THESE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

pour obtenir le grade de

DOCTEUR TROISIEME CYCLE

Spécialité : Informatique

par

Christian VERCAUTER

maitre ès sciences

SUR

UN ENSEMBLE D'OUTILS D'AIDE A LA SPECIFICATION ET A LA CONCEPTION DES SYSTEMES INDUSTRIELS

Soutenue le 5 Juillet 1982 devant le Jury d'examen

Président	M. V.CORDONNIER	Professeur
Rapporteurs	M. D.CORBEEL	Maitre Assistant
	M. J-C.GENTINA	Professeur
Examinateurs	M. P.BORNE	Professeur
	M. M.LATTEUX	Maitre Assistant
Invité	M. B.DOLLE	

AVANT PROPOS

Le travail présenté dans ce mémoire a été effectué au Laboratoire d'Informatique Industrielle de l'INSTITUT INDUSTRIEL DU NORD et au Laboratoire de Systématique de l'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE I.

Je tiens tout d'abord, à remercier très vivement Monsieur CORDONNIER, Professeur à l'Université de Lille I, d'avoir accepté de présider mon Jury de Thèse. Qu'il trouve ici, le témoignage de ma respectueuse gratitude.

Les travaux présentés ici, ont été menés sous la Direction de Messieurs CORBEEL, Maître-Assistant à l'Institut Industriel du Nord, et GENTINA, Professeur à l'I. D. N. et Directeur du Laboratoire d'Informatique Industrielle de l'I. D. N.. Par leurs conseils et leur amitié, ils ont su m'initier à la recherche et guider mon travail. Très conscient de ce qu'ils m'ont apporté, je les remercie très sincèrement.

Je suis très reconnaissant à Monsieur BORNE, Professeur à l'Institut Industriel du Nord et à Monsieur LATTEUX, Maître-Assistant à l'Université des Sciences et Techniques de Lille I, pour l'honneur qu'ils me font en participant à mon Jury de Thèse.

Je suis également très flatté de la présence à ce Jury, de Monsieur DOLLE, Ingénieur à la R. N. U. R.. C'est un agréable devoir pour moi de l'en remercier ici.

Qu'il me soit permis de rendre hommage à tous les techniciens, chercheurs du laboratoire d'Informatique Industriel de l'I. D. N. pour l'aide précieuse qu'ils m'ont apportée tant sur le plan scientifique que sur le plan humain.

Enfin, je tiens à remercier les personnes qui, avec compétence, ont assuré la réalisation matérielle de ce mémoire : Madame TRICOT pour la dactylographie, Madame DELTOUR et Monsieur SOYEZ, pour la reprographie.

SOMMAIRE

INTRODUCTION GENERALE

CHAPITRE I: PROBLEMES POSES PAR L'ETUDE DES SYSTEMES INDUSTRIELS

CHAPITRE II: METHODOLOGIE DE SPECIFICATION ET DE CONCEPTION DES PROCESSUS INDUSTRIELS

CHAPITRE III: VALIDATION DES SYSTEMES DECRITS PAR DES RESEAUX DE PETRI

CONCLUSION GENERALE

INTRODUCTION GENERALE

Depuis quelques années, on constate dans le domaine de la réalisation des automatismes logiques, une évolution vers des méthodes programmées. Les principales raisons de cette évolution sont les suivantes :

- le besoin des industriels de systèmes de commande de plus en plus complexes et performants permettant de rentabiliser au maximum des chaînes de fabrication (ex : atelier et machines flexibles), que seuls les systèmes programmables permettent d'appréhender correctement tout en assurant une bonne sûreté de fonctionnement,

- le développement extraordinaire de la technologie électronique offrant sur le marché une large gamme de composants puissants et fiables et par voie de conséquence, la diminution du coût de production des systèmes programmables.

Cette évolution s'est faite en deux temps :

- depuis l'apparition des premiers systèmes programmables spécialisés dans le traitement des problèmes logiques (automates programmables) les méthodes de synthèse avaient toutes un même objectif :

Transposer les schémas de réalisation cablée en un programme.

- et depuis quelques années, une deuxième méthode de conception est venue concurrencer la première. Elle se caractérise par le passage quasi-direct de la description formalisée du cahier des charges de l'automatisme à sa réalisation.

Les outils se sont développés (automates programmables incorporant des fonctions de calcul numérique, de régulation analogique, système à microprocesseurs, mini-ordinateur, traitant en plus des problèmes de gestion), et les méthodes de cette "seconde génération" se sont rapprochées de celles utilisées en informatique pour, par exemple, l'étude :

- des systèmes d'exploitation
- d'exécutif temps réel
- des systèmes de gestion de base de données réparties, ...

Le passage du cahier des charges initial à la réalisation du projet s'est affiné, faisant apparaître un certain nombre d'étapes :

- définition, modélisation
- analyse, conception
- implémentation
- maintenance
- test

qui ne sont pas sans rappeler les phases d'études d'un projet typiquement informatique.

L'ensemble des problèmes rencontrés lors de la définition du cahier des charges initial jusqu'à la définition d'un "bon modèle" permettant un codage sûr à l'aide d'un langage structuré de haut niveau a amené de nombreux chercheurs et industriels sur le problème général des spécifications d'un système informatique. Nous entendons ici, par système informatique, un système comprenant à la fois du matériel et du logiciel.

Nous reviendrons sur ces problèmes dans la première partie de ce mémoire en dégageant les éléments essentiels que doit posséder un bon outil de spécification. Ces caractéristiques nous permettront d'analyser quelques méthodologies de spécification et fera ainsi apparaître les points forts et les limites de chacune de ces méthodes.

La seconde partie sera consacrée à l'étude d'un outil de spécification plus particulièrement adapté à l'analyse des systèmes parallèles asynchrones : les réseaux de Pétri. Après avoir rappelé les principales définitions relatives à ce modèle, nous justifierons le choix que nous avons fait en précisant dans quelle mesure ils constituent un "bon outil de spécification". Nous serons, en particulier, amenés à proposer l'utilisation d'une classe particulière de réseaux de Pétri : les réseaux de Pétri structurés.

Cet outil constitue d'ailleurs la base d'une méthodologie de spécification et de conception dont nous donnons ensuite les principales directions.

Toute spécification utilise une démarche descendante, il est donc important de vérifier que chaque étape respecte les propriétés du système (les données et contraintes du cahier des charges). Cette phase nécessaire de validation procède de deux manières complémentaires :

- vérification de la conformité du modèle
- vérification du respect des spécificités du système.

Le troisième volet de ce mémoire permettra d'aborder les différentes méthodes d'analyse, qu'autorise l'utilisation du modèle choisi, c'est-à-dire :

- les méthodes classiques par énumération de marquages,
- les méthodes structurelles qui s'intéressent essentiellement à la structure du graphe,
- les méthodes par réduction dont l'utilisation précède généralement l'application de l'une ou l'autre méthode déjà citée.

Nous nous attacherons plus particulièrement à montrer que l'utilisation de réseaux de Pétri structurés apporte, par rapport à l'utilisation de réseaux de Pétri quelconques, un certain nombre de simplifications qui facilitent la mise en œuvre des méthodes d'analyse.

CHAPITRE I

PROBLEMES POSES PAR L'ETUDE DES SYSTEMES INDUSTRIELS

CHAPITRE I

page

→ I,1.2

PROBLEMES POSES PAR L'ETUDE DES SYSTEMES INDUSTRIELS

INTRODUCTION

Dans le contexte industriel de ces quinze dernières années, les problèmes de rentabilité et de productivité ont été les préoccupations essentielles des milieux industriels.

Pour répondre à ces nécessités, ce sont davantage des besoins de méthodes plutôt que des besoins en matériel (d'ailleurs largement satisfaits par les progrès technologiques en matière de systèmes programmables, par exemple) qui ont été ressentis par l'ensemble des utilisateurs sensibilisés par ces problèmes.

Ces besoins se sont exprimés principalement dans les domaines suivants :

- Spécification de définition,

- Méthodologie d'analyse et de synthèse.

Ils permettent notamment :

- d'aborder correctement l'analyse et la description des systèmes actuels parfois très complexes,

- de réduire les coûts d'étude et de maintenance,

- d'établir un document complet, contractuel entre le client et le maître d'œuvre (utilisateur), qui sera utilisé comme base de travail pour la réalisation du projet.

Nous reviendrons, de façon plus précise, sur chacun de ces aspects dans la première partie de ce chapitre ; ce qui permettra dans un deuxième temps, de dégager les caractéristiques principales que doit posséder un bon outil de spécification, dans le contexte du génie logiciel.

Nous nous intéresserons plus particulièrement à la spécification de logiciels, c'est à dire à la spécification d'un ensemble de programmes implémentant une application sur un support informatique.

La troisième partie de ce chapitre sera consacrée à l'étude et à l'analyse des formalismes de spécification et de conception les plus significatifs :

- Méthode HIPO,
- Méthodes utilisant un diagramme de flux de données,
- Méthode SADT,
- Langage Z.

I.1 - SPECIFICATIONS

I.1.1 - Généralités - Définition

Après avoir été pendant longtemps, un domaine d'étude relativement peu exploré, la spécification est devenue l'un des thèmes de recherche les plus abordés comme le témoigne l'abondance d'articles et de publications parus depuis 1976 |22|, |1|, |2|, |3|, ... Ces études ont effectivement mis en évidence que cette étape représente l'une des plus délicates de la réalisation d'un système.

Les projets qui pouvaient être alors traités par un petit noyau de personnes, d'ailleurs le plus souvent à la fois comcepteur et utilisateur, se traitent désormais, compte tenu de leur complexité, par plusieurs équipes pour lesquelles les problèmes de répartition de tâche, d'interface se posent de manière aigue (interface client/concepteur, utilisateur/concepteur, concepteur/système, interface entre les différentes équipes d'étude, ...).

On se trouve ainsi dans des situations où les intermédiaires entre maître d'œuvre et maître d'ouvrage se multiplient dans un rapport qui peut certes, réduire le temps de développement, mais qui peut également amener certains déboires.

Pour résoudre ces problèmes, il a été nécessaire de spécifier davantage c'est à dire définir précisèment et complétement les objectifs qu'il faut véritablement atteindre, à tous les niveaux d'avancement du projet. C'est pourquoi les termes Spécification de définition, Spécifications internes et externes du système, du logiciel, du matériel, Spécifications fonctionnelles, Spécification de conception globale, détaillée, sont apparus dans la littérature.

I.1.2 - Objectifs atteints par la spécification des systèmes

Une autre motivation qui a suscité l'intérêt des industriels pour la spécification, a été la publication d'études statistiques sur les coûts d'études, de réalisation et de maintenance de systèmes |1|, |14|, |15|, |16|, |19|, |27|, ...

Elles indiquent qu'en particulier, la contribution des coûts de maintenance au coût total d'un système est prédominante par rapport au coût de développement ; ce qui tendrait à penser qu'une réduction du coût total passe par une meilleure fiabilité et disponibilité et par conséquent par la prise en compte au niveau de l'analyse, de contraintes supplémentaires qu'il faut spécifier et vérifier.

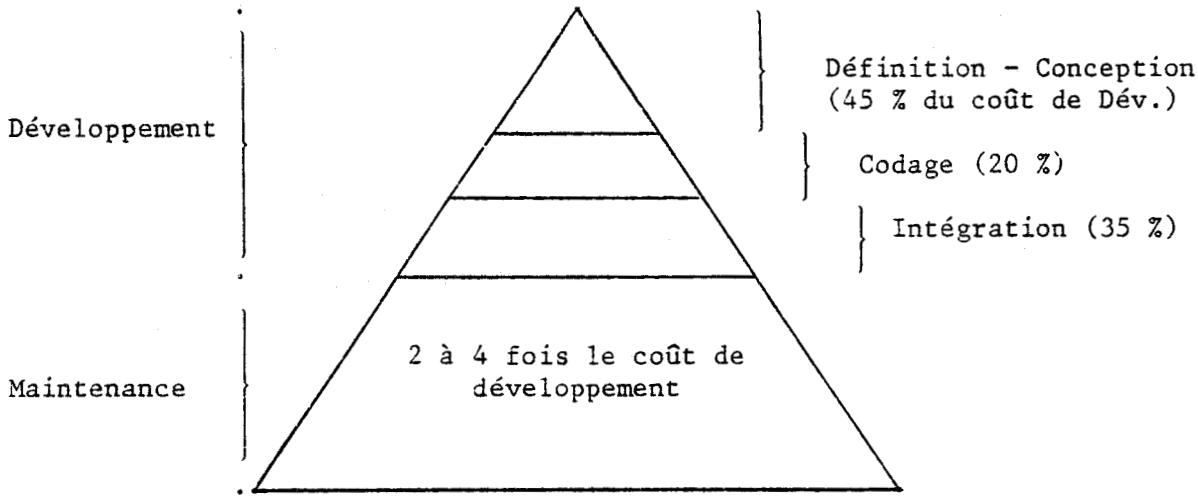


Figure I.1

En effet, la plupart des erreurs sont commises lors de la définition même du système. Ce sont en général les plus graves, dans le sens où elles sont détectées les dernières, parfois lors de l'exploitation, et sont donc, par voie de conséquence, les plus difficiles à corriger et les plus coûteuses |19|.

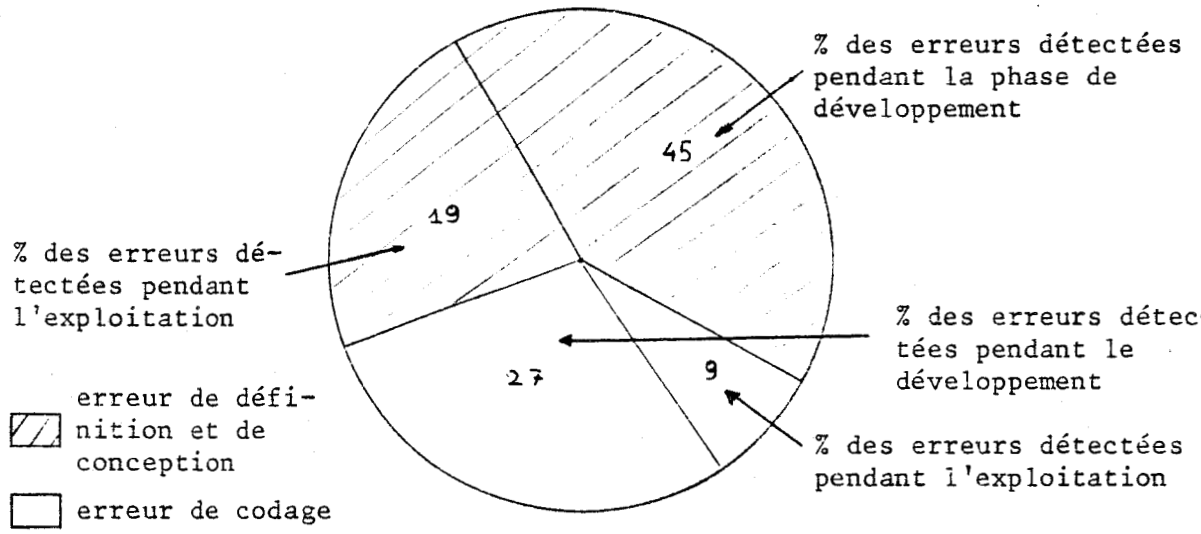


Figure I.2

Pour obtenir une meilleure fiabilité, il est nécessaire que les concepteurs puissent travailler sur un ensemble de documents stable et complet : le cahier des charges.

Toutes les questions, les problèmes (contraintes de tout ordre, conflits, choix, ...) doivent avoir été abordés, résolus et approuvés par les différents partis intéressés ; ceci permet d'établir un document contractuel sur lequel apparaissent les objectifs, les moyens qui permettent de vérifier que les cibles visées ont été atteintes. Il convient par ailleurs, d'y adjoindre des éléments d'ordres économiques et commerciaux (date de livraison, prévision des coûts, formation des utilisateurs, ...). Tous ces éléments ne devront alors plus être sujets à des modifications ultérieures. Dans ce sens, on cherchera à éviter les conséquences désastreuses provoquées par l'effet d'une modification prévue trop tardivement dans l'avancement du projet. Cette modification peut en effet, remettre en cause des parties importantes de l'étude.

Un dernier point essentiel réside dans le fait que de bonnes spécifications peuvent amener une conception de programmes informatiques beaucoup plus efficace. Elles ont permis, en effet, de résoudre tous les conflits, toutes les alternatives, de lever toute ambiguïté permettant ainsi au concepteur de se consacrer exclusivement au codage et à l'implémentation à partir de documents stables et complets, résultat d'un long processus d'analyse.

Les figures I.3 et I.4 montrent d'ailleurs qu'une meilleure définition du problème, qui prend par conséquent un temps d'étude plus long, amène une phase de conception plus courte et une réduction substancielle de la durée totale de développement du projet et son influence sur les coûts |1|.

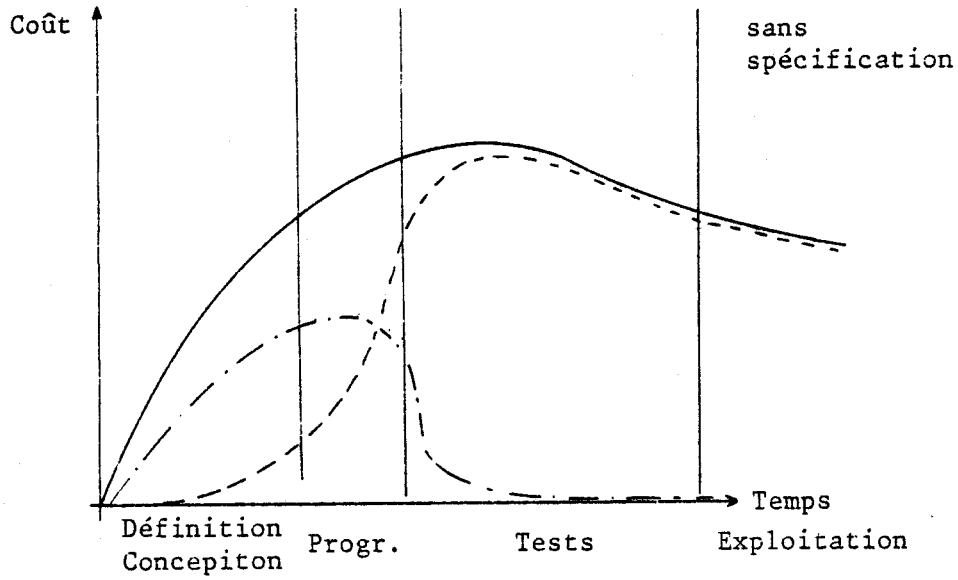


Figure I.3

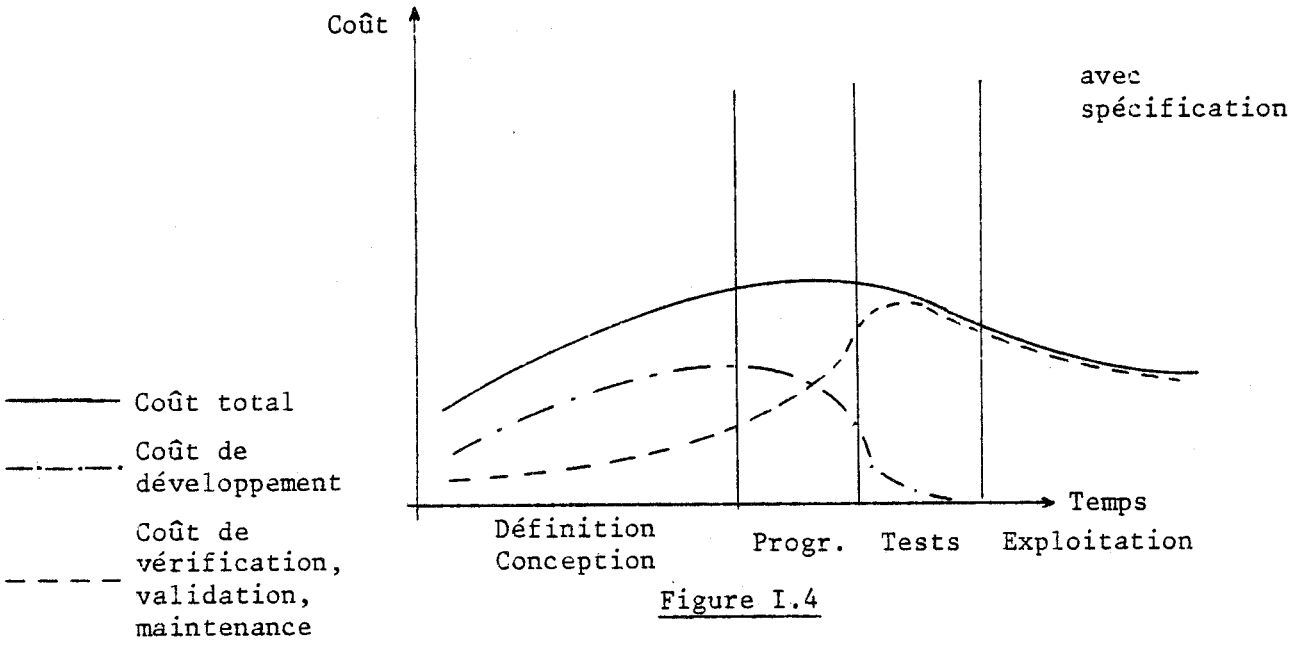


Figure I.4

I.1.3 - Organisation des étapes de spécification

Le problème de la spécification ne se résume pas seulement à la production du cahier des charges, mais il s'attache à toutes les étapes de la vie d'un projet. Certes, l'établissement de ce cahier des charges, réalisé et accepté par le maître d'ouvrage et le maître d'œuvre, est une étape cruciale dans le développement du projet : il constitue la synthèse de tout un processus |1|, |2|, |3| comprenant :

(i) l'évaluation du système existant,

(ii) la mise en évidence de besoins nouveaux,

(iii) l'étude de faisabilité, c'est à dire l'analyse des besoins (coût et avantages, ...) et ébauche des solutions,

(iv) la définition du système et de ses objectifs, des critères de choix,

(v) la mise au point d'un planning de travail.

On y trouvera d'ailleurs les éléments décrits sur le plan suivant :

1) Présentation générale du système :
   - situer le système : utilisation (à quoi il sert ? qui l'utilise ?)
     environnement géographique (contexte)
   - objectif (le pourquoi de l'étude)

2) Fonctions et relations :
   - hiérarchie
   - description de chaque fonction

3) Contraintes de conception :
   - performance (rapidité, précision, ...)
   - matériel (taille mémoire, ce qu'il faut garder du système existant, ...)

4) Interfaces

5) Base de données

6) Contraintes de développement :
   - délai d'étude
   - coût (estimation, limites)
   - planning de travail
   - test d'acceptation

7) Définition, notation, glossaire.

Tout ceci constituant un ensemble qui devra être :

- conforme au besoins réels
- communicable
- faisable
- modifiable
- complet
- cohérent
- maintenable
- exploitable

et qui reprend toutes les spécifications de définition et des besoins. L'analyse de ce travail permettra de définir les spécifications fonctionnelles décrivant les différentes fonctions et les flots de données, eux-mêmes utilisés dans une phase de conception physique pour établir les spécifications de réalisation comme le montre le diagramme suivant :

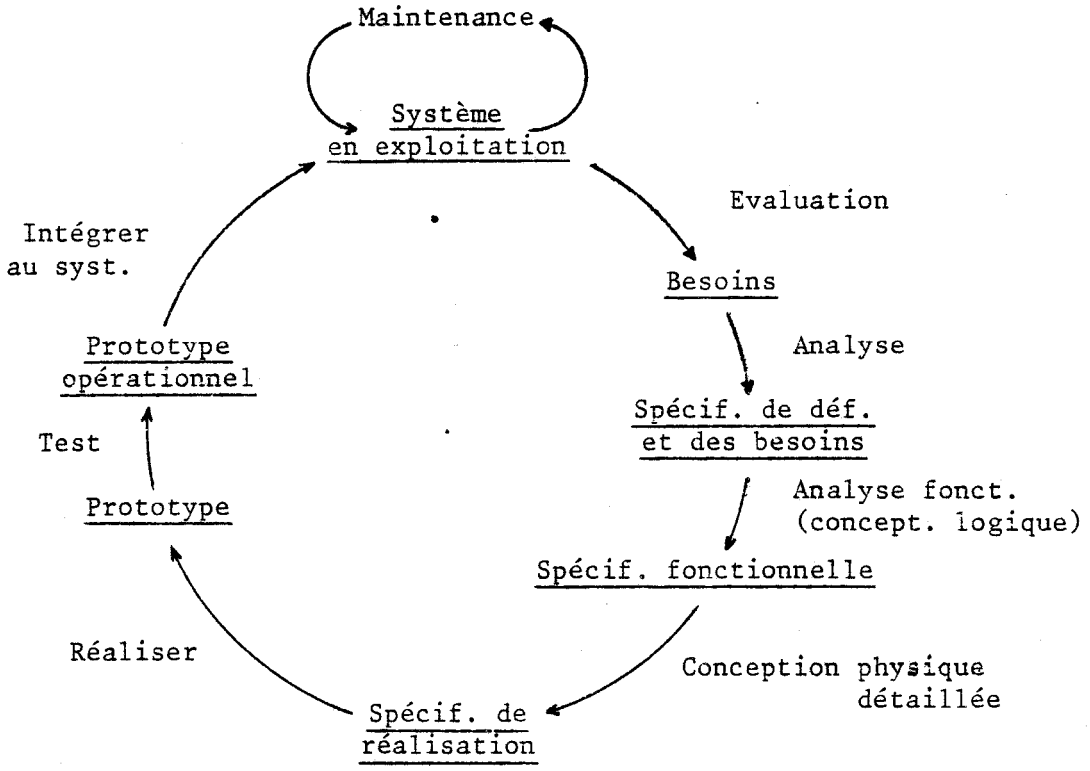


Figure I.5

Remarque : L'utilisation du mot "cycle" pour indiquer une activité continue en cyclique a pour but de mettre l'accent sur le fait que les systèmes sont re-fabriqués à intervalles réguliers lors d'une maintenance par exemple.

La phase de conception physique a pour entrée les spécifications fonctionnelles qu'elle transforme en programme et en machine. Elle produit également les documents suivants :

- plan matériel
- description du système (diagramme hiérarchique, structure des données, ...)
- analyse des compromis, évaluation des alternatives
- conception et écriture des tests
- plan d'exploitation et de maintenance,

qui serviront ensuite à la phase de construction du système sur la machine de développement.

I.1.4 - Principe d'approche de la spécification

Une approche efficace consiste à décrire le système, son environnement, ses objectifs, à l'aide d'une succession de définitions de plus en plus précises. Cette démarche descendante (du plus général vers le plus détaillé) fait apparaître un certain nombre de niveaux d'abstraction et représente la technique clef de spécification et de développement d'un projet.

Le travail de spécification est un travail d'équipe qui réunit concepteurs et utilisateurs et nécessite, pour garantir une poursuite correcte de l'étude, d'être vérifié, testé à chaque niveau. La démarche est donc également itérative dans le sens où chaque oubli, chaque anomalie ou ambiguïté, provoque la reprise du travail au niveau concerné. Il s'agit là également d'une caractéristique essentielle d'une bonne méthode d'analyse. Elle permet en particulier, de minimiser voire supprimer les rebouclages trop grands ; en effet, la non-validation d'une phase de l'étude ne peut entraîner, dans le pire des cas, que la reprise de l'étape précédente. On évite de cette façon, les rebouclages sur des phases

situées chronologiquement bien avant celle qui n'a pas été validée.

Une autre idée importante consiste à repousser le plus loin possible la prise en compte des moyens physiques de réalisation. Des détails d'ordre technique considérés trop tôt, restreignent les choix et les possibilités et dirigent l'étude vers une solution qui ne satisfait pas toujours les objectifs initiaux. Il est en effet evident que la question qui doit se poser en dernier est la question du COMMENT. Seules les questions POURQUOI (motivation), QUOI (objet de l'étude, objectifs, ...) et COMBIEN (performances du système) doivent être envisagés dans les différentes phases d'analyse.

De plus, pour une bonne compréhension entre les divers partenaires, qui sont en général des formations différentes, il est nécessaire d'utiliser un langage commun, simple à assimiler et qui possède un grand pouvoir descriptif. Il s'agit souvent d'éléments graphiques soutenus par des textes qui commentent chaque figure |3|, |17|, |24|, ...

I.2 - LES OUTILS DE SPECIFICATION

I.2.1 - Les différents types d'outils

Que ce soit pour des applications de type informatique, gestion, temps réel ou automatique, il existe aujourd'hui de nombreuses méthodes et outils d'aide à la spécification. Ils peuvent être classés en trois catégories qui expriment leur degré de formalisme et reflètent l'ordre dans lequel ils sont apparus.

I.2.1.1 - Les spécifications informelles :

Ce sont les plus utilisées (99 % d'après |1|). Elles se composent des éléments suivants :

- un texte en langage naturel représentant en moyenne 25 à 35 % du volume total des documents de spécification

- des listes, tableaux (50 à 65 % du volume total)

- des représentations graphiques, figures (10 à 15 % du volume total) |12|.

D'abord faites manuellement, elles fournissent des documents complexes, lourds (pris ici au sens propre comme au sens figuré), présentant, mis à part ceux qui viennent d'être cités, les défauts suivants :

- ambiguïté, car les langages naturels ne sont pas assez précis pour décrire des systèmes : plusieurs lecteurs peuvent avoir une interprétation différente d'un même texte,

- incohérence : les vérifications sur un document volumineux et complexe sont difficiles aussi est-il très difficile d'assurer qu'il ne contient aucune erreur,

- incomplétude : pour les mêmes raisons, il est très difficile d'assurer que rien n'a été oublié.

Pour pallier à ces défauts, des méthodes standardisées ont été développées. Elles proposent :

- un plan type (Ex : dossier standard d'analyse informatique |18|)
- un contenu imposé
- une convention d'écriture
- glossaire, index
- règles de vérification, de mise à jour ...

qui de plus en plus sont assistées par ordinateur (utilisation de fonctions de traitement de texte, analyse de cohérence de données, index ...).

I.2.1.2 - Les spécifications semi-formelles :

Elles font appel en règle générale, soit à un langage formel, soit à une représentation graphique soit également aux deux (Ex : SADT |17|).

Elles permettent ainsi d'exprimer, à l'aide d'une forme syntaxiquement analysable, toutes les informations que l'on retrouve communément dans une spécification. Elles décrivent en particulier les types d'objets et leurs relations.

Des outils automatiques peuvent ainsi être développés afin de tester la cohérence et la complétude et de générer des simulations, des programmes d'essai et de tests |12|, |11|, |13|, |14|.

I.2.1.3 - Les spécifications formelles :

Elles utilisent également des langages de spécification, des représentations graphiques qui ont en plus la particularité d'avoir une syntaxe et une sémantique définies formellement.

Une validation totale des spécifications est ainsi permise. Des méthodologies utilisant de tels formalismes permettent une conception assistée des programmes (Ex : le langage Z |7|).

I.2.2 - Critères d'évaluation

Afin de pouvoir faire la comparaison entre diverses méthodes et divers outils de spécification et de conception, il est nécessaire de définir des critères d'évaluation. Les problèmes rencontrés lors de leur analyse sont les suivants |8| |9| :

1) L'outil de spécification permet-il simplement de décrire les objectifs ou introduit-il également des éléments de solutions ? Cette remarque permet d'éliminer la plupart des langages de programmation qui, dans leur structure, sont essentiellement procéduraux. En effet, ils obligent à décrire comment obtenir les résultats alors que seul le "que doit-on faire" doit être envisagé dans cette phase d'étude.

Un bon outil de spécification doit être statique et non pas dynamique (ou procédural).

2) Permet-il de décrire un système en utilisant plusieurs niveaux d'abstraction ? (Ici encore, il apparaît que la plupart des langages de programmation sont mal adaptés à la spécification, dans le sens où ils réclament d'entrer immédiatement dans les détails). Jusqu'à quel point peut-on faire abstraction des contraintes d'implémentation ?

3) L'outil est-il adapté à des types particuliers d'application ou bien

possède-t-il un domaine d'application général ? (Plus le champ d'application est vaste et plus l'apprentissage et la mise en œuvre sont lourds.

4) Est-il inclus dans une méthodologie permettant d'étudier les grands systèmes en proposant par exemple des règles de décomposition ?

5) Existe-t-il des dispositifs d'aide automatique proposant des facilités de traitement (modification, documentation), des facilités de vérification, de simulation, ... ?

6) Quel est le degré de formalisme de l'outil de spécification ? Celui-ci possède-t-il une base théorique (algèbre de Boole, théorie des ensembles, des graphes, ...) permettant de valider complètement des spécifications ? En particulier, l'outil de spécification a-t-il des possibilités de vérifier en plus de la syntaxe du système (c'est à dire sa structure et les relations entre ses composants), la sémantique (c'est à dire la signification et le rôle de ses éléments) ?

7) L'outil et la méthode sont-ils simples à utiliser ? (Un support graphique permet en général de faciliter leur approche : "mieux vaut un beau dessin ..."). Leur apprentissage est-il aisé pour n'importe quel type d'utilisateur (spécialistes, non spécialistes, ...) ?

Remarque : L'évolution des langages de programmation est telle que leurs deux défauts cités en 1) et 2) tendent à disparaître. En effet, les langages récents autorisent de plus en plus l'abstraction procédurale ; on a ainsi la possibilité de déclarer un sous-programme (lui donner un nom, des arguments) sans en préciser, dans un premier temps, le contenu.
Il est à noter qu'à côté de ces types de langages, les langages non procéduraux comme LISP, ont été développés.

I.3 - ETUDE ET ANALYSE DE QUELQUES FORMALISMES DE SPECIFICATION ET DE CONCEPTION

I.3.1 - Méthodes et outils semi-formels

I.3.1.1 - HIPO |24| :

HIPO (Hierarchy plus Input Process Output) est une méthode de documentation hiérarchisée fondée sur l'utilisation de diagrammes de structure hiérarchique appelés VTOC (Virtual Table Of Contents) qui sont utilisés à 3 niveaux de description :

- niveau Système
- niveau Programme
- niveau Module.

EXEMPLE DE DIAGRAMME DE STRUCTURE D'UN SYSTEME
(Exemple d'une machine flexible) |25| |28|

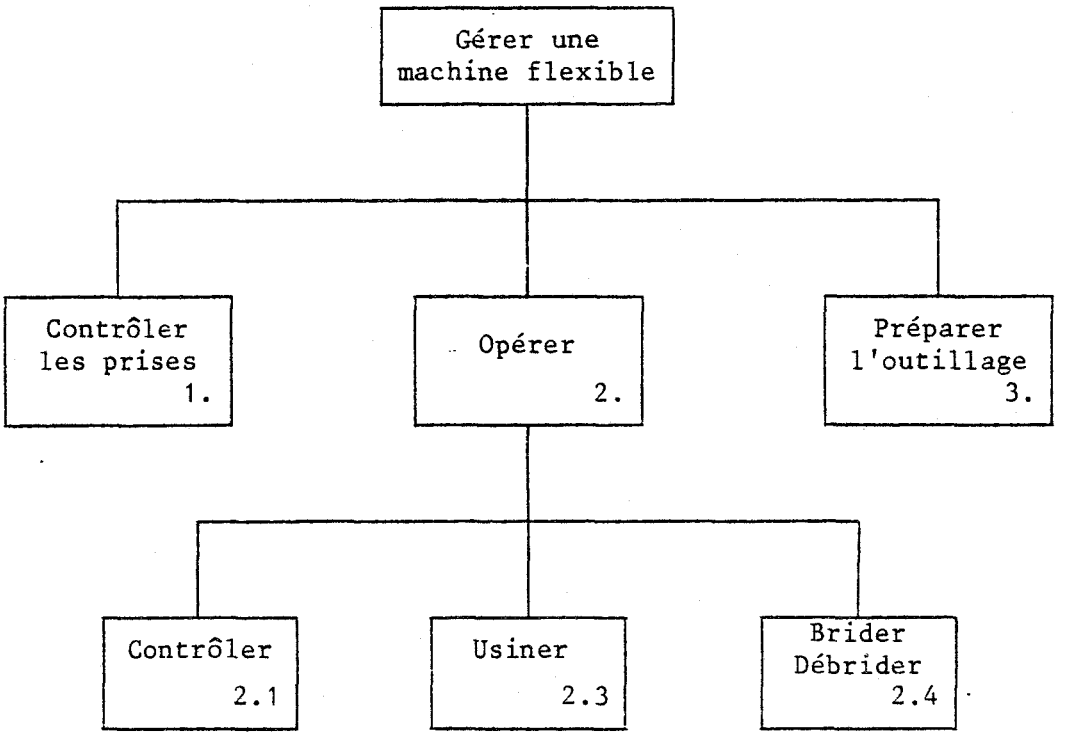


Figure I.6a

EXEMPLE DE DIAGRAMME DE STRUCTURE DE PROGRAMME

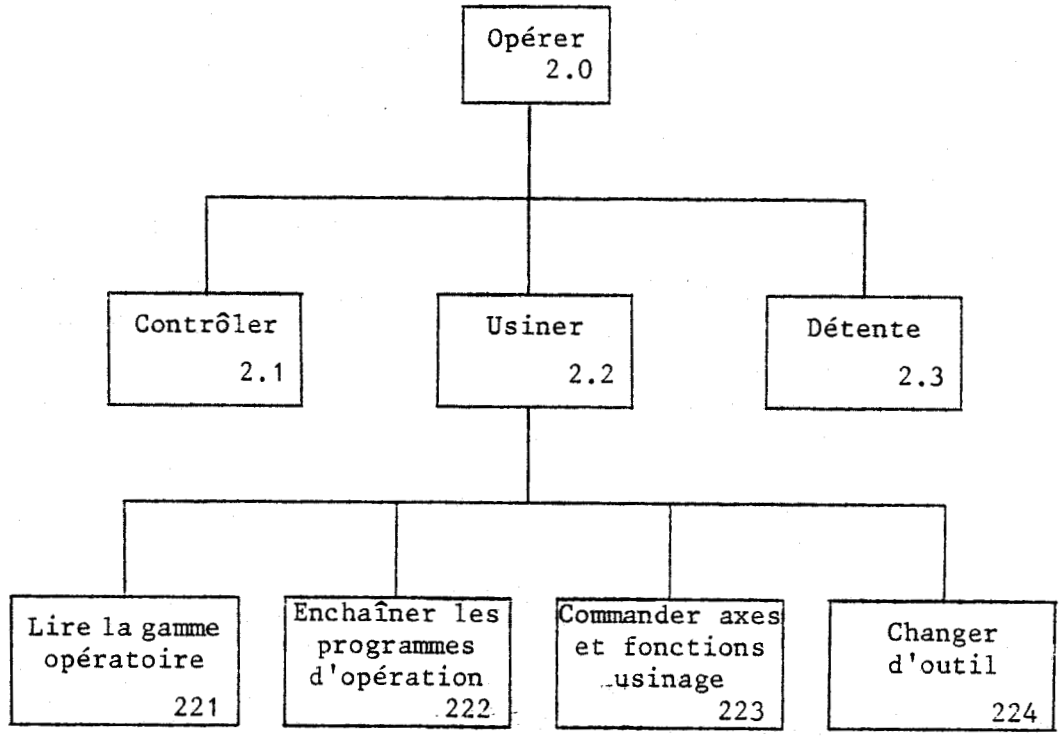


Figure I.6b

EXEMPLE DE DIAGRAMME DE STRUCTURE DE MODULE

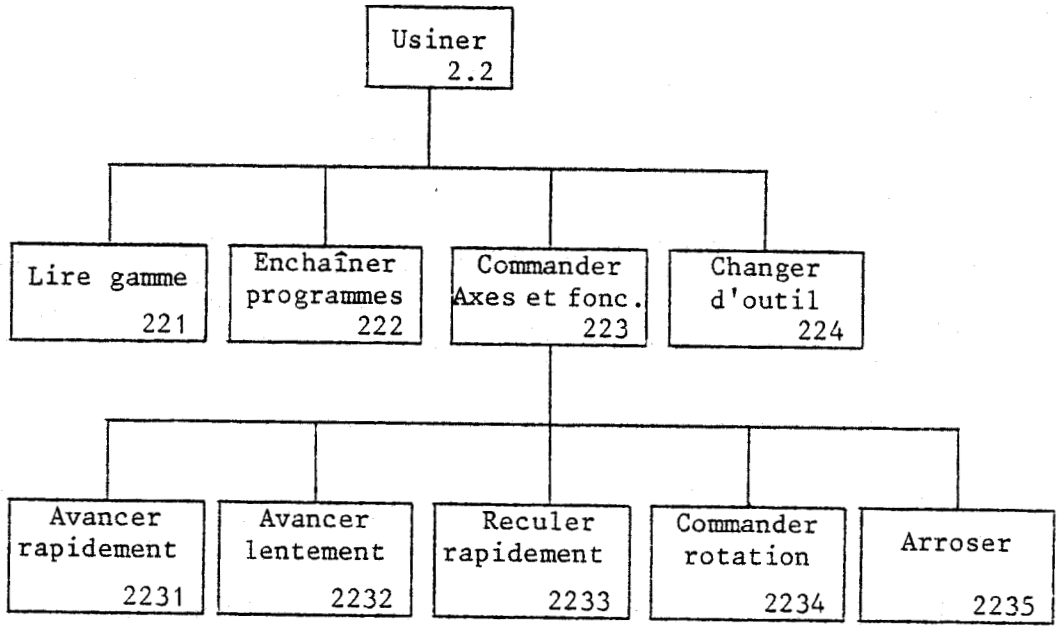


Figure I.6c

Les diagrammes du niveau module décrivent de façon détaillée et séparée, les entrées, les traitements et les sorties du module comme sur la figure ci-dessous.

