N° d'ordre : 782

50376
1979
209

THESE

présentée à

L'UNIVERSITÉ DES SCIENCES ET TECHNIQUES DE LILLE

pour obtenir le grade de

DOCTEUR DE SPECIALITÉ
(Informatique)

par

Didier CORBEEL

SCHEMA DE CABLAGE ET SCHEMA DE CONTROLE.
Application à la simulation et à la gestion de processus industriels

Soutenue le 12 octobre 1979, devant la Commission d'Examen

Membres du Jury :

MM.	V. CORDONNIER	Président
	G. JACOB	Rapporteur
	A. ARNOLD	Examinateur
	M. FLIESS	Examinateur
	J. C. GENTINA	Examinateur
	G. ROUCAIROL	Examinateur
	M. MORIAMEZ	Invité

A ma femme

A mes parents

A mon fils

Je tiens à témoigner toute ma gratitude à Monsieur le Professeur CORDONNIER qui me fait l'honneur d'accepter la présidence du jury de cette thèse.

Je tiens à exprimer à Monsieur le Professeur JACOB ma profonde reconnaissance pour avoir assumé la direction de cette recherche et les nombreux conseils qu'il m'a prodigués.

Je remercie très vivement

Monsieur le Professeur ARNOLD qui n'a jamais cessé de s'intéresser à mon travail en y apportant des critiques constructives ; je lui exprime ma reconnaissance d'avoir été à l'origine de la première partie de cette étude,

Monsieur le Professeur GENTINA qui, par sa compétence et l'intérêt qu'il a toujours manifesté pour le Contrôle de Processus industriels et la simulation, a été pour moi un soutien constant,

Messieurs les Professeurs FLIESS et ROUCAIROL qui ont accepté de juger ce travail, pour avoir bien voulu s'y intéresser et pour leurs remarques bienveillantes,

Monsieur le Professeur MORIAMEZ, directeur de l'IDN, pour ses encouragements et l'intérêt qu'il a manifesté pour cette étude.

Je tiens à remercier également :

mes collègues de l'équipe d'Informatique Théorique de Lille, et plus particulièrement Michel LATTEUX, pour les discussions et les conseils qui ont jalonné ce travail,

les membres du laboratoire d'Informatique Industrielle de l'IDN pour les échanges de vues féconds que nous avons eus et les diverses réalisations effectuées.

Mademoiselle FIEVET, Mesdames CARON, COPIN et CRETON, qui toutes quatre, avec gentillesse, compétence et efficacité ont assuré dans des délais rapides la réalisation matérielle de cet ouvrage,

Monsieur SOYEZ, ainsi que Madame et Monsieur DEBOCK pour l'impression et la réalisation de ce document.

INTRODUCTION GENERALE

L'étude des systèmes complexes ne peut généralement être totalement apprehendée sur le plan théorique et nécessite de ce fait le recours aux méthodes expérimentales. La simulation peut-être considérée comme un outil de base permettant de résoudre ce type de problème.

Le degré de complexité d'un système apparaît essentiellement au niveau de ses fonctions de base ou découle tout simplement de ses caractéristiques de taille et de structure. Il convient toutefois de préciser qu'une simulation ne peut-être envisagée en toute rigueur que dans l'hypothèse où il est possible de définir un modèle de référence proche du processus. Les lois de la similitude jouent donc un rôle fondamental en simulation.

Les moyens techniques concernant la simulation des systèmes peuvent être classés différemment selon d'une part la nature des informations traitées et d'autre part la nature des modèles utilisables.

Si l'on dispose d'un modèle à caractére mathématique on peut être conduit à mettre en oeuvre des simulations électroniques, nous en distingueron essentiellement deux classes.

L'ordinateur a tout d'abord été considéré comme l'outil le plus puissant, il a pleinement justifié ce point de vue par ses nombreuses applications au traitement des informations discrétes, booléennes ou codées (numériques). Toutefois sa mise en oeuvre devient délicate lorsqu'il s'agit de traiter une information continue par nature. Il est alors préférable de recourir à l'utilisation d'une autre classe de calculateurs : les calculateurs analogiques.

Pour ce type de calculateur, l'information est traitée en continu selon un modèle parallèle et le modèle mathématique se traduit par un cablage (F, J, T).

Dans le chapitre 1, nous proposons un outil formel permettant la description d'un cablage : le schéma de cablage. Cet outil repose sur la structure algébrique du magmoïde étudié par l'équipe Lilloise d'Informatique théorique (A, AD1, AD2, D, J,).

La théorie des schémas de programmes (I, L, G, E, M, D^1, L1) permet l'étude des propriétés structurelles et des transformations d'un programme. Nous montrons, dans la partie 2 du chapitre 1, que le schéma de cablage permet la même étude des structures et des transformations d'un cablage.

Une étude des systémes continus, discrets ou séquentiels fait apparaître une certaine similitude de propriétés
Le schéma de cablage offre une représentation unifiée de ces systèmes, et l'interprétation d'un schéma de cablage précise l'espace de définition et la signification des divers opérateurs.

L'évolution des calculateurs analogiques s'est effectuée en deux étapes. Dans une première phase, l'adjonction aux opérateurs analogiques, d'opérateurs booléens et hybrides a permis d'accroître le champ d'utilisation de ces matériels.

Pour pouvoir tenir compte de ce nouveau type de cablage, nous avons été amené à étendre la structure de magmoïde à celle du magmoïde typé. Dans cette structure, il est possible de définir le type des entrées et des sorties d'un opérateur. Cette extension du magmoïde permet de représenter par un schéma de cablages, un modèle de simulation hybride (E1, B1, L2). Nous proposons en annexe, quelques exemples de telles représentations.

La seconde évolution a constitué à coupler un calculateur à opérateurs logiques et analogiques cablés et un calculateur digital. L'ensemble de traitement ainsi obtenu, appelé calculatrice hybride de type II, présente donc deux avantages principaux. En effet, aux instructions à exécution séquentielle des calculateurs numériques il associé la possibilité d'opérations spatiales selon un mode parallèle. De plus, la possibilité de traitements simultanés accroît la vitesse d'exécution des calculs.

Désormais, au modèle simulé sur la calculatrice hybride de type II, doit être associée une logique de commande pilotant l'enchaînement des tâches, synchronisant des calculs où des tâches complexes s'effectuent en parallèle.

D'une manière tout à fait analogue, dans la gestion de processus industriels, apparaît également une logique de commande chargée d'activer des tâches, de les synchroniser, de répartir les ressources, il nous a donc semblé intéressant d'aborder selon la même démarche l'étude de la logique de commande d'un processus industriel et la simulation sur calculatrice hybride de type II.

L'étude des problèmes liés à l'exécution parallèle d'un ensemble d'opérateurs présente un intérêt particulier dans le pilotage de processus industriels de grandes dimensions. Dans ce contexte, le parallélisme engendre une multitude d'états qui sont souvent difficiles à relier entre eux. Par conséquent, la description et l'analyse des phénomènes parallèles requièrent l'utilisation de modèles permettant de représenter à la fois les différents états possibles et les régles de transition entre ces états. De plus, il doit également être possible de vérifier des propriétés de comportement dynamique.

Parmi les modèles existants, la mise en oeuvre des réseaux de Petri basée sur une représentation graphique est actuellement l'un des moyens les plus efficaces qui permet concrétement de traduire l'évolution des contraintes de précédence entre les actions à réaliser et les états qui résultent de leurs exécutions. Leur simplicité d'emploi les rend très accessibles aux domaines qui relèvent de l'informatique et de l'automatique. Il convient cependant de rappeler l'existence de formatismes équivalents, toutefois moins commodes, "Vector Addition Systems" de Karp et Miller (K) "UCLA graphs" de Gostelow (G3).

Le chapitre 2 débute par quelques brefs rappels sur les réseaux de Petri.

Cette étude a essentiellement pour objectif de faciliter à la fois la description de la logique de commande d'une gestion de processus et d'en effectuer la synthèse. Elle porte, donc en premier lieu, sur la formalisation de la notion de processus, dont nous déduirons le schéma de processus libre de toute interprétation (chapitre 2-1). D'une manière informelle le schéma de processus correspond à un schéma de programme, ce dernier étant chargé de piloter une tâche ou un "processus industriel".

La logique de commande d'une tâche ne peut-être quelconque, elle doit offrir à l'utilisateur certaines sécurités : inexistance d'état de blocage (vivant) et possibilité à tout instant de réinitialisation (propre). Le schéma de processus devra donc, lui aussi, posséder ces deux propriétés.

La logique de commande d'un processus pilote, le plus souvent, plusieurs tâches simultanément, à chaque tâche étant associé un schéma de processus muni d'une interprétation. La seconde partie du chapitre 2 nous amène ainsi à présenter une classification et une étude des diverses liaisons existant entre schéma de processus. Nous définissons ainsi trois classes principales de liaisons : synchronisation et partage de ressource à une ou deux états.

La dernière partie du chapitre 2, porte sur la détermination des conditions définissant l'absence de blocage d'un système de processus. Bien que les propriétés des réseaux de Petri traduisant le bon comportement d'un système parallèle aient été établies, leurs vérifications se révèlent du fait de la complexité inhérente au parallélisme, assez difficile dans le cas général (L3 R, C1). Notre démarche a consisté à étudier les diverses liaisons indépendamment, et à offrir à l'utilisateur une méthodologie de conception des systèmes de schéma de processus très proche de celle qui découle de l'utilisation de la programmation structurée.

La description d'une logique de commande à l'aide de schémas de processus structurés permet d'éviter de manière systématique de nombreux cas de blocages.

Le chapitre 3 est consacré à la conception et à la mise en oeuvre d'un système de logique de commande. Cette étude porte sur la réalisation matérielle et logicielle d'un système de contrôle numérique et présente deux approches, l'une utilisant un calculateur industriel, l'autre des circuits microprocesseur. Chaque réalisation fait l'objet d'un exemple pris dans les domaines de la simulation ou de la gestion de processus industriel.

BIBLIOGRAPHIE DE L'INTRODUCTION

(A) ARNOLD A.
"Systèmes d'équations dans le magmoïde. Ensembles rationnels et algébriques d'arbres". Thèse es Sc. Math. Lille I, 1977.

(AD1) ARNOLD A. et DAUCHET M.
"Théorie des magmoïdes ; Introductions communes aux thèses es Sc. Math. d'Arnold et de Dauchet". Lille 1, 1977.

(AD2) ARNOLD A. et DAUCHET M.
"Théorie des magmoïdes ; Colloque de Lille : *Les arbres en algèbre et en programmation"*, 1976.

(B) BORNE P., GENTINA J.C., LAURENT F., TOULOTTE J.M.
"Extension à un corps quelconque des méthodes de simulations usuelles"
7ème AICA Congrès, Prague, Août 1973.

(B1) BEKEY G.A., and KARPLUS W.J.
"Hybrid Computation"
Wiley Int. Ed. 1968.

(C) CONNELY M.E.
"Real-Time Analog - Digital Computation"
IRE Transactions on Electronic Computers, Feb. A62.

(C1) COTRONIS J.Y., LAUER P.E.
"Vérification of concurrent systems of processes"
Proc. of the International Computing Symposium 1977
Liège, North-Holland. Pub. Co.

(D) DAUCHET M.
"Transductions de forêts - Bimorphismes de Magmoïdes"
Thèse d'Etat. Lille 1, 1977.

(D1) DAHL O.J., DIJKSTRA E.W., HOARE C.A.R.
"Structured Programming"
Academic Press, New-York, 1972.

(E) ENGELFRIET J.
"Simple program schemes and formal languages"
Lecture Notes in Computer Sciences (Springer), 1974.

(E1) Electronic Associates, Inc.
"Introduction to hybrid computations"
Long Branch, N.J. 1963.

(F) FIFER S.
"Analogue Computation : Theory, Techniques and Applications"
Mc Graw Hill, New-York, 1961.

(G) GREIBACH S.
"Theory of programs structures : schemes, semantics, verification"
Lecture Notes in Computer Science, 36, Springer. Berlin (1975).

(G1) GENTINA J.C.
"Sur la notion de modèle dans la description d'un processus"
Collogue ANRT. Ecole Centrale, Paris. (Juin 1975).

(G2) GENTINA J.C., BORNE P., LAURENT F., TOULOTTE J.M.
"Extension à un corps quelconque des méthodes de simulations usuelles"
7ème congrés AICA, Prague, Août 1973.

(G3) GOSTELOW K., CERF V.G.
"Proper termination of flow of control in programs in volving concurrent processes"
Sigplan Notice, 7.11.72.

(H) HACK M.
"Decision problems for Petri Nets and Vector Additions Systems"
MJT Project Mac Technical memorandum 59 (1975).

(I) IANOV Y.I.
"The logical Schemes of Algorithms"
Problems of Cybernetics - Pergamon Press, New-York, 1960.

(J) JACKSON A.S.
"Analog Computation"
Mc Graw-Hill, 1960.

(J1) JACOB G.
"Substitution dans les arbres et non déterministe. Appel par nom et appel synchrone"
2ème Colloque de Lille *"Les arbres en algèbre et en programmation"* (Fév. 1977).

(K) KARP R.M., MILLER R.E.
"Parallel program schemata"
JCSS, Vol. 3, 1969.

(L) LUCKHAM D.C., PARK D.M.R., PATERSON M.S.
"On formalized computer programs"
JCSS, Vol. 4, 1970.

(L1) LEDGARD H., MARCOTTY M.
"A genealogy of Control structures"
C ACM 18, 1975.

(L2) LAURENT F.
"Les machines hybrides et leurs applications en Automatique"
Thèse Doctorat Lille, 1968.

(L3) LAUTENBACH K., SCHMID H.A.
"Use of Petri-nets for proving correctness of concurrent process systems"
IFIP 74, North-Holland. Pub. Co. (1974).

(M) MANNA Z.
"Mathematical theory of computation"
Mc Graw Hill, New-York (1974).

(R) RAMCHANDANI C.
"Analysis of asynchronous concurrent systems by Petri-Nets"
M.I.T. Project MAC TR-120, Feb. 1974.

(T) TOMOVIC R. and KARPLUS W.I.
"High-speed analog Computers"
John Wiley, New-York, 1962.

C H A P I T R E 1

SCHEMA DE CABLAGE

SCHEMA DE CABLAGE

1.0 INTRODUCTION

Dans les concepts de simulation, modèle et système sont intrinséquement liés, la simulation étant d'une façon générale une méthode pour étudier les systèmes, à l'aide de modèles. Schématiquement, on peut dire que la simulation est une manipulation de modèle, et le modèle, une représentation de système.

La notion de système n'est pas nouvelle. Darwin, dans sa théorie de l'évolution, intégrait toute vie dans un "système de la nature". Avant lui, Newton a construit un "système du monde", fondé sur la loi de la gravitation universelle. Toute discipline particulière, et notamment la physique, a pour objet une construction explicative, dont la matière est un ensemble, arbitraire et révisable, d'éléments de témoignage sensible. De manière concrète, l'observateur porte son attention sur une réunion d'objets ou de parties d'objets ; il n'en retient que certaines qualités, celles qui l'intéressent pour l'étude qu'il se propose ; il la délimite par rapport aux objets qui l'entourent et précise par des hypothèses ses rapports avec ces objets extérieurs. Il définit ainsi ce qu'on peut appeler un système.

La notion de processus est lié au temps, un processus est un système évolutif (V), c'est-à-dire évoluant dans le temps. Les sorties du processus caractérisent les grandeurs susceptibles d'être observées ou mesurées.

Supposons qu'il soit possible de décrire un processus P au moyen d'un modèle mathématique M. Ce modèle peut présenter diverses propriétés.

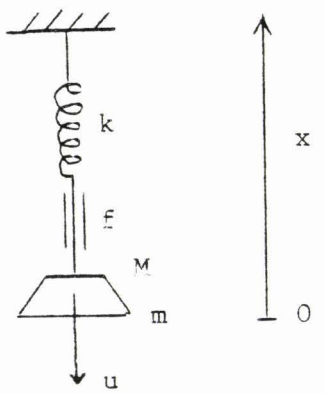
Le modèle est dit observable si la connaissance d'un nombre fini de valeurs de sorties permet d'en déduire l'état initial du processus (B).

Le modèle est dit <u>commandable</u> s'il existe une entrée telle qu'à partir de conditions initiales quelconques, il soit possible d'amener en un temps t_c, un état arbitraire (B).

A partir de ce modèle, il peut s'averrer possible de définir d'autres modèles mathématiques du processus P choisi de manière à ce qu'ils soient observables ou commandables.

Illustrons ces définitions sur un exemple tiré de la mécanique. Considérons le dispositif mécanique de la figure 1 où :

- k note le coefficient de raideur du ressort,
- f est le coefficient de frottement fluide,
- m désigne la masse du ressort,
- g l'accélération de la pesanteur,
- u la force dirigée selon Ox appliquée à la masse.
  (l'axe Ox est dirigé verticalement).



Soit x_o l'abscisse du point M à l'équilibre : $mg = k\,x_o$ (∇1)

L'équation fondamentale de la dynamique s'écrit dans ce cas :

$$m\overset{oo}{x} = -mg + k(x_o - x) - f\overset{o}{x} + u$$

soit en tenant compte de ∇1 :

$$m\overset{oo}{x} + f\overset{o}{x} + kx = u$$

Cette équation est de la forme : $\overset{oo}{x} = -(a_1 \overset{o}{x} + a_2 x) + b$ (∇2)

A ce système, nous pouvons associer les deux modèles suivants :

(i) posons $y^T = [\overset{o}{x}, x]$, l'équation ∇2 se réécrit :

$$M_1 \begin{cases} \overset{o}{y} = \begin{bmatrix} -a_1 & -a_2 \\ 1 & 0 \end{bmatrix} y + \begin{bmatrix} 1 \\ 0 \end{bmatrix} b \\ S = [0,1]^{T_1} y \end{cases}$$

(ii) l'équation ∇2 peut s'écrire :

$$x = \int(\int b^{dt} - a_2 \int x\,dt - a_1 x)dt$$

posons $y_1 = \int(b - a_2 x)dt$

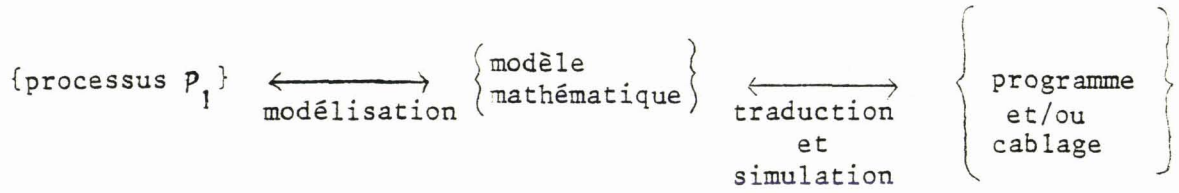
$$y_2 = x$$

nous obtenons le modèle M_2 :

$$M_2 \begin{cases} \overset{o}{y}_1 = -a_2 y_2 + b \\ \overset{o}{y}_2 = y_1 - a_1 y_2 \\ S = y_2 \end{cases}$$

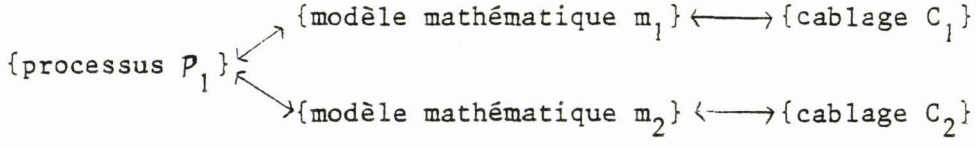
le modèle M_1 est commandable mais non observable, et le modèle M_2 est observable mais non commandable [B], ainsi nous pouvons proposer deux modèles du même processus, ayant des propriétés différentes.

Reprenons notre processus P ayant pour modèle mathématique M, si ce modèle est implémenté sur une calculatrice analogique, digitale ou hybride, ceci nous conduit à écrire un programme pour le calculateur digital ou analogique. Nous obtenons la chaîne de traitement suivante que doit suivre l'utilisateur désirant simuler un processus



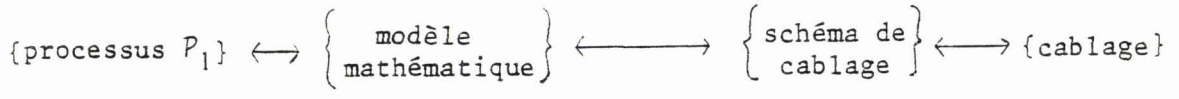
Dans la suite de ce chapitre nous supposerons que l'implantation du modèle se fasse sur un calculateur analogique ou hybride.

Nous venons de montrer que pour un même processus il peut s'averrer possible de définir plusieurs modèles mathématiques et donc obtenir plusieurs cablages.



Puisque les divers modèles peuvent posséder des propriétés différentes il peut s'averrer nécessaire de passer d'un modèle ne possédant pas une propriété à un autre modèle du même processus possédant cette propriété. Cette transformation s'effectue plus ou moins facilement sur le modèle mathématique (B), mais ne permet pas d'obtenir de manière systématique tous les modèles mathématiques du même processus.

Nous nous proposons d'introduire un intermédiaire formel entre le modèle mathématique et le cablage physique du calculateur, que nous appelerons schéma de cablage, d'où la nouvelle chaîne de traitement :



Nous montrons au moyen des propriétés combinatoires des schémas de cablage, que le passage d'un modèle mathématique à un autre, pour un même processus peut se faire facilement. (Une conséquence sera le passage d'un cablage à un autre).

Le schéma de cablage est au cablage, ce qu'est le schéma de programme à un programme ; à savoir un objet purement formel (syntaxique). Si on attribue une signification aux symboles, un schéma de programmes devient un programme, ici de même si on propose une interprétation des opérateurs et si on définit le domaine de calcul le schéma de cablage devient un cablage. Dans ce chapitre seront proposées diverses interprétations du même schéma de cablage pour des domaines de calcul tels que l'ensemble $\mathbb{N}$ ou les champs de Gallois modulo n, ce qui correspond respectivement aux simulations d'un processus ou d'une machine séquentielle.

Le dernier intérêt qu'offre le schéma de cablage est d'être une première approche de l'automatisation du cablage des calculatrices analogiques et/ou hybrides. Ce qui nous a amené à développer un algorithme de factorisation des expressions formelles et étendre la notion de magmoïde, afin de tenir compte des types des opérateurs.

1. ALGEBRE DE DECOMPOSITION - RAPPELS SUR LE MAGMOÏDE [A, AD1, AD2, AD3]

1.1 OBJET FONCTIONNEL [C_1]

Considérons une machine abstraite composée de cellules 'opérateurs' et de cellules 'mémoires'. Une cellule mémoire est définie sur un certain domaine (un ensemble E, par exemple). Par extension, nous appelerons mémoire de degrè p une séquence de p cellules mémoires, et nous la noterons par l'une des écritures suivantes, équivalentes :

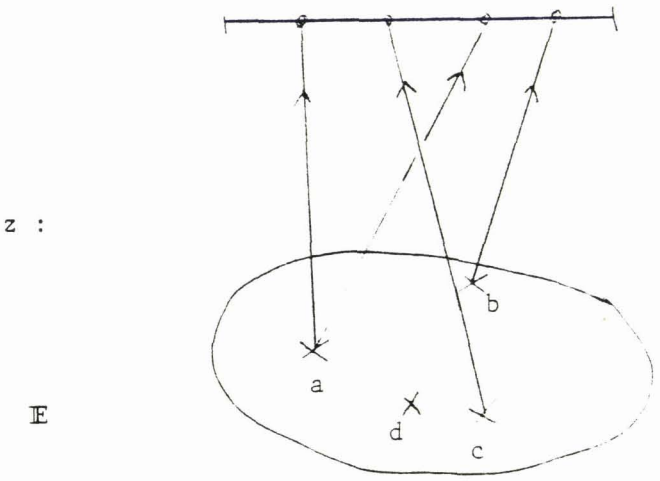
- —•—•—•— mémoire de degrè 4
- [p] l'intervalle des entiers compris entre 1 et p
- $\langle p ; x_1, x_2, \ldots, x_p \rangle$ notation en systèmes de variables

Le contenu d'une mémoire de degrè p, z sera noté $(z_1, z_2, \ldots, z_p)$, et z est une application de [p] dans $\mathbb{E}$:

$z : [p] \longrightarrow \mathbb{E}$

Cette application se nommera assignation de degrè [p].

Exemple :



$z : [4] \longrightarrow E$ et $z(1) = z_1 = a$

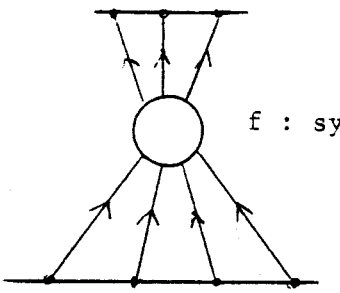
$z(2) = z_2 = c$

$z(3) = z_3 = a$

$z(4) = z_4 = b$

Les cellules opérateurs sont des symboles fonctionnels de type $\binom{q}{p}$ et une cellule opérateur f possédant p entrées et q sorties est décrite par :

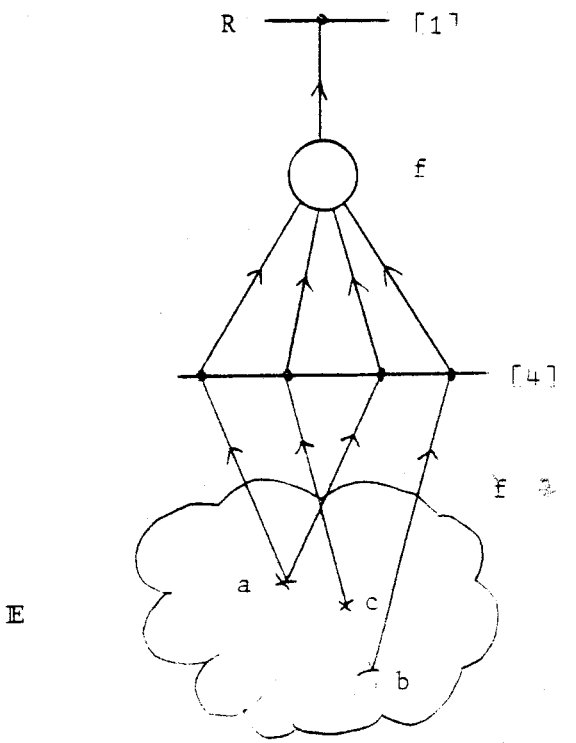
f :



f : symbole fonctionnel de type $\binom{3}{4}$.

Remarque : $f \in F_p^q$ où F_p^q représente l'ensemble des opérateurs ayant p variables d'entrées et q variables de sorties.
(F pour objet fonctionnel)

Exemple : Soit f une application de E_4 dans E, la structure fonctionnelle sera décrite comme suit :



Soit z : [4] ⟶ E alors le résultat f.z est mémorisé dans la cellule mémoire R.

1.2 ASSOCIATION D'OPERATEURS

1.2.1 PRODUIT TENSORIEL

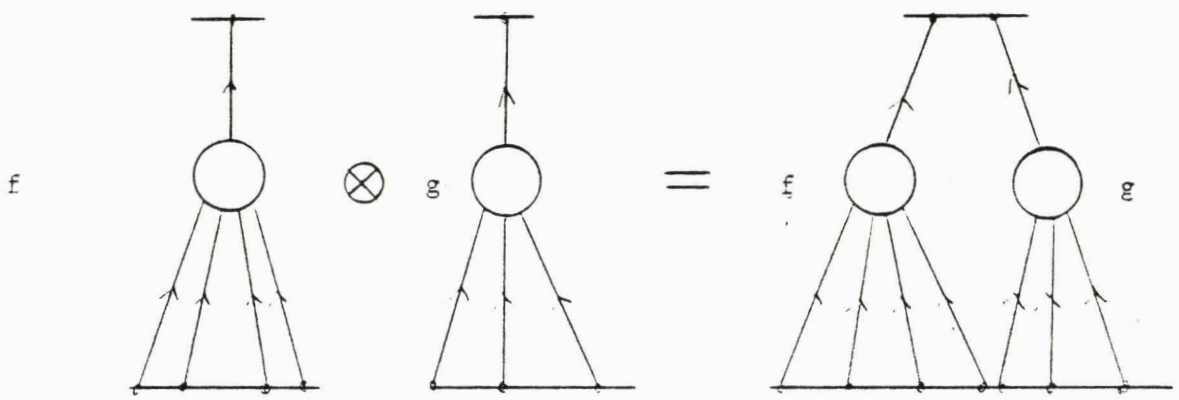
L'introduction du produit tensoriel, note ⊗, permet la composition des cellules mémoires, des cellules opérateurs et des assignations.

i Cellules mémoires

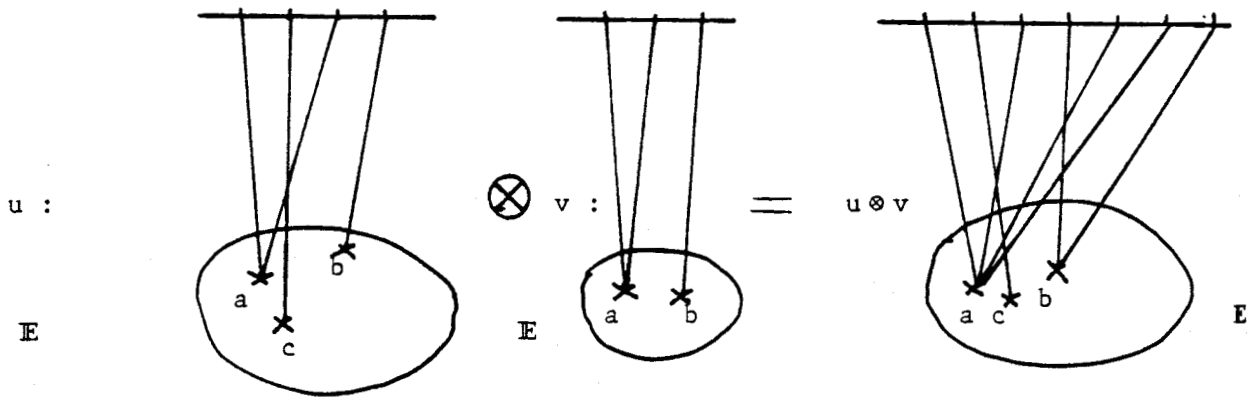
$[p] \otimes [q] = [p+q]$

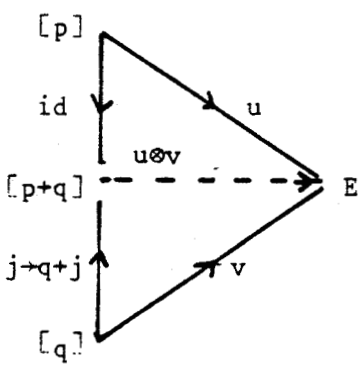
$\langle p ; x_1, x_2, \ldots, x_p \rangle \otimes \langle q ; x_1, x_2, \ldots, x_q \rangle = \langle p+q ; x_1, x_2, \ldots, x_{p+q} \rangle$

ii Cellules opérateurs



iii Assignations





L'application u ⊗ v : [p+q] ⟶ 𝔼 est définie par

(u ⊗ v) (i) = u(i) si i ∈ [p]
(u ⊗ v) (p+j) = (v(j) si j ∈ [q]

$\langle u_1, u_2, \ldots, u_p \rangle \otimes \langle v_1, v_2, \ldots, v_q \rangle$
$= \langle u_1, u_2, \ldots, u_p, v_1, v_2, \ldots v_q \rangle$

L'exemple ci-dessus s'écrit :

<a, c, a, b> ⊗ <a, a, b> = <a, c, a, b, a, a, b>

iv Composition parallèle de machines abstraites

Soient M_1 et M_2 deux machines abstraites, définies sur les domaines D_1 et D_2, nous définirons la composée parallèle M de ces deux machines sur le domaine 𝔻 par :

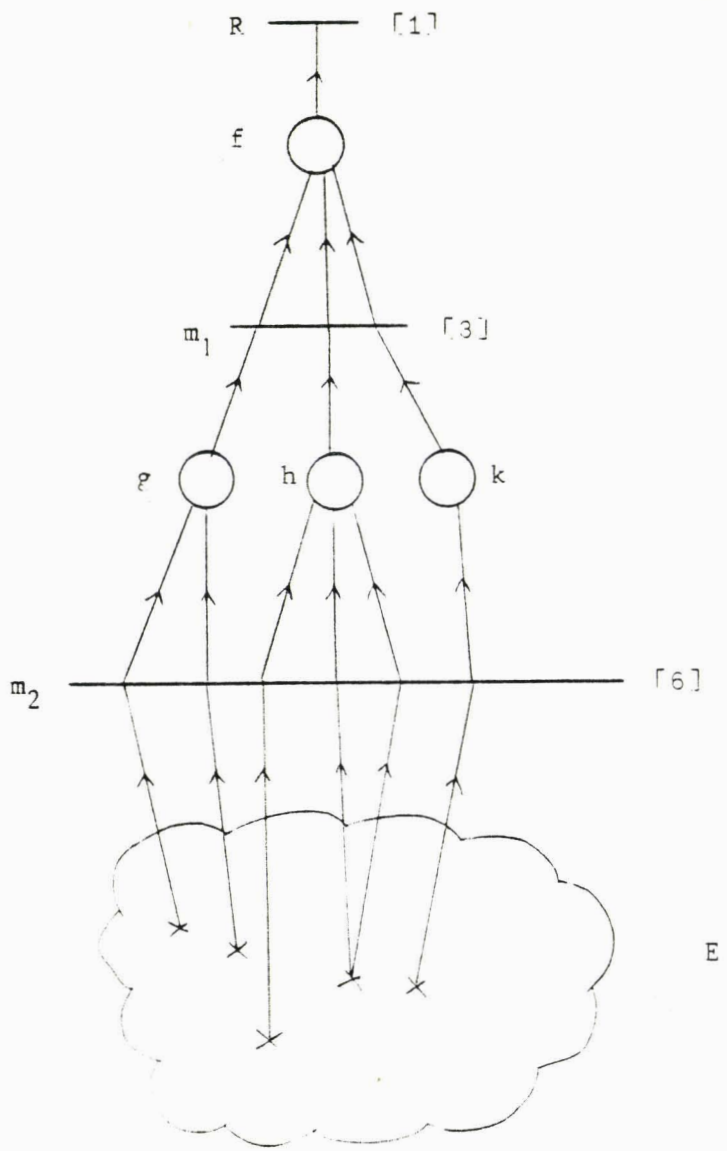
$$M = M_1 \otimes M_2$$

$$D = D_1 \times D_2$$

1.2.2 COMPOSITION DE MACHINES ABSTRAITES

La composition consiste à composer en série les machines abstraites décrites, les haut. Elle ne peut se faire que si les deux objets considérés peuvent se "souder" en identifiant la cellule mémoire de "sortie" du premier à la cellule mémoire d'entrée du second. Exemple :

Soient les 4 cellules opérateurs : $f \in F_3^1$, $g \in F_2^1$, $h \in F_3^1$ et $k \in F_1^1$. Ces opérateurs sont utilisés afin de définir la machine abstraite T suivante :



Les contenus des opérateurs mémoires m_1 et m_2 sont notés respectivement y et z :

$$y = \langle y_1, y_2, y_3 \rangle$$

$$z = \langle z_1, z_2, z_3, z_4, z_5, z_6 \rangle$$

et nous avons les relations : $y_1 = g(z_1, z_2)$

$$y_2 = h\ (z_3, z_4, z_5)$$
$$y_3 = k(z_6)$$

Celles-ci nous permettent de définir la machine abstraite T :

$$T = f(y) = f(y_1, y_2, y_3)$$
$$T = f(g(z_1, z_2), h(z_3, z_4, z_5), k(z_6))$$

(écriture dans le magma libre)

L'écriture dans le magma libre d'une machine abstraite s'averre malcommode car ne permettant pas une modification de la machine abstraite par ajout d'un opérateur supplémentaire. Reprenons l'exemple précédent et ajoutons l'opérateur ℓ, redéfinissant z_1 :

$$z_1 = \ell(x_1, x_2)$$

Alors, T se réécrit :

$$T = f(g(\ell(x_1, x_2), z_2), h\ (z_3, z_4, z_5), k(z_6))$$

ou

$$T = f(g(\ell(z_1, z_2), z_3), h(z_4, z_5, z_6), k(z_7))$$

Cet exemple, nous montre que chaque fois que nous affinerons la description d'une machine abstraite, nous devrons réécrire '<u>totalement</u>' l'ensemble des paramètres.

L'introduction du produit tensoriel, corrige ce défaut.

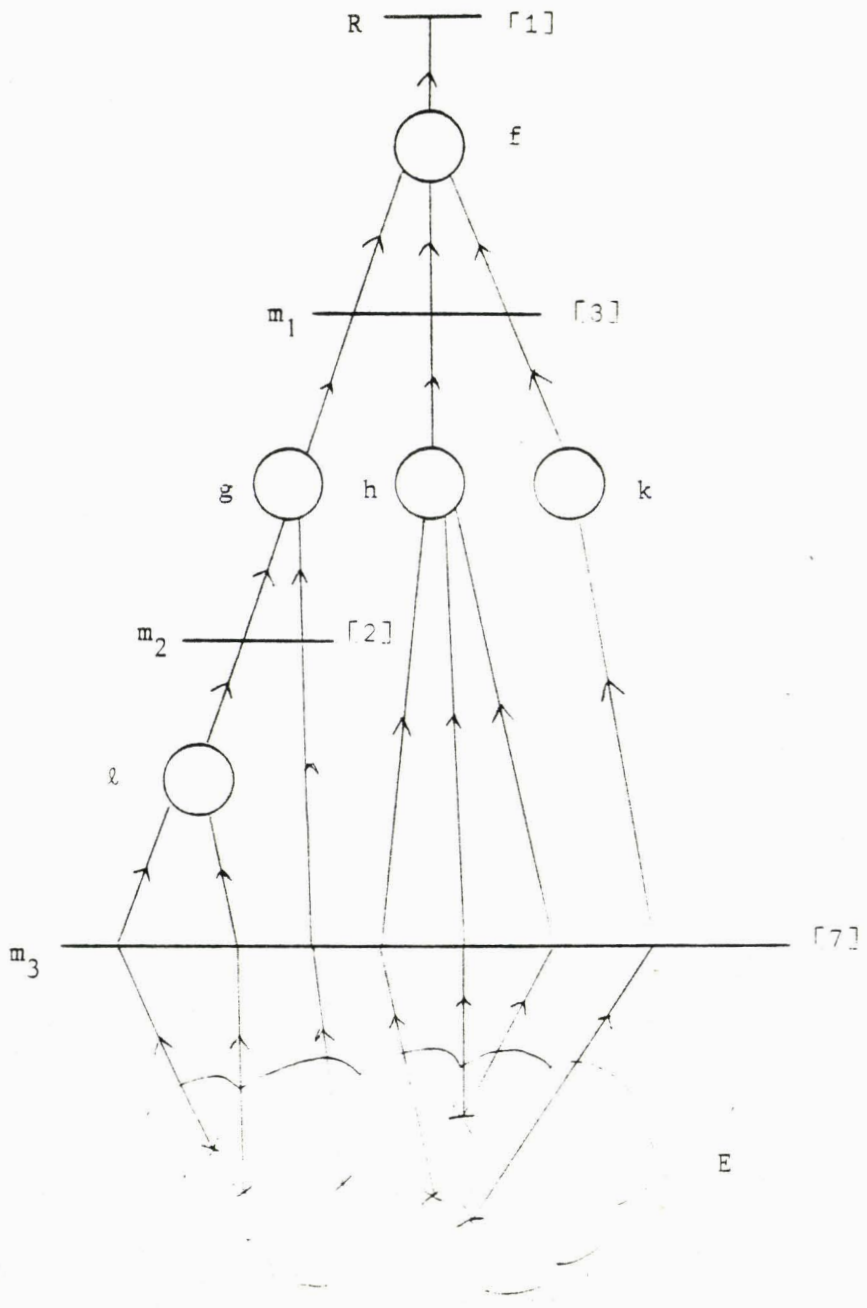
L'exemple précédent, définissant T, s'écrit, d'une manière équivalente :

$$T = f.(g(z_1, z_2) \otimes h(z_1, z_2, z_3) \otimes k(z_1))$$

ou

$$T = f.(g \otimes h \otimes k)$$

L'ajout de l'opérateur ℓ conduisant à la machine abstraite T suivante dont l'équation est :



$$T = f.((g.(\ell \otimes Id_1)) \otimes h \otimes k)$$

où Id_1 représente l'opérateur identité de F_1^1.

1.3 MAGMOÏDE [A, AD1, AD2, AD3]

La structure d'une machine abstraite, définie à l'aide des deux opérations de produit tensoriel et de produit de composition est un cas particulier d'une structure algébrique plus générale : le magmoïde. Rappelons brièvement les axiomes du magmoïde :

i - les deux opérations $\otimes$ et $\cdot$ sont associatives

ii - si $g \in F_p^q$ et $h \in F_{p'}^{q'}$: $g \otimes h \in F_{p+p'}^{q+q'}$

iii - si $g \in F_q^p$ et $h \in F_r^q$: $g \cdot h \in F_r^p$

vi - les deux opérations $\otimes$ et $\cdot$ vérifient une propriété de distributivité :

si les produits de composition u•u' et v•v' sont définis, alors nous avons l'égalité suivante, dans laquelle les deux membres sont bien définis :

$$(u \cdot u') \otimes (v \cdot v') = (u \otimes v) \cdot (u' \otimes v')$$

1.4 OBJETS "CHANGEMENT DE VARIABLE" : LES TORSIONS

Lorsque nous devons réaliser un cablage, il apparait souvent la situation suivante :

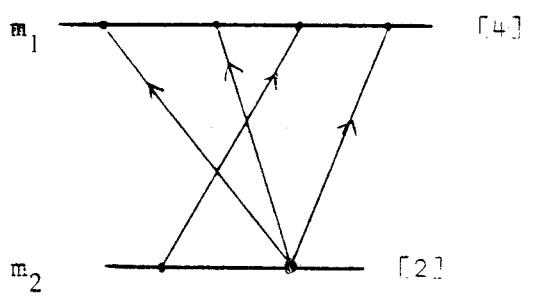


Figure 1.4

Afin de résoudre ce problème, nous introduisons les torsions. Celles-ci permettent de 'recopier' le contenu des cellules mémoires d'une mémoire à l'autre, avec éventuellement des oublis, des modifications dans l'ordre, des duplications.

A) Torsions et composition de torsion.

Le cablage de la figure précédente est décrit par la torsion θ : [4] ⟶ [2], application de [4] dans [2], d'où $\theta \in (H)_2^4$ (ensemble des applications de [4] dans [2]).

Sur l'exemple : $\theta(1) = \theta(2) = \theta(4) = 2$

$\theta(3) = 1$.

