Spécification et validation d'automatismes logiques interconnectés

Joseph Albukerque

N° d'ordre : 2747

Année 1982

THÈSE

présentée

DEVANT L'UNIVERSITE PAUL SABATIER DE TOULOUSE

en vue de l'obtention

du TITRE de DOCTEUR de 3e CYCLE

Spécialité : Electronique, Electrotechnique, Automatique
Option : Automatique

par

Joseph ALBUKERQUE

Maître ès-Sciences

SPECIFICATION ET VALIDATION D'AUTOMATISMES LOGIQUES INTERCONNECTES

Soutenue le 16 décembre 1982, devant la Commission d'examen :

MM.	A. COSTES	*Président*
	P. ALANCHE	*Examinateurs*
	G. BLONDET-GONTE	
	M. COURVOISIER	
	J. FANTUZZO	
	Y. QUENEC'HDU	
	R. VALETTE	

ALBUKERQUE (Joseph). - Spécification et validation d'automatismes logiques interconnectés. - 150 p.

Th.Doctorat de 3ème Cycle : E.E.A. : Automatique : Toulouse III, 1982 ; 2746

RESUME :

Ce mémoire est composé de trois chapitres. Le premier pose le problème de la spécification des systèmes de commande complexes formés d'un ensemble d'automatismes communicants. Après avoir introduit les réseaux de Petri en tant qu'outil formel pour la spécification, il est montré que cet outil n'est pas contradictoire avec une approche structurée. Quelques règles de structuration sont proposées. Cette démarche est illustrée par un exemple concret. Le second chapitre montre comment une spécification structurée peut être validée. Le troisième chapitre propose un langage de spécification adapté à la description structurée d'automatismes interconnectés. Ce langage est fondé sur l'utilisation des réseaux de Petri. Un logiciel d'analyse syntaxique et sémantique a été développé sur microcalculateur en langage PASCAL. Ce logiciel traduit la spécification en tables et est conçu de façon à permettre le téléchargement d'automates programmables spécialisés.

MOTS CLES :

- Spécification
- Validation
- Automatismes logiques
- Réseaux de Petri

JURY et date de soutenance : 16 décembre 1982

Président : A. COSTES
Membres : P. ALANCHE
G. BLONDET-GONTE
M. COURVOISIER
J. FANTUZZO
Y. QUENEC'HDU
R. VALETTE (LAAS)

DEPOT à la Bibliothèque Universitaire en 4 exemplaires

" *Il essuiera toute larme de leurs yeux, et la mort ne sera plus, et il n'y aura plus ni deuil, ni cri, ni douleur, car les premières choses ont disparu*".

(La Sainte Bible : "Apocalypse" 21.4)

AVANT-PROPOS

Le travail présenté dans ce mémoire a été réalisé au Laboratoire d'Automatique et d'Analyse des Systèmes du C.N.R.S, au sein de l'équipe "Parallèlisme, Analyse, Spécification et Test en Ligne des Strcutures".

Je remercie Monsieur D. ESTEVE, Directeur du laboratoire pour m'y avoir accueilli et ainsi permis la réalisation de ce travail.

Je tiens à remercier Monsieur A. COSTES, directeur-adjoint du laboratoire et Professeur à l'Institut National Polytechnique de Toulouse pour l'honneur qu'il me fait en acceptant de présider ce Jury de Thèse.

Je suis très reconnaissant à :

- Monsieur J. ALANCHE, Ingénieur à la DAST-RNUR.
- Monsieur G. SONDET-GONTE, Chef du Département des Systèmes Electriques au Centre National d'Etudes Spatiales (CNES) EVRY.
- Monsieur COURVOISIER, Professeur à l'Université Paul Sabatier de Toulouse et responsable de l'équipe P.A.S.T.E.L.S.
- Monsieur J. FANTUZZO, Chef du Département Electronique à l'Aérospatiale Toulouse.
- Monsieur Y. QUENEC'HDU, Chef du Service Automatismesà l'Ecole Supérieure d'Electricité (E.S.E) RENNES.
- Monsieur R. VALETTE, Chargé de Recherche au C.N.R.S.

Pour l'honneur qu'ils me font en participant à la commission d'examen.

Ce travail a été réalisé sous la direction de Messieurs COURVOISIER et VALETTE à qui je suis particulièrement reconnaissant pour leurs nombreux conseils et la sympathie qu'ils m'ont témoignée.

Que Monsieur GEFFROY, Maître-Assistant à l'Institut National des Sciences Appliquées de Toulouse et tous les camarades de l'équipe P.A.S.T.E.L.S. trouvent ici l'expression de ma profonde gratitude pour leur soutien et leur amitié.

Je tiens à remercier la direction du C.N.E.S Toulouse et en particulier Monsieur G. BLONDET-GONTE pour les bonnes relations de travail que nous avons eues lors du contrat concernant la spécification et la validation des automatismes de l'étage cryogénique H8 du lanceur ARIANE.

Enfin je remercie Mesdames Joëlle PENAVAYRE et Dannie HERAL qui ont assuré la dactylographie, ainsi que Messieurs D. DAURAT, R. ZITTEL et R. LORTAL qui ont permis la réalisation matérielle de ce mémoire.

SOMMAIRE

°O°

INTRODUCTION

L'automatisation poussée des procédés industriels est devenue une obligation pour assurer une bonne compétitivité. Ceci implique une complexité croissante des automatismes logiques de commande et pose le problème de leur spécification et celui de leur mise en oeuvre. Il faut par conséquent fournir aux concepteurs et aux utilisateurs des outils et des méthodes qui, tout en permettant de maîtriser cette augmentation de complexité restent simples à manipuler.

Les méthodes classiques de spécifications utilisées pour décrire des systèmes simples avec un faible degré de complexité tels que l'organigramme, le graphe d'états, les diagrammes de transitions,..., ne permettent pas de maîtriser le problème de la spécification d'un système complexe et en particulier lorsque celui-ci comporte des évolutions simultanées. Cependant, une première solution peut être une décomposition par rapport aux différents matériels commandés. Cette solution apporte effectivement une simplification du graphe spécifiant le système à décrire, mais elle ne permet pas de représenter de façon suffisamment claire les mécanismes de synchronisation entre les divers automates. Ainsi, la spécification d'automatismes complexes et notamment la maîtrise du parallèlisme (évolutions simultanées interférant entre elles) nécessite à la fois la décomposition du système global en sous-systèmes (automates "indépendants") pour éviter l'explosion combinatoire des états et la possibilité d'avoir une description précise des liens existants entre les divers sous-systèmes afin de permettre la détection de mauvais fonctionnements (blocages, ordres contradictoires, etc,...).

Il est donc nécessaire de faire appel à un modèle de description plus puissant. Notre choix s'est porté sur les réseaux de PETRI qui à nos yeux constituent actuellement le meilleur compromis entre les exigences que doit satisfaire l'outil de description idéal. Les raisons de de choix sont exposées dans le chapitre premier. Nous pouvons d'ores et déjà souligner que l'aspect graphique et surtout le caractère analysable des réseaux de Pétri leur confèrent un atout important.

Toutefois, une utilisation "anarchique" des réseaux de Pétri peut donner un graphe complètement illisible, c'est pourquoi une bonne structuration est nécessaire et nous verrons que les réseaux de Pétri s'y

prêtent particulièrement bien.

D'autre part, le caractère analysable des réseaux de Pétri permet, sans passer par de coûteuses simulations, de détecter la plupart des erreurs grossières de conception et de vérifier certaines propriétés spécifiques au système décrit. Le chapitre II expose les propriétés que doit posséder un réseau de Pétri correct et les méthodes d'analyse permettant de les vérifier.

Afin d'illustrer l'application des réseaux de Pétri, nous présentons un travail effectué dans le cadre d'un contrat avec le Centre National d'Etudes Spatiales (CNES). Il s'agit de la spécification et validation des automatismes de sécurité de l'étage cryogénique de la base de lancement ARIANE 2. On peut noter à ce sujet que le seul fait de spécifier avec un modèle non ambigu fait se poser des questions et éventuellement remettre en cause les spécifications initiales. L'analyse permet de pousser plus loin cette discussion et cette remise en cause.

La validation des spécifications est très importante et donne des garanties quant au comportement futur du système spécifié. Mais celle-ci est un vain effort si des erreurs se glissent lors du passage de la spécification à la réalisation du système. D'où l'intérêt d'une mise en oeuvre directe des spécifications. Une mise en oeuvre logicielle basée sur l'utilisation d'un automate programmable est une solution pour des automatismes où la vitesse ne constitue pas une contrainte forte. Pour cela, il faut utiliser un langage capable de décrire les spécifications, c'est-à-dire le réseau de Pétri interprété.

Nous proposons dans cette optique un langage haut niveau (DESY2) qui, après avoir vérifié que la description du système à l'aide de réseaux de Pétri interprété est syntaxiquement et sémantiquement correcte le transmet sous forme de tables à l'automate programmable qui se charge alors de le faire évoluer en temps réel et compte tenu des évènements en provenance du processus commandé.

CHAPITRE I

SPECIFICATION DES AUTOMATISMES COMPLEXES

I.1. INTRODUCTION

Les automatismes complexes, de par leur grand nombre d'états, posent des problèmes de spécification. Outre celui de l'explosion combinatoire des états, on peut noter celui de la représentation de la synchronasation entre les divers automatismes.

Un outil de spécification puissant et formel est alors nécessaire ; notre choix s'est porté sur les réseaux de Pétri.

L'utilisation d'un outil de spécification aussi puissant soit-il ne met pas à l'abri de descriptions lourdes et touffues si un effort de structuration n'est pas entrepris. Nous proposons dans cette optique quelques approches pour la structuration lors de la spécification à l'aide de réseaux de Pétri.

Les automatismes de sécurité de l'étage cryogénique H8 de la base de lancement ARIANE 2 sont ensuite pris comme exemple illustratif d'application.

I.2 QUE SONT LES AUTOMATISMES COMPLEXES.

I.2.1. Introduction

Pour être compétitifs et performants, les procédés industriels sont l'objet d'une automatisation très poussée, ce qui implique une complexité croissante des automatismes logiques de commande.

D'autre part, les tâches à accomplir étant très nombreuses et pouvant évoluer simultanément tout en interférant entre-elles (parallèlisme), cela se traduit par des automatismes ayant un grand nombre d'états.

En plus du parallèlisme interne à chaque automatisme, on rencontre le problème de l'interconnexion de plusieurs automatismes. On peut noter à ce sujet que la composition de k automates à n états donne un automate ayant n^k états.

Un atelier flexible (atelier modulaire et adaptatif) est un exemple type de tels systèmes commandés complexes. En effet une machine donnée joue souvent plusieurs rôles et le fait que plusieurs types de pièces soient fabriqués simultanément introduit de sérieux problèmes de coopération et de compétition entre les automatismes logiques de commande.

Les méthodes classiques de spécification (graphe d'états, diagrammes de transition,...) ne permettent pas de maîtriser le problème de la spécification d'un système complexe comportant des évolutions simultanées. Il est donc nécessaire de faire appel à un outil de description plus puissant : les réseaux de Pétri, objet du paragraphe suivant.

I.2.2. Un outil de description : les réseaux de PETRI

L'outil idéal de spécification devrait satisfaire les exigences suivantes :

- Etre d'une utilisation simple et aisée
- Etre facilement transmissible, c'est-à-dire un bon outil de dialogue
- Pouvoir traiter la majorité des cas pouvant se présenter (généralité)
- Offrir la possibilité de validation par l'analyse
- Pouvoir donner lieu à des méthodes de mise en oeuvre directe qui respectent ses règles d'évolutions.

Il n'existe actuellement aucun modèle qui puisse satisfaire pleinement ces exigences, mais parmi les modèles existants nous considérons que les réseaux de Pétri constituent le meilleur compromis. Il a été montré [VAL 80] comment spécifier divers types de problèmes avec les réseaux de Pétri et notamment les automatismes logiques.

Ce modèle a son origine dans les travaux de C.A PETRI concernant la communication entre automates [PET 62] . Il doit sa forme actuelle à HOLT [HOL 70] . Le modèle est non ambigu : ses règles de fonctionnement sont parfaitement précisées. Il est facilement manipulable car ses constituants, les places et les transitions, sont associés respectivement à l'état du système et aux évènements conditionnant son évolution.
Il est cohérent quel que soit le niveau choisi pour la spécification et permet en particulier de définir et utiliser une représentation multi-niveaux. Il est concis car il ne spécifie que les états et les évènements qui sont significatifs pour l'évolution de la commande.

La représentation obtenue est en général claire et lisible, du moins pour des réseaux de taille raisonnable (c'est-à-dire n'excédant pas 20 places - 20 transitions). Ses règles d'évolution sont simples et

permettent de transmettre très vite et aisément le fonctionnement décrit. Cet avantage et sa nature graphique lui permettent de représenter la plupart des problèmes liés à la commande parallèle des processus industriels. Bien que n'étant pas le modèle idéal dans tous les cas, il a été largement utilisé et a fait preuve de son aptitude à représenter des spécifications analysables, ce qui constitue sa principale caractéristique.

Les éléments de base de ce modèle sont rappelés en annexe 1 de ce mémoire. Il sera utilisé pour décrire de façon formelle la commande à réaliser et nous permettra alors de vérifier si la spécification est correcte, c'est-à-dire remplit un certain nombre de propriétés qui auront une signification réelle parfaitement précise, afin d'assurer le bon fonctionnement du système.

I.2.3. *Utilisation des réseaux de PETRI pour la spécification d'automatismes logiques.*

Représentation du système de commande et réseau de Pétri étiqueté

On peut représenter l'ensemble système de commande, système commandé (ou procédé) par la figure I.1.

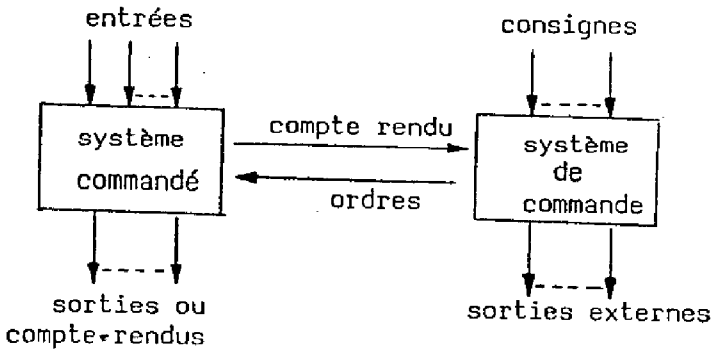


Figure I.1

Le système de commande doit être capable de lire les comptes-rendus ou mesures fournies par le système commandé (capteurs par exemple) ou les consignes provenant d'un opérateur. A partir de ces données il engendre des actions ou des ordres vers le système commandé et les sorties externes (signaux de visualisation, d'alarme, sortie sur imprimante de certains résultats).

Ainsi, pour spécifier des automatismes logiques à l'aide des réseaux de Pétri, il faut être capable de prendre en compte les évènements provenant du monde extérieur et d'envoyer les actions vers le système commandé. Pour cela on associe des étiquettes aux transitions et aux places. Les étiquettes associées aux transitions sont constituées par les évènements qui sont des fonctions logiques définies sur les consignes provenant de l'opérateur et les comptes rendus d'états provenant du procédé. Elles permettent également l'introduction de variables auxiliaires pour la représentation de certains problèmes de synchronisation que l'on ne peut pas représenter facilement graphiquement. Le tir d'une transition T est alors soumis à deux conditions :

1) - La condition de marquage qui fait que T est ou non sensibilisée,

2) La valeur booléene de l'étiquette (associée à T) qui, si elle est VRAIE, permet alors (si la transition T est sensibilisée) le tir de la transition T.

L'émission d'actions du système de commande vers le procédé peut être modèlisé de deux façons :

- s'il s'agit d'actions impulsionnelles du type, par exemple, signaux de déclenchement, celles-ci doivent être associées au tir des transitions, aucune notion de durée n'y étant alors rattachée.
- s'il s'agit d'actions se traduisant par des niveaux logiques, elles doivent être associées aux places du réseau de Pétri, une action étant maintenue tant qu'une des places où elle est spécifiée est marquée.

L'ensemble des évènements associés aux transitions et des opérations et actions associées aux places et aux transitions constituent l'interprétation du réseau de Pétri.

En résumé : L'outil "réseau de Pétri" permet de décrire des systèmes logiséquentiels et parallèles en spécifiant divers sous-systèmes évoluant simultanément et les liens existants entre-eux.

Toutefois, les réseaux de Pétri ont des limites [VLM 81] et il n'est pas toujours possible de décrire simplement un système très complexe. Mais nous verrons dans le paragraphe suivant comment cet outil utilisé avec rigueur permet malgré tout d'aborder de nombreux problèmes. Si tous les problèmes complexes ne s'expriment pas simplement, du moins un certain nombre de mécanismes qu'il était difficile de décrire à l'aide des outils classiques deviennent plus aisés et plus clairs.

I.3. QUELQUES APPROCHES POUR LA STRUCTURATION

I.3.1. Introduction

Le réseau de Pétri permet de spécifier à la fois les automatismes et la communication entre ceux-ci en évitant au maximum l'explosion combinatoire des états. Dans le cas d'un système formé de sous-systèmes on aboutit ainsi à un réseau de Pétri exprimant les liens existants entre les divers sous-systèmes. Toutefois, si les sous-réseaux sont interconnectés de manière anarchique, l'on aboutit inévitablement à un gros réseau de Pétri dont la taille peut être inacceptable.

C'est pourquoi une démarche rigoureuse s'impose lors de la spécification. Il faut structurer , et les réseaux de Pétri peuvent être utilisés comme outil de structuration pour répondre aux questions :

- où couper entre sous-systèmes
- comment recomposer les sous-systèmes lorsque celà est possible et nécessaire.

La première décomposition est toujours un peu arbitraire. L'intérêt des réseaux de Pétri, en représentant de la même manière (à l'aide de places et transitions) les sous-systèmes et leur communication, est qu'il est possible de déplacer assez facilement les frontières.

Une structuration correcte commence par une bonne délimitation entre la partie commande et la partie données du système à spécifier. On peut ensuite envisager une approche descendante c'est-à-dire une description par affinements successifs. La rigueur d'une telle approche

assure une spécification bien structurée et garantit des propriétés de bon fonctionnement quant au réseau ainsi obtenu. Mais cette démarche dans certains cas n'est pas possible et il faut alors raisonner "de bas en haut", c'est-à-dire spécifier le système au moyen de descriptions partielles et obtenir le réseau global par composition des différents sous-réseaux.

Nous exposons les grandes lignes de ces approches :

I.3.2. Délimitation entre commandes et données

L'utilisation des réseaux de Pétri comme outil de spécification et de structuration impose une réflexion approfondie sur la délimitation entre la commande (séquencement des actions) et les données. En effet un réseau de Pétri non étiqueté est totalement inadapté pour décrire un traitement d'information sophistiqué. Il ne doit décrire que la partie correspondant à l'enchaînement des tâches. Cette délimitation peut être floue mais c'est justement l'intérêt des réseaux de Pétri d'obliger le concepteur à se poser cette question et de lui permettre d'analyser les conséquences de son choix au niveau de la spécification et à celui de la validation.

Exemple : Considérons le système suivant formé de deux chaînes de fabrication A et B. Les tapis roulants A1 et B1 amènent des pièces qui s'accumulent sur les tapis de stockage A2 et B2. Une seule machine étant disponible, elle devra être attribuée à la chaîne A ou à la chaîne B suivant les demandes.

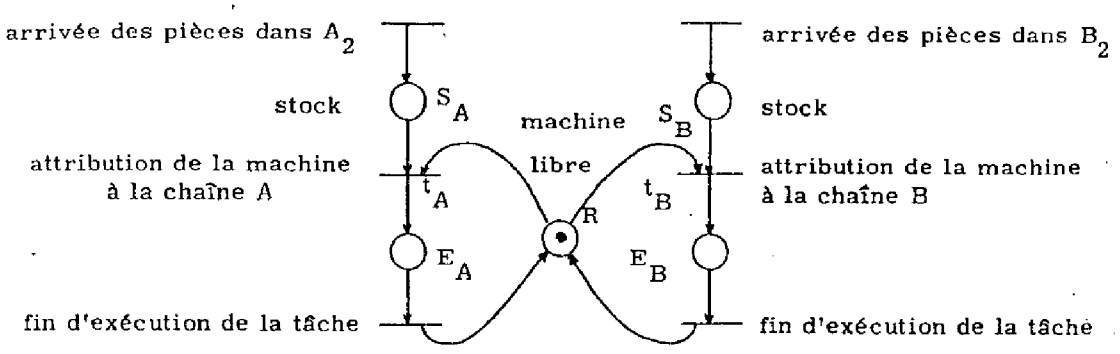


Figure I.2

Le réseau de la figure I.2 représente ce problème d'allocation de ressource et met bien en évidence le conflit qui existe entre les transitions t_A et t_B, ces deux transitions peuvent être simultanément sensibilisées alors que seule l'une d'entre elles peut être tirée (il n'y a qu'une seule machine et donc un seul jeton dans la place R). Le choix entre t_A et t_B implique une priorité entre les chaînes A et B. Supposons que l'on veuille exprimer que B est prioritaire par rapport à A. Il faut rajouter une condition exprimant que t_A ne peut être tirée que lorsque la place S_B est vide ("test-à-zéro"). Cela ne peut pas s'exprimer directement par un arc liant S_B à t_A. Diverses solutions sont possibles :

- la première consiste à introduire une place S'_B complémentaire de S_B, il suffit alors de tester S'_B, ce qui donne le réseau de la figure I.3.

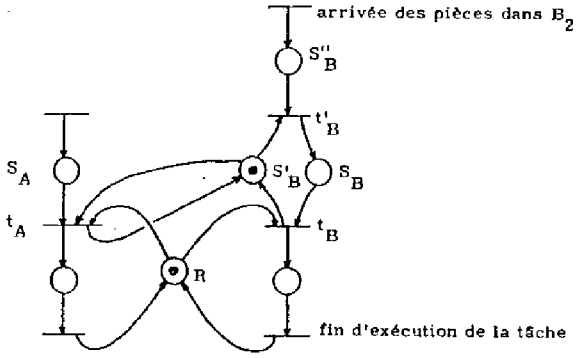


Figure I.3

L'on s'aperçoit aisément que cette solution rend le graphe rapidement touffu et illisible par l'introduction de boucles élémentaires.

- Une autre solution consiste à introduire des arcs inhibiteurs permettant d'exprimer directement la condition "tirer t_A si S_B est vide". Malheureusement l'introduction d'arcs inhibiteurs remet en cause l'ensemble des résultats théoriques obtenus à propos des réseaux de Pétri et le réseau ainsi obtenu n'est pas analysable. Nous ne les utiliserons donc pas.

- Enfin, on peut jouer sur la délimitation commande/données pour contourner le problème de "test-à-zéro". Tout ce qui introduit trop de boucles élémentaires est mis dans la partie données, ce qui conduit à une partie commande simple et facilement analysable. En utilisant des variables auxiliaires pouvant dépendre du marquage de certaines places, on obtient des spécifications plus simples. Ainsi est obtenu le réseau

de Pétri de la figure I.4 où la variable B est à tout instant le marquage de la place SB.

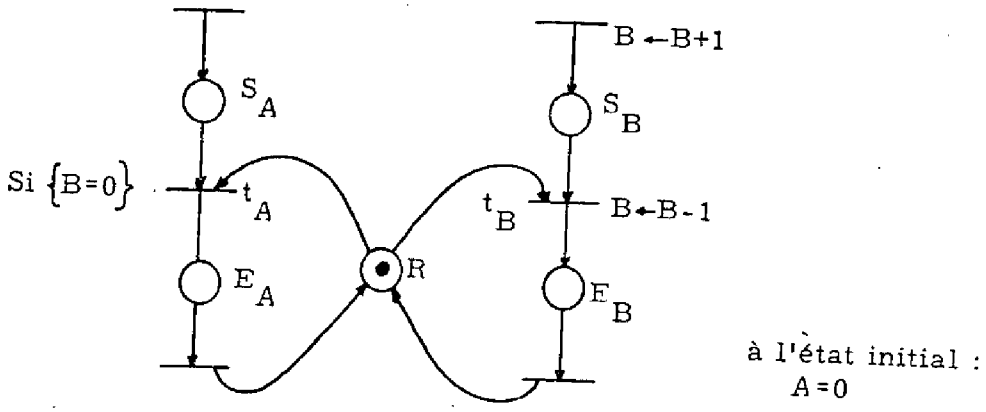


Figure I.4

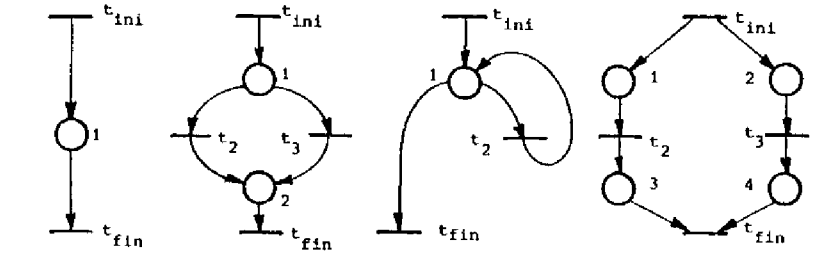
Les mécanismes décrits par les figures I.3 et I.4 sont identiques, les spécifications ne sont différentes que parce que la délimitation entre les parties commande et données a été modifiée.

Le richesse de l'outil"réseau de Pétri" résulte justement de cette liberté laissée au concepteur. Il est ainsi possible de choisir une représentation où le maximum est mis dans la partie commande, ou bien de rejeter dans la partie données tout ce qui n'est pas lourd à décrire par un réseau de Pétri.

I.3.3 Approche par affinements successifs

Cette solution consiste à décrire grossièrement le comportement du système par un réseau de Pétri (qui est relativement simple) et à le valider. Cette première spécification à un haut niveau d'abstraction utilise nécessairement des opérateurs complexes qui doivent à leur tour être spécifiés et validés. Plusieurs niveaux d'abstraction peuvent ainsi s'enchaîner. Si la démarche a été "correcte", la validation du système global n'est pas nécessaire, celle-ci résultant directement des validations effectuées, d'où l'intérêt de cette méthode.

On entend par démarche"correcte" le fait de respecter certaines règles et notamment de remplacer les transitions représentant les opérateurs uniquement par des "blocs bien formés" [VAL 76] . La figure I.5 donne les "blocs bien formés " de base :



a) Bloc "séquence" b) Bloc "si-alors-sinon" c) Bloc "faire-tant-que" d) Bloc "faire-en-parallèle"

Figure I.5

L'avantage d'une telle approche (issue des règles de programmation structurée) est que sa rigueur garantit l'obtention d'un réseau de Pétri ayant les propriétés désirées.

Toutefois, utilisée seule cette approche peut s'avérer insuffisante. En particulier lorsque la synchronisation dans un système n'est pas purement fonctionnelle mais qu'elle résulte d'un "effet de bord" provoqué par l'utilisation de ressources communes logiques ou physiques, il n'est pas possible de raisonner par affinements successifs. C'est aussi le cas des systèmes complètement décentralisés, c'est-à-dire sans aucun élément effectuant la coordination des divers composants, ces derniers communiquant entre eux de manière très dynamique. On retrouve ce phénomène dans l'exemple classique des pompiers qui font la chaîne pour se passer le seau d'eau [PET 79] .

D'autre part, une démarche par affinements peut donner une synchronisation globale trop forte, entraînant des prises de ressources (machines) prématurées et des restitutions tardives. Exemple :

On veut spécifier une opération utilisant deux machines A et B, et à un niveau de détail plus bas on précise que l'on veut en fait utiliser la machine A puis la machine B. Une telle approche peut s'illustrer pour la figure I.6.a , puis par la figure I.6.b, où l'opérateur t_1 de la figure I.6.a, a été remplacé par le bloc"séquence" t'_1, P1, $t"_1$.

L'on s'aperçoit que le tir de t'_1 (utilisation de la machine A) prend les deux machines A et B mais en fait une seule est utilisée (A) et celle-ci ne sera disponible qu'après l'utilisation de la machine B (tir de t''_1). Une telle démarche entraînerait par conséquent des pertes de temps considérables.

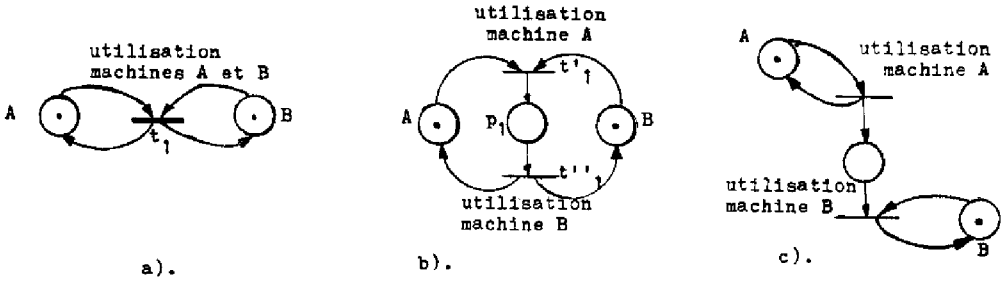


Figure I.6

La figure I.6.c, représente une utilisation optimale des deux machines A et B, mais celle-ci résulte d'une spécification partant directement d'un niveau de description assez bas, qui n'a plus rien de commun avec une approche descendante et dont l'expression serait "Utiliser la machine A et la rendre disponible aussitôt après, puis utiliser la machine B et la rendre disponible aussitôt après son utilisation ...".

A la lumière de cet exemple, nous voyons que dans certains cas il est nécessaire, pour avoir une synchronisation efficace de partir d'un niveau de détail très bas, si bas parfois qu'il n'y a plus d'affinements à faire. Il s'agit alors de spécifier au moyen de descriptions partielles, la synchronisation propre à chaque sous-sytème et le réseau global est ensuite obtenu par composition des différents sous-réseaux.

1.3.4. Approche par composition de réseaux

Cette démarche consiste à donner des spécifications partielles du système et à essayer d'obtenir une synchronisation globale par la composition des divers sous-réseaux décrivant les spécifications partielles.

1/ - Communication par envoi de message simple

La fin de l'exécution d'une tâche TA1 peut être une condition nécessaire à l'exécution d'une tâche TA2 (dans un atelier flexible de

masticage par exemple, la fin de chargement d'un tapis roulant, lance l'exécution d'une tâche "transport"). La figure I.7 montre comment cet échange de message est spécifié dans chacune des tâches. La place P_M représente le registre tampon où sont stockés les messages émis par la tâche TA1. L'analyse de la synchronisation se fera alors à l'aide de la figure I.7.b. Dans un tel cas on dit que l'obtention du réseau représentant la synchronisation globale se fait par "fusion de places".

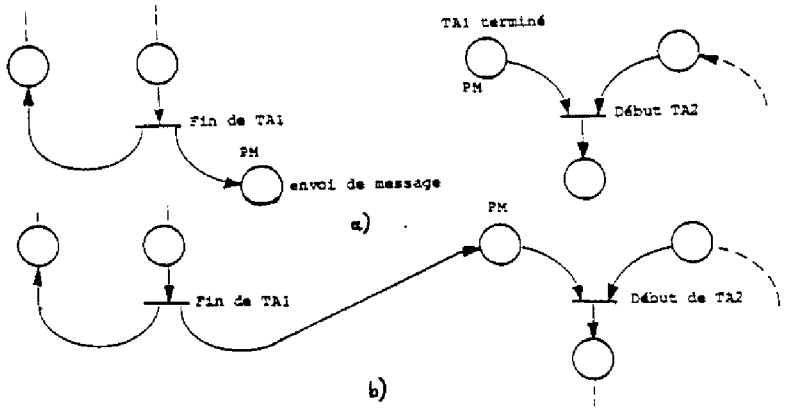


Figure I.7 - Communication par envoi de message

Remarque : Les places "registre tampon de message" (comme PM) peuvent apparaître dans plus de deux tâches mais pour que la mise en oeuvre sur un système réparti soit correcte, il faut qu'il n'y ait qu'une seule tâche consommatrice (comme TA2) toutes les autres étant productrices (comme TA1). Une communication telle que celle de la figure I.8 est en effet interdite, car elle va créer un conflit non résolu entre les tâches TA2 et TA3. Pour éviter ce problème, il faut, soit changer la communication entre les tâches (par exemple, en rendant TA3 esclave de TA2) soit regrouper les tâches TA2 et TA3 dans le même automatisme, ce dernier se chargeant de résoudre le conflit.

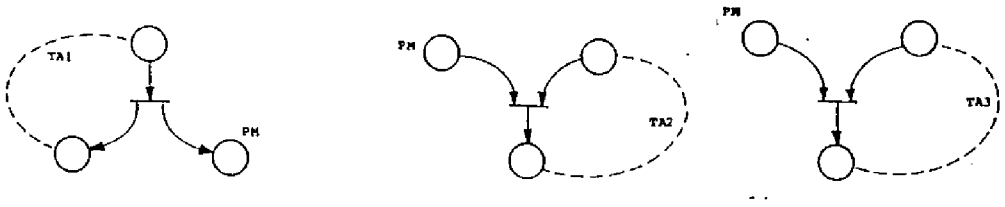


Figure I.8 - Conflit non résolu entre deux tâches

La figure I.9 montre comment on peut résoudre ce problème en créant une hiérarchie entre les tâches. La tâche esclave doit faire une demande (place P'3) au maître. Si ce dernier est d'accord (tir de T'3) il lui donnera l'autorisation (place p"3) de tirer T3.

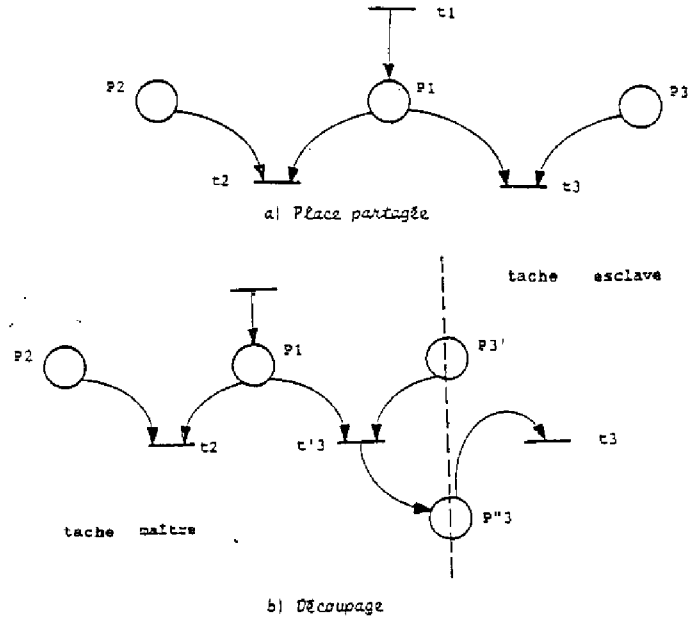


Figure I.9 - Répartition en cas de conflit
Relation Maître-Esclave.

Cet exemple met en évidence la possibilité de modification du découpage d'un système en sous-système lorsque la spécification est faite par un réseau de Pétri.

2/ - Communication par "appel-réponse"

A certains moments on a besoin d'une synchronisation plus fine entre les tâches. Par exemple, on veut spécifier que l'évènement Ev de la tâche TA1 doit avoir lieu en synchronisation parfaite avec l'évènement Ev de la tâche TA2. Cette synchronisation peut se faire par un échange de message du type "appel-réponse". Ce mécanisme est aussi appelé "rendez-vous". Le réseau de Pétri correspondant à cet exemple est donné par la figure I.10.a. On parle dans ce cas de communication par fusion de transitions.

La synchronisation globale sera obtenue directement à partir des descriptions partielles en faisant fusionner les transitions de même nom (figure 1.10.b).