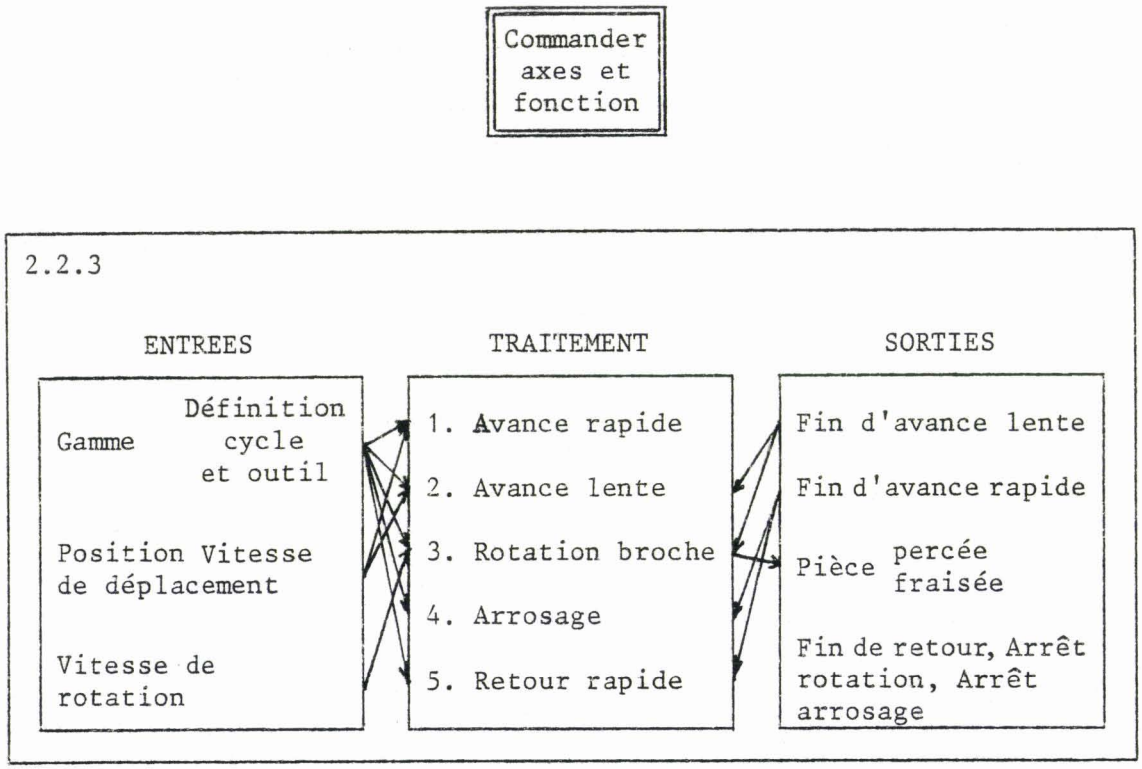


Figure I.6d

Cette méthode présente les défauts suivants :

1) Elle ne propose pas de règles permettant les décompositions successives du système en programmes et en modules. Cela vient du fait que cette technique de spécification et de conception est basée sur les fonctions

2) Elle procède de façon trop procédurale dans le sens où le concept de traitement doit être détaillé.

3) Elle repose sur l'utilisation d'un langage naturel, source d'ambiguïté et d'erreur.

I.3.1.2 - Les diagrammes de flux de données et les dictionnaires de données |3| |19| |20| |21| :

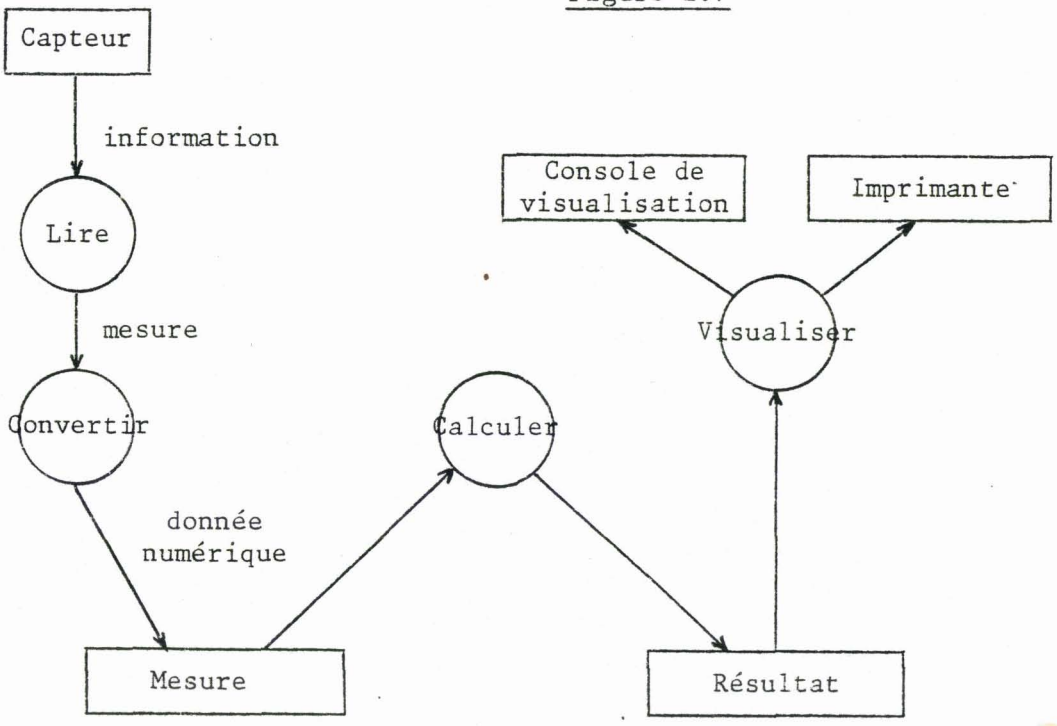
La démarche utilisée dans cette méthode diffère de celle de HIPO dans le sens où la décomposition fonctionnelle ne se fait plus de manière intuitive mais est orientée, dirigée par les flots de données.

Ces diagrammes sont composés des éléments suivants :

- la "bulle" de traitement
- les boîtes "source" et "réservoir" de données
- les arcs qui représentent les flots de données.

Exemple : Un capteur analogique fournit des informations à un système d'acquisition et de traitement de mesure, qui va ensuite convertir ces mesures, les stocker dans un fichier "mesure". Le traitement de ces mesures fournit des résultats qui seront stocker dans un fichier "résultat". Le système se donne également la possibilité de sortir ces résultats, soit sur une console de visualisation, soit sur imprimante. le diagramme correspondant est présenté ci-dessous.

Figure I.7



ACQUISITION, TRAITEMENT DE MESURES ET VISUALISATION

La description d'un système se fait par niveau d'abstraction faisant apparaître des traitements de plus en plus détaillés.

Les diagrammes permettent d'exprimer la syntaxe du système mais il est indispensable pour la représenter totalement, de préciser également leur signification. Aussi, les diagrammes de flux de données sont-ils toujours accompagnés de dictionnaires de données (Data Dictionnary) qui commentent et définissent le rôle de chacun des éléments des diagrammes.

La description des dictionnaires de données est obtenue en utilisant par exemple, un métalangage (dérivé du formalisme de Backus-Nauer) ou d'autres types de diagrammes comme ceux utilisés par WIRTH pour la description du langage PASCAL |22| ou encore ceux que l'on rencontre dans la méthode WARNIER |23|.

Cette méthode est intéressante à plusieurs titres :

- elle fait apparaître la dualité donnée-traitement
- elle permet une décomposition fonctionnelle basée sur les données et donc propose des règles de décomposition bien définies
- elle exprime la syntaxe du système à l'aide des diagrammes de flux de données et dans une certaine mesure, sa sémantique à l'aide des dictionnaires de données.

On peut cependant regretter le manque d'intégrité de cette méthode qui est basée sur deux modèles différents.

I.3.1.3 - SADT |6| |16| |17| :

C'est une méthode développée par SOFTECH qui reprend les grandes idées de l'analyse structurée |6|.

Elle utilise comme modèle de langage, un ensemble structuré et hiérarchisé de diagrammes dont la construction est réalisée en respectant un ensemble de règles cohérentes qui permettent une approche systématique des spécifications d'un système.

Le langage SADT est basé sur deux types de schémas duaux :

- les diagrammes d'activités ou d'actigrammes

- les diagrammes de données ou datagrammes.

La décomposition d'un système permet de faire apparaître des éléments (3 à 6 éléments) représentés chacun par une boîte. Chaque boîte-mère est de nouveau décomposée de façon à fournir un diagramme-enfant et proposer ainsi progressivement de plus en plus de détails.

Les boîtes d'un même diagramme sont interconnectées par des flèches de 4 types :

- flèches d'entrée (qui représentent les données d'entrées ou les activités d'entrées)
- flèches de sortie (données de sorties ou activités de sorties)
- flèches de contrôle
- flèche de support (qui précisent, si nécessaire, le support physique de l'activité ou de la donnée).

On obtient ainsi les éléments de base suivants :

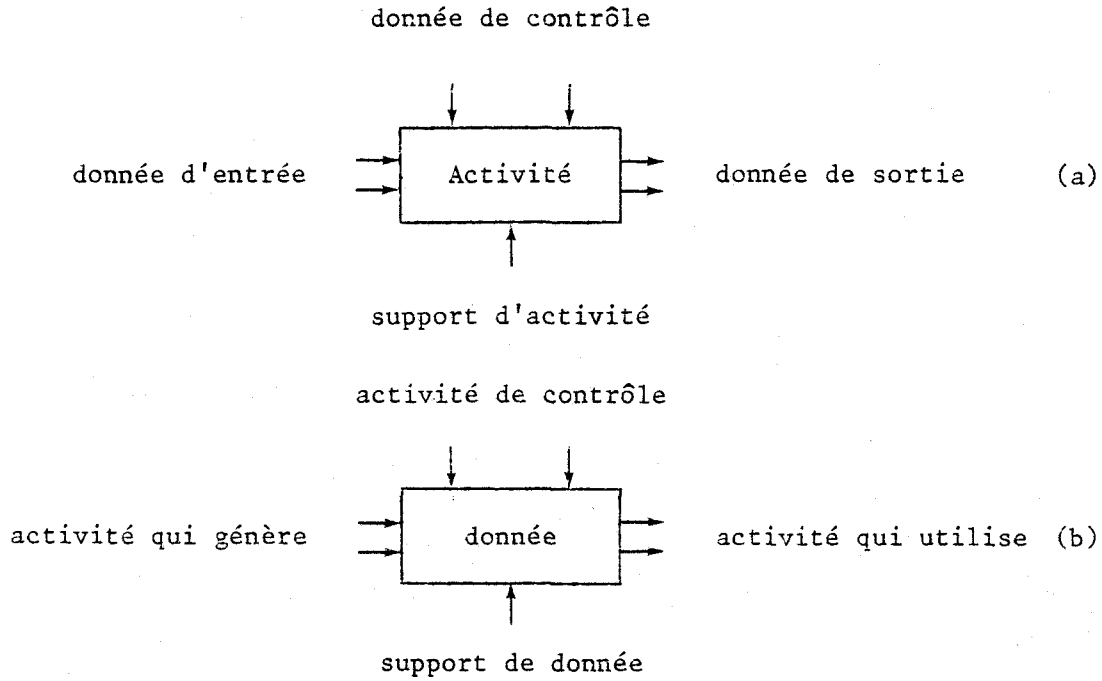


Figure I.8

Les figures ci-après illustrent comment sont interconnectés les différents modules d'un système.

Ainsi, une boîte-mère (une boîte) dispose d'un contexte (entrée, sortie, contrôle, support) que l'on doit retrouver complètement sur le diagramme-enfant.

Exemple : Description des différentes activités d'un système de fabrication de produits composites (adapté de 4).

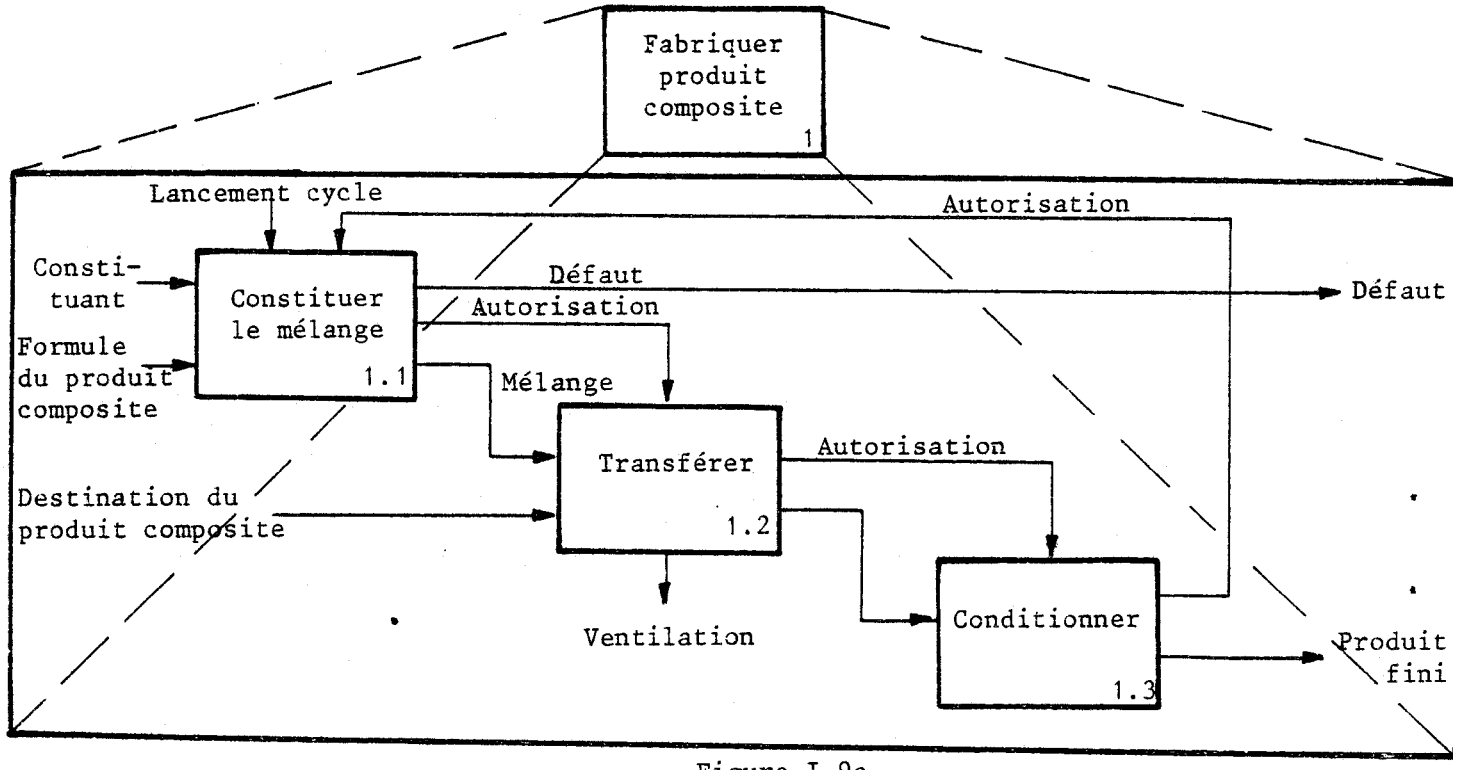


Figure I.9a

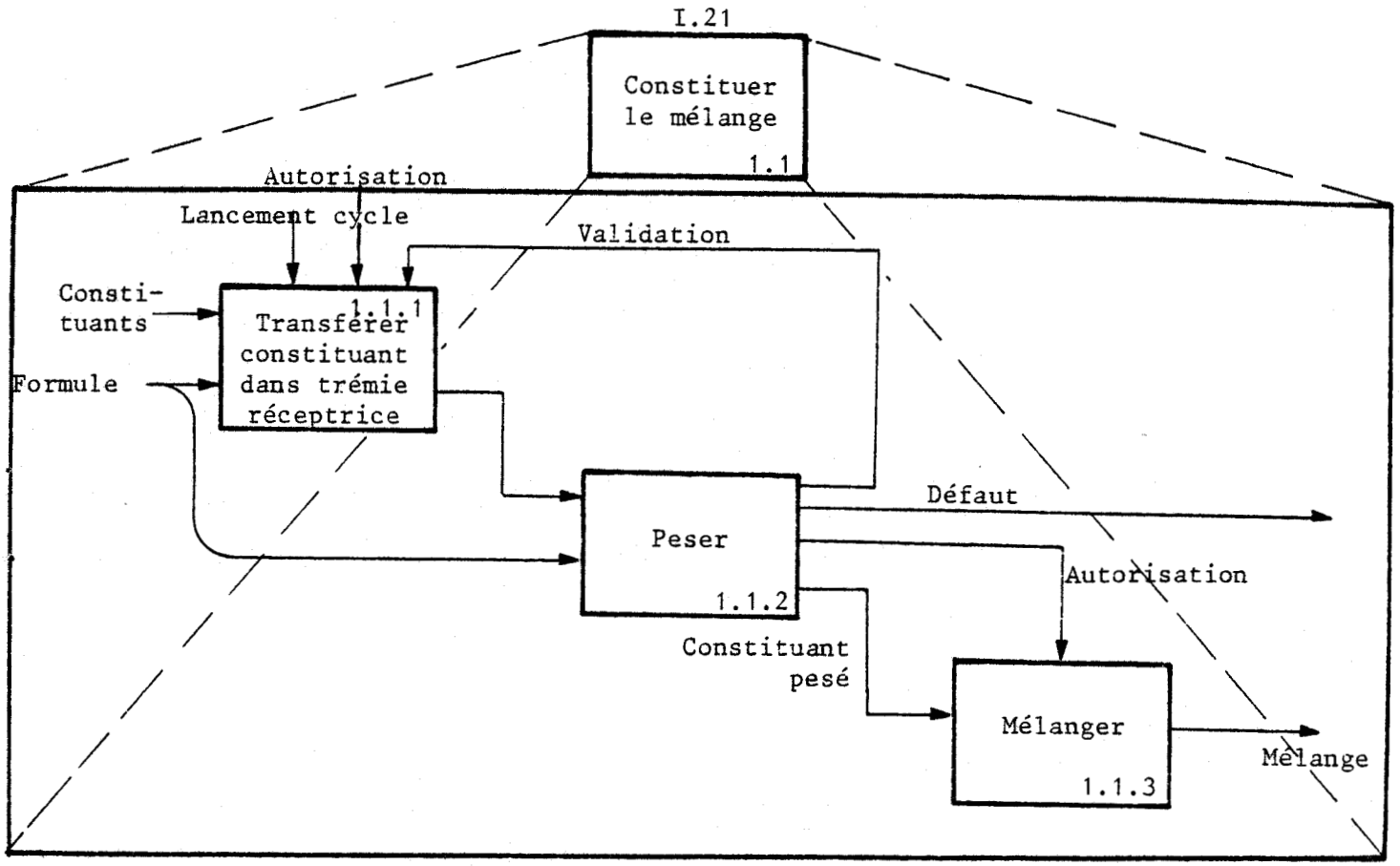


Figure I.9b

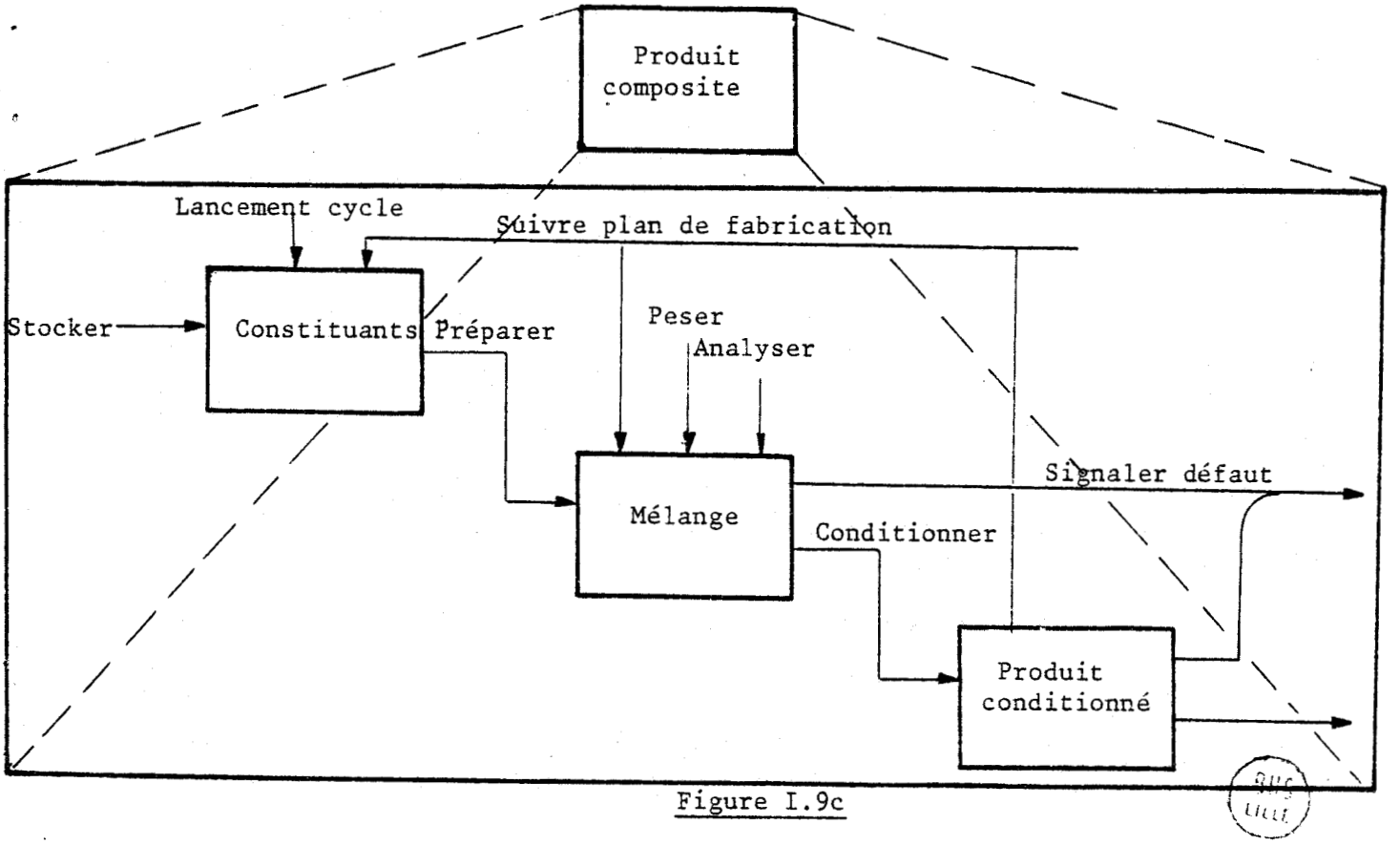


Figure I.9c

EXEMPLE D'UN DATAGRAMME

Ce qui est généralement reproché à la méthode SADT repose sur le fait que celle-ci permet essentiellement la description de l'organisation structurelle des modules et de leurs relations. Des commentaires en langage naturel accompagnant les diagrammes sont nécessaires pour décrire totalement les propriétés du système. De plus, elle n'autorise que des vérifications croisées entre les décompositions de données et activités qui ne permettront donc que de valider la grammaire du système.

I.3.2 - Un outil formel : le langage Z

Le langage Z |7| |8| |9| |10| est un formalisme adapté à la spécification et à la conception des logiciels implémentant tout type d'application.

Remarque : Nous appelons conception, le processus qui, à partir de spécifications initiales (le cahier des charges par exemple) et par une succession de raffinements de spécifications, va fournir un produit permettant une implémentation.

L'idée de base du langage est que la définition des objets manipulés par les programmes, indépendamment de leur représentation physique, conduit naturellement à la décomposition fonctionnelle du système. Dans ce sens, Z utilise la notion de type abstrait de données |20|.

Le formalisme utilisé est celui de la théorie des ensembles.

I.3.2.1 - Méthode :

Elle est itérative et de nature descendante. Trois types de CLAUSES permettent de formaliser une partie du cahier des charges :

- les clauses de TYPE qui énumèrent les différentes sortes d'objets qui apparaissent dans le système.

- les clauses de RELATION qui établissent les relations ou les fonctions existant entre les objets définis précédemment.

- les clauses ASSERTION qui expriment les lois logiques du système (ses propriétés).

Remarque : Les clauses de type et de relation décrivent la structure ou syntaxe du système, les clauses d'assertion, sa sémantique.

Lorsque toutes les parties du cahier des charges ont été formalisées, une première spécification du système est obtenue. Des transformations conservant les propriétés du système, des améliorations nécessitant de reprendre tout le processus de définition par clause, sont ensuite effectuées afin d'améliorer de manière itérative, les versions successives.

Cette approche apparaît comme l'une des plus prometteuse mais ne répond pas exactement à tous les problèmes qui ont été soulevés dans les paragraphes précédents.

Son principal défaut est d'être d'un abord difficile, ce qui ne la rend accessible qu'à des spécialistes. Elle n'est pas actuellement rapportée par des éléments graphiques et son formalisme mathématique la rend peu communicable.

CONCLUSION

Nous avons, dans ce chapitre, mis en évidence les motivations qui poussent les maîtres d'œuvre et d'ouvrage à spécifier de façon précise et efficace leurs problèmes (taille, complexité des systèmes, réduction des erreurs et des coûts de développement et de maintenance).

Ce travail ne peut se faire correctement qu'en respectant des règles cohérentes (description par niveau d'abstraction, décomposition fonctionnelle et structurée basée sur les flux de données, vérification et validation de chaque niveau d'abstraction, ...).

Les outils qui permettent cette phase importante de la vie d'un projet sont nombreux et de natures différentes (informels, semi-formels, formels). Aussi est-il important de pouvoir les évaluer en fonction d'un certain nombre de critères.

Quelques formalismes de spécification sont ensuite présentés et analysés. Aucun d'entre eux ne satisfait totalement les critères qui ont été définis. La décomposition fonctionnelle de HIPO est itérative. Les méthodes adaptées de l'analyse structurée (SADT, ...) ne permettent pas de vérification des propriétés du système et manquent d'intégrité. La méthode Z permet des vérifications sémantiques mais reste, dans sa forme, peu communicable. De plus, les mécanismes de transformation ne sont pas aujourd'hui parfaitement maîtrisés.

Aussi est-il intéressant de se tourner vers d'autres méthodes que nous proposons d'aborder dans la suite de ce mémoire.

BIBLIOGRAPHIE DU CHAPITRE I

|1| G. GERMAIN
"Les nouvelles techniques de spécification du logiciel"
Journée de synthèse, Afcet Informatique, 1980, Nancy.

|2| ZAHNISER traduit par PALLETA pour ICS
"Analyse et programmation structurée. Applications industrielles et temps réel"
Cours 320, 1981.

|3| Tom de MARCO
"Structured analysis and system specification"
Yourdon Inc, 1978.

|4| J.M. CHARTRES
"Méthode et analyse de conception et de spécification de logiciels"
Mémoire d'examen général CNAM, Lille, 1981.

|5| C.V. RAMAMOORTHY, BASTANI, G.M. CHIN, K. SUZUKI
"Application of a methodology for the development and validation of reliable Process Control Software" IEEE Trans. on S.E., Vol. SE-7, n° 6, 1981.

|6| D.T. ROSS
"Structured analysis (SA). A language for communicatif ideas"
I.E.E.E. Trans. on S. E., Vol. SE-3, n° 1, January 1977.

|7| J.R. ABRIAL, S.A. SCHUMAN, B. MEYER
"Specification language (draft)"
August 1979.

|8| Ecole de l'IRIA
"Méthodologie de la programmation"
Support du cours, Tomes 1 et 2, 19 au 23 Mars 1979.

|9| DE MUYNCK, B. MEYER
"Les langages de spécifications"
EDF DERIMA - Atelier Logiciel n° 14, Novembre 1978.

|10| B. MEYER, B. HEILBRONN, A. POUJOL
"Avantages et limites de la spécification formelle du logiciel. Le cas d'un système de protection de réacteur nucléaire"
Journée d'étude AFCET "Validation des spécifications fonctionnelles des systèmes automatique et informatique", 23 Octobre 1980, PARIS.

|11| C.A. IRVINE, J.W. BRACKETT
"Automated software engineering through structured management"
IEEE Trans. on S. E., Vol. SE-3, n° 1, January 1977.

|12| D. TEICHROEW, A.HERSCHEY
"PSL/PSA. A computer-aided for structured documentation and analysis of information processing systems"
IEEE Trans. on S. E., Vol. SE-3, n° 1, January 1977.

|13| T.E. BELL, D.C. BIXLER, M.E. DYER
"An extendable approach to computer-aided software requirements engineering"
IEEE Trans. on S. E., Vol. SE-3, n° 1, January 1977.

|14| M.W. ALFORD
"A requirement engineering methodology for real-time processing requirements"
IEEE Trans. on S.E., Vol. SE-3, n° 1, Januray 1977.

|15| C.G. DAVIS, C.R. VICK
"The software development system"
IEEE Trans. on S. E., Vol. SE-3, n° 1, January 1977.

|16| D.T. ROSS, K.E. SCHOMAN
"Structured analysis for requirements definition"
IEEE Trans. on S. E., Vol. SE-3, n° 1, January 1977.

|17| SOFTECH
"An introduction to SADT. Structured analysis and design techniques"
9022-78R, Novembre 1976.

|18| X. CASTELLANI
"Dossier standard d'analyse informatique"
Diffusé par MA CASTELLANI, 1980.

|19| G.J. MYERS
"Composite. Structured design"
Van Nostrand Reinhold Company, 1978.

|20| E. YOURDON
"Techniques of program. Structure and design"
Prentice-Hall, 1975.

|21| G.J. MYERS
"Reliable software trough composite design"
Van Nostrand Reinhold Company, 1978.

|22| N. WIRTH
"Algorithms + Data structures = Programs"
Prentice-Hall series in Automatic Computation, 1976.

|23| S.D. WARNIER
"Les procédures de traitement et leur donnée"
Editions L'Organisation, PARIS.

|24| STAY
"HIPO and integral program design"
IBM System Journal, Vol. 15, n° 2, 1976.

|25| DAST RENAULT
"Spécification d'une machine flexible pour pièces palettisables"
Document de travail, 1980.

|26| B. LISKOV, S. ZILLES
"Programming with abstract data types"
Sigplan Notices, 9-5 Mai 1974 (également dans 8).

|27| A.B. ENPRES
"An analysis of errors and their causes in system programs"
IEEE Trans. on S. E., June 1975.

|28| "La machine modulaire convertible"
Bulletin technique n° 39.

CHAPITRE II

METHODOLOGIE DE SPECIFICATION ET DE CONCEPTION DES PROCESSUS INDUSTRIELS

CHAPITRE II

UNE METHODOLOGIE DE SPECIFICATION ET DE CONCEPTION BASEE SUR UNE CLASSE PARTICULIERE DE RESEAUX DE PETRI

INTRODUCTION

Les réseaux de Pétri constituent un modèle à la fois formel et graphique ayant suscités des travaux de natures différentes relativement récents.

En considérant l'outil formel, de nombreux auteurs se sont intéressés à l'analyse des propriétés des réseaux de Pétri |2| |3| |17| |18| |20| |24| |27|.

L'utilisation de leurs possibilités de description ou de définition des flots d'information et de contrôle a permis d'aborder de manière originale l'étude des systèmes à activités asynchrones concurrentes ou parallèles |8| |9| |10| |11| |14| |22| |25| |34|.

Ces deux types de travaux permettent, par un rapprochement des résultats, la définition de méthodes de spécification et de conception pour la réalisation de systèmes fiables.

Dans ce chapitre, nous allons rappeler les définitions et caractéristiques des réseaux de Pétri et présenter dans quel sens ils constituent un outil de spécification.

La deuxième partie sera consacrée à la définition d'une méthodologie de spécification et de conception qui utilise comme modèle une classe particulière de réseaux de Pétri : les réseaux de Pétri structurés, permettant une approche simple et sûre des systèmes parallèles |6| à |11|.

II.1 - RESEAU DE PETRI

II.1.1 - Définitions - Notations

II.1.1.1 - Graphe de Pétri |2| |6| |24| :

Définition 1 : Un graphe de Pétri est un graphe biparti orienté, défini par le triplet $G = (P, T, \Gamma)$ où :

- P est un ensemble de sommets appelés places et représentés par des cercles,
- T est un ensemble de sommets appelés transitions et représentés par des barres (les transitions peuvent aussi être représentées par des rectangles ; représentation très utilisée dans les travaux publiés en Allemagne |26|),
- Γ est l'application multivoque qui fait correspondre à un élément d'une classe de sommets, l'ensemble de ses successeurs. Ce dernier est contenu dans l'ensemble des sommets de l'autre classe.

On définit de la même manière, l'application multivoque réciproque Γ^{-1} qui associe à un sommet, l'ensemble de ses prédécesseurs.

Propriété 1 :
$$\forall t \in T \quad \begin{array}{r} \Gamma(t) \subset P \\ \Gamma^{-1}(t) \subset P \\ \tilde{\Gamma}(t) \subset P \end{array}$$

($\tilde{\Gamma}(t)$ représente l'ensemble des voisins du sommet t)

$$\tilde{\Gamma}(t) = \Gamma(t) \cup \Gamma^{-1}(t)$$

$$\forall p \in P \quad \begin{array}{r} \Gamma(p) \subset T \\ \Gamma^{-1}(p) \subset T \\ \tilde{\Gamma}(p) \subset T \end{array}$$

Définition 2 : Soit $G = (P, T, \Gamma)$ un graphe biparti orienté, nous définissons alors :

(i) L'application valuation $V : P \times T \cup T \times P \to (0 , 1)$ tel que :

$V(x,y) = 1$ si $y \in \Gamma(x)$
$V(x,y) = 0$ sinon

Remarques : . Nous noterons $V(x,y)$ pour $V((x,y))$, ...
. Nous considérons ici que G est 1.graphe (entre 2 sommets de types différents, il existe au plus 1 arc. Nous généraliserons par la suite cette définition (§ II.1.4.2)).

(ii) S, la restriction de V définie par :

$S : T \times P \to \mathbb{N}$ tel que :

$\forall x,y \in T \times P \quad S(x,y) = V(x,y)$

S est appelée fonction de sortie.

(iii) E, la restriction de V définie par :

$E : T \times P \to \mathbb{N}$ tel que :

$\forall x,y \in T \times P \quad E(x,y) = V(y,x)$

E est appelée fonction d'entrée.

Exemple :

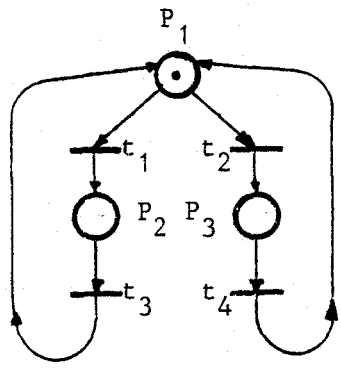


Figure II.1

Soit le réseau $R = (P, T, \Gamma, M_o)$ présenté ci-contre :

$E(t_1,p_i) = 1$ pour $p_i = P_1$ et
$E(t_1,p_i) = 0$ pour $p_i \in P - \{P_1\}$
$E(t_1,p_1) = V(p_1,t_1) = 1$
De même :
$S(t_2,p_i) = 1$ pour $p_i = P_3$
$S(t_2,p_i) = 0$ pour $p_i \in P - \{P_3\}$
et donc :
$S(t_2,p_3) = V(t_2,p_3) = 1$

Définition 3 : L'ensemble des places (resp transitions) prédécesseurs et l'ensemble des places (resp transitions) successeurs d'une transition t (resp place p) sont appelés respectivement ensemble des entrées et ensemble des sorties de la transition t (resp place p) et seront notés respectivement :

$E(t,\bullet)$ (resp $E(\bullet,p)$)
$S(t,\bullet)$ (resp $S(\bullet,p)$)

ou encore :

$E(t)$ (resp $E(p)$)
$S(t)$ (resp $S(p)$)

Remarque : On peut également définir le graphe de Pétri comme un 4.uple $G = (P, T, E, S)$ |18| |27| |3|. Les deux définitions sont équivalentes.

II.1.1.2 - Réseau de Pétri :

Définition 4 : On appelle Réseau de Pétri, le couple $\mathcal{R} = (G,M_0)$ dans lequel G représente le graphe de Pétri et M_0 l'application définie sur P et à valeur dans $\mathbb{N}$.

$$M_0 : P \rightarrow \mathbb{N}$$

M_0 est appelé marquage initial du réseau.

II.1.1.3 - Représentation d'un réseau de Pétri :

a) Représentation graphique :

Le marquage M_0 peut être modélisé par une distribution d'objets appelés marqueurs dans les places du réseau. Il peut être schématisé en inscrivant dans chaque place la quantité $M_0(p)$ si celle-ci est non nulle ou encore en plaçant $M_0(p)$ "jetons" dans chaque place p.

Exemple : La figure II.2 représente un réseau de Pétri avec :

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

$$t = \{t_1, t_2, t_3, t_4\}$$

$$M_0(P_1) = 3 \qquad M_0(P_2) = 0 \qquad M_0(P_3) = 1$$

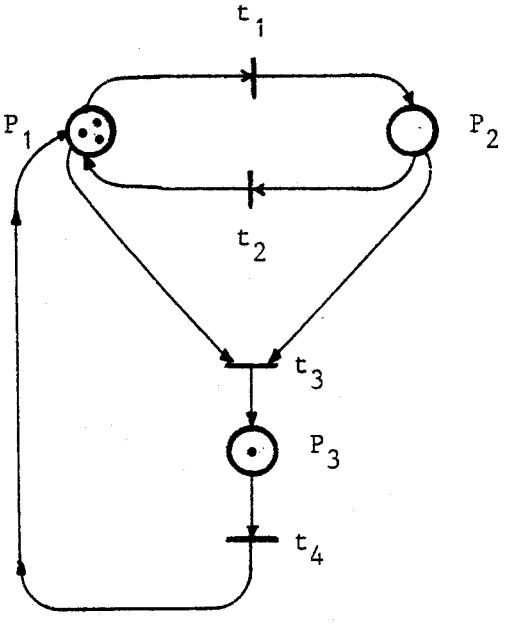


Figure II.2

b) Représentations matricielles :

On considère un réseau de Pétri $\mathcal{R} = (P, T, \Gamma, M_0)$ dont les ensembles P, T sont finis et ordonnés.

$$P = p_1, p_2, \ldots, p_m \qquad |P| = m \quad \text{(m places)}$$

$$T = t_1, t_2, \ldots, t_n \qquad |T| = n \quad \text{(n transitions)}$$

alors, l'application V peut être associée à la matrice :

$$V = ((v_{ij})) \quad \text{avec} \quad V_{ij} = V(x_i, y_j)$$

où x_i et y_j appartiennent à l'ensemble ordonné, défini par la réunion

de P et T.

Si l'on considère l'exemple de la figure II.1, sa matrice d'adjacence V a la forme suivante :

$$V = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{ccc} P_1 & P_2 & P_3 \end{array} \\ \left[\begin{array}{ccc|cccc} & & & 1 & 0 & 1 & 0 \\ & 0 & & 0 & 1 & 1 & 0 \\ & & & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 0 & & & & \\ 1 & 0 & 0 & & & 0 & \\ 0 & 0 & 1 & & & & \\ 1 & 0 & 0 & & & & \end{array}\right] \\ \\ \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \end{array}$$

Le mode de représentation par la matrice d'adjacence sommet-sommet n'est guère utilisé ; on lui préfère celui que l'on obtient à partir des applications E et S définies plus haut.

Les matrices associées sont définies par :

$$E = ((e_{i,j})) \quad \text{avec} \quad e_{i,j} = E(t_i,p_j)$$
$$S = ((s_{i,j})) \quad \text{avec} \quad s_{i,j} = S(t_i,p_j)$$

Pour l'exemple de la figure II.1, nous avons ainsi :

$$E = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{ccc} P_1 & P_2 & P_3 \end{array} \\ \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] \end{array} \qquad S = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{ccc} P_1 & P_2 & P_3 \end{array} \\ \left[\begin{array}{ccc} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{array}\right] \end{array}$$

Remarque : La matrice V peut s'écrire :

$$V = \begin{bmatrix} 0 & E^T \\ S & 0 \end{bmatrix}$$

La description du graphe à l'aide des matrices E et S est donc moins lourde : la quantité d'informations à stocker (sur calculateur, par exemple), est $2(n \times m)$ au lieu de $(n+m)^2$ avec la matrice V.

Ces deux représentations matricielles peuvent être regroupées pour ne former qu'une seule matrice appelée matrice caractéristique ou d'incidence.

Cette matrice, notée C, est obtenue en retranchant E à S.

Ainsi, $C = S - E$

où $C = ((c_{ij}))$ avec $c_{ij} = S(t_i,p_j) - E(t_i,p_j)$.

La matrice d'incidence du réseau de la figure II.1 peut alors s'écrire :

$$C = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{c} \begin{array}{ccc} P_1 & P_2 & P_3 \end{array} \\ \begin{bmatrix} -1 & 1 & 0 \\ 1 & -1 & 0 \\ -1 & -1 & 1 \\ 1 & 0 & -1 \end{bmatrix} \end{array}$$

Propriété : La matrice C est représentative du graphe si et seulement si celui-ci ne contient pas de chemin de longueur 2 qui se reboucle sur un même sommet.

Définition : La longueur d'un chemin est égale au nombre d'arcs parcourus en suivant ce chemin. Autrement dit, le réseau de Pétri ne doit pas comporter de boucles élémentaires.

Exemples de boucles élémentaires (boucles de longueur 2) :

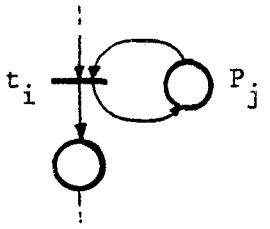


Figure II.3a

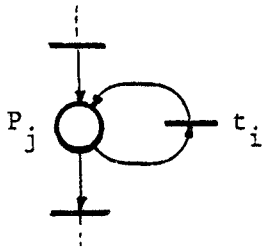


Figure II.3b

En effet, il existe un bouclage de longueur 2, alors :

$$E(t_i,p_j) = S(t_i,p_j)$$

D'où, pour l'élément $C(i,j) = 0$, ce que l'on peut interpréter de la manière suivante : il n'y a pas d'arête qui relie la transition t_i à la place p_j.

Le marquage M_0 de $\mathcal{R}$ est défini par la donnée d'un vecteur de $\mathbb{N}^n$ où $n = |P|$

$$M = ((m_i)) \quad \text{avec} \quad m_i = M_0(p_i)$$

c) Autres représentations :

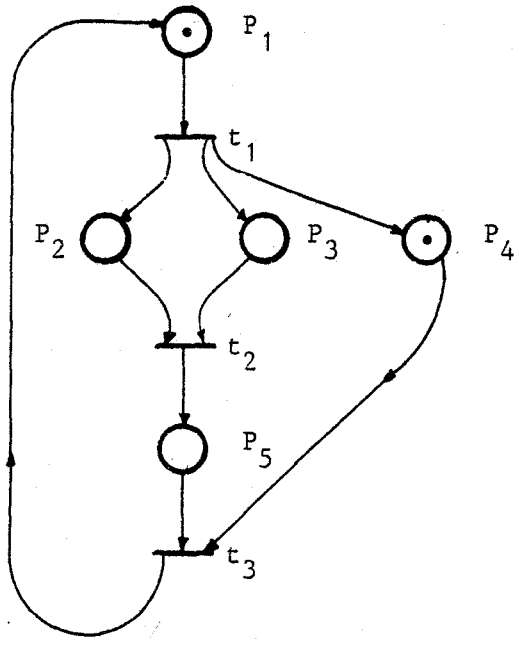
Un graphe de Pétri peut être également représenté à l'aide d'autres formes de tableau :

- tableau des successeurs
- liste des successeurs.