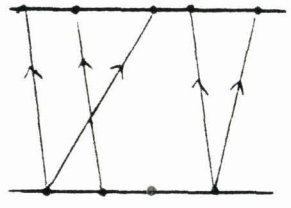
La torsion θ sera définie par : $<2 ; \theta(1), \theta(2), \theta(3), \theta(4)>$ c'est-à-dire par : $<2 ; 2, 2, 1, 2>$

D'une manière plus générale, si θ est une torsion de $(H)_p^q$ alors $\theta = <q ; \theta(1), \theta(2), \ldots, \theta(p)>$

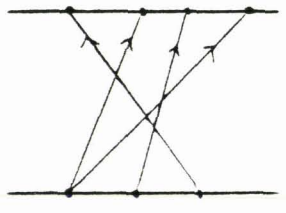
Les torsions peuvent également être composées à l'aide des deux lois : produit de composition et produit tensoriel.

i - produit de composition

Soient les deux torsions suivantes, θ_1 et θ_2 :

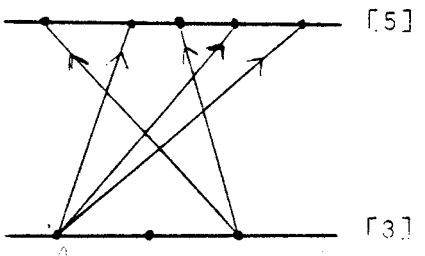


θ_1 : ⌈5⌉ → ⌈4⌋



θ_2 : ⌈4⌉ → ⌈3⌋

L'application composée $\theta_1 \cdot \theta_2$: [5] ⟶ [3] définie la torsion :



Et plus généralement, si θ_1 est une application de [p] dans [q] et θ_2 une application de [q] dans [r], la torsion composée $\theta_1 \cdot \theta_2$ est une application de [p] dans [r], définie de la manière suivante :

$$\theta_1 = \langle q \ ; \ \theta_1(1), \ \theta_1(2), \ldots, \theta_1(p) \rangle$$

$$\theta_2 = \langle r \ ; \ \theta_2(1), \ \theta_2(2), \ldots, \theta_2(q) \rangle$$

et $\theta_1 \cdot \theta_2 = \langle r \ ; \ \theta_2(\theta_1(1)), \ \theta_2(\theta_1(2)), \ldots, \theta_2(\theta_1(p)) \rangle$

Reprenons le dernier exemple :

$$\theta_1 = \langle 4 \ ; \ 1, \ 2, \ 1, \ 4, \ 4 \rangle$$

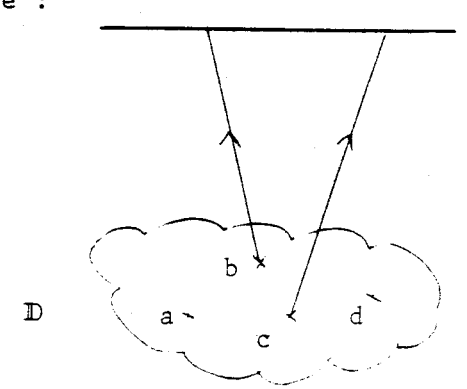
$$\theta_2 = \langle 3 \ ; \ 3, \ 1, \ 2, \ 1 \rangle$$

et $\theta_1 \cdot \theta_2 = \langle 4 \ ; \ 1, \ 2, \ 1, \ 4, \ 4 \rangle \cdot \langle 3 \ ; \ 3, \ 1, \ 3, \ 1, \ 1 \rangle$

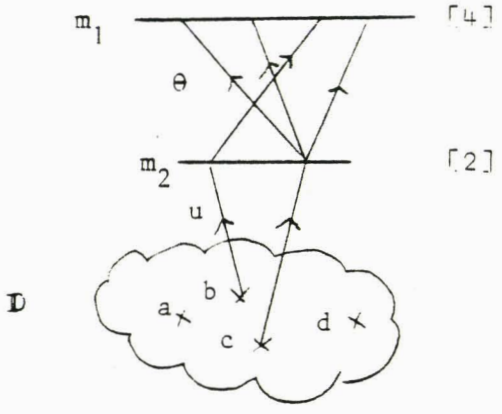
Lemme : Le composition des torsion est associative·[AD1].

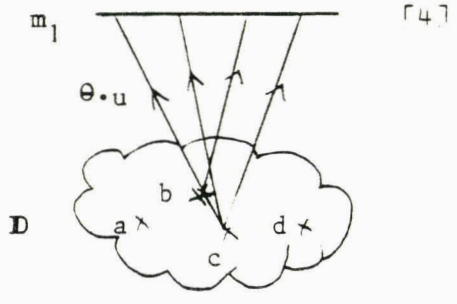
B) Action des torsions sur les assignations

Reprenons la torsion θ décrite figure 1.4 et considérons l'assignation u suivante :

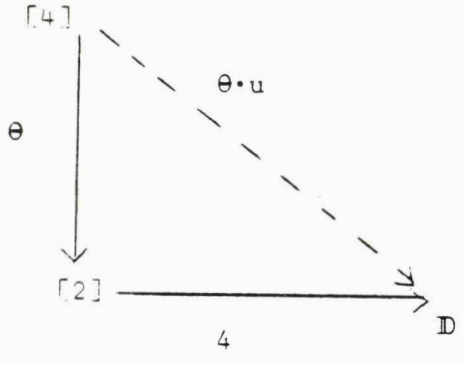


Alors les cablages (1) et (2) opèreront toujours de manière identique sur les assignations.





Remarquons que θ•u est obtenu par simple composition d'applications.



SCHEMA DE CABLAGE ASSOCIE A UN PROCESSUS

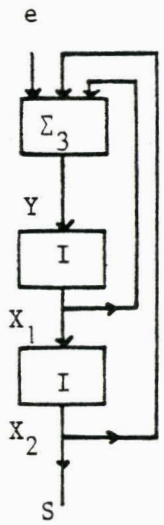
2.0 INTRODUCTION

Les propriétés de l'algèbre de décomposition, nous permettront de décrire formellement le schéma de cablage d'un processus.

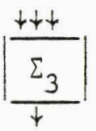
Considérons un processus P_1 et le modèle mathématique le décrivant :

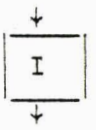
$$(1) \quad \begin{cases} \overset{\circ}{x}_1 = x_1 + x_2 + e \\ \overset{\circ}{x}_2 = x_1 \\ s = x_2 \end{cases}$$

Si nous disposons d'opérateurs d'intégration et de sommation, le modèle mathématique (1) peut être représenté par le schéma de cablage suivant [B2] :



qui utilise les cellules opérateurs Σ_3 et I :

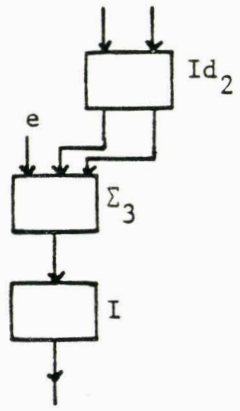
(i)  représente l'opérateur de sommation $\Sigma_3 \in F_3^1$

(ii)  représente l'opérateur d'intégration $I \in F_1^1$

En utilisant les résultats du paragraphe précédent, ce schéma peut être décrit par le système formel d'équations :

$$(\varepsilon_1)\begin{cases} Y = \Sigma_3(e_1, X_1, X_2) = \Sigma_3.(e_1 \otimes X_1 \otimes X_2)^* \\ X_1 = I.(Y) \\ X_2 = I.(X_1) \\ S = X_2 \end{cases}$$

A présent, soit le schéma de cablage suivant, où Id_2 représente l'opérateur identité de F_2^2 ($Id_2.\langle X_1 \otimes X_2\rangle = \langle X_1 \otimes X_2\rangle$) :



Nous considérons, dans la suite de cet exemple, ce schéma comme une machine abstraite, ayant deux entrées et une sortie et définie par l'équation :

$$M = I.\Sigma_3.(e_1 \otimes Id_2)$$

M est un opérateur de F_2^1 σ

Remarque : Rappel sur le magmoïde [AD]

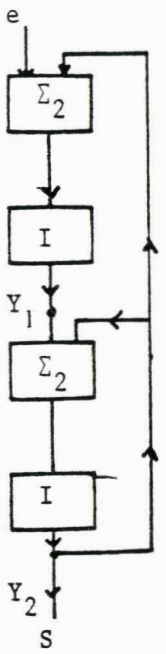
$\langle u \otimes v\rangle = \langle u, v\rangle$ si $u, v \in F_1^o$

2.1 AUTRES CABLAGES D'UN MEME PROCESSUS

Considérons l'équation différentielle linéaire, suivante :

$$(2)\begin{cases} \overset{\circ}{y}_1 = y_2 + e \\ \overset{\circ}{y}_2 = y_1 + y_2 \\ s = y_2 \end{cases}$$

et le schéma de cablage associé :



Les modèles (1) et (2) définissent le même processus, et par conséquent les schémas de cablage associés également.

Pour montrer ceci nous proposons de faire subir des manipulations élémentaires aux schémas sans faire usage des modèles mathématiques.

Ceci nous permet de réécrire le système (ε_1) :

$$X_1 = I.\Sigma_3(e \otimes X_1 \otimes X_2)$$

$$= M.\langle X_1, X_2\rangle$$

$$= \langle I.\Sigma_3.(e \otimes 1d_2)\rangle \langle X_1, X_2\rangle$$

$$X_2 = I.(X_1)$$

$$S = X_2$$

En introduisant les torsions, le système (ε_1) peut être réécrit :

$$S = \langle 1d_1\rangle \langle 3 ; 2\rangle \langle X_1, X_2, S\rangle$$

$$X_2 = \langle I\rangle \langle 3 ; 1\rangle \langle X_1, X_2, S\rangle$$

$$X_1 = \langle I.\Sigma_3.(e \otimes 1d_2)\rangle \langle 3 ; 1, 2\rangle \langle X_1, X_2, S\rangle$$

$$\ldots \Rightarrow \langle X_1, X_2, S\rangle = \langle \Sigma_2.(I.\Sigma_2.(e \otimes Id_1) \otimes Id_1) \otimes I \otimes Id_1\rangle \qquad (\gamma)$$
$$\langle 3\ ;\ 2, 2, 1, 2\rangle \langle S, X_1, X_2\rangle$$

Nous allons introduire deux nouveaux termes afin d'identifier le schéma de la figure 4 avec le schéma de cablage (S2) .

Posons $Y_1 = \langle I.\Sigma_2.(e_1 \otimes Id_1)\rangle \langle X_2\rangle$
et $Y_2 = X_2$,
ce qui nous permet de réécrire l'équation (γ) définissant le schéma de la figure 4 :

$$\langle X_1, X_2, S, Y_1, Y_2\rangle = \langle \Sigma_2 \otimes I \otimes Id_1 \otimes I.\Sigma_2(e_1 \otimes Id_1) \otimes Id_1\rangle$$
$$\langle 5\ ;\ 2, 4, 1, 2, 2, 2\rangle \langle X_1, X_2, S, Y_1, Y_2\rangle \quad \text{(fig 5)}$$

Nous remarquons que $Y_2 = \langle I.\Sigma_2\rangle \langle Y_1, Y_2\rangle$, d'où finalement
$\langle S, Y_1, Y_2\rangle = \langle Id_1 \otimes I.\Sigma_2.(e \otimes Id_1) \otimes I.\Sigma_2\rangle \langle 3\ ;\ 3, 3, 2, 3\rangle$
$\langle S, Y_1, Y_2\rangle$

Cette équation est la description formelle du schéma de cablage (S2).

Remarque : Cette démonstration repose sur les propriétés R1 et R2, qui sont vérifiées pour l'interprétation présente (le domaine est l'ensemble des réels où les opérateurs sommateur et intégrateur sont respectivement associatif et linéaire).

et finalement, le produit tensoriel permet d'écrire le système (ε_1) sous la forme :

$$(\varepsilon_1) : \langle X_1, X_2, S\rangle = \langle I.\Sigma_3.(e \otimes Id_2) \otimes I \otimes 1d_1\rangle \langle 3 ; 1, 2, 1, 2\rangle \langle X_1, X_2, S\rangle$$

L'équation (ε_1) décrit formellement le schéma de cablage (S_1) associé au processus P_1.

Montrons que les seules propriétés combinatoires de l'algèbre de décomposition et les propriétés R_1 et R_2 suivantes, entrainent l'équivalence des schémas S1 et S2.

R1 : $\Sigma_i = \Sigma_2.(\Sigma_{i-1} \otimes Id_1) = \Sigma_2.(Id_1 \otimes \Sigma_{i-1})$

de manière générale, Σ_i est un opérateur de F_i^1, représente le sommateur ayant i entrées.

R2 : $I.\Sigma_i = \Sigma_i.(I \otimes I \otimes \ldots \otimes I)$ (⟵ i fois ⟶)

L'opérateur I est linéaire.

En partant de l'écriture formelle du schéma de cablage (ε_1) ; nous allons modifier le système d'équations en remplaçant des termes par des termes équivalents.

$$(\varepsilon_1) : \langle X_1, X_2, S\rangle = \langle I.\Sigma_3.(e \otimes Id_2) \otimes I \otimes Id_1\rangle \langle 3 ; 1, 2, 1, 2\rangle \langle X_1, X_2, S\rangle$$

par définition : $d_2 = Id_1 \otimes Id_1$

propriété R1 : $\Sigma_3 = \Sigma_2.(\Sigma_2 \otimes 1d_1) = \Sigma_2.(Id_1 \otimes \Sigma_2)$

⟹ (fig 1)

$$\Rightarrow \quad \langle X_1, X_2, S\rangle = \langle I.\Sigma_2.(\Sigma_2.(e \otimes Id_1) \otimes | \otimes Id_1\rangle \langle 3 ; 1, 2, 1, 2\rangle \langle X_1, X_2, S\rangle$$

utilisons la propriété R_2 : $I.\Sigma_2 = \Sigma_2.(I \otimes I)$

⟹ (fig 2)

$$\ldots \Rightarrow \langle X_1, X_2, S\rangle = \langle \Sigma_2.(I.\Sigma_2.(e \otimes Id_1) \otimes I) \otimes I \otimes Id_1\rangle \langle 3 ; 1, 2, 1, 2 \quad X_1, X_2, S\rangle$$

⟹ (fig 4)

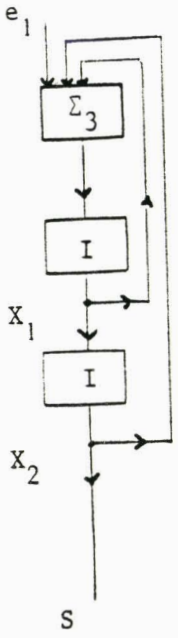


Fig. 1

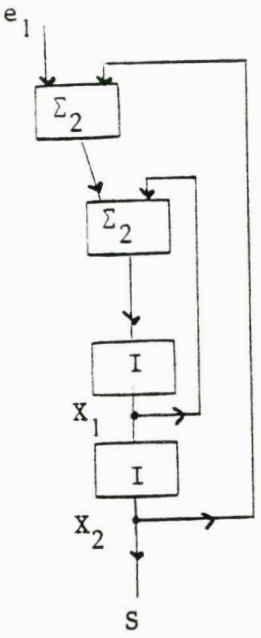


Fig. 2

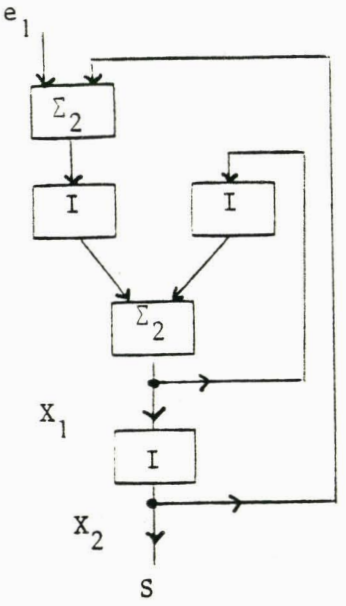


Fig. 3

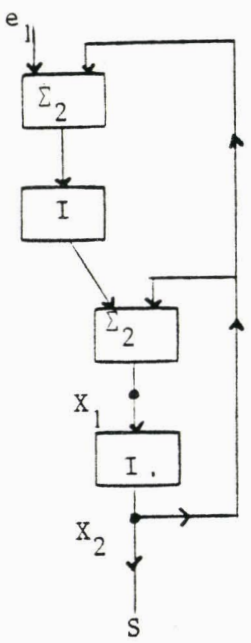


Fig. 4

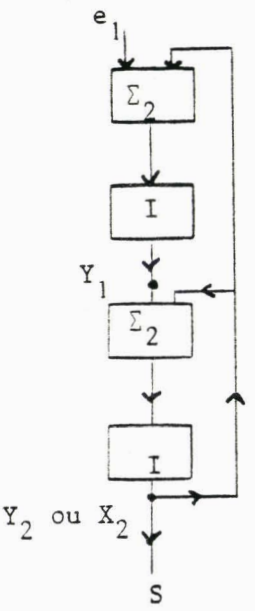


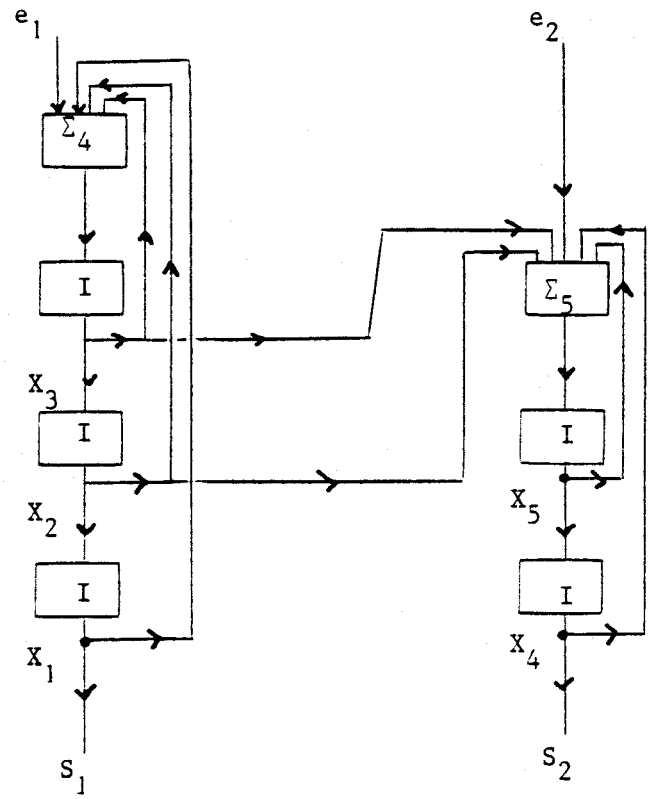
Fig. 5

2. EXEMPLE 2

Considérons, maintenant, les systèmes d'équations différentielles (3) et (4) et leur schéma de cablage respectif (S3) et (S4) :

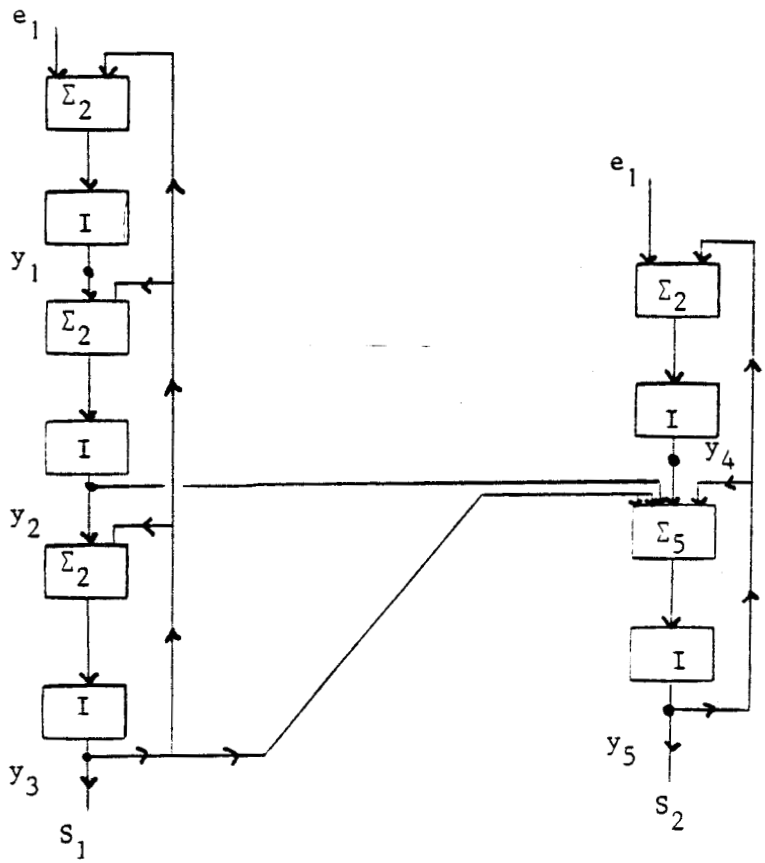
$$(3)\quad \begin{cases} \overset{o}{x}_1 = x_2 \\ \overset{o}{x}_2 = x_3 \\ \overset{o}{x}_3 = x_1 + x_2 + x_3 + e_1 \\ \overset{o}{x}_4 = x_5 \\ \overset{o}{x}_5 = x_2 + x_3 + x_4 + x_5 + e_2 \\ S_1 = x_1 \\ S_2 = x_2 \end{cases}$$

(S3)



$$(4):\begin{cases} \overset{\circ}{y}_1 = y_3 + e_1 \\ \overset{\circ}{y}_2 = y_1 + y_3 \\ \overset{\circ}{y}_3 = y_2 + y_3 \\ \overset{\circ}{y}_4 = y_5 + e_2 \\ \overset{\circ}{y}_5 = y_2 + 2.y_3 + y_4 + y_5 \\ S_1 = y_3 \\ S_2 = y_5 \end{cases}$$

(S4)



Les modèles (3) et (4) sont associés au même processus mais la dualité [B3] ne peut être utilisée pour le démontrer comme dans le cas précédent, pourtant l'application des propriétés combinatoires de l'algèbre de décomposition aux schémas de cablage S3 et S4 permet de mettre en évidence que ces deux modèles définissent le même processus.

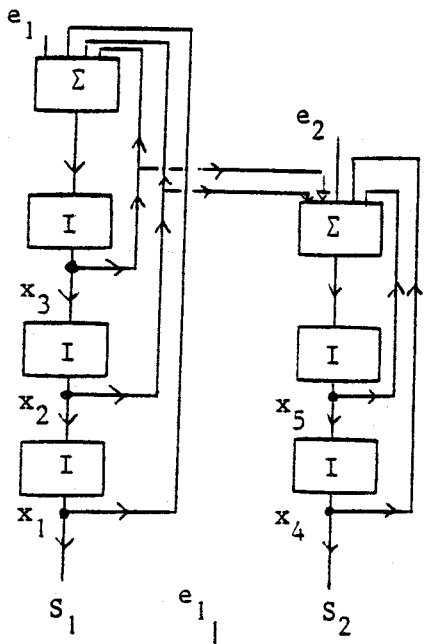


Fig. 1

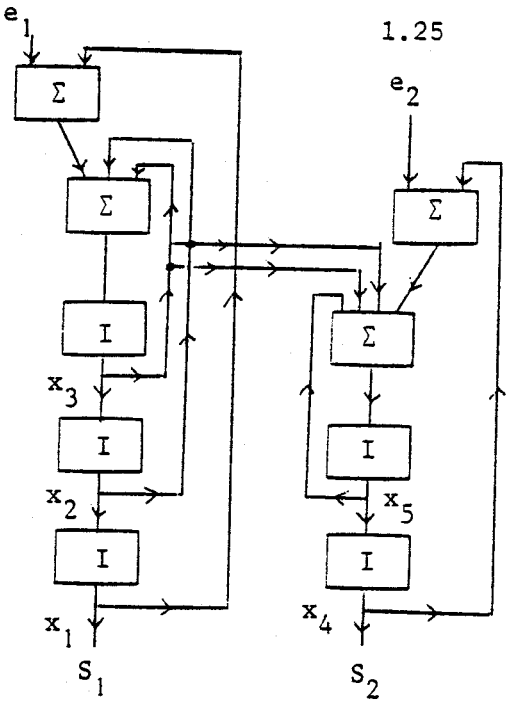


Fig. 2

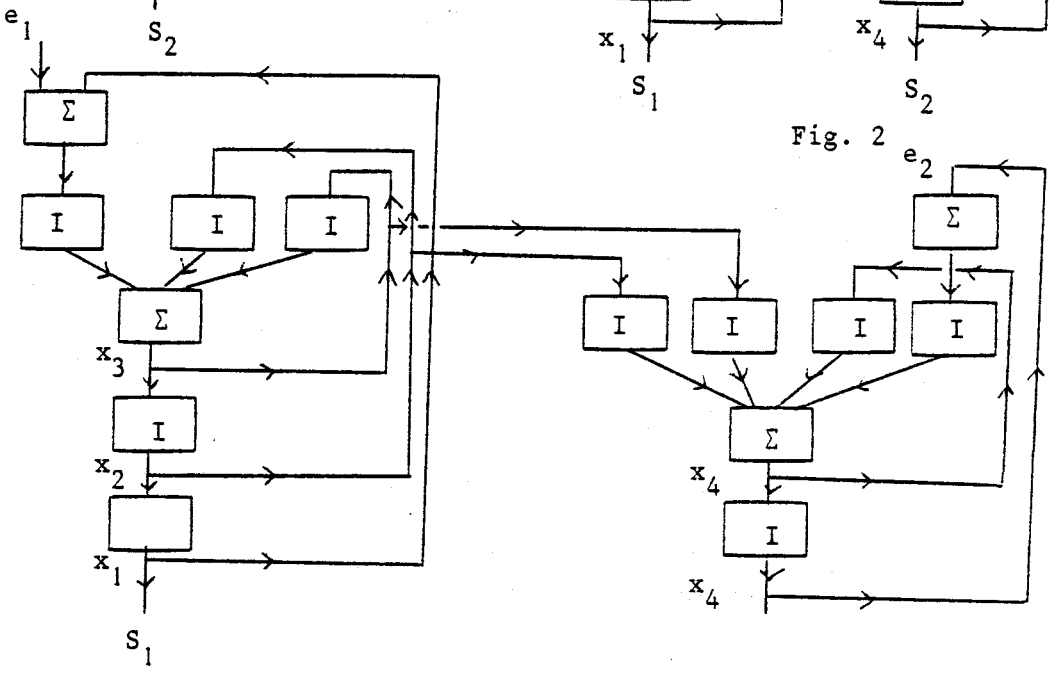


Fig. 3

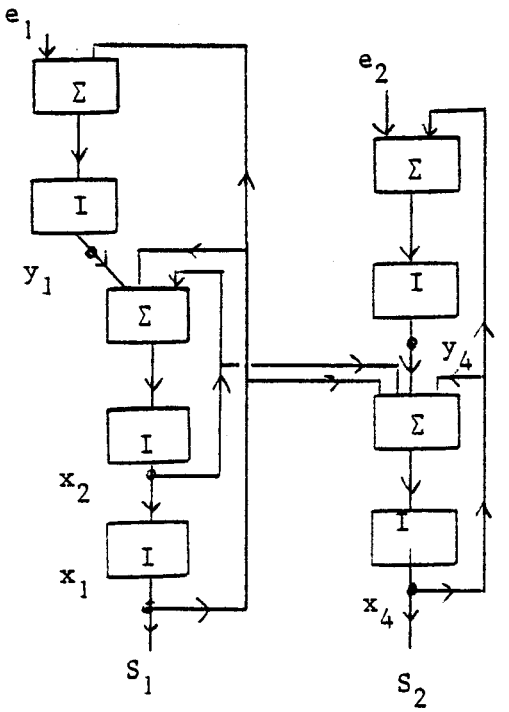


Fig. 4

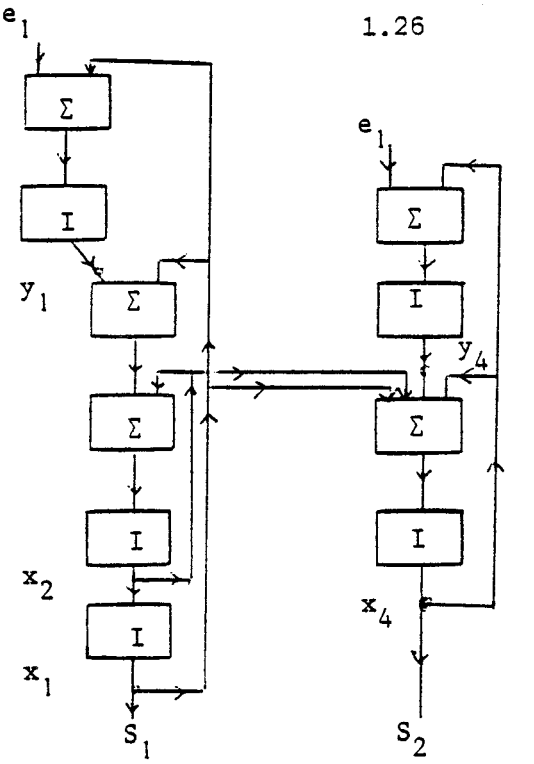


Fig. 6

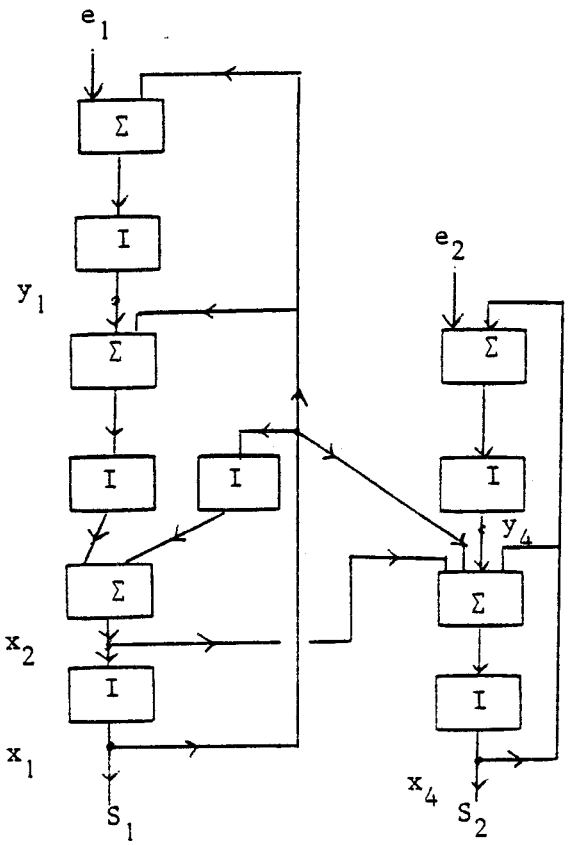


Fig. 6

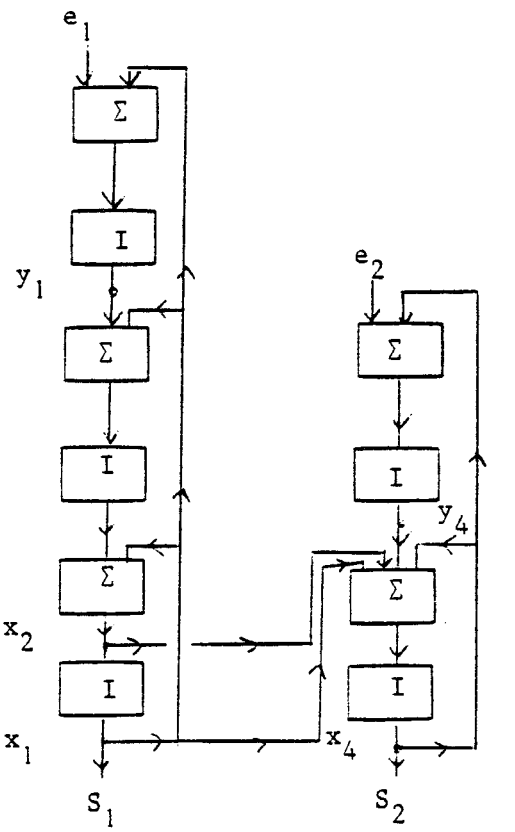


Fig. 7

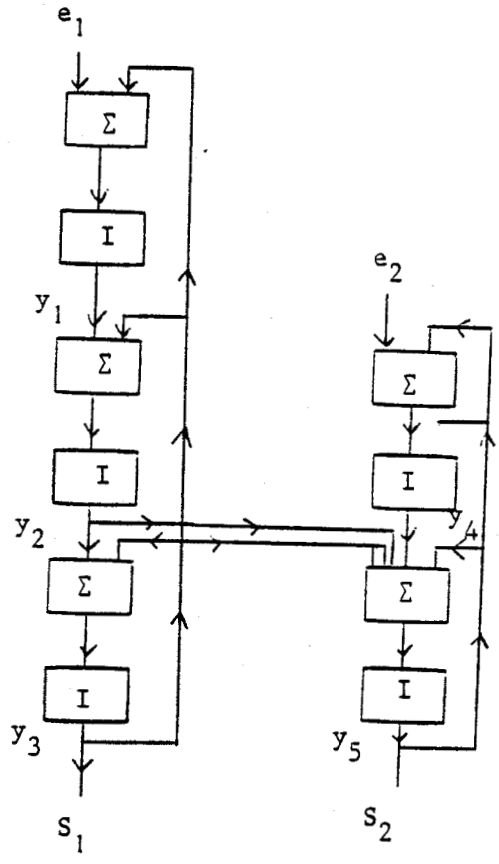


Fig. 8

Considérons le système d'équations représentant le schéma de cablage (S3) : (fig. 1)

$$\langle X_1, X_2, X_3, X_4, X_5\rangle = \langle I_2 \otimes I.\Sigma_4.(e_1 \otimes Id_3) \otimes I \otimes I.\Sigma_5.(e_2 \otimes Id_4)\rangle$$
$$\langle 5\ ;\ 2,\ 3,\ 1,\ 2,\ 3,\ 5,\ 2,\ 3,\ 4,\ 5\rangle \langle X_1, X_2, X_3, X_4, X_5\rangle$$

(I_2 représente l'opérateur de F_2^2 : $I \otimes I$)

La propriété R1 entraine :

$$\langle X_1,X_2,X_3,X_4,X_5\rangle = \langle I_2 \otimes I.\Sigma_3.(\Sigma_2(e_1 \otimes Id_1) \otimes 1d_2) \otimes I \otimes I.\Sigma_4.(\Sigma_2(e_2 \otimes Id_1) \otimes Id_3)\rangle$$
$$\langle 5\ ;\ 2,\ 3,\ 1,\ 2,\ 3,\ 5,\ 2,\ 3,\ 4,\ 5\rangle \langle X_1, X_2, X_3, X_4, X_5\rangle \quad \text{(fig 2)}$$

En utilisant la propriété R2 nous obtenons

$$\langle X_1,X_2,X_3,X_4,X_5\rangle = \langle I_2 \otimes \Sigma_3.(I.\Sigma_2.(e_1 \otimes Id_1) \otimes I_2) \otimes I \otimes \Sigma_4.(I.\Sigma_2(e_2 \otimes Id_1) \otimes I_3)$$
$$\langle 5\ ;\ 2,\ 3,\ 1,\ 2,\ 3,\ 5,\ 2,\ 3,\ 4,\ 5\rangle \langle X_1, X_2, X_3, X_4, X_5\rangle \quad \text{(fig 3)}$$

ou

$$\langle X_1,X_2,X_3,X_4,X_5\rangle = \langle I_2 \otimes \Sigma_3.(I.\Sigma_2.(e_1 \otimes Id_1) \otimes 1d_2) \otimes I \otimes \Sigma_4.(I.\Sigma_2.(e_2 \otimes Id_1) \otimes Id_3)\rangle$$
$$\langle 5\ ;\ 2,\ 3,\ 1,\ 1,\ 2,\ 5,\ 4,\ 1,\ 2,\ 4\rangle \langle X_1, X_2, X_3, X_4, X_5\rangle \quad \text{(fig 4)}$$

Introduisons les variables auxiliaires Y_1 et Y_4 dont les équations les définissant sont :

$$Y_1 = I.\Sigma_2.(e_1 \otimes Id_1).\langle X_1\rangle$$

et $$Y_4 = I.\Sigma_2.(e_2 \otimes Id_1).\langle X_4\rangle$$

Les variables Y_1 et Y_4 vont nous permettre de réécrire le système S3 :

$$\langle X_1,X_2,Y_1,X_4,Y_4\rangle = \langle I \otimes I.\Sigma_3 \otimes I.\Sigma_2(e_1 \otimes Id_1) \otimes I.\Sigma_4 \otimes I.\Sigma_2(e_2 \otimes Id_1)\rangle$$
$$\langle 5\ ;\ 2,\ 3,\ 1,\ 2,\ 1,\ 5,\ 1,\ 2,\ 4,\ 4\rangle \langle X_1, X_2, Y_1, X_4, Y_4\rangle \text{ (fig 4)}$$

La propriété R1 entraine :

$$\langle X_1,X_2,Y_1,X_4,Y_4\rangle = \langle I \otimes I.\Sigma_2(\Sigma_2 \otimes Id_1) \otimes I.\Sigma_2(e_1 \otimes Id_1) \otimes I.\Sigma_4$$
$$\otimes I.\Sigma_2.(e_2 \otimes Id_1)\rangle \langle 5\ ;\ 2,\ 3,\ 1,\ 2,\ 1,\ 5,\ 1,\ 2,\ 4,\ 4\rangle \langle X_1,X_2,Y_1,X_4,Y_4\rangle \text{ (fig 5)}$$

En utilisant la propriété R2 nous obtenons :

$$\langle X_1,X_2,Y_1,X_4,Y_4\rangle = \langle I \otimes \Sigma_2(I.\Sigma_2 \otimes I) \otimes I.\Sigma_2(e_1 \otimes Id_1) \otimes I.\Sigma_4 \otimes I.\Sigma_2(e_2 \otimes Id_1)\rangle$$
$$\langle 5\ ;\ 2,\ 3,\ 1,\ 2,\ 1,\ 5,\ 1,\ 2,\ 4,\ 4\rangle \langle X_1, X_2, Y_1, X_4, Y_4\rangle \text{ (fig 6)}$$

ou

$$\langle X_1, X_2, Y_1, X_4, Y_4\rangle = \langle I \otimes \Sigma_2 . (I.\Sigma_2 \otimes Id_1) \otimes I.\Sigma_2 . (e_1 \otimes Id_1) \otimes I.\Sigma_4 \otimes I.\Sigma_2 (e_2 \otimes Id_1)\rangle$$
$$\langle 5 ; 2, 3, 4, 1, 5, 1, 2, 4, 4\rangle \langle X_1, X_2, Y_1, X_4, Y_4\rangle \quad \text{(fig 7)}$$

Introduisons les variables intermédiaires Y_2, Y_3, Y_5 définies par :

$$Y_3 = X_1$$

$$Y_5 = X_5$$

$$Y_2 = \langle I.\Sigma_2\rangle \langle X_1, Y_1\rangle = \langle 1.\Sigma_2\rangle \langle Y_3, Y_1\rangle$$

D'où

$$\langle Y_3, Y_2, Y_1, Y_5, Y_4\rangle = \langle I.\Sigma_2 \otimes 1.\Sigma_2 \otimes I.\Sigma_2 . (e_1 \otimes Id_1) \otimes I.\Sigma_5 \otimes I.\Sigma_2 (e_2 \otimes Id_1)$$
$$\langle 5 ; 2, 1, 3, 1, 1, 1, 2, 4, 5, 4\rangle \langle Y_3, Y_2, Y_1, Y_5, Y_4\rangle \text{ (fig 8)}$$

2.2 INTERPRETATION D'UN SCHEMA DE CABLAGE [B, B1]

Pour les quatre schémas de cablage S1, S2, S3 et S4 nous nous sommes intéressés à une interprétation particulière, associée aux systèmes d'équations différentielles linéaires mais le schéma de cablage est un "outil" formel donc indépendant de toute interprétation.

Reprenons les schémas S1 et S2 et changeons d'interprétation. Le processus P_1 évolue en fonction d'une variable indépendante, non plus continue, mais discrète. L'opérateur Σ_i reste le même, tandis que l'opérateur I devient l'opérateur Retard : R. Dans ce cas les schémas S1 et S2 sont associés aux systèmes linéaires discrets (1') et (2') (équations récurrentes) :

$$(1') \begin{cases} x_1^{k+1} = x_2^k \\ x_2^{k+1} = x_1^k + x_2^k + e_1^k \\ s^{k+1} = x_1^{k+1} \end{cases} \qquad (2') \begin{cases} y_1^{k+1} = y_2^k + e_1^k \\ y_2^{k+1} = y_1^k + y_2^k \\ s^{k+1} = y_1^{k+1} \end{cases}$$

que nous pouvons noter :

$$(1') \begin{cases} x_1^+ = x_2 \\ x_2^+ = x_1 + x_2 + e_1 \\ s = x_1 \end{cases} \qquad (2') \begin{cases} y_1^+ = y_2 + e_1 \\ y_2^+ = y_1 + y_2 \\ s = y_2 \end{cases}$$

Les propriétés R1 et R2 sont vérifiées par les 2 opérateurs de sommation discrets et de retard pur,les deux modèles (1') et (2') définissent le même processus.

Fournissons une dernière interprétation des schémas S1 et S2. Les variables sont définies sur un corps de Gallois module 4, l'opérateur Σ_i devient l'addition module 4 et l'opérateur I le retard pur. Dans ce cas, les schémas S1 et S2 sont associés aux systèmes logiques séquentiels linéaires (1") et (2") :

$$(1'') \quad \begin{cases} x_1^+ = x_2 \\ x_2^+ = (x_1 + x_2 + e_1) \pmod 4 \\ s = x_1 \end{cases} \qquad (2'') \quad \begin{cases} y_1^+ = (y_2 + e_1) \pmod 4 \\ y_2^+ = (y_1 + Y_2) \pmod 4 \\ s = y_2 \end{cases}$$

Ici encore les propriétés R1 et R2 sont vérifiées, ce qui entraine que les deux modèles (1") et (2") définissent le même processus.

L'étude des processus continus, discrets ou séquentiels fait apparaître une certaine similitude de propriété [G]. Dans cet esprit, les schémas de cablage permettent une représentation unifiée de ces systèmes, ainsi apparait une approche commune à bien des problèmes de modélisations quelle que soit la nature du processus envisagé.

3. FACTORISATION D'UNE EXPRESSION FORMELLE [C]

3.0 DEFINITIONS

Nous supposerons, maintenant, que le schéma de cablage est donné, ou obtenu à partir des résultats précédents, la prochaine étape que nous envisagerons concerne la factorisation d'une expression formelle.

Un schéma de cablage se caractérise par un système d'équations formelles dans le magmoïde. Nous nommerons expression formelle le membre droit d'une équation formelle dans le magmoïde.

La factorisation d'une expression formelle permettra de minimiser le nombre d'opérateurs nécessaires lors du cablage du schéma. Avant d'aborder l'étude de cet algorithme de factorisation, nous commencerons par faire quelques rappels sur les arbres.

3.0.1 RAPPELS SUR LES ARBRES(D)

Une expression formelle E est un arbre étiqueté. On appelle noeud de E toute occurence dant E d'un symbole ou d'une variable. Ce symbole ou cette variable sont l'étiquette du noeud.

Puisque deux noeuds de E peuvent avoir la même étiquette nous distinguerons nominalement les noeuds en les numérotant injectivement.

La fonction d'étiquetage L associe au noeud i son étiquette (symbole d'opérateur ou variable).

Soit $i, \in F_q^p$, i noeud de E, nous noterons dsup(i) = p le degrè supérieur de i et dinf(i) = q son degrè inférieur.

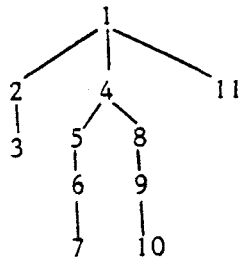
Illustrons sur un exemple les définitions ci-dessus. Considérons l'expression formelle E,

$$E = \Sigma_3 \cdot (I \cdot \bar{z} \otimes M \cdot (I \cdot I \cdot \bar{z} \otimes I \cdot I \cdot \bar{z}) \otimes e)$$

E se schématise en :

$$E = 1 \cdot (2 \cdot 3 \otimes 4 \cdot (5 \cdot 6 \cdot 7 \otimes 8 \cdot 9 \cdot 10) \otimes 11)$$

qui peut être représenté par l'ordre étiqueté suivant



avec $L(1) = \Sigma_3$

$L(2) = L(5) = L(6) = L(8) = L(9) = I$

$L(3) = L(7) = L(10) = \bar{z}$

$L(4) = M$

$L(11) = e$

b. ordre $\leqslant_E$ [D]

E étant fixé, nous définissons entre ses noeuds un ordre partiel, noté $\leqslant_E$, défini comme suit :

(i) $a R_E b \iff$ E se décompose en $E = t_1(t_2, a(t_4, b \cdot t_6, t_5), t_3)$

(ii) $\leqslant_E$ est la clôture transitive de R_E

Dauchet [D] a montré que $\leqslant_E$ est un ordre partiel et que $a \leqslant_E b$ ssi a et b confondus ou E se décompose en $E = u_1(u_2, a \cdot u_4(u_5, b \cdot u_7, u_6), u_3)$. Si $a R_E b$, on dit que a est <u>prédécesseur immédiat</u> de b et b <u>successeur immédiat</u> de a.