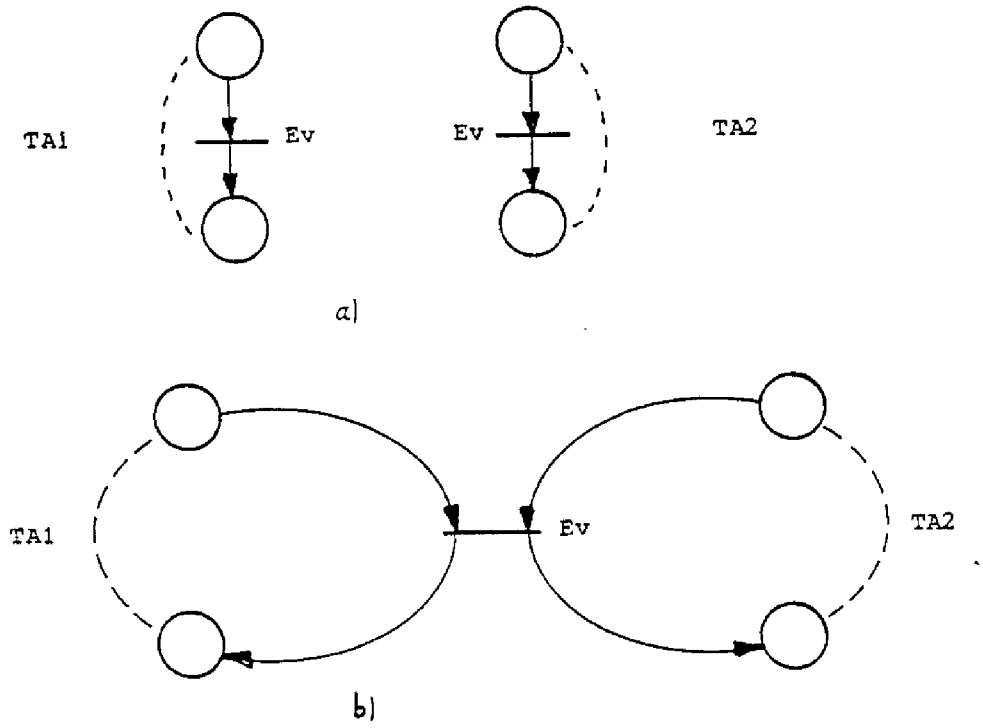


Figure I.10 - Communication par fusion de transition

Remarque : Alors que dans le cas d'une approche par affinements successifs, les propriétés du réseau de Pétri sont conservées par construction, il est nécessaire dans le cas d'une approche par composition de réseaux de recourir à la validation du système global.

1.3.5. Communication par relation "maître-esclave".

Certaines communications sont difficiles à représenter par des places et transitions et en particulier le mécanisme suivant où un automatisme de sécurité (maître) force hors service et verrouille dans cette position un (ou plusieurs) automatisme(s) de fonctionnement "normal" (esclaves). En effet, une représentation complète et détaillée de ce mécanisme impliquerait l'introduction d'autant de transitions (et par conséquent d'arcs nécessaires à la liaison place-transition-place) supplémentaires que l'automate esclave possède d'états. La figure I.11 illustre

un cas très simple de communication entre un réseau "maître" et un réseau "esclave" où le maître est chargé de verrouiller le réseau "esclave" en cas d'alarme, d'exécuter une séquence d'opérations (ORDRE X) et déverrouiller ensuite le réseau esclave I.11.a . Pour vérouiller le réseau "esclave" il faut le mettre hors service quelque soit l'état dans lequel il se trouve ; il faut par conséquent associer à chaque place du réseau esclave une transition supplémentaire dont le tir lui enlèvera le jeton en cas d'alarme; à cela s'ajoutent les arcs nécessaires à cette correspondance. L'on aboutit dans cet exemple très simple au réseau de Pétri de la Figure I.11.b, et l'on s'aperçoit qu'avec une telle représentation la structure du réseau de Pétri peut très vite disparaître derrière un foisonnement d'arcs.

Remarque : Dans la représentation de la figure I.11.b, les ordres "verrouiller esclave" et "déverouiller esclave" n'apparaissent plus en étiquette car ils sont directement spécifiés par le réseau de Pétri. (La place P3 disparaît car l'ordre qui lui est associé est directement représenté par un arc liant t_3 à P_4).

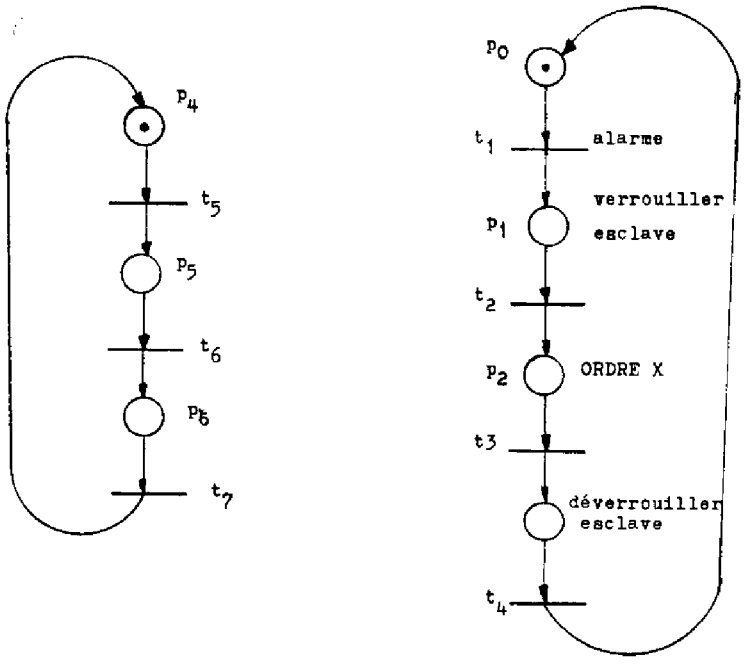


Figure I.11.a

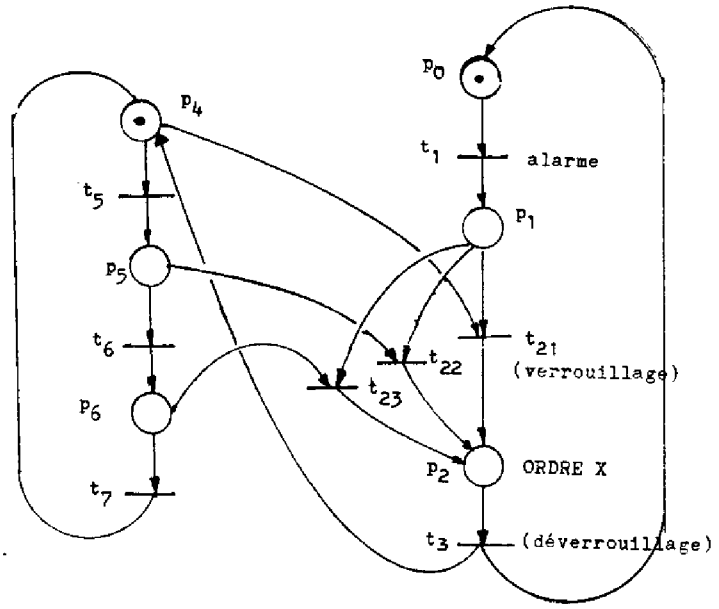


Figure I.11.b

- Dans la pratique une telle représentation est bien évidemment totalement inacceptable. C'est le problème bien connu que posent les arrêts d'urgence dans les automatismes logiques.

- Une approche possible consiste à introduire une variable auxiliaire de synchronisation "arrêt d'urgence" (A.U) intervenant dans toutes les étiquettes associées aux transitions de réseau esclave de telle manière que toutes les transitions sauf celle de mise en service soient tirées. Cette approche est satisfaisante lorsque la commande d'arrêt d'urgence est une entrée du système. Par contre lorsque l'on a toute une hiérarchie d'automatismes logiques capables d'émettre de telles commandes, il est nécessaire d'avoir une représentation plus claire et surtout plus facilement analysable de leur communication.

- Une autre approche consiste à supposer que l'automatisme "maître" est effectivement capable "en une seule opération" d'enlever tous les jetons de l'automate esclave quel que soit leur emplacement. Or la condition pour que cette opération soit possible est que l'automate esclave n'ait pas déjà été verrouillé en position hors service. Donc du point de vue de l'automate maître, l'automate esclave se comporte comme une place

initialement marquée qui est vidée de son jeton lorsqu'il est verrouillé et hors service et dans laquelle on remet un jeton lorsqu'il est déverouillé. En prenant pour exemple le réseau de la figure I.11 on obtient celui de la figure I.12.

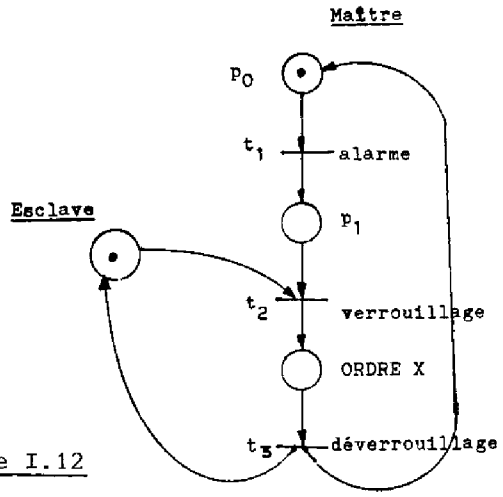


Figure I.12

Remarque : Dans le cas d'un ensemble d'automatismes interconnectés, une telle approche oblige à spécifier clairement à chaque instant quels automatismes ont le droit de verrouiller quels autres.

I.3.6. *Conclusion*

Face aux problèmes de spécification que posent les automatismes complexes, nous avons proposé le réseau de Pétri comme outil de spécification et examiné les méthodes de structuration que permet celui-ci.

La structuration commence par une décomposition du système en sous-systèmes, ce qui permet de décrire clairement le rôle et la structure de chacun de ceux-ci et parallèlement de définir globalement les liens entre eux.

Une bonne délimitation entre la partie commande et la partie données du système à spécifier, ainsi que des sous-systèmes s'impose. La structure de commande est alors concise et précise et on peut ensuite la spécifier en détail :

- lorsque la synchronisation globale et interne à chaque sous-système est importante, nous avons vu qu'il est avantageux d'adopter une

une démarche consistant en descriptions partielles, puis en composant les réseaux ainsi obtenus.

Par contre, pour un gros système ayant peu d'interconnexions entre les divers sous-systèmes, une approche par affinements successifs est alors très efficace et garantit des propriétés de bon fonctionnement.

Nous allons illustrer ces différentes démarches au paragraphe suivant à l'aide d'un exemple concret.

I.4. *UN EXEMPLE : LES AUTOMATISMES DE SECURITE DE LA BASE DE LANCEMENT ARIANE 2.*

1.4.1. *Présentation Générale*

Dans le cadre d'un contrat avec le Centre National d'Etudes Spatiales (CNES) nous avons été amenés à étudier les automatismes de sécurité de la base de lancement ARIANE 2 [ARI 81] . Il s'agit plus précisément des automatismes de l'étage cryogénique H8 du lanceur ARIANE. Cet étage est composé de deux réservoirs cryogéniques hydrogène (H2) et oxygène (O2) séparés par un fond intermédiaire commun. Celui-ci devant être le plus léger possible, est par conséquent fragile, et une trop grande différence de pression entre les deux compartiments O2 et H2 peut provoquer sa fissure et par suite l'explosion de l'étage. C'est ainsi qu'une régulation permanente de la différence de pression entre O2 et H2 doit être assurée. Cette différence de pression est détectée au moyen de capteurs propres à chacun des deux réservoirs. La régulation de pression est assurée par des modules de pressurisation au sol, essentiellement composés de vannes commandées en ouverture, fermeture et demi-ouverture. La liaison entre l'infrastructure au sol et l'étage cryogénique H8 se fait au moyen de conduits appelés "ombilicaux". A chaque réservoir est associé un (ou plusieurs) clapet(s) dont la commande en fermeture peut à tout moment interrompre cette liaison et isolerle(s) réservoir(s) concerné(s). C'est le cas notamment lorsqu'il y a une fuite importante au niveau des "ombilicaux".

La pressurisation des deux réservoirs et la surveillance de celle-ci est assurée par un système automatique. En cas de défaillance de ce dernier, des automatismes assurant des fonctions de reprise en secours peuvent être manuellement mis en service. De plus, un dispositif de sécurité fonctionne en permanence quelle que soit la configuration. Ce dernier

est appelé "automatisme de sécurité" (A.S) et est prioritaire sur les autres automatismes.

L'ensemble de ces automatismes tel que défini dans [X78] comporte :

. 2 fonctions de régulation de pression :
  - Automatisme AX : régulation de pression du réservoir O2
  - Automatisme AY : régulation de pression du réservoir H2

. 1 fonction de régulation de niveau :
  - Automatisme A11 : régulation grossière des niveaux des réservoirs O2 et H2.

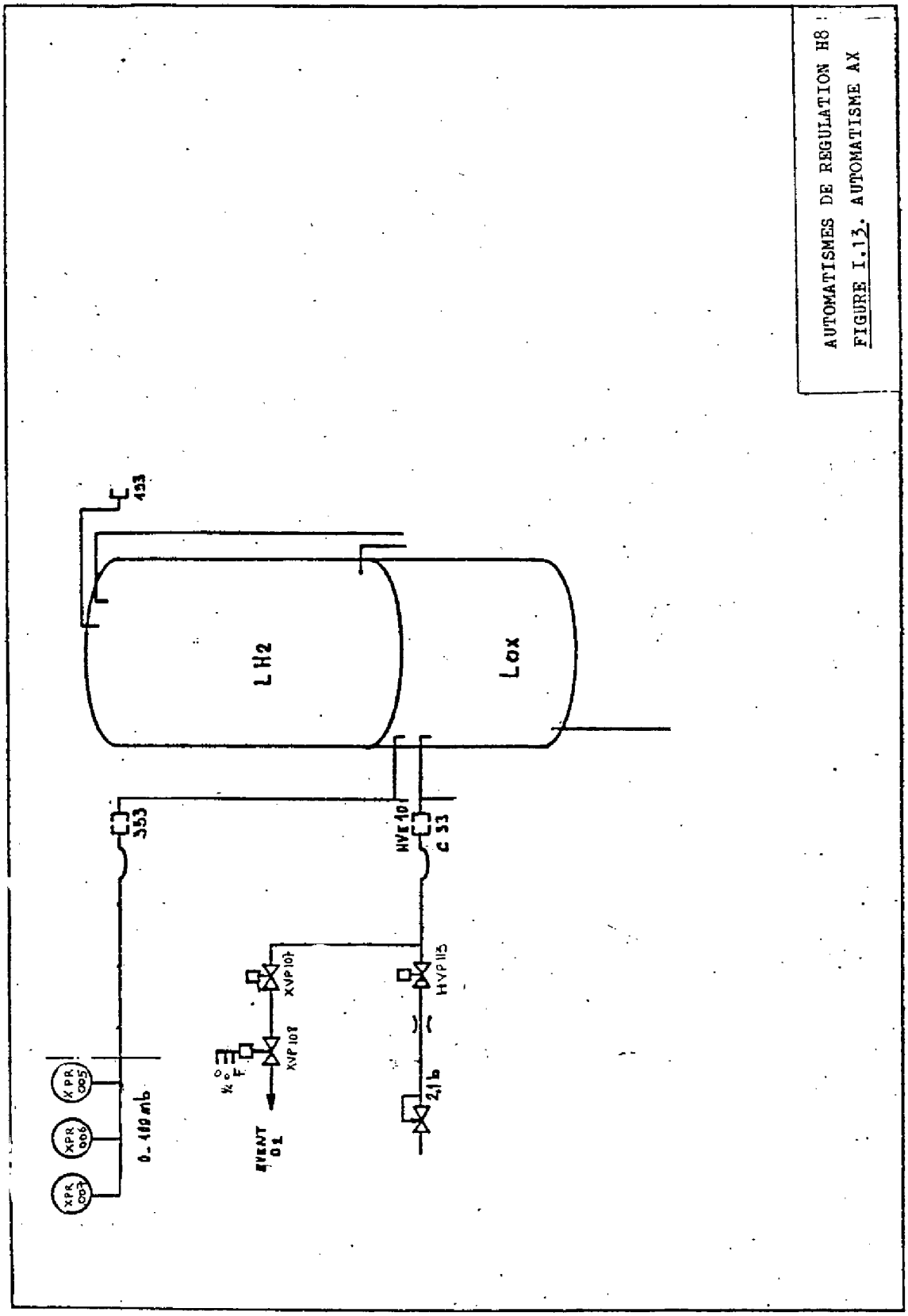
. 3 fonctions de surveillance de pression :
  - Automatisme PX : surveillance de la pression maximale du réservoir O2
  - Automatisme A7 : Maintien de la pression des réservoirs O2 et H2 dans des créneaux prédéfinis et surveillance de la différence de pression du fond intermédiaire.
  - Automatisme AS : surveillance des pressions des réservoirs O2 et H2 et de la différence de pression du fond intermédiaire. Sa mise en service est prévue pendant toutes les opérations entraînant des variations de pression incompatibles avec les seuils limites définis.

Nous allons à partir de la description en langage naturel qui nous a été communiquée [X78] , spécifier à l'aide des réseaux de Pétri chacun de ces automatismes et éventuellement mettre en évidence certaines ambiguités ou omissions.

I.4.2. Spécification des automatismes à l'aide des réseaux de Pétri

La spécification qui va être faite est une spécification fonctionnelle et non une spécification d'implémentation. Il s'agit de préciser ce que les automatismes doivent faire sans entrer dans les détails technologiques, c'est-à-dire sans détailler leur mise en oeuvre.

Comme nous l'avons vu (cf.I.2.3) , les actions sont en général associées aux places, sauf celles qui sont considérées comme des actions de nature impulsionnelle (leur durée d'exécution est très courte) qui sont


AUTOMATISMES DE REGULATION H8 :
FIGURE I.13. AUTOMATISME AX
133
L H2
Lox
353
HVE 101
C 53
XVP 101
XVP 102
HVP 115
2,1 b
EVENT
0,1
0..10 mb
XPR 005
XPR 006
XPR 003

associées aux transitions. Dès qu'un jeton est mis dans une place, l'action associée est exécutée et cette exécution ne s'arrêtera que lorsque que le jeton sera enlevé.

Les automatismes du H8 ne sont pas indépendants les uns des autres. Néanmoins pour des raisons de clarté, ils seront d'abord spécifiés individuellement et ce n'est qu'ensuite que la synchronisation globale sera considérée.

Par convention, on suppose que les commandes des vannes sont à trois états. Quand un ordre de fermeture est donné, la vanne est fermée et elle reste verrouillée dans cette position tant que l'ordre est maintenu. Quand un ordre d'ouverture est donné, la vanne est ouverte et elle le reste tant que l'ordre est maintenu. Quand aucun ordre n'est donné, la vanne est déverrouillée et peut être commandée manuellement dans n'importe quelle position.

1.4.2.1. Spécification des automatismes pris individuellement

a/ Automatisme de régulation fine de la pression du réservoir d'oxygène liquide AX.

1) Spécification en langage naturel (voir aussi figure I.13)

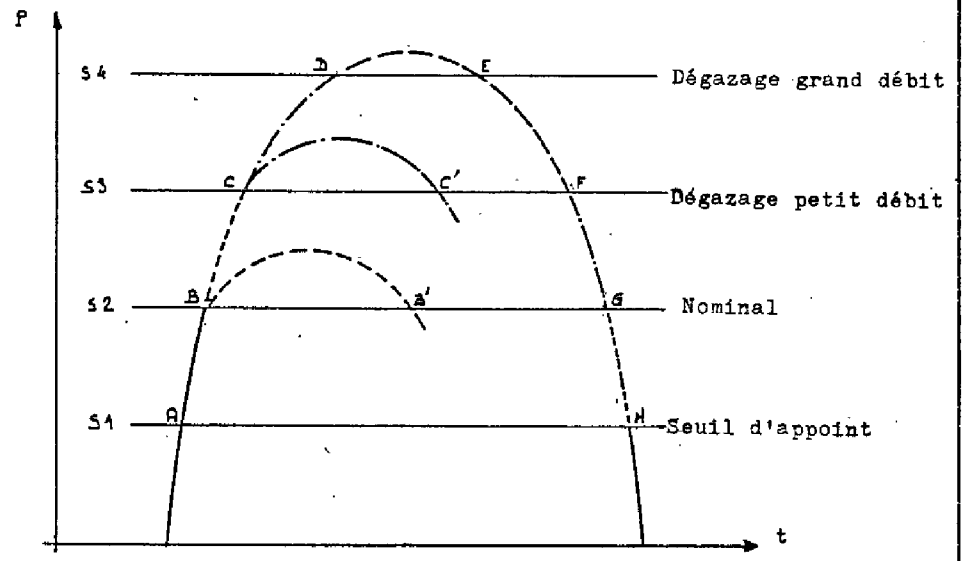
Cet automatisme est mis en service manuellement. Il utilise trois capteurs : XPR005, XPR006, XPR007. La régulation de pression est réalisée autour de quatre circuits appelés dans l'ordre croissant S1, S2, S3, S4 :

- S1 : seuil bas d'appoint
- S2 : seuil normal
- S3 : seuil de petit dégazege
- S4 : seuil de grand dégazege

Trois vannes sont utilisées en cours de fonctionnement : HVP 113, XVP 107, XVP 108, cette dernière étant une vanne trois positions (ouverture, demi-ouverture, fermeture).

Le fonctionnement est schématisé par la figure I.14 qui donne les ordres à exécuter en fonction des seuils de pression (P) atteints :

AUTOMATISMES A_X ET A_Y



	A_X	A_Y
ordre 5	Fermeture XVP 107 " " XVP 108 Ouverture HVP 113	Fermeture YVP 514 " " YVP 515 Ouverture HVP 513
ordre 6	Fermeture XVP 107 " " HVP 113 1/2 ouverture XVP 108	Fermeture YVP 514 " " HVP 513 Ouverture YVP 515
ordre 7	Fermeture HVP 113 Ouverture XVP 107 1/2 ouverture XVP 108	Fermeture HVP 513 Ouverture YVP 514 1/2 ouverture YVP 515
ordre 8	Fermeture HVP 113 Ouverture XVP 107 " " XVP 108	Fermeture HVP 513 Ouverture YVP 514 " " YVP 515

FIGURE I.14

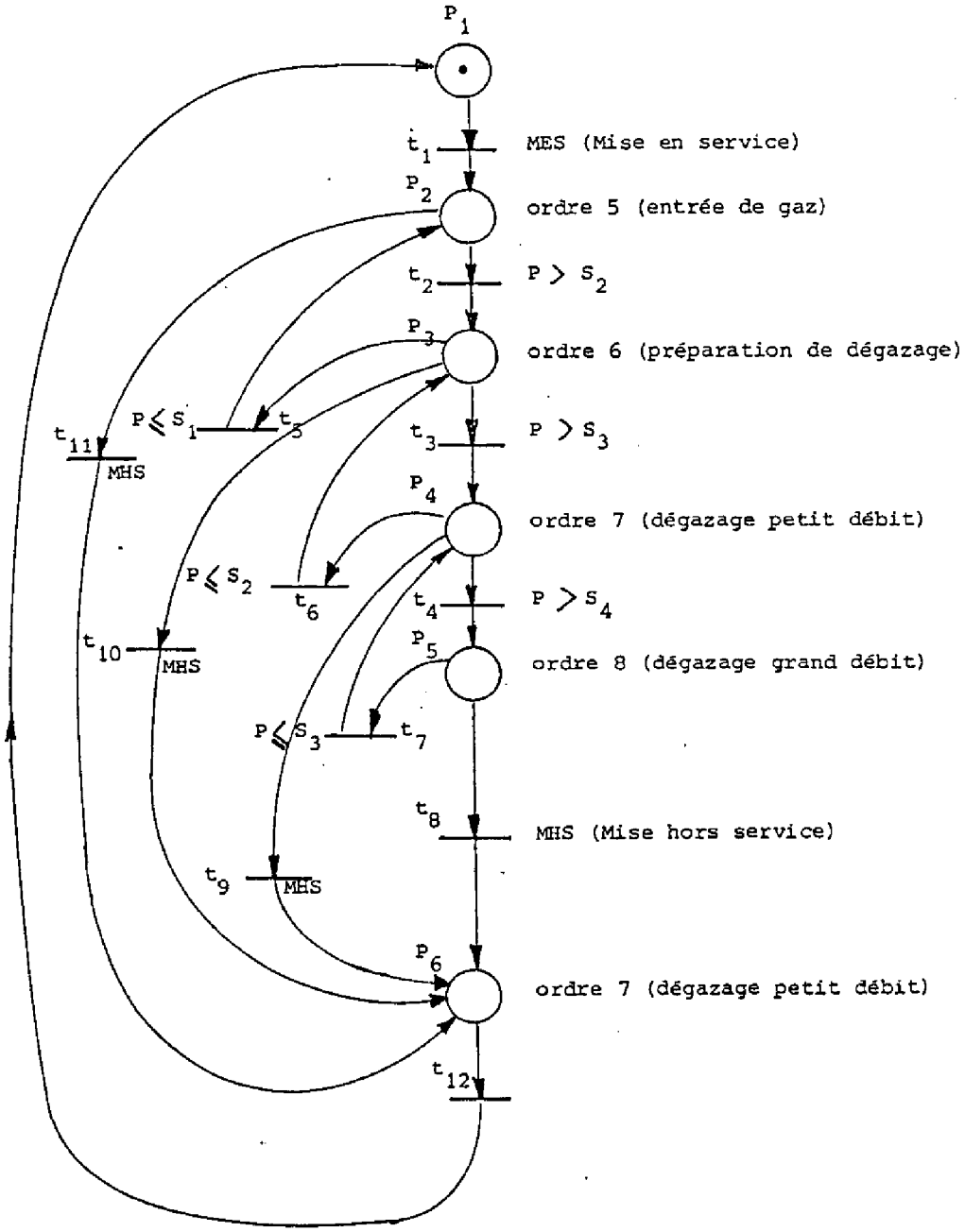


Figure I.15 *Spécification de AX*

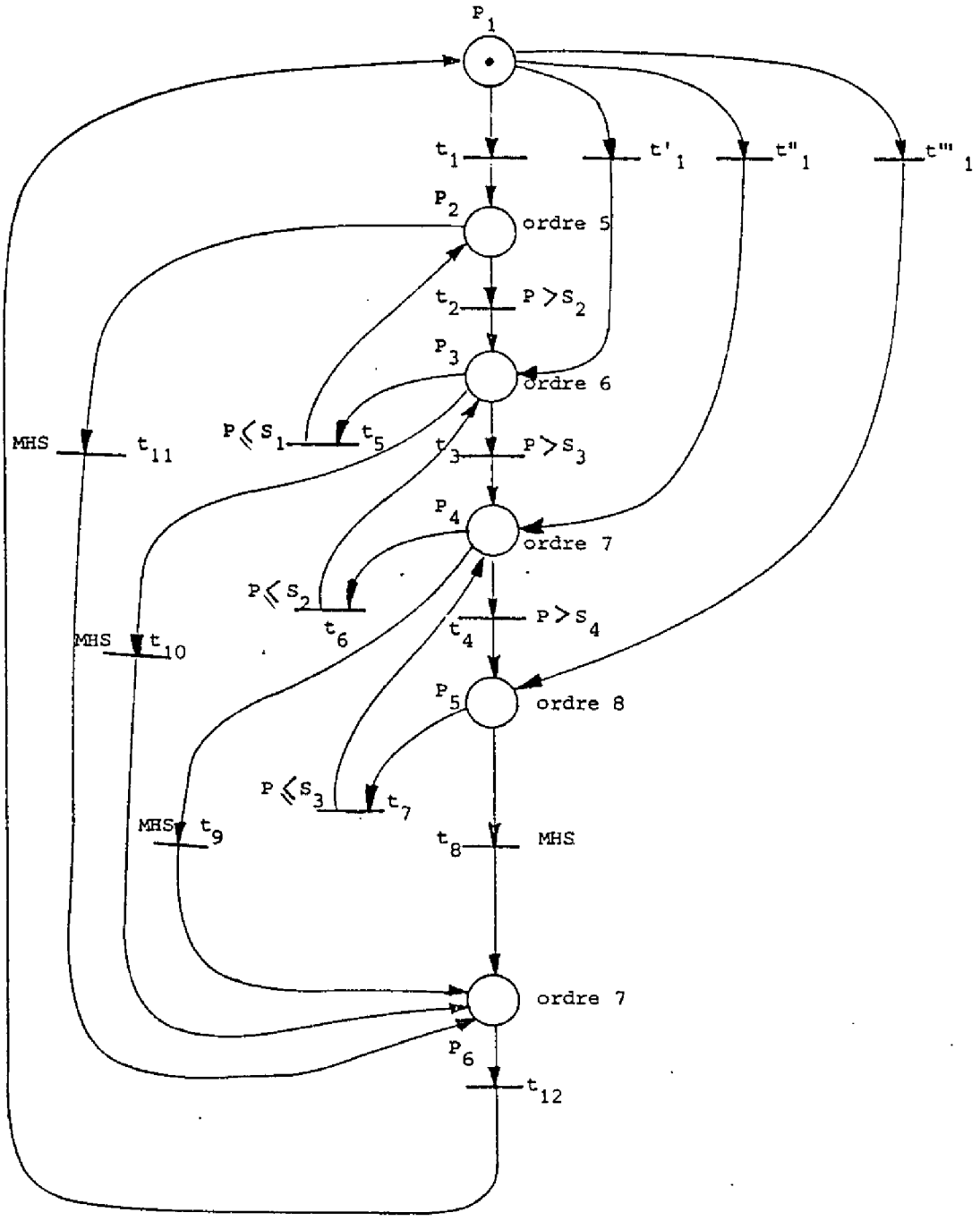
. La mise hors fonctionnement force en ouverture la XVP 107, en fermeture la AVP 113 et en demi-ouverture la XVP 108. Cette commande n'est autre que l'ordre 7.

2) Spécification à l'aide des réseaux de Pétri

Le réseau de Pétri de la figure I.15 correspond aux spécifications en langage naturel de cet automatisme. L'élaboration de ce réseau de Pétri fait surgir deux questions :

La première concerne la mise en service de l'automatisme. La spécification donnée en langage naturel laisse supposer que cette mise en service n'a lieu que lorsque la pression P est inférieure au seuil S2. Dans le cas contraire, la spécification de la figure I.15 implique que l'ordre 5 sera donné de manière fugitive lors de la mise en service, jusqu'à ce que la transition t_2 soit tirée. Si la possibilité d'ordre fugitif est inacceptable, la spécification peut être complétée de deux manières différentes. La première consiste à associer à la transition t_1 la réceptivité (condition logique) suivante : "Mise en Service et P<S2". Il en résultera un verrouillage de la mise en service de AX lorsque la pression est "incorrecte". La seconde solution consiste à dédoubler la transition t_1 de mise en service en quatre transitions de façon à passer directement dans l'état correct. Dans ce cas, la spécification de AX est donnée par le réseau de la figure I.16.

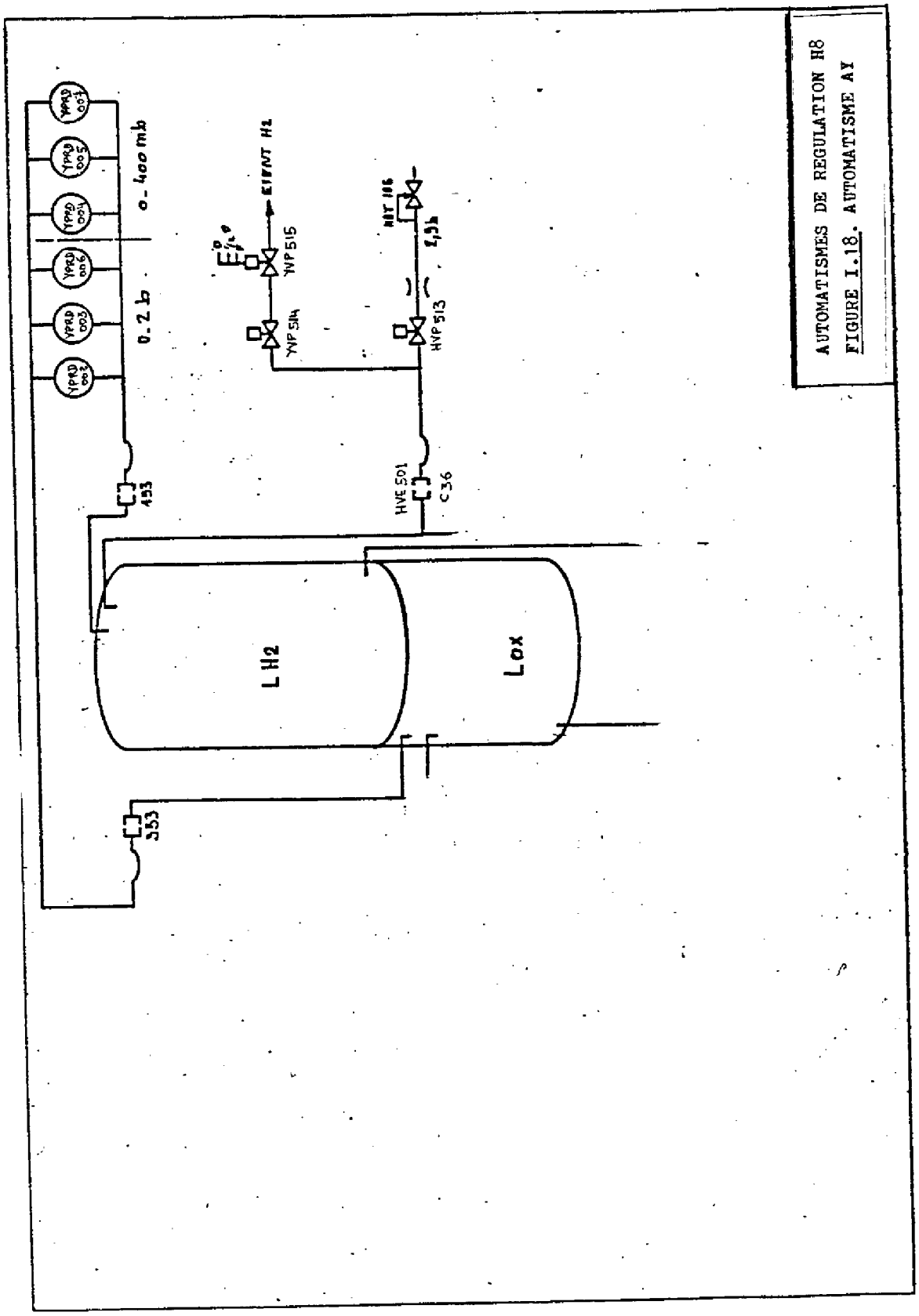
La seconde question posée par l'élaboration du réseau de Pétri de la figure 1.15 provient de la transition t_{12} car elle ne représente pas une synchronisation interne et aucune réceptivité n'y est associée. Ceci provient du manque de précision de la phrase "la mise hors fonctionnement force en ouverture la XVP 107...". Jusqu'à quand faut-il maintenir ce forçage ? Si l'on désire garantir que l'ordre donné sera exécuté complètement sans être interrompu par un contre-ordre, il est nécessaire d'introduire une temporisation qui fera que la transition t_{12} ne sera tirée qu'au bout d'un temps donné. Ceci correspond à la modification du réseau proposé dans la figure I.17. Il faut souligner que l'opération "lancement de la temporisation" étant une action instantanée et indivisible, elle est associée à des transitions et non à des places.