Ils permettent en particulier, de réduire la quantité d'information à stocker en mémoire et sont particulièrement adaptés au graphe dont la matrice d'incidence est creuse.

A titre indicatif, il est également possible de concevoir un réseau de Pétri à l'aide d'un système de réécriture sur l'alphabet des noms de places |2|.

A chaque transition est associée une ou plusieurs règles de réécriture où la partie gauche représente la fonction d'entrée de la transition et la partie droite sa fonction de sortie. Le marquage initial constitue l'axiome de départ. Pour l'exemple de la figure II.3, nous avons le système de réécriture suivant :



M_0 : $P_1\ P_4$ ← axiome

t_1 : $P_1 \rightarrow P_2\ P_3\ P_4^2$

t_2 : $P_2\ P_3 \rightarrow P_5$

t_3 : $P_4^2\ P_5 \rightarrow P_1\ P_4$

Figure II.4

II.1.2 - Règles d'évolution d'un réseau de Pétri

Deux aspects d'un système sont décrits par un réseau de Pétri :

- sa structure (ou sa statique) à l'aide du graphe de Pétri

- son comportement dynamique à l'aide du marquage dont l'évolution est déterminée par les règles qui suivent.

Définition : Transition sensibilisée

Une transition t est dite sensibilisée par un marquage M si et seulement si pour tout $p \in \Gamma^{-1}(t)$, on a $M(p) \geq 1$; c'est à dire toute place d'entrée de la transition contient au moins une marque.

En utilisant la notation matricielle définie précédemment, t est sensibilisée si et seulement si :

$$\boxed{\forall p \in P \qquad M(p) \geqq E(t,p)} \qquad (1)$$

Définition : Tir d'une transition sensibilisée

Soit M, un marquage vérifiant la relation (1) pour la transition t. Le tir (on dit également mise à feu ou déclenchement) de la transition va provoquer une évolution de marquage. Le marquage obtenu M' par le tir de t depuis M est défini par :

$$\forall p \in P \qquad M'(p) = M(p) + V(t,p) - V(p,t)$$
$$\text{ou} \qquad M'(p) = M(p) + S(t,p) - E(t,p)$$

soit aussi :

$$\boxed{M'(p) = M(p) + C(t,p)} \qquad (2)$$

Exemple de tir :

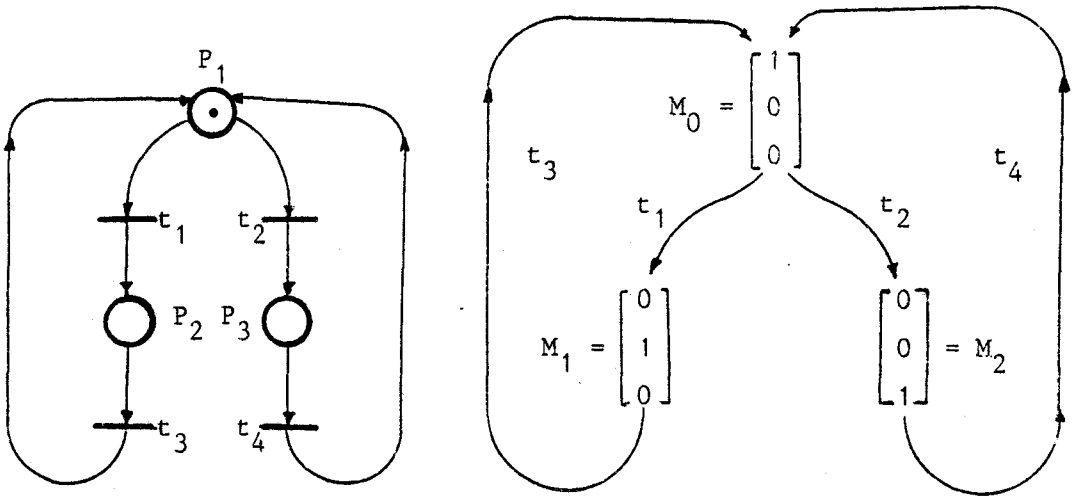


Figure II.5

Un déclenchement δ_t associé à une transition t est une semi-application de $\mathbb{N}^n$ dans $\mathbb{N}^n$ qui, à un marquage M vérifiant (1) fait correspondre un marque M' obtenu par (2) : $\delta_t(M) = M'$.

Définition : séquence de tir

Une séquence de déclenchement σ est une composition de déclenchement

$$\sigma = \delta_{t_n} \circ \ldots \circ \delta_{t_1}$$

Soit M un marquage vérifiant (1) pour la transition t_1, alors $\sigma(M)$ est défini si pour tout $i > 1$, le marquage $\delta_{t_{i-1}} \circ \ldots \circ \delta_{t_1}(M)$ sensibilise la transition t_i.

Définition : marquage accessible

Un marquage M' est accessible depuis M si il existe au moins une séquence de $\sigma(M)$, tirable depuis M qui conduit en M'.

Nous noterons $\mathcal{E}(M)$ l'ensemble des marquages accessibles depuis M_0.

Définition : degré d'une séquence de tir

Soit σ, une séquence de déclenchement et t_i une transition de T.

Nous appellerons degré en t_i de la séquence σ le nombre d'occurrence de déclenchement δ_{t_i} de la transition t_i. Le degré de la séquence σ est un vecteur de $\mathbb{N}^m$ (avec $m = |T|$) dont le $i^{ème}$ composante est le degré en t_i de la séquence σ.

II.1.3 - Propriétés associées au marquage d'un réseau de Pétri

II.1.3.1 - Propriété borné :

Soit un réseau de Pétri $R = (G, M_0)$ avec $G = (P, T, \Gamma)$.

Définition : Une place $p \in P$ est k bornée si et seulement si :

- $\exists M \in \mathcal{E}(M_0)$ tel que $M(p) = k$
- $\forall M \in \mathcal{E}(M_0)$ $M(p) \leq k$

Si k = 1, on dit que la place p est saine.

Définition : Une place sera dite bornée si et seulement si il existe $k \in \mathbb{N}$ tel que cette place soit k bornée.

Définition : Le réseau de Pétri est borné (resp. sain ou sauf) si et seulement si toutes ses places sont bornées (resp. saines).

Propriété : Les propositions suivantes sont équivalentes :

- le réseau est borné
- $\mathcal{E}(M_0)$ est fini.

La démonstration donnée dans |20| est la transposition aux réseaux de Pétri des résultats obtenus pour un formalisme équivalent : les systèmes d'addition de vecteurs.
Un réseau de Pétri non borné contient au moins une place dont le marquage peut croître à l'infini. Il s'ensuit que l'ensemble des marquages atteints par un réseau non borné contient une infinité d'éléments.

II.1.3.2 - Propriété "vivant" :

Plusieurs formulations de la propriété de vivance ont été proposées |2| |24|.

Définition : Un réseau de Pétri est pseudo-vivant pour un marquage initial si et seulement si tout marquage accessible depuis M_0 sensibilise au moins une transition.

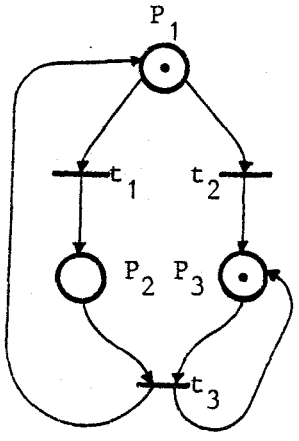
$$\forall M \in \mathcal{E}(M_0) \qquad \exists t \in T, \forall p \in P \qquad M(p) \geq E(t,p)$$

Cette propriété traduit l'absence de blocage total du réseau.
(Un réseau totalemant bloqué, ou mort est un réseau dont le marquage ne sensibilise aucune transition $t \in T$, interdisant ainsi l'évolution du marquage M.

Définition : Une transition t est quasi-vivante pour un marquage M si et seulement si il existe une séquence de déclenchement σ contenant t déclenchable pour M.
Un réseau de Pétri est quasi-vivant pour un marquage M si et seulement si toutes ses transitions sont quasi-vivantes pour M.

Définition : Une transition est vivante pour un marquage M si et seulement si elle est quasi-vivante pour tout marquage M' atteint à partir de M_0.
Un réseau de Pétri est vivant pour un marquage M si et seulement si toutes ses transitions sont vivantes pour M.

Exemples :



t_1 quasi vivante.
t_2
t_3 quasi vivante ∃ t_1, t_3.

$$M_0 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

Figure II.6a

Le réseau de la figure ci-dessus est quasi-vivant pour le marquage M_0 (toutes les transitions peuvent au moins être déclenchées une fois) mais non pseudo-vivant (le marquage $\begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$ obtenu par le déclenchement de t_2 à partir de M_0 ne sensibilise aucune transition). Le réseau est donc non vivant.

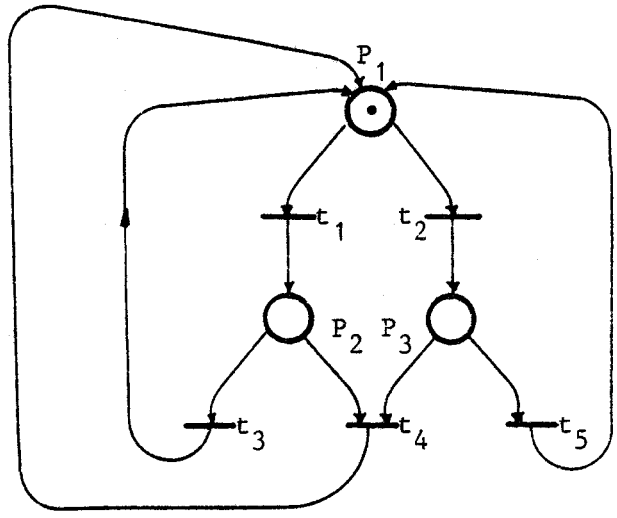


Figure II.6b

Le réseau ci-dessus est pseudo-vivant (il existe toujours une transition déclenchable pour $M \in \delta(M_0)$ mais non quasi-vivant pour M_0 (il n'existe pas de séquence de déclenchement qui depuis M_0, contient la transition t_4). Le réseau est donc également non-vivant.

II.1.3.3 - Propriété réinitialisable (ou propre) |34| :

Cette propriété intéressante traduit le fait qu'un système modélisé par le réseau a un fonctionnement cyclique. Sur le plan des applications, il s'agit d'une propriété de réversibilité très intéressante en cas de panne : retour à un état antérieur.

Définition : Un réseau de Pétri $R = (G, M_0)$ est propre pour un marquage M_0 si et seulement si pour tout marquage M accessible depuis M_0, il existe une séquence de déclenchement σ qui conduit à M_0 ; c'est à dire si :

$$\forall M \in \delta(M_0) \qquad \exists \sigma \qquad \sigma(M) = M_0$$

Remarque : Par analogie avec les propriétés d'un système en automatique, nous pouvons considérer que les propriétés "borné" et "réinitialisable" caractérisent la stabilité, alors que la propriété de vivance indique que toutes les parties du réseau sont accessibles, c'est à dire que le réseau est commandable.

II.1.4 - Réseaux de Pétri particuliers

Depuis la définition originelle des réseaux de Pétri, de nombreuses variantes ont été proposées, ceci dans deux buts :

1) faciliter l'analyse du comportement du réseau en restreignant les possibilités d'expression,

2) ou au contraire étendre la puissance de représentation du modèle au détriment des facilités d'analyse.

On retrouve parmi les variantes du premier type :

- les graphes d'évènements
- les graphes d'état
- les réseaux libre -choix, ...

et parmi celles de la deuxième catégorie :

- les réseaux de Pétri généralisés
- les réseaux de Pétri arc inhibiteur, ...

II.1.4.1 - Les classes restrictives des réseaux de Pétri :

a) Les graphes d'évènements |29| :

Définition : Un graphe d'évènement est un graphe de Pétri pour lequel toute place $p \in P$ n'admet qu'une et une seule transition d'entrée et une et une seule transition de sortie.

$$\forall p \in P \qquad |\Gamma(p)| = |\Gamma^{-1}(p)| = 1$$

Exemple : Réseau de Pétri d'évènement (Figure II.7a)

Ils permettent de modéliser le parallélisme (le tir d'une transition peut faire évoluer le marquage de plusieurs places d'entrée et plusieurs places de sortie de cette transition) mais n'expriment pas les choix (un marquage M(p) ne peut sensibiliser au maximum qu'une transition).

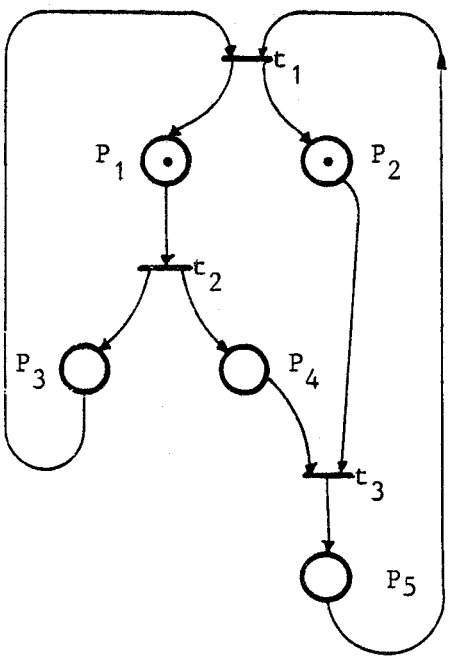


Figure II.7a

b) Les graphes d'états |18| |29| :

Définition : Un graphe d'état est un graphe de Pétri pour lequel toute transition t ∈ T n'admet qu'une et une seule place d'entrée et une et une seule place de sortie.

$$\forall\, t \in T \qquad |\Gamma(t)| = \Gamma^{-1}(t) = 1$$

Exemple de réseau de Pétri d'état :

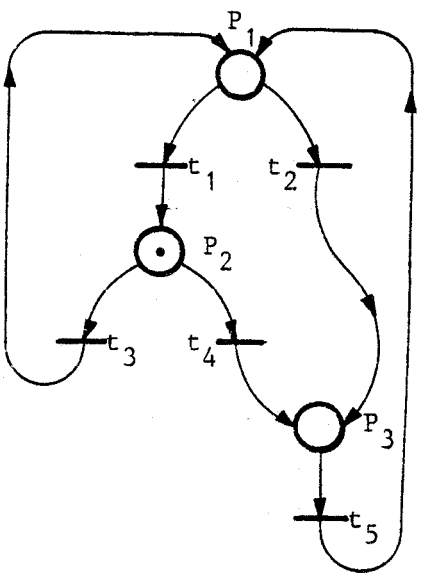


Figure II.7b

Remarque : Dualité place-transition

On passe d'un graphe d'évènement à un graphe d'état en transformant chaque place en transition et réciproquement (Figures II. a et II. b).
Compte tenu de cette remarque, un réseau de Pétri d'état permet d'exprimer les alternatives mais interdit la représentation du parallélisme.

c) Les réseaux de Pétri libre-choix |18| |19| |24| :

Ils apparaissent comme un compromis entre les deux classes définies plus haut.

Définition : Un réseau de Pétri libre-choix est tel que chacun de ses arcs ayant comme origine une place p et comme extrémité une transition t, soit :

- ou le seul arc sortant de p
- ou le seul arc entrant de t.

Exemple :

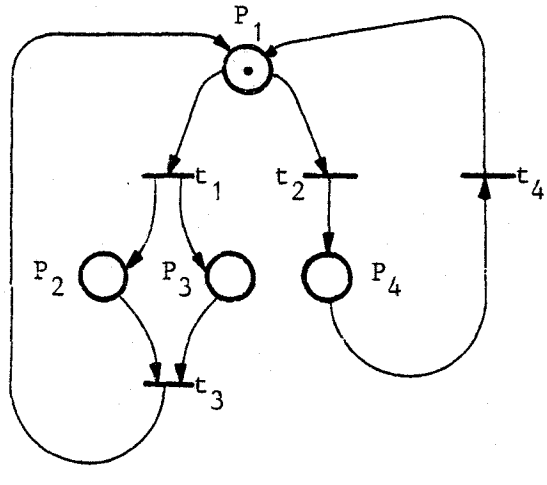


Figure II.7c

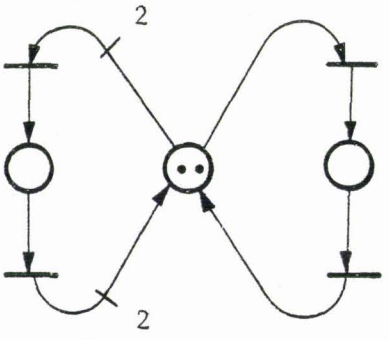
Une extension de ces réseaux a été présentée dans |24| les réseaux à choix non imposé.

II.1.4.2 - Extension des réseaux de Pétri :

a) Réseau de Pétri généralisé :

Définition : On appelle réseau de Pétri généralisé, un réseau de Pétri $R = (G, M_0)$ pour lequel G est un multigraphe biparti orienté. Entre deux sommets adjacents, il peut exister un ou k arcs ($k \in \mathbb{N}$; $k > 1$). Dans ce cas, une étiquette associée à l'arc indique le poids qui lui est attribué.

Exemple :



(par défaut, le poids de l'arc vaut 1)

Figure II.8

Toutes les définitions, les règles données précédemment, s'appliquent aux réseaux généralisés en remarquant que les applications V, S, E sont désormais à valeur dans $\mathbb{N}$.

b) Réseau à arc inhibiteur |1| |17| :

Définition : Un réseau à arc inhibiteur est un doublet $R_Z = (R, I)$ tel que :

- R est un réseau de Pétri ordinaire = (G, M_0)
- I un ensemble d'arc inhibiteur : $I \subseteq P \times T$

On représente un arc inhibiteur en plaçant un petit cercle à l'extrémité incidente aux transitions.

Exemple :

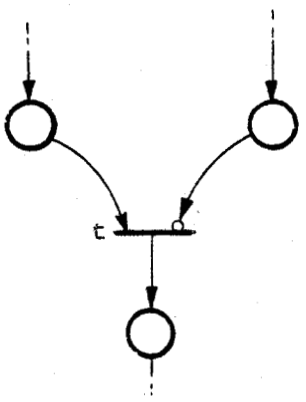


Figure II.9

Règles d'évolution :

t ∈ T est sensibilisée par un marquage M si chacune de ses places d'entrée reliée à t par un arc ordinaire est marquée, et si chaque place reliée à t par un arc inhibiteur est vide.

Le déclenchement d'une transition sensibilisée par un marquage M consiste à enlever un marqueur de toutes les places d'entrée reliées par un arc ordinaire et à déposer une marque dans toutes ses places de sortie.

La transformation de R_Z en R est réalisée en introduisant pour chaque place p reliée à t par un arc inhibiteur, une place complémentaire P'.

Exemple :

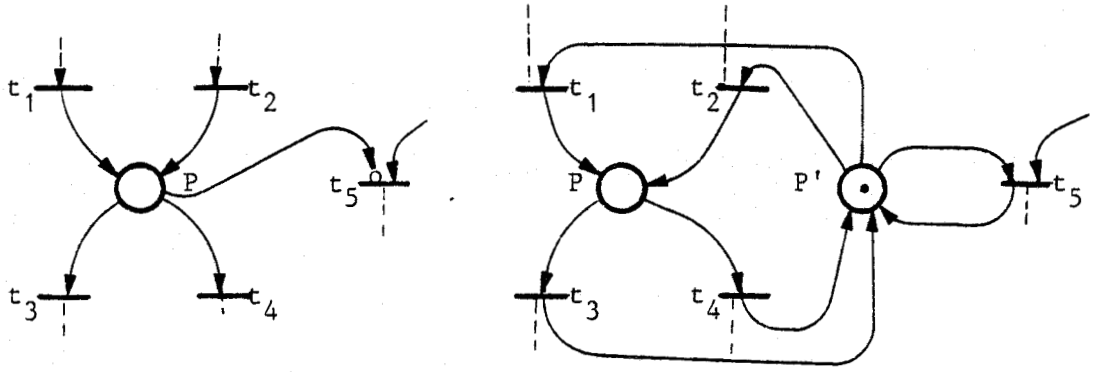


Figure II.10

c) Réseau de Pétri étiqueté :

Un réseau de Pétri étiqueté est un triplet $R_e = (R, A, f)$ tel que :

- R représente un réseau de pétri
- A est un alphabet
- f une application définie par $f : T \rightarrow A \cup \{\lambda\}$
  où λ est l'élément neutre (le mot vide) du monoïde libre A* construit sur l'alphabet A.

Remarque : Tout réseau de Pétri peut être considéré comme étant un réseau de Pétri étiqueté pour lequel A = T et f représente l'application identité.

II.2 - RESEAU DE PETRI : OUTIL DE SPECIFICATION ET DE CONCEPTION

II.2.1 - Modélisation

II.2.1.1 - Généralités :

L'intérêt de la modélisation tient essentiellement à ce qu'elle permet de représenter le comportement dynamique d'un système sans nécessiter ou son existence physique (il peut exister sous forme de projet) ou des manipulations sur celui-ci, dans le but d'évaluer, d'analyser le système réel ou en projet (mesure des performances, vérification des spécifications, ...).

La plupart des modèles utilisent un formalisme mathématique. En automatique par exemple, les systèmes continus sont très souvent modélisés à l'aide d'équations différentielles, et leur comportement à diverses sollicitations peut être obtenu soit en résolvant ces équations soit par simulation sur calculateur analogique digital ou hybride.

Les réseaux de Pétri constituent également un outil de modélisation, plus particulièrement adaptés aux systèmes d'évènements discrets présentant des activités parallèles et concurrentes. Ils permettent effectivement de représenter les flots de données et les structures de contrôle de tels systèmes.

II.2.1.2 - Interprétation :

La description du fonctionnement d'un système par un réseau de Pétri exige que l'on tienne compte des spécificités de ce système, c'est à dire :

- ses différents états,
- les conditions qui lui permettent d'évoluer,
- leurs relations.

L'ensemble de ces caractéristiques fournit un vocabulaire dont chaque élément va permettre d'attribuer une signification aux symboles qui étiquettent les places et les transitions.

Ainsi, les symboles relatifs aux conditions sont généralement associés aux places alors que les transitions représentent les conséquences (Event and Condition |19|). Cependant, cette règle n'est pas stricte, elle dépend :

i) du type de modélisation - flux de données
- flux de contrôle

ii) du stade de développement du projet (en phase de conception détaillée, les places représentent des actions, les transitions sont, quant à elles, relatives aux évènements externes ou internes (prédicats) (Ex : RdPI |25|)).

Remarque : Ces derniers points de vue s'expliquent à la fois par la dualité place-transition et par la dualité donnée-contrôle.

Exemple :

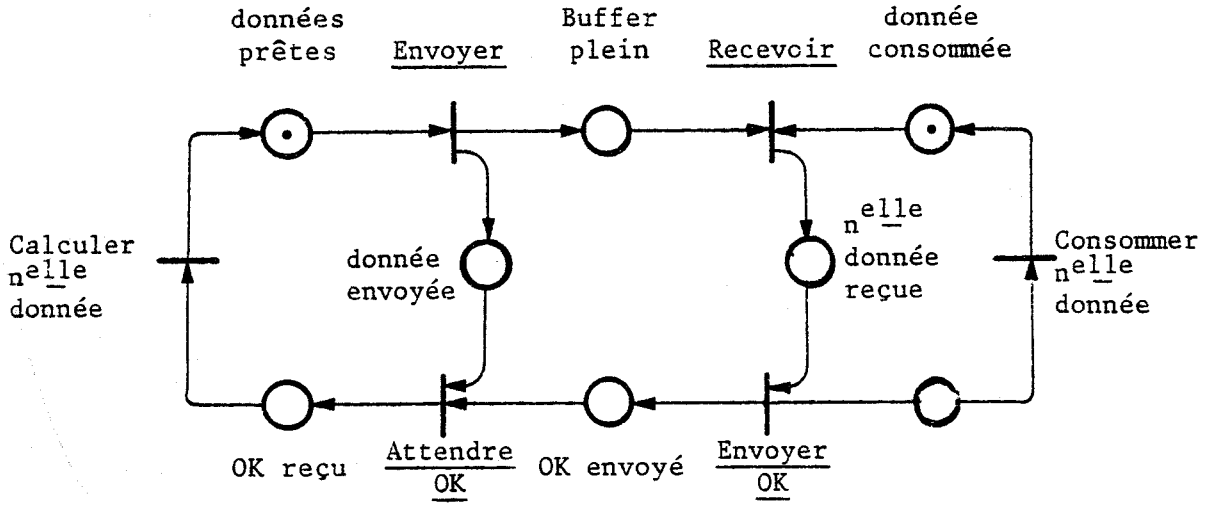


Figure II.11a

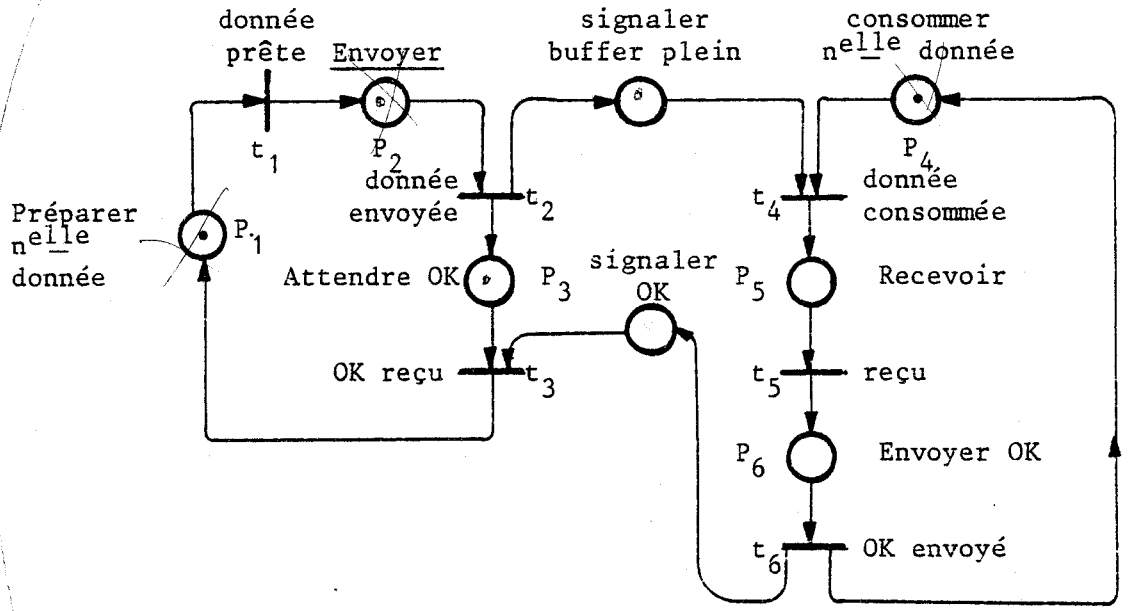


Figure II.11b

PROTOCOLE DE COMMUNICATION ENTRE PROCESSUS

Les réseaux des figures II.11a et II.11b modélisent un protocole de communication entre deux processus ; par exemple, un UART (Universal Asynchronous Receiver Transmitter) qui pilote une console de visualisation et un programme de calcul dont les résultats successifs doivent être visualisés.

Les deux figures représentent le même système mais ont été établies à partir de deux optiques différentes. Ainsi, sur la figure II.11a, la description du protocole de communication est orientée vers les données (résultat, acquittement) alors que la seconde fait plutôt apparaître la structure de contrôle.

Définition : Pour un réseau de Pétri représentant la structure de contrôle, l'interprétation est définie par la donnée :

- d'un domaine OP qui est l'ensemble des opérateurs
- d'un domaine C qui est un ensemble de conditions

prenant leur valeur dans {Vrai - Faux}

- d'une application I_{op} : $P \rightarrow OP$
- d'une application I_t : $T \rightarrow C$

Remarque : Une condition $c \in C$ représente la présence ou la non présence d'un évènement externe x, (on dit alors que la transition labellée par c est réceptive à l'évènement x) ou également l'état (vrai ou faux) d'un prédicat (évènement interne).

Les règles d'évolution pour ce type de réseau sont quelque peu modifiées. Une transition t est tirée :

- si elle est sensibilisée et
- si la condition qui lui est associée est vraie.

Ainsi, si l'on considère l'exemple de la figure II.11b, la transition t_1 ne pourra être tirée que si la place P_1 est marquée et la condition "donnée reçue" est vraie.

L'interprétation du marquage est également différente selon que l'on s'intéresse à l'un ou à l'autre type de représentation. Une marque d'un RdP décrivant le flux de donnée représente l'état d'une donnée ou d'une

ressource, alors que pour l'autre classe de réseau, elle représente le niveau d'évolution, le pointeur d'activité de chaque processus.

En définitive, un réseau de Pétri permet de modéliser la statique du système à l'aide de la structure du graphe et non de son interprétation, et également de son comportement dynamique par les règles d'évolution de son marquage.

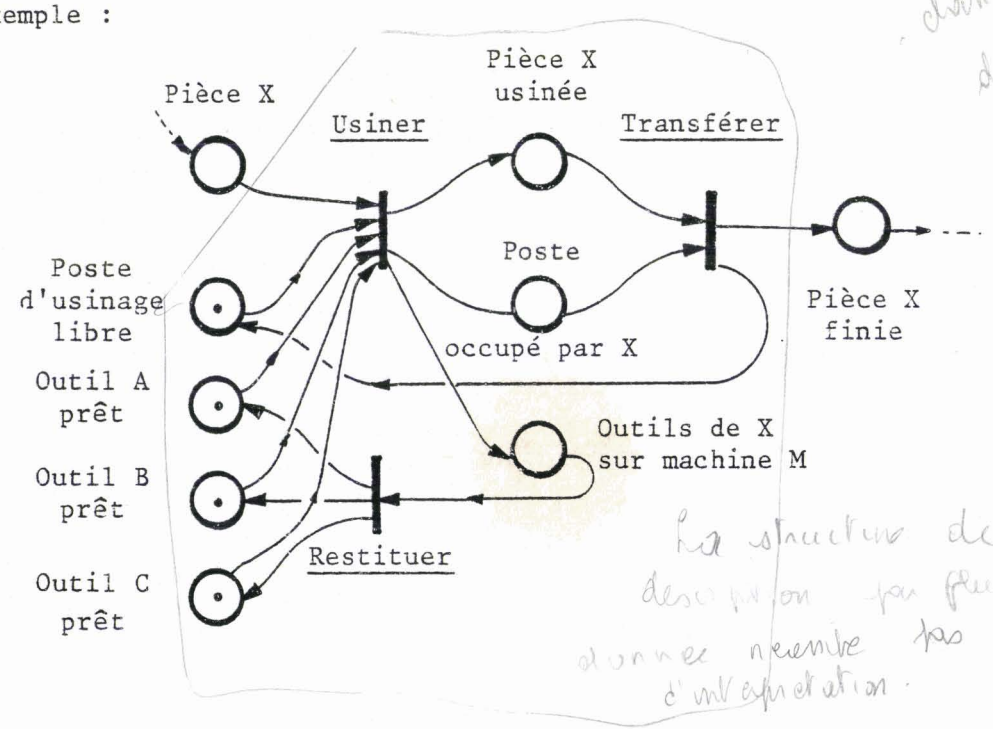
II.2.2 - Justification du choix des réseaux de Pétri

Les principales caractéristiques qui justifient l'utilisation de ce modèle pour la spécification et la conception de la classe des systèmes définis plus haut, sont les suivantes :

1) Les réseaux de Pétri permettent de représenter un système selon deux points de vue duaux :

- description par le flux de donnée
- description par le flux de contrôle

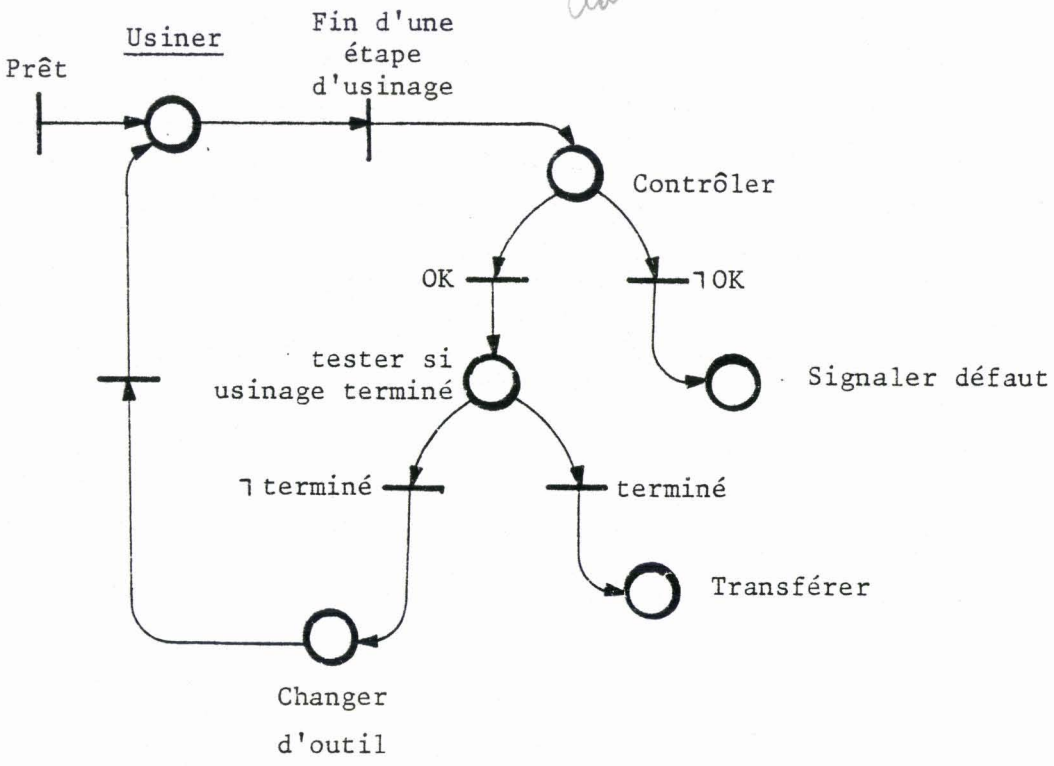
Exemple :



Portion de réseau de Pétri qui modélise une unité d'usinage d'une machine flexible dans un atelier flexible. Cette unité permet d'usiner plusieurs types de pièces et nécessite donc plusieurs types d'outils.

Figure II.12

Exemple :



Portion de RdP représentant un cycle usinage-contrôle-transfert

Figure II.13

Les éléments graphiques de représentation sont de plus, un atout important de cet outil permettant un dialogue facile entre client, maître d'œuvre et maître d'ouvrage.

2) La description des flux de donnée permet de réaliser de manière systématique la décomposition fonctionnelle du système.

Dans l'exemple de la figure II.12, les différentes fonctions ont été mises en évidence :

- Usiner pièce
- Transférer pièce usinée
- Restituer outillage.

(La décomposition de "Usiner" fait apparaître le traitement d'usinage, de contrôle et de chargement d'outil).

3) La description d'un système se fait par niveau d'abstraction. Il est en effet possible de remplacer une transition ou une place par un sous-réseau qui fournira un modèle plus précis de ce système.

Soit $\mathcal{R}$, un ensemble de réseaux de Pétri $\mathcal{R}'$ défini de la manière suivante :

$$\mathcal{R}'_{abs} = (G', M'_o)$$

où $G' = (P', T')$ est un graphe simplement connexe dont on distingue un ensemble de sommets initiaux S_i et un ensemble de sommets finaux S_f.

Définition : Un raffinement de représentation par réseau de Pétri $\mathcal{R} = (P, T, \Gamma, M_o)$ est défini par :

$$A_p : P \to \mathcal{R}$$

$$A_p(p) = G \quad \text{avec} \quad S_i \subset P' \text{ et } S_f \subset P'$$

Pour une seule place $p' \in S_i$, $M'_o(p') = M_o(p)$.

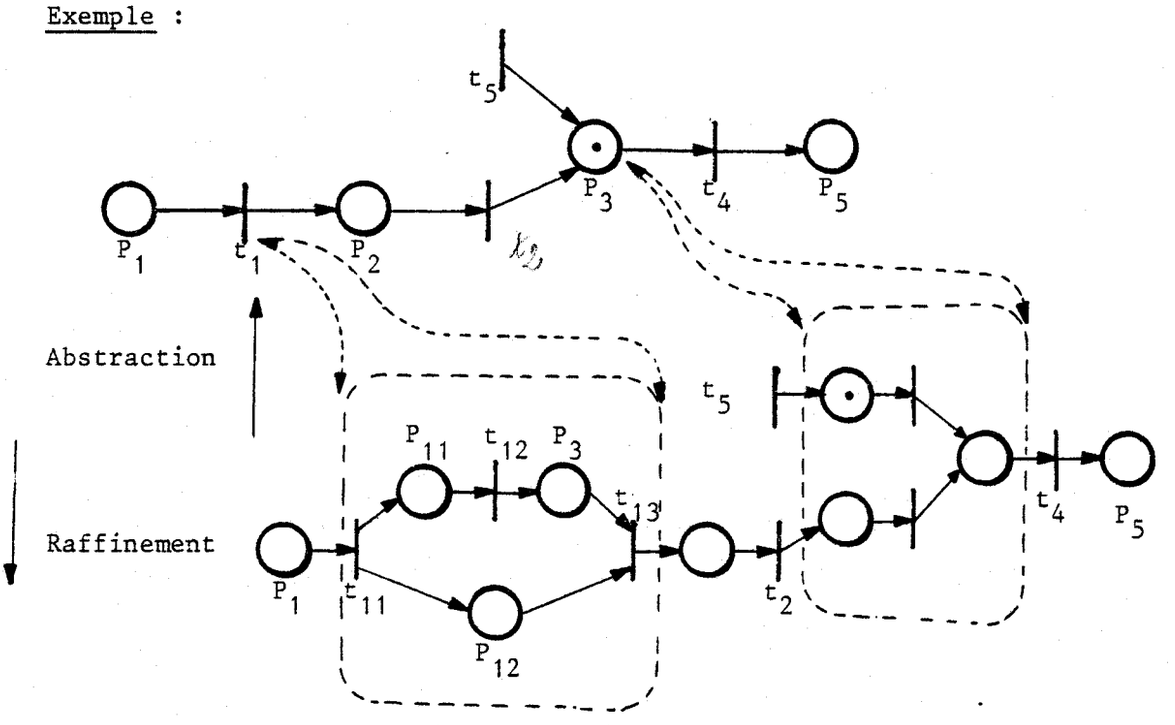
Pour toutes les autres $M'_o(p') = 0$

$$A_t : T \rightarrow R$$

telle que $A_t(t) = G$ avec $S_i \subset T$, $S_f \subset T$

$$\forall p' \in P' \qquad M_o(p') = 0$$

Exemple :



Raffinement de représentation

Figure II.14

4) Le passage du réseau à un programme peut être facilité soit par des transformations à postériori de la structure du graphe, soit en structurant à priori ces réseaux.

5) Ils constituent un outil formel dont les bases théoriques sont la théorie des graphes et l'algèbre linéaire.

6) Leur formalisme autorisent une vérification pratiquement totale des spécifications, à savoir :

- la vérification de la syntaxe du système par une analyse des propriétés intrinsèques du réseau

- la vérification partielle des propriétés du système (sa sémantique).

7) L'évaluation de certaines performances est également permise |32|. On peut par exemple, associer des durées, aux transitions ou aux places (suivant que l'on s'intéresse à tel ou tel autre type de réseau) et déterminer ainsi la durée totale d'exécution de certaines opérations.

Remarque : Le modèle obtenu est appelé réseau de Pétri temporisé |29|. L'unité centrale du CDC 6600 a été modélisé par ce type de réseau afin de déterminer dans quelle mesure il fallait augmenter le parallèlisme entre les différentes unités fonctionnelles pour minimiser le temps d'exécution |33|.

II.2.3 - Conclusion

Les réseaux de Pétri apparaissent comme un bon outil de spécification et de conception de système parallèle et concurrent. Les points forts ont été soulignés ; ils possèdent cependant certaines faiblesses qu'il faudra réduire ou éliminer afin de proposer une méthode cohérente et sûre de spécification et de conception.

Ainsi, ils ne proposent pas dans la phase de conception logique, de mécanisme d'aide à la définition des structures de données.

L'aspect graphique est intéressant mais peut également nuire à la compréhension du système, les décompositions successives font apparaître trop de détails. On obtient alors un graphe touffu et illisible (la représentation de tous les fonctionnements d'exception conduit en général à de tels graphes).

C'est en quelque sorte pour remédier à ce défaut, qu'ont été développées certaines variantes de réseau de Pétri ayant des possibilités de représentation plus puissante |16| |36| |23| au détriment des facilités d'analyse.

Il est de plus possible de proposer une conception automatique ou

à défaut assistée, des programmes en structurant à priori les réseaux c'est à dire en y incluant certaines primitives de contrôle séquentiel et parallèle.

II.3 - METHODOLOGIE DE SPECIFICATION ET DE CONCEPTION DE PROCESSUS INDUSTRIELS

II.3.1 - Principe

L'analyse structurée a apporté de nombreuses améliorations dans la spécification et la conception des systèmes. Elle constitue le principe de base de nombreuses méthodes de définition du cahier des charges. Elle s'applique également à la conception des programmes et a influencé la plupart des langages de programmation récents |13|.

La prise en compte des fondements de l'analyse et de la programmation structurée, au niveau de la définition même des réseaux de Pétri, constitue l'approche la plus sûre permettant d'aborder la complexité des grands systèmes de processus industriels. Elle conduit à la définition d'une classe particulière de réseaux de Pétri : les réseaux de Pétri structurés.

La structure de ce modèle permet une traduction aisée en langage de programmation, qui supprime toute interprétation erronée lors de l'écriture des programmes. Son intégration dans une méthodologie de spécification et de conception permet aussi d'assurer que les raffinements successifs de spécification (depuis la définition informelle du cahier des charges jusqu'au modèle final) n'ont pas introduit d'erreur.

II.3.2 - Réseau de Pétri structuré

II.3.2.1 - Processus assimilé à un réseau de Pétri :

La notion de processus correspond ici à celle rencontrée en informatique dans les systèmes opératoires |12|. Un processus peut être assimilé à un programme* séquentiel. (* Programme est pris au sens large, il correspond à un enchaînement d'instructions ou plus généralement à un enchaînement de tâches).