Si $a \leqslant_E b$, on dit que a est prédécesseur de b (ou a précède b) et que b est successeur de a (ou que b succède à a). Les <u>variables</u> d'une expression formelle sont les noeuds ayant aucun successeur, les <u>opérateurs</u> sont les noeuds ayant au moins un successeur.

Dans la suite de ce chapitre, nous nous intéresserons aux arbres étiquetés E dont chaque opérateur i est tel que dsup(i) = 1. L'ensemble des noeuds de E, muni de $\leq_E$ est un inf-demi treilli. L'élément minimal est appelé racine de E.

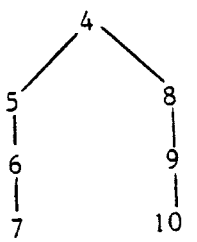
Les sous-branches de E sont les suite (a) = $(a_1,\ldots,a_p)$ telles que, pour tout i, a_{i+1} soit successeur immédiat de a_i. La longueur de la sous-branche (a), notée long (a), est égale à p. La branche est une sous-branche où α_p est maximal et a_1 minimal. La profondeur de E, notée prof(E), est la borne supérieure des longueurs des branches de E.

Dans l'exemple précédent, l'ensemble des successeurs immédiats du noeud 1 est (2, 4, 11).

C) Sous-arbres

Soit un arbre E, E' est un sous-arbre de E ssi E se décompose en $E = E_1(E_2, E', E_3)$. Le noeud i de E racine du sous-arbre E' défini ce sous-arbre.

Reprenons l'exemple précédent, le noeud (4) défini le sous-arbre E' :



3.1 EXEMPLE PRELIMINAIRE

Afin d'illustrer, à priori, l'algorithme de factorisation, nous nous proposons de définir le cablage d'un modèle mathématique, ayant le nombre minimal d'opérateurs.

Considérons le modèle mathématique suivant :

$$\overset{oo}{y} = \overset{o}{y} + y^2 + e$$

nous pouvons lui associer le schéma de cablage (ε_5) :

$$(\varepsilon_5) : \begin{cases} Z = \Sigma_3\ (I \cdot \bar{z} \otimes M \cdot (I^2 \cdot Z \otimes I^2 \cdot \bar{z}) \otimes e) \\ Y = I^2 \cdot \bar{z} \end{cases}$$

Le schéma de cablage peut être réécrit sous forme symbolique

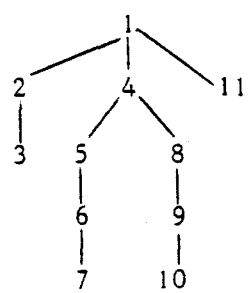
$$(\varepsilon_6) \quad \begin{cases} Z = 1 \cdot (2 \cdot 3 \otimes 4 \cdot (5 \cdot 6 \cdot 7 \otimes 8 \cdot 9 \cdot 10) \otimes 11) \\ Y = 12 \cdot 13 \cdot 14 \end{cases}$$

avec la fonction d'étiquetage L :

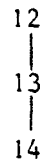
$L(1) = \Sigma_3$,
$L(2) = L(5) = L(6) = L(8) = L(9) = L(12) = L(13) = I$
$L(3) = L(7) = L(10) = L(14) = \bar{z}$,
$L(4) = M$,
$L(11) = e$

Nous avons deux expressions formelles E_1 et E_2, représenté par les arbres étiquetés suivants :

E1:



E2:



Maintenant, nous cherchons à obtenir les expressions formelles dont l'implantation sur une calculatrice analogique nécessitera le nombre minimal d'opérateurs : Σ_3, I, M et de variables $\bar{z}$, e.

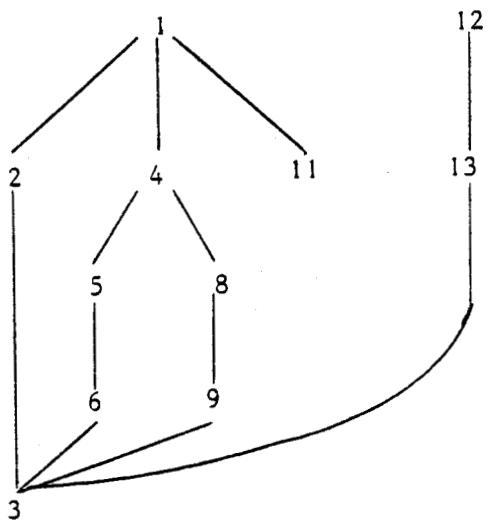
L'algorithme de factorisation sera développé en trois étapes.

1er Pas : Soit X l'ensemble des noeuds des deux expressions formelles E_1 et E_2. Parmi ces noeuds, isolons l'ensemble X_1 des variables ayant même fonction d'étiquetage.

$X_1 = (3, 7, 10, 14)$ et $L(3) = \bar{z}$

Nous pouvons substituer aux noeuds 7, 10, 14 le noeud 3. Nous obtenons le graphe G_1 équivalent aux deux arbres E_1 et E_2 :

G1 :



D'une manière formelle, nous introduisons une nouvelle variable W_1 caractérisée par l'équation suivante :

$$W_1 = 3$$

Le système d'équation (ε6) peut être réécrit :

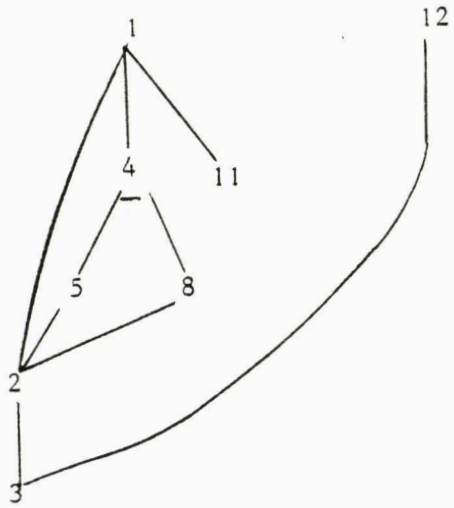
$$\begin{cases} Z = 1 \cdot (2 \cdot W_1 \otimes 4 \cdot (5 \cdot 6 \cdot W_1 \otimes 8 \cdot 9 \cdot W_1) \otimes 11) \\ Y = 12 \cdot 13 \cdot W_1 \\ W_1 = 3 \end{cases}$$

Pas 2 : Les noeuds de l'ensemble X_1 admettent respectivement pour prédécesseur immédiat les noeuds (2, 6, 9, 13), qui sont tous de degrè inférieur 1.

La fonction d'étiquetage des noeuds appartenant à cet ensemble X_2 leur associe le même symbole opérateur : I, donc les sous arbres de racines 2, 6, 9, 13 sont identiques.

Nous pouvons substituer, dans le graphe G_1, aux noeuds 6, 9, 13 le noeud 2, ce qui nous donne le graphe G_2 :

G2:



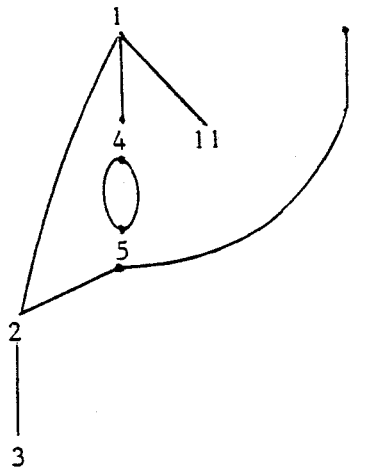
Si nous introduisons une nouvelle variable W_2, au graphe G_2 correspond le système d'équation :

$$\begin{cases} Z = 1 \cdot (W_2 \otimes 4 \cdot (5 \cdot W_2 \otimes 8 \cdot W_2) \otimes 11) \\ Y = 12 \cdot W_2 \\ W_1 = 3 \\ W_2 = 2 \cdot W_1 \end{cases}$$

Pas 3 : Les noeuds de l'ensemble X_2 admettent respectivement pour prédécesseur immédiat les noeuds (1, 5, 8, 12). Le degrè inférieur du noeud 1 est 3 et des noeuds 5, 8, 12 : 1. Puisque la fonction d'étiquetage des noeuds 5, 8, 12 leur associe le même symbole opérateur I, les trois sous-arbres de racines 5, 8, 12 sont identiques. Nous pouvons substituer aux noeuds 8 et 12 du graphe G_2, le noeud 5.

Nous obtenons le graphe G_3 :

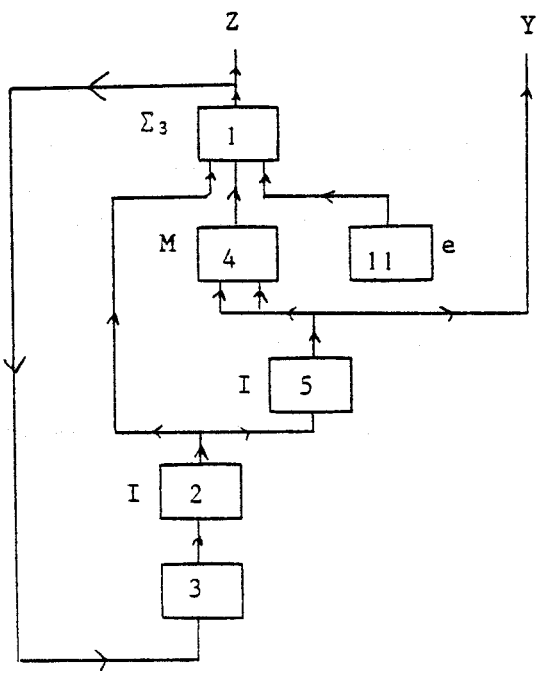
G3:



Après introduction d'une nouvelle variable W_3, correspond au graphe G_3, le système d'équations suivant :

$$(A)\quad \begin{cases} Z = 1 \cdot (W_2 \otimes 4\ (W_3 \otimes W_3) \otimes 11) \\ Y = W_3 \\ W_3 = 5 \cdot W_2 \\ W_2 = 2 \cdot W_1 \\ W_1 = 3 \end{cases}$$

Nous remarquons que le graphe G_3 correspond au système d'équations (A) après factorisation des sous-arbres identiques (2, 3) et (5, 2, 3). La fonction d'étiquetage permet, à partir du graphe G_3, d'obtenir le cablage final :



De cet exemple, il est possible de tirer l'algorithme suivant de factorisation.

3.2 ALGORITHME DE FACTORISATION D'UNE EXPRESSION FORMELLE E

L'expression formelle E est représentée par un arbre étiqueté. Soit X l'ensemble des noeuds de E, nous définissons entre ces noeuds des relations d'équivalences, noté $\tilde{n}$, défini comme suit :

I. Soit (j, j') un couple de variables de X :

$$j \overset{\sim}{0} j' \iff L(j) = L(j')$$

II. Soit (j, j') un couple d'opérateurs de X :

$$j \tilde{n} j' \iff \begin{array}{ll} \text{(i)} & L(j) = L(j') \\ \text{(ii)} & d\ \inf(j) = d\ \inf\ (j') = k \\ \text{(iii)} & \exists\ i \in (1,\ldots,h)\ \text{tq}\ j_i \overset{\sim}{n-1} j'_i \\ \text{(iv)} & \forall\ i \in (1,\ldots,k) : \\ & \qquad \exists\ p < k\ \ \text{tq}\ \ j_{i'} \tilde{p} j'_{i'} \end{array}$$

(nous notons $(j_1,\ldots j_k)$ et $(j_1,\ldots,j'_{k'})$ les successeurs immédiats respectifs des noeuds j et j').

□ Lemme préliminaire : Si deux noeuds i et i' sont équivalents modulo n, il existe, dans l'arbre associé à l'expression formelle E, deux sous-arbres de profondeur n et de racines i et i' identiques.

Preuve : La démonstration se fera par induction sur n.

(1) $n = 0 : i \overset{\sim}{0} i'$

Les noeuds i et i' sont des variables et L(i) = L(i'). Les deux sous-arbres se réduisent à deux feuilles étiquetées par L(i).

(2) $n = 1 : i \overset{\sim}{1} i'$

D'où a - $i, i' \in X_1$

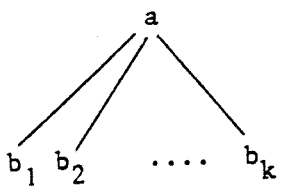
b - $L(i) = L(i')$

c - $dinf(i) = dinf(i') = k$

d - $\forall m \in \{1,\ldots,k\} : j_m \overset{\sim}{0} d'_m$,

(si nous notons $(j_1,\ldots,j_k) = S(i)$, $(j'_i,\ldots,j'_1) = S(i')$)(1)

Le cas trivial (1) entraîne : $\forall m \in \{1,\ldots,h\} : L(j_m) = L(j'_m,)$. Les deux sous-arbre sont de la forme :



et sont étiquetés par $L(i)$, $L(j_1)$..... $L'j_n)$.

Considérons les deux sous-arbres respectivement de racines de i et i', nous avons : $\forall r \in \{1,\ldots,k\} : \exists\, p \leq m : j_r, \tilde{p} j'_r,$; les k couples de sous-arbres de racines respectives j_r et j'_r, sont identiques. Puisque $L(i) = L(i')$ $dinf(i) = dinf(i')$, les deux sous-arbres de racines i et i' sont identiques.

cqfd

□ Construction de $X/\tilde{n}$, connaissant $X/\overset{\sim}{n-2},\ldots, X/\overset{\sim}{n-2}, X/\overset{\sim}{0}$

Puisque, par définition de la relation d'équivalence modulo n, deux noeuds i et i' équivalents modulo n possédent deux successeurs immédiats j et j' de même rang équivalents modulo n-1, pour construire $X_1/\tilde{n}$, il suffit d'appliquer l'algorithme suivant :

(1) Remarque : S(i) représente l'ensemble des successeurs immédiats du noeud i.

pour tous les couples j, j' tq $j \underset{n-1}{\sim} j'$ ex P(j) = i et P(j') = i' faire

si L(i) ≠ L(i') alors ¬(i $\underset{n}{\sim}$ i') (définition de $\underset{n}{\sim}$)

sinon si dinf(i) ≠ dinf(i') alors ¬(i $\underset{n}{\sim}$ i') (lemme précédent)

sinon si prof(i) ≠ prof(i') alors ¬(i $\underset{n}{\sim}$ i') (lemme précédent)

sinon si prof(i) ≠ n alors ¬(i $\underset{n}{\sim}$ i') (lemme précédent)

sinon pour r variant de 1 à deg inf (i) faire
si (j_r, j'_r) ∉ même classe d'équivalence alors ¬(i $\underset{n}{\sim}$ i')
(définition de $\underset{n}{\sim}$) fait fait

Remarques : 1 - Par abus d'écriture, nous noterons prof(i) la profondeur du sous-arbre de racine i.

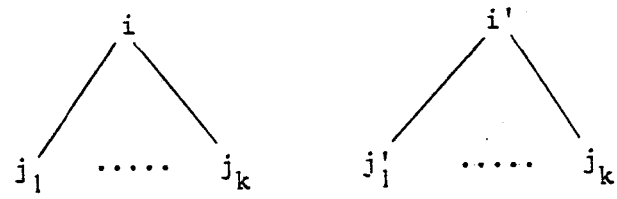
2 - Pour examiner tous les couples (j, j') d'une classe d'équivalence modulo n, il suffit de numéroter injectivement les éléments de cette classe : $(a_1,\ldots,a_n)$ et l'algorithme portera sur les couples (a_1, a_2), $(a_1, a_3),\ldots,(a_1, a_n)$ puis sur les couples $(a_2, a_3),\ldots,(a_2, a_n)$, et ainsi de suite. Il y aura donc $\frac{n.n+1}{2}$ examens pour une classe de n éléments.

□ Lemme : L'algorithme de construction des classes d'équivalences mod n pour n variant de 1 à prof (E), d'une expression formelle E (dont nous notons prof(E) la profondeur de l'arbre associé) détecte tous les sous-arbres identiques.

Preuve : Soit une expression formelle E dont l'arbre associe possède deux sous-arbres identiques de racines i et i' et de profondeur n. Montrons par induction sur n que ces deux sous-arbres appartiennent à la même classe d'équivalence.

(i) $n = 0$ les sous-arbres sont des feuilles et ce cas est trivial

(ii) $n = 1$ les deux sous-arbres de racines i et i' sont de la forme :



avec $k = \deg \inf(i) = \deg \inf(i')$

$L(i) = L(i')$

$L(j_r) = L(j'_r)$ $\quad \forall r \in \{1,\ldots,k\}$ et

(j_r, j'_r) sont des variables $\forall r \in \{1,\ldots,h\}$

$L(j_1) = L'd'_1)$,

j_1, j'_1 sont des variables $\Rightarrow j_1 \overset{\sim}{0} j'_1$

L'examen de la classe d'équivalence à laquelle appartiennent j_1 et j'_1 détectera l'identité des deux sous-arbres i et i'.

(iii) $n = \ell$ Supposons que l'algorithme détecte tous les sous-arbres identiques de profondeur inférieur à ℓ.
Soient deux sous-arbres de profondeur ℓ, ils possèdent deux successeurs immédiats j et j' de même rang et racines de deux sous-arbres de profondeur $\ell-1$ identiques. L'hypothèse de récurrence entraîne que lors de la construction de $X/\overset{\sim}{\ell-1}$, j et j' aient été placés dans la même classe d'équivalence. De même pour tous les autres successeurs immédiats des noeuds i et i'. L'examen des éléments de la classe d'équivalence modulo $\ell-1$ à laquelle appartiennent j_1 et j_2 détectera l'identité des deux sous-arbres i et i'.

□ Algorithme de factorisation d'une expression formelle et construction du cablage réduit.

Soit une expression formelle E, nous pouvons lui associer un arbre étiqueté, et un cablage. Nous nous proposons de fournir un algorithme de factorisation de cette expression formelle et d'en déduire un algorithme de construction du cablage réduit.

(i) algorithme de factorisation d'une expression formelle E

Soit X l'ensemble des noeuds de l'arbre étiqueté représentant l'expression formelle E. Nous noterons $(S_1,\ldots,S_m)$ les éléments d'une classe d'équivalence de $X/\tilde{n}$ et $(p_2,\ldots,p_m)$ l'ensemble des prédécesseurs immédiats respectifs des noeuds $(S_2,\ldots,S_m)$. L'algorithme de factorisation s'écrit :

Pas 1 : Construire les classes d'équivalences de $X/\overset{\sim}{0}$
n = 0

Pas 2 : tant que ∃ une classe d'équivalence modulo n possédant plus d'un élément
faire
pour chaque classe d'équivalence de $X/\tilde{n}$, possédant plus d'un élément
faire
. introduire une nouvelle variable W_{n_j}
. pour tout élément s_i de cette classe d'équivalence
faire substituer, dans l'arbre E, au sous-arbre de racine s_i, la variable W_{n_j}.
fait
. introduire l'équation $W_{n_j} = S$, où S représente l'expression formelle associée au sous-arbre de racine S_1.

fait
n = n + 1
construire $X/\tilde{n}$
fait

(ii) algorithme de construction du cablage réduit.

Nous considérons l'arbre associé à l'expression E, comme un graphe. L'algorithme précédent s'écrit :

Pas 1 : . Construire les classes d'équivalences de $X/\overset{\sim}{0}$
. n = 0

Pas 2 : tant que $\exists$ une classe d'équivalence modulo n possédant plus d'un élément
faire
pour chaque classe d'équivalence de $X/\tilde{n}$, possédant plus d'un élément
faire
pour tout élément s_i de cette classe d'équivalence
faire remplacer l'arc (s_i, p_i) par l'arc (s_1, p_i)
fait
fait
n = n + 1
construire $X/\tilde{n}$
fait

□ Proposition : L'algorithme de factorisation d'une expression formelle E est fini et le nombre d'itérations sera au plus de n, où n est la profondeur de l'arbre associé à l'expression E.

Preuve :

(i) La construction des classes d'équivalence modulo k de l'ensemble X, des noeuds de l'arbre E, est fini puisque cet ensemble X est fini.

(ii) Supposons qu'il existe deux noeuds i et i' équivalent modulo S avec S > prof(E).(H).

Puisque i est un noeud de l'arbre E, la profondeur du sous-arbre de racine i est inférieure ou égale à prof(E) :
prof(i) ≤ prof(E) (1)
Puisque $i \tilde{s} i'$, prof(i) = s'2) (lemme préliminaire)
L'hypothèse (H) est absurde, il n'existe donc pas de classes d'équivalence modulo s, pour s > prof(E).
Le nombre d'opération de construction de classes d'équivalence est borné par prof(E).

MAGMOÏDE TYPE DES OBJETS FONCTIONNELS

4.0 La structure algébrique du magmoïde nous permet de traiter formellement le problème du schéma de cablage mais limite le domaine d'interprétation à un ensemble unique.

Ceci est suffisant si le domaine de calcul est homogène (exemple : ensembles $\mathbb{R}$, $\mathbb{Z}$, champs de Gallois modulo n correspondant respectivement aux domaines de calcul du continu, du discret et de la logique séquentielle), mais, de plus en plus, dans la simulation nous avons affaire à un domaine "composite" formé de n ensembles disjoints comme nous le montre les deux exemples traités en annexe du chapitre 1.

Lorsque le domaine d'interprétation D est formé de la réunion de n ensembles disjoints E_i, les entrées et les sorties d'un opérateur fonctionnel appartiendront à des ensembles E_i quelconques. Pour pouvoir contrôler plus efficacement un schéma de cablage, il sera nécessaire de préciser le type des paramètres et des résultats d'un opérateur fonctionnel c'est à dire l'ensemble E_i auquel ils appartiennent. De même, nous devions typés les objets mémoires, les assignations, ...

Ceci n'est pas sans rappeler les langages de programmation récents comme(Algol 60, Algol 68, Pascal ...) où la notion de mode permet de particulariser le traitement des objets informatiques et où les procédures, correspondent à nos opérateurs. Ces procédures nécessitent que l'utilisation précise le mode des paramètres formels afin d'offrir au compilateur le moyen de contrôler le mode des paramètres effectués et de permettre un traitement efficace.

Ces remarques nous ont amenés à étendre la structure du magmoïde en magmoïde typé.

4.1 MAGMOIDE TYPE

Soit Σ un alphabet fini, notons Σ^* le monoïde libre sur Σ.
Un magmoïde typé est un sextuplet $<M, \Sigma, ., \otimes, \hat{e}, e_o>$, où M est un ensemble, Σ un alphabet fini, . une opération binaire partielle sur M, $\otimes$ une opération binaire sur M, e_o un élément distingue de M, $\hat{e}$ un ensemble d'éléments distingués de M, qui vérifie les axiomes suivants :

M1. $\forall\, p \in \Sigma^*$, $\forall\, q \in \Sigma^*$, il existe une partie M_q^p de m, appelée fibre p-q de M. Les fibres de Msont disjointes deux à deux et M est la réunion de toutes ses fibres.

M2. Produit de composition :
$\forall\, p \in \Sigma^*$, $\forall\, q \in \Sigma^*$, $\forall\, p' \in \Sigma^*$, $\forall\, q' \in \Sigma^*$, $\forall\, m \in M_q^p$, $\forall\, m' \in M_{q'}^{p'}$:
m.m' est défini ssi $q = p'$
on a alors $m.m' \in m_{q'}^{p}$.

M'2. Le produit de composition est associatif

M3. Produit tensoriel :
$\forall\, p \in \Sigma^*$, $\forall\, q \in \Sigma^*$, $\forall\, p' \in \Sigma^*$, $\forall\, q' \in \Sigma^*$:
$\forall\, m \in M_q^p$, $\forall\, m' \in M_{q'}^{p'}$, $m \otimes m' \in M_{q.q'}^{p.p'}$

M4. $\forall\, m_1, m_2, m'_1, m'_2 \in M$ si $(m_1.m_2) \otimes (m'_1.m'_2)$ est défini, alors cette quantité est égale à $(m_1 \otimes m'_1).(m_2 \otimes m'_2)$

M5. Eléments neutres :
. $e_o \in M_1^1$,
$\forall\, \hat{e}_i \in \hat{e}$, $\exists\, a \in \Sigma$ tq $\hat{e}_i \in M_a^a$ ..
notons $\hat{e}_a$, l'élément de $\hat{e}$ appartenant à M_a^q.

. En posant, pour $p = a_1.a_2 \ldots a_k$, $a_i \in \Sigma$.
$\hat{e}_p = \hat{e}_{a_1}.\hat{e}_{a_2}.\hat{e}_{a_3}. \ldots . \hat{e}_{a_k} \in M_p^p$

. on a : $\forall\, m \in M_q^p$: $e_p.m = m$, et
$\forall\, m \in M_q^p$: $m.e_p = m$

De plus $\forall\, m \in M_q^p$: $e_o \otimes m = m \otimes e_o = m$

Proposition : Les éléments neutres d'un magmoïdes typé, $<M, \Sigma, ., \otimes, \hat{e}, e_o>$, sont uniques dans chaque fibre M_q^p.

Proposition : Si $<M, \Sigma, ., \Theta, \hat{e}, e_o>$ est un magmoïde typé, la restriction à une lettre de Σ de ce magmoïde typé est un magmoïde.
(Nous n'envisagerons que les fibres p-q telles que $p = a^i$, $q = a^j$, si nous restreignons l'alphabet Σ à la seule lettre a)

4.2 EXEMPLE DE MAGMOÏDE TYPE : LE MAGMOÏDE TYPE DES FONCTIONS

Soit un ensemble E tel que $E = E_1 \cup E_2 \cup \ldots \cup E_n$. Le magmoïde typé des fonctions ; $\tilde{FT}(E)$ sera défini par :

i. $\Sigma = \{1, 2, \ldots, n\}$,
$\forall\, p \in \Sigma^* : E^{[u]} = E_{u_1} \times E_{u_2} \times \ldots \times E_{u_n}$

pour $u = u_1\, u_2 \ldots u_n$ et $u_i \in \Sigma$

Exemple : Soit $E = E_1 \times E_2 \times E_3$, alors $\Sigma = \{1, 2, 3\}$
et $E^{[1213]} = E_1 \times E_2 \times E_1 \times E_3$

ii. a. L'ensemble des constantes appartenant à E_i, c'est à dire l'ensemble des applications de $E^{[1]}$ dans $E^{[x_i]}$, sera noté $\tilde{FT}(E)_1^{x_i}$

b. L'ensemble des fonctions dont les variables sont typées par p, $p \in \Sigma^*$, et dont le résultat est de typé E_i sera noté $\tilde{FT}(E)_p^{x_i}$ (ensemble des applications de $E^{[P]}$ dans $E^{[x_i]}$).

c. D'une manière plus générale, l'ensemble des fonctions dont les variables sont typées par p et dont les résultats dont typés par a, $(P, q \in \Sigma^*)$ sera noté $\tilde{FT}(E)_p^q$

d. $\tilde{FT}(E)_o^o$ contient un seul élément noté e_o.

iii. $\hat{e}$ est l'ensemble des application identiques de $E^{[P]}$ dans $E^{[P]}$, $\forall\, p \in \Sigma^*$.

iv. Si $f \in \tilde{FT}(E)_q^p$, $f' \in \tilde{FT}(E)_{q'}^{p'}$,
$f \,\Theta\, f'$ est l'application de $E^{[qq']}$ dans $E^{[pp']}$

définie par :

$$f \circledast f'(x_1, \ldots, x_q, x'_1, \ldots, x'_{q'}) = (y_1 \cdots y_p, y'_1, \ldots, y'_{p'})$$

$$\text{ssi} \quad f(x_1, \ldots, x_q) = (y_1, \ldots, y_p)$$
$$f'(x'_1, \ldots, x'_{q'}) = (y'_1, \ldots, y'_{p'})$$

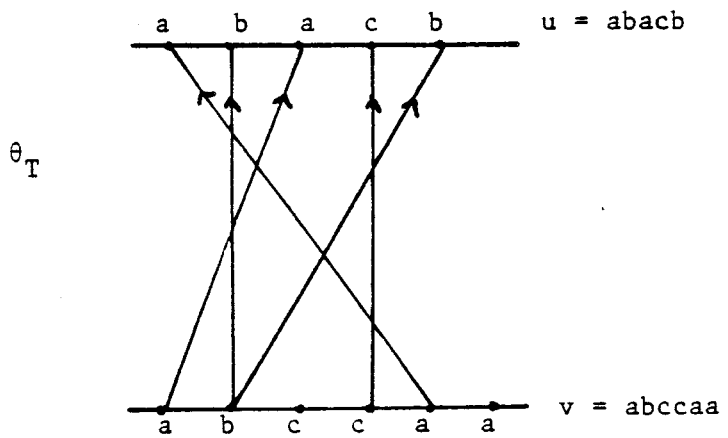
v. Si $f \in \overset{\gamma}{F}T(E)^p_q$ et $f' \in \overset{\gamma}{F}T(E)^q_r$, f.f' est l'application f o f' de $_E[r]$ dans $_E[q]$, où o est la composition usuelle des applications.

4.3 MAGMOÏDE DES TORSION TYPEES. θ_T

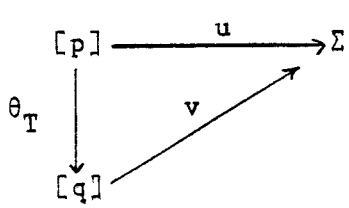
Les torsions typées sont définies, sur un alphabet Σ,

i. Concrétement : par des cablages compatibles avec les types définis par les lettres de l'alphabet Σ :

Exemple : $\Sigma = \{a, b, c\}$



ii. Par l'application θ_T de [p] dans [q] rendant commutatif le triangle suivant :



Nous avons donc :

a - $|u| = p$ et $u = u_1\ u_2 \ldots u_p$, $u_i \in \Sigma$

$|v| = q$ et $v = v_1\ v_2 \ldots v_q$, $v_j \in \Sigma$

b - $\forall\ i \in [1, p]$, si $u_i = z$ alors $v_{\theta_T(i)} = z$ (avec $z \in \Sigma$).

En reprenant l'exemple, ci-dessus, la torsion typée θ_T sera définie par :

. $\theta_T : [5] \to [6]$

. $u = a\ b\ a\ c\ b$

. $v = a\ b\ c\ c\ a\ a$

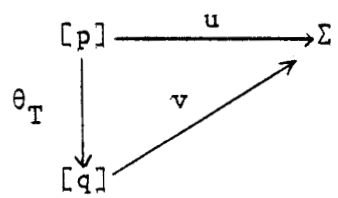
et $\theta_T(1) = 5$

$\theta_T(2) = \theta_T(5) = 2$

$\theta_T(3) = 1$

$\theta_T(4) = 4$

Posons $\theta_{T_v}^{u}$ égal à l'ensemble des applications θ_T de [p] dans [q] rendant commutatif le triangle :



avec $p = |u|$ et $q = |v|$

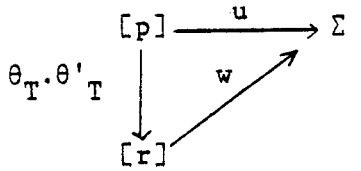
$u \in \Sigma^*$, $v \in \Sigma^*$

Puisqu'il n'existe aucune application d'un ensemble non vide dans l'ensemble vide, si $p \neq 0$ alors $\theta_{T_\emptyset}^{u}$ est un ensemble vide. D'autre part, comme il existe une seule application de l'ensemble vide dans un ensemble non vide, quel que soit le mot v : $\theta_{T_v}^{\emptyset}$ contient un seul élément noté $0_{T\ v}$.

Comme pour les torsions non typées, nous définissons

i. Le produit de composition de torsions typées comme le produit de composition des applications respectant les types,

si $\theta_T \in \Theta_{T_v}^{u}$ et $\theta'_T \in \Theta_{T_v}^{w}$, $\theta_T.\theta'_T$ est l'application de [p] dans [r] rendant commutatif le triangle.

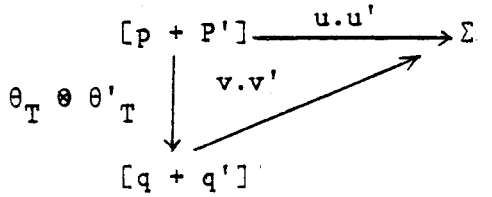


avec u, v, w $\in \Sigma^*$

$p = |u|$, $q = |v|$, $r = |w|$

ii. De même, nous définissons le produit tensoriel par :
si $\theta_T \in \Theta_{T_v}^{u}$ et $\theta'_T \in \Theta^{u'}_{v'}$, alors $\theta_T \otimes \theta'_T$

est l'application de [p + p'] dans [q + q'] rendant commutatif le triangle



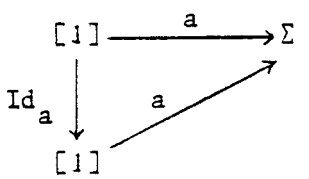
avec $p = |u|$, $p' = |u'|$
$q' = |v|$, $q' = |v'|$

et qui vaut $- \theta_T(i)$ si $i \leq p$

$- q + \theta'_T(i - p)$ si $p < i \leq p + p'$

iii. Nous posons $\hat{e}_a \in \Theta_{T_a}^{a}$ comme l'unique application de [1] dans [1]

rendant commutatif le triangle :



et que nous écrirons Id_a.

Nous posons $e_{T_1} = 0_1$ l'unique élément de $\Theta^1_{T_1}$ que nous écrirons Id_1.

D'après la définition du produit tensoriel $e_u \in \Theta^u_u$ est l'application identique de [p] dans [p] (où p = |u|) que nous noterons Id_u.

Lemme : $\Theta_T = \langle m, \Sigma, \theta, ., e_{T^1}, \hat{e} \rangle$ est un magmoïde typé

$(\hat{e} = \{e_{T^u} \mid u \in \Sigma^+\})$

Parmi les torsions typées nous retrouvons les torsions particulières :

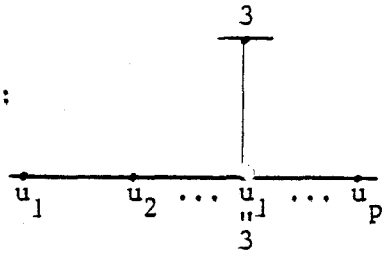
- les zéros types : 0_u
- les identités typées : Id_{u_i}
- les projections typées : π^i_u

Les projections typées sont définies comme suit :

$\pi^i_u : [1] \to [p]$ où p = |u|

et $\pi^i_u(1) = i$

π^i_u :



$(u = u_1\ u_2 \ldots u_p)$

4.4 MAGMOÏDE TYPE DES OBJETS FONCTIONNELS

L'introduction de la notion de type permet d'étendre le magmoïde des objets fonctionnels décrit par Jacob [J].

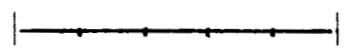
Considérons un alphabet de type Σ, et le monoïde associé que nous noterons Σ^*.

Nous définirons successivement l'objet mémoire typé, l'assignation typée et l'objet opérateur typé.

i. Mémoire typée

L'objet mémoire du magmoïde typé des objets fonctionnels doit être compatible avec les types représentés par un mot u du monoïde Σ^*. Il sera décrit par l'une des écritures suivantes équivalentes :

. $\langle u ; x_1, x_2, \ldots, x_p \rangle$ où $u \in \Sigma^*$ et le type de x_i étant u_i

.  $\langle u \rangle$: mémoire de degré 4 typé par u.

Le produit tensoriel de deux objets mémoires typées doit respecter le type et le degré de chaque objet mémoire typé.

Considérons deux objets mémoire typée : $\langle u ; x_1, \ldots x_p \rangle$ et $\langle v ; x_1, \ldots x_q \rangle$, que nous écrirons $\langle u \rangle$ et $\langle v \rangle$. Alors $\langle u \rangle \otimes \langle v \rangle$ est de degré p + q, si p et q sont les degrés respectifs des mémoires $\langle u \rangle$ et $\langle v \rangle$.

Le produit tensoriel des deux mémoires $\langle u \rangle$ et $\langle v \rangle$ devant respecter les types nous aurons la relation :

$$[p + q] \xrightarrow{u.v} \Sigma$$

si $[p] \xrightarrow{u} \Sigma$

et $[q] \xrightarrow{v} \Sigma$

D'où finalement, le produit tensoriel des deux objets mémoires $\langle u ; x_1, x_2, \ldots x_p \rangle$ et $\langle v ; x_1, \ldots, x_q \rangle$ sera défini par $\langle u.v ; x_1, x_2, \ldots, x_{p+q} \rangle$.

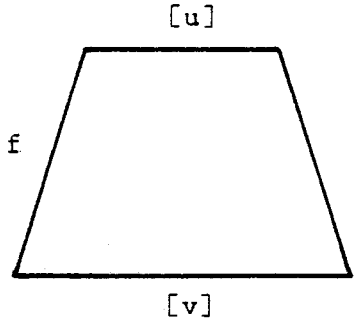
ii. Symbole fonctionnel typé

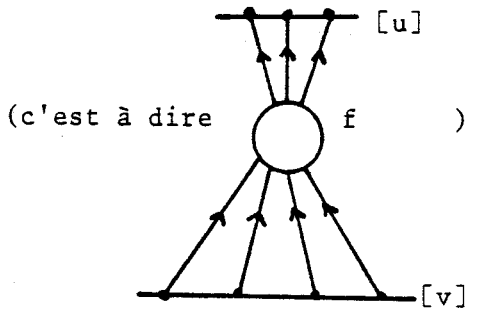
Considérons deux mots du monoïde Σ^* :

$$u = u_1 u_2 \ldots u_p \quad \text{et}$$

$$v = v_1 v_2 \ldots v_q$$

Un symbole fonctionnel f de type $\binom{[u]}{[v]}$, d'arité inférieure q et d'arité supérieure p sera décrit par





Nous pouvons définir les produits tensoriels et de composition des symboles fonctionnels types.

. Produit tensoriel

Soient f et g, deux symboles fonctionnels typés,

f de type $\binom{[u]}{[v]}$ et

g de type $\binom{[u']}{[v']}$

alors f ⊗ g est de type $\binom{[u,u']}{[v,v']}$, d'arité supérieure p + p' et

d'arité inférieure q + q'.

. Composition de symboles fonctionnels,

Cette opération permet de composer en série les objets décrits plus haut. Elle ne peut se faire que si les deux objets considérés peuvent se "souder" en identifiant la mémoire typée de "sortie" du premier avec la mémoire d'entrée du second.

Soient f et g deux symboles fonctionnels types,

f de type $\binom{[u]}{[v]}$

et g de type $\binom{[v]}{[w]}$

alors f.g est de type $\binom{[u]}{[w]}$

. Objets fonctionnels typés et magmoïde typé des objets fonctionnels

Un objet fonctionnel typé est défini par composition de symboles fonctionnels typés à l'aide des deux opérations définies plus haut.

L'ensemble $\bar{F}$ des objets fonctionnels typés muni des deux opérations de produit tensoriel et de produit de composition est le magmoïde typé des objets fonctionnels et sera noté $\tilde{T}_t(F)$.

iii. Domaine typé

Si la cardinalité de l'ensemble des types Σ, vaut n, alors le domaine de calcul D est formé de n ensembles disjoints D_i.

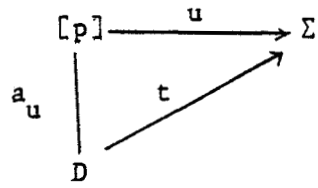
Nous écrirons $D = D_1 \cup D_2 \cup D_3 \ldots \cup D_n$, si $\Sigma = \{1, 2, \ldots, n\}$

Nous introduirons l'application t qui à un élément x d'un ensemble D_i associe son type i :

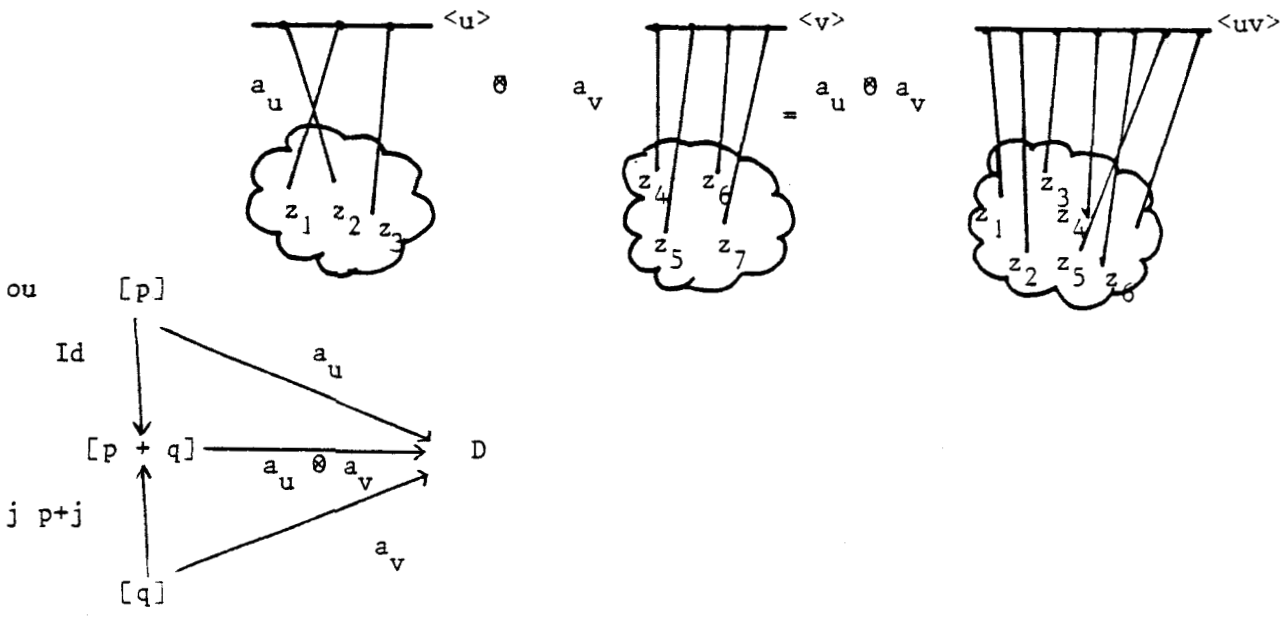
$$t : D \to \Sigma$$

iv. Assignation typée

L'assignation typée d'un objet mémoire typée $p : \langle u ; x_1, \ldots x_p \rangle$ sera définie par l'application a_u de [p] dans D respectant les types, c'est à dire rendant commutatif le triangle suivant :



Nous pouvons introduire le produit tensoriel des applications typées :

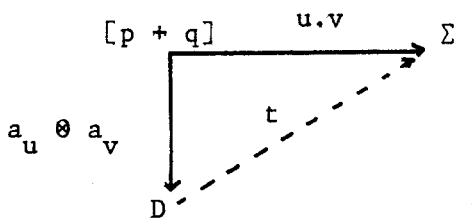


L'application $a_u \otimes a_v : [p + q] \to D$ est définie par

$(a_u \otimes a_v)\ (i) = a_u(i)$ si $i \in [p]$

$(a_u \otimes a_v)\ (p + j) = v(j)$ si $j \in [q]$

Cette application respecte les types, puisque a_u et a_v sont deux assignations typées, ce qui se traduit par le triangle suivant :



v. Interprétation typées à valeur dans D

Nous appellerons interprétation du magmoïde fonctionnel $\tilde{T}_t(F)$
la donnée I = (D, *)
d'un domaine typé D
d'une application à gauche * de $\tilde{T}_t(F)$ sur les assignations typées à valeur dans D.

BIBLIOGRAPHIE DU CHAPITRE 1

[A] ARNOLD A.
"Systèmes d'équations dans le magmoïde. Ensembles rationnels et algébriques d'arbres" ; Thèse es Sc. Math., Lille I, 1977.

[AD1] ARNOLD A. et DAUCHET M.
"Théorie des magmoïdes" ; Introduction commune aux thèses es Sc. Math. d'ARNOLD et de DAUCHET. Lille I, 1977.