Réceptivités :

t_1	:	MES $\wedge$ $(P \leqslant S_2)$
t'_1	:	MES $\wedge$ $(S_2 < P \leqslant S_3)$
t''_1	:	MES $\wedge$ $(S_3 < P \leqslant S_4)$
t'''_1	:	MES $\wedge$ $(S_4 \leqslant P)$

Figure I.16 *Deuxième spécification de AX*


AUTOMATISMES DE REGULATION H8
FIGURE I.18. AUTOMATISME AY
0 - 400 mb
0.2 b
YPRD 001
YPRD 002
YPRD 003
YPRD 004
YPRD 005
YPRD 006
ETAT H2
YVP 515
YVP 514
YVP 513
HVT 116
1,5 b
HVE 501
C36
153
353
L H2
Lox

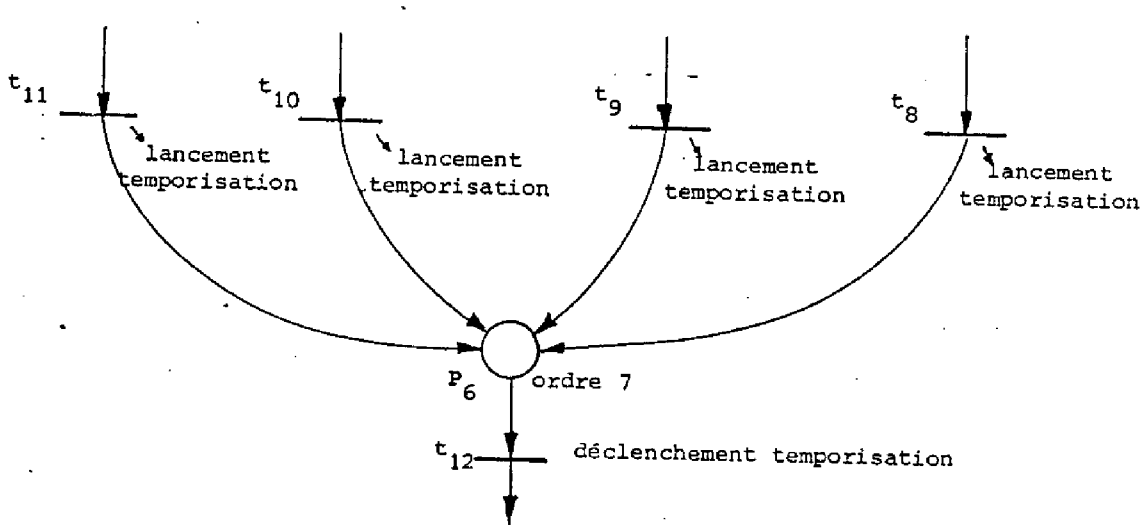


Figure I.17 - Spécification de la temporisation

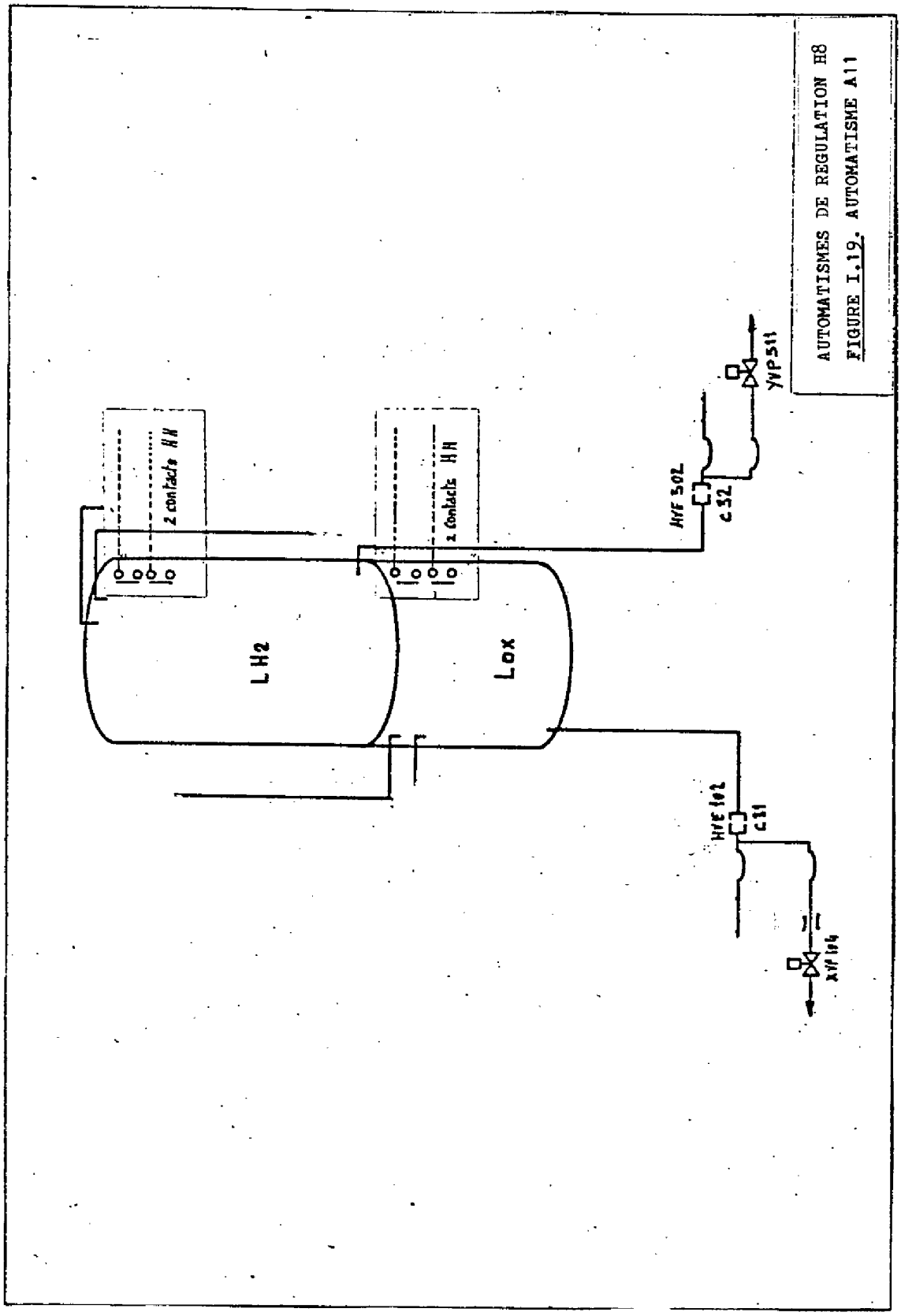
b/ Automatisme de régulation fine de la pression du réservoir d'hydrogène liquide : AY

Le fonctionnement de cet automatisme est tout à fait similaire à celui de l'automatisme AX (voir figure I.14). Les seules différences sont les noms des vannes commandées (voir figure I.18) et l'ordre associé à la place P6 qui devient : "fermeture de HVP 513, YVP 514 et YVP 515" et que nous appelons ordre 9.

c/ Automatisme de régulation de niveau des compléments de plein des réservoirs du H8 : A11.

1) Spécification en langage naturel (voir aussi figure I.9)

Cet automatisme est mis en fonctionnement manuellement. Il utilise deux capteurs-bord tout-ou-rien de niveaux "haut-haut" par réservoir. Il assure les compléments de pleins pour chacun des réservoirs à partir des comptes rendus délivrés par l'un ou l'autre des deux capteurs ou par les deux.


AUTOMATISMES DE REGULATION H8
FIGURE I.19. AUTOMATISME A11
2 contacts HH
2 contacts HH
LH2
Lox
HYF 502
C32
YVP511
HYE 102
C11
XVF 104

Les compléments de plein sont ainsi définis :

- apparition d'un des niveaux "haut-haut" : ouverture de la vanne XVP 104 (ou XVP 511) et ouverture de la HVE 102 (ou HVE 502). L'ouverture de la HVE 502 correspond à la fermeture clapet associé. Démarrage de la temporisation ;

- En fin de temporisation et si disparition du (ou des) capteur(s) "haut-haut" sélectionné(s), fermeture de la XVP 104 (ou YVP 511) et fermeture de la HVE 102 (ou HVE 502).

- A la fin de la temporisation et si non disparition du (ou des) capteur(s) sélectionné(s), la temporisation est relancée.

2) Spécification à l'aide des réseaux de Pétri

Le réseau de Pétri de la figure I.20 correspond à la spécification de l'automatisme A11 avec l'interprétation suivante des symboles :

MES : Mise en Service de A11.

MHS : Mise Hors Service de A11.

HH : Niveau "Haut-Haut" d'un des capteurs-bord atteint.

∿HH : Niveau "Haut-Haut" du (ou des) capteur(s) sélectionné(s) non atteint.

Ordre 1 : Fermeture de la XVP 104 (YVP 511) - Fermeture de la HVE 102 (HVE 502)

Ordre 2 : Ouverture de la XVP 104 (YVP 511)
Ouverture de la HVE 102 (HVE 502)

↗T : Lancement temporisation

↘T : Fin temporisation

RAZT : Désactivation temporisation

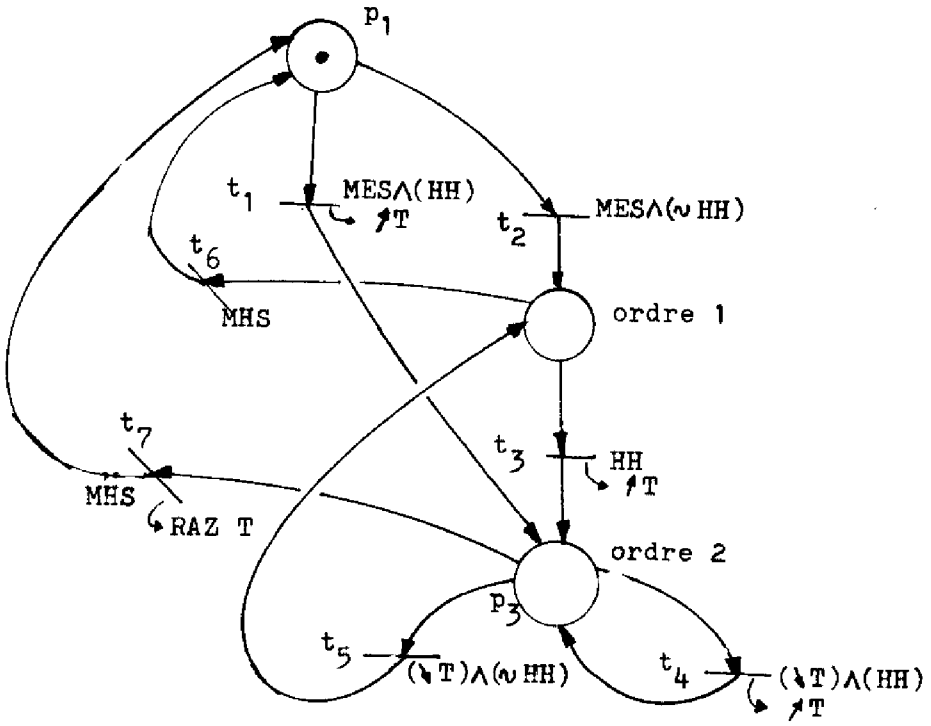
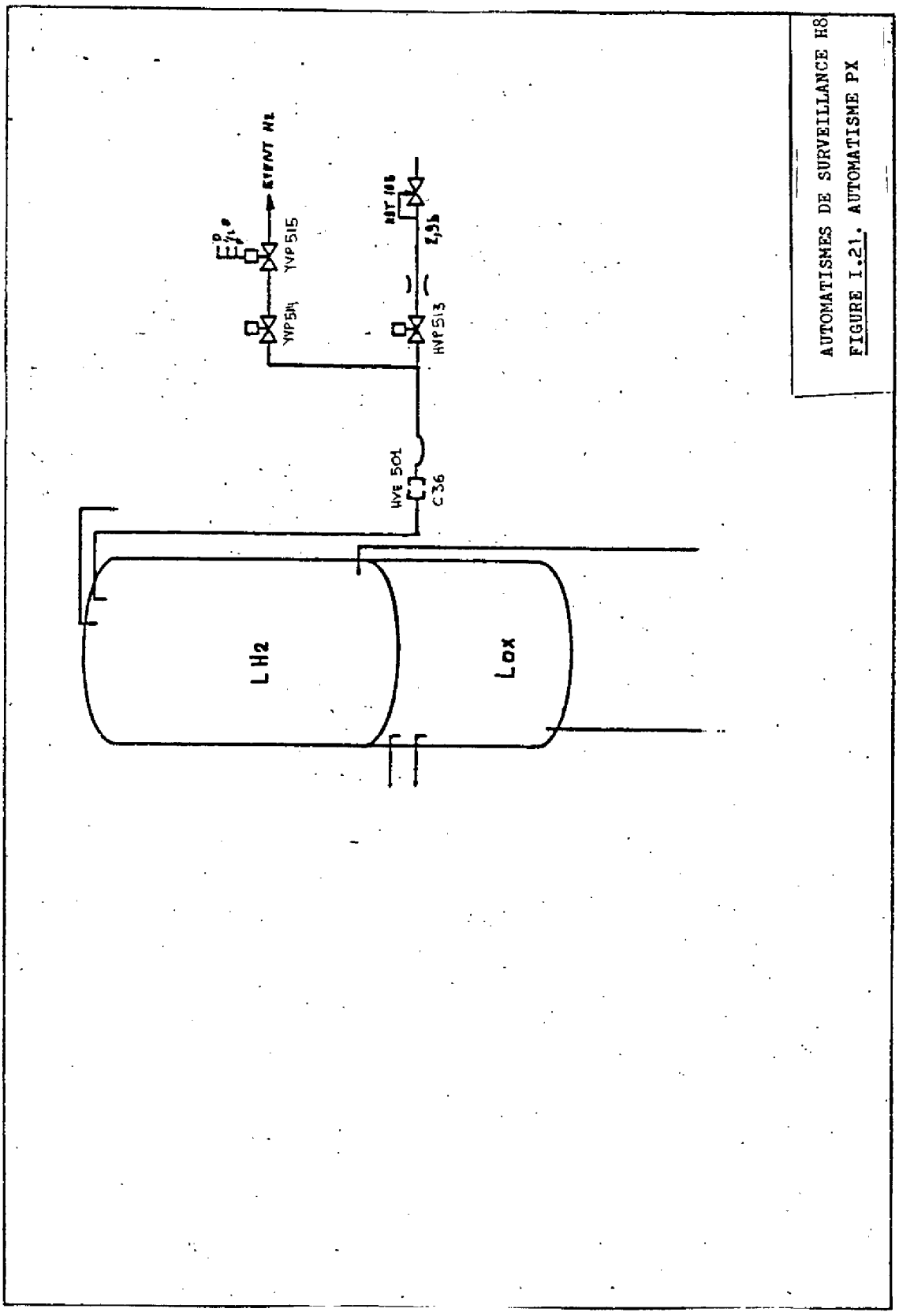


Figure I.20 - Automatisme A11



AUTOMATISMES DE SURVEILLANCE H8

FIGURE I.21. AUTOMATISME PX

d/ Automatisme de surveillance de la pression maximale du réservoir d'oxygène liquide (PX)

1) Spécification en langage naturel (voir aussi figure I.21)

Cet automatisme est mis en fonctionnement manuellement. Afin de protéger le fond intermédiaire, cet automatisme isole les réservoirs H2 en cas de franchissement du seuil de pression maximale autorisé dans le réservoir O2. Trois vannes sont utilisées au cours du fonctionnement (voir figure I.21) : HVP 513, YVP 514 et YVP 515, cette dernière étant une vanne trois positions : ouverture, demi-ouverture et fermeture. Par dépassement du seuil de pression maximale autorisée, les vannes sont fermées et l'automatisme AY est mis hors fonctionnement.

Cet automatisme est dit "transparent à verrouillage" car c'est son déclenchement au lieu de sa mise en service qui interdit toute commande des vannes citées ci-dessous en verrouillant leur position.

La reprise manuelle des commandes après déclenchement de l'automatisme ne peut se faire que par la mise hors-service de celui-ci.

2) Spécification à l'aide des réseaux de Pétri

Le réseau de la figure I.22 correspond à la spécification de l'automatisme PX en considérant l'interprétation suivante pour les étiquettes :

MES : mise en service de PX

PO2 > 115 : le seuil de pression maximale autorisée dans O2 a été franchi

Ordre 1 : mise hors fonctionnement de l'automatisme AY

Ordre 2 : isoler le réservoir H2 (fermeture des trois vannes)

MHS : mise hors service de PX

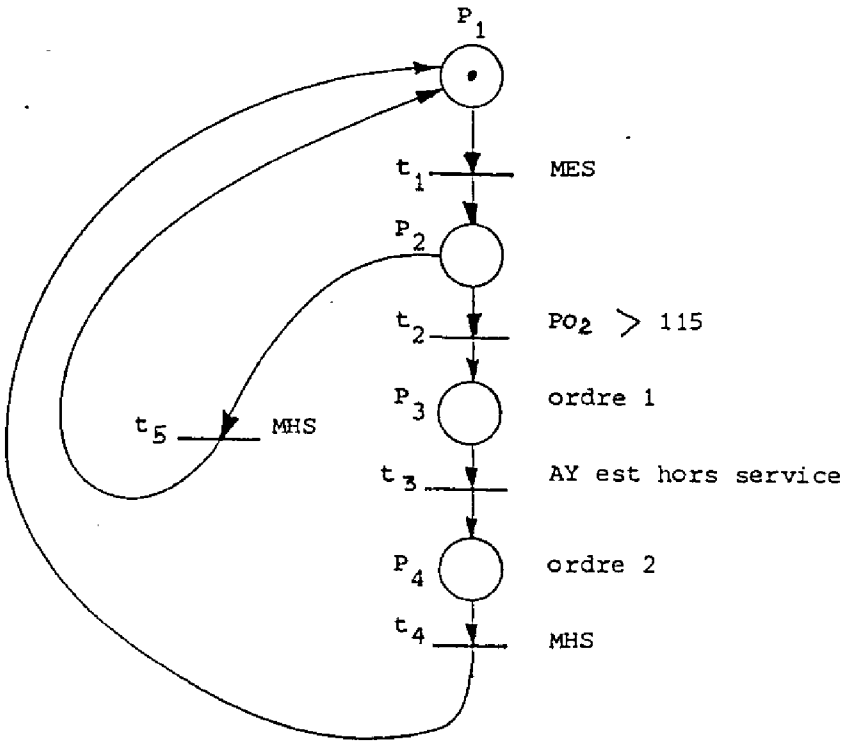


Figure I.22 - Automatisme PX

La spécification du fait que l'automatisme PX est "transparent à verrouillage" est donnée simplement par les actions associées aux places. A la place P2 aucune action n'est associée, cela veut dire que la manipulation des vannes reste possible. L'automate est donc "transparent". Dès que la pression PO2 dépasse le seuil limite, l'automatisme AY est mis hors service (ordre 1) et le tir de t_3 met un jeton dans la place P_4. Les vannes HVP 513, YVP 514 et YVP 515 sont fermées et cet ordre est maintenu tant que la transition t_4 n'est pas tirée, c'est-à-dire tant qu'un ordre de mise hors service de PX n'a pas été donné ; l'automatisme PX est donc verrouillé dans cette position.

e/ Automatisme de surveillance de la pressurisation grossière du réservoir H8 (A7).

1) Spécification en langage naturel (voir aussi figure I.23)

Cet automatisme est mis en fontionnement manuellement. Il utilise les informations de capteurs propres à chaque réservoir O2 et H2 (voir figure I.23, celle-ci donne également la liste des vannes utilisées).

· Réservoir Oxygène : le réservoir oxygène est pressurisé grâce au maintien en ouverture des vannes HVP 107, HVP 108 et du clapet 31 de l'étage (HVE 102). La pression est maintenue dans un créneau prédéfini et la protection du fond intermédiaire impose une limite de pression minimale. Il faut donc empêcher la dépressurisation totale du réservoir qui peut avoir trois origines :

- défaillance du module de pressurisation,
- fuite interne de l'étage, ou ouverture d'une vanne moteur,
- fuite au niveau des"ombilicaux"

Dans les deux premiers cas, la sécurité du réservoir est assurée par la mise en service d'un module dont la tâche consiste en la séquence suivante :

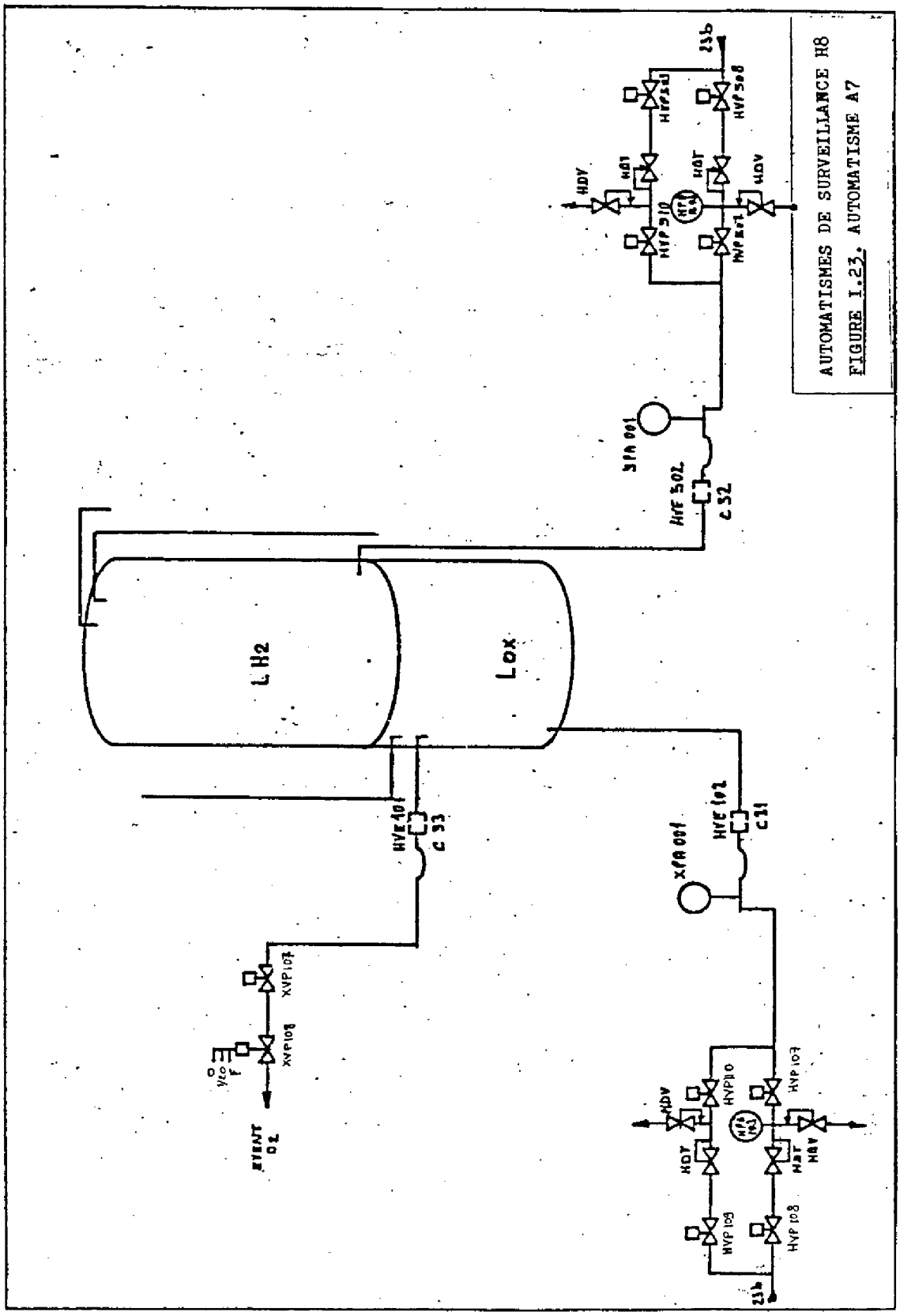
- ouverture des vannes HVP 110 et HVP 109 après fermeture des vannes HVP 107 et HVP 108.

Dans le troisième cas, la fuite pouvant être importante, le clapet 31 est commandé en fermeture.

· Réservoir Hydrogène : le réservoir hydrogène est pressurisé grâce au maintien en ouverture des vannes HVP 507 et HVP 508 et du clapet 32 de l'étage (HVE 502) (voir figure I.23). La pression est maintenue dans un créneau prédéfini. La tenue du fond intermédiaire entre les deux réservoirs impose une pression côté H2 supérieure à celle du côté O2, il faut donc empêcher la dépressurisation de H2 en dessous du seuil limite.

Cette dépressurisation peut avoir trois origines :

· défaillance du module de pressurisation
· fuite interne de l'étage ou ouverture d'une vanne moteur
· fuite au niveau des "ombilicaux"



AUTOMATISMES DE SURVEILLANCE H8

FIGURE 1.23. AUTOMATISME A7

Dans le premier cas et dans le deuxième cas, si la fuite est faible, la sécurité est assurée par la mise en service d'un module qui se charge de l'ouverture des vannes HVP 509 et HVP 510 après fermeture des vannes HVP 507 et HVP 508.

Dans le deuxième et troisième cas, si la fuite est importante, les commandes suivantes sont envoyées :

. fermeture du clapet 32 (HVE 502)
. fermeture du clapet 31 (HVE 102)
. ouverture du clapet 33 (HVE 101)
. ouverture des vannes XVP 107 et XVP 108

Cet automatisme est dit "non transparent à verrouillage" car sa mise en service interdit toute commande manuelle des vannes citées ci-dessus et son déclenchement verrouille les vannes dans les positions commandées.

La mise en service de l'automatisme force la position nominale des vannes citées ci-dessus si la pression est nominale.

A la mise hors service, les vannes retrouvent la position correspondant à la dernière commande passée (avant ou pendant la mise en service de A7).

2) Spécification à l'aide des réseaux de Pétri

Cet automatisme est en fait décomposé en deux parties : l'une concerne le réservoir oxygène et l'autre le réservoir hydrogène. Mais ces deux parties ne peuvent pas être complètement indépendantes car chacune peut verrouiller le clapet 31. Bien que cela ne soit pas explicitement mentionné dans les spécifications en langage naturel, il semble logique que le déclenchement, en cas de fuite importante de la partie concernant le réservoir hydrogène entraîne le verrouillage de la partie concernant le réservoir oxygène.

i) Spécification de la partie oxygène

La spécification de la partie concernant le réservoir d'oxygène est donnée par le réseau de la figure I.24. L'interprétation des étiquettes est la suivante :

MES : Mise en Service de A7

MHS : Mise Hors Service de A7

FF : Fuite faible du réservoir d'oxygène

FI : Fuite importante du réservoir d'oxygène

V : Verrouillage de l'automatisme A7 en conséquence d'une fuite importante du réservoir d'hydrogène.

Ordre 1 : Commande en position nominale des vannes HVP 110, HVP 109, HVP 108, HVP 107 et HVP 102 (clapet 31)

Ordre 2 : Fermeture de XVP 107, fermeture de HVP 108, ouverture de HVP 110 et de HVP 109, maintien de la position nominale pour HVE 102 (clapet 31)

Ordre 3 : Ouverture de HVE 102 (fermeture du clapet 31)

Ordre 4 : Commande correspondant à la dernière commande passée pour les vannes HVP 110, HVP 109, HVP 108, HVP 107 et HVE 102 (clapet 31).

On remarquera que la communication avec la partie hydrogène se fait à l'aide d'étiquettes associées aux transitions (t_6, t_9, t_{12}) c'est-à-dire qu'il s'agit d'une communication par fusion de transition (t_6, t_9, t_{12} du réseau O2 avec t_{10} du réseau H2).

La transition t_{13} n'a pas d'étiquette associée. Comme dans le cas de l'automatisme AX, il semble logique d'introduire une temporisation lancée lors du tir de l'une des transitions t_4, t_8 ou t_{11} et dont le déclenchement provoquera le tir de t_{13}. Ainsi l'ordre 4 sera maintenu le temps nécessaire à son exécution complète.

L'exécution de l'ordre 3 n'agit pas sur les vannes HVP 107, HVP 108, HVP 109 et HVP 110, car elles peuvent être dans la position nominale ou non suivant que l'ordre a été appliqué auparavant ou non. Vu les conventions retenues, ces vannes cessent donc d'être verrouillées lorsque la place P_4 est marquée et que le clapet 31 est fermé. Si cela

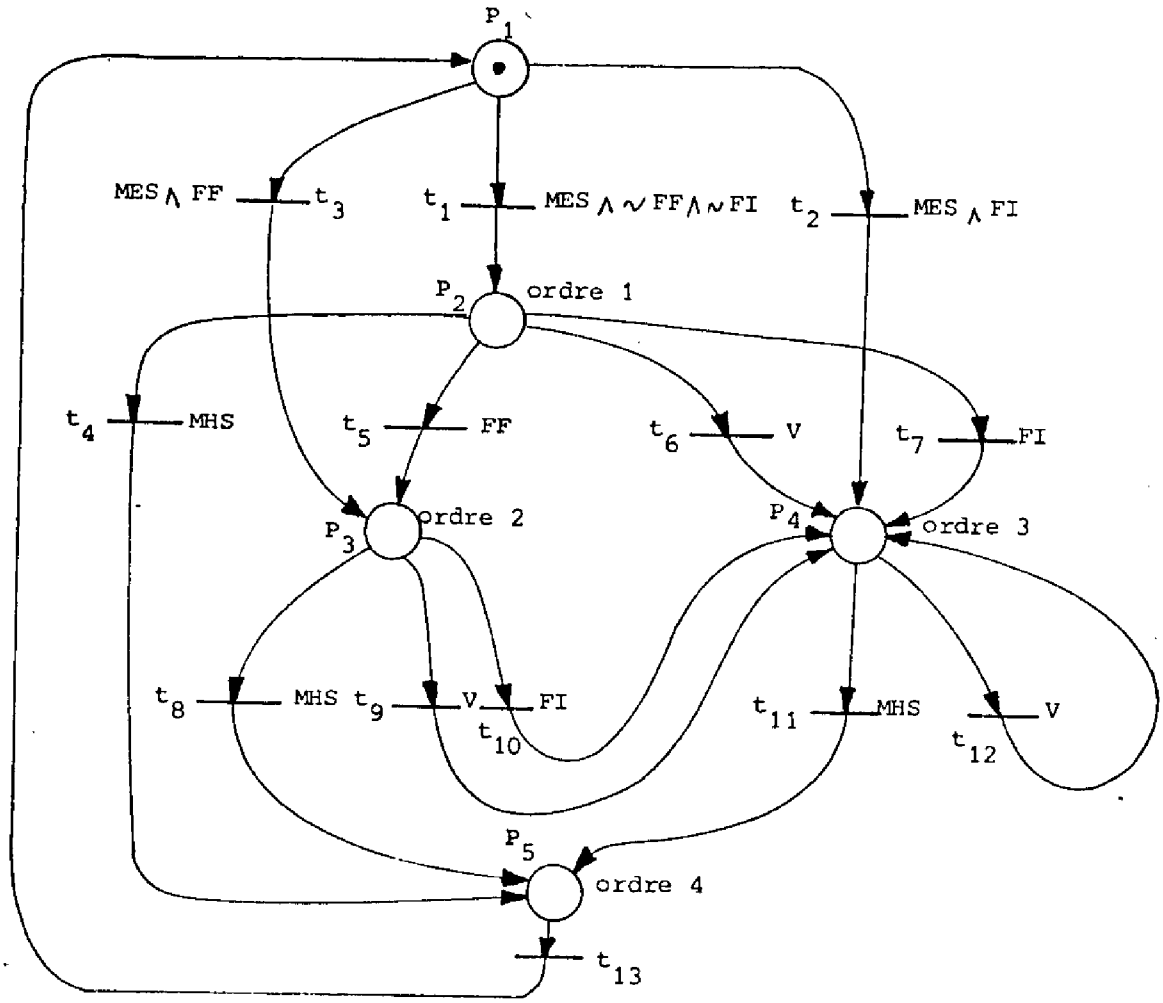


Figure I.24 *Automatisme A7 partie Oxygène*

n'est pas admissible, on peut, soit les verrouiller en position nominale, soit dédoubler la place P4 pour les verrouiller soit dans la position nominale, soit dans la position correspondant à l'ordre .

ii) Spécification de la partie hydrogène

Le réseau de Pétri correspondant est donné par la figure I.25. L'interprétation des étiquettes est la suivante :

MES : Mise en Service de A7

MHS : Mise Hors Service de A7

FF' : Fuite faible du réservoir d'hydrogène

FI' : Fuite importante du réservoir d'hydrogène

V : Verrouillage de la partie oxygène de l'automatisme A7

Ordre 5 : Commande en position nominale des vannes HVP 507, HVP 508, HVP 509, HVP 510, HVE 502 (clapet 32), HVE 101 (clapet 33) XVP 107 et XVP 108.

Ordre 6 : Ouverture des vannes HVP 509 et HVP 510 après fermeture des vannes HVP 507 et HVP 508. Maintien en position nominale des autres vannes.

Ordre 7 : Fermeture du clapet 32 et ouverture du clapet 33, ouverture des vannes XVP 107 et XVP 108

Ordre 8 : Commande correspondant à la dernière commande passée pour les vannes mentionnées dans l'ordre 5.

L'absence d'étiquette pour la transition t_{12} du réseau de la figure I.25 appelle la même remarque que celle concernant la transition t_{13} du réseau de la figure I.24.

D'autre part, la spécification retenue pour les ordres 7 et 8 entraîne un déverrouillage des vannes HVP 507, HVP 508, HVP 509 et HVP 510. Ceci est analogue au problème concernant l'ordre 3 de la partie oxygène.

iii) Spécification de l'ensemble de l'automatisme A7

Le réseau global est obtenu par composition des réseaux des figures I.24 et I.25. Cette composition se fait en particulier en effectuant la fusion de la transition t_{10} du réseau de la figure I.25 avec les transitions t_6, t_9 et t_{12} du réseau de la figure I.24. En effet, le tir

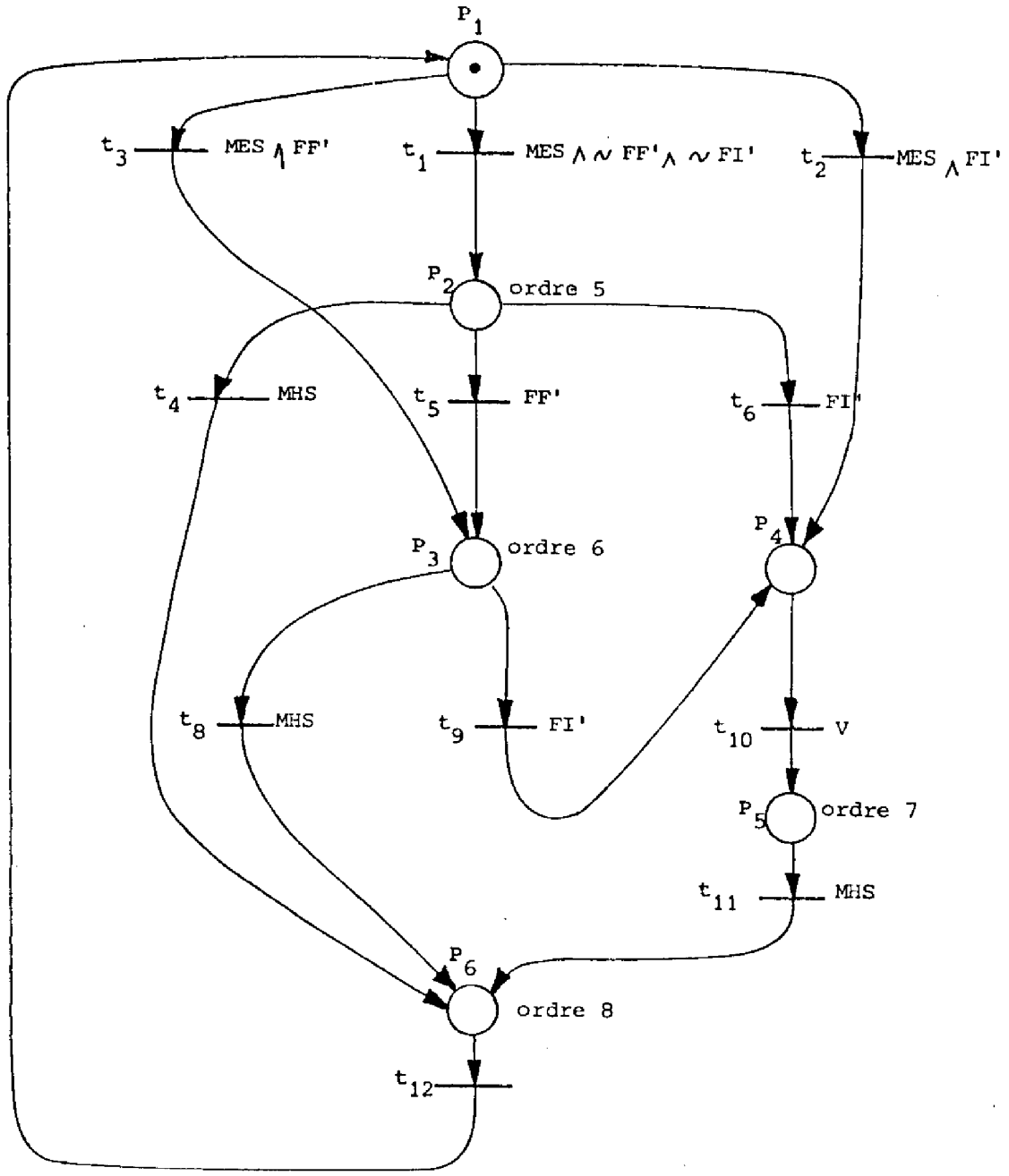


Figure I.25 *Automatisme A7 Partie Hydrogène*

de la transition t_{10} du réseau de la figure I.25 doit être simultané au tir de l'une des transitions t_6, t_9 ou t_{12} du réseau de la figure I.24 (verrouillage de la partie oxygène de l'automatisme A7 en conséquence d'une fuite importante du réservoir d'hydrogène). De même, une mise en service de A7 doit entraîner le tir simultané de l'une des transitions t_1, t_2 ou t_3 du réseau de la figure I.24 avec l'une des transitions t_1, t_2 ou t_3 du réseau de la figure I.25. Le même phénomène apparaît également lors de la mise hors service et dans ces deux cas également, il faut opérer des fusions de transitions. Le réseau composé résultant n'est utile que pour la validation de la synchronisation globale de l'automatisme A7. Il est obtenu automatiquement par caluculateur à partir des réseaux des figures I.24 et I.25, mais sa représentaion graphique serait trop complexe pour apporter quelque chose au niveau des spécifications.

f/ Automatisme de sécurité des réservoirs du H8 (AS)

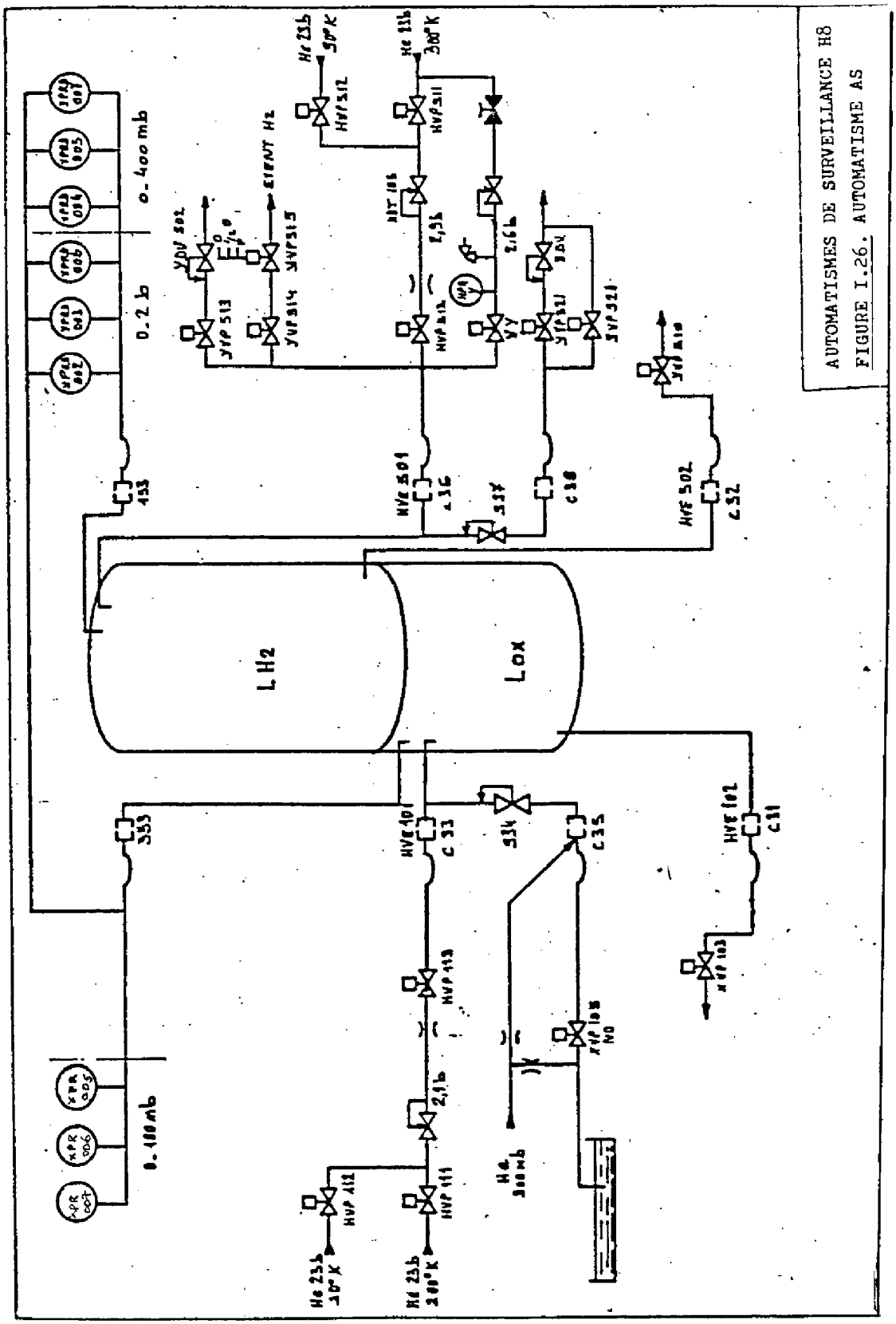
1) Spécification en langage naturel (voir aussi figure I.26)

Il est mis en service manuellement. Il utilise 9 capteurs (voir figure I.26, celle-ci donne également la liste des vannes mises en service) Le dépassement de seuils limites détectés au moyen de ces capteurs entraîne les actions de verrouillage suivantes :

- fermeture de la vanne HVE 501 et suivantes [X 78]
- ouverture de la vanne XVP 111 et suivantes [X 78]
- mise hors des automatismes AX, AY, A11, PX et A7.
- coupure des commandes de l'automatisme principal en ce qui concerne les réseaux hydrogène et oxygène
- signalisation du déclenchement. Le déverrouillage n'est possible que manuellement. Cet automatisme est dit "transparent à verrouillage".
- A la suite de la mise hors service de AS, les vannes retrouvent la position de la dernière commande passée.