Dans ce sens, il peut être représenté par un réseau de Pétri $R = (P, T, \Gamma, M_o)$ qui possède les caractéristiques suivantes :

(1) $\forall t \in T$ alors $|\Gamma^{-1}(t)| = |\Gamma(t)| = 1$

(chaque transition n'a qu'une et une seule place d'entrée et une et une seule place de sortie).

(2) $\forall M \in \mathcal{E}(M_o)$ alors $\sum_{p \in P} M(p) = 1$

(le nombre total de marqueurs de R est toujours égal à 1).

(3) Le graphe G est fortement connexe.

(il existe un chemin qui relie deux sommets distincts).

En particulier :

$\forall p \in P$ $\Gamma(p)$ et $\Gamma^{-1}(p)$ existent

$\forall t \in T$ $\Gamma(t)$ et $\Gamma^{-1}(t)$ existent

Le réseau R est donc un graphe d'état marqué.

Propriété 1 : Le réseau de Pétri d'un processus est sain.

Preuve : Soit $m_i = M(p_i)$, $n = |P|$, $\forall M \in \mathcal{E}(M_o)$ $\forall p \in P$

$m_i \in \mathbb{N}^+$

alors $\forall M \in \mathcal{E}(M_o)$ d'après (2), nous avons

$\sum_{i=1}^{n} m_i = 1$; d'où $m_i \leq 1$ $\forall i = 1$ à n

Le réseau est sain.

Si $m_j = 1$ alors $\forall i \neq j$, nous avons $m_i = 0$ (une et une seule place est marquée pour un marquage donné).

Propriété 2 : Le réseau de Pétri d'un processus est vivant.

Preuve : Soit M, un marquage quelconque de $\mathcal{E}(M_o)$.

$\exists p_o \in P$ tq $M(p_o) = 1$ (d'après la propriété 1).

Supposons que la transition t soit non-vivante. Alors, l'unique place d'entrée p de cette transition ($p = \Gamma^{-1}(t)$) est telle que $\forall M \in \mathcal{E}(M_o)$ $M(p) = 0$ et donc, toutes les transitions de sortie de la place p (y compris t) sont non-vivantes ainsi que toutes ses transitions d'entrée. Ainsi, si une transition est non-vivante, toutes celles à qui elle est reliée par un chemin de longueur 2* sont également non-vivantes. Le graphe étant fortement connexe, on en déduit que toutes les transitions sont non-vivantes et donc que $\forall M \in \mathcal{E}(M_o)$ et $\forall p \in P$, $M(p) = 0$ ce qui contredit la propriété de ce réseau :

$$\exists\, p_o \in P \quad \text{et} \quad \exists\, M \in \mathcal{E}(M_o) \qquad M(p) = 1$$

Ce réseau est donc vivant.

Propriété 3 : Le réseau de Pétri d'un processus est propre (réinitialisable).

Preuve : Soit $p_o \in P$ tq $M_o(p_o) = 1$ et $M_o(p) = 0$ pour $p \in P - \{p_o\}$.
Soit t_o, une place d'entrée de p_o.
Le réseau est sain et, d'après la propriété 2, t_o est vivante ; il existe donc une séquence de tir depuis un marquage $M \in \mathcal{E}(M_o)$ contenant t_o qui conduit à un marquage M' tel que :

$$M'(p_o) = 1 \quad \text{et} \quad M'(p) = 0 \quad \text{pour} \quad p \in P - \{p_o\}$$

Le marquage M' est donc identique à M_o.

Le réseau est donc également propre.

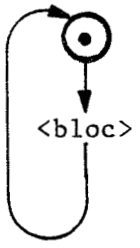
* Rappelons que la longueur d'un chemin est égale au nombre d'arcs qu'il comporte.

La construction d'un graphe de processus utilise un nombre limité de structures élémentaires qui sont :

- la structure séquentielle
- la structure alternative
- la structure répétitive.

Leur organisation respecte les règles de production qui définissent la grammaire du graphe présentée ci-dessous.

L'axiome de départ se traduit de la manière suivante :



(a)

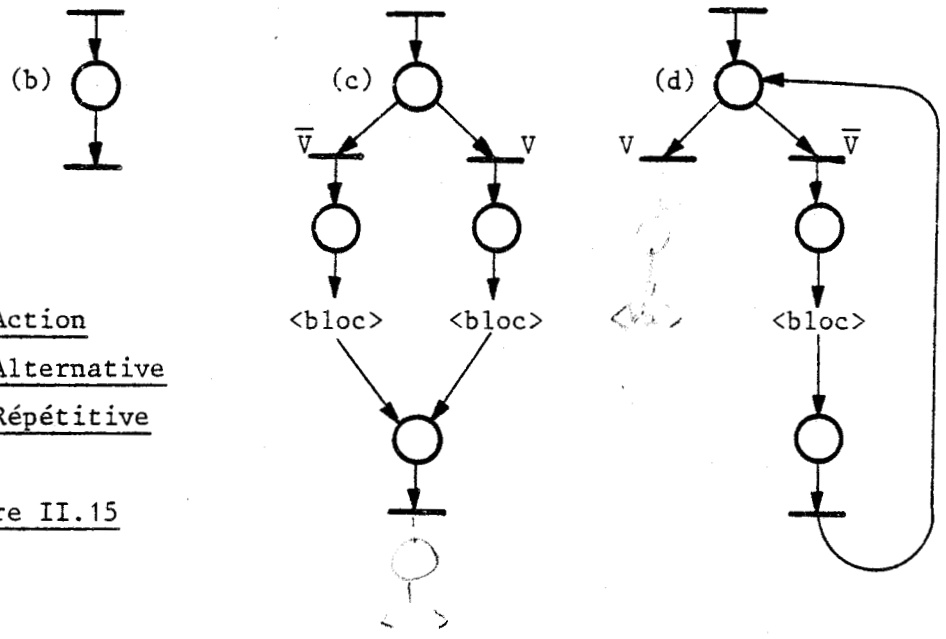
ou en utilisant la notation BNF (Backus-Naur Form) :

<processus> ::= <tâche initiale> <bloc>

<tâche initiale> ::= PLACE MARQUEUR

<bloc> ::= <action> | <action> PLACE <bloc> |
<alternative> | <alternative> PLACE <bloc> |
<répétitive> | <répétitive> PLACE <bloc> |

Les 3 structures de base (action, alternative, répétitive) s'écrivent en termes de réseaux de Pétri de la manière suivante :



(b) Action
(c) Alternative
(d) Répétitive

Figure II.15

Remarque : Böhm et Jacopini |4| ont montré que tout schéma de programme peut être obtenu à partir de ces 3 structures. Leur utilisation ne restreint donc pas les possibilités de représentation du modèle. Par contre, l'introduction des clauses SI (pour l'alternative) et TANT QUE (pour la répétitive) permet une traduction rapide en langage de programmation.

II.3.2.2 - Système de processus :

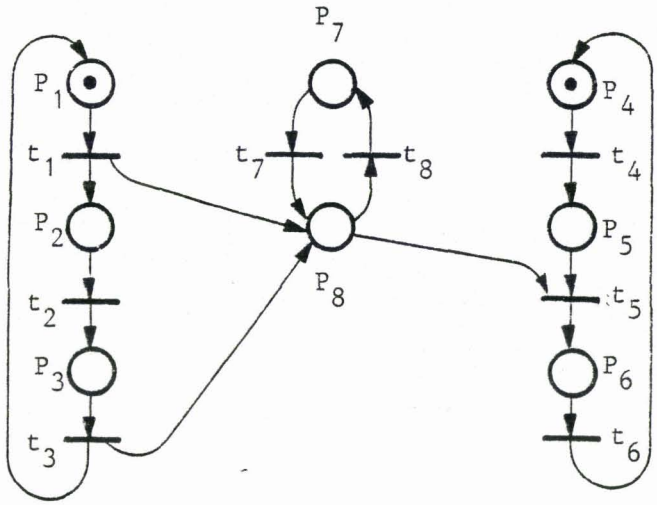
Pour prendre en considération le parallélisme d'un système, il est nécessaire d'introduire plusieurs processus (un processus est un graphe d'état qui ne permet donc pas d'exprimer le parallélisme). L'évolution de chacun de ces processus tient compte, en règle générale, de l'état des autres processus. Il est donc nécessaire de pouvoir représenter leur relation par des liaisons cohérentes.

Soit $\mathcal{P}$ un ensemble de processus décrits chacun par un réseau de Pétri $\mathcal{P}_i = (G_i, M_o^i)$.

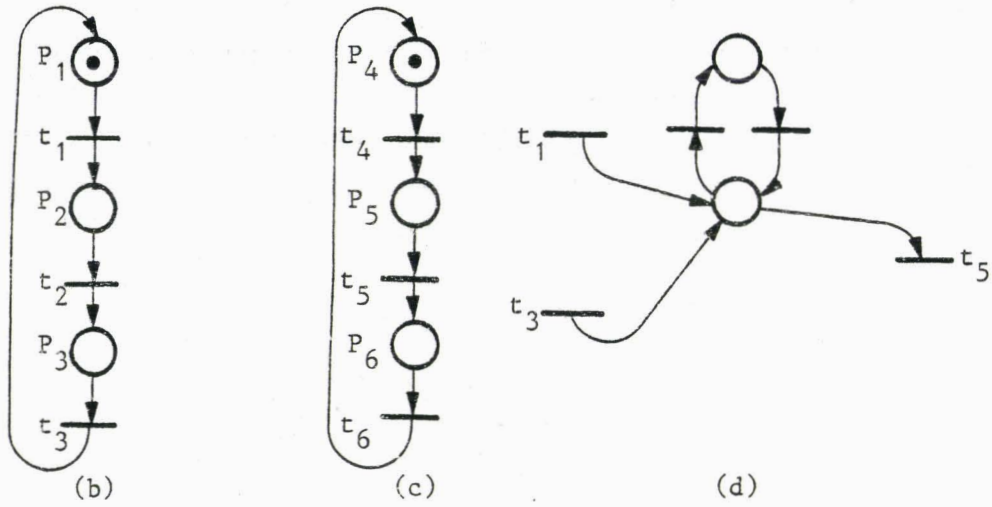
Une liaison entre deux processus $\mathcal{P}_i$ et $\mathcal{P}_j \in S$ est définie par un graphe simplement connexe $\ell = (P, T, \Gamma)$, éventuellement marqué, dont les sommets et extrémités appartiennent aux graphes des processus $\mathcal{P}_i$ et $\mathcal{P}_j$.

L'ensemble des liaisons d'un système de processus constitue le graphe de liaison L |6|.

Exemple : Le système présenté ci-dessous fait apparaître deux processus et une liaison.



(a)
Système de processus



(b) (c) (d)

Le processus P_1 Le processus P_2 La liaison ℓ_{12}

DECOMPOSITION D'UN SYSTEME DE PROCESSUS

Figure II.16

Définition : Un système de processus est défini par la donnée :

- de n processus $\mathcal{P}_1, \ldots, \mathcal{P}_n$
- d'un graphe de liaison $\mathcal{L}$

et donc $S = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{L})$

L'étude des liaisons de longueur 2 de la forme (t, p, t') |6| montre qu'elles permettent :

- de représenter les principaux types de communication et d'intéraction qui apparaissent dans le système

- de minimiser les risques de blocage entre processus.

A partir de ces remarques, 4 types de liaison ont été proposés.

* Liaison de partage de ressource à 1 seul état (ou encore liaison de type exclusion mutuelle) :

Définition : n ($n \geq 2$) processus sont liés par une liaison de type exclusion mutuelle si leur déroulement respectif nécessite l'utilisation d'une ressource à accès exclusif.

Exemple : Deux processus $\mathcal{P}_1$ et $\mathcal{P}_2$ se partagent l'utilisation d'un fichier texte, le premier pour y écrire, le second pour y effectuer des lectures.

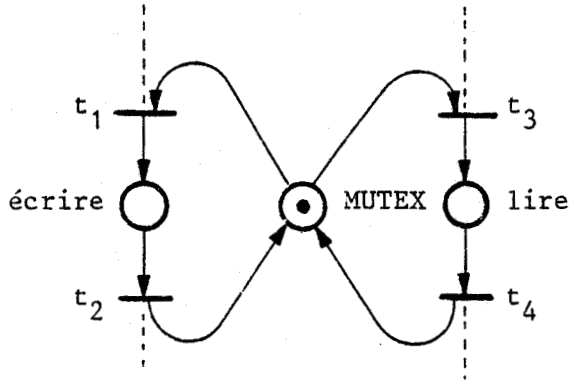


Figure II.17

L'utilisation du fichier par le processus rédacteur place le processus lecteur, qui était éventuellement prêt à acquérir la ressource, en état d'attente. La libération de celle-ci par le processus rédacteur permet à P_2 qui était en état d'attente, de l'utiliser.

Ce mécanisme est semblable à ceux qui utilisent des sémaphores . La place étiquetée par MUTEX peut, en effet, être interprétée comme un sémaphore booléen initialisé à 1. Les tirs des transitions t_1 et t_3 correspondent à des opérations P sur MUTEX et ceux de t_2 et t_4 à des opérations V sur MUTEX.

Exemple : Problème du lecteur-rédacteur de la figure II.17

```
Sémaphore MUTEX ; (valeur initiale = 1)
Processus rédacteur : début          Processus lecteur : début
                ... ;                                ... ;
                P (MUTEX) ;                          V (MUTEX) ;
                ECRIRE ;                             LIRE ;
                V (MUTEX) ;                          P (MUTEX) ;
                ... ;                                ... ;
                fin                                  fin
```

De même, nous définirons une section critique de processus comme étant l'ensemble des opérations qui utilisent une ressource dont l'accès est protégé par un sémaphore. Dans l'exemple ci-dessus, la section critique de REDACTEUR est réduite à l'opération "écrire".

Remarque : La possibilité de demande de ressourec simultanée nécessite d'introduire la notion de priorité d'accès. Le conflit peut par exemple, se résoudre de la manière suivante :

Exemple : le processus rédacteur est prioritaire par rapport au processus lecteur en cas de demande simultanée.

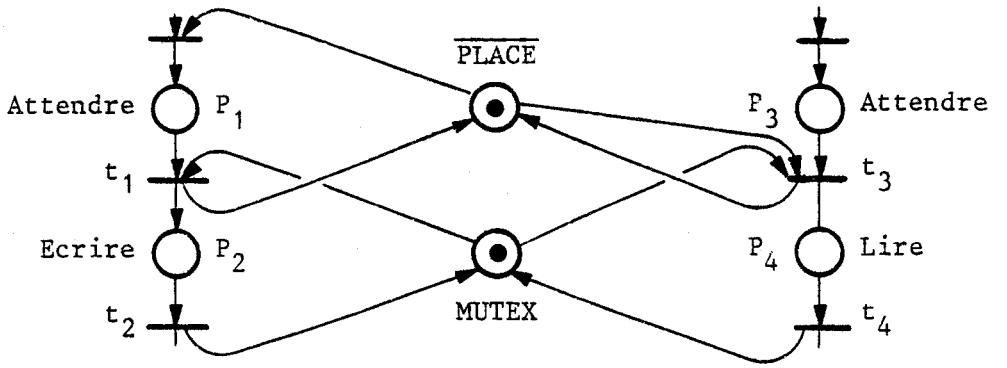


Figure II.18

Si P_1 et P_2 sont tous deux prêts à acquérir la ressource (les places p_1 et p_3 sont marquées), alors, la place complémentaire n'est plus marquée et seule la transition t_1 est déclenchable.

Remarque : La marque de la place MUTEX peut être interprétée comme étant la ressource partagée (objet).
Il est alors possible de généraliser le mécanisme d'exclusion mutuelle en considérant que MUTEX est un sémaphore à compte, initialisé au nombre de ressources disponibles.

* Liaison de partage de ressources à deux états (ou encore liaison de type producteur-consommateur) :

Définition : n ($n \geq 2$) processus sont liés par une relation de producteur-consommateur si leur déroulement respectif nécessite l'utilisation d'une ressource qui peut prendre deux états.

Chacun de ces processus acquiert une ressource dans un état, et la restitue dans l'autre état. Parmi ces n processus, il existe au moins un processus qui l'utilise dans un état et un processus qui l'utilise dans l'autre état.

Exemple : Une flotte de camions fait la navette entre une usine et un entrepôt. Chaque camion est déchargé à l'entrepôt des marchandises qui avaient été chargées à l'usine.

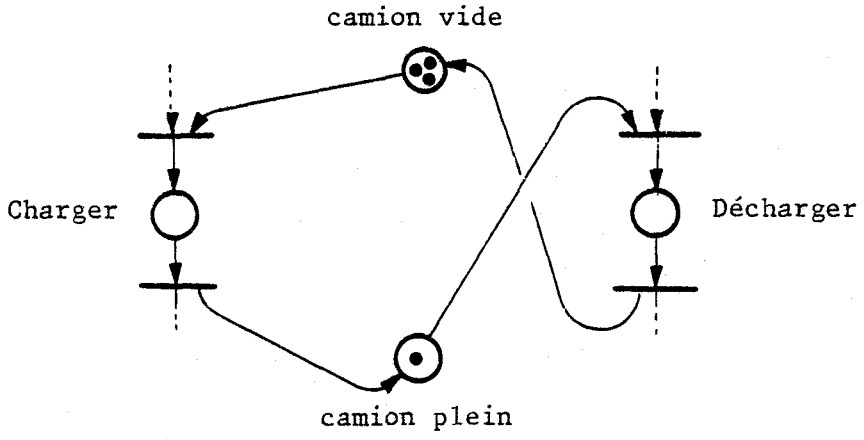


Figure II.19

Remarque : Les marqueurs que contiennent les places de liaison symbolisent, à un instant donné, les divers exemplaires de la ressource dans l'un ou l'autre état.
Dans notre exemple, à l'instant où est prise l'image du réseau, il y a 3 camions vides et 1 camion plein.

* Liaison de synchronisation :

Elle permet d'indiquer dans quel ordre des opérations appartenant à des processus différents, vont s'enchaîner.

Exemple :

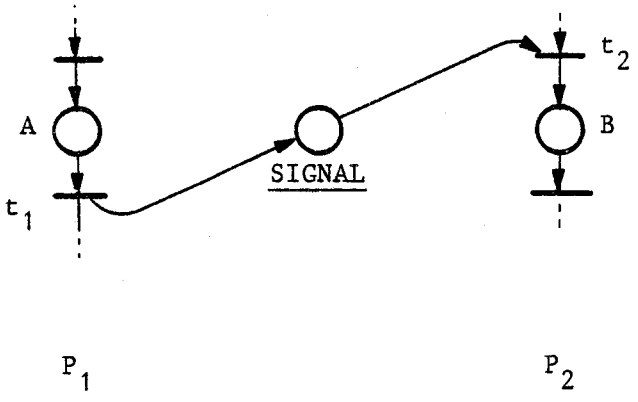


Figure II.20

Dans l'exemple ci-dessus, l'opération B est précédée dans le temps, par l'opération A du processus P_1. La transition t_2, qui correspond à A l'activation de la tâche B, est dite synchronisée par la transition t_1 qui synchronise la fin de l'opération A.

Propriété : La place de synchronisation a une propriété particulière : comme les étapes du GRAFCET, elle est absorbante (le nombre de jetons qu'elle possède est au plus, égal à 1).

Justification : Le marqueur de cette place n'a pas la même signification que celle des places d'exclusion mutuelle ou de producteur/consommateur (elle ne symbolise pas une ressource objet). Elle correspond plutôt à un signal qui ne peut prendre que deux états : absent ou présent. Cette propriété permet d'assurer que le réseau ci-dessous est vivant.

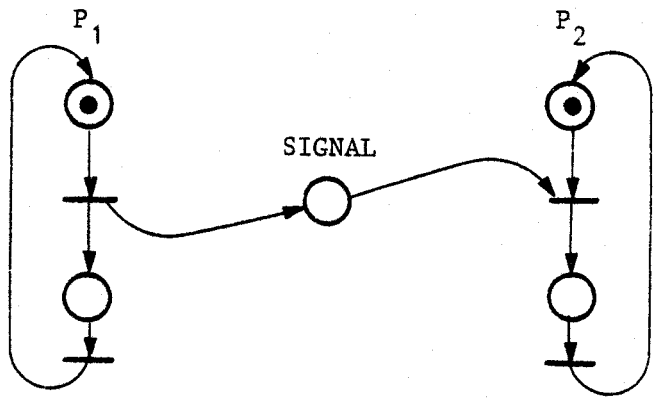


Figure II.21

(Si aucune contrainte sur le marquage de la place SIGNAL n'est faite, alors celui-ci peut croître indéfiniment si le processus P_2 n'évolue pas depuis son marquage initial).

Un mécanisme de synchronisation également important dans la modélisation des systèmes industriels est présenté ci-dessous.

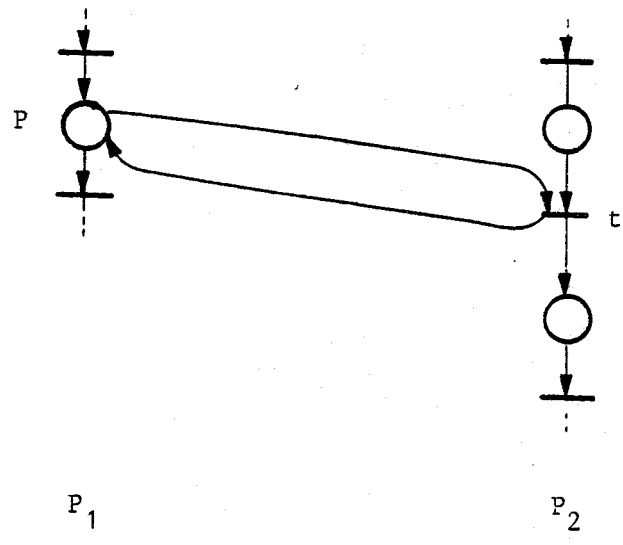


Figure II.23

Le franchissement de t est conditionné par la présence d'une marque dans la place P du processus P_1. Le tir de t ne modifie pas le marquage de P. Il s'agit donc simplement d'une lecture de l'état d'une place.

Ce type de synchronisation est utilisé en Grafcet |29|. La norme Grafcet n'impose pas la représentation des arcs, il faut alors rappeler dans l'étiquette de déclenchement de la transition t, le numéro de l'étape (place) dont l'activité est testée.

Remarque : La suppression des arcs a pour conséquence de masquer les possibilités de blocage du graphe.

Exemple :

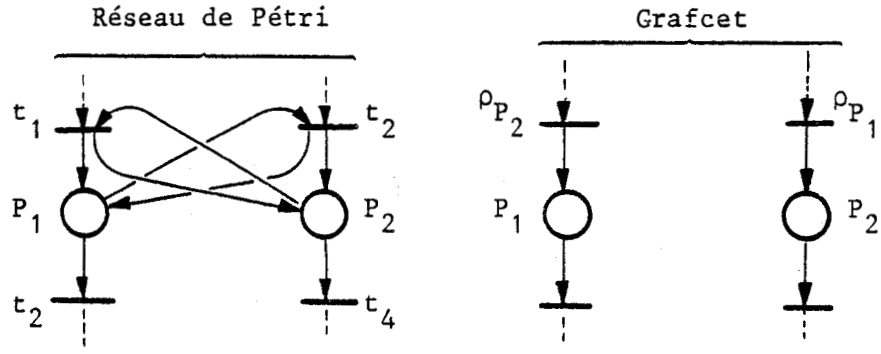


Figure II.24

Ce type d'intéraction entre deux processus peut également être représenté à l'aide d'une liaison de synchronisation, comme le montre la figure suivante.

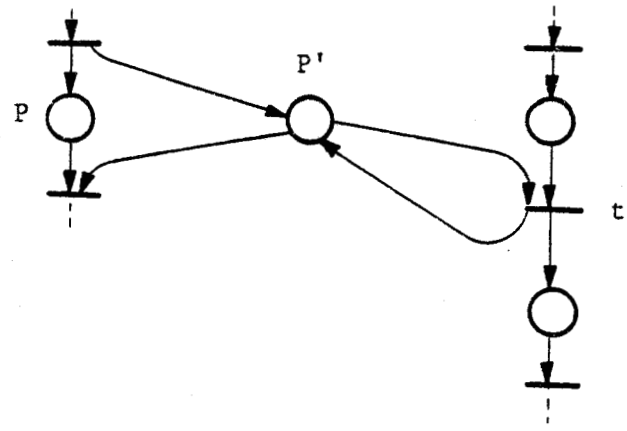


Figure II.25

Le principe est un peu analogue à celui que nous avons présenté pour décrire le test de non présence d'une marque dans une place (qui peut

également être décrit à l'aide d'un arc inhibiteur).

On crée ici une place supplémentaire qui constitue notre place de synchronisation.

* Liaison hiérarchique (de type appel de sous-programme) :

Elle permet de définir une relation de hiérarchie entre processus d'un même système.

Un processus P_1 utilise à plusieurs stades de son évolution une même ressource représentée par un processus P_2 qui joue ainsi le rôle d'un sous-programme.

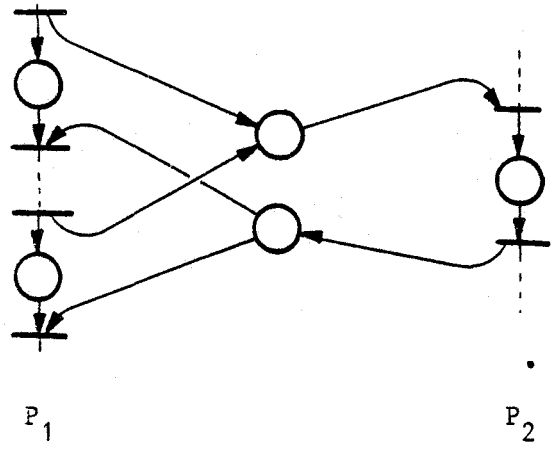


Figure II.26

Les 4 liaisons qui représentent les principaux modes de communication et d'intéraction entre processus, mettent en évidence les 3 types de ressources susceptibles d'être utilisées par une tâche :

- ressource processeur qui correspond au marqueur d'un processus hors des sections critiques
- ressource objet symbolisée par un ou plusieurs marqueurs dans les places des liaisons d'exclusion mutuelle et de producteur/consommateur
- ressource processus dans le cas de la liaison de type appel de sous-programme.

Il semble alors que le choix de ces liaisons et de leur nombre, n'amènent aucune restriction au niveau du pouvoir de représentation du modèle. Nous verrons d'ailleurs plus loin, qu'ils permettent de simplifier considèrablement l'analyse du modèle. Il est d'ailleurs toujours possible de combiner leur emploi pour réaliser des modèles de communication plus complexes.

II.3.3 - Méthodologie de spécification et de conception |7| à |11|

II.3.3.1 - Décomposition du système :

La procédure qui permet de faire apparaître les différentes unités fonctionnelles utilisent, comme il a été indiqué en II.2, une description du système par les flots de donnée. L'organisation de ces unités dans un système de processus doit respecter un certain nombre de contraintes qui sont :

- le parallélisme possible de ces fonctions (ce qui permet de minimiser la durée d'exécution du procédé)

- la minimisation du nombre de processus (qui constitue une contrainte opérationnelle)

- la cohérence fonctionnelle qui permet une maintenance du système plus fiable

- etc ...

Les critères sont en génèral conflictuels. Aussi, des analyses permettant d'évaluer le modèle sont souvent nécessaires avant d'établir la structure définitive du système |32|.

La conception se fait par raffinement successif et utilise des règles qui assurent que chaque niveau d'abstraction du modèle est correct.

Dans la phase de conception, lorsque la structure du système est définie, c'est-à-dire lorsque la décomposition en processus a été réalisée, la modélisation s'effectue en deux étapes :

- description de chaque processus exécuté indépendamment des autres processus,

puis, lorsque tous les processus ont été définis,

- description des intéractions entre les différents processus, afin d'exprimer les contraintes sur l'utilisation des ressources.

L'utilisation des réseaux de Pétri structurés est particulièrement adaptée à cette phase d'étude. Elle amène cependant quelques contraintes supplémentaires, en particulier sur les mécanismes de raffinement.

Un raffinement permet de transformer un sommet du réseau en un graphe de Pétri simplement connexe à un seul point d'entrée et un seul point de sortie qui respecte la grammaire des réseaux de Pétri structurés. En fait, chaque niveau de définition d'un système utilise l'abstraction procédurale. Celle-ci autorise la déclaration d'un bloc sans nécessiter immédiatement de définir son contenu. Un raffinement de bloc consiste donc à préciser le contenu d'un bloc en utilisant la syntaxe définie au § II.3.2 . Cette règle permet de conserver les propriétés du réseau initial.

II.3.3.2 - Règle de construction du modèle :

La notion de bloc est intéressante à deux titres :

- elle permet de définir les sommets d'entrée et les sommets de sortie du graphe de liaison. On dira en effet, que le bloc A de P_1 synchronise le bloc B de P_2, ou encore, le bloc A de P_1 et le bloc B de P_2 se partagent en exclusion mutuelle la ressource R ...

- elle permet également de définir les points de reprise indispensable à l'expression des procédures de reprise |8|.

Ainsi, par exemple, une ressource utilisée dans un bloc, doit être restituée à la fin de ce bloc. Cette règle permet d'éviter le blocage dû à une mauvaise utilisation des ressources, comme dans l'exemple ci-dessous.

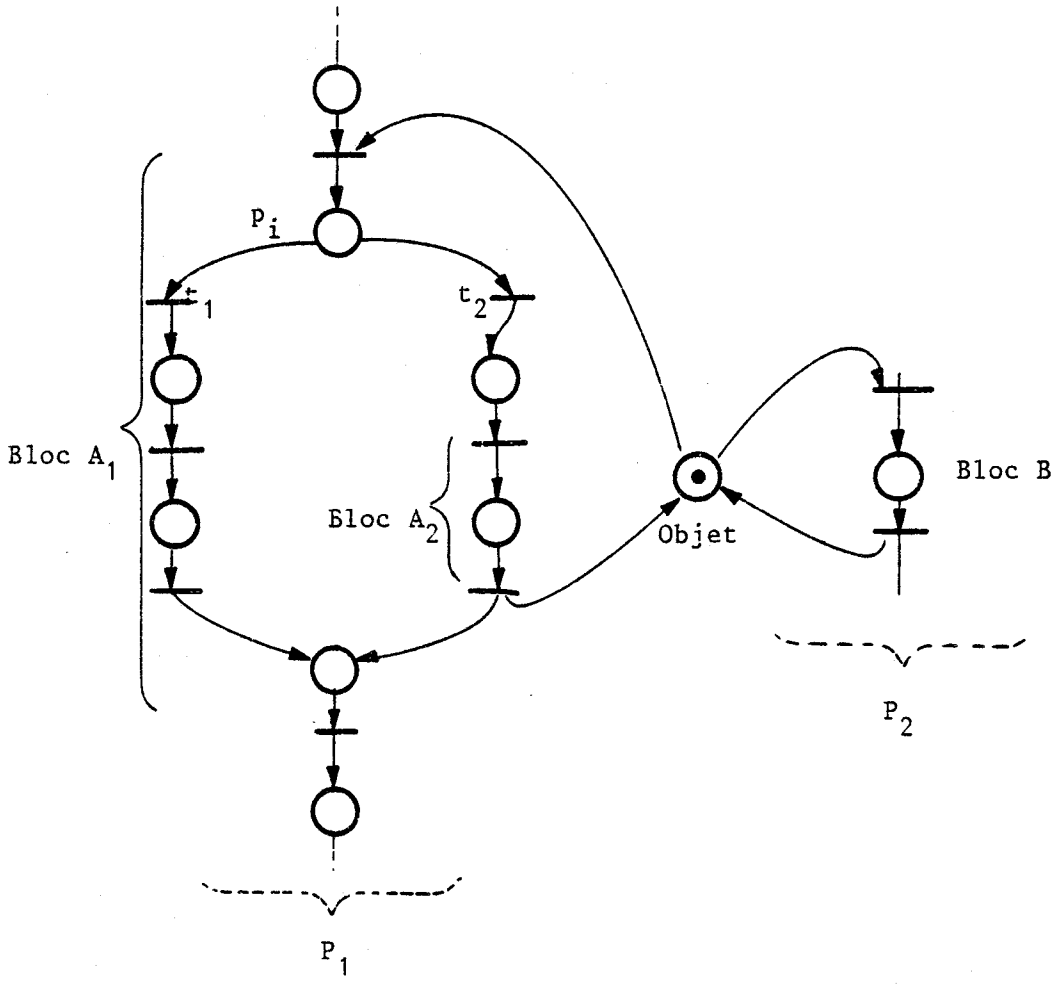


Figure II.27

Considérons l'exemple présenté ci-dessus. Supposons que le bloc A_1 du processus P_1 acquiert la ressource "objet". Deux utilisations de cette ressource sont possibles et correspondent sur le graphe aux deux chemins qui partent de la place p_i.

- Si la ressource est utilisée dans les opérations qui s'enchaînent depuis le tir de la transition t_2, alors le fonctionnement du système de processus semble correct.

- Par contre, si "objet" n'est pas utilisé dans les opérations qui succèdent à la mise à feu de la transition t_1, alors le bloc A_1 se termine sans avoir restitué la ressource. A la prochaine exécution du bloc A_1, le système sera totalement bloqué puisque "objet" n'existera plus.

Les deux représentations ci-dessous, respectivement en pointillé et en trait mixte, sont autorisées. Dans le premier cas, on dit que les blocs A_2 de P_1 et B de P_2 se partagent la ressource R ; dans l'autre cas, ce sera A_1 et B qui utiliseront en exclusion mutuelle la ressource R.

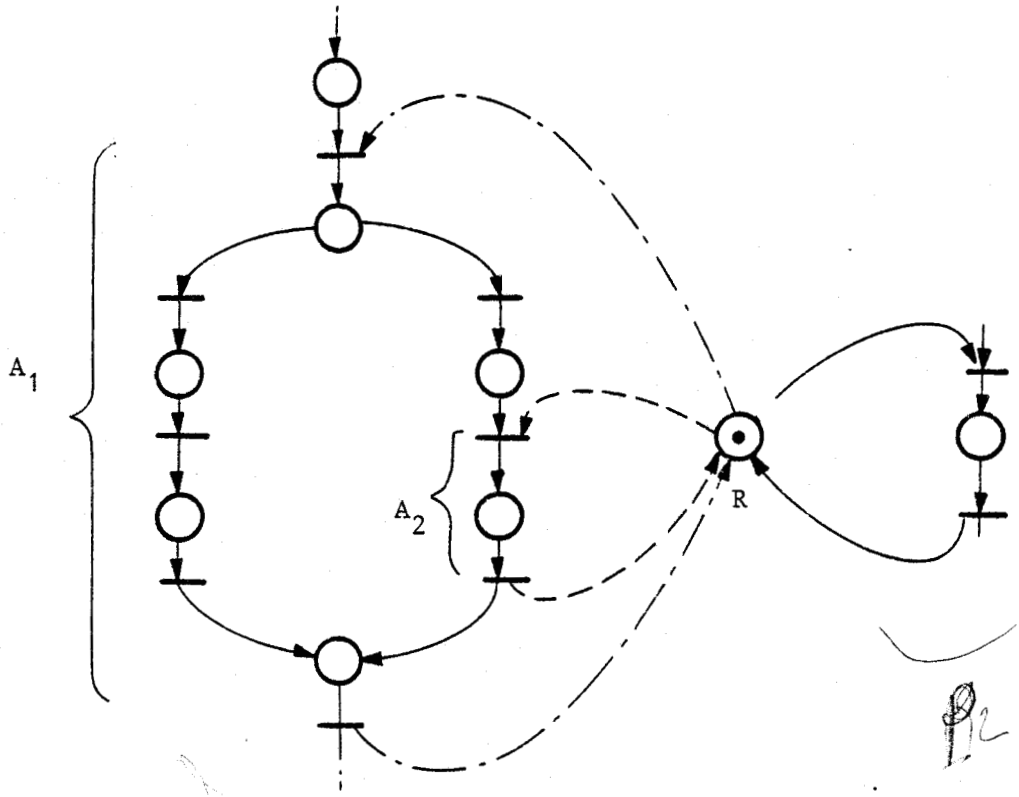


Figure II.28

De plus, il est nécessaire que les sections critiques d'un processus associées à des ressources partagées en exclusion mutuelle, soient disjointes ou incluses totalement l'une dans l'autre, ceci pour éviter le type de blocage présenté ci-dessous.

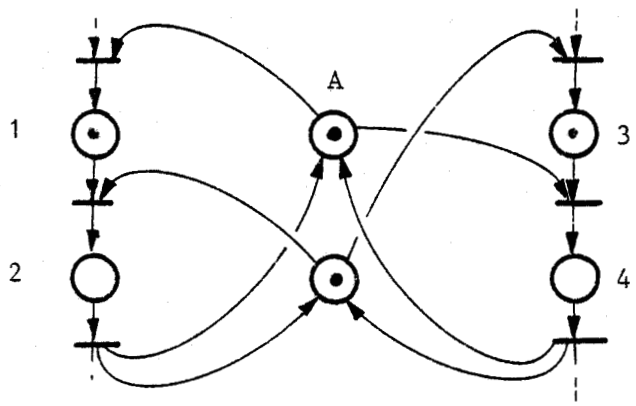


Figure II.29

Les sections critiques pour la ressource A comprennent les places de $P_{SCA} = (1,2,4)$.

Les sections critiques pour la ressource B comprennent les places de $P_{SCB} = (2,3,4)$.

Il est alors facile de vérifier que si les places 1 et 3 sont marquées, alors le système constitué de P_1 et P_2 est bloqué. Pour cet exemple, on a :

$$P_{SCB} \not\subset P_{SCA}$$
$$P_{SCA} \not\subset P_{SCB}$$
$$P_{SCA} \cap P_{SCB} \neq \emptyset$$

Le découpage en bloc de chaque processus doit donc être tel que, ou la relation d'inclusion des supports de sections critiques associées aux ressources partagées par 2 processus (dans l'exemple P_{SCA} et P_{SCB}) est vérifiée, ou que l'intersection de ces supports est vide.

$$P_{SCA} \cap P_{SCB} = \emptyset$$

Les règles qui viennent d'être citées visent à minimiser les erreurs de conception, il est cependant nécessaire de valider chaque niveau de définition avant de passer à un niveau de définition plus détaillée. (Ces vérifications seront abordées par la suite).

Remarque : Selon la complexité du problème, il est possible de construire directement les réseaux de Pétri structurés du système sans passer par le graphe qui exprime les flots de données.

II.3.3.3 - Les outils complémentaires :

Cette méthode intègre également d'autres outils qui sont d'ailleurs complémentaires aux réseaux de Pétri et qui permettent dans ce sens d'augmenter le pouvoir de représentation, de conception des systèmes.

a) Un pseudo-langage respectant la grammaire du langage des réseaux de Pétri peut être utilisé pour pallier au défaut que peuvent présenter les réseaux de Pétri lorsque le niveau de détail est trop avan-

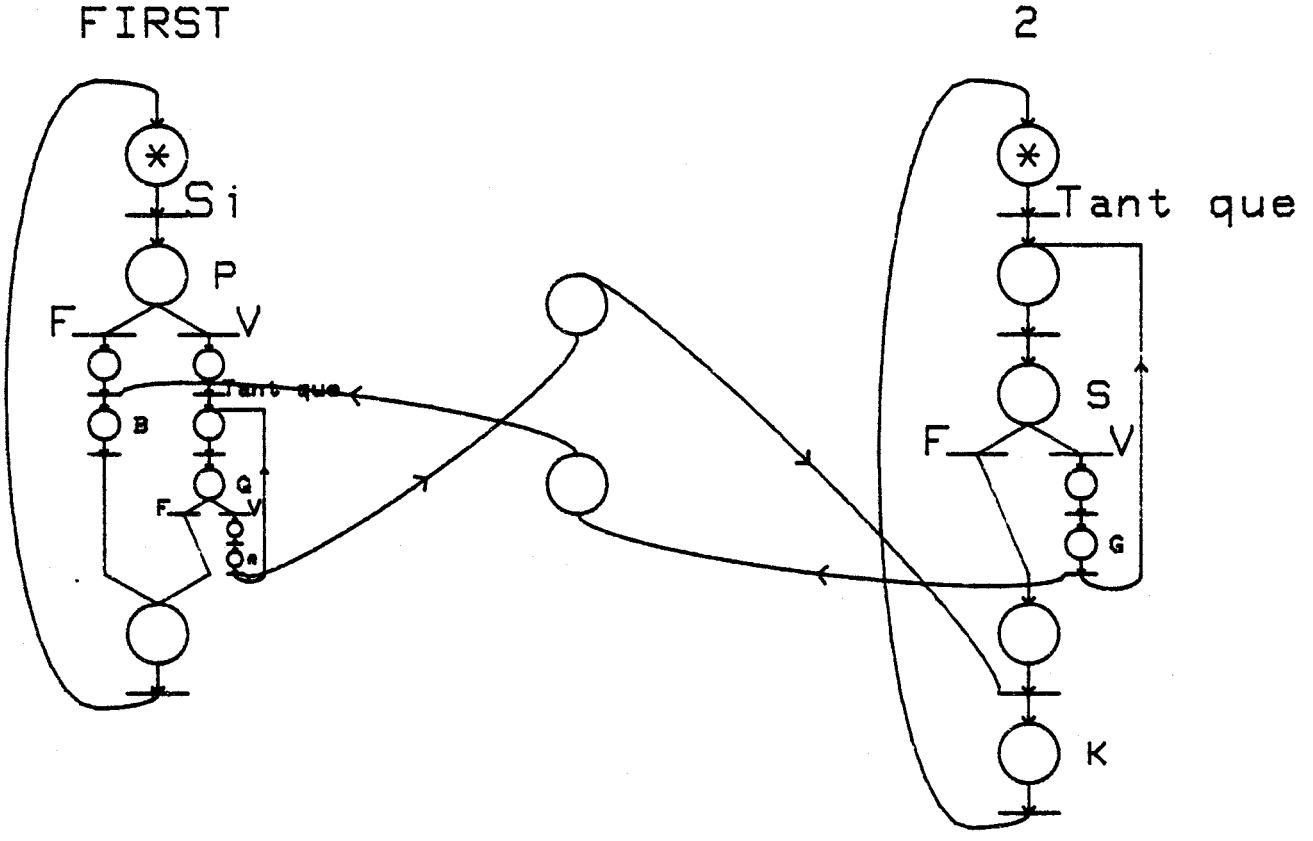
cé. Il permet également une implémentation directe. Des outils permettant le passage à l'un ou l'autre des modèles ont été développés.