[AD2] ARNOLD A. et DAUCHET M.
"Théorie des Magmoïdes" ; Colloque de Lille "Les arbres en algèbre et en programmation", 1976.

[AD3] ARNOLD A. et DAUCHET M.
"Bimorphismes de Magmoïdes" ; Colloque de Lille "Les arbres en algèbre et en programmation ", 1976.

[B] BORNE P., GENTINA J.C., LAURENT F., TOULOTTE J.M.
"Extension à un corps quelconque des méthodes de simulations usuelles" ; 7ème AICA Congrès Prague Aout 1973 (pp 26-31).

[B1] BORNE P., GENTINA J.C., LAURENT F.
"On the coding of information and modelisation of complex systems"
1st World Conference on mathematics at the service of Man
Barcelone 1977

[B2] BOUDAREL R., DELMAS J., GUICHET P.
"Commande optimale des processus" Tome 1.
Dunod Paris 1967

[C] CORBEEL D., GENTINA J.C.
"Formal descriptions of processes : application to simulation" ;
UKSC Conference on computer simulation. Chester Avril 1978.

[D] DAUCHET M.
"Transductions de forêts. Bimorphismes de magmoïdes".
Thèse es Sc. Math. , Lille I Juin 1977.

[G] GENTINA J.C.
"Sur la notion de modèle dans la description d'un processus"
Colloque ANRT. Ecole Centrale Paris Juin 1975

[J] JACOB G.
"Substitution dans les arbres et non déterminisme. Appel par nom et appel synchrone" ; 2ème colloque de Lille : "Les arbres en algèbre et en programmation" 1977.

[V] VOGEL T.
"Théorie des sytèmes évolutifs" Ed. Gauthier-villars 1965.

ANNEXE DU CHAPITRE I

UTILISATION DU MAGMOÏDE TYPE DES OPERATEURS FONCTIONNELS EN SIMULATION ANALOGIQUE ET HYBRIDE

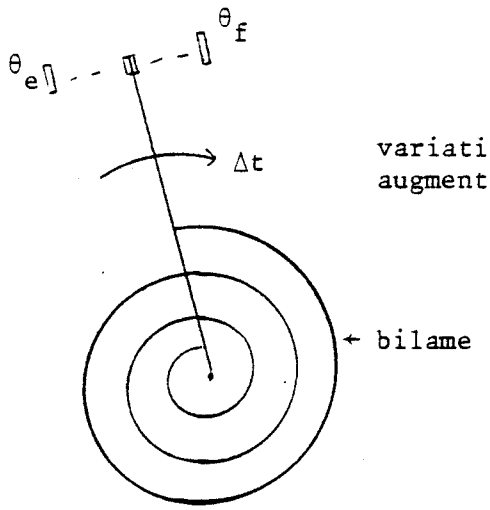
Les calculatrices analogiques et hybrides permettent, grace au cablage des opérateurs, d'appréhender le parallèlisme d'exécution. Nous nous proposons, dans cette annexe, de décrire formellement le schéma de cablage associé à certaines simulations.

Cette description formelle est le premier pas vers un cablage automatique de ce type de calculatrice.

1. SIMULATION DU CYCLE D'HYSTERESIS CARRE

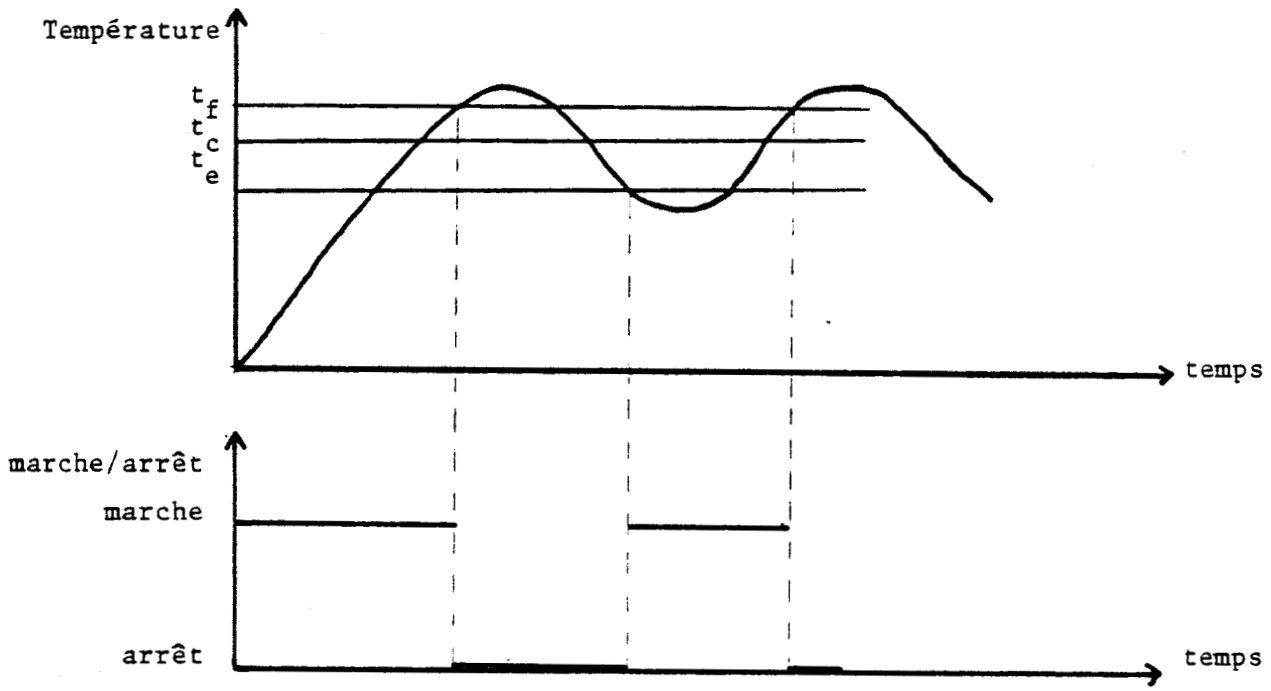
1.1 Exemple introductif de l'hystérésis carré : modélisation du thermostat d'un four.

Le thermostat d'un four est généralement constitué par un capteur dont le principe est le suivant : il utilise la dilatation d'un bilame sous l'influence d'une variation de température. Le déplacement de l'extrémité libre du bilame commande la rotation d'une aiguille dont l'extrémité ferme le contact soit de la mise en marche, soit de l'arrêt du four.

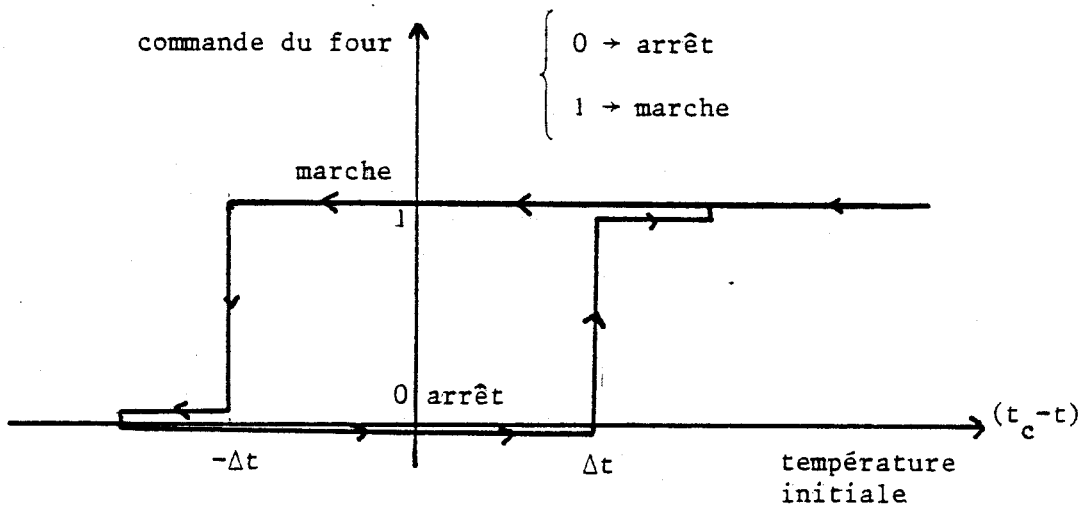


La position moyenne de l'aiguille correspond à la température de consigne t_c c'est à dire à la température souhaitée par l'utilisateur. Les positions extrêmes correspondent respectivement à une variation de $\pm \Delta t$ (20°C pour une température de consigne de 500°C par exemple).

La position θ_e de l'aiguille correspond à la température $t_c - \Delta t$ et commande la mise en marche du four, la position θ_f de l'aiguille associée à la température $t_c + \Delta t$ commande l'arrêt du four. Nous pouvons dresser les courbes fournissant la température intérieure du four et la commande de marche-arrêt du four en fonction du temps.

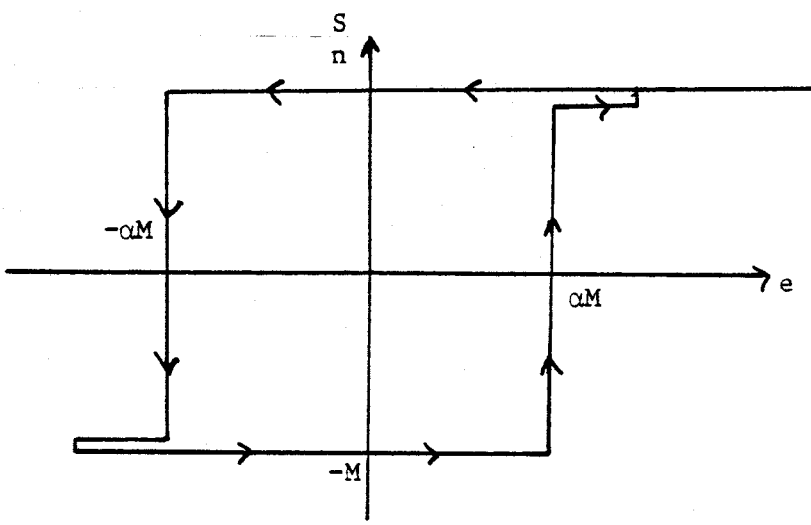


Si nous représentons la commande marche-arrêt du four fonction de la variation de la température t du four autour de la température de consigne t_c nous obtenons un hystérésis carré.



1.2 Simulation du cycle d'hystérésis carré.

Nous désirons simuler le processus ayant pour entrée la variation de température $(t_c - t)$ et pour sortie la commande de marche-arrêt du four. Le modèle de ce processus correspond à un hystérésis carré :



Dans un but d'unification des opérateurs on représente cette caractéristique centrée. A ce modèle peut être associé le schéma de cablage et l'interprétation suivante :

1) $\langle S\rangle = \langle f.(M_1 \otimes M_2 \otimes Id_1)\rangle \langle 2\rangle \langle S, L, V\rangle$

$\langle L\rangle = \langle f.(e_1 \otimes Id_1)\rangle \langle 3\rangle \langle S, L, V\rangle$

$\langle V\rangle = \langle h.(A \otimes Id_1)\rangle \langle 1\rangle \langle S, L, V\rangle$

avec $f \in F^{a}_{a^2b}$

$g \in F^{b}_{a^2}$

$h \in F^a_{a^2}$

$M_1, M_2, A \in F^a_o$

2) Interprétation

Le domaine de calcul D = a ∪ b, ayant pour interprétation

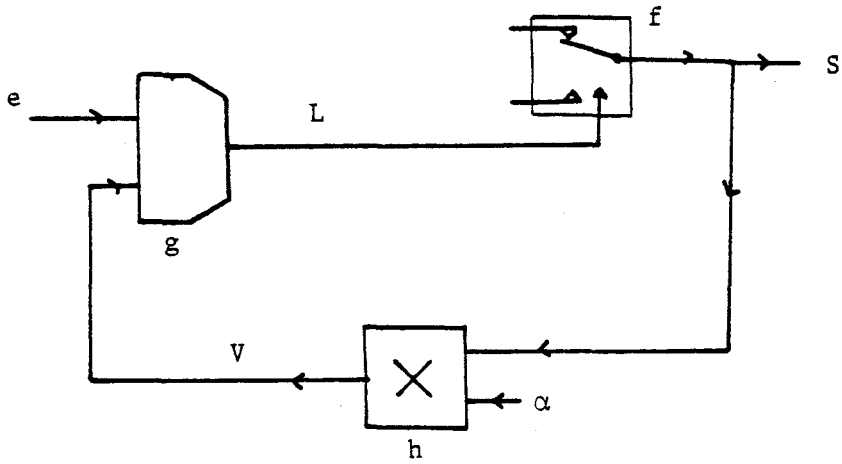
a = $\mathbb{R}$

b = CG(2) (booléen)

L'interprétation des opérateurs :

f	relai
g	comparateur
h	multiplieur
M_1	constante M
M_2	constante -M
A	constante α

Pour mémoire, voici le cablage de la calculatrice hybride



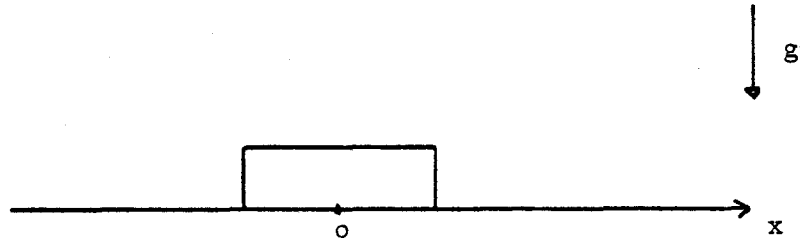
2. SIMULATION D'UN RELAI TOUT OU RIEN AVEC SEUIL

2.1 Exemple introductif au relai tout ou rien avec seuil :

Nous illustrons le modèle du relai tout ou rien avec seuil par un exemple emprunté à la mécanique statique.

Considérons un solide homogène d'angle de frottement sec ∅ reposant sur une surface horizontale. Le module de la force F horizontale

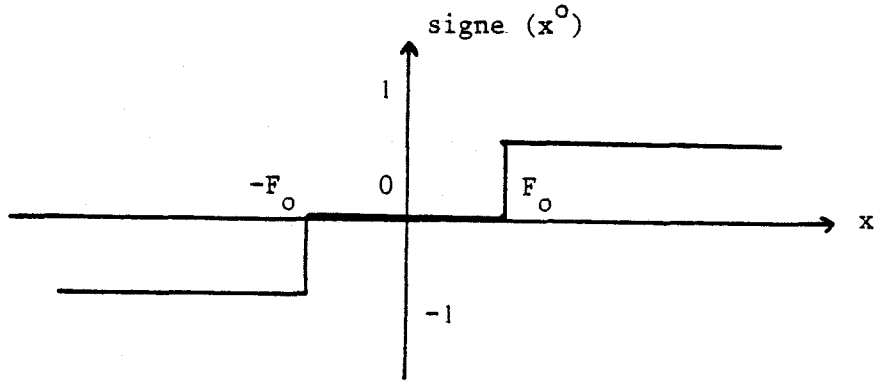
qu'il faut lui appliquer pour le déplacer doit au moins être égal à $F_o = M.g. tg(\emptyset)$ où M représente la masse du solide, g l'accélération terrestre et $tg(\emptyset)$ le coefficient de frottement sec.



Orientons l'espace du solide et étudions le signe de la vitesse du déplacement du solide en fonction de la force F appliquée sur ce solide :

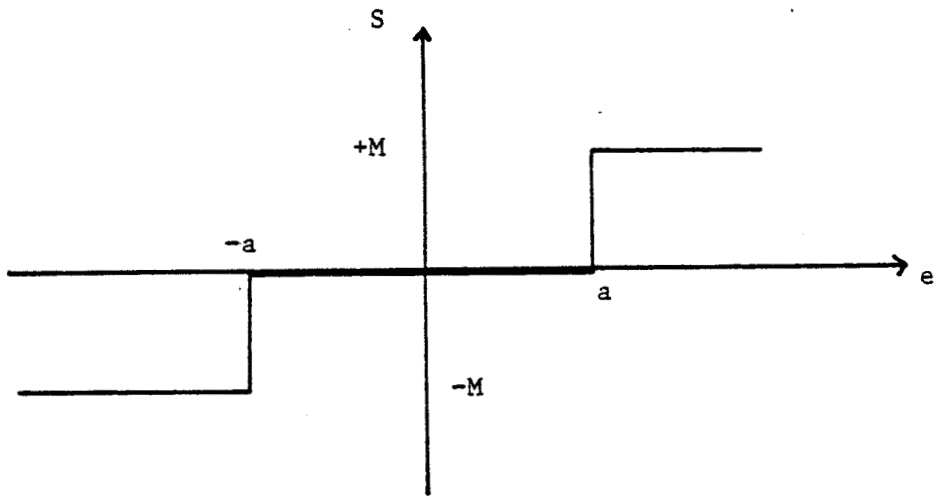
- la vitesse est positive pour $F > F_o$
- la vitesse est nulle pour $-F_o \leq F \leq F_o$
- la vitesse est négative pour $F < -F_o$

Si nous représentons le signe de la vitesse de déplacement fonction de la force de poussée F, nous obtenons une courbe appelée relai tout ou rien avec seuil



2.2 Simulation d'un relai Tout ou rien avec seuil

Nous désirons simuler le processus admettant pour entrée la face F et ayant pour sortie le signe de la vitesse de déplacement. Le modèle de ce processus correspond à un relai tout ou rien avec seuil :



Le schéma de cablage du modèle simulant le relai tout ou rien avec seuil et l'interprétation seraient :

$\langle S \rangle = \langle f.(ID_1 \otimes g.(M_1 \otimes M_2 \otimes ID_1)) \rangle \langle 2, 3 \rangle \langle S, L_1, L_2, F, E \rangle$

$\langle L_1 \rangle = \langle h.(ID_1 \otimes A_1) \rangle \langle 4 \rangle \langle S, L_1, L_2, F, E \rangle$

$\langle L_2 \rangle = \langle h.(ID_1 \otimes A_2) \rangle \langle 5 \rangle \langle S, L_1, L_2, F, E \rangle$

$\langle F \rangle = \langle k.(ID_1 \otimes f.(ID_1 \otimes k)) \rangle \langle 5, 3, 5, 5 \rangle \langle S, L_1, L_2, F, E \rangle$

avec $f \in F^a_{ba}$

$g \in F^a_{a^2_b}$

$h \in F^b_{a^2}$

$k \in F^a_{a^2}$

$M_1, M_2, A_1, A_2 \in F^a_\emptyset$

et $D = a \cup b$

L'interprétation du domaine de calcul et des divers opérateurs étant :

a : $\mathbb{R}$

b : CG(2)

f : interrupteur

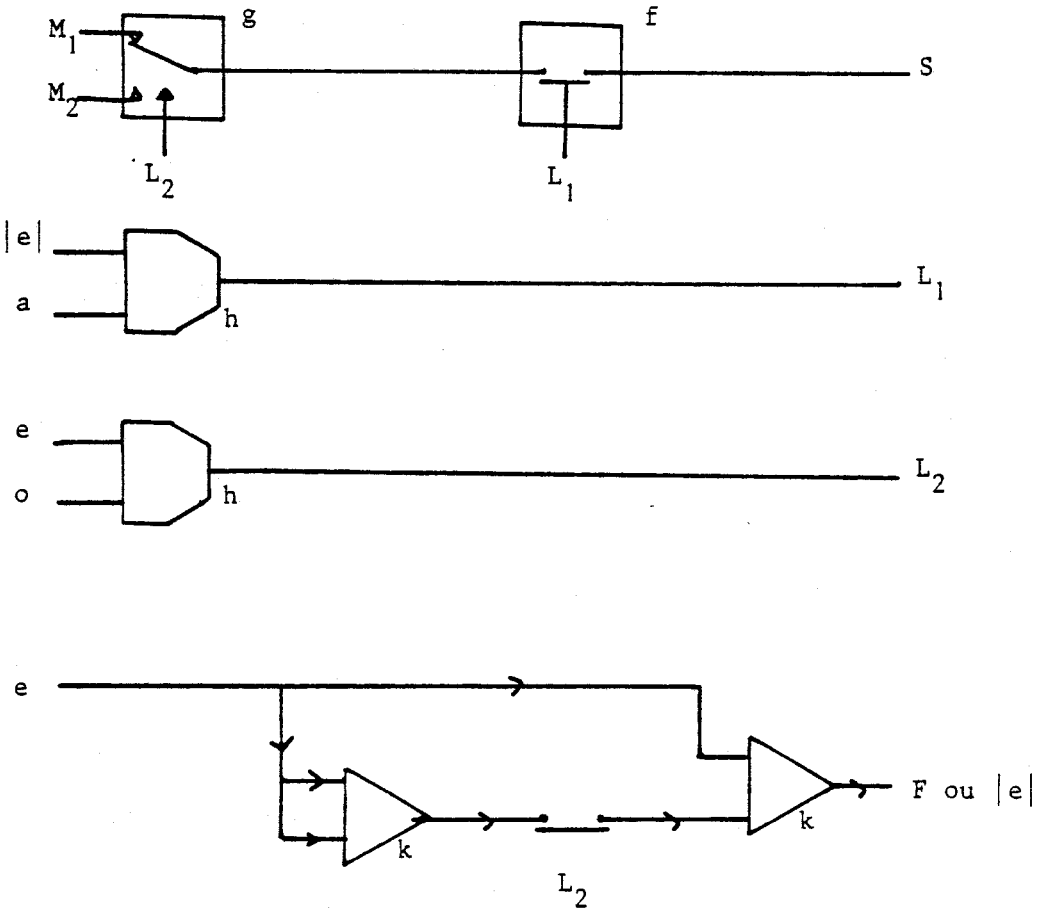
g : relai

h : comparateur

k : additionneur-inverseur

M_1 : constante M
M_2 : constante -M
A_1 : constante a
A_2 : constante 0

Pour mémoire, le cablage de la calculatrice serait :



C H A P I T R E 2

PARALLELISME DE CONTROLE

PARALLELISME DE CONTROLE

2.0 GENERALITES SUR LES RESEAUX DE PETRI

Ce chapitre contient quelques rappels succints de définition et de notations classiques concernant les réseaux de Petri (HAC 75, GIR 78, VAL 67).

2.0.1 GRAPHE DE PETRI

a - Définition : Graphe de Petri

Un graphe de Petri est un triplet $G = (P, T, \Gamma)$ où

- P est un ensemble de sommets appelés places et représentés graphiquement par des cercles.
- T est un ensemble de sommets appelés transitions et représentés graphiquement par des barres.
- Γ est la correspondance associant à un sommet ses successeurs (Γ^{-1} celle associant à un sommet ses predecesseurs)
- $\Gamma(T) \subset P$ et $\Gamma(P) \subset T$

b - Graphe de Petri particulier

Définition : Graphe d'état

Un graphe d'état est un graphe de Petri, G (P, T, Γ), tel que

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

autrement dit, tel que toute transition a une place d'entrée et une place de sortie.

c - Définition auxilliaire

Soit G = (P, T, Γ), un graphe de Petri, nous définissons alors V l'application de P x T U T x P dans N telle que V((x,y)) = 1 si $y \in \Gamma(x)$, 0 sinon.

Nous noterons V(x,y) pour V((x,y))

2.0.2 RESEAU DE PETRI (BER 78)

Définition : Un réseau de Petri est le couple $\mathcal{R} = (G, M)$ où G (P, T, Γ) est un graphe de Petri, M appelé marquage initial du réseau est une application de P dans N.

Dans le cas où G a un nombre fini de places, $|P| = n$, un marquage M de $\mathcal{R}$ est défini par la donnée d'un vecteur de N^n

Nous notons :

$$M = \begin{vmatrix} m_1 \\ m_2 \\ . \\ . \\ . \\ m_n \end{vmatrix} \qquad \text{et} \quad m_i = M(P_i)$$

Sur le graphe associé à $\mathcal{R}$, le marquage M peut être représenté par une distribution, dans les places, d'objets appelés marqueur. Une place p contient k marques schématisées par k points à l'intérieur du cercle p ssi $M(p) = k$.

Remarque : Dans la suite de notre travail, nous nous intéressons au réseau de Petri dont le graphe a un nombre fini de sommets.

2.0.3. REGLES DE DECLENCHEMENT D'UNE TRANSITION

L'une des plus intéressantes possibilités des réseaux de Petri est qu'ils permettent de représenter le comportement d'un ensemble de processus concurrents. Le marquage du réseau de Petri représente alors l'état de l'ensemble des processus et doit donc être susceptible d'évoluer. Ceci peut se faire par le déclenchement des transitions.

Une transition t est dite déclenchable pour un marquage M ssi pour tout p de $\Gamma^{-1}(t)$: $M(p) \geq 1$, autrement dit, ssi toute place d'entrée de t contient au moins une marque.

Un déclenchement δ_t associé à une transition t est une semi application de N^n dans N^n qui à un marquage M, pour lequel t est déclenchable, associé un marquage M' tel que pour tout p de P :

$$M'(p) = M(p) + V(t,p) - V(p,t)$$

Une séquence de déclenchements σ est une composition de déclenchement $\sigma = \delta_{t_n} \circ \ldots \circ \delta_{t_1}$, soit M un marquage, $\sigma(M)$ est définie ssi : t_1 est déclenchable pour le marquage M et pour tout $i > 1$, t_i est déclenchable pour le marquage $\delta_{t_{i-1}} \circ \ldots \circ \delta_{t_1}(M)$.

Nous dirons que M' est un marquage atteint à partir de M ssi il existe une séquence de déclenchements σ telle que $\sigma(M)$ soit définie et $\sigma(M) = M'$.

Nous noterons ε_M l'ensemble des marquages atteints à partir de M.

2.0.4. PROPRIETES ASSOCIEES AUX MARQUAGES D'UN RESEAU DE PETRI

204.1 Propriétés des places

Définition : Dans un réseau de Pétri, $R = (G, M_o)$, une place est k-bornée ssi :
(i) il existe M appartenant à ε_{M_o} tel que $M(p) = k$
(ii) pour tout M' appartenant à ε_{M_o} : $M'(p) \leq k$

Définition : Dans un réseau de Petri, une place sera dite bornée ssi il existe un entier k tel que cette place soit k-bornée.

Définition : Le réseau est borné (resp. k-bornée) ssi toute place est bornée (resp. k-bornée).

Définition : Une place est saine ssi elle est 1-bornée

Définition : Le réseau est sain ssi toute place est saine

204.2 Propriétés des transitions

Définition : Dans un réseau de Petri, $R = (G, M_o)$, une transition t est vivante ssi pour tout marquage M appartenant à ε_{M_o}, il existe un marquage M' appartenant à ε_M tel que t soit déclenchable à partir de M'.

Définition : Le réseau R est vivant ssi toutes ses transitions sont vivantes.

204.3 Réseau propre

Définition : Un réseau, $R = (G, M_o)$ est propre ssi pour tout marquage M appartenant à ε_{M_o}, il existe une séquence de déclenchement σ telle que $\sigma(M) = M_o$.

2.0.5. GRAPHE DE PETRI A ARCS INHIBITEURS (GRAPHE INHIBITEUR) GdPZ

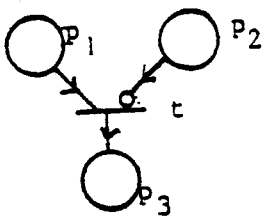
Définition : Un GdPZ est un doublet $G_z(G\ I)$ tel que :
(i) G soit un graphe de Petri : (P, T, Γ)
(ii) I est la correspondance associant à une place p les transitions qu'elle inhibe, et $\Gamma(p) \cap I(p) = \emptyset$

Définition : Une place p d'un GdPZ, est inhibée ssi $I(p) \neq \emptyset$

Représentation :

Sur la représentation graphique d'un GdPZ, nous ajouterons les arcs inhibiteurs et afin de les différencier des arcs ordinaires, nous placerons un petit cercle aux extrémités incidentes aux transitions :

exemple :



$\Gamma(p_1) = t$, $I(p_1) = \emptyset$

$\Gamma(p_2) = \emptyset$, $I(P_2) = t$

Définition : Un réseau de Pétri à arcs inhibiteurs (réseau inhibiteur, RdPZ) est le couple $R_z = (G_z, M)$ où $G_z = (G, I)$ est un graphe à arcs inhibiteurs et M le marquage initial du réseau.

Opération sur les marquages :

Définition : Une transition t est déclenchable pour un marquage M ssi

(i) pour tout p de $\Gamma^{-1}(t)$: $M(p) \geq 1$

(ii) pour tout p de $I^{-1}(t)$: $M(p) = 0$

Définition : Le déclenchement δ_t associé à une transition t déclenchable pour un marquage M est une semi-application de N^n dans N^n qui au marquage M associe de marquage M' tel que pour tout p de P :

$$M'(p) = M(p) + V(t,p)\ V(p,t)$$

Remarque : l'application V de PxTU TxP dans N est définie en 2.0.1.

Propriété d'un réseau inhibiteur (GI)

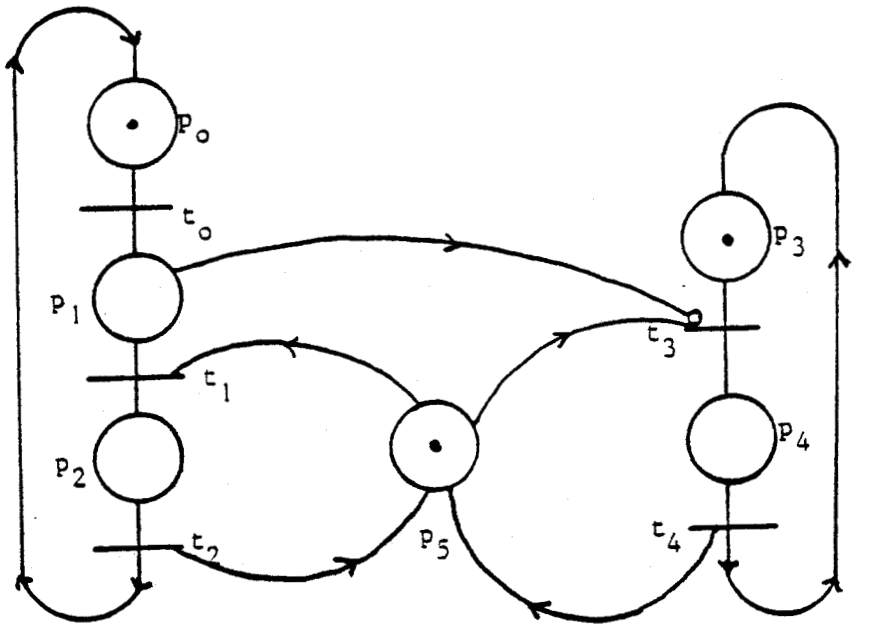
Si une place p inhibée, d'un GdPZ, est 1-bornée, nous pouvons la transformer en une place non inhibée de la manière suivante :

(i) - nous gardons les transitions du GdPZ

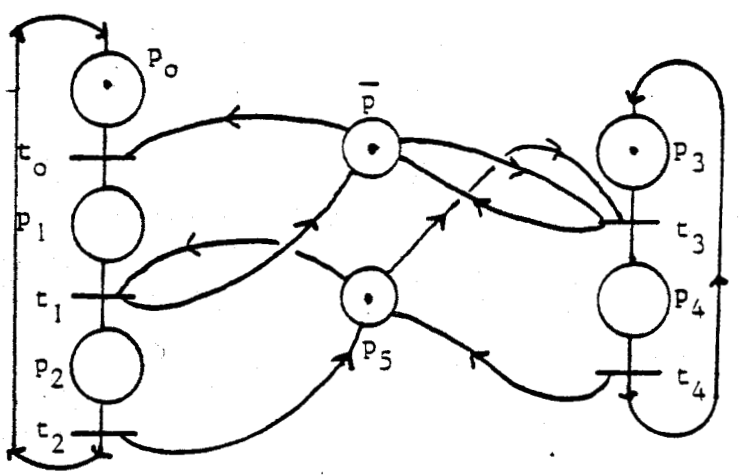
(ii) - nous adjoignons à cette place p une place $\overline{p}$ contenant toujours le nombre de marques complémentaires à 1.

(iii) - nous remplaçons l'arc inhibiteur par des arcs associés à la place $\overline{p}$

Exemple : Soit le GdPZ suivant :



La place p_1 est inhibée et 1-bornée, nous pouvons supprimer l'arc inhibiteur (p_1, t_3):



Remarque : L'algorithme précédent peut être étendu au cas d'une place inhibée k-bornée, mais nécessite la définition des réseaux de Petri généralisés

2.0.6. RESEAU DE PETRI A PRIORITES (RdPP)

Définition : Un RdPP est un couple $R_p = (R,\sigma)$ tel que :

(i) R soit un RdP

(ii) σ soit une relation d'ordre stricte sur T Si $(t, t') \in \sigma$, nous dirons sur t est plus prioritaire que t'

Opération sur les marquages:

Une transition t est déclenchable pour un marquage M ssi pour tout $p \in \Gamma^{-1}(t)$: $M(p) \geq 1$.

Le déclenchement de t défini pour M ssi t est déclenchable pour M et s'il n'existe aucune transition déclenchable pour M plus prioritaire. Son exécution s'effectue comme pour les réseaux de Petri.

Propriété :

Hack (HA) a montré l'équivalence de cette extension et de la précédente.

2.0.7. RESEAUX DE PETRI NON AUTONOMES

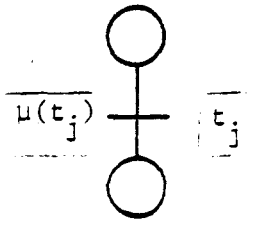
Définition : Réseau de Petri synchronisé (RdPS) (VAL 75, MOA)

Un RdPS est un triplet $R_s = (R, E, \mu)$ où :

- R est un RdP
- E est un ensemble d'événements externes
- μ est une application de l'ensemble T des transitions de R dans E U {e} , où e est l'élèment neutre de E^* appelé "événement toujours présent".

Représentation :

Nous présentons un RdPS par le RdP associé en portant sur chaque transition tj le renseignement $\mu(tj)$ où $\mu(tj)$ est l'événement associé à tj.



Opération sur les marquages :

Une transition déclenchable pour le marquage M est déclenchée sur l'événement correspondant à cette transition se produit.

2.0.8. CONCLUSION

Bien d'autres extensions ont été proposées (NOE 77, VALK 77) ainsi que des formalismes équivalents (PET 74) , nous nous sommes restreints aux seules extensions utiles pour la suite de notre travail.

2.1 PROCESSUS ET SCHEMA DE PROCESSUS

Nous allons définir formellement la notion de processus que nous rencontrons en automatique (contrôle de processus industriels) et en informatique (schéma opératoire). (CRO)

2.1.1. PROCESSUS ASSIMILE A UN RESEAU DE PETRI

D'une manière intuitive, un processus correspond à un programme séquentiel (donc à un enchaînement séquentiel de tâches). Avant de formaliser cette notion de processus, nous définirons le nombre de marque d'un sous-ensemble de place P' d'un réseau de Petri.

Le nombre de marque d'un sous-ensemble de place P' d'un réseau de Petri $R = (G, M_o)$ pour un marquage M' de ε_{M_o} représente la quantité $\sum_{p \in P'} M'(p)$ et sera noté N(P', M').

Un réseau de Petri, $R = (G, M_o)$ est un processus ssi

(i) pour toute place p de P : $|\Gamma(p)| \leq 2$

$|\Gamma^{-1}(p)| \leq 2$

(ii) pour toute transition de T : $|\Gamma(t)| = |\Gamma^{-1}(t)| = 1$ (G est un graphe d'état

(iii) $N(P, M_o) = 1$

(IV) Le graphe G est fortement connexe.

Remarque : La place p d'un processus, telle que $M_o(p) = 1$, sera notée p_{init} dans la suite de notre travail.

Proposition 1 Un processus est sain et pour tout marquage M appartenant à ε_{M_o}, nous avons N(P, M) = 1.

Preuve :

Montrons que tout marquage M appartenant à ε_{M_o}, N(P, M) = 1. Seul le déclenchement d'une transition peut modifier le nombre de marqueur du processus. Puisque le processus est un graphe d'état (condition (ii) de la définition), le déclenchement d'une transition supprime exactement un marqueur du réseau et ajoute exactement un marqueur au réseau. Ainsi le déclenchement d'une transition ne change pas le nombre de marqueur du réseau.

La condition (iii) implique que pour tout marquage M appartenant à M_o N (P, M) = 1.
Ce résultat entraîne que chaque place est 1-bornée, le processus est sain.

Proposition 2 Un processus est un réseau de Petri Propre.

Preuve :

Considérons un marquage M appartenant à ε_{M_o}, la proposition précédente entraîne l'existence d'une seule et unique place p marquée. Le graphe, définissant le processus, étant fortement connexe, il existe un chemin de la place p à la place initiale p_{init}. Le processus étant un graphe d'état, le chemin (p, p_{init}) définit une séquence de déclenchement δ_t : il suffit de choisir les transitions appartenant à ce chemin. Cette séquence de déclenchement nous conduit à un marquage M' tel que $M'(p_{init}) = 1$ et la proposition précédente entraîne que N(P, M') = 1, nous avons donc identité entre ce marquage M' et le marquage M_o.

Proposition 3 Un processus est un réseau de Petri Vivant
Cette propriété est une conséquence immédiate de la forte connexité et de la proposition 1.

2.1.2. SCHEMA DE PROCESSUS ET INTERPRETATION D'UN PROCESSUS

Définition : Réseau de Petri étiqueté

Soit σ un ensemble fini d'opérateurs.

Soit $\Sigma = \Sigma_d \cup \Sigma_f$ un alphabet tel que :

- $\Sigma_d = \bigcup_{a \in \sigma} \{\overline{a}\}$ où $\overline{a}$ est un symbole d'activation d'opérateur
- $\Sigma_f = \bigcup_{a \in \sigma} \underline{a}$ où $\underline{a} = \{a_1, \ldots, a_{k(a)}\}$ est un alphabet de terminaison d'opérateur

Un réseau de Petri étiqueté est un réseau de Petri auquel nous associons à chaque transition un symbole d'activation ou de terminaison d'un opérateur appartenant à un alphabet Σ.

Un réseau de Petri $R = (G, M_o)$, étiqueté est défini comme un triplet (R, Σ, ℓ) où :

$$\ell : T \to \Sigma \cup \{1\} \quad (1)$$

est une fonction totale et surjective dite d'étiquetage.

(1) $\{1\}$ représente le mot vide du monoide construit sur l'alphabet Σ.

Définition : Schéma de processus

Un schéma de processus est un <u>processus étiqueté</u> tel que :

(i) l'ensemble d'opérateurs σ se compose de deux sous-ensembles disjoints d'actions, A , et de test, P , respectivement de graduation 1 et 2.

$$\sigma = A \cup P$$

(ii) L'alphabet Σ se compose de quatre sous-ensembles disjoints :

$$A_d \ , \ A_f \ , \ P_d \ , \ P_f$$

Les sous-ensembles A_d et P_d correspondent respectivement au début d'action et de test, tandis que les sous-ensembles P_f et P_f définissent respectivement la fin d'action ou de test.

Nous noterons <u> l'opérateur dont u marque le début ou la fin.

(iii) 1- pour toute transition t étiquetée par une lettre du sous-ensemble A_d, nous avons :

- $|\Gamma^2(t)| = 1$
- $\ell(\Gamma^2(t)) \in A_f$
- $<\ell(t)> = <\ell(\Gamma^2(t)>$

2- pour toute transition t étiquetée par une lettre du sous-ensemble A_f, nous avons :

- $|\Gamma^{-2}(t)| = 1$
- $\ell(\Gamma^{-2}(t)) \in A_d$
- $<\ell(t)> = <\ell(\Gamma^{-2}(t))>$

(IV) 1- Une transition t est étiquetée par une lettre du sous-ensemble P_d ssi $\Gamma^2(t) = 2$, et nous avons :

- $\ell(t_1), \ell(t_2) \in P_f$ et $\ell(t_1) \neq \ell(t_2)$
  (si $\Gamma^2(t) = \{t_1, t_2\}$)
- $<\ell(t)> = <\ell(t_1)> = <\ell(t_2)>$

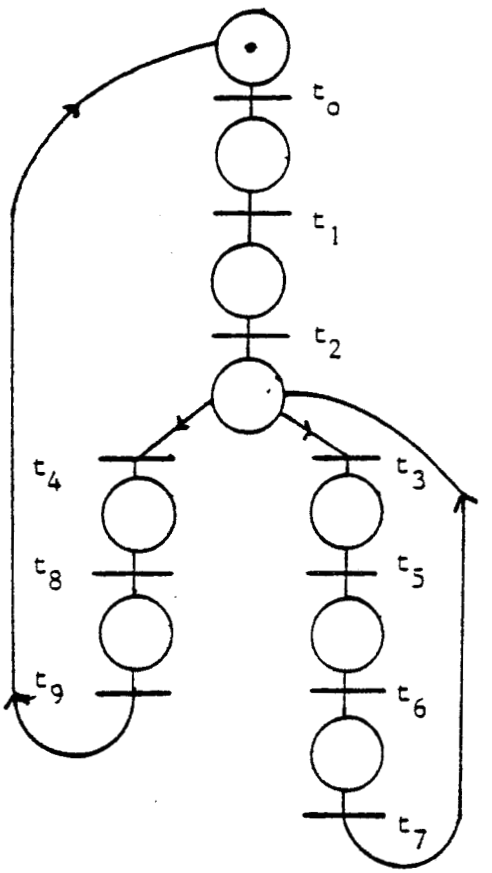
(IV) 2- pour toute transition t étiquetée par une lettre du sous-ensemble P_f nous avons :

- $|\Gamma^{-2}(t)| = 1$ notons $t_o = \Gamma^{-2}(t)$
- $|\Gamma^2(t_o) = 2$ et $\Gamma^2(t_o) = \{t, t'\}$

$- \ell(t_o) \in P_d \ , \quad \ell(t') \in P_f \ , \quad \ell(t') \neq \ell(t)$

$-<\ell(t_o)> = <\ell(t_1)> = <\ell(t_2)>$

Exemple : Soit $\mathcal{T}$ le processus défini par le réseau de Petri suivant :



Nous considérons :

(i) L'ensemble d'opérateur σ formé de deux ensembles d'actions et de tests : A et P

$\sigma = A \cup P$ avec $A = \{a,b\}$

$P = \{p\}$

(ii) Les alphabets A_d, A_f, P_d, P_f se définissant somme suit :

$A_d = \{\overline{a}, \overline{b}\}$

$A_f = \{a_1, b_1\}$

$P_d = \{\overline{p}\}$

$P_f = \{p_1, p_2\}$

(iii) et la fonction ℓ d'étiquetage

$$\ell(t_o) = \overline{a} \quad , \quad \ell(t_1) = a_1$$
$$\ell(t_2) = \overline{p} \quad , \quad \ell(t_3) = p_1 \quad , \quad \ell(t_4) = p_2 \quad , \quad \ell(t_7) = \overline{p}$$
$$\ell(t_5) = \overline{a} \quad , \quad \ell(t_6) = a_1$$
$$\ell(t_8) = \overline{b} \quad , \quad \ell(t_9) = b_1$$

Définition : Schéma de processus structuré

Nous représentons le schéma de contrôle d'un processus structuré par un réseau de Pétri étiqueté (R, Σ, ℓ)

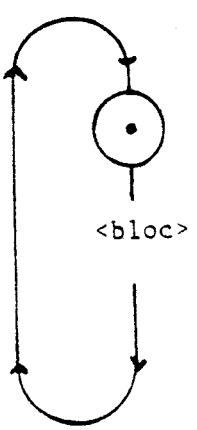
Le graphe représentant le réseau de Petri est engendré par la grammaire suivante :

Cette grammaire est formée à partir d'un vocabulaire terminal contenant notamment 4 types de transitions que nous distinguons, pour des raisons sémantiques, en associant à chacune d'elle un label :

- (t_a^a), (resp - (t_t^a)) : désigne une transition qui sera étiquetée par l'activation (resp- la terminaison) d'un opérateur d'action.
- (t_{p_a}), (resp- $(t_{p_{fa}})$) : désigne une transition qui sera étiquetée par l'activation (resp- la terminaison) d'un opérateur de test gouvernant l'exécution d'une alternative.
- (t_{p_d}), (t_{p_r}) : désignent les transitions qui seront étiquetées par l'activation d'un opérateur de test gouvernant l'exécution d'une respective.
- (t_p^f) , (t_p^v) : désignent des transitions qui seront étiquetées chacune par une terminaison différente d'un opérateur de test.