2) Spécification à l'aide des réseaux de Pétri

Le réseau de Pétri de le figure I.27 donne la spécification de cet automatisme. La réceptivité "alarme" correspond au franchissement de seuils limites. L'action associée à la place P4 correspond au verrouilage des vannes listées dans la spécification en langage naturel [X 78] .


AUTOMATISMES DE SURVEILLANCE H8
FIGURE I.26. AUTOMATISME AS
LH2
LOX
0-400 mb
0-100 mb
0,2 b
He 23 b 50°K
He 11 b 300°K
EVENT H2
He 23 b 50°K
He 23 b 300°K
H2 300 mb
HVF 501
HVF 502
HVF 101
HVF 102
HVF 103
HVF 111
HVF 112
HVF 113
HVF 511
HVF 512
YVF 513
YVF 514
YVF 515
YVF 521
YVF 522
YVF 523
YVF 524
YVF 525
YVF 526
YVF 527
YVF 528
YVF 529
YVF 530
YVF 531
YVF 532
YVF 533
YVF 534
YVF 535
YVF 536
YVF 537
YVF 538
YVF 539
YVF 540
YVF 541
YVF 542
YVF 543
YVF 544
YVF 545
YVF 546
YVF 547
YVF 548
YVF 549
YVF 550
YVF 551
YVF 552
YVF 553
YVF 554
YVF 555
YVF 556
YVF 557
YVF 558
YVF 559
YVF 560
YVF 561
YVF 562
YVF 563
YVF 564
YVF 565
YVF 566
YVF 567
YVF 568
YVF 569
YVF 570
YVF 571
YVF 572
YVF 573
YVF 574
YVF 575
YVF 576
YVF 577
YVF 578
YVF 579
YVF 580
YVF 581
YVF 582
YVF 583
YVF 584
YVF 585
YVF 586
YVF 587
YVF 588
YVF 589
YVF 590
YVF 591
YVF 592
YVF 593
YVF 594
YVF 595
YVF 596
YVF 597
YVF 598
YVF 599
YVF 600
YVF 601
YVF 602
YVF 603
YVF 604
YVF 605
YVF 606
YVF 607
YVF 608
YVF 609
YVF 610
YVF 611
YVF 612
YVF 613
YVF 614
YVF 615
YVF 616
YVF 617
YVF 618
YVF 619
YVF 620
YVF 621
YVF 622
YVF 623
YVF 624
YVF 625
YVF 626
YVF 627
YVF 628
YVF 629
YVF 630
YVF 631
YVF 632
YVF 633
YVF 634
YVF 635
YVF 636
YVF 637
YVF 638
YVF 639
YVF 640
YVF 641
YVF 642
YVF 643
YVF 644
YVF 645
YVF 646
YVF 647
YVF 648
YVF 649
YVF 650
YVF 651
YVF 652
YVF 653
YVF 654
YVF 655
YVF 656
YVF 657
YVF 658
YVF 659
YVF 660
YVF 661
YVF 662
YVF 663
YVF 664
YVF 665
YVF 666
YVF 667
YVF 668
YVF 669
YVF 670
YVF 671
YVF 672
YVF 673
YVF 674
YVF 675
YVF 676
YVF 677
YVF 678
YVF 679
YVF 680
YVF 681
YVF 682
YVF 683
YVF 684
YVF 685
YVF 686
YVF 687
YVF 688
YVF 689
YVF 690
YVF 691
YVF 692
YVF 693
YVF 694
YVF 695
YVF 696
YVF 697
YVF 698
YVF 699
YVF 700
YVF 701
YVF 702
YVF 703
YVF 704
YVF 705
YVF 706
YVF 707
YVF 708
YVF 709
YVF 710
YVF 711
YVF 712
YVF 713
YVF 714
YVF 715
YVF 716
YVF 717
YVF 718
YVF 719
YVF 720
YVF 721
YVF 722
YVF 723
YVF 724
YVF 725
YVF 726
YVF 727
YVF 728
YVF 729
YVF 730
YVF 731
YVF 732
YVF 733
YVF 734
YVF 735
YVF 736
YVF 737
YVF 738
YVF 739
YVF 740
YVF 741
YVF 742
YVF 743
YVF 744
YVF 745
YVF 746
YVF 747
YVF 748
YVF 749
YVF 750
YVF 751
YVF 752
YVF 753
YVF 754
YVF 755
YVF 756
YVF 757
YVF 758
YVF 759
YVF 760
YVF 761
YVF 762
YVF 763
YVF 764
YVF 765
YVF 766
YVF 767
YVF 768
YVF 769
YVF 770
YVF 771
YVF 772
YVF 773
YVF 774
YVF 775
YVF 776
YVF 777
YVF 778
YVF 779
YVF 780
YVF 781
YVF 782
YVF 783
YVF 784
YVF 785
YVF 786
YVF 787
YVF 788
YVF 789
YVF 790
YVF 791
YVF 792
YVF 793
YVF 794
YVF 795
YVF 796
YVF 797
YVF 798
YVF 799
YVF 800
YVF 801
YVF 802
YVF 803
YVF 804
YVF 805
YVF 806
YVF 807
YVF 808
YVF 809
YVF 810
YVF 811
YVF 812
YVF 813
YVF 814
YVF 815
YVF 816
YVF 817
YVF 818
YVF 819
YVF 820
YVF 821
YVF 822
YVF 823
YVF 824
YVF 825
YVF 826
YVF 827
YVF 828
YVF 829
YVF 830
YVF 831
YVF 832
YVF 833
YVF 834
YVF 835
YVF 836
YVF 837
YVF 838
YVF 839
YVF 840
YVF 841
YVF 842
YVF 843
YVF 844
YVF 845
YVF 846
YVF 847
YVF 848
YVF 849
YVF 850
YVF 851
YVF 852
YVF 853
YVF 854
YVF 855
YVF 856
YVF 857
YVF 858
YVF 859
YVF 860
YVF 861
YVF 862
YVF 863
YVF 864
YVF 865
YVF 866
YVF 867
YVF 868
YVF 869
YVF 870
YVF 871
YVF 872
YVF 873
YVF 874
YVF 875
YVF 876
YVF 877
YVF 878
YVF 879
YVF 880
YVF 881
YVF 882
YVF 883
YVF 884
YVF 885
YVF 886
YVF 887
YVF 888
YVF 889
YVF 890
YVF 891
YVF 892
YVF 893
YVF 894
YVF 895
YVF 896
YVF 897
YVF 898
YVF 899
YVF 900
YVF 901
YVF 902
YVF 903
YVF 904
YVF 905
YVF 906
YVF 907
YVF 908
YVF 909
YVF 910
YVF 911
YVF 912
YVF 913
YVF 914
YVF 915
YVF 916
YVF 917
YVF 918
YVF 919
YVF 920
YVF 921
YVF 922
YVF 923
YVF 924
YVF 925
YVF 926
YVF 927
YVF 928
YVF 929
YVF 930
YVF 931
YVF 932
YVF 933
YVF 934
YVF 935
YVF 936
YVF 937
YVF 938
YVF 939
YVF 940
YVF 941
YVF 942
YVF 943
YVF 944
YVF 945
YVF 946
YVF 947
YVF 948
YVF 949
YVF 950
YVF 951
YVF 952
YVF 953
YVF 954
YVF 955
YVF 956
YVF 957
YVF 958
YVF 959
YVF 960
YVF 961
YVF 962
YVF 963
YVF 964
YVF 965
YVF 966
YVF 967
YVF 968
YVF 969
YVF 970
YVF 971
YVF 972
YVF 973
YVF 974
YVF 975
YVF 976
YVF 977
YVF 978
YVF 979
YVF 980
YVF 981
YVF 982
YVF 983
YVF 984
YVF 985
YVF 986
YVF 987
YVF 988
YVF 989
YVF 990
YVF 991
YVF 992
YVF 993
YVF 994
YVF 995
YVF 996
YVF 997
YVF 998
YVF 999
YVF 1000

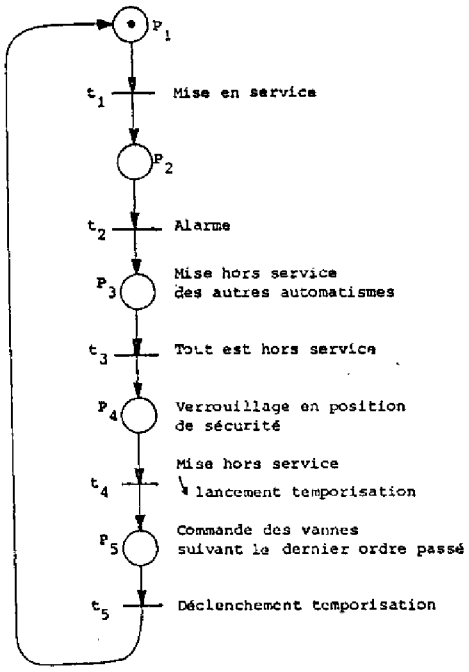


Figure I.27 - Automatisme AS.

I.4.2.2 Spécification de la synchronisation entre les automatismes du H8

D'après les spécifications en langage naturel, les liens existants entre les automatismes sont les suivants :

- la mise en service simultanée des automatismes A7 et A11, ainsi que A7 et AX est interdite.
- le déclenchement de PX provoque la mise hors service de AY
- le déclanchement de AS provoque la mise hors service de tous les autres automatismes.

Nous allons considérer les points les uns après les autres.

a/ Synchronisation entre A7, A11 et AX

La synchronisation entre A7, A11 et AX ne concerne que les mises en service et les mises hors service. Ces automatismes peuvent donc être considérés à un niveau d'abstraction supérieur où l'on ne prend en compte que les états "en service" et "hors service". La spécification de leur synchronisation est alors donnée par le réseau de la figure I.28 où les automatismes AX, A7 et A11 ont été spécifiés de manière globale, la place PS1 exprime que la mise en fonctionnement simultané de AX et A7

est interdite (un seul jeton pour deux transitions de mise en service : MES) et la place PS2 exprime que la mise en service simultanée de A7 et A11 est interdite.

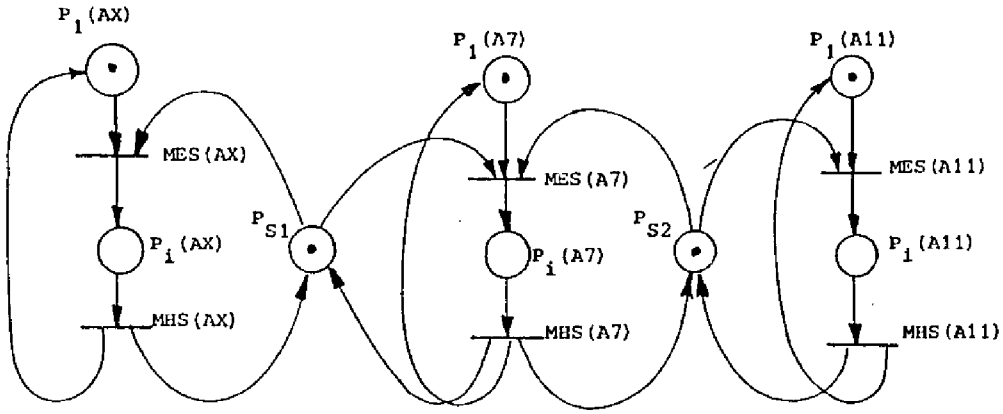


Figure I.28 - Synchronisation entre A7, A11 et AX

b/ Synchronisation entre PX et AY

Du point de vue de l'automatisme AY, seuls interviennent les évènements "mise en service" et "mise hors service". Il faut toutefois souligner que cette "mise hors service" doit avoir lieu quel que soit l'état interne de cet automatisme. On rencontre ici une difficulté classique concernant la spécification graphique des automatismes logiques : les arrêts d'urgence.

Nous avons vu au paragraphe I.3 de ce même chapitre qu'une telle synchronisation pouvait être représentée par une relation "maître-esclave", le réseau esclave étant réduit à une place initialement marquée. C'est ainsi que la synchronisation entre PX et AY est donnée par le réseau de Pétri de la figure I.29, où le réseau de Pétri représentant l'automatisme AY (esclave) est condensé en une place. D'autre part, l'ordre 1 associé à la place P3 du réseau de Pétri de la figure I.22 représentant l'automatisme PX disparaît car il est directement explicité par le réseau de Pétri.

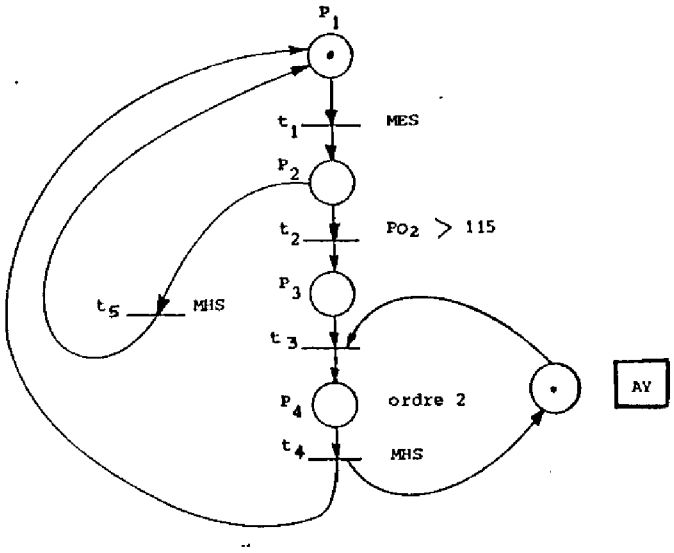


Figure I.29 - Représentation condensée de la synchronisation entre PX et AY

Le fonctionnement de la synchronisation est le suivant : en cas d'alarme (PO2 > 115) le tir de la transition t_3 va enlever le jeton de la place représentant l'automatisme AY (c'est-à-dire quelle que soit la place interne marquée de AY), ce jeton ne sera rendu que lors de la mise hors service de PX (tir de t_4).

Nous verrons au chapitre suivant qu'aucun blocage ou mauvais fonctionnement ne peut être introduit par la synchronisation entre les automatismes PX et AY.

c/ Synchronisation entre AS et les autres automatismes du H8

Nous rencontrons à nouveau une relation "maître-esclave", mais cette fois plus de deux automatismes interviennent. On peut dans ce cas envisager une représentation hiérarchisée, c'est-à-dire considérer la mise hors service des ensembles constitués par des réseaux liés par une synchronisation. On a deux ensembles : (PX - AY) et (AX - A7 - A11) et l'on obtient ainsi le réseau condensé de la figure I.30 où les ensembles (PX - AY) et (AX - A7 - A11) sont considérés comme réseaux esclaves de AS. Du côté maître et du côté esclave les interactions restent celles définies

au paragraphe I.3.5 de ce même chapitre. Le tir de t_3 verrouille les deux ensembles esclaves et ceux-ci sont réactivés après le tir de t_5.

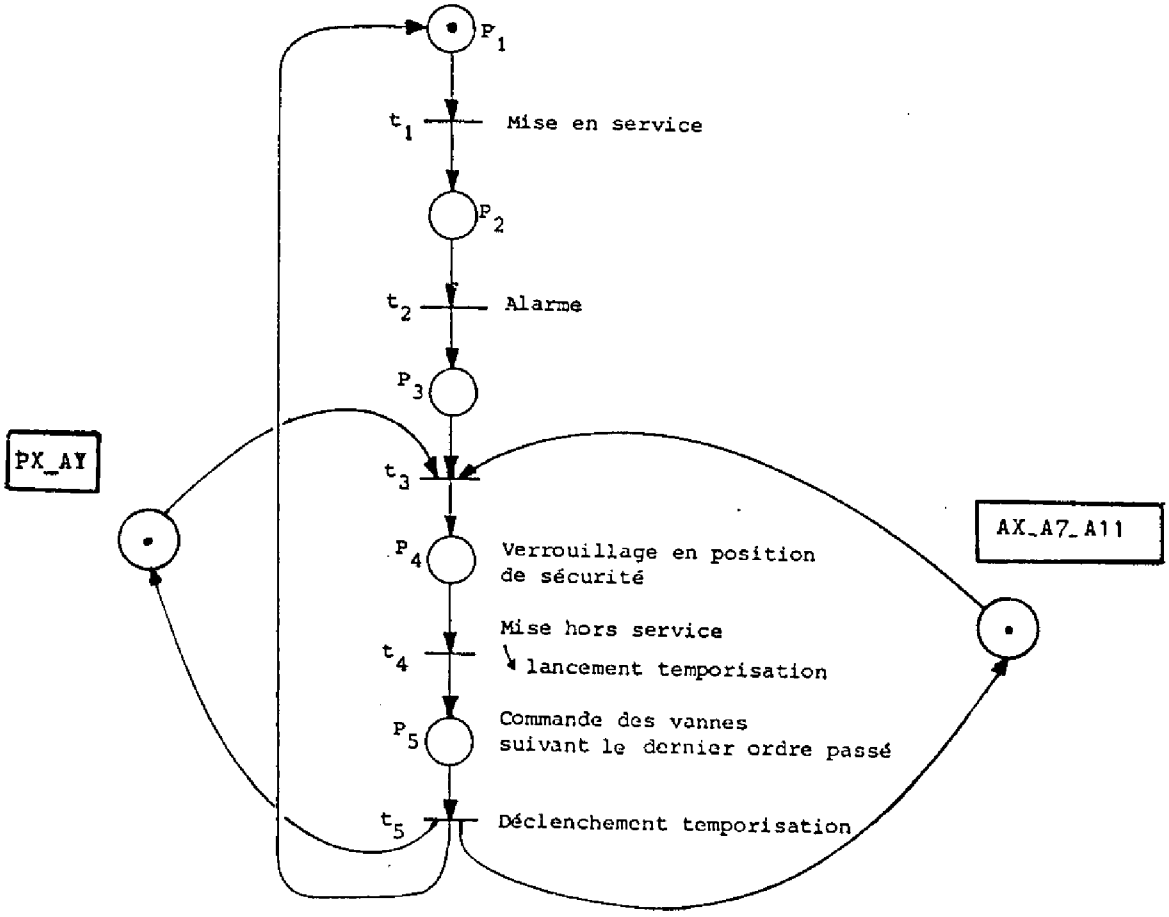


Figure I.30 - Représentation condensée de la synchronisation entre AS et les autres automatismes du H8

I.4.3. Conclusion

Nous avons vu comment les réseaux de Pétri permettent d'aborder plus aisément les problèmes de spécifications des automatismes complexes.

De par leur nature, ils facilitent et imposent une approche structurée obligeant ainsi le concepteur à lever toute ambiguité de spécification.

L'application aux automatismes de la base de lancement ARIANE 2 nous a permis de mettre à l'épreuve cet outil et les possibilités qu'il offre.

De plus, la certification des spécifications peut être poursuivie en utilisant les possibilités d'analyse des réseaux de Pétri. Ceci est l'objet du chapitre suivant.

CHAPITRE II

ANALYSE ET VALIDATION.

II.1. INTRODUCTION

Il est classique de décomposer la vie d'un système en quatre phases [WEI 80] [VAL 80] .

. La spécification
. La mise en oeuvre
. Les tests de conformité
. La vie utile

Il est très important de n'aborder la deuxième phase - la mise en oeuvre - que lorsqu'on est sûr qu'il ne persiste pas d'ambiguité, ni d'imprécision, ni d'erreur dans les spécifications fonctionnelles. Pour cela il faut valider les spécifications. La validation des spécifications fonctionnelles avant l'implémentation permet de diminuer les coûts de réalisation. En effet, la plupart des erreurs de conception ayant été corrigées, les tests de bon fonctionnement se réduisent à la vérification de la conformité de la réalisation vis-à-vis des spécifications.

On peut noter parmi les moyens dont on dispose pour valider les spécifications, les points suivants :

. Validation par dialogue
. Validation par simulation
. Validation par analyse
. Validation par preuve formelle

Les deux derniers points nécessitent un outil de spécification formel. En effet, lorsque le modèle est formel, une analyse de la description peut être suffisamment riche pour permettre de détecter les erreurs de conception sans qu'il soit nécessaire d'effectuer de coûteuses simulations. Nous allons voir les possibilités des réseaux de Pétri dans ce domaine.

Un réseau de Pétri dont on a ôté toutes les étiquettes associées aux transitions et aux places, ne représente plus que la partie commande du système considéré, sans les liens qui l'unissent à la partie données et au monde extérieur.

Si la délimitation entre la commande et les données est bien faite, la partie commande prise isolément doit avoir un sens. La première étape de la validation consistera donc à analyser cette partie commande, c'est-à-dire à rechercher les propriétés du réseau de Pétri non étiqueté. Ainsi les erreurs les plus grossières sont détectées très vite d'une manière systématique.

Ensuite, certaines propriétés spécifiques peuvent être prouvées sur un réseau de Pétri, par exemple que deux places P1 et P2 ne sont jamais marquées simultanément. Ceci permet de prouver que des contraintes spécifiques sont bien vérifiées par le système de commande.

Nous allons dans un premier temps définir les "bonnes propriétés" qu'on est amené à vérifier sur un réseau de Pétri, puis énumérer les méthodes d'analyse permettant d'y arriver.

Seront ensuite proposées des méthodes de validation permettant de contourner les problèmes posés par la communication entre les réseaux et enfin sera abordé le problème de la vérification des contraintes technologiques.

II.2. BONNES PROPRIETES

II.2.1. Définitions

Le réseau de Pétri est donc considéré sans les étiquettes qui lui sont associées et représente ainsi la structure de commande du système. Nous allons définir les "bonnes propriétés" qu'il doit posséder :

a/ Réseaux de Pétri borné - Réseau de Pétri Sauf

Définition : Un réseau de Pétri est dit borné si, quel que soit le marquage accessible et quelle que soit la place considérée, le nombre de jetons contenus dans cette place est inférieur à une borne donnée.

Si cette borne est égale à 1, on dit que le réseau est sauf (binaire)

Remarque : Un système réel étant nécessairement un système borné, il est clair que toute spécification de système implémentable doit avoir un réseau de Pétri borné pour graphe de commande.

b/ Réseau de Pétri Propre (réinitialisable)

Définition : Un réseau de Pétri est propre si pour tout marquage M accessible à partir du marquage initial, il existe une séquence de tir de transitions tirables à partir de M et qui ramène au marquage initial.

Remarque : La plupart des systèmes de commande ont des fonctionnements répétitifs et doivent donc retrouver leur état initial périodiquement. Il est donc clair que les réseaux de Pétri correspondant aux graphes de commande de tels systèmes doivent être propres.

c/ Réseau de Pétri vivant

Définition : Un réseau de Pétri est vivant si pour tout marquage accessible M et pour toute transition T il existe une séquence de tir tirable à partir de M et qui comprend la transition T.

Remarque : Un réseau de Pétri vivant garantit qu'aucun blocage ne sera introduit par la structure de commande du système considéré. Il assure qu'à tout instant il existe toujours une séquence d'évènements permettant d'atteindre une transition donnée et donc qu'aucune partie de la commande ne devient morte.

d/ Résumé

Dans la plupart des automatismes logiques les réseaux de Pétri saufs sont généralement suffisants. Un réseau involontairement non borné est certainement le siège d'une erreur de conception.

Un réseau de Pétri propre correspond à un fonctionnement répétitif qui est généralement désiré dans un automatisme logique.

Un réseau de Pétri vivant est un réseau ne possédant pas de parties mortes (blocage mortel). Dans le cas contraire, une erreur de conception en est sûrement la cause.

Il est donc important de vérifier ces propriétés et nous allons énumérer les différentes méthodes d'analyse qui le permettent.

II.2.2. Méthodes d'analyse

L'analyse des réseaux de Pétri a déjà fait l'objet de nombreux travaux. Une présentation détaillée peut être trouvée dans la littérature [BRA 80, PET 81, BRA 82, VAL 80]. Les principales méthodes d'analyse sont :

- L'analyse par établissement du graphe des marquages accessibles
- L'analyse par réduction
- L'analyse par affinements
- L'analyse structurelle

a/ Analyse par établissement du graphe des marquages accessibles.

Le recherche de l'arbre de couverture des marquages accessibles s'effectue par un algorithme dû à KARP & MILLER [KAR 69] . Il permet de mettre en évidence le fait qu'un réseau est non borné. Les autres propriétés sont ensuite déterminées sur le graphe des marquages accessibles dans le cas où le réseau est borné.

b/ Analyse par réduction

Cette méthode est due à BERTHELOT [BER 77] [BER 78]. Elle consiste à appliquer des règles de réduction (du graphe) qui conservent les "bonnes propriétés". Celles-ci sont alors plus facilement déductibles sur un réseau simplifié. En effet, la taille du graphe des marquages accessibles est fortement réduite.

c/ Analyse par affinements

Issue des règles de programmation structurée, c'est à la fois une méthode d'analyse et de conception proposée par VALETTE [VAL 76]. En concevant un système de façon descendante, en s'imposant de n'utiliser que certaines classes de réseaux, l'analyse est extrêmement simplifiée.

d/ Analyse structurelle

L'analyse structurelle, due à LAUTENBACH [LAU 74], consiste à chercher des invariantes linéaires de places et de transitions. Cette

recherche s'effectue indépendamment du marquage initial et utilise la programmation linéaire en nombres entiers. La valeur du marquage initial est introduite ensuite.

- Les invariants linéaires de places sont particulièrement utiles pour montrer des propriétés spécifiques. Par exemple dans le cas où les actions sont associées aux places, on peut dans la plupart des cas montrer que des actions contradictoires portant sur un matériel donné sont exclues (sans avoir à énumérer les marquages accessibles). En effet, les invariants linéaires de places suffisent en général pour montrer que dans un certain ensemble de places, une seule d'entre elles peut posséder un (ou plusieurs) jeton(s) à un instant donné (les autres étant vides).

- Les invariants linéaires de transitions correspondent à des séquences cycliques de tir de transitions (le marquage final est égal au marquage de départ). Ce type d'invariant est très utile pour montrer des propriétés spécifiques faisant intervenir des séquences de tir.

Ces méthodes d'analyse ont été implémentées sur calculateur et à l'heure actuelle le laboratoire dispose du système OGIVE [PRA 79] qui, outre les possibilités d'analyse, propose une description graphique du réseau de Pétri.

II.2.3. Exemple

Considérons les automatismes de sécurité de la base de lancement ARIANE 2. L'automatisme A7 ayant été spécifié en deux parties, nous ne le considèrerons que dans le paragraphe suivant. Par contre tous les autres ont été prouvés saufs, vivants et propres. Il faut souligner d'une part, que ces preuves ont été faites par réduction et analyse structurelle sans jamais avoir à énumérer tous les marquages accessibles. D'autre part, une approche descendante ayant été entreprise, la preuve que les réseaux détaillés correspondant aux automatismes A7, A11 et AX et que le réseau global correspondant à leur synchronisation possèdent les bonnes propriétés implique que leur coopération est correcte. La construction du réseau global détaillé, et à fortiori de son graphe des marquages, n'est pas nécessaire. Le cas des coopérations maître-esclave est traité au paragraphe suivant.

II.3. PROBLEME POSES PAR LA COMMUNICATION ENTRE LES RESEAUX

Nous avons vu (cf.I.3) que lors de la description d'un automatisme complexe, uns spécification bien structurée se traduisait par une décomposition du système et par conséquent par des descriptions partielles. Chaque automatisme est ainsi pris séparément et sa structure est clairement définie.

Une bonne validation doit prendre en considération non seulement la structure de chaque automatisme mais également l'interaction entre ces automatismes.

II.3.1. Preuve par fusion de réseaux

Lorsque les communications entre les automatismes sont du type envoi de message asynchrone ou "rendez-vous" (cf.I.3.4), nous avons vu que cela se traduisait par l'utilisation de places ou de transitions communes (de même nom). Dans ce cas la preuve de l'absence d'interblocage peut pratiquement être automatique car le réseau global représentant la communication peut être obtenu directement par calculateur en fusionnant les places et les transitions de même nom. Il suffit alors d'analyser ce réseau global (à condition bien sûr qu'il reste de taille raisonnable).

Un exemple d'une telle approche est donné par la spécification de l'automatisme A7 (cf I.4.2.) qui se compose en fait de deux automatismes : un automatisme de surveillance de la pressurisation du côté oxygène et un automatisme de surveillance de la pressurisation du côté hydrogène. Ces deux automatismes ont été spécifiés séparément figures I.24 et I.25 . Les liens entre ces deux automatismes ont été définis et concernent en particulier la manipulation du clapet 31 ainsi que les mises en service et hors service de ces automatismes.

Le réseau résultant de la composition des réseaux des figures I.24 et I.25 et représentant la synchronisation globale de l'automatisme A7 a été obtenu et analysé par le système OGIVE : Ce réseau est sauf, propre et vivant, ce qui prouve que la communication entre les parties oxygène et hydrogène de l'automatisme A7 est structurellement correcte. En particulier aucun interblocage ne peut avoir lieu.

Lors de la fusion des réseaux, les fusions multiples (fusion d'une ou plusieurs transitions d'un réseau avec plusieurs transitions d'un autre réseau) doivent être effectuées avec précautions. Ainsi lors de la fusion concernant la mise hors service de l'automatisme A7, la transition t_{11} côté hydrogène (H2) n'a pas été fusionnée avec les transitions t_4 et t_8 côté oxygène (O2). En effet, lorsque la transition t_{11}(H2) est sensibilisée, il va de soi que la transition t_{10} (H2) a été tirée (verrouillage de O2) et que la fusion des transitions ayant l'étiquette "V" a provoqué du côté oxygène la prise du jeton dans les places P2 (O2) ou P3 (O2) (tir de t_6 ou t_9 côté oxygène), celui-ci se trouve alors dans la place P_4 (O2).Ainsi les transitions t_4 et t_8 de O2 et t_{11} (H2) ne peuvent jamais être sensibilisées simultanément. Par conséquent les transitions résultant de la fusion de t_4 (O2) et t_{11} (H2) ainsi que t_8 (O2) et t_{11} (H2) ne peuvent jamais être tirées et le réseau global résultant de la fusion n'est donc pas vivant.

Cet exemple montre qu'il est souvent difficile de prévoir toutes les conséquences d'une fusion multiple automatique et de les maîtriser du premier coup. C'est une des raisons pour lesquelles nous ne proposerons pas de fusion multiple automatique au niveau du langage de spécifications (cf. Chapitre III) mais uniquement des fusions dans le cas des occurences simples.

II.3.2. Preuve par équivalence : relation maître-esclave

Dans le cas d'automatismes pouvant en verrouiller d'autres, nous avons vu comment l'on pouvait se ramener à une relation "maître-esclave" et représenter cette relation de façon très compacte en remplaçant le réseau de l'automate esclave par une place initialement marquée (cf.I.3.5). Nous allons montrer que cette représentation permet d'analyser la synchronisation de l'ensemble "maître-esclave" sans avoir à détailler les mécanismes précis autorisant à l'automate maître de forcer l'automate esclave dans un état "hors-service" à n'importe quel instant.

Cette analyse peut en effet être déduite de l'analyse séparée du réseau "esclave" seul et de celle de l'ensemble "réseau maître-place représentant le réseau esclave", à condition toutefois que le réseau esclave soit propre (réinitialisable).

Effectivement, l'influence du réseau maître sur le réseau esclave se résume en un passage d'un marquage accessible quelconque au marquage initial.

Si le réseau représentant l'automate esclave est propre, on ne change pas le graphe des marquages accessibles et donc on n'altère pas les propriétés du réseau (si ce n'est au niveau des séquences qui peuvent être raccourcies).

Pour le réseau maître, aucun autre influence n'est introduite que celle modèlisée par la place représentant l'automate esclave qui exprime que la seule condition pour que l'on puisse forcer l'esclave lors service, c'est qu'il ne soit pas déjà dans cet état.

Ce résultat permet de compléter la validation de la coopération entre les automatismes de sécurité spécifiés au chapitre I. En effet tous les réseaux ayant été prouvés saufs, vivants et propres, aucun problème ne peut être introduit par les relations maître-esclave.

De plus, comme nous allons le voir, il est même possible d'obtenir des invariants reliant les places de deux réseaux.