Exemple :

Réseau de PETRI TEST3

```
10    ! PROCESSUS FIRST
20    !  SI P
30    !  ALORS
40    !   TQUE Q
50    !   FAIRE
60    !    BLOC R
70    !     ACTION A
80    !    FBLOC
90    !   FTQUE
100   !  SINON
101   !   BLOC F
110   !    ACTION B
111   !   FBLOC
120   !  FSI
130   ! FPROCESSUS
140   !
150   ! PROCESSUS 2
160   !  TQUE S
170   !  FAIRE
171   !   BLOC W
180   !    ACTION G
181   !   FBLOC
190   !  FTQUE
200   !  BLOC J
210   !   ACTION K
220   !  FBLOC
230   ! FPROCESSUS
240   !
250   ! LIEN
260   !  SYNCHRO (R>J),(W>F)
270 END

Syntaxe correcte
```



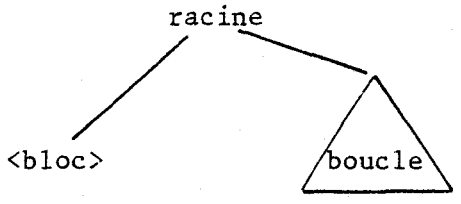
b) Un schéma de processus peut également être défini à l'aide d'une grammaire d'arbre adaptée de celle définie pour les réseaux de Pétri structurés. Ce modèle permet la preuve formelle du processus.

Cependant, les possibilités de représentation de ce modèle sont réduites. Il ne permet pas en effet, de faire apparaître de façon très explicite les intéractions entre processus.

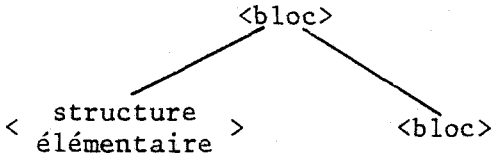
Le passage de ce modèle au réseau de Pétri structuré est aisé compte tenu des règles de production semblables de ces 2 modèles.

Exemple : Pour l'arbre programmatique :

a)



b)



Une structure élémentaire est décrite selon les règles suivantes :

c)

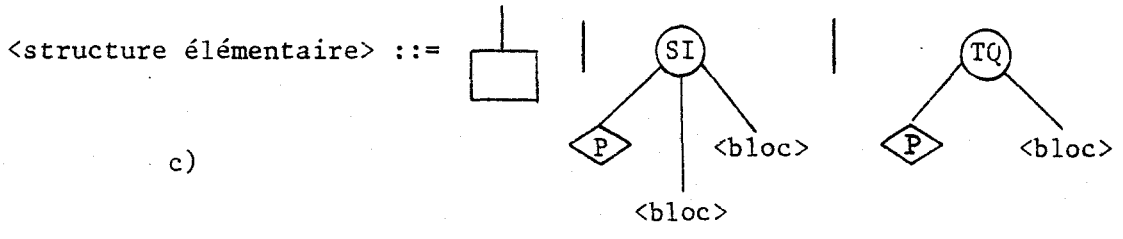
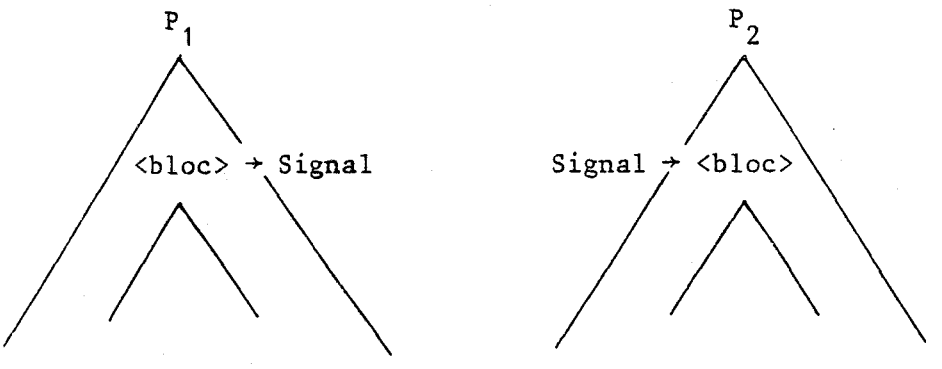


Figure II.30

La représentation d'une intéraction entre un bloc d'un processus P_1 et un bloc d'un processus P_2 est représenté de la façon suivante :



Synchronisation entre deux blocs appartenant à deux processus différents

Figure II.31

c) Un logiciel permettant la traduction du réseau de Pétri dans un code qui est ensuite interprété et exécuté sur différentes machines cibles |22|.

CONCLUSION

Nous avons dans ce chapitre, rappelé les principales définitions et propriétés des réseaux de Pétri et indiqué dans quelle mesure ils pouvaient constituer un outil de spécification et de conception.

Ils apparaissent particulièrement adaptés aux systèmes de commande parallèle mais présentent cependant quelques imperfections nécessitant l'introduction, au niveau de la définition même du modèle, de contraintes supplémentaires visant à :

- Conserver une bonne sûreté de fonctionnement dans le processus de raffinement

- structurer le modèle pour assurer une bonne sûreté de fonctionnement de l'ensemble et permettre une implémentation aisée.

Ces remarques nous ont conduit à utiliser une classe restrictive des réseaux de Pétri : les réseaux de Pétri structurés, conciliant le caractère graphique de description des réseaux de Pétri et la rigueur de l'analyse et de la programmation structurée.

BIBLIOGRAPHIE DU CHAPITRE II

|1| AGERWALA T., FLYNN M.
"Comments on capabilities, limitations and correctness of Petri nets"
First Annual Symposium on Computer Architecture, Florida, 1973.

|2| BERTHELOT B.
"Vérification des réseaux de Pétri"
Thèse de 3ème Cycle, Paris, 1978.

|3| BERTHOMIEU B.
"Analyse structurelle des réseaux de Pétri. Méthodes et outils"
Thèse de Doct. Ing., Toulouse, 1979.

|4| BOHM C., JACOPINI G.
"Flow diagrams Turins machines and languages with only two formation rules"
Comm. of the ACM 9.5, 1966.

|5| CHEZAVIEL-PRADIN B.
"Un outil graphique intéractif pour la vérification des systèmes à évolution parallèle décrits par réseaux de Pétri"
Thèse de Doct. Ing., Toulouse, 1979.

|6| CORBEEL D.
"Schéma de cablage et schéma de contrôle. Application à la simulation et à la gestion de processus industriels"
Thèse de Doct. de Spécialité, Lille, 1979.

|7| CORBEEL D., VERCAUTER C., GENTINA J.C.
"Méthodologie de description des systèmes de processus et de gestion d'erreur"
Mini and Micro, Budapest, 1980.

|8| CORBEEL D., VERCAUTER C., GENTINA J.C.
"Formal description of processes'systems and exceprion handling"
IASTED Inter. Symp. Modelling, Identification and Control, Davos, 1981.

|9| CORBEEL D., VERCAUTER C., GENTINA J.C.
"Méthodologie de description des systèmes de processus et de gestion d'erreur"
Convention Informatique Latine, Barcelone, 1981.

|10| CORBEEL D., VERCAUTER C., GENTINA J.C.
"Specification and conception of real-time control systems"
IASTED Inter. Symp. Measurement and Control, Le Caire, 1981.

|11| VERCAUTER C., CORBEEL D.
"Analyse des systèmes décrits par des réseaux de Pétri structurés"
MIMI'82, Davos, Mars 1982.

|12| CROCUS
"Systèmes d'exploitation des ordinateurs"
Dunod, 1975.

|13| DAHL O.J., DIJKSTRA E.W., HOARE C.A.R.
"Structured programming"
Academic Press, London, 1972.

|14| DAVID R., MOALLA M., SAUCIER G., SILVA M.
"Conception d'automatismes logiques répartis : outils et mise en œuvre"
Rapport D. G. R. S. T., n° 77.7.0646, 1979.

|15| DIJKSTRA E.W.
"Cooperating sequential processes"
in "Programming languages", F. GENUYS, Ed. Academic Press, 1967.

|16| GENRICH H.J., LAUTENBACH K.
"System modelling with high-level Petri nets"
Theoritical Computer science 13 (1981), pp. 109-136, North-Holland Publ. Co.

|17| HACK M.
"Petri net languages"
MAC, Memo, 124, MIT, 1975.

|18| HACK M.
"Analysis of production schemata by Petri nets"
MS Thesis, MIT, 1972.

|19| HOLT A., COMMONER F.
"Events and conditions"
Record of the project Mac Conference on Concurrent Systems and Parallel Computation, 1970.

|20| KARP R.M., MILLER R.E.
"Parallel Program Schemata"
Journal of Computer and System Sciences, Vol. 3, 1969.

|21| KELLER R.M.
"Formal verification and parallel Programs"
Communication of the ACM, Vol. 19, n0 7, 1976.

|22| LAHMAR N.
"Sur un système de gestion de processus multitâches : Application à la commande et à la simulation"
Thèse de 3ème Cycle, Lille, 1982.

|23| MEKLY L.J., YAU S.
"Software design representation using abstract process networks"
IEEE Trans on S.E., Vol. SE-6, n° 5, 1980.

|24| MEMMI G.
"Fuites et semi-flots dans les réseaux de Pétri"
Thèse de Doct. Ing., Paris, 1978.

|25| MOALLA M., SIFAKIS J., SILVA M.
"A la recherche d'une méthodologie sûre des automatismes logiques basés sur les réseaux de Pétri"
Monographie AFCET informatique "Sûreté de fonctionnement des systèmes informatiques", Ed. Hommes et techniques, 1980.

|26| NET THEORY AND APPLICATIONS
Proc. of the Advanced Course on General Net theory of Processes and Systems, Hamburg, 1979, Springer-Verlag 1980.

|27| PETERSON J.L.
"Petri nets"
Computing Surveys, Vol. 9, n° 3, 1977.

|28| RAMAMOORTHY C.V., HO G.S.
"Performance evaluation of asynchronous concurrent systems using Petri"
IEEE Trans. on S.E., Vol. SE-6, n°5, 1980.

|29| RAMCHANDANI C.
"Analysis of asynchronous concurrent systems by Petri Nets"
PhD Thesis, MIT 1973.

|30| RAPPORT FINAL DE LA COMMISSION AFCET
"Normalisation de la représentation du cahier des charges d'un automatisme logique"
1977, (publié dans Automatique et Informatique Industrielle n° 61-62).

|31| RENALIER J.
"Analyse et simulation en langage APL de systèmes de commandes décrits par des réseaux de Pétri"
Thèse de Doct. de Spécialité, Toulouse, 1976.

|32| ROLLIN P.
"Exploitation d'un modèle d'évaluation de Nutt au cours de la vie d'un produit"
Journée d'étude AFCET "Validation et spécification", 1981.

|33| SHAPIRO R.M., SAINT H.
"A new approach to optimization of sequencing decisions"
Annual review of automatic programming 6,5, 1970.

|34| VALETTE R.
"Sur la description, l'analyse et la validation des systèmes de commandes parallèles"
Thèse d'Etat, Toulouse, 1976.

|35| VALETTE R., LATAPIE M., COURVOISIER M.
"Possibilités et limites des réseaux de Pétri dans le cadre des systèmes à haute sûreté"
Journée d'étude AFCET "Validation et spécification" 1981.

|36| VOSS K.
"Using Predicat/Transition-Nets to model and analyse distributed database systems"
IEEE Trans. on S.E., Vol. SE-6, n° 6, 1980.

CHAPITRE III

VALIDATION DES SYSTEMES DECRITS PAR DES RESEAUX DE PETRI

CHAPITRE III

page

VALIDATION DES SYSTEMES DECRITS PAR DES RESEAUX DE PETRI

INTRODUCTION

La validation d'un modèle constitue une étape déterminante dans le processus de raffinement des spécifications. Elle représente une condition essentielle permettant à ce processus de se réitérer afin de produire un nouveau modèle qui devra à son tour être validé.

D'une manière générale, la méthode proposée procède en deux étapes :

- la première consiste à vérifier les propriétés du modèle indépendamment de son interprétation c'est-à-dire de sa sémantique.

- la seconde permet de vérifier que les spécificités du système sont respectées.

Afin de garantir une bonne sûreté de fonctionnement du système construit à partir du modèle final, il est donc nécessaire que la description d'un système s'appuie sur un outil formel permettant ces deux types de vérification.

Pour un système décrit par un réseau de Pétri, il faut donc vérifier les propriétés internes du réseau, c'est-à-dire les caractères :

- borné, afin d'exprimer que le système possède un nombre d'état fini

- réinitialisable, caractéristique du fonctionnement cyclique d'un système

- vivant, ce qui permet d'assurer qu'il n'existe pas de situation de blocage

Deux classes de méthodes ont été développées :

- les méthodes classiques qui nécessitent l'énumération de tous les marquages possibles obtenus à partir d'un marquage initial donné que nosu présenterons dans le premier paragraphe

- les méthodes "structurelles", qui feront l'objet de la troisième partie de ce chapitre, pour lesquelles le marquage n'apparaît qu'en tant que paramètre et qui fournissent des invariants linéaires permettant le test de certaines propriétés spécifiques.

Pour ces deux types de méthodes, il est souvant nécessaire de réduire la taille du graphe en utilisant des règles de transformation qui conservent les propriétés des graphes analysés.

Dans ce sens, nous présenterons dans le cadre des réseaux structurés qui concernent cette étude, une synthèse et une adaptation des méthodes de réduction.

Pour chaque méthode, nous présenterons les simplifications qu'apporte l'utilisation des réseaux de Pétri structurés.

III.1 - METHODES CLASSIQUES

III.1.1 - Introduction

Ces méthodes nécessitent l'énumération de tous les marquages conséquents à un marquage initial donné. D'abord conçues pour résoudre le problème de l'accessibilité d'un marquage, elles permettent de décider du caractère borné d'un réseau de Pétri. Elles ont été développées initialement pour un formalisme équivalent (les systèmes d'addition de vecteur), elles ont ensuite été adaptées au réseau de Pétri |11| |13|. L'énumération de tous les marquages permet de construire l'arbre des marquages conséquents à M_o (ou le graphe).

III.1.2 - L'arbre des marquages ou arborescence recouvrante

Chaque sommet du graphe des marquages est étiqueté pour un marquage, alors que chacun de ses arcs est étiqueté pour la transition dont le tir a permis de faire évoluer la fonction de marquage. A la racine de l'arbre correspond le marquage initial du réseau.

Cependant, pour obtenir une arborescence fine lorsque le réseau est non borné, un symbole spécial w est utilisé pour étiqueter le marquage d'une place non bornée. Le symbole w représente un nombre infiniment grand pour lequel les opérations d'addition, de soustraction et de comparaison donnent les résultats suivants :

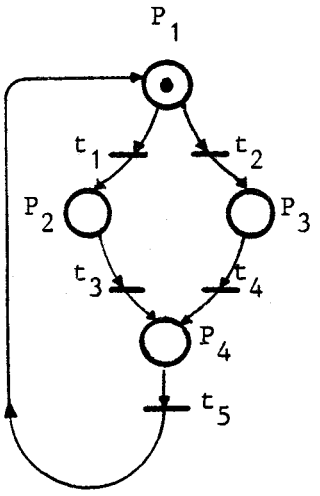
$$\forall a \in \mathbb{N} \qquad a < w$$
$$a + w = w$$
$$- a + w = w$$

On note w dans l'étiquette M, le marquage de toute place p tel que, parmi les ancêtres du nœud d'étiquette M figure un sommet d'étiquette M' telle que :

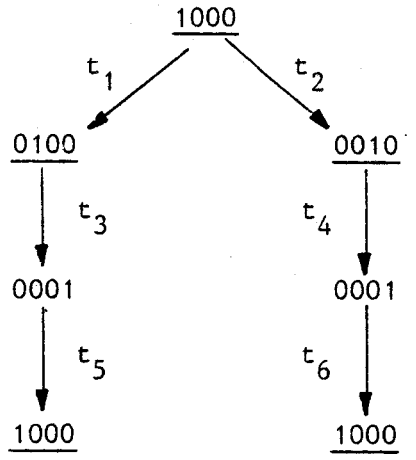
$$M' \leqslant M$$
$$M'(p) < M(p)$$

L'examen d'une branche est arrêté dès que l'on retrouve un sommet dont l'étiquette est identique à celle de l'un de ses ancêtres.

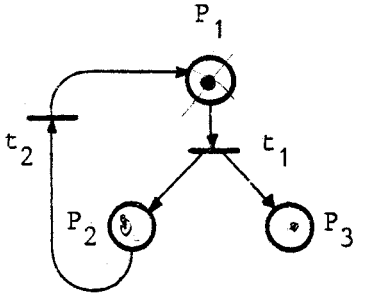
Exemples :

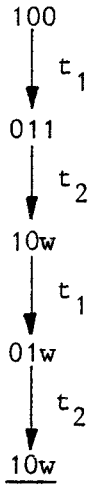


(a) Réseau de Pétri $\mathcal{G}$



Arbre des marquages du RdP $\mathcal{G}$





(b) Réseau de Pétri non borné

Figure III.1

Proposition : Un réseau de Pétri est borné si le symbole w n'apparaît pas dans l'étiquette d'un sommet de l'arbre des marquages conséquents.

Preuve : Evidente, d'après la définition même du symbole w.

Proposition : Si le réseau de Pétri est borné, tout marquage M est accessible depuis le marquage initial M_o, s'il existe un sommet de l'arbre labellé par M.

Preuve : Evidente (l'arborescence recouvrante contient tous les marquages accessibles depuis M_o).

Cette procédure qui, initialement, visait à démontrer que le caractère borné était décidable, peut cependant être simplifié en tenant compte des remarques suivantes :

1 - Le fait que le symbole w apparaisse dans l'étiquette d'un sommet permet de conclure que le réseau est non borné. La poursuite de la procédure d'énumération ne présente alors plus d'intérêt.

2 - Compte tenu de la commutativité des tirs de transitions simultanément déclenchables, il est également inutile de poursuivre l'examen d'une branche, si on retrouve parmi tous les sommets déjà obtenus, c'est-à-dire ceux de cette branche et ceux des autres branches, un sommet étiqueté de la même manière.

Définition : Deux transitions sont dites simultanément déclenchables si le tir de l'une n'affecte pas le tir de l'autre.

Exemple :

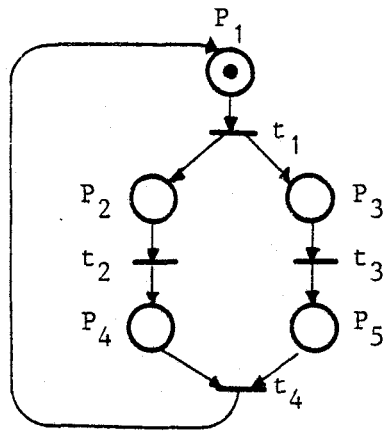


Figure III.2

Sur cet exemple, les transitions t_2 et t_3 sont simultanément déclenchables. La séquence de tir $t_1\ t_2\ t_3$ exécutée depuis M_o conduit au même marquage que la séquence $t_1\ t_3\ t_2$.

Exemple :

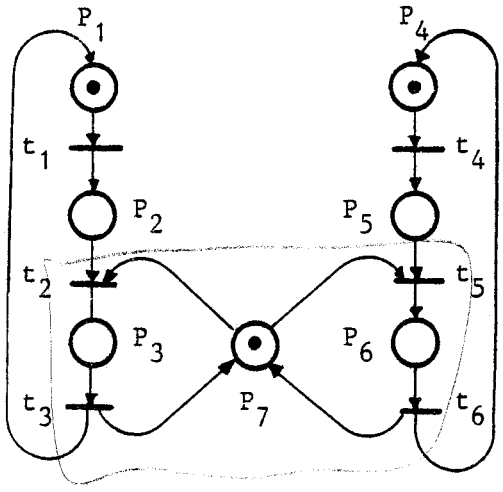
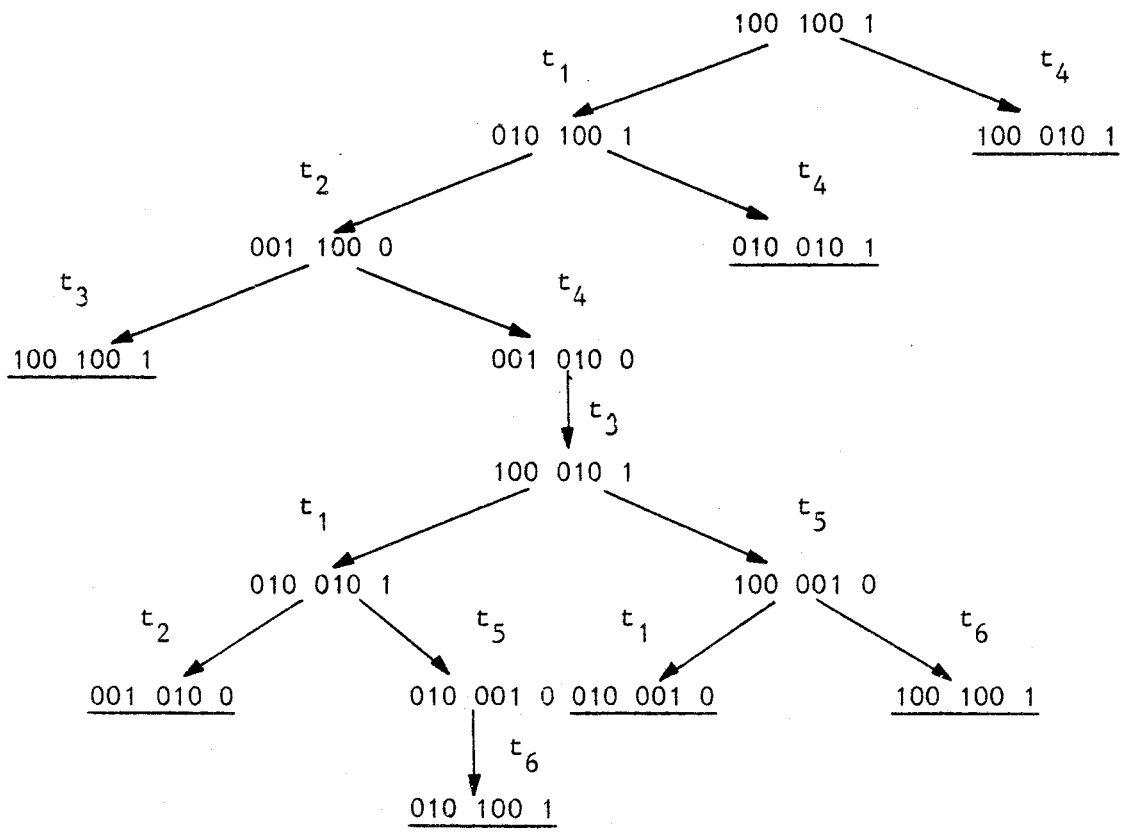


Figure III.3 $\quad M_o^T = [100\ 100\ 1]$



Sur l'exemple ci-dessus, l'utilisation de la remarque (2) nous conduit à construire un arbre ne comportant que 16 sommets, l'arborescence recouvrante établie selon la procédure de Keller disposerait de 57 sommets.

La réduction du nombre de sommets de l'arbre est d'autant plus grande que le degré de parallélisme du réseau est important. Cette remarque est importante dans le sens où celui-ci apparaît explicitement sur les réseaux de Pétri structurés. En effet, le degré de parallélisme dans un réseau de Pétri structuré correspond au nombre de processus de ce système.

3 - Il est cependant possible d'obtenir une représentation plus concise en regroupant les sommets qui portent la même étiquette. On obtient alors le graphe des marquages |27| qui possède autant de sommets que l'arbre a de marquages différents.

Ainsi, dans l'exemple de la figure III.3, le graphe de marquage du réseau possède 8 sommets (cette représentation n'est bien évidemment possible que si le réseau est borné).

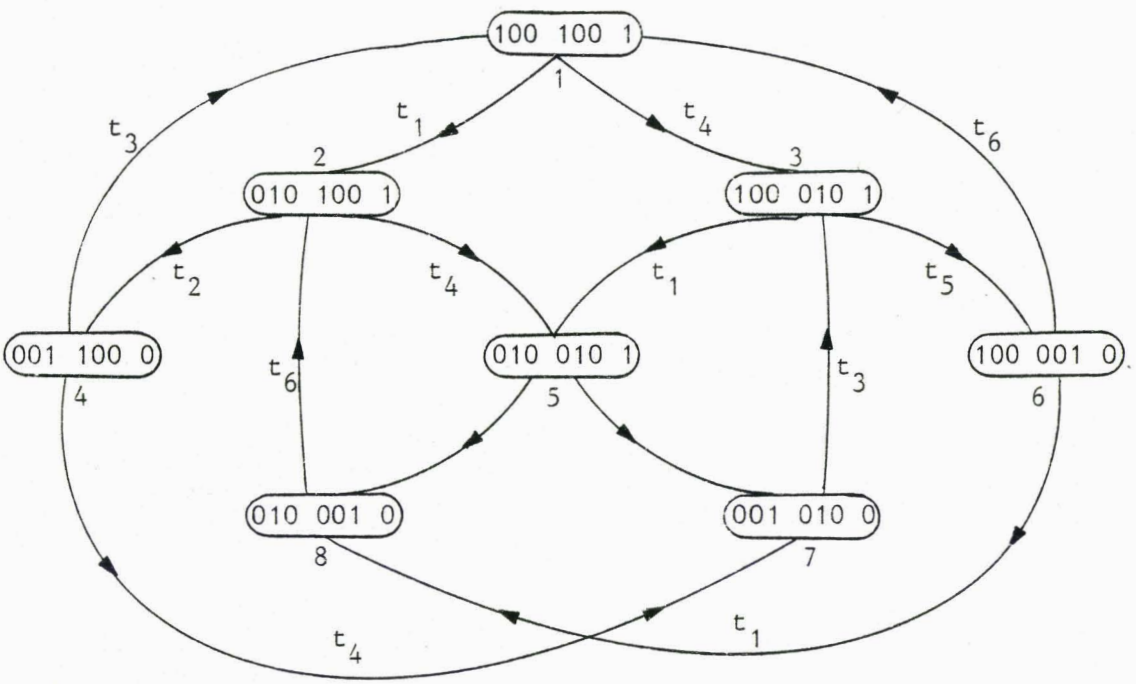


Figure III.4

Graphe de marquage du
réseau de Pétri de la figure III.3

La construction du graphe de marquage d'un réseau de Pétri structuré doit tenir compte du fait qu'une place de liaison de synchronisation est absorbante.

Le programme dont l'algorithme de principe est présenté Figure III.5 fournit pour l'exemple de la figure III.3, le tableau de résultats suivants :

Numéro du sommet	Marquage	Nombre de successeurs	Numéro de chaque successeur	Indice des transitions déclenchées
1	100 100 1	2	2, 3, ...	1, 4
2	010 100 1	2	4, 5, ...	2, 4
3	100 010 1	2	5, 6, ...	1, 5
4	001 100 0	2	1, 7, ...	3, 4
5	010 010 1	2	7, 8, ...	2, 5
6	100 001 0	2	8, ...	1, 6
7	001 010 0	1	3, ...	3
8	010 001 0	1	2, ...	6

Remarque : Sur ce tableau, les successeurs et les transitions sont ordonnés : l'arc qui relie le sommet n°i à son $j^{ème}$ successeur est étiqueté par la $j^{ème}$ transition. Ainsi, le sommet n°3 étiqueté par 100 010 1 a deux sommets successeurs. Il est relié au sommet n°5 par un arc labellé par (t)1, et au sommet n°6 par un arc labellé par (t)5.

```
Procédure "Construire le GRAPHE DES MARQUAGES"

(* INITIALISATION *)
Pseudo-vivant := VRAI
Borné := VRAI
Borné ou Vivant := VRAI
J := 1 (* N° d'ordre du marquage *)
N := 1 (* Nombre de marquages obtenus *)

M := MO (* Le premier marquage considéré est le marquage initial MO *)

(* DEBUT DE L'ALGORITHME *)

    Tant que J ≤ N   Faire
        NTS(J) := 0 (* NTS(J) = Nombre de transition sensibilisées par
                                 le marquage N° J *)
        Rechercher toutes les transitions sensibilisées
        Stocker leurs indices
        Si NTS(J) = 0   Alors Quasi-Vivant := FAUX

            Sinon Pour chaque transition sensibilisée   Faire
                Calculer le nouveau marquage
                Pour chaque place de synchro. Faire
                    Si le marquage de cette place > 1   Alors
                        on lui affecte la valeur 1
                    FSi
                FPour
                Pour chaque marquage déjà obtenu   Faire
                    Si le nouveau lui est supérieur   Alors
                        Borné ou Vivant := FAUX
                        Exit Procédure
                                                    Sinon
                        Stocker le nouveau marquage
                        N := N + 1
                    FSi
                FPour
            FPour
        J := J + 1      (* Marquage suivant *)
    FTant que           (* Le réseau est borné *)
FProcédure
```

Figure III.5

Ce programme permet de décider si un réseau de Pétri structuré ou un réseau de Pétri quelconque (ordinaire, généralisé, réseau libre-choix, ...) est borné. Il permet également de détecter les situations de blocage et par conséquent, décider également du caractère pseudo-vivant du réseau pour le marquage initial M_o. Lorsque la conjecture bornée n'est pas vérifiée, alors le réseau de Pétri est soit non borné, soit non vivant.

La définition de cette propriété est donnée dans |6|.

III.1.3 - Analyse des caractères Vivant et Réinitialisable

L'analyse de cette propriété n'est envisageable que si le réseau de Pétri est borné (le réseau de Pétri est, dans ce mémoire, utilisé comme modèle d'un système réel, le fait qu'il soit non borné révèle un défaut de structure du modèle qu'il faut donc réécrire).

Elle nécessite de vérifier que pour chaque transition t et pour chaque marquage M, s'il existe une séquence de déclenchement depuis M_1 qui contient t. Elle peut être effectuée sur le graphe des marquages à partir du théorème suivant :

Soit GR, le graphe des composantes fortement connexes du graphe de marquage d'un réseau $R = (G, M_o)$.

Définition : On appele composante fortement connexe pendante, tout sommet sans successeur du graphe des composantes fortement connexes.

Théorème : Un réseau de Pétri borné pour un marquage initial M_o est vivant si et seulement si chaque transition du réseau est au moins une fois étiquette d'un arc de chacune des composantes fortement connexes pendantes du graphe des marquages.

Démonstration : Soit M', un marquage d'une composante fortement connexe pendante (cfcp) et M n'appartenant à aucune cfcp. Il existe alors un chemin (c'est-à-dire une séquence de déclenchement) entre M et M'. Si, à partir de M', toutes les transitions du réseau sont quasi-vivantes (elles étiquettent chaque arc du cfcp) alors, elles sont également quasi-vivantes pour tout marquage M n'appartenant pas à une cfcp.

L'analyse du caractère vivant nécessite donc la recherche des composantes fortement connexes d'un graphe.

Les méthodes permettant cette recherche sont assez nombreuses |9|. Elles utilisent :

- des algorithmes d'exploration en profondeur "d'abord" (Peep First Search)

- des algorithmes spécifiques aux graphes décrits par piles

- des algorithmes qui utilisent la matrice associée à l'application Γ, (matrice caractéristique du graphe de marquage).

Elles conduisent à déterminer pour chaque sommet, l'ensemble de ses descendants et de ses ancêtres.

Soit GM le graphe de marquage défini par :

$$GM = (X , \Gamma)$$

où X est l'ensemble des sommets du graphe
Γ l'application multivoque qui associe à un sommet, l'ensemble de ses successeurs
(Γ^{-1} représente l'application qui associe à un sommet, l'ensemble de ses antécédents).

Définition : On appelle fermeture transitive de l'application multivoque Γ, l'application multivoque $\hat{\Gamma}$ définie par :

$$\hat{\Gamma}(i) = \Gamma(i) \cup \Gamma(i)^2 \cup \ldots \cup \Gamma(i)^{n-1} \qquad n = |X|$$

avec $\Gamma(i)^k$ représentant l'ensemble des sommets que l'on peut atteindre par des chemins de longueur k.

$\hat{\Gamma}(i)$ est l'ensemble des descendants du sommet i, de même, $\hat{\Gamma}^{-1}$ est l'ensemble des ancêtres du sommet i.

La recherche des composantes fortement connexes utilise la remarque suivante :

la composante fortement connexe $\mathcal{C}$ contenant un sommet i donné est égale à :

$$\boxed{\mathcal{C}(i) = \hat{\Gamma}(i) \cap \hat{\Gamma}(i)^{-1}}$$

c'est-à-dire qu'elle est l'intersection de l'ensemble de ses descendants et de ses ancêtres.

De plus, cette composante fortement connexe est pendante si, pour chacun des marquages qui étiquette un nœud de cette composante, l'ensemble des successeurs est inclus dans l'ensemble des antécédents.

La figure III.6 donne l'algorithme de principe du programme d'analyse du caractère vivant et réinitialisable d'un réseau de Pétri borné quelconque. Il succède dans l'analyse des propriétés d'un réseau au programme que nous avons présenté peu avant (celui-ci lui fournit l'arbre des marquages).

Le théorème suivant du à |27| permet la vérification de la propriété réinitialisable.

Théorème : Un réseau de Pétri est réinitialisable si et seulement si son graphe de marquage est fortement connexe.

La démonstration est évidente. En effet, si le réseau est réinitialisable, il existe depuis tout marquage appartenant à $\mathcal{E}(M_o)$, une séquence de déclenchement qui conduit à M_o. Or, sur le graphe de marquage GM, chaque chemin représente une séquence de tir et chaque sommet un marquage de $\mathcal{E}(M_o)$, si ce graphe est fortement connexe, alors depuis tout sommet, il existe un chemin qui conduit au sommet dont l'étiquette est celle du marquage initial.

```
Procédure "ANALYSE DES CARACTERES VIVANT ET PROPRE"
(* INITIALISATION *)
REINIT := VRAI
VIVANT := VRAI
(* DEBUT DE L'ALGORITHME *)
     Calculer la fermeture transitive de l'application GAMMA
     (* GAMMA est l'application qui associe à un sommet, l'ensemble de
     ses successeurs immédiats. La fermeture transitive nous donne pour
     chaque sommet, ses successeurs et ses prédécesseurs *)
     Déterminer les NCFC composantes fortement connexes du graphe
     (* Une CFC contenant un sommet est obtenu par l'intersection de ses
     successeurs et de ses prédécesseurs *)

     Si NCFC <> 1   Alors

          REINIT := FAUX
     FSi

     Rechercher les CFC pendantes
     (* Une CFC est pendante si, pour chacun de ses sommets, l'ensemble
     des prédécesseurs est inclus dans celui de ses successeurs *)

     Pour chaque CFC pendante    Faire

          Pour chaque marquage de cette CFC

               Si il existe une transition non quasi-vivante pour ce
                         marquage   Alors   VIVANT := FAUX
               FSi

          FPour

     FPour

FProcédure
```

Figure III.6

L'analyse du réseau suivant est borné, vivant mais non réinitialisable. Son graphe de marquage possède 3 composantes fortement connexes dont deux sont pendantes.

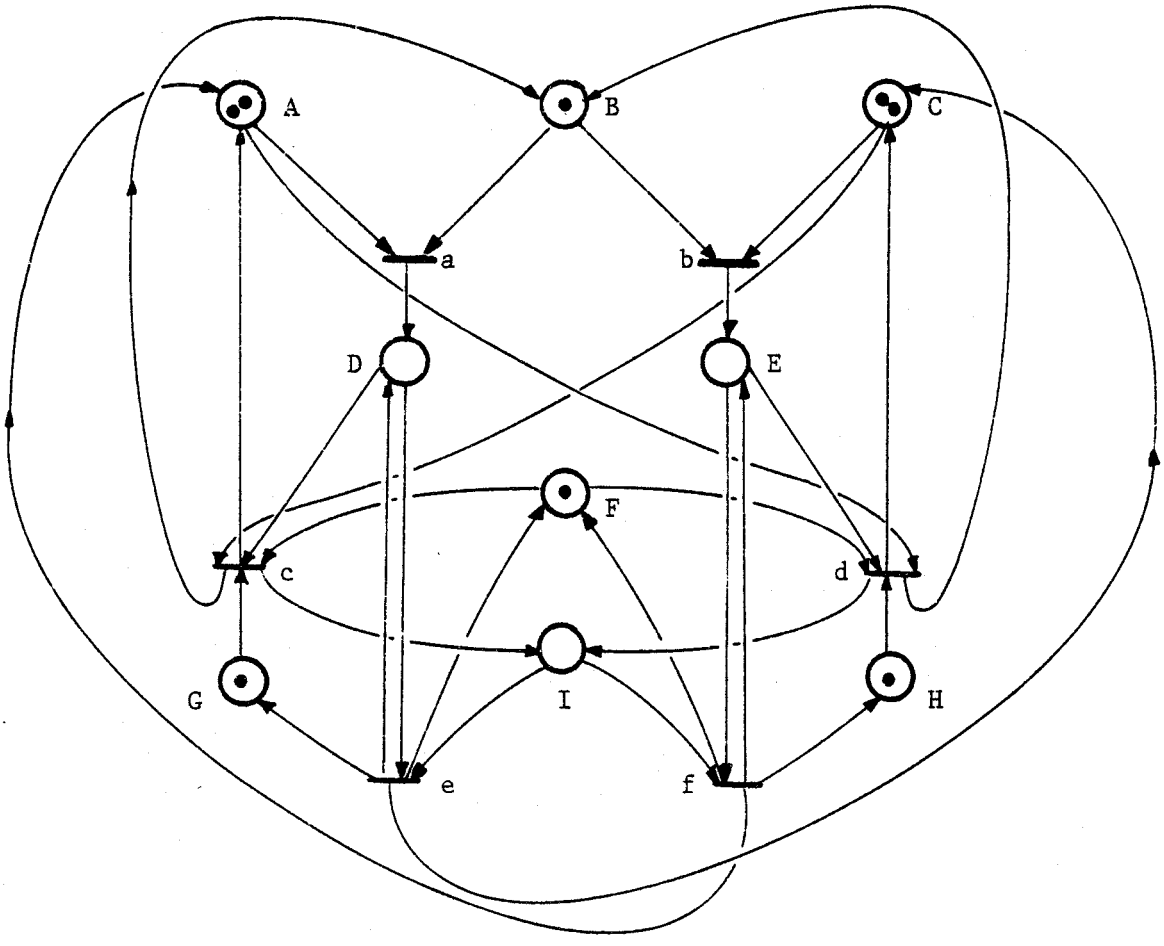


Figure III.7

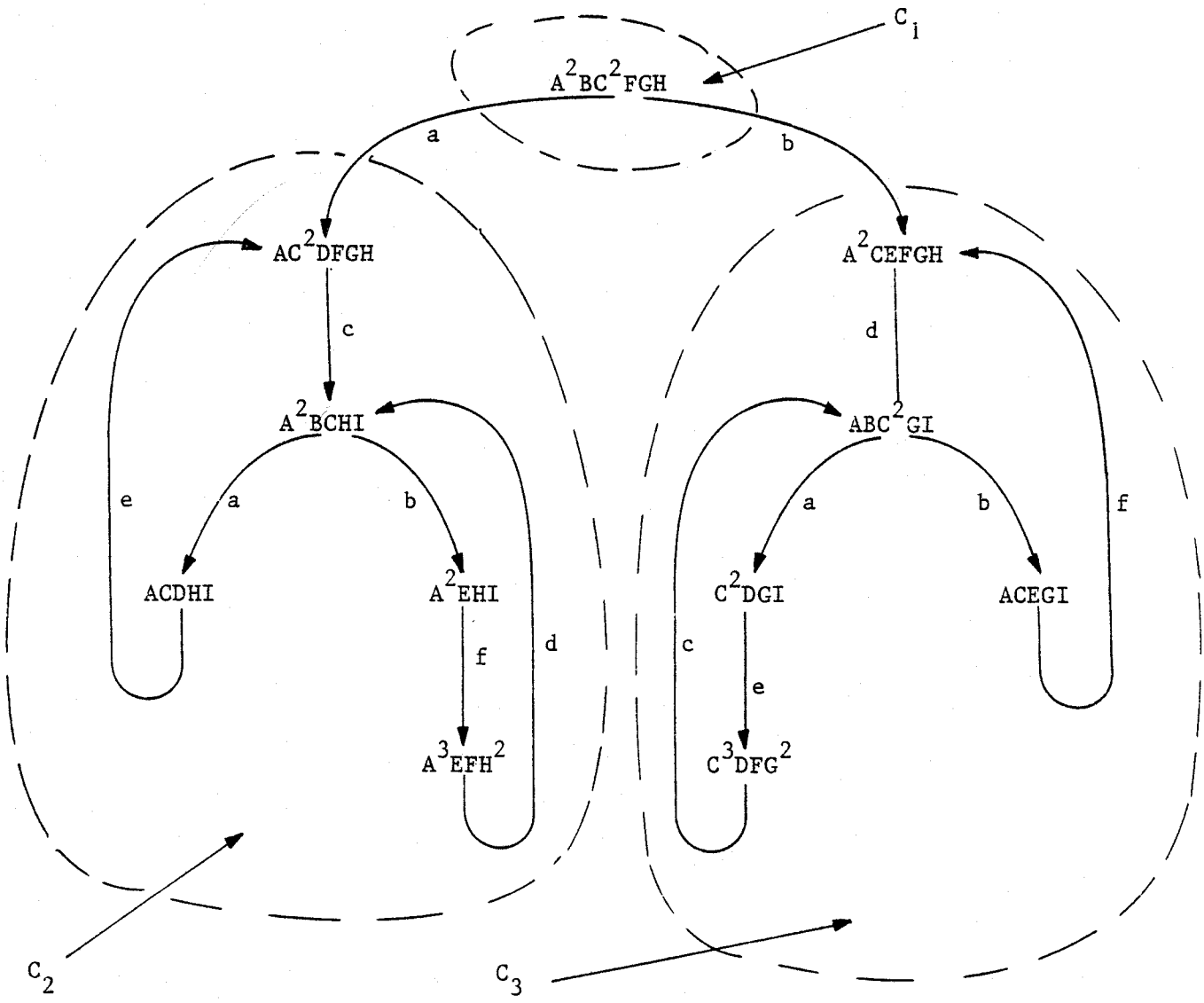


Figure III.8

Réseau de Pétri et graphe de marquage

Les composantes fortement connexes C_2 et C_3 sont pendantes. Cependant, chaque transition du réseau étiquette au moins une fois un arc de chacune de ces composantes. Le réseau de la figure III.7 est donc vivant. Son graphe de marquage n'étant pas fortement connexe (il existe deux CFC pendantes), il n'est donc pas réinitialisable.