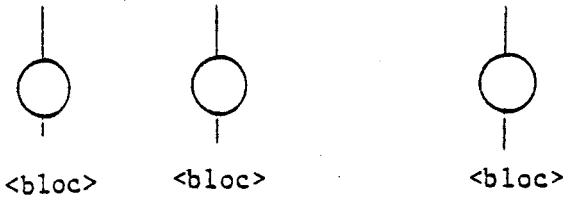
La grammaire de graphe définissant un schéma de processus structuré est alors la suivante :

< R > → <bloc>

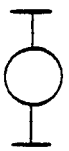


(R représente l'axiome)

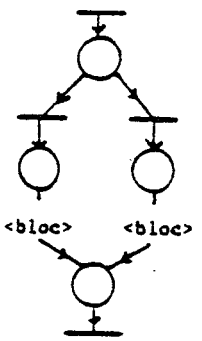
<bloc> → <action> | <alternative> | <répétitive> | <action> | <alternative> | <répétitive>



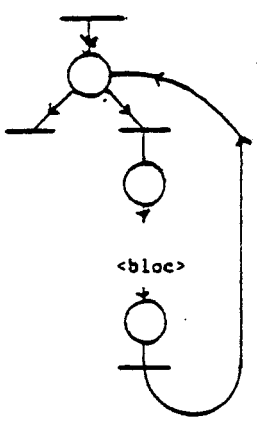
<action> →



<alternative> →



<répétitive> →



Nous pouvons nous demander si la classe des schémas de processus structurés suffit pour décrire tout schéma de processus. En fait nous n'avons introduit aucune restriction, Böhm et Jacopini ont démontré en 1966 (Böhm 66) le lemme suivant :

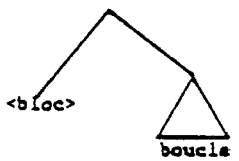
Lemme : pour tout schéma de processus, il existe un schéma de processus structuré équivalent (c.a.d. effectuant la même transformation données-résultats), et où

- les actions réalisées sont les mêmes que dans le processus initial
- les prédicats sont les mêmes augmentées éventuellement d'un certain nombre de prédicat

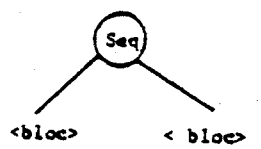
Arbre programmatique d'un schéma de processus structuré

Un schéma de processus structuré · peut-être décrit à l'aide de la grammaire précédemment définie mais également à l'aide de la grammaire d'arbre suivante :

<R> →

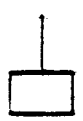


<bloc> → <structure élèmentaire> |

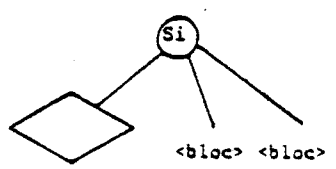


<structure élèmentaire> → <action> | <alternative> | <répétitive>

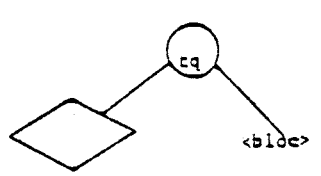
<action> →



<alternative > —



<répétitive> —



Nous nommerons cette seconde représentation d'un processus P : arbre programmatique, que nous noterons $A(P)$.

Définition : arbre programmatique étiqueté

Un arbre programmatique étiqueté est défini comme un triplet (A, Σ, ℓ) où :

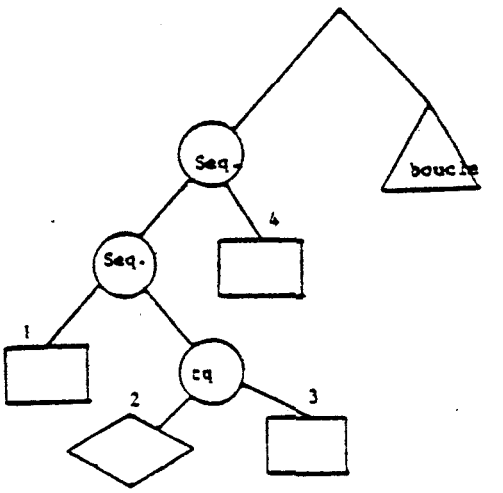
(i) Σ représente l'alphabet des actions et des tests, c.a.d. Σ est formé de deux ensembles disjoints : A l'ensemble des actions et P l'ensemble des tests.

(ii) ℓ représente la fonction d'étiquetage de certains noeuds de l'ordre programmatique A :

$$\ell : N_a \rightarrow A$$
$$N_p \rightarrow P$$

où N_a (resp. N_p) représente l'ensemble des noeuds de la forme

Exemple :



Nous considérons l'alphabet : $\Sigma = A \cup P$ avec $A = \{a,b\}$ et $P = \{p\}$ et la fonction ℓ d'étiquetage avec $\ell(1) = a$

$\ell(2) = p$

$\ell(3) = a$

$\ell(4) = b$

Lemme : Soit un processus physique défini par un arbre programmatique étiqueté (A, Σ, ℓ), nous pouvons lui associer un schéma de processus structuré (R, Σ_1, ℓ_1) de la manière suivante :

(i) Définition de Σ_1

Soit $\Sigma = A \cup P$ avec $= \{a_i\}$ et $\{q_j\}$

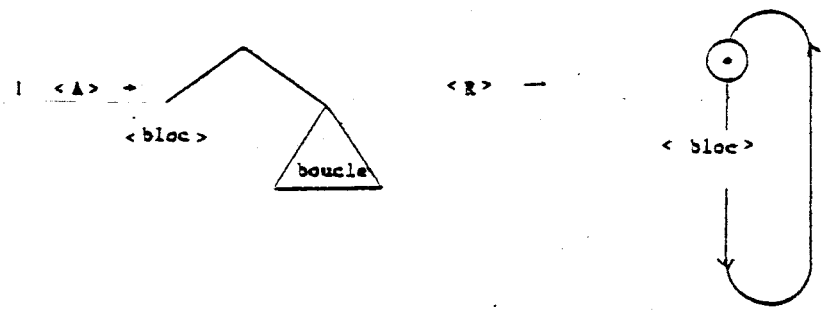
$\Sigma_1 = A_1 \cup P_1$ est construit de la manière suivante :

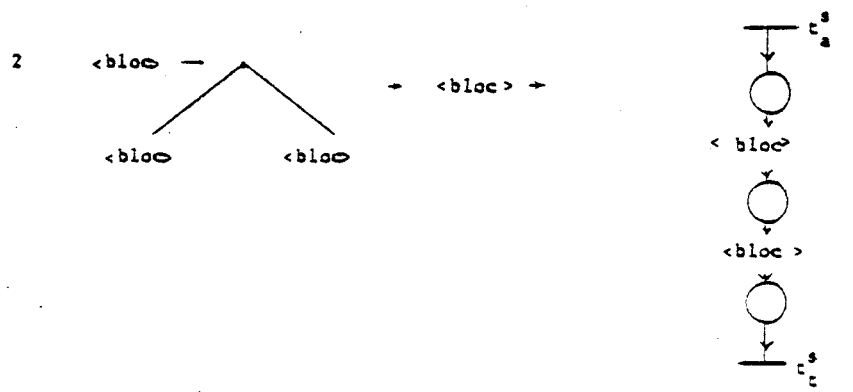
$$A_1 = \{\overline{a_i}\} \cup \{a_{i_1}\}$$

$$P_1 = \{\overline{q_j}\} \cup \{q_{j_1}, q_{j_2}\}$$

(ii) Définition du réseau de Petri R et de la fonction d'étiquetage ℓ_1.

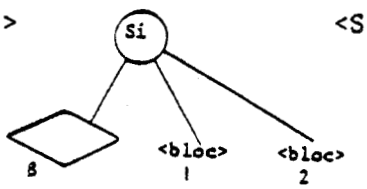
A chaque règle de production, définissant A, nous associons un réseau de Petri.

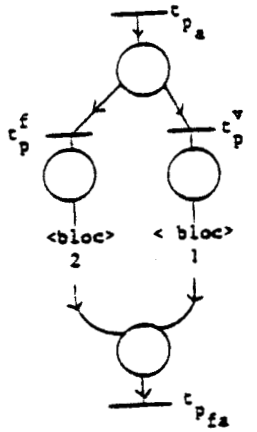




3 <Struc. Elem.> →  → <Struc. Elem.> → 

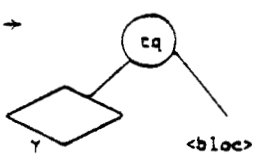
si $\ell(\alpha) = a \in A$ $\quad \ell_1(t^a_a) = a \in A_d$ et $\quad \ell_1(t^a_t) = a_1 \in A_f$

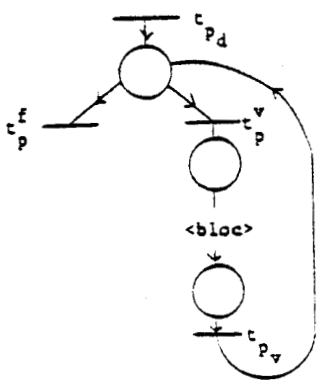
4 <Struct. Elem.> 

<Struc. Elem.> → 

si $\ell(\beta) = q \in P \rightarrow \ell_1(t_{p_a}) = \overline{q} \in P_d$, $\quad \ell_1(t^v_p) = q_1 \in P_f$

$\ell_1(t^f_p) = q_2 \in P_f$, $\quad \ell_1(t_{p_{fa}}) = 1$

5 <Struc. Elem.> → 

<Struc. Elem.> → 

si $\ell(\gamma) = q \in P$ $\ell_1(t_{p_d}) = \ell_1(t_{p_r}) = \overline{q} \in P_d$

$\ell_1(t^v_p) = q_1 \in P_f$, $\quad \ell_1(t^f_p) = q_2 \in P_f$

Définition : Interprétation d'un processus

Pour transformer un schéma de processus en un processus physique, il nous faut donner un sens aux symboles d'actions et de prédicats. La sémantique d'un schéma de processus est définie opérationnellement, par son calcul dans une interprétation.

Une <u>interprétation</u> I des symboles d'un schéma de processus est la donnée :

- d'un ensemble non vide, $\mathcal{D}$, appelé domaine de l'interprétation
- pour chaque symbole a de A , une application $I_o(a)$ de $\mathcal{D}^A$ vers $\mathcal{D}$
- pour chaque symbole p de P , d'une application $I_o(p)$ de $\mathcal{D}^P$ vers {vrai, faux}

On appelle <u>initialisation</u> un point d de $\mathcal{D}$

Introduisons une classe d'interpretations qui presente un caractère universel et qui permet de rendre compte de toutes les autres ; les interprétations libres ou interprétations de Herbrand.

Le domaine considéré est celui des schémas fonctionnels. L'initialisation est toujours l'initialisation canonique, qui est x, l'interprétation de est définie par :

$$I_o(a)(x) = a(x)$$

et aucune condition n'est imposée aux interprétations des prédicats.

Lemme : Soit un schéma de processus. Il existe une interprétation I telle que $\sigma_1\ \sigma_2 \ldots\ \sigma_m$ soit une séquence de déclenchement ssi c'est une séquence de déclenchement pour une interprétation libre.

Définition : Trace du calcul (LANOV)

Lors d'une séquence de déclenchement, $\sigma = \sigma_1\ \sigma_2 \ldots\ \sigma_m$ nous noterons

(i) La suite des symboles prédicats associés aux transitions de cette séquence de déclenchement : ceci forme le <u>mot des tests</u> du calcul.

(ii) la suite des symboles actions associés aux transitions de cette séquence de déclenchement : ceci forme le <u>mot valeur</u> du calcul.

(iii) la suite des symboles rencontrés : ceci forme le <u>mot trace</u> du calcul

mot trace de $\sigma = \ell(\sigma_1)\ .\ \ell(\sigma_2)\ \ldots.\ \ell(\sigma_m)$

Définition : Langage de déclenchement d'un schéma de processus

Le langage de déclenchement d'un schéma de processus, noté $L_d(P)$, définit l'ensemble des séquences de déclenchement du schéma de processus . Il sera défini en deux étapes :

(i) évaluation de L_o : ensemble des séquences de déclenchements allant de la transition $\Gamma(p_{init})$ à la transition $\Gamma^{-1}(p_{init})$, sans déclencher une seconde fois la transition $\Gamma(p_{init})$.

(ii) $L_d(P) = L_o^{\omega}$ (langage infini).

Proposition

L_o est solution d'un système d'équations linéaires à droite, dérivé du graphe de Petri de la manière suivante :

(i) numérotons toutes les transitions de 0 à n où (n+1) représente le nombre de transitions, (0 pour $\Gamma(p_{init})$ et n pour $\Gamma^{-1}(p_{init})$)

(ii) L_i représente l'ensemble des séquences de déclenchements allant de la transition i à la transition n, sans déclencher deux fois la transition 0.

(iii) nous obtenons n+1 équations linéaires à droite, L_o, L_1, ... L_n, dont la solution est un langage rationnel.

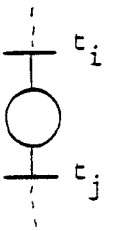
Preuve : nous noterons t_i la transition i dans la suite de cette démonstration. La proposition 2 du paragraphe 2.1, nous assure l'existence d'au moins une séquence de déclenchement menant d'une quelconque transition t_i à la transition t_n.

Pour i = n, alors $L_n = t_n$

Pour i ≠ n, envisageons les deux seuls cas possible, puisque le réseau de Petri est structure.

(i) $|\Gamma^2(t_i)| = 1$

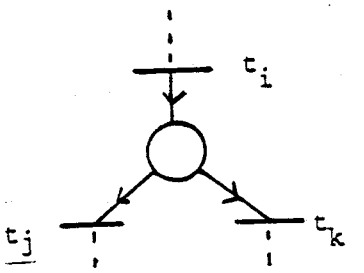
nous avons le graphe suivant :



alors $L_i = t_i \cdot L_j$

(ii) $|\Gamma^2(t_1)| = 2$

nous avons le graphe suivant :



alors $L_1 = t_i \cdot (L_j + L_k)$

$= t_i \cdot L_j + t_i \cdot L_k$

Les n + 1 équations $L_o, \ldots, L_i, \ldots, L_n$ sont donc linéaire à droite dont la solution L_o est un langage rationnel

Définition : Langage trace d'un schéma de processus

Le langage trace, L_T, définit l'ensemble des mots traces du schéma de processus P. Il se déduit par la fonction d'étiquetage, du langage de déclenchement $L_d(P)$:

$$L_T(P) = \ell(L_d(P))$$

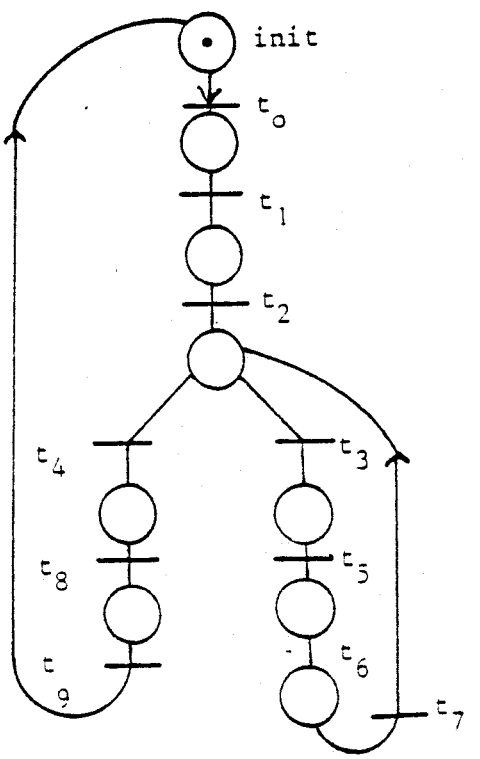
Nous noterons L_{T_o}, le langage obtenu par le morphisme , à partir du langage L_o

$$L_{T_o} = \ell(L_o)$$

Lemme L_{T_o} est un langage rationnel.

Preuve : Immédiat puisque L_o est rationnel (lemme précédent)

Exemple : Considérons le schéma de processus suivant :



Nous pouvons construire k système d'équation L_i

$$L_o = t_o \cdot L_1$$
$$L_1 = t_1 \cdot L_2$$
$$L_2 = t_2 \cdot L_3 + t_2 \cdot L_4$$
$$L_3 = t_3 \cdot L_5$$
$$L_4 = t_4 \cdot L_8$$
$$L_5 = t_5 \cdot L_6$$
$$L_6 = t_6 \cdot L_7$$
$$L_7 = t_7 (L_3 + L_4)$$
$$L_8 = t_8 \cdot L_9$$
$$L_9 = t_9$$

d'où $L_o = t_o t_1 t_2 (t_3 t_5 t_6 t_7)^* t_4 t_8 t_9$

et $L_{T_o} = (L_o) = \overline{a}\, a_1\, \overline{p}\, (p_1\, \overline{a}\, a_1\, \overline{p})^*\, p_2\, \overline{b}\, b_1$

$= \overline{a}\, a_1\, (\overline{p}\, p_1\, \overline{a}\, a_1)^*\, \overline{p}\, p_2\, \overline{b}\, b_1$

Définitions :

1) Deux schémas de processus P et P' sont <u>strictement équivalent</u> si leur langage trace L_o sont identiques (noté $P \sim P' \Leftrightarrow L_o(P) = L_o(P')$

2) Deux schémas de processus P et P' sont Ianov-équivalents si

val (P) = val (P') avec P défini sur le domaine D

P' défini sur le domaine D x V

l'interprétation de D et en conservant la même interprétation de V (noté $P \overset{1}{\sim} P'$)

3) Deux schémas de processus P et P' sont équivalent si leur langage trace infini (L_o^{ω}) sont identiques (noté $P \overset{2}{\sim} P'$) $L_o(P) = L_o^{\omega}(P')$

<u>Lemme</u>

Nous avons les relations suivantes :

i $P \sim P' \Rightarrow P \overset{2}{\sim} P'$

ii $P \overset{1}{\sim} P' \Rightarrow P \overset{3}{\sim} P'$

iii $P \overset{2}{\sim} P' \Rightarrow P \overset{3}{\sim} P'$

2.2. SYSTEME DE PROCESSUS ET RELATION ENTRE PROCESSUS

2.2.1. GRAPHE DE LIAISON

Il est indispensable de pouvoir "lier" des processus pour former un système de processus, chaque processus a une finalité propre et son déroulement tient compte de l'état des autres processus.

Afin de définir formellement un système de processus, nous introduirons quelques notions nouvelles.

Définition :

Parmi l'ensemble des sommets d'un graphe de Petri G = (P, T, Γ), nous distinguerons deux sous-ensembles particuliers :

(i) le sous-ensemble des sommets origines du graphe G, noté SO(G) :

$$SO(G) = \{x \mid x \in PUT \text{ et } \Gamma^{-1}(x) = \emptyset \}$$

(ii) le sous-ensemble des sommets terminaux du graphe G, noté ST(G) :

$$ST(G) = \{y \mid y \in PUT \text{ et } \Gamma(y) = \emptyset \}$$

Par extension, étant donné un réseau de Petri, R = (G, M), nous parlerons des sous-ensembles SO(R) et ST(R) pour SO(G) et ST(G) respectivement.

Définition : Graphes (Réseaux) de Petri indépendants

a - Deux graphes de Petri (P_1, T_1, Γ_1) et (P_2, T_2, Γ_2) sont indépendants ssi :

$$P_1 \cap P_2 = \emptyset$$

$$T_1 \cap T_2 = \emptyset$$

b - N graphes de Petri $G_1, G_2, \ldots, G_n$ sont indépendants ssi

$$\forall_i, \forall_j,\ i \neq j \qquad G_i, G_j \text{ sont indépendants}$$

$$i,j \in \{1,N\}$$

c - Par extension, nous dirons que N réseaux de Petri $(G_1, M_1), \ldots (G_N, M_N)$ sont indépendants ssi les N graphes de Petri $G_1 \ldots, G_N$ sont indépendants.

Définition : Liaison de N graphes (réseaux) de Petri indépendants

a - Le graphe de Petri L lie les n graphes de Petri $G_1, G_2, \ldots, G_N$ ssi

$$SO(L) \subset \left(\bigcup_{i=1} P_i\right) \cup \left(\bigcup_{i=1} T_i\right)$$

$$ST(L) \subset \left(\bigcup_{i=1} P_i\right) \cup \left(\bigcup_{i=1} T_i\right)$$

Remarque : Un sommet du graphe de Petri L n'appartenant ni à SO(L) ni à ST(L) appartient à aucun des graphes P_i.

b - Par extension, le réseau de Petri (L, M) lie N réseaux de Petri indépendants ssi le graphe de Petri L lie les N graphes de Petri associés respectivement aux n réseaux de Petri.

c - Le graphe (réseau) de Petri L sera, par la suite, appelé graphe (réseau) de liaison.

Remarque : Le graphe que définissent le graphe de liaison L et les n graphes de Petri $G_1, \ldots, G_n$ est un graphe de Petri.

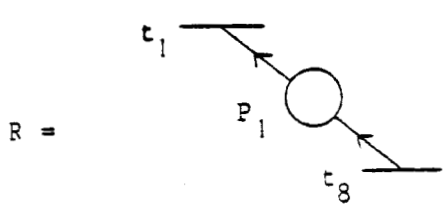
Définition : Système de N processus

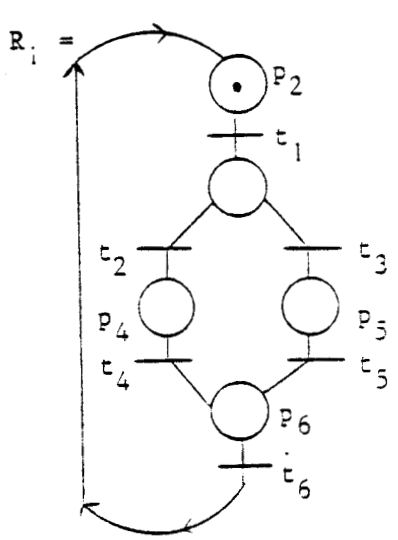
Lorsque les réseaux de Petri indépendants $P_1, P_2, \ldots, P_n$ sont des processus, tout réseau de Petri R les liant défini un système de processus noté $< R; P_1, P_2, \ldots, P_n>$ ssi

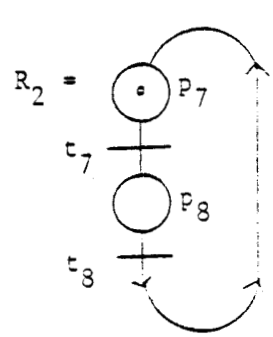
$$N(P_i, M) = 1 \quad , \quad \forall_i \in \{1,\ldots,n\}$$
$$\forall M \in \varepsilon_{M_o}$$

où P_i représente l'ensemble des places du processus P_i;
et M_o le marquage initial du réseau de Petri
$< R; P_1, P_2, \ldots, P_n>$

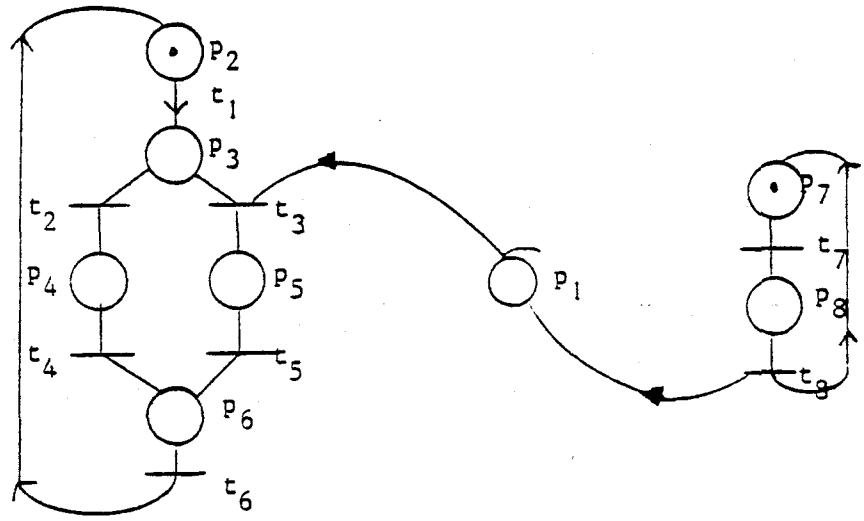
Exemple : Les réseaux de Petri R, R_1, R_2 forment un système de processus $<R ; R_1, R_2>$:







$<R ; R_1, R_2>$ =



Dans la suite de cette étude, nous ne nous préoccuperons que des systèmes de Processus.

Etude du graphe (resp- réseau) de liaison d'un système de Processus

Présentons deux nouvelles notions :

Définition : Chemin de liaison

Soient les deux graphes de Petri G_1 et G_2 indépendants, liés par le graphe de Petri L, nous appelerons chemin de liaison du graphe G_1 vers le graphe G_2 tout chemin tel que les sommets parcourus soient des sommets du graphe L, l'origine étant également un sommet de G_1 et l'extrémité un sommet de G_2.

Remarque : L'origine du chemin de liaison de G_1 vers G_2 appartient à :
$SO(L) \cap (P_1 \cup T_1)$
L'extrémité du chemin de liaison de G_1 vers G_2 appartient à
$ST(L) \cap (P_2 \cup T_2)$.

Définition : Longueur d'un chemin de liaison

(i) La longueur d'un chemin de liaison est égale au nombre d'arcs parcourus en suivant ce chemin.

(ii) Un k-graphe de liaison est un graphe de liaison tel que la longueur de tout chemin soit inférieure ou égale à k.

(iii) Un k-système de processus est un système de processus possédant un k-graphe de liaison.

Puisque nous envisageons de représenter le schéma de contrôle d'un processus industriel par un schéma de processus (Chap. 3), il nous faut d'une part vérifier que ce schéma ne présente aucun blocage (réseau

vivant), et d'autre part pouvoir réinitialiser le processus à tout moment et ce quelque soit son état (réseau propre).

Par ailleurs, nous essayerons de minimiser la longueur des chemins de liaison, ce qui facilitera l'analyse des propriétés de réseau de Petri associé (puisque nous diminuerons le nombre de places et de transition), tout en respectant le cahier des charges du processus à commander.

Dans la suite de ce paragraphe, nous étudierons les propriétés structurelles des graphes de liaison de longueur 1 et 2.

Etude des 1-systèmes de processus

Puisque les systèmes de processus sont des 1-systèmes de processus tout arc (x,y) appartenant au graphe de liaison, de l'un quelconque d'entre eux, est un chemin de liaison.

Les chemins de liaison (x,y) des 1-systèmes de processus se présentent donc sous l'une des deux formes suivantes :

(i) x est une place, y une transition

(ii) x est une transition, y une place

Proposition : Si un 1-système de processus est vivant, alors pour tout chemin de liaison (x,y) existe :

(i) si x est une place, un chemin de liaison (y,x')

(ii) si x est une transition, un chemin de liaison (y',x)

où x,x' et y,y' sont respectivement sommets du même processus.

Preuve : Considérons un 1-système de processus vivant possédant un chemin de liaison (x,y) ne respectant pas les conditions (i) ou (ii). Il nous faut envisager successivement les deux cas :

(1) x est une place

(2) x est une transition

1er cas : x est une place

L'arc (x,y) peut s'écrire (p,t) où p est une place d'un processus P et t une transition d'un processus P' indépendant de P. Le système de processus étant vivant, il existe une séquence de déclenchement σ contenant t déclenchable. Le déclenchement de t supprime une marque de p, or p est une place d'un processus P, et le nombre de marques de ce processus vaut

désormais 0, puisqu'il n'existe aucun chemin de liaison de la forme (t, p') (où p' serait une place de P).

Ceci contredit le fait que P soit un processus, il doit donc exister un chemin de liaison de la forme (t, p') avec p' place de P .

$2^{ème}$ cas : x est une transition

L'arc (x,y) peut s'écrire (t,p) où t est une transition d'un processus P et p une place d'un processus P' indépendant de P . Le système de processus étant vivant, il existe une séquence de déclenchements σ contenant t déclenchable. Le déclenchement de t ajoute une marque dans la place p et donc le nombre de marques du processus P est augmenté d'une unité. Et $N(P')$ est supérieur à 1, puisqu'il n'existe aucun chemin de liaison de la forme (p',t), où p' serait une place du processus P', ceci contredit l'hypothèse que P soit un processus. Il doit donc exister un chemin de liaison de la forme (p',t), avec p' place de P'.

Définition

Un 1-système de processus conforme est un 1-système de processus tel que tout chemin de liaison vérifie l'une des deux conditions (i) ou (ii) de la proposition précédente.

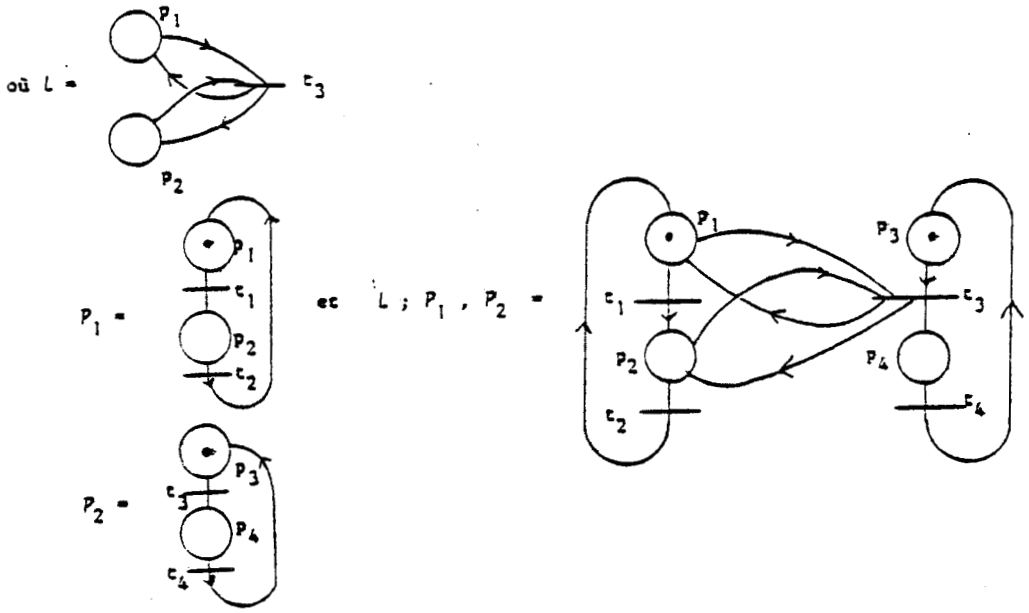
Proposition 2

La classe des 1-systèmes de processus <u>vivant</u> est incluse strictement dans la classe des 1-systèmes de processus conforme.

Démonstration

La proposition 1 entraîne l'inclusion de la classe des 1-systèmes de processus vivant dans la classe des 1-systèmes de processus conforme.

Pour montrer la stricte inclusion, il suffit de trouver un 1-système de liaison conforme qui ne soit pas vivant. Considérons le système suivant $< L ; P_1, P_2 >$



La transition t_3 du processus P_2 ne sera jamais déclenchable puisque les places p_1 et p_2 du processus P_1 ne peuvent être simultanément marquées.

Etude des 2-systèmes de processus

Proposition 3

Si un 2-système de processus est vivant alors pour tout chemin de liaison de la forme (p, t, p') existe au moins un chemin de liaison de la forme (p'_1, t, p_1) où les couples (p,p_1) et (p',p_1) sont respectivement places des mêmes processus P et P'.

Démonstration

Soit un 2-système de processus vivant formé de 2 processus P et P'. Supposons qu'il existe un chemin de liaison de la forme (p'_1, t, p_1) avec (p,p_1) et (p',p'_1) respectivement sommets des mêmes processus P et P', et montrons qu'il y a contradiction.

Le système de processus étant vivant, il existe une séquence de déclenchement σ contenant t déclenchable. Lorsque t sera déclenché, il y aura fuite de l'unique marqueur du processus P vers le processus P'. L'hypothèse faite entraîne que $N(P) = 0$ et $N(P') > 1$, ceci contredit le fait que P et P' appartiennent à un système de processus.

2.2.2. ETUDE DES CHEMINS DE LIAISON DE LA FORME (t,p,t')

Parmi les chemins de liaison de la forme (t,p,t'), nous pouvons distinguer deux classes principales :

- la première définie par une relation de partage de "ressource" entre plusieurs processus.
- la seconde définie une relation de synchronisation entre processus.

222.1 Relation de partage de ressources

Définition : N processus P_1, P_2,..., P_n sont dits liés par une relation de partage de ressource R_{PR} ssi leur déroulement nécessite l'accès exclusif à une ressource (non-partageable).

Remarque : Cette relation est symétrique : $P\ R_{PR}\ P' \Leftrightarrow P'\ R_{PR}\ P$

Définition : Le graphe de liaison L, L = (P, T, Γ) d'un système S de n processus P_i, $i \in \{1,...n\}$, partageant une même ressource sera un 2-graphe de liaison tel que :

(i) $P = \{p_r\}$

$T = SO(L) \cup ST(L)$

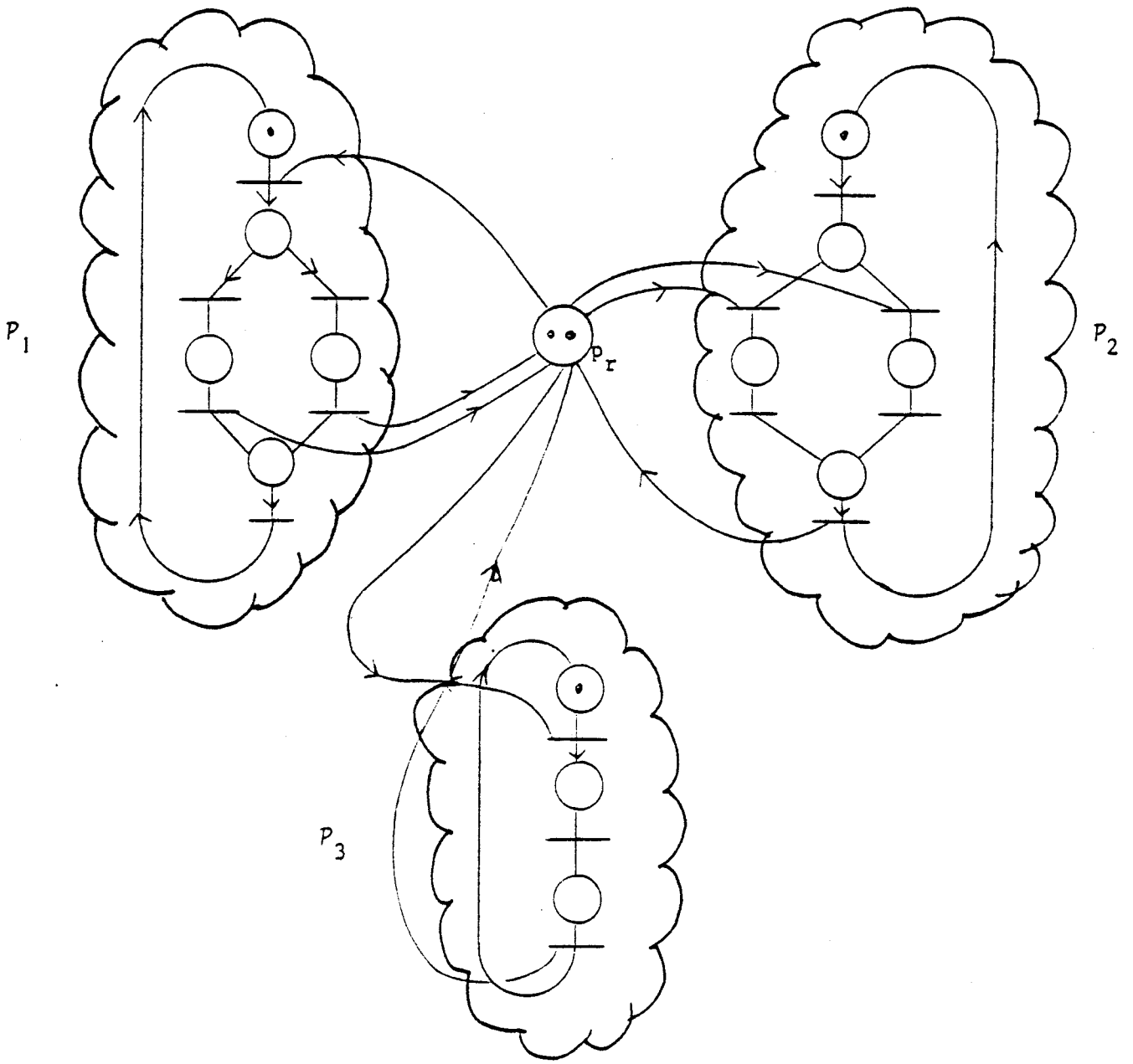
(ii) pour tout $i \in \{1,...,n\}$, $t, t' \in P_i$ t_q

. $t \in \Gamma(p_r)$

. $t' \in \Gamma^{-1}(p_r)$

(iii) $M_o(p_r) \geq 1$

Exemple : Le système suivant, (L, P_1, P_2, P_3), des 3 processus P_1, P_2, P_3 sont liés par une relation de partage de ressource :



Remarques : 1 - les marqueurs que contient la place p_r à un instant particulier symbolisent les divers exemplaires disponibles de la ressource.

2 - l'arc (p_r,t) , où t est une transition du processus P_i, représente la requête d'un exemplaire de la ressource par le processus P_i.

3 - l'arc (t',p_r), où t' est une transition du processus P_j, représente la restitution d'un exemplaire de la ressource par le processus P_j.

4 - Pour un processus P_i, le nombre d'arc de la forme (p_r,t) peut-être différent du nombre d'arc de la forme (t',p_r) (exemple : les processus P_1 et P_2 du système défini plus haut).

Partage de ressource à deux états. Relation producteur-consommateur

Définition : N processus P_i, $i \in \{1,\ldots,n\}$, sont liés par une relation de producteur-consommateur R_{PC} ssi leur déroulement nécessite l'accès à une ressource pouvant prendre deux états distincts, notés R_1 et R_2, de la manière suivante :

(i) Chaque processus P_i, $i \in \{1,\ldots,n\}$, requiert un exemplaire de la ressource se trouvant dans l'un des deux états R_1 ou R_2 et la restitue dans l'autre état.

(ii) Il existe deux processus ayant besoin d'un exemplaire de la ressource l'un dans l'état R_1 et l'autre dans l'état R_2.

Le graphe de liaison L, d'un système, S, de n processus P_i, $i \in \{1,\ldots,n\}$, liés par une relation producteur-consommateur R_{PC}, peut-être défini formellement comme suit :

Soient $S = \{L ; P_1,\ldots, P_n\}$, M le marquage initial du système S et $L = (P, T, \Gamma)$, alors :

(i) $P = \{p_{r_1}, p_{r_2}\}$

(ii) $T = SO(L) \quad ST(L)$

(iii) P_i, $i \in \{1,\ldots,n\}$ $\quad t, t' \in P_i$ $\quad t_q$

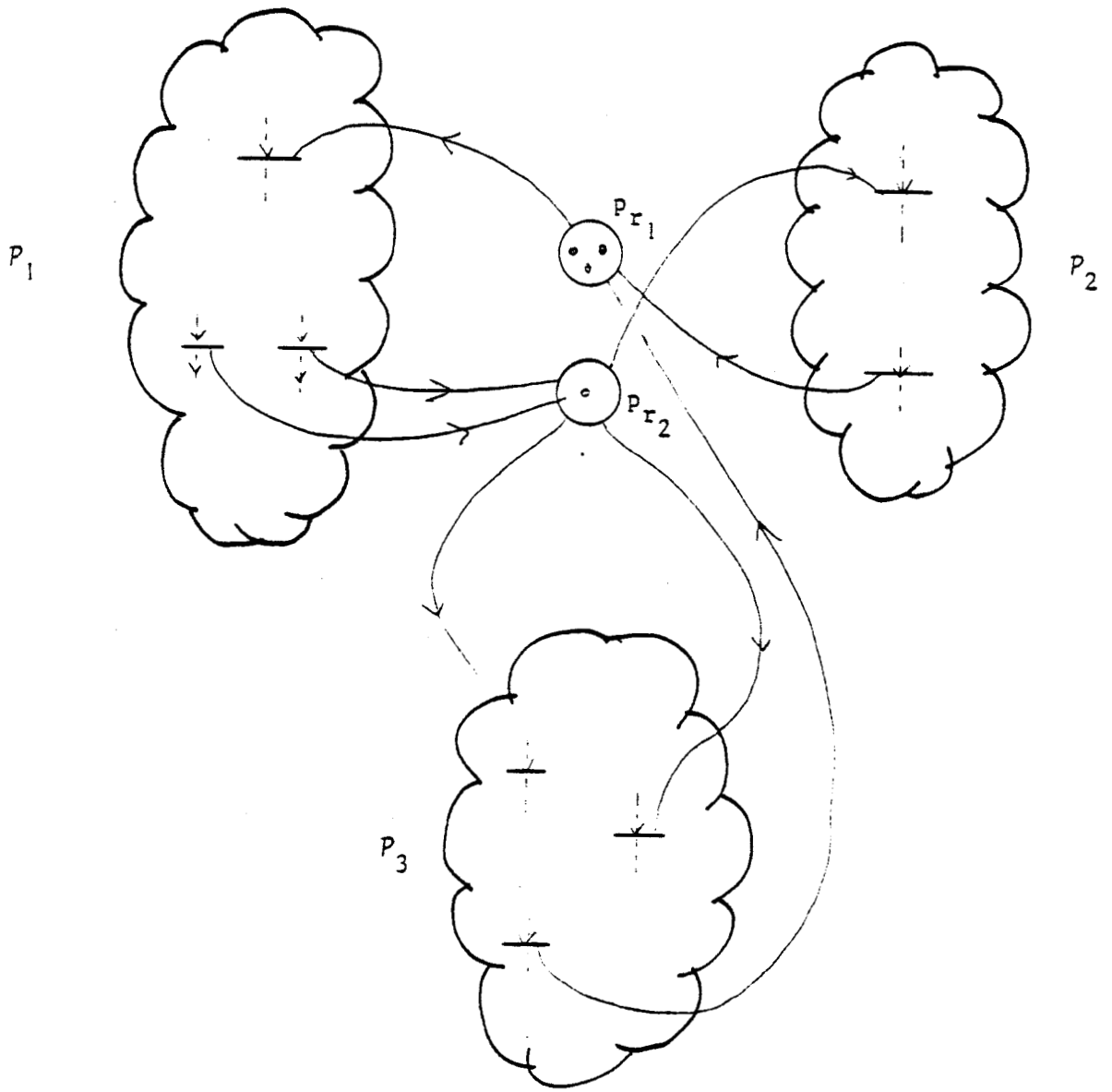
$$t \in \Gamma(p_{r_j})$$

$$t' \in \Gamma^{-1}(p_{r_{3-j}}) \qquad , j \in \{1,2\}$$

(IV) $\Gamma^{-1}(p_{r_1}) \neq \{\emptyset\}$, $\Gamma^{-1}(p_{r_2}) \neq \{\emptyset\}$

(V) $M_o(p_{r_1}) + M_o(p_{r_2}) \geq 1$

Exemple : Le système suivant, $(L ; P_1, P_2, P_3)$, de trois processus P_1, P_2, P_3 sont liés par une relation producteur-consommateur.