Ces invariants sont importants car ils aident à prouver le bon fonctionnement de la synchronisation entre les automates. Ils peuvent être nécessaires en particulier pour mettre en évidence l'absence d'ordres contradictoires. En effet, considérons les invariants du réseau maître en faisant intervenir la place P_E symbolisant l'automate esclave. Si le marquage de cette place est nul, cela implique que toutes les places du réseau esclave sont vides sinon, puisque cette place est binaire, les places du réseau esclave vérifient les invariants du réseau esclave.

Il suffit donc de remplacer M(PE) (marquage de PE) par les deux valeurs (0 et 1) successivement dans les invariants du réseau maître pour obtenir deux ensembles exclusifs d'invariants liant les places des deux réseaux. Prenons par exemple la synchronisation entre PX et AY (voir figure I.29). Les invariants linéaires de places du réseau-maître donnent les relations suivantes :

$$\begin{cases} M[P1\ (PX)] + M[P2\ (PX)] + M[P3\ (PX)] + M[P4\ (PX)] = 1 & (1) \\ M[P4\ (PX)] + M[P_E\ (AY)] = 1 & (2) \end{cases}$$

L'invariant de place du réseau esclave AY dont la structure est similaire à celle de l'automatisme AX (Figure I.16)donne la relation suivante :

$$\sum_{i=1}^{6} M\left[Pi\ (AY)\right] = 1. \qquad (3)$$

. 1er cas : $M\left[P_E\ (AY)\right] = 1.$

Les relations (2) (1) (3) impliquent :

$$\begin{cases} . M\left[P_4\ (PX)\right] = 0 \text{ et} \\ M\left[P1\ (PX)\right] + M\left[P2\ (PX)\right] + M\left[P_3\ (PX)\right] = 1 \\ . \sum_{i=1}^{6} M\left[Pi\ (AY)\right] = 1. \end{cases}$$

. 2ème cas : $M\left[P_E\ (AY)\right] = 0$

Les relations (2) (1) (3) impliquent :

$$\begin{cases} . M\left[P_4\ (PX)\right] = 1 \text{ et} \begin{cases} M\left[P_1\ (PX)\right] = 0 \\ M\left[P_2\ (PX)\right] = 0 \\ M\left[P_3\ (PX)\right] = 0 \end{cases} \\ . \forall i,\ M\left[P_i\ (AY)\right] = 0 \end{cases}$$

En particulier ont peut déduire de ces deux ensembles que la relation suivante est toujours vérifiée :

$$\sum_{i=1}^{6} M\left[(Pi\ (AY)\ )\right] + M\left[P_4\ (PX)\right] = 1$$

Nous utiliserons cet invariant au paragraphe suivant. Néanmoins nous pouvons remarquer qu'il implique que une fois mis hors service, l'automatisme AY ne peut être remis en service tant que l'automatisme PX n'aura pas lui-même été mis hors service. C'est-à-dire que PX non seulement met AY hors service mais également le verrouille dans cette position. Ceci n'était pas explicité dans les spécifications en langage naturel [X 78].

II.3.3. Preuve dans le cas d'une communication par variables auxiliaires

Certaines procédures de communication entre automatismes peuvent ne pas pouvoir s'exprimer simplement ni par des fusions de places, ni par des fusions de transitions, ni par des relations maître-esclave. On est alors amené à jouer sur l'interprétation des réseaux de Pétri, c'est-à-dire à introduire des variables auxiliaires testées et modifiées par les divers automatismes. La validation de la communication se ramène alors à celle d'un réseau de Pétri interprété.

Malheureusement les résultats obtenus sur le réseau de Pétri non étiqueté ne restent pas toujours valables lorsque l'interprétation est prise en compte. En effet, même dans le cas où le nombre de marquages accessibles - en tenant compte de l'interprétation - est fini, le nombre d'états accessibles peut être infini (un état d'un réseau de Pétri interprété est formé d'un marquage et de l'état des variables auxiliaires de synchronisation).

Une démarche possible consiste à baser la preuve sur l'utilisation d'invariants. Un invariant consiste alors en une hypothèse faite sur le contenu en jetons des places et sur les variables auxiliaires. Cet invariant doit être vérifié pour tous les états du système.

Les invariants linéaires de places pouvant être obtenus directement à partir de la structure du réseau de Pétri, il suffit donc de compléter ces invariants par d'autres qui font également intervenir les variables auxiliaires.

III.4. VERIFICATION DES CONTRAINTES TECHNOLOGIQUES

Après avoir vérifié les bonnes propriétés de la structure de commande et montré par exemple qu'elle était sans blocage, il est nécessaire de vérifier certaines contraintes spécifiques du système considéré, par exemple qu'aucun des enchaînements de tâches spécifiés n'est contradictoire avec le matériel utilisé.

II.4.1. Absence d'ordres contradictoires

On ne peut tolérer qu'un système de commande puisse envoyer deux ordres contradictoires au même moment que le même matériel (par exemple commandes simultanées d'ouverture et de fermeture d'une même vanne). Afin d'éviter ce genre d'incohérence, une démarche possible est la suivante :

- chercher l'ensemble des places concernées par une commande
- trouver un invariant (ou un ensemble d'invariants) concernant ces places tel que l'on puisse prouver que si une de ces places possède un jeton, toutes les autres sont vides.
- si la recherche d'invariant ne permet pas de conclure, ou bien on énumère les marquages pour prouver qu'une seule place concernée par la commande est marquée, ou bien on reprend la spécification car cela veut dire que la structure du réseau n'implique pas à elle seule l'absence d'ordres contradictoires comme dans l'exemple suivant :

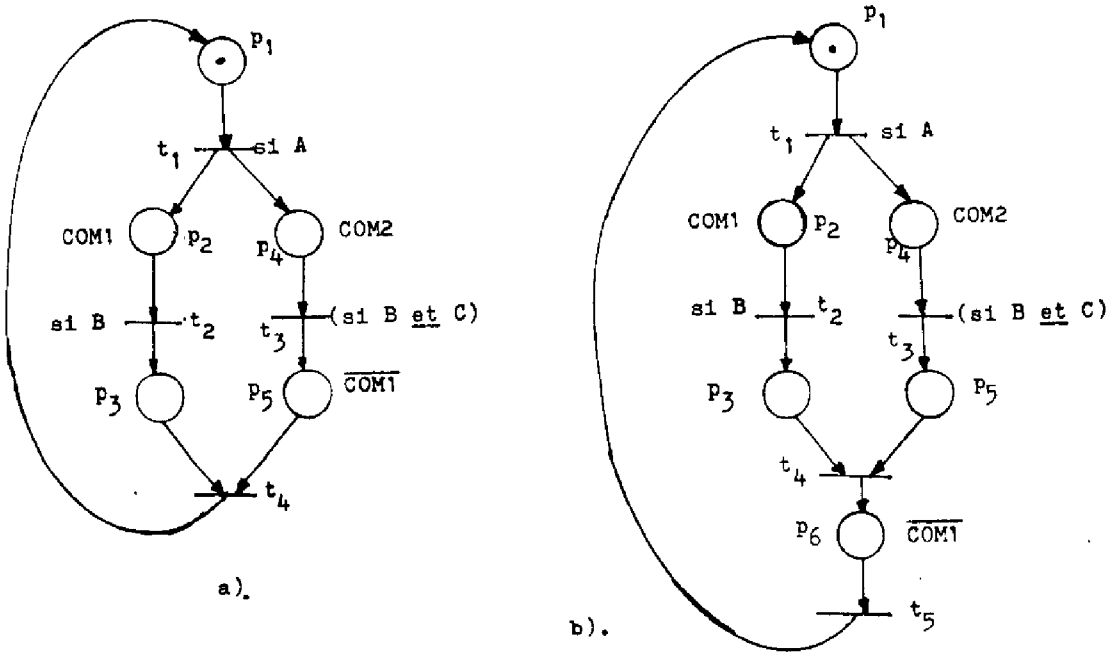


Figure II.1

La structure du réseau de la figure II.1.a est dangereuse car un changement mineur de réceptivité peut introduire l'apparaition d'ordres contradictoires, en effet il suffirait que la condition associée à t_3 devienne "(si B ou C)" pour que l'occurence de l'évènement"C vrai et B faux"entraîne le marquage simultané des places P_2 et P_5 auxquelles sont associées respectivement les actions COM_1 et $\overline{COM_1}$, qui sont contradictoires. On peut noter que même sans cette modification, la commande contradictoire peut être émise pendant un temps très court si t_3 est tirée avant t_2.

Par contre, la spécification donnée par le réseau de la figure II.1.b assure une meilleure sécurité car il y a un invariant linéaire entre les places P_2 et P_6 : $M(P_2) + M(P_6) = 1$, ainsi les ordres COM1 et $\overline{COM1}$ ne seront jamais exécutés simultanément.

Cet exemple fait apparaître la notion de robustesse vis-à-vis des modifications. Sur le réseau de la figure II.1.b, quelles que soient les modifications sur les réceptivités, l'absence d'ordres contradictoires est vérifié (robustesse de la spécification). Ce n'est pas le cas pour le réseau de Pétri de la figure II.1.a.

Reprenons maintenant les automatismes de sécurité de la base de lancement ARIANE 2 spécifiés au paragraphe I.4 pour illustrer notre démarche.

Considérons d'abord l'automatisme AX spécifié par le réseau de la figure I.16., dans ce réseau les ordres 5,6,7,8 manipulent les mêmes vannes (XVP107, XVP 108, HVP 113) soit en fermeture, soit en ouverture. Mais du fait qu'il n'existe aucun parallèlisme interne, on a directement l'invariant linéaire de place suivant : $\sum_{i=1}^{6} M[(Pi)] = 1$, c'est-à-dire qu'à tout moment une place et une seule est marquée et par conséquent deux ordres contradictoires ne peuvent être envoyés simultanément sur ces vannes.

Un cas moins évident concerne les automatismes PX et AY (similaire à AX). En effet, ils travaillent sur des vannes communes (HVP 513, YVP 514, YVP 515). Mais ces deux automatismes sont en relation maître-esclave (voir figure I.29). Nous avons vu au paragraphe précédent comment on peut trouver l'invariant nécessaire à la preuve d'ordres contradictoires, c'est-à-dire un invariant liant la place P_4 de P_X aux places de AY.

En effet, la relation :

$$\sum_{i=1}^{6} M\left[P_i\ (AY)\right] + M\left[P_4\ (PX)\right] = 1$$

implique que si la place P_4 (PX) possède un jeton, toutes les places de AY sont vides. De plus, la place P_4 (PX) est la seule place de PX agissant sur des vannes communes à AY. En conséquence il ne peut y avoir d'ordres contradictoires ou redondants à l'intérieur de l'ensemble formé par les automatismes PX et AY.

Remarque : Le fait que deux automatismes soient en relation maître-esclave n'exclut pas la possibilité d'ordres contradictoires. Prenons par exemple la spécification en langage naturel de l'automatisme PX (cf. I.4) : Le réseau de Pétri de la figure II.2 satisfait les exigences de cette spécification.

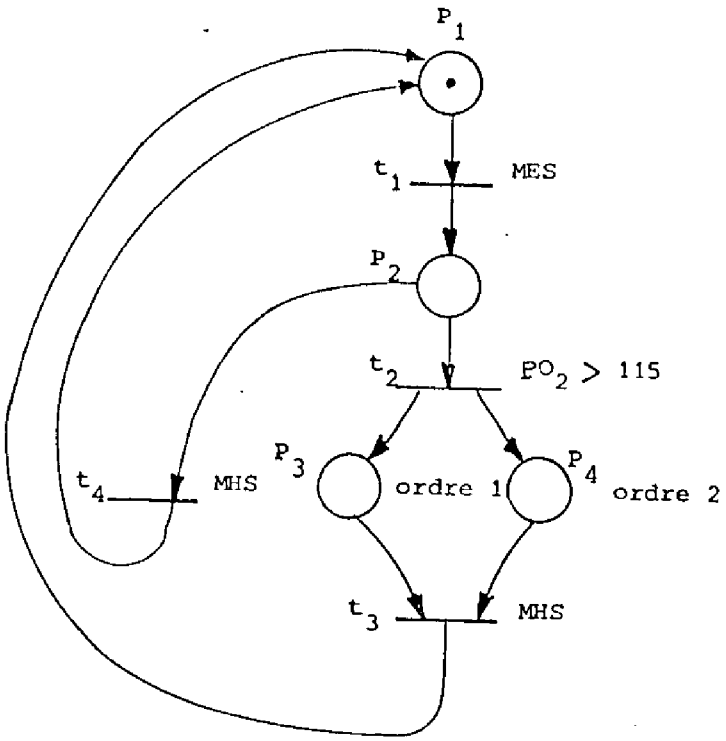


Figure II.2

Si l'on veut représenter sa communication avec l'automatisme AY (Relation maître-esclave) on aboutit au réseau de Pétri de la figure II.3.

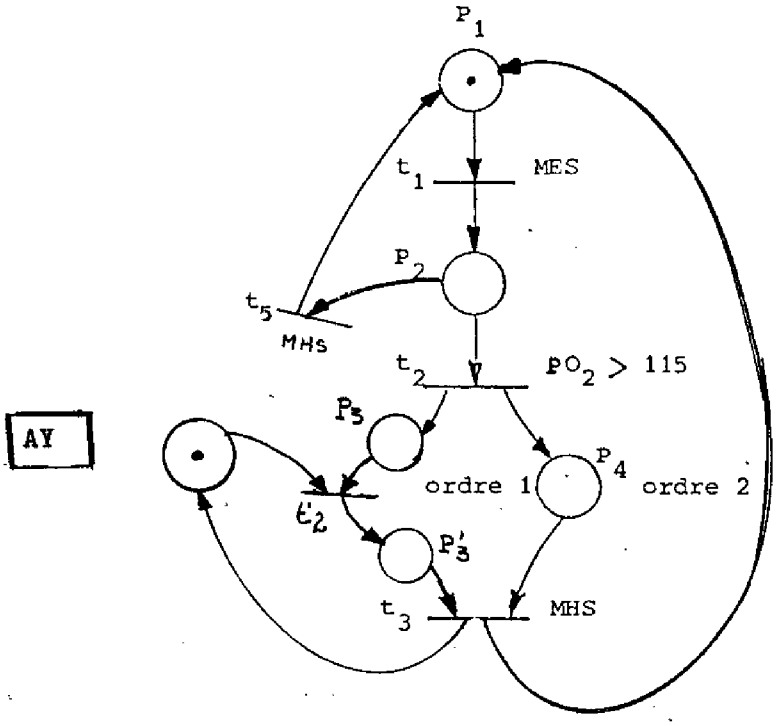


Figure II.3

Il n'existe aucun invariant linéaire entre l'ordre 2 (isoler le réservoir H2) et la mise hors service de l'automatisme AY. Par contre le marquage simultané des places P_4 et P_3 apparaîtra certainement, même si c'est pendant un temps très court. Suivant l'état interne de l'automatisme AY, des ordres contradictoires pourront alors être émis.

II.4.2. Absence d'ordres intempestifs

Un autre danger apparaît lorsqu'il existe dans le réseau de Pétri des chemins tels que toutes les réceptivités (conditions logiques) sont vraies à un instant donné. En effet, s'il arrive que, à un instant donné, les réceptivités de deux transitions se suivant soient vraies, ces dernières seront tirées à des temps proches et le jeton restera pendant un temps court mais non négligeable dans la place intermédiaire. Si une action est associée

à cette place, et n'est associée ni à la place précédente, ni à la place suivante, elle sera exécutée pendant un temps court. Ainsi des actions peuvent être abandonnées avant d'avoir été complètement exécutées.

Un tel phénomène crée d'abord des ambiguités de spécification car on ne sait si une telle action appelée action intempestive sera prise en compte ou non car cela fait intervenir des considérations technologiques de bas niveau qui ne doivent pas apparaître lors d'une spécification fonctionnelle correcte. D'autre part, les actions intempestives peuvent être dommageables pour le matériel, et dans les systèmes comprenant des mécanismes d'auto-surveillance elles peuvent provoquer des signalements de fautes et des reconfigurations inutiles et dangereuses. Par ailleurs, si un chemin bouclé comprenant de telles actions existe, le défaut de fonctionnement est catastrophique car l'automatisme risque d'exécuter indéfiniment la séquence correspondante, bloquant toute autre évolution si aucune procédure de sécurité n'a été prévue.

Une spécification correcte ne doit donc comprendre aucune possibilité d'ordre intempestif. Cette vérification se fait pas à pas, au niveau des spécifications détaillées ou même des spécifications d'implémentation en analysant les réceptivités une à une.

Par exemple, dans le cas du réseau de Pétri de la figure I.15 représentant l'automatisme AX, des chemins pouvant être parcourus de manière quasi instantanée existent :

- le chemin t1, P2, t2, P3, t3, P4, t4, P5, si lors de la mise en service la pression P est supérieure au seuil S4,
- les chemins comprenant la transition t_{12} (aucune étiquette n'y étant associée, l'ordre 7 (place P6) risque d'être interrompu avant d'avoir été exécuté).

Toutefois, aucun de ces chemins ne forme un circuit. Si les ordres intempestifs ne sont pas tolérables, une autre spécification de cet automatisme est nécessaire et c'est ainsi qu'a été proposé le réseau de Pétri de la figure I.16 . On peut montrer qu'aucun ordre intempestif n'y apparaît. En effet, considérons par exemple la place P_3, lorsqu'elle reçoit un jeton (tir de t'1, t2 ou t6) on a $S1 < P < S3$ et donc ni t3 ni t5 ne seront tirables. On peut raisonner de manière analogue pour les autres places.

En ce qui concerne l'ordre associée à la place P6, si l'on tient compte de la spécification proposée dans la figure I.17, la transition t_{12} ne sera tirée qu'au bout d'un temps donné, ainsi l'ordre 7 pourra être complètement exécuté.

II.5. CONCLUSION

Nous avons vu les propriétés que doit posséder une structure de commande représentée par réseau de Pétri pour être correcte (vivant, borné, propre). Les méthodes d'analyse de ces propriétés ont été énumérées. Nous avons ensuite vu comment assurer une bonne synchronisation entre les divers sous-sytèmes en enfin comment prouver des propriétés spécifiques au système décrit et vérifier des contraintes technologiques.

Le but du chapitre suivant est de proposer un langage haut niveau pour la description sur micro-calculateur de réseau de Pétri à transmettre à un automate programmable après avoir été analysé et trouvé correct. Le module "Analyse" du réseau de Pétri sur micro-calculateur est en cours d'élaboration dans l'équipe PASTELS.

CHAPITRE III

UN LANGAGE DE SPECIFICATION

III.1. INTRODUCTION

Nous proposons dans ce chapitre un langage de description des réseaux de Pétri auquel est associé un analyseur syntaxique et sémantique.

Ce langage a été conçu comme langage d'entrée d'un automate programmable permettant de passer directement de la spécification structurée sous la formed'un ensemble de réseaux de Pétri interprétés communicants à une mise en oeuvre. La programmation ne se fera pas directement sur l'automate programmable, mais par l'intermédiaire d'une console de programmation. Cette console peut être un micro-système de développement (dans notre cas, un un micro-système bâti autour d'un Z.80). Cela donne suffisamment de puissance pour supporter un langage haut niveau tout en restant d'un coût peu élevé. Une fois la description analysée, la console de programmation télécharge l'automate de tables représentant les réseaux. L'automate émule alors le fonctionnement des réseaux en temps réel suivant les valeurs des entrées.

Dans un premier temps sont exprimées les raisons qui nous ont poussés à élaborer un langage de description de réseaux de Pétri. La structure du langage est ensuite présentée, puis vient la description du langage, les possibilités de représentation de communication entre réseaux sont alors mises en valeur.

Vient ensuite la description de l'analyseur syntaxique et sémantique. Les différentes phases de l'analyse du réseau de Pétri décrit sont explicitées. Un exemple d'application est présenté (cf. Annexe 3).

III.2. POURQUOI UN LANGAGE

Les méthodes de description des réseaux de Pétri à l'aide de calculateur se font soit par :

a/ Entrée graphique (exemple OGIVE) [PRA 79]

Ce mode de description est séduisant car le réseau de Pétri est avant tout un graphe et il est appréciable de le visualiser.

La description se fait généralement de manière assistée. Le Calculateur dans les versions simples impose le nom des noeuds, ainsi que la manière de tracer les arcs.

Au delà d'un certain nombre d'arcs, le réseau peut devenir illisible s'il n'y a pas la possibilité de maîtriser le tracé des arcs sur l'écran (afin d'éviter que plusieurs arcs se croisent). On peut toutefois y remédier à l'aide d'une composition de réseaux et de "zooms" (possibilité d'isoler une partie du réseau).

Nous voyons donc que pour qu'un système graphique soit réellement pratique, il doit offrir les possibilités suivantes :

. Laisser à l'opérateur le choix du nom des noeuds,
. La possibilité de définir le tracé des arcs,
. La possibilité de "zoom",
. La possibilité de composition de réseaux,

Le support de logiciel nécessaire pour un tel système doit être très puissant. A l'heure actuelle une implémentation sur microprocesseur est exclue, ce qui est contradictoire avec le projet d'un automate programmable dont la console de programmation resterait un microprocesseur simple.

b/ Entrée conversationnelle (non graphique OGIVE, SIRENA) [PRA 79, SIR 82]

La description consiste dans ce cas à répondre à des questions posées par le calculateur. Ce mode de description a l'avantage d'être très facile d'accès et par conséquent bien adapté à l'enseignement puisque l'apprentissage est très aisé.

L'inconvénient majeur est qu'en cas d'erreur de frappe ou d'oubli, il faut soit recommencer la description, soit continuer jusqu'à ce que le calculateur termine la description et ensuite demander à rentrer dans un module de modifications (suppression ou ajout d'arcs ou de noeuds par exemple).

Cette méthode peut devenir longue et fastidieuse pour des systèmes de grande taille car il faut se souvenir des erreurs pour les corriger en bloc après la description. Toute faute de frappe devient vite une catastrophe. D'autre part, l'opérateur entraîné peut trouver lassante la répétition des questions et juger l'ordinateur un peu trop bavard.

c/ Langage

Une troisième approche consiste à découpler les problèmes d'édition des problèmes de vérification de la cohérence de la description. Un langage de description clair et de haut niveau doit être défini. A l'aide d'un éditeur de texte standard ayant toute la puissance et la souplesse nécessaire, l'ensemble des réseaux de Pétri interprétés correspondant à la spécification est décrit et commenté. Ensuite un programme d'analyse syntaxique et sémantique est exécuté et une liste d'erreurs est produite. Les corrections éventuelles se font sous l'éditeur, facilement et sans ambiguité.
En résumé les avantages d'une telle démarche sont les suivants :

- Edition très aisée avec possibilité de mise en page,
- Analyse performante puisque le programme l'effectuant est déchargé de tout problème d'édition,
- Correction et modification facilitée par l'édition standard déchargé de tout problème d'analyse syntaxique,
- Listage clair de la description finale du système grâce à l'utilisation d'un langage haut niveau autorisant l'usage des commentaires,
- Stockage direct du listage en clair sur disquette.

III.3. STRUCTURE DU LANGAGE

III.3.1. Structure générale

Une analyse sémantique ne peut se faire que si le langage est fortement structuré et utilise la notion de type.

La description du réseau de Pétri interprété consistera en une suite de déclarations non procédurales.

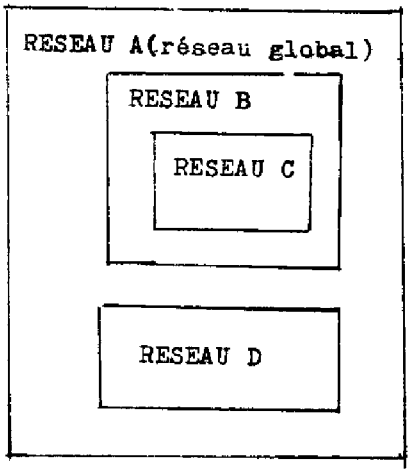
III.3.2. Structure de commande

La description du réseau de Pétri se fait suivant une structure de bloc, ce qui permet la spécification de gros réseaux en plusieurs morceaux : le réseau principal et un ou plusieurs sous-réseaux pouvant être inclus les uns dans les autres ou totalement indépendants.

Des réseaux peuvent être ainsi déclarés à l'intérieur d'autres réseaux. Les noms des noeuds, des variables d'interprétation et des sous-

réseaux sont dits locaux au réseau dans lequel ils sont déclarés, ce qui veut dire qu'ils n'ont de signification qu'au sein du morceau de programme que constitue le réseau et qui est appelé la portée (ou domaine de validité) de ces noms. Puisque les réseaux peuvent être imbriqués, les portées peuvent l'être aussi. Les objets qui sont déclarés dans le réseau principal et qui ne sont donc locaux à aucun sous-réseaux sont dits globaux et ont une signification dans tout le programme constitué par le réseau global du système.

La figure III.1 illustre cette structure de bloc :



les objets définis dans le bloc(réseau)	sont accessibles dans le bloc(réseau)
A	A , B , C , D
B	B , C
C	C
D	D

Figure III.1 - Structure de bloc

L'application à l'ensemble des automatismes de sécurité de la base de lancement ARIANE 2 (cf. I.4) est donnée par la figure II.2

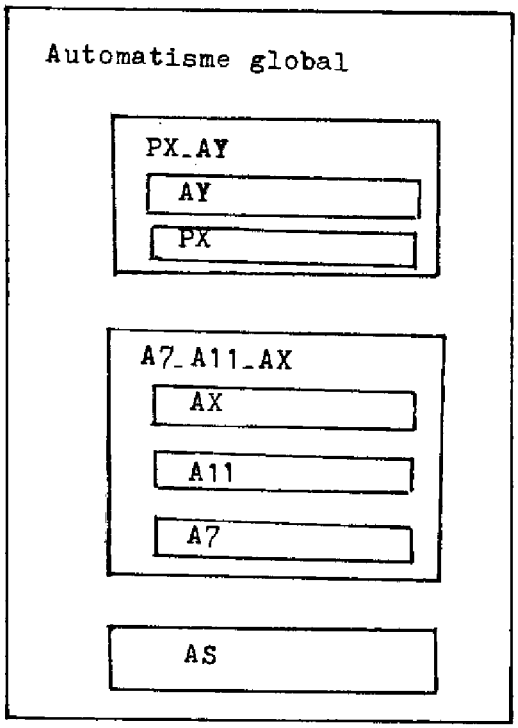


Figure III.2 - Structure de l'ensemble des automatismes du H8

Le réseau de Pétri interprété est décrit sous forme d'instructions composées (ou modules), cette notion est différente de celle utilisée en PASCAL car il est strictement interdit d'inclure une instruction composée dans une autre, mais les instructions élémentaires sont analogues aux instructions de déclaration en langage PASCAL.

Il n'y a pas de fermeture explicite d'un module (instruction composée) : un mot réservé de début d'instruction composée entraîne automatiquement la fermeture de l'instruction composée précédente.

Les instructions composées sont au nombre de quatre (chacune pouvant être absente ou vide ou multiple suivant le désir de l'opérateur) :

. 'NOEUDS' : module de déclaration des noeuds (places, marquage, transitions)
. 'ARCS' : module de déclaration des arcs du réseau (et du poids qui leur est associé)
. 'VAR' : module de déclaration des variables d'interprétation.
. 'INTERPRETATION' : module de déclaration des conditions logiques et prédicats associées aux transitions et des actions associées aux places et aux transitions.

Chaque module est composé d'instructions de déclarations, celles-ci n'étant soumises à aucune contrainte procédurière ; par exemple dans le module NOEUDS, l'ordre dans lequel sont déclarées les places et/ou les transitions n'a aucune importance.

La fin d'une instruction de déclaration est marquée par un point virgule qui est utilisé ici comme terminateur d'instruction et non comme séparateur d'instruction comme en langage PASCAL.

Il n'y a pas d'ordre imposé par la syntaxe entre les différentes instructions composées, celles-ci pouvant apparaître plusieurs fois dans la description d'un réseau. Toutefois l'ordre NOEUDS, ARCS, VAR, INTERPRETATION est l'ordre logique normal car les arcs ne peuvent être définis qu'après les noeuds et l'interprétation après les déclarations de noeuds et de variables.

Ceci permet une structuration plus fine d'un réseau (regroupements des noeuds en diverses instructions composées par exemple). La description suivante où un réseau global comprend un sous-réseau local en est une illustration :

DEBUT *RESEAU* *GLOBAL*

NOEUDS { déclaration des seuls noeuds de communication entre GLOBAL et LOCAL }

DEBUT *RESEAU* *LOCAL*

{ description réseau LOCAL }

FIN *RESEAU* *LOCAL*

NOEUDS { noeuds propres à GLOBAL }

FIN *RESEAU* *GLOBAL*

III.3.3. Structuration des données

Le langage utilise abondamment la notion de type. Tout identificateur doit avoir été déclaré avant d'être utilisé par exemple tout noeud du réseau apparaissant dans une déclaration d'arcs et toute variable d'interprétation (ainsi que le noeud auquel elle est associée) doivent au préalable avoir été déclarés dans leur module de déclaration respectif (NOEUDS et ARCS) avec leur type.

Ainsi sont détectées (et donc évitées par la suite) des erreurs grossières de description généralement révélées par suite de conflit de type, par exemple un arc reliant deux places, une variable déclarée comme comme entrée, puis redéclarée comme sortie, opération arithmétique sur des variables déclarées booléennes, etc...

Les différents types sont :

P : Place
T : Transition
B : Booléen
E : Entier
S : Symbolique(fontion pour booléens sont à définir des sous-expressions booléennes).

De plus, pour les types "booléen" et "entier", quatre sous-types ont été définis :

- Entrée (variable d'entrée de l'automate, c'est-à-dire sortie des capteurs)
- Sortie (actionneurs du système commandé)
- Temporisations (variables booléennes dépendant du temps)
- Internes (lorsque ce sont des variables auxiliaires utilisés dans l'interprétation)

Les sous-types sont également déclarés explicitement sauf le sous-type variable interne qui est pris par défaut.

III.4. DESCRIPTION DU LANGAGE

III.4.1. Notations et vocabulaire

Le vocabulaire de base du langage est constitué d'un certain nombre de symboles : lettres, chiffres et symboles spéciaux.

. Une lettre est un élément de l'ensemble formé par tous les caractères alphabétiques majuscules et minuscules plus un ensemble de caractères spéciaux propres au langage :

$$\{(A \rightarrow Z\ ,\ a \rightarrow z) + (@, [,], \backslash, \uparrow, -)\}$$

Définition : nous appellerons ASCII (64,122) tout élément de l'ensemble comprenant des caractères alphabétiques (majuscules et minuscules) et les symboles spécifiés ci-dessus.

. Un chiffre est un élément de l'ensemble 0,1,2,3,4,5,6,7,8,9

. L'ensemble des symboles spéciaux peut être décomposé en deux parties :

1°) Opérateurs et délimiteurs

+	,	'	<=
-	;	=	>=
*	:	<>	)
:=	[	<	(
.	]	>	)

2°) Mots clés ou mots réservés

ACTION	CONST	FAUX	OU	SORTIE
ARCS	DEBUT	FIN	PAS	TEMPORISATION
BOOLEEN	ENTIER	INTERPRETATION	PLACE	TRANSITION
CHEMIN	ENTREE	LANCER	RAZ	VAR
COND	ET	NOEUDS	RESEAU	VRAI

Symboles à signification prédéfinie, ils (les mots réservés) ne peuvent en aucun cas être utilisés pour désigner autre chose que ce pour quoi ils ont été définis.

Ils sont formés d'une suite de caractères, sans qu'aucun symbole particulier ne les distingue. Dans le cadre de ce mémoire, ils sont le plus souvent soulignés dans les descriptions écrites à la main afin de mettre en valeur leur rôle de symbole à signification prédéfinie.

Commentaire : Le langage permet l'insertion d'un commentaire en tout point de la description entre deux identificateurs, nombres ou symboles spéciaux, ceci ne modifiant en rien la signification du programme.

Un commentaire est respectivement ouvert et fermé par les symboles "{" et "}" et contient n'importe quelle suite de caractères autre que les accolades et le symbole ";".

exemples : {ceci est un commentaire correct}
{ceci n' est pas un ; commentaire correct}

Identificateur : Tout élément apparaissant dans la description du réseau (types de données, variables) est décrit à l'aide d'un identificateur. Celui-ci commence par un ASCII (64,122) qui peut être suivi par un nombre quelconque d'ASCII (64,122) ou de chiffres. Le diagramme de la figure III.3. en représente la syntaxe.

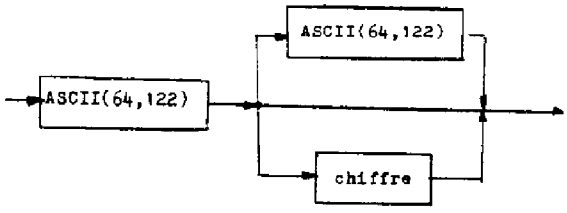


Figure III.3 - Identificateur

Remarque : Bien qu'un identificateur puisse être de longueur quelconque, l'analyseur syntaxique ne prend en compte que les huit premiers caractères.

Séparateurs : Les blancs, les fins de lignes et les commentaires sont considérés comme des séparateurs. Il peut y avoir un nombre quelconque de séparateurs entre deux symboles consécutifs mais aucun séparateur ne peut se trouver au milieu d'un identificateur, d'un nombre ou d'un symbole spécial.

Enfin, il doit y avoir au moins un séparateur entre deux nombres, identificateurs ou mots réservés.

III.4.2. Descriptions des différents modules

A/ - INTRODUCTION

Il est nécessaire pour la descritpion précise et détaillée de la syntaxe d'un langage d'avoir un outil de description structuré, simple d'usage et facilement transmissible.

Tenant compte de ces critères, nous avons préféré le diagramme syntaxique de CONWAY à la forme de BACKUS-NAUR. Le premier présentant un aspect graphique qui rend plus facile et plus agréable la lecture.

Rappel : Le diagramme syntaxique de CONWAY est un graphe orienté (un chemin quelconque à travers le graphe aboutissant sur un arc de sortie du graphe définit une description syntaxiquement correcte). Les noeuds du graphe sont essentiellement des rectangles, des cercles et des cases à bouts arrondis.

- Les rectangles font référence à un autre diagramme défini par ailleurs.
- Les symboles spéciaux se trouvent dans les cercles ou les cases à bouts arrondis.
- On indique une éventuelle répétition d'une forme à l'aide d'un arc de retour.

Un exemple est donné par la figure III.4

NOTA : Dans le cadre de ce mémoire les cases à bouts arrondis contiendront exclusivement les mots réservés , les cercles contiendront les opérateurs et symboles délimiteurs (:, =, ;, ,, ., ').

B/ - MODULE DECLARATION DE NOEUDS

Tout noeud (place, transition) appartenant au réseau doit être déclaré dans un module de déclaration de noeuds. A chaque noeud est alors associé un identificateur et un type.

Le mot réservé NOEUDS introduit ce module dont la structure générale (semblable à la déclaration de variable en langage PASCAL) est représenté par le diagramme syntaxique de la figure III.4.

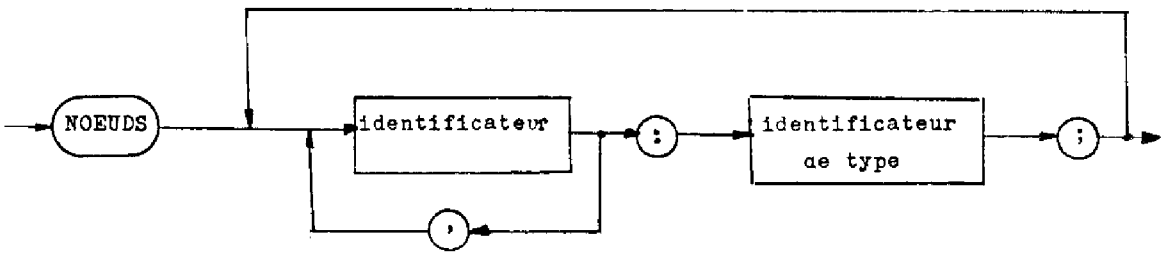


Figure III.4

Suivant que les identificateurs décrivent des transitions ou des places, l'identificateur de type sera le mot réservé TRANSITION ou le mot réservé PLACE.

Au type PLACE est associé un entier optionnel, celui-ci permettant de donner le marquage initial des places décrites. Le marquage initial zéro est pris par défaut.

Fort de ceci, le diagramme syntaxique complet et détaillé du module NOEUDS est donné par la figure III.5.