III.1.4 - Conclusion

Les deux procédures présentées dans ce paragraphe, permettent la vérification des propriétés internes du réseau de Pétri. Elles s'enchaînent de la manière suivante :

Programme "ANALYSE DES PROPRIETES INTERNES"

Construire le graphe des marquages

Si BORNE et QUASI-VIVANT Alors

Analyse des caractères Vivant et Propre

Si VIVANT et REINIT Alors

Le RESEAU est CONFORME

Sinon

Le RESEAU est INCORRECT

FSi

Sinon

Le RESEAU est INCORRECT

FSi

FProgramme

Figure III.9

L'utilisation de ces méthodes pour des systèmes industriels nous amènent cependant, à formuler quelques réserves pour l'analyse de modèles complexes.

En effet, la dimension du graphe conduit très souvent à une taille mémoire et un temps de calcul trop important. Ceci est du à la nature fortement combinatoire des algorithmes d'énumération (complexité exponentielle en temps et en espace mémoire) |18|.

Aussi est-il souvent nécessaire de réduire au préalable la taille des réseaux de Pétri.

III.2 - REDUCTION DES RESEAUX DE PETRI

III.2.1 - Introduction

Au cours de ce chapitre, nous aborderons la présentation de méthodes de réduction générales, déduites d'un formalisme proche de celui des réseaux de Pétri, les graphes UCLA |3| |4|. Nous nous intéresserons ensuite au cas particulier des réseaux de Pétri structurés pour indiquer les simplifications importantes permises par rapport au cas général. Dans ce cas en effet, la mise en œuvre devient plus simple et les algorithmes de réduction seront en conséquence plus efficaces et plus rapides.

III.2.2 - Règles de réduction générales |3| |4|

III.2.2.1 - Règle 1 : Substitution d'une place :

* Place substituable :

Soit un réseau de pétri $R = (G , M_o)$ avec $G = (P, T, \Gamma)$ ou également $G = (P, T, S, E)$.

Définition : Une place $p \in P$ est substituable si et seulement si il existe un entier positif non nul et deux sous-ensembles non vides de transitions H et F tels que :

(i) H : ensemble des transitions d'entrée de p

c'est-à-dire $H : \{t \mid S(t,p) \neq 0\}$

(ii) F : ensemble des transitions de sortie de p

c'est-à-dire $F : \{t \mid E(t,p) \neq 0\}$

(iii) $F \cap H = \emptyset$

(iv) $\forall t \in F$ alors $E(t,p) = m$ avec $m \in \mathbb{N}^*$ et
$\forall p' \in P - \{p\}$ $E(t,p') = 0$

Toute transition t de F n'a que la place p comme entrée. Les arcs qui relient le place p à chaque transition de F ont le même poids m.

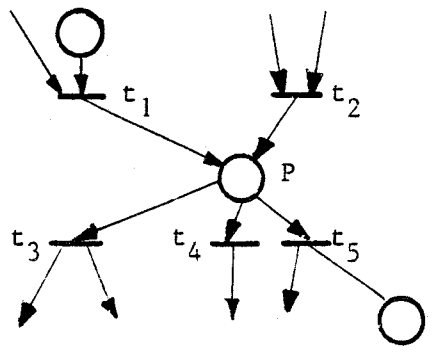
(v) $\exists k \in \mathbb{N}^*$ tel que $\forall t \in H$ $S(t,p) = k.m$

Le nombre de marqueurs mis dans la place p par chacune des transitions de H est égal à un multiple de m.

(vi) $\exists t \in H$ $\exists p' \in P - \{p\}$ tel que $S(t,p') \neq 0$

Une au moins des transitions de sortie de H possède une sortie vers une autre place du réseau.

Le schéma type d'une place substituable est le suivant :



$M = \{t_1, t_2\}$

$F = \{t_3, t_4, t_5\}$

Figure III.10

Remarque : Si l'on ne tient pas compte de la règle (vi), il est possible de transformer un réseau non borné en réseau borné.

Exemple :

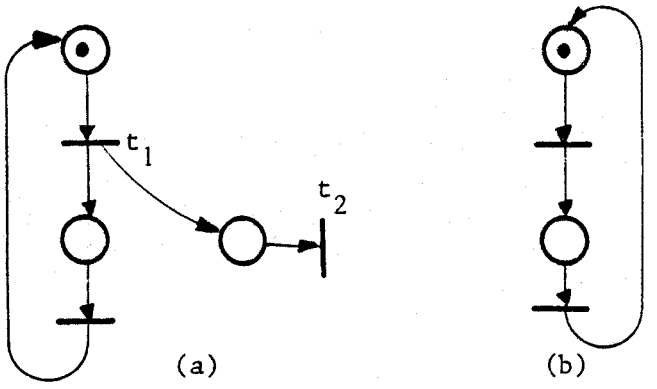


Figure III.11

Sur l'exemple de la figure III.11, le réseau (a) est non borné, celui de (b) l'est.

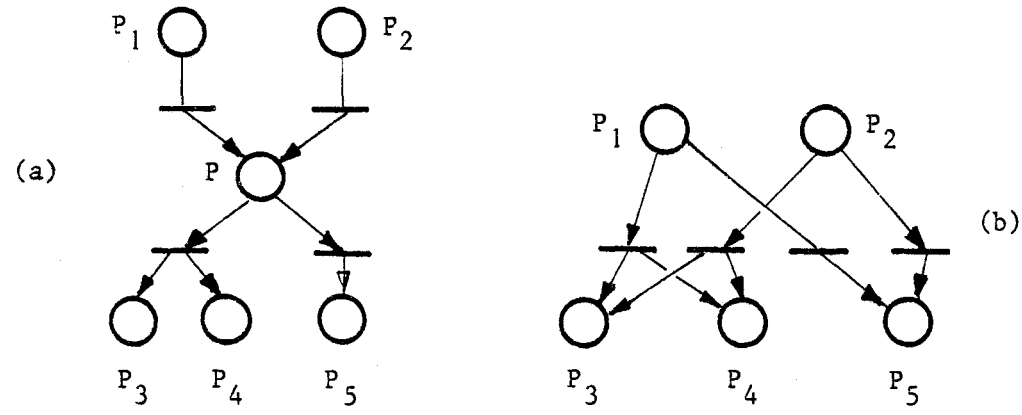
* Substitution d'une place substituable :

Le principe consiste à remplacer la place p et les ensembles H et F, par un ensemble de transitions S de telle sorte que le déclenchement des transitions de S conduise aux mêmes marquages que ceux obtenus par les tirs des différentes combinaisons h, f (au marquage de la place p près).

Si cette place était marquée pour le marquage initial, le marquage initial est transformé.

Nous appellerons MI, l'ensemble des marquages initiaux qui peuvent ainsi être obtenus.

Exemple :



Substitution d'une place substituable

Figure III.12

On constate ainsi que le tir de t'_1 conduit au même marquage que ceux de t_1 et t_3 (de même t'_2 et t_2t_3, t'_3 et t_1t_4, t'_4 et t_2t_3).

III.2.2.2 - Règle 2 : Simplification d'une place implicite :

* Place implicite :

Soit un réseau $R = (G , M_o)$ avec $G = (P, T, E, S)$

Une place p est implicite s'il existe :

- un sous ensmeble P' (éventuellement vide) de $P - \{p\}$
- une valuation $V : P' \cup \{P\} \rightarrow \mathbb{N}^*$

tels que : $\exists b_{M_o} \in \mathbb{N}$

(i) $\forall M_o \in MI \qquad V(p).M_o(p) - \sum_{p' \in P'} V(p').M_o(p') = b_{M_o}$

Quelquesoit le marquage initial, la différence entre le marquage pondéré de p et la somme des marquages pondérés de P' est positive ou nulle. Autrement dit, le marquage initial d'une place implicite ne constitue pas un obstacle pour le tir d'une transition.

$$\text{(ii)}\quad \forall t \in T \qquad V(p)\ E(t,p) - \sum_{p' \in P'} V(p')\ E(t,p') = b'_t$$

où $b'_t \leqslant \min\ (b_{M_o})$

Pour chaque transition t de T, la différence entre le marquage pondéré de p nécessaire au tir de t, et la somme des marquages pondérés des places de P' nécessaires au tir de t n'est pas supérieure à celle de chacun des marquages initiaux.

$$\text{(iii)}\quad \forall t \in T \qquad \exists C_t \in \mathbb{N}$$

$$V(p)\ (S(t,p) - E(t,p)) - \sum_{p' \in P'} V(p')\ (S(t,p') - E(t,p') = C_T$$

Pour le déclenchement de chaque transition t, la différence entre l'accroissement du marquage pondéré de p ($M(p) = M_o(p) + S(t,p) - E(t,p)$) et celui de la somme des marquages pondérés des places de P' n'est pas négative.

La recherche des places implicites n'est pas comme pour les places substituables, immédiate. Elle conduit généralement à la résolution d'un problème de programmation linéaire en nombres entiers |6|.

Cependant, deux cas particuliers de place implicite sont intéressants, leur détection étant relativement aisée. Il s'agit :

1) Des places identité (cas où $P' = \emptyset$) pour lesquelles les définitions que nous venons de citer se réduisent à :

- $\forall t \in T,\ \exists M_o \in MI \qquad M_o(p) \geqslant E(p,t)$
- $\forall t \in T,\ \exists C_T \in \mathbb{N} \qquad S(p,t) - E(p,t) = C_t$

Exemple : PLACE IDENTITE

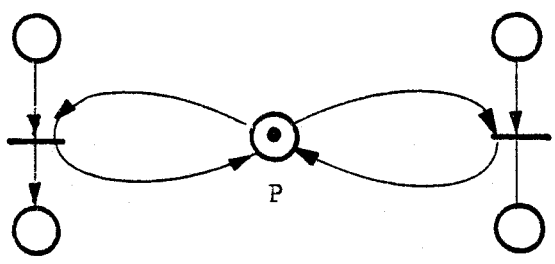


Figure III.13

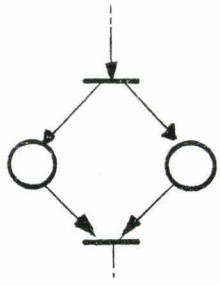
2) Les places identiques (cas où $P' = \{p'\}$). Alors, nous avons :

$$E(t,p) = E(t,p') \qquad \forall t \in T$$
$$S(t,p) = S(t,p')$$

Les places p et p' ont mêmes entrées et mêmes sorties.

La place p qui est telle que $\forall M_o \in MI \quad M_o(p) - M_o(p') = b_{M_o} \geqslant 0$ sera toujours simplifiée.

Exemple :



PLACE IDENTIQUE

Figure III.14

* Simplification d'une place implicite :

Ici, également, afin de conserver les propriétés du réseau initial, certaines précautions doivent être prises.

Exemple :

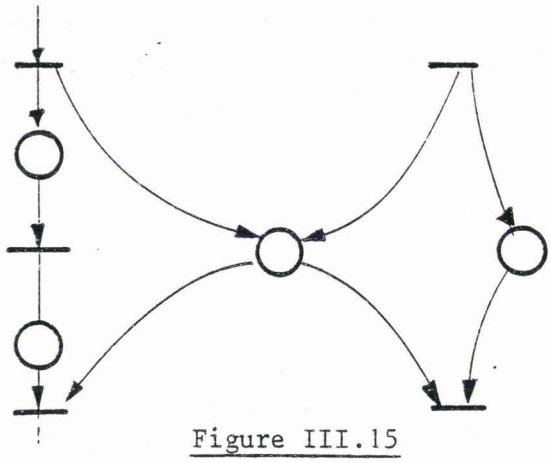


Figure III.15

La place p est implicite $P' = \{p_1 , p_2 , p_3\}$

La suppression de la place p conduirait à transformer un réseau non borné (si toutes les transitions t_2 à t_5 sont vivantes) en un réseau borné.

Le réseau réduit par cette règle R2 est tel qu'une place implicite n'est plus entrée d'aucune transition et n'est sortie que des transitions pour lesquelles $C_T \neq 0$ (le nombre d'arcs vaut C_T).

Si tous les C_T sont nuls, elle est simplement supprimée.

III.2.2.3 - Règle 3 : Suppression des transitions identité et transitions identiques :

* Transition identité :

Définition : Une transition t d'un réseau $R = (G , M_o)$ est appelée transition identité si et seulement si :

$$E(t) = S(t)$$

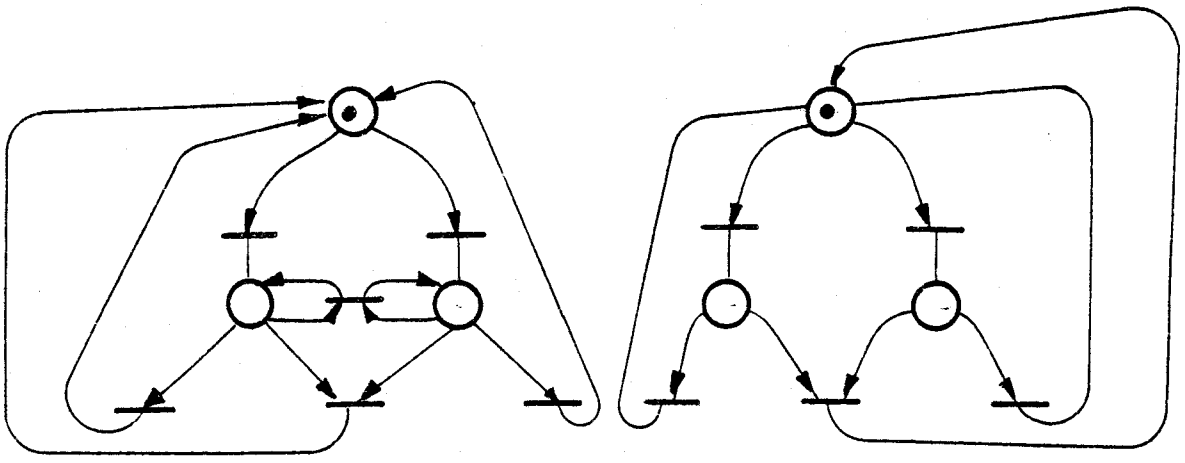
Si cette transition est vivante, alors il est évident que sa mise à feu ne modifie pas le marquage. Il est donc possible de la supprimer.

Par contre, si elle n'est pas vivante, elle ne pourra être supprimée que s'il existe une autre transition non vivante, ceci pour assurer que le caractère non vivant est conservé lors de la transformation du réseau.

Définition : Une transition identité t est supprimable si et seulement si il existe une transition t' différente de t qui vérifie :

$$S(t') \geqslant E(t)$$

Exemple :



SIMPLIFICATION D'UNE TRANSITION IDENTITE

Figure III.16

La transition identité t, bien que non vivante a pu être supprimée car dans le réseau, elle n'était pas la seule transition non vivante (t_5 l'est également).

* Transition identique :

Définition : Deux transitions t et t' d'un réseau R = (P, T, T, M_o) sont dites identiques si et seulement si elles ont mêmes entrées et mêmes sorties :

$$E(t) = E(t')$$
$$S(t) = S(t')$$

L'une quelconque de ces deux transitions peut alors être supprimée.

Exemple :

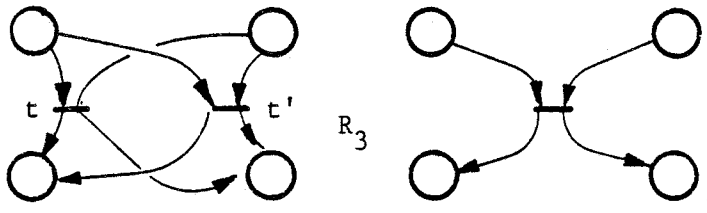


Figure III.17

III.2.2.4 - Analyse du réseau réduit :

Ces trois règles de réduction d'un réseau de Pétri conduisent :

- soit à un réseau totalement réduit

- soit à un réseau irréductible par ces trois règles, pour lequel une analyse classique ou utilisant les méthodes structurelles peut être effectuée.

Dans le premier cas, nous pouvons conclure que le réseau initial est vivant (quasi-vivant, pseudo-vivant) et borné. En effet, Berthelot |3| montre que ces règles conservent les propriétés du réseau.

Ces méthodes ont d'ailleurs un intérêt supplémentaire. Elles permettent en effet de décider, dans certains cas, si un réseau borné ou non est vivant. (Cette propriété de vivance a été démontrée décidable dans tous les cas mais les algorithmes et les programmes sont d'un usage très lourd). Pour vérifier cela, il suffit de ne pas tenir compte des contraintes d'utilisation des règles 1 et 2 que nous avons alors soulignées (exemples des figures III.11 et III.15).

L'ordre d'application de ces règles est quelconque mais compte tenu de leur difficulté de mise en œuvre informatique, l'ordre dans lequel elles ont été effectuées est souvent : R3 puis R1 puis R2.

Ces règles nous conduisent à formuler néanmoins, quelques remarques :

- La 1ère règle présentée a l'inconvénient de ne pas systématiquement réduire le nombre de sommets du réseau. A chaque fois qu'elle est utilisée, le nombre de places diminue d'une unité alors que celui des transitions peut augmenter.

- La 2ème règle a une mise en œuvre difficile. Elle nécessite l'utilisation d'un logiciel de programmation linéaire en nombre entier qui doit tenir compte de la taille du réseau initial (dans un cas réel |P| et |T| sont de l'ordre de la centaine, typiquement).

- Ces réductions peuvent être menées à bien plus facilement si l'on considère à priori les caractèristiques des réseaux de Pétri structurés. Ce point de vue nous a amenés à adapter dans ce sens, les méthodes que nous venons de décrire.

III.2.3 - Méthodes spécifiques aux réseaux de Pétri structurés

L'idée de base découle de la remarque suivante :

la réduction d'un réseau de Pétri structuré peut être interprétée comme une procédure d'abstraction de la structure du modèle (dans le sens où nous avons défini l'abstraction et le raffinement de description d'un modèle).

La première réduction que nous proposons consiste en fait, en un fusionnement de places.

III.2.3.1 - Fusionnement de places :

* Places fusionnables :

Définition : Deux places P et P' d'un réseau de Pétri structuré $R = (P, T, \Gamma, M_o)$ sont fusionnables si et seulement si :

- P est la seule place d'entrée de $\Gamma^{-1}(P')$
- P' est la seule place de sortie de $\Gamma(P)$
- $|\Gamma(p)| = 1 = |\Gamma^{-1}(p)|$
- $|\Gamma^{-1}(p')| = 1$

Remarque : Cette règle, certes limitative dans sa mise en œuvre, correspond à une simplification dans le cadre des réseaux de Pétri structurés qui conserve la structure du graphe de liaison.

En fait, quatre cas de figure ne correspondent pas à cette définition. Ils sont présentés ci-après :

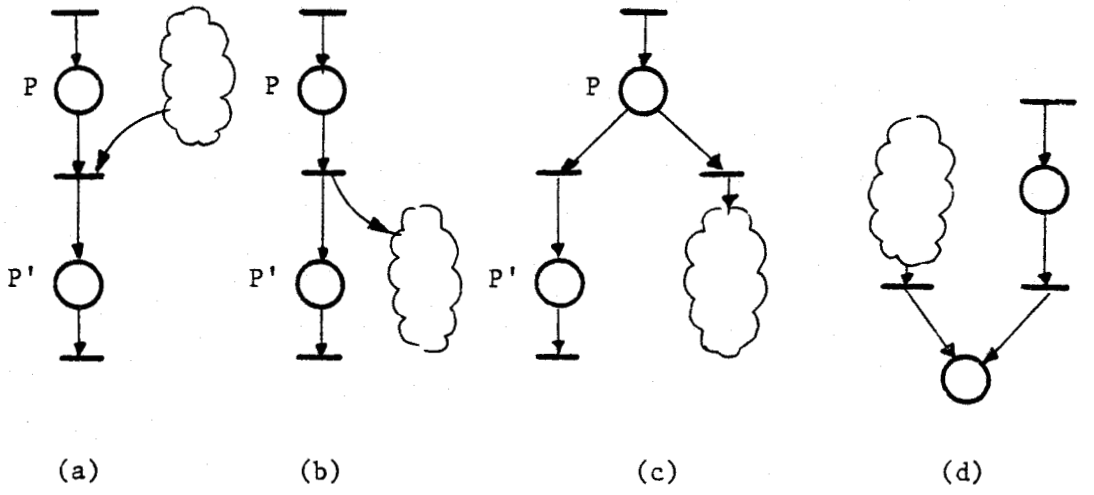


Figure III.18

Les figures III.18 (a) et (b) font apparaître des liaisons entre plusieurs processus et expriment ainsi une certaine forme de parallélisme alors que (c) et (d) décrivent une structure de choix.

Remarque : Ces deux types de structure se déduisent l'un de l'autre compte tenu de la dualité place/transition.

* Fusionnement :

La mise en œuvre de cette règle est aisée. Elle consiste tout d'abord à rechercher les transitions $t_k \in T$ telles que :

- p_i est la seule place d'entrée de t_k
- p_j est la seule place de sortie de t_k
- t_k est la seule transition de sortie de p_i
- t_k est la seule transition d'entrée de p_j

Le fusionnement consiste alors à supprimer la transition t_k et l'une ou l'autre place (p_i ou p_j).

En règle générale, si une ou les deux places sont marquées, celle dont le marquage est le plus élevé est conservée.

Exemple :

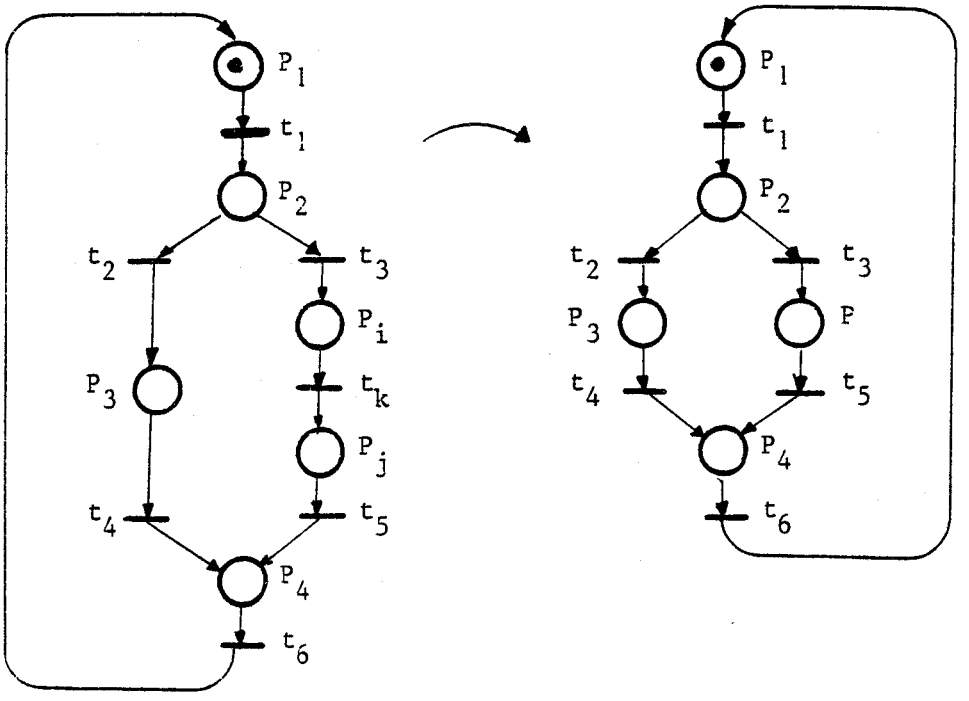


Figure III.19

Dans ces conditions, les fonctions d'entrée et de sortie de la place conservée, que nous noterons P_f et qui peut être soit p_i, soit p_j, sont telles que :

$$E(P_f) = E(p_i)$$
$$S(P_f) = S(p_j)$$

★ Conservation des propriétés :

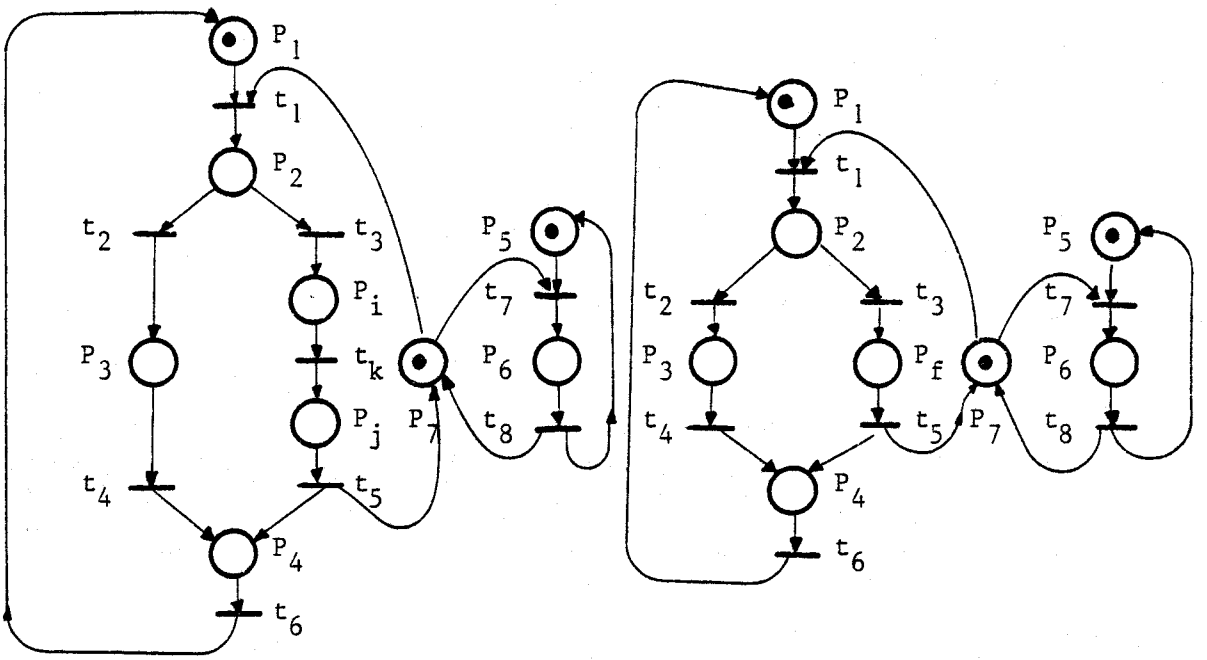
Montrons maintenant que cette règle conserve les propriétés initiales du réseau.

- Caractère vivant :

Si la transition t_k était non vivante, cela signifie que sa seule place d'entrée p_i n'est jamais marquée et donc que la transition d'entrée de cette place est également non vivante. La suppression de t_k conserve ainsi le caractère non vivant du réseau puisqu'il existe une autre transition non vivante.

Par contre, si le réseau est vivant, t_k est vivante. La suppression ne modifie pas le caractère vivant (il existe toujours une séquence de déclenchement δ_f, qui contient les transitions $E(P_f)$ et $S(P_f)$, tirable depuis tout marquage M de $\mathcal{E}(M_o)$).

Exemple :



P_i et P_j sont fusionnables

RESEAU NON VIVANT INITIAL — RESEAU REDUIT NON VIVANT

Figure III.20

- Caractère borné :

Cette vérification est également évidente. Si P_i ou P_j est non bornée, alors l'autre place est également non bornée. La place P_f qui remplace l'ensemble (P_i, t_k, P_j) sera également non bornée. Si l'une au moins est bornée, l'autre l'est également et par conséquent, P_f sera également bornée.

Remarque : On retrouve avec cette règle, un des aspects de la décomposi-

tion descendante d'un système, présenté dans le chapitre II. La démarche est ici inverse, il ne s'agit plus d'un raffinement de place mais d'une abstraction de place : c'est-à-dire, il est possible de remplacer un réseau simplement connexe possédant une place d'entrée et une place de sortie, par une seule place. De plus, nous venons de voir que cette transformation conserve les propriétés du réseau ce qui permet de prouver que la transformation inverse (raffinement), que nous avons présenté au chapitre II, gardait correct une réseau initialement correct.

* Mise en œuvre de cette règle :

Parenthèse : La représentation graphique d'un système à l'aide des règles de construction définies au chapitre II, conduit également à une représentation matricielle structurée du réseau.

En effet, la matrice d'incidence peut être structurée en blocs qui sont représentatifs soit d'un processus, soit des liaisons entre 2 processus.

Pour un réseau de Pétri structuré, la matrice d'incidence se met sous la forme suivante :

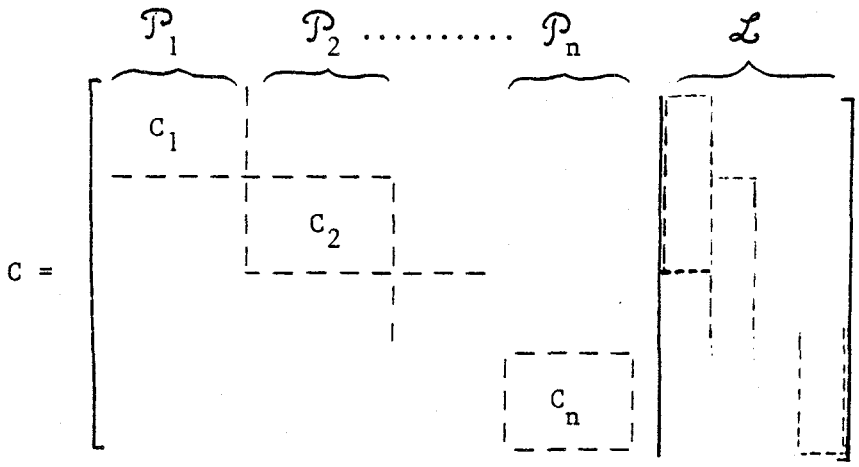
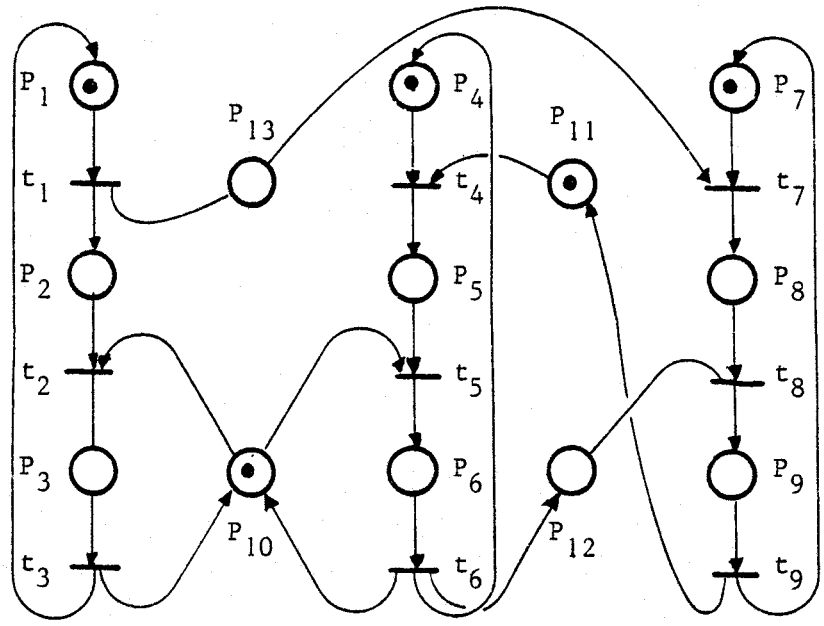


Figure III.21

Cette matrice a autant de blocs que le réseau comporte de processus (le nombre est donc représentatif du degré de parallélisme du système). Le bloc de liaison peut également être décomposé en blocs L_{ij} où L_{ij} est caractéristique des liaisons entre le processus i et le processus j.

Exemple :



$C =$

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}
t_1	1	-1	0							0			-1
t_2	0	1	-1		0			0		1		0	0
t_3	-1	0	1							-1			0
t_4				1	-1					0	1	0	0
t_5	0			0	1	-1		0		1	0	0	0
t_6				-1	0	1				-1	0	-1	0
t_7							1	-1	0		0		1
t_8	0				0		0	1	-1	0	0	1	0
t_9							-1	0	1		-1		0

.../...

.../...

$$C = \left[\begin{array}{ccc|c|c} C_{11} & 0 & 0 & & L_{31} \\ 0 & C_{22} & 0 & L_{12} & 0 \\ 0 & 0 & C_{33} & L_{23} & L_{31} \end{array}\right]$$

Processus Liaison

Figure III.22

Cette représentation matricielle est intéressante car elle permet, dans le processus de fusionnement de places, de ne tenir compte que d'une partie de la matrice d'incidence. Il est en effet inutile de chercher si deux places appartenant à 2 processus différents sont fusionnables.

La recherche des transitions t_k définies plus haut, commence par éliminer toutes les transitions d'un processus qui reçoivent ou transmettent une liaison (l'examen d'une partie du bloc de liaison le permet aisément).

- Procédure Fusionner

```
(* Gamma = application qui associe à un sommet, l'ensemble de ses succes-
         seurs immédiats *)
Pour chaque processus     Faire
    Pour chaque transition     Faire
        Si transition n'appartient pas au graphe de liaison   Alors
            Pplus := Gamma (transition)
            Pmoins := Gamma-moins-un (transition)
                Si Pplus a plus d'un prédécesseur   OU
                   Pmoins a plus d'un successeur   Alors
                     "Pplus et Pmoins ne sont pas fusionnables"
                                                   Sinon
                     (* Fusionnement *)
                     Si Marquage (Pplus) =  Marquage (Pmoins)   Alors
                          E(Pplus) := E(Pmoins)
```

.../...

```
.../...                          (* E = fonction d'entrée d'un sommet *)
                                 Supprimer (Pmoins et transition)
                                 Renuméroter
                                 Fusionner
                                                              Sinon
                                 E(Pmoins) := E(Pplus)
                                 Supprimer (Pplus et transition)
                                 Renuméroter
                                 Fusionner
                            FSi
                       FSi
            FPour
      FPour
FProcédure
```

ALGORITHME DE PRINCIPE DE FUSIONNEMENT

Figure III.24

Exemple :

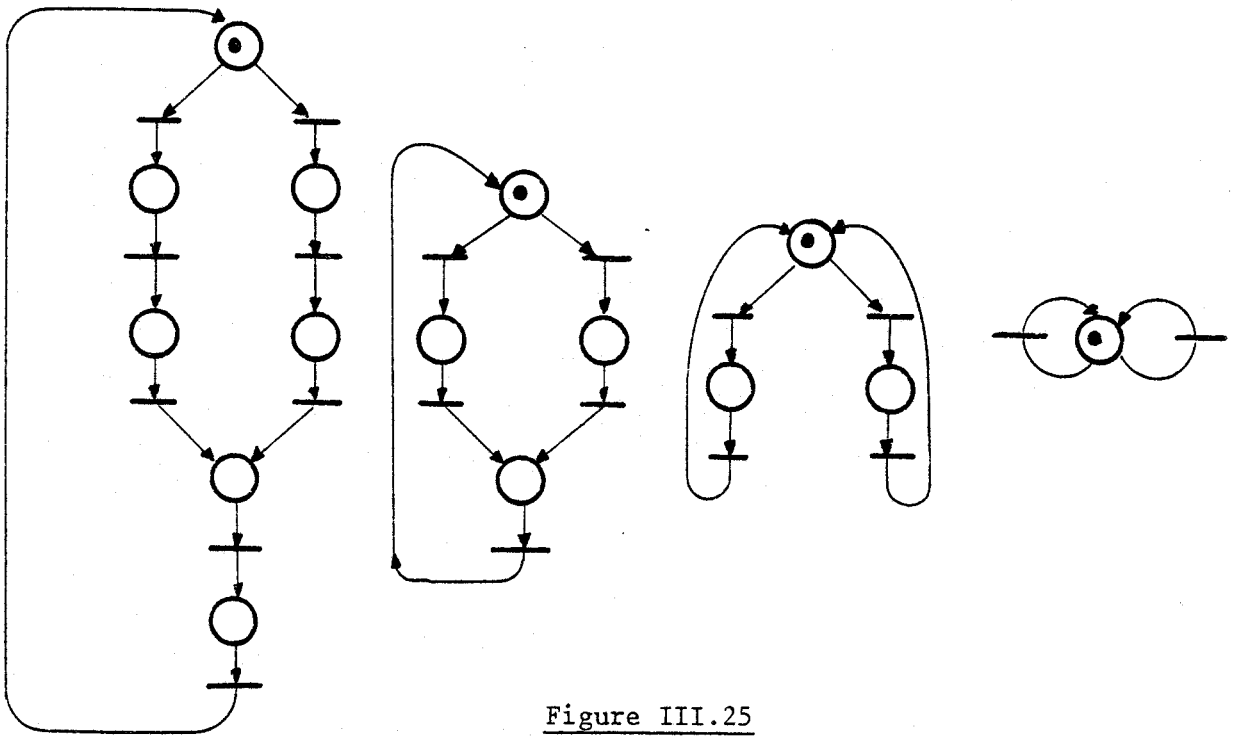


Figure III.25

On constate sur l'exemple ci-dessus, que cette règle, bien que restrictive dans sa définition, permet de réduire de façon importante un processus. Elle n'est cependant pas suffisante pour traiter des réseaux plus complexes. D'autres règles sont nécessaires et nous nous proposons de les présenter dans la suite de ce chapitre.

III.2.3.2 - Fusionnement de transition :

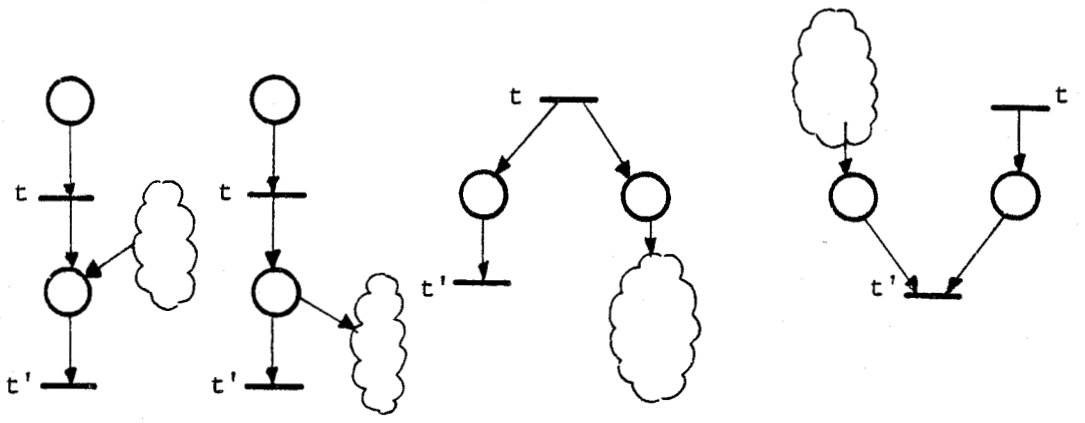
Cette règle est duale de celle que nous venons de présenter.

* Transition fusionnable :

Définition : Nous dirons que deux transitions t et t' d'un réseau de Pétri structuré $R = (P , T , \Gamma , M_o)$ sont fusionnables si et seulement si :

$$- \Gamma^{-1}(t) = \Gamma(t)$$
$$- \left|\Gamma^{-1}(t)\right| = \left|\Gamma(t)\right| = 1$$

Comme pour la règle précédente, nous définissons quatre cas de figures qui ne correspondent pas à cette définition. On remarque qu'ils se déduisent de ceux présentés sur la figure III.18 en remplaçant les transitions par des places et réciproquement.



(a) (b) (c) (d)

Figure III.26

* Fusionnement :

Le principe est identique à celui présenté pour le fusionnement de places. Il consiste à remplacer un graphe tpt' par une transition t_f qui aura mêmes entrées que t et mêmes sorties que t'.

La cas où la place p ($p = \Gamma(t) = \Gamma^{-1}(t')$) est marquée, pose un problème. En effet, si on supprime cette place (et son marquage), le réseau initial qui est vivant devient non vivant. Il est donc nécessaire de modifier le marquage initial du réseau réduit.

Les deux solutions suivantes sont possibles :

- ajouter une marque dans chaque place appartenant à $\Gamma^{-1}(t_f)$
- ajouter une marque dans chaque place de $\Gamma(t_f)$.

Exemple :

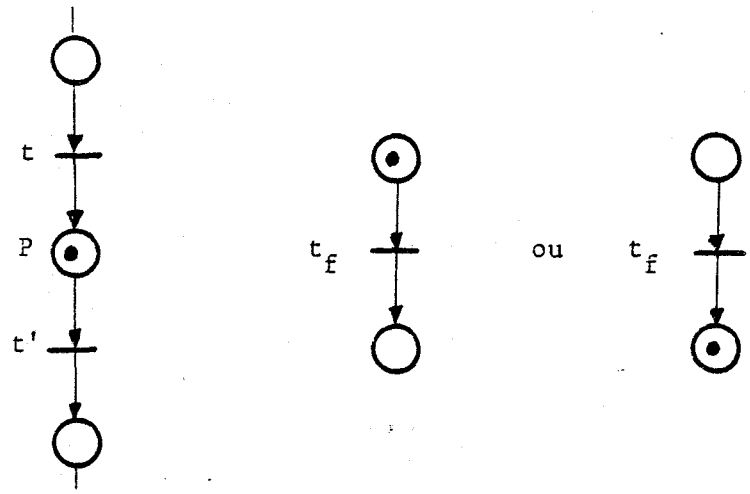


Figure III.27

* Conservation des propriétés :

a) Vivance :

Propriété : Le réseau réduit à l'aide de cette règle est vivant si le

réseau initial était vivant, et non vivant si le réseau initial était non vivant.

Preuve : Si t ou t' est non vivante, comme $\Gamma(t) = \Gamma^{-1}(t')$, alors l'autre transition est non vivante. Le fusionnement consistant à en supprimer l'une d'entre elles ainsi que leur place voisine commune, il reste donc au moins une transition non vivante d'où t_f est non vivante.

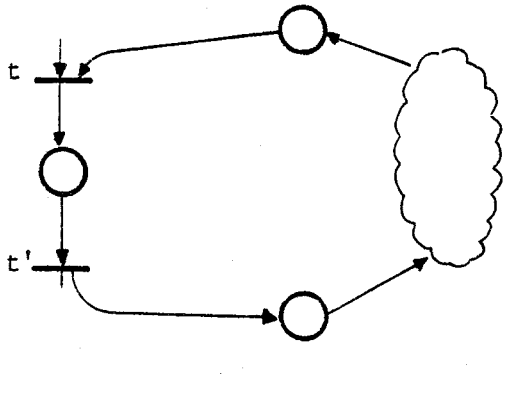
b) Borné :

Propriété : Un réseau réduit est borné si le réseau initial était borné, et non borné si le réseau initial était non borné.