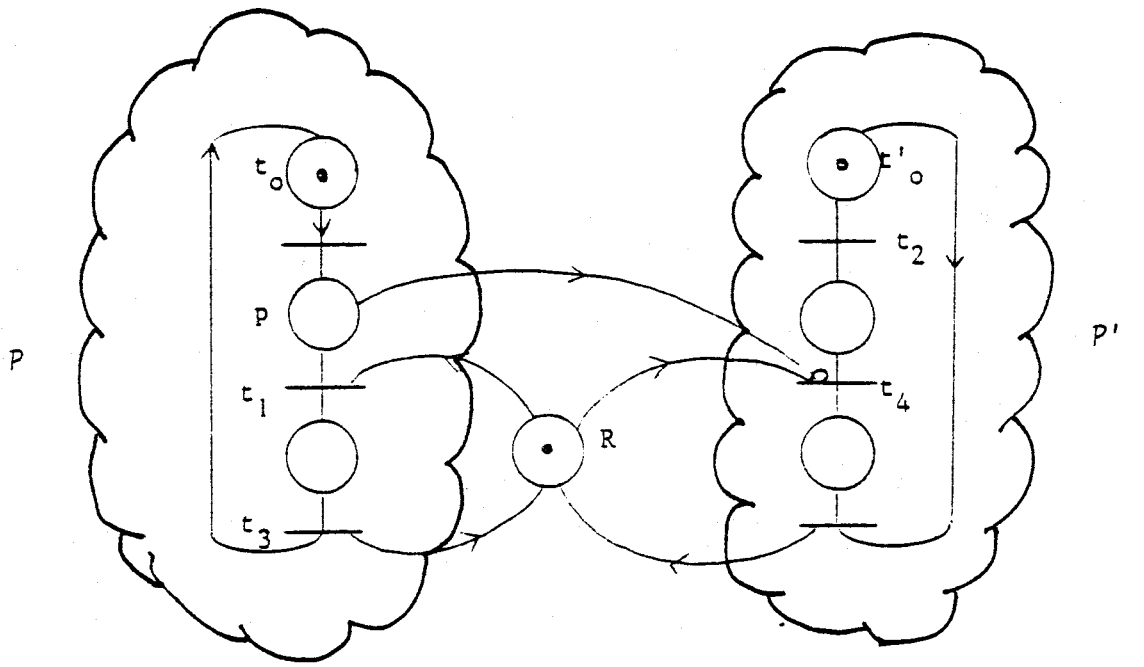
Remarque : Les marqueurs que contient la place p_{t_1} (resp. p_{r_2}) à un instant particulier, symbolisent les divers exemplaires disponibles de la ressource dans l'état R_1 (resp. dans l'état R_2).

Relation de partage de ressource avec priorité

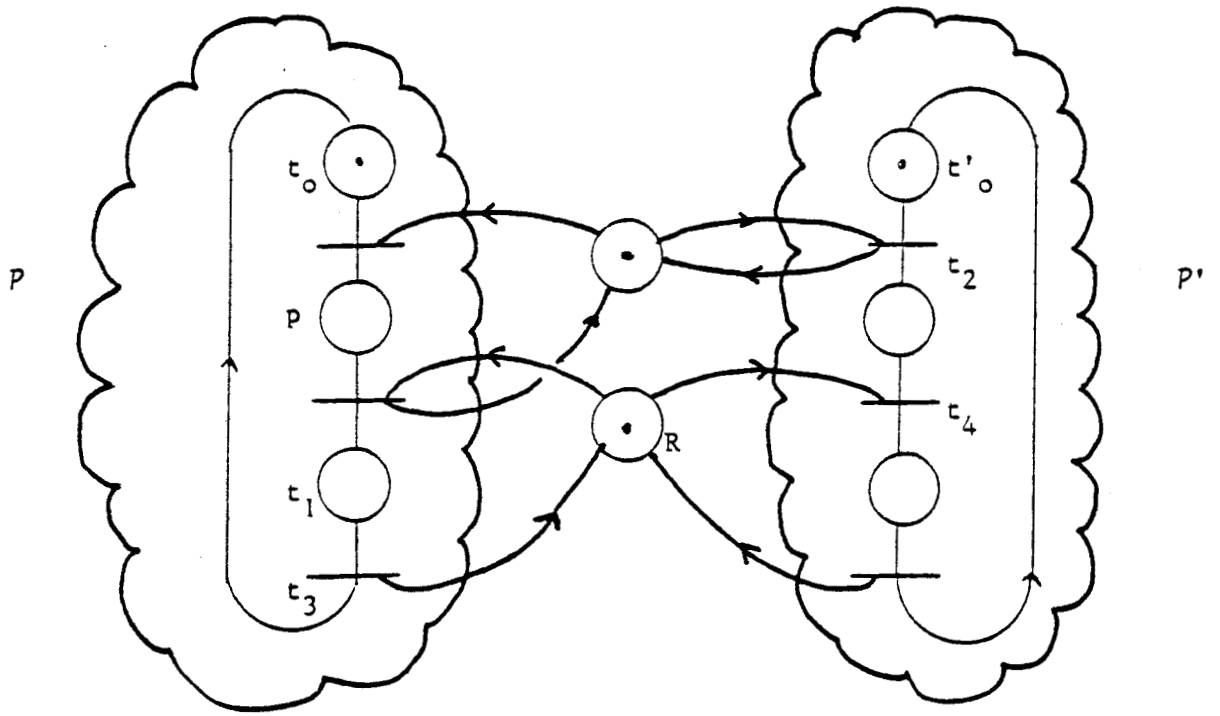
Lors d'un partage de ressource entre deux processus peut se produire une situation de conflit : les deux processus effectuent la requête de la même ressource simultanément.

Afin de lever cette ambiguité, peut-être introduite une priorité entre les deux processus qui indiquera le processus ayant accès à la ressource ; nous obtenons une relation de partage de ressource avec priorité.

Le graphe de liaison de deux processus P et P' liés par une relation de partage de ressource avec priorité sera un 2-graphe de liaison de la forme :



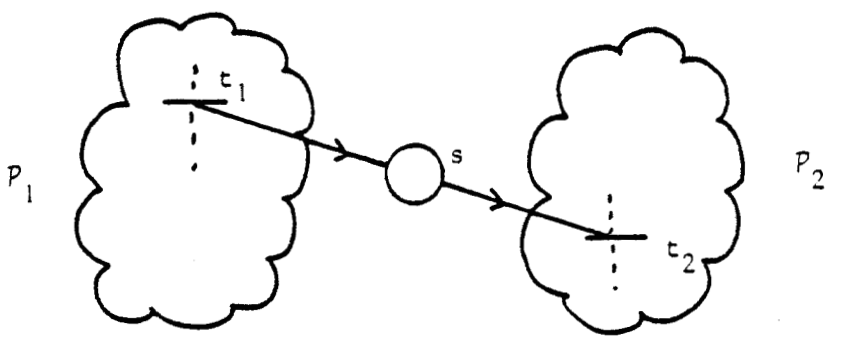
Puisque les processus sont des graphes de Petri 1-bornés, l'arc inhibiteur (p,t_2) peut-être supprimé, nous obtenons ainsi le graphe de liaison :



Ceci peut-être étendu, sans difficulté, (GIR 78) au cas de n processus.

222.2 Relation de synchronisation

Le graphe de liaison d'un système de deux processus P_1 et P_2 liés par une relation de synchronisation se réduit à la liaison (t_1, s, t_2), où t_1 et t_2 sont respectivement transitions de P_1 et P_2 et le marquage initial de la place s est nul.

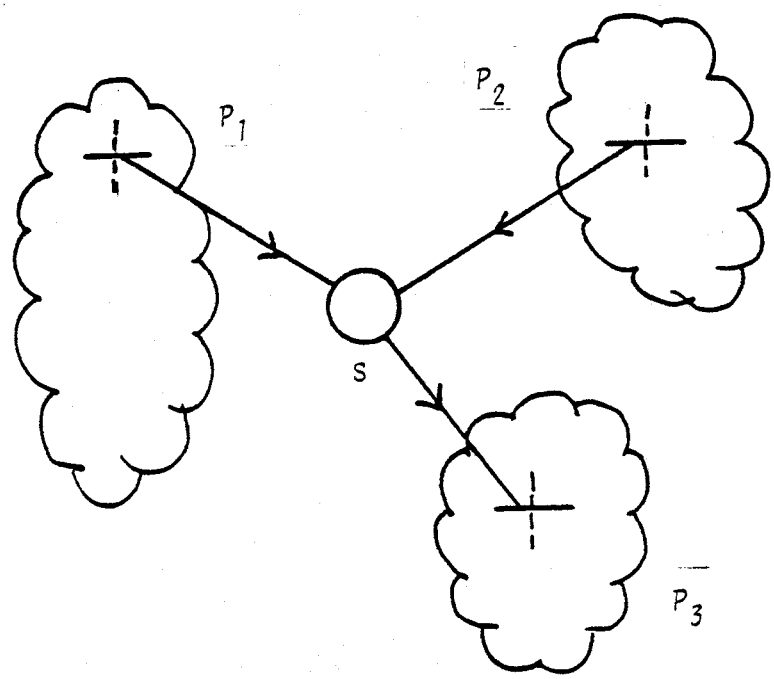


Remarques : 1 - La transition t_2 est dite synchronisée par la transition t_1 et nous noterons cette relation de synchronisation entre les processus P_1 et P_2 : $t_1 R_s t_2$.

2 - La relation de synchronisation R_s est anti-symétrique.

Cette liaison présente quelques variantes :

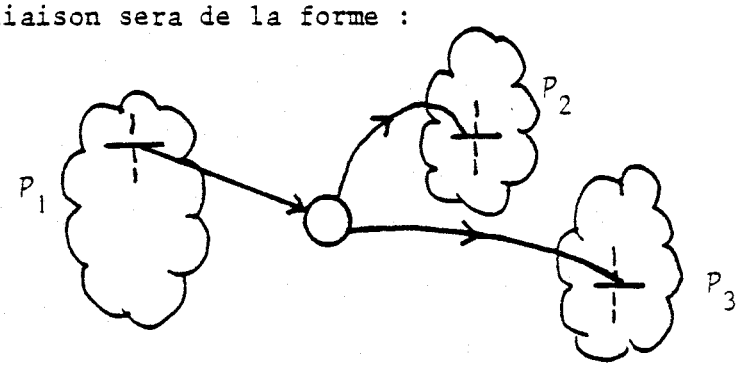
(i) La transition t_3 du processus P_3, d'un système de 3 processus, est synchronisée par la transition t_1 du processus P_1 <u>ou</u> par la transition t_2 du processus P_2 ; le 2-graphe de liaison associé au système des trois processus P_1, P_2, P_3 sera de la forme :



La relation sera notée : $(t_1 \vee t_2)\ R_s\ t_3$

(ii) Variante 2

La transition t synchronisé n transitions t_1, t_2, ..., n, le 2-graphe de liaison sera de la forme :



Remarque : Cette liaison est non-deterministe s'il existe au moins 2 transitions t_i et t_j de l'ensemble des n transitions (t_1, t_2, ..., t_n) appartenant à 2 processus différents.

Dans le paragraphe suivant, nous reprendrons en détails l'étude de ces graphes de liaisons dans le cas des systèmes de schéma de processus.

2.3 CARACTERE VIVANT D'UN SYSTEME DE SCHEMAS DE PROCESSUS

Nous envisagerons successivement les systèmes de schémas de processus dont le graphe de liaison sera de la forme :

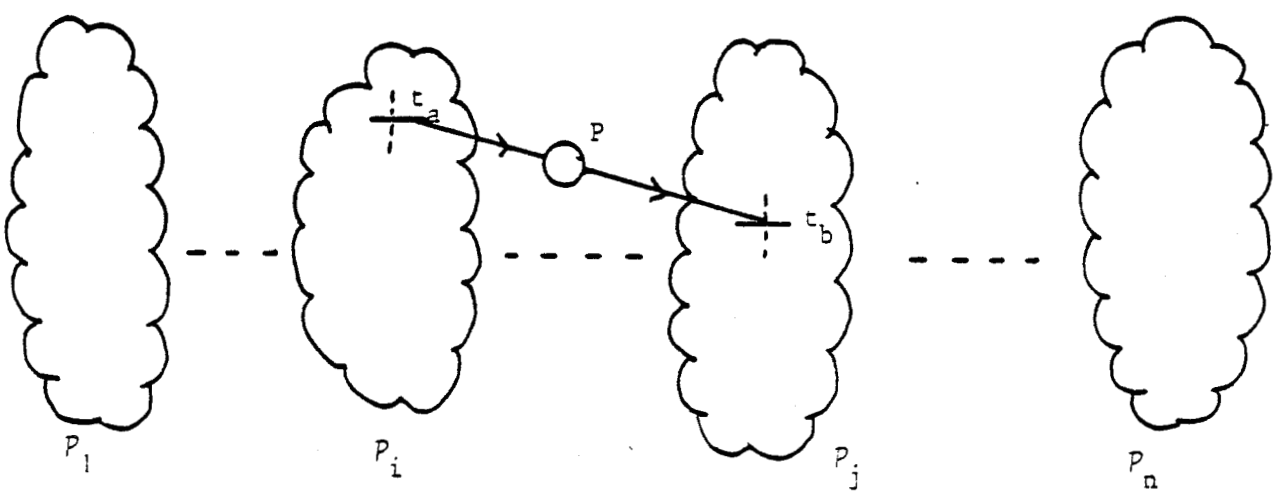
(i) synchronisation
(ii) partage de ressource
(iii) producteur-consommateur

2.3.1. SCHEMAS DE PROCESSUS LIES PAR DES RELATIONS DE SYNCHRONISATION

Considérons un système de n schémas de processus ; si deux schémas de processus P et P' sont liés par une relation de synchronisation, il existe, au moins, un opérateur de P dont la fin synchronise un opérateur de ou dont le début est synchronisé par un opérateur de P'. Si l'opérateur "a" synchronise l'opérateur "b", il existe une 2-liaison que nous noterons (t_a, p, t_b) où :

(i) t_a et t_b sont les transitions telles que $\ell(t_a) = a$ et $\ell(t_b) = \bar{b}$
(ii) p est une place du graphe de liaison L du système de schémas de processus.

Exemple : Soit un système n schémas de processus, $P_1, \ldots, P_n$, dont P_i et P_j sont liés par une relation de synchronisation



Dans la suite de ce paragraphe, nous noterons P_E l'ensemble des places du graphe de liaison appartenant à une liaison de type synchronisation et Σ l'alphabet des début et fin d'opérateur des schémas de processus.

Une place p du sous-ensemble P_E symbolise un événement, mais afin de traduire fidélement le fait qu'un schéma de processus P synchronise un schéma de processus P', nous ferons la distinction entre :

(i) l'envoi de l'événement par P, noté $\overline{p}$

(ii) l'attente de ce même événement par P', noté p

L'envoi d'un événement par le schéma de processus P se traduit par l'ajout d'un marqueur dans la place p. La fin de l'attente de cet événement par le schéma de processus P' se tratuit par le retrait d'un marqueur de la place p. L'ensemble des envois et attentes d'événement sera noté E.

Le langage trace, $L_T(P)$, caractérise le comportement d'un schéma de processus pris isolément, sans tenir compte de l'environnement du processus au sein du système de schémas de processus c'est-à-dire en ignorant les interactions du schéma de processus avec les autres schémas de processus du système.

Si nous abordons l'étude d'une famille de schémas de processus liés par des relations de synchronisation, nous pouvons caractériser les liaisons d'un schéma de processus P avec les autres schémas de processus, par un <u>langage d'événement</u>, noté $L_E(P)$. Ce langage, $L_E(P)$, se déduit du langage de déclenchement $L_d(P)$ du schéma de processus P par le <u>morphisme Ø</u>.

<u>Définition</u> : Le morphisme $\emptyset : T \to E^*$ sera tel que :

(i) $\emptyset(t) = 1$ si t ST(L) SO(L)

(ii) $\emptyset(t) = \overline{p}_1 \ldots \overline{p}_n$ où $\{p_1 \ldots p_n\} = \{p \in P_E$ tq(t,p) soit un arc du graphe de liaison$\}$

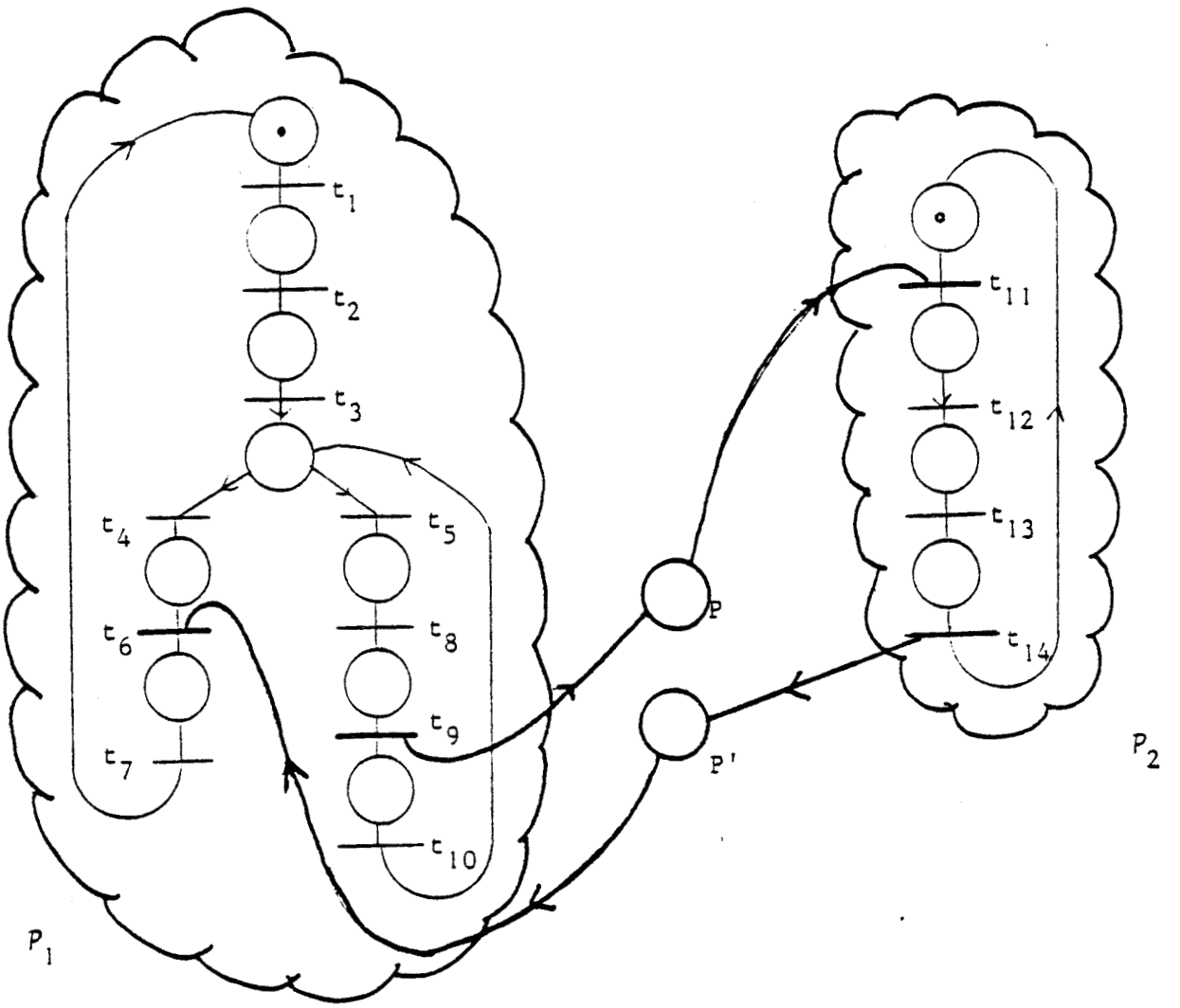
(iii) $\emptyset(t) = p'_1 \ldots p_k$ où $\{p'_1 \ldots p'_k\} = \{p \in P_E$ tq (p,t) soit un arc du graphe de liaison$\}$

Considérons un mot w du langage de déclenchement $L_d(P)$ du schéma de processus P s'écrivant : $w = t_{i_1} t_{i_2} \ldots t_{i_n}$; le morphisme Ø nous permet d'obtenir le mot u du langage d'événement $L_E(P^n)$:

$$u = \emptyset(w) = \emptyset\,(t_{i_1} t_{i_2} \ldots t_{i_n})$$

$$= \emptyset\,(t_{i_1}) \,.\, \emptyset\,(t_{i_2}) \,.\, \ldots\, \emptyset\,(t_{i_n})$$

Exemple : Soit le système de deux schémas de processus P_1 et P_2, liés par deux liaisons de synchronisation :



Nous pouvons définir successivement les ensembles d'actions et de prédicats, la fonction d'étiquetage ℓ, et le morphisme $\emptyset$.

Les ensembles d'actions et de prédicats étant les suivants :

$$A = \{a, b, c, d, e\}$$

$$P = \{q\}$$

La fonction d'étiquetage ℓ est telle que :

$$\ell(t_1) = \overline{a},\ \ell(t_2) = a$$

$$\ell(t_3) = \overline{q},\ \ell(t_4) = q_1,\ \ell(t_5) = q_2,\ \ell(t_{10}) = \overline{q}$$

$$\ell(t_6) = \overline{b},\ \ell(t_7) = b$$

$\ell(t_8) = \overline{c}$, $\ell(t_9) = c$

$\ell(t_{13}) = \overline{e}$, $\ell(t_{14}) = e$

Considérons maintenant le morphisme $\emptyset$:

$\emptyset(t_9) = \overline{p}$

$\emptyset(t_6) = p'$

$\emptyset(t_{11}) = p$

$\emptyset(t_{14}) = \overline{p}'$

$\emptyset(t_i) = 1$ pour $i \in \{1, 2, 3, 4, 5, 7, 8, 12, 13\}$

Les langages de déclenchement des schémas de processus P_1 et P_2 seront définis par :

$$L_d(P_1) = t_1\ t_2\ t_3\ (t_5\ t_8\ t_9\ t_{10})^*\ t_4\ t_6\ t_7$$

$$L_d(P_2) = t_{11}\ t_{12}\ t_{13}\ t_{14}$$

Le morphisme $\emptyset$ permet de déduire les deux langages d'événements $L_E(P_1)$ et $L_E(P_2)$:

$$L_E(P_1) = \emptyset(L_d(P_1)) = \emptyset(t_1\ t_2\ t_3(t_5\ t_8\ t_9\ t_{10})^*\ t_4\ t_6\ t_9)$$

$$= \overline{p}^*\ p'$$

$$L_E(P_2) = \emptyset\ (L_d(P_2)) = p\ \overline{p}'$$

Etude d'un système de schémas de processus liés par des relations de synchronisation.

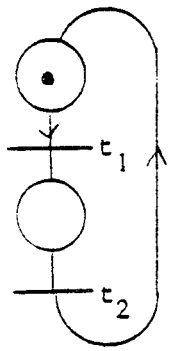
Définition : Un système de n schémas de processus liés par des relations de synchronisation posséde un état de blocage total s'il existe n transitions $(t_1, t_2, \ldots t_n)$, appartenant respectivement aux n schémas de processus, de type "début d'action", non vivantes.

Exemple 1 : Envisageons le système des trois schémas de processus, $\langle L ; P_1, P_2, P_3 \rangle$, caractérisé par :

(i) Les réseaux de Petri PN_1, PN_2, PN_3 et la fonction d'étiquetage ℓ

(ii) le graphe de liaison L et le morphisme $\emptyset$

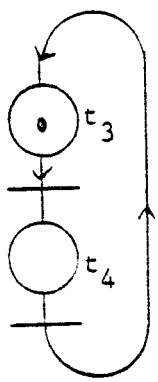
PN_1 :



$\ell(t_1) = \overline{a}$

$\ell(t_2) = a$

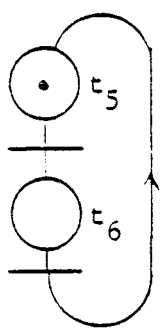
PN_2 =



$\ell(t_3) = \overline{b}$

$\ell(t_4) = b$

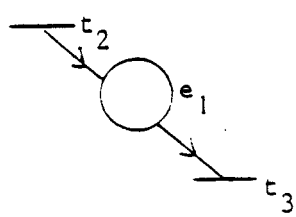
PN_3 =



$\ell(t_5) = \overline{c}$

$\ell(t_6) = c$

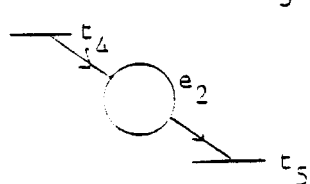
R_1 =



$\phi(t_2) = \overline{e}_1$

$\phi(t_3) = e_1$

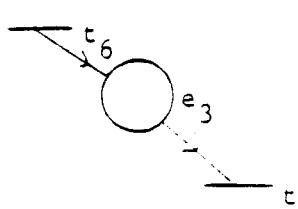
R_2 =



$\phi(t_4) = \overline{e}_2$

$\phi(t_5) = e_2$

R_3 =



$\phi(t_6) = \overline{e}_3$

$\phi(t_1) = e_3$

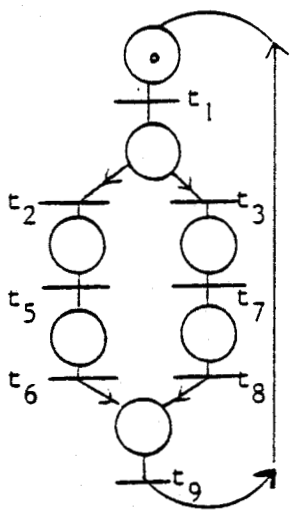
Ce système posséde un état de blocage total :< t_1, t_3, t_5>

Exemple 2 : Considérons le système des deux schémas de processus, $< L ; P_1, P_2 >$ caractérisé par :

(i) Les réseaux de Petri PN_1, PN_2 et la fonction d'étiquetage ℓ

(ii) Le graphe de Liaison L et le morphisme Φ :

PN_1 :



$\ell(t_1) = \overline{p}$

$\ell(t_2) = p_1$

$\ell(t_3) = p_2$

$\ell(t_5) = \overline{a}$

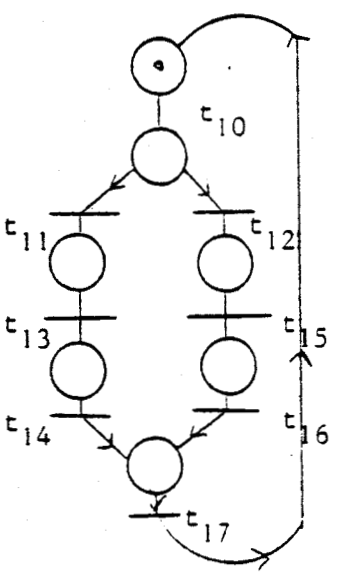
$\ell(t_6) = a$

$\ell(t_7) = \overline{b}$

$\ell(t_8) = b$

$\ell(t_9) = 1$

PN_2 :



$\ell(t_{10}) = \overline{q}$

$\ell(t_{11}) = q_1$

$\ell(t_{12}) = q_2$

$\ell(t_{13}) = \overline{c}$

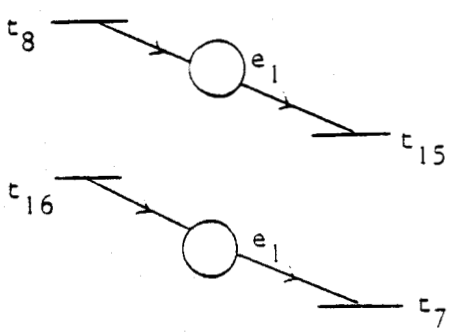
$\ell(t_{14}) = c$

$\ell(t_{15}) = \overline{d}$

$\ell(t_{16}) = d$

$\ell(t_{17}) = 1$

L :



$\Phi(t_8) = \overline{e}_1$

$\Phi(t_{15}) = e_1$

$\Phi(t_{16}) = e_2$

$\Phi(t_7) = e_2$

Ce système posséde un état de blocage total : $<t_7, t_{15}>$, mais il se peut que ce blocage ne survienne jamais si les tests p et q forcent le choix des transitions t_2 et t_{11}.

Caractérisation d'un système de schémas de processus totalement bloqué.

Les liaisons d'un schéma de processus avec les autres processus d'un système de n processus liés par des relations de synchronisation peuvent être décrites par son langage d'événement $L_E(P)$.

Lemme : Si un système de n schémas de processus liés par des relations de synchronisation : $< L ; P_1, P_2, \ldots, P_n >$, posséde un état de blocage total alors :

n mots $\quad w_1 = u_1 x_1 v_1 \quad L_E(P_1)$

$w_2 = u_2 x_2 v_2 \quad L_E(P_2)$

.

.

.

$w_n = u_n x_n v_n \quad L_E(P_n)$

$u_i, v_i \in E^*$

$x_i \in E$

tels que : $\sum_{j=1}^{n} \#_{x_i}(u_j) = \sum_{j=1}^{n} \#_{\overline{x_i}}(u_j), \quad i \quad \{1, 2, \ldots, n\}$

Remarque :

Nous définissons le nombre d'occurrence d'une lettre a d'un alphabet Σ , dans un mot u de Σ^+, par la fonction $\#_a(u)$ de Σ^+ dans

Démonstration du lemme :

Considérons un système de schémas de processus, liés par des relations de synchronisation et possédant un état de blocage total.

Ce systéme se caractérise par l'existence de n transitions $(t_1, t_2, \ldots, t_n)$ du graphe de liaison appartenant respectivement aux n schémas de processus de type "début d'action", non vivantes.

Ces n transitions sont non vivantes car en attente d'un événement puisque chaque schéma de processus pris isolément est vivant par définition.

Notons y_i, les n attentes d'événements c'est-à-dire :

$y_i = \emptyset \ (t_i)_i = \{1, 2, \ldots n\}$

Il existe n mots, w_1, w_2, ..., w_n, tels que :

$$w_i = u_i\, y_i\, v_i \in L_E(P_i) \qquad i = \{1, 2, \ldots, n\}$$

et

$$u_i, v_i \in E^*$$

$$y_i \in E$$

Considérons l'une quelconque de ces n transitions, t_i, et posons $\alpha = \Gamma^{-1}(t_i)$ l'événement qui lui est associé.

Etudions la relation liant l'attente et l'envoie de l'événement α au n-uple $(y_1, y_2, \ldots, y_n)$, il nous faut distinguer trois cas :

(i) $$\sum_{j=1}^{n} \#_\alpha(u_j) < \sum_{j=1}^{n} \#_{\overline{\alpha}}(u_j)$$

Cette relation implique qu'il existe au moins un envoie d'événement e, et par conséquent l'attente d'événement y_i doit être satisfaite. La transition t_i est déclenchable ce qui contredit l'hypothèse.

(ii) $$\sum_{j=1}^{n} \#_{\overline{\alpha}}(u_j) > \sum_{j=1}^{n} \#_\alpha(u_j)$$

. il existe un mot w_k parmi les mots w_i tel que :

$$\alpha \in u_k \text{ , (si } w_k = u_k\, y_k\, v_k)$$

ce préfixe u_k peut être réécrit :

$$u_k = u_{1k}\, x_k\, u_{2k} \quad , \qquad \text{avec } x_k = \alpha$$

. Puisque $\sum_{j=1}^{n} \#_\alpha(u_j) > \sum_{j=1}^{n} \#_{\overline{\alpha}}(u_j)$, nous avons la relation :

$$\sum_{\substack{j=1 \\ j\neq k}}^{n} \#_\alpha(u_j) + \#_\alpha(u_{1k}) > \sum_{j=1}^{n} \#_{\overline{\alpha}}(u_j) - 1$$

Cette dernière relation entraîne que la transition associée à l'attente d'événement x_k est non vivante , ce qui contredit le fait que l'état y_k puisse être atteint.

La relation liant α et $\overline{\alpha}$, au point $(y_1, y_2, \ldots, y_n)$, ne peut être que :

$$\sum_{j=1}^{n} \#_{\alpha}(u_j) = \sum_{j=1}^{n} \#_{\overline{\alpha}}(u_j)$$

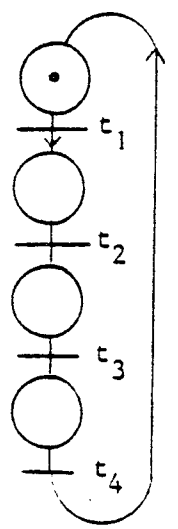
La réciproque de ce lemme est fausse comme le montre le contre-exemple suivant :

Soit le système de schémas de processus, $< L ; P_1, P_2, P_3, P_n >$, dont :

(i) Les schémas de processus P_1, P_2, P_3, P_4 sont définis par les réseaux de Petri PN_1, PN_2, PN_3, PN_4 et la fonction d'étiquetage ℓ,

(ii) Le graphe de liaison L est défini par le réseau de Petri PN_L et le morphisme $\emptyset$.

PN_1 =



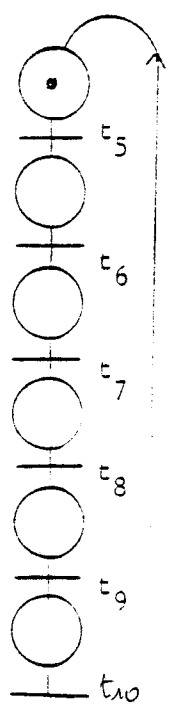
$\ell(t_1) = \overline{a}$

$\ell(t_2) = a$

$\ell(t_3) = \overline{b}$

$\ell(t_4) = b$

PN_2 =



$\ell(t_5) = \overline{c}$

$\ell(t_6) = c$

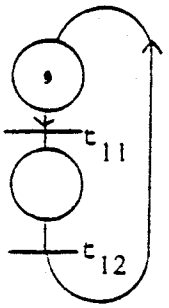
$\ell(t_7) = \overline{d}$

$\ell(t_8) = d$

$\ell(t_9) = \overline{e}$

$\ell(t_{10}) = e$

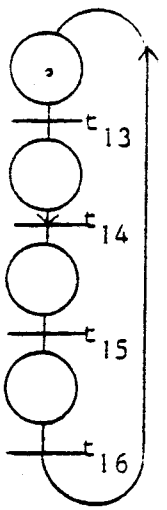
PN_3 =



$\ell(t_{11}) = \overline{f}$

$\ell(t_{12}) = f$

PN_4 =



$\ell(t_{13}) = \overline{g}$

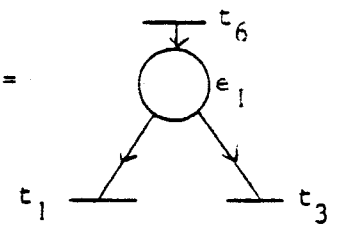
$\ell(t_{14}) = g$

$\ell(t_{15}) = \overline{h}$

$\ell(t_{16}) = h$

$PN_L = (R_1, R_2, R_3, R_4)$

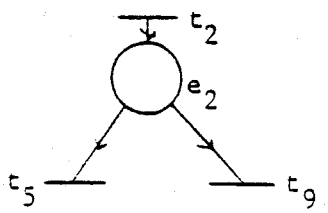
R_1 =



$\phi(t_6) = \overline{e}_1$

$\phi(t_1) = \phi(t_3) = e_1$

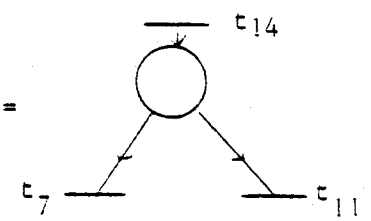
R_2 =



$\phi(t_2) = \overline{e}_2$

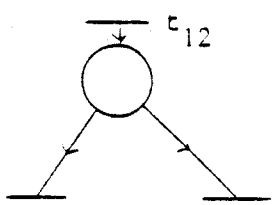
$\phi(t_5) = \phi(t_9) = e_2$

R_3 =



$\phi(t_{14}) = \overline{e}_3$

$\phi(t_7) = (t_{11}) e_3$



$\phi(t_{12}) = \overline{e}_4$

$\phi(t_{15}) = e_4$

Du langage de déclenchements des schémas de processus P_1, P_2, P_3, P_4, nous pouvons déduire leur langage d'événements notés $L_E(P_1)$, $L_E(P_2)$, $L_E(P_3)$ et $L_E(P_4)$:

$$L_E(P_1) = e_1 \bar{e}_2 e_1 = w_1$$

$$L_E(P_2) = e_2 \bar{e}_1 e_3 e_2 = w_2$$

$$L_E(P_3) = e_3 \bar{e}_4 = w_3$$

$$L_E(P_4) = \bar{e}_3 e_4 = w_4$$

Si nous procédons au découpage des mots w_1, w_2, w_3, w_4 de la manière suivante :

$$w_1 = u_1, x_1, v_1 \text{ avec } u_1 = e_1 \bar{e}_2$$
$$x_1 = e_1$$
$$v_1 = 1$$

$$w_2 = u_2, x_2, v_2 \text{ avec } u_2 = e_2 \bar{e}_1 e_3$$
$$x_2 = e_2$$
$$v_2 = 1$$

$$w_3 = u_3, x_3, v_3 \text{ avec } u_3 = 1$$
$$x_3 = e_3$$
$$v_3 = \bar{e}_4$$

$$w_4 = u_4, x_4, v_4 \text{ avec } u_4 = \bar{e}_3$$
$$x_4 = e_4$$
$$v_4 = 1$$

Le 4- uple (x_1, x_2, x_3, x_4) est tel que :

$$\sum_{j=1}^{4} \#_{e_i}(u_j) = \sum_{j=1}^{4} \#_{\bar{e}_i}(u_j), \quad \forall i \in \{1,2,3,4\} \qquad (A)$$

Mais ce n'est pas un blocage total car les transitions t_3 et t_9 sont inaccessibles, les transitions t_1 et t_6 étant non vivantes !

De plus, le découpage des mots w_1, w_2, w_3, w_4 vérifiant la condition (A) est le seul existant. Ce contre-exemple nous conduit à introduire la notion de système partiellement bloqué.

Système partiellement bloqué

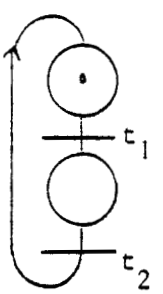
Le système totalement bloqué du paragraphe précédent n'est qu'un cas particulier de système de schémas de processus, liés par des relations de synchronisation, non vivant.

Définition : Un système de n schémas de processus liés par des relations de synchronisation posséde un état de blocage partiel s'il existe p transitions (P_{i_1} , P_{i_2} ,..., P_{i_p}) appartenant à p processus, de type "début d'action", non vivantes.

Exemple : Considérons le système formé des 3 schémas de processus SP_1, SP_2, SP_3 et du graphe de liaison L.

Les trois schémas de processus SP_1, SP_2, SP_3 sont définis par les 3 réseaux de Petri PN_1, PN_2, PN_3, et la fonction d'étiquetage ℓ :

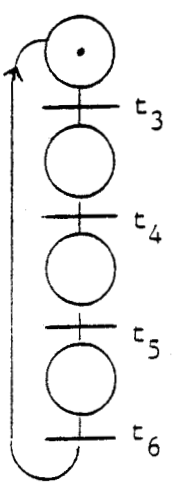
PN_1 :



$\ell(t_1) = \overline{a}$

$\ell(t_2) = a$

PN_2 :



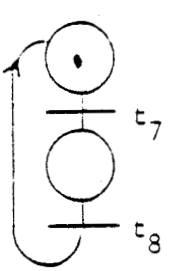
$\ell(t_3) = \overline{b}$

$\ell(t_4) = b$

$\ell(t_5) = \overline{c}$

$\ell(t_6) = c$

PN_3 :

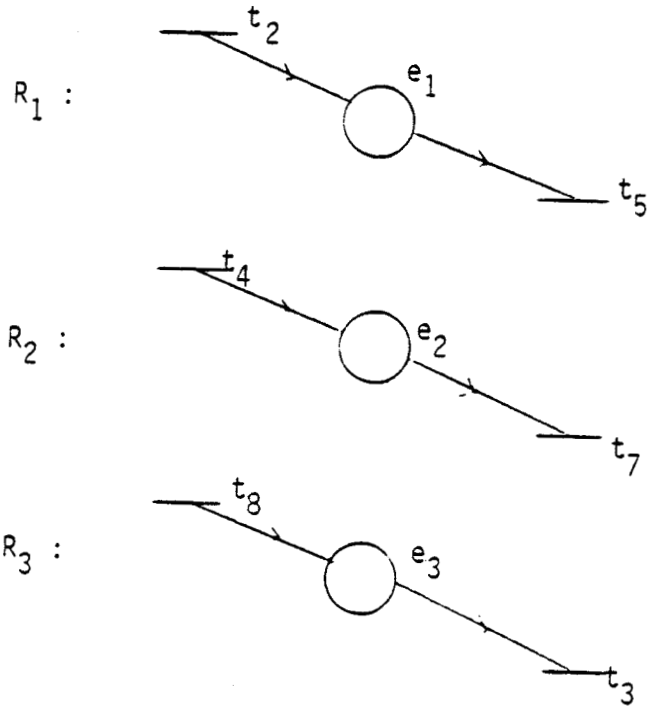


$\ell(t_7) = \overline{d}$

$\ell(t_8) = d$

Le graphe de liaison étant défini par les réseaux de Petri R_1, R_2, R_3 et le morphisme $\emptyset : T \to E$

$$L = (R_1 \cup R_2 \cup R_3)$$



$\phi(t_2) = \bar{e}_1$
$\phi(t_5) = e_1$

$\phi(t_4) = \bar{e}_2$
$\phi(t_7) = e_2$

$\phi(t_8) = \bar{e}_3$
$\phi(t_9) = e_3$

Ce système posséde un état de blocage partiel caractérisé par les 2 transitions t_3, t_7 appartenant respectivement aux schémas de processus SP_1 et SP_2.

Lemme : Un système de n schémas de processus liés par des relations de syncrhonisation :< L ; P_1, P_2, ..., P_n> , posséde un état de blocage partiel ssi :

un sous-ensemble de p schémas de processus, et p mots w_1, w_2, ..., w_p tels que :

$$w_i \in L_e(P_{j_i}) \qquad i \in \{1,\ldots p\}$$

$$w_i = u_i\, x_i\, v_i \qquad u_i, v_i \in E^* \qquad x_i \in E$$

$$(i)\quad \sum_{j=1}^{p} \#_{x_i}(u_j) = \sum_{j=1}^{p} \#_{\overline{x}_i}(u_j)\ , \qquad i \in \{1,\ldots,p\}$$

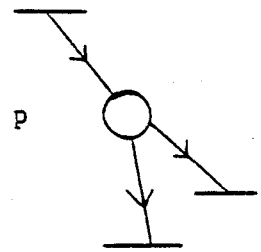
$$(ii)\quad \#_{\alpha}(L_e(P_{j_k})) = 0\ , \qquad k \in \{p+1,\ldots,n\}$$
$$\alpha\ ,\ \alpha = x_i$$

Remarque : Lorsque le sous-ensemble est en fait formé des n schémas de processus, nous retrouvons le cas du systéme totalement bloqué.

Preuve :

Soit un système de n schémas de processus liés par des relations de synchronisation et possédant un état de blocage partiel.

Il existe p transitions (t_{i_1}, t_{i_2}, t_{i_p}) non vivantes. Une transition t_{i_j} appartient au processus P_{i_j} et au graphe de liaison L. Les relations liant les schémas de processus étant du type synchronisation, sont de la forme : t-p-t et plus précisément de la forme :



$|\Gamma(p)| = k \geq 1$

Les p transitions (t_{i_1}, t_{i_2}, ..., t_{i_p}) étant non vivantes, sont du type attente d'événement, c'est-à-dire que :

$$\forall j \in \{1,2,\ldots,p\}\ ,\ \exists\, p_k \in L \quad t_q\ t_{i_j} \in \Gamma(p_k)$$

Notons W l'ensemble des événements attendus par les p transitions.