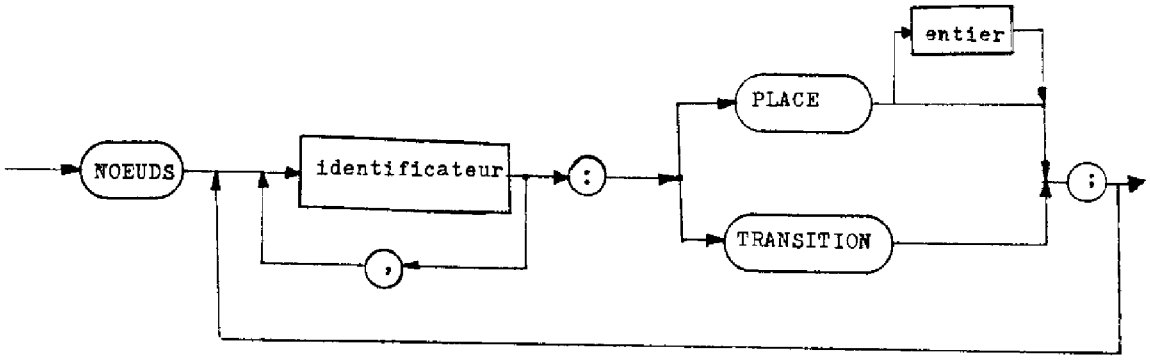


FIGURE III.5 - Module de déclaration de noeuds

Exemple : Considérons le réseau de Pétri de la figure III.6 (exemple des Lecteurs-Ecrivains). Une déclaration correcte des noeuds de ce réseaupeut être la suivante :

```
NOEUDS
        LIRE, ECRIRE : PLACE 0 ;        { zéro étant facultatif }
        RESSOURCE : PLACE 3 ;
    AUT_LEC, AUT-ECRI   : TRANSITION ;
    FIN_LEC, FIN_ECRI   : TRANSITION ;
```

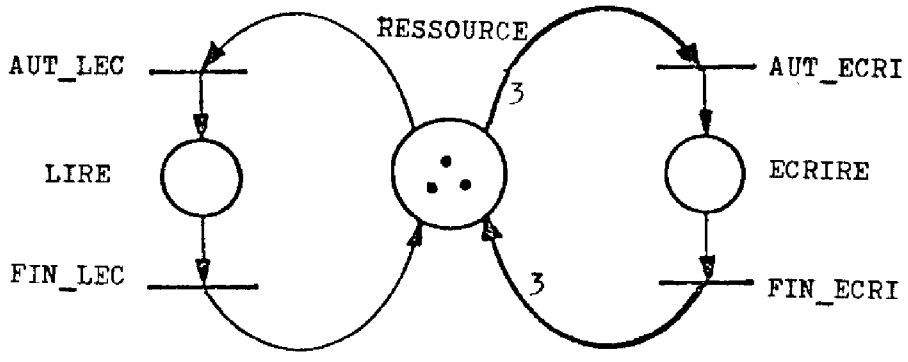


FIGURE III.6

C/ - MODULE DE DECLARATION D'ARCS

Les arcs sont définis par rapport aux transitions (Entrants/ Sortants des transitions).

Ils peuvent être décrits en spécifiant les listes des entrants et des sortants de chaque transition ou encore en décrivant des chemins du graphe constituant le réseau (Place/Transition→Transition/Place...). Le choix étant laissé à l'appréciation de l'utilisateur.

Dans les deux cas, le poids d'un arc n'apparaît explicitement que lorsqu'il est supérieur à un (1 est pris par défaut). Le module de déclaration d'ARCS est introduit par le mot réservé ARCS . Il comprend ensuite les déclarations de listes des entrants et/ou de sortants de transitions et/ou les déclarations de chemins.

Rappel : Tout noeud apparaissant dans une déclaration d'arcs doit avoir été déclaré au préalable avec son type.

a) Déclaration des listes d'entrants/sortants des transitions.

Cette déclaration est introduite par le mot réservé ENTREE ou SORTIE suivant que l'on décrit des précédents ou des suivants de transition suivi de l'identificateur représentant la transition dont on veut spécifier les places précédentes/suivantes, vient ensuite la liste complète de celles-ci et des poids respectifs. Le diagramme de la figure III.7 en représente la syntaxe.

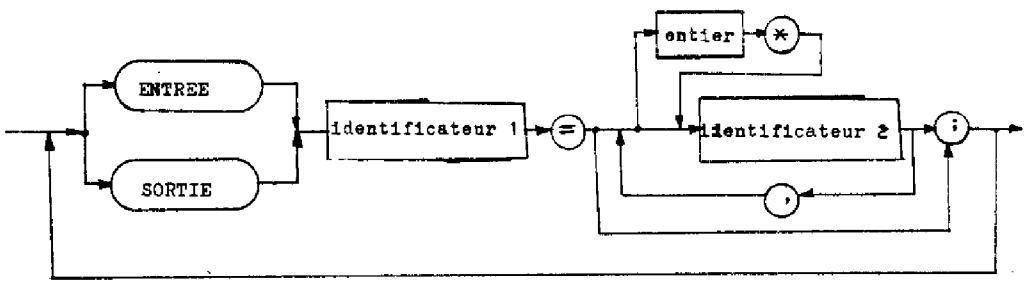


Figure III.7

où :

- Identificateur 1 désigne la transition considérée (type T).
- Identificateur 2 désigne une place suivante ou précédente de la transition (type P).
- Entier désigne le poids de l'arc reliant la transition aux différentes places (ce paramètre est optionnel, la valeur 1 étant prise par défaut).

Exemple :

Prenons le réseau de la figure III.6 en y ajoutant la place ATTENTE comme place précédente de la transition AUT_ECRI à l'aide d'un arc de poids 1. (Figure III.6.bis).

Voici pour quelques transitions, des possibilités de déclarations d'entrants/sortants syntaxiquement correcte.

```
ENTREE      AUT_ECRI = 3 * RESSOURCE , ATTENTE ;
SORTIE      AUT_ECRI = ECRIRE ;
SORTIE      AUT_LEC  = LIRE
```

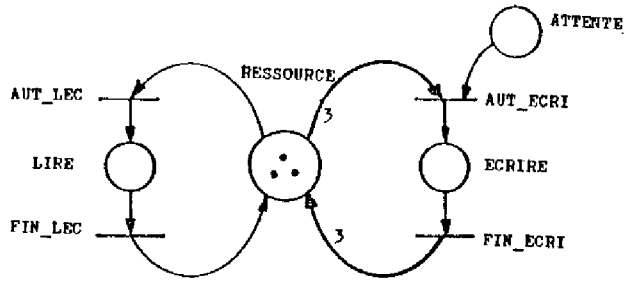


Figure III.6.bis

N.B : L'instruction vide est possible mais inutile étant donné que par défaut des listes d'entrées/sorties des transitions sont considérées comme vides.

Exemple : ENTREE TSOURCE = ; (noeud source)

b) Déclarations de chemins

Un choix judicieux peut amener l'utilisateur à décrire les arcs de réseau sous forme de chemins (par exemple pour la description d'automatismes logiques comprenant de longues séquences). Suivant les cas il peut être décrit un ou plusieurs chemins ; ceux-ci sont séparés entre eux par une virgule et sont compris entre une ouverture et une fermeture de parenthèse. Dans une déclaration de chemin, deux identificateurs successifs ne peuvent désigner deux noeuds du même type, car dans un réseau de Pétri un arc ne peut relier qu'une transition à une place ou une place à une transition. Le poids d'un arc est représenté par un entier apparaissant entre deux identificateurs représentant respectivement le noeud de départ et le noeud d'arrivée de l'arc. Lorsque le poids est égal à un, l'entier qui le représente est facultatif.

La déclaration de chemins est introduite par le mot réservé CHEMIN . Le diagramme de la figure III.8 en représente la syntaxe.

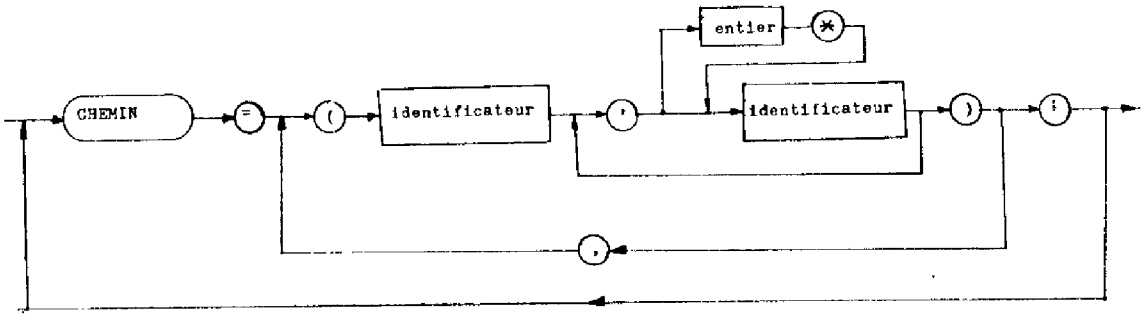


Figure III.8

Exemple : En considérant le réseau de la figure III.6, une déclaration d'arcs sous forme de chemins peut être la suivante :
CHEMIN = (ECRIRE,FIN_ECRI, 3 , RESSOURCE, AUT_LEC, LIRE, FIN_LEC, RESSOURCE) , (RESSOURCE, 3, AUT_ECRI, ECRIRE) ;

Remarque :

De par leur sémantique même, les chemins sont additfs, c'est-à-dire qu'un identificateur de place ou de transition peut apparaître dans plusieurs chemins différents. Ainsi on peut déclarer les entrants et sortants d'une même transition en plusieurs reprises lors d'une déclaration d'arcs au moyen du mot réservé CHEMIN (voir exemple ci-dessus). Il n'en est pas de même des déclarations au moyen des mots réservés ENTREE ou SORTIE qui doivent comprendre en une seule déclaration la liste complète des entrants/sortants.

Remarque :

Un chemin peut contenir un circuit.

c) Déclaration globale

La déclaration peut être résumée par le diagramme syntaxique condensé de la figure III.9.

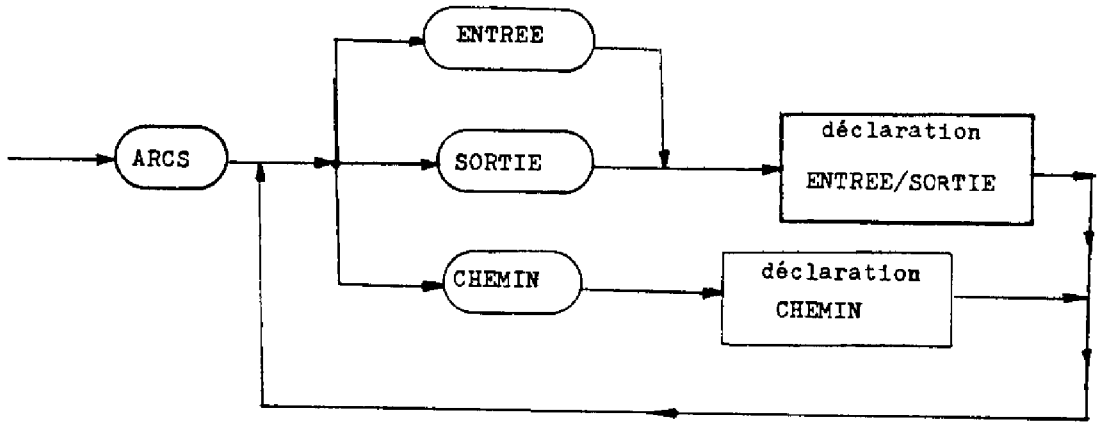


Figure III.9 - Déclaration d'arcs

Exemple :

Reprenons le réseau de la figure III.6 et donnons un exemple où l'on se sert des deux moyens de déclaration d'arcs :

```
ARCS
    ENTREE AUT_ECRI = 3 * RESSOURCE ;
    SORTIE AUT_ECRI = ECRIRE ;
    CHEMIN  = (ECRIRE, FIN_ECRI, 3 , RESSOURCE, AUT_LEC, LIRE),
              (LIRE, FIN_LEC, RESSOURCE) ;
```

D/ - MODULE DE DECLARATION DE VARIABLES (VAR)

a) Le mot réservé VAR introduit ce module. Les variables apparaissant dans l'interprétation du réseau (Expressions logiques, prédicats, actions) sont déclarées dans ce module. A chaque variable est alors associé un identificateur, un type et comme nous le verrons plus loin, un sous-type.

Les types ENTIER ou BOOLEEN peuvent être associés à une variable.

La syntaxe de déclaration des variables est similaire à celle de déclaration des noeuds et répond au diagramme syntaxique de la figure III.10.

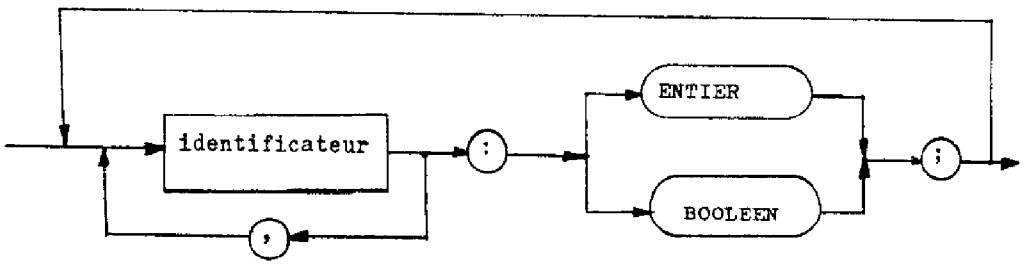


Figure III.10

Exemple : VAR

```
        B1, B2, B3, B4 : BOOLEEN ;
        E1, E2, O2, H2 : ENTIER ;
```

Lors de l'interprétation du réseau de Pétri, les variables sont utilisées en tant que variables d'entrée, de sortie, variables symboliques, variable interne ou temporisation.

Pour une vérification de l'utilisation correcte de ces variables (une action ne peut pas, par exemple, porter sur une entrée ou sur une variable symbolique,...), il est indispensable de définir les sous-types correspondants.

C'est ainsi qu'après avoir déclaré la liste des variables et le type qui leur est associé, il convient de préciser leur sous-type :

b) Déclaration des variables d'entrée et de sortie

La déclaration est introduite par le mot réservé ENTREE ou SORTIE, vient ensuite la liste des variables concernées suivies d'une chaîne de caractères entre cotes donnant le registre et le bit correspondant physiquement à la variable déclarée.

Cette déclaration répond au diagramme syntaxique de la figure III.11.

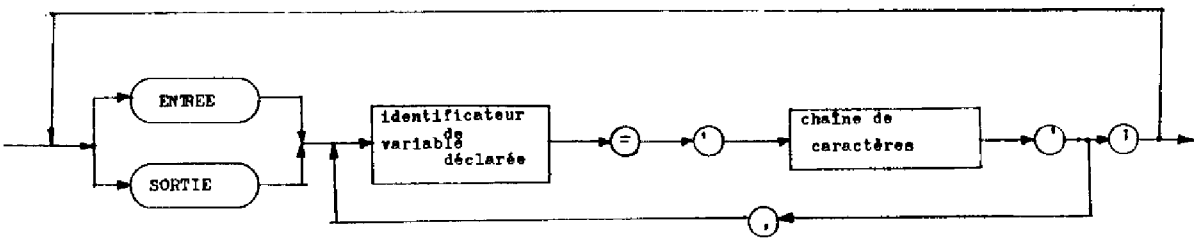


Figure III.11

Exemple :

```
VAR
    B1, B2, B3, B4 : BOOLEEN ;
    E1, E2, O2, H2 : ENTIER ;
    ENTREE
          B1 = 'O1' ,
          E1 = 'O2' ;
    SORTIE
          B2 = '11' ,
          B3 = '12' ;
```

c) Déclaration des temporisations

Leur spécification syntaxique est similaire à celle des variable d'entrée et de sortie. Elles doivent être décrites par des variables booléennes. La durée de la temporisation est exprimée entre cotes par un entier.

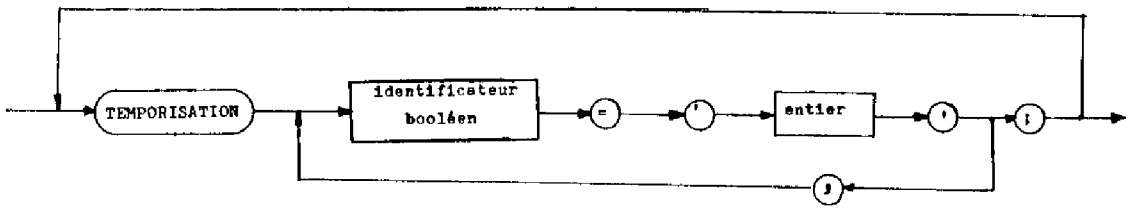


Figure III.11.bis.

Exemple : TEMPORISATION

```
T1 = 'O3' , T2 = '6O' ;
```

d) Déclaration de variables symboliques

Lorsqu'une expression logique apparaît souvent dans une condition logique ou dans plusieurs conditions logiques, il est intéressant de ne pas avoir à la réécrire chaque fois, pour celà on lui donne un nom (de

variable) et c'est ce qui nom qui représentera l'expression chaque fois qu'elle sera sollicitée.

La variable symbolisant l'expression logique doit avoir été déclarée, et la syntaxe de la déclaration de cette variable comme variable symbolique est représentée par le diagramme de la figure III.12.

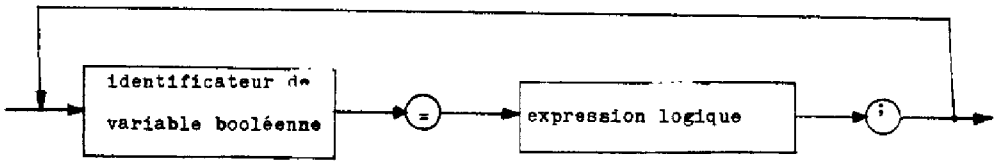


Figure III.12

Définissons la syntaxe d'une expression logique :

Les variables apparaissant dans une expression logique doivent avoir été déclarées, elles peuvent être de type BOOLEEN ou de type ENTIER.

Les opérateurs s'appliquant aux variables booléennes d'une expression logique sont les mots réservés suivants :

ET : "et" logique ou conjonction

OU : "ou" logique ou disjonction

PAS : "non" logique ou négation

Les variables de type ENTIER n'apparaissent dans une expression logique que sous la forme de prédicats élémentaires. Ces derniers sont formés d'une variable entière suivie d'un opérateur de relation et d'un second membre qui est un entier ou une autre variable entière.

L'ensemble des opérateurs de relation est composé des symboles spéciaux suivants : $\{ >, >=, <=, <>, = \}$

Une expression logique peut être décrite soit par une expression ne faisant intervenir que des variables booléennes, soit par des expressions ne comprenant que des variables entières (Prédicats), soit par une

expression plus générale qui peut faire intervenir des variables booléennes et des variables entières.

Exemple : B1 OU B2 ;
E1 = O ;
B1 OU B2 ET (E1 = O) ;

Le diagramme syntaxique global d'une expression logique est donné par la figure III.13

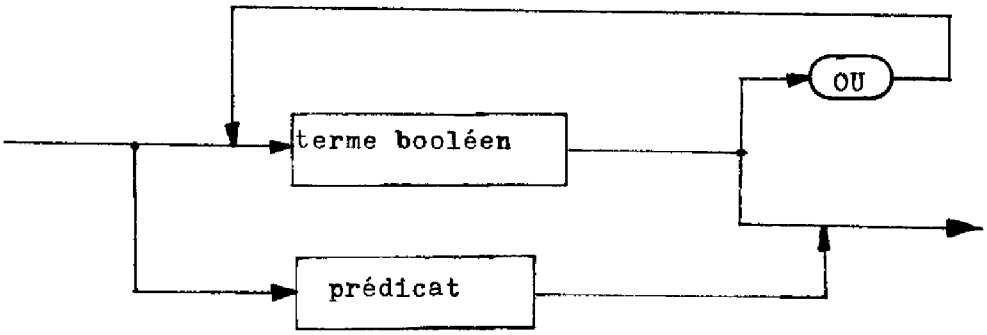


Figure III.13 - Expression logique

• Prédicat :

Seuls des prédicats simples, comprenant au plus deux variables sont autorisés. Le diagramme syntaxique d'un prédicat est donné par la figure III.14.

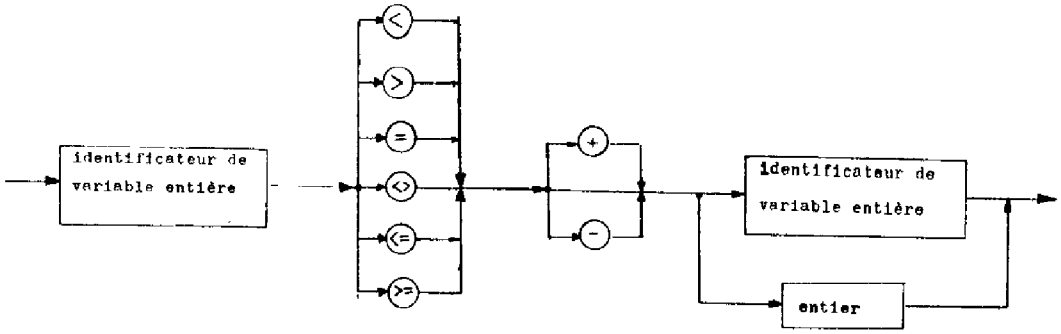


Figure III.14 - Prédicat

• Terme Booléen.

Une expression logique simple est une combinaison de variables et d'opérateurs logiques mais elle peut comporter des expressions parenthèsées qui elle-mêmes peuvent en comprendre. A l'intérieur des parenthèses peuvent aussi bien intervenir des variables booléennes que des variables entières (prédicats).

Le diagramme syntaxique d'un terme boléen est donné par la figure III.15

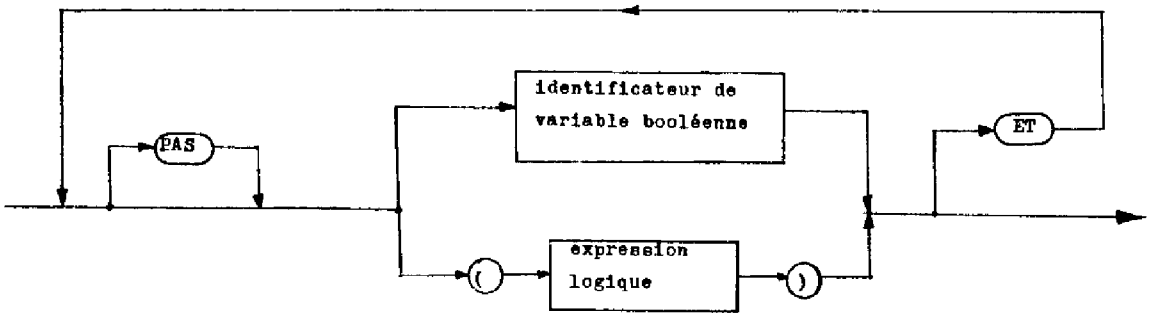


Figure III.15 - Terme booléen

En tenant compte des diagrammes des figures III.13, III.14, III.15 on vérifier que l'expression logique :

B1 ET (((B2 OU B3) ET PAS (E1= -5)) ET B4)

est syntaxiquement correcte.

Ainsi une déclaration de variable symbolique peut être donnée par l'exemple suivant :

B4 = B3 ET PAS (B1 OU B2) OU (E1 = -5) ;

e) Variables internes

Les variables de type BOOLEEN ne figurant ni dans une déclaration de variable d'entrée ou de sortie, ni dans une déclaration de temporisation ni dans une déclaration de variable symbolique sont considérées comme

variables internes. Quant aux variables de type ENTIER, elles pourront être utilisées dans l'expression d'un prédicat ou comme variables internes.

f) Description globale du module VAR

Le diagramme syntaxique du module VAR comprenant la déclaration des variables, de leur type et de leur sous-type est représenté par la figure III.16.

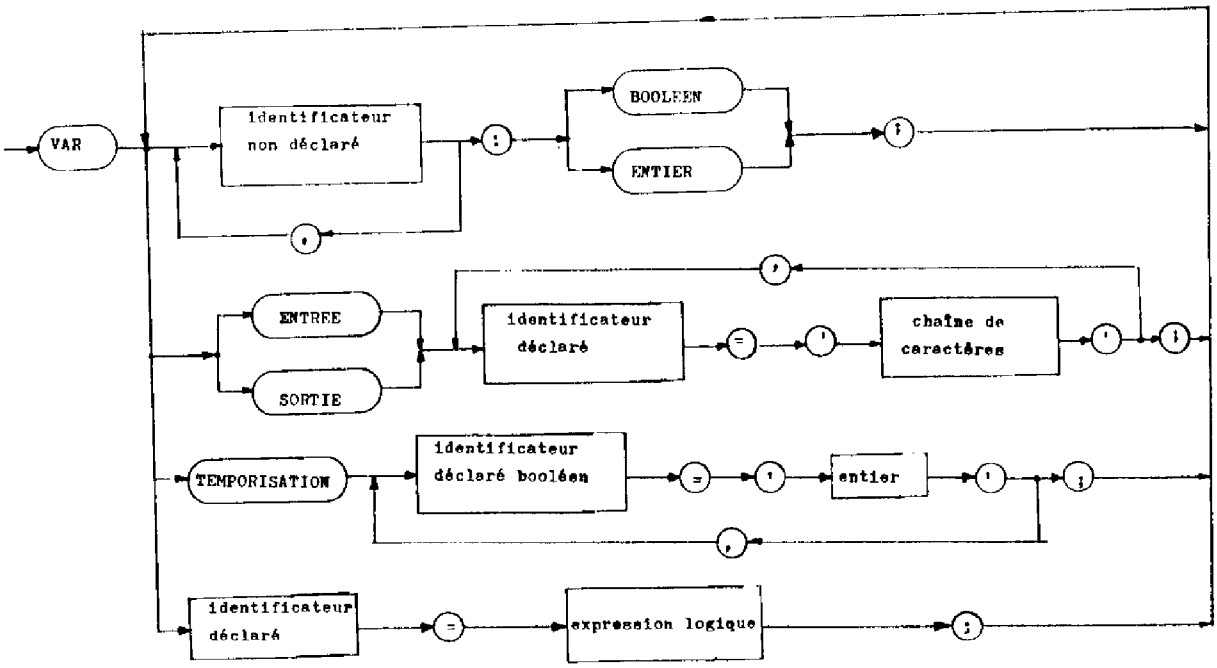


Figure III.16 - Module VAR

E/ - MODULE INTERPRETATION

Nous avons vu au chapitre I que les automatismes logiques sont en relation constante avec le monde extérieur et que cette dépendance est matérialisée par l'interprétation du réseau de Pétri (celui ne représentant par lui-même que la structure de commande).

Aux transitions sont associées des conditions logiques, des actions impulsionnelles, des opérations d'incrémentation et de décrémentation sur des variables internes, des lancements et arrêts de temporisations.

Aux places sont associées des actions portant sur des sorties booléennes. Le module prenant en compte l'interprétation du réseau de Pétri est introduit par le mot réservé INTERPRATION , viennent ensuite les déclarations des conditions logiques et/ou des actions associées aux places et aux transitions.

a) Conditions logiques

La déclaration d'une condition associée à une transition est introduite par le mot réservé COND suivi de l'identificateur spécifiant la transition considérée puis de l'expression de la condition logique.

Cette déclaration répond à la syntaxe du diagramme de la figure III.17.

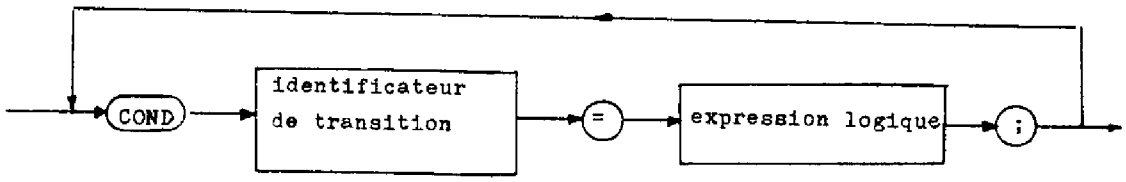


Figure III.17

Exemple : COND AUT_ECRI = B2 OU B3 ET PAS (E1=O) ;

b) Actions

La déclaration d'actions associées à une place ou à une transition est introduite par le mot réservé ACTION suivi de l'identificateur de place ou de transition puis de l'expression de l'action ou de la séquence d'actions à effectuer (voir figure III.18).

Les actions associées aux places et aux transitions étant de nature différentes, il convient de les considérer distinctement.

Il est à noter qu'aucune action ne doit porter ni sur des variables d'entrées, ni sur des variables symboliques.

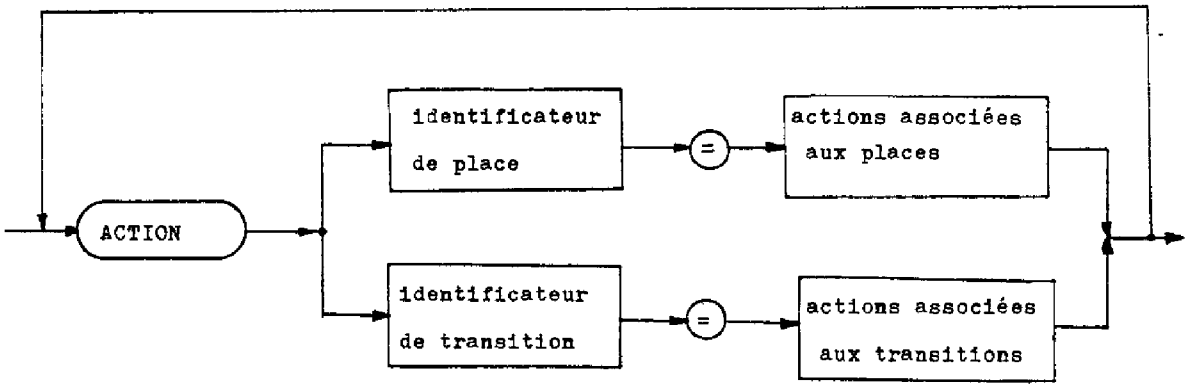


Figure III.18 - Déclaration d'actions associés aux places et aux transitions.

1) Actions associées aux transitions

Comme nous l'avons vu, ce sont des actions impulsionnelles du type signaux de déclenchement (par exemple lancer ou arrêter une temporisation) ou des opérations d'initialisation, d'incrémentation ou de décrémentation d'une ou plusieurs unités sur des variables entières (compteurs variable de synchronisation,...) ou encore des opérations sur des variables booléennes.

Lorsqu'à une transition est associée plusieurs actions, la séquence d'actions doit être comprise entre les mots réservés DEBUT et FIN suivi d'un (;) qui marque la fin de l'instruction DEBUT - FIN. C'est en fait le seul endroit du programme où une suite d'instructions semblable à une procédure est autorisée.

La fin de chaque action élémentaire est ponctuée par un point virgule. Le diagramme de la figure III.19 représente la syntaxe de déclaration globale des actions associées aux transitions.

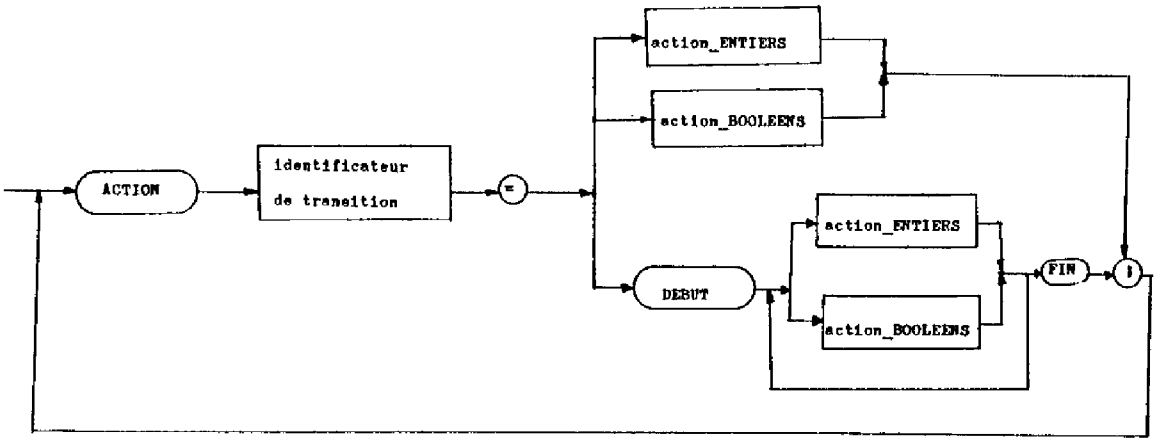


Figure III.19 - Déclaration d'actions associées aux transitions

Les diagrammes syntaxiques correspondants aux actions associées aux entiers et aux booléens sont respectivement donnés par les figures III.20 et III.21.

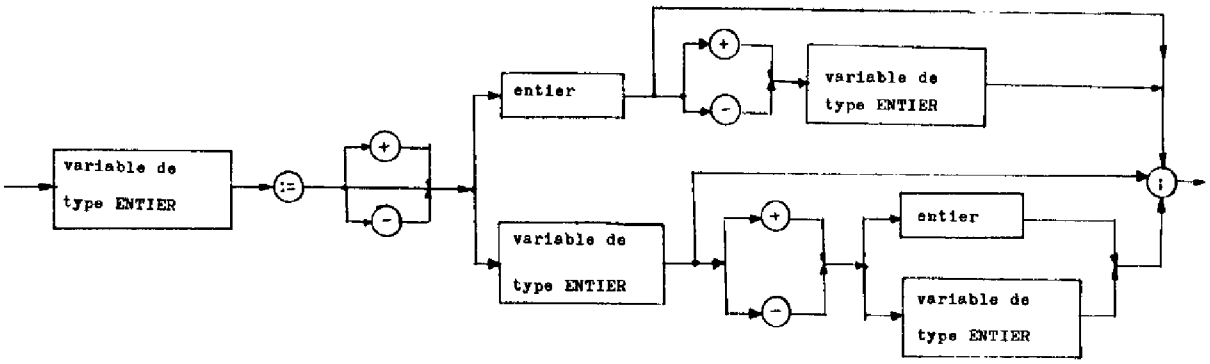


Figure III.20 - Action portant sur une variable entière

Exemples : E1 : = - 10 ;

E1 : = E2 + 5 ;

E1 : = E2 - E3 ;

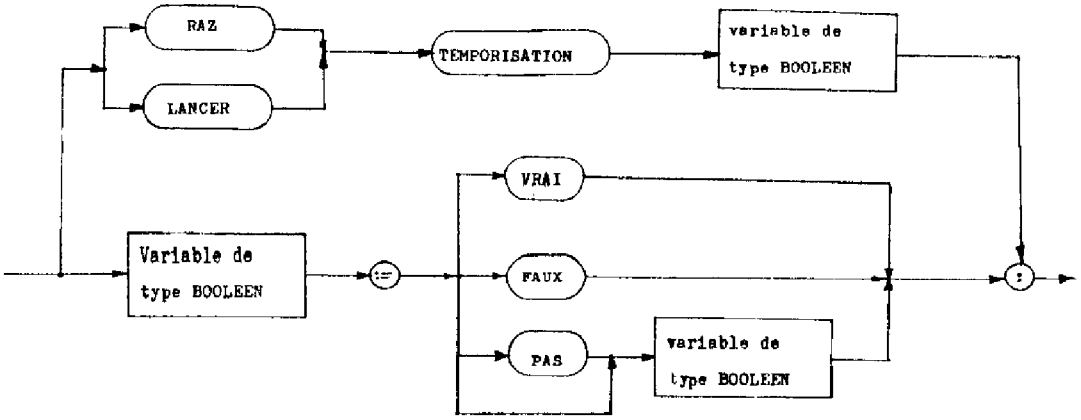


Figure III.21 - Action portant sur une variable booléenne.

Exemples : B1 := VRAI ;
B2 := FAUX ;
B1 := PAS B2 ;
LANCER TEMPORISATION T ;

Exemple de déclaration d'actions associées aux transitions :

```
ACTION T1  =  DEBUT
              LANCER TEMPORISATION T ;
              E2 := E1 + 5 ;
              B1 := PAS B2 ;
              RAZ TEMPORISATION TMP ;
              FIN ;
ACTION T2  =  E1 := E2 -10 ;
```

2) Actions associées aux places

Rappelons que les actions associées aux places sont maintenues tout le temps que la place est marquée et annulées dès que la place est démarquée. Les seules variables sur lesquelles portent ces actions doivent être des variables booléennes de sortie et d'autre part on ne peut associer une séquence à une place. Ainsi la syntaxe retenue pour la déclaration des actions associées aux places est donnée par la figure III.22.