Preuve : La place p n'a qu'une entrée et qu'une sortie, elle est donc par principe forcèment bornée. Si cette place est supprimée, le réseau sera borné si toutes les places du réseau initial p' telles que $p' \in P - \{p\}$ sont bornées, ou le réseau réduit sera non borné s'il existe $p' \in P - \{p\}$ qui est non borné.

* Mise en œuvre :

Elle se fera en plusieurs étapes. En effet, le but que nous nous sommes fixés consiste à réduire le réseau de façon à faire apparaître la structure de bloc de chaque processus et le graphe de liaison du réseau. Aussi, dans un premier temps, nous interdisons nous le fusionnement de deux transitions dont la première reçoit un arc de liaison et la seconde transmet un arc de liaison selon le schéma décrit sur la figure suivante.



Si $|\Gamma^{-1}(t)|$ et $|\Gamma(t')| \geq 1$ alors t et t' ne seront pas fusionnées.

Figure III.28

Cette contrainte supplémentaire étant imposée, la mise en œuvre de cette méthode est identique à celle proposée dans le § III.2.2.

De plus, une remarque intéressante permet de réutiliser le programme de fusionnement de places :

Remarque : Compte tenu de la dualité Place/Transition, il est possible d'exécuter cette règle à l'aide du programme défini pour le fusionnement de places. Il suffit pour cela de transposer la matrice d'incidence du réseau.

Cette remarque a été en particulier, mise à profit dans la recherche des transitions fusionnables et dans le renumérotation des transitions.

Exemple de fusionnement d'une structure TANT QUE
(impossible à réaliser avec fusionnement de places).

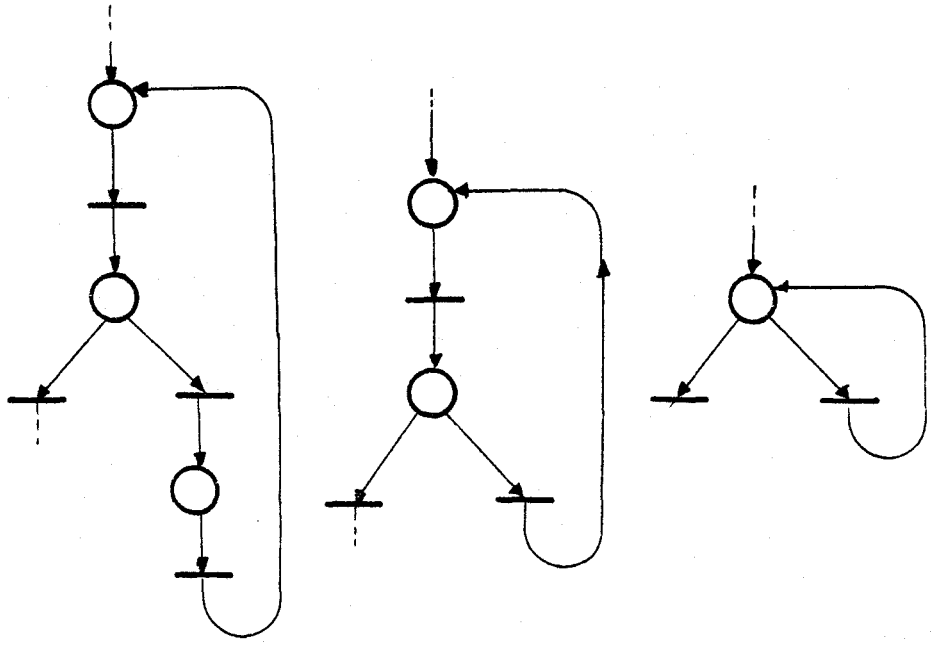


Figure III.29

Cette règle constitue en réalité une abstraction de transition. C'est l'application réciproque de celle présenté au § II. . .

Elle permet de transformer un sous-graphe ayant une transition d'entrée et une transition de sortie, en une seule transition.

L'invariance des propriétés du réseau par cette transformation, prouve ainsi que le processus de raffinement que nous avions alors proposé, laisse également correct un réseau correct.

III.2.3.3 - Simplification de transitions et de places identiques :

Définition : Deux transitions (ou deux places) sont identiques si elles ont même fonction d'entrée et même fonction de sortie.

* Mise en œuvre :

Cette règle regroupe en quelque sorte, une partie de la règle R3 (présentée au § III.2.2.3) et une partie de la règle R2, pour le cas particuliers des places identiques.

Elles sont réunies ici car leur mise en œuvre exploite la dualité Place/Transition. Un même programme permet de traiter ces deux cas.

L'algorithme de réduction de places identiques est le suivant :

```
Procédure REDUIRE (NP ...)
  (* NP : nombre de places du réseau de Pétri *)

   Pour   I = 1  à  NP - 1
       Pour J = I + 1  à  NP
           Si  E(Pi) = E(Pj) et S(Pi) = S(Pj)
               Alors   Si M(pi)   M(pj)
                         Alors   Supprimer Pj
                                 Renuméroter P
                                 NP = NP - 1
                                 Réduire (NP)
                         Sinon   Supprimer Pi
                                 Renuméroter P
                                 NP = NP - 1
                                 Réduire (NP)
                       FSi
           FSi
       FPour
   FPour
```

La place dont le marquage est le plus élevé est conservée, l'autre est supprimée.

L'algorithme de réduction de transitions se déduit de celui-ci en remplaçant p_i par t_i, NP par NT, P par T (dans ce cas, le test des marquages est inutile).

En pratique, un même et seul programme traite ces deux types de réduction.

Il faut cependant noter que dans la mesure où l'on ne désire que se ramener à la structure de bloc et au graphe de liaison, il n'est pas nécessaire d'appliquer la règle de simplification des places identiques. En effet, dans un système de processus, si deux places sont identiques, l'une au moins d'entre elles appartient au graphe de liaison ; or, nous voulons précisèment conserver le graphe de liaison.

* Conservation des propriétés :

Il est évident que si deux places ou deux transitions sont identiques, elles ont les mêmes propriétés. Supprimer l'une d'entre elles ne modifie donc pas le caractère de l'autre.

Ainsi, si une transition identique est non vivante, l'autre l'est aussi et comme la non-vivance de ces transitions est liée au marquage de leurs places d'entrée, la suppression de l'une d'entre elles n'affecte pas le caractère non vivant de l'autre. Le réseau réduit sera donc également non vivant. S'il était vivant, pour les mêmes raisons, il resterait vivant après transformation.

De même, si une place identique est non bornée, l'autre l'est également. Si l'on supprime celle dont le marquage initial est le plus faible, il restera encore une place non bornée.

III.2.3.4 - Simplification des places et des transitions identités :

* Rappels et principes :

Cette règle comprend également une partie de la règle R2 et une partie de la règle R3 de |3|.

Ces deux parties sont regroupées ici afin de tenir compte de la dualité Place/Transition.

Cette propriété permet en effet de simplifier la mise en œuvre de ces règles. On se ramène souvent à l'écriture d'un seul programme permettant de traiter l'une ou l'autre des simplifications.

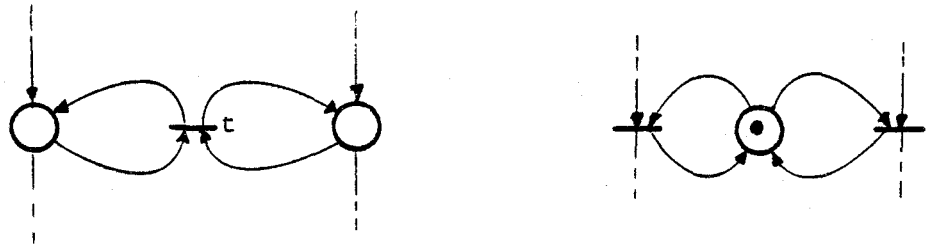
Définition : Une transition $t \in T$ d'un réseau de Pétri structuré $R = (G , M_o)$ est une transition identité si et seulement si

$$\forall p \in P \qquad E(t,p) = S(t,p)$$

Définition : Une place $p \in P$ d'un réseau de Pétri structuré $R = (G , M_o)$ est une place identité si et seulement si

$$\forall t \in T \qquad E(t,p) = S(t,p)$$
$$\text{et } M_o(p) \geqslant 1$$

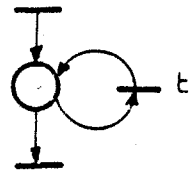
Exemple :

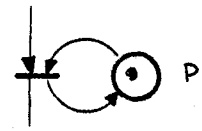


(a) transition identité (b) place identité

Figure III.30

Les cas particuliers qui nous intéressent plus particulièrement sont les suivants :





(a) transition identité (b) place identité

Figure III.31

En effet, les exemples de la figure III.31 font apparaître des liaisons entre processus que nous conserverons dans une première phase de réduction.

Une place identité p peut être supprimée dans la mesure où elle ne constitue pas un obstacle de tir de la (ou des) transition(s) t telle(s) que $E(t,p) = S(t,p)$.

De même, une transition identité t peut être supprimée dès lors que pour un réseau non vivant, il existe $t' \in T - \{t\}$ également non-vivante. La suppression n'affecte pas le caractère borné de la place sur laquelle elle se reboucle. En effet, comme $S(t,p) = E(t,p)$ alors le tir de cette transition conduit au marquage :

$$M'(p) = M'p) + S(t,p) - E(t,p)$$

soit encore : $M'(p) = M'(p)$.

Le marquage de cette place reste inchangé par la mise à feu de cette transition (ceci est vrai pour chaque place p qui est à la fois entrée et sortie de t).

Cette règle conserve donc également les propriétés du réseau initial.

III.2.4 - Heuristique de réduction

III.2.4.1 - Notation :

Notation : Nous noterons

(1) SP : la règle de fusionnement des places

(2) ST : la règle de fusionnement des transitions

(2.1) S'_t : la règle de fusionnement de transitions qui ne prend pas en compte les transitions fusionnables t et t' telles que $|\Gamma^{-1}(t)| > 1$ et $|\Gamma(t)| > 1$ (c'est-à-dire les transitions t et t' qui correspondent à la figure ci-dessous).

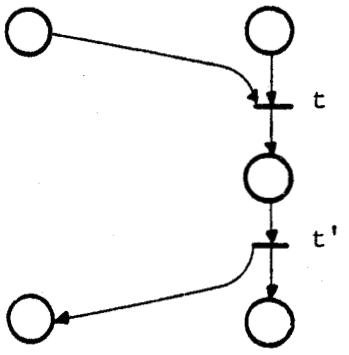


Figure III.32

(3) SPI : la règle de simplification d'une place identique

(4) STI : la règle de simplification d'une transition identique

(5) SPIté : La règle de simplification d'une place identité

(6) STIté : la règle de simplification d'une transition identité.

III.2.4.2 - Heuristique :

Les règles que nous venons de proposer peuvent être exécutées indépendamment les unes des autres.

Dans un premier temps, nous nous attacherons à faire apparaître la structure de bloc et les liaisons. Cette démarche est importante car l'interprétation des places de liaisons, des structures de blocs et donc également les sections critiques associées aux ressources partagées, permet de vérifier certaines spécificités du système. En fait, la structure du graphe est réduite mais reste inchangée ce qui permet donc de retrouver sur le graphe réduit, une forme d'interprétation des propriétés du système par le marquage des places de liaison, des sections critiques, ...

Sur ce graphe partiellement réduit, une analyse de certaines particularités du système (et non plus du modèle) pourra alors être faite.

Les règles qui permettent d'obtenir un tel graphe sont successivement :

- SP
- ST'
- STI
- SPIté
- STIté.

L'ordre dans lequel elles sont appliquées importe peu, nous retiendrons la proposition ci-dessus.

Afin de réduire davantage le réseau, on appliquera la règle notée ST à la place de ST' et également la règle de simplification des places identiques.

Remarque : Un réseau totalement réductible se réduit à une transition ou à une place.

Exemple :

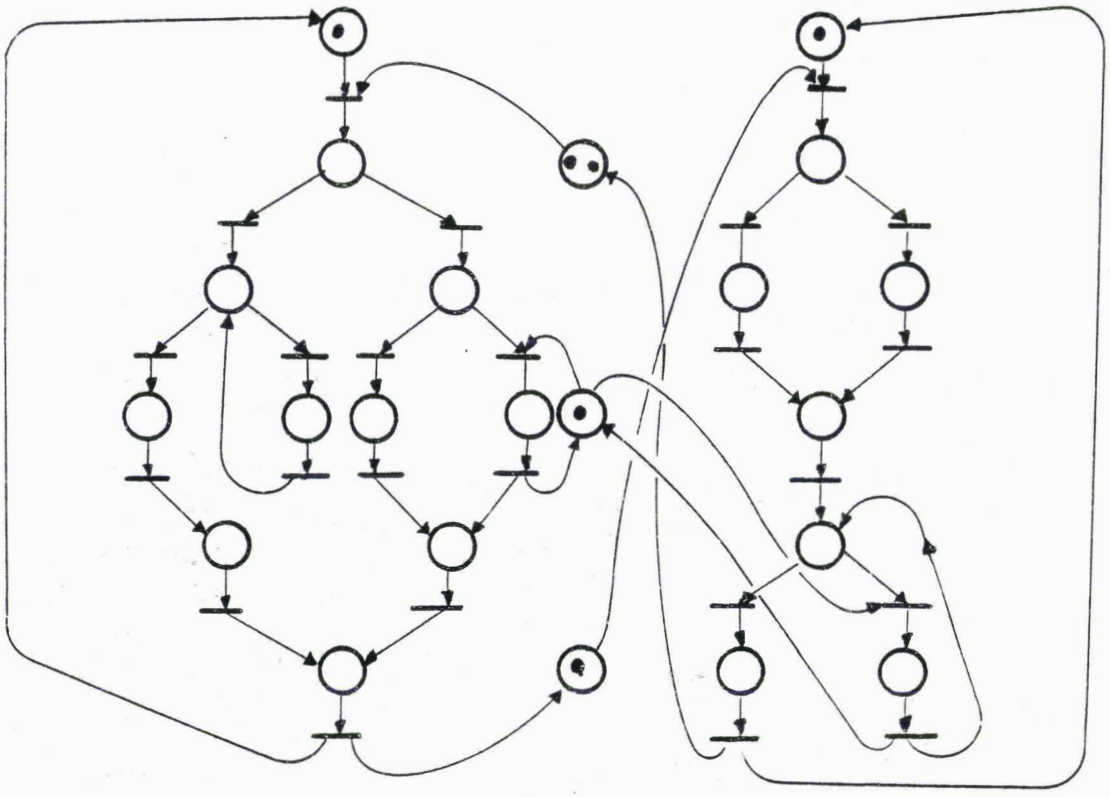
Si l'application de toutes ces règles ne conduit pas à un réseau totalement réductible, nous avons encore la possibilité d'utiliser les règles définies pour des réseaux quelconques.

Si le graphe n'est toujours pas totalement réduit, il faut alors se tourner vers d'autres méthodes :

- méthodes classiques
- méthodes structurelles

et les appliquer sur le réseau réduit.

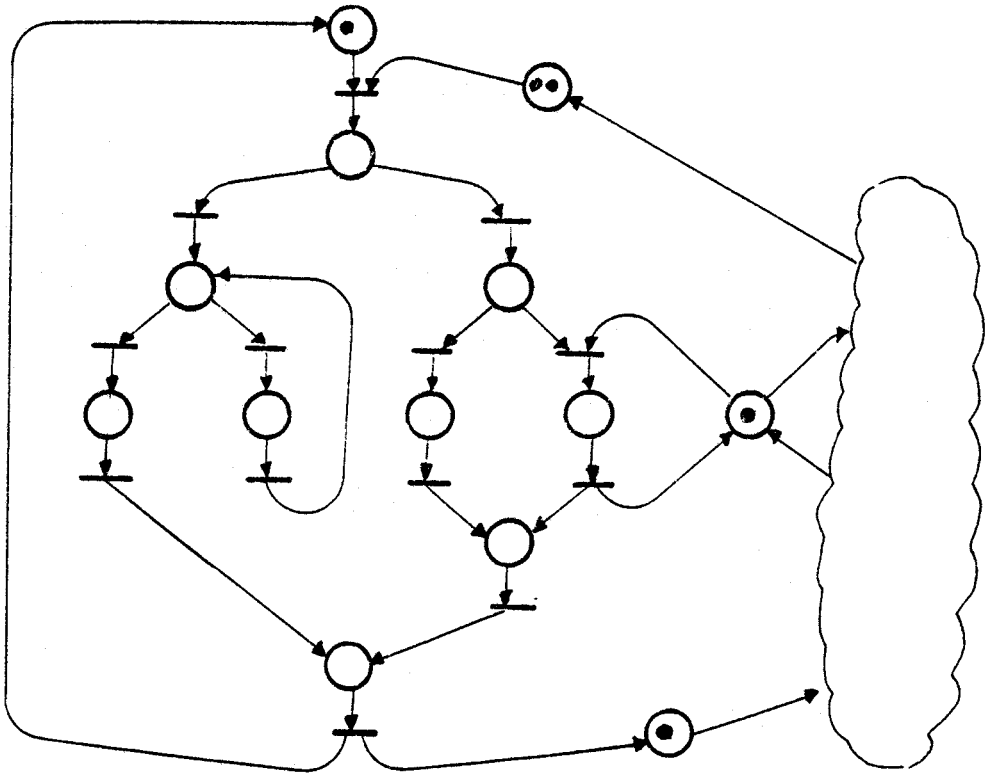
Exemple :



Réseau initial

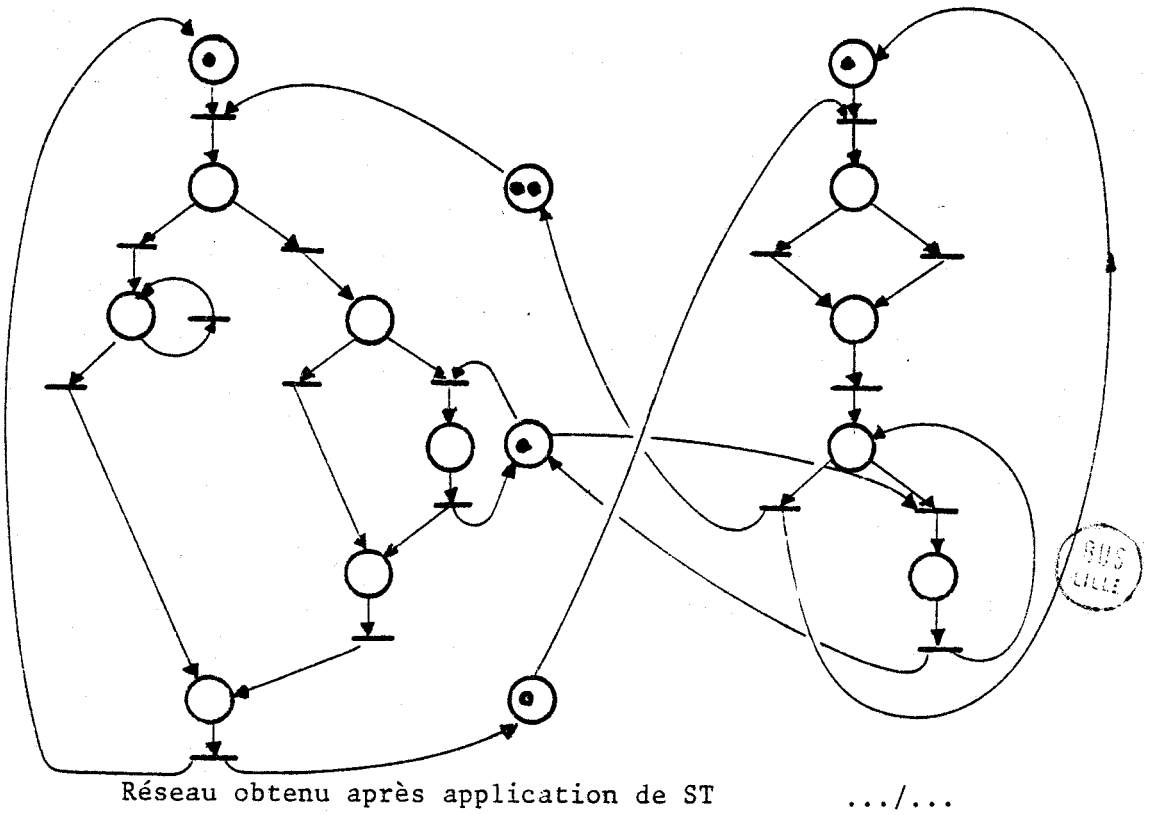
.../...

SP



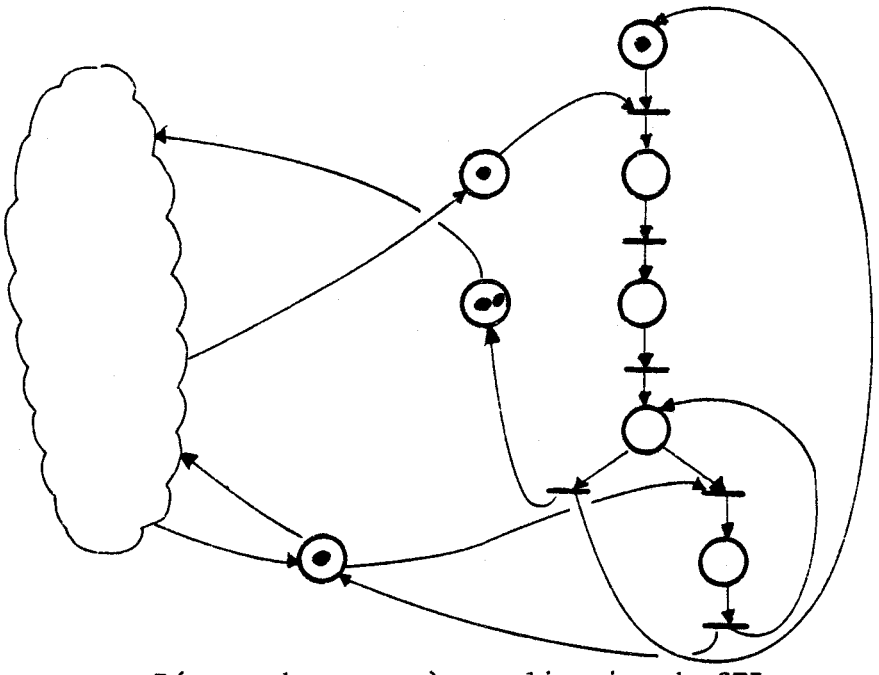
Réseau obtenu après application de SP

ST

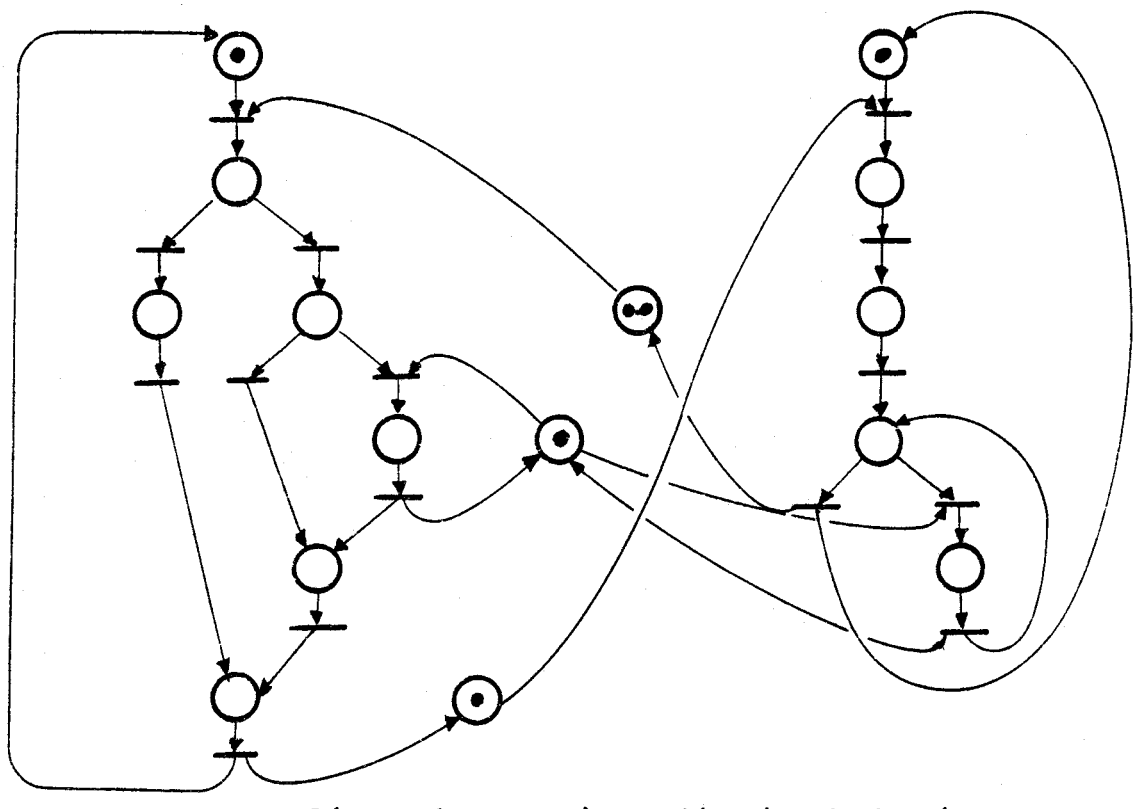


Réseau obtenu après application de ST

.../...

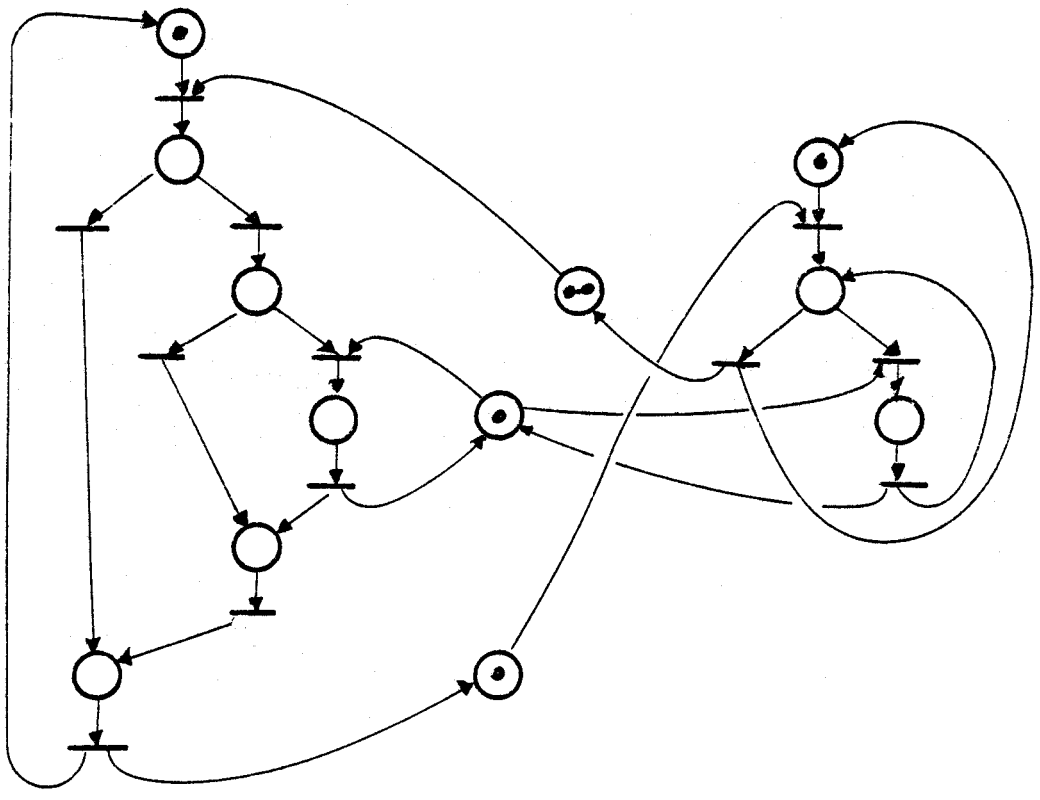


Réseau obtenu après application de STI

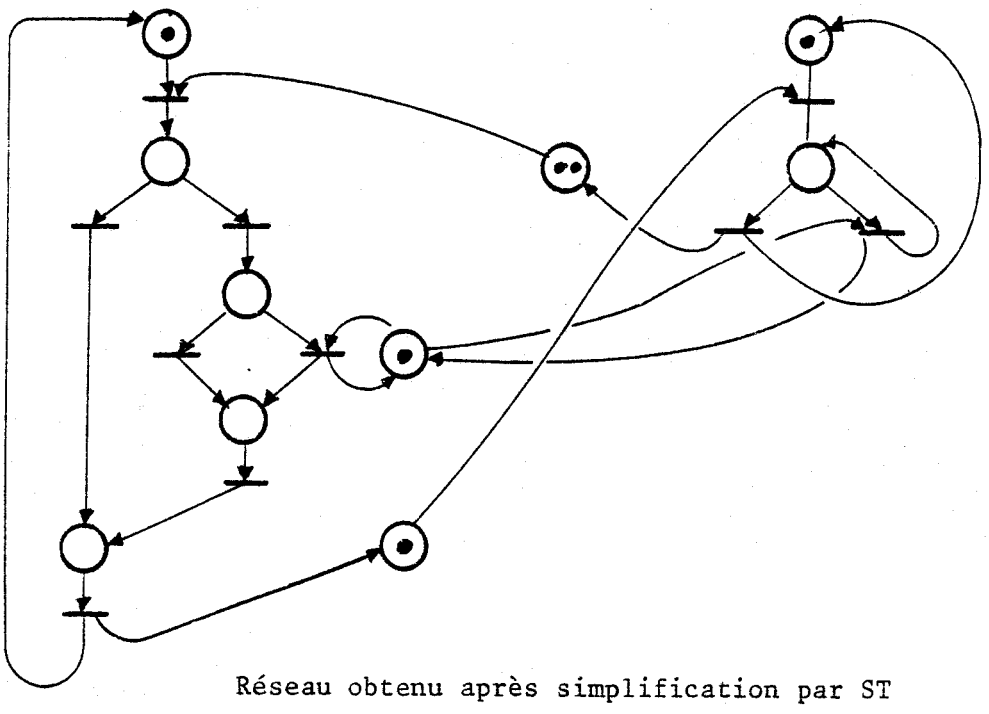


Réseau obtenu après application de STIté

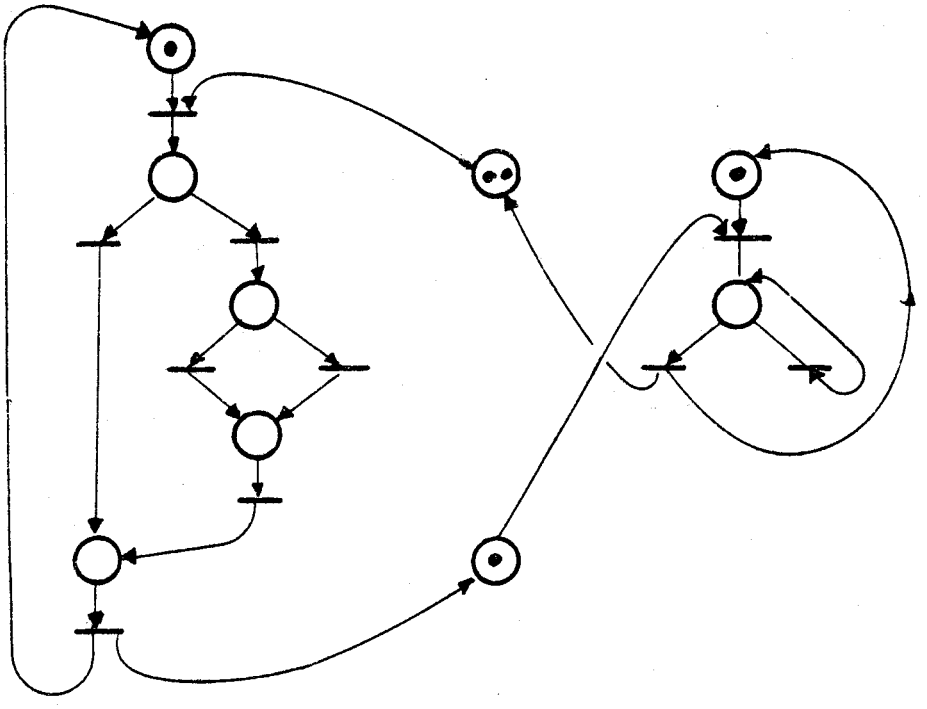
On recommence ensuite le processus jusqu'au moment où aucune simplification n'est possible.



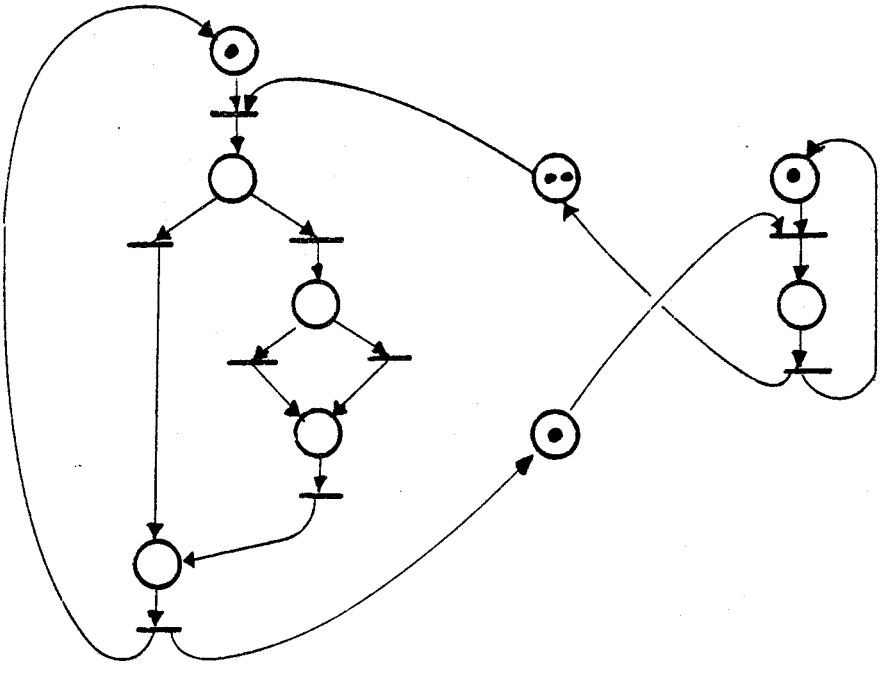
Réseau réduit à la structure de bloc et au graphe de liaison.



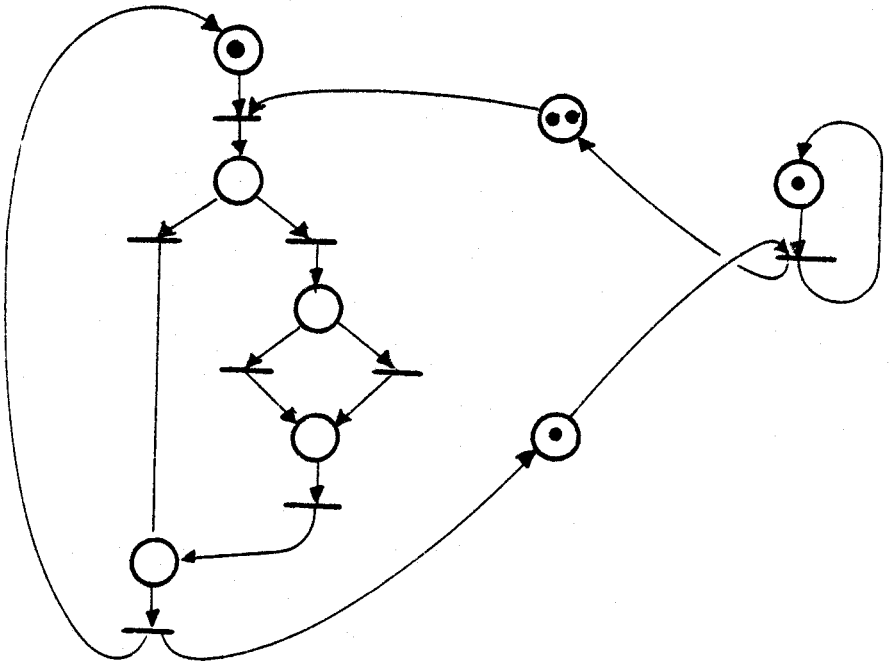
Réseau obtenu après simplification par ST



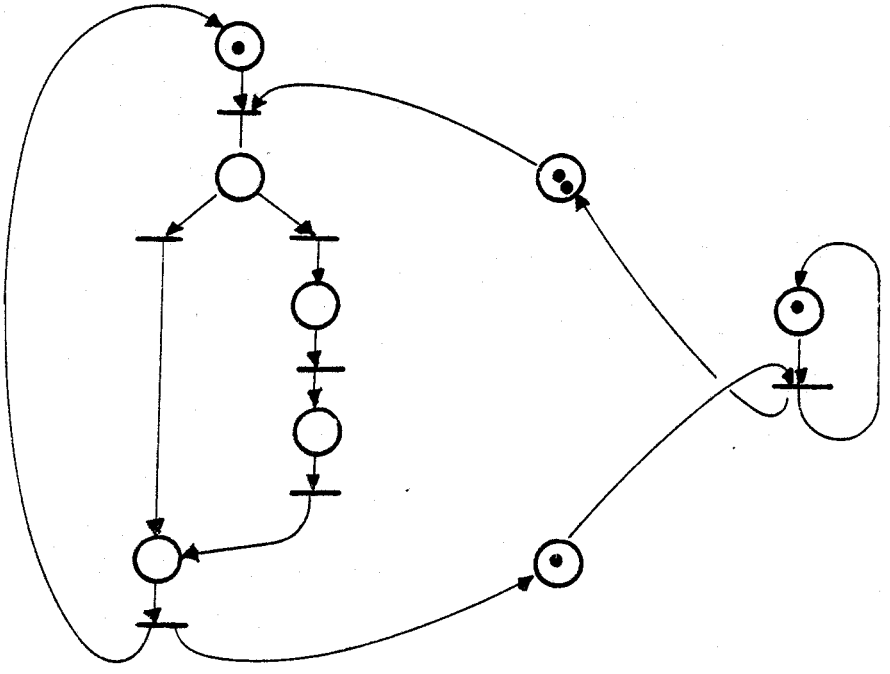
Réseau obtenu après simplification par SPIté



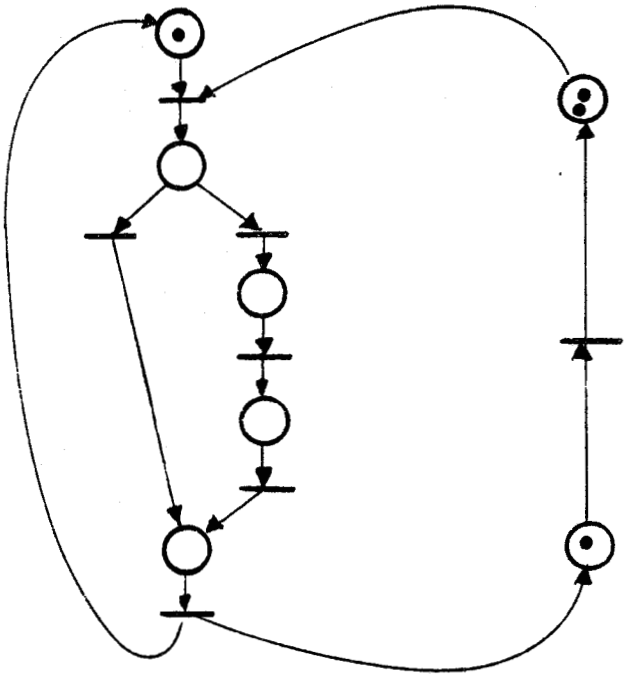
Réseau obtenu après simplification par STIté



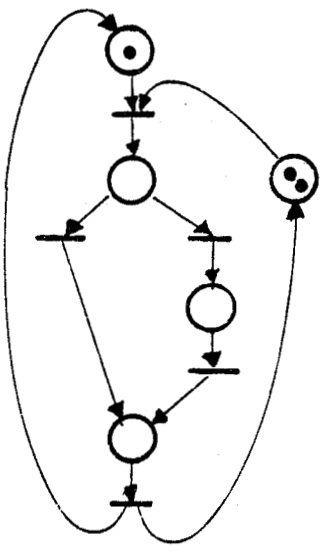
Règle ST



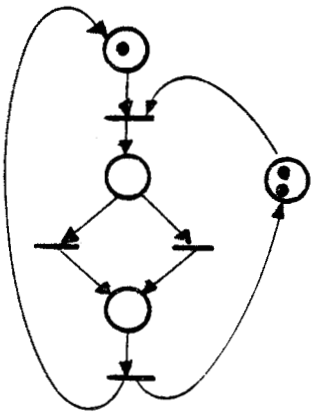
Règle STI



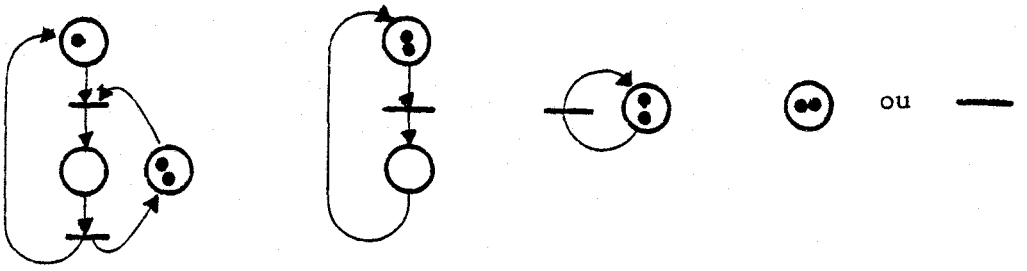
Règle SPIté



(SP)



(ST)



Le réseau initial était donc borné et vivant.

III.2.5 - Conclusion

La réduction d'un réseau de Pétri est intéressante à plusieurs titres. Elle permet :

- une analyse plus aisée à l'aide des méthodes classiques ou des méthodes structurelles.

- de conclure, dans certains cas; directement sur les propriétés du modèle.

Afin d'exploiter les caractéristiques du modèle que nous utilisons, il est nécessaire d'adapter les règles générales de réduction [3].

Ce point de vue nous a donc amenés à définir des règles plus restrictives mais dont la mise en œuvre est plus aisée car elles tiennent compte de la dualité Place/Transition. Ainsi, les programmes de réduction des places peuvent être utilisés pour réduire les transitions simplement en transposant les matrices caractéristiques du réseau (c'est à dire les matrices E et S). Ces règles conservent les propriétés du réseau initial.