Le systéme des n schémas de processus se divisant en un sous-ensemble de p schémas de processus non vivants et (n-p) schémas de processus vivants. Les p schémas de processus sont bloqués en attente d'événément,(appartenant à W) et ne seront pas débloqués par les (n-p) autres schémas de processus d'où :

$$(1)\quad \sum_{j=1}^{n-p} \#_{\alpha}(L_e(P_{i_j})) = 0 \qquad \alpha \in W$$

Puisque les (n-p) autres schémas sont vivants, nous aurons la seconde relation :

$$(2) \quad \sum_{j=1}^{n-p} \#_{\overline{\alpha}} (L_e (P_{i_j})) = 0 \quad , \qquad \forall \alpha \in W$$

Les relations (1) et (2) entraînent donc :

$$\sum_{j=1}^{p} \#_{\alpha} (u_i) = \sum_{j=1}^{p} \#_{\overline{\alpha}} (u_i) \quad , \qquad \forall i \in \{1,2,\ldots,p\}$$

2.3.2. SCHEMAS DE PROCESSUS LIES PAR DES RELATIONS DE PARTAGE DE RESSOURCES

Nous limiterons notre étude aux schémas de processus structurés puisque tout schéma de processus peut-être défini, d'une manière équivalente, par un schéma de processus structuré.

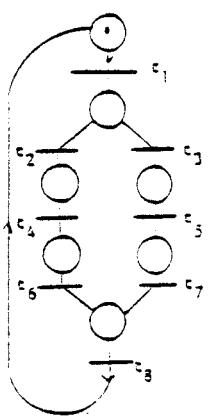
Un schéma de processus structuré, P , peut-être représenté par un arbre programmatique, noté $A(P)$.

Parmi la classe des schémas de processus structurés liés par des relations de partage de ressources, nous n'envisagerons que la sous-classe des schémas de processus structurés dont les partages de ressource sont associés à des noeuds de l'arbre programmatique de la manière suivante : chacun de ces noeuds est la racine d'un sous-arbre de $A(P)$, et la requête d'une ressource précédera le premier traitement de ce sous-arbre et la restitution suivra le dernier traitement du sous arbre.

Illustrons cette définition par un exemple :

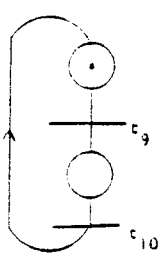
Soit S = < L ; P_1, P_2 >, un système de processus structuré

P_1

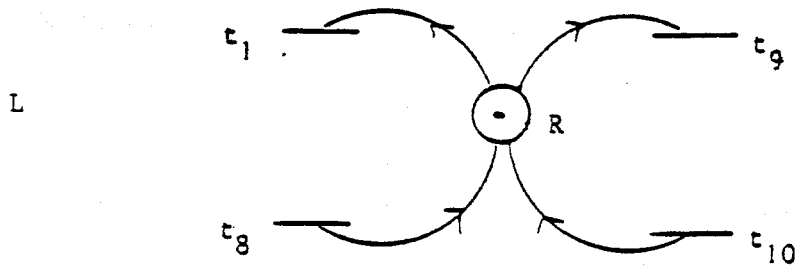


$\ell(t_1) = \overline{p}$
$\ell(t_2) = p_1$
$\ell(t_3) = p_2$
$\ell(t_4) = \overline{a}$
$\ell(t_6) = a$
$\ell(t_5) = \overline{b}$
$\ell(t_7) = b$
$\ell(t_8) = 1$

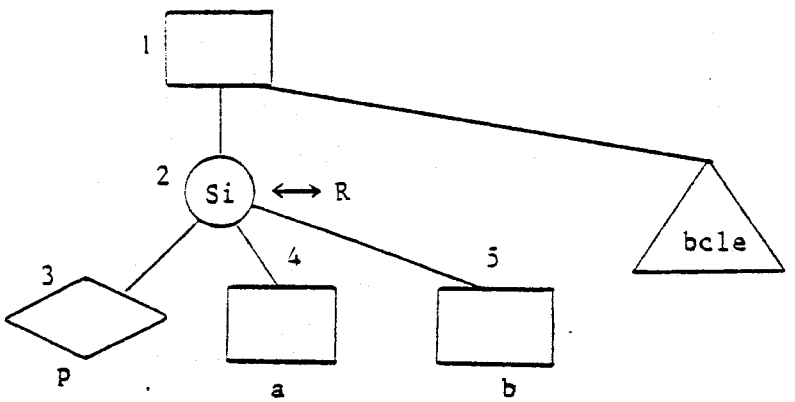
P_2



$\ell(t_9) = \overline{c}$
$\ell(t_{10}) = c$



et en représentant le schéma de processus P_1 à l'aide d'un arbre programmatique, cette liaison se symbolisera par le graphisme suivant $\leftrightarrow$ R, associé au noeud 2, racine du sous arbre de $A(P_1)$ partageant la ressource R.



Cette constante entraîne que toute requête d'une ressource par un schéma de processus sera suivie de sa restitution et toute restitution d'une ressource aura été précédée d'une requête. Et comme conséquence, nous avons le lemme suivant :

Lemme : La place associée à une relation de partage de ressource respectant cette constante est k-bornée où k représente le nombre initial d'exemplaires de cette ressource.

Remarque : Dans la suite de ce paragraphe nous nous intéresserons qu'aux ressources ne possédant qu'un exemplaire.

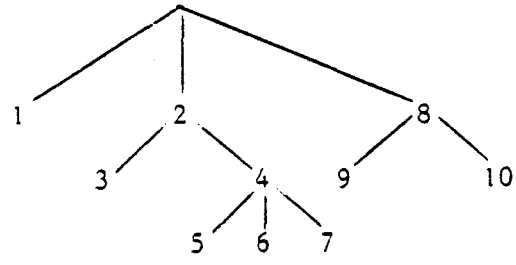
La requête et la restitution d'une ressource s'apparentant à un parenthésage et la restriction approtée assure la correction de ce celui-ci.

Langage d'un schéma de processus

Dans la théorie des arbres (THA 73), un langage de branches peut-être associé à un arbre particulier. (A une branche $(a_1, a_2, \ldots a_p)$ correspond le mot $a_1\ a_2 \ldots a_p$, où $a_1, \ldots, a_p$ sont des noeuds de l'arbre). Nous noterons $L_B(A(P_1)$ le langage de branche de l'arbre programmatique du schéma de processus P_1 et par extension $L_B(P_1)$.

Illustrons ceci sur un exemple. Soit l'arbre programmatique suivant :

A =



Le langage de branches de cet arbre est le suivant :

$L_B(A) = \{1,\ 2.3,\ 2.4.5,\ 2.4.6,\ 2.4.7,\ 8.9,\ 8.10\}$

Langage de ressources d'un schéma de processus structuré

Considérons un schéma de processus structuré P lié aux autres schémas de processus par des relations de partages de ressource respectant notre constante. Le morphisme ψ permet d'associer à son langage de branches $L_B(P)$, un langage de ressource, noté $L_R(P)$.

Définition : Soient R l'ensemble des ressources partagées et N l'ensemble des noeuds de l'arbre programmatique du schéma de processus P, le morphisme ψ, de N dans RU 1 est défini comme suit :

$\psi(i) = 1$ si le noeud i n'effectue aucune requête de ressource

$\psi(i) = j$ si le noeud i effectue la requête de la ressource j

A un mot $w = i_1,\ldots,i_n$, du langage de branches du schéma de processus , le morphisme ψ associera le mot u du langage de ressource de ce même processus:

$$u = \psi(w) = \psi(i_1\ i_2 \ldots i_n) = \psi(i_1)\ \psi(i_2)\ \ldots\ \psi(i_n)$$

Lemme : Un systéme 1-borné, formé de deux schémas de processus P_1 et P_2 liés par des relations de partage de ressource, présente un état de blocage total ssi il existe deux mots w_1 et w_2 appartenant respectivement aux langages de ressources de P_1 et P_2 tels que :

$$w_1 = u_1\ r\ v_1\ s\ z_1$$

$$w_2 = u_2\ s\ v_2\ r\ z_2$$

avec $u_1, u_2, v_1, v_2, z_1, z_2 \in R^*$

$r, s \in R$

et $u_1\ r\ v_1 \cap u_2\ s\ v_2 = \phi$

Preuve :

Considérons un système 1-borné, de deux schémas de processus P_1 P_2 liés par des relations de partage de ressource et où celui-ci se fait aux niveaux des sous-arbres.

(i) Supposons que le système soit vivant, donc sans blocage mais qu'il existe deux mots w_1, w_2 appartenant respectivement à $L_R(P_1)$, et $L_R(P_2)$ tels que :

$$w_1 = u_1\ r\ v_1\ z_1$$

$$w_2 = u_2\ s\ v_2\ r\ z_2$$

avec $u_1, v_1, z_1, u_2, v_2, z_2 \in R^*$

$r, s \in R$

et $u_1\ r\ v_1 \cap u_2\ s\ v_2 = \phi$ (A)

Puisque le systéme est vivant, il existe une séquence de déclenchement amenant le schéma de processus P_1 sur la branche associée à w_1 et le schéma de processus P_2 sur la branche associée à w_2

Puisque $u_1\ r\ v_1 \cap u_2\ s\ v_2 = \phi$, les schémas de processus P_1 et P_2 vont faire respectivement la requête des ressources s et r. Le système étant 1-borné, il n'existe qu'un exemplaire unique de chacune de ces ressources r et s, qui, à cet instant, est en possession respectivement de P_1 et P_2.
Le système présente un état de blocage total.

(ii) Supposons que les deux schémas de processus P_1 et P_2 sont bloqués. Puisque toutes les liaisons sont du type partage de ressources, les deux schémas de processus P_1 et P_2 sont bloqués pour une requête de ressource non satisfaite, notons r et s les ressources respectivement attendues par P_1 et P_2.

L'état de blocage du système entraîne que l'unique exemplaire de la ressource r est en possession de P_2, de même s est en possession de P_1.

Puisque le partage de ressource est associé à un sous arbre, il existe deux mots w_1 et w_2 appartenant respectivement à $L_R(P_1)$ et $L_R(P_2)$ de la forme :

$$w_1 = u_1\ s\ v_1\ r\ z_1 \in L_R(P_1)$$

$$w_2 = u_2\ r\ v_2\ s\ z_2 \in L_R(P_2)$$

avec $u_1, v_1, z_1, u_2, v_2, z_2 \in R^*$; $r, s \in R$

Puisque les deux schémas de processus P_1 et P_2 ont pu attendre les noeuds associés respectivement aux ressources r et s par le morphisme ψ, nous avons la relation

$$u_1\ s\ v_1 \cap u_2\ r\ v_2 = \phi$$

2.3.3. SCHEMAS DE PROCESSUS LIES PAR DES RELATIONS PRODUCTEUR-CONSOMMATEUR

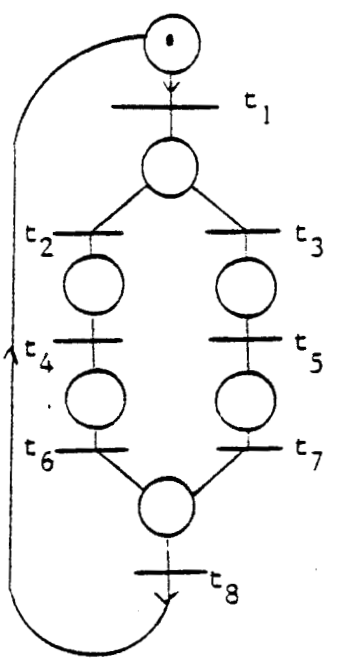
Comme dans le paragraphe précédent, notre étude se limitera aux systèmes de schémas de processus structurés.

Les relations producteur-consommateurs sont associées à des sous-arbres de l'arbre programmatique définissant le schéma de processus structuré. Si un schéma de processus structuré est lié à d'autres schémas par une relation producteur-consommateur, il prélève un exemplaire de la ressource dans l'état 1 (resp. 2) avant le premier traitement du sous-arbre auquel est attachée cette liaison et la restitue dans l'état 2 (resp. 1) après le dernier traitement du sous arbre.

Illustrons cette définition par un exemple.

Soit $S = \langle L ; P_1, P_2 \rangle$, un système de processus structuré

P_1



$\ell(t_1) = \overline{p}$

$\ell(t_2) = p_1$

$\ell(t_3) = p_2$

$\ell(t_4) = \overline{a}$

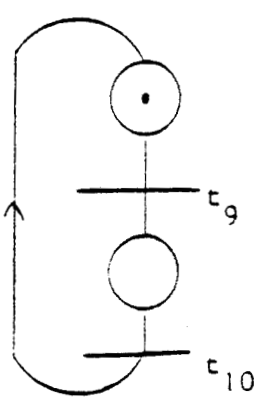
$\ell(t_6) = a$

$\ell(t_5) = \overline{b}$

$\ell(t_7) = b$

$\ell(t_8) = 1$

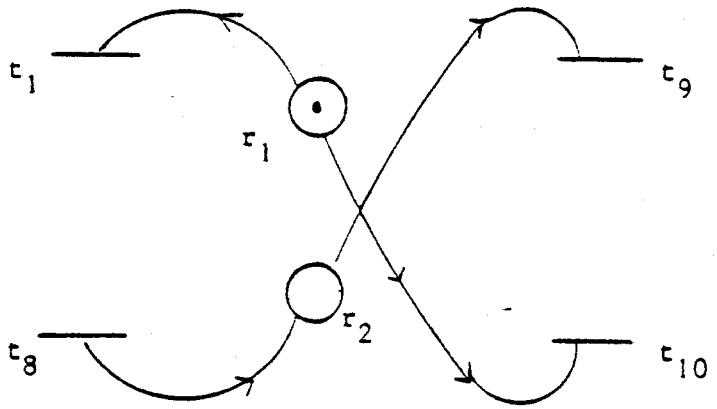
P_2



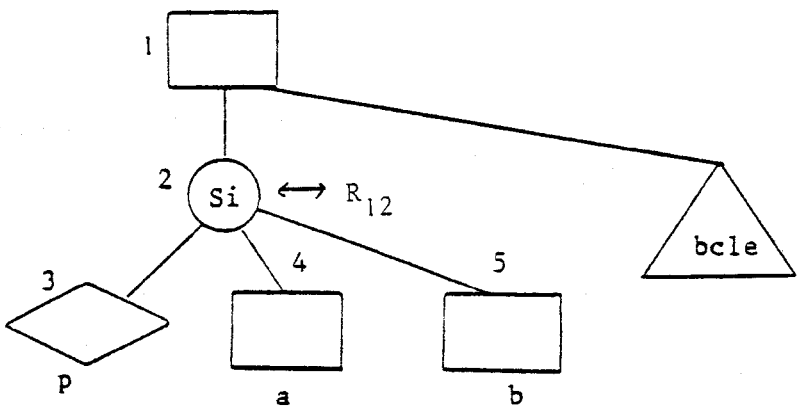
$\ell(t_9) = \overline{c}$

$\ell(t_{10}) = c$

L



et en représentant le schéma de processus P_1 à l'aide d'un arbre programmatique, cette liaison se symbolisera par le graphique suivant $\leftrightarrow R_{12}$ (ou $\leftrightarrow R_{21}$) si le schéma de processus prend la ressource dans l'état 1 (resp. état 2).



La remarque faite dans le paragraphe précédent reste valable : la contrainte apportée aux liaisons en les attachant à des sous-arbres permet d'éviter une appropriation d'une ressource dans un état au dépend des autres schémas de processus. (Les liaisons producteur-consommateur se comportent comme un parenthésage).

Langage de ressources 2-états d'un schéma de processus structuré.

Considérons un système de schémas de processus structurés liés par des relations producteur-consommateur. Nous noterons R_{PC} l'ensemble des ressources 2-états. A cet ensemble correspond deux sous-ensembles de places du graphe de liaison, notés R_1 et R_2, symbolisant les ressources soit dans l'état 1, soit dans l'état 2. Nous identifirons R_{PC} et R_1 UR_2 dans la suite de ce paragraphe.

A un schéma de processus P peut-être associé un arbre programmatique et de manière équivalente un réseau de Pétri structuré.

Considérons le langage de déclenchement de P et le morphisme ψ', de l'ensemble des transitions T dans l'ensemble des ressources 2-états auquel nous ajouterons l'élèment neutre {1} : $\psi' : T \to R_1 \cup R_2 \cup \{1\}$

(i) $\psi'(t) = 1$ si t ST(L) U SO(L)

(ii) $\psi'(t) = \bar{r}_1$ (resp. $\bar{r}_2$) où r_1 R_1 et (t, r_1) est un arc du graphe de liaison

(resp. r_2 R_2 et (t, r_2) est un arc du graphe de liaison

(iii) $\psi'(t) = r_1$ (resp. r_2) où r_1 R_1 et (r_1, t) est un arc du graphe de liaison

(resp. r_2 R_2 et (r_2, t) est un arc du graphe de liaison

Considérons un mot w du langage de déclenchement $L_d(P)$ du schéma de processus P s'écrivant : $w = t_{i_1} t_{i_2} \ldots t_{i_n}$; le morphisme ψ' nous permet d'obtenir le mot u du langage de ressources 2-états $L_{R_2}(P)$ =

$$u = \psi'(w) = \psi'(t_{i_1}) \psi'(t_{i_2}) \ldots \psi'(t_{i_n})$$

Etude d'un système de n schémas de processus liés par des relations producteur-consommateur.

Introduisons un schéma de processus virtuel P_{n+1}, chargé d'initialiser le graphe de liaison, c'est-à-dire de déposer les marqueurs dans les places des sous-ensembles R_1 et R_2. Après cette initialisation ce processus fait la requéte d'une ressource inexistante R_{fant}.

Notons $R_{PC} = \{r_1, \ldots, r_n\}$ et r_{i_1} (resp. r_{i_1}) la place associée à la ressource r_i dans l'état 1 (resp. état 2).

Si le marquage initial de la place r_{i_1} est j, le mot $(\bar{r}_{i_1})^j$ caractérise $N(r_{i_1}, M_o)$.

Le langage de ressource 2-état de ce schéma de processus virtuel P_{n+1} sera de la forme :

$$L_{PC}(P_{n+1}) = (\bar{r}_{i_1})^{j_1} . (\bar{r}_{i_2})^{j_2} \ldots (\bar{r}_{n_2})^{j_{2n}} . r_{fant}$$

L'ajout de ce schéma de processus P_{n+1} permet de considérer le système de n schémas de processus comme lié par des relations de synchronisation (l'envoie d'un évènement est assimilé à une restitution de ressource et l'attente d'un évènement à une requête, l'ensemble des évènements étant $R_1 \cup R_2 \cup \{R_{fant}\}$. Cette dernière remarque permet d'utiliser les résultats du paragraphe 2.3.1.

BIBLIOGRAPHIE DU CHAPITRE 2

(BER) BERTHELOT G.
"Vérification de réseaux de Pétri" ; doctorat 3ème cycle - Paris, 1978

(BOHM 66) BOHM C., JACOPINI G.
"Flow diagrams turing machines and languages with only two formations rules" ; CACM 9.5, 1966

(CRO) CROCUS
"Systèmes d'exploitation des ordinateurs" ; Dunod - 1975

(GIR 78) GIRAULT C.
"Réseaux de Pétri et Synchronisation de processus" ; Publication Institut de Programmation n°78-02 (1978)

(HAC) HACK M.
"Analysis of production schemata by Petri Nets" ; M.S. thesis M.I.T., TR 94, (1974)

(HAC 75) HACK M.
"Petri Nets Languages" ; M.I.T., TR 159, March 1975

(IANOV) IANOV Y.I.
"The Logical Schemes of Algorithms" ; Problems of cybernetics, Pergamon Press - New York 1960

(MOA) MOALLA M., PULOU J., SIFAKIS J.
"Réseaux de Petri Synchronisés" ; Journées AFCET Nice 1977

(NOE 77) NOE J.O.
"Machine aided modelling using modified Petri Nets" ; Journées d'étude AFCET sur les réseaux de Petri, Paris 1977

(PET) PETERSON J.L.
"Petri Nets" ; Computing Surveys vol.9 - n°3 Sept. 1977

(PET 74) PETERSON J.L., BIELOT T.D.
"A comparaison of models of parallel computation" ;
IFIP Stockholm 1974

(VAL 76) VALETTE R.
"Sur la description, l'analyse et la validation des systèmes de commandes parallèles" ; thèse d'état, Toulouse 1976

(VALK 77) VALK R.
"Self varging nets" ; Journées d'étude AFCET sur les réseaux de Petri, Paris 1977.

C H A P I T R E 3

EXEMPLES PRATIQUES D'UTILISATION DES SCHEMAS DE CABLAGE ET DE CONTROLE

CHAPITRE 3 - EXEMPLES PRATIQUES D'UTILISATION DES SCHEMAS DE CÂBLAGE ET DE CONTRÔLE

3.0 INTRODUCTION

Nous proposons dans ce chapitre d'illustrer par deux exemples particuliers, les techniques spécifiques de la simulation et du contrôle de processus découlant de l'utilisation des schémas de câblage et de contrôle présentés dans les deux chapitres précédents.

Lorsqu'un problème de simulation nécessite l'utilisation d'un couplage numérique analogique, certains processus mis en oeuvre lors de la simulation doivent être considérés comme processus hybride (BEK 68). Un processus hybride résulte de l'association globale et synchronisée d'un câblage analogique, d'un câblage logique et d'un programme numérique chargé de certains calculs et assurant l'enchaînement des différentes actions. Plus formellement, le schéma de processus hybride est tel que certaines actions sont définies par un schéma de câblage.

Un premier exemple illustre globalement ce point de vue. Il s'agit d'un problème relatif à la résolution d'équations aux dérivées partielles.

En dernier lieu un problème relatif au contrôle d'un système industriel, un réseau ferroviaire, permet d'évaluer l'intérêt pratique des schémas de contrôle.

Mais avant d'aborder ces deux exemples, nous nous proposons d'exposer la manière de réaliser concrétement le contrôle numérique d'un processus puisque ce contrôle intervient également dans une simulation.

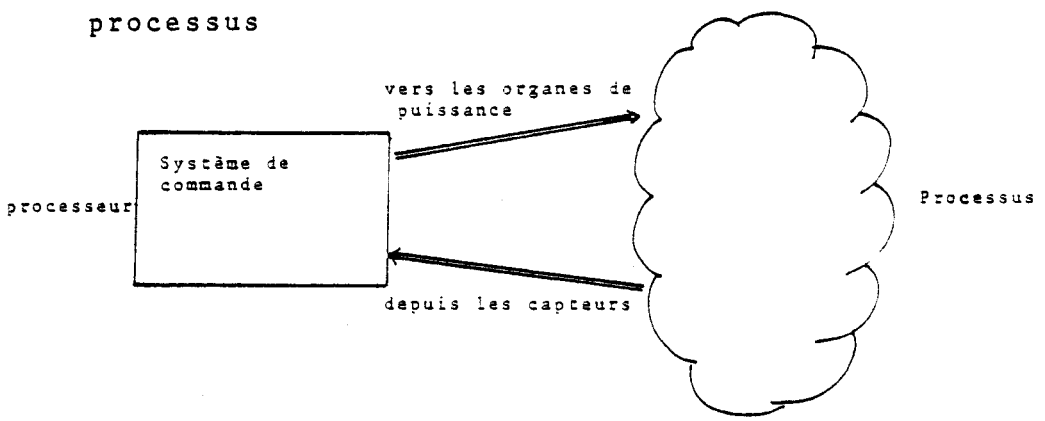
3.1. CONTRÔLE NUMERIQUE D'UN PROCESSUS

31.0 Introduction

Dans le contrôle numérique d'un processus industriel, le problème est de donner les ordres de marche et d'arrêt aux organes de puissance (moteurs, vannes, vérins...) dans une succession définie par le cahier des charges et sans contrôle de deux ensembles d'informations :

(i) celles en provenance du processeur,

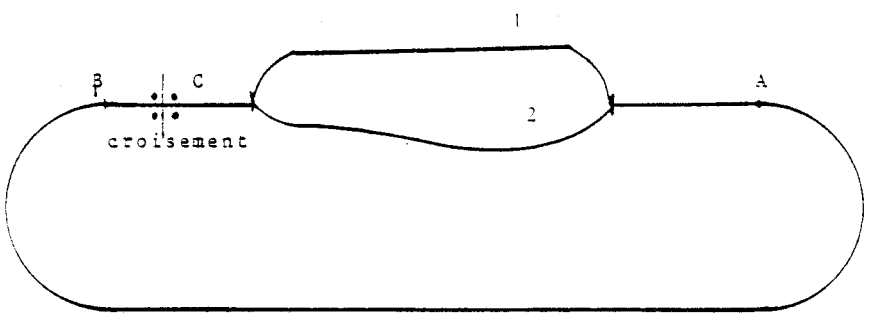
(ii) celles en provenance du processus et de son environnement, sur lequel un certain nombre de capteurs (fin de course...) sont chargés de recueillir les renseignements significatifs quant à l'évolution du processus



Nous proposons d'illustrer ce point de vue par l'exemple suivant :

Exemple : pilotage automatique d'un véhicule

Considérons l'itinéraire routier décrit sur le schéma ci-dessous :



Les tronçons 1 et 2 sont de natures différentes (longueur, état de la route, relief...).
Nous envisageons de contrôler la bonne marche d'un véhicule sur ce circuit, et en reprenant la terminologie du paragraphe précédent nous considérons : comme processus, le véhicule et son environnement et comme processeur le pilote.
Le contrôle d'un véhicule doit tenir compte des impératifs suivants :

(i) les contraintes soumises au véhicule par le circuit tels que l'itinéraire, le profil de la route, les feux de croisement.

(ii) le cahier des charges définissant le plan de route et les critères de choix ou d'optimisation.

(iii) la sécurité du passager face aux diverses perturbations pouvant survenir lors du trajet.

Pour cet exemple élémentaire, divers cahiers des charges peuvent être proposés pour un même processeur

(i) aller du point A au point B en un minimum de temps,

(ii) mener n passagers du point A au point B en minimisant la consommation,

(iii) assurer un service régulier de transport du point A au point B,

(iv) transporter tous les passagers du point A et B en tenant compte de la capacité du véhicule.

Le contrôle de processus doit donc permettre de satisfaire toutes les situations rencontrées dans un cahier des charges.

31.2 Problèmes fondamentaux posés par le contrôle numérique d'un processus.

Dans la suite de ce chapitre, nous n'envisageons que les processeurs du type calculateur numérique.

Les difficultés rencontrées dans le contrôle numérique d'un processus industriel se répartissent en trois catégories, de complexités croissantes.

(i) Catégorie I
Nous dissocions les différents processus composant le système à contrôler et étudions le système de commande d'un processus sans tenir compte des relations existantes entre les taches. Dans ce cas, le processeur est assimilé à un mono-processeur et le système de commande est décrit à l'aide des trois structures classiques de la programmation structurée : séquence, alternative, répétitive.

(ii) Catégorie II : Prise en compte des événements
Le système de commande d'une tâche doit pouvoir tenir compte d'événements aussi bien internes c'est à dire traites au niveau processeurs mais traduisant des relations entre tâches, ou externes, c'est à dire émanant directement du processus.

Exemple : dans le contrôle du véhicule, les divers événements dont le pilote doit tenir compte sont les feux de circulation, l'itinéraire, les données du véhicule (vitesse, consommation, régime moteur...), les piétons, les autres véhicules, l'horaire...

(iii) Catégorie III : Relation du processus avec les autres processus :
Nous retrouvons les trois classes de relations étudiées dans le chapitre précédent, à savoir :

(i) synchronisation,

(ii) partage de ressource, (exemple : voie étroite interdisant le croisement de deux véhicules de front),

(iii) producteur - consommateur.

De plus, à un processus peuvent être attachés les trois états suivants : actif, attente, inactif permettant respectivement de traduire qu'un processus puisse être en phase d'attente d'un événement lui permettant d'atteindre l'état suivant ou enfin de ne pouvoir être active même par erreur par l'un quelconque des événements existants.

31.3 Représentation d'un système de commande par un Réseau de Petri.

Auparavant, le système de commande d'un processus était représenté par une machine séquentielle (DAC 76) dont nous rappelons la définition :

Définition : une machine séquentielle est un quintuplet
M = {X,E,S,f,g},où
X est l'ensemble fini, non vide des états,
E est l'ensemble fini, non vide des entrées (alphabet d'entrée
S est l'ensemble fini,non vide des sorties (alphabet de sortie
f est la fonction'état suivant',qui réalise l'application

$$X \times E \rightarrow X$$

g est la fonction'sortie', qui réalise l'application

$$X \times E \rightarrow S$$

Dans le cas d'un contrôle numérique de processus industriel, l'ensemble E représente les informations en provenance soit des capteurs, soit du pupitre de commande, et l'ensemble S les informations destinées soit aux organes de puissance du processus soit au pupitre de commande.

D'un autre point de vue, une machine séquentielle peut être représentée par un Réseau de Petri en affectant les sorties à un état du réseau et à un événement (DAC 76).
Dans ce cas, on définit : $<E,S,M,\rho,\mu,\sigma>$ où

- $E = \{e_1,\ldots e_n\}$ est un ensemble fini, non vide, d'événements (construits sur l'alphabet d'entrée de la machine)

- $S = \{s_1,\ldots s_k\}$ est un ensemble fini,non vide, de sorties

- $M = \{m_1,\ldots m_i\}$ est un ensemble fini,non vide, de marquages du réseau

- ρ est une application multivoque de M dans E. A chaque élément m de M, elle fait correspondre un sous-ensemble $\rho(m)$ de E auquel la machine est réceptive. ρ est appelée fonction de réceptivité. L'ensemble $\rho(m)$ est composé d'événements qui font évoluer le marquage.

- μ est la fonction de transition (ou marquage suivant)
  $\forall\ m \in M \quad \rho(m) \xrightarrow{\mu} M.$

Pour chaque élement m de M, la fonction μ n'est définie que sur l'ensemble de réceptivité $\rho(m) \underline{C} E$. On écrira, par extension :
$\rho(m,e) = m' \quad (e \in \rho(m))$

- σ est une fonction multivoque (fonction de sortie) :

  $\forall m \in M : \sigma(m) \xrightarrow{\sigma} S$

  Pour chaque élément m de M, σ est définie sur l'ensemble de réceptivité ρ (m) $\subseteq$ E.

  On écrira, par extension : $\sigma(m,e) = S \quad (e \in \rho(m))$

Les fonctions ρ, μ, σ sont représentées sur le réseau de Petri lui-même.

Lors d'une représentation d'un système de commande par une machine séquentielle , une démarche usuelle basée sur différentes méthodes de synthèse (GIR 75, DAC 76) conduit à réduire le nombre d'états en entraînant cependant une complexité extrême pour les systèmes de grandes dimensions.
Lors d'une représentation d'un système de commande on peut se demander s'il faut conserver la même démarche : conduisant à réduire systématiquement le graphe à partir de transformations équivalentes (BOU,DUM). Nous proposons ici de représenter le système de commande par un réseau de Petri sans toutefois réduire le réseau. Il nous est paru essentiel de construire à priori un réseau "propre et vivant", en offrant de ce fait aux utilisateurs non seulement des qualités de fonctionnement idéales mais aussi une méthodologie se rapprochant de celle de la programmation structurée.

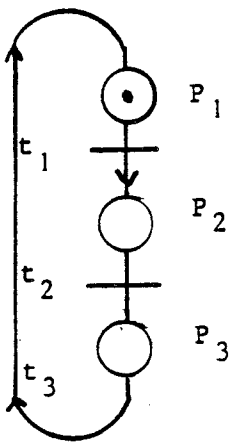
31.4 Schéma de processus et système de commande

Le système de commande de chaque processus industriel est représenté par un schéma de processus structuré $(A,\Sigma,1)$, muni d'une interpretation I. Plusieurs exemples de système de commande seront fournis dans ce chapitre.

Mais afin de tenir compte de l'état du processus industriel qui peut être actif, en attente, inactif ; une fonction ε est définie de l'ensemble des places d'un schéma de processus dans l'ensemble des états possibles : (actif, attente, inactif). Cette fonction ε n'est définie que pour les places marquées du schéma de processus.

Exemple : soit le schéma de processus structure :

$\mathcal{P}$:



$\varepsilon (P_1) = \varepsilon (P_2)$ = actif

$\varepsilon (P_3)$ = inactif

Le processus réalise la tâche associée aux transitions (t_1, t_2) et est inhibé e.

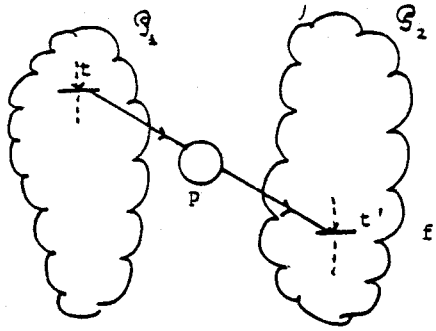
31.5 Synchronisation des processus industriels.

Pour tenir compte de toutes les synchronisations que nous pouvons rencontrer dans le contrôle de processus industriel, nous avons introduit les divers événements suivants.

(i) événement externe au processeur, fugitif ou mémorisé
(ii) événement provenant des autres processus, fugitif ou mémorisé,
(iii) fonctions d'événements (fonctions Bouleennes. OU et ET)

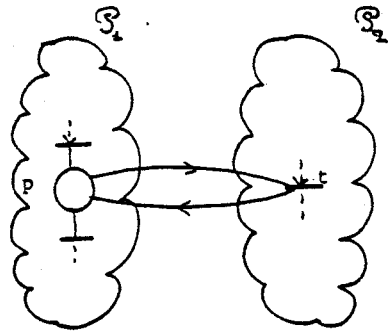
Les événements de type (ii) se traduisent par des liaisons de type (t,p,t') ou (p,t) comme exposés dans le chapitre précédent (figures a et b)

figure a



événement mémorisé échangé entre les 2 schémas de processus 1 et 2

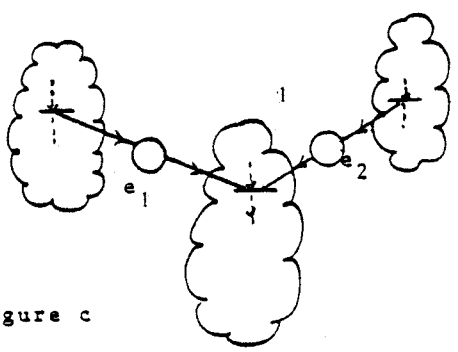
figure b



événement fugitif échangé entre les 2 schémas de processus 1 et 2

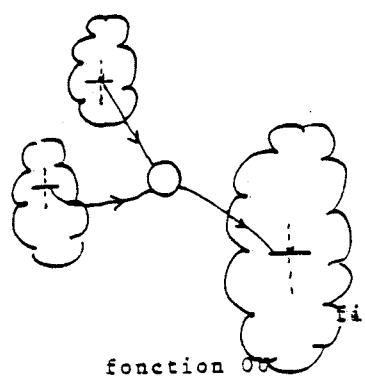
Les événements de type (iii) peuvent être pris en compte par des liaisons de longueurs 2 (figure c et d)

figure c



fonctions : e, e_2

figure d



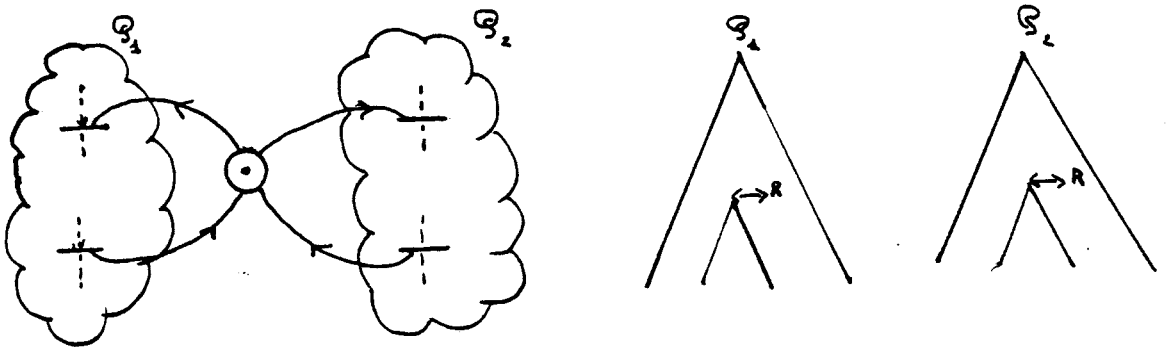
fonction 00

31.6 Relations de partage de ressource.

Les utilisateurs ont à leur disposition deux types de relations de partages de ressources suivant le nombre d'état distinct présenté par la ressource (1 ou 2).

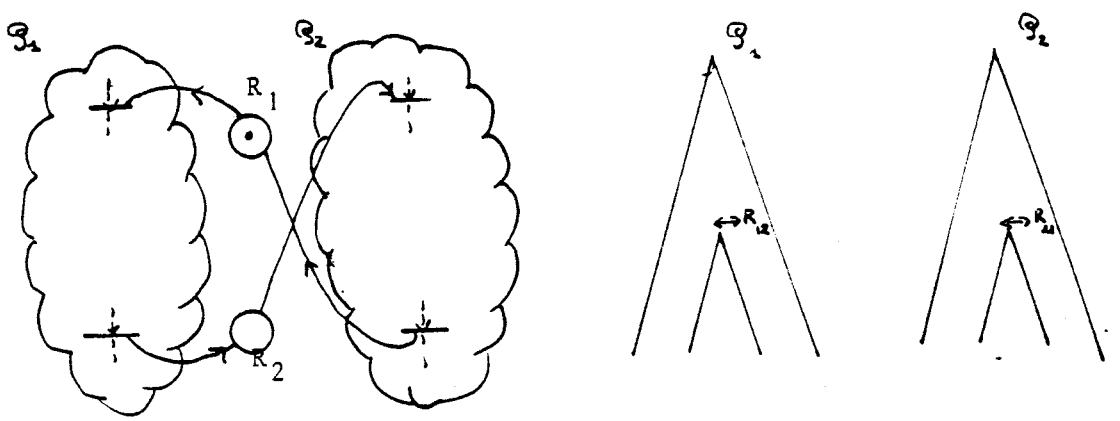
Nous rappelons sur les figures e et f la représentation de ces relations lorsque les schémas de processus sont structurés.

figure e partage d'une ressource possédant 1 état r



utilisation des réseaux de Petri — Utilisation des arbres programmatiques

figure f partage d'une ressource possédant 2 états r_1 et r_2



utilisation des réseaux de Petri — Utilisation des arbres programmatiques

31.7 Implementation de cette structure de contrôle (COR 78 et DAL 79)

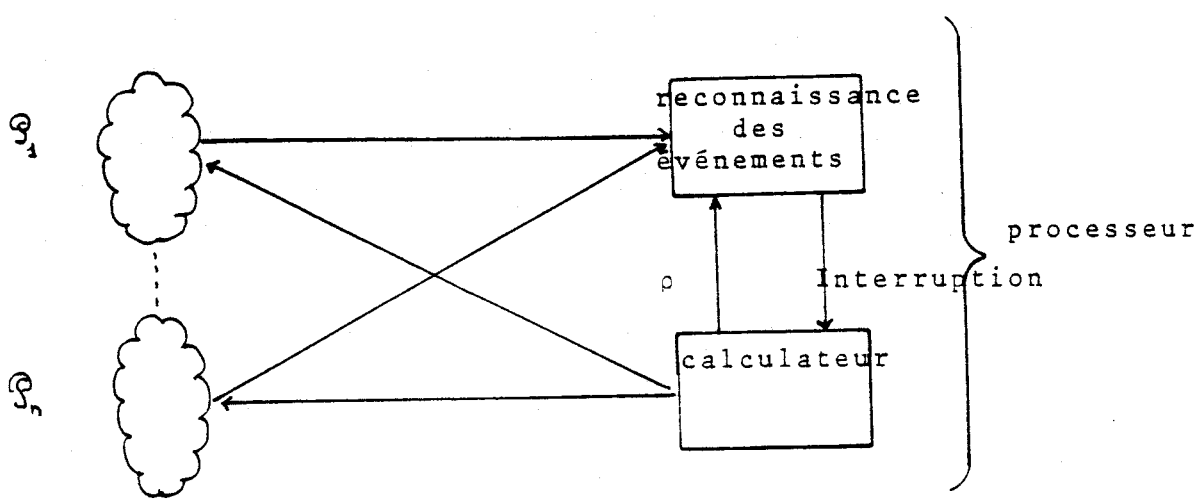
(i) aspect matériel

La représentation d'un réseau de Petri $<E,S,M,\rho,\mu;\sigma>$ sur un calculateur peut se faire de la façon suivante :

- à un événement, $e \in E$, est associé une interruption ou/et une entrée sur le processeur,
- au marqueur du réseau de Petri correspond le pointeur d'instruction,
- à la fonction de réceptivité ρ correspond le masquage des interruptions associées et/ou entrées associées aux événements,
- à la fonction de marquage μ correspond l'évolution du pointeur,
- à la fonction de sortie σ les diverses actions à réaliser.

L'intérêt de cette représentation est l'utilisation de la structure prioritaire du calculateur pour gérer les événements.

La représentation d'un système de gestion de processus $\mathcal{P}_1,\ldots\mathcal{P}_n$ peut être symbolisée par le schéma suivant :

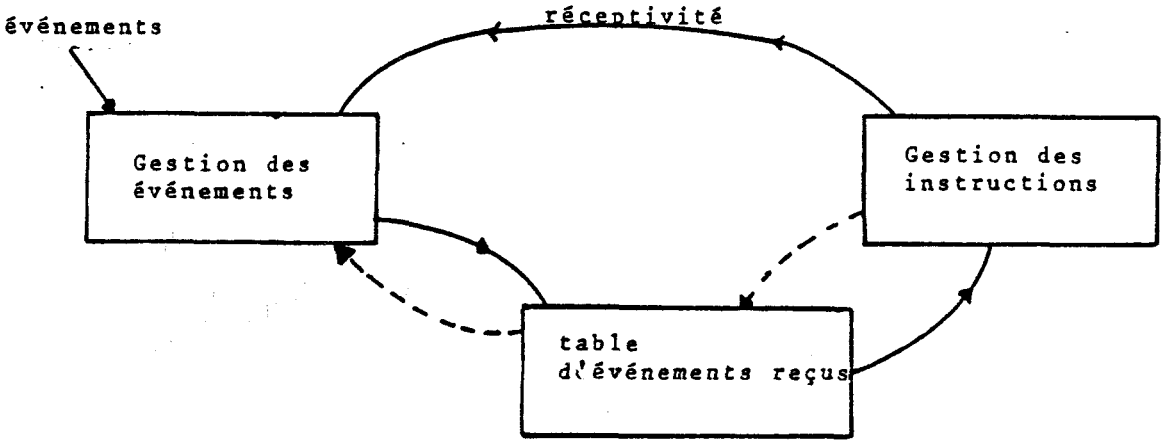


D'une manière pratique, le traitement peut être effectué à deux niveaux :

(a) traitement des événements et de la réceptivité,

(b) traitement de l'évolution du marquage et des taches associées

Les événements provenant des divers processus industriels à l'intention d'autres processus industriels transitent par la logique de commande. Cette logique de commande reconnaît l'événement reçu et interrompt le calculateur qui se charge de l'action appropriée.



Le calculateur, suivant la nature de l'événement, met à jour

- soit une table d'événements fugitifs,

- soit celle des événements mémorisés.

La logique de commande choisit, à chaque instant, les événements en attente permettant une évolution des processus. Ce schéma peut être réalisé plus ou moins parfaitement suivant le type de calculateur employé. Les réalisations que nous proposons dans la suite de ce chapitre nous ont conduit à mettre en oeuvre successivement :

(a) un calculateur industriel P 856 (Philips) disposant d'une structure prioritaire à 64 niveaux hiérarchisés mais qui ne peuvent être masqués individuellement

(b) un microprocesseur 8085 (Intel) utilisant les circuits d'interruptions 8259 et permettant une implementation convenable de la méthode proposée.

(ii) language associé à la description des chémas de processus

a - Langage

l'analyse et la synthèse de la commande d'un processus industriel conduit à une transcription directe du cahier des charges sous forme d'un système de schéma de processus. Ce système se prête très bien à une étude de ces propriétés : vivant, propre et chaque schéma de processus constitue l'arbre programmatique du programme associé.
La description programmée de ce système est facilitée par la mise en oeuvre d'un langage approprié (COR 78, DAL 79). Ce langage est adapté à la programmation structurée, offre la possibilité d'activer et d'inhiber des taches.
Les événements étiquettent les instructions ou bloc d'instructions. A la fonction de réceptivité sont associées des instructions de démasquages et de mémorisation d'événements.
Plusieurs instructions permettent de gérer les deux types de ressources (1 ou 2 états). Il convient de préciser qu'il nous a paru intéressant de pouvoir associer aux requêtes de ressource un niveau de priorité.

b - Compilateur

Un compilateur fournit un langage objet adapté à différents calculateurs ou microprocesseur.

Ce compilateur offre plusieurs avantages

1 - transportabilité

Le compilateur est écrit dans un langage de haut niveau (Basic) et peut donc être adapté sur la plupart des calculateurs possédant un interpréteur Basic .

2 - extensibilité du code opération

Ce langage permet à l'utilisateur d'étendre ou de restreindre facilement le jeu d'instruction afin de l'adapter à son problème. Nous avons utilisé cette propriété lors de son adoption aux deux calculateurs précédemment cités.

3 - le compilateur est inter-actif

Ce qui facilite la mise au point des programmes.

c - Logiciel adapté au langage et à la structure-machine

Le langage-objet fourni par le compilateur est interprété sur le calculateur cible (calculateur industriel ou micro-processeur). L'interpréteur gère le parallélisme des schémas de processus, c'est à dire qu'il se charge de l'enchaînement des actions simultanées en prenant en compte les événements émis par les processus industriels. Il doit également gérer le partage des ressources. Enfin, l'interpréteur réalise les différentes actions déclenchées par les schémas de processus : E/S diverses, opérations arithmétiques et logiques...

3.2 SIMULATION DU CHOC THERMIQUE

32.1 Description du problème étudié

Nous proposons à titre d'illustration de résoudre un problème de conduction thermique usuel mieux connu sous le nom de "choc thermique". Il s'agit essentiellement d'étudier par simulation la dynamique de la température θ d'un mur d'épaisseur constante soumis à un échelon de température à partir d'un état initial supposé en équilibre.

La Loi de Fourier exprimant le principe de base de la conductibilité thermique conduit à exprimer l'évolution dynamique de la température d'une masse sous la forme du modèle suivant :

$$\text{Div}\ [\lambda\ \text{Grad}\ \theta\ ] = \rho.c.\ \frac{\partial\theta}{\partial t} \qquad (1)$$

ρ et c sont des constantes désignant respectivement la masse volumique du mur, et sa chaleur massique. Nous supposerons λ constant et le mur de dimension infini, ce qui permet d'étudier le problème dans la seule dimension relative à l'épaisseur du mur, en négligeant les effets limites.

Dans ces conditions l'équation (1) peut être écrite sous la forme simplifiée suivante :

$$\begin{cases} \lambda = b = \text{constante} \\ b.\ \dfrac{\partial^2\theta}{\partial m^2} = \rho.c.\ \dfrac{\partial\theta}{\partial t} \end{cases} \qquad (2)$$

Cette équation (2) ne comporte que deux variables : la variable d'espace n et le temps. La résolution d'un système de cette nature peut être envisagée selon la méthode proposée par Marquette (MAR 77).

Nous ne discrétisons que la seule variable t, nous obtenons un modéle décrit par un système différentiel par rapport à la variable x. Dans ce cas, il est possible dans un premier temps d'envisager une solution analogique par câblage répétitif d'une cellule de base. Nous nous proposons donc de résoudre le système à partir de la gestion numérique d'un sous-programme analogique affecté à la résolution d'une cellule de base selon une technique itérative (MAR 77) .

Selon que l'on envisage la résolution dans le sens rétrograde ou dans le sens normal d'évolution du temps la discrétisation de la variable t dans l'équation (2) conduit aux deux modèles (3) et (4)

$$\frac{\lambda}{\rho c}\ \frac{\partial^2 \theta_i}{\partial x^2} = \theta_i - \theta_{i-1} \quad , \text{ t croissant} \qquad (3)$$

$$\frac{\lambda}{\rho c}\ \frac{\partial}{\partial x^2} = \theta_{i+1} - \theta_i \quad , \text{ t rétrograde} \qquad (4)$$

Nous supposerons les changements de variables effectués. En conséquence θ représente la température en variables réduites évoluant dans l'échelle $[0,1]$, $\theta \in [0,1]$, $x \in [0,1]$

Les conditions limites variables à un instant t_i donné sont donc sur la surface du mur $\theta\ (1,t_i) = 1$, et à l'intérieur du mur $\theta\ (n,t_i) = 0$, $\forall\ x \in [0,1]$

Les équations différentielles (3) ou (4) étant du second ordre il est nécessaire de se donner une condition initiale sur la dérivée $\frac{\partial \Theta}{\partial x}$. Celle-ci n'est pas donnée à priori par les conditions limites, il est donc nécessaire de prévoir un algorithme de recherche paramétrique de cette variable.

Marquette (MAR 77) a montré que le choix du modèle de simulation de la cellule de base dans le sens des équations (3) ou (4) peut être déduit d'une étude de la stabilité du système décrit par l'équation aux dérivées partielles (1). Suite à cette étude, nous choisissons le modèle associé à l'équation (3).

32.2 Méthode de résolution.

Nous proposons donc pour résoudre une cellule de base de simuler l'équation différentielle (3) en recherchant préalablement la condition initiale sur la dérivée($\Theta (0,t_i)$)

La solution obtenue $\Theta (n,t_i)$ est enregistrée pour servir de condition initiale au pas suivant de résolution à l'instant t_{i+1}.

La solution de l'équation différentielle (3) est définie de manière unique à partir des conditions aux limites $\Theta (0,t_i)$, $\Theta (1,t_i)$. Une résolution itérative conduit à effectuer une recherche paramétrique de la condition initiale $\mathring{\Theta} (0,t_i)$. La contrainte à respecter sur la condition finale $\Theta (1,t_i)$ permet alors de valider ou de rejeter la solution obtenue.

32.3 Schéma de contrôle

Le système de schéma de processus définissant cette simulation se compose de deux schémas de processus $\mathcal{P}_1$, $\mathcal{P}_2$ et d'un graphe de liaison L.

Le premier schéma de processus $\mathcal{P}_1$ gère la recherche de la condition initiale $\overset{\circ}{\theta}$ et l'enregistrement des valeurs de la température θ pour la prochaine étape.

Le second schéma de processus $\mathcal{P}_2$ visualise les résultats obtenus sur un périphérique externe (traceur de courbe, oscilloscope).

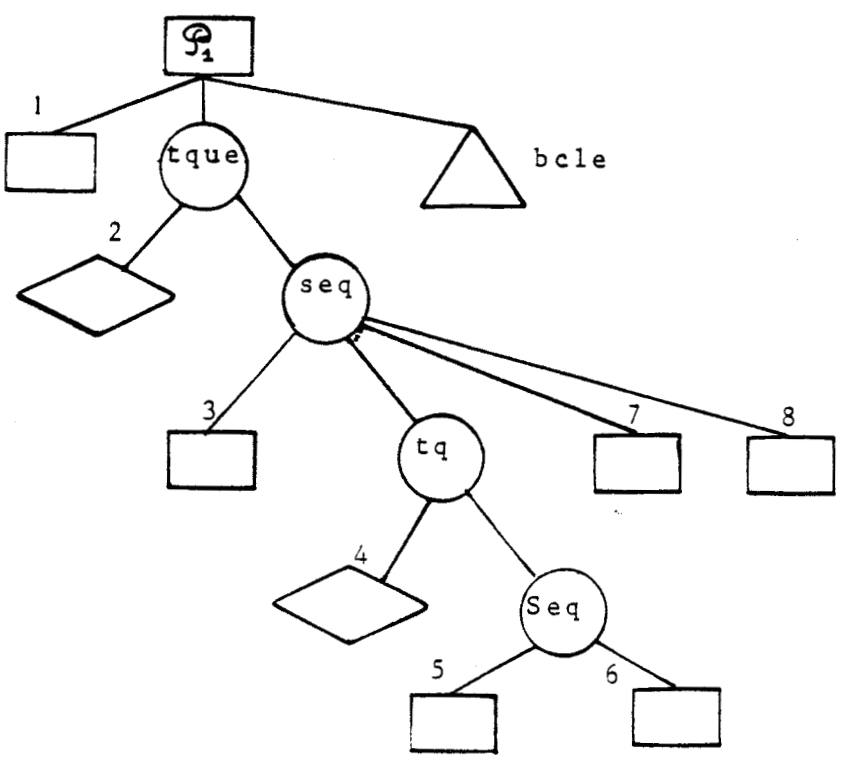
Le graphe de liaison L assure l'échange d'information entre les deux schémas de processus $\mathcal{P}_1$ et $\mathcal{P}_2$.

323.1 Le schéma de processus $\mathcal{P}_1$.

Le schéma de processus $_1$ peut être caractérisé par l'arbre programmatique étiqueté (A,Σ,λ), où :

(i) $\Sigma = \mathcal{A} \cup \mathcal{P}$ avec $\mathcal{A} = \{a,b,c,d,e,f\}$
$\mathcal{P} = \{p,q\}$

(ii) A =

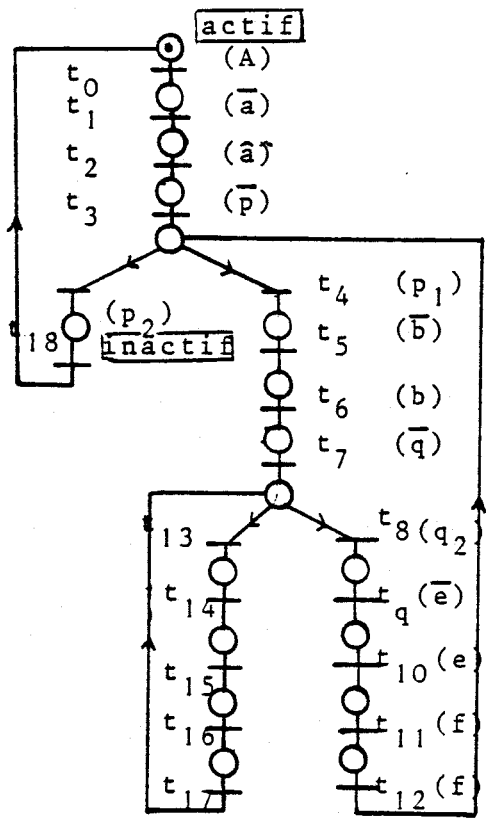


(iii) 1 : $1(1) = a$, $1(2) = p$, $1(3) = b$,
$1(4) = q$, $1(5) = c$, $1(6) = d$,
$1(7) = e$, $1(8) = f$

Ce même processus physique $\mathcal{P}_1$ peut être représenté par le schéma de processus $(R, \Sigma_1, 1_1)$ où
$\Sigma_1 = \mathcal{A}_1 \cup \mathcal{P}_1$ avec $\mathcal{A}_1 = \{\bar{a},\bar{b},\bar{c},\bar{d},\bar{e},\bar{f}\} \cup \{a,b,c,d,e,f\}$

$$\mathcal{P}_1 = \{\bar{p},\bar{q}\} \cup \{p_1,p_2,q_1,q_2\}$$

R_1 =



actif

inactif

symbolisent respectivement les mises en activation et désactivation du schéma de processus $_1$.

1_1 : $1_1(t_0) = \Lambda$, $1_1(t_1) = \bar{a}$, $1_1(t_2) = a$,
$1_1(t_3) = \bar{p}$, $1_1(t_4) = p_1$, $1_1(t_5) = \bar{b}$, $1_1(t_6) = b$,
$1_1(t_7) = \bar{q}$, $1_1(t_8) = q_2$, $1_1(t_q) = \bar{e}$, $1_1(t_{10}) = e$,
$1_1(t_{11}) = \bar{f}$, $1_1(t_{12}) = f$, $1_1(t_{13}) = q_1$, $1_1(t_{14}) = \bar{c}$,
$1_1(t_{15}) = c$, $1_1(t_{16}) = \bar{d}$, $1_1(t_{17}) = d$, $1_1(t_{13}) = p_2$

Donnons une interprétation I de l'arbre programmatique étiqueté $(A,\Sigma,1)$:

(i) interprétation des actions

$= \{a,b,c,d,e,f,\}$

I (a) ⟶ initialisation générale

I (b) ⟶ première initialisation de $\mathring{y}$

I (c) ⟶ calcul de $\mathring{y} + \varepsilon$

I (d) ⟶ résolution de l'équation (3) par calcul hybride

I (e) ⟶ enregistrements des valeurs

I (f) ⟶ passage au pas suivant

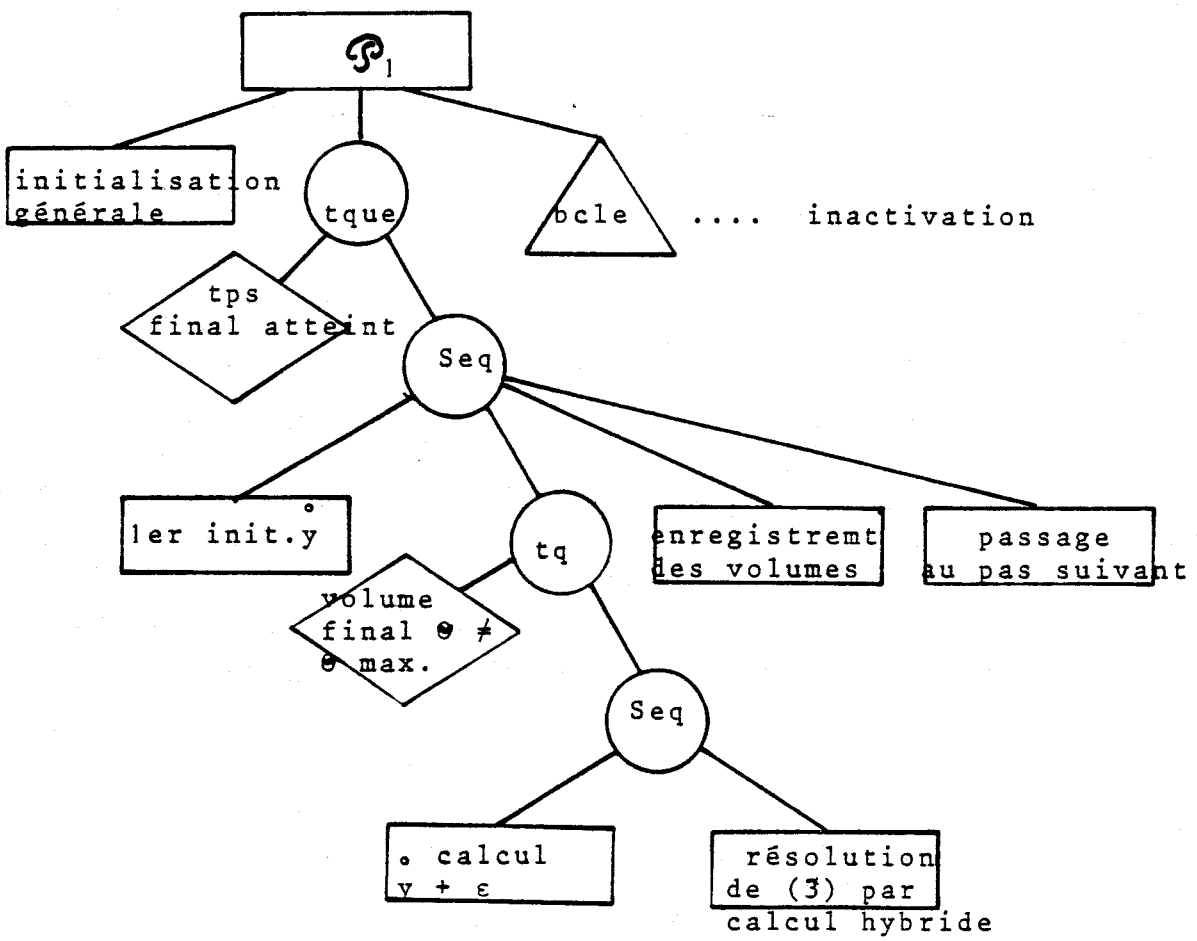
(ii) Interprétation des prédicats

$= \{p,q\}$

et I (p)—— le temps final de l'expérimentation est-il atteint ?

I (q)—— la valeur finale de θ est-elle différente de θ max ?

Arbre programmatique interprété en utilisant I



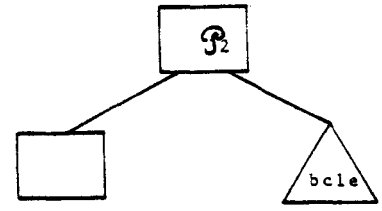
L'interprétation I_1 du schéma de processus structuré se déduit aisément de l'interprétation I

323.2 Le schéma de processus $\mathcal{P}_2$

Le schéma de processus $\mathcal{P}_2$ se caractérise par l'arbre programmatique étiqueté (A',Σ',1') où :

(i) $\Sigma' = \mathcal{A}'$ avec $\mathcal{A}' \{d'\}$

(ii) A' =

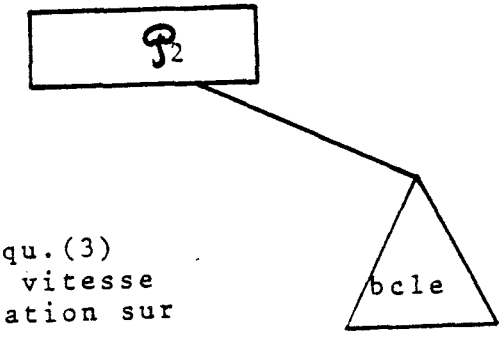


(iii) 1 : 1 (9) = d'

Donnons une interprétation I' de l'arbre programmatique étiqueté (A',Σ',1') :

I' (d') —— résolution de l'équation (3) par calcul hybride à vitesse lente et visualisation des températures sur sur table traçante.

L'arbre programmatique interprété devient donc :



résolution de l'équ.(3) par calcul hyb. à vitesse lente et visualisation sur table

323.3 Etude du graphe de liaison L :

Les deux schémas de processus $\mathcal{P}_1$ et $\mathcal{P}_2$ sont liés par une relation producteur-consommateur et se partage la ressource à 2 états R.

La liaison se situe aux noeuds (7) de $\mathcal{P}_1$ et (9) de $\mathcal{P}_2$: le noeud (7) nécessite la ressource dans l'état 1 et le noeud (9) dans l'état 2.

L'interprétation I'' de cette liaison est la suivante : le schéma de processus $_1$ calcule une valeur initiale $\mathring{y}$ à grande vitesse et la communique au second processus $\mathcal{P}_2$, qui, grâce à cette valeur peut résoudre l'équation à vitesse lente. Ce découpage des fonctions entre les deux schémas de processus assure une meilleure utilisation du matériel.

32.4 Etude de l'action d

L'action d a une double finalité :

(i) résoudre l'équation différentielle suivante

$$\frac{\lambda}{\rho \cdot c} \frac{\partial^2 \theta_i}{\partial x^2} = \theta_i - \theta_{i-1},$$

(ii) vérifier si la valeur θ_{max} est atteinte.

Au cas où l'interprétation de l'action d nous conduit à choisir une résolution de type hybride, nous pouvons définir I (d) par un schéma de câblage muni d'une interprétation I_o.

Le schéma du câblage étant :

$T = \langle I.\ (1d^2 \otimes I\ (1d^2 \otimes \Sigma_2)\rangle\ \langle 1,2,1,3,4,5\rangle\langle m,a,b,T,c\rangle$

$L = \langle N.\ (C.(1d, \otimes \Sigma_3) \otimes C\ (1d, \otimes \Sigma_3\ (1d_2 \otimes I_1))\ )\rangle$

$\langle 1,4,2,3,1,4,2,3\rangle\ \langle d,e,f,T\rangle$

avec $D = \{u,v\}$

et $m \in \mathcal{F}^{u}_{01}$

$a,b,c,d,e,f \in \mathcal{F}^{\sigma}_{1}$

$I \in \mathcal{F}^{v}_{uv^2}$

$\Sigma_2 \in \mathcal{F}^{v}_{uv^2}$

$T \in \mathcal{F}^{v}_{v}$

$L \in \mathcal{F}^{u}_{.u}$

$N \in \mathcal{F}^{u}_{.u^2}$

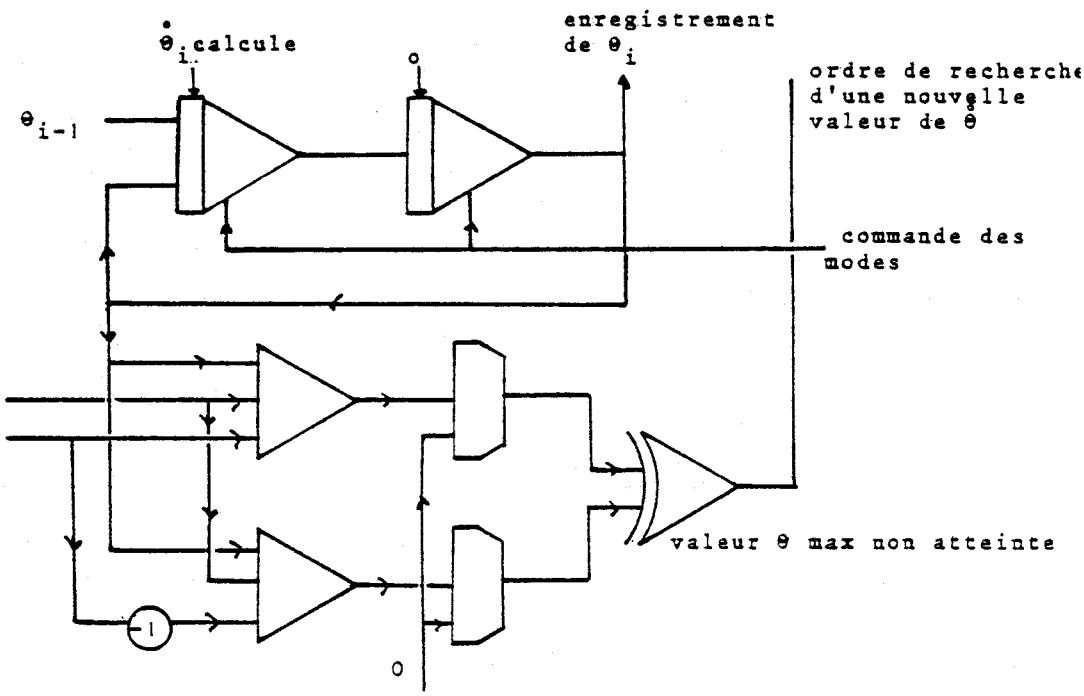
$C \in \mathcal{F}^{u}_{v^2}$

$I_1 \in \mathcal{F}^{v}_{v}$

L'interprétation I_0 du domaine de calcul et des divers opérateurs étant :

u : CG (2)
v : R
m : commande des modes de l'intégrateur
a : constante 0
b : constante $\overset{\circ}{\theta}_i$
c : constante θ_{i-1}
I : intégrateur analogique
I_1 : inverseur
Σ_2 : sommateur
N : ou exclusif
C : comparateur
d : constante 0
e : constante - 1
f : constante ε

Pour mémoire le câblage de la calculatrice hybride serait :

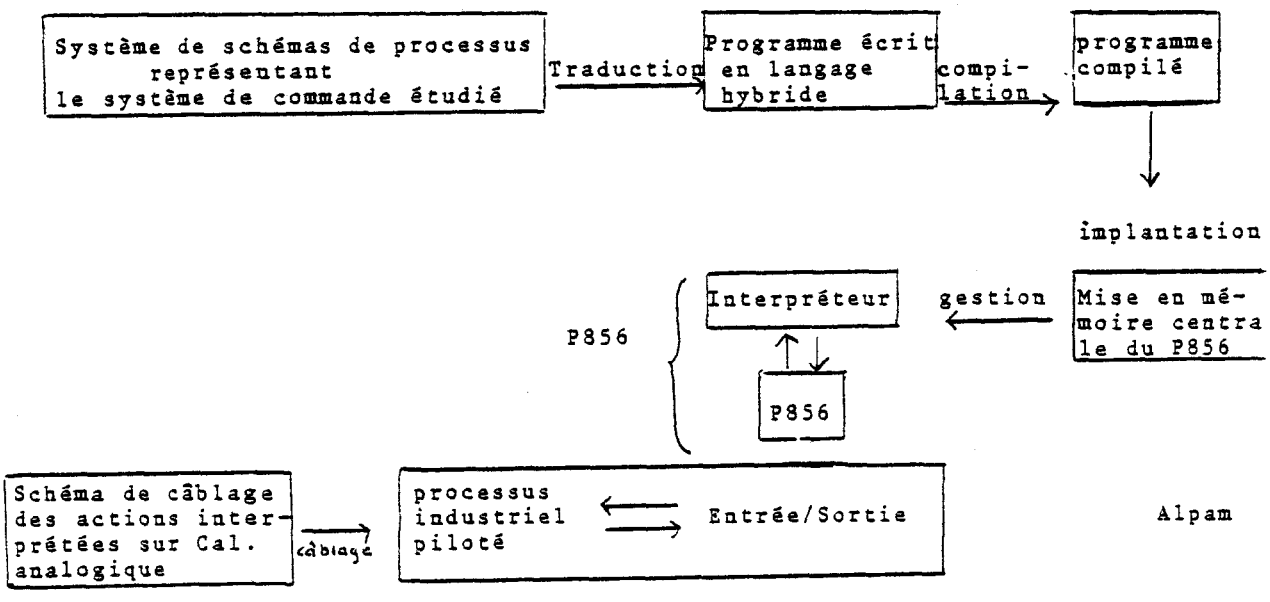


32.5 Réalisation concrète de cette simulation.

La simulation décrite nécessite le couplage d'un calculateur numérique et d'un calculateur analogique. Cette connexion est réalisée, dans le cas présent, entre un calculateur numérique Philips P 856 et un calculateur analogique ALPAM 300 (MAR 76).

La programmation des schémas de processus a nécessité l'élaboration d'un logiciel de simulation hybride ou INTERPRETEUR, associé à un langage hybride particulier (DAU 79).

La figure suivante définit les différents traitements à effectuer, depuis le schéma de processus jusqu'au déroulement de la simulation sur la calculatrice hybride.



32.6 Conclusion

Le système de simulation présenté constitue sur le plan matériel et logiciel un outil remarquablement adapté à l'étude des processus industriels. Il permet une mise en oeuvre directe des méthodes d'analyse et de synthèse des systèmes de commande grâce à un logiciel élaboré autour du concept de schéma de contrôle à une méthodologie qui représente la figure précédente.

L'utilisation de deux calculateurs de natures différentes et d'un interpréteur chargé de les piloter permet de répartir les tâches tout en adaptant au mieux les opérateurs disponibles qu'il s'agisse des éléments logiques, analogiques ou des fonctions numériques.

Le calculateur analogique donne au système la possibilité d'une simulation en temps réel, accéléré ou ralenti des processus d'évolution continue. Les opérateurs hybrides associés aux possibilités de mémorisation, de calcul arithmétique et de décision du calculateur numérique permettent d'agir à tout moment sur l'évolution de ce processus.

Les domaines d'applications d'un tel système sont multiples. En effet, la possibilité offerte par le calculateur analogique d'un traitement parallèle de l'information peut dans un premier temps permettre de décharger le calculateur digital des algorithmes d'intégration et de résolution des systèmes différentiels. Le calculateur analogique est tout à fait adapté aux traitements de cette nature. Un autre domaine d'utilisation, extrêmement important concerne les problèmes relatifs à la simulation et au contrôle des processus. Le logiciel hybride a permis d'étendre très sensiblement le champ potentiel des utilisateurs possibles.

En effet, l'appréhension de la méthodologie et du langage hybride est relativement rapide, et conduit à mettre rapidement ce matériel à la portée d'utilisateurs non spécialisés.

3.3 COMMANDE DE MACHINES NUMERIQUES OU DE PROCESSUS INDUSTRIELS

33.0 Introduction

L'utilisation de machine à commande numérique prend une importance croissante dans le domaine industriel. Après la logique câblée, puis programmée (automate programmable), la commande numérique par microprocesseur nous apparaît comme étant à l'heure actuelle la plus adaptée aux problèmes de gestion de processus.

Le système présenté repose sur l'utilisation des schémas de contrôle, exposé dans le chapitre 2. Ce système ne représente qu'un premier pas vers le développement d'un matériel et surtout d'un logiciel destiné à simplifier au maximum la tâche d'un utilisateur non informaticien.

Le domaine d'application du système de commande réalisé est celui des automatismes par tout ou rien ou automatismes logiques, il peut toutefois s'étendre aisément aux contrôles de type analogique et numérique.

Le processeur chargé de réaliser le "système de commande" de notre système est conçu autour de circuits microprocesseurs afin de pouvoir adapter la configuration en fonction du processus à piloter (COR 78,DAL 79).

Dans la suite de ce chapitre, nous mettrons en évidence l'intérêt d'un schéma de contrôle, en étudiant un problème relatif au contrôle d'un réseau ferroviaire. Cette maquette de réseau ferroviaire nous offre une structure très riche en entrées-sorties et fait apparaître un certain nombre de conflits internes.

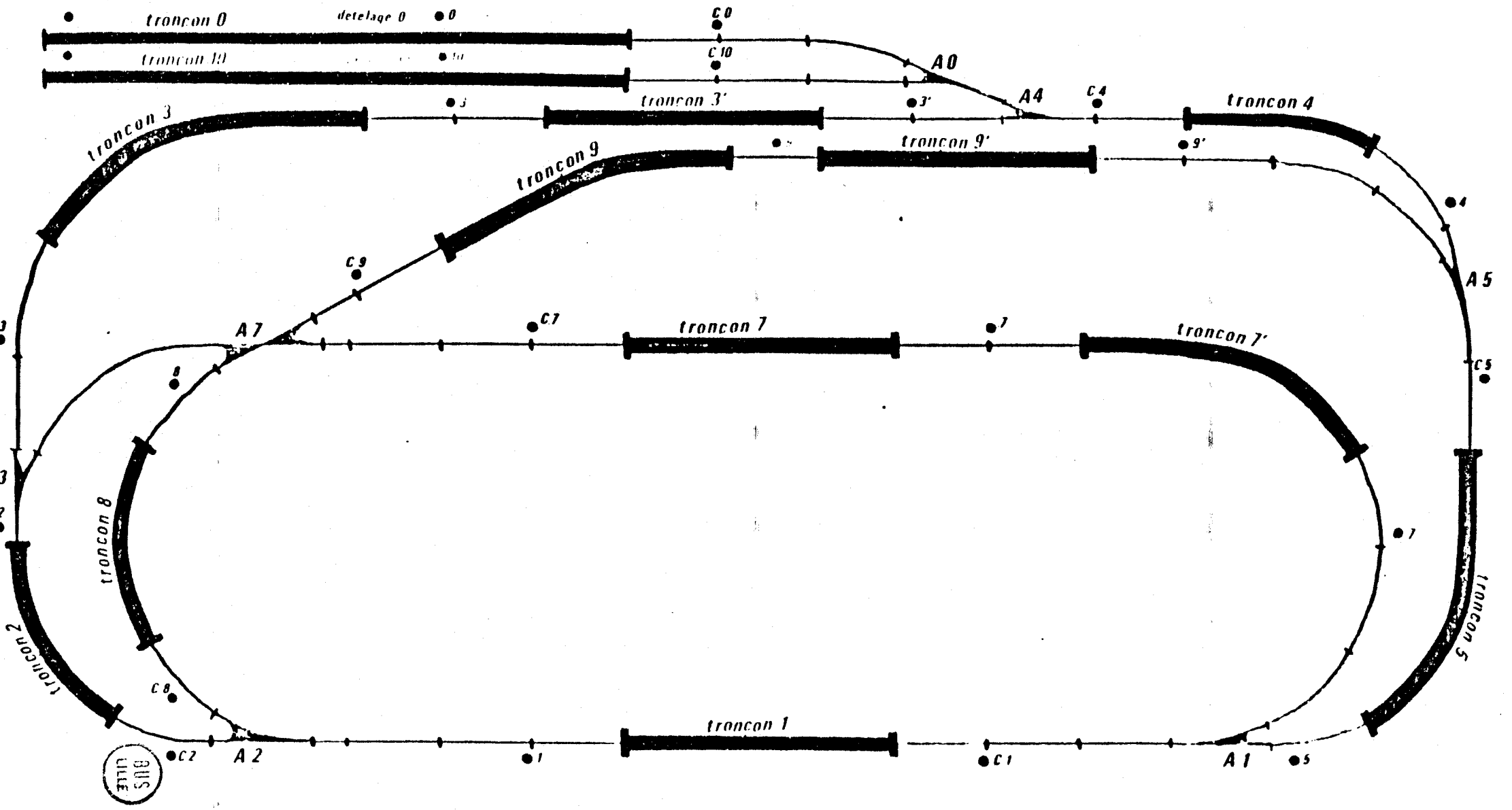
33.1 Présentation de la maquette du réseau ferroviaire

Le réseau, que décrit la figure suivante, comporte six aiguillages et un croisement qui sont autant de ressources critiques . Le réseau se subdivise en zones alimentées en permanence et en zones appelées tronçons , pouvant être alimentés ou non, régulés en vitesse ou pouvant recevoir un signal permettant l'inversion du convoi si toutefois celui-ci est à l'arrêt sur le tronçon correspondant.

Ces tronçons constituent les sorties du système qui doit être piloté. Les entrées de ce système sont constituées par des capteurs placés en bordure des zones alimentées en permanence. Ces capteurs délimitent l'entrée et la sortie d'un tronçon, et indiquent le passage d'un élément de traction (locomotive, wagon).

Par un traitement numérique, il est alors possible d'identifier la longueur et la vitesse du convoi.

REPERAGE DU RESEAU



33.2 Logique de commande chargé de la surveillance des convois (COR 78)

La logique de commande est implémentée comme exposé au paragraphe 31.7, le processeur choisi est un microprocesseur muni d'une structure prioritaire élaborée.

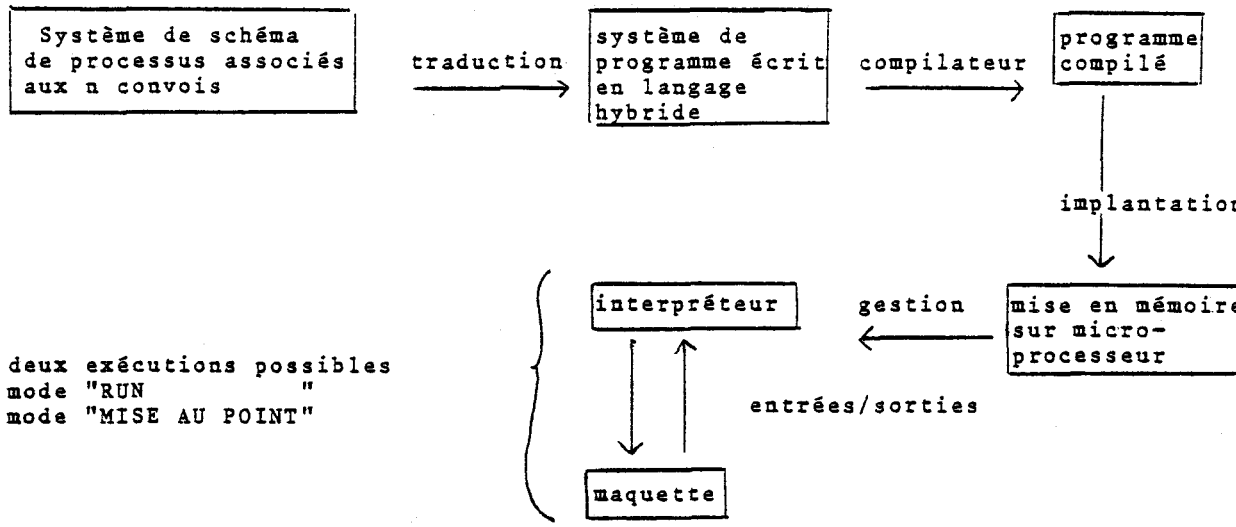
Chaque convoi constitue une tâche à laquelle est associé un schéma de processus interprété sous forme d'un programme destiné au micro processeur.

Ce schéma de processus interprété, $<E,S,M,\rho,\mu,\sigma>$, est défini comme suit :

- à un événement e $\in$ E, (information en provenance de l'un des capteurs ou du pupitre de commande) est associé un niveau d'interruption,
- au marqueur du réseau de Petri correspond le pointeur d'instruction du programme,
- à la fonction de marquage μ correspond l'évolution du pointeur d'instruction,
- à la fonction de réceptivité ρ correspond le masquage des interruptions associées aux événements,
- à la fonction de sortie σ, les commandes d'alimentation des tronçons sélectionnés par le programme.

Le système de commande actuellement réalisé permet le pilotage de huit convois simultanés.

La figure suivante caractérise les différents traitements à effectuer pour passer du cahier des charges à l'expérimentation de la maquette ; ces différentes étapes comprennent notamment l'analyse et la synthèse du problème posé ainsi que l'écriture et la mise au point des programmes.



Remarque : cette chaîne de transformation réalisée, ici, pour un microprocesseur ne se différencie de celle associée au calculateur industriel P 856, que par l'interpréteur, le langage hybride et le compilateur restant le même dans les deux cas. (COR 78, DAL 79)

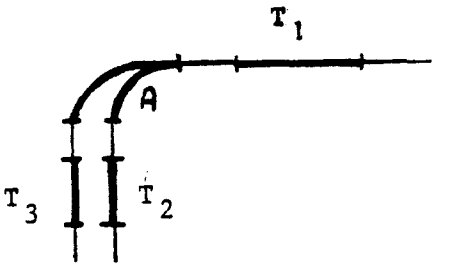
33.3 Etudes de quelques exemples élémentaires

L'étude d'un cas complexe sera précédée de celle de quelques situations simples mais fréquentes afin de préciser notre formalisme.

333.1 Choix d'un tronçon à la sortie d'un aiguillage

a - définition du problème

Considérons un aiguillage A et trois tronçons T_1, T_2, T_3, que symbolise le schéma suivant :



Le train se trouve immobilisé sur le tronçon T_1 et doit prendre l'un des deux tronçons T_2, T_3, en tenant compte du cahier des charges (Plan de route). Deux cahiers des charges peuvent être proposés :

(i) le choix du tronçon T_2 ou T_3 est imposé(règle A

(ii) le train se dirige vers le tronçon libre, mais si les deux sont disponibles, le choix est imposé au sens d'un critère déterministe (règle B)

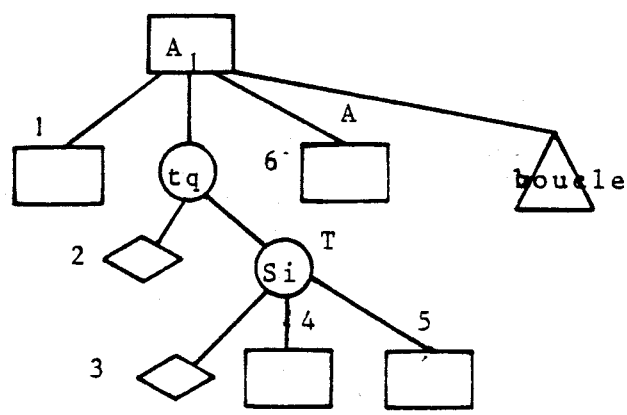
Etudions successivement les deux cahiers des charges qui résument des situations très fréquentes.

3331.b - règle A

Nous considérons l'aiguillage A comme une ressource pouvant être partagée par tous les convois se déplaçant sur le réseau. Nous associons à chaque tronçon T_1, T_2, T_3 un prédicat p_1, p_2, p_3 indiquant si le tronçon est libre ou occupé. Ces prédicats sont fonction des événements en provenance des capteurs, et sont rangés dans une table considérée comme une seconde ressource partagée par tous les convois.

Le schéma de processus structure associé au train est défini par l'arbre programmatique étiqueté, (A_1, Σ_1, l_1), suivant :

.. A_1 =



.. Σ_1 = avec $\{a,b,c,d\}$

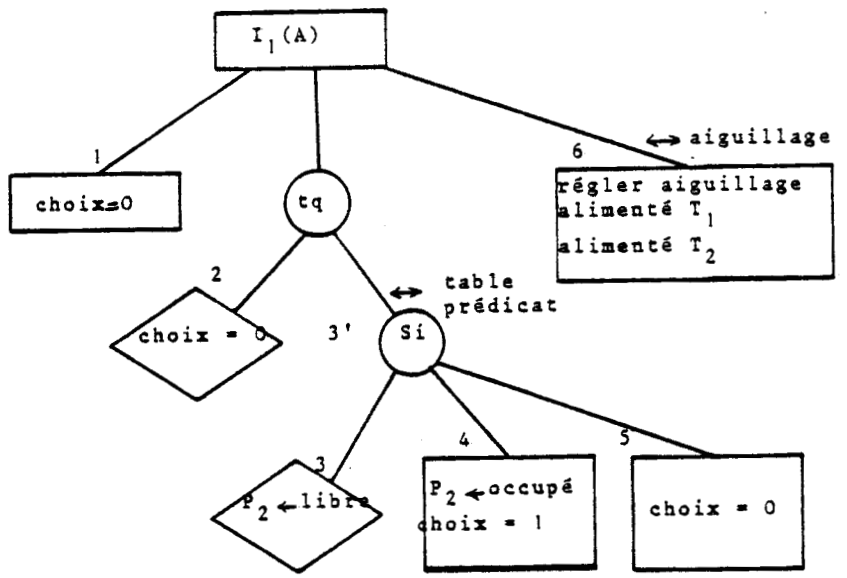
$\Phi_1 = \{p,q\}$

.. fonction d'étiquetage l_1 :
$l_1(1)=a$, $l_1(2)=p$, $l_1(3)=q$, $l_1(4)=b, l_1(5)=c, l_1(6)=d$

.. la liaison de ce schéma de processus avec les autres schémas de processus tient en deux ressources partagées T et A ; la ressource T au niveau du noeud 3' et la ressource A au niveau du noeud 6.

Une interprétation, I_1, possible de ce schéma de processus structuré est fournie dans le cas où le choix du tronçon se porte sur T_2 :

$I_1(A)$ =



Cet exemple met en évidence un mécanisme que nous rencontrerons fréquemment à savoir : détecter si les tronçons en aval d'une ressource critique du réseau (croisement, aiguillages) sont disponibles (noeud 2) avant de s'engager sur celle-ci (noeud 6).

3331.c - règle B

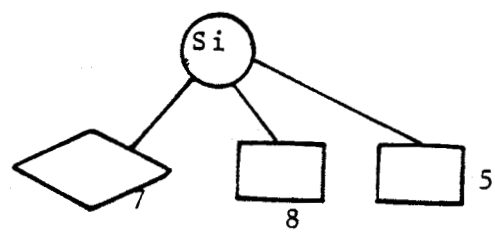
Nous supposerons maintenant, afin d'éviter une attente du convoi, qu'un train arrivant sur T_1 puisse choisir la voie restant libre si T_2 ou T_3 est occupé. De plus, si les deux tronçons sont disponibles le choix est imposé.

Le schéma de processus structure chargé de piloter le train B est défini par l'arbre programmatique étiqueté (A_2,Σ_2,l_2) se déduisant de l'arbre précédent (A_1,Σ_1,l_1) de la manière suivante :

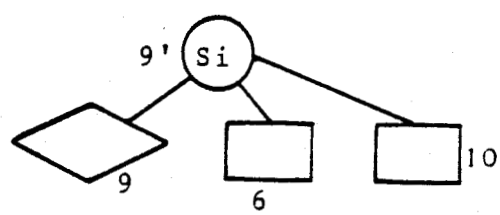
.. $\Sigma_2 =$ $_2$ $_2$ avec $_2 =$ $_1$ {e,f}

$_2 =$ $_1$ {r }

.. l'arbre programmatique A_2 se dérive de l'A.P. A_1 en substituant au noeud 5 la structure :



et au noeud 6 la structure :