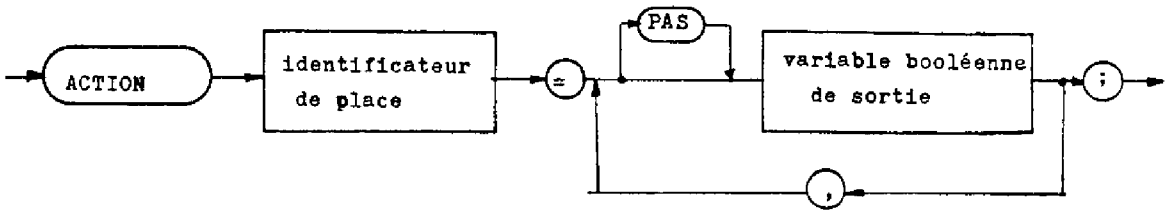


Figure III.22 - Déclaration d'une action associée à une place

Exemple :

ACTION P1 = B3, PAS B4, B5 ;

ce qui signifie que la sortie représentée par la variable B3 est positionnée à 1, celle représentée par B4 à 0 et celle représentée par B5 à 1.

REMARQUE :

Le fait que les actions associées aux places soient maintenues tout le temps que la place est marquée nécessite une validation plus poussée du réseau. En effet il faut vérifier qu'une variable booléenne donnée n'est pas affectée à deux places pouvant être simultanément marquées.

F/ - EXEMPLE DE DECLARATION D'UN RESEAU DE PETRI INTERPRETE

Considérons l'automatisme A11 de la base de lancement ARIANE 2. Il est spécifié par le réseau de la figure I.20. La description de ce réseau de Pétri, muni de son interprétation à l'aide du langage peut être la suivante (en considérant que la fermeture d'une vanne correspond à la mise à zéro de la variable de sortie qui lui est associée et vice-versa)

```
{ description de l'automatisme de régulation de niveau des compléments
  de pleins des réservoirs du H8 (pris individuellement) }

NOEUDS
        P1 : PLACE 1 ;
        P2, P3 : PLACE ;
        T1, T2, T3, T4, T5, T6, T7 : TRANSITION ;

ARCS
        CHEMIN = (P1, T2, P2, T3, P3, T4, P3, T5, P2, T6, P1) ,
                 (P1, T1, P3, T7, P1) ;

VAR     MES, MHS, HH, T : BOOLEEN ;
        XVP_104 , HVE_102 : BOOLEEN ;
        SORTIE
              XVP_104 = '14', HVE_102 = '17' ;
```

```
ENTREE
        HH = '01' , MES = '02' , MHS = '06' ;

TEMPORISATION
                T = '60' ;

INTERPRETATION

        COND T1 = MES ET HH ;
        COND T2 = MES ET PAS HH ;
        COND T3 = HH ;
        COND T4 = T ET HH ;
        COND T5 = T ET PAS HH ;
        COND T6 = MHS
        COND T7 = MHS ;

        ACTION T1 = LANCER TEMPORISATION T ;
        ACTION T3 = LANCER TEMPORISATION T ;
        ACTION T4 = LANCER TEMPORISATION T ;
        ACTION T7 = RAZ TEMPORISATION T ;

        ACTION P2 = PAS  XVP_104, PAS HVE_102 ;
        ACTION P4 = XVP_104, HVE_102 ;

FIN. {fin de description de A11}
```

III.4.3. Déclaration de sous-réseaux

Comme nous l'avons vu au chapitre I, lorsque le réseau de Pétri à décrire est de taille importante, il est souvent commode de le décrire par morceaux, et cela se traduit par un réseau principal et un ou plusieurs sous-réseaux qui peuvent être imbriqués les uns dans les autres ou indépendants. Nous avons vu (cf.III.3.2) qu'une structure de bloc permet une telle description.

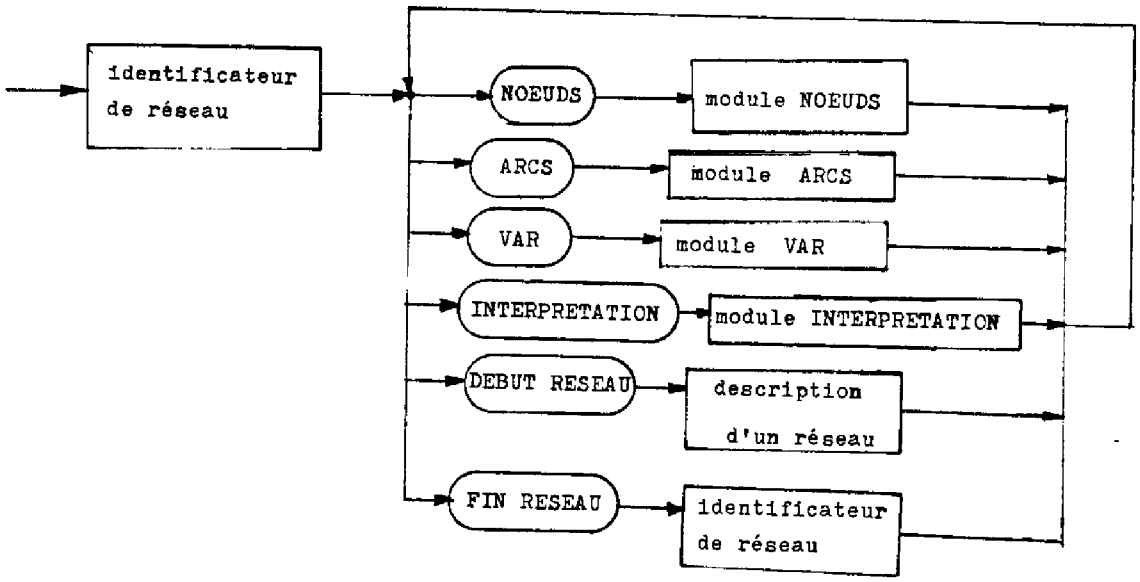
La déclaration d'un sous-réseau est introduite par les mots réservés DEBUT RESEAU suivis du nom du sous-réseau, viennent ensuite les différents modules de description (NOEUDS - ARCS - VAR - INTERPRETATION) La fermeture du sous-réseau est marquée par les mots réservés FIN RESEAU suivis du nom du sous-réseau.

Afin de respecter la structure de bloc, les ouvertures et fermetures de sous-réseaux doivent obéir à la règle suivant laquelle le dernier sous-réseau ouvert doit être le premier fermé.

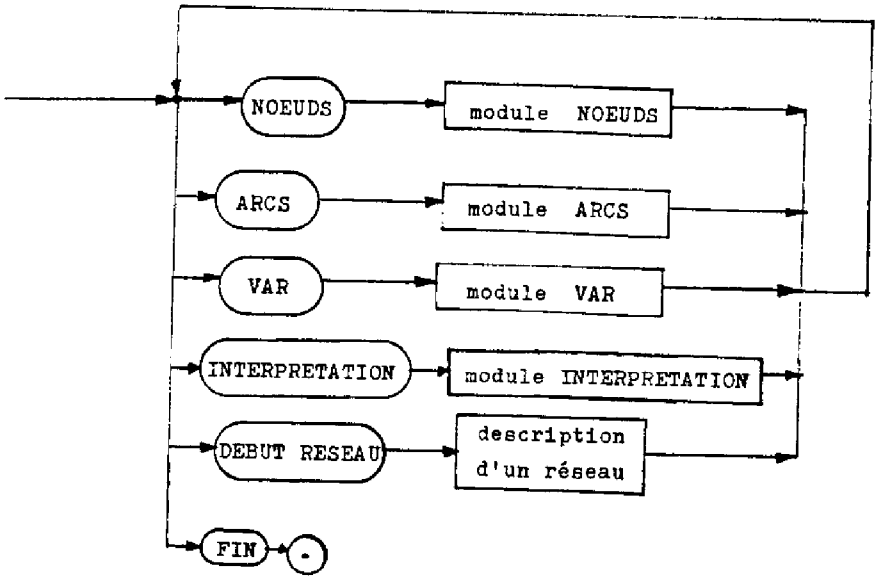
Il est bien évident que le nombre de fermetures de sous-réseaux doit être égal au nombre d'ouvertures de ceux-ci.

Le diagramme de la figure III.23.a représente la syntaxe de déclaration de réseau et sous-réseaux. Quant à la figure III.23.b, elle représente la syntaxe de déclaration du réseau global. Un exemple est donné en considérant l'ensemble des automatismes de sécurité du H8 dont la structure globale a été donnée par la figure III.2 :

```
{ ensemble des automatismes de sécurité du H8 }
DEBUT RESEAU GLOBAL
|
|      DEBUT RESEAU [PX_AY]
|      |
|      |   DEBUT RESEAU AY
|      |   |      ʃ
|      |   FIN RESEAU AY
|      |          ʃ   {description du corps de PX }
|      FIN RESEAU PX_AY
|
|      DEBUT RESEAU [A7_A11_AX]
|      |      NOEUDS {déclaration des noeuds communs à A7, A11 et AX}
|      |
|      |      DEBUT RESEAU AX
|      |      |      ʃ
|      |      FIN RESEAU AX
|      |
|      |      DEBUT RESEAU A11
|      |      |      ʃ
|      |      FIN RESEAU A11
|      |
|      |      DEBUT RESEAU A7
|      |      |      ʃ
|      |      FIN RESEAU A7
|      FIN RESEAU [A7_A11_AX]
|             ʃ   {description du corps de AS}
FIN RESEAU GLOBAL
FIN.
```



a) Description d'un réseau



b) Description réseau global

Figure III.23 - Diagrammes syntaxiques de déclaration de sous-réseaux et réseau global.

III.4.4. Représentation de la communication entre réseaux (cf.I.3)

1) Communication par place (envoi de message simple)
Soit la place P3 commune aux réseaux local 1 et au réseau local 2 (figure III.24).

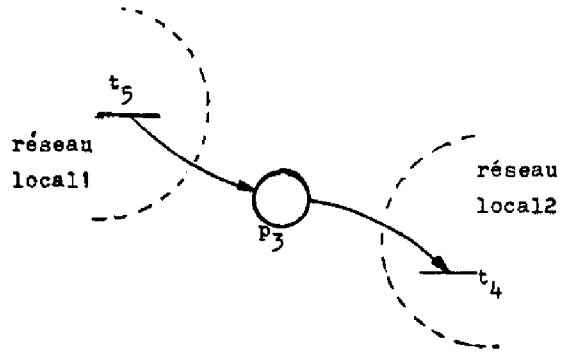


Figure III.24

La déclaration de cette communication, compte tenu des règles de portée énoncées et en ne faisant intervenir que les noeuds et arcs concernés par la communication peut être la suivante :

{ Réseau principal correspondant à la communication seule }

```
NOEUDS
      P3 : PLACE ;
      DEBUT  RESEAU LOCAL_1
      |      NOEUDS
      |            T5 : TRANSITION ;
      |      ARCS
      |            CHEMIN = (T5, P3);
      FIN  RESEAU LOCAL_1

      DEBUT  RESEAU  LOCAL_2
      |      NOEUDS
      |            T4 : TRANSITION ;
      |      ARCS
      |            CHEMIN = (P3, T4) ;
      FIN  RESEAU LOCAL_2
FIN.  fin de la description du réseau principal
```

Considérons l'ensemble des automatismes AX, A7 et A11 de la base de lancement ARIANE 2. Leur synchronisation est donnée par le réseau de Pétri de la figure I.28 où les trois réseaux communiquent à l'aide des places de synchronisation PS1 et PS2.

```
DEBUT  RESEAU [A7_A11_AX]
       NOEUDS
            PS1, PS2 : PLACE 1 ; { noeuds correspondant à la communication
                                   entre les trois réseaux }
       DEBUT  RESEAU AX
              NOEUDS
                   T1, TPRIM1, TSECOND1 , TTIERCE1 : TRANSITION ;
                        { noeuds t 1, t '1, t "1, t "' 1 }
                   T8 : TRANSITION ;
                                   { mise hors service }
              ARCS
                   CHEMIN = (PS1, T1) (PS1, TPRIM1), (PS1,TSECOND1),
                            (PS1, TTIERCE1) ;
                   CHEMIN = (T12, PS1) ;
       FIN  RESEAU  AX

       DEBUT  RESEAU  A11
              NOEUDS
                    T1, T2 : TRANSITION ; { mise en service }
                    T6, T7 : TRANSITION ; { mise hors service}
              ARCS
                    CHEMIN = (PS2, T1),(PS2,T2) ;
                    CHEMIN = (T6, PS2),(T7,PS2) ;
       FIN RESEAU A11

       DEBUT RESEAU A1
                   {déclarations de nœuds et chemins liés à PS1 et PS2}
       FIN  RESEAU  A7

FIN  RESEAU [A7_A11_AX]
```

2) Communication par transition ("rendez-vous")

La communication par "rendez-vous" représenté par la figure III.25

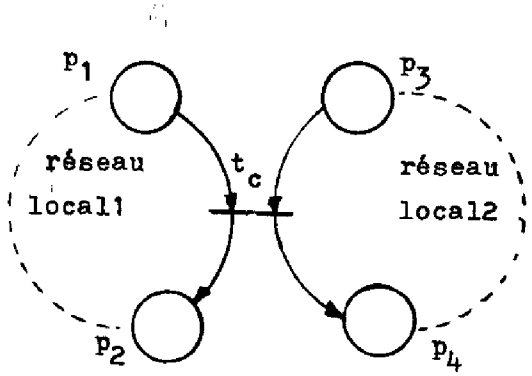


Figure III.25

peut être décrite de la manière suivante (compte tenu des règles de portée des identificateurs énoncés), où seuls les noeuds et arcs concernés par la communication sont explicités :

réseau principal correspondant à la communication seule

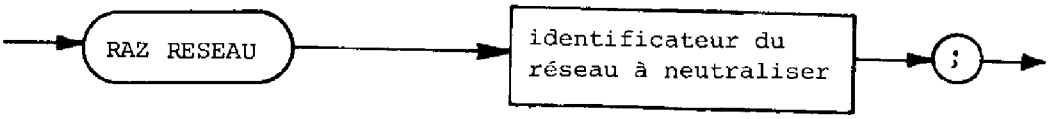
```
NOEUDS
      TC : TRANSITION ;   {noeud commun aux deux réseaux}
   DEBUT  RESEAU  LOCAL_1
   |      NOEUDS
   |            P1, P2 : PLACE ;
   |      ARCS
   |            CHEMIN = (P1, TC, P2) ;
   FIN  RESEAU  LOCAL_1

   DEBUT  RESEAU  LOCAL_2
   |      NOEUDS
   |            P3, P4 : PLACE ;
   |
   |      ARCS
   |            CHEMIN = (P3, TC, P4) ;
   FIN  RESEAU  LOCAL_2

FIN.   {fin réseau principal}
```

REMARQUE :

La transition TC n'est découpée qu'au niveau de la spécification. A partir de cette spécification décomposée, un réseau de Pétri unique, ne comportant qu'une transition TC est construit et c'est ce réseau qui sera chargé dans l'automate programmable. Le problème serait tout autre si la mise en oeuvre devait se faire sur un réseau d'automates.

3) Relation maître-esclave - Arrêt d'urgence.

Lorsqu'un automatisme peut en neutraliser un autre dans un cas d'arrêt d'urgence, cela peut s'exprimer comme suit : à une transition du réseau maître est associé l'ordre suivant :



Prenons pour exemple la synchronisation entre les automatismes PX et AY représentée par la figure I.29. Le fait que l'automatisme PX peut verrouiller l'automatisme AY peut s'exprimer comme suit :

ACTION T2 = RAZ RESEAU AY ;

Le déverrouillage (retour au marquage initial) du réseau esclave par le réseau maître se traduit par la syntaxe suivante :

En considérant l'exemple ci-dessus on peut exprimer le fait que l'automatisme PX déverrouille l'automatisme AY comme suit :

ACTION T3 = LANCER RESEAU AY ;

4) Exemple

Considérons la synchronisation entre les automatismes PX et AY donnée par le réseau de Pétri interprété de la figure I.29. La description complète et détaillée de cette synchronisation est la suivante :

```
{ Description de la synchronisation entre PX et AY }
DEBUT RESEAU  PX_AY
      DEBUT RESEAU AY
            NOEUDS
             P1 : PLACE 1 ;
             P2,P3,P4,P5,P6 : PLACE ;
             T1,T2,T3,T4,T5,T6,T7 : TRANSITION ;
             T8,T9,T10,T11,T12 : TRANSITION ;
             TPRIM1,TSECOND1,TTIERCE1 : TRANSITION ;  { t'1,t''1,t'''1 }

            ARCS
             CHEMIN=(P1,T1,P2,T2,P3,T3,P4,T4,P5,T8,P6,T12,P1),
                    (P5,T7,P4,T6,P3,T5,P2,T11,P6),(P3,T10,P6),
                    (P4,T9,P6),(P1,TPRIM1,P3),(P1,TSECOND1,P4),
                    (P1,TTIERCE1,P5);

            VAR
             MES,MHS,ORDRE_5,ORDRE_6,ORDRE_7,ORDRE_8 : BOOLEEN ;
             ORDRE_9,TMP,BS1,BS2,BS3,BS4 : BOOLEEN ;
                    { TMP : temporisation }
             P : ENTIER ;  { P : pression }
             ENTREE
              P='0',      { Entrée analogique }
              MES='10',MHS='11';
             SORTIE
              ORDRE_5='20',ORDRE_6='21',ORDRE_7='22',
              ORDRE_8='23',ORDRE_9='24';
             TEMPORISATION
              TMP='60';

             BS1=(P>60);
             BS2=(P>75);        { Variables symboliques }
             BS3=(P>150);
             BS4=(P>350);

           INTERPRETATION
            COND T1 = MES ET PAS BS2;
            COND TPRIM1 = MES ET BS2 ET PAS BS3;
            COND TSECOND1 = MES ET BS3 ET PAS BS4;
            COND TTIERCE1 = MES ET BS4;
            COND T2 = BS2;
            COND T3 = BS3;
            COND T4 = BS4;
            COND T5 = PAS BS1;
            COND T6 = PAS BS2;
            COND T7 = PAS BS3;
            COND T8 = MHS;
            COND T9 = MHS;
            COND T10 = MHS;
            COND T11 = MHS;
            COND T12 = TMP;
```

```
            ACTION T8 = LANCER TEMPORISATION TMP;
            ACTION T9 = LANCER TEMPORISARION TMP;
            ACTION T10 = LANCER TEMPORISATION TMP;
            ACTION T11 = LANCER TEMPORISATION TMP;

            ACTION P2 = ORDRE_5;
            ACTION P3 = ORDRE_6;
            ACTION P4 = ORDRE_7;
            ACTION P5 = ORDRE_8;
            ACTION P6 = ORDRE_9;

      FIN RESEAU AY

      DEBUT RESEAU PX
            NOEUDS
             P1 : PLACE 1;
             P2,P3,P4 : PLACE;
             T1,T2,T3,T4,T5 : TRANSITION;

            ARCS
             CHEMIN = (P1,T1,P2,T2,P3,T3,P4,T4,P1),(P2,T5,P1);

            VAR
             A_U,MES,MHS,ORDRE_2 : BOOLEEN;
             PO2 : ENTIER;
             ENTREE
                  MES='30', MHS='31', PO2='32' ;
             SORTIE
                  ORDRE_2='25' ;

             A_U = (PO2>115); { variable symbolique}

             INTERPRETATION
              COND T1 = MES;
              COND T2 = A_U;
              COND T4 = MHS;
              COND T5 = MHS;

              ACTION T3 = RAZ RESEAU AY;
              ACTION T4 = LANCER RESEAU AY;

              ACTION P4 = ORDRE_2;

      FIN RESEAU PX

FIN RESEAU [PX_AY]

FIN. { Fin de la description}
```

III.5. ANALYSE SYNTAXIQUE ET SEMANTIQUE

III.5.1. Introduction

Un analyseur syntaxique et sémantique adapté au langage a été élaboré. Il est écrit en langage haut-niveau(PASCAL). Le code objet de l'analyseur occupe 35 K octets, les données occupent environ 15 K octets . (Le sytème d'exploitation employé (CP/M) laisse encore 10 K octets libres environ, cette place peut être utilisée soit pour rentrer des réseaux de Pétri de grande taille, soit pour sophistiquer le programme analyseur).

L'analyse syntaxique suit pas à pas les diagrammes syntaxiques de CONWAY donnés aux paragraphes précédents.

Lorsque la description est correcte, les identificateurs ainsi que leur type et éventuellement leur sous-type sont rangés dans des tableaux.

Le balayage de ces tableaux sera la base de l'analyse sémantique (détection de conflit de type, d'utilisation d'identificateurs non déclarés, de redéclaration d'identificateurs, etc...).

Lorsqu'une erreur (syntaxique ou sémantique) est détectée, un message est alors envoyé à la console et l'analyse repart à la demande, jusqu'à trouver les symboles marquant la fin de la description du réseau de Pétri, ou le cas échéant la fin de fichier.

L'analyseur syntaxique et sémantique a été construit autour d'un noyau de procédures élémentaires permettant une lecture cohérente du fichier contenant la description du réseau de Pétri.

Le noyau est consitué des procédures élémentaires suivantes :

. Lecture d'un caractère (LIRE-C)

. Recherche d'un caractère significatif (REC - CAR). Un caractère significataif est n'importe quel caractère autre que le blanc, la fin de ligne et le commentaire . Chaque commentaire rencontré est envoyé sur la console.

. Lecture d'un identificateur(LEC-IDEN)

. Lecture d'un entier(LEC-ENT)

. Tester si le caractère lu est bien le caractère attendu(TEST-C)

. Indentification d'un identificateur et de son type (ID-IDEN). Cette procédure balaie le tableau des identificateurs (TAB-IDEN). Des messages d'erreur sont envoyés si le type rencontré n'est pas celui attendu ou bien si l'identificateur n'a pas été trouvé dans le tableau ou encore si celui-ci a déjà été défini.

. Envoi des messages d'erreur(ERREUR). Le message d'erreur (apparaissant à la console) contient le code de l'erreur (cf annexe 3), le numéro de la ligne où l'erreur a été détectée et le dernier identificateur lu. Suivant la nature de l'erreur un message complémentaire est envoyé précisant le type d'erreur commis.

. Restauration : lorsqu'une erreur a été détectée et signalée, l'analyseur syntaxique et sémantique se restaure et reprend l'analyse. La restauration se fait sur un délimiteur prédéfini, en général le point-virgule, ainsi l'analyseur - au moyen de la procédure CHERCHE - ignore tout ce qu'il rencontre jusqu'à trouver le délimiteur attendu d'où il reprendra l'analyse.

REMARQUE :

Il existe des cas d'erreurs pour lesquels la restauration n'a pas lieu, il s'agit surtout des cas de dépassement du nombre maximum d'éléments dans les tableaux (nombre de noeuds, d'arcs, d'identificateurs, d'entrées ou de sorties, etc... supérieur au maximum autorisé).

D'autre part, dans le cas de description de sous-réseaux, l'ouverture ou la fermeture illégale de ceux-ci ou leur omission entraîne l'arrêt de l'analyse syntaxique et sémantique.

III.5.2. Analyse syntaxique et sémantique des différents modules

Les autres procédures de l'analyseur syntaxique et sémantique sont calquées sur les diagrammes syntaxiques de CONWAY définissant le langage.

A/ - MODULE NOEUDS

Le mot réservé NOEUDS a été rencontré, l'analyseur scrute alors la liste des identificateurs qui devra être suivie du mot réservé TRANSITION

ou PLACE (éventuellement suivi du marquage initial) suivant la syntaxe définie par le diagramme de la figure III.5.

Lors de la"scrutation" de la liste des identificateurs au moyen du noyau de l'analyseur et notamment au moyen de la procédure ID - IDEN, on vérifie que les identificateurs n'ont pas été déclarés ni ne figurent parmi les mots réservés.

Lorsque la déclaration s'est avérée correcte, les identificateurs, ainsi que leur type (P : place, T : transition), sont rangés dans le tableau des identificateurs TAB - IDEN.

En cas de déclaration incorrecte, tous les identificateurs compris dans la liste déclarée sont effacés puisque leur type n'est pas encore connu, l'analyseur se restaure sur le (;) marquant la fin de la déclaration erronée et analyse la suivante, ceci jusqu'à ce qu'il trouve une ouverture de module ou éventuellement une déclaration de sous-réseaux ou le mot réservé FIN.

B/ - MODULE ARCS

Le mot réservé ARCS ayant été rencontré, l'analyseur cherche alors les mots réservés ENTREE, SORTIE ou CHEMIN.

Après avoir trouvé ENTREE ou SORTIE il vérifie que l'identificateur suivant désigne une transition et que celle-ci ne fait l'objet d'une redéclaration de la liste de ses places précédentes ou suivantes, puis il scrute la liste de ses entrants et sortants qui doivent être des identificateurs de place éventuellement précédés du poids de l'arc les reliant à la transition concernée et ceci suivant la syntaxe définie par le diagramme de la figure III.7.

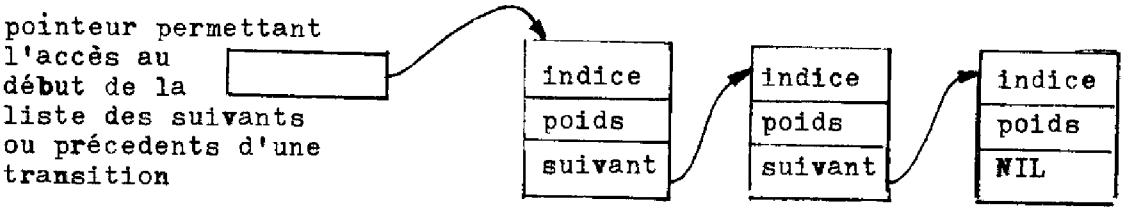
Dans le cas d'une description par chemin du graphe que constitue le réseau de Pétri, le diagramme de la figure III.8 illustre les exigences syntaxiques que vérifie l'analyseur.

L'analyse sémantique consiste à vérifier que deux indentificateurs successifs sont de type différent (type PLACE ou type TRANSITION).

La description d'arcs au moyen de chemins est additive. L'analyseur ajoute à la liste des entrants/sortants d'une transition les noeuds et les

poids des arcs rencontrés dans les chemins décrits.

Lorsque la description d'arcs est correcte (par l'une quelconque des deux formes), à chaque transition est alors associée une liste chaînée des éléments constituant les entrants et sortants de la transition. Ces éléments sont accessibles au moyen de pointeurs et comprennent chacun l'indice de la place dans le tableau des identificateurs (TAB - IDEN), le poids de l'arc reliant la place à la transition et un pointeur permettant d'accéder aux autres éléments de la liste. Cette structure dynamique de stockage du réseau permet le traitement de gros réseaux sur un microprocesseur 8 bits.



stockage des entrants/sortants d'une transition

Il est ainsi formé un vecteur qui contient pour chaque transition la liste chaînée des éléments contenant les renseignements sur les arcs (poids de l'arc et noeud de départ ou noeud d'arrivée) précédents (vecteur D - L - E : Début de la liste des Entrants) ou suivants (vecteur D - L - S : Début de la liste des Suivants)

Une redéclaration d'arcs provoque un envoi de message d'erreur. En cas d'erreur de description, l'analyseur signale celle-ci et se restaure sur le (;) marquant la fin de la déclaration concernée et les éléments décrits lors de celle-ci sont ignorés. Sauf dans le cas de description à l'aide de chemins où la restauration se fait sur la première virgule rencontrée.

C/ - MODULE VAR

L'analyseur vient de rencontrer le mot réservé VAR et scrute alors la liste des identificateurs qui devront être suivis du type qui leur est associé, celui-ci est symbolisé par les mots réservés ENTIER ou BOOLEEN suivant la syntaxe définie par le diagramme de la figure III.10.

L'analyse sémantique consiste, comme dans le module NOEUDS à vérifier que les identificateurs n'ont pas déjà été déclarés et ne figurent pas parmi les mots réservés.

En cas de déclaration incorrecte, tous les identificateurs figurant dans celle-ci sont ignorés. L'analyseur se restaure sur le (;) marquant la fin de la déclaration.

Les déclarations de sous-types ENTREE et SORTIE sont définis par le diagramme syntaxique de la figure III.11. L'analyse syntaxique est le reflet de ce même diagramme (au moyen des procédures que nous avons décrites et qui constituent le noyau de l'analyseur.

Lorsque la déclaration est correcte, le sous-type et le code (correspondance physique) de chaque variable sont rangés dans le tableau des entrées/sorties (TAB-ES).

L'analyse sémantique consistera à vérifier que les variables dont le sous-type est ENTREE ou SORTIE ont déjà été déclarées comme variables booléennes ou entière et le balayage du tableau TAB - ES permet de vérifier qu'aucun sous-type ne leur a déjà été affecté (par exemple une variable ne pourra être déclarée comme sortie si elle est déjà déclarée comme entrée, temporisation ou variable symbolique et vice-versa).

L'analyse de la déclaration d'une temporisation est similaire à celle d'une déclaration des ENTREES/SORTIES et suit le diagramme syntaxique de la figure III.11.bis.

L'analyse syntaxique de la déclaration d'une variable symbolique suit le diagramme de la figure III.12 explicitée par les figures III.13, III.14, III.15.

Par ailleurs l'analyseur vérifiera que la variable a bien été déclarée booléenne et qu'aucun sous-type ne lui a déjà été affecté.

En cas de déclaration incorrecte de sous-type, l'analyseur se restaure sur la fin de la déclaration symbolisée par un (;) et tous les éléments de la déclaration seront ignorés.

D/ - MODULE INTERPRETATION

L'analyseur vient de rencontrer le mot réservé INTERPRETATION et il cherche soit le mot réservé COND, soit le mot réservé ACTION.

Lorsque la description s'est avérée correcte, les conditions logiques et les actions associées aux transitions sont rangées dans un tableau (TAB - EL) . Les actions associées aux places sont rangées dans un autre tableau(A.A.P : Actions Associées aux Places). Ces tableaux sont communiqués à l'automate programmable qui évolue à partir de ceux-ci.

1) Conditions logiques

Après avoir trouvé le mot réservé COND, l'analyseur vérifie que l'identificateur suivant est bien une transition et que celle-ci ne fait pas l'objet d'une redéfinition de la condition logique qui lui est associée, vient ensuite l'analyse de la condition logique proprement dite. Les diagrammes syntaxiques des figures III.13, III.14, III.15 illustrent cette analyse.

Parallèlement à cette vérification syntaxique et sémantique, l'analyseur procède à la traduction de l'expression logique concernée, celle-ci produit une structure en arbre. L'algorithme développé à ce sujet (cf.Annexe 2) effectue l'implantation tabulaire TAB - EL de l'arbre obtenu. L'arbre obtenu n'est pas optimal , l'élaboration d'un algorithme d'optimisation du tableau est en cours.

En cas d'erreur, un message spécifique est envoyé et l'analyseur reprend l'analyse après avoir trouvé la fin de l'expression logique c'est-à-dire le (;).

2) Actions

Après avoir trouvé le mot réservé ACTION, l'analyseur vérifie que l'identificateur suivant est bien de type PLACE ou TRANSITION.

a) Dans le cas d'une action associée à une transition, l'analyse syntaxique s'appuie sur les diagrammes des figures III.19, III.20, III.21 qui marquent notamment la différence de syntaxe lorsqu'il s'agit d'une action portant sur une variable entière ou booléenne.

L'analyse sémantique consiste à vérifier qu'il ne s'agit pas d'une redéclaration d'action associée à la transition concernée et que les variables sur lesquelles portent les actions ont été déclarés et qu'elles ne sont ni des entrées ni des variables symboliques.

Lorsque la description s'est révélée correcte, les variables et le le type d'opérations portant sur celles-ci sont mémorisées sous forme tabulaire TAB - EL directement accessible à l'exécuteur de réseau de Pétri.

b) Dans le cas d'une action associée à une place, la vérification syntaxique repose sur le diagramme de la figure III.22. Quant à l'analyse sémantique, elle consiste à vérifier que les variables engagées représentent des sorties booléennes.

Lorsque la description est correcte, à chaque place est alors associée une liste chaînée d'élément contenant l'adresse de la variable sur laquelle porte l'action, le résultat de l'action (0 ou 1) et un pointeur permettant d'accèder aux autres éléments de la liste.

E/ - DECLARATIONS DE SOUS-RESEAUX

La déclaration de sous-réseaux suit la syntaxe du diagramme de la figure III.23.a. Après avoir trouvé les mots réservés DEBUT RESEAU, l'analyseur vérifie que l'identificateur du sous-réseau ne figure pas déjà dans le tableau des identificateurs (TAB - IDEN). Vient ensuite l'analyse des différents modules du sous-réseau (NOEUDS, ARCS, VAR, INTERPRETATION).

Lorsque l'analyseur rencontre une fermeture de sous-réseaux (symbolisée par les mots réservés FIN RESEAU), il vérifie que le sous-réseau fermé a déjà été déclaré et qu'il a été le dernier à être ouvert, cette vérification se faisant au moyen du tableau TAB - IDEN qui sert alors de pile. Ainsi toute omission de fermeture de sous-réseaux ou enchevêtrement de ceux-ci est systématiquement détectée.

Les erreurs sur les ouvertures et fermetures de sous-réseaux provoquent l'arrêt de l'analyse. En effet, l'analyseur n'ayant pas trouvé la fermeture poursuivrait l'analyse et déclencherait une cascade de messages d'erreurs du fait que l'ouverture du module suivant est ignorée (car la restauration se fait sur le premier (;) rencontré d'où envoi de message d'erreurs car on attend une ouverture de module et autre cascade d'erreurs par la suite car toute la première déclaration a été ignorée).

La fin de la description du réseau global se fait par l'intermédiaire du mot réservé FIN suivi d'un point. Lorsque cette syntaxe n'est pas respectée, un message d'erreur concernant la fin du fichier est envoyé et l'analyse est interrompue.

III.6. CONCLUSION

Nous avons présenté le langage d'entrée de l'automate programmable en soulignant ses caractéristiques, facilitant la structuration des spécifications. Ces possibilités ont été illustrées sur des exemples concrets, les automatismes de sécurité de la base de lancement ARIANE 2 présentés au chapitre I.

La structure de l'analyseur syntaxique et sémantique a été explicitée et nous avons vu que, outre les procédures d'analyse syntaxique et sémantique il comprend une implantation du réseau de Pétri interprété sous formes de tables qui seront transmises à l'automate qui se chargera de son évolution.

L'étude effectuée dans le cadre de l'élaboration du langage et de son analyseur syntaxique et sémantique est bien évidemment non exhaustive et il reste certainement beaucoup de choses à faire pour le rendre plus complet et plus performant.

Une amélioration de ce travail peut être la prise en compte de fusion multiple de transitions, avec toutes les précautions que demande une telle entreprise.

D'autre part un algorithme est en cours d'élaboration qui procèdera à l'optimisation des tableaux et permettra notamment dans le cas d'une expression logique, d'éviter de tester plusieurs fois une même variable, permettant ainsi un gain de temps appréciable lors de l'émulation du réseau de Pétri interprété.

CONCLUSION

Le but principal de ce mémoire a été d'apporter des éléments de solution au problème de la spécification et de la mise en oeuvre d'automatismes logiques complexes. Nous entendons par automatisme logique complexe un ensemble de mécanismes de commande interdépendants les uns des autres. Dans un tel contexte l'utilisation d'un modèle formel et de règles de structuration est absolument nécessaire pour atteindre un degré de confiance suffisant et satisfaire aux contraintes de sûreté imposées par l'environnement industriel des applications considérées.

Après avoir montré les avantages d'une spécification par réseaux de Pétri. Le chapitre I du mémoire souligne le fait que si la seule utilisation de cet outil n'est pas en soi une garantie de structuration, elle ne constitue pas pour autant un obstacle à une telle approche. Quelques éléments de solution sont donnés pour structurer une description fondée sur l'emploi des réseaux de Pétri.

Dans le chapitre II, les grandes lignes des méthodes de validation et d'analyse sont rappelées. L'utilisation des approches classiques dans le contexte d'une spécification structurée est étudiée et il est montré que la validation peut être poussée jusqu'à un niveau de détail suffisant pour permettre la vérification de certaines contraintes technologiques.

Les concepts et démarches présentés dans les chapitres I et II sont illustrés à partir d'un exemple réel, celui des automatismes de sécurité de l'étage cryogénique de la base de lancement ARIANE 2.

Le travail de spécification correspondant a été fait dans le cadre d'un contrat de recherche avec le Centre National d'Etudes Spatiales.

La validation détaillée d'une spécification formelle n'a de sens que si l'on dispose de moyens automatiques ou quasi-automatiques de mise en oeuvre à partir de cette spécification. L'utilisation d'automates programmables est une solution à condition qu'il accepte un langage d'entrée adapté. Le chapitre III présente un langage permettant directement la description structurée d'un ensemble d'automatismes spécifiés

par réseaux de Pétri interprétés. Un analyseur syntaxique et sémantique a été écrit. Son rôle est en outre de produire un ensemble de tables correspondant à la spécification, l'automate émulant alors directement à partir de ces tables les réseaux de Pétri interprétés en étant piloté en temps réel par les entrées provenant du procédé industriel.