Un autre aspect important qui a pu être signalé dans ce chapitre concerne l'analogie qui existe entre la réduction d'un réseau de Pétri et l'abstraction d'un modèle.

Les réductions que nous proposons s'opèrent en deux phases :

- la 1ère phase permet de conserver le graphe de liaison. Les places de liaisons symbolisent génèralement une ou plusieurs ressources,

elles peuvent également introduire le concept de synchronisation entre tâches, une analyse structurelle sur le modèle partiellement réduit permet alors de vérifier les propriétés relatives aux contraintes de liaison (assertions de liaison).

- La 2ème phase conduit soit à un réseau totalement réduit (le réseau initial était borné et vivant), soit à un réseau irréductible qui doit alors être analysé par des méthodes classiques ou structurelles.

III.3 - METHODES DE L'ANALYSE STRUCTURELLE

III.3.1 - Introduction

Les méthodes que nous nous proposons d'exposer ici permettent l'analyse d'un réseau de Pétri indépendamment de son marquage initial. Celui-ci est considéré comme un paramètre.

Deux types de vérification sont ainsi permises :

- vérification des propriétés internes du modèle
- vérification des propriétés spécifiques du système modélisé.

Cependant, ce que nous avons appelé au chapitre II la sémantique du réseau ne peut être totalement validée que dans la mesure où il s'avère possible de traduire les propriétés du système sous forme de marquage ou de déclenchement. Cette contrainte doit donc être prise en considération dès la définition du modèle par un réseau de Pétri structuré afin de permettre une validation plus complète des spécifications du système.

Ces méthodes, dont les fondements appartiennent à la théorie des graphes et à l'algèbre linéaire, fournissent des conditions suffisantes ou nécessaires permettant de garantir le bon fonctionnement d'un système. Elles sont, dans ce sens, complémentaires des méthodes présentées auparavant. En effet, si le modèle n'est pas validé, il est nécessaire de recourir aux méthodes traditionnelles exposées dans les paragraphes III.1 et III.2 (énumération de marquages).

Nous mettrons en évidence dans quelle mesure l'utilisation des réseaux de Pétri structurés permet de faciliter la mise en œuvre de ces méthodes que nous aurons présentées au préalable.

III.3.2 - Propriété structurelle

III.3.2.1 - Equation d'état d'un réseau de Pétri [5] :

Nous avons vu au chapitre II que la mise à feu d'une transition sensibilisée t par le marquage initial M_o, conduisait à un marquage M' dont les composantes étaient obtenues de la façon suivante :

$$\forall p \in P \qquad M'(p) = M_o(p) + C(t,p)$$

Cette relation se généralise pour une transition t sensibilisée par un marquage quelconque M appartenant à l'ensemble des marquages $\mathcal{C}(M_o)$.

$$\forall p \in P \qquad M'(p) = M(p) + C(t,p)$$

De plus, si l'on considère non plus la mise à feu d'une seule transition mais celle d'une séquence de déclenchement U_k de degré $\overline{U_k}$ (cf II.12) tirable depuis un marquage $M_k \in M_o$ conduisant à un marquage M_{k+1}, nous obtenons la relation qui suit :

$$\boxed{M_{k+1} = M_k + C^T \overline{U}_k} \qquad (1)$$

Cette formulation est semblable à celle que l'on rencontre en théorie des systèmes linéaires, pour la représentation dans l'espace d'état d'un système échantillonné linéaire :

$$\boxed{X_{k+1} = M X_k + F U_k} \qquad (2)$$

pour laquelle X_k représente le vecteur d'état, et U_k le vecteur de commande du système.

L'état d'un réseau de Pétri est quant à lui, représenté par son marquage ; le degré d'une séquence de tir pouvant être interprété comme un

vecteur d'entrée.

Remarque : Pour un système échantillonné linéaire, la matrice M est caractéristique de la structure du système alors que pour un réseau de Pétri il s'agit plutôt de la matrice C.

III.3.2.2 - Consistance et conservation :

Comme pour les systèmes en théorie de l'automatique, il est possible de faire apparaître les notions de stabilité linéaire et de commandabilité : les composantes consistantes et conservatrices d'un réseau de Pétri.

La recherche de ces composantes et des propriétés qu'elles introduisent ont susciter de nombreux travaux |16| |17| |19| |24| |25|.

Elle implique généralement la résolution de système linéaire obtenus à partir de la matrice d'incidence du réseau.

* Composante conservatrice. Invariant de place :

a) Définition :

Soit $\mathcal{R}$ un réseau de Pétri défini par $\mathcal{R} = (P, T, \Gamma, M_o)$ de matrice d'incidence C.

Définition : Soit I_p solution à composantes non négatives de l'équation :

$$\boxed{CX = [0]} \qquad (3)$$

où $[0]$ est un vecteur colonne à $|T|$ composantes.
Alors cette solution constitue un invariant de place et définit, à partir de l'ensemble des places du réseau correspondant à ces composantes non nulles et des transitions qui leur sont adjacentes, une composante conservatrice.
(I_p est un vecteur colonne à $|P|$ composantes).

Définition : Soit I_p un invariant de place d'un réseau $\mathcal{R} = (P, T, \Gamma, M_o)$, on appelle support d'un invariant I_p le sous-ensemble SUPP de l'ensemble des places du réseau R tel que :

$$SUPP = \left\{ p \in P \;/\; I_p(p) \neq 0 \right\}$$

b) Interprétation :

Soit l'équation d'état du réseau $\mathcal{R} = (G , M_o)$

$$M = M_o + C^T X,$$

en multipliant à gauche, les deux membres de cette égalité par le vecteur I_p^T, nous obtenons la relation suivante :

$$\boxed{I_p^T M = I_p^T M_o} \qquad (4)$$

Ce qui traduit le fait que la somme des nombres de marqueur des places de la composante conservatrice pondérée par les composantes de I_p^T est constante quelquesoit l'évolution du marquage depuis M_o.

Exemple :

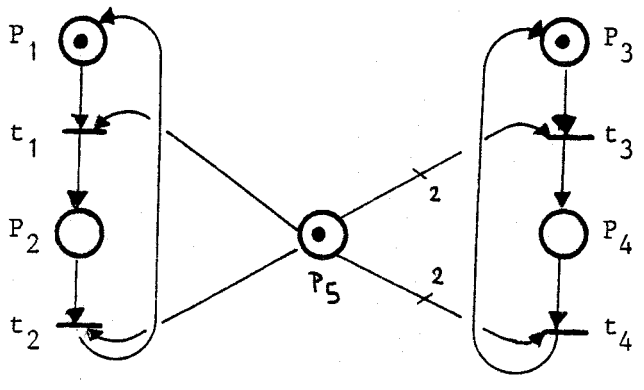


Figure III.34

La matrice d'incidence de ce réseau est présenté ci-dessous :

$$C = \begin{array}{c} \begin{matrix} P_1 & P_2 & P_3 & P_4 & P_5 \end{matrix} \\ \left[\begin{array}{cccc|c} -1 & 1 & & & -1 \\ 1 & -1 & & 0 & 1 \\ \hline & & -1 & 1 & -2 \\ & 0 & 1 & -1 & 2 \end{array}\right] \end{array} \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{matrix}$$

Les invariants ci-dessous vérifient la relation C X = 0 :

$$I_p^{1T} = [1\ 1\ 0\ 0\ 0] \qquad SUPP^1 = \{P_1 , P_2\}$$

$$I_p^{2T} = [0\ 0\ 1\ 1\ 0] \qquad SUPP^2 = \{P_3 , P_4\}$$

$$I_p^{3T} = [0\ 1\ 0\ 2\ 1] \qquad SUPP^3 = \{P_2 , P_4 , P_5\}$$

Remarque : Il est intéressant de constater que pour cet exemple, I_p^1 et I_p^2 déterminent des composantes conservatrices qui sont en fait le réseau de Pétri de chaque processus, alors que I_p^3 définit une composante conservatrice relative au graphe de liaison.

Théorème :

Soit I_p^i et I_p^j deux invariants de places d'un réseau $\mathcal{R} = (G , M_o)$.

Alors $\forall \lambda, \mu \in N^* ; \lambda I_p^i + \mu I_p^j$ est également un invariant de places du réseau.

Démonstration :

La relation $C(\lambda I_p^i + \mu I_p^j)$ peut encore s'écrire :

$$\lambda C I_p^i + \mu C I_p^j$$

or, I_p^i et I_p^j sont des invariants du réseau ; ils vérifient donc la relation C X = [0] et par conséquent : $C(\lambda I_p^i + \mu I_p^j) = [0]$.

Soit $SUPP^i$ le support de I_p^i et $SUPP^j$ celui de I_p^j ; alors le support de $\lambda I_p^i + \mu I_p^j$ est égal à la réunion de $SUPP^i$ et de $SUPP^j$.

Exemple : Pour l'exemple de la figure III.34, la somme des trois invariants trouvés, permet d'obtenir un invariant de place dont le support vaut P.

$$I_p^{\text{total}} = I_p^1 + I_p^2 + I_p^3 = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 3 \\ 1 \end{bmatrix}$$

$$SUPP = SUPP^1 \cup SUPP^2 \cup SUPP^3 = P$$

Définition : Nous dirons qu'un invariant de place est total si son support est l'ensemble des places du réseau. Ce dernier constitue à lui seul, une composante conservatrice (c'est le cas de la figure III.34).

Définition : Un réseau de Pétri est dit invariant s'il admet un invariant total de places.

La relation $I_p^T M = I_p^T M_o$ où I_p représente un invariant de place, permet de définir un invariant de marquage k tel que :

$$\boxed{k = I_p^T M = I_p^T M_o}$$

Exemple : L'invariant I_p^3 du réseau de la figure III. , définit l'invariant de marquage k suivant :

$$\forall M \in \mathcal{E}(M_o) \qquad M(P_2) + 2\,M(P_4) + M(P_5) = 2 = k$$

Propriété 1 : Tout invariant de marquage k associé à un invariant de place I_p, définit une borne supérieure pour le marquage de chaque place du support de I_p.

En effet, en annulant les coefficients des marquages des autres places du support de I_p, et compte tenu du fait que ceux-ci sont non négatifs,

on obtient pour chaque place p du support de I_p :

$$\forall M \in \mathcal{C}(M_o) \qquad M(p) \leq \frac{I_p^T M_o}{I_p(p)} \qquad (5)$$

Exemple : Pour I_p^3 du réseau de la figure III. , nous obtenons l'invariant de marquage :

$$M(p_2) + 2\,M(p_4) + M(p_5) = 2$$

qui nous permet d'affirmer d'après (5) :

$$M(p_4) \leq 1$$

$$M(p_2) \leq 2$$

$$M(p_5) \leq 2$$

Propriété 2 : Soit I_p, un ensemble d'invariants de place de support contenant la place p.

$$I_p = (I_p^1, I_p^2, \ldots, I_p^r) \qquad \text{avec } r = |I_p|$$

Alors, nous avons :

$$\forall M \in \mathcal{C}(M_o) \qquad M(p) \leq \min_{i=1 \text{ à } r} \frac{I_p^i M_o}{I_p(p)} \qquad (6)$$

Autrement dit, l'ensemble I_p permet de définir une borne minimum au marqeage de la place p.

$$(p \in SUPP^1 \cap SUPP^2 \cap \ldots \cap SUPP^r)$$

Exemple : Soient

$$I_p^1 = \begin{vmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{vmatrix} \qquad I_p^2 = \begin{vmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{vmatrix} \qquad I_p^3 = \begin{vmatrix} 0 \\ 1 \\ 0 \\ 2 \\ 1 \end{vmatrix}$$

Ces invariants de places fournissent les invariants de marquage suivants :

$$\forall M \in \mathcal{E}(M_o) \qquad M(p_1) + M(p_2) = 2$$

$$M(p_3) + M(p_4) = 1$$

$$M(p_2) + 2\,M(p_4) + M(p_5) = 2$$

Une borne supérieure du marquage de la place p_2 est :

$$M(p_2) = \min\,(1\ ,\ 2)$$

Ces propriétés sont importantes car elles permettent de décider du caractère borné de certaines places du réseau.

Propriété 3 : Si un réseau de Pétri est lui-même une composante conservatrice, alors il est borné pour tout marquage.

Cette propriété peut également se formuler de la façon suivante :

> Si un réseau de Pétri admet un invariant total de place, alors il est borné.

La démonstration de cette propriété est évidente. Si le réseau admet un invariant total de place, alors chaque place du réseau appartient au support de cet invariant et par conséquent, d'après les propriétés 1 et 2, cette place est bornée.

Exemple : Le réseau de Pétri de la figure III.34 admet l'invariant total I_p^{total} :

$$I_p^{total} = \begin{vmatrix} 1 \\ 2 \\ 1 \\ 3 \\ 1 \end{vmatrix}$$

il est donc borné.

* Composante consistante. Invariant de transition :

a) Définition :

Définition : Soit I_t, solution à composantes non négatives de l'équation

$$\boxed{C^T Y = [0]} \qquad (7)$$

où $[0]$ est un vecteur colonne à $|P|$ composantes. Alors, nous dirons que I_t constitue pour le réseau $\mathcal{R} = (P, T, \mathbf{I}, M_o)$ un invariant de transitions et définit à partir de l'ensemble des transitions du réseau correspondant à ces composantes non nulles et des places qui leurs sont adjacentes (places amont et aval), une composante consistante.

b) Interprétation :

Propriété : Les invariants I_t correspondent aux degrés de séquences de tirs cycliques non vides.

Définition : Une séquence de tir cyclique est une séquence de tir tirable depuis un marquage $M \in \mathcal{C}(M_o)$ et conduisant à ce même marquage.

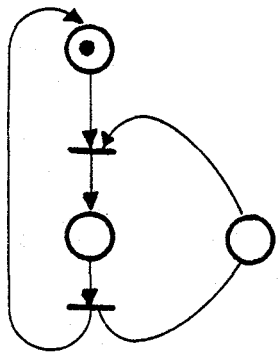
En effet, si Y est une séquence de tir cyclique, de degré $\overline{Y}$, alors nous avons :

$$M = M + C^T \overline{Y} = 0$$

d'où $C^T \overline{Y} = 0 \iff \overline{Y} = I_t$ est un invariant de transition.

Cependant, la réciproque n'est pas toujours vérifiée, comme le montre l'exemple ci-après.

Exemple :



$$C = \begin{array}{c} \begin{matrix} P_1 & P_2 & P_3 \end{matrix} \\ \begin{bmatrix} -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} \end{array} \begin{matrix} t_1 \\ t_2 \end{matrix}$$

$$M_o^T = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

Figure III.35

La résolution de $C^T Y = 0$ permet d'obtenir la solution $I_t = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Or, cette solution ne définit pas une séquence de tir cyclique, tirable depuis M_o (le réseau est bloqué, aucune transition n'est sensibilisée).

Il est donc nécessaire de vérifier que les séquences de tir définies à partir des solutions de $C^T Y = 0$ soient tirables.

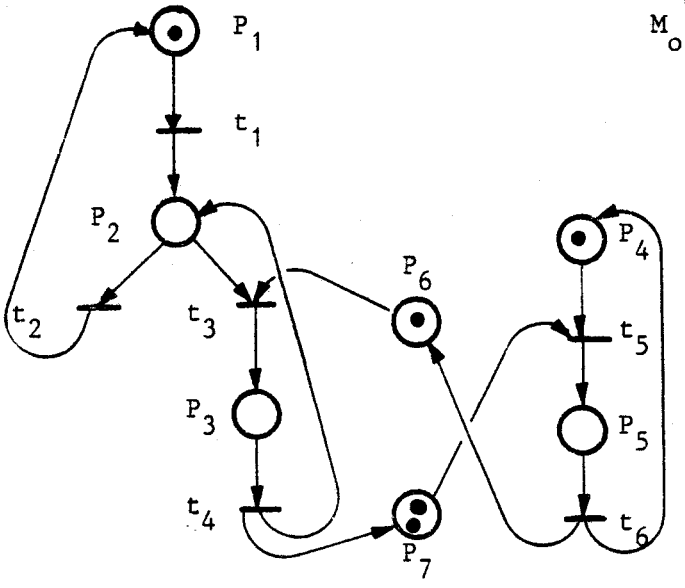
L'existence de solution du système (7) fournit ainsi des conditions nécessaires permettant de prouver que le réseau dispose de séquences de tir cycliques.

Comme pour les invariants de place, on définit le support d'un invariant de transition I_t de la façon présentée ci-dessous.

$$SUPT = \left\{ t \in T \ / \ I_t(t) \neq 0 \right\}$$

De même, un invariant I_t est dit total si son support est égal à l'ensemble T des transitions du réseau.

Exemple :



$$M_o^T = [1\ 0\ 0\ 1\ 0\ 1\ 2]$$

Figure III.36

La résolution du système (7) donne les invariants suivants :

$$(1)\quad I_{t_1}^{1T} = [1\ 1\ 0\ 0\ 0\ 0] \qquad SUPT_1^1 = \{t_1\ ,\ t_2\}$$

$$(2)\quad I_{t_2}^{2T} = [0\ 0\ 1\ 1\ 2\ 2] \qquad SUPT^2 = \{t_3\ ,\ t_4\ ,\ t_5\ ,\ t_6\}$$

qui correspondent respectivement aux séquences de déclenchements cycliques suivants :

$$\sigma_1 = \delta t_1 . \delta t_2$$

$$\sigma_2 = \delta t_3 . \delta t_4 . \delta t_5 . \delta t_6$$

Remarque : Il est alors nécessaire que ces séquences de tir soient tirables pour le marquage initial M_o.

Ces deux invariants définissent l'invariant total :

$$I_t^{\ total} = [1\ 1\ 1\ 1\ 2\ 2]$$

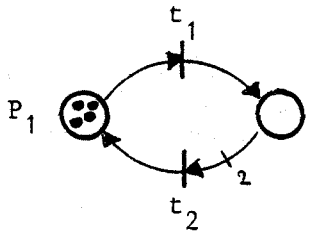
Définition : Un réseau de Pétri qui possède un invariant total de transition est dit consistant.

III.3.2.3 - Propriétés :

Nous présentons ici les principales incidences sur le comportement interne du modèle (le réseau de Pétri) qu'apportent les propriétés structurelles que sont l'invariance et la consistance. Ces relations ont fait l'objet de nombreuses études

(1) Un réseau de Pétri invariant est borné pour tout marquage initial (la preuve a été présentée au § III.2.2.1).

Cette propriété fournit une condition suffisante, l'exemple suivant montre que la réciproque n'est pas toujours vraie.



$$C = \begin{array}{c} \begin{matrix} P_1 & P_2 \end{matrix} \\ \begin{bmatrix} -1 & 1 \\ 1 & -2 \end{bmatrix} \end{array} \begin{matrix} t_1 \\ t_2 \end{matrix}$$

Figure III.37

Ce réseau n'est pas invariant, il est néanmoins borné.

(2) Un réseau de Pétri est consistant s'il admet pour un marquage initial depuis lequel une séquence de tir cyclique tirant au moins une fois chaque transition du réseau est déclenchable.

(3) Un réseau de Pétri borné et consistant est invariant.

(4) Un réseau de Pétri borné et vivant est invariant et consistant.

(5) Un réseau de Pétri vivant et réinitialisable est consistant.

Ces propriétés qui fournissent des conditions qui sont ou nécessaires ou suffisantes permettent néanmoins de conclure dans les cas suivants |6| :

- Si le réseau est non invariant ou non consistant, alors il est non vivant ou non borné (ou logique).

- S'il est non invariant, il est non borné ou non consistant.

- Si le réseau est non consistant, il est non vivant ou non réinitialisable.

III.3.2.4 - Méthode de vérification :

Nous avons pu étudier dans le paragraphe précédent, les conditions de non existence d'invariant total de place ou de transition impliquant de ce fait la non conformité du modèle. La validation du modèle nécessite donc dans un premier temps, la recherche de ces invariants.

Cependant, l'intérêt de cette analyse ne tient pas uniquement à la détermination des propriétés intrinsèques d'un réseau. Elle permet également la vérification de propriétés spécifiques du système modéllisé.

Exemple :

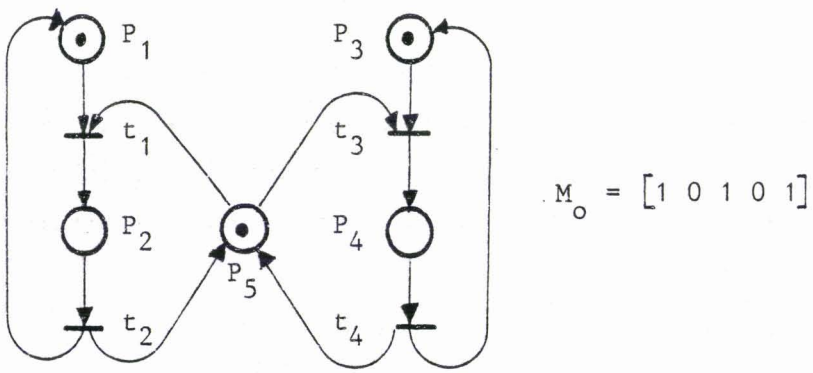


Figure III.38

La détermination de l'invariant $I_p^T = [0\ 1\ 0\ 1\ 1]$ permet de vérifier les contraintes d'exclusion mutuelle entre les places p_2 et p_4 du réseau. En effet, cet invariant de place permet d'écrire l'invariant de marquage suivant :

$$M(p_2) + M(p_4) + M(p_5) = 1$$

ce qui prouve qu'une et une seule place est marquée quelquesoit $M \in \mathcal{C}(M_o)$.

Le principe de validation que nous avons adopté s'apparente aux méthodes de vérification par assertion développées pour les programmes séquentiels |10| et parallèles |22|. La notion d'assertion apparaît également ici.

La démarche peut ainsi être décrite de la façon suivante |6| :

Etape 1 : Exprimer les spécifications du système à réaliser sous forme d'assertion.

Etape 2 : Traduire ces assertions en invariant de marquage de place, de transition sur le réseau qui modèlise le système.

Etape 3 : Rechercher les invariants réels du modèle.

Etape 4 : Vérifier la concordance entre les invariants obtenus à l'étape 2 et à l'étape 3.

Remarque : La traduction des assertions en invariant pose souvent un délicat problème et apparaît comme une limitation de cette méthode. Il n'est en effet pas toujours possible d'effectuer cette transformation et la validation de la sémantique du réseau peut ne pas être totale.

Cependant, la prise en compte de cette contrainte au niveau de l'écriture du modèle permet de faciliter cette phase.

De plus, cette méthode permet la vérification des aspects principaux rencontrés dans l'analyse des systèmes de commande parallèle, à savoir :

- respect des contraintes d'exclusion mutuelle,
- absence de situation de blocage,
- respect des capacités en jetons des places (dans la mesure où ces jetons symbolisent une ressource-objet).

L'étape 3 nécessite l'extraction de tous les invariants du réseau et plus particulièrement ceux dont les supports satisfont à un certain nombre de contraintes exprimées par l'utilisateur. En particulier, dans le cadre des réseaux de Pétri structurés, les contraintes à vérifier reposent essentiellement sur le graphe de liaison (contrainte d'exclusion mutuelle, respect de capacité dans une liaison de producteur/consommateur, ...). Il est intéressant de rechercher les invariants dont les supports contiennent les places ou les transitions de ce graphe de liaison.

Remarque : Ce point de vue justifie en quelque sorte le choix que nous avons fait dans le chapitre des réductions, c'est-à-dire de scinder en deux parties le processus de réduction du réseau. La première partie permettait de conserver la structure du graphe de liaison, les sections critiques associées aux ressources symbolisées par les jetons des places de liaison.

L'utilisation des réseaux de Pétri structurés permet ainsi une recherche systématique des invariants utiles dans la mesure où les places de liaison sont déclarées comme telles.

Cette recherche utilise un logiciel de programmation linéaire en nombre entier présenté plus loin.

III.3.2.5 - Simplification apportée pour l'utilisation de réseaux de Pétri structurés :

Nous avons montré qu'un processus décrit par un réseau de Pétri structuré était :

- sauf
- vivant
- propre.

L'analyse structurelle nous montre que le caractère sauf est conservé pour chaque processus d'un système de processus interconnecté par des liaisons de longueur 2. Il est en effet toujours possible de déterminer un invariant élémentaire de support contenant toutes les places d'un processus.

Cette remarque est importante dans le sens où elle permet de simplifier la procédure d'analyse structurelle. La recherche de tous les invariants élémentaires de place n'est plus nécessaire, car seuls désormais sont significatifs ceux dont les supports contiennent les places de liaison.

Exemple :

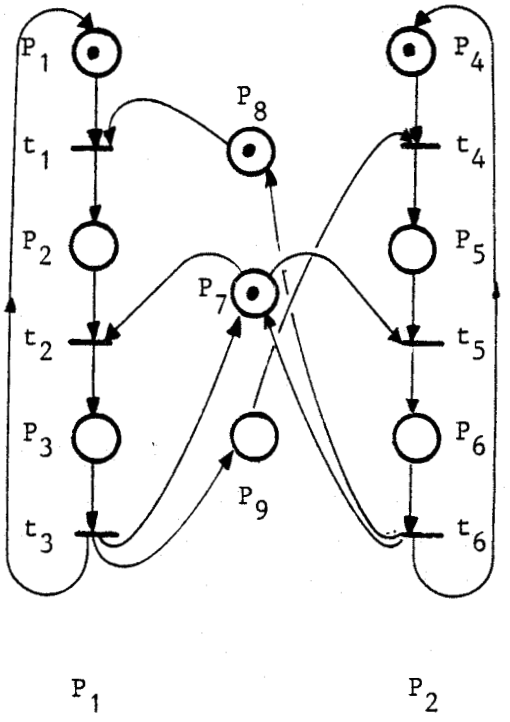


Figure III.39

On trouve pour ce réseau les invariants :

$$I_p^{\,1} = [1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0]^T$$
$$I_p^{\,2} = [0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0]^T$$

qui définissent respectivement les composantes conservatrices relatives au processus P_1 et au processus P_2 et qui prouvent que les places de chaque processus sont bornées (ce que l'on avait déjà démontré).

L'introduction de liaison n'influe pas sur le caractère borné des places des processus, seul le caractère vivant de leur transition peut être affecté. L'obtention de ces deux invariants ne nous fournit donc pas de nouvelles informations.

Nous nous intéresserons donc désormais exclusivement à la recherche d'invariants élémentaires dont les supports contiennent des places de liaisons et des sections critiques associées aux ressources.

Ainsi, nous chercherons plus particulièrement :

1) l'invariant I_p^3 associé à la place d'exclusion mutuelle P_7 du réseau de la figure III.39.

$$I_p^3 = [0\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0]^T$$

Cet invariant permet d'écrire l'invariant de marquage suivant :

$$M(p_3) + M(p_6) + M(p_7) = 1 \qquad \forall M \in \mathcal{C}(M_o)$$

qui prouve que la contrainte d'exclusion mutuelle entre la place P_3 et P_6 est respectée et que la place d'exclusion mutuelle est bornée à 1.

2) l'invariant I_p^4 associé aux places de la liaison de producteur/consommateur P_8 et P_9.

$$I_p^4 = [0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1]^T$$

Ce dernier permet d'écrire la relation invariante ci-dessous :

$$M(p_2) + M(p_3) + M(p_5) + M(p_6) + M(p_7) + M(p_8) = 2$$

Les places p_2 à p_6 sont des places de processus, elles sont donc saines.

La relation ci-dessus permet alors d'affirmer que les places p_7 et p_8 sont bornées à 2, chiffre qui correspond au nombre total de ressources dans l'un ou l'autre état.

III.3.2.6 - Mise en œuvre :

* Mise en forme du problème :

L'analyse structurelle nécessite la recherche

- d'invariants de places
- d'invariants de transition.

Ceux-ci sont obtenus en résolvant un système d'équations linéaires de forme :

$$A\,X = [0] \qquad (1)$$

où A est une matrice de dimension (n , m) ou (m , n) suivant le type de problème à résoudre (rappelons que n = |P| et m = |T|), et X un vecteur colonne de dimension n ou m, dont au moins une composante est non nulle (afin d'éliminer la solution triviale X = [0]).

L'ensemble des solutions de (1) pouvant être infini ou vide, il est intéressant d'en déterminer la solution minimale.

Nous avons mis en évidence également dans le paragraphe III.3 qu'il était intéressant pour une analyse des propriétés spécifiques d'un système modélisé par un réseau de Pétri structuré de déterminer des invariants dont les supports contiennent certaines places ou transitions particulières. Nous noterons S, l'ensemble des indices de ces sommets particuliers.

Le problème général de la recherche d'invariant peut donc se poser de la manière suivante :

Déterminer X où $X = (x_i)$ $\quad i = 1$ à n
tel que :

$$I \quad \begin{cases} A\,X = 0 & A = ((a_{i,j})) \quad i = 1 \text{ à } n \; ; \; j = 1 \text{ à } m \\ \forall\, s \in S & x_s \geq 1 \\ \forall\, i \; ; & x_i \in \mathbb{N} \end{cases}$$

et qui minimise la fonction $z = \sum_i x_i$.

Nous nous intéressons dans la suite, à la recherche d'invariants de place. Pour déterminer les invariants de transition, il suffit de remplacer P par T, et la matrice d'incidence C par sa matrice transposée C^T.

Exemple 1 : Invariant total.

Le problème est alors le suivant :

Déterminer X avec $X = (x_i)$ $i = 1$ à n $n = |P|$ tel que

$$C\,X = 0$$

$$\forall\, i \in \{1, \ldots, n\} \quad x_i \geq 1 \; ; \; x_i \in \mathbb{N}$$

qui minimise $z = \sum_{i=1}^{n} x_i$.

Exemple 2 : Invariant dont le support contient une place d'indice s (par exemple une place d'exclusion-mutuelle).

Déterminer X tel que $C\,X = 0$ $x_s \geq 1$

qui minimise $z = \sum_{i=1}^{n} x_i$.

L'utilisation de programme linéaire en nombres entiers permet de résoudre ces différents problèmes.

* Méthodes de programmation linéaire en nombre entier |26| |15| :

- Formulation d'un programme linéaire en variables entières :

La forme standard d'un programme linéaire en variables entières s'énonce de la façon suivante :

II { Déterminer X tel que f X est minimal sous les contraintes
   { A X ≦ B et $X \in \mathbb{N}^n$ (contrainte d'intégrité)

avec X : vecteur colonne d'entiers naturels à n composantes
f : vecteur ligne à n composantes
A : une matrice à m lignes et n colonnes
B : un vecteur à m composantes (vecteur colonne).

- Choix d'une méthode :

A la différence de la méthode classique de programmation linéaire en variables continues, il existe pour les problèmes en variables entières, autant de méthodes que de problèmes à traiter. Ces méthodes peuvent être regroupées dans les deux classes suivantes :

- les méthodes exploiratoires
- les méthodes à base simpliciale.

Les méthodes exploiratoires : les techniques employées sont diverses :

- les procédures booléennes
- les procédures combinatoires arborescentes (méthodes du type branch and bound, ...).

Ces méthodes utilisent des critères d'investigation heuristiques qui sont souvent dépendant du cas de programme traité. De plus, la diversité et le nombre de ces méthodes témoignent d'un champ d'application restreint de celles-ci et d'autre part rendent difficile toute comparaison.

Les méthodes à base simpliciale présentent essentiellement l'avantage de pouvoir traiter la plupart des problèmes de programmation linéaire en nombre entier.

Parmi ces méthodes, nous retiendrons essentiellement celles développées par GOMORY qui, à partir de l'algorithme de Danzig et de ses résultats en variables continues permet d'obtenir une solution entière en procédant par troncature du polyèdre des solutions réalisables.

Le principe de ces algorithmes est le suivant : à partir d'un programme optimal, pour lequel le polyèdre primitif des solutions réalisables est obtenu, un nombre fini de troncatures de ce domaine conduit à l'ob-

tention d'une solution entière ou à l'affirmation de sa non-existence. Chacune de ces troncatures réduit le polyèdre en conservant tous les points à coordonnées entières. Les réductions successives de ce polyèdre se font en ajoutant au programme des contraintes additionnellles (contraintes de Gomory) dont les coefficients sont issus de ceux de l'une des contraintes du programme linéaire par une congruence module. Chacune de ces contraintes introduit une variable d'écart qui transforme le programme initialement optimal en un programme dual réalisable. L'application de l'algorithme dual simplicial permet de rendre alors à nouveau la programme optimal.

Si les contraintes d'intégrité des solutions obtenues ne sont pas respectées, une nouvelle troncature du domaine de solutions réalisables est effectuée jusqu'à l'obtention d'une solution entière.

Trois algorithmes ont, sur ce principe, été proposés |15|.

- l'algorithme mixte pour lequel toutes les variables ne sont pas astreintes à être entières

- le premier algorithme avec toutes les variables assujetties à être entières

- le second algorithme de Gomory pour lequel les calculs sont effectués en fixe.

Pour remédier au défaut que présente le premier algorithme, c'est-à-dire la transformation de solution en nombre rationnel en nombre entier, |15| propose une variante pour laquelle chaque solution est décomposée en un quotient et un reste. Cette remarque permet ainsi d'effectuer les calculs en fixe, pour cet algorithme.

- Mise en œuvre : Afin de résoudre le programme I, il faut auparavant le transformer afin de le mettre sous la forme du programme II

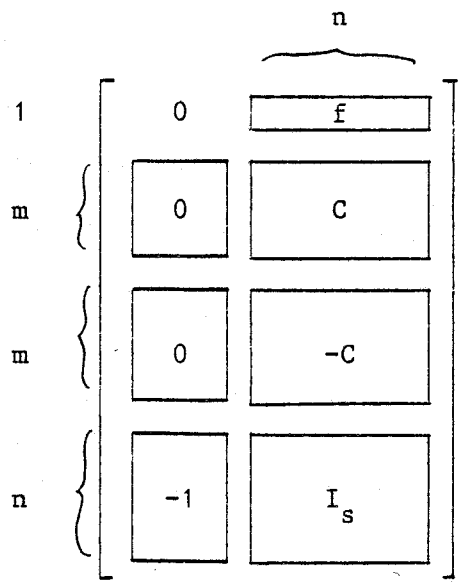
Ceci est réalisé de la façon suivante :

$$\begin{cases} C\,X \leqq 0 \\ -C\,X \leqq 0 \\ I^s\,X \leqq -1 \\ X \in \mathbb{N}^n \\ \min f\,X \end{cases} \longrightarrow \begin{cases} A\,X \leqq B \\ \\ \\ X \in \mathbb{N}^n \\ \min f\,X \end{cases}$$

avec une matrice carrée n × n dont seule la diagonale est non nulle, et qui est telle que :

si $s \in S$ alors $I^s(s,s) = -1$; sinon 0.

Le tableau simplicial aura alors la forme suivante :



La taille maximum de ce tableau sera donc de (2m+n+1) × (n+1). On constate alors que la mise en œuvre d'une analyse structurelle ne peut être efficace que sur des réseaux de petite taille et donc sur des réseaux réduits.

CONCLUSION

Nous avons présenté et analysé dans ce chapitre, les différentes méthodes permettant de valider un modèle décrit par réseau de Pétri.

Cependant, cette validation n'est que partielle. En effet, elle repose essentiellement sur la structure du modèle : le réseau de Pétri autonome, c'est-à-dire non interprété, même si dans une certaine mesure, il est possible d'attacher aux places et aux séquences de tir une signification dont il est possible de vérifier les caractéristiques sur le modèle.

Néanmoins, l'analyse des réseaux ne doit pas être négligée, car elle permet de détecter la plupart des erreurs de conception par des méthodes d'analyse structurelle, notamment :

- la non finitude des états

- les blocages du système

- les sous-ensembles du système non accessibles, par les méthodes classiques ou structurelles et également

- le non respect de contraintes d'exclusion mutuelle, de signalisation ou de producteur-consommateur.

L'utilisation de réseaux de Pétri structurés, pour la définition du modèle d'un système dans le cadre d'une méthodologie sûre de spécification et de conception est intéressante à plusieurs titres :

- elle permet dès la définition du modèle, d'éliminer la plupart des erreurs de conception (la construction du modèle obéit à des règles bien cohérentes),

- elle apporte également de nombreuses simplifications par rapport au cas des réseaux de Pétri ordinaires, dans la mise en œuvre des méthodes d'analyse (leur structure est particulièrement mise à profit dans les méthodes visant à réduire la taille du réseau).

BIBLIOGRAPHIE DU CHAPITRE III

|1| BABICH A.F.
"Proving total correctness of parallel programs"
IEEE Trans. on S.E., Vol. SE-5, n° 6, Nov. 1979.

|2| BERGE C.
"Graphe et hypergraphe"
DUNOD, 1970.

|3| BERTHELOT B.
"Vérification des réseaux de Pétri"
Thèse de Dr 3ème Cycle, Paris, 1978.

|4| BERTHELOT G. ROUCAIROL G., VALK R.
"Reductions of nets and parallel programs"
Proc. of the advanced course on general net theory of processes and Systems, Hamburg, 1979, Springer-verlag, 1980.

|5| BERTHOMIEU B.
"Analyse structurelle des réseaux de Pétri. Méthodes et outils"
Thèse de Dr Ing., Toulouse, 1979.

|6| CHEZAVIEL-PRADIN B.
"Un outil graphique intéractif pour la vérification des systèmes à évolution parallèle décrits par réseaux de Pétri"
Thèse de Dr Ing., Toulouse, 1979.

|7| CORBEEL D., VERCAUTER C., GENTINA J.C.
"Formal description of processes'systems and exception handling"
IASTED Inter. Symp. Modelling, Identification and Control, Davos, Fév. 1981.

|8| CORBEEL D., VERCAUTER C., GENTINA J.C.
"Specifications and conception of real-time control systems"
IASTED Inter; Symp. Measurement and Control, Le Caire, Sept. 1981.

|9| DERNIAME J.C., PAIR C.
"Problèmes de cheminement dans les graphes"
Monographie d'informatique AFCET, n° 8, Dunod, 1971.

|10| FLOYD
"Assigning meaning to programs"
Proc. Symp. in Applied Mathematics 19, American Math. Soc., 1967.

|11| HACK M.
"Decision problem for Petri nets and vector addition systems"
MIT Project MAC Tech. Memo. 59, 1975.

|12| HOLT A., COMMONER F.
"Events and conditˆons"
Record of the Project Mac Conference on concurrent systems and Parallel Computation, June 1970.

|13| KARP R.M., MILLER R.E.
"Parallel Program Schemata"
Journal of Computer and System Sciences, Vol. 3, 1969.

|14| KELLER R.M.
"Formal verification of parallel programs"
Communication of the ACM, Vol. 19, n° 7, July 1976.

|15| KORTAS M.
"Programmation linéaire en nombres entiers : comparaison de méthodes"
Thèse de Docteur de Spécialité, Lille, 1971.

|16| LAUTENBACH K., SCHMID H.A.
"Use of Petri nets for proving correctness of concurrent process systems"
Proc. of IFIP Congress 74, North-Holland Publ. Co., Amsterdam.

|17| LIEN Y.E.
"Termination properties of generalized Petri nets"
SIAM Journal of Computing, Vol. 5, n° 2, June 1976.

|18| LIPTON R.
"The reachability problem and the boundedness problem for Petri nets are exponential-space hard"
TR-62, Dept. Comp. Sc., Yale Univ., New Haven, Conn., Janv. 1976.

|19| MEMMI G.
"Fuites et semi-flots dans les réseaux de Pétri"
Thèse de Dr Ing., Paris, 1978.

|20| MEMMI G., ROUCAIROL G.
"Linear algebra in net theory"
Proc. of the advanced course on general net theory of processes and systems, Hamburg, 1979, Springer-Verlag 1980.

|21| MURATA T.
"Synthesis of decision-free concurrent systems for prescribed resources and performance"
IEEE Trans. on S.E., Vol. SE-6, n° 6, Nov. 1980.

|22| OWICKI S., GRIES D.
"An axiomatic proof technique for parallel programs"
Acta Informatica, n° 6, 1976.

|23| PETERSON J.L.
"Petri nets"
Computing Surveys, Vol. 9, n° 3, Sept. 1977.

|24| RAMCHANDANI C.
"Analysis of asynchronous concurrent systems by timed Petri nets"
PH.D Thesis, MAC-TR-20, Cambridge Mass.1974.

|25| SIFAKIS J.
"Le contrôle des systèmes asynchrones : concepts, propriétés, analyse statique"
Thèse Dr ès Sciences, Grenoble, 1979.

|26| SIMONNARD M.
"Programmation linéaire"
Tome 2, DUNOD, 1973.

|27| VALETTE R.
"Sur la description, l'analyse et la validation des systèmes"
Thèse d'Etat, Toulouse, 1976.

|28| VERCAUTER C., CORBEEL D.
"Analyse des systèmes décrits par des réseaux de Pétri structurés"
MIMI'82, Davos, Mars 1982.

CONCLUSION GENERALE

Nous avons présenté dans ce mémoire, un certain nombre d'outils d'aide à la spécification et à la conception des systèmes parallèles.

Ces outils sont les suivants :

- un modèle adapté aux problèmes de spécification des systèmes de commande parallèle, les réseaux de Pétri, que nous avons choisi en fonction de caractéristiques qui ont été proposées dans le premier chapitre et que nous avons justifié par la suite,

- les bases d'une méthodologie de spécification et de conception utilisant une classe particulière des réseaux de Pétri, les réseaux de Pétri structurés qui permettent de minimiser les erreurs de conception,

- des méthodes et outils d'analyse permettant une validation presque complète des spécifications à un niveau d'abstraction donné. L'utilisation de réseaux de Pétri structurés présente dans ce sens, l'avantage de simplifier la mise en œuvre des différentes méthodes que sont :

  - les méthodes classiques par énumération de marquages
  - les méthodes structurelles.

Les limites d'utilisation de chacun de ces outils ont été présentées. Ainsi, pour le modèle, nous avons indiqué que les fonctionnements d'exception conduisaient à des graphes illisibles nuisant ainsi à la compréhension du système, de plus les validations proposées sont partielles. Pour répondre à ces contraintes, plusieurs directions d'investigation sont possibles :

- utilisation de réseaux de Pétri de puissance de description supérieure comme les réseaux transitions prédicats, et qui seront transformés en réseaux ordinaires (ou généralisés) afin de pouvoir être analysés,

- utilisation des méthodes de vérification proches de celles utilisées en programmation séquentielle pour prouver totalement la conformité du système (les propriétés relatives au fonctionnement parallèle étant vérifiées à l'aide des méthodes qui ont été présentées au § III.2).

- utilisation de méthodes de simulation en vue de valider le modèle et d'évaluer les performances du système.