variables 124
4.3.2.2 Déclaration des conditions et des événements 127
4.3.2.3 Déclaration des actions 128
4.3.3 Partie interprétation 130
4.3.4 Partie description du flux d'activité 131
4.3.4.1 Déclaration des files d'attente 132
4.3.4.2 Déclaration des transactions 133
4.3.4.3 Description des écoulement de flux 134
4.4 L'analyseur du langage 136
4.5 Extensions 138
5 L'éditeur graphique : PETRI-EDIT 141
5.1 Les mécanismes utilisés pour la gestion des interactions homme/machine 141
5.1.1 Les fenêtres 141
5.1.2 Les menus 142
5.1.3 Le mécanisme de désignation 145
5.1.4 Le mini-éditeur de textes 145
5.2 Les fonctionnalités 145
5.2.1 Menu général 145
5.2.2 Menu de création modification 146
5.2.3 Menu environnement 148
5.2.4 Menu interprétation 151
5.3 Remarques précisions et extensions 154
5.4 Conclusion 154
6 Interface utilisateur de PETRI-S 155
7 Le simulateur PETRI-S 155
7.1 La phase de préparation 155

7.1.1 Description du comportement de l'environnement 156
7.1.1.1 L'éditeur d'acquisition de la description 156
a) La construction 156
b) La modification 158
c) La visualisation 158
d) La fonction de sauvegarde 158
e) La fonction de chargement 159
7.1.2 Spécification de l'état initial 159
7.1.3 Spécification du début de la simulation 160
7.1.4 La durée de la simulation 160
7.1.5 Vérification des propriétés 160
7.1.6 Les modes d'exploitation du simulateur 161
7.1.6.1 Le mode pas à pas 161
7.1.6.2 Le mode semi-automatique 161
7.1.6.3 Le mode automatique 162
7.2 La phase de simulation 162
7.2.1 La gestion du temps 162
7.2.2 L'approche de simulation 163
7.2.3 L'algorithme de simulation 164
7.2.3.1 Procédure d'initialisation 166
7.2.3.2 Procédure effectuant un tir sur occurrences d'événements 166
7.3 La phase de génération des résultats 174
8 Conclusion 174

CONCLUSION 175

BIBLIOGRAPHIE 177

ANNEXE A : MOTS CLES DU LANGAGE 186

ANNEXE B : LISTE DES ERREURS DE COMPILATION 187

ANNEXE C : SCENARIOS D'UTILISATION DE L'EDITEUR GRAPHIQUE 190

ANNEXE D : SCENARIOS D'UTILISATION DE PETRI-S 206

INTRODUCTION

1 CONTEXTE DU TRAVAIL

Ce travail s'insère dans le cadre d'un projet, qui concerne l'élaboration **d'un système d'aide à la conception** des systèmes de commande répartis d'ateliers de production [Tankoano 88].
Un système de commande se distingue des autres systèmes informatiques par le fait que son fonctionnement est assujetti à l'évolution d'un environnement (ou procédé industriel) qui lui est connecté et dont il doit commander et contrôler le comportement. Pour suivre et piloter l'environnement, le système de commande reçoit par le biais de capteurs des comptes rendus sur l'état de l'environnement (événements et mesures), et à partir de ces entrées, élabore des ordres de commandes à l'aide d'actionneurs sur l'environnement selon les lois de commande de ce dernier. L'élaboration des commandes doit respecter des contraintes temporelles imposées par l'environnement, c'est-à-dire que le système de commande doit réagir dans des laps de temps qui garantissent la bonne évolution de l'environnement.
On parle de système réparti pour signifier que le système de commande est partitionné en modules qui peuvent être répartis sur des sites distribués géographiquement et communiquant via un réseau.

La conception de ces systèmes de commande pose de sérieuses difficultés du fait du parallélisme inhérent au problème à résoudre, des contraintes temporelles à respecter et de la sûreté de fonctionnement à garantir. Elle nécessite de ce fait l'emploi de méthodes et d'outils d'aide très avancés.

Le projet de recherche dans lequel s'insère notre travail vise la mise en place d'un système d'aide à la conception bâti autour des réseaux de Petri qui couvre les quatre tâches intimement liées du processus de conception de tels systèmes. Ces tâches sont :

• la **spécification** du service que doit rendre le système de commande. Ce service décrit la relation d'ordre qui doit être maintenue entre les entrées et les sorties du système,

• l'**organisation** du système en un réseau de modules communiquant par échange de messages. Cette phase concerne la conception de la structure interne

du système,

• la **validation** qui consiste à montrer que la spécification du service que doit assurer le système de commande possède les propriétés de sûreté et de vivacité et qu'elle est conforme aux besoins exprimés par l'utilisateur,

• et l'**évaluation** des performances en vue du dimensionnement de l'atelier.

Pour cela, le système d'aide s'appuie sur :

-Une **démarche de conception** qui permet d'ordonner les différentes étapes du processus de conception.

-Un **modèle de spécification unique (les réseaux de Petri interprétés)** qui permet d'aborder à la fois les problèmes de spécification, d'organisation, de validation et d'évaluation de systèmes.

-Et **trois techniques d'aide à la conception :**

- la technique de conception par raffinements qui permet la construction progressive de la spécification du comportement d'un système avec de bonnes propriétés

- la technique de conception par décomposition, qui permet l'organisation du système en un réseau de modules communicants

- la technique de prototypage rapide qui permet de simuler le système de commande.

2 OBJECTIFS DU TRAVAIL

Dans cette thèse, nous nous intéressons tout particulièrement à la technique de prototypage rapide.
Notre contribution porte essentiellement sur la conception et la réalisation d'un outil (**PETRI-S**) qui s'appuie sur une description du comportement du système de commande et sur une modélisation adéquate de l'environnement commandé et

du flux d'activité pour réaliser une **simulation hors site** du système de commande. Cette mise en œuvre simulée de la spécification du système de commande vise les objectifs suivants:

• **La clarification des besoins**, en aidant l'utilisateur à préciser ou à compléter ses besoins et en facilitant la communication entre les différentes personnes concernées par la réalisation et l'utilisation d'un système.

• **La validation et la mise au point fine** de la spécification en vérifiant que le comportement du système de commande est conforme aux besoins exprimés par l'utilisateur et qu'il possède les bonnes propriétés liées à la vivacité et à la sûreté de fonctionnement.

• **L'évaluation des performances**, soit pour le choix de stratégies de pilotage soit pour le dimensionnement de l'atelier.

L'apport fondamental de cet outil réside dans sa très **grande facilité de mise en œuvre** (qualité essentielle d'un outil d'aide au prototypage rapide) et surtout dans le fait qu'il permet d'atteindre l'ensemble de ces objectifs à partir d'une **même modélisation** du système que le concepteur enrichit. Ceci permet dans un processus de conception de renforcer la **cohérence** et la **continuité** entre les tâches de spécification, validation et évaluation et d'éviter ainsi au concepteur l'utilisation de plusieurs outils avec des modélisations et des concepts différents. En effet, la multiplicité des outils constitue une contrainte et une surcharge de travail qui peut devenir prohibitive et compromettre leur utilisation. Elle pose au concepteur les problèmes de vérification de leur cohérence et du maintien de cette cohérence lors de modifications.

3 PLAN DE LA THESE

Cette thèse comporte deux parties :

La première partie réalise un tour d'horizon des principales approches utilisées pour la validation des systèmes de commandes, et pour l'évaluation des performances. Elle vise à mettre en évidence les limites de ces approches et à justifier les choix retenus dans le développement de notre outil. Elle est

composée de trois chapitres :

Le premier chapitre est consacré à une présentation générale des réseaux de Petri qui constituent le modèle de base sur lequel nous nous appuyons. Notre choix a été motivé par le fait que ce modèle permet de décrire finement les contraintes de synchronisation et de temps, et il est bien adapté à la validation et à la simulation.

Nous abordons dans le chapitre 2, la présentation des différentes approches de validation de systèmes de commande décrits par des réseaux de Petri. Nous décrivons d'abord les méthodes de validation formelles basées sur l'analyse des réseaux de Petri. Ces méthodes fournissent des résultats précis mais ne permettent pas de tenir compte de l'influence du comportement de l'environnement commandé. Nous présentons ensuite des méthodes plus empiriques basées sur la simulation conjointe du système de commande et du système commandé. Ces méthodes permettent de détecter des incohérences et des erreurs mais ont l'inconvénient bien connu d'être non exhaustives. En effet, comme on ne peut dans la plupart des cas essayer toutes les possibilités, on ne peut garantir que toutes les erreurs ont été détectées. Nous terminons ce chapitre par la proposition d'une approche mixte qui combine les apports d'une approche formelle et d'une approche par simulation.

Dans le chapitre 3, nous présentons les différentes approches préconisées pour l'évaluation des performances des ateliers de production. Ces approches se répartissent en deux grandes familles : les méthodes analytiques et la simulation à événements discrets. Les méthodes analytiques permettent de fournir rapidement des résultats pour une évaluation grossière de l'atelier. Plus couramment utilisée, la simulation discrète permet de décrire et d'évaluer l'atelier avec un grand degré de précision.

La seconde partie est consacrée à la description du système PETRI-S, elle comporte deux chapitres :

Le chapitre 4, présente l'ensemble des modèles retenus pour décrire le comportement du système de commande, le comportement de l'environnement commandé et le flux d'activité. Ces modèles permettent de décrire de façon non

redondante la dynamique des systèmes de production et d'éviter des doubles définitions avec tous les risques d'incohérence que cela comporte.

Dans le chapitre 5, nous présentons le logiciel d'exécution du simulateur avec dans un premier temps les outils d'édition de la spécification (l'analyseur syntaxique et l'éditeur graphique) et dans un deuxième temps le simulateur (approche de simulation, gestion du temps de simulation, les résultats issus de la simulation...).

Enfin des propositions visant à améliorer certains aspects de l'outil seront présentées dans la conclusion.

PREMIERE

PARTIE

INTRODUCTION

L'automatisation des ateliers de production tels les ateliers flexibles conduit de plus en plus à des systèmes très complexes, qui permettent de fabriquer quasi simultanément les diverses composantes d'un produit donné, puis de passer assez facilement de la fabrication d'un type de produit à celle d'un autre. Ceci permet de mieux adapter la production à la demande, d'accroître le taux d'occupation des machines, de réduire le volume des stocks et d'obtenir ainsi une rentabilité maximale.

En contrepartie, la conception du système de commande d'un atelier de production nécessite, par sa complexité et par les intérêts économiques mis en jeu des méthodes et des outils spécifiques qui permettent la validation du système de commande et l'évaluation des performances en vue du dimensionnement de l'atelier.

Les démarches de conception traditionnellement préconisées conduisent le concepteur à aborder les tâches de validation et d'évaluation en s'appuyant sur plusieurs modélisations de son système et en manipulant des outils et des formalismes différents. Ceci entraîne des redondances dans les spécifications avec des risques d'incohérences et rend mal aisée l'étude de différents scénarios. C'est pourquoi, l'objectif que nous nous étions fixé était de développer **un outil unique** d'aide au prototypage rapide de systèmes de commande qui peut être utilisé à diverses fins : pour la validation, pour l'évaluation des performances et pour l'aide au dialogue avec l'utilisateur.

Pour mieux montrer l'apport de notre travail, nous allons dans cette première partie faire une revue critique des approches actuellement employées pour la validation et l'évaluation des performances des systèmes automatisés de production. Notre but n'est pas de les décrire précisément mais plutôt de mettre en évidence leurs limites. Mais auparavant, nous allons présenter les Réseaux de Petri (noté RdP) qui constituent un modèle bien connu pour décrire le comportement de systèmes de commande.

CHAPITRE 1

LES RESEAUX DE PETRI

GENERALITES

1 INTRODUCTION

Les réseaux de Petri ont été développés à partir des travaux de recherche de CARL ADAM PETRI [Petri 62] en vue de représenter graphiquement le cheminement de l'information dans les systèmes asynchrones. Mais ce n'est que beaucoup plus tard, grâce notamment aux travaux de A.W. HOLT [Holt 70] et ceux développés au MIT [Hack 72] [Ramchandani 73] [Peterson 77] dans le cadre des problèmes posés par le parallélisme dans les systèmes informatiques, que ce modèle commença à être utilisé de façon plus large. Les nombreux résultats théoriques accumulés, en font aujourd'hui un excellent outil de spécification et de validation dans de nombreux domaines d'application, et plus particulièrement dans le domaine des automatismes industriels [Valette 76] [Sifakis 79] [André 81] [Moalla 81] et des protocoles de communication [Berthelot 83] [Ayache 85].

2 DEFINITIONS

2.1 DEFINITION DES RDP [BRAMS 83]

Un RdP généralisé (dit aussi autonome) est défini par un quadruplet **R = (P, T, Pré, Post)** où

P est un ensemble fini non vide d'objets appelés **places**,

T est un ensemble fini non vide d'objets appelés **transitions** tel que P et T sont disjoints ($P \cap T = \varnothing$),

Pré une application de $P \times T \rightarrow \mathbb{N}$ appelée **fonction d'incidence avant**,

et **Post** une application de $P \times T \rightarrow \mathbb{N}$ appelée **fonction d'incidence arrière**.

2.2 REPRESENTATION GRAPHIQUE

On représente un RdP par un graphe biparti orienté dont les arcs sont valués. Dans ce graphe les nœuds correspondent à des places et des transitions représentées respectivement par des cercles et des barres.
Un arc relie une place p à une transition t si et seulement si la valeur pré(p,t) est supérieure à 0. Pré(p,t) représente alors la valuation ou "poids" de l' arc.
De la même façon un arc relie une transition t à une place p si et seulement si

post(p,t) est supérieure à 0. Post(p,t) représente ici également la valuation de l'arc. De façon générale, dans un réseau, la valuation d'un arc n'est indiquée de façon explicite que lorsqu'elle est supérieure à 1.

Les RdP qui ont été étudiés en tout premier lieu sont ceux pour lesquels tous les arcs sont valués par 1. Ces réseaux, sont qualifiés de RdP ordinaires. Les réseaux qui possèdent des arcs ayant une valuation supérieure à 1 sont appelés réseaux généralisés, ces réseaux peuvent se ramener à des réseaux ordinaires [Hack 72].

Notations

Etant donné un RdP R = (P, T, pré, post), t une transition, et p une place, on utilise les notations suivantes :

- ${}^{\bullet}t = \{p \in P / pré(p,t) > 0\}$ désigne l'ensemble des places en entrée de la transition t;

- $t^{\bullet} = \{p \in P / post(p,t) > 0\}$ désigne l'ensemble des places en sortie de la transition t;

- ${}^{\bullet}p = \{t \in T / post(p,t) > 0\}$ désigne l'ensemble des transitions en entrée de la place p;

- $p^{\bullet} = \{t \in T / pré(p,t) > 0\}$ désigne l'ensemble des transitions en sortie de la place p.

2.3 MARQUAGE D'UN RDP

Le marquage M d'un réseau R = (P, T, pré, post) est une application de l'ensemble des places P dans l'ensemble des entiers naturels $\mathbb{N}$.

Il peut être défini comme suit, par la donnée d'un vecteur de $\mathbb{N}^{|P|}$

$$M = \begin{bmatrix} m_1 \\ m_2 \\ . \\ . \\ m_n \end{bmatrix}$$

$m_i = M(p_i)$ désigne alors le marquage de la place p_i, c'est-à-dire le nombre de marques contenues dans la place p_i.

Sur la représentation graphique d'un réseau, on représente généralement le marquage M soit en inscrivant à l'intérieur de chaque place p du réseau la valeur M(p) si celle ci est non nulle, soit en représentant les M(p) marques par des points.

Le marquage M d'un réseau peut servir à représenter l'état d'un système. Il évolue en respectant des règles bien précises.

2.4 REGLES D'EVOLUTION D'UN RDP

L'évolution d'un réseau s'effectue par une succession d'opérations de franchissement des transitions. Le franchissement d'une transition ne peut avoir lieu que si sa précondition de franchissement est satisfaite. On dit qu'elle est validée ou encore qu'elle est sensibilisée, ou franchissable.

2.4.1 TRANSITION VALIDEE

Une transition t est validée pour un marquage M si et seulement si chacune de ses places d'entrée contient un nombre de marques au moins égal au poids de l'arc qui la connecte à t.

$$\forall p \in {}^{\cdot}t \quad M(p) \geq pré\ (p, t)\ \textit{préconditions de franchissement de } t$$

2.4.2 FRANCHISSEMENT D'UNE TRANSITION

Une transition t validée peut être franchie. L'opération de franchissement ou de tir consiste alors à faire passer le réseau d'un marquage M (marquage validant la transition t) à un nouveau marquage M' et tel que :

$$\forall p \in P \quad M'(p) = M(p) - pré(p,t) + post(p,t).$$

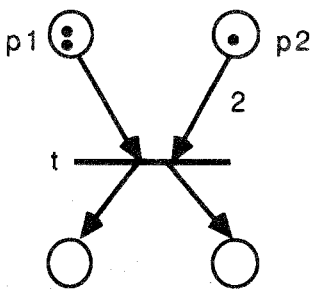
On note $M \xrightarrow{t} M'$

Le marquage après le tir d'une transition s'obtient donc en retirant des marques

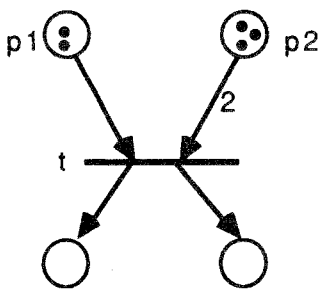
des places en entrée de la transition et en ajoutant des marques aux places de sortie de cette transition. Le nombre de marques déplacées correspondant aux poids des arcs reliant la transition aux places d'entrée et de sortie.

Le franchissement d'une transition est instantané et indivisible.

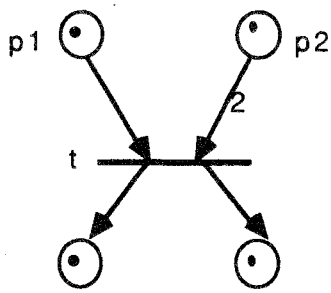
Exemple



transition non validée
Figure 1.1



transition validée
Figure 1.2



transition franchie
Figure 1.3

La transition t de la figure 1.1 ne peut être franchie car une de ses préconditions n'est pas satisfaite : la place p2 ne contient pas suffisamment de marques. Par contre, la transition t de la figure 1.2 est franchissable, la figure 1.3 indique le marquage que l'on obtient après le franchissement de cette transition.

2.5 MARQUAGES ACCESSIBLES

2.5.1 ENSEMBLE DES MARQUAGES ACCESSIBLES

Soient T^* le monoïde libre construit sur T et $s \in T^*$ une séquence finie de transitions $t_{i1}t_{i2}.....t_{il}$. On dira que s est une **séquence de franchissement** à partir du marquage M_i si et seulement si il existe des marquages intermédiaires M_{i1} $M_{i2}.....$ M_{il} tels que :

$$M_i \xrightarrow{ti1} M_{i1} \xrightarrow{ti2} M_{i2}.....\xrightarrow{til} M_{il}$$

On écrit plus simplement $M_i \xrightarrow{s} M_{il}$,

et on dit que M_{il} est un **marquage accessible** à partir du marquage M_i.
On note alors par **A(R, M_0) l'ensemble des marquages accessibles** à partir du marquage initial M_0 du RdP R

$$A(R, M_0) = \{ M_i \in \mathbb{N}^{|P|} / \exists s \in T^* : M_0 \xrightarrow{s} M_i \}$$

2.5.2 GRAPHE DES MARQUAGES ACCESSIBLES

Soit un RdP R de marquage initial M_0, on appelle **graphe des marquages accessibles GA(R, M_0)**, le graphe orienté (A(R, M_0), X) où A(R, M_0) est l'ensemble des sommets et X l'ensemble des arcs (M_i, M_j) tels que :

- M_i et $M_j \in$ A(R, M_0),
- il existe une transition $t_i \in$ T telle que $M_i \xrightarrow{ti} M_j$.

Dans un graphe des marquages, chaque arc est étiqueté par la transition qui permet de passer du marquage de départ au marquage d'arrivée.

3 EXEMPLE

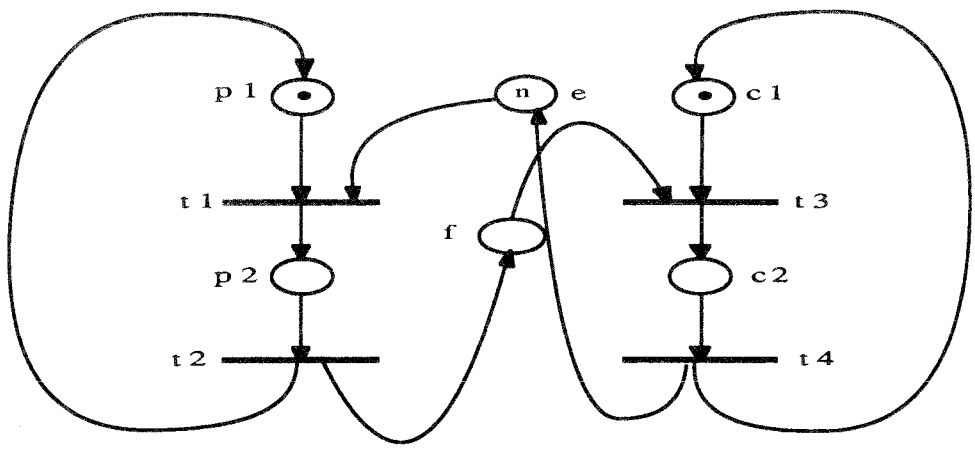
La figure 1.4 présente un exemple de RdP modélisant un système constitué d'un producteur et d'un consommateur reliés par un magasin dans lequel sont déposés et retirés les objets produits. Le magasin a une capacité finie N.
Dans ce réseau les places p1 et p2 décrivent l'état du producteur : la présence de marque dans la place p1 signifie que le producteur est en attente de production et le marquage de p2 qu'il produit.
De la même manière, les places c1 et c2 modélisent l'état du consommateur : la présence de marque dans la place c1 signifie que le consommateur est en attente de consommation et la présence de marque dans c2 signifie qu'il est en train de consommer.
Les places e et f représentent l'état du magasin, la place e comptabilise le nombre de positions vides dans le magasin, alors que f comptabilise le nombre de positions occupées.
Les transitions t1 et t2 représentent respectivement le début et la fin de la production, les transitions t3 et t4 le début et la fin de la consommation.

Le marquage initial indique qu'au départ le système est dans un état tel que :

-le producteur est prêt à produire (place p1 marquée)

-le consommateur est en attente de consommation (place c1 marquée)

-et le magasin est vide (la place e contient N marques)



producteur magasin consommateur

Figure 1.4

Comme on peut le constater, les RdP permettent de représenter de façon simple et naturelle le comportement des systèmes discrets à évolutions parallèles à l'aide seulement de deux types d'objets : les places et les transitions reliées par des arcs. L'ensemble des places permet de représenter les états du système et l'ensemble des transitions l'ensemble des événements dont l'occurrence provoque le changement d'état du système.

4 CONCLUSION

Ce court chapitre nous a permis de présenter un outil puissant de modélisation de systèmes complexes : les RdP.

En plus de ses qualités descriptives (modélisation élégante des mécanismes de synchronisation, parallélisme, concurrences....), ce modèle a l'avantage d'être un excellent outil de validation. C'est cette particularité qui va être exposée dans le chapitre suivant.

CHAPITRE 2

VALIDATION DES SYSTEMES DE COMMANDE

1 INTRODUCTION

La validation constitue l'une des tâches les plus importantes mais aussi les plus complexes du processus de conception d'un système de commande.
Cette tâche consiste à montrer que le fonctionnement d'un système est **conforme au service attendu** et qu'il possède **de bonnes propriétés de vivacité et de sûreté.**

Les propriétés liées à la vivacité permettent de garantir à des degrés divers, la capacité du système à prolonger indéfiniment des séquences d'opérations et les propriétés liées à la sûreté, sa capacité à ne pas engendrer des ordres erronés pouvant entraîner des conséquences désastreuses (perte de vie humaine, détérioration de matériels coûteux....).

L'utilisation des RdP comme modèle de description permet d'aborder ces différents problèmes liés à la validation d'un système de commande.

Les approches de validation généralement utilisées peuvent se regrouper en deux grandes familles : celles se fondant sur des approches statiques plus formelles, et celles se fondant sur des approches dynamiques plus empiriques. Nous consacrons ce chapitre à une présentation générale de ces approches.

2 LES METHODES STATIQUES

Les méthodes statiques ont essentiellement été étudiées pour les RdP autonomes, caractérisés par le fait que les évolutions de leurs marquages ne sont soumises à aucune contrainte particulière (synchronisation à l'occurrence d'événements externes, temporisation...).
Elles permettent d'analyser et de garantir **formellement les propriétés qui caractérisent le bon fonctionnement** d'un système de commande.
L'intérêt de ces méthodes réside dans le fait qu'elles sont **automatisables**, ce qui a permis la réalisation de nombreux outils OVIDE [Chezalviel 79] [Montel 83], OGIVED [Dufau 84], PETRIREVE [Choppy et Johnen 85], pouvant être intégrés dans des environnements d'aide à la conception.

Dans ce qui suit, nous allons d'abord présenter les différents types de

propriétés qui peuvent servir à caractériser le bon fonctionnement d'un RdP modélisant un système de commande. Nous verrons ensuite les méthodes d'analyse des RdP les plus importantes et nous les comparerons par rapport aux critères suivants:

- les types de propriétés qu'elles permettent de vérifier
- les classes de réseaux qu'elles permettent d'analyser
- et leur facilité de mise en œuvre.

2.1 PROPRIETES D'UN RdP

Les propriétés des RdP sont généralement réparties en deux catégories :
Les propriétés générales, elles sont requises pour tout type de réseau décrivant un système de commande. Elles permettent de garantir le bon comportement du système modélisé.

Les propriétés spécifiques, elles sont plus liées au problème traité. Elles expriment des contraintes particulières que doit respecter le système modélisé.

2.1.1 PROPRIETES GENERALES

Soit R = (P, T, Pré, Post) un RdP décrivant un système de commande, M_0 le marquage initial du réseau R et A(R, M_0) l'ensemble des marquages accessibles à partir de M_0

2.1.1.1 Propriété "borné" [Brams 83]

Définitions

On dit qu'une place p de R est **"k-bornée"** (k appartenant à l'ensemble des entiers naturels) pour M_0 si et seulement si pour tout marquage accessible M, le nombre de marque dans cette place est inférieur ou égal à l'entier k.
Dans la cas contraire la place p est non bornée.

Le réseau R est **borné** pour un marquage initial M_0 si et seulement si toutes ses places sont bornées.

La propriété "borné" permet de caractériser **la finitude des états** du système de commande modélisé, ce qui est en général le cas pour les systèmes réels. Sa non vérification peut de ce fait correspondre à une erreur de spécification.

EXEMPLE

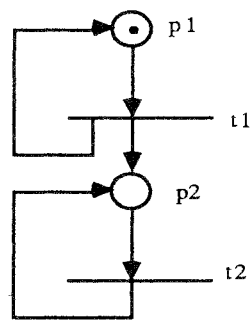


Figure 2.1

Ce réseau est non borné car la place p2 peut accumuler un nombre illimité de marques.

2.1.1.2 Propriété sauf [Brams 83]

Définition

On dit que R est un réseau **sauf** si et seulement si toutes ses places ont une borne inférieure ou égale à 1 pour tout marquage accessible à partir du marquage initial M_0.

Cette propriété est très intéressante car elle réduit le marquage d'une place à deux états vrai ou faux, bien adapté à la représentation des systèmes logiques.

2.1.1.3 Propriété de vivacité [Brams 83]

Cette propriété exprime le fait que quelle que soit l'évolution du système, toute action de ce système **pourra se reproduire**. La non satisfaction de cette propriété caractérise des situations de **blocage total** (i.e. état du système à partir duquel aucune évolution n'est plus possible, ce qui est bien évidemment une défaillance du système de commande) ou de **blocage partiel** c'est-à-dire

qu'une partie du système de commande est inaccessible.

Définitions

On dit qu'une transition t de R est **vivante** pour le marquage initial M_0 si et seulement si pour tout marquage M accessible depuis le marquage M_0, il existe un marquage M' accessible à partir du marquage M et tel que la transition t soit franchissable à partir du marquage M'.

Le réseau R est **vivant** pour le marquage initial M_0 si toutes ses transitions sont vivantes.

EXEMPLE

Soit le RdP de la figure 2.2, qui modélise l'allocation de deux ressources q (modélisée par la place p4) et r (modélisée par la place p5) à deux processus a et b. Le processus a, demande d'abord la ressource q, ensuite la ressource r puis libère ces deux ressources. Le processus b demande la ressource r ensuite la ressource q puis libère les deux ressources. Ce réseau est non vivant (il y a un interblocage entre les deux processus) car il suffit de franchir la séquence t1t4 ou t4t1 (le processus a possède la ressource q et le processus b possède la ressource r) pour que le réseau se bloque, les transitions t2 et t5 ne peuvent pas être franchies.

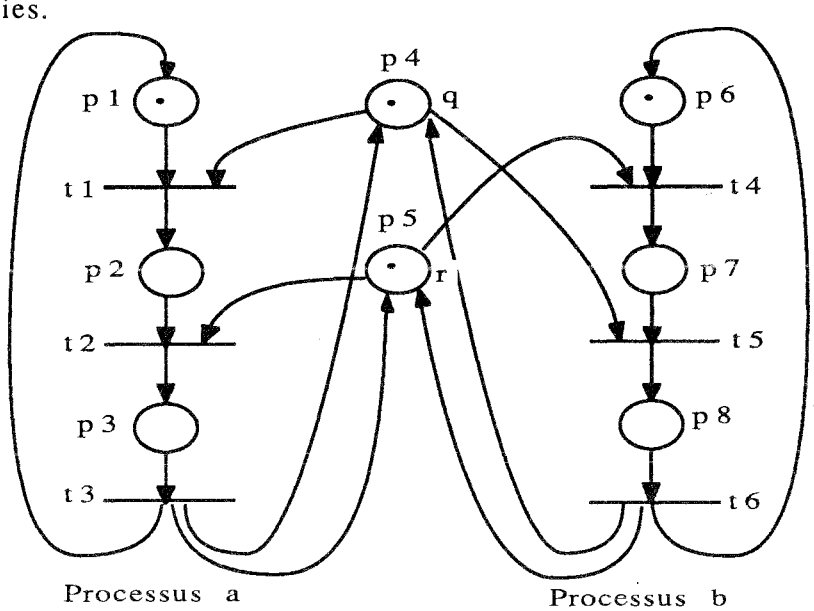


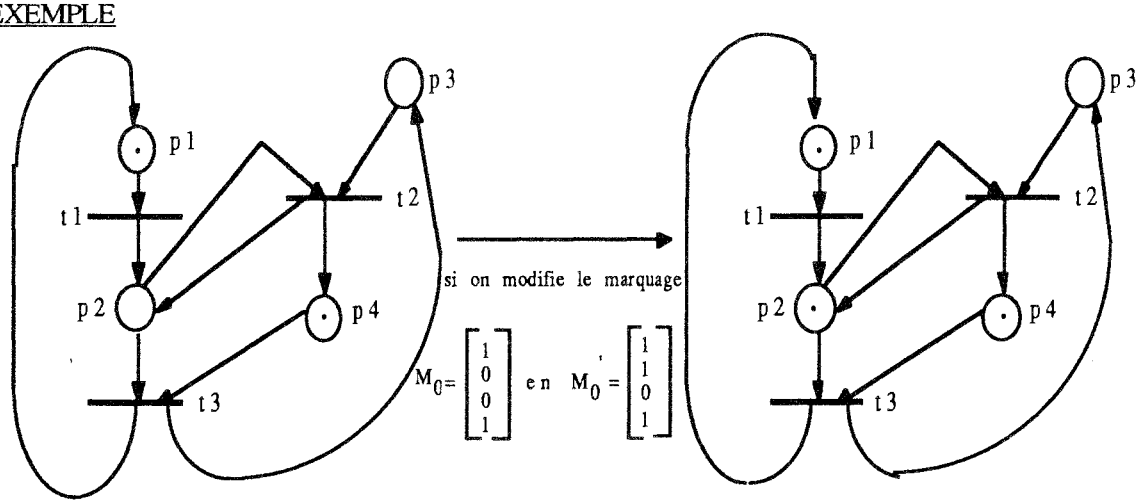
Figure 2.2

2.1.1.4 Propriété de réinitiabilité ou "propre" [Brams 83]

Définition

On dit que le réseau R est réinitialisable pour un marquage initial M_0 si et seulement si pour tout marquage accessible M il existe une séquence de franchissement s ramenant le réseau à l'état initial

EXEMPLE



réseau non réinitialisable réseau réinitialisable

Le réseau est non réinitialisable pour le marquage M_0. D'après le graphe des marquages accessibles du réseau (cf. figure 2.3), on voit bien qu'une fois t1 franchie, le réseau ne pourra plus revenir au marquage M_0. Par contre, pour le marquage M_0' le réseau est réinitialisable (cf figure 2.4) puisque pour tout marquage M accessible à partir de M_0' on peut trouver une séquence de franchissement qui ramène le réseau à M_0'.

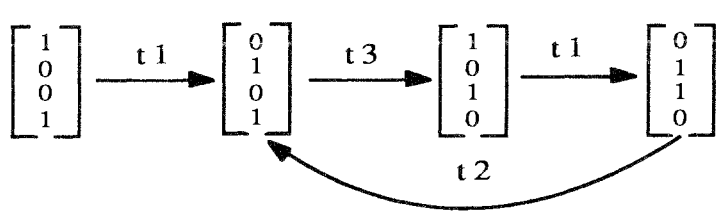


Figure 2.3

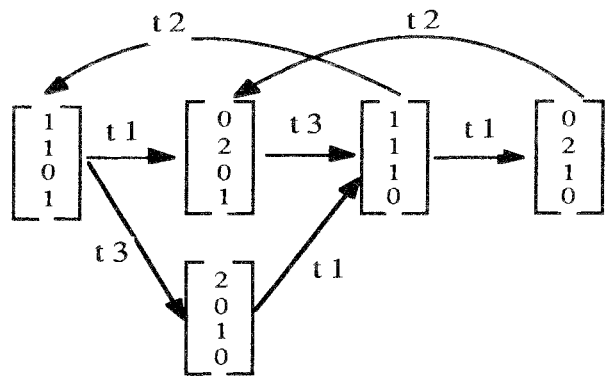


Figure 2.4

2.1.2 PROPRIETES SPECIFIQUES

Les propriétés spécifiques peuvent être utilisées pour prouver que l'évolution d'un système de commande respecte bien certaines contraintes. Parmi ces contraintes, on peut vouloir vérifier que le système dans son fonctionnement réalise bien le service attendu, ou vérifier des propriétés liées à la sûreté de fonctionnement tels l'exclusion mutuelle, l'absence d'ordres contradictoires..... etc.

L'étude de ces propriétés s'appuie sur des recherches d'invariants.

2.1.2.1 Invariant de places

Définition [Brams 83]

Un invariant de places du réseau R est un vecteur I_p de $|P|$ éléments tel que :

$$I_p{}^{tr}.\ C = 0 \text{ et } I_p \neq 0 \text{ où}$$

- C est **la matrice d'incidence** du réseau R définie par:

$$\forall p \in P\ ,\ \forall t \in T\ :\ C\ (p,t) = post(p,t)\ -\ pré(p,t)$$

- Ip^{tr} est le transposé du vecteur I_p

On notera $C = POST - PRE$ où $POST$ et PRE sont deux matrices qui représentent les fonctions post et pré du réseau c'est-à-dire que le coefficient (i,j) de la matrice POST (respectivement PRE) est donné par post(p_i,t_j) (respectivement pré(p_i,t_j)).

Théorème [lautenbach 74]

Si I_p est un invariant de places alors pour tout couple (M, M') de marquages accessibles à partir du marquage initial M_0, I_p satisfait la relation suivante:

$$I_p{}^{tr}.\ M = I_p{}^{tr}.\ M'$$

en particulier pour le marquage initial M_0

$$I_p{}^{tr}.M = I_p{}^{tr}.\ M_0 = k$$

k est une constante.

Une propriété spécifique qui est exprimée par une relation entre le marquage des places pourra être vérifiée par l'existence ou non de l'invariant correspondant.

2.1.2.2 Invariant de transitions

Définition [Brams 83]

Un invariant de transitions du réseau R est un vecteur I_t de $|T|$ éléments tel que :

$$C.\ I_t = 0 \text{ avec } I_t \neq 0$$

Les invariants de transitions caractérisent des séquences de franchissement cycliques du RdP.

2.2 METHODES D'ANALYSE

2.2.1 METHODE BASEE SUR L'ENUMERATION DES MARQUAGES

Cette méthode est due aux travaux de Karp et Miller [Karp et Miller 69], elle se fonde sur la construction de l'arbre des marquages, appelé **arbre de**

couverture. Le principe de base de la construction de cet arbre consiste à examiner, au cours de l'énumération des marquages, les cas où, à partir d'un marquage M, on peut accéder à un marquage M' tel que pour toute place p $M'(p) \geq M(p)$. D'après la définition de franchissement d'une transition (cf.chapitre 1), on voit bien que dans de telles situations, toute séquence de transitions s franchissable à partir de M l'est également à partir de M'. Il s'ensuit que si, pour une place p, M'(p) est strictement supérieur à M(p), alors en répétant la séquence de transitions qui fait passer de M à M', on peut augmenter de manière arbitrairement grande le nombre de marques dans la place p. Ce nombre très grand est représenté par un symbole spécial ω.

A partir de l'arbre de couverture est construit le graphe de couverture en groupant les sommets de l'arbre portant des étiquettes identiques (cf. figure 2.5). Le graphe de couverture est identique au graphe des marquages accessibles (cf. chapitre 1) dans le cas où le réseau est borné.
L'algorithme de construction de l'arbre de couverture et la preuve de sa terminaison peuvent être retrouvés dans [karp et Miller 69] [Peterson 81] [Brams 83].

2.2.1.1 Vérification des propriétés

La vérification des propriétés s'effectue simplement en examinant le graphe de couverture. Toutefois, dans le cas de réseaux non bornés, l'analyse du graphe de couverture seul, ne permet pas de montrer dans le cas général les propriétés de vivacité et de réinitiabilité.

Propriété "borné": le réseau R est borné si et seulement si il n'existe pas de sommets du graphe de couverture pour lequel le contenu d'une place est égal à ω.

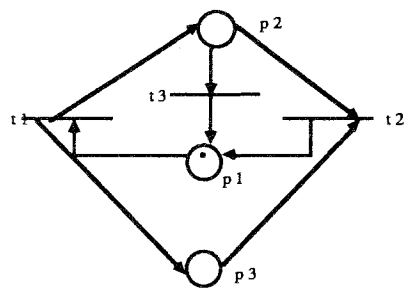
Propriété "sauf": le réseau R est sauf si et seulement si pour tout marquage du graphe, il n'existe pas de place qui contient plus d'une marque.

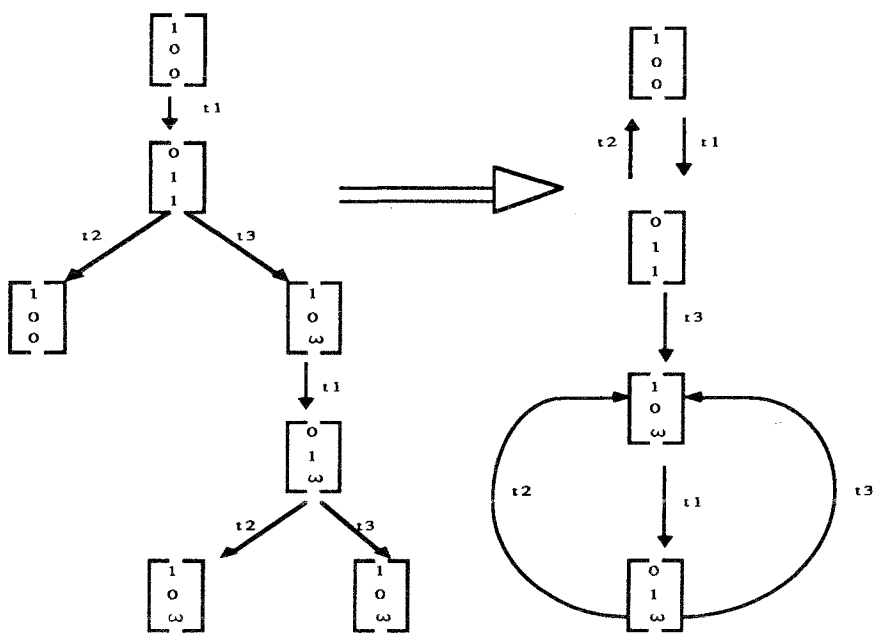
Propriété de réinitiabilité: si le réseau est borné pour un marquage initial M_0 alors ce réseau est réinitialisable pour ce marquage si et seulement si le graphe

de couverture est fortement connexe.

Propriété de vivacité: si le réseau est borné, il est vivant si et seulement si toutes ses transitions apparaissent au moins une fois dans chacune des composantes fortement connexes du graphe.
Si le réseau est borné et réinitialisable pour M_0, pour qu'il soit vivant, il suffit que toutes les transitions apparaissent au moins une fois comme étiquette d'un arc du graphe.

EXEMPLE





arbre de couverture graphe de couverture

Figure 2.5

D'après le graphe de couverture on voit bien que le réseau est non borné. La place p3 est non bornée (elle contient le symbole ω) .

2.2.1.2 Commentaires

La méthode par énumération des marquages ne permet que la vérification des propriétés générales d'un réseau (caractère borné, vivant, réinitialisable). D'autre part, elle se limite à l'analyse de réseaux bornés. Elle ne permet pas de vérifier les propriétés de vivacité et de réinitiabilité dans le cas où le réseau est non borné.
Par ailleurs, l'analyse du réseau est à refaire à chaque fois qu'on change d'état initial.
Enfin, l'explosion combinatoire des marquages accessibles rend l'utilisation de cette méthode très coûteuse, voire même impossible dans certains cas.

2.2.2 METHODE DE REDUCTION

Le principe de base de cette méthode consiste à réduire un réseau (R, M_0) à analyser en un autre réseau (R', M'_0) plus simple à analyser et ayant exactement les même propriétés que le réseau de départ.
Pour cela un ensemble de règles de réduction a été défini et étudié dans [Berthelot 84] [Brams 83]. L'application de ces règles peut permettre d'aboutir à un réseau complètement réduit : réseau formé d'une transition.
Lorsqu'un réseau est complètement réductible, on en déduit qu'il est borné et vivant. Dans le cas contraire, le réseau partiellement réduit peut être aisément analysé en utilisant d'autres méthodes comme l'énumération des marquages.

2.2.2.1 Commentaires

Cette méthode aussi ne permet que la vérification des propriétés générales. De plus, elle se limite à l'analyse de réseaux complètement réductibles, pour les autres réseaux, il faut faire appel à d'autres méthodes d'analyse.
D'autre part, la réduction d'un réseau peut conduire à une perte d'information. Lorsque le réseau réduit ne possède pas les propriétés voulues, cette perte d'information peut rendre délicate toute analyse fine du réseau et généralement tout le processus de conception et de description est à refaire.

2.2.3 METHODE D'ANALYSE STRUCTURELLE

Cette méthode s'attache à étudier les propriétés d'un RdP à partir de sa structure en faisant donc abstraction de l'ensemble des marquages accessibles. Elle consiste à extraire du réseau à analyser des structures particulières qui sont des invariants linéaires de places et de transitions. La connaissance de ces structures permet de caractériser des propriétés invariantes du système.

Soit un RdP R = (P, T, Pré, Post) et C sa matrice d'incidence.
Le passage du marquage initial M_0 à un marquage M par une séquence de franchissement s est alors décrit par l'équation fondamentale suivante:

$$M = M_0 + C.\,\bar{s}$$

$\bar{s}$: est le **vecteur caractéristique** de s, dont les composantes s(t) sont les nombres d'occurrences de chaque transition t dans la séquence s.

Cette équation est le point de départ de l'analyse structurelle par les techniques de l'algèbre linéaire. Son étude a conduit plusieurs auteurs [Lautenbach 74] [Ramchandani 73] [Berthomieu 79] [Genrich 80] [Memmi 83] [Toudic 82] [Alaiwan 85] à s'intéresser aux deux types d'invariants suivants :

- Les composantes consistantes ou répétitives construites sur l'ensemble des transitions.
- Les composantes conservatives construites sur l'ensemble des places.

2.2.3.1 Les composantes conservatives [Brams 83]

Définition

Soit $X_C \subseteq P$, X_C est une composante conservative si et seulement si il existe un vecteur f d'entiers positifs ou nuls tel que :

$f^{tr}.\,C = 0$ (f^{tr} : le transposé de f)

$||f|| = X_C$ i.e. la composante X_C est le support de f

Le vecteur f est appelé **p_semi_flot.**

Rappelons que le support d'un vecteur f de $\mathbb{N}^{|P|}$ noté $||f||$ est l'ensemble des éléments i de $\{1,\ldots,|P|\}$ tels que la i-ième composante de f soit non nulle.

Considérons l'équation fondamentale

$$M = M_0 + C.\vec{s}$$

En multipliant chaque membre de cette équation par le vecteur f^{tr} nous obtenons la relation :

$$f^{tr}.M = f^{tr}.M_0 \quad (1)$$

ce qui signifie que pour tout marquage M accessible à partir du marquage initial M_0, le produit scalaire <M, f> est constant et égal à $k = \Sigma\, f^{tr}(pi).M_0(pi)$, ainsi l'équation (1) s'écrit :

$$\Sigma\, f^{tr}(pi).\, M\,(pi) = k$$

Le p-semi-flot f, permet de déduire les invariants de places nécessaires pour vérifier les propriétés spécifiques (cf. chapitre 5) et le caractère borné d'un réseau. En effet, du fait que les composantes de f^{tr} sont non négatives, les places qui font partie de la composante conservative X_C, support de f, sont obligatoirement bornées pour tout marquage initial.

D'après la relation $f^{tr}.\, M = f^{tr}.\, M_0$, nous avons $\forall pi \in X_C\ f(pi).M(pi) \leq f^{tr}.M_0$

$$\forall\, pi \in X_C \quad M(pi) \leq \left\lfloor \frac{f^{tr}.M_0}{f\,pi)} \right\rfloor$$

$\lfloor X \rfloor$: représente la partie entière du réel X

De manière générale, si le réseau est lui même une composante conservative, c'est-à-dire qu'il existe f solution de $f^{tr}.C = 0$ ayant toutes ses composantes positives, alors le réseau est borné pour tout marquage initial.

2.2.3.2 Les composantes répétitives stationnaires [Brams 83]

Définition

Soit $Y_C \subseteq T$, Y_C est une composante répétitive stationnaire si et seulement si il existe un vecteur f d'entiers positifs ou nuls tel que :

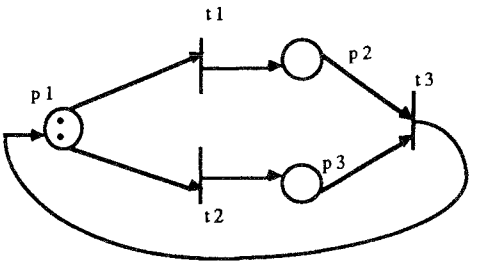
(i) $C.f = 0$

(ii) $|| f || = Y_C$

f est appelé **t_semi_flot.**

Les composantes répétitives stationnaires correspondent à des séquences de franchissement dont le franchissement ne modifie pas le marquage de départ.

Il est démontré dans [Ramchandani 73] que si le réseau n'est pas lui même une composante répétitive stationnaire, alors le réseau est soit non borné soit non vivant. Par contre s'il existe un marquage M pour lequel le réseau (R, M) est vivant alors le réseau est une composante répétitive. Comme le montre l'exemple suivant [Brams 83] la réciproque est fausse.



La séquence t1t2t3 est une séquence de franchissement qui ne modifie pas le marquage initial, donc le réseau est une composante répétitive. Pourtant le réseau (R, M_0) n'est pas vivant. Il suffit en effet de franchir t1t1 ou t2t2 pour qu'il y ait blocage.

2.2.3.3 Détermination des composantes conservatives et des composantes répétitives

Les composantes conservatives et répétitives s'obtiennent par le calcul de semi_flots solutions du système d'équations

$F^{tr}.C = 0$ *et* $C.F = 0$

Ces semi_flots sont caractérisés par les deux propriétés suivantes :

Propriété 1

Si f est un p_semi_flot (respectivement un t_semi_flot) alors $\forall \lambda \in \mathbb{Q}^+$ et $\lambda.f \in \mathbb{N}^{|P|}$ (respectivement $\lambda.f \in \mathbb{N}^{|T|}$), $\lambda.f$ est aussi un p_semi_flot (respectivement $\lambda.f$ est un t_semi_flot).

propriété 2

Si f et g sont deux p_semi_flots (respectivement t_semi_flots) alors f+g est aussi un p_semi_flot (respectivement un t_semi_flot).

D'après ces deux propriétés, il est clair que le nombre de semi_flots est infini, puisque toute combinaison linéaire de semi_flots est un semi_flot. C'est pourquoi la recherche des semi_flots s'est orientée vers la détermination d'une **base génératrice minimale** (de nombre fini) tel que chaque semi_flot peut s'écrire comme une combinaison linéaire des éléments de la base.

EXEMPLE

Pour mieux illustrer l'intérêt de ces composantes, considérons le RdP modélisant le système du producteur et du consommateur (cf. figure 2.6)

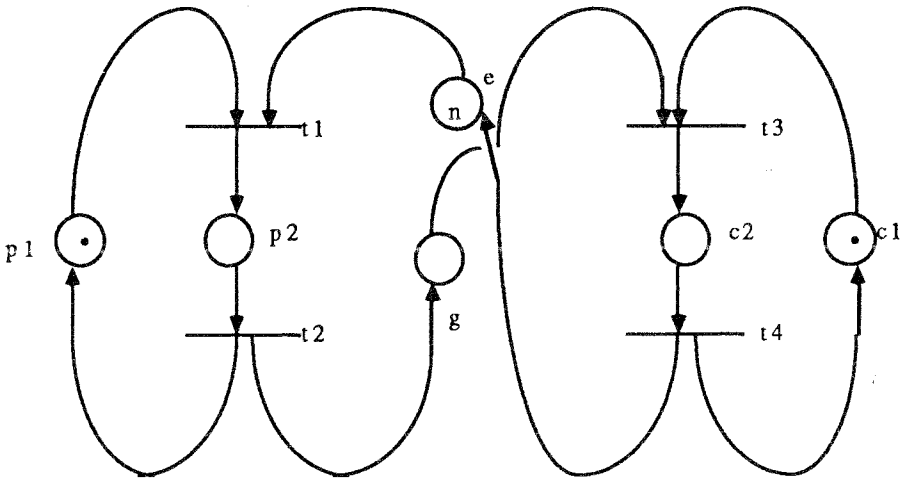


Figure 2.6

La matrice C = POST - PRE

$$C = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \end{bmatrix}$$

La résolution de l'équation $F^{tr}.\ C = 0$ fournit la base minimale suivante :

$$B = \begin{bmatrix} f1 & f2 & f3 \end{bmatrix} \quad \text{avec} \quad f1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad f2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad f3 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Ces trois p_semi_flots admettent comme support les trois composantes conservatives suivantes :

Xc1 = {p1, p2}, Xc2 = {c1, c2} et Xc3 = {p2 , c2, e, g}

L'obtention de ces composantes permet de prouver que :
le réseau est borné car il est une composante conservative. En outre, elles fournissent les invariants suivants :

• $f1^{tr}.\ M = f1^{tr}.\ M0 \Rightarrow M(p1) + M(p2) = 1$, qui signifie que le producteur est soit en attente de produire soit en train de produire,

• $f2^{tr}.M = f2^{tr}.M0 \Rightarrow M(c1) + M(c2) = 1$, qui signifie que le consommateur consomme ou en attente de consommer,

• et $f3^{tr}.M = f3^{tr}.M0 \Rightarrow M(p2) + M(c2) + M(e) + M(g) = N$, qui signifie qu'il y a conservation de positions dans le magasin.

2.2.3.4 Commentaires

L'analyse structurelle est la seule méthode qui permet de vérifier les propriétés spécifiques d'un réseau, les méthodes précédentes se limitent à la vérification des propriétés générales.
D'autre part, elle a l'avantage de pouvoir être appliquée à tous les réseaux sans exception (réseaux non bornés, réseaux avec un fort taux de parallélisme,...) .
Enfin, elle permet de s'affranchir de l'explosion combinatoire du nombre de marquages à laquelle se heurte la méthode par énumération et d'être indépendante du marquage initial. Elle est basée sur la seule connaissance du graphe du réseau et donc sur sa matrice d'incidence.
Ainsi, compte tenu de ces remarques, nous avons choisi de nous appuyer sur cette méthode pour aborder dans PETRI-S, le problème de la validation.

2.2.4 METHODE BASEE SUR LA REECRITURE

Cette approche utilise les systèmes de réécriture pour analyser et vérifier les propriétés d'un RdP. Le principe dû à Choppy et Johnen [Choppy 85] consiste à décrire le fonctionnement du RdP à analyser par un système de règles de réécriture et à vérifier les propriétés de ce système de réécriture afin d'en déduire celles du RdP.

Avant de décrire le principe de cette méthode nous allons d'abord donner quelques notions de bases sur la réécriture.

2.2.4.1 Notions de bases sur la réécriture [Jouannaud et Lescanne 85]

La réécriture est, à l'origine, une méthode de preuve en logique équationnelle. Dans cette logique, les axiomes sont des équations, et les théorèmes sont déduits des axiomes par une règle d'inférence très simple: le remplacement d'égaux par des égaux. En fait, il s'agit de relier les deux membres de l'équation par une chaîne de remplacements utilisant les axiomes. Cette méthode s'avère inefficace car elle nécessite de nombreux retours en arrière sur les choix des axiomes à appliquer : elle est **indéterministe**

L'idée de la réécriture est de limiter les choix de remplacement d'égaux par des égaux jusqu'à rendre le processus **déterministe**. Elle consiste à orienter les axiomes, notés sous forme a==b, en équations à sens unilatéral, ou **règles de réécriture**, notées $a \rightarrow b$.
Ces règles de réécriture forment ce que l'on appelle **un système de réécriture**.