Notre travail est donc à replacer dans l'ensemble du programme de recherche concernant les automates programmables de l'équipe PASTELS du LAAS-CNRS. Il peut être prolongé de diverses manières.

Une première limitation du langage DESY/2, présenté au chapitre III concerne le cas de systèmes où l'on rencontre un certain nombre d'automatismes tous semblables. A l'heure actuelle la description doit être répétée autant de fois qu'il est nécessaire. La création d'un type RESEAU indicé serait intéressante puisqu'elle éviterait cette procédure répétitive et fastidieuse. Néanmoins, elle pose quelques problèmes. Le premier concerne les variables d'entrées et de sorties car la connaissance de la localisation physique de ces variables est nécessaire pour le câblage des liens entre le procédé industriel et les plaques d'entrées/sorties de l'automate programmable. Si l'affectation des variables aux registres de sortie est faite automatiquement il faudra prévoir la génération d'un plan de câblage correspondant mais l'inconvénient sera que le concepteur ne sera plus maître de la disposition finale de ce câblage. Le second problème concerne les liens entre automatismes. En effet, dans le cas général, s'il est fréquent de rencontrer des ensembles d'automatismes dont la structure interne est semblable, leurs liens avec les autres automatismes (ou les liens entre eux) doivent être différenciés. Les noeuds servant à la communication doivent donc être traités à part pour permettre de ne pas répéter la description des arcs les liant à la structure interne des automatismes tout en différenciant les arcs les liant aux autres automatismes.

Un autre prolongement concerne la partie sémantique de l'analyseur syntaxique et sémantique. En effet, ce logiciel se contente de vérifier que le réseau de Pétri ne comporte pas de conflit de type (arc liant des noeuds de même type, action portant sur des entrées,...). La console de programmation peut être utilisée pour développer des procédures d'analyse des propriétés de réseaux simples mais intéressantes au niveau des

automatismes logiques. Ceci permettrait d'éviter dans la plupart des cas l'utilisation d'un centre de calcul important pendant l'étape de validation. De plus, il serait possible de simuler les automatismes dans la console de programmation avant leur chargement dans l'automate programmable. Ces travaux sont en cours et en particulier le dernier point a été ébauché en collaboration avec la RNUR-DAST au cours d'un contrat de recherche [VAL 82]. D'autre part un prototype d'automate programmable est en voie d'achèvement.

ANNEXES.

ANNEXE 1

DEFINITION

(pour plus ample information, une présentation détaillée peut être trouvée dans la littérature [BRA 80, PET 81, BRA 82, VAL 80]).

Un réseau de Pétri est un quintuplet

$\mathcal{P} = \langle P, T, Mo, I, O \rangle$ où :

- $P = \{ P1, P2, \ldots, Pm \}$ est un ensemble fini de places,
- $T = \{ t1, t2, \ldots, t_n \}$ est un ensemble fini de transitions,
- Mo est le marquage initial, c'est une fonction de l'ensemble des places dans l'ensemble N des entiers positifs ou nul :

$$M_O : P \longrightarrow N$$

Mo associe à chaque place le nombre de jetons contenus initialement dans cette place.

- I est la fonction "place précédente". C'est une fonction de l'ensemble TxP dans l'ensemble N des entiers positifs ou nuls :

$$I : T \times P \longrightarrow N$$

La fonction I donne le poids d'un arc liant une place à une transition. La valeur de la fonction I est nulle quand il n'a pas d'arc.

- O est la fonction "place suivante"

$$O : T \times P \longrightarrow N$$

Elle donne le poids d'un arc liant une transition à une place.
Elle prend la valeur zéro s'il n'y a pas d'arc.

Graphiquement on représente les places par des cercles, les transitions par des barres, le marquage par des points appelés marques ou jetons situés à l'intérieur des places. Alors les fonctions I et O se représentent respectivement par des arcs pondérés liant les places aux transitions, et les transitions aux places.

Exemple

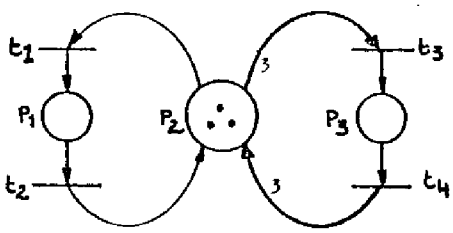


Figure A.1

$$P = \{P_1, P_2, P_3\}$$

$$T = \{t_1, t_2, t_3\}$$

$$Mo = [0 \quad 3 \quad 0]$$

$$I = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad O = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 3 & 0 \end{bmatrix}$$

Règles d'évolution

a) M(p) indique le nombre de jetons contenus dans la place p pour le marquage M. Ce nombre est un entier positif ou nul. Une transition t est sensibilisée par un marquage M si I (t,) M.

b) Une transition t sensibilisée par le marquage M peut être tirée.

Le tir d'une transition est constitué de 2 opérations indivisibles :

. I (t, p) jetons sont enlevés de chaque place p

. O (t, p) jetons sont ajoutés à chaque place p

Si M_{i+1} est le vecteur :

$$M_{i+1} = M_i - I(t,) + O(T,)$$

alors M_i est le marquage obtenu par le tir de la transition t à partir du marquage M_i : $M_i \xrightarrow{t} M_{i+1}$

c) Soit S une séquence finie de transitions : $t_i, t_{i+1}, \ldots, t_{i+k}$;

S est une séquence de tir à partir du marquage M_i si l'on peut trouver des marquages $M_{i+1}, M_{i+2}, \ldots, M_{i+n}$ tels que :

$$M_i \xrightarrow{ti} M_{i+1}, \quad M_{i+1} \xrightarrow{ti+1} M_{i+2}, \ldots, M_{i+n} \xrightarrow{ti+n} M_{i+n+1}$$

On peut écrire : $M \xrightarrow{s} M_{i+n+1}$

d) On appelle ensemble des marquages accessibles à partir du marquage initial M_o : $\vec{M}_o$, l'ensemble définit comme suit :

$$M_i \in \vec{M}_o \Leftrightarrow \exists s / M_o \xrightarrow{s} M_i$$

e) Si $\vec{M}_o$ est un ensemble fini, on peut lui associer un graphe appelé graphe des marquages accessibles $\vec{GMo}$, défini comme suit :

$\vec{GMo}$ = (X, U) avec :

- X = $\vec{Mo}$ est l'ensemble des sommets
- U = ensemble des arcs (M_i, M_j) tels que $\exists t_i \in \mathcal{P}$ et $M_i \xrightarrow{ti} M_j$

Remarque : le graphe $\vec{GMo}$ est la machine à états correspondant au réseau de Pétri.

Exemple : Reprenons l'exemple de la figure A.1 :

Mo = [0 3 0]

le tir de la transition t_1 sensibilisée par Mo donne :

[0 3 0] - [0 1 0] + [1 0 0] = [1 2 0]

le tir de t3 depuis Mo donne :

[0 3 0] - [0 3 0] + [0 0 1] = [0 0 1]

Le graphe est le suivant:

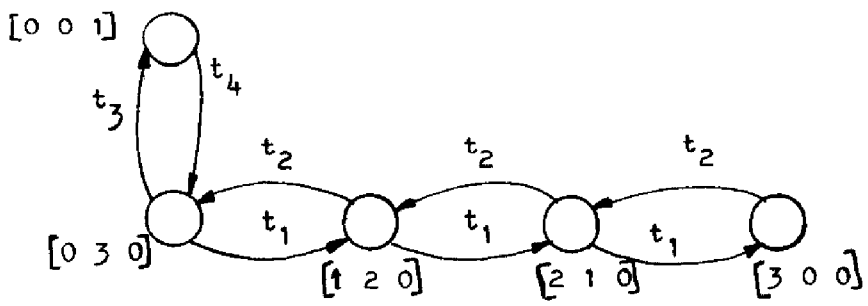


Figure A.2

ANNEXE 2

. ALGORITHME D'EVALUATION DES EXPRESSIONS LOGIQUES

Cet algorithme effectue parallèlement à l'analyse syntaxique et sémantique, l'implantation sous forme d'arbre des expressions logiques décrites à l'aide du langage (voir diagrammes syntaxiques correspondants aux expressions logiques).

L'avantage d'une telle structure (en arbre) est qu'elle permet un gain de temps considérable lors de l'émulation du réseau de Pétri interprété car on ne teste pas nécessairement toutes les variables de l'expression logique. En effet si l'on considère une expression logique de la forme X1 OU X2 ET X3 ; lors du test de la variable X1, si celle-ci est vraie, l'évaluation s'arrête et l'expression logique est vraie (on a ainsi économisé le test de deux variables). Néanmoins, si l'expression logique a été donnée sous forme non factorisée, une même variable peut avoir à être testée plus d'une fois. Un module d'optimisation évitant ce problème, du moins dans les cas simples, est en cours d'écriture.

Structure des données :

Les expressions logiques sont stockées dans le tableau TAB-EL et à chaque variable il correspond 4 colonnes définies comme suit :

- TAB-EL[I]. ADR : contient à la ligne I l'indice de la variable dans le tableau des identificateurs (TAB-IDEN).
- TAB-EL[I]. VRAI : contient à la ligne I l'adresse dans TAB-EL de la variable suivante à tester (où le cas échéant un indice (0) de fin d'évaluation) dans le cas où la variable testée est VRAIE.
- TAB-EL[I]. FAUX : contient à la ligne I l'adresse dans TAB-EL de la variable suivante à tester (ou le cas échéant un indice (-1) de fin d'évaluation) dans le cas où la variable testé est **FAUSSE**
- TAB-EL[I]. CLE : contient à la ligne I la "clé" de la variable testée : variable, variable symbolique, prédicat.

. Dans le cas de prédicat et variable symbolique, la colonne TAB-EL . ADR contient directement l'adresse dans TAB-EL où est stocké le prédicat ou la sous-expression logique correspondant à la variable symbolique.

. Le traitement des variables et prédicats complémentés se fait comme celui des variables non complémentées, simplement en fin d'évaluation les adresses comprises dans les colonnes TAB-EL.VRAI et TAB-EL.FAUX sont interverties.

L'établissement du tableau TAB-EL est possible grâce à la gestion de trois piles : P_V, P_F, P_P, qui permettent l'élaboration de l'adresse de la variable suivante à tester. Grâce à ces trois piles l'agorithme de traitement peut être fait symbole par symbole sans aucun retour en arrière (dans le fichier) lors de la lecture de l'expression logique.

Au fur et à mesure que l'expression logique est scrutée, les éléments rencontrés (variables, parenthèses, complémentation (PAS))sont rangées de manière spécifique dans ces trois piles et à chaque opérateur rencontré (ET/OU) un traitement particulier de chacun de ces trois piles est entrepris et ainsi que le remplissage des colonnes TAB-EL . VRAI et TAB-EL . FAUX de TAB-EL.

Dans la pile P_P sont rangés les éléments suivants : "(" , ") " , "PAS". Elle sert à vérifier que les parenthèses sont utilisées correctement et à délimiter la portée de ces symboles sur des variables stockées dans les piles P_V et P_F.

Les piles P_V et P_F contiennent l'adresse dans TAB-EL des variables rencontrées et ainsi que les ouvertures de parenthèses : "(". Ces dernières servant à délimiter les variables à effacer dans ces piles. Deux piles sont nécessaires car les règles pour déduire les éléments suivants dans l'arbre de décision sont différentes dans le cas où la variable testée est vraie et lorsqu'elle est fausse.

Au fur et à mesure que pour une variable testée, l'adresse de la variable suivante à tester est déterminée et inscrite dans TAB-EL (colonne TAB-EL . VRAI et TAB-EL . FAUX correspondant à la variable testée), celle-ci disparaît dans la pile P_V ou P_F suivant que la variable testée

est vraie ou fausse.

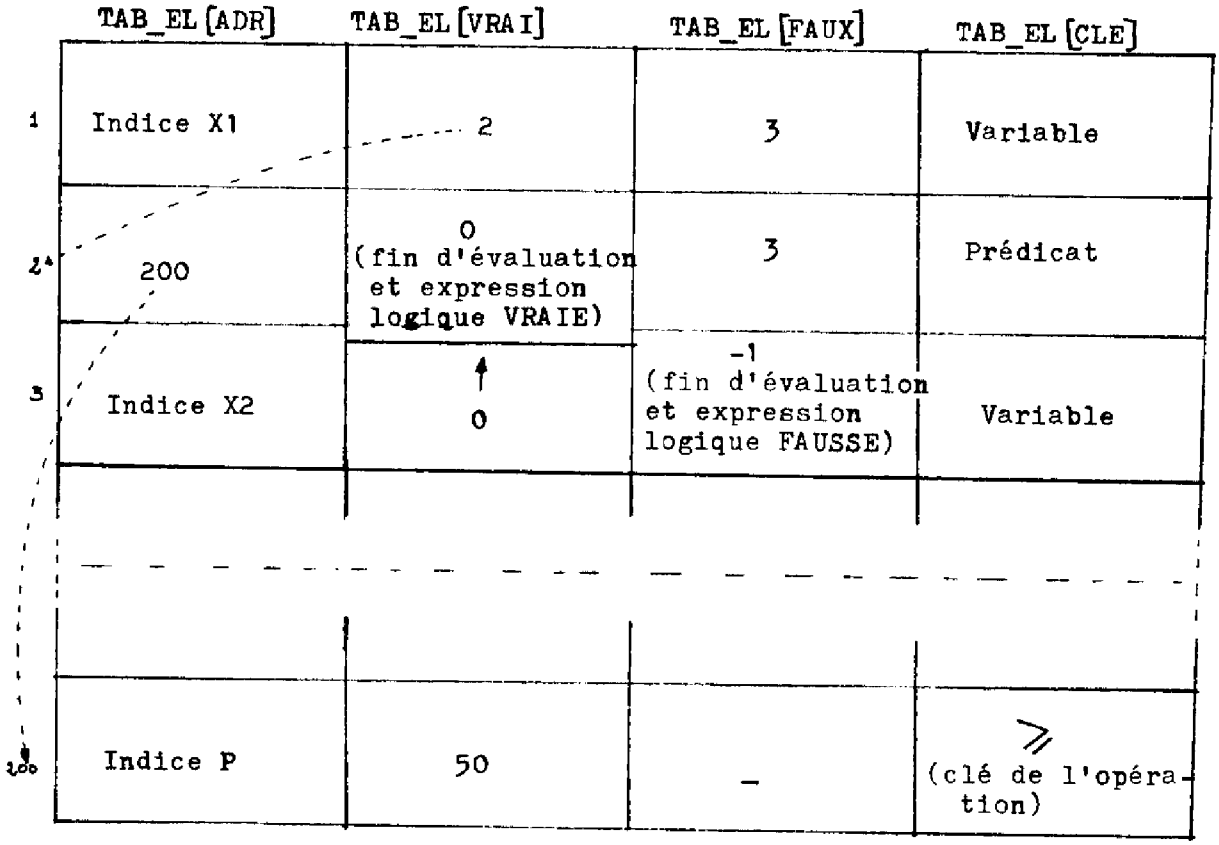
Il en est de même pour la pile P_P ; au fur et à mesure que toutes les variables sur lesquelles portent les parenthèses ou la complémentation sont déterminées, il y a effacement dans la pile des symboles correspondants ('PAS', '(', ')').

Exemple :

Considérons l'expression logique suivante :

X1 ET (P⩾50) OU X2 ;

L'évaluation de cette expression logique donnerait le rangement suivant dans le tableau TAB-EL :



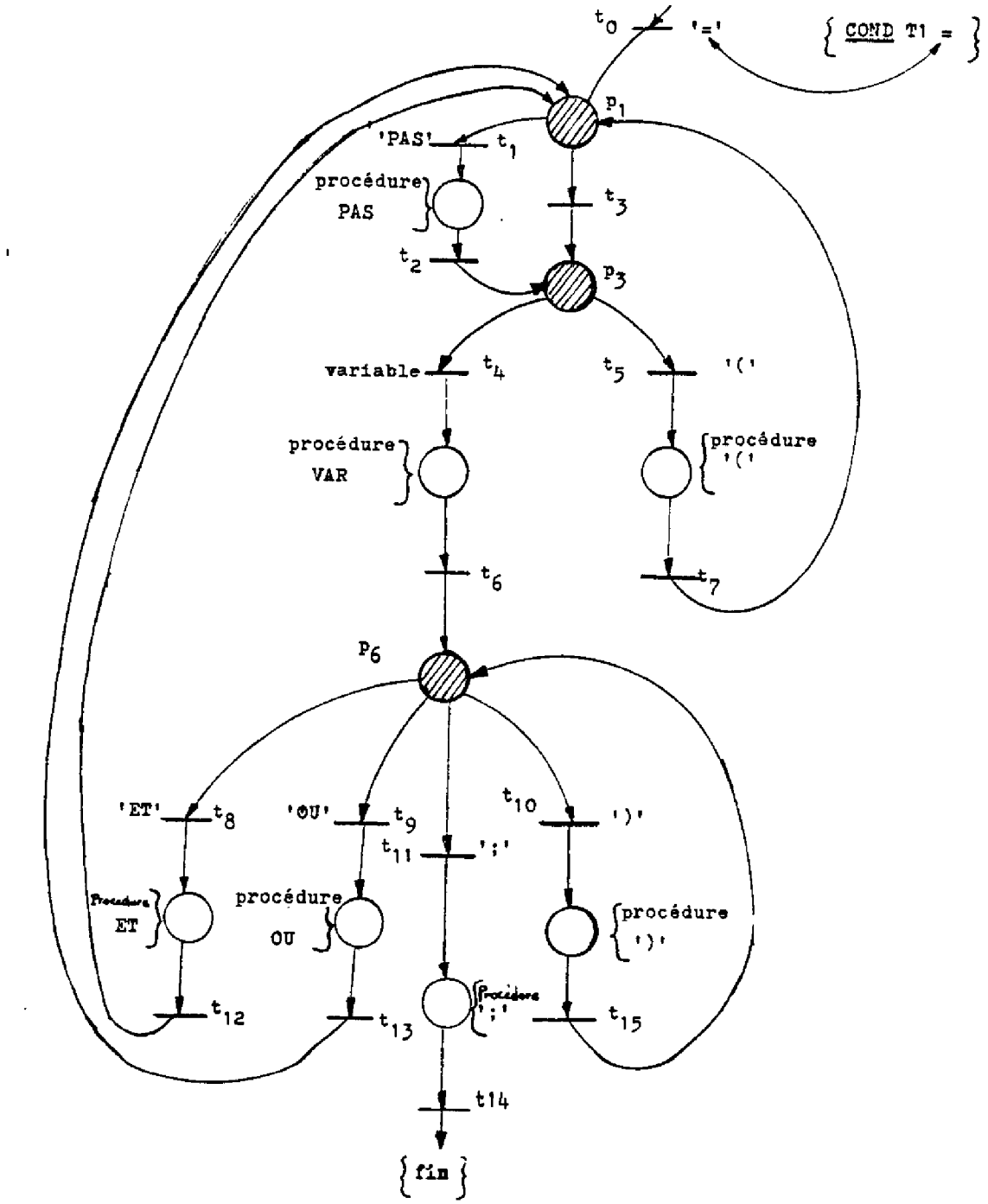


Figure A.3 - Structure de commande de l'algorithme

Structure de commande :

Contrairement aux diagrammes syntaxiques des expressions logiques (voir chapitre III), la structure de commande n'est pas récursive ; elle est donnée par le réseau de Pétri de la figure A.3 qui est complètée par une gestion des piles P_P, P_V, P_F qui aide notamment à l'analyse syntaxique. Aux transitions sont associées les éléments susceptibles d'être rencontrés (en respectant la syntaxe) dans une expression logique ; aux places les traitements correspondants.

Comme l'indique le réseau de Pétri, pour chaque élément de l'expression logique rencontré, un traitement spécifique est effectué. Nous allons donner, en suivant l'ordre croissant des places du réseau de Pétri, l'algorithme correspondant à chaque procédure. Il convient d'abord d'expliciter les notations utilisées et de présenter les procédures de base :

Notations :

- I représente l'adresse des variables dans le tableau TAB-EL.
- Tête (Pile) représente le dernier élément rangé dans la pile dont le nom est entre paranthèses.
- $\overline{\text{Mode}}$ et mode servent à mémoriser le fait qu'une expression est complémentée ou non.
- TAB-EL [I].CLE représente la colonne CLE dans TAB-EL à l'adresse I, il en est de même pour les colonnes VRAI et FAUX.

Procédures de bases :

EFFACE P.P ; {procédure d'effacement dans la pile P.P}

k : = 0;

- Tant que (tête(P-P) = ")") ou (tête (P_P) = "PAS")
  - si tête (P_P) = "PAS" faire mode : = $\overline{\text{mode}}$;
  - si tête (P_P) = ")" faire k : = k+1
  - effacer tête (P_P)
- Effacer k fois tête (P_P)
- Si tête P_P = "PAS" faire
  - mode : = $\overline{\text{mode}}$;
  - effacer tête (P_P)

ET/OU - 1 ; {procédure commune aux procédures ET et OU}

- Tant que tête (P_V)<> "(" faire
  - TAB-EL [tête (P-V)] . V : = I + 1
  - effacer tête (P-V)
- Effacer tête (P-V) ;
- Effacer le premier "(" rencontré dans P-F

ET/OU - 2 ; {procédure commune aux procédures ET et OU}

- TAB-EL [tête (P-V)].V : = I + 1
- Effacer (P-V) ;

ET/OU - 3 ; {procédure commune aux procédures ET et OU}

- Effacer le premier "(" rencontré dans la pile P-V
- Si P-P est vide faire
  - TAB-EL [tête (P-V)] . V : = O ; indice d'arrêt d'évaluation expression logique VRAIE
  - effacer tête (P-V) ;
  - jusqu'à ce que P-V soit vide
- Effacer le premier "(" rencontré dans P-F ;
  tant que tête (P-F)<> "(" faire
  - TAB-EL [tête (P-F)] . F : = I + 1 ;
  - effacer tête (P-F)

ET/OU - 4 ; {procédure commune aux procédures ET et OU}

- si la pile P-P est vide faire TAB-EL[tête (P-V)] . V : = O ;
- tant que tête (P-F) <> "(" faire
  - TAB-EL [tête (P-F)]. Faux : = I + 1 ;
  - effacer tête (P-F) ;

Procédures principales :

Place P2 : { on a rencontré l'opérateur PAS }

Procédure PAS ;

. On range "PAS" dans la pile P-P ;

. On inverse le "mode" (mode : = $\overline{\text{mode}}$) ;

Place P4 : { on a rencontré une variable dans l'expression logique }

Procédure VAR ;

- I : = I + 1 ;
- I est mis dans les piles P-V et P-F ;
- TAB-EL[I]. ADR : = indice dans TAB-IDEN de la variable rencontrée ;
- Si $\overline{\text{mode}}$ faire
  - si variable symbolique alors TAB-EL I clé : = 1
  - si non TAB-EL [I] clé : = v
- si non
  - si variable symbolique TAB-EL [I] clé : = S
  - si non TAB-EL[I] clé : = V
- Si tête (P-P) : 'PAS' faire
  - mode : = $\overline{\text{mode}}$;
  - efface tête (P-P)

Place P5 : { on vient de recontrer une ouverture de parenthèse }

Procédure OUV-PARENTHSE

- mettre "(" dans la pile P-P ;
- mettre "(" dans la pile P-V ;
- mettre "(" dans la pile P-F ;

Place P7 : { on vient de rencontrer l'opérateur ET }

Procédure ET ;

- si tête (P-P) = ")" faire | EFFACE - PP ;
  | Si mode procédure ET/OU - 1 sinon ET/OU -3
- si non faire | si mode alors procédure ET/OU - 2
  | si non procédure ET/OU - 4

Place P8: { on vient de rencontrer l'opérateur OU }

Procédure OU ;

- Si tête (P-P) = ")" faire | EFFACE - PP ;
  | Si mode alors procédure ET/OU -3
  | si non ET/OU -1
- Si non | Si mode alors procédure ET/OU - 4
  | sinon porcédure ET/OU - 2

Place P9 : { on vient de rencontrer une fermeture de parenthèse }

Procédure FERM PARENTHESE ;

- Si tête (P-P) = ")" effacer la première ouverture de parenthèse ("(") rencontrée dans les piles P-V et P-F
- Mettre ")" dans la pile P-P

Place P10 : {on vient de rencontrer le point-virgule qui marque la fin de l'expression logique (E.L)}

Procédure POINT VIRGULE ;

- Si tête (P-P) = ")" faire
  - EFFACE - PP ;
  - Effacer la première "(" rencontrée dans P-V et P-F
- Tant que tête (P-V) <> "(" faire
  - TAB-EL [tête (P-V)].V : = 0 ; {zéro indique de l'avaluation s'arrête et que l'E.L est VRAIE}
  - Effacer tête (P-V)
- Tant que tête (P-F) "(" faire
  - TAB-EL [tête (P-F)]. F : = - 1 ; {-1 indique de l'évaluation s'arrête et que l'E.L est fausse}
  - Effacer tête (P-F) ;
- Vérifier que les piles P-P, P-V, P-F sont vides et que mode = vrai

ANNEXE 3

CODE D'ERREURS DU PROGRAMME D'ANALYSE SYNTAXIQUE ET SEMANTIQUE

ERREUR 1 : fin de fichier inattendue :

mot clé FIN
- oublié
- non reconnu à cause d'une erreur précédente (par exemple oubli de fermeture d'un commentaire ou bien recherche d'un caractère au cours d'une restauration).
- non suivi du caractère '.' (point final).

ERREUR 2 : Erreur dans un commentaire :

- oubli du caractère '}' qui ferme le commentaire ou (;) dans un commentaire (en effet le (;) n'est pas autorisé dans un commentaire).

N.B : Le numéro de ligne apparaissant n'est pas forcément la ligne où débute le commentaire.

L'analyseur ne repart qu'après avoir trouvé le caractère "}".

ERREUR 3 : Séparateur erroné :

- Le symbole lu ne correspond pas à la syntaxe de déclaration du réseau. Il est signalé et le symbole attendu est affiché.
- La restauration se fait sur le prochain point-virgule (;).

ERREUR 4 : Caractère illégal au début d'un identificateur :

- Le premier caractère d'un identificateur ne peut être ni un chiffre, ni une virgule, ni un point-virgule (ce doit être un ASCII (64,122) : voir III.4.1
- Dans le cas où le caractère illégal est ";" ou ",", l'analyseur se restaure aussitôt.

ERREUR 5 : Utilisation incorrecte d'un identificateur ou erreur de type.

- La restauration se faisant sur le premier (;) rencontré dans le cas d'une ouverture de module, celuici sera ignoré.
- Lorsqu'un identificateur non déclaré est rencontré on lui affecte le type Z.

ERREUR 6 : Erreur de parenthèse dans une expression logique.

ERREUR 7 : Caractère illégal. Opérateur attendu dans un prédicat.

ERREUR 8 : Type inattendu dans une expression logique.

N.B : Pour toutes les erreurs concernant une expression logique, la restauration se fait sur le (;) qui en marque la fin.

ERREUR 9 : Identificateur illégal dans une séquence d'actions

ERREUR 10 : "Type erroné→X, type Y attendu"

ERREUR 11 : Poids erroné pour un arc (négatif ou nul) :

. le nom de la place à laquelle est relié l'arc apparaît sur la console et l'analyseur repart en attribuant un poids égal à 1.

ERREUR 12 : Arc ou action ou condition logique redéfini

. Les actions, conditions logiques et arcs (sauf au moyen de chemins) ne doivent pas être déclarés en plusieurs reprises.

ERREUR 13 : Erreurs relatives à l'utilisation de sous-type.

. Sémantique incorrecte : soit le sous-type est erroné (notamment dans les actions associées aux places et aux transitions) soit il s'agit d'une attribution de sous-type à une variable qui en a déjà un.

ERREUR 14 : Erreur relative à la déclaration d'une variable symbolique.

. Une variable symbolique doit être de type booléen (B) et aucun sous-type n'a dû lui être attribué.

ERREUR 20 : Relative aux oublis de fermeturesou ouvertures illégales de sous-réseaux.

. L'analyseur s'arrête (pas de restauration).

ERREUR 100 : Nombre de places supérieur à MAXP (100)

ERREUR 101 : Nombre de transitions supérieur à MAXT (100)

ERREUR 102 : Nombre d'identificateurs supérieur à MAXI (400)

ERREUR 103 : Nombre d'arcs supérieur à MAXE (400)

ERREUR 104 : Relatif au dépassement du nombre de caractères (20) dans les piles P-P, P-V, P-F.

ERREUR 105 :
- Nombre d'entiers supérieur à MAXV2 (100)
- Nombre de variables d'interprétation supérieur à MAXTAB (200)
- Nombre de booléens supérieur à MAXV1 (100)
- Nombre d'entrées, de sorties, de temporisations et variables internes supérieur à MAXES (200)

Remarque : Pour toutes les erreurs citées ci-dessus (dépassement de tableau) il n'y a pas de restauration. L'analyse est interrompue.

ERREUR 999 / 998 : L'analyseur se trouve dans un état incohérent suite à une erreur dans une expression logique.

Exemple : Reprenons la description de la synchronisation entre les automatismes PX et AY donnée en III.4.4.4) et correspondant au réseau de Pétri de la figure I.29 : supposons que dans un moment d'inattention l'utilisateur ait oublié de déclarer la place P2 de AY, redéclaré la variable MES comme sortie et oublié de fermer le réseau PX.

Avant de donner les messages envoyés par l'analyseur syntaxique il convient de noter que, outre les messages d'erreurs, les noms des modules et des sous-réseaux apparaissent au fur et à mesure qu'ils sont analysés.

```
BONJOUR! ICI DESY2 VERSION Z81/10
{ Description de la synchronisation entre PX et AY }
RESEAU [PX_AY]
RESEAU AY
NOEUDS
ARCS
*****
ERREUR 5
LIGNE :          12
DERNIER IDENTIFICATEUR LU :P2
IDENTIFICATEUR NON DECLARE - TYPE Z
** RC **

*****
ERREUR 5
LIGNE :          13
DERNIER IDENTIFICATEUR LU :P2
IDENTIFICATEUR NON DECLARE - TYPE Z
** RC **

VAR
*****
ERREUR 13
LIGNE :          26
DERNIER IDENTIFICATEUR LU :MES
SOUS-TYPE DEJA DEFINI : E
** RC **

INTERPRETATION
*****
ERREUR 10
LIGNE :          58
DERNIER IDENTIFICATEUR LU :P2
TYPE Z ERRONE _ TYPE P OU T ATTENDU
** RC **

RESEAU PX
NOEUDS
ARCS
VAR
INTERPRETATION
*****
ERREUR 20
LIGNE :          98
DERNIER IDENTIFICATEUR LU :[PX_AY]
RESEAU PX : NON FERME OU RESEAUX ENCHEVETRES
*****
          9 PLACES,          20 TRANSITIONS

FICHIER DE LISTAGE?
PX_AY.LST
SAUVEGARDE
FIN   AU REVOIR
```

BIBLIOGRAPHIE.

ARI 81 : M. COURVOISIER, R. VALETTE, J. ALBUKERQUE : "Etude de la spécification et de l'analyse d'automatismes logiques à l'aide des réseaux de Pétri", Rapport Final Convention CNES N 80/CNES 3542, Janvier 1981.

BER 77 : G. BERTHELOT : "Une méthode de vérification des réseaux de Pétri" Journée d'Etude AFCET sur les réseaux de Pétri, PARIS 1977.

BER 78 : G. BERTHELOT : "Vérification des réseaux de Pétri". Thèse de troisième cycle. Université de PARIS VI. 1978.

BRA 80 : W.BRAUER : "Net theory and application". Proceding of the advanced course on general theory of processes and systems. Hamburg, Lectures Notes in Computer Science n°84, Springer Verlag. 1980.

BRA 82 : G.W BRAMS (C. ANDRE, G. BERTHELOT, G. GIRAULT, G. MEMMI, G.ROUCAIROL, J.SIFAKIS, R. VALETTE, G. VIDAL-NAQUET) : "Réseaux de Pétri, Théorie et Pratique" Edition MASSON EAP 1982.

HOL 70 : A. HOLT, F. COMMONER : "Events and conditions. Record of the Project MAC conference on concurrent systems ans Parallel computation, June 1970, pp3.33

KAR 69 : R.M. KARP, R.E MILLER : "Parallel program schemata". Journal of computer and system sciences, Vol.3, 1969, pp147-195.

LAU 74 : K. LAUTENBACH, H.A SCHMID : "Use of Petri Nets for proving correctness of concurrent process systems". IFIP Congress 1974 pp 187-191.

PET 62 : C.A PETRI : "Communication with automata". Supplement 1 to technical report RADC-TR-65-377, vol.1, New-York, 1966. Traduit de "Kommunikation mit Automaten". Université de BONN, 1962.

PET 79 : C.A. PETRI : "Introduction to général net théory - Net theory and applications". Lecture Notes in Computer Science n°84. Springer Verlag 1979, pp 1-19

PET 81 : J.L PETERSON : "Petri net and the modelling of systems". Prentice Hall Inc. Englewood Cliffs ISBN 0-13-66 1983-5, 1981.

PRA 79 : B. CHEZALVIEL - PRADIN : "Un outil graphique interactif pour la vérification des systèmes à évolutions parallèles décrits par réseaux de Pétri". Thèse de Docteur Ingénieur. Université Paul Sabatier, Toulouse, 1979.

SIR 82 : P. BOURNAI : "Un système de CAO pour la simulation des systèmes physiques contrôlés par réseaux de Pétri". Thèse de Docteur Ingénieur. Institut National des Sciences Appliquées de RENNES, 1981.

VAL 76 : R. VALETTE : "Sur la description, l'analyse et la validation des systèmes de commande parallèles. Thèse de Doctorat es-sciences, Toulouse, Novembre 1976.

VAL 80 : R. VALETTE, M. COURVOISIER : "Systèmes de commande en temps réel : Description, Analyse et Réalisation". Editions SCM, PARIS 1980.

VLM 81 : R. VALETTE; G. LATAPIE, D. MAYEUX : "Possibilités et limites des réseaux de Pétri dans le cadre des systèmes à haute sûreté", Journées d'Etudes AFCET, Validation et Spécification, PARIS, Mars 1981.

WEI 80 : C. WEITZMAN : "Distributed micro/minicomputer systems. Structure, Implementation and application". Prentice Hall, Inc, 1980.

VAL 82 : R. VALETTE : "Ecriture d'un simulateur à évènements discrets basé sur l'utilisation des réseaux de Pétri". Rapport du contrat 03.03.21/L.A.A.S.11 avec le G.I.E RENAULT RECHERCHE INNOVATION. Juillet 1982.

X 78 : Revue de Projet des automatismes de l'étage cryogénique H8 du lanceur ARIANE. 18 Octobre 1978, n°369 DLA/EIS/01. Direction des lanceurs.

°0°

TABLE DES MATIERES

Thèse de Monsieur Joseph ALBUKERQUE

« Spécification et validation d'automatismes logiques interconnectés »

RESUME

Ce mémoire est composé de trois chapitres. Le premier pose le problème de la spécification des systèmes de commande complexes formés d'un ensemble d'automatismes communicants. Après avoir introduit les réseaux de Petri en tant qu'outil formel pour la spécification, il est montré que cet outil n'est pas contradictoire avec une approche structurée. Quelques règles de structuration sont proposées. Cette démarche est illustrée par un exemple concret. Le second chapitre montre comment une spécification structurée peut être validée. Le troisième chapitre propose un langage de spécification adapté à la description structurée d'automatismes interconnectés. Ce langage est fondé sur l'utilisation des réseaux de Petri. Un logiciel d'analyse syntaxique et sémantique a été développé sur microcalculateur en langage PASCAL. Ce logiciel traduit la spécification en tables et est conçu de façon à permettre le téléchargement d'automates programmables spécialisés.

MOTS CLES : Spécification, Validation, Automatismes logiques, Réseaux de Petri.

ABSTRACT

This thesis has three chapters. The first one presents the specification problem of complex control systems which comprise a set of communicating automata. After having introduced Petri nets as a formal specification tool, it is shown that the use of this tool is compatible with a structured approach. Some structuration rules are proposed. This step is illustrated by a concrete exemple. The second chapter shows how to validate a structured specification. The third chapter proposes a specification language for the structured description of interconnected control automata. The language is based on Petri nets. A syntactic and semantic analysis program has been developed in PASCAL on a micro-computer. This program translates the specification into tables and it allows the loading of Petri net based programmable logic controllers.

KEY WORDS : Specification, Validation, Automata, Petri nets.