La contrainte d'orientation constitue cependant un appauvrissement de l'ensemble initial des axiomes. Knuth et Bendix proposèrent une procédure connue sous le nom de **la procédure de complétion** [Knuth et Bendix 70] qui consiste à enrichir l'ensemble des règles issues des axiomes par de nouvelles règles appelées **paires critiques**. Une paire critique traduit le fait qu'un membre gauche de la règle peut se réécrire à l'aide d'une autre règle du système. Le système de réécriture ainsi obtenu est équivalent à l'ensemble initial d'axiomes, et a donc la même puissance de déduction.

Ce système de règles de réécriture est utilisé pour calculer des termes, c'est-à-dire réécrire des termes. Réécrire un terme t consiste à choisir un sous_terme t' de t et une règle de réécriture $g \rightarrow d$, à vérifier que t' est une instance de g puis à remplacer t' dans t par l'instance correspondante de d. On itère ce processus jusqu'à ce que le terme ne puisse plus être réécrit par aucune règle du système de réécriture. On dit alors que le terme est irréductible ou en **forme normale.**
Le calcul de la forme normale d'un terme nécessite aussi bien la terminaison du calcul que son unicité (le résultat du calcul d'un terme ne dépend pas du choix de la règle à appliquer). Ces deux caractéristiques traduisent deux propriétés essentielles d'un système de réécriture : **la terminaison** et **la confluence**.

Un système de réécriture qui possède ces deux propriétés est dit **convergent**. Si le système de réécriture est convergent alors prouver la validité d'un théorème équationnel $t_1 = t_2$ revient à calculer les formes normales de t_1 et t_2 et à vérifier si elles sont syntaxiquement égales.

EXEMPLE

Soient les deux axiomes équationnels suivants qui définissent l'addition sur les entiers:

$$X+0=X$$

et $$X + S(Y) = S(X+Y)$$

où S est la fonction successeur des entiers naturels.

En orientant ces deux équations de la gauche vers la droite, nous obtenons le système de réécriture suivant :

$$(1)\ X + 0 \rightarrow X$$

$$(2)\ X + S(Y) \rightarrow S(X+Y)$$

Pour effectuer la démonstration du théorème S(X)+0 = X+S(0), il suffit de montrer que les membres de l'équation ont la même forme normale.

Considérons le terme S(X)+ 0 qui n'est pas en forme normale. Ce terme se réécrit en S(X) en utilisant la première règle du système de réécriture. Le terme S(X) est en forme normale.

Considérons l'autre terme X + S(0). Ce terme se réécrit en S(X+0) en utilisant la deuxième règle. Ce terme n'étant toujours pas en forme normale, il peut se réécrire à l'aide la première règle en S(X). Les deux formes normales étant syntaxiquement égales, on en déduit que le théorème est valide dans le système de réécriture.

2.2.4.2 Représentation d'un RdP à l'aide d'un système de réécriture

Le fonctionnement d'un RdP est décrit par un ensemble de règles de réécriture. A chaque transition t du réseau est associée une règle de réécriture définie comme suit :

• La partie gauche de la règle représente la précondition de franchissement de la transition t, c'est-à-dire le marquage validant t. Ce marquage est décrit par un symbole opératoire "state" d'arité n (n est le nombre de places du réseau). Chaque argument de cet opérateur correspond à une place pi du réseau, et sa

valeur vpi au nombre de marques dans cette place.

• La partie droite de le règle représente la postcondition de franchissement de t, c'est-à-dire le marquage résultant du tir de la transition.

A cet ensemble de règles est ajoutée la règle qui correspond au blocage total du réseau, décrite comme suit :

STATE(arg1, arg2, ...) → DEADLOCK(arg1, arg2, ...)

Cette règle exprime le fait que la partie droite ne peut plus être réécrite en utilisant les autres règles.

EXEMPLE

Considérons le RdP suivant qui modélise l'opération de multiplication par deux d'un entier donné.

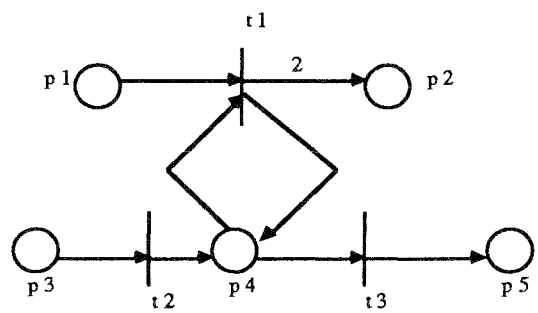


Figure 2.7

p1 : contient l'entier qui va être doublé
p2 : contient le résultat de l'opération
p3 : marque le début de l'opération
p4 : indique que l'opération est en cours
p5 : marque la fin de l'opération

Ce réseau peut être décrit par le système de réécriture suivant :

t1 : STATE(vp3, S(vp4), vp5,S(vp1),vp2)→STATE(vp3,S(vp4),vp5,vp1, S(S(vp2)))

t2 : STATE(S(vp3),vp4,vp5,vp1,vp2) → STATE(vp3,S(vp4),vp5,vp1,vp2)

t3 : STATE(vp3,S(vp4),vp5,vp1,vp2) → STATE(vp3,vp4,S(vp5),vp1,vp2)

STATE(0, 0, vp5, vp1, vp2) → DEADLOCK(0, 0, vp5, vp1, vp2)

S désigne la fonction successeur. Par exemple S(vp1) = vp1 + 1, signifie que la

place p1 contient au moins une marque.

La première règle du système correspond au franchissement de la transition t1. La partie gauche de la règle représente le marquage validant t1 (i.e. les places p4 et p1 doivent avoir au moins une marque).
La partie droite de la règle représente le marquage résultant du franchissement de t1. Ce marquage est obtenu en retirant une marque de la place p1 et une marque de la place p4, puis en déposant une marque dans la place p4 et deux marques dans la place p2. On peut donc en déduire qu'après franchissement, le marquage de p1 passe à vp1, le marquage de p4 reste inchangé et le marquage de p2 passe à S(S(vp2)).

La deuxième et la troisième règle du système de réécriture, correspondent aux franchissements des transitions t2 et t3.
La dernière règle exprime l'état de blocage du réseau. Les transitions t1 et t3 ne peuvent plus être franchies que si le marquage de p4 est nul. De même, la transition t2 devient infranchissable que si le marquage de p3 est nul.

La construction de cet ensemble de règles de réécriture est faite automatiquement par le système PETRIREVE. Ce logiciel 'extrait' ces règles à partir de la représentation graphique du RdP crée à l'aide de l'éditeur PETRIPOTE [Beaudouin 83]. l'ensemble de règles est ensuite soumis sous forme d'équations (STATE(arg1,arg2,.....) == STATE(arg1, arg2,.....)) au logiciel REVE (laboratoire de réécriture) [Lescanne 83].

2.2.4.3 Principe de la méthode

Le principe de base de cette méthode est de montrer que le système de réécriture décrivant le fonctionnement d'un RdP à analyser est convergent. Pour cela la procédure de complétion du logiciel REVE commence d'abord par transformer l'ensemble d'équations en un ensemble de règles de réécriture. Ensuite elle ajoute à cet ensemble de règles toutes les paires de termes de taille minimale qui proviennent de la réécriture d'un terme de deux façon différentes (paires critiques).Une paire critique, traduit le fait que pour un marquage

donné deux transitions sont franchissables et que le franchissement de l'une invalide l'autre, on dit que les transitions sont en **conflit**.
Par exemple le marquage décrit par STATE(vp3, S(vp4), vp5, S(vp1), vp2) du réseau de la figure 2.7 valide les deux transitions t1 et t3.
Ainsi le terme STATE(vp3, S(vp4), vp5, S(vp1), vp2) se réécrit en STATE(vp3,S(vp4),vp5, vp1, S(S(vp2))) dans le cas où la transition t1 est franchie et il se réécrit en STATE(vp3,vp4,S(vp5),S(vp1),vp2), lorsque la transition t3 est franchie.

Les paires critiques sont ensuite orientées de manière à ce que le système soit à terminaison finie. Ce processus est itéré jusqu'à ce que, le cas échéant, la procédure s'arrête en fournissant un système convergent.

2.2.4.4 Analyse des propriétés du RdP

Réseau borné : la propriété "borné" d'un RdP est liée à la terminaison du système de réécriture décrivant le fonctionnement du réseau. Ainsi si le système de réécriture est à terminaison finie alors, le RdP est borné. Ceci s'explique par le fait que si le système termine, alors la réécriture des termes (marquages du réseau) ne peut pas engendrer des chaînes de dérivation infinies, ce qui équivaut au fait que les places du réseau ne peuvent contenir qu'un nombre limité de marques. Par contre, si le système de réécriture ne termine pas, le caractère borné du réseau n'est pas décidable.

Réseau vivant et réinitialisable : ces deux propriétés ne peuvent pas être vérifiées par cette méthode. Seule la **quasi-vivacité** d'une transition peut être vérifiée. La quasi-vivacité d'une transition désigne la possibilité de franchir au moins une fois cette transition depuis le marquage initial. Pour prouver cette propriété, il suffit de montrer que le marquage permettant le franchissement de cette transition est accessible à partir du marquage initial. L'accessibilité à un marquage M depuis un marquage initial M_0 peut être prouvée en ajoutant l'équation M_0==M au système de réécriture et en montrant que M_0 peut se réécrire en une ou plusieurs étapes en M.

Enfin, cette méthode ne permet pas de déterminer les différents invariants

nécessaires pour vérifier les propriétés spécifiques. Le concepteur peut toutefois s'assurer de la vérification de ces propriétés en écrivant lui même ces invariants sous forme d'équations et en les ajoutant au système de réécriture décrivant le fonctionnement du RdP. Si la procédure de complétion termine, alors cela suffit pour affirmer que ces propriétés sont vérifiées. Dans le cas contraire, il est impossible de tirer une conclusion.

2.2.4.5 Commentaires

Les résultats obtenus par cette méthode d'analyse semblent avoir une portée limitée :
elle ne donne pas d'informations concernant les propriétés de vivacité et de réinitiabilité. Par ailleurs, la propriété "borné" ne peut pas être vérifiée dans tout les cas de figure. Ceci est dû au fait que la propriété de bornage du RdP est liée à la propriété de terminaison du système de réécriture décrivant le fonctionnement du RdP et que la terminaison des systèmes de réécriture est une propriété indécidable [Huet et Lankford 78]. Toutefois, il existe des algorithmes qui permettent de résoudre le problème de la terminaison dans certains cas particuliers.

2.2.5 DISCUSSION

Il convient maintenant de conclure sur la présentation des différentes méthodes d'analyse qui viennent d'être présentées. C'est surtout sur les limites de ces méthodes pour valider complètement un système de commande que notre discussion va porter. On peut d'abord noter, qu'aucune méthode ne suffit à elle toute seule pour vérifier toutes les propriétés d'un réseau. Le concepteur peut s'appuyer sur une combinaison "intelligente" de ces méthodes. Il effectue ses analyses progressivement en choisissant judicieusement les méthodes qui s'appliquent le mieux en fonction de la structure du réseau et de la propriété à étudier.
Toutefois, même une utilisation combinée de l'ensemble des méthodes, reste insuffisante pour valider un système de commande. Ceci est dû au fait que ces méthodes ne permettent d'analyser que des réseaux autonomes, et que ces derniers s'avèrent insuffisants pour modéliser les systèmes de commande.

Comme nous l'avons déjà souligné, un système de commande est un système non autonome, dont l'évolution est soumise à l'influence du comportement de l'environnement qu'il commande. Il est donc absolument nécessaire que le modèle de description de ce type de systèmes soit capable de prendre en compte les échanges d'informations qui existent entre le système de commande et son environnement.
Or les RdP autonomes ne servent qu'à représenter les contraintes fonctionnelles (séquencement des actions, parallélisme, allocation de ressources...) laissant de côté tous les problèmes liés à l'interaction du système de commande avec son environnement. Des extensions des RdP ont été étudiées, afin de prendre en compte ces interactions, c'est le cas des schémas à RdP de Valette [Valette 76] (ce modèle est formé d'un graphe de commande qui est le RdP étiqueté, d'un graphe de données et d'une interprétation) et des RdP interprétés [Brams 83] [Moalla 85]. Tout en augmentant la puissance de description des RdP autonomes, ces extensions ont l'inconvénient de remettre en cause une partie des résultats théoriques des RdP autonomes en rendant indécidables un certain nombre de propriétés importantes. Ainsi les méthodes formelles de validation de systèmes décrits par ces extensions sont loin d'être disponibles.

Les approches généralement employées consistent à analyser la structure du système de commande, c'est-à-dire le RdP 'dépouillé' de son interprétation en utilisant les méthodes d'analyse des RdP autonomes. Cette analyse permet au concepteur de détecter certaines erreurs de description et de conception. Toutefois, elle reste insuffisante, elle ne permet pas par exemple de montrer que le système ne peut pas se bloquer. La propriété de blocage et plus généralement la propriété de vivacité est étroitement liée à l'interprétation superposée au réseau (ce point sera traité en détail dans le chapitre 5).
Pour pallier cet inconvénient, des approches basées sur le principe de la certification a priori ont été proposées dans [Valette 76] et [Tankoano 88]. Elles proposent un ensemble de règles de construction de réseaux, qui permettent de garantir, par avance la vérification des propriétés recherchées. Ces approches, ont cependant l'inconvénient d'être trop restrictives quant à la classe de réseaux qu'elles permettent de construire.
Dans de nombreux cas, le recours à la simulation reste inévitable.

2 METHODES DYNAMIQUES

Les méthodes dynamiques permettent de valider un système de commande en simulant l'ensemble du système de commande et de l'environnement avec lequel il interagit. Ces méthodes ont l'inconvénient de ne pas être exhaustives. Elles permettent de détecter des erreurs mais jamais de garantir leur absence. En effet, comme l'a dit Dijkstra "faire fonctionner un programme ne peut que démontrer qu'il est faux, jamais qu'il est correct".

Nous allons expliciter les principes de ces méthodes à travers la présentation d'un logiciel spécialisé dans la validation par simulation de systèmes de commande décrits par des RdP.

2.1 LE SYSTEME SICLOP

Le système SICLOP (SImulateur de Commande LOgique et de Procédé) développé au LAAS [Benzakour et Valette 86] est un logiciel spécialisé dans la validation des systèmes de commande. Il permet, en simulant conjointement le fonctionnement du système de commande et le comportement du système commandé, de détecter les incohérences dans les échanges d'informations et les erreurs de conception. Le système de commande ainsi que le système commandé sont décrits par des RdP auxquels est associée une interprétation particulière.

Nous décrirons tout d'abord, les différents modèles utilisés dans SICLOP. Ensuite nous présenterons les principes de fonctionnement du simulateur.

2.1.1 DESCRIPTION DES MODELES

2.1.1.1 Modèle de description du système de commande

La spécification du système de commande est basée sur des RdP 'décorés' d'une interprétation. Cette interprétation sert à préciser l'interaction du système de commande avec le système commandé. Ainsi, comme le montre la figure 2.8, aux places du RdP sont associées des ordres de commande destinés au

système commandé, et aux transitions des conditions et des actions.

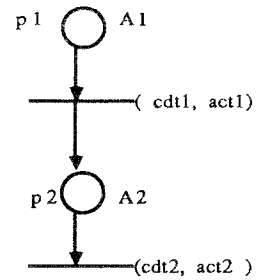


Figure 2.8

A1, A2 : sont des commandes d'actionneurs,

cdt1, cdt2 : expressions booléennes composées de variables internes, externes (capteurs) ou des deux,

act1, act2 : des actions sur les variables internes

2.1.1.2 modèle de description du système commandé

Dans SICLOP on modélise le comportement du système commandé par des RdP particuliers qui sont des machines à état. Ces réseaux sont caractérisés par le fait que chaque transition est reliée à une seule place en entrée et à une seule place en sortie par des arcs de valuation 1.

Chaque grandeur physique dont on veut suivre l'évolution est décrite par une machine à état qui ne doit contenir à tout instant qu'une seule marque à laquelle est associée une variable X indiquant la position courante de la grandeur physique. Les franchissements des transitions de la machine à état peuvent être synchronisés à des ordres de commande engendrés par le système de commande. Par ailleurs, à chacune des places du réseau, il peut être associé une vitesse constante (V) et une liste (L).

(V) désigne la vitesse à laquelle la grandeur physique évolue. Cette vitesse peut être positive si la grandeur augmente, ou négative si au contraire la grandeur diminue.

Chaque composante de la liste (L) est un couple désignant

-un événement de modification d'une variable d'entrée du système de commande

-et la position de la grandeur physique à cet instant.

Lorsqu'une marque est introduite dans une place à laquelle sont associées une vitesse (V) et une liste (L) (cf. figure 2.9). La position de la grandeur physique associée à cette marque évolue à la vitesse (v) jusqu'à ce que tous les événements répertoriés dans la liste (L) soient engendrés. Pendant ce temps, la marque est considérée comme indisponible. Toutefois, la génération par la partie commande de l'action associée à la transition en sortie de cette place (action A'), avant que la marque ne soit devenue disponible, provoque un tir forcé de cette transition, et un arrêt de l'évolution de la grandeur physique.

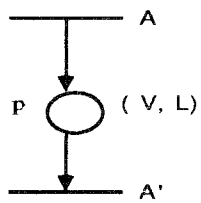


Figure 2.9

EXEMPLE

Considérons un chariot qui se déplace sur une voie à double sens entre un poste de chargement et un poste de déchargement (cf. figure 2.10). Devant chaque poste, il existe un capteur permettant de détecter la présence ou non du chariot. La liste (L) (figure 2.11) indique les différentes valeurs des variables d'entrée (états des capteurs) en fonction des différentes valeurs de la variable X représentant la grandeur physique 'chariot'.

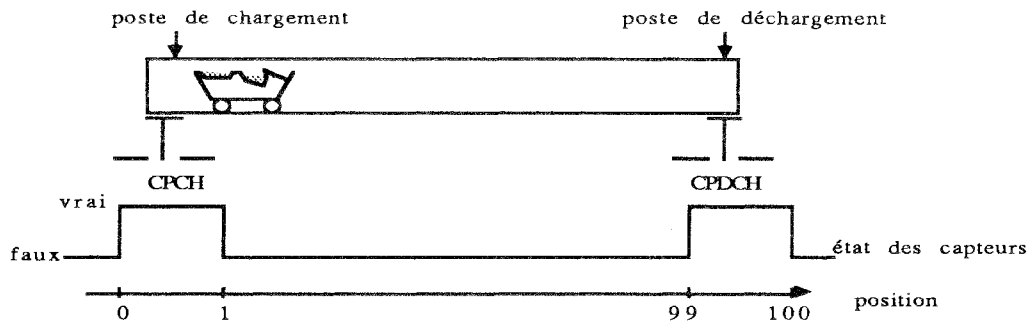


Figure 2.10

CPCH : capteur du poste de chargement

CPDCH : capteur du poste de déchargement

(L)

Position de la grandeur X	Capteurs	Etat des capteurs
0	CPCH	vrai
1	CPCH	faux
99	CPDCH	vrai
100	CPDCH	faux

Figure 2.11

La figure 2.12 modélise le comportement du chariot.

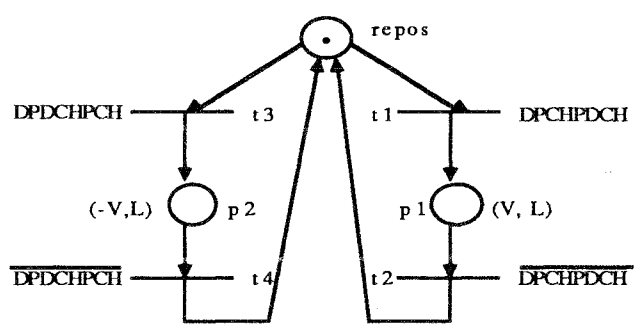


Figure 2.12

DPCHPDCH et DPDCHPCH sont deux actionneurs.

DPCHPDCH : actionne le déplacement du chariot dans le sens poste de chargement vers le poste de déchargement

DPDCHPCH : actionne le déplacement du chariot dans le sens poste de déchargement vers le poste de chargement

2.1.2 PRINCIPE DE FONCTIONNEMENT DE SICLOP

Comme le montre la figure 2.13, le système SICLOP est constitué de deux simulateurs : un pour la simulation du système de commande et l'autre pour La simulation du système commandé. Les deux simulateurs ont en commun un échéancier qui assure le dialogue entre eux. L'échéancier comprendra donc, tous les changements d'état des capteurs (destinés à la partie commande) et des actionneurs (destinés à la partie physique) classés en temps croissant.

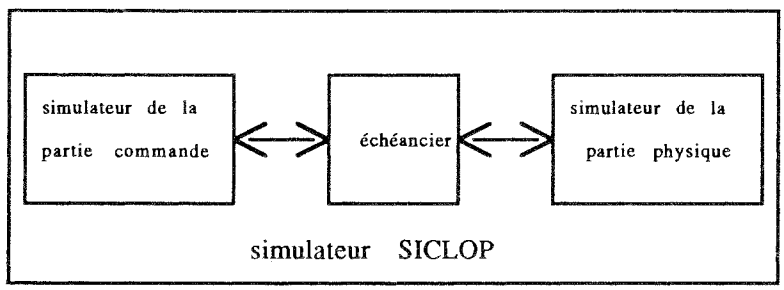


Figure 2.13

Le système SICLOP commence par lire le premier événement se trouvant en tête de l'échéancier. Il active ensuite le simulateur correspondant au type de l'événement (simulateur de la partie physique s'il s'agit d'un changement d'état d'un actionneur, ou le simulateur de la partie commande, lorsque l'événement est un changement d'état d'un capteur). Le simulateur désigné commence par prendre en compte tous les événements le concernant et correspondant au temps courant. Ensuite il fait évoluer le réseau jusqu'à atteindre un état stable, état à partir duquel le réseau ne peut plus évoluer sans prendre en compte de nouveaux événements. Au cours de la recherche de l'état stable les événements résultant des franchissements des transitions seront introduits dans l'échéancier, et ne seront utilisés qu'après l'état stable. La simulation se poursuit en considérant l'événement suivant.

EXEMPLE

Nous allons illustrer sur un exemple le fonctionnement du simulateur SICLOP. Reprenons l'exemple précédent du chariot qui se déplace sur une voie à

double sens. Le réseau de la figure 2.13 modélise la commande de déplacement du chariot.

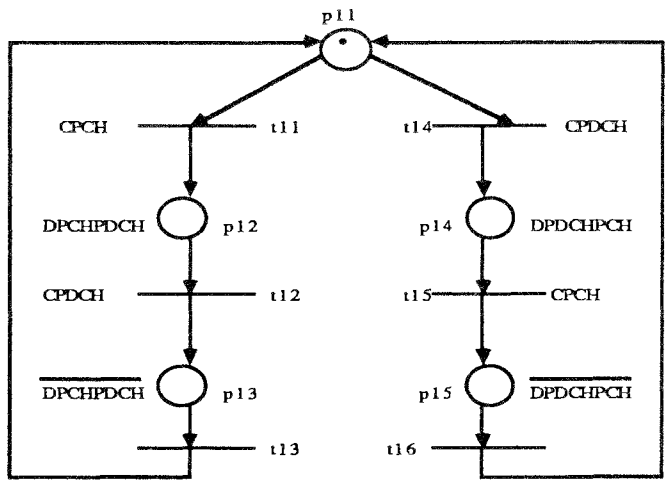


Figure 2.13

Si initialement le chariot est devant le poste de chargement, la place "repos" (cf. figure 2.12) est marquée et la valeur de la grandeur physique est nulle (X=0), alors d'après la liste (L) le capteur CPCH est vrai. Si cette information est contenue dans l'échéancier initial alors le simulateur de la partie commande est activé. Ce simulateur commence par tirer la transition t11 et engendre donc l'ordre de déplacement du chariot dans le sens 'poste de chargement-poste de déchargement'. Ceci se traduit par l'insertion dans l'échéancier de l'événement mise à vrai de l'actionneur DPCHPDCH. Cet événement va permettre au simulateur de la partie physique de tirer la transition t1 de la machine à état (la place 'repos' est marquée et la condition DPCHPDCH est vraie). La place qui vient d'être marquée p1 possédant une vitesse V (vitesse de déplacement du chariot) il est donc possible de calculer à partir de la liste (L), l'instant où un événement changement d'état d'un capteur va se produire. Cet événement correspondant à la mise à faux du capteur CPCH (X = 1), est alors inséré dans l'échéancier pour la partie commande. Le système de commande n'ayant pas réagi à cette information, la grandeur X continue d'augmenter jusqu'à la nouvelle valeur X = 99 correspondant à la mise à vrai du capteur CPDCH.

Si la commande continue toujours à ignorer les comptes rendus de la partie physique, alors la variable X continue d'augmenter jusqu'à atteindre la valeur 100, équivalent à la mise à faux du capteur CPDCH. La marque qui est contenue

dans p1 devient alors disponible, c'est-à-dire qu'elle peut être utilisée pour le déclenchement de la transition t1 : ce qui correspond à un fonctionnement anormal. Ces types d'anomalies sont détectables dans SICLOP au moyen des statistiques, il suffit de vérifier si les places qui modélisent un déplacement comme p1 et p2 ont un temps moyen de disponibilité de la marque différent de zéro.
Si, par contre, la commande réagit à l'événement 'mise à vrai' du capteur CPDCH en envoyant un ordre d'arrêt du chariot (tir de t12 et marquage de p13) c'est-à-dire mise à faux de l'actionneur DPCHPDCH alors la transition t2 de la machine à état sera tirée. Le chariot passe à l'état d'arrêt (place "repos" marquée) et la grandeur X cesse d'augmenter.

La place "repos" étant de nouveau marquée, si la commande actionne le déplacement du chariot dans le sens poste de déchargement-poste de chargement ('mise à vrai de DPDCHPCH'), alors la transition t3 sera tirée et l'on reprend de la même façon que pour le déplacement dans le sens poste de chargement vers le poste de déchargement sauf que dans ce cas la variable X sera décrémentée dans p2 (-V).

4 CONCLUSION

Nous avons présenté à travers cette étude les limites des méthodes employées pour valider les systèmes de commande décrits par des RdP. Nous avons d'abord expliqué que les méthodes d'analyse statique se limitent aux RdP autonomes et que ces derniers sont insuffisants pour modéliser les systèmes de commande. Les RdP interprétés qui sont des extensions des RdP, proposés pour représenter de tels systèmes ne se prêtent malheureusement pas à une analyse statique des propriétés. Ces extensions ont la même puissance qu'une machine de Turing [Brams 83]. De ce fait, un bon nombre de propriétés de ces réseaux sont indécidables.
Dans cette étude nous avons également présenté les approches de validation par simulation. Ces approches permettent de tenir compte de l'influence de l'environnement sur le comportement du système de commande et de détecter donc les incohérences dans les échanges d'information entre eux ainsi que les

erreurs de conception. L'inconvénient majeur de ces approches est de ne pas pouvoir garantir que toutes les erreurs ont été détectées.

L'approche que nous avons retenue est une **approche mixte** qui permet de combiner les apports des approches statiques et des approches par simulation. Le principe de cette approche consiste, en utilisant la méthode d'analyse structurelle, (cf. § 2.3) à déterminer **deux catégories de structures** du RdPI :

• Des structures où le nombre de marques reste invariant pendant toute l'évolution du réseau (composantes conservatives et composantes répétitives). Ces structures invariantes sont utilisées pour la vérification des propriétés spécifiques et du caractère borné du RdPI modélisant le système de commande. Dans le cas où l'analyse statique de ces structures n'est pas concluante, alors, elle procède à la vérification de ces propriétés au cours de la simulation du fonctionnement du RdPI.

• Des structures appelées **verrous**. Un verrou est un ensemble de places tel que s'il ne contient pas suffisamment de marques, il ne peut plus en recevoir et ses transitions en entrée comme en sortie ne sont plus franchissables. Ces verrous sont utilisées au cours de la simulation du fonctionnement du réseau pour détecter d'éventuels blocages du système.
Cette approche sera décrite en détail dans le chapitre 5.

CHAPITRE 3

EVALUATION DES PERFORMANCES DES SYSTEMES DE PRODUCTION

1 INTRODUCTION

Comme nous l'avons déjà souligné, la conception de systèmes de production nécessite, par sa complexité et ses implications économiques, des outils d'aide qui permettent d'évaluer le plus tôt possible les performances d'un système. Cette évaluation est nécessaire pour le choix de stratégies de pilotage et pour le dimensionnement du système de production (nombre de chariots, capacité des stocks, nombre de postes d'usinage,...).

Pour évaluer les performances d'un système de production deux approches sont possibles, la **simulation** ou les **méthodes analytiques**.

Dans ce chapitre, nous présentons dans les grandes lignes ces approches, en faisant surtout ressortir leurs avantages et leurs inconvénients. Nous terminons ce chapitre par la présentation de l'approche que nous avons adoptée dans notre système.

2 LES METHODES ANALYTIQUES

Les méthodes analytiques permettent de décrire le fonctionnement du système de production **en confondant partie commande et partie opérative**, puis de calculer directement des indices de performances tels :
le taux d'activité des machines, le taux de production du système, les temps d'attente, la capacité de chaque stocks....
Ces méthodes ont l'avantage de **fournir rapidement des résultats**. Elles sont, donc souvent utilisées dans les premières phases du processus de conception pour évaluer différentes solutions possibles.
On distingue deux types de méthodes :

- les méthodes basées sur la théorie des files d'attente,

- et celles basées sur les RdP : RdP temporisés et RdP stochastiques.

Ces méthodes diffèrent par le mode de représentation et par la finesse de description qu'elles permettent.

2.1 LES METHODES ANALYTIQUES BASEES SUR LA THEORIE DES FILES D'ATTENTE

La théorie des files d'attente a été introduite pour l'analyse des systèmes de production et de communication [Jackson 63] [Gordon et Newell 67]. Cependant, c'est pour l'analyse des performances des systèmes informatiques que cette théorie a été surtout développée. Mais la mise en œuvre de systèmes de production automatisée capables de réaliser plusieurs types de produits simultanément (les ateliers flexibles), a amené un regain d'intérêt dans l'utilisation de la théorie des files d'attente pour l'analyse des systèmes de production.

Dans ce qui suit, nous allons brièvement définir le modèle de files d'attente. Le lecteur désirant plus d'informations pourra se référer à [Dubois et Cavaille 82] [Gelenbe et al 80] [Gelenbe et Pujolle 85].

2.1.1 GENERALITES SUR LES RESEAUX DE FILES D'ATTENTE

Un réseau de files d'attente est composé d'un ensemble de stations de service ou guichets et d'un ensemble de clients. Chaque station se compose d'une file d'attente et d'un ou de plusieurs serveurs.

Les clients sont caractérisés par des attributs précisant leur routage à travers le réseau et leurs temps de service aux stations. Ces clients peuvent être partitionnés en classes. Les clients d'une même classe sont identiques du point de vue des probabilités de routage et des temps de service. Ainsi, un réseau de files d'attente sera du type **monoclasse** s'il n'existe qu'une seule classe de clients, ou de type **multiclasse** s'il en existe plusieurs. Une seconde caractéristique d'un réseau de files d'attente est d'être **ouvert** ou **fermé.** Dans un réseau ouvert, les clients entrent dans le réseau, circulent à travers les stations pour recevoir des services puis quittent le réseau. Un réseau fermé par contre est caractérisé par un nombre constant de clients qui circulent indéfiniment dans le réseau.

L'état instantané d'un réseau est défini par le nombre de clients présents dans chaque station.

Ce modèle peut être analysé quantitativement par des techniques de résolution permettant de calculer de façon exacte ou approchée des indices de performance du réseau comme :

- le taux d'utilisation des serveurs
- le temps moyen d'attente d'un client dans une file

- le nombre moyen de clients dans une file
- le nombre moyen de clients dans le réseau
-....

2.1.2 PRINCIPE DE MODELISATION D'UN SYSTEME DE PRODUCTION PAR UN RESEAU DE FILES D'ATTENTE ET EVALUATION DE SES PERFORMANCES [BEL et DUBOIS 85]

Le modèle réseau de files d'attente permet de décrire de façon **simplifiée** un système de production. Chaque station du réseau peut représenter suivant le cas :

• un poste de travail (poste d'usinage, poste de chargement, poste de déchargement...) et l'encours de produits (des pièces par exemple) en attente de traitement (station monoserveur)

• plusieurs postes de travail identiques auxquels est associée une file d'attente (station multiserveurs)

• un système de transport. Par exemple un groupe de chariots filoguidés peut être représenté par une station à plusieurs serveurs.

La notion de client peut servir quant à elle à modéliser des produits à fabriquer, par exemple des pièces ou des palettes sur lesquelles sont bridées les pièces à usiner.

Si on ne distingue pas les produits entre eux, le réseau modélisant un système de production sera du type monoclasse sinon il sera du type multiclasse.

Les réseaux monoclasses permettent une vision agrégée du système et leur analyse ne fournit qu'une estimation grossière. Ceci est dû au fait que ces réseaux ne permettent pas de connaître les encours et le taux de production de chaque type de produit. Les réseaux multiclasses permettent, quant à eux, une description plus "réaliste" du système. Malheureusement leur emploi est moins souple car les méthodes de résolution ne s'appliquent qu'à des stations monoserveur. Les réseaux multiclasses sont essentiellement utilisés pour déterminer la distribution des palettes nécessaire pour assurer un taux de production donné ainsi que des pourcentages des différents produits.

Les réseaux de files d'attente ont été étudiés selon deux approches :

- l'approche stochastique
- et l'approche opérationnelle.

2.1.2.1 L'approche stochastique [Bel 85] [Dubois et Cavaille 82]

Cette approche suppose que le système se comporte comme un processus stochastique, que les temps de service à chaque station sont distribués selon des lois exponentielles, que les mouvement des pièces dans l'atelier sont décrits par une chaîne de Markov et que la capacité de chaque file d'attente est infinie.
Rappelons qu'une chaîne de Markov est une suite $(X_n)_{n\geq 0}$ de variables aléatoires à valeurs $\{i_n \in \{1, 2,..., q\}\}$ satisfaisant la propriété $Pr(X_{n+1} = i_{n+1} \mid X_0 = i_0, X_1 = i_1,....X_n = i_n) = Pr(X_{n+1} = i_{n+1} \mid X_n = i_n)$ ie. que seul l'état présent a une influence sur le futur. Les probabilités conditionnelles $Pr(X_{n+1} = j \mid X_n = i)$ sont des probabilités de transition, supposées indépendantes du temps n, et notées P_{ij} (un client quittant la station i a une probabilité p_{ij} de se rendre à la station j).

Sous ces hypothèses, le réseau de files modélisant le système a une solution en "forme de produit" pour les probabilités conjointes des longueurs de files [Pujolle 85], c'est-à-dire que la probabilité d'état $P(n_1, n_2,...n_m)$ (probabilité d'avoir n_1 clients à la station 1, n_2 clients à la station 2,..., n_m clients à la station m) peut s'exprimer comme le produit de facteurs, chacun étant relatif à une station du réseau.
A partir de cette formule de base, des algorithmes permettent de calculer les indices de performances (les taux d'activité des machines, les longueurs moyennes de files,...) valables pour une longue période de fonctionnement du système.

L'inconvénient majeur de cette approche réside dans le fait que les hypothèses de travail sont loin de correspondre aux conditions de fonctionnement des systèmes de production. Par exemple, les temps de service du système (usinage,transfert,...) sont souvent déterministes. De même, les zones de stockage sont de dimension limitée, et le fait de supposer des files d'attente infinies ne permet pas de prendre en compte les phénomènes de blocage dûs à la limitation des zones de stockage. Malgré ces restrictions certains travaux ont constaté empiriquement que cette approche permet une assez bonne évaluation du comportement des systèmes de production, du moins pour ce qui concerne le taux d'activité des machines.

2.1.2.2 L'approche opérationnelle [Dallery 84] [Buzen 80]

Contrairement à l'approche stochastique qui ne permet d'étudier des systèmes que sur une longue période de fonctionnement, l'approche opérationnelle permet d'étudier des systèmes sur une période de temps quelconque. De plus, elle s'applique à des systèmes de nature quelconque, c'est-à-dire des systèmes ayant un comportement aléatoire ou déterministe.
Elle permet d'obtenir la solution à "forme de produit" et d'en déduire les indices de performance mais avec des hypothèses plus réalistes :
- L'équilibre des flots : l'état final du système est le même que l'état initial, ce qui revient à dire que le flux d'entrée est pour chaque machine égal au flux de sortie.
- Evolution pas à pas : il n'y a jamais de départ simultané des pièces.
- Homogénéité des stations : le temps de service de chaque station ne dépend que du nombre de pièces à la station.
- Homogénéité des routages : les routages des pièces ne dépendent pas de l'état du système.
- Comportement homogène du système : les nombres minimums de pièces à chaque station sont nuls ($\underline{n}_m = 0$) et le temps moyen de service $S_m(n)$ à chaque station est indépendant du nombre de pièces n.

Le gros problème de cette approche est que en l'absence de comportement homogène, on a recours à la simulation pour déterminer les grandeurs $\underline{n}_m$ et $S_m(n)$. On se trouve alors devant le paradoxe de la nécessité d'une simulation pour calibrer une méthode analytique, laquelle a pour but d'éviter le recours à la simulation.

2.1.3 COMMENTAIRES

L'intérêt essentiel des méthodes analytiques basées sur la théorie des files d'attente réside dans le fait qu'elles sont faciles à mettre en œuvre, et qu'elles fournissent rapidement et à moindre coût des indices de performance du système. Elles sont donc particulièrement intéressantes pour tester plusieurs configurations. Toutefois dans les réseaux de files d'attente, les contraintes de synchronisation qui caractérisent le fonctionnement d'un système de production ne sont que très partiellement prises en compte. C'est pour pallier cette carence que des méthodes analytiques fondées sur les RdP ont été introduites.

2.2 LES METHODES ANALYTIQUES BASEES SUR LES RdP

Les RdP permettent de décrire et de valider très aisément les contraintes de synchronisation qui caractérisent le fonctionnement d'un système de production.
Ces réseaux permettent de modéliser de manière fine le système physique (ressources, activités, pièces, aires de stockage,...) et son fonctionnement. Les objets qui circulent dans l'atelier (pièces, palettes, chariots,...) sont représentés par des marques. L'état des entités physiques (machines, stocks,..) est représenté par le marquage des places.
Pour permettre l'évaluation des performances, Les RdP (a priori qualitatifs) ont été enrichis **d'une temporisation.** Ce temps représente en général la durée d'une activité (temps d'usinage d'une pièce, le temps de transport d'une pièce usinée jusqu'à un stock,...). L'intégration du temps à conduit à deux types d'extensions, les **RdP temporisés** caractérisés par un **temps fixe** et **les RdP stochastiques** dans lesquels le **temps est aléatoire.**

Dans ce qui suit nous allons présenter comment ces deux modèles peuvent être utilisés pour analyser les performances d'un système.

2.2.1 LES RdP TEMPORISES

2.2.1.1 Présentation intuitive du modèle

Deux principaux modèles ont été définis, les RdP temporels [Merlin 74] et les RdP temporisés. Dans le premier modèle, à chaque transition du réseau est associée un intervalle de temps [τ_1 τ_2]. τ_1 représente le délai minimal que la transition doit attendre, après sa validation, pour être franchie. τ_2 correspond au temps maximal pendant lequel la transition reste valide sans être franchie. Une fois ce temps écoulé, si la transition n'a pas été franchie alors son franchissement est déclenché immédiatement. Le franchissement de la transition reste toujours instantané.
Les RdP temporels s'avèrent très bien adaptés à la modélisation du mécanisme du **time-out** : si une transition n'est pas franchie avant le temps maximum qui lui est imparti, le time-out déclenche obligatoirement son franchissement.

Les RdP temporels ont surtout été utilisés pour modéliser les protocoles de

communication [Merlin 76] [Berthelot 83]. Leur emploi pour l'évaluation des performances reste inexistant.

Le second modèle a été étudié selon deux approches: la première concerne les réseaux dits **T-temporisés**, pour lesquels le temps est associé aux transitions.
La seconde approche les réseaux dits **P-temporisés**, où le temps est associé aux places.

les RdP T-temporisés [Ramchandani 73] [Chrétienne 84]

Ce modèle a été introduit par Ramchandani en associant une durée fixe τ à chaque transition t du réseau. Dans ce modèle les marques ont deux états **réservé** et **non réservé**. Seules les marques à l'état non réservé peuvent être utilisées pour valider une transition. Si une transition t est validée, son franchissement commence en rendant réservées les marques qui l'ont validée pendant un temps égal au temps attribué à cette transition. Après l'écoulement de ce temps, le franchissement de la transition se termine en enlevant les marques réservées des places d'entrée de la transition et en déposant dans les places de sortie de t des marques non réservées.
La figure 3.1 illustre l'évolution d'un réseau de Petri T-temporisé

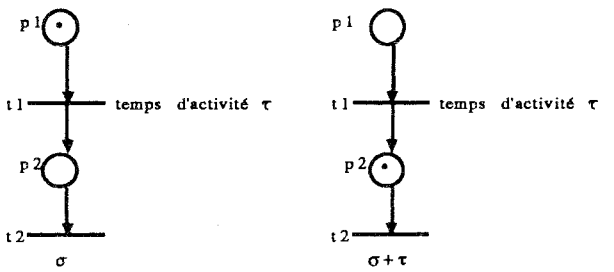


Figure 3.1

A l'instant σ, la marque se trouvant dans la place p1 est non réservée rendant la transition t1 franchissable. Le franchissement commence en rendant réservée la marque dans la place p1 (l'état réservé de la marque correspond à l'activation de l'activité associée à la transition t1). Le franchissement de transition t1 se termine à l'instant $\tau + \sigma$, en enlevant la marque réservée de la place p1 et en déposant une marque non réservée dans la place p2 (l'état non réservé représente la fin de l'activité).

Les RdP P-temporisés [Sifakis 79]

Dans ce modèle, une durée τ est associée à chaque place du réseau. Cette durée représente le temps pendant lequel une marque qui rentre dans une place reste dans l'état **indisponible**. Si une marque arrive à l'instant σ dans une place, elle ne peut quitter cette place avant l'instant $\tau + \sigma$. A l'issue de cette durée, la marque passe à l'état **disponible** et peut participer à l'évolution du réseau comme dans le cas d'un réseau autonome.
Le franchissement des transitions reste toujours instantané.
La figure 3.2 illustre l'évolution d'un réseau de Petri P-temporisé

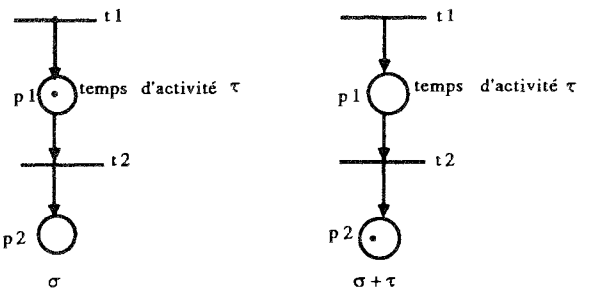


Figure 3.2

A l'instant σ, une marque est déposée dans la place p1 (franchissement de la transition t1). Cette marque est dans l'état indisponible du fait de la temporisation de la place p1 (l'état indisponible de la marque correspond à l'activation de l'activité de durée τ attribuée à la place p1). La marque devient disponible à l'instant $\tau + \sigma$, ce qui représente la fin de l'activité. La transition t2 est alors franchissable.
Il est montré dans [Chretienne 84] [Sifakis 79] [Valette 85] que les deux modèles sont équivalents. Ainsi un système de production pourra être modélisé par un RdP T-temporisé ou par un RdP P-temporisé selon que le spécifieur préfère représenter les activités du système par des places en associant le temps aux places, ou préfère représenter les activités par des transitions en utilisant une temporisation sur celles-ci.

2.2.1.2 Utilisation des RdP temporisés pour l'évaluation des performances

L'étude principale des RdP temporisés a été de déterminer des conditions nécessaires et/ou suffisantes pour que ces réseaux réalisent des fonctionnements particuliers au cours du temps, comme le fonctionnement

périodique ou "au plus tôt", dit aussi, fonctionnement à vitesse maximale. Ramchandani a donné des conditions nécessaires et suffisantes sur la structure du réseau et le marquage initial pour que le réseau ait un fonctionnement périodique. En fait, les principaux résultats établis par Ramchandani concernent les graphes d'événements temporisés. Ces derniers sont des réseaux particuliers caractérisés par l'absence de conflits. Ils ne possèdent aucune transition qui partage ses places d'entrée avec d'autres transitions. Par conséquent si à un instant donné plusieurs transitions sont franchissables elles peuvent être franchies dans n'importe quel ordre (ces réseaux ne permettent pas de modéliser des choix de tir de transitions qui dépendent de l'état du système).

Chrétienne dans [Chrétienne 84] étend les résultats obtenus par Ramchandani à des RdP temporisés quelconques, et pour d'autres types de fonctionnements : Le fonctionnement fini et "au plus tôt". Ce dernier fonctionnement est caractérisé par le fait qu'une transition est franchie dès qu'elle est franchissable, c'est-à-dire qu'aucun retard n'est admis dans le fonctionnement. Pour ce type de fonctionnement, Chretienne a proposé un algorithme de calcul des **dates de franchissements** des transitions. Il a proposé également un algorithme de calcul des **fréquences de franchissement** des transitions en cas de fonctionnement permanent du réseau de Petri temporisé.

Sifakis a donné dans [Sifakis 77] des conditions pour que chaque transition d'un RdP P-temporisé puisse être franchie périodiquement. Il en déduit des fréquences de franchissement des transitions à partir d'inégalités liant les matrices d'incidence du réseau et des temps affectés aux places, le vecteur des fréquences et le marquage initial. Il a aussi défini le fonctionnement à vitesse maximale et une méthode d'évaluation des fréquences de franchissement des transitions pour ce type de fonctionnement.

2.2.1.3 Commentaires

En dépit de l'intérêt que présente les RdP temporisés pour la présentation formelle de la synchronisation dans les systèmes de production, ils se révèlent insuffisants pour évaluer analytiquement ces systèmes sur le plan des performances. Les seules résultats établis concernent des conditions à vérifier pour qu'un fonctionnement fixé à l'avance soit possible (exemple le fonctionnement périodique ou "au plus tôt"). Ainsi, les RdP temporisés sont très peu exploitables analytiquement et la simulation est actuellement la seule voie

qui permette l'obtention de résultats numériques pour ces réseaux.

2.2.2 LES RdP STOCHASTIQUES [Florin et Natkin 85]

Les RdP stochastiques constituent une nouvelle approche de modélisation et d'évaluation des performances des systèmes de production. Ces réseaux sont des RdP auxquels les notions de temps et de probabilités de résolution de conflit sont ajoutées. A chaque transition représentant une activité du système est associée une distribution de probabilité qui détermine son instant de franchissement quand elle est franchissable. Les distributions de probabilités considérées par les RdP stochastiques sont les distributions exponentielles, elles sont les seules qui permettent l'étude numérique de ces réseaux. Des transitions instantanées peuvent être utilisées dans ces réseaux pour modéliser des contraintes de synchronisation.
Le principe de l'évaluation quantitative à l'aide de ce type de réseaux repose sur l'analyse Markovienne. Il a été démontré que les RdP stochastiques sont isomorphes aux chaînes de Markov. La démarche utilisée pour l'évaluation des indices de performances consiste alors à engendrer le graphe des marquages accessibles du réseau comme pour les réseaux autonomes (cf. chapitre 1), puis à valuer ce graphe par des taux de franchissement exponentiels lorsqu'il s'agit de transition temporisées et par des taux infinis lorsqu'il s'agit de transitions instantanées. Ce graphe est ensuite réduit en supprimant les états instantanés. Le graphe réduit est isomorphe à une chaîne de Markov. A partir du graphe réduit on construit la matrice stochastique A des taux de transitions entre états. Cette matrice est utilisée pour calculer les probabilités des différents états en régime permanent en résolvant le système d'équations linéaires suivant :

$$\mathbb{P}^* . A = 0$$

avec la norme du vecteur $\mathbb{P}^*$ égale à 1.

A partir de ces probabilités d'états on en déduit par des calculs matriciels simples, les indices de performances désirés : les fréquences moyennes de franchissement des transitions du réseau (débit du système modélisé), les temps moyens de séjour des marques dans les places (temps d'attente) et les nombres moyens de marques dans les places (capacités moyennes des stocks).

2.2.2.1 Commentaires

Les RdP stochastiques constituent un outil puissant de modélisation et d'évaluation des performances des systèmes. Par rapport aux réseaux de files d'attente cet outil a l'avantage de permettre la représentation de la synchronisation. Cependant, comme la plupart des méthodes analytiques, ces réseaux ne s'intéressent qu'au fonctionnement en régime permanent. De ce fait, ils ne permettent qu'une évaluation grossière des performances du système. De plus ils se restreignent au cas markovien (temps de franchissement exponentiel), qui n'est pas toujours valide pour les systèmes de production. Le plus souvent, ces systèmes sont déterministes, notamment quant aux temps de traitement des produits.

2.3 CONCLUSION

Les méthodes analytiques ont le mérite d'être faciles à mettre en œuvre et de donner rapidement et à moindre coût une évaluation grossière du fonctionnement d'un système pour ce qui concerne la capacité des aires de stockage, le taux d'activité des machines et du système de transport, les temps d'attente. Elles sont donc souvent utilisées pour le prédimensionnement d'un système. Toutefois, ces méthodes sont en général trop agrégées et deviennent vite insuffisantes à une étape avancée de la conception.
L'analyse détaillée d'un système nécessite l'utilisation d'outils plus puissants pouvant permettre l'étude des problèmes de blocage, des pannes et des règles de gestion et de pilotage. La simulation est sans contester l'outil qui permet de résoudre au mieux ce type de problèmes. Elle conduit le concepteur à modéliser son système par un ensemble de ressources (passives et actives), de files d'attente et de règles de fonctionnement, et à évaluer les performances de ce système en reproduisant pas à pas, événement par événement, son fonctionnement dans le temps.

3 LA SIMULATION DISCRETE

La simulation discrète ou par événements discrets est l'outil le plus couramment utilisé pour l'évaluation des performances. Elle permet de décrire un système de production en prenant en compte la plupart de ces caractéristiques essentielles, puis d'en déduire avec le maximum de précision les

indices moyens de performances comme le taux de productivité, le taux d'utilisation, la longueur moyenne des files d'attente... pour un meilleur dimensionnement.

Cette simulation peut être réalisée selon trois approches [Bel et Dubois 85]:

Approche par événements

Ce type d'approche conduit le concepteur à répertorier tous les événements (changements d'état du système) pouvant se produire au cours de la vie du système, puis à associer à chaque type d'événements une procédure qui a pour rôle d'effectuer les traitements nécessaires lors de l'occurrence d'un événement et d'engendrer éventuellement des occurrences qui se produisent dans le futur. Ainsi, par exemple, l'événement arrivée à un poste d'une pièce pour usinage peut provoquer selon l'état du poste de travail, soit son usinage immédiat, si le poste est libre (un événement "fin d'usinage" qui se produira dans le futur est alors engendré), soit l'introduction de la pièce dans la file d'attente du poste de travail si celui ci est occupé.

Le principe de la simulation du comportement dynamique du système consiste alors à activer dans l'ordre chronologique les traitements associés aux événements classés dans un échéancier. L'avance du temps se fait par saut d'une date d'événement à la suivante.

Approche par activités

Cette approche, conduit le concepteur à identifier les activités caractéristiques du système à étudier, ainsi que les conditions de début et de fin de ces activités. La simulation du fonctionnement dynamique d'un système s'appuie alors sur une horloge qui progresse à pas fixe. Chaque progression de l'horloge déclenche un examen des conditions de début et de fin des différentes activités. Si l'une de ces conditions est satisfaite l'état du système est modifié en conséquence. Sinon l'état du système reste inchangé.

Cette approche a l'inconvénient de nécessiter un temps de calcul trop important (à chaque progression de l'horloge, il faut évaluer toutes les conditions associées aux différentes activités). Par contre, elle est intéressante lorsqu'il s'agit de systèmes dont les durées d'activités ne sont pas connues a priori.

Approche par processus

Dans ce type d'approche, le concepteur identifie les processus en jeu et leurs interactions, un processus est défini comme une succession d'événements

ou de façon duale comme une succession d'activités. Cette approche est largement utilisée pour l'étude des systèmes de production. Elle a l'avantage de conduire à une structure du modèle proche de celle du système réel.

Les outils de simulation sont généralement axés sur l'une et/ou l'autre de ces approches. Ils peuvent être classés en deux grandes familles :

- Les progiciels généraux de simulation.
- Les simulateurs basés sur les RdP.

Dans ce qui suit, nous allons présenter dans les grandes lignes ces deux familles de simulateurs.

3.1 LES PROGICIELS GENERAUX DE SIMULATION (SLAM, GPSS, QNAP, PAWS,....)

Les progiciels généraux de simulation permettent de modéliser facilement le fonctionnement d'un système de production en confondant partie commande et partie opérative. Ensuite de déterminer au cours de la simulation du fonctionnement du système des statistiques sur les files d'attente longueurs moyennes, maximales et minimales, les durées d'attente, les débits et taux d'occupation. Pour cela, ces progiciels reposent sur :

- des primitives standards qui permettent de décrire le fonctionnement d'un système,
- des règles de gestion des files d'attente,
- et des lois de distributions probabilistes (exponentielle, uniforme, constante, Erlang...) des intervalles d'arrivée des événements, des temps de service et des routages.

qui permettent de faciliter la mise au point de la simulation et de décharger le concepteur d'un gros effort de programmation.

L'inconvénient majeur de ces progiciels réside dans le fait qu'ils ne permettent pas de décrire aisément et d'analyser formellement les contraintes de synchronisation qui caractérisent le fonctionnement des systèmes de production. Une solution consiste à utiliser des modèles formels qui s'y prêtent mieux à l'expression et à l'analyse de la synchronisation (tels les RdP), ensuite de traduire la description par RdP du système dans le formalisme propre au progiciel retenu. Cette traduction peut entraîner des erreurs et ralentir la

progression correcte de la conception. Le développement de simulateurs fondés sur le modèle RdP est donc souhaitable.

3.2 LES SIMULATEURS DE RDP

Ces simulateurs consistent à décrire par un RdP le fonctionnement d'un système en confondant partie commande et partie opérative, puis à évaluer le comportement du système pour définir des dimensionnements.
L'avantage d'utiliser des simulateurs fondés sur les RdP est double:
- d'une part, le modèle RdP permet de représenter aisément les contraintes de synchronisation et de montrer que certaines propriétés telles que l'absence de blocage ou de configurations non désirées (exemple la collision de convoyeurs) sont vérifiées.
- D'autre part, le modèle RdP peut être utilisé pour la description du système de commande (cf. chapitre 2), ce qui permet dans une démarche de conception de conserver le même modèle.
Nous présentons quelques simulateurs de RdP destinés à l'évaluation des performances des systèmes de production.

3.2.1 PSI : a Petri net based SImulator [Alanche 84]

PSI est un simulateur de RdP développé au LAAS dans le cadre d'un contrat avec la Régie Renault. Il est basé sur les concepts suivants:

- les places du réseau représentent les activités du système(activité d'usinage, activité de chargement...) ainsi que les états des stocks,

- les transitions représentent des événements,

- les marques modélisent des entités physiques (les pièces, les chariots,...),

- le temps (d'activité) est associé aux places du réseau qui modélisent des activités. Ce temps peut être constant ou aléatoirement distribué selon une loi uniforme,

- des conditions et des actions peuvent être associées aux transitions du réseau.

Les conditions sont utilisées pour exprimer **des règles de décision** dans le choix de franchissement des transitions conflictuelles.

Au cours de la simulation du fonctionnement du RdP décrivant le système, des mesures sont effectuées pour fournir des statistiques sur les places et les transitions du réseau.

EXEMPLE

Nous précisons les principes de modélisation à travers cet exemple tiré de [Valette 84]. Il concerne un système d'inspection par échantillonnage constitué d'un poste. Les pièces circulant sur un tapis roulant sont saisies et chargées sur le poste lorsque ce dernier est libre et qu'elles passent à proximité (capteur CAP). Le déchargement ne se fait que lorsque la section CAPE/CAPS est vide (cf. figure 3.4).
La figure 3.5 décrit le fonctionnement du système à l'aide d'un RdP :
la place PCP représente la condition "poste libre". Si la place PCT contient 4 marques, cela signifie que la section CAPE/CAPS est vide.
La place PPOSTE exprime le fait que, si une pièce n'est pas chargée sur le poste, elle doit continuer sur le tapis. De cette manière le conflit entre les transitions FCAP et ALD est résolu.
La transition ARP représente l'événement "arrivée d'une pièce sur le tapis". Ainsi, chaque fois qu'une pièce arrive la transition ARP est franchie.

Les temps d'acheminement d'une pièce sur les sections du tapis sont associés aux places SEC1, SEC2 et SEC3. Le temps d'alimentation du poste est associé à la place ALIM et le temps d'inspection d'une pièce à la place POSTE (temps aléatoire et distribué selon une loi uniforme).

L'objet de la simulation de ce système est de vérifier les points suivants:
- il n'y a aucune attente dans la place PBUF. Dans le cas contraire, il faudrait augmenter le marquage initial de PCT et les poids des arcs correspondants,
- il n'y a aucune attente dans la place CAP, sinon cela voudrait dire qu'une nouvelle pièce est arrivée avant la fin de l'alimentation du poste et donc qu'il y a eu collision.

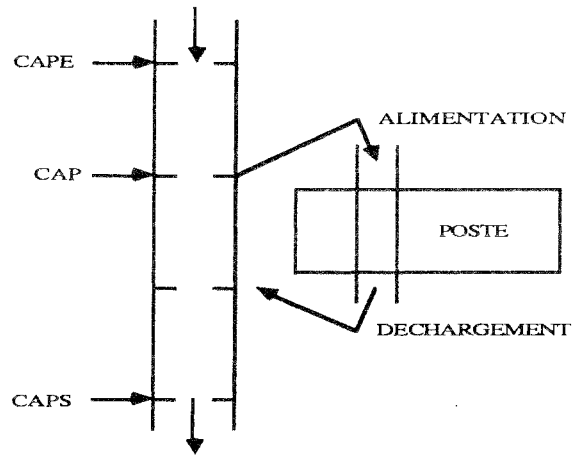


Figure 3.4

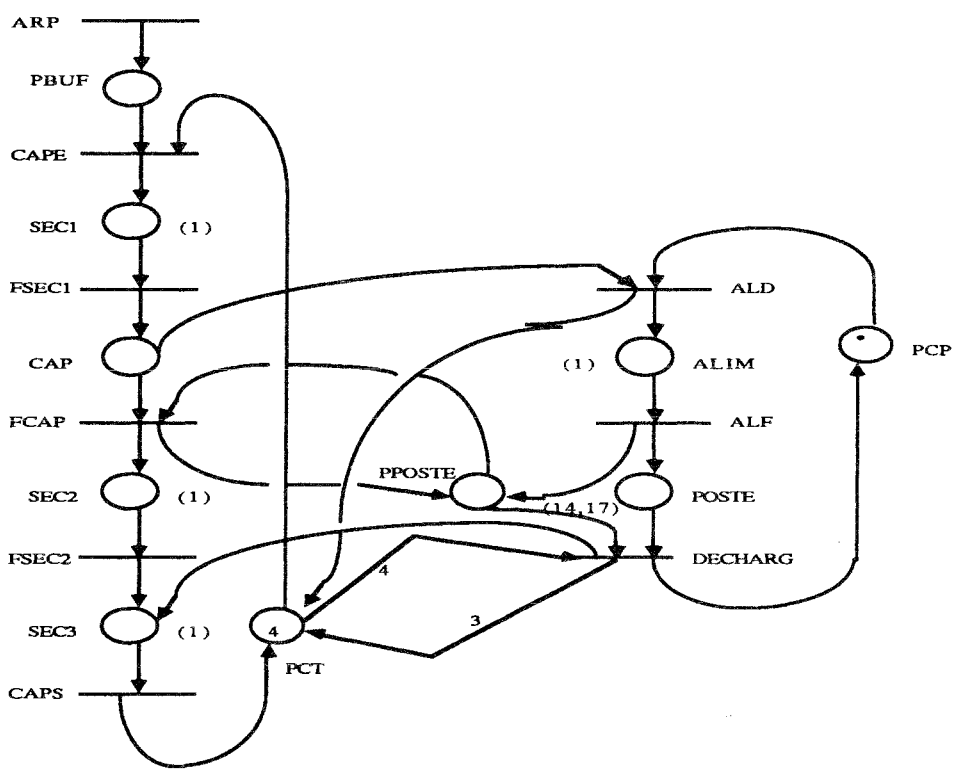


Figure 3.5

3.2.1.1 Commentaires

Du fait de la nature des RdP utilisés, PSI se révèle très limité pour l'évaluation des performances des systèmes. En effet, les marques qui modélisent les objets qui circulent dans le système sont banalisées (ces marques ne portent pas d'identité et ne sont pas distinguables). Or pour pouvoir évaluer finement un système il est nécessaire d'associer aux marques des attributs afin de les

individualiser, et pouvoir ainsi rendre compte de la circulation dans le système d'objets diversifiés.
Les RdP colorés permettent l'affectation d'attributs aux marques du réseau et de prendre en compte des flux multiples.

3.2.2 SIMULATEURS A EVENEMENTS DISCRETS BASES SUR L'UTILISATION DES RdP COLORES : [Alla 86], SEDRIC [Valette 85]

Les RdP colorés, dûs à Jensen [Jensen 81], permettent de modéliser de façon condensée des systèmes de grande taille en introduisant un élément d'information supplémentaire qui est la couleur.
Ainsi, un RdP coloré se caractérise par:
• l'association de couleurs aux marques,
• l'association d'ensembles de couleurs aux places et aux transitions,
• et l'étiquetage des arcs qui joignent places et transitions par des fonctions indiquant les marques (couleurs) à considérer.

Nous allons introduire ce modèle d'une façon intuitive sur un exemple, le lecteur désirant plus de détails pourra consulter [Jensen 81] [Brams 83].

EXEMPLE

Considérons l'exemple de la file FIFO (First In First Out), cet exemple est doublement intéressant, car d'une part la file FIFO est un modèle qu'on rencontre souvent dans les systèmes de production (modélisation d'un stock), et que d'autre part, cet exemple convient parfaitement à une description par des réseaux de Petri colorés.

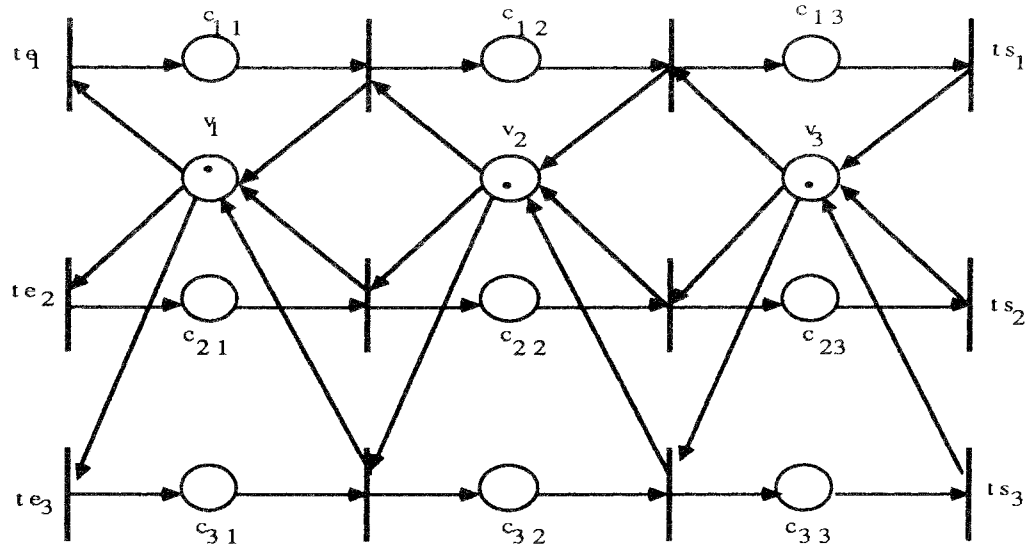


Figure 3.6

La figure 3.6 donne le RdP généralisé de la file FIFO de capacité 3 pouvant contenir des produits de trois types différents (p_1, p_2, p_3). Dans ce réseau, les transitions te_1, te_2, te_3 représentent respectivement l'entrée des produits de type p_1, p_2, p_3, et les transitions ts_1, ts_2 et ts_3 la sortie de ces produits.
Les places c_{1i}, c_{2i}, c_{3i} et v_i ($i \in [1, 3]$) représentent les quatre états possibles de la position i de la file : occupée par un produit de type p_1 si la place c_{1i} est marquée, par un produit de type p_2 si la place c_{2i} est marquée, par un produit de type p_3 si la place c_{3i} est marquée et enfin, vide lorsque la place v_i est marquée.
Si nous essayons d'allonger la file de n positions et de p types de produits, le réseau représentant la file devient illisible. Il faut donc condenser l'information. Cette condensation s'effectue grâce au RdP colorés.

La figure 3.7 représente le RdP coloré modélisant la file FIFO [Alla 87]. Ce réseau est obtenu par un pliage du RdP généralisé de la figure 3.6 suivant des symétries entre des évolutions parallèles et équivalentes d'éléments différents. Ainsi en fusionnant toutes les places v_i $i \in [1, 3]$, toutes les places c_{ij} $j \in [1, 3]$ et $i \in [1, 3]$, on obtient les deux places V et C.
La transition te est obtenue en fusionnant les trois transitions d'entrée du réseau généralisé (te_1, te_2, te_3). La transition ts résulte de la fusion des transitions ts_1, ts_2 et ts_3, et la transition t de la fusion de toutes les autres transitions. La transition te permet l'entrée des produits dans la file, ts leur sortie de la file et la transition t permet à un produit p_i de progresser dans la file d'une position à la suivante. La fusion des arcs conduit à ce que la valuation du

nouvel arc soit représentée par une fonction qui définira par son image les couleurs à supprimer ou à déposer, respectivement, dans les places d'entrée et de sortie.

Les ensembles de couleurs des marques sont :

$\{p_i / i \in [1, 3]\}$ ensemble des produits;

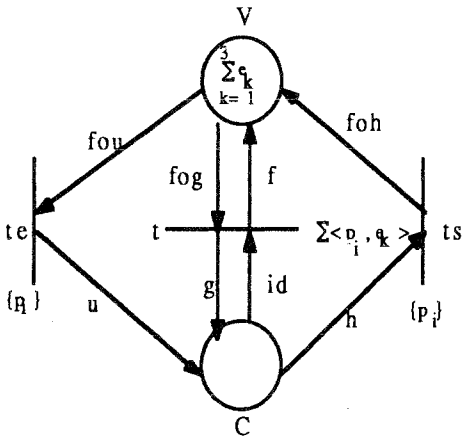
$\{e_k / k \in [1, 3]\}$ ensemble des positions dans la file;

$\{<p_i, e_k> / k \in [1, 3], i \in [1, 3]\}$ ensemble des positions dans la file qui sont occupées par des produits. Par exemple une marque de couleur $<p_1, e_1>$ signifie qu'il y a un produit de type p_1 dans la première position de la file.

Les fonctions u et fou permettent aux différents produits d'entrer dans la file, la fonction composée fou permet de s'assurer que la position e1 est libre, et la fonction u introduit un produit pi dans la position e1 de la file.

La fonction h permet d'évacuer un produit pi de la file, elle extrait le produit pi de la dernière position de la file. La fonction foh sert à libérer la dernière position de la file.

Les fonction fog, f, g et identité gèrent la progression des produits dans la file.



$u : \{p_i\} \rightarrow \{<p_i, e_1>\} / u(p_i) = \{<p_i, e_1>\}$

$h : \{p_i\} \rightarrow \{<p_i, e_3>\} / h(p_i) = <p_i, e_3>$

$f : \{<p_i, e_k>\} \rightarrow \{e_k\} / f(<p_i, e_k>) = e_k$

$g : \{<p_i, e_k>\} \rightarrow \{<p_i, e_k>\} / g(<p_i, e_k>) = <p_i, e_{k+1}>$ $k \in [1,2]$

id : fonction identité

le marquage initial $M_0(V) = \sum e_k$; $M_0(c) = 0$

Figure 3.7

Fonctionnement du réseau de la figure 3.7

Lorsqu'un produit p_i entre dans la file, il provoque le franchissement de la transition te (te est validée par rapport à la couleur p_i) si la première position de la file est vide, c'est-à-dire que la place V contient une marque de couleur e_1 (ie. $fou(p_i) = e_1$).

Le franchissement de la transition te retire une marque de couleur e1 de la place V, et dépose une marque de couleur $u(p_i) = <p_i, e_1>$ dans la place C, ce qui signifie que la position 1 de la file est occupée par le produit p_i.

Le franchissement de la transition t permet la progression du produit p_i dans la file. Supposons que la place C contient une marque de couleur $<p_i, e_k>$, la transition t sera franchie selon cette couleur à condition que la place V contienne une marque de couleur $fog(p_i,e_k) = e_{k+1}$, ce qui signifie que la position e_{k+1} est vide. Le franchissement de t dépose une marque de couleur $g(p_i,e_k) = <p_i,e_{k+1}>$ dans la place C (ie. la position e_{k+1} est occupée par le produit p_i) et une marque de couleur $f(p_i, e_k) = e_k$ dans la place V (ie. la position e_k dans la file est libérée).

Lorsque le produit pi se trouve dans la dernière position de la file, il peut sortir de la file par le franchissement de la transition ts. Cela consiste à enlever une marque de couleur $h(p_i) = <p_i, e_3>$ de la place C et à déposer une marque de couleur $foh(p_i) = e_3$ dans la place V (ie. que la dernière position e_3 est libérée).

En se basant sur les RdP colorés, ces simulateurs permettent de décrire de façon concise le fonctionnement d'un système de production en confondant partie commande et partie opérative et de prendre en compte des flux multiples. Le modèle est ensuite temporisé, des durées de séjour des marques sont associées aux places du réseau. La simulation du réseau permet de fournir des statistiques plus fines et plus pertinentes.

4 CONCLUSION

Au vu de ce qui précède, nous avons choisi de nous appuyer sur la simulation pour aborder le problème de l'évaluation des performances. L'approche adoptée consiste à **compléter** la description du système de commande par la **description du flux d'activité**, qui permet de décrire à l'aide d'un modèle proche des RdP colorés les contraintes imposées par l'organisation du système de transport, des stocks intermédiaires et des machines, en précisant les mouvements logiquement et physiquement possibles. Ensuite d'en déduire au cours de la simulation conjointe du système de commande et de l'environnement commandé des statistiques pertinentes sur les files d'attente (cf. chapitre 5).

Cette approche a l'avantage d'une part, de faciliter l'évaluation de différents scénarios de pilotage. D'autre part, de permettre au concepteur de s'assurer de la cohérence entre le fonctionnement du système qu'il évalue et la spécification du système de commande.

CONCLUSION

Nous avons abordé dans cette première partie les principales approches préconisées pour la validation et l'évaluation des performances des systèmes automatisés de production.
Dans le chapitre 2, nous avons présenté les différentes approches de validation des systèmes de commande fondés sur une description par les RdP. Deux types d'approches ont été exposées : les approches statiques et les approches dynamiques. Dans le chapitre 3 nous avons présenté les approches d'évaluation des performances des systèmes de production.

De cette étude il ressort :

1°) Que certaines approches n'ont pas été motivées principalement par la conception des systèmes de commande des systèmes automatisés, mais par d'autres disciplines, et que leur adaptation à ces systèmes n'a pas toujours permis de prendre en comptes toutes les spécificités de ces systèmes. On peut citer, à titre d'exemple, les réseaux de files d'attente stochastiques qui ont été étudiés pour les systèmes informatiques et dont les hypothèses de travail sont injustifiables pour les systèmes de production.

2°) Que ces approches se limitent à un seul aspect : la validation ou l'évaluation des performances, obligeant ainsi le concepteur à gérer deux spécifications, l'une décrivant le fonctionnement du système de production en confondant partie commande et environnement commandé et l'autre décrivant le comportement du système de commande. Du fait de la redondance qui peut exister entre ces deux spécifications, cette gestion pose des problèmes de vérification de leur cohérence et du maintien de cette cohérence lors de modifications.

L'objectif de notre travail est d'apporter un élément de réponse à ces problèmes en proposant **un outil d'aide au prototypage** qui permet d'aborder **la validation et l'évaluation des performances** des systèmes de production automatisée.

La description de cet outil fera l'objet de la seconde partie de ce document.

DEUXIEME

PARTIE

INTRODUCTION

Cette seconde partie est consacrée à la présentation de l'outil d'aide au prototypage rapide des automatismes industriels: le logiciel PETRI-S. Cet outil s'appuie sur la simulation qui est considérée comme étant l'une des techniques de base du prototypage [Johnson 83].
La simulation d'un système s'effectue généralement en deux étapes:
- l'étape de modélisation qui consiste à construire un modèle du comportement dynamique du système,
- et l'étape de simulation qui utilise ce modèle pour étudier le comportement du système au cours du temps.

PETRI-S est donc décrit en deux chapitres :
le premier chapitre présente les différents modèles de description d'un système. Les modèles utilisés sont les mêmes que ceux de la méthode de conception proposée dans [Tankoano 88]. Ceci permet l'utilisation de cet outil tout au long du processus de conception et non pas seulement au cours des dernières étapes.

Le second chapitre décrit le logiciel de simulation.

CHAPITRE 4

DESCRIPTION DES MODELES DE PETRI-S

1 INTRODUCTION

Dans ce chapitre, nous abordons **la modélisation** des systèmes de production. L'approche de modélisation que nous préconisons s'appuie sur trois vues différentes mais complémentaires de la dynamique d'un système, à savoir:

- la description du comportement du système de commande,

- la modélisation de l'environnement commandé,

- et la modélisation du flux d'activité.

La spécification du comportement du système de commande définit les stratégies de pilotage du système en précisant les règles d'enchaînement des décisions à prendre pour coordonner le déroulement des différentes activités.
La description du comportement de l'environnement précise, en fonction des caractéristiques des composantes mécaniques, les lois qui régissent l'évolution des grandeurs physiques. La description du flux d'activité permet de fixer les contraintes imposées par l'organisation du réseau de transport, des stocks intermédiaires et des machines, en précisant les mouvements physiquement et logiquement possibles.

Dans une optique de prototypage rapide, cette approche offre plusieurs avantages:

1) Elle facilite la réutilisation. Dans les premières phases du processus de conception, c'est la spécification du comportement du système de commande utilisée pour la simulation que le concepteur affine dans les étapes suivantes pour obtenir une solution plus élaborée. Dans la dernière phase du processus de conception, cette spécification peut directement servir pour l'exploitation.

2) Elle facilite l'expérimentation. Les trois descriptions n'étant pas redondantes, le spécifieur peut changer l'une d'entre elles sans remettre en cause les deux autres. Ainsi, étant donné un ensemble de lois caractéristiques de l'évolution des grandeurs physiques et une organisation du système, le spécifieur peut évaluer différentes stratégies de pilotage, ou encore pour une stratégie de pilotage et une organisation, évaluer différentes hypothèses sur les

lois d'évolution des grandeurs physiques.

3) Enfin, elle permet de vérifier la cohérence. Par rapport à la spécification du comportement du système de commande, la description du flux d'activité offre une vue plus clarifiée des contraintes du fonctionnement que le système de commande doit assurer. Au cours d'une simulation, il est aisé de vérifier que l'évolution de la commande respecte bien ces contraintes. Inversement, le spécifieur peut s'assurer que le fonctionnement de l'atelier qu'il évalue respecte bien les contraintes exprimées dans la spécification du système de commande.

Nous allons décrire ces différents modèles. Dans la section 2, nous présentons le modèle de description du comportement du système de commande. Le modèle que nous avons retenu sont les RdP Interprétés (notés par RdPI).

Dans la section 3, nous proposons un modèle original pour décrire de façon "réaliste" l'évolution des grandeurs physiques de l'environnement avec lequel le système de commande interagit.

Nous terminons ce chapitre par la présentation du modèle de description du flux d'activité. Le modèle retenu est un réseau de files permettant de représenter les différentes files d'attente qui se créent autour des postes de travail, les flux possibles entre ces files d'attente et les événements qui déclenchent ces flux.

2 MODELE DE DESCRIPTION DU COMPORTEMENT DU SYSTEME DE COMMANDE

Comme nous l'avons déjà souligné les systèmes de commande sont des systèmes particuliers qui se caractérisent par:

- l'influence du temps dans leur définition,
- leur parallélisme,
- et leur interaction avec l'environnement.

Un modèle de description de tels systèmes doit donc premettre d'exprimer le parallélisme, les contraintes de synchronisation, les containtes temporelles, les interactions avec l'environnement et offrir des possibilités de validation.

Pour cela, notre choix s'est porté sur les RdPI. Ce modèle permet de décrire graphiquement le parallélisme et la synchronisation, de décrire séparément le système de commande, le système commandé et leurs interactions puis d'effectuer de façon automatique certaines vérifications.

Dans ce qui suit, nous allons présenter le modèle RdPI auquel nous avons apporté certains aménagements afin d'étendre son utilisation à la spécification des systèmes de commande répartis. A cet effet, des règles de décomposition des RdPI ont été définies dans [Tankoano 88]. Ces règles permettent le passage automatique de la spécification externe du système à répartir à l'aide de RdPI à la spécification interne décrivant le comportement d'un réseau de modules communicants via des ports.

2.1 PRESENTATION DU MODELE RdPI

La description du comportement d'un système s'effectue à l'aide de RdPI, en synchronisant l'évolution d'un RdP généralisé aux interactions que le système doit avoir avec son environnement. Les interactions du système avec l'environnement externe sont traduites par des réceptivités associées aux transitions et par des opérations associées aux places. Les réceptivités sont des prédicats définis par les comptes rendus provenant de l'environnement et les opérations représentent les actions de commande destinées pour l'environnement.

2.1.1 DEFINITION

Un RdPI R_I est défini par la donnée :

- d'un RdP R = <P, T, Pré, Post>;
- d'un triplet <V, OP, PR> caractérisant les interactions du système avec son environnement tel que :

  -V = E ∪ S ∪ C ∪ H désigne l'union de quatre ensembles disjoints de variables avec

  - E l'ensemble des **variables d'entrée** du système dont les valeurs successives sont engendrées par l'environnement
  - S l'ensemble des **variables de sortie** du système dont les valeurs successives déterminent la commande appliquée à l'environnement
  - C l'ensemble des **variables internes** au système, qui peuvent être des

compteurs, des variables de synchronisation, ...

- et H un ensemble de **variables de temporisation** servant à initialiser la génération de top d'horloge au bout d'intervalles de temps prédéfinis

-OP = $\{op_1, op_2, ..., op_n\}$ désigne un ensemble d'opérations où chaque opération op_i est une application de $E \cup C \cup H \rightarrow S \cup C$

-PR = $\{pr_1, pr_2, ..., pr_m\}$ désigne un ensemble de prédicats définis sur $E \cup C \cup H$

-d'une application $\varphi : P \rightarrow OP$, qui associe à chaque place p une **opération**

-et d'une application $\Psi : T \rightarrow PR$, qui associe à chaque transition t un prédicat ou **réceptivité**

2.1.2 REPRESENTATION GRAPHIQUE

On représente un RdPI R_I par la représentation graphique usuelle du RdP R associé en spécifiant à côté de chaque place p_i l'opération $\varphi(p_i)$ et à côté de chaque transition t_{ij} la réceptivité $\Psi(t_{ij})$ (cf. figure 4.1).

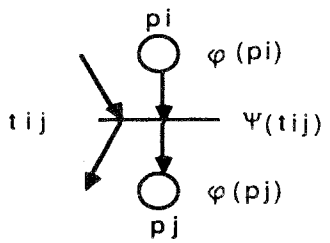


Figure 4.1 : Représentation graphique d'un RdPI

2.1.3 ELEMENTS DU MODELE

Nous allons préciser la définition d'un RdPI par la description des différents éléments du modèle.

2.1.3.1 La notion de réceptivité

La réceptivité associée aux transitions est composée d'un **événement**

externe e et d'une **condition c**. Elle est notée sous la forme <e.c>.

-L'événement: traduit un changement d'état de l'environnement.

Le concept d'événement est très important pour la conception des systèmes de commande [Thomesse 80] [Ladet 82] [Benmaïza 84]. Il permet de déterminer les instants de déclenchement des opérations de commande, exemple sur l'événement arrivée d'une pièce à un poste de travail déclencher l'ordre d'usinage.

Nous exprimons les changements d'état à l'aide de deux types d'événements notés respectivement par :

↑exp désigne le passage de "faux" à "vrai" de l'expression logique "exp" (front montant de "exp") et correspond à l'instant où l'environnement entre dans l'état caractérisé par l'expression "exp".

↓exp désigne le passage de "vrai" à "faux" de l'expression logique "exp"(front descendant de "exp") et correspond à l'instant où l'environnement sort de l'état caractérisé par "exp".

Exemple l'événement "dépassement de la température t d'un seuil donné s " est défini par ↑(t > s).

-La condition : caractérise un état particulier de l'environnement.

2.1.3.2 Les opérations

Les opérations associées aux places peuvent être des actions élémentaires (commandes à appliquer au procédé) ou des actions complexes selon le niveau d'abstraction où l'on veut se situer.

Les actions peuvent être de deux types : **impulsionnelles** ou **continues**. Les actions impulsionnelles sont exécutées une seule fois, quand les places auxquelles elles sont associées sont marquées. Les actions continues sont exécutées en permanence tant que les places où elles sont spécifiées sont marquées.

Pour distinguer les actions continues des actions impulsionnelles, nous suffixons les actions continues d'une étoile.

2.1.3.3 Prise en compte du temps

La prise en compte des contraintes temporelles peut s'exprimer à l'aide de

variables de temporisation qui sont testées au niveau des transitions. Les temporisations sont activées ou désactivées au niveau des places grâce à deux opérations prédéfinies :

Armertemporisation(temp,n) : force la variable de temporisation temp à vrai au bout de n unités de temps

Désarmertemporisation(temp) : désactive l'effet de armertemporisation(temp,n)

Exemple

Dans la figure 4.2, nous exprimons le fait " après avoir donné l'ordre de déplacement d'un chariot, si l'événement fincourse ne survient pas avant 15 unités de temps, on lance l'opération opd qui traite le mauvais fonctionnement du chariot".

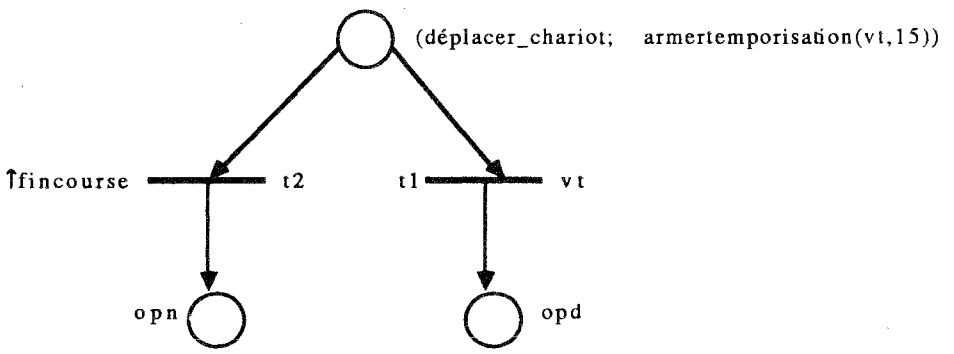


Figure 4.2

2.1.4 LES REGLES D'EVOLUTION D'UN RdPI

Les évolutions des marquages d'un RdPI modélisant le système de commande sont synchronisées aux occurrences de l'ensemble E des événements engendrés par l'environnement. On définit les règles d'évolution d'un RdPI par rapport à l'ensemble de séquences de parties de E, chaque élément X de la séquence modélisant l'occurrence simultanée des événements qui le composent.

REGLE 1 [Moalla 85]

Une transition t à laquelle est associée un événement e et une condition c est franchissable pour un marquage M accessible depuis M_0 et le sous-ensemble d'événements X ssi :

• t est validée pour le marquage M, c'est-à-dire que chaque place p en entrée de t contient un nombre de marque supérieur ou égal à la valeur de l'arc liant la

place p à la transition.
- la condition c est vérifiée
- et l'événement e $\in$ X

La transition t est dite aussi réceptive au sous-ensemble d'événements X.
Le franchissement de la transition provoque l'évolution du marquage et l'exécution des opérations associées aux places apparaissant en sortie de la transition.

Toutefois, pour un marquage M, le sous-ensemble d'événements X peut rendre franchissables plusieurs transitions et l'ordre de franchissement de ces transitions n'est pas toujours indifférent : le franchissement de certaines transitions peut en invalider d'autres et certaines transitions peuvent être franchies plusieurs fois. C'est pourquoi des règles précises concernant le choix de transitions réceptives à franchir ont été définies. Ces règles permettent d'éviter toute ambiguïté d'interprétation des intentions du spécifieur.

REGLE 2 [Moalla 85]
Si pour un marquage M de $<R_I, M_0>$, $T_{X,M}$ = { t1, t2, ..., tl} désigne l'ensemble des transitions réceptives au sous-ensemble d'événements X, on appelle **séquence de simulation complète** par rapport à X pour le marquage M toute séquence de franchissement s_c à partir de M qui vérifie les propriétés suivantes :
- les transitions de s_c sont les transitions de $T_{X,M}$
- toute transition t de $T_{X,M}$ n'apparaît qu'une fois au plus dans s_c
- toute séquence s_c' obtenue en permutant les transitions de s_c est aussi une séquence de franchissement à partir de M
- s_c est maximale, c'est-à-dire il n'existe pas d'autres séquences plus longues qui contiennent toutes les transitions de s_c et qui vérifient les 3 propriétés.

Définition
On dit qu'un RdPI marqué $<R_I, M_0>$ est dans un état stable si le marquage atteint est tel qu'aucune transition ne peut être franchie sans prendre en compte de nouveaux événements.

Le RdPI évolue selon le principe suivant: lorsque dans un état stable plusieurs transitions sont réceptives à un sous-ensemble d'événements, on franchit d'abord une séquence de simulation complète, ensuite on effectue des tirs répétés des transitions franchissables sans prendre en compte de nouveaux

événements jusqu'à ce que l'on atteigne un nouvel état stable.
Lorsque pour un marquage accessible M et un sous-ensemble d'événements on peut pas atteindre un état stable on dit alors que le réseau n'est pas **prompt**.

Exemple
Considérons le RdPI de la figure 4.3

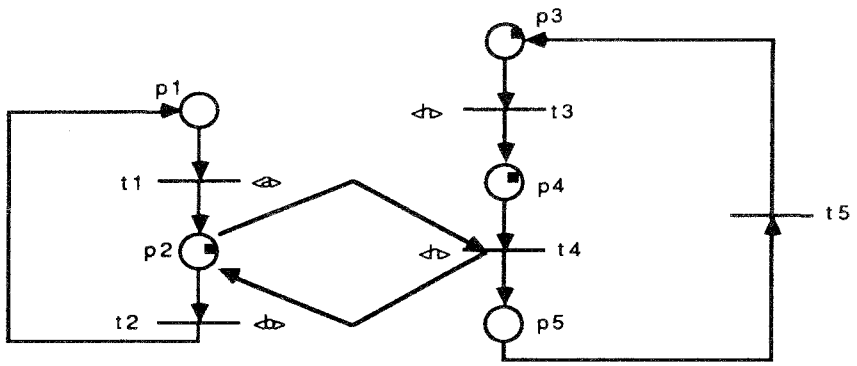


Figure 4.3

Si l'on suppose l'occurrence simultanée des deux événements b et h, alors l'ensemble des transitions réceptives aux événements b et h pour le marquage M est :
$T_{\{b,h\},M}$= { t2, t3, t4}
les séquences de simulation complètes par rapport à {b,h} pour le marquage M :
s_{c1} = (t2t3)
s_{c2}= (t3t4)

En pratique, le comportement d'un RdPI peut être étudié pour deux types d'environnement :
-l'environnement le moins contraignant qui admet l'occurrence simultanée d'événements externes
- et l'environnement le plus strict dans lequel les occurrences des événements externes sont toujours disjointes dans le temps.
Notons qu'on peut parler de simultanéité d'occurrences d'événements que relativement à des règles strictes qui permettent de décider de façon unique quand les occurrences d'événements distincts doivent être considérées comme étant simultanées. C'est pourquoi dans un contexte réparti et en l'absence d'horloge commune il est difficile de définir des critères rigoureux pour

comparer les événements engendrés par l'environnement sur des sites répartis. Ainsi pour garantir une interprétation unique de la spécification on convient de considérer les occurrences d'événements comme étant toujours disjointes dans le temps.

2.2 LES EXTENSIONS APPORTEES AUX RdPI

2.2.1 LA PRISE EN COMPTE DES OCCURRENCES D'EVENEMENTS

Nous avons vu dans le paragraphe 2.1.4 (règle 1) qu'une transition t à laquelle est associé un événement e et une condition c est franchissable ssi, t est validée, la condition c est vérifiée et une occurrence de l'événement e existe. Or si la transition t n'est pas validée ou si la condition c n'est pas vérifiée lors de l'occurrence de l'événement e, alors cette occurrence est perdue. De ce fait le mécanisme de synchronisation de cette règle ne permet pas de synchroniser le franchissement de la transition à une évolution passée de l'environnement. Ainsi; si le spécifieur désire exprimer des contraintes nécessitant une connaissance sur le passé de l'environnement, il doit modifier la structure de contrôle de son RdPI afin de mémoriser ce passé. Cette modification peut alourdir considérablement la structure du réseau, comme elle peut introduire des places non bornées et des transitions non vivantes nuisibles à l'analyse du réseau décrivant le système de commande [Tankoano 88]. Pour pallier cet inconvénient, nous proposons une solution qui consiste à associer aux transitions des types de communication qui caractérisent la politique de prise en compte des occurrences d'événements.

REGLE 3 : On peut alors considérer une transition t à laquelle est associé un événement e, un type de communication τ et une condition c, comme franchissable ssi les trois conditions suivantes sont vérifiées:

1- la transition est validée,

2- la condition c est vérifiée,

3- il existe une occurrence de e consommable selon τ.

Parmi l'ensemble des types de communication proposé dans [Perrin 85] nous avons retenu les types suivants :

Le type de communication "**égalité**": ce type implique que toute occurrence

émise par l'environnement est reçue une et une seule fois par le système. L'ordre des réceptions des occurrences est identique à l'ordre des émissions. Dans ce type de communication on n'admet aucune perte d'occurrences, qui peuvent être utilisées pour un traitement différé.

Le type de communication **"aléatoire"** : dans ce type de communication, l'occurrence consommée correspond à la plus récente occurrence détectée à cet instant. Ce type de communication implique que certaines occurrences peuvent être consommées plusieurs fois, alors que d'autres peuvent ne jamais être consommées.

Le type de communication **"rafraîchie"** : ce type de communication est semblable au type "aléatoire", sauf qu'aucune occurrence ne peut être consommée plus d'une fois.

Le type de communication **"strictement rafraîchi"**: l'occurrence consommée est la première occurrence détectée après la demande de consommation. Ce type de communication conduit aux mêmes règles de fonctionnement que la règle 1.

Le type de communication par **"bascule"** : ce type de communication implique que l'occurrence consommée est la première occurrence détectée après la dernière consommation.

2.2.2 LA COMMUNICATION INTER-MODULAIRE

Dans un contexte réparti, l'interaction entre les différentes entités modulaires ne peut se faire que par communication de messages. Pour permettre la prise en compte de cette interaction dans la spécification du comportement d'un module, nous associons en plus aux transitions du RdPI décrivant le module des opérations de communication synchrones et asynchrones.
Dans ce qui suit, nous précisons comment ces opérations de communication modifient les règles d'évolution des marquages des RdPI.

2.2.2.1 Cas des communications synchrones (CSP [Hoare 78])

Nous désignons par PE?m une opération de réception du message m sur le port

d'entrée PE, et par PS!m une opération d'émission du message m vers le port de sortie PS. Contrairement à CSP, on peut ainsi définir des rendez-vous entre plusieurs partenaires via des liaisons convergentes entre n émetteurs et un récepteur, ou via des liaisons divergentes entre un émetteur et n récepteurs. Dans le cas d'une liaison convergente, le récepteur n'acquitte un rendez-vous que ssi les n émetteurs ont tous émis le même message, tandis que dans le cas d'une liaison divergente, l'émetteur n'acquitte un rendez-vous que ssi les n récepteurs attendent le même message.

Les opérations de communication synchrone modifient les règles d'évolution d'un RdPI de la façon suivante :

REGLE 4 : Dans les cas (1) et (2) de la figure 4.4, si la transition t est validée, elle devient franchissable dès qu'une occurrence de e devient consommable selon τ, à condition que c soit vérifiée. Le tir de la transition est alors différé jusqu'à ce que le rendez-vous exprimé par ?M ou ! M devienne exécutable. Pendant ce temps, les marques dans les places en entrée qui rendent la transition valide sont réservées et ne sont donc plus disponibles pour le tir des autres transitions.

Par contre, les cas (3) et (4) imposent les mêmes règles d'évolution qu'un événement externe : si à un instant donné la transition t est validée, elle devient franchissable dès que le rendez-vous peut avoir lieu.

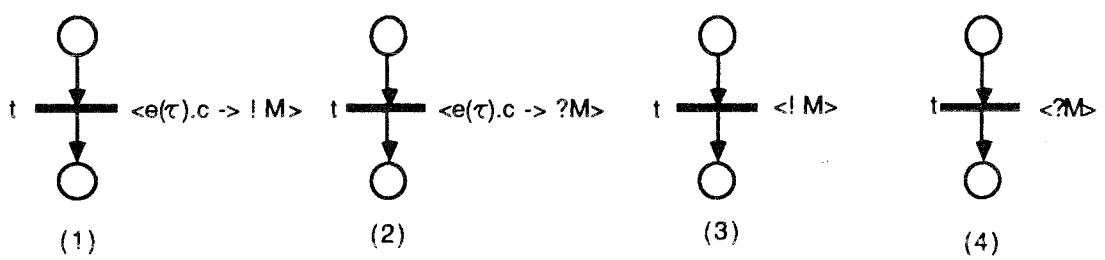


Figure 4.4

2.2.2.2 Cas des communications asynchrones

Les conventions de notation que nous retenons pour les opérations de communication asynchrones via des tampons non bornés sont les suivantes: PE?/m pour la réception et PS!/m pour l'émission. Ici également, nous admettrons des communications entre plusieurs partenaires via des liaisons

convergentes et divergentes. Dans le cas d'une liaison convergente, chaque message émis par chacun des n émetteurs est reçu par le récepteur. Dans le cas d'une liaison divergente, chaque message émis par l'émetteur ne sera reçu que par un seul des n récepteurs choisi au hasard parmi les récepteurs prêts à recevoir le message. Ce choix de politique de communication est imposé par les règles de production et de consommation des marques dans un RdP.

REGLE 5 : Dans la figure 4.5, dans (1) et (3), la réception du message M est traitée comme une condition pouvant rendre la transition t franchissable. Dans (1), si la transition t est validée, elle devient franchissable dès qu'une occurrence de e devient consommable selon τ à condition que c soit vérifiée et que le message M soit recevable . Par ailleurs, dans (3), si la transition t est validée, elle devient franchissable si le message M est recevable. Par contre, dans (2) et (4), les opérations de communication ne modifient en rien les règles d'évolution des marquages.

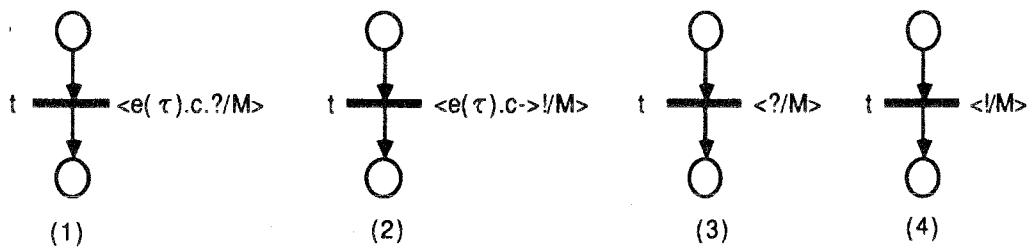


Figure 4.5

2.2.2.3 L'utilisation combinée des communications synchrones et asynchrones

A partir des définitions précédentes, nous en déduisons des formes syntaxiques plus générales, qui permettent d'associer au tir d'une transition, à la fois des interactions synchrones et asynchrones (figure 4.6). Ces interactions s'effectuent toujours dans un ordre bien déterminé : réception asynchrone, rendez-vous, et émission asynchrone.

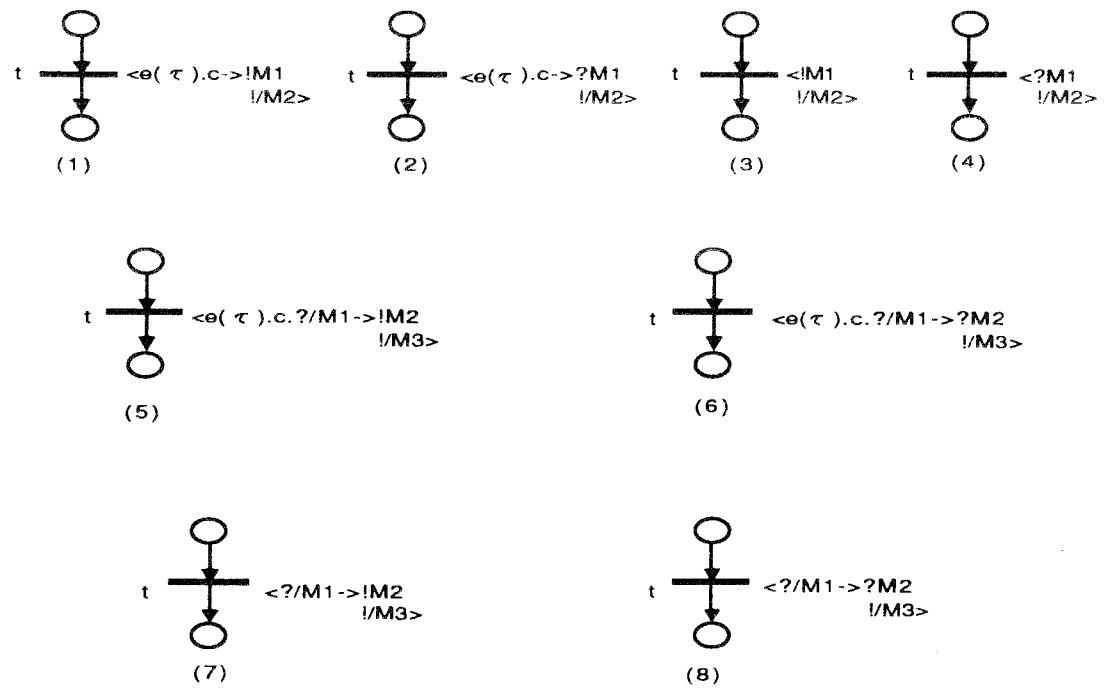


Figure 4.6

2.3 EXEMPLES

2.3.2 L'ATELIER DE BOBINAGE [Hollinger 85]

Cet exemple concerne un atelier de production de bobines de fil textile. L'atelier est composé de bobinoirs groupés en métiers. Nous ne nous intéresserons qu'au fonctionnement d'un métier. Une bobine est obtenue à l'aide d'un bobinoir entraînant en rotation un axe équipé d'un tube en carton, sur lequel est accroché l'amorce d'un fil en défilement continu. Lorsque la bobine est achevée, le fil est coupé en amont du bobinoir puis dévié vers un collecteur de déchets. La bobine achevée est retirée, et remplacée par un tube vide.

La coupe du fil, le remplacement d'une bobine et l'accrochage de l'amorce du fil sont assurés par un système embarqué; une navette assure le transport des tubes vides et des bobines achevées. Le système embarqué et la navette se déplacent chacun sur une voie parallèle au métier, à partir de leurs positions de dégagement. Lorsqu'une bobine est en voie d'achèvement, une requête est transmise au système embarqué et à la navette qui déclenchent leur déplacements vers le bobinoir concerné. Le remplacement de la bobine par un tube vide débute alors dès que le système embarqué termine sa course, et dès que la fin effective du bobinage intervient. Si un incident de casse de fil se produit

sur le bobinoir, le système embarqué et la navette retournent à leur position de dégagement.

Description formelle de l'énoncé :

Nous définissons comme suit les variables d'E/S du système de commande d'un métier:

Les variables d'entrée

Ereq[i] : indique qu'une bobine est en voie d'achèvement sur le bobinoir no. i
Ecasse[i] : indique qu'il y a casse de fil sur le bobinoir no. i
Efbob[i] : indique qu'une bobine est achevée sur le bobinoir no. i
Efdepse[i] : indique que la fin de course du système embarqué vers le bobinoir no. i est détectée.
Efretse : indique la fin de course du système embarqué vers sa position de dégagement.
Efdepn[i] : indique que la fin de course de la navette vers le bobinoir no. i est détectée.
Efretn : indique la fin de course de la navette vers sa position de dégagement.

Les variables de sortie

Adepse : ordre de déplacement du système embarqué vers un bobinoir.
Aretse : ordre de déplacement du système embarqué vers sa position de dégagement.
Aarrse : ordre d'arrêt du système embarqué.
Acf : ordre de coupe du fil par le système embarqué.
Aech : ordre d'échange de la bobine avec un tube vide par le système embarqué.
Adepn : ordre de déplacement de la navette vers un bobinoir.
Aretn : ordre de déplacement de la navette vers sa position de dégagement.
Aarrn : ordre d'arrêt de la navette.

Le comportement du système de commande du métier est décrit par le RdPI de la figure 4.7.

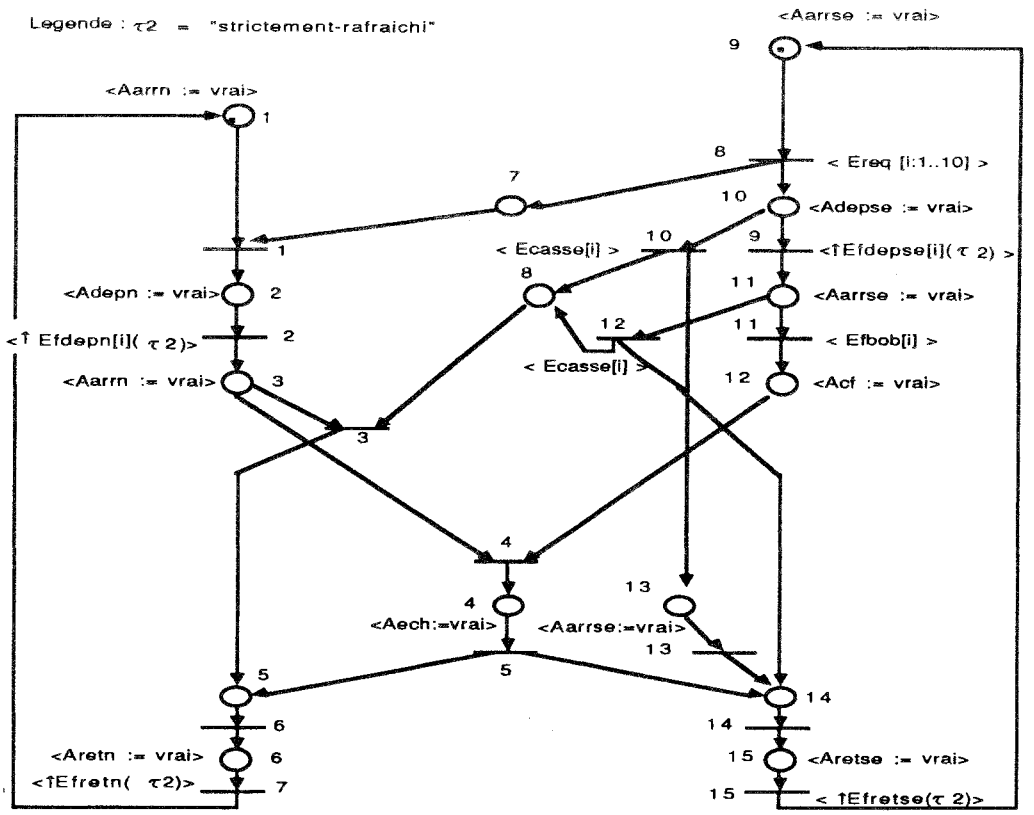


Figure 4.7

2.3.2 LA POMPE DE DRAINAGE [Kramer 80]

Il s'agit de maintenir en dessous d'un certain seuil le niveau de l'eau, et d'assurer la surveillance de l'environnement atmosphérique, dans une galerie souterraine d'une mine de charbon. On dispose pour cela

- d'une pompe de drainage pouvant recevoir les commandes de fonctionnement "dem" pour sa mise en route et "stop" pour son arrêt,

- d'un capteur de niveau servant à détecter les seuils "haut" et "bas" de l'eau dans la galerie,

- et de capteurs de méthane, de monoxyde de carbone (CO_2), et d'aération qui permettent à tout instant de connaître le taux de ces gaz.

La conduite du dispositif est assurée en surface par un opérateur qui peut le mettre en service (commande "marche") ou hors service (commande "arrêt"). Cet opérateur doit par ailleurs avoir la possibilité de connaître l'état de la pompe et des gaz sur simple requête, et doit être informé sous forme d'alarmes de tout

dépassement de seuil des gaz.

La pompe doit fonctionner automatiquement une fois qu'elle a été mise en service par l'opérateur, en fonction du niveau de l'eau dans la galerie. Pour des raisons de sécurité, elle ne doit pas démarrer ou continuer à fonctionner, dès que le pourcentage de méthane atteint un certain seuil.

Description formelle de l'énoncé :

Nous définissons les variables d'E/S du système de commande de ce dispositif comme suit :

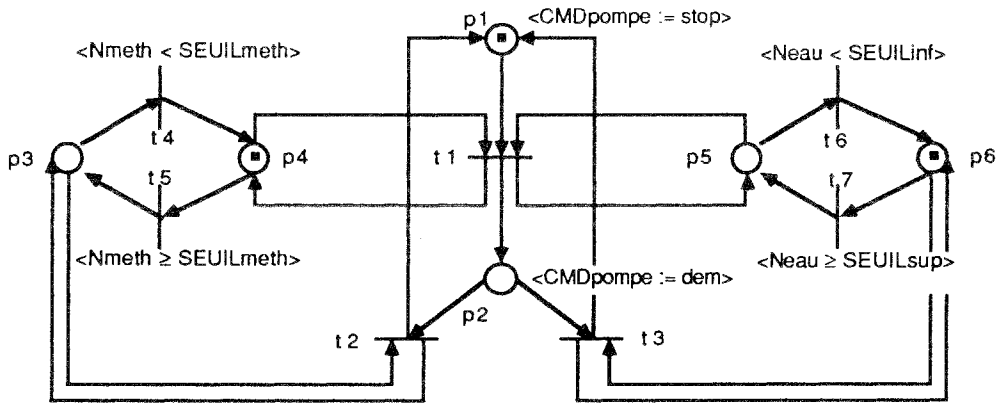
Les variables d'entrée

Neau : indique le niveau de l'eau dans la galerie.
Nmeth : indique le taux de méthane dans l'atmosphère.
Nco_2 : indique le taux de CO_2 dans l'atmosphère.
Nair : indique le niveau d'aération dans l'atmosphère.
MARCHEop: sert à détecter l'ordre de mise en service du dispositif par l'opérateur
ARRETop: sert à détecter l'ordre de mise hors service du dispositif par l'opérateur
REQpompe : sert à détecter les requêtes de l'opérateur sur l'état de la pompe.
REQmeth : sert à détecter les requêtes de l'opérateur sur le taux de méthane.
$REQco_2$: sert à détecter les requêtes de l'opérateur sur le taux de CO_2.
REQair : sert à détecter les requêtes de l'opérateur sur le niveau d'aération.

Les variables de sortie

CMDpompe: ordre de démarrage et d'arrêt de la pompe.
REPetatp : fournit à l'opérateur l'état de la pompe.
REPmeth : fournit à l'opérateur le taux de méthane.
$REPco_2$: fournit à l'opérateur le taux de CO_2.
ALMmeth : ordre de déclenchement de l'alarme du méthane.
$ALMco_2$: ordre de déclenchement de l'alarme de CO_2.
ALMair : ordre de déclenchement de l'alarme d'air.

Le RdPI de la figure 4.8 décrit seulement comment la mise en route et l'arrêt de la pompe alternent en fonction du niveau de l'eau et du niveau de méthane dans la galerie. La description du reste du système de commande peut être retrouvée dans [Tankoano 88].



N.B.: "pompe en état d'arrêt" = M(p1) = 1

"pompe en état de marche" = M(p2) = 1

"taux du méthane en dessous du seuil" = M(p4) = 1

"taux du méthane au dessus du seuil" = M(p3) = 1

"le niveau de l'eau a atteint le seuil supérieur et n'est pas redescendu jusqu'au seuil inférieur" = M(p5) = 1

"le niveau de l'eau est descendu jusqu'au seuil inférieur et n'est pas remonté jusqu'au seuil supérieur" = M(p6) = 1

Figure 4.8

3 MODELE DE DESCRIPTION DE L'ENVIRONNEMENT

Dans cette partie nous abordons la présentation du modèle de description du comportement de l'environnement avec lequel le système de commande interagit. Le modèle que nous proposons s'appuie sur des types génériques qui permettent d'instancier des **générateurs d'événements** de modification des variables d'entrée du système de commande. Les événements engendrés par ces générateurs permettent au cours d'une simulation du système de commande, de rendre compte, le plus fidèlement possible, de l'évolution des grandeurs physiques de l'environnement commandé.

Ce modèle a l'avantage d'être simple et facile à utiliser. D'autre part, il ne demande aucun effort de programmation de la part de l'utilisateur ce qui est fondamental dans un contexte de prototypage rapide.

3.1 PRINCIPE DE MODELISATION

Pour modéliser le comportement de l'environnement d'un système de

commande, le spécifieur identifie dans un premier temps les grandeurs physiques qui doivent être observées par le système de commande. Ensuite, il instancie les générateurs d'événements qui rendent compte de l'évolution de ces grandeurs. Les événements qui caractérisent l'évolution de ces grandeurs physiques sont de deux types :

- les événements **exogènes** dont les occurrences peuvent être considérées comme indépendantes des ordres engendrés par le système de commande,
- et les événements **endogènes** dont les occurrences dépendent des ordres engendrés par le système de commande.

Par ailleurs, la prise en compte d'un événement par le système de commande peut s'effectuer soit en détectant le passage d'une grandeur physique par sa position nominale à l'aide de capteurs du type tout ou rien dans des variables **d'entrée binaires**. Soit en observant la position courante d'une grandeur physique au moyen de capteurs numériques dans des variables **d'entrée continues**.

Dans le cas d'une variable binaire, si la grandeur physique ne peut occuper la position nominale que pendant un temps très court, nous dirons qu'il s'agit d'une variable **binaire fugitive**. A l'inverse si la grandeur physique peut occuper la position nominale pendant un temps long, nous dirons qu'il s'agit d'une variable **binaire non fugitive**. Comme exemple de variables binaires fugitives, on peut citer les variables d'entrée servant pour la détection du passage d'un chariot devant des postes de travail, et comme exemple de variables binaires non fugitives une variable d'entrée servant pour la détection de la présence d'une pièce à usiner dans un chariot.

En nous basant sur ces notions nous proposons sept types génériques de générateurs d'événements. Chacun des six premiers types est un modèle qui permet d'instancier une famille de générateurs d'événements, qui ont en commun le fait que les événements engendrés sont tous de même type (endogène ou exogène) et modifient le même type de variables (binaire fugitif, binaire non fugitif, ou continu). Chaque instanciation permet au spécifieur de préciser les conditions d'occurrence des événements de modification d'une variable d'entrée, qui rendent compte de l'évolution d'une grandeur physique. Toutefois, dans un environnement complexe, l'occurrence d'un autre événement correspondant par exemple à un incident peut changer le cours normal de l'évolution de cette grandeur physique, et remettre donc en cause ces conditions.

Aussi, le rôle du septième type générique sera de permettre au spécifieur de provoquer la purge d'un événement, lorsque celui-ci doit être invalidé du fait de l'occurrence d'un autre événement.

3.2 LES TYPES GENERIQUES DE GENERATEURS D'EVENEMENTS

3.2.1 LES GENERATEURS D'EVENEMENTS EXOGENES BINAIRES FUGITIFS

Un générateur d'événements exogènes binaires fugitifs permet d'engendrer selon une périodicité constante ou aléatoire, des événements qui forcent une variable binaire fugitive à sa valeur significative le temps d'un cycle de simulation. Après le cycle de simulation, la variable concernée reprend sa valeur non significative.
Un générateur d'événements exogènes binaires fugitifs s'instancie en précisant les paramètres suivants:

- le nom d'une variable d'entrée du système de commande,
- la valeur significative de cette variable,
- et la loi de distribution de l'intervalle de génération des événements.

3.2.2 LES GENERATEURS D'EVENEMENTS EXOGENES BINAIRES NON FUGITIFS

Un générateur d'événements exogènes binaires non fugitifs permet d'engendrer selon une périodicité fixe, des événements qui forcent une variable binaire non fugitive à vrai selon une probabilité p, et à faux selon une probabilité 1- p. Il s'instancie en précisant :

- le nom d'une variable d'entrée du système de commande,
- la probabilité avec laquelle cette variable prend la valeur vrai ou faux,
- et la fréquence de génération des événements modifiant cette variable.

3.2.3 LES GENERATEURS D'EVENEMENTS EXOGENES CONTINUS

Un générateur d'événements exogènes continus permet d'engendrer selon une périodicité fixe, des événements qui font varier le contenu d'une variable continue d'un Δx qui peut être fixe ou aléatoire. Il s'instancie en précisant les paramètres suivants:

- le nom d'une variable d'entrée du système de commande,

- la loi de distribution des Δx correspondant aux variations de cette variable,
- et la fréquence de génération des événements modifiant cette variable.

3.2.4 LES GENERATEURS D'EVENEMENTS ENDOGENES BINAIRES FUGITIFS

A l'inverse des générateurs d'événements exogènes, les générateurs d'événements endogènes sont soumis à des conditions de génération qui portent sur l'état du système de commande. Un générateur d'événements endogènes binaires fugitifs permet, chaque fois que la condition de génération qui lui est associée devient vrai, d'engendrer avec une probabilité p un événement de modification de variable binaire fugitive. La variable v_i à modifier peut être choisie avec une probabilité p_i parmi n variables. La somme des probabilités associées aux n variables doit alors être égale à p. Chaque événement engendré force une variable à sa valeur significative au bout d'un temps Δt qui peut être fixe ou aléatoire. Après un cycle de simulation, la variable modifiée reprend sa valeur non significative. Un générateur d'événements endogènes binaires fugitifs s'instancie en précisant :

- l'état dans lequel doit se trouver le système de commande pour qu'un événement soit engendré (la condition de génération),
- et pour chaque variable d'entrée du système de commande pouvant être modifiée
  - son nom,
  - sa valeur significative,
  - la probabilité pour qu'un événement engendré la concerne,
  - et la loi de distribution du temps Δt au bout duquel cet événement provoque sa modification

3.2.5 LES GENERATEURS D'EVENEMENTS ENDOGENES BINAIRES NON FUGITIFS

Les générateurs d'événements endogènes binaires non fugitifs se différencient des générateurs d'événements endogènes binaires fugitifs uniquement par le fait que les événements engendrés ne modifient pas les variables d'entrée que le temps d'un cycle de simulation. Les variables modifiées conservent leur modification après le cycle de simulation. Un générateur

d'événements endogènes binaires non fugitifs s'instancie en précisant :

- l'état dans lequel doit se trouver le système de commande pour qu'un événement soit engendré,
- et pour chaque variable d'entrée du système de commande pouvant être modifiée
  - son nom,
  - la valeur qu'elle prend,
  - la probabilité pour qu'un événement engendré la concerne,
  - et la loi de distribution du temps Δt au bout duquel cet événement provoque sa modification.

3.2.6 LES GENERATEURS D'EVENEMENTS ENDOGENES CONTINUS

Les générateurs d'événements endogène continus sont caractérisés par une condition d'activation et par une condition d'arrêt de la génération, qui portent sur l'état du système de commande. Chaque fois que la condition d'activation qui lui est associée devient vrai, un générateur d'événements endogènes continus engendre selon une périodicité fixe, des événements qui font varier le contenu d'une variable continue d'un Δx fixe ou aléatoire, jusqu'à ce que la condition d'arrêt devienne vraie. Le contenu de la variable d'entrée peut être forcée au début de chaque cycle de génération à une valeur particulière. Un générateur d'événements endogènes continus s'instancie en précisant :

- le nom d'une variable d'entrée du système de commande,
- l'état dans lequel doit se trouver le système de commande pour que la génération débute (condition d'activation de la génération),
- l'état dans lequel doit se trouver le système de commande pour que la génération s'arrête (condition d'arrêt de la génération),
- la valeur de départ de la variable (optionnel),
- la loi de distribution des Δx correspondant aux variations de la variable,
- et la fréquence de génération des événements.

3.2.7 LES GENERATEURS DE PURGES

Un générateur de purges permet, chaque fois qu'un événement donné est engendré, de provoquer la purge de tous les événements que son occurrence invalide. La date prévue pour l'occurrence d'un événement qui fait l'objet d'une

purge doit être supérieure à la date prévue pour l'occurrence de l'événement qui provoque sa purge. Un générateur de purge s'instancie en précisant les paramètres suivants:

- le nom de la variable d'entrée dont les événements de modification provoquent des purges,
- et la liste des variables d'entrée dont les événements de modification doivent être purgés.

3.3 EXEMPLE

L'exemple que nous présentons ici montre comment on peut utiliser les types génériques proposés pour instancier des générateurs d'événements modélisant le comportement de l'environnement du système de commande de la pompe de drainage (cf. §2.3.2) décrit par le RdPI de la figure 4.8. Ce réseau décrit seulement comment la mise en route et l'arrêt de la pompe doivent alterner en fonction de l'évolution de deux grandeurs physiques : le niveau de l'eau et le taux de méthane dans la galerie souterraine.

L'évolution de la grandeur physique "taux de méthane" dans la galerie n'est pas directement liée aux ordres engendrés par la partie commande, alors qu'à l'inverse, le sens de l'évolution du niveau de l'eau dépend de l'état de fonctionnement de la pompe. Selon que la pompe est en état d'arrêt ou de marche, le niveau de l'eau dans la galerie souterraine augmente ou diminue.
Les événements de modification de la variable "Nmeth" qui rendent compte de l'évolution du méthane peuvent de ce fait être engendrés par un générateur d'événements exogènes continus, instancié avec les paramètres suivants:

- nom de la variable d'entrée : Nmeth
- fréquence de génération des événements de modification de la variable : toutes 5 unités de temps
- loi de distribution des Δx correspondant à la variation du taux de méthane toutes les 5 unités de temps:uniformément distribué entre -2 et +2

Quant aux événements de modification de la variable d'entrée "Neau" qui servent à rendre compte de l'évolution du niveau de l'eau, ils peuvent être engendrés par deux générateurs d'événements endogènes continus. Le premier générateur engendre les événements qui rendent compte de la baisse du niveau de l'eau quand la pompe est en état de fonctionnement, et le deuxième

générateur les événements qui rendent compte de l'augmentation du niveau de l'eau quand la pompe s'arrête de fonctionner. Ils s'instancient de la façon suivante:

1er générateur:

- nom de la variable d'entrée : Neau
- condition d'activation de la génération d'événements : $M(p_2) = 1$
- condition d'arrêt de la génération d'événements : $M(p_2) = 0$
- fréquence de génération des événements de modification de la variable: toutes les 5 unités de temps
- loi de distribution des Δx correspondant à la baisse du niveau de l'eau toutes les 5 unités de temps : variation constante de -10.

2ème générateur:

- nom de la variable d'entrée : Neau
- condition d'activation de la génération d'événements : $M(p_1) = 1$
- condition d'arrêt de la génération d'événements : $M(p_1) = 0$
- fréquence de génération des événements de modification de la variable: toutes les 5 unités de temps
- loi de distribution des Δx correspondant à l'augmentation du niveau de l'eau toutes les 5 unités de temps : variation constante de +5.

3.4 COMPARAISON AVEC L'APPROCHE PRECONISEE DANS SICLOP [BENZAKOUR 86]

Nous avons vu que dans SICLOP (cf. chapitre 2 §3.1), on modélise le comportement de l'environnement d'un système de commande en associant à chaque grandeur physique une machine à états, qui décrit l'enchaînement des différentes activités qui peuvent la faire évoluer. Ensuite, on définit par rapport à cette machine à états des générateurs d'événements (le couple (v) (L)) servant à rendre compte de son évolution.

Dans l'approche que nous préconisons, les générateurs d'événements sont plutôt définis par rapport à l'état de la partie commande.

Vu de cette façon, on peut dire que les deux approches se rejoignent, puisqu'il existe une certaine dualité entre l'état du système de commande et les

activités en cours dans l'environnement.

Les différences qui existent entre les deux propositions proviennent essentiellement du fait que dans SICLOP on n'offre qu'un seul type de générateurs d'événements :

1°) Ce type de générateurs d'événements ne permet que l'instanciation de générateurs d'événements endogènes.

2°) Les événements endogènes qui peuvent être engendrés ne peuvent concerner que l'observation de grandeurs physiques évoluant à vitesse constante. Même s'il est vrai que dans les procédés industriels la plupart des grandeurs physiques évoluent effectivement de façon presque linéaire dans les deux sens, cette contrainte peut devenir un grand handicap. En effet, la génération des événements à vitesse constante conduit au cours d'une simulation, à n'observer qu'un seul comportement possible du système de commande, alors que dans une phase de mise au point des spécifications, à défaut d'observer tous les comportements possibles, on aimerait bien au moins pouvoir en observer plusieurs différents. Par exemple, il peut être intéressant de pouvoir engendrer au cours d'une simulation des événements de pannes qui peuvent intervenir à différentes étapes de l'évolution du système.

3°) Enfin, le modèle n'offre pas de possibilité de purge.

4 MODELE DE DESCRIPTION DU FLUX D'ACTIVITE

Le but de la modélisation du flux d'activité est de compléter la description du système de commande par une description macroscopique des éléments d'organisation et de fonctionnement d'un atelier qui peuvent permettre au cours d'une simulation de collecter des statistiques fines et pertinentes sur les files d'attente : longueur maximale, minimale et moyenne, temps d'attente, débit et taux d'occupation pour l'évaluation des performances.

Le modèle proposé pour cela conduit à percevoir l'atelier comme un réseau de files d'attente. Il s'appuie sur un formalisme largement inspiré des RdP colorés tels que introduits dans [Jensen 81]. La particularité de ce formalisme par rapport aux RdP colorés, tient essentiellement au fait que les marques sont traitées dans les places comme dans des files d'attente. Il en résulte les deux

avantages suivants :

- la cohérence entre les différents formalismes que nous utilisons,
- la clarté qu'apporte les description à l'aide de files d'attente et la concision des RdP colorés.

4.1 LES ELEMENTS DU MODELE RETENU

4.1.1 LES MARQUES

Les marques circulant dans le réseau du flux d'activité servent à modéliser des entités dynamiques dans l'atelier, exemple pièces, outils, chariots, ... Ces marques sont appelées transactions.
Aux marques peut être associé un ensemble d'attributs (identifiant de la transaction, priorité de la transaction, classe, identifiant de la file d'attente à laquelle appartient la transaction, ...), elles seront ainsi **différenciées** les unes des autres et **individualisées**.

4.1.2 LES PLACES

Chaque place du réseau sert à modéliser une ou plusieurs files d'attente et est caractérisée par la politique de gestion de ces files (PG) et leur capacité (C).

La politique de gestion permet de spécifier comment seront prises en compte les transactions au moment du traitement. Les politiques de gestion que nous avons retenues sont :
la politique FIFO (First In First Out) permet de gérer les transactions selon leur ordre d'arrivée, la politique LIFO (Last In First Out) selon l'ordre inverse de leur arrivée, la politique PRIOR selon l'ordre de priorité affecté aux transactions et la politique RAND dans un ordre quelconque.

La capacité d'une file permet de préciser le nombre maximum de transactions qu'elle peut contenir. Dans le cas où la capacité d'une file d'attente est inconnue, nous noterons par convention qu'elle est infinie (i.e $C = \infty$).

4.1.3 LES TRANSITIONS

A chaque transition est associé un ensemble E_i de types possibles d'écoulement du flux d'activité noté par FA (i.e l'ensemble exhaustif des transactions pour lesquelles la transition peut être franchie et des files où ces transactions doivent être sélectionnées) et le lien de synchronisation de ces écoulement de flux à l'évolution de la commande.
Ce lien est décrit par la liste des transitions et/ou la liste des événements externes apparaissant dans le réseau décrivant la partie commande.
Les liens de synchronisation permettent au cours de la simulation de faire évoluer conjointement le réseau décrivant le système de commande et le réseau décrivant le flux d'activité.

4.1.4 LES ARCS

Chaque arc (p_i, t_j) du réseau est étiqueté par des expressions qui définissent les transactions à sélectionner et les files où ces transactions doivent être sélectionnées. Ces expressions sont de deux types :

• $<\varphi>$ où φ désigne une fonction de l'ensemble E_j de types d'écoulement de flux associé à la transition t_j dans l'ensemble des files X [l'ensemble des classes de transactions].

• < IDX > avec X une variable libre.

Le premier type d'expression permet de préciser l'identifiant de la file dans la place p_i et éventuellement la classe de transactions, la transaction est choisie selon la politique de gestion associée à p_i. Quant au second type d'expression, il indique que les identifiants des transactions sélectionnées dans les files qui sont liées à la transition t_j par des arcs portant des étiquettes utilisant la même variable doivent être égaux. Si dans les files considérées, il existe à la fois des files FIFO, LIFO ou PRIOR et RAND, les transactions doivent être sélectionnées dans les files RAND de manière à ce que leur identifiant soit le même que celui des transactions en tête, en queue ou avec la plus grande priorité des files FIFO, LIFO ou PRIOR.
Ce type de sélection permet de gérer l'évolution d'une transaction dans plusieurs files d'attente modélisant des sections imbriquées les unes dans les autres.

On peut ainsi par exemple obtenir parallèlement des statistiques sur le temps de transit d'un type de pièces dans une section et des statistiques sur le temps de transit de ces pièces dans chacun des postes de travail qui compose la section.

De même, chaque arc (t_i, p_j) est étiqueté par des expressions qui définissent les transactions à produire des files en sortie de la transition t_i et leurs caractéristiques (priorité, la classe, l'identifiant de la file dans laquelle la transaction doit être introduite, ...). Les étiquettes peuvent être de la forme :

• $<\varphi>$: φ est une fonction qui précise les transactions à produire et l'identifiant des files dans lesquelles ces transactions doivent être produites.

• < IDX > : permet d'indiquer (dans le cas où au moins un arc d'entrée à la transition t_i porte une étiquette de la forme < IDX > utilisant la même variable) que l'identifiant de la transaction à produire est le même de celui de la transaction de l'arc portant l'étiquette utilisant la même variable X.
Si cette étiquette apparaît sur d'autres arcs de sortie, le même identifiant est affecté à toutes les transactions à produire dans les différentes files de sortie correspondantes.

4.2 EXEMPLE

Cet exemple [Alla 86] concerne une section de travail que l'on peut retrouver dans divers types d'ateliers. Cette section permet l'usinage de trois types de pièces nécessitant trois types de postes de travail différents. On souhaite disposer les postes de travail de part et d'autre d'un convoyeur central qui sert à l'acheminement des pièces (fig 4.9). Pour éviter la congestion de ce convoyeur central, des stocks intermédiaires sont prévus au niveau de chaque poste. Le problème pour ce type d'organisation est double :

- Déterminer le nombre de postes de travail de chaque type, et la taille des stocks intermédiaires, qui permettent d'éviter tout ralentissement des autres sections situées en amont et en aval, pour un ordonnancement des pièces en entrée.

- Garantir un bon niveau du taux d'utilisation des postes.

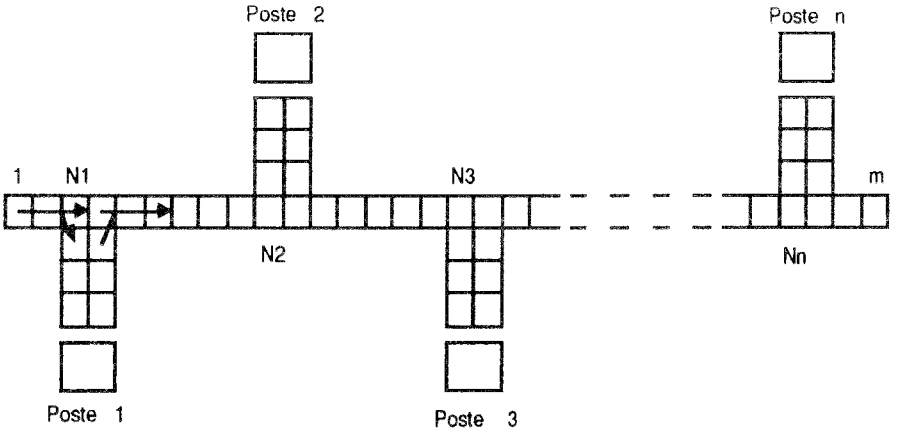
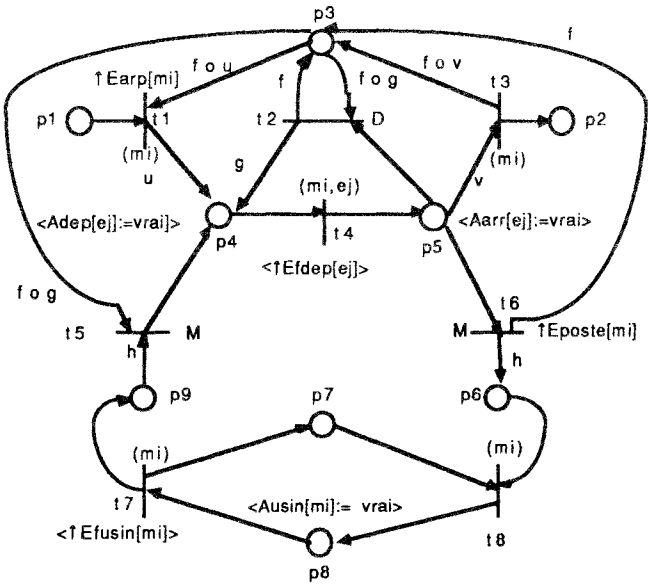


Figure 4.9

Ce problème peut être résolu par une série de simulations du fonctionnement de l'atelier. Sans faire au départ d'hypothèse sur la taille des stocks intermédiaires, on recherche le point où la diminution du nombre de postes n'influe plus sur l'amélioration de leur taux d'utilisation, et où le temps de transit des pièces dans la section commence à croître de façon sensible. On obtient ainsi non seulement le nombre de postes de travail à prévoir, mais aussi la taille des stocks intermédiaires qui correspond à la plus grande taille observée pendant la simulation.

La figure 4.10 décrit le comportement du système de commande, nous utilisons les RdP colorés pour réduire la taille du réseau.



MARQUAGE INITIAL

$M(p3) = \sum_{j=1}^{m} e_j$ $\quad M(p7) = \sum_{i=1}^{n} m_i$

ENSEMBLE DES COULEURS

{ mi, ej, <mi, ej> } $\forall i \in (1,n)$ $j \in (1,m)$

$C = \{ <mi, ej> \} \forall i \in (1,n)$ $j \in (1,m)$

$M = \{ <mi, eNi> \} \forall i \in (1,n)$

D = C - M

DEFINITION DES FONCTIONS

f (mi,ej) = ej

u (mi) = mi, e1

g (mi,ej) = mi,ej+1

v (mi) = mi, em

h (mi,ej) = mi

l'identité est associée aux arcs non étiquetés

VARIABLES D'ENTREE

Efdep : tableau [1..m] de booléen	Efdep[i]=vrai si fin de déplacement sur la portion i du convoyeur
Efusin: tableau [1..n] de booléen	Efusin[i]=vrai si fin usinage de la pièce mi
Earp : tableau [1..n] de booléen	Earp[i]=vrai si arrivée de la pièce mi
Eposte: tableau [1..n] de booléen	Eposte[i]=vrai si la pièce mi est devant le poste d'usinage

VARIABLES DE SORTIE

Adep : tableau [1..m] de booléen	Adep[i]:=vrai déclenche un déplacement sur la portion i
Aarr : tableau [1..m] de booléen	Aarr:=vrai déclenche un arrêt sur la portion i
Ausin: tableau[1..n] de booléen	Ausin[i]:=vrai déclenche l'usinage de la pièce mi

Figure 4.10

Dans ce réseau :

- Les couleurs m_i servent à différencier les postes d'usinage et les pièces (par rapport aux postes où elles doivent être usinées).

- Les couleurs e_j identifient les différentes portions du convoyeur.

- Les places p_3, p_4, p_5 et les transitions t_1, t_2, t_3 décrivent les règles de synchronisation liées à l'évolution des pièces sur le convoyeur selon le principe FIFO.

Chaque marque de la place p_3 est caractérisée par une couleur e_j qui sert à indiquer que la portion de rang j du convoyeur est libre. Chaque marque de la place p_4 est caractérisée par une couleur représentée par le couple (m_i, e_j) qui signifie qu'une pièce de type i occupe la portion de rang j du convoyeur. De même pour la place p_5. L'interprétation associée à la place p_4 définit l'action de commande qui déclenche le déplacement de la pièce vers la portion j du

convoyeur. L'interprétation associée à la place p_5 définit l'action de commande qui déclenche l'arrêt sur la portion j lorsqu'on atteint la fin de la portion j.

- Les places p_6 et p_9 modélisent les stocks intermédiaires des postes de travail. Ces stocks sont de capacité inconnue a priori. Chaque marque de la place p_6 est caractérisée par une couleur m_i qui indique le poste de travail sur lequel la pièce va être usinée. De même, chaque marque dans la place p_9 est caractérisée par une couleur m_i qui indique dans quel poste de travail la pièce a été usinée.

- Les marques dans la place P_8 servent quant à elles à modéliser des postes en cours d'usinage, et les marques dans la place p_7 des postes en attente.

- Les couleurs m_i associées à la transition t_1 indiquent que chaque franchissement de t_1 doit correspondre à l'entrée d'une pièce dans la section de travail. Les étiquettes des arcs (p_1, t_1) et (p_3, t_1) imposent qu'on ne puisse franchir la transition t_1 que si on a une marque de couleur m_i dans la place p_1 (l'arc (p_1, t_1) étiqueté par la fonction identité) et une marque de couleur e_1 dans la place p_3 indiquant que la première portion du convoyeur est libre.
Le franchissement de t1 consiste à enlever une marque de couleur e_1 dans la place p_3 et une marque de couleur m_i dans la place p_1, et à produire une marque de couleur (m_i, e_1) dans la place p_4, ce qui permet de déclencher le déplacement sur la portion 1.

- Chaque tir de la transitions t_4 est synchronisé à l'occurrence de l'événement ($\uparrow$ Efdep[e_j]) indiquant la fin de déplacement sur la portion j du convoyeur.

- La transition t_2 sert à modéliser les déplacements des pièces sur les différentes portions du convoyeur. Chaque couleur (m_i, e_j) associée à cette transition permet de faire déplacer la pièce m_i de la portion j vers la portion j+1. Le tir de cette transition enlève une marque de couleur (m_i, e_j) de p_5 et une marque de couleur e_{j+1} de p_3, et dépose une marque de couleur (m_i, e_{j+1}) dans p_4 et une marque de couleur e_j dans p_3 (ie. libère la portion j).

- Les couleurs m_i associées à la transition t_3 indiquent que chaque franchissement de t_3 correspond à la sortie d'une pièce m_i usinée.
L'étiquette de l'arc (p_5, t_3) impose qu'on ne peut franchir t_3 que si la place p_4 contient une marque de couleur (m_i, e_N) indiquant que la dernière portion du

convoyeur est occupée par une pièce usinée. Le franchissement de t_3 enlève alors cette marque, et dépose dans p_2 une marque de couleur m_i et dans p_3 une marque de couleur e_N libérant ainsi la dernière portion du convoyeur.

- L'ensemble des couleurs $\{<m_i, e_{Ni}>\}$ associé à la transition t_6 indique que chaque tir de t_6 correspond à la mise de pièces dans le stock d'entrée modélisé par la place p_6.
Le franchissement de t_6 retire de p_5 une marque de couleur (m_i, e_{Ni}) et dépose dans p_3 une marque e_{Ni} (ie. la portion Ni est libre) et dans p_6 une marque m_i.

- Les couleurs m_i associées à la transition t_8 indiquent que chaque franchissement de t_8 correspond à l'usinage de la pièce m_i sur le poste m_i.

- Le tir de la transition t_7 est synchronisé à l'occurrence de l'événement ($\uparrow$Efusin[m_i]) indiquant la fin de l'usinage de la pièce m_i. Le tir de t_7 enlève une marque m_i de p_8 et dépose dans la place p_7 une marque de couleur m_i (indiquant que le poste m_i est libre) et dans la place p_9 une marque m_i (ie. mise de la pièce dans le stock de sortie).

- Enfin, la transition t_5, permet d'enlever les pièces usinées du stock de sortie pour les acheminer vers la sortie de la station de travail.

Le réseau décrivant la partie commande (c.f. figure 4.10) est complété par une description du flux d'activité (Voir réseau de la figure 4.11).
Dans ce réseau, pour chaque type de pièces, une file est associée à chaque place.

Les transitions sont étiquetées par les couleurs pour lesquelles elles sont franchissables et par les liens de synchronisation avec le réseau de la partie commande (réseau de la figure 4.10).
Les étiquettes associées aux arcs permettent de déterminer les marques à retirer et à produire dans les files.

- Les marques contenues dans chaque file d'attente de la place pp_1 correspondent aux pièces présentes dans la section.

- De même pour les marques contenues dans la place pp_6. Cette place permet d'obtenir des statistiques sur le temps de transit des pièces.

- Les marques contenues dans chaque file d'attente de la place pp_2 correspondent aux pièces présentes dans le stock d'entrée.

- Chaque file de la place pp_3 permet d'indiquer la présence d'une pièce dans un poste de travail. Le tir de transition tt_3 permet de faire sortir une pièce du stock d'entrée pour usinage sur le poste correspondant. Les tirs de tt_3 sont liés aux tirs de la transition t_8 du réseau de la partie commande (cf. figure 4.10).

- Les marques contenues dans chaque file d'attente de la place pp_4 correspondent aux pièces usinées présentes dans le stocks de sortie.
Les places pp_3, pp_2 et pp_4 permettent d'obtenir des statistiques sur le taux d'utilisation des postes de travail et sur les la taille des stocks intermédiaires.

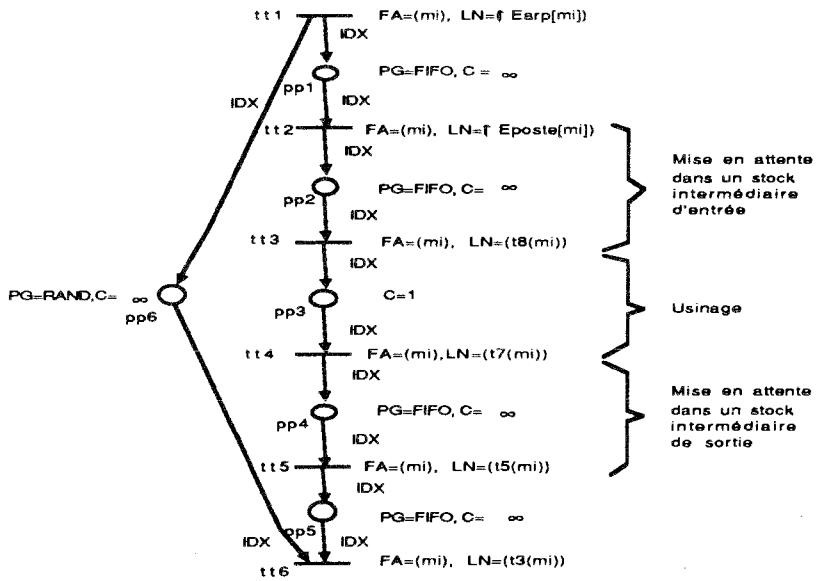


Figure 4.11

5 CONCLUSION

Nous venons de présenter les trois modèles retenus pour la description de systèmes de production. Ces modèles ont l'avantage de décrire de façon non redondante la dynamique de ces systèmes et de permettent d'utiliser un seul outil qui peut être exploité à diverses fins : pour la communication avec l'utilisateur , pour la validation et la mise au point de la spécification du système de commande et pour l'évaluation des performances.
La description de cet outil va faire l'objet du chapitre suivant.

CHAPITRE 5

DESCRIPTION DU SYSTEME PETRI-S

1 INTRODUCTION

Ce chapitre est consacré à la présentation de l'outil d'aide au prototypage rapide de systèmes de commande de procédés industriels : le logiciel PETRI-S.
Cet outil utilise les modèles de description du comportement du système de commande, de l'environnement commandé et du flux d'activité pour réaliser une simulation hors site du système de commande. Cette mise en œuvre simulée de la spécification du comportement du système de commande vise les objectifs suivants:

• **La clarification de l'expression des besoins**, elle constitue un support objectif permettant de débloquer les difficultés de l'expression des besoins, elle aide l'utilisateur à préciser ou compléter la spécification et constitue un moyen de communication "agréable" entre les différentes personnes concernées par la réalisation et l'utilisation du système.

• **La validation et la mise au point fine** de la spécification du système de commande, elle permet de vérifier que le système réalise bien le service pour lequel il est conçu et qu'il vérifie les bonnes propriétés liées à la vivacité et à la sûreté de fonctionnement.

• **L'évaluation des performances**, elle offre une assistance pour le choix de stratégies de pilotage et pour le dimensionnement de l'atelier.

Au cours du développement du logiciel PETRI-S nous avons plus particulièrement soigné les points suivants :

• la facilité de mise en œuvre, qualité essentielle d'un outil d'aide au prototypage,

• la souplesse d'utilisation, en mettant à la disposition de l'utilisateur un ensemble de fonctionnalités qu'il peut choisir et utiliser à son gré,

• la diversité des résultats engendrés : traces des entrées et sorties du système, marquages du réseau, propriétés du système et statistiques sur les files d'attente, le marquage des places et le tir des transitions,

• et la communication homme-machine, en fournissant une interface simple et hautement interactive avec multifenêtrage, menus et dispositif de désignation.

Nous abordons la description de ce logiciel en plusieurs parties:
La section 2, décrit en détail les objectifs de l'outil. La section 3, présente l'environnement de l'outil.
Dans la section 4, nous décrivons le langage de description de la spécification et l'analyseur qu'il lui est associé. Dans la section 5 est présenté l'éditeur graphique des RdPI.
L'outil fera l'objet des sections 6 et 7, nous décrivons les principes de fonctionnement, les modes d'exploitation et les résultats engendrés.

2 LES OBJECTIFS ET PRINCIPES DE BASE

PETRI-S peut être exploité à différents niveaux du processus de développement et à des fins diverses.

2.1 EXPRESSION DES BESOINS

Le développement de systèmes de commande est une activité très complexe. Cette complexité provient à la fois de la nature intrinsèque des systèmes développés et des nécessaires rapports entre spécialistes de domaines différents. Ce processus de développement est décomposé en plusieurs étapes, l'ensemble de ces étapes est appelé "cycle de développement du système". (Voir la figure 5.1).

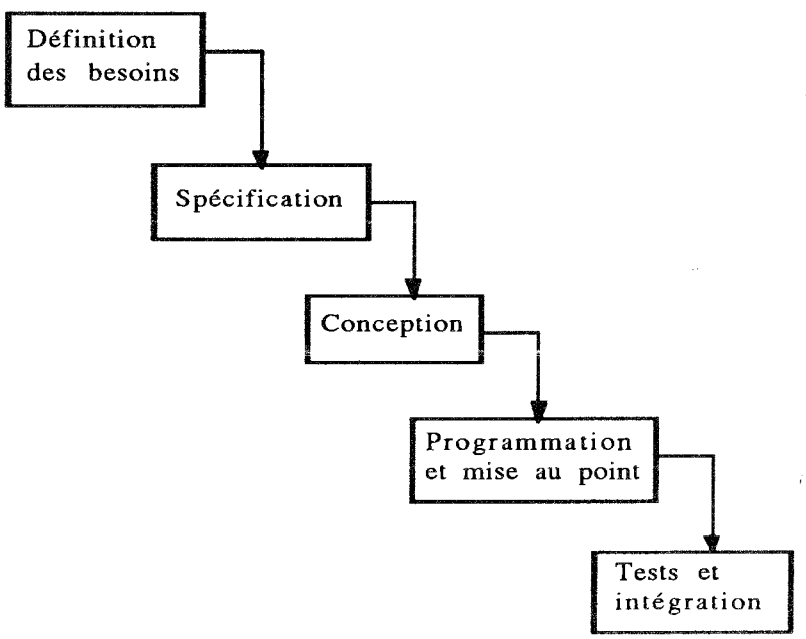


Figure 5.1: un exemple de développement d'un système la cascade [Boehm]

Etape de définition des besoins

Cette étape est consacrée à l'établissement du cahier des charges, dans lequel on exprime de façon informelle:

- les fonctions globales que doit réaliser le système,
- les contraintes opérationnelles et technologiques imposées,
- les sécurités à respecter,
- ...

Cette première étape est généralement menée par plusieurs personnes d'horizons très divers (le spécialiste du processus physique, l'exploitant, le spécialiste de la commande, ...). Ces différents intervenants doivent s'entretenir pour définir les caractéristiques du système, ils ont besoin d'outils pour communiquer entre eux.

Etape de spécification

Cette étape a pour vocation de fournir les spécifications du système à développer.

Ces spécifications sont élaborées à la suite de l'analyse des besoins et permettent de délimiter les grands contours du système en termes de services offerts. Elles consistent à décrire le comportement externe du système.
De nombreux travaux sur le développement de logiciels soulignent l'intérêt d'utiliser des outils et des méthodes de spécifications formelles. Une spécification est formelle, lorsqu'elle est écrite à l'aide de notation précise, non ambiguë, dont la syntaxe et la sémantique sont définies [Choppy 88].
De plus, le fait d'écrire des spécifications formelles rend possible la vérification des propriétés de ces spécifications, comme :

-**la fidélité** : qui exprime la conformité de la spécification aux besoins exprimés;

-**la complétude** : signifie que la spécification couvre l'ensemble des besoins;

-**la cohérence** (ou non-contradiction) : la spécification ne comporte pas d'éléments contradictoires;

Etape de conception

Il s'agit d'affiner la spécification et de définir l'architecture du système

(ie. l'ensemble des modules constituant le système et les liens entre eux).

Etape de programmation

Etape d'intégration

Dans ce processus de développement la définition des besoins et l'établissement de spécifications rigoureuses sont des tâches difficiles et coûteuses. Ceci est dû :

- aux imprécisions des besoins, l'utilisateur est généralement incapable de définir dès le départ de façon précise et complète l'ensemble de ses besoins, souvent ces besoins sont flous,

- à la difficulté pour l'utilisateur à exprimer ses besoins dans un langage autre que son langage naturel avec les inconvénients bien connus : il est imprécis et sujet à de nombreuses équivoques conduisant à des comportements surprenant pour l'utilisateur et le spécifieur une fois le système achevé,
- à la participation conjointe de plusieurs partenaires aux vocabulaires différents (utilisateurs, spécifieurs, ...). Ces différents intervenants doivent s'entendre pour définir les caractéristiques du système : ils doivent donc avoir une même compréhension mais aussi un langage technique commun pour dialoguer. Or force est de constater que souvent on se limite au langage naturel.

Un nouvel élément de réponse est apporté à présent aux problèmes de la définition et de la spécification des besoins : **le prototypage**.
Le prototypage permet avant tout de contribuer à la clarification des besoins de l'utilisateur, de consolider l'élaboration des spécifications et de faciliter le dialogue lors de l'expression des besoins.

L'outil que nous proposons permet au spécifieur de réaliser **rapidement** et à **moindre frais** des prototypes du système en cours de définition et de les soumettre à l'utilisateur. Ce dernier peut ainsi observer le comportement de son système, il peut alors préciser certains points, revenir sur certaines décisions ou formuler de nouvelles exigences. Le spécifieur prend en compte ces nouveaux besoins, re-spécifie le système et présente un autre prototype à l'utilisateur.
Ce processus peut être exécuté plusieurs fois jusqu'à ce que l'utilisateur ait défini de façon précise et complète l'ensemble de ses besoins (cf. figure 5.2).

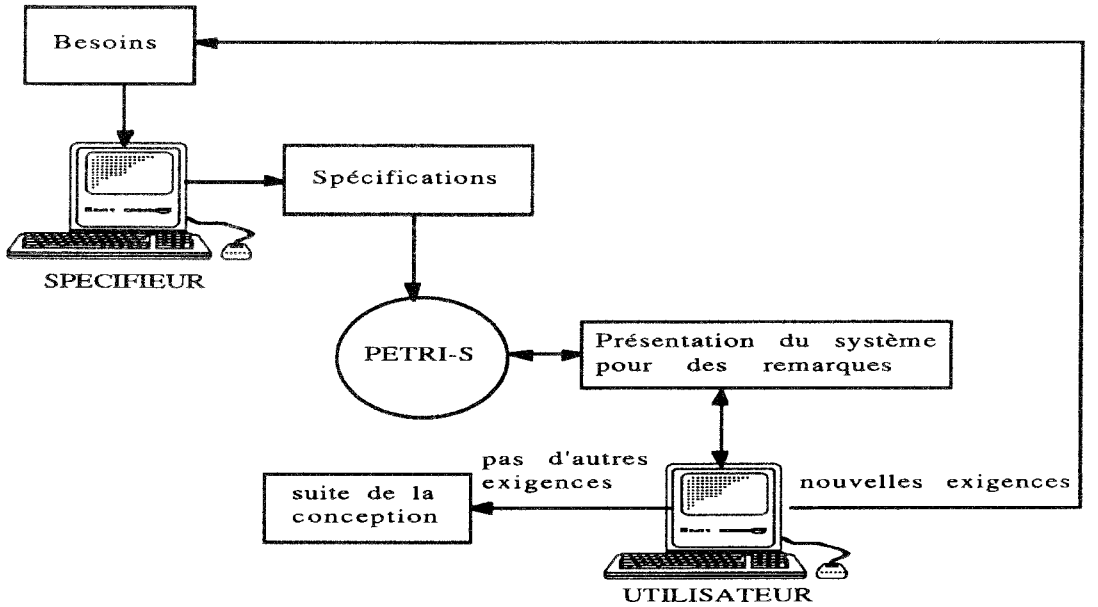


Figure 5.2

Le cahier des charges ainsi mis au point de façon interactive peut devenir un contrat entre l'utilisateur et le développeur, dans une forme compréhensible par les deux partenaires.

Par ailleurs, dans les approches traditionnelles de développement, l'utilisateur ne pourra voir fonctionner son système qu'à des étapes avancées de la réalisation ou au moment de la livraison du produit. Or, il arrive dans certains cas (à cause de malentendus qui persistent entre utilisateur et développeur) que le système réalisé ne réponde pas (du moins en partie) aux besoins réels de l'utilisateur. Ceci induit une série de modifications pour adapter le système aux besoins réels. Ces modifications présentent généralement de grands risques d'introduction d'erreurs. Elles sont coûteuses et détruisent tout effort d'optimisation.

PETRI-S peut être utilisé pour pallier les inconvénients de ce type de constatations tardives. Il permet à l'utilisateur de simuler la spécification : prend en compte les entrées du système et engendre les sorties conformément à la spécification introduite par le spécifieur. L'utilisateur compare alors le comportement du prototype avec ses besoins et signale les différences au spécifieur.

L'apport de PETRI-S est de permettre à l'utilisateur de porter une appréciation sur la spécification et donc sur ce qu'à compris le spécifieur de son problème sans l'obliger à lire la spécification.

conclusion

Au niveau de l'expression des besoins PETRI-S joue un rôle de premier plan, il aide l'utilisateur à clarifier ses besoins en lui permettant d'essayer son système et constitue un moyen de communication dont le rôle principal est d'éliminer des malentendus coûteux qui sont généralement détectés qu'au moment de la réalisation.

2.2 LA VALIDATION

Le second objectif de PETRI-S est la validation et la mise au point de systèmes de commande. La validation consiste à montrer que le comportement du système est conforme aux besoins exprimés par l'utilisateur ainsi qu'à vérifier que le système possède les propriétés liées à la vivacité et à la sûreté de fonctionnement.

Cette validation est basée sur l'analyse des propriétés du **RdPI global** décrivant le système de commande. L'approche que nous utilisons consiste à enrichir les résultats partiels d'une analyse statique du RdP autonome sur lequel est construit le RdPI par une simulation, car les extensions introduites dans les RdP autonomes tout en augmentant la puissance d'expression des RdP font en contrepartie perdre un certains nombre de résultats bien établis (cf.chapitre 2). En effet, ces extensions induisent des contraintes sur les évolutions possibles des marquages et modifient par conséquent les hypothèses d'application des méthodes d'analyse statiques des RdP autonomes.

2.2.1 L'APPROCHE DE VALIDATION RETENUE

Elle consiste d'abord à analyser le RdP sous-jacent et à montrer que ce réseau possède certaines propriétés caractéristiques du bon fonctionnement du système qu'il modélise en utilisant les méthodes d'analyse statiques des RdP autonomes. Suivant le résultat de cette analyse nous pouvons soit affirmer que le RdPI possède lui aussi ces propriétés soit recourir à la simulation pour surveiller la vérification ou non de ces propriétés. Il est clair que dans ce cas l'approche est à même de prouver la présence d'erreurs mais elle ne **suffit** pas pour montrer leur absence, à moins que l'interprétation associée au RdPI soit relativement simple et que le système soit destiné a avoir un fonctionnement périodique auquel cas la simulation peut être faite de façon exhaustive.

Parmi l'ensemble des méthodes d'analyse statique des RdP (cf chapitre 2), nous avons choisi d'utiliser l'analyse structurelle pour vérifier les propriétés du RdP sous-jacent. Ce choix se justifie par les points suivants :
1- l'analyse structurelle est basée sur la seule connaissance du graphe du réseau et donc de sa matrice d'incidence,

2- elle permet d'éviter l'explosion combinatoire des marquages à laquelle se heurte la méthode par énumération des marquages accessibles,

3- elle est indépendante de l'état initial, ainsi nous n'aurons pas à recommencer l'analyse à chaque fois que nous changeons d'état initial,

4- et surtout, elle permet de vérifier aisément les propriétés spécifiques.

Nous allons dans ce qui suit présenter quelles sont les propriétés que nous vérifions et comment.

2.2.2 LES PROPRIETES

L'approche de validation présentée ici recouvre deux classes de propriétés:
- les propriétés générales qui permettent de montrer que le système est borné et qu'il n'a pas de blocage. Ces propriétés doivent a priori être valables pour tout système de commande,

- les propriétés spécifiques qui permettent de montrer que le système rend bien le service pour lequel il a été conçu.

2.2.2.1 Propriétés générales

Soient R_I un RdPI décrivant un système de commande, définit par :
$R_I = (R, V, OP, C, \mu, \varphi)$,

$A(R, M_0)$: l'ensemble des marquages accessibles à partir du marquage initial M_0 du RdP autonome R = (PRE, POST, P, T),

$A(R_I, M_0)$: l'ensemble des marquages accessibles à partir du marquage initial M_0 du RdPI R_I pour un environnement représenté par le langage

$L \subseteq (P(E) - \varnothing)^*$ avec P(E) l'ensemble des parties de l'ensemble E des événements externes.

a- Propriété borné

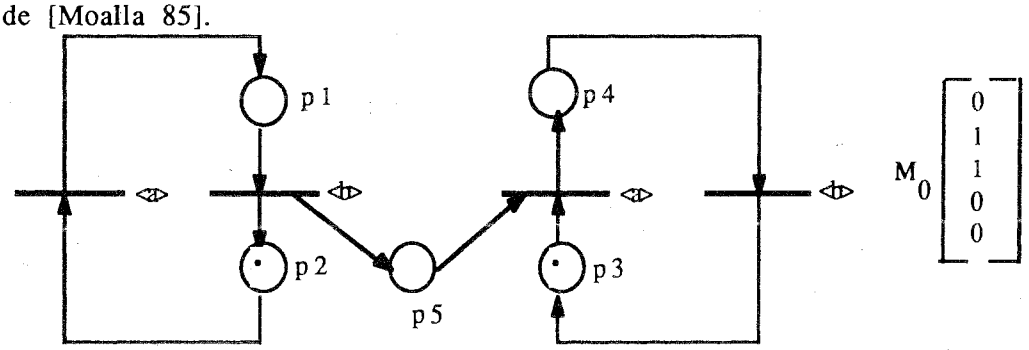
Cette propriété permet d'affirmer que le système a un nombre fini d'états. Un RdPI est borné pour un marquage initial M_0 et un environnement L ssi le marquage de chacune de ses places est borné pour tout marquage accessible depuis M_0.

La propriété borné est **indécidable** pour les RdPI, c'est-à-dire qu'il n'existe pas d'algorithme permettant de décider dans tout les cas si oui ou non le RdPI est borné. Cependant, il existe une condition **suffisante** pour prouver cette propriété.

Proposition

Pour qu'un RdPI $<R_I, L>$ soit borné pour un marquage M_0 il **suffit** que le RdP R sous-jacent soit borné pour le marquage M_0.

Cette condition n'est pas nécessaire. La preuve que cette condition est suffisante est triviale : le RdP R étant borné cela est équivalent à $A(R, M_0)$ est fini (propriété de Karp [Karp 72]) et puisque $A(R_I, M_0) \subseteq A(R, M_0)$ cela signifie que $A(R_I, M_0)$ est fini ce qui est équivalent à R_I est borné.
Pour monter que cette condition n'est pas nécessaire prenons le contre-exemple tiré de [Moalla 85].



Le RdP R est non borné pour le marquage M_0, car la place p5 peut accumuler un nombre illimité de marques. Par contre, le RdPI est borné.

Ainsi, la proposition énoncé ci-dessus sera utilisée pour la vérification du bornage d'un RdPI.

Technique de vérification

Le principe de cette technique de vérification consiste d'abord à déterminer si le RdP sous-jacent est borné ou non. Si le RdP est borné, alors le RdPI est borné.
Par contre, si le RdP est non borné, nous ne pouvons rien conclure quant au bornage du RdPI. Dans ce cas nous procédons à une vérification dynamique de cette propriété. L'outil demande à l'utilisateur la valeur de la borne (cette valeur doit être supérieure ou égale à 1) et vérifie au cours de la simulation du fonctionnement du RdPI que le marquage de chaque place reste toujours inférieur ou égal à la borne donnée. Remarquons toutefois que cette méthode de permet pas d'affirmer de façon absolue que le RdPI est borné mais elle permet de mettre en évidence des cas de violation de cette contrainte.
Pour vérifier que le RdP R est borné nous utilisons l'analyse structurelle dont nous allons rappeler le principe (cf chapitre 2 pour plus de détail).
La méthode d'analyse structurelle consiste à extraire du réseau des **composantes conservatives** ou **p-semi-flots** et à montrer que le réseau est lui même un p-semi-flot.
Pour montrer que le réseau R est borné nous procédons selon le principe suivant:

1- Nous déterminons la **base de p-semi-flots** du réseau. Pour cela, nous utilisons l'algorithme de Alaiwan [Alaiwan 85]. Cet algorithme est a notre avis beaucoup plus performant que celui de Berthomieu [Berthomieu 79] fondé sur la programmation linéaire en nombres entiers, et qui nécessite beaucoup de temps de calcul, car il faut résoudre au minimum autant de programmes linéaires qu'il y a de p-semi-flots dans la base.

2- Ensuite, nous vérifions si chaque place p du réseau est contenue dans au moins un support des vecteurs de la base. Si cette condition est vérifiée alors le réseau R est borné et par conséquent le RdPI est aussi borné. Par contre s'il existe des places qui ne satisfont pas à cette condition alors nous ne pouvons rien conclure quant au bornage du réseau R. En effet, cette condition est suffisante mais pas nécessaire.

Exemple

Soit le RdPI de la figure 5.3, décrivant le système de commande de la pompe de drainage dans la mine de charbon (cf. chapitre 4 § 2.3.2). Le système vérifie que ce réseau est borné.

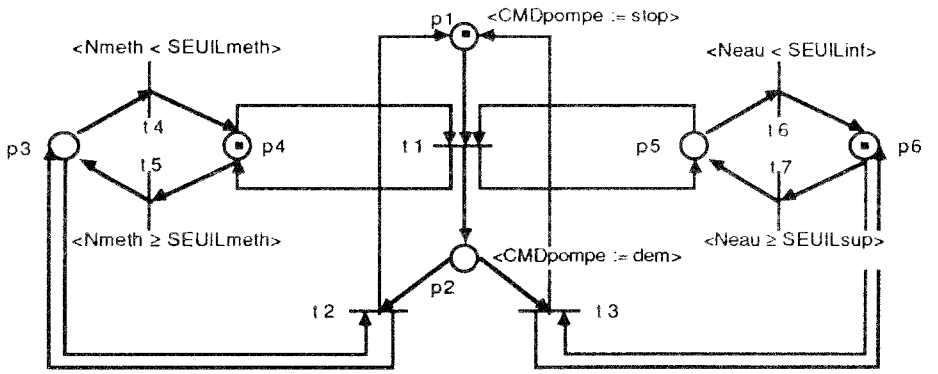


Figure 5.3

```
BASE DES P_SEMI_FLOTS
---------------------

P1      P2
P3      P4
P5      P6
************************
* le reseau est borne *
************************
```

b-Propriété de vivacité

La vivacité est une propriété très importante, elle permet de garantir l'absence de blocage total et partiel du système.

La propriété de vivacité est aussi une propriété indécidable pour les RdPI [Valette 76] [Moalla 85]. De plus, le fait que le RdP R sous-jacent soit vivant ne constitue **ni une condition nécessaire ni suffisante** pour que le RdPI soit vivant. En effet, si le RdP R est vivant cela signifie que pour tout marquage M appartenant à $A(R, M_0)$, il existe au moins une transition franchissable; cependant l'interprétation du réseau peut être telle que la réceptivité associée à cette transition soit fausse. Inversement, l'interprétation en rendant inaccessibles certains marquages, peut être telle que pour tout marquage M appartenant à $A(R_I, M_0)$ le RdPI possède au moins une transition franchissable, même si des blocages existent au niveau du RdP.

Pour plus d'éclaircissement, prenons les exemples suivants [Valette 76] [Moalla 85] :

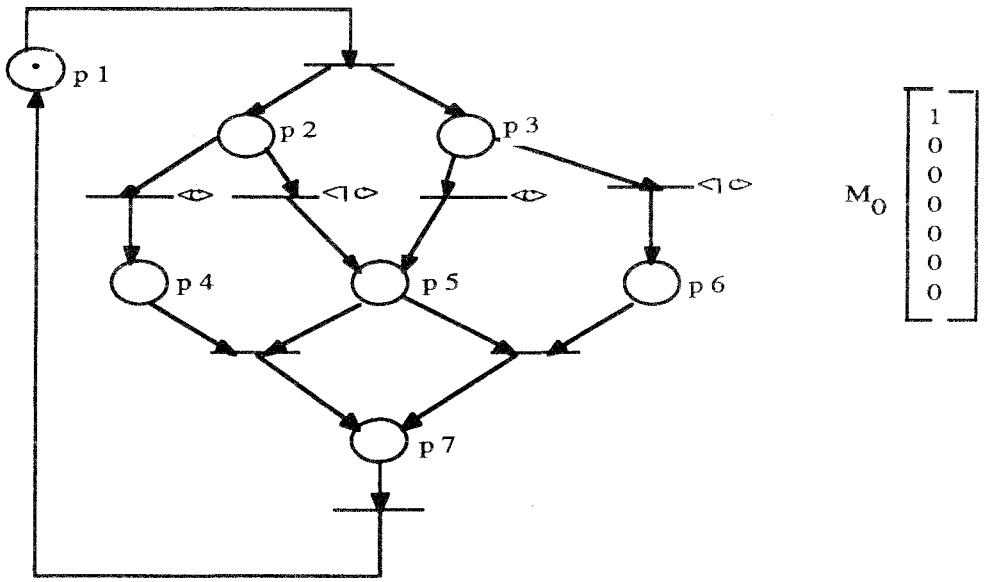


Figure 5.4

Le RdP R de la figure 5.4 est non vivant pour le marquage initial M_0 car le marquage M ne valide aucune transition.

$$M = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 0 \\ 0 \end{bmatrix}$$

Par contre, le RdPI est vivant puisque le marquage M n'est plus accessible.

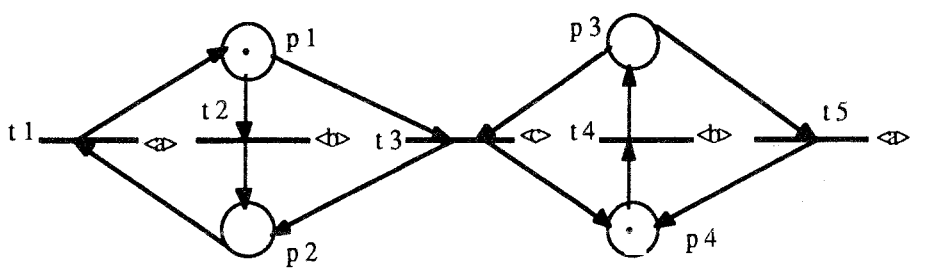


Figure 5.5

Le RdP R (cf. figure 5.5) est vivant alors que le RdPI n'est pas vivant pour ce marquage initial et pour l'ensemble des événements externes {a, b, c} : la transition t_3 ne peut jamais être franchie.

Ainsi nous ne vérifions pas la vivacité du RdP R puisque elle n'apporte aucune

preuve sur celle du RdPI R_I.

Nous proposons une approche qui permet de **détecter au cours de la simulation des situations de blocage total ou partiel**. Elle consiste à déterminer des structures du réseau correspondant à des **verrous** et à utiliser ces derniers pour détecter d'éventuels blocages au cours de la simulation.

Nous allons d'abord donner quelques définitions utiles pour la compréhension du principe de validation.

Définition 1 [Brams 83]

Un verrou est un ensemble non vide de places du réseau telles que l'ensemble de leurs transitions d'entrée est contenu dans l'ensemble de leurs transitions de sortie.

V est un verrou alors $\Gamma^{-1}(V) \subseteq \Gamma(V)$ *avec*

Γ : la fonction qui donne les successeurs d'un nœud

Γ^{-1} : donne les prédécesseurs.

Propriété des verrous

Une propriété fondamentale d'un verrou V est que si les transitions en sortie de V sont **infranchissables** pour un marquage M alors **elles le resteront** pour tout marquage atteint à partir de M. On dit alors que V est **déficient** pour le marquage M.

Définition 2 [Brams 83]

Un verrou V est déficient pour un marquage M si est seulement si :

$$\forall p \in V,\ \Gamma(p) \neq \varnothing \text{ et } M(p) < \min_{t \in \Gamma(p)} \{ v(p,t) \}$$

Corollaire

Soit V un verrou déficient pour un marquage M, alors pour tout marquage accessible à partir de M, V reste déficient.

La connaissance de l'ensemble des verrous permet de mettre en évidence un **éventuel blocage partiel** ou **total** du réseau. En effet, si en partant du marquage initial on trouve une séquence de franchissements qui rend déficient un verrou on peut alors conclure que le réseau est **non vivant**.

principe de vérification

Pour la détection de blocages du RdPI, l'approche que nous proposons procède donc en deux étapes:

1- déterminer l'ensemble des verrous du réseau.

2- Utiliser ces verrous au cours de la simulation pour vérifier s'il existe un ou plusieurs verrous déficients. Donc pour chaque marquage M résultant du tir de transitions franchissables nous vérifions s'il existe au moins un verrou qui est déficient pour ce marquage. Si oui, d'après le corollaire, ce verrou reste alors déficient pour tout marquage accessible à partir de M et par conséquent le RdPI décrivant le système de commande est non vivant. Par contre, si nous ne réussissons pas à trouver pendant la simulation de verrous déficients nous ne pouvons rien conclure quant à la vivacité du RdPI.

Exemple

Pour le RdPI de la figure 5.3, l'ensemble des verrous déterminer par le système est:

```
ENSEMBLE DES VERROUS DU RESEAU
------------------------------

        { P1  P2  }

        { P3  P4  }

        { P2  P3  P4  }

        { P5  P6  }

        { P1  P3  P4  P5  P6  }

        { P2  P5  P6  }
```

c- Propriété de persistance

Cette propriété permet de démontrer l'absence de **conflits effectifs** dans le réseau.

Définitions [Brams 83] [Moalla 85]

Définition 1

Soit T_C un ensemble de transitions. Ces transitions sont en **conflit structurel** si elles ont des places d'entrée communes. Si de plus il existe un marquage M accessible à partir du marquage initial M_0 qui valide ces transitions et tel que le franchissement de tout sous-ensemble $\{t_i\}$ de T_C conduit à un marquage pour lequel il existe une transition $t_j \in T_C - \{t_i\}$ qui ne soit plus franchissable alors l'ensemble T_C de transitions sont en conflit effectif.

Définition 2

Un RdPI est persistant pour un marquage M_0 et un environnement représenté par le langage L si et seulement si pour tout marquage M appartenant à $A(R_I, M_0)$, lorsque deux ou plusieurs transitions sont franchissables à partir de M alors toutes ces transitions peuvent être franchies dans n'importe quel ordre, en d'autre termes il n'existe pas de conflits effectifs.

Principe de vérification de la propriété de persistance

La procédure de vérification que nous proposons commence d'abord par vérifier qu'aucun couple de transitions n'a de places d'entrée en commun: absence de conflits structurels. Si cette condition est vérifiée alors le RdPI est **persistant**. Toutefois, cette condition n'est pas **nécessaire** car on peut résoudre les conflits structurels en associant aux places d'entrée communes des marquages qui valident toutes les transitions qui sont en conflit structurel, ou en associant aux transitions conflictuelles des réceptivités incompatibles. Ainsi dans le cas où il existe des conflits structurels, la procédure vérifie à chaque pas de la simulation si ces conflits sont effectifs ou non.

Notons que dans ce dernier cas, la procédure permet de montrer la présence d'éventuels conflits effectifs (situation de non persistance) mais jamais que le RdPI est persistant.

2.2.2.2 Propriétés spécifiques

Ces propriétés sont propres au système, elles permettent au concepteur de vérifier l'adéquation entre le service effectivement rendu par le système et le service à rendre, ainsi que de vérifier certaines propriétés liées à la sûreté de fonctionnement du système comme l'absence d'ordres contradictoires.

Les propriétés spécifiques sont exprimées par des **assertions** qui concernent, soit l'ensemble des états accessibles du système de commande, soit l'ensemble des séquences de commande exécutables soit les deux.

Nous allons présenter, dans un premier temps, la méthode utilisée pour vérifier ces propriétés lorsqu'elles sont exprimées par des assertions de marquage. Nous verrons ensuite la vérification des propriétés lorsqu'elles sont traduites par des séquences de franchissement.

a-Vérification des assertions de marquage

Soit A l'ensemble des assertions traduisant les propriétés spécifiques à vérifier.

A est un ensemble de relations (équations ou inéquations) entre les marquages des places du RdPI, que doit vérifier tout marquage M accessible à partir du marquage initial M_0.
Le principe de la méthode de vérification que nous utilisons consiste d'abord à vérifier ces assertions sur le RdP autonome sur lequel est construit le RdPI en utilisant la méthode d'analyse structurelle.
Si ces assertions sont vraies sur l'ensemble des marquages accessibles du RdP autonome,alors elles le sont également sur l'ensemble des marquages accessibles du RdPI, puisque $A(R_I, M_0) \subseteq A(R, M_0)$. Par contre, si ces assertions ne sont pas vraies sur l'ensemble des marquages accessibles du RdP autonome alors nous ne pouvons rien dire quant à leur véracité sur l'ensemble des marquages accessibles depuis M_0 du RdPI. Ces assertions peuvent être soit vraies donc corrigées par l'interprétation qui est associée au RdP autonome, soit fausses, et comme il n'existe pas d'algorithme qui permet de vérifier ces assertions de façon statique, on a recours alors à la simulation qui reste le seul moyen de validation. Le principe de la vérification par simulation consiste pour chaque marquage résultant du tir de transitions à vérifier si ces assertions sont vraies ou non.

Nous allons dans ce qui suit décrire la technique utilisée pour vérifier ces assertions sur le RdP autonome.

a1- technique de vérification des assertions sur le RdP

La vérification de ces assertions peut être faite selon deux approches:
1- La première approche consiste à extraire du RdP des invariants de places et à les fournir au concepteur. Celui ci vérifie lui même si ces invariants impliquent ou non les assertions qu'il désire vérifier. Ainsi le concepteur doit non seulement déterminer les assertions qui traduisent les propriétés spécifiques mais il doit aussi se charger de leur vérification à partir des invariants qu'il lui sont donnés.

2- La seconde approche, celle que nous utilisons permet de décharger le concepteur de la tâche difficile de vérification des assertions qui est confié à l'outil. Le concepteur doit seulement fournir l'ensemble des assertions à vérifier en procédant de manière suivante:
-il sélectionne dans le menu "vérification des propriétés" la fonction "Inv.places". Une fenêtre de saisie s'affiche sur l'écran de son poste de travail et

le concepteur se trouve sous le contrôle d'un éditeur de texte pour taper le texte de son assertion (les assertions sont introduites une par une). Lorsque le concepteur termine de taper le texte de l'assertion, celle-ci est analysée. Si cette assertion est syntaxiquement correcte elle est alors mémorisée dans des structures de données pour la vérification. Par contre, si une erreur a été détectée au cours de l'analyse syntaxique, cette erreur est signalée au concepteur pour la correction.
Une fois que le concepteur a fini de fournir l'ensemble des assertions, la procédure de vérification est lancée. Cette procédure est basée sur le principe qui consiste d'abord à déterminer une base B de **p-flots** du RdP.
rappelons qu'un p-flot f est un vecteur à composantes entières ($f \in \mathbb{Z}^{|P|}$) solution de l'équation $f^{tr}.C = 0$ avec C la matrice d'incidence du RdPI.

La propriété d'un p-flot f est que pour tout marquage $M \in A(R, M_0)$

$$f^{tr}.M = f^{tr}.M_0$$

c'est-à-dire que f définie un invariant de places.

La vérification des assertions sur $A(R, M_0)$ est ensuite ramenée à leur vérification sur l'ensemble des marquages solutions du système d'équations $B.M = B.M_0$, qui peut être réalisée sans énumérer les marquages de cet ensemble. En effet, il est démontré dans [Berthomieu 79] [Toudic 82] que $A(R, M_0) \subset \{M \in \mathbb{N}^{|P|} \wedge B.M = B.M_0\}$ et donc si ces assertions sont satisfaites sur cet ensemble alors elles sont également sur $A(R, M_0)$.

La vérification de l'ensemble des assertions A sur $\{M \in \mathbb{N}^{|P|} \wedge B.M = B.M_0\}$ est facile à réaliser, elle consiste à résoudre le système linéaire suivant:

$$B.M = B.M_0$$
$$\neg A$$
$$M \in \mathbb{N}^{|P|}$$

Si ce système n'admet pas de solution, c'est-à-dire qu'il n'existe pas de marquage pour lequel ces assertions ne sont pas vérifiées, on peut alors conclure que ces assertions sont satisfaites sur l'ensemble $\{M \in \mathbb{N}^{|P|} \wedge B.M = B.M_0\}$ et par conséquent sur $A(R, M_0)$.
Par contre, si le système admet une solution ($M \in \mathbb{N}^{|P|}$ pour lequel ces assertions ne sont pas satisfaites) nous ne pouvons pas en conclure que le système ne vérifie pas les propriétés spécifiques, une étude plus fine est alors nécessaire.

Ceci est dû à ce que l'existence de M solution de B.M = B.M_0 n'implique pas nécessairement que M est accessible à partir du marquage M_0 (ie. que M $\in$ A(R,M_0)). En effet, considérons le RdP de la figure 5.6

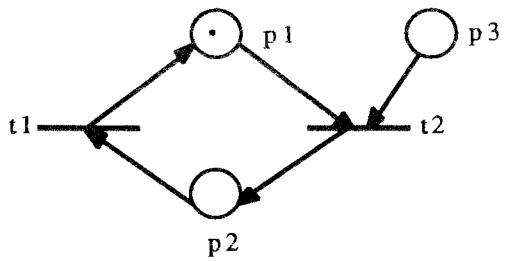


Figure 5.6

Ce réseau admet pour p_flot le vecteur (1 1 0) qui définit l'invariant de places $M(p_1) + M(p_2) = 1$; le marquage M

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

vérifie bien l'invariant de places; pourtant, M n'est pas accessible à partir de M_0 puisque la transition t_2 n'est pas franchissable.

Dans le cas où nous ne pouvons pas affirmer de façon formelle que les assertions sont vérifiées sur A(R, M_0), nous procédons à leur vérification par simulation.

b-Vérification des propriétés exprimées par des séquences de tirs

Ces propriétés sont très intéressantes, elles permettent au concepteur de vérifier par exemple que l'ordre d'enchaînement de certaines actions est bien l'ordre voulu, ou de vérifier l'absence de famine ("fairness"). Le concepteur fournit les séquences de tirs de transitions à l'outil PETRI-S de la même manière que pour les assertions de marquages et lors de la simulation du fonctionnement du RdPI, l'outil vérifie si les tirs des transitions respectent les séquences de tirs du concepteur ou non. Cette vérification se fait uniquement de façon dynamique.

2.3 L'EVALUATION DES PERFORMANCES

La complexité croissante des systèmes de production et l'exigence de compétitivité toujours accrue rendent nécessaire l'évaluation des performances de ces systèmes dès les premières phases de leur conception. L'évaluation a pour

objectif de déterminer les meilleurs compromis pour réduire les encours et gérer de façon optimale l'utilisation des ressources selon les profils de la production. PETRI-S peut être utilisé à cette fin.
Il permet de guider efficacement le concepteur dans le choix de stratégies de pilotage et dans le dimensionnement du système (nombre de chariots, nombre de postes de travail, capacité des zones de stockage...).

En utilisant PETRI-S, l'évaluation des performances d'un système est conduite comme indiqué dans la figure 5.7

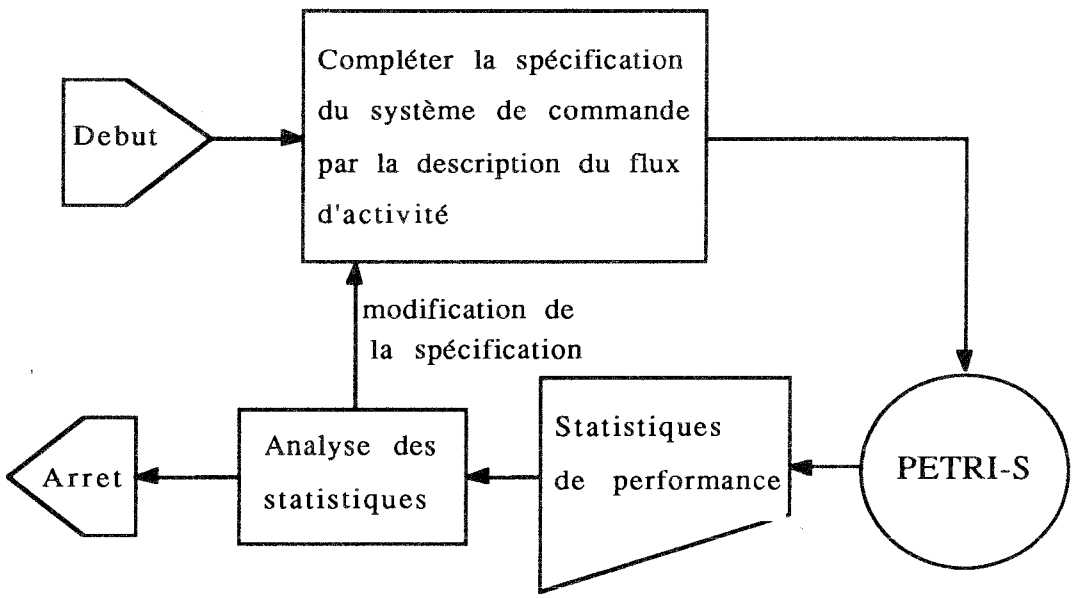


Figure 5.7

Le concepteur complète la spécification du système de commande par la description du flux d'activité. Ensuite, il utilise cette description au cours de la simulation du système de commande et de son environnement. Des mesures sont effectuées automatiquement pendant cette simulation de façon à fournir des statistiques sur les files d'attente du réseau décrivant le flux d'activité.

Pour chaque file f PETRI-S calcule :

T(f,n) : durée totale pendant laquelle la file f a contenu n transactions
R(f) : la somme totale des durées passées dans la file f par l'ensemble des transactions (sans tenir compte des transactions non encore sorties)

N(f) : nombre total de transactions ayant quitté la file f
Tmax(f) : durée maximale d'attente dans la file f

Tmin(f) : durée minimale d'attente dans la file f
Tmoy(f) : durée moyenne de résidence dans la file f

$$Tmoy(f) = R(f) / N(f)$$

Lmax(f) : longueur maximale de la file
Lmin(f) : longueur minimale de la file
Lmoy(f) : longueur moyenne de la file

$$Lmoy(f) = (1*T(f,1) + 2*T(f,2) + ...) / horloge$$

(l'horloge fournit la valeur du temps)

Le débit moyen : $D(f) = N(f) / horloge$

Notons que les deux premières grandeurs sont inaccessibles à l'utilisateur.

Le concepteur peut faire tourner plusieurs variantes du système, chacune correspondant à une stratégie de conduite du système. Ces variantes seront comparées au point de vue des performances par analyse des différentes valeurs fournies et le concepteur pourra alors choisir la meilleure stratégie.

3 ENVIRONNEMENT GENERAL DE PETRI-S

Avant d'aborder de façon plus détaillée la description de PETRI-S, nous allons d'abord présenter l'environnement de conception élaboré au cours du projet dans lequel cette étude a été effectuée.
L'objectif de cet environnement est de contribuer à la maîtrise de la complexité de conception des systèmes de commande répartis, en offrant un ensemble d'outils (cf. figure 5.7) comprenant :

- des outils d'édition évolués,
- un outil de décomposition,
- et un outil de simulation.

Ces outils assistent le concepteur tout au long de son travail.

3.1 LES OUTILS D'EDITION EVOLUES

Ils permettent la construction et l'édition aisée du RdPI décrivant le comportement externe du système. Trois outils ont été proposés : un outil de

construction des RdPI, un analyseur de la description textuelle des RdPI et un éditeur graphique des RdPI.

L'outil de construction

La vocation de cet outil est d'aider le concepteur à construire progressivement le RdPI décrivant le comportement externe du système conformément à la méthode proposée dans [Tankoano 88].

L'avantage principal de cet outil est qu'il permet de garantir par construction les bonnes propriétés (vivacité et invariants de places) du réseau décrivant le système et d'éviter au concepteur les étapes délicates de validation. Son seul inconvénient est sa limitation à une classe de réseaux.

L'analyseur du langage de description des RdPI

Le langage de description des RdPI est de type PASCAL, et emprunte un certains nombre de constructions au langage DESY [Albukerque 82] utilisé comme langage d'entrée d'automates programmables. Il permet de décrire :

- le RdP décrivant le système de commande;
- l'ensemble des variables, des événements, des conditions et des actions;
- l'interprétation, c'est-à-dire le lien entre les événements, les conditions, les actions et les places et les transitions du réseau;
- et le flux d'activité.

L'analyseur se trouvant dans la figure 5.7 permet d'analyser la spécification du système décrite dans ce langage et de la traduire en une représentation interne qui sera exploitée par l'outil de simulation et/ou l'outil de décomposition.

L'éditeur graphique des RdPI

Le rôle de cet outil est le même que celui assigné au langage de description à savoir l'édition du RdPI global décrivant le système, sauf que dans ce cas l'édition est faite de façon plus agréable et visuelle (graphique).

3.2 L'OUTIL DE DECOMPOSITION

Cet outil assiste le concepteur dans la conception de la structure interne de son système.

En s'appuyant sur les règles de décomposition présentées dans [Tankoano 88], il permet de décomposer le RdPI décrivant le comportement externe du système en

un ensemble de modules communicants (chaque module est décrit par un RdPI) et pouvant être répartis sur un réseau de processeurs.

3.3 L'OUTIL DE SIMULATION

Comme nous l'avons déjà indiqué dans la section 2, cet outil permet de simuler conjointement le système de commande et l'environnement commandé dans le but d'aider l'utilisateur dans l'expression de ses besoins, de valider et tester le système de commande et de dimensionner correctement l'atelier.

L'apport de cet outil est qu'il conduit à des systèmes sûrs et conformes aux besoins et donc à une meilleure satisfaction de l'utilisateur.

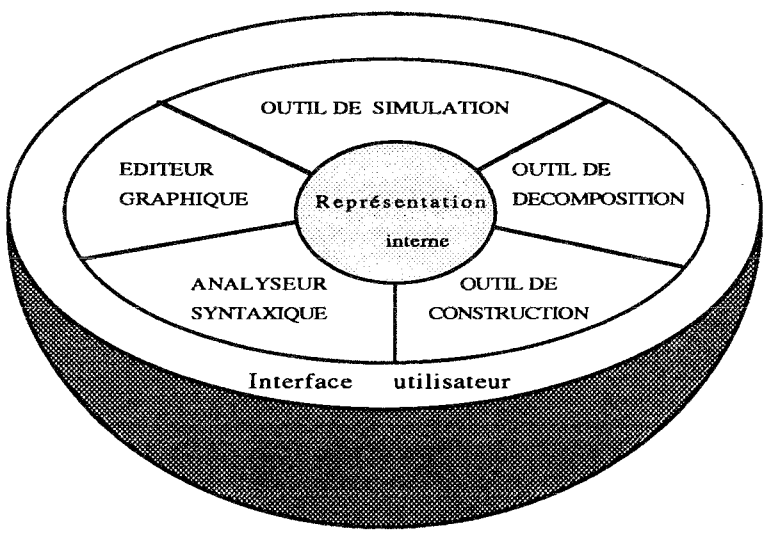


Figure 5.7 : environnement de conception

Dans ce qui suit nous allons aborder la présentation du langage de description des RdPI, de l'éditeur graphique et de l'outil de simulation, les autres outils (outil de construction et outil de décomposition) ont été décrit dans [Tankoano 88].

4 LANGAGE DE DESCRIPTION DES SPECIFICATIONS

Dans cette partie, nous présentons le langage de description de RdPI décrivant le comportement du système de commande et du modèle décrivant le

flux d'activité (cf. chapitre 4 pour la description de ces deux modèles).
La proposition de ce langage déclaratif et de haut niveau permet une description aisée de la spécification en découplant les problèmes liés à l'édition de ceux liés à la vérification de la cohérence de la spécification.
L'utilisateur décrit à l'aide d'un éditeur de texte classique (type EMACS [Stallman 84]) la spécification par des RdPI de son système. Il peut ainsi disposer de toutes les facilités et souplesses d'édition offertes par l'éditeur. Cette spécification est ensuite analysée syntaxiquement et sémantiquement par l'analyseur du langage. Au cours de cette analyse, des erreurs peuvent être détectées et signalées à l'utilisateur pour la correction.
Lorsque la spécification est jugée syntaxiquement correcte, elle est traduite en une représentation interne qui sera utilisée par le simulateur.

Nous allons, dans un premier temps décrire les éléments du langage, nous présenterons ensuite la réalisation de l'analyseur.

4.1 ELEMENTS LEXICAUX

4.1.1 JEU DE CARACTERES

Les spécifications sont construites à partir du jeu de caractères suivant :
- les 26 lettres majuscules : A → Z
- les 26 lettres minuscules : a → z
- les 10 chiffres décimaux : 0 → 9
- les caractères spéciaux :. + - * / ^? ! = <> [ ] $ () <= >= := { }
- le caractère espace.

Les minuscules sont équivalentes aux majuscules dans les identificateurs et les mots-clés.

4.1.2 LES DELIMITEURS

Ce sont :
- les caractères spéciaux définis au dessus
- une suite de un ou de plusieurs caractères espace.

4.1.3 LES MOTS-CLES

Ils sont listés dans l'annexe A.
Ces mots-clés ne peuvent en aucun cas être utilisés pour désigner autre chose

que ce pour quoi ils ont été définis et en particulier ils ne peuvent pas être utilisés comme identificateur.

4.1.4 LES IDENTIFICATEURS

Les identificateurs sont des noms qui servent à nommer les éléments de la description du système : place, transition, variable, événement, ...
Un identificateur commence par une lettre qui peut être suivie par un nombre quelconque de lettres ou de chiffres.

4.1.5 LES COMMENTAIRES

Il est possible d'insérer des commentaires en tout point de la spécification. Un commentaire commence par le caractère "{" et s'étend jusqu'au prochain caractère "}". Les commentaires sont d'une grande utilité pour documenter les différentes spécifications.

4.2 STRUCTURE D'UNE SPECIFICATION

La structure d'une spécification comprend quatre parties principales :

- La première partie comprend la description du graphe, c'est-à-dire le RdP décrivant la partie commande. Elle débute par le mot-clé **RDP** et se termine par le mot-clé **FRDP**.

- La deuxième partie comprend la déclaration de l'environnement : déclarations des variables du système, des opérations agissant sur les variables et des conditions et événements qui conditionnent et synchronisent les évolutions du réseau décrit au dessus.
Cette partie est délimitée par les mots-clés **ENVIRONNEMENT** et **FENVIRONNEMENT**.

- La troisième partie contient la définition de l'interprétation associée au RdP. Elle spécifie l'ensemble des liens entre les événements, les conditions et les transitions et les liens entre les opérations et les places du réseau.
Cette partie est délimitée par les mots-clés **INTERPRETATION** et **FINTERPRETATION**.

- La dernière partie comprend la description du flux d'activité. Elle est introduite par le mot-clé **RDFA** et se termine par le mot-clé **FRDFA**. Cette partie peut être vide dans le cas où l'outil n'est pas utilisé pour l'aide au dimensionnement.

La figure 5.8 illustre la structure d'une spécification d'un système. Cette spécification débute par le mot-clé **RDPI** suivi du nom donné au système de commande. La fin de la spécification est repérée par le mot-clé **FRDPI** suivi du nom du système et d'un point.

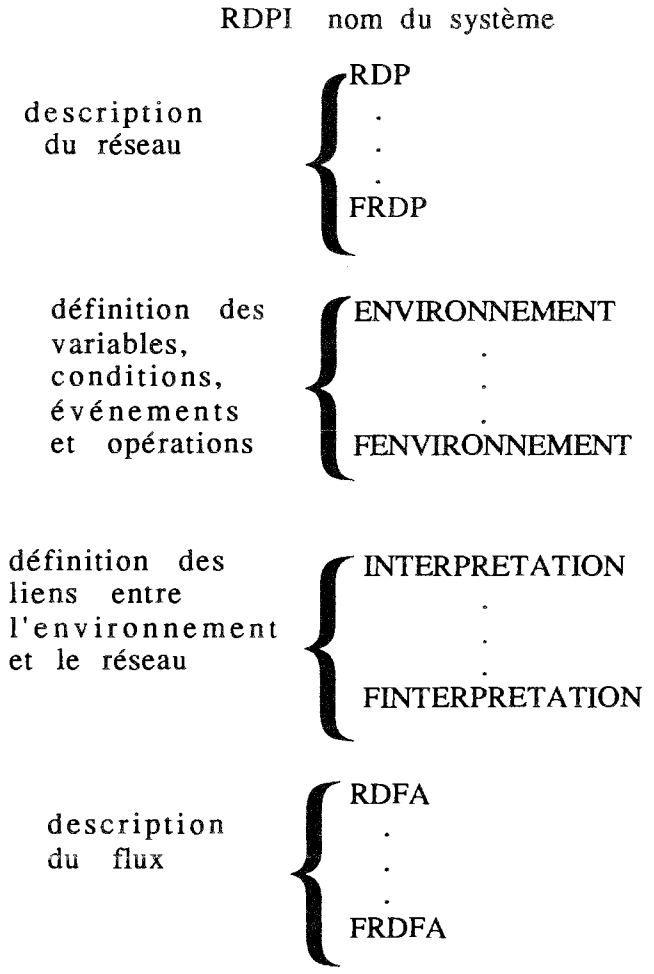


figure 5.8 : structure d'une spécification

4.3 DESCRIPTION DES DIFFERENTES PARTIES

4.3.1 PARTIE DESCRIPTION DU RESEAU

Cette partie comporte deux sections :

- une section de déclaration des nœuds
- une section de déclaration des arcs.

La première section, annoncée par le mot-clé **NŒUDS** est utilisée pour définir tous les nœuds (places et transitions) appartenant au réseau décrivant le système.
La déclaration de ces nœuds consiste à associer à chaque nœud un identificateur (nom du nœud) et un type. Ce dernier peut être le mot-clé **PLACE** ou **TRANSITION** suivant que l'identificateur décrit un nœud place ou un nœud transition. De plus pour chaque nœud de type PLACE est associé un entier optionnel, placé entre parenthèses, qui représente le marquage initial de la place. Par défaut nous prenons le marquage nul.

La deuxième section, introduite par le mot-clé **ARCS** comprend la déclaration des arcs qui relient les nœuds déclarés au préalable dans la première section.
Les arcs sont définis par rapport aux transitions: Pour chaque transition nous spécifions la liste des places en entrée et la liste des places en sortie. Le mot-clé **ENTREE** suivi de l'identificateur de la transition introduit la déclaration des places en entrée. Cette liste contient les identificateurs des places précédé chacun par un entier optionnel, qui indique le poids de l'arc qui relie la place à la transition. Le poids n'apparaît explicitement que lorsqu'il est supérieur à 1 (la valeur 1 est prise par défaut).
De la même manière le mot-clé **SORTIE** annonce la déclaration de la liste des places en sortie.

Syntaxe

NB: Les symboles suivants sont les méta-symboles de la notation BNF (Backus-Naur Form) et non pas des symboles du langage.

::= | { }

Les accolades indiquent une éventuelle répétition des symboles qu'elles englobent, zéro, une ou plusieurs fois.
En général S ::= { R } est une forme abrégée de S ::=vide | S R

<partie description du réseau> ::= **RDP** <corpsrdp> **FRDP** ;
<corpsrdp> ::= <partie déclaration des nœuds>
<partie déclaration des arcs>
<partie déclaration des nœuds> ::= **NŒUDS** <déclaration de nœuds>
{; <déclaration de nœuds> } ;
<déclaration de nœuds> ::= <définition nœud>
{, <définition nœud> } : <type nœud>
<définition nœud> ::= <identificateur> | <identificateur> (<constante>)
<type nœud> ::= **PLACE | TRANSITION**
<constante> ::= <chiffre> { <chiffre> }

<partie déclaration des arcs> ::=**ARCS** <déclaration d'arc> {;<déclaration d'arc>};
<déclaration d'arc>::=<type arc><identificateur de transition> = <liste de places>
<type arc> ::= **ENTREE | SORTIE**
<identificateur de transition> ::= <identificateur>
<liste de places> ::= <nplace> {, <nplace>}
<nplace> ::= <constante>*<identificateur de place> |<identificateur de place>
<identificateur de place> ::= <identificateur>

Exemple

Considérons le RdP modélisant l'exemple du producteur-consommateur (cf. figure 5.9), la partie description du réseau est la suivante :

RDP
NŒUDS
attente-prod(1), produire : PLACE;
attente-cons(1), consommer : PLACE;
tampon-vide(n), tampon-plein : PLACE;
perm-prod, fin-prod, perm-cons, fin-cons : TRANSITION;
ARCS
ENTREE perm-prod = attente-prod, tampon-vide;
SORTIE perm-prod = produire;
ENTREE fin-prod = produire;
SORTIE fin-prod = attente-prod, tampon-plein;
.
. même chose pour les transitions concernant le processus
. consommateur
FRDP

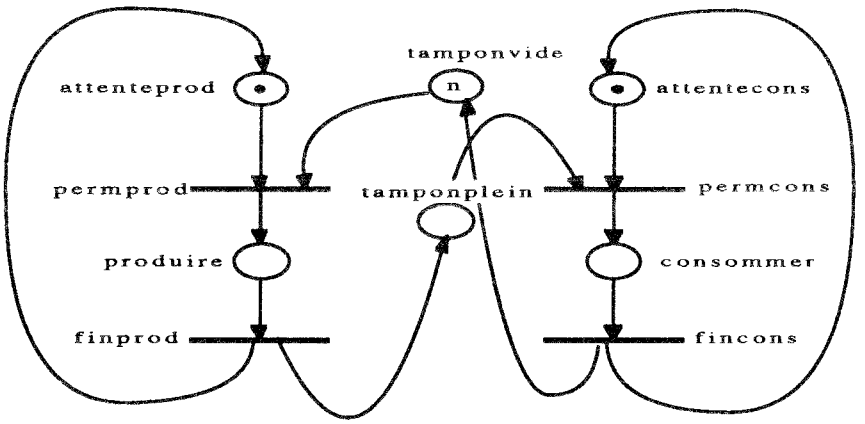


Figure 5.9

4.3.2 PARTIE DECLARATION DE L'ENVIRONNEMENT

La notion d'environnement est utilisée pour réunir sous le même nom un ensemble de déclarations de variables, de conditions, d'événements et d'actions.

La syntaxe de cette partie est la suivante :

<partie déclaration environnement> ::= **ENVIRONNEMENT**
<corpsenvironnement>
FENVIRONNEMENT ;

<corpsenvironnement> ::= <partie déclaration de variables>
<partie déclaration de conditions>
<partie déclaration d'événements>
<partie déclaration d'actions>

4.3.2.1 Déclaration des variables

La section de déclaration de variables débute par le mot-clé VARIABLES. Elle comprend la déclaration des variables d'entrée/sortie qui servent à la communication entre le système de commande et le procédé et la déclaration de variables internes au système de commande. Les variables internes peuvent être des variables calculées (compteurs, variables de synchronisation, ...) ou des temporisations.

La syntaxe:

<partie déclaration de variables> ::= **VARIABLES**
<déclaration de variables d'entrée>
<déclaration de variables de sortie>
<déclaration de variables calculées>
<déclaration de temporisations>

a- Déclaration des variables d'entrée

La rubrique de déclaration de variables d'entrée est introduite par le mot-clé ENTREE. Celui-ci permet de préciser la classe de ces variables afin de vérifier lors de la déclaration des conditions, événements, actions et interprétation que ces variables sont utilisées correctement. En effet, les variables d'entrée ne peuvent être utilisées qu'en consultation et n'apparaissent par conséquence que dans les conditions et événements associés aux transitions du réseau, ou dans les expressions apparaissant dans les affectations.
La déclaration de ces variables consiste à associer un identificateur et un type à chaque variable. Le type pouvant être entier, réel ou booléen.

Syntaxe :

<déclaration de variables d'entrée> ::= ε | **ENTREE** <liste de variables>
{; <liste de variables> };

<liste de variables> ::= <identificateur> {, <identificateur> } : <type>

<type> ::= **ENTIER** | **REEL** | **BOOLEEN**

b- Déclaration des variables de sortie

Cette déclaration est analogue à celle des variables d'entrée; elle est introduite par le mot-clé SORTIE. Par ailleurs ces variables n'apparaissent qu'au niveau des actions associées aux places du réseau (envoyer des ordres vers le procédé).

La syntaxe :

<déclaration de variables de sortie> ::= ε | **SORTIE** <liste de variables>
{; <liste de variables> } ;

c- Déclaration des variables calculées

La déclaration des variables calculées commence par le mot-clé CALCULEES suivi des identificateurs des variables et de leur type. Ces variables peuvent être également de type entier, réel ou booléen.

Aucun restriction n'est imposée sur l'utilisation de ces variables, elles peuvent apparaître dans les conditions et événements ainsi que dans les actions.

La syntaxe

<déclaration de variables calculées> ::= ε | **CALCULEES** <liste de variables>
{; <liste de variables> } ;

d- Déclaration des variables de temporisation

Cette déclaration débute par le mot-clé TEMPORISATION suivi des identificateurs des variables et de leur type. Celui-ci ne peut être que le type booléen.

Les temporisations sont manipulées par des opérations prédéfinies: armertemporisation et désarmertemporisation (cf. chapitre 4 §2) au niveau des places et tester au niveau des transitions.

La syntaxe

<déclaration de temporisations> ::= ε | **TEMPORISATION** <liste de variables>
{; <liste de variables> } ;

Exemple

Considérons l'exemple de la pompe de drainage dans une mine de charbon (cf. chapitre 4 § 2.3.2) dont le RdPI de la figure 5.3 décrit le comportement de la pompe en fonction du niveau de l'eau et celui de méthane. La déclaration de quelques variables de ce système est la suivante :

```
VARIABLES
    ENTREE
        Neau, Nmeth, Nco2 : REEL ;
        Reqmeth, Reqco2 : BOOLEEN ;
    SORTIE
        ALMmeth, CMDpompe : BOOLEEN ;
        Repmeth, Repco2 : REEL ;
    CALCULEES
        seuilinf, seuilsup : REEL ;
    TEMPORISATION
        temp1, temp2 : BOOLEEN ;
```

4.3.2.2 Déclaration des conditions et événements

Les conditions et événements associés aux transitions peuvent être décrits soit dans la partie interprétation soit dans la partie environnement et repérés donc par des identificateurs auxquels on fait référence dans la partie interprétation. La description des conditions et événements dans la partie environnement donne à l'utilisateur la possibilité de définir de nouveaux événements et conditions à partir de conditions et événements déclarés auparavant en utilisant seulement leur identificateur et sans avoir donc à réécrire leur expressions.

a- Déclaration des conditions

Le mot-clé CONDITIONS annonce la déclaration des conditions. Celle-ci associe à chaque condition un identificateur (identificateur repérant la condition) et une expression booléenne.
Une expression booléenne est une expression dont les opérandes sont de type booléen ou sont des expressions de comparaison dont le résultat est de type booléen.
Les opérateurs s appliquant aux opérandes booléens sont :

-l'opérateur unaire PAS : la négation
- les opérateurs binaires ET : la conjonction
OU : la disjonction

L'expression est après analyse syntaxique, transcrite en notation polonaise postfixée.

b-Déclaration des événements

Introduite par le mot-clé EVENEMENTS, elle sert à définir les événements qui synchronisent l'évolution du système à celle de l'environnement avec lequel il interagit. Ces événements traduisent les changements d'état des grandeurs physiques. Leur déclaration consiste à associer à chaque événement un identificateur qui sera utilisé dans l'interprétation et une expression qui permet de l'évaluer.
L'expression peut être de la forme :
↑(expression) : pour exprimer le passage de faux à vrai de expression
↓(expression) : pour exprimer le passage de vrai à faux de expression

ou une expression composée qui est définie à partir d'événements décrits au préalable et d'opérateurs (ET, OU, PAS)

La syntaxe des déclarations des conditions et événements est la suivante:

<partie déclaration de conditions>::= ε|**CONDITIONS** <déclaration de condition> {; <déclaration de condition> } ;

<déclaration de condition> ::= <identificateur> = <expression>

<partie déclaration événements>::=ε|**EVENEMENTS** <déclaration événement> {; <déclaration événement> } ;

<déclaration événement>::= <identificateur> = <expression>

<expression>::= <expression simple> |<expression simple> <opérateur de comparaison> <expression simple>

<opérateur de comparaison> ::= = |<> | < | <= | > | >=

<expression simple> ::= <terme>|<expression simple> <opérateur additif> <terme> | <signe> <terme>

<opérateur additif> ::= + | - | **OU**

<terme> ::= <facteur> | <terme> <opérateur multiplicatif> <facteur>

<opérateur multiplicatif> ::= * | / | **ET**

<facteur> ::= <identificateur> | <constante> | (<expression>) | **PAS** <facteur>| \^ <facteur> | \! <facteur>

<signe> ::= + | -

NB: les symboles \^ et \! sont utilisés pour représenter respectivement le front montant ↑ et le front descendant ↓.

Exemple

Reprenons l'exemple de la pompe de drainage, et donnons quelques déclarations de conditions et d'événements.

CONDITIONS

cond1 = Neau < seuilinf ;

EVENEMENTS

evetalm = ↑ ALMmeth ;

4.3.2.3 Déclaration des actions

Le mot-clé ACTIONS annonce la déclaration des actions associées aux places du réseau représentant le système de commande. Ces actions peuvent être des actions de commande qui agissent sur l'environnement par l'intermédiaire des

variables de sortie en vue de le contraindre à respecter un comportement donné, ou des actions de calcul.

Cette déclaration consiste à associer à chaque action un identificateur et un corps. Le corps est formé soit d'une instruction soit d'une séquence d'instructions.

Une séquence d'instructions est un ensemble d'instructions qui doivent être exécutées dans l'ordre où elles apparaissent, les mots-clés DEBUT et FIN délimitent la séquence.

Quatre types d'instructions ont été proposés:

- l'affectation: elle permet d'affecter à une variable calculée ou de sortie une valeur calculée,

- l'activation d'une temporisation: elle affecte la valeur vraie à une variable de temporisation après l'écoulement du temps spécifié,

- la désactivation d'une temporisation : désactive l'effet de l'activation,
- et la conditionnelle qui spécifie que si une condition est vraie, les instructions qui suivent le mot-clé ALORS sont exécutées.

La syntaxe

<partie déclaration actions>::= ε| **ACTIONS** <identificateur> = <corpsaction>
{; <identificateur> = <corpsaction> } ;

<corpsaction> ::= <instruction> | <blocinstructions>

<blocinstructions> ::= **DEBUT** <instruction> {, <instruction> } **FIN**

<instruction> ::= <instruction simple> | <instruction si>

<instruction si> ::= **SI** <expression> **ALORS** <instruction simple>
{; <instruction simple> } **FSI**

<instruction simple> ::= <affectation> | <instruction activertemp> |
<instruction desactivertemp>

<affectation> ::= <variable> := <expression>

<variable> ::= <identificateur>

<instruction activertemp> ::= **ARMERTEMPORISATION**
(<identificateur>,<constante>)

<instruction desactivertemp> ::= **DESARMERTEMPORISATION**
(<identificateur>)

Exemple de déclarations d'actions

ACTIONS

act1 = CMDpompe := vrai ;

act2 = ARMERTEMPORISATION (temp1, 10) ;

act3 = SI Nmeth >= seuilmeth ALORS ALMmeth := vrai FSI ;

4.3.3 PARTIE INTERPRETATION

Il s'agit dans cette troisième partie, d'associer aux transitions du réseau décrit dans la première partie des réceptivités (conditions et/ou événements) et aux places des actions.

L'association de réceptivités aux transitions est réalisée au moyen de l'instruction RECEPTIVITE suivie de la liste des transitions ayant une même réceptivité et de la description de celle-ci. Cette description comporte la définition de l'événement et/ou la définition de la condition.

La définition de l'événement comprend :

1- la politique de communication associée à cet événement, cette politique spécifie comment seront consommer les occurrences de cet événement au moment du franchissement des transitions synchronisées à cet événement. L'utilisateur choisit une politique parmi l'ensemble des cinq politiques prédéfinies suivantes : la politique rafraîchie, strictement rafraîchie, par bascule, égalité et aléatoire (voir chapitre 4 pour la définition de ces politiques).

2- L'expression de l'événement ou l'identificateur de l'événement dans le cas où celui-ci a été déjà défini dans la partie environnement.

De même, la définition de la condition comprend soit l'expression booléenne soit l'identificateur de la condition si celle-ci a été définie dans la partie environnement.

L'instruction ACTION p_{i1}, p_{i2}, ...= [acti1, acti2, ...] permet d'associer aux places du réseau des actions impulsionnelles.

Alors que l'instruction ACTION p_{j1}, p_{j2}, ...= [actj1, actj2, ...]* permet d'associer aux places p_{j1}, p_{j2}, ... des actions continues.

Syntaxe

<partie interprétation> ::= ε|**INTERPRETATION** <corpsinterprétation>
FINTERPRETATION

<corpsinterprétation> ::= <interprétation transition>| <interprétation place>
{; <interprétation transition>| <interprétation place>} ;

<interprétation transition> ::= ε| **RECEPTIVITE** <définition réceptivité>

<définition réceptivité> ::= <identificateur> {,<identificateur>} = <association
événement> <association condition>

<association événement>::= ε|**SUR CONSOMMATION** <typcommunication> **DE**
OCCURRENCE DE <expression>| **SUR OCCURRENCE DE** <expression>

<typcommunication>::=**RAFRAICHIE**|**STRICTEMENTRAFRAICHIE**|**BASCULE**|
EGALITE | **ALEATOIRE**

<association condition> ::= ε | **SI** <expression>

<interprétation place> ::= ε|**ACTION** <définition place>

<définition place> ::= ε|<identificateur> {,<identificateur>} =
[<corpsaction>|<identificateur action>{,<identificateur action>}]* |
[<corpsaction>| <identificateur action> {, <identificateur action>}]

<identificateur action> ::= <identificateur>

Exemple

Considérons le RdPI de la figure 5.3, la description de l'interprétation de ce réseau est la suivante :

INTERPRETATION
RECEPTIVITE t6 = si cond ;
ACTION p1 = [act1] ;
ACTION p2 = [CMDpompe := faux] ;

4.3.4 PARTIE DESCRIPTION DU FLUX D'ACTIVITE

Cette partie comprend la description du réseau de files d'attente décrivant le flux d'activité du système. Ce réseau, est après analyse syntaxique et sémantique traduit en des structures de données internes qui seront utilisées pendant la simulation.

La simulation consistera à faire évoluer ce réseau en fonction de l'évolution du système de commande et à collecter des mesures pour déterminer des statistiques dans le but d'aider l'utilisateur dans ses choix de dimensionnement.

La description du réseau modélisant le flux d'activité comprend trois sections :

la section de déclaration des files d'attente, la section de déclaration des classes de transactions et celle de description des écoulements de flux.

Syntaxe

<partie déclaration du flux> ::= ε | **RDFA** <corpsrdfa> **FRDFA** ;

<corpsrdfa> ::= <partie déclaration des files>
<partie déclaration des transactions>
<partie description du flux>

4.3.4.1 Déclaration des files d'attente

Le mot clé FILES introduit la section de déclaration des files. La déclaration consiste à fournir pour chaque file un identificateur (nom de la file), une politique de gestion et une capacité.
La politique de gestion permet de spécifier comment seront prises en compte les transactions au moment du traitement. La politique de gestion peut être de type
FIFO (premier arrivé, premier servi),
LIFO (dernier arrivé, premier servi),
PRIOR (gestion selon l'ordre de priorité. Dans le cas où deux transactions ont la même priorité alors c'est la première arrivée qui sera prise en considération),
ou RAND (choisi de façon aléatoire).

La capacité de la file indique le nombre maximum de transactions que la file peut contenir.

NB : Dans le cas où l'utilisateur omet de préciser ces deux paramètres, les valeurs FIFO et une capacité infinie sont prises par défaut.

Syntaxe

<partie déclaration de files>::= **FILES** <déclaration de files>
{ ; <déclaration de files> } ;

<déclaration de files> ::= <identificateur> {, <identificateur>} :
PG = <politique de gestion>, **C** = <capacité>

<politique de gestion> ::= FIFO | LIFO |PRIOR |RAND

<capacité> ::= <nombre entier> | INFINI

<nombre entier> ::= <chiffre> {<chiffre>}

Exemple

Considérons l'exemple de la section de travail (cf. chapitre 4 § 4.2), nous supposons que la section est composée de 3 postes de travail qui permettent l'usinage de 3 types de pièces m1, m2 et m3. La déclaration des files à l'aide de ce langage est la suivante :

FILES

fm1pp1, fm2pp1, fm3pp1 : **PG = FIFO, C =INFINI** ; {déclaration des 3 files des pièces m1,m2 et m3, qui sont associées à la place pp1 du réseau de flux d'activité (cf. figure 4.11) }

fm1pp2, fm2pp2, fm3pp2 : **PG = FIFO, C =INFINI** ; {idem pour la place pp2}

.

.

fm1pp6, fm2pp6, fm3pp6 : **PG =RAND, C=INFINI** ; {idem pour la place pp6}

4.3.4.2 Déclaration des transactions

Le mot-clé TRANSACTIONS annonce la déclaration des différentes classes de transactions. Chaque transaction peut être individualisée en lui assignant des attributs qui la décrivent et la caractérisent.

La déclaration d'une classe de transactions consiste en la donnée de l'identificateur de la classe (nom qui sert à désigner toutes les transactions de cette classe) et la liste des attributs. Chaque attribut est défini par un identificateur (nom de l'attribut) et un type. Les valeurs de ces différents attributs permettent de distinguer les transactions d'une même classe.

Syntaxe

<partie déclaration des transactions> ::= **TRANSACTIONS**
<déclaration transaction>
{ ; <déclaration transaction> } ;

<déclaration transaction>::= <identificateur> {, <identificateur>} :
(<liste des attributs>) |
<identificateur> {, <identificateur>}

<liste des attributs> ::= <identificateur-attribut> : <type-attribut>
{,<identificateur-attribut> : <type-attribut>}

<identificateur-attribut> ::= PRIORITE | <identificateur>

<type-attribut> ::= REEL | ENTIER

Exemple de déclaration de transactions

TRANSACTIONS

m1, m2, m3 ;

4.3.4.3 Description des écoulements de flux

Il s'agit dans cette section de définir l'ensemble des écoulements de flux. La définition de chaque écoulement est délimitée par les mots-clés FLUX et FFLUX. Cette définition se fait explicitement en spécifiant :

-L'ensemble des transactions et éventuellement les files où ces transaction doivent être sélectionnées associées à la transition du flux d'activité.

-Le lien de synchronisation du réseau du flux d'activité aux événements de la partie commande. Ce lien est décrit soit par la liste des transitions du RdPI, qui permettent lors de leur franchissement de faire évoluer les transactions dans le réseau de files d'attente, soit par la liste des événements externes.

-La liste des transactions à retirer, en donnant les noms des transactions et les noms des files où ces transactions doivent être sélectionnées. L'utilisateur a la possibilité de désigner une transaction particulière de la classe au moyen d'une précondition. Cette précondition est une expression booléenne portant sur les valeurs des attributs et qui permet de déterminer la transaction à sélectionner.

-La liste des transactions à produire, en fournissant les noms des transactions et les noms des files dans lesquelles ces transactions vont être introduites. De plus, aux différents attributs de la transaction à produire peuvent être affectés des valeurs qui permettent de la différencier.

Notons que dans le cas de création initiale de transactions, la liste des transactions à retirer est vide. De même dans le cas de destruction de transactions, la liste des transactions à produire est vide.

Syntaxe

<partie description du flux > ::= **FLUX** <définition de flux> **FFLUX**
{; **FLUX** <définition de flux> **FFLUX**} ;

<définition de flux> ::= <identificateur de transition du RdFA> :
<types d'écoulements possibles> <lien de synchronisation>
<liste de transactions à retirer> <liste de transactions à produire>

{;<lien de synchronisation><liste de transactions à retirer>
<liste de transactions à produire>}
<identificateur de transition du RdFA>::= <identificateur>
<types d'écoulements possibles>::= <identificateur de transaction> |
<identificateur de transaction> **DE** <identificateur de file>
{,<identificateur de transaction> | <identificateur de transaction> **DE**
<identificateur de file> };
<lien de synchronisation>::=**SUR TIR DE** <liste transitions> |**SUR OCCURRENCE**
DE <liste des événements>
<liste transitions>::= <nom de transition> {, <nom de transition>}
<nom de transition>::= <identificateur>
<liste des événements>::= <nom d'événement> | <expression d'événement>
{, <nom d'événement> | <expression d'événement> }
<nom d'événement>::= <identificateur>
<expression d'événement>::= <expression>
<liste de transactions à retirer> ::= ε| **RETIRER** <transactions à sélectionner>
{ ; <transactions à sélectionner> } ;
<transactions à sélectionner> ::= <liste de transactions> **DE** <liste de files> |
<liste de transactions> **DE** <liste de files> **TELQUE** <précondition>
<liste de transactions>::= <nom de transaction> {, <nom de transaction>}
<liste de files>::= <nom de file> {, <nom de file>}
<nom de file> ::= <identificateur>
<nom de transaction> ::= <identificateur>
<précondition>::= <expression>
<liste de transactions à produire>::= ε| **PRODUIRE** <transactions produites>
{ ; <transactions produites>} ;
<transactions produites>::= <liste de transactions> **DANS** <liste de files> |
<liste de transactions> **DANS** <liste de files> **TELQUE** <postcondition>
<postcondition>::= <énoncé>| **DEBUT** <énoncé> {; <énoncé>} **FIN**
<énoncé>::= <nom attribut> := <expression>

Exemple

Reprenons l'exemple de la section de travail (cf. chapitre 4 § 4.2). La description du réseau décrivant le flux d'activité (cf. figure 4.11) à l'aide du langage est la suivante :

FLUX

tt1 : m1, m2, m3 ;

SUR OCCURRENCE DE ↑ Earpm1 PRODUIRE m1 DANS fm1pp1, fm1pp6 ;
SUR OCCURRENCE DE ↑ Earpm2 PRODUIRE m2 DANS fm2pp1, fm2pp6 ;
SUR OCCURRENCE DE ↑ Earpm3 PRODUIRE m3 DANS fm3pp1, fm3pp6 ;
FFLUX;

.

.

FLUX
tt4 : m1, m2, m3 ;
SUR OCCURRENCE DE ↑ Efusinm1 RETIRER m1 DE fm1pp3 ;
PRODUIRE m1 DANS fm1pp4;
. { même déclaration pour les pièces m2 et m3}
.
FFLUX ;

4.4 L'ANALYSEUR DU LANGAGE

Dans ce paragraphe, nous présentons dans les grandes lignes l'analyseur du langage qui vient d'être décrit. Cet analyseur assure les fonctions de vérification syntaxique et sémantique de la description du système et de traduction de cette description en des structures de données qui seront exploitées par le logiciel de simulation ou par tout autre outil de l'environnement de conception : le décompilateur, l'outil de transformation, ...

La première fonction de l'analyseur est de lire et d'analyser la spécification contenue dans le fichier source, il vérifie si cette spécification est conforme ou pas à la syntaxe du langage. Il effectue ensuite des contrôles de cohérences au niveau sémantique : l'utilisation d'identificateurs non déclarés, que tout identificateur déclaré est utilisé, la détection de conflits de types, ...
En cas d'erreur, un message comportant le code de l'erreur (cf annexe B pour la liste des erreurs) et les numéros de la ligne et de la colonne où l'erreur a été détectée est envoyé à l'utilisateur et l'analyse se poursuit automatiquement à partir du premier point de reprise (le point virgule par exemple).

La seconde fonction de l'analyseur est de fournir, si la spécification est syntaxiquement et sémantiquement correcte les structures de données suivantes:

• La liste des places : un élément de cette liste contient l'ensemble des attributs de la place (nom, marquage, pointeur vers la liste des transitions entrantes, pointeur vers la liste des transitions sortantes, pointeur vers la liste des actions associées à la place et leur type : continue ou impulsionnelle).

• La liste des transitions : elle contient l'ensemble des transitions du réseau. Chaque élément de la liste comprend les informations concernant une transition (nom, liste des places en entrée, liste des places en sortie, pointeur vers la condition, pointeur vers l'événement et la politique de communication).

• La table des variables : contient les informations de chaque variable comme son nom, son type (entier, booléen ou réel), sa classe (entrée, sortie, calculée ou temporisation) et ses valeurs (ancienne et nouvelle).

• La table des prédicats : comprend les conditions et événements associés aux transitions du réseau. Chaque élément de la table contient le nom du prédicat, le type du prédicat (condition ou événement) et le pointeur vers la table de code de l'expression du prédicat.

• La table des actions : contient les informations de chaque action comme son nom et son pointeur vers la table de code.

• La table de code : elle contient sous forme postfixée le code des conditions, des événements et des actions.

• La liste des files : un élément de liste contient les attributs de la file (nom, politique de gestion et capacité).

• La liste des transactions : contient les informations concernant les types de transactions avec leurs attributs.

• La table de description de l'écoulement du flux : contient les informations concernant l'aspect structurel du graphe du flux d'activité et de sa synchronisation aux tirs de transitions et/ou aux occurrences d'événements externes.

4.5 EXTENSIONS

Nous venons de présenter un langage de description des RdPI, ce langage permet de décrire un système par un seul RdPI, le réseau pouvant être décrit par parties et l'interconnexion entre les parties étant effectuée par superposition de places ou de transitions communes. La description est d'abord analysée syntaxiquement et sémantiquement par l'analyseur du langage, ensuite elle est traduite en une représentation interne sous forme de structures de données. Ces structures de données peuvent être exploitées directement par le logiciel de simulation: emploi de la simulation au niveau conceptuel de la démarche de conception [Tankoano 88]. Elles peuvent aussi être utilisées par le logiciel de structuration de l'application en un ensemble de modules communicants, avant la phase de simulation (cf. figure 5.10). Dans ce dernier cas la simulation est utilisée au niveau organisationnel de la démarche pour tester différentes configurations du système à répartir.

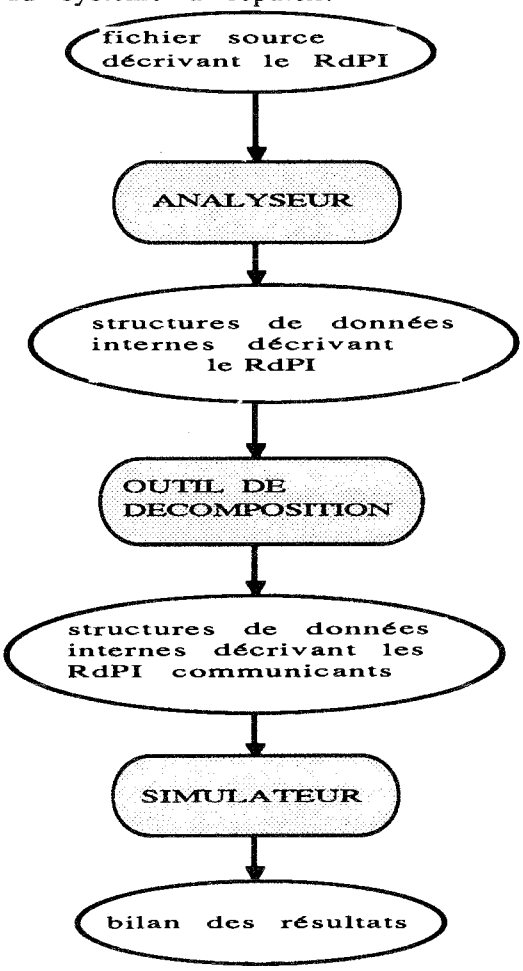
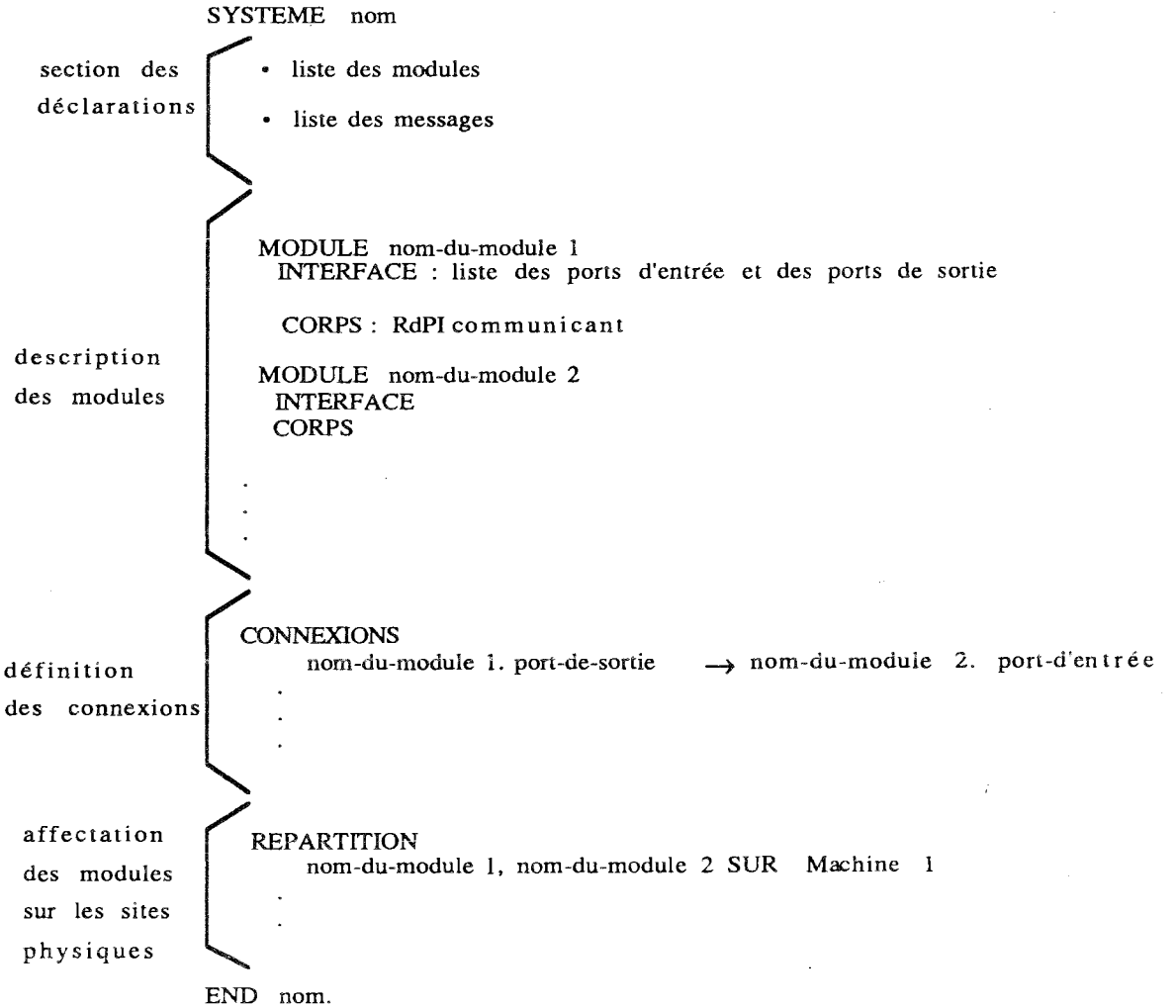


Figure 5.10 : emploi de la simulation au niveau organisationnel

Notons que la version actuelle du langage ne permet pas de décrire le comportement interne du système de commande à répartir, ie. un système défini en termes de modules communicants par échanges de messages. La notion de module est définie au sens de CONIC [Kramer 83] et de FLEXI [Derniame 83] [Zakari 84], un module constitue une unité de répartition (ie. le module ne peut pas être réparti sur plusieurs sites), il ne peut communiquer avec son environnement que par échanges de messages (pas de variables partagées) et peut être constitué de une ou plusieurs tâches parallèles.

Des extensions sont prévues dans la prochaine version de ce langage. Celui-ci doit permettre de décrire l'ensemble des modules du système de commande (chaque module étant décrit par un RdPI), de spécifier la liste des connexions logiques entre les modules communicants et de décrire la répartition des modules sur les sites physiques.

Ainsi la description d'un système réparti dans ce langage se fera selon le schéma suivant:



description d'un système réparti

La première section décrit la liste des modules utilisés par le système et la liste des messages de communication avec leur types. Nous précisons que ces messages servent à la communication inter-modules uniquement.

La seconde section définit les modules du système, chaque module est décrit par une interface et un corps. L'interface est constituée d'un ensemble de ports qui servent au transit des messages. Chaque port est identifié par un nom et caractérisé par un type (qui spécifie le sens de l'échange) et le type de messages admissibles.
Il y a deux types de ports :

- les ports utilisés uniquement en entrée (pour les réceptions de messages) ou uniquement en sortie (pour les émissions de messages).

- Les ports de services utilisables en entrée et en sortie :
  - les ports de sortie-entrée, utilisés d'abord en sortie pour les émissions des requêtes ensuite en entrée pour les réceptions de réponses.
  - Les ports de entrée-sortie, utilisés pour les réceptions de requêtes et ensuite pour les émissions des réponses après traitement.

L'interface est définie par la syntaxe suivante :

Module < nom de module >

Interface

entree-port < nom de port > : < type de message >

sortie-port < nom de port > : < type de message >

es-port <nom de port > : < type de message1 > **Réponse** < type de message2 >

se-port < nom de port > : < type de message1 > **Réponse** < type de message2 >

La partie corps du module comprend la description du RdPI spécifiant le comportement du module. Elle comprend la déclaration des nœuds et des arcs du réseau, la déclaration des variables, des opérations, des conditions et événements et la description de l'interprétation.

5 L'EDITEUR GRAPHIQUE : PETRI-EDIT [Treille 87]

Le second outil que nous présentons pour l'édition des RdPI est graphique. La réalisation d'un tel outil a été motivée par le fait que le modèle RdPI peut utiliser un mode de représentation graphique plus agréable.

Cet outil a la particularité d'intégrer l'édition graphique du réseau et l'édition textuelle de l'interprétation qui lui est associée. Les éditeurs graphiques existants ne permettent en général que l'édition graphique du réseau et par conséquent ne permettent que l'édition des RdP généralisés (cf. chapitre 1).

Nous allons dans ce qui suit présenter cet outil, nous commençons d'abord par décrire les mécanismes d'interaction utilisés dans cet éditeur, nous présentons ensuite ses fonctionnalités.

5.1 LES MECANISMES UTILISES POUR LA GESTION DES INTERACTIONS HOMME/MACHINE

Dans PETRI-EDIT la description de la spécification du système se fait de manière très interactive à l'aide de fenêtres, de menus et de mécanismes simple de désignation.

5.1.1 LES FENETRES

Elles permettent d'utiliser l'écran de manière très souple pour visualiser ou saisir des informations. Plus particulièrement sur un écran on peut faire cœxister plusieurs fenêtres dans lesquelles se déroulent des activités différentes. Dans PETRI-EDIT, l'écran est ainsi découpé en quatre fenêtres:

• Une fenêtre supérieure qui est la **fenêtre de dialogue**, elle permet l'entrée textuelle de certaines informations complémentaires, l'édition de l'interprétation superposée au réseau ainsi que l'affichage des messages d'erreurs.

• Une fenêtre sur la droite où apparaissent les différents menus.

• Une fenêtre centrale qui est la **fenêtre de dessin**, dans laquelle apparaît le réseau en cours d'édition.

• Une fenêtre en bas de l'écran, réduite à une seule ligne et qui donne des informations sur la fonction de déplacement (scrolling) du réseau dans la fenêtre de dessin (voir écran 1 et 2 de l'annexe C).

La représentation du réseau qui apparaît dans la fenêtre de dessin est la représentation classique des RdP :

• les places sont représentées par des octogones, car un cercle sur le matériel utilisé se traduit par une élipse inesthétique,

• les transitions sont représentées par des rectangles,

• les arcs sont représentés par une suite de segments de droites, une flèche positionnée sur l'un des segments donne la direction de l'arc,

• le marquage d'une place est représenté par un nombre entier qui apparaît à l'intérieur de la place,

-il est possible de nommer les places et les transitions et de visualiser ou non ces noms,

-les arcs peuvent être également valués. Toutefois, pour éviter une surcharge du réseau cette valuation n'y apparaît pas directement.

5.1.2 LES MENUS

Ils présentent l'ensemble des commandes prédéfinies de l'éditeur. L'activation de la commande sélectionnée par l'utilisateur entraîne l'exécution d'une fonctionnalité de l'éditeur.

L'avantage des menus comme mécanisme d'interaction est double :

- ils permettent d'une part de guider l'utilisateur en lui présentant l'ensemble des commandes disponibles à un instant donné et uniquement celles-là ,

-d'autre part de limiter les erreurs dues à une méconnaissance des différentes possibilités de l'éditeur.

La sélection d'une commande dans un menu se fait en tapant une touche sur le clavier, cette touche peut correspondre
- soit à une lettre qui est le plus souvent la première lettre de la commande,
- soit à un chiffre servant de clé de sélection,
- soit à une touche de fonction.

La sélection d'une commande peut conduire soit à l'exécution directe d'une fonction simple non décomposable, soit à un autre menu qui peut a son tour entraîner l'exécution d'une fonction ou permettre l'accès à un nouveau menu, et ainsi de suite.

La structure hiérarchique des menus est donnée dans la figure 5.11.

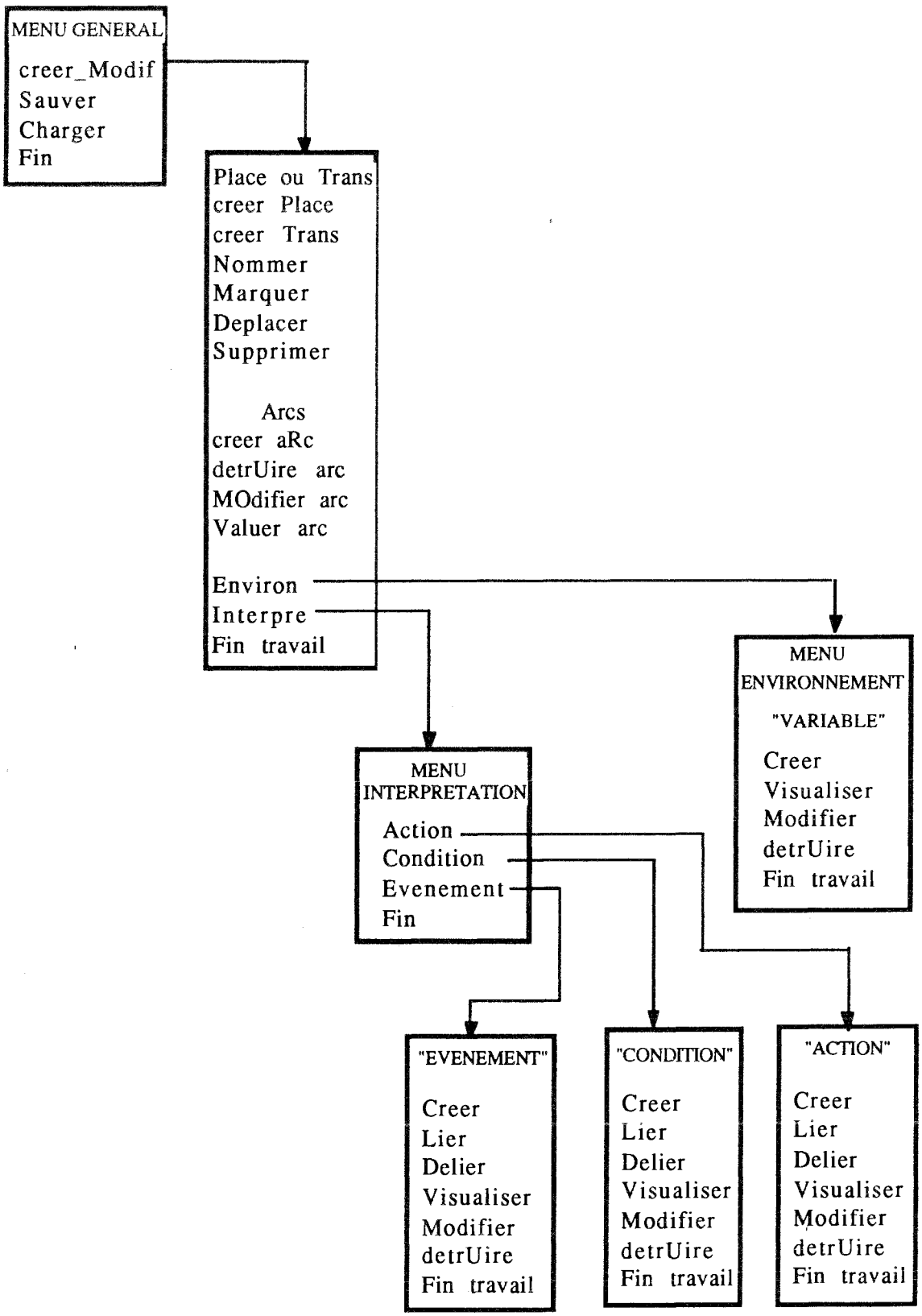


Figure 5.11: structure des menus

5.1.3 LE MECANISME DE DESIGNATION

Ce mécanisme permet de préciser pour certaines commandes les objets sur lesquels elles vont porter.
A l'aide des touches ($\rightarrow$, $\leftarrow$,$\uparrow$,$\downarrow$) l'utilisateur déplace la tortue dans la fenêtre de dessin et désigne un emplacement libre ou un objet du réseau (place, transition) sur lequel il veut effectuer une opération. Ensuite il sélectionne la commande qui permet de réaliser l'opération.

5.1.4 LE MINI-EDITEUR DE TEXTES

Un point important de l'éditeur PETRI-EDIT est que en plus de l'édition graphique du réseau, il permet l'édition textuelle de l'interprétation qui lui est associée. L'utilisateur peut ainsi éditer et analyser les prédicats (conditions et événements) et les actions associés aux transitions et aux places du réseau grâce à un mini-éditeur de textes.

L'édition textuelle se fait dans une fenêtre de quatre lignes, toutefois un mécanisme de "scrolling" donne la possibilité d'éditer de grands textes.
Les fonctionnalités de cet éditeur de textes seront détaillées dans le paragraphe suivant.

5.2 LES FONCTIONNALITES

Nous allons présenter les différentes commandes de l'éditeur graphique. Ces commandes sont regroupées dans des menus.

Nous noterons en **gras** la lettre qui permet d'activer la commande : la clé

5.2.1 MENU GENERAL

Dès l'appel de l'éditeur PETRI-EDIT, ce menu apparaît sur l'écran. Il contient les commandes suivantes:

Charger

charge en mémoire centrale la description d'un réseau contenu dans un fichier et affiche dans la fenêtre de dessin le réseau.

Sauver

sauvegarde dans un fichier la description d'un réseau (dessin du graphe et interprétation superposée).

Créer Modif

entre dans le menu de création et de modification d'un réseau.

Fin

sort de l'éditeur.

Une image de l'écran est présentée dans l'annexe c.

5.2.2 MENU DE CREATION MODIFICATION

Ce menu est destiné à l'édition proprement dite du réseau, il se compose des commandes suivantes :

créer Place

cette commande permet la création d'une place à la position désignée par la tortue. Si la position est déjà occupée, l'éditeur refuse la création de la place et affiche un message d'erreur dans la fenêtre de dialogue. Par contre si la position est libre la place s'affiche.

créer Trans

cette commande est similaire à créer Place, elle permet de créer une transition.

Nommer

permet de nommer ou de changer le nom de l'objet (place ou transition) désigné par la tortue. Si la tortue désigne un emplacement vide, un message d'erreur est affiché. Dans le cas contraire, l'utilisateur donne dans la fenêtre de dialogue le nom de l'objet.

Dans le cas où l'objet a été déjà nommé, le nom apparaît dans la fenêtre de dialogue et l'utilisateur peut soit le conserver, soit le modifier en tapant un nouveau nom.

L'utilisateur peut ensuite choisir de visualiser ce nom sur le dessin ou non afin de ne pas trop le surcharger. Dans le premier cas, l'utilisateur désigne l'endroit où il désire que le nom s'affiche à l'aide de la tortue et la touche fonction F2.

Le nom apparaîtra alors à l'endroit désigné.

Dans le second cas, le nom restera cacher.

Marquer

permet de marquer ou de modifier le marquage de la place désignée

par la tortue. L'éditeur commence par vérifier que l'objet désigné est bien une place, ensuite il affiche dans la fenêtre de dialogue l'ancien marquage de la place s'il existe et demande à l'utilisateur d'entrer le marquage qu'il désire affecter à la place (nombre entier positif ou nul). L'éditeur vérifie d'abord que ce marquage est correct avant de l'inscrire au centre de la place considérée.

Déplacer

déplace l'objet (place ou transition) désigné par la tortue.

L'utilisateur pointe le nouvel emplacement à l'aide de la tortue et tape la touche F1. Si cet emplacement est libre, l'objet apparaît aussitôt à cet endroit accompagné de son nom s'il est visible et de son marquage s'il s'agit d'une place. L'ensemble des arcs qui lui sont attachés se déplace également avec lui.

Si au contraire l'emplacement est occupé alors un nouveau pointage est nécessaire.

Supprimer

supprime l'objet désigné par la tortue.

Cette commande agit d'abord au niveau externe ensuite au niveau interne. L'objet est tout d'abord effacé du dessin ainsi que l'ensemble des arcs qui lui sont connectés, si l'utilisateur valide la destruction, la suppression est répercutée au niveau interne (structures de données), sinon on réaffiche le tout et rien n'est changé.

créer aRc

permet de créer un arc entre deux objets place-transition ou transition-place. L'utilisateur doit désigner au moyen de la tortue, la place ou la transition qui sera le point de départ de l'arc et l'objet d'arrivée de l'arc en utilisant toujours la tortue et la touche F1.

L'éditeur vérifie si l'arc respecte la syntaxe des RdP, c'est-à-dire aller d'une transition vers une place ou vise versa. Si c'est oui, l'arc est alors crée et tracé. Dans le cas contraire, un message d'erreur est affiché.

mOdification arc

permet de modifier l'apparence géométrique de l'arc.

Lorsqu'un arc est crée, il est composé d'une seule ligne droite, cette commande permet de remplacer cette ligne droite par une ligne brisée.

Pour cela, l'utilisateur, identifie d'abord l'arc sur lequel va porter la

modification en désignant les sommets de départ et d'arrivée de l'arc. Ensuite, à l'aide de la tortue et de la touche F1, il indique les différents points intermédiaires successifs. Pour chaque point montré, l'éditeur détruit l'ancien segment et affiche deux autres segments qui lient l'ancien point au nouveau point et le nouveau point au sommet d'arrivée.

détrUire arc

détruit l'arc désigné par l'utilisateur. La désignation de l'arc se fait en précisant ses sommets de départ et d'arrivée. L'éditeur procède d'abord à la destruction de l'arc au niveau externe donc efface l'arc du dessin. La destruction se poursuit au niveau interne après validation de la part de l'utilisateur, sinon la commande est annulée et l'arc est réaffiché.

Valuer arc

Cette commande sert à affecter à un arc une valeur ou "poids" donnée par l'utilisateur dans la fenêtre de dialogue. L'éditeur vérifie que cette valeur est supérieure ou égale à 1. Pour des raisons de clarté la valeur de l'arc n'apparaît pas sur le dessin. Toutefois, l'utilisateur peut à tout instant connaître la valeur de l'arc en faisant appel à cette même commande. Elle peut également servir pour la modification de la valeur de l'arc.

NB : Tous les arcs ont par défaut une valuation de 1.

Environ

active le menu qui permet la déclaration des variables du système de commande.

Interpre

passe au menu d'édition des conditions, événements et actions.

Fin de travail

retourne au menu général.

5.2.3 MENU ENVIRONNEMENT

Ce menu permet à l'utilisateur de définir

- l'ensemble des variables de communication du système de commande avec le procédé : les variables d'entrée et les variables de sortie,

- et l'ensemble des variables internes : les variables calculées et les temporisations.

Ces différentes variables seront ensuite référencées dans les conditions, événements et actions qui seront traités ultérieurement dans le menu interprétation.

Les commandes disponibles dans ce menu sont:

Créer

appelle le menu de création d'une variable.

Visualiser

lance le menu de visualisation des variables.

Modifier

entre dans le menu de modification d'une variable.

Détruire

active le menu de destruction d'une variable.

Fin de travail

retourne au menu 'création modification' (cf. § 2.2.2).

Pour toutes ces commandes (sauf fin de travail) le menu qui s'affiche est généralement le même. Il s'agit du menu permettant le choix de l'ensemble de variables sur lequel l'utilisateur désire exécuter ces commandes.
Ce menu se présente comme suit :

1 : l'ensemble des variables d'entrée
2 : l'ensemble des variables de sortie
3 : l'ensemble des variables calculées
4 : l'ensemble des variables de temporisation

L'utilisateur fait son choix en entrant un chiffre entre 1 et 4. A la suite de ce choix, l'éditeur réalise la fonction correspondante à la commande précédemment sélectionnée :

Pour la création d'une variable

L'utilisateur doit entrer dans la fenêtre de dialogue le nom de la variable à créer. Si ce nom existe déjà, l'éditeur signale une erreur et lui demande de fournir un nouveau nom. L'utilisateur doit ensuite préciser le type de la variable, il a le choix entre trois types prédéfinis:

'1' pour le type entier
'2' pour le type booléen
'3' pour le type réel

La variable est enfin créée dans l'ensemble choisi et un retour au menu environnement est effectué.

Pour la visualisation d'un ensemble de variables

Après décompilation, l'éditeur affiche dans la fenêtre de dialogue les variables de l'ensemble choisi sous la forme d'un texte de déclaration PASCAL.
Ce texte contient le nom de l'ensemble (entrée, sortie, calculées ou temporisation) et le nom de toutes les variables de cet ensemble regroupées par type.
L'utilisateur a la possibilité de se déplacer dans le texte à l'aide des touches (↑,↓), page-up, page-down et ctrl-page-up et ctrl-page-down pour aller au début et à la fin du texte. Pour terminer la visualisation l'utilisateur tape la touche "END" et revient donc au menu environnement.
NB : dans le cas où l'ensemble est vide, un message prévient l'utilisateur.

Pour la modification d'une variable

L'utilisateur commence par donner dans la fenêtre de dialogue le nom de la variable à modifier. L'éditeur vérifie que cette variable existe, si oui, il fournit dans la même fenêtre les informations concernant cette variable : le nom de l'ensemble auquel elle appartient et son type. Ces deux informations peuvent être modifiées par l'utilisateur. Il peut choisir un autre ensemble et/ou un autre type de variable en entrant les chiffres clés correspondant à ses choix:

1. entrée 2. sortie 3. calculée 4. temporisation

pour l'ensemble, et par la suite

1. entier 2. booléen 3. réel

pour le type.

Les différentes modifications sont effectuées et on revient au menu environnement.

Pour la destruction d'une variable

L'utilisateur donne le nom de la variable à éliminer. Si cette variable n'est référencée dans aucun prédicat (condition et événement) ni aucune action, alors l'éditeur détruit la variable et on revient au menu environnement. Par contre si la variable est utilisée, l'éditeur refuse sa destruction.

5.2.4 MENU INTERPRETATION

Ce menu permet la saisie et l'analyse des textes correspondant aux conditions, événements et actions. Il permet aussi de lier ces derniers aux places et aux transitions du réseau apparaissant dans la fenêtre de dessin.

Les commandes qui le composent sont :

Action

active le menu concernant les actions.

Condition

lance le menu concernant les conditions.

Evénement

lance le menu concernant les événements.

Fin de travail

retourne au menu "création modification".

Pour les trois commandes, un même menu appelé "menu des réalisations" est proposé.

Les commandes contenues dans ce menu des réalisations permettent de :

Créer

une action, une condition ou un événement. l'utilisateur donne le nom de l'action suivie de son corps (dans le cas de la création d'une action), le nom de la condition suivie de son expression booléenne ou le nom de l'événement suivi de l'expression événementielle. Pour cela, L'utilisateur est placé sous un mini-éditeur pour y taper son texte. Ce texte est entré en mode "réécriture" et non en mode "insertion".

Un certain nombre de fonctions du mini-éditeur sont mises à la disposition de l'utilisateur afin de lui simplifier l'édition des textes. Ainsi il lui est possible d'insérer une ligne avec la touche RETURN, d'en détruire une avec F1, de se déplacer sur le texte à l'aide des touches ($\rightarrow$, $\leftarrow$,$\uparrow$,$\downarrow$), ...

Pour sortir de sous le contrôle de l'éditeur de texte, l'utilisateur utilise la touche END. La sortie de l'éditeur lance aussitôt l'analyse syntaxique et sémantique du texte saisi. Si une erreur est détectée, le numéro de la ligne et de la colonne de celle-ci sont affichés. L'utilisateur a le choix entre abandonner son texte ou effectuer une correction. Dans le premier cas, l'action, la condition ou l'événement ne sont pas créés et l'utilisateur retourne au menu des réalisations. Dans le second cas, l'utilisateur est replacé sous le mini-éditeur où le texte a été réaffiché, le curseur se trouve à l'endroit où l'erreur a été détectée. L'utilisateur peut corriger l'erreur et continuer jusqu'à ce que le texte soit syntaxiquement et sémantiquement correct.

Le nom et le code engendré sont rangés dans les structures de données et le retour au menu des réalisation est effectué.

Lier

une action à une place ou une condition ou un événement à une transition. L'action, la condition ou l'événement doivent être créés au préalable. L'utilisateur donne dans la fenêtre de dialogue le nom de l'action, la condition ou l'événement et désigne l'objet du réseau qu'il désire lier.

L'éditeur vérifie que l'utilisateur n'a pas tenter de lier une action à une transition ou un événement à une place. Si, c'est oui alors la liaison est effectuée dans les structures internes et rien n'apparaît sur le dessin (pour ne pas trop l'alourdir).

En outre, s'il s'agit de la liaison d'une action, l'utilisateur doit préciser en plus le type de l'action à savoir :

1- continue 2- impulsionnelle

NB : l'utilisateur peut lier plusieurs actions à une même place.

Délier

permet de détruire la liaison d'une action à une place ou d'une condition ou d'un événement à une transition. L'utilisateur spécifie la liaison à supprimer en désignant l'objet du réseau (place ou transition).

L'éditeur s'assure d'abord de l'existence de cette liaison avant de la détruire. Cette destruction est effectuée au niveau des structures internes et rien n'apparaît sur le dessin.

Visualiser

est un menu de trois commandes qui proposent les modes de visualisation suivantes :

D'une place ou D'une transition

permet, une fois la place ou la transition désignée de visualiser l'interprétation associée.

L'éditeur décompile le code des actions ou des expressions et affiche le texte obtenu dans la fenêtre de dialogue. L'utilisateur peut à l'aide des touches ($\rightarrow$, $\leftarrow$,$\uparrow$,$\downarrow$), page-up, page-down se déplacer sur le texte. Il peut retourner au menu des réalisations en tapant la touche END.

L'ensemble

visualise l'ensemble des actions, conditions ou événements qui ont été crées.

Une action ou Une condition ou Un événement

permet de visualiser le texte d'une action, d'une condition ou d'un événement dont le nom a été fourni par l'utilisateur.

Modifier

permet de modifier le texte d'une action, d'une condition ou d'un événement dont le nom est fourni par l'utilisateur.

L'éditeur vérifie si effectivement l'action, la condition ou l'événement existe. Si oui, il décompile le code correspondant et le présente par le mini-éditeur de texte dans la fenêtre de dialogue. L'utilisateur peut alors modifier son texte à loisir. Lorsqu'il quitte le mini-éditeur l'analyse du texte est lancée suivant le même

principe que pour la fonction créer (correction jusqu'à validation du texte ou abandon).

Détruire

une action, une condition ou un événement donnés et de ses liaisons (si elles existent) avec les objets du réseau.

Fin de travail

retourne au menu interprétation.

5.3 REMARQUES PRECISIONS ET EXTENSIONS

Comme le langage de description textuelle PETRI-EDIT ne permet que l'édition d'une application décrite par **un seul RdPI** (pouvant être de taille importante), l'édition d'une application définie par un réseau de RdPI communicants n'étant pas encore prise en compte. D'autre part, PETRI-EDIT ne permet pas l'édition de la description du flux d'activité. Des extensions sont prévues dans la prochaine version de PETRI-EDIT qui concernent la prise en compte de la description du flux d'activité, l'édition de l'ensemble des RdPI communicants de l'application et l'élaboration de fonctions de transformation comme la fusion de réseaux, le raffinement d'un réseau et la décomposition d'un réseau en un ensemble de sous_réseaux communicants par échange de messages. Notons que si les premières extensions sont faciles à réaliser, il n'en est pas de même pour la mise en œuvre des fonctions de raffinement et de décomposition qui pose le problème de placement des objets du réseau sur l'écran après transformation. Des travaux sont en cours dans notre équipe de recherche [Touzani 90] pour résoudre ce type de problèmes.

5.4 CONCLUSION

Cette partie nous a permis de présenter un outil d'édition graphique de RdPI spécifiant un système de commande. Cet éditeur a été réalisé sur un IBM PC dans le langage PASCAL. Il utilise les primitives graphiques ainsi que le mécanisme de fenêtrage et de tortue qui remplace la souris.

6 L'INTERFACE UTILISATEUR DE PETRI-S

Le rôle de l'interface est de permettre la communication entre le système et l'utilisateur du système. Une bonne interface doit être homogène, c'est-à-dire que les mêmes mécanismes d'interaction doivent être utilisés à tous les niveaux. Elle doit être aussi agréable à utiliser par un "habitué" et compréhensible par un débutant. L'interface proposée par PETRI-S possède l'ensemble de ces qualités. Cette interface est homogène, simple et hautement interactive. Elle s'appuie sur les fenêtres, les menus et un mécanisme de désignation simple.

Les fenêtres sont utilisées comme moyen d'entrée des informations et comme support d'affichage. Plusieurs fenêtres peuvent coexister sur l'écran, mais une seule fenêtre est active (c'est-à-dire connectée au clavier) à un instant donné.

Le principe de menu permet de regrouper des commandes de même nature et de visualiser cet ensemble sur l'écran afin que l'utilisateur puisse sélectionner une des commandes. La sélection d'une commande dans PETRI-S s'effectue par une désignation directe à l'aide du curseur, ce qui offre l'avantage à l'utilisateur de ne pas avoir de texte à taper.

D'autre part, le menu principal de PETRI-S est statique ce qui permet d'offrir un accès rapide aux différentes commandes. En effet, ce menu est une barre horizontale de titres de menus qui est affichée de façon **permanente**, le menu associé à chaque titre apparaît lorsque l'utilisateur désigne le titre en positionnant le curseur et en faisant un retour chariot (voir écran 1 de l'annexe D).

7 LE SIMULATEUR PETRI-S

Le processus de simulation se déroule en trois phases : la phase de préparation des informations nécessaire à la simulation, la phase de simulation proprement dite et la phase de génération des résultats.

7.1 LA PHASE DE PREPARATION

Cette phase permet à l'utilisateur de préciser les conditions de simulation qu'il souhaite : il sollicite ou non la vérification des propriétés du système, il décrit le comportement de l'environnement avec lequel le système à étudier doit interagir, il spécifie l'état initial du système (marquage initial du réseau et valeurs initiales des variables du système) et il définit les conditions d'arrêt de la

simulation ainsi que le mode de fonctionnement du simulateur.

7.1.1 LA DESCRIPTION DU COMPORTEMENT DE L'ENVIRONNEMENT

Cette description permet d'engendrer automatiquement au cours de la simulation les différentes valeurs des variables d'entrée qui rendent compte de l'évolution des grandeurs physiques de l'environnement. Elle est effectuée à l'aide de types génériques de générateurs d'événements (cf. chapitre 4 § 3) que l'utilisateur instancie en utilisant un éditeur spécial d'acquisition.
L'utilisateur lance l'éditeur en sélectionnant la fonction "gestion des événements" dans le menu principal de PETRI-S.

7.1.1.1 L'éditeur d'acquisition de la description de l'environnement

Cet éditeur permet la construction, la modification et la visualisation interactive de la description du comportement de l'environnement avec lequel le système de commande interagit.
L'utilisateur a accès aux différentes fonctions de l'éditeur par l'intermédiaire de menus. Le menu principal fait apparaître l'ensemble des fonctions réalisées par l'éditeur à savoir:

- la construction d'une description
- la modification d'une description
- la visualisation d'une description
- le chargement d'une description construite auparavant
- et la sauvegarde d'une description

Nous allons dans ce qui suit présenter ces différentes fonctions.

a- La construction

La construction de la description du comportement de l'environnement s'effectue par l'acquisition de paramètres permettant d'instancier les types génériques de générateurs d'événements prédéfinis (sept types ont été définis dans le chapitre 4 section 3). Chaque instanciation donne lieu à la création d'un schéma de génération d'événements utilisable au cours de la simulation.
L'acquisition des paramètres s'effectue selon le principe suivant:
L'utilisateur procède d'abord au choix du type générique pour lequel il veut

définir un générateur d'événements en désignant un type parmi la liste des types proposés par l'éditeur. Le type sélectionné apparaît en inversion vidéo, en même temps que l'éditeur affiche sur l'écran deux fenêtres :

- une fenêtre qui présente l'ensemble des variables d'entrée du système de commande ainsi que leur type,

- une fenêtre de saisie, qui décrit les différents paramètres d'instanciation que l'utilisateur doit fournir.

L'utilisateur commence par sélectionner dans la première fenêtre la variable pour laquelle il désire instancier un générateur des valeurs successives que prendra cette variable. Ceci est fait en désignant directement (à l'aide du curseur) la variable et en tapant un "retour chariot". Cette sélection provoque la désactivation de la fenêtre contenant l'ensemble des variables d'entrée et l'activation de la fenêtre de saisie dans laquelle l'éditeur affiche le nom de la variable sélectionnée.

L'utilisateur a ensuite le choix entre abandonner l'instanciation du générateur correspondant à la variable choisie et revenir à la première fenêtre, ou poursuivre sa construction en fournissant les valeurs des paramètres d'instanciation. L'utilisateur peut introduire ces valeurs dans l'ordre qu'il lui plaît en sélectionnant tour à tour les paramètres qu'il veut définir et en tapant sur le clavier leur valeurs. L'arrêt de la saisie des paramètres se fait lorsque tous les paramètres ont été définis et que l'utilisateur a tapé la touche ESC. En effet, la saisie n'est considérée comme terminée que si l'utilisateur l'indique, ce qui lui donne la possibilité de changer les valeurs de certains paramètres déjà définis autant qu'il veut. A la fin de la saisie, l'éditeur effectue une analyse syntaxique et sémantique des données fournies par l'utilisateur.

Si l'éditeur ne détecte pas d'erreurs l'instance est dite "valide", elle est alors rangée dans les structures de données pour être utilisée au cours de la simulation. Par contre s'il détecte des erreurs, il les signales à l'utilisateur en affichant des messages sur l'écran. L'utilisateur peut alors corriger ces erreurs et relancer l'analyse. Ce processus ce poursuit jusqu'à ce que l'instance soit "valide".

L'utilisateur peut soit recommencer ce processus d'instanciation pour une autre variable d'entrée, soit retourner au menu qui propose la liste des types génériques. Il peut ainsi continuer la description du comportement de l'environnement en instanciant d'autres types de générateurs selon le principe

qui vient d'être décrit.

b- La modification

Lorsque l'éditeur est utilisé pour modifier une description, deux cas peuvent être considérés selon les intentions de l'utilisateur:

1ercas- L'utilisateur désire seulement supprimer certaines instances de schémas de génération, pour cela il fait appel à la fonction **détruire instances** de l'éditeur. Dès l'invocation de cette fonction, l'éditeur décompile et affiche dans une fenêtre l'ensemble des instances existantes à cet instant, l'utilisateur peut ensuite sélectionner, grâce au curseur, les instances à détruire. L'éditeur supprime ces instances et affiche le résultat.

2^{em}cas- L'utilisateur désire remplacer certaines instances de schémas de génération par d'autres. Pour cela il fait appel a deux fonctions de l'éditeur:

- la fonction détruire instances pour supprimer les instances qu'il désire modifier,
- et la fonction créer instances (vue dans a) pour définir d'autres instances.

c- La visualisation

Cette fonction permet d'afficher dans une fenêtre la description du comportement de l'environnement caractérisée par l'ensemble des schémas de génération d'événements.

d- La fonction de sauvegarde

Elle permet la mémorisation des schémas de génération dans un fichier. Lors de l'invocation de cette fonction, l'éditeur demande à l'utilisateur le nom du fichier dans lequel il désire mémoriser les schémas. Si le fichier existe déjà l'éditeur demande à l'utilisateur s'il faut toujours conserver le même nom de fichier; si oui, l'ancien contenu du fichier est remplacé par le nouveau; sinon, l'utilisateur donne un autre nom de fichier dans lequel on mémorise les schémas.

e- La fonction de chargement

Cette fonction permet de charger en mémoire centrale à partir d'un fichier dont le nom est fourni par l'utilisateur des schémas de génération d'événements préalablement sauvegardés.

Remarque

Les fonctions de sauvegarde et de chargement permettent un gain de temps appréciable lorsqu'il s'agit d'effectuer plusieurs simulations en faisant varier uniquement certains paramètres du modèle de description de l'environnement.

7.1.2 SPECIFICATION DE L'ETAT INITIAL

Cette opération permet de préciser le marquage initial du réseau et les valeurs initiales des différentes variables du système. Pour cela l'utilisateur sélectionne la fonction "Modification" du menu principal qui propose les trois fonctions suivantes:

• Visualiser variables : elle permet d'afficher dans une fenêtre de l'écran l'ensemble des variables du système avec leur valeur courante. Le rôle de cette fonction est d'une part d'offrir à l'utilisateur le moyen d'observer l'évolution des variables au cours de la simulation, d'autre part de voir pendant la phase de préparation, les variables qui ont été initialisées et celles qui ne le sont pas encore (en effet, pour les variables auxquelles aucune valeur n'a été attribuée leur champ valeur est vide).

• Modifier variables : elle permet d'initialiser les variables du système ou de modifier leur valeur pendant la phase de simulation. Sur invocation de cette fonction, l'outil PETRI-S affiche l'ensemble des variables du système avec leur type. L'utilisateur peut alors sélectionner la variable qu'il désire initialiser ou modifier et taper au clavier la valeur à affecter. Si la valeur fournie par l'utilisateur n'est pas conforme au type de la variable, une erreur est signalée. La correction s'effectue simplement en retapant une nouvelle valeur.

• Modifier marquage : cette fonction permet de fournir le marquage initial du réseau. L'intérêt de cette fonction est qu'elle permet de relancer la simulation avec des marquages initiaux différents sans avoir à passer par une étape

d'édition. Cette fonction peut également servir en cours de simulation pour modifier le marquage de certaines places du réseau et créer ainsi des situations de panne par exemple.
La modification du marquage des places se fait selon les mêmes principes que pour la modification des variables.

7.1.3 SPECIFICATION DU DEBUT DE LA SIMULATION

Cette opération permet de spécifier l'heure à partir de laquelle la simulation commence.
Pour cela, l'utilisateur désigne la fonction "Initialiser horloge" dans le menu "conditions de simulation". La sélection de cette fonction provoque l'affichage d'une fenêtre "horloge" dans laquelle l'utilisateur peut taper la valeur initiale de l'horloge.
L'horloge est affichée de façon permanente pendant la simulation, elle indique la valeur courante du temps simulé.

NB : dans le cas où l'utilisateur ne fournie pas de valeur, l'horloge est mise à zéro par défaut.

7.1.4 LA DUREE DE LA SIMULATION

La durée de la simulation peut être spécifiée par la fonction "Durée de la simulation" du menu "conditions de simulation" de deux manières :

- soit en indiquant la durée de la simulation,
- soit en indiquant le nombre de cycles de simulation.

Nous rappelons qu'un cycle de simulation est une séquence de franchissements de transitions qui fait passer le système d'un état stable à un autre.
Cette information est utilisée pour arrêter la simulation (lorsque le nombre de cycles ou le temps fournie par l'utilisateur est atteint). La simulation peut toutefois s'arrêter lorsque l'échéancier est vide ou lorsqu'il y a blocage.

7.1.5 VERIFICATION DES PROPRIETES

L'outil peut vérifier, à la demande, les propriétés du RdPI décrivant le système de commande (bornage, blocage total et partiel, la persistance et les

invariants de places et de transitions). L'utilisateur désigne les propriétés qu'il désire vérifier et l'outil se charge de leur preuve selon les principes décrits dans le paragraphe 2.2.

7.1.6 LES MODES D'EXPLOITATION DU SIMULATEUR

L'utilisateur peut disposer de plusieurs modes d'exploitation qui dépendent de ses objectifs : l'aide à l'expression des besoins, la validation et mise au point ou l'évaluation.

7.1.6.1 Le mode pas à pas

Dans le mode pas à pas, grâce aux commandes précédentes, l'utilisateur peut intervenir sur le déroulement de la simulation. Il peut :

- modifier les valeurs des variables ou le marquage des places du réseau (cf. § 7.1.2),

- réinitialiser la simulation (cf. § 7.1.3),

- modifier le mode d'exploitation (cf. § 7.1.6),

- insérer ou retirer des événements de l'échéancier, pour simuler des situations dangereuses et observer le comportement du système de commande face à ces situations (cf. § 7.1.1),

- suivre l'évolution de son système en demandant de visualiser le marquage courant de son réseau, l'état du procédé, les statistiques sur les files d'attente, ...

Ce mode est particulièrement utile pour la mise au point.

7.1.6.2 Le mode semi-automatique

Ce mode permet à l'utilisateur d'observer, sur son poste de travail, l'évolution ralentie de l'état de son système. Cet état peut être caractérisé soit par le marquage des places du réseau, soit par l'état des ports d'entrée et de sortie du

système de commande...

Dans ce mode, l'utilisateur ne peut toutefois pas intervenir sur le déroulement de la simulation.

7.1.6.3 Le mode automatique

Dans ce mode, l'utilisateur ne peut pas non plus intervenir sur le déroulement de la simulation. La fin de la simulation retourne dans un fichier les résultats des traitements effectués (vérification des différentes propriétés, calcul des statistiques, ...). Ceux-ci peuvent être consultés à l'écran ou édités sur imprimante.

Ce mode de fonctionnement est essentiellement utilisé pour l'aide au dimensionnement.

7.2 LA PHASE DE SIMULATION

L'utilisateur active la phase de simulation en sélectionnant la fonction "lancement de la simulation" du menu principal de PETRI-S. Cette phase consiste en fonction des entrées engendrées par l'environnement à faire évoluer le système et à engendrer les sorties destinées aux actionneurs.

Nous abordons, la description de cette phase selon les principaux points suivants :

- la gestion du temps de simulation,

- l'approche de simulation retenue,

- et l'algorithme de simulation.

7.2.1 LA GESTION DU TEMPS

Il existe deux approches pour faire progresser le temps au cours d'une simulation :

• Le temps simulé est mis à jour à des intervalles de temps réguliers. On dit, dans ce cas, que la simulation est dirigée par le temps.

• Le temps est actualisé à chaque fois qu'un événement se produit. On dit, alors, que la simulation est dirigée par les événements.

Dans le premier cas, on définit une unité de temps appropriée au problème et on dispose d'une horloge qui progresse par pas d'unité de temps. A chaque incrémentation de l'horloge on examine la liste des événements pour voir si l'un d'eux apparaît à cette date. Cette approche est peu utilisée car elle demande souvent un très grand nombre de tests et consomme ainsi beaucoup de temps de calcul.

Dans le second cas, on utilise un échéancier constitué d'une liste d'événements placés en ordre croissant selon leur date d'occurrence. Puisqu'on sait qu'il n'y aura aucun changement d'état en dehors des dates d'événements contenus dans l'échéancier, l'horloge est mise à jour seulement chaque fois qu'un nouvel événement est pris en compte.

C'est cette seconde approche que nous utilisons. Son principal avantage est de ne considérer que les dates pour lesquelles un changement survient dans le système.

7.2.2 L'APPROCHE DE SIMULATION

Dans le chapitre 3, nous avons présenté les trois approches de simulation possibles:

• l'approche par événements s'intéresse à la suite des événements qui se produisent au cours de l'évolution du système,

• l'approche par activités consiste à chercher l'état du système à intervalle de temps constant,

• l'approche par processus consiste à décrire le système par un ensemble de processus interagissant.

PETRI-S repose sur une approche par événements. Il utilise pour cela, d'une part, un échéancier dans lequel sont rangés les événements provenant de l'environnement externe et les fins de temporisations et d'autre part, un joueur de RdPI. Chaque fois qu'un nouvel événement est pris en compte, le joueur fait évoluer le réseau décrivant le comportement du système de commande et celui décrivant le flux d'activité d'un état stable à un autre. Les événements qui dépendent de l'état de la commande (événements endogènes) sont engendrés au

cours de la recherche de cet état stable. En ce qui concerne les événements indépendants de la commande (événements exogènes), une occurrence de l'événement est engendrée selon une loi d'arrivée à la suite de la prise en compte de l'occurrence de l'événement précédent.

7.2.3 L'ALGORITHME DE SIMULATION

L'algorithme de simulation consiste à faire évoluer les marquages du réseau décrivant la partie commande et du réseau décrivant le flux d'activité. On considère l'événement qui se trouve en tête de l'échéancier, on fait progresser le temps de simulation et l'on cherche à franchir toutes les transitions franchissables jusqu'à parvenir à un état stable caractérisé par le fait qu'à cet instant précis le réseau modélisant le système ne peut plus évoluer.
Si le réseau n'évolue pas vers un état stable, le logiciel le signale à l'utilisateur et arrête la simulation. Au cours de cette recherche on fait également évoluer le réseau représentant le flux d'activité.

Par ailleurs, si l'outil est utilisé pour la validation nous vérifions pour chaque marquage résultant du tir de ces transitions:
1°) si ce marquage est déficient ou non pour l'ensemble des verrous calculés, ce qui permet de détecter d'éventuels blocages,
2°) si ce marquage vérifie les propriétés (bornage, les invariants de places et de transitions et la persistance) lorsque ces propriétés n'ont pas pu être vérifiées de manière statique.

algorithme
début
|
| procédure d'initialisation
| Répéter
| | - considérer le prochain temps noté dans l'échéancier et mettre à
| | jour l'horloge
| | -prendre en compte les occurrences d'événements associées à cet
| | instant
| | Pour chaque événement pris en compte Faire
| | | - engendrer la prochaine occurrence de l'événement;
| | | - introduire cette occurrence dans l'échéancier
| | Finpour

```
|   |   - modifier les variables qui rendent compte de ces occurrences
|   |     d'événements
|   |   - évaluer les expressions des événements ( soit EV l'ensemble des
|   |   événements présents )
|   |   Pour chaque événement e de EV Faire
|   |   |   Si le réseau du flux d'activité est synchronisé à e
|   |   |   Alors
|   |   |   |   -faire évoluer le réseau du flux d'activité
|   |   |   Finsi
|   |   Finpour
|   |   - effectuer un tir sur occurrences de ces événements
|   |   Si non état stable (* il existe des transitions franchissables *)
|   |   Alors
|   |   |   Répéter
|   |   |   |   -considérer une transition non synchronisée à aucun
|   |   |   |     événement
|   |   |   |   Si cette transition est franchissable
|   |   |   |   Alors
|   |   |   |   |   franchir-transition
|   |   |   |   Finsi
|   |   |   |   Si le nombre de transitions franchie > nombre très grand
|   |   |   |   Alors
|   |   |   |   |   bouclage
|   |   |   |   Finsi
|   |   |   Jusqu'à état stable ou bouclage
|   |   |   Si bouclage
|   |   |   Alors
|   |   |   |   signaler erreur à l'utilisateur: "ABSENCE D'ETAT STABLE"
|   |   |   |   fin de simulation
|   |   |   Finsi
|   |   Finsi
|   Jusqu'à fin de simulation
fin
```

Nous allons dans ce qui suit expliciter les différents points importants de l'algorithme.

7.2.3.1 Procédure d'initialisation

Cette procédure permet :

• de vérifier que le marquage initial est stable compte tenu des valeurs initiales des variables;

• de vérifier (sur la demande de l'utilisateur) les propriétés du RdPI décrivant le comportement du système de commande en analysant de façon statique le RdP global sur lequel est construit le RdPI. Dans le cas où ces vérifications ne sont pas concluantes, le système procédera alors à des vérifications en cours de simulation.

• et d'établir l'échéancier initial dans lequel sont notés par ordre croissant les occurrences des événements exogènes et les occurrences des événements endogènes dépendant de l'état initial du système.

7.2.3.2 Procédure effectuant un tir sur occurrences d'événements

Le rôle de cette procédure est de déterminer une séquence de simulation complète s_c et de franchir les transitions de cette séquence. Pour cela, elle détermine d'abord l'ensemble $T_{EV,M}$ des transitions réceptives à EV, où M est le marquage courant du RdPI et EV est l'ensemble des événements présents à l'instant courant, c'est-à-dire les événements qui viennent de se produire et les événements qui sont déjà arrivés et qu'on n'a pas pu prendre en compte, mais qui ont été mémorisés pour un traitement en différé.

A partir de l'ensemble $T_{EV,M}$, elle détermine la séquence de simulation complète selon les règles de construction d'une séquence de simulation décrites dans le chapitre 4 § 2.2. Ensuite elle franchit la séquence de simulation.

Notons que dans le cas où plusieurs séquences de simulation complètes existent, la procédure choisit de façon aléatoire la séquence à franchir.

Procédure de détermination de $T_{EV,M}$

algorithme

début

| $T_{EV,M} := \emptyset$

| Répéter

| | considérer une place p marquée

| | Répéter

| | | Si p contient suffisamment de marques pour franchir une
| | | transition t et t ∉ $T_{EV,M}$
| | | Alors
| | | | vérifier condition de franchissement de t
| | | | Si ok
| | | | Alors
| | | | | $T_{EV,M} := T_{EV,M} \cup \{t\}$
| | | | Finsi
| | | Finsi
| | Jusqu'à toutes les transitions en sortie de p soient examinées
| Jusqu'à plus de places marquées non examinées
fin

En partant de l'ensemble des places marquées du réseau, cette procédure examine pour chaque place p de cet ensemble les transitions qu'elle valide. Pour chaque transition t validée par cette place, elle vérifie la **condition de franchissement** de cette transition. La condition de franchissement dépend des informations associées à la transition, plusieurs cas peuvent être considérés:

1- Ni réceptivité ni primitive de communication ne sont associées à la transition dans ce cas la condition de franchissement d'une transition est que :
- toutes les places en entrée soient suffisamment marquées, c'est-à-dire que le marquage de chaque place soit supérieur ou égal à la valuation de l'arc qui lie la place à la transition.

2- Une réceptivité (condition et/ou événement) est associée à la transition
la condition de franchissement pour ce type de transition est que:
-toutes les places en entrée soient suffisamment marquées,
-la condition associée à la transition soit vérifiée,
-et il existe une occurrence de l'événement consommable selon la politique de communication associée à la transition.

3- Une communication asynchrone est associée à la transition
3.1- cas de l'émission
la condition de franchissement est la même que celle du cas 2 ou du cas 1 selon qu'à la transition est associée une réceptivité ou non,

3.2- cas de la réception

la condition de franchissement est celle du cas 3.1 avec en plus la condition que le message soit recevable.

4- Une communication synchrone est associée à la transition

4.1- une réceptivité est associée à la transition

la condition de franchissement dans ce cas est que :

- toutes les places en entrée soient suffisamment marquées
- la réceptivité soit vérifiée : la condition est vraie et il existe une occurrence de l'événement consommable selon la politique de communication
- et le rendez-vous peut avoir lieu.

Un rendez-vous associé à un transition peut avoir lieu si chaque transition partenaire du rendez-vous est telle que:

- si une réceptivité est associée à la transition
  - les places en entrée de la transition sont suffisamment marquées et la réceptivité est vérifiée
  - ou la transition est dans un état bloqué en attente de la satisfaction du rendez-vous
- si aucune réceptivité n'est associée à la transition
  -les places en entrée sont suffisamment marquées.

Si les deux premières conditions sont vérifiées et que le rendez-vous n'est pas réalisable alors

- mettre t à l'état bloqué en attente de la satisfaction du rendez-vous
- et réserver les marques des places qui valident t

Finsi.

4.2- Aucune réceptivité n'est associée à la transition

la condition de franchissement dans ce cas est que :

- toutes les places en entrée soient suffisamment marquées
- et le rendez-vous soit réalisable.

Si le rendez-vous ne peut pas avoir lieu, la transition n'est pas bloquée comme dans le cas 4.1 car dans ce dernier cas la communication est considérée comme un événement (cf. chapitre 4 § 2).

Procédure de détermination d'une séquence de simulation complète

Cette procédure se compose de deux procédures : la procédure

déterminer-séquences qui détermine l'ensemble S des séquences de simulation possibles et la procédure déterminer-séquence-maximale qui choisit parmi les éléments de l'ensemble S une séquence de simulation complète.

1-procédure déterminer-séquences

Soit S l'ensemble des séquences de simulation, initialement S = ø

déterminer-séquence (TxEV,M)

début

| Si conflit ($T_{xEV,M}$, t_{c1}, t_{c2})

| Alors

| | éclater($T_{xEV,M}$, $T_{x1EV,M}$, $T_{x2EV,M}$)

| | déterminer-séquence ($T_{x1EV,M}$)

| | déterminer-séquence ($T_{x2EV,M}$)

| Sinon

| | s := $T_{xEV,M}$

| | S := S ∪ s

| Finsi

fin

Cette procédure commence par examiner l'ensemble des transitions de $T_{EV,M}$, si ces transitions ne sont pas en conflit effectif alors il existe une seule séquence de simulation qui est aussi la séquence de simulation complète s_c. L'ensemble S contient un seul élément qui est la séquence s. s comprend l'ensemble des transitions de $T_{EV,M}$.

Par contre, si un conflit effectif est détecté entre deux transitions t_i et t_j, alors ces transitions ne peuvent être franchies en même temps puisque le franchissement de l'une invalide le franchissement de l'autre. Ceci conduit à éclater $T_{EV,M}$ en deux sous- ensembles $T_{1EV,M}$ et $T_{2EV,M}$ et tel que :

$T_{1EV,M} = T_{EV,M}$ - {t_j et tous ses partenaires, dans le cas où à t_j est associée une communication par rendez-vous }

$T_{2EV,M} = T_{EV,M}$ - {t_i et tous ses partenaires}

La procédure reprend le même traitement pour chacun des sous-ensemble

$T_{iEV,M}$ (i = 1,2), elle vérifie s'il existe encore des conflits, si oui elle rééclate $T_{iEV,M}$ en deux autres sous ensembles $T_{i1EV,M}$ et $T_{i2EV,M}$ et refait le même traitement pour ces nouveaux sous ensembles (en s'appelant récursivement). Cette procédure s'arrête lorsqu'il n'y a plus de conflits dans aucun des sous-ensembles obtenus au cours de ce traitement. Cette procédure retourne dans l'ensemble S les différentes séquences possibles et tel que le franchissement des transitions de chaque séquence peut se faire dans n'importe quel ordre.
L'ensemble S est fourni à la procédure déterminer-séquence-maximale pour le choix de la séquence la plus complète.

2-déterminer-séquence-maximale

En partant de l'ensemble S des séquences de simulation, cette procédure calcule la longueur de chaque s appartenant à S. Ensuite en comparant ces longueurs entre elles, elle détermine la séquence de simulation complète s_c (séquence de longueur maximale). Dans le cas où plusieurs séquences ont la même longueur maximale, elle choisie de façon aléatoire une séquence.

Procédure franchir séquence de simulation complète

Cette procédure permet de franchir l'ensemble des transitions de la séquence s_c, pour chaque transition t de s_c on exécute l'algorithme suivant :

cas transition t **faire**

transition sans communication : franchir transition(t)

transition d'émission asynchrone : début
| -déposer le message dans un tampon;
| -franchir transition(t)
fin

transition de réception asynchrone: début
| - consommer le message;
| - franchir transition(t)
Fin

transition synchrone : franchir transition(T_{rv}) avec T_{rv} l'ensemble des transitions partenaires du rendez-vous

Fin cas

Procédure franchir transition

Elle permet :

- de déterminer le nouveau marquage du réseau
- d'exécuter les actions des places nouvellement marquées
- de vérifier les propriétés du réseau
- d'activer les générateurs d'événements endogènes dépendant du nouveau marquage
- de désactiver les générateurs d'événements endogènes continus dont la condition d'arrêt de génération est vérifiée par le nouveau marquage
- et de faire évoluer le réseau du flux d'activité dans le cas où l'évolution de ce réseau est synchronisé au franchissement de la transition.

Dans les lignes qui suivent, nous allons expliciter les différents points de cette procédure.

Calcul du nouveau marquage

Soient M : l'ancien marquage du RdPI

T_f : l'ensemble des transitions à franchir, cet ensemble est généralement formé d'une seule transition, sauf dans le cas de la communication par rendez-vous, il contient l'ensemble des partenaires du rendez-vous qui doivent être franchis en même temps

et M' : le nouveau marquage obtenu par le franchissement de T_f

M' est défini par :
pour toute transition t de T_f

pour toute place p en entrée de t : $M'(p) = M(p) - pré(p,t)$
pour toute place p en sortie de t : $M'(p) = M(p) + post(p,t)$

et pour le reste des places du réseau (ie. places n'appartenant ni à l'ensemble des places en entrée de T_f ni à l'ensemble des places en sortie de T_f) $M'(p) = M(p)$

Exécution des actions

Cette procédure permet d'exécuter les actions associées aux places qui viennent d'être marquées.
L'exécution des actions d'une place commence par l'accès au code de ces actions par l'intermédiaire d'un pointeur mémorisé dans la structure des places. Ensuite par l'évaluation de ce code.
Notons que l'exécution d'une opération de type armertemporisation(vt, n), consiste à insérer dans l'échéancier un événement vt qui deviendra vrai lorsque n unités de temps se seront écoulées.

NB Les actions continues sont exécutées à chaque cycle de simulation tant que les places auxquelles elles sont associées restent marquées.

Vérification dynamique des propriétés du RdPI

Le rôle de cette procédure est de contôler si le nouveau marquage M' vérifie les propriétés du réseau.
Propriété borné : le système ne prend en charge la vérification dynamique de cette propriété que si l'analyse statique n'a pas pu prouver que le réseau est borné.
La vérification de cette propriété pour le marquage M', consiste à montrer que le marquage M'(p) de chaque place p du réseau est inférieur ou égal à une borne donnée (cette borne est fournie par l'utilisateur).
En cas de violation de cette propriété, un message est affiché informant l'utilisateur de cet état. L'utilisateur pourra alors choisir d'arrêter ou de poursuivre la simulation.

Propriété de blocage : elle vérifie si le marquage M' peut conduire à un blocage (total ou partiel) ou non. Pour chaque verrou v de l'ensemble des verrous V du réseau, elle détermine si v est déficient pour M' ou non.
En cas de blocage total, un message est produit et la simulation est arrêtée.

Pour ce qui concerne les propriétés spécifiques exprimées par des invariants de places, elle évalue pour le marquage M' les expressions de ces invariants mémorisées sous la forme postfixée dans le tableau CODE et vérifie si ces expressions sont vraies ou fausses pour M'.
Pour les invariants de transitions, elle vérifie que l'ordre de franchissement des transitions est bien l'ordre spécifié par ces invariants.

Activation des générateurs d'événements

La procédure d'activation de générateurs commence par parcourir la liste des générateurs d'événements endogènes en attente d'activation et par vérifier pour le marquage M' la condition d'activation de chaque générateur. Si la condition est vérifiée, elle active alors le générateur. Celui ci engendre une occurrence de l'événement et l'introduit dans l'échéancier.
Tant que ce générateur est à l'état actif, et chaque fois qu'une occurrence de l'événement est prise en compte il engendre la prochaine occurrence.

Désactivation des générateurs d'événements

Cette procédure permet d'arrêter la génération des occurrences des événements endogènes continus dont la condition d'arrêt de génération est vérifiée par le marquage M'.
Ces générateurs passent de l'état actif à l'état en attente d'activation.

Faire évoluer le réseau du flux d'activité

Le rôle de cette procédure est de faire évoluer le réseau de files d'attente décrivant le flux d'activité et d'effectuer des calculs pour déterminer des statistiques sur les différentes files.
En considérant la description de l'écoulement du flux associée à l'événement déclencheur (tir d'une transition du RdPI ou occurrence d'un événement externe) elle effectue dans l'ordre les opérations suivantes :
1- déterminer les transactions à enlever des files en entrée de l'écoulement du flux
2- vérifier si ces transactions existent bien dans ces files
3- si oui, enlever ces transactions et effectuer des calculs sur ces files d'entrée
4- déterminer les transactions à produire dans les files en sortie
5- introduire ces transactions dans les files en sortie et effectuer des calculs sur ces files.
Notons que dans le cas d'une destruction des transactions (ie pas de files en sortie) seules les opérations 1, 2 et 3 seront exécutées. De même dans le cas d'une générations de transactions (ie pas de files en entrée) seules les opérations 4 et 5 seront exécutées.

7.3 LA PHASE DE GENERATION DES RESULTATS

Les résultats issus de la simulation sont nombreux et correspondent à :

- l'évolution du marquage du réseau,

- l'évolution de l'état du procédé,

- les statistiques sur les files d'attente, les places et les transitions,

- la vérification des propriétés (blocage, bornage, invariants de places et de transitions).

- les statistiques sur les communications dans le cas où le système est décrit par un réseau de modules communicants. Ces statistiques peuvent aider à préparer le niveau opérationnel de la démarche de conception.

8 CONCLUSION

Nous venons de voir l'outil d'aide au prototypage rapide de système de commande. Cet outil est réalisé sur un IMB PC AT en PASCAL. Son utilisation ne nécessite pas un apprentissage important. L'aide est intégrée, ce qui évite à un utilisateur débutant d'avoir à recourir à une documentation volumineuse dans laquelle il ne sait jamais où retrouver les informations pertinentes.
Le lecteur peut trouver dans l'annexe D différents scénarios d'utilisation de cet outil.

CONCLUSION

Le travail que nous avons présenté concerne la conception et la réalisation d'un outil d'aide au prototypage rapide des systèmes automatisés de production. Cet outil s'appuie sur une modélisation adéquate du système de commande, de l'environnement commandé et du flux d'activité pour réaliser une simulation hors site du système de commande.

Ce travail a été guidé par le souci de proposer un outil d'aide à la conception prenant en compte les critères suivants :

• être simple d'utilisation,

• permettre de tenir compte des spécificités des systèmes de commande,

• permettre d'aborder à partir d'une modélisation que le concepteur enrichit les problèmes de l'expression des besoins, la validation du système de commande et l'évaluation des performances, ce qui permet de renforcer la cohérence et la continuité entre les différentes étapes du cycle de développement d'un système.

Cet outil peut également intervenir au cours de l'exploitation d'un système pour l'aide à la décision. Il permet de tester rapidement les conséquences des différentes décisions que les responsables envisagent de prendre, ainsi que de mesurer l'impact d'une modification. Il peut aussi servir comme outil pédagogique. Il permet aux étudiants d'apprendre à spécifier et à concevoir des systèmes de commande.

En ce qui concerne les possibilités d'amélioration et de prolongement de notre travail, plusieurs points peuvent être traités :

• L'écriture d'une version du logiciel qui utilise les RdP colorés et interprétés car ces réseaux permettent de décrire un système de façon plus concise. Cela permettrait de posséder plusieurs versions de l'outil correspondant chacune à un type particulier de RdP.

• La prise en compte des durées de communication au cours de la simulation.

• La réalisation d'une version industrielle de l'outil en incluant le maximum de

facilités de mise en œuvre et en développant une interface graphique.

- L'étude des possibilités d'analyse des RdPI, en proposant des règles de transformation des RdPI en des RdP pour lesquels il existe des méthodes d'analyse.

BIBLIOGRAPHIE

[**Alaiwan 85**] H. Alaiwan, J.M. Toudic
Recherche des sémi-flots, des verrous et des trappes dans les réseaux de Petri.
TSI Vol 4, no. 1 ,1985, pp. 103-112.

[**Alanche 84**] P. Alanche, K. Benzakour, F. Dollé, P. Gillet, R. Rodrigues, R.Valette
PSI : A Petri Net based simulator for flexible manufacturing systems.
LNCS 188, Springer-Verlag, 1984, pp. 1-14.

[**Albukerque 82**] J. Albukerque
Spécification et validation d'automatismes logiques interconnectés.
Thèse de Docteur 3ème cycle, Toulouse, 1982.

[**Alla 84**] H. Alla, P. Ladet, F. Martin, J. Martinez, M. Silva
Modeling and validation of complex systems by coloured Petri nets . Application to a flexible manufacturing system.
LNCS 188, Springer-Verlag 1984, pp. 13-31.

[**Alla 86**] H. Alla, P. Ladet, F. Martin, J. Martinez, M. Silva
Spécification de la commande des ateliers flexibles à l'aide de réseaux de Petri colorés.
Journées d'études organisées par l'AFCET Automatique: Méthodes et outils modernes de conception et d'exploitation de la commande des procédés discontinus complexes, Montpellier, 1986, pp. 85-95.

[**Alla 87**] H. Alla
Réseaux de Petri colorés et réseaux de Petri continus : Application à l'étude des systèmes à événements discrets.
Thèse de Docteur d'Etat, Grenoble 1987.

[**André 81**] C. André
Systèmes à évolutions parallèles : modélisation par réseaux de Petri à capacité et analyse par abstraction.
Thèse de Docteur d'Etat ès-sciences, Nice, 1981.

[**Ayache 85**] J.M. Ayache, J.P. Courtiat, M. Diaz, G. Juanole
Utilisation des réseaux de Petri pour la modélisation et la validation des protocoles.
TSI Vol 4, no. 1, 1985, pp. 51-71.

[**Beaudouin 83**] M. Beaudouin-Lafon
PETRIPOTE : a graphic system for Petri net design and simulation.
4th European workshop on application and theory of Petri nets, Toulouse, Septembre 1983.

[**Bel 83**] G. Bel
Méthodes et langages de simulation pour la production automatisée : principes, choix, utilisation.
Congrès AFCET Automatique : Production et Robotique Intelligente, Besançon, 1983.

[**Bel et Dubois 85**] G. Bel, D. Dubois
Modélisation et simulation de systèmes automatisés de production.
APII, volume 19, no 1, 1985.

[**Benmaïza 84**] M. Benmaïza
Le concept d'événement dans la spécification et la programmation d'applications temps réel.
Thèse de Docteur Ingénieur, Nancy 1984.

[**Benzakour 86**] K. Benzakour, R. Valette
Simulation conjointe de commandes logiques et de systèmes mécaniques.
Journées d'études organisées par l'AFCET Automatique: Méthodes et outils modernes de conception et d'exploitation de la commande des procédés discontinus complexes, Montpellier, 1986, pp. 97-105.

[**Berthelot 83**] G. Berthelot
Transformation et analyse des réseaux de Petri, Applications aux protocoles.
Thèse de Docteur d'Etat ès-sciences, Paris VI, 1983.

[**Berthelot 84**] G. Berthelot
Transformations of Petri nets.
5th European workshop on application and theory of Petri Nets, june 1984.

[**Berthelot 85**] G. Berthelot
Transformations de réseaux de Petri.
TSI, Vol 4, no. 1, 1985, pp. 91-126.

[**Berthelot 85**] G. Berthelot
Analyse de processus parallèles par transformation de réseaux de Petri : Application à un protocole de réseau.
TSI, Vol 4, no. 1, 1985, pp. 73-83.

[**Berthomieu 79**] B. Berthomieu
Analyse structurelle des réseaux de Petri, methodes et outils.
Thèse Docteur Ingénieur, Toulouse 1979.

[**Blanchard 79**] M. Blanchard
Comprendre, maîtriser, et appliquer le GRAFCET.
Cepadus - éditions, 1979.

[**Brams 83**] G.W. Brams
Réseaux de Petri : Théorie et pratique
Edition Masson, 1983.

[**Buzen 80**] J.P. Buzen, P.J. Denning
Measuring and calculating queue length distributions.
IEEE Computer, Avril 1980.

[**Calvez 84**] J.P. Calvez, Y. Thomas
Méthodologie de conception pour les systèmes complexes de commande en temps réel.
R.A.I.R.O. Automatique/Systems Analysis and control, Vol 18, no. 2, 1984
pp. 251-266.

[**Campbell 74**] R. H. Campbell
The specification of processes synchronization by path expressions.
LNCS 16, Springer-Verlag, 1974, pp. 89-102.

[**Chen 83**] Bo-Shoe Chen, Raymond T. Yeh
Formal specification and verification of distributed systems.
IEEE Trans. on soft. eng., Vol SE-9, no. 6, 1983, pp. 710-722.

[**Chezalviel 79**] B. Chezalviel Pradin
Un outil interactif pour la vérification des systèmes à évolution parallèle décrits par réseaux de Petri.
Thèse de Docteur-Ingénieur, UPS Toulouse 1979.

[**Choppy et Johnen 85**] C. Choppy, C. Johnen
Petrireve: Proving Petri Net Properties with Rewriting System.
1st Conference on Rewriting Techniques and Applications, Lecture Notes in Computer Science, vol. 202, pp. 271-286, Springer Verlag, Dijon, 1985.

[**Choppy 86**] C. Choppy
Maquettage et prototypage : panorama des outils et des techniques.
Génie logiciel et systèmes experts no 11, Mars, 1988.

[**Chrétienne 84**] P. Chrétienne
Les réseaux de Petri temporisés.
Thèse de Docteur d'Etat ès-sciences, Université. P. et M. Curie , 1984.

[**Chrétienne 85**] P. Chrétienne
Analyse des régimes transitoire et asymptotique d'un graphe d'événements temporisé.
TSI Vol 4, no. 1, 1985, pp. 127-142.

[**Commoner 72**] F. Commoner
Deadlock in Petri Nets.
Applied data research Inc., Wakefield MASS, CA7206-2311, 1972.

[**Courtiat 87**] J.P. Courtiat, P. Demminski, R. Groz, C. Jard
Estelle : un langage ISO pour les algorithmes distribués et les protocoles.
TSI Vol 6, no. 2, 1987, pp. 89-102.

[**Courvoisier 83**] M. Courvoisier, R. Valette, J.M. Bigou, P. Esteban
Réseaux d'automates et ateliers flexibles.
Congrès AFCET Automatique : Productique et robotique intelligente", Besançon, 1983.

[**Dallery 84**] Y. Dallery
Une méthode analytique pour l'évaluation des performances d'un atelier flexible.
Thèse de Docteur Ingénieur, Grenoble, 1984.

[**Dennis 87**] A.R. Dennis, R.N. Burns, R.B. Gallupe
Phased design : a mixed methodology for application system development.
Data base Summer 1987 pp. 31-37.

[**Derniame 83**] J.C. Derniame, A. Zakari
Spécification d'application de conduite d'un atelier flexible.
Convention informatique Latine, Barcelone, 1983.

[**Deschizeaux 82**] P. Deschizeaux
Temps et événements dans les systèmes distribués de contrôle de procédé.
R.A.R.O. Automatique/Systems Analysis and Control, Vol 16, no.3, 1982, pp. 259-274.

[**Dijkstra 76**] E. W. Dijkstra
A discipline of programming.
Prentice-Hall, Englewood Cliffs, N.J., 1976.

[**Dubois et Cavaille 82**] D. Dubois, J.B. Cavaille
Théorie des réseaux de files d'attente et algorithmes pour l'évaluation des performances d'un atelier de production.
Rapport CERT-DERA, Mai 1982.

[**Dufau 84**] J. Dufau
Un outil pour la vérification des protocoles décrits par réseaux de Petri.
Thèse de Docteur Ingénieur, Toulouse, 1984.

[**El Fazziki 85**] A. El Fazziki
Contribution à la structuration et à la programmation des applications de contrôle de procédés industriels.
Thèse de Docteur de 3ème cycle, Nancy, 1985.

[**Florin et Natkin 85**] G. Florin, S. Natkin
Les réseaux de Petri stochastiques.
TSI Vol 4, no. 1, 1985, pp. 143-160.

[**Floyd 85**] C. Floyd
A systematic look at prototyping, in approaches to prototyping.
(Eds. R. Budde, K. Kuhlenkamp, L. Mathiasen, H. Züllinghoven), Springer-Verlag, 1984.

[**Gelenbe et al 80**] E. Gelenbe, J. Labetoulle, R. Marie, M. Metivier, G. Pujolle, W. Stewart.
Réseaux de files d'attente: modélisation et traitement numérique.
Edition hommes et techniques, 1980.

[**Gelenbe et Pujolle 85**] E. Gelenbe, G. Pujolle
Introduction aux réseaux de files d'attente.
Edition Eyrolles, 1985.

[**Genrich 81**] H.J. Genrich, K. Lautenbach
System modelling with high-level Petri Nets.
Theoritical Computer Science 13, 1981, pp. 109-136.

[**Genrich 80**] H.J. Genrich, K. Lautenbach, P.S. Thiagarajan
Elements of general net theory in net theory and applications.
LNCS 84, Springer-Verlag, 1980, pp. 21-164.

[**Giorgio 85**] B. Giorgio, M. Giuseppe
Rapid prototyping of control systems using high level Petr Nets.
IEEE on software engineering, 1985 pp. 230-235.

[**Gordon et Newell 67**] W.J. Gordon, G.F. Newell
Closed queuing systems with exponential servers.
Operations Research, 15, 1967, pp. 252-267.

[**Hack 72**] M. Hack
Analysis of production schemata by Petri nets.
MAC TR. 94, MIT, Février 1972.

[**Hoare 78**] C.A.R. Hoare
Communicating Sequential Processes.
CACM Vol 21, no. 8, 1978, pp. 666-677.

[**Hollinger 85**] D. Hollinger
Utilisation pratique des réseaux de Petri dans la conception des systèmes de production.
TSI Vol 4, no. 6, 1985, pp. 509-522.

[**Hollinger 87**] D. Hollinger
Réseaux de Petri et Flavors pour la spécification et la simulation de systèmes productiques.
Proc. des journées AFCET sur les langages orientés objets.
Bigre+Globule 48, 1986, pp. 206-215.

[**Holt 70**] A.W. Holt, F. Commoner
Events and Conditions, Record of the project MAC Conference on Concurrent Systems and Parallel Computation.
ACM, NEW YORK, 1970, pp. 3-52.

[**Huet et Lankford 78**] G. Huet, D.S. Lankford
On the uniform Halting problem for term rewriting system.
Rapport Laboria 283, Iria, Mars 1978.

[**Jackson 63**] J.R. Jackson
Jobshop like queuing systems
Management Sciences 10, 1963, pp. 131-142.

[**Jensen 81**] K. Jensen
Coloured Petri Nets and invariant method.
Theoritical Computer Science 14,North. Holland publ. co, 1981, pp. 317-336.

[**Johnson 83**] J.R Johnson
A ptototypical success story.
Data information, Nov. 1983, pp. 251-256.

[**Jouannaud et Lescanne 85**] J.P. Jouannaud, P. Lescanne
La réécriture.
Rapport CRIN 85-R-28, 1985.

[**Karp et Miller 69**] K.M. Karp, R.E. Miller
Parallel program schemata.
Journal of computer and system sciences, Vol 3, no. 2, 1969 pp. 147-195.

[**Keller 76**] R. M. Keller
Formal verification of parallel programs.
CACM Vol 19, no. 7, 1976, pp. 371-384.

[**Knuth et Bendix 70**] D. Knuth, P. Bendix
Simple word problems in universal algebras.
Computational Problems in Abstract Algebra Ed. Leech J. Pergamon Press 1970, pp. 263-297.

[**Kramer 80**] J. Kramer, A. Lister, J. Magee, M. Sloman
Distributede process control systems : programming and configuration
Rep. RR 80/12 Imp. College Londres, Mai, 1980.

[**Kramer 81**] J. Kramer, J. Magee, M. sloman
Intertask communication primitives for distributed computer control systems.
Proc. 2nd Int. Conf. on distr. Comp. Syst., Paris, 1981, pp. 1-8.

[**Kramer 83**] J. Kramer, J. Magee, M. Sloman, A. Lister
CONIC : An integrated approach to distributed computer control system.
IEEE Proc. , Vol 130, 1, 1983.

[**Krämer 87**] B. Krämer
Segras : A formal and semigraphique language combining Petri net and Abstract Data Type for the Specification of Distributed Systems.
Proc.9th international conference on software engineering, 1987
pp. 116-125.

[**Ladet 82**] P. Ladet
Contribution à l'étude des systèmes informatiques répartis pour la commande des procédés industriels.
Thèse d'Etat ès-sciences, Grenoble, 1982.

[**Lamport 83**] L. Lamport
What good is temporal logic.
Inf Proc. 83, R.E.A. North-Holland, 1983, pp. 657-668.

[**Lautenbach 74**] K. Lautenbach, H.A. Schmid
Use of Petri nets for proving correctness of concurrent process systems.
Proc. IFIP Congress 1974, North-Holland Publ. Co., 1974 pp. 187-191.

[**Lee 85**] S. Lee
On executable models for rule-based prototyping.
Proc. 8th Int. Conf. on Soft. Eng., London , 1985.

[**Leroudier 80**] J. Leroudier
La simulation à événements discrets.
Edition Hommes et Techniques, 1980.

[**Lescanne 83**] P. Lescanne
Computer experiments with the REVE term rewriting system generator.
10th ACM Conf. on Principles of Programming Languages Austin Texas, January 1983, pp. 99-108.

[**Lonchamp 87**] J. Lonchamp
Conception des applications informatiques réparties en commande de procédés industriels : une démarche, des outils.
Thèse de Docteur d'Etat ès-sciences, Nancy 1987.

[**Manna 84**] Z. Manna, P. Wolper
Synthesis of communicating processes from temporal logic specifications.
ACM Trans. on prog. lang. and syst., Vol 6, no. 1, 1984, pp. 68-93.

[**Martin 87**] F. Martin
Méthodologie de modélisation et simulation de systèmes complexes décrits par réseaux de Petri colorés.
Thèse de Docteur de l'Institut National Polytechnique de Grenoble, 1987.

[**Memmi 83**] G. Memmi
Méthodes d'analyse des réseaux de Petri, réseaux à files, et applications aux systèmes temps réel.
Thèse d'Etat, Paris VI, 1983.

[**Merlin 74**] P. Merlin
A study of the converability of computer system.
Thèse Ph.D. Computer Science University of California, Irvine, 1974.

[**Merlin 76**] P. Merlin, D.J. Farber
Recoverability of communication protocols. Implication of a theorical study.
IEEE Transactions on Comm. Vol 24, septembre 1976.

[**Moalla 78**] M. Moalla, J. Pulou, J. Sifakis
Réseaux de Petri synchronisés.
R.A.I.R.O. Automatique/Systems Analysis and control, Vol 12, no. 2, 1978, pp. 103-130.

[**Moalla 85**] M. Moalla
Réseaux de Petri interprétés et Grafcet.
TSI Vol 4, no. 1, 1985, pp. 17-30.

[**Montel 83**] B. Montel
OVIDE : a software package for the validation of systems represented by Petri nets based models.
4th European workshop on application and theory of Petri nets, Toulouse, Septembre, 1983.

[**Owicki 82**] S. Owicki, L. Lamport
Proving liveness properties of concurrent programs.
ACM Trans. on progr. lang. and syst. , Vol 4, no 3, 1982, pp. 455-495.

[**Pascal 87**] J.C. Pascal, J.M. Pons, Y. Bandin, M. Courvoisier, R. Valette
Programmation modulaire d'une application distribuée dans le contexte du projet S.E.CO.I.A.
Journées d'étude S.E.E, Gif-sur-Yvette, 1987.

[**Perrin 85**] G. R. Perrin
La communication : Un outil pour la spécification, la construction et la vérification de systèmes parallèles.
Thèse de Docteur d'Etat ès-sciences, Nancy, 1985.

[**Peterson 77**] J.L. Peterson
Petri Nets,
ACM Comp. Surveys, Vol. 9, n° 3, Septembre 1977, pp. 223-251.

[**Peterson 81**] J.L. Peterson
Petri Net theory and the modeling of systems.
Prentice-Hall, inc., Englewood Cliffs, N.J. 07632, 1981.

[**Petri 62**] C.A. Petri
Kommunikation mit automaten, Schriften des Rheinisch-Westfalischen Institutes für instrumentelle Mathematik an der Universitat Bonn, 1962 (Thèse traduite en anglais dans Communication with automata, Tech. Rep. n° RADC-TR-65-337, Vol. 1, Rome Air Develop. Center, Griffis Air Force Base, NEW YORK, Janvier 1966.

[**Ramchandani 73**] C. Ramchandani
Analysis of asynchronous concurrent systems by timed Petri nets.
PhD. Thesis, M.I.T, Septembre 1973.

[**Raynal 81**] M. Raynal
Contribution à l'étude de la coopération entre processus dans les langages et systèmes informatiques.
Thèse de Docteur d'Etat ès-sciences, Rennes I, 1981.

[**Sifakis 77**] J. Sifakis
Use of Petri nets for performance evaluation.
Measuring, Modeling and Evalaution Computer Systemes, North-Holland Publ. Co., 1977, pp. 75-93.

[**Sifakis 79**] J. Sifakis
Contrôle des systèmes asynchrones : concepts, propriétés, analyse statique.
Thèse de Docteur d'Etat ès-sciences, Grenoble, 1979.

[**Stallman 84**] R.M. Stallman 84
Emacs : The Extensible, customizable, self documenting display Editor.
Interactive programming Environments, ed David R. Barstow, Howard E, Erik Sandewall, Mc Graw Hill, 1984, pp. 300-325.

[**Tankoano 88**] J. Tankoano
M2C: Une approche méthodique pour la conception certifiée des systèmes de commande des automatismes industriels répartis.
Thèse de Docteur d'Etat ès-sciences, Nancy, 1988.

[**Thomesse 80**] J.P. Thomesse
SYGARE : Une structuration pour la conception d'applications en temps réel et réparties.
Thèse de Docteur d'Etat ès-sciences, Nancy, 1980.

[**Toudic 82**] J.M. Toudic
Algorithmes d'algèbre linéaire pour l'analyse structurelle des réseaux de Petri. Revue technique Thomson-CSF, 14 (1), Mars 1982.

[**Touzani 90**] A. Touzani
Thèse de l'Université de Nancy 1 à paraitre 1990.

[**Travendale 85**] R.D. Travendale
A technique for prototyping directly from a specification.
IEEE on software engineering 1985, pp. 224-229.

[**Treille 87**] A. Treille
Réalisation d'un éditeur graphique pour réseaux de Petri interprétés.
Rapport de DESS Informatique Option Génie logiciel, 1987.

[**TSI 85**] Numéro spécial réseaux de Petri.
Vol 4, no 1, 1985.

[**Valette 76**] R. Valette
Sur la description, l'analyse et la validation des systèmes de commande parallèles.
Thèse de Docteur d'Etat ès-sciences, Toulouse, 1976.

[**Valette 82**] R. Valette, M. Courvoisier, J.M. Bigou
Les réseaux d'automates : analyse de la coopération.
Journées d'études S.E.E., Gif-sur-Yvette, 1982, p.733-740.

[**Valette 84**] R. Valette, K. Benzakour
Simulateur P.S.I.
Rapport Scientifique 84.065, Septembre 1984.

[**Valette 85**] R. Valette, V. Thomas, S. Bachmann
SEDRIC : un simulateur à évènements discrets basé sur les réseaux de Petri.
APII Informatique industrielle, 19, 1985, pp. 423-436.

[**Zakari 84**] A. Zakari
FLEXI : Langage de conception d'applications de conduite de procédés industriels.
Thèse de Docteur de 3ème cycle, Nancy, 1984.

ANNEXE A

LES MOTS CLES DU LANGAGE

ACTION
ACTIONS
ALEATOIRE
ALORS
ARCS
ARMERTEMPORISATION
BASCULE
BOOLEEN
C
CALCULEES
CONDITIONS
CONSOMMATION
DANS
DE
DEBUT
DESARMERTEMPORISATION
EGALITE
ET
ENTIER
ENTREE
EVENEMENTS
ENVIRONNEMENT
FAUX
FENVIRONNEMENT
FFLUX
FIFO
FILES
FIN
FINTERPRETATION
FLUX
FRDFA
FRDP
FRDPI
FSI
INFINI
INTERPRETATION
LIFO
NŒUDS
OCCURRENCE
OU
PAS
PG
PLACE
PRIOR
PRIORITE
PRODUIRE
RAFRAICHIE
RAND
RDFA
RDP
RDPI
RECEPTIVITE
REEL
RETIRER
SI
SORTIE
STRICTEMENTRAFRAICHIE
SUR
TELQUE
TEMPORISATION
TIR
TRANSACTIONS
TRANSITION
√ARIABLES
VRAI

ANNEXE B

LISTE DES ERREURS

100: "RDPI" attendu
101: "FRDPI" attendu
104: "FRDP" attendu
110: "PLACE" ou "TRANSITION" attendu
112: "ENTREE" ou "SORTIE" attendu
117: "FENVIRONNEMENT" attendu
122: identificateur attendu
123: constante attendu
131: "ENTIER", "REEL" ou "BOOLEEN" attendu
145: ":=" attendu
147: "ALORS" attendu
148: "FSI" attendu
184: "," attendu
185: ";" attendu
186: "." attendu
187: ":" attendu
188: "=" attendu
195: "(" attendu
196: ")" attendu
197: "*" attendu
202: "FINTERPRETATION" attendu
208: "FIN" attendu
209: "SUR" attendu
210: "CONSOMMATION" attendu
211: "DE" attendu
212: "RECEPTIVITE" ou "ACTION" attendu
213: "[" attendu
214: "]" attendu
215: "OCCURRENCE" attendu
300: identificateur,"SI","ARMERTEMPORISATION","DESARMERTEMPORISATION" ou "DEBUT" attendu
301: variable calculée ou de sortie attendu
302: nom de variable ou de condition ou d'événement attendu
303: variable, constante, "(", "PAS", "\^" ou "\!" attendu
304: variable de temporisation attendu
305: "EGALITE", "RAFRAICHIE", "STRICTEMENTRAFRAICHIE", ALEATOIRE" ou "BASCULE" attendu

306: variable non déclarée
307: identificateur, armertemporisation, desarmertemporisation ou si attendu
308: identificateur ou "DEBUT" attendu
499: symbole non attendu
500: identificateur dans "FRDPI" doit être le même que dans "RDPI"
501: les sections réseau, environnement, interprétation et réseau de files doivent être définies une fois au plus et dans cet ordre
502: les variables d'entrée, de sortie, calculées et de temporisation doivent être déclarées dans cet ordre
503: les variables, les conditions, les événements et les actions doivent être déclarées dans cet ordre
504: identificateur déjà déclaré
505: une variable de temporisation doit être de type booléen
506: expression conditionnelle attendu
507: incompatibilité de type
508: variable modifiable ou nom d'action attendu
509: expression arithmétique attendu
510: les nœuds et les arcs doivent être déclarés dans cet ordre
511: nom de transition attendu
512: nom de place attendu
513: expression événementielle attendu
514: réceptivité déjà associée à une transition
515: expression événementielle ou conditionnelle attendu
516: action déjà associée à une place
517: nom d'action attendu
600: "FRDFA" attendu
601: "FLUX" attendu
602: "FFLUX" attendu
603: "FILES" attendu
604: "TRANSACTIONS" attendu
610: "OCCURRENCE" ou "TIR" attendu
611: "DANS" attendu
612: "PRODUIRE" attendu
613: "RETIRER" attendu
614: les files, les transactions et les flux doivent être déclarés dans cet ordre
615: paramètre attendu
616: paramètre déjà déclaré

617: double définition de la transaction
618: nom de transaction attendu
619: file déjà déclarée dans le flux
620: nom de file attendu
621: transaction n'appartenant pas aux transactions de l'écoulement du flux
622: nom d'événement attendu
623: nom de transition attendu
624: le paramètre n'appartient pas à la liste des paramètres de la transaction
625: définition incorrecte du flux.

ANNEXE C

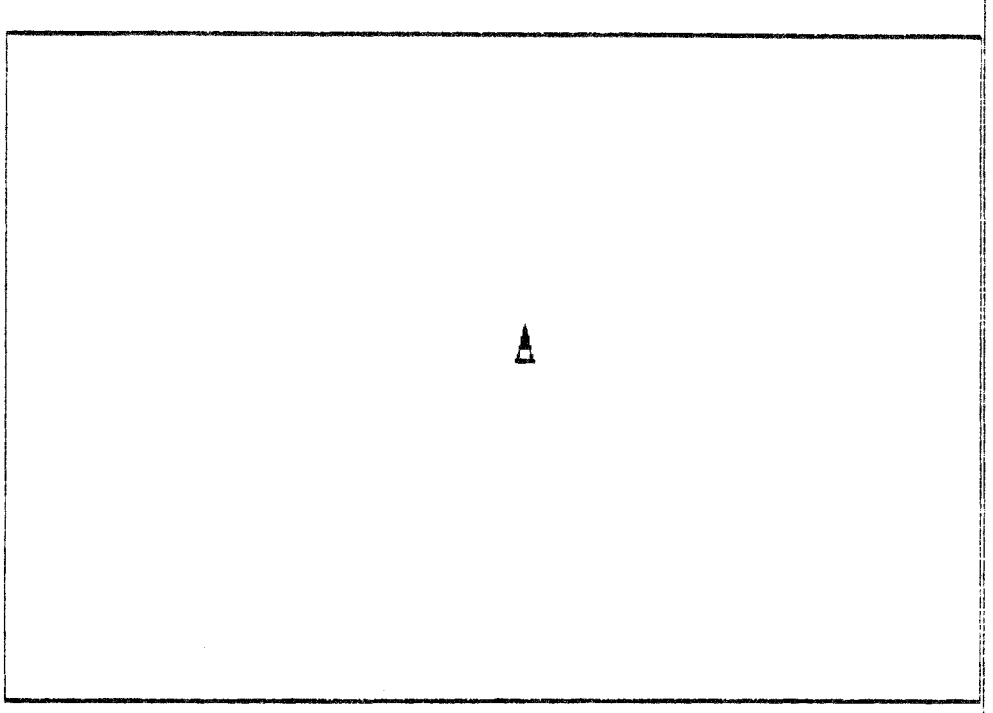
SCENARIOS D'UTILISATION DE L'EDITEUR GRAPHIQUE

Cette annexe montre un scénario de construction d'un RdPI avec PETRI-EDIT.
L'exemple présenté est celui de la pompe de drainage dans une mine de charbon(cf. chapitre 4 §2.3.2). Nous construisons que le RdPI qui décrit comment la mise en route et l'arrêt de la pompe alternent en fonction du niveau de l'eau et du niveau de méthane dans la galerie.

ECRAN 1: Menu principal

L'écran 1 montre les quatre fenêtres:
•la fenêtre supérieure qui est la fenêtre de dialogue,
•la fenêtre sur la droite où apparaît le menu principal,
•la fenêtre centrale qui est la fenêtre de dessin. Dans cette fenêtre apparaît la tortue qui remplace la souris que nous n'avions pas à notre disposition sur le matériel utilisé,
•et la fenêtre en bas de l'écran qui fournit des informations sur le déplacement du réseau dans la fenêtre de dessin.

Faites un choix



MENU GENERAL

creer_M.odif

S.auver

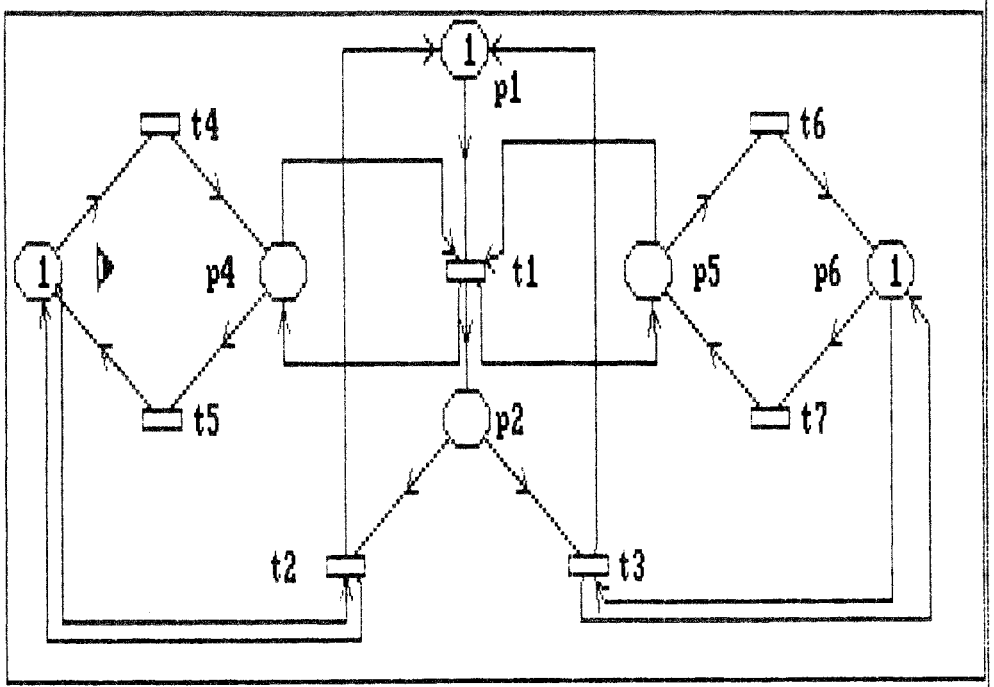
C.harger

F.in

ECRAN 2: Création d'un réseau

En tapant la lettre "M" du menu principal, l'utilisateur est placer sous le menu de création et modification d'un réseau. L'utilisateur pourra ainsi créer son réseau.
L'écran 2 montre le réseau en cours de construction plus précisément l'exécution de la fonction de nommage pour la place p3.

Nom initialise a :
Entrez le nouveau nom et tapez 'RETURN' : p3
Voulez_vous le visualiser sur le dessin o/n ?o
Montrez l'emplacement du nom (F2)



SCROLLING --> vertical : 0 horizontal : 0

MENU

PLACE ou TRANS

creer P.lace
creer T.rans
N.ommer
M.arquer
D.eplacer
S.upprimer

ARCS

creer aR.c
detrU.ire arc
mO.difier arc
V.aluer arc

E.nviron
I.nterpre

F.in travail

ECRANS 3, 4, 5, 6, 7, 8: Déclaration des variables

Les écrans 3, 4 montrent respectivement la déclaration de la variable d'entrée Nmeth et celle de la variable d'entrée Neau.

L'écran 5, indique que la création de ces variables s'est bien déroulée.

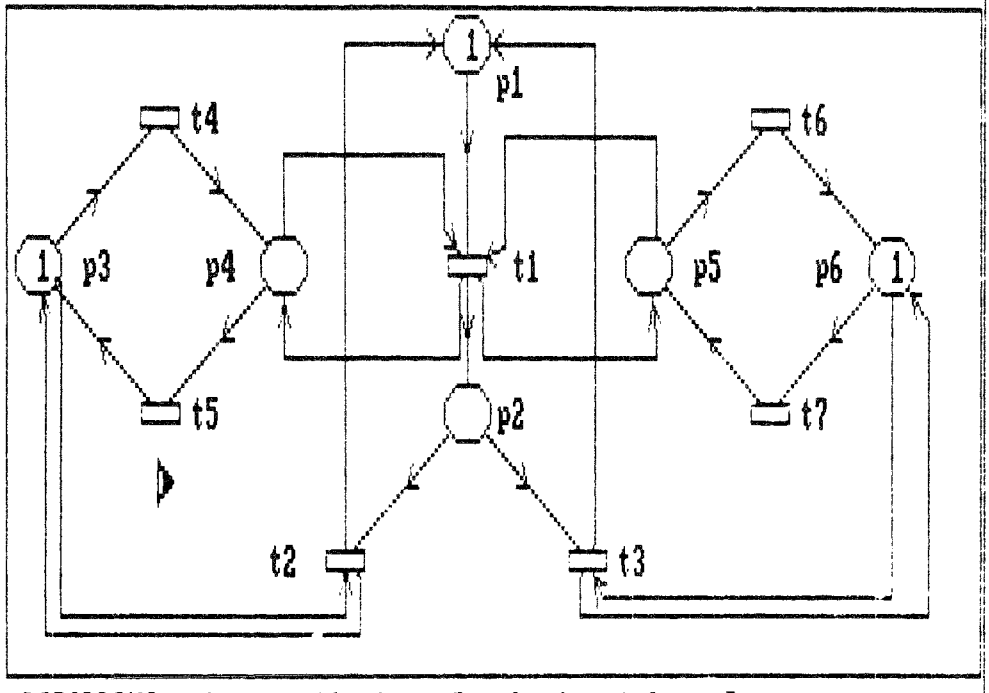
L'écran 6 montre la déclaration de la variable de sortie CMDpompe.

L'utilisateur pourra ensuite visualiser les déclarations des variables. Pour cela, il sélectionne d'abord la commande Visualiser du menu, ensuite il indique la classe de variables, c'est-à-dire s'il s'agit de variables d'entrée, de sortie, calculée ou de temporisation.
Après décompilation, l'éditeur affiche dans la fenêtre supérieure la déclaration des variables de la classe choisie.

L'écran 7 et 8 présentent respectivement dans la fenêtre supérieure de l'écran, les déclarations des variables d'entrée et de la variable de sortie du réseau.

CREATION D'UNE VARIABLE
Entrez le nom de la variable puis (RETURN) : Nmeth

Entrez le TYPE de cette variable puis RETURN : 1
1. entier 2. boolean

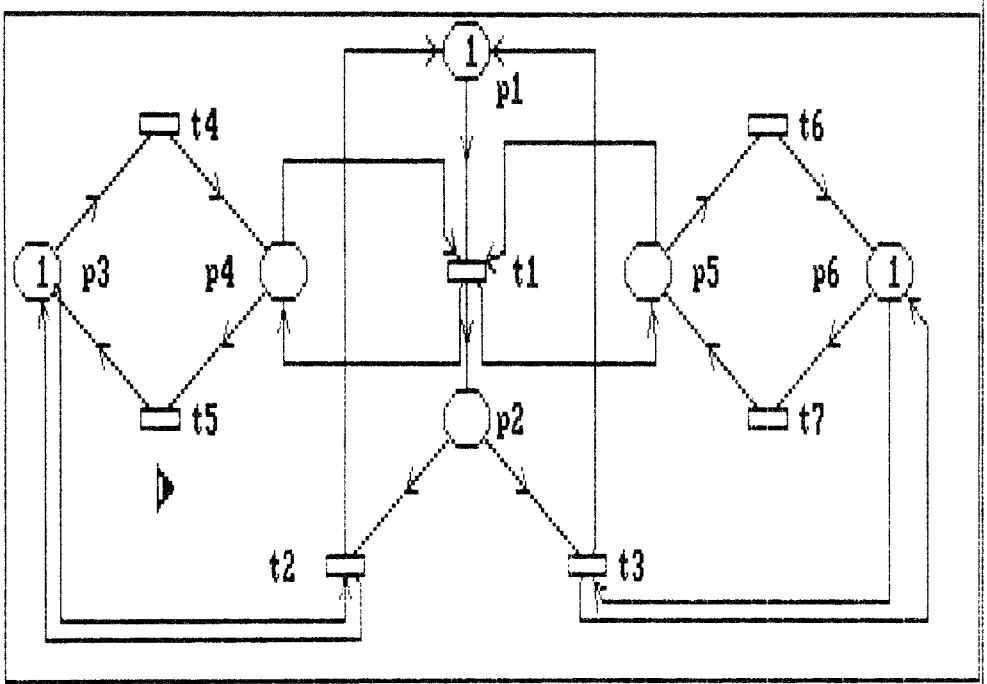


SCROLLING --> vertical : 0 horizontal : 0

MENU
"VARIABLE"
QUEL ENSEMBLE?
1. en entree
2. en sortie
3. calculee
4. tempori.

CREATION D'UNE VARIABLE
Entrez le nom de la variable puis (RETURN) : Neau

Entrez le TYPE de cette variable puis RETURN : 1
1. entier 2. boolean



SCROLLING --> vertical : 0 horizontal : 0

MENU

"VARIABLE"

QUEL ENSEMBLE?

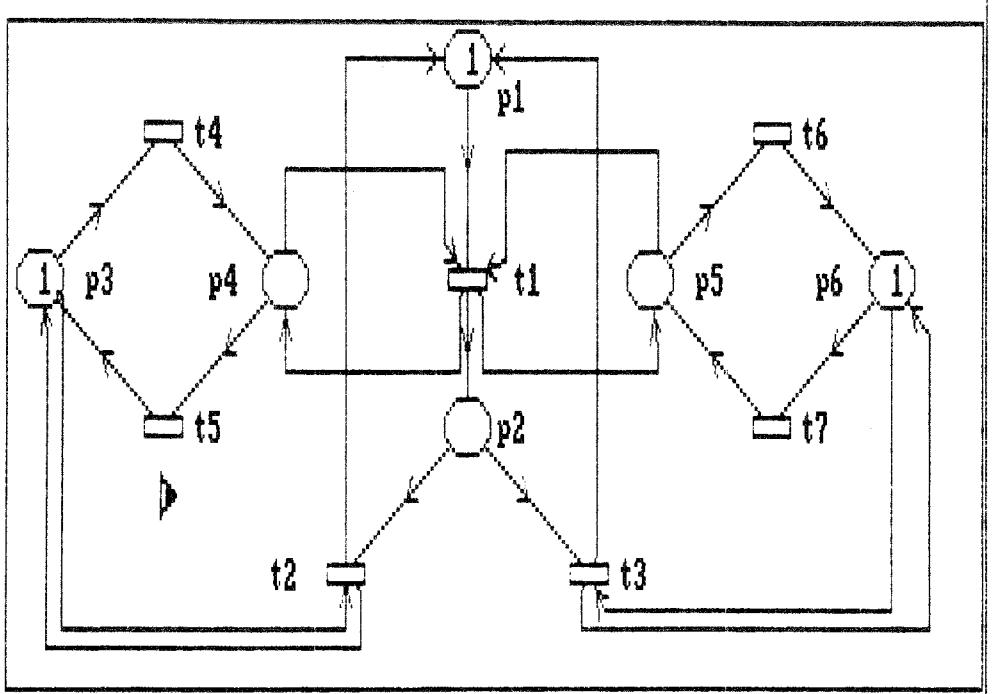
1. en entree

2. en sortie

3. calculee

4. tempori.

CREATION EFFECTUEE AVEC SUCCES



SCROLLING --> vertical : 0 horizontal : 0

MENU
ENVIRONNEMENT

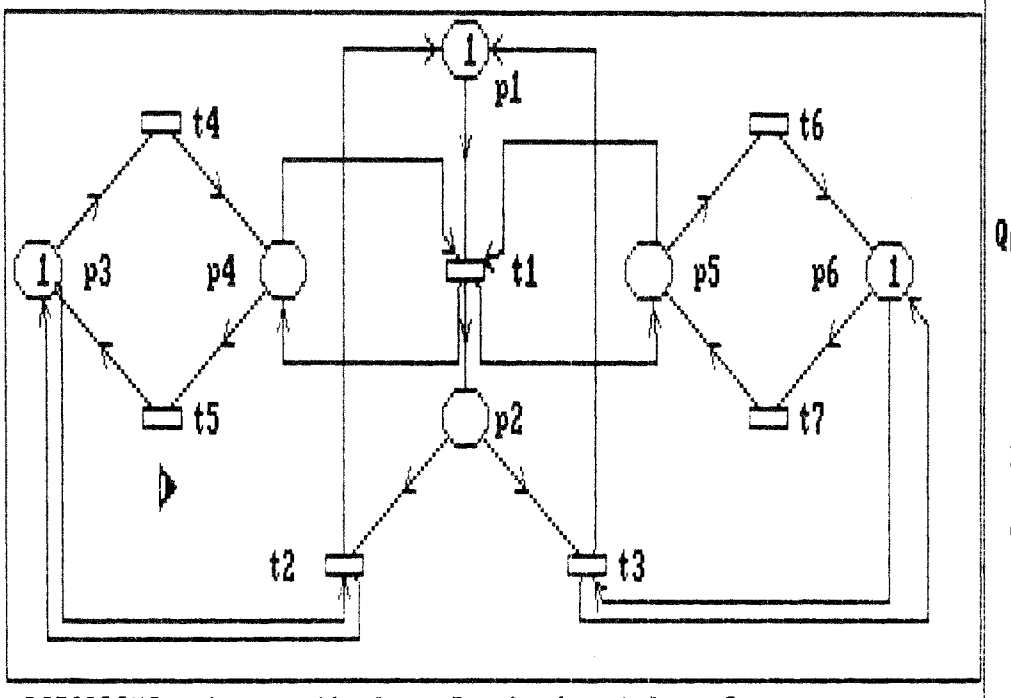
"VARIABLE"

C.reer
V.isualiser
M.odifier
detr.U.ire

F.in travail

CREATION D'UNE VARIABLE
Entrez le nom de la variable puis (RETURN) : CMDpompe

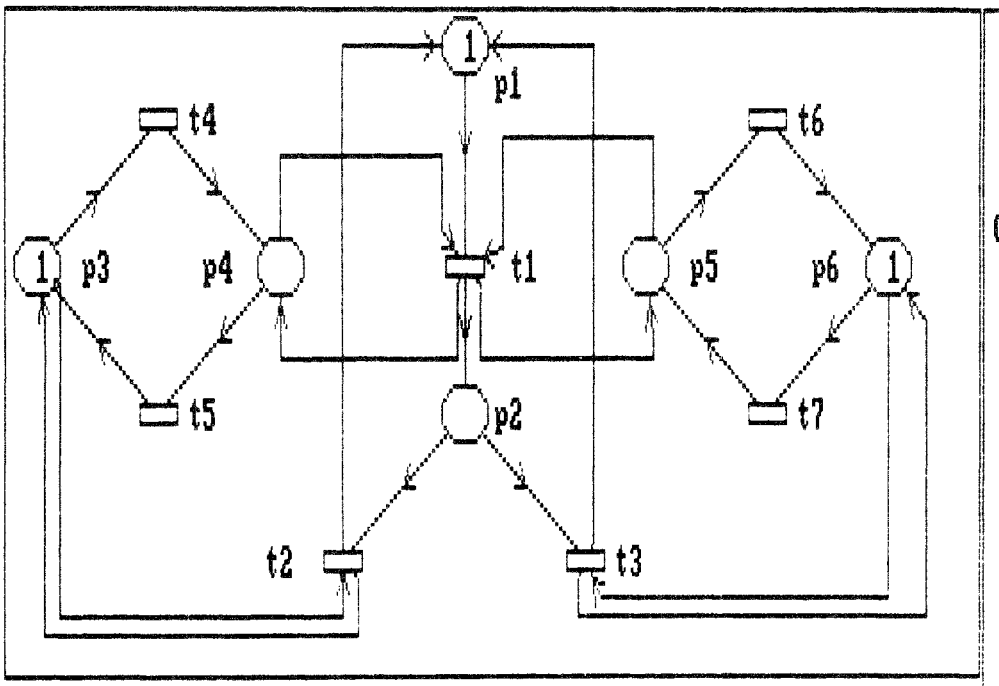
Entrez le TYPE de cette variable puis RETURN : 2
1. entier 2. boolean



SCROLLING --> vertical : 0 horizontal : 0

MENU
"VARIABLE"
QUEL ENSEMBLE?
1. en entree
2. en sortie
3. calculee
4. tempori.

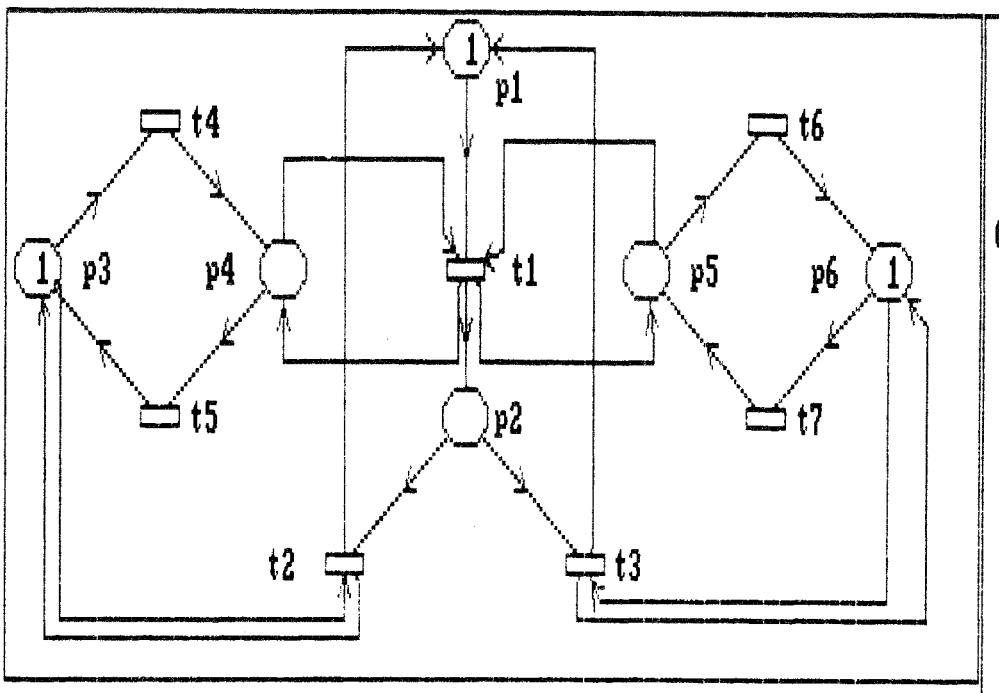
```
EDITION TEXTE  :  (END) pour sortir
entree NMETH, NEAU : entier ;
```




```
SCROLLING -->   vertical :  0   horizontal :  0
```

```
   MENU
"VARIABLE"
QUEL ENSEMBLE?
1. en entree
2. en sortie
3. calculee
4. tempori.
```

```
EDITION TEXTE  :  (END) pour sortir
sortie CMDPOMPE : booleen ;
```




```
SCROLLING -->   vertical :  0   horizontal :  0
```

```
    MENU
"VARIABLE"
QUEL ENSEMBLE?
1. en entree
2. en sortie
3. calculee
4. tempori.
```

ECRAN : Interprétation

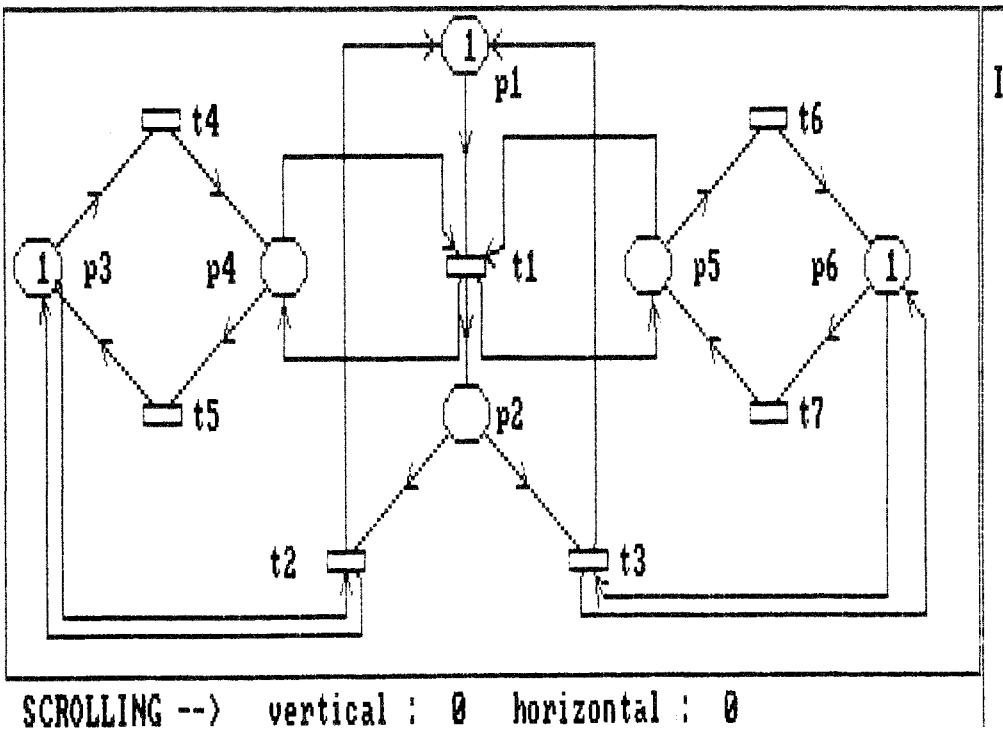
L'écran 9 présente un cas de création de conditions.
Pour créer une condition l'utilisateur est placé sous un mini-éditeur de texte (qui apparaît dans la fenêtre supérieure de l'écran) pour taper son texte.
Ce texte comporte le nom de la condition et son expression booléenne.
Lorsque l'utilisateur quitte le mini-éditeur de texte, la compilation de la condition est lancée.
L'écran 10 indique que la compilation s'est bien effectuée.

L'utilisateur peut ensuite lier la condition à une ou plusieurs transitions du réseau.
L'écran 11 montre la liaison de cette condition avec la transition t4 (transition désignée par la tortue).

De même, les écrans 12, 13 et 14 présentent respectivement la création de l'action ACT1, le succès de cette création et l'association de cette action à la place désignée par la tortue (association de ACT1= CMDpompe := vrai à la place p2 du réseau).

Le dernier écran montre les différentes actions qui ont été créées.

EDITION : destruction ligne (F1) , fin (END)
condi = Nmeth < 50 ; Δ



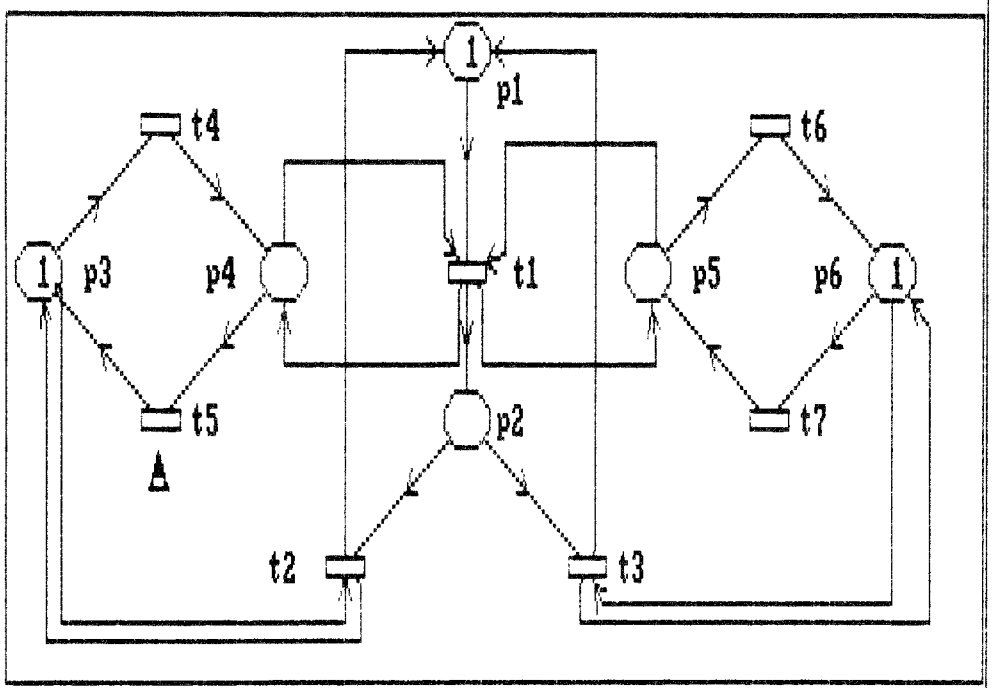
SCROLLING --> vertical : 0 horizontal : 0

MENU
INTERPRETATION

"CONDITION"

C.reer
L.ier
D.elier
V.isualiser
M.odifier
detr.U.ire
F.in travail

LA COMPILATION S'EST BIEN DEROULEE



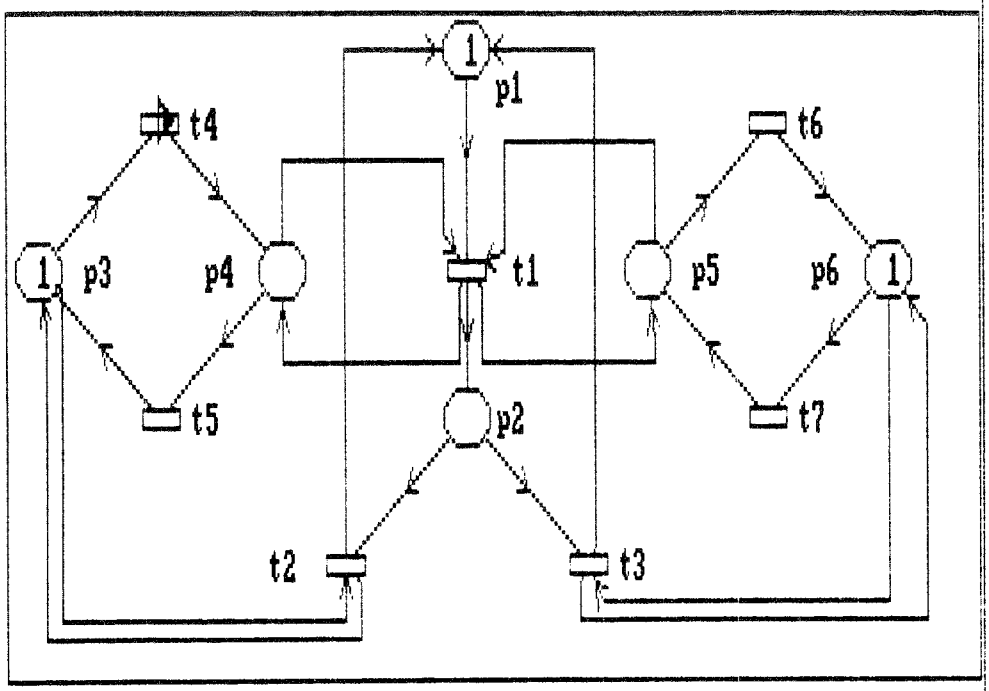
SCROLLING --> vertical : 0 horizontal : 0

MENU
INTERPRETATION

"CONDITION"

C.reer
L.ier
D.elier
V.isualiser
M.odifier
detr.U.ire
F.in travail

LIAISON D'UN ENVIRONNEMENT A UN COMPOSANT
Entrez le nom de la CONDITION puis (RETURN) : condi
Montrez avec quelle transition vous vouler travailler (F3)



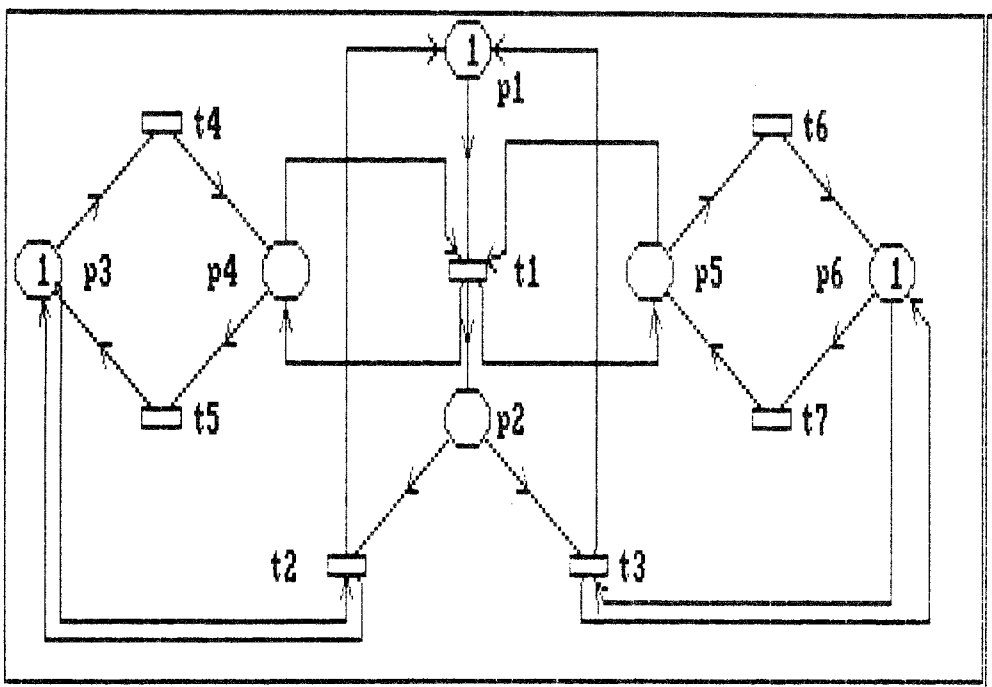
SCROLLING --> vertical : 0 horizontal : 0

MENU
INTERPRETATION

"CONDITION"

C.reer
L.ier
D.elier
V.isualiser
M.odifier
detr.U.ire
F.in travail

EDITION : destruction ligne (F1) , fin (END)
act1 = CMDpompe := vrai ; Δ



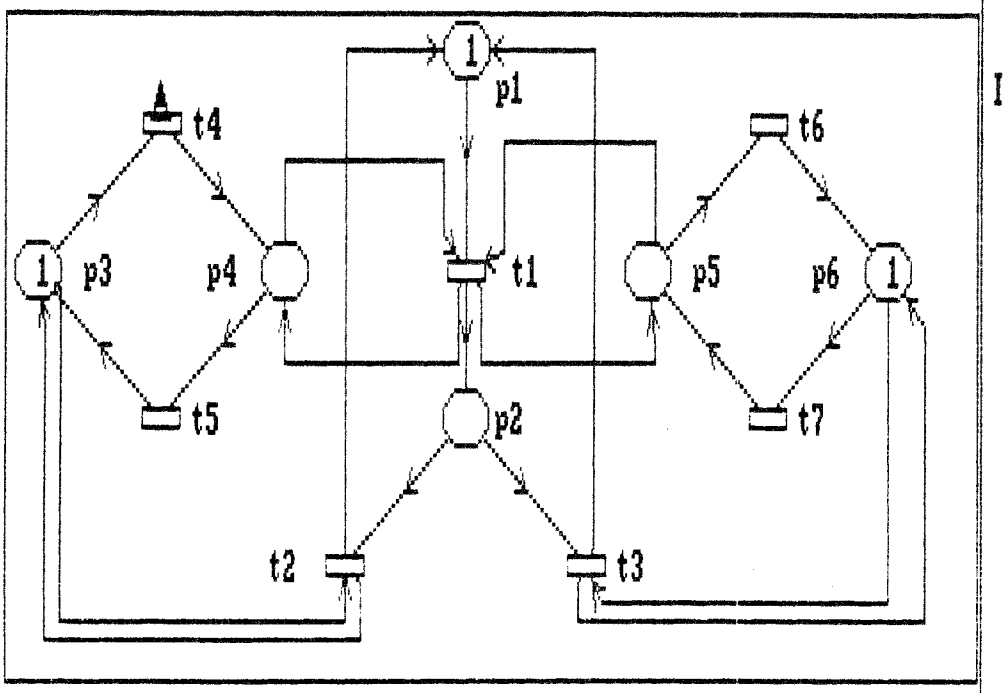
SCROLLING --> vertical : 0 horizontal : 0

MENU
INTERPRETATION

"ACTION"

C.reer
L.ier
D.elier
V.isualiser
M.odifier
detr.U.ire
F.in travail

LA COMPILATION S'EST BIEN DEROULEE



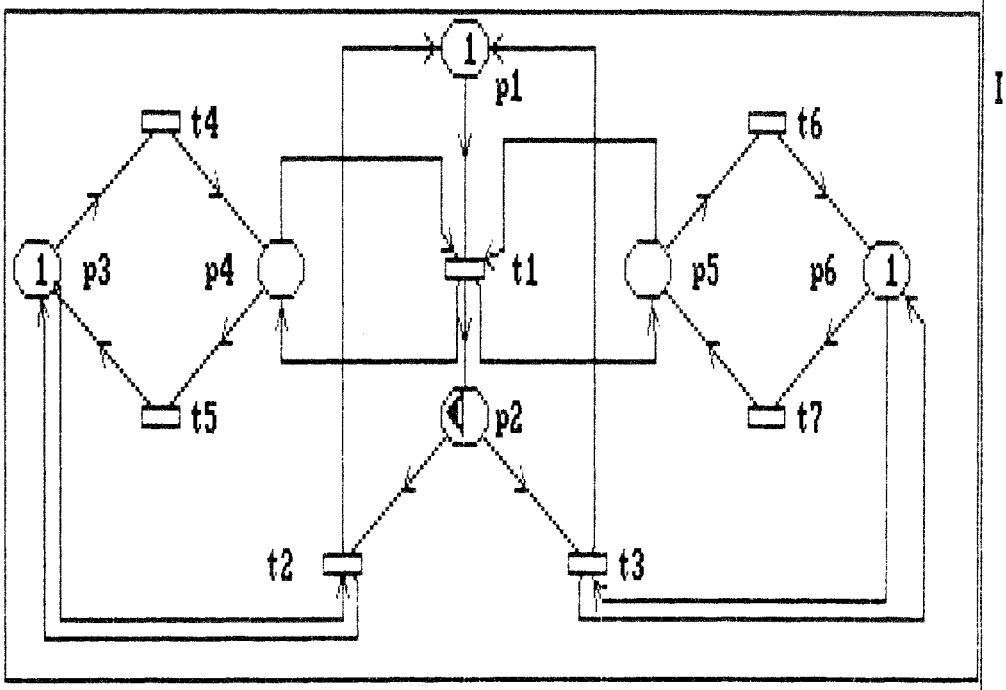
SCROLLING --> vertical : 0 horizontal : 0

MENU INTERPRETATION

"ACTION"

C.reer
L.ier
D.elier
V.isualiser
M.odifier
detr.U.ire
F.in travail

LIAISON D'UN ENVIRONNEMENT A UN COMPOSANT
Entrez le nom de l'ACTION puis (RETURN) : act1
Montrez avec quelle place vous vouler travailler (F3)
classe de l'ACTION ? (1.continue, 2.impulsionnelle :



SCROLLING --> vertical : 0 horizontal : 0

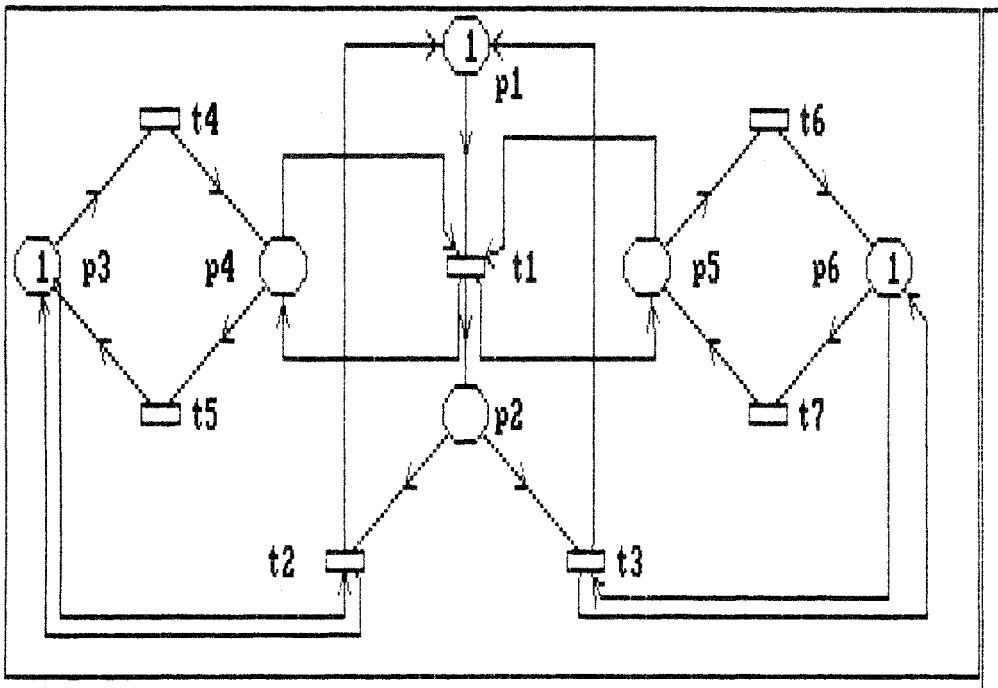
MENU
INTERPRETATION

"ACTION"

C.reer
L.ier
D.elier
V.isualiser
M.odifier
detr.U.ire
F.in travail

```
EDITION TEXTE  :  (END) pour sortir

  ACT1 = CMDPOMPE := vrai  ;
  ACT2 = CMDPOMPE := faux  ;
```




```
SCROLLING -->  vertical :  0  horizontal :  0
```

```
MENU

"ACTION"

D.'une place
L.'ensemble
U.ne action
```

ANNEXE D

SCENARIOS D'UTILISATION DE PETRI-S

Cette annexe présente des scénarios d'utilisation de PETRI-S
L'exemple présenté est celui de l'atelier de bobinage (cf. chapitre 4 §2.3.1).
Nous nous limitons au cas d'un seul bobinoir car nous ne pouvons pas déclarer des variables de type tableau.

La figure 1 présente la description textuelle du RdPI modélisant le fonctionnement de l'atelier.
Pour lancer l'outil, l'utilisateur tape PETRIS et donne le nom du fichier contenant la description du réseau. A la suite de ça, le menu principal s'affiche sur l'écran (voir écran 1).

```
rdpi atelierbobinage;

  rdp

    noeuds p1(1),p2,p3,p4,p5,p6,p7,p8,p9(1),p10,p11,p12,p13,p14,p15 : place;
           t1,t2,t3,t6,t7,t8,t9,t10,t11,t12,t13,t14,t15,t4,t5: transition;


    arcs   entree t1 = p1,p7;
           sortie t1 = p2;
           entree t2 = p2;
           sortie t2 = p3;
           entree t3 = p3,p8;
           sortie t3 = p5;
           entree t4 = p3,p12;
           sortie t4 = p4;
           entree t5 = p4;
           sortie t5 = p5,p14;
           entree t6 = p5;
           sortie t6 = p6;
           entree t7 = p6;
           sortie t7 = p1;
           entree t8 = p9;
           sortie t8 = p7,p10;
           entree t9 = p10;
           sortie t9 = p11;
           entree t10= p10;
           sortie t10= p8, p13;
           entree t11= p11;
           sortie t11= p12;
           entree t12= p11;
           sortie t12= p8,p14;
           entree t13= p13;
           sortie t13= p14;
           entree t14= p14;
           sortie t14= p15;
           entree t15= p15;
           sortie t15= p9;

  frdp;

  environnement

    variables

       entree ereq,ecasse,efech,efbob,efdepse,efretse,efdepn,efretn : booleen;

       sortie adepse,aretse,aarrse,acf,aech,adepn,aretn,aarrn : booleen;


    conditions

           cond1 = ereq ;
           cond2 = efbob ;
           cond3 = ecasse ;

    evenements

           evt1 = \^ efdepn ;
           evt2 = \^ efretn ;
           evt3 = \^ efdepse ;
           evt4 = \^ efretse;
           evt5 = \^ efech;
```

```
    actions

            act1 = aarrn := vrai ;
            act2 = adepn := vrai ;
            act3 = aretn := vrai ;
            act4 = aech := vrai ;
            act5 = aarrse := vrai ;
            act6 = adepse := vrai ;
            act7 = acf := vrai ;
            act8 = aretse := vrai;
            act9 = aretn := faux;
            act10= aarrn := faux;
            act11= adepn := faux;
            act12= aech := faux;
            act13= aretse := faux;
            act14= aarrse := faux;
            act15= adepse := faux;
            act16= acf := faux;

  fenvironnement ;

  interpretation

  receptivite t2= sur consommation strictementrafraichie de occurrence de evt1;
  receptivite t7= sur consommation strictementrafraichie de occurrence de evt2;
  receptivite t8 = si cond1 ;
  receptivite t9= sur consommation strictementrafraichie de occurrence de evt3;
  receptivite t15=sur consommation strictementrafraichie de occurrence de evt4;
  receptivite t11 = si cond2 ;
  receptivite t10,t12 = si cond3 ;
  action p1 = [act1,act9] ;
  action p2 = [act2,act10] ;
  action p3 = [act1,act11] ;
  action p4 = [act4] ;
  action p6 = [act3,act10] ;
  action p9 = [act5,act13];
  action p5,p14 = [act16,act12] ;
  action p10 = [act6,act14];
  action p11,p13 = [act5,act15];
  action p12 = [act7] ;
  action p15 = [act8,act14] ;

  finterpretation ;

frdpi atelierbobinage.
```

RETURN : selection ESC : fenetre suivante CTRL/END : fin

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

Instancier generateurs d evenements
Detruire schema de generation d evenements
Lister schemas de generation d evenements
Charger schemas de generation d evenements
Sauvegarder schemas de generation
Creer evenement
Lister evenements
Quitter

RETURN : selection ESC : aide

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

Endogene binaire
Endogene continu
Purge
Exogene fugitif binaire
Exogene non fugitif binaire
Exogene continu
Quitter

ments
d evenements
d evenements
d evenements
tion

PHASE DE PREPARATION DES INFORMATIONS

L'écran 2 montre l'instanciation d'un type générique de générateur d'événement endogène binaire.

L'écran 3 présente dans la fenêtre centrale, les différents générateurs d'événements de modification des variables d'entrée de l'atelier.

L'initialisation des variables du système est décrite dans l'écran 4.

L'écran 5 visualise les différentes variables du système et leur valeur, et présente aussi le cas de modification du marquage du réseau.

L'écran 6 présente respectivement l'initialisation du début de la simulation, la durée de simulation et le mode d'exploitation du simulateur.

RETURN : champ suivant F1 : champ precedent ESC : valider CTRL/END : fin

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

```
Nom      : EFBOB
Etat     :  P11 = 1

Valeur                    : VRAI
1-Fugitive 2-Non fugitive :    1
Probabilite               : 0.9
Loi de distribution du temps de passage a la valeur 1/2/3:3
  1-Exponentielle ( moyenne ) :
  2-Uniforme (petite valeur,grande valeur):
  3-Constante ( valeur ) : 7
```

	Type
EREQ	booleen
ECASSE	booleen
EFECH	booleen
EFBOB	booleen
EFDEPSE	booleen
EFRETSE	booleen
EFDEPN	booleen

RETURN : champ suivant F1 : champ precedent ESC : valider CTRL/END : fin

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

```
Nom      : ECASSE
Etat     :  P11 = 1

Valeur                    : VRAI
1-Fugitive 2-Non fugitive :    2
Probabilite               : 0.1
Loi de distribution du temps de passage a la valeur 1/2/3:2
  1-Exponentielle ( moyenne ) :
  2-Uniforme (petite valeur,grande valeur):5     7
  3-Constante ( valeur ) :
```

	Type
EREQ	booleen
ECASSE	booleen
EFECH	booleen
EFBOB	booleen
EFDEPSE	booleen
EFRETSE	booleen
EFDEPN	booleen

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

```
Instancier generateurs d evenements
Detruire schema de generation d evenements
Lister  schemas de generation d evenements
```

```
si P10 = 1
EFDEPSE := vrai  avec 1.0000 au bout d un temps constant (5)

si P15 = 1
EFRETSE := vrai  avec 1.0000 au bout d un temps constant (4)

si P2 = 1
EFDEPN := vrai  avec 1.0000 au bout d un temps constant (3)

si P6 =1
EFRETN := vrai  avec 1.0000 au bout d un temps constant (2)
```

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

```
Instancier generateurs d evenements
Detruire schema de generation d evenements
Lister  schemas de generation d evenements
```

```
si P9 =1
EREQ := vrai  avec 1.0000 au bout d un temps constant (6)

si P11 =1
EFBOB := vrai  avec 0.9000 au bout d un temps constant (7)

ECASSE := vrai  avec 0.1000 au bout d un temps uniforme( 5, 7)

si P4 =1
EFECH := vrai  avec 1.0000 au bout d un temps constant (2)
```

RETURN : selection CTRL/END : fin

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

Modification des places
Modification des variables
Visualisation des variables
Quitter

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

Modification des places
Modification des variables
Visualisation des variables
Quitter

Nom : EFDEPSE

Valeur : FAUX

Variable	Valeur
EREQ	faux
ECASSE	faux
EFECH	faux
EFBOB	faux
EFDEPSE	
EFRETSE	
EFDEPN	
EFRETN	
ADEPSE	

RETURN : selection CTRL/END : fin

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

Modification des places
Modification des variables
Visualisation des variables
Quitter

Variable	Valeur
EREQ	faux
ECASSE	faux
EFECH	faux
EFBOB	faux
EFDEPSE	faux
EFRETSE	faux
EFDEPN	faux
EFRETN	faux
ADEPSE	faux

RETURN : selection CTRL/END : fin

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

Modification des places
Modification des variables
Visualisation des variables
Quitter

Nom :

Valeur :

MARQUAGE COURANT	
Place	Nbre de jetons
P1	1
P2	0
P3	0
P4	0
P5	0
P6	0
P7	0

RETURN : selection

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

HORLOGE
0

Initialisation de l"horloge
Initialisation duree de la simulation
Mode de la simulation
Verification des proprietes
Quitter

RETURN : selection

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

HORLOGE
0

Initiali
Initiali
Mode de
Verifica
Quitter

Temps :1000
ou
Nombre de cycles :

n

RETURN : selection

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

HORLOGE
0

Initiali
Initiali
Mode de
Verifica
Quitter

Pas a pas
Semi_automatique
Automatique

imulation

RESULTATS DE LA SIMULATION

Pour suivre le déroulement de la simulation, l'utilisateur peut sélectionner la fonction "Demande d'informations" dans le menu principal qui est chargée d'afficher sur le poste de travail le type d'informations choisi par l'utilisateur. L'utilisateur doit auparavant préciser le type d'informations qu'il désire observer (voir écran 7):

- vue externe du système (les traces des variables d'entrée/sortie du système),
- le marquage courant du réseau,
- les statistiques sur les places,
- les statistiques sur les transitions,
- les propriétés du réseau,
- les communications inter-modulaires,
- les statistiques sur les files du réseau décrivant le flux d'activité.

L'écran 8 donne une vue externe du système à l'instant 0.
les écrans 9, 10 présentent respectivement l'état externe du système et le marquage du réseau à l'instant 9.

Les écrans 11, 12 et 13 montrent les propriétés du réseau et statistiques sur les places et les transitions à l'instant 33.

RETURN : selection

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

HORLOGE
0

Initiali
Initiali
Mode de
Verifica
Quitter

Borne
Blocage partiel
Persistance
Invar.places
Seq.transitions
Quitter

ulation

RETURN : selection ESC : fenetre suivante CTRL/END : fin

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

HORLOGE
0

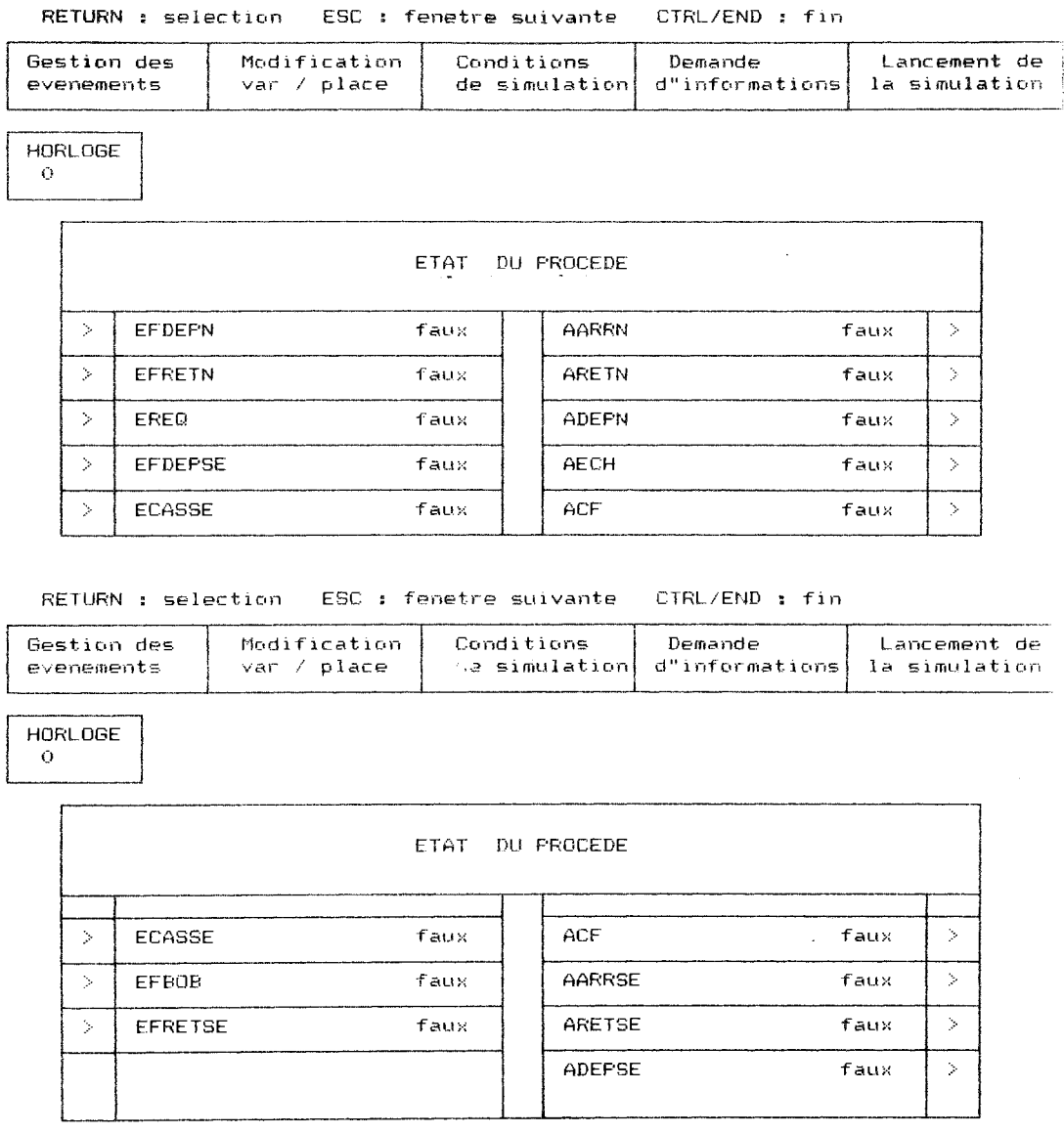
ATELIERBOBINAGE

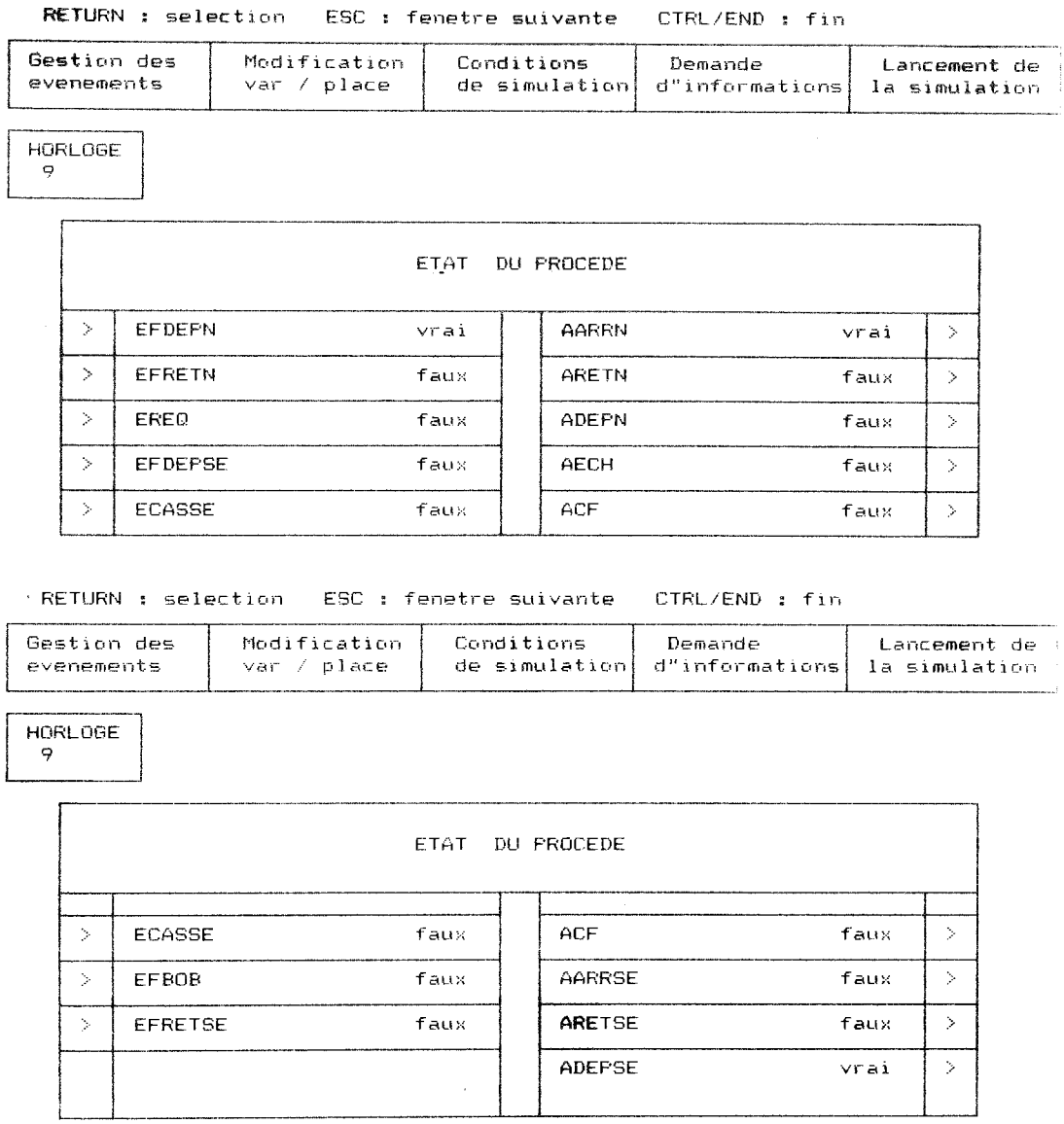
RETURN : selection ESC : fenetre suivante CTRL/END : fin

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

HORLOGE
0

Vue externe du systeme
Marquage courant du reseau
Statistiques sur les places
Statistiques sur les transitions
Proprietes du reseau
Etat des communications
Statistiques sur les files d attente


RETURN : selection ESC : fenetre suivante CTRL/END : fin
Gestion des evenements
Modification var / place
Conditions de simulation
Demande d"informations
Lancement de la simulation
HORLOGE
0
ETAT DU PROCEDE
EFDEPN faux
EFRETN faux
EREQ faux
EFDEPSE faux
ECASSE faux
AARRN faux
ARETN faux
ADEPN faux
AECH faux
ACF faux
RETURN : selection ESC : fenetre suivante CTRL/END : fin
Gestion des evenements
Modification var / place
Conditions de simulation
Demande d"informations
Lancement de la simulation
HORLOGE
0
ETAT DU PROCEDE
ECASSE faux
EFBOB faux
EFRETSE faux
ACF faux
AARRSE faux
ARETSE faux
ADEPSE faux


RETURN : selection ESC : fenetre suivante CTRL/END : fin
Gestion des evenements
Modification var / place
Conditions de simulation
Demande d"informations
Lancement de la simulation
HORLOGE 9
ETAT DU PROCEDE
EFDEPN vrai
EFRETN faux
EREQ faux
EFDEPSE faux
ECASSE faux
AARRN vrai
ARETN faux
ADEPN faux
AECH faux
ACF faux
RETURN : selection ESC : fenetre suivante CTRL/END : fin
Gestion des evenements
Modification var / place
Conditions de simulation
Demande d"informations
Lancement de la simulation
HORLOGE 9
ETAT DU PROCEDE
ECASSE faux
EFBOB faux
EFRETSE faux
ACF faux
AARRSE faux
ARETSE faux
ADEPSE vrai

RETURN : selection ESC : fenetre suivante CTRL/END : fin

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

HORLOGE
9

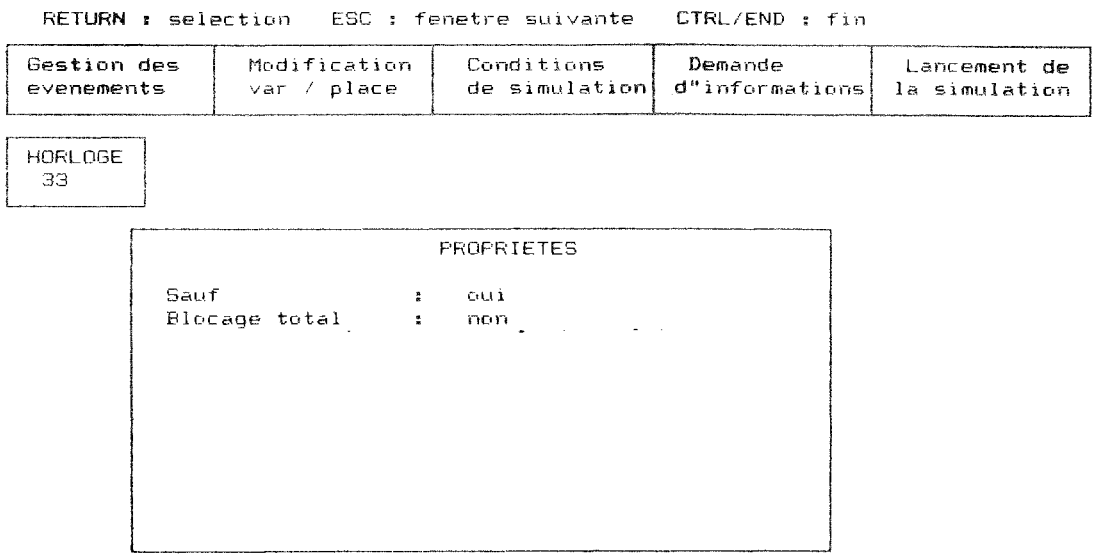
MARQUAGE COURANT	
Place	Nbre de jetons
P1	0
P2	0
P3	1
P4	0
P5	0
P6	0
P7	0

RETURN : selection ESC : fenetre suivante CTRL/END : fin

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

HORLOGE
9

MARQUAGE COURANT	
Place	Nbre de jetons
P7	0
P8	0
P9	0
P10	1
P11	0
P12	0
P13	0


RETURN : selection ESC : fenetre suivante CTRL/END : fin
Gestion des evenements
Modification var / place
Conditions de simulation
Demande d"informations
Lancement de la simulation
HORLOGE
33
PROPRIETES
Sauf : oui
Blocage total : non

RETURN : selection ESC : fenetre suivante CTRL/END : fin

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

HORLOGE
33

STATISTIQUES SUR LES PLACES			
Place	Nbre de fois marquee	Nbre max de jetons	Nbre min de jetons
P1	1	1	1
P2	2	1	0
P3	2	1	0
P4	1	1	0
P5	1	1	0
P6	1	1	0
P7	2	1	0
P8	0	0	0

RETURN : selection ESC : fenetre suivante CTRL/END : fin

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

HORLOGE
33

STATISTIQUES SUR LES PLACES			
Place	Nbre de fois marquee	Nbre max de jetons	Nbre min de jetons
P8	0	0	0
P9	1	1	1
P10	2	1	0
P11	2	1	0
P12	1	1	0
P13	0	0	0
P14	1	1	0
P15	1	1	0

RETURN : selection ESC : fenetre suivante CTRL/END : fin

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

HORLOGE
33

STATISTIQUES SUR LES TRANSITIONS	
Transition	Nbre de tirs
T1	2
T2	2
T3	0
T6	1
T7	1
T8	2
T9	2
T10	0
T11	1

RETURN : selection ESC : fenetre suivante CTRL/END : fin

Gestion des evenements	Modification var / place	Conditions de simulation	Demande d"informations	Lancement de la simulation

HORLOGE
33

STATISTIQUES SUR LES TRANSITIONS	
Transition	Nbre de tirs
T9	2
T10	0
T11	1
T12	0
T13	0
T14	1
T15	1
T4	1
T5	1

NOM DE L'ETUDIANT : Mademoiselle BOUDEBOUS Dalila

NATURE DE LA THESE : Doctorat de l'Université de NANCY I en Informatique

VU, APPROUVE ET PERMIS D'IMPRIMER

NANCY, le - 5 DEC. 1989 n° 2413

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

UNIVERSITE NANCY I
LE PRESIDENT,
M. BOULANGE

Résumé :

Ce mémoire présente la conception et la réalisation d'un outil d'aide au prototypage rapide des systèmes de production automatisés. Cet outil s'appuie sur une description du comportement du système de commande exprimée à l'aide des réseaux de Petri interprétés, et sur une approche de modélisation de l'environnement commandé et du flux d'activité, pour réaliser une simulation hors site de la spécification d'un système de commande.
Dans un processus de conception, cet outil peut être utilisé pour dialoguer avec l'utilisateur afin de clarifier ses besoins, pour la validation et la mise au point de la spécification du comportement d'un système de commande et pour évaluer les performances liées au choix de stratégies de pilotage et au dimensionnement d'un atelier.

Mots-clés : Systèmes de commande, Modélisation de l'environnement, Modélisation du flux d'activité, Réseaux de Petri interprétés, Prototypage rapide, Simulation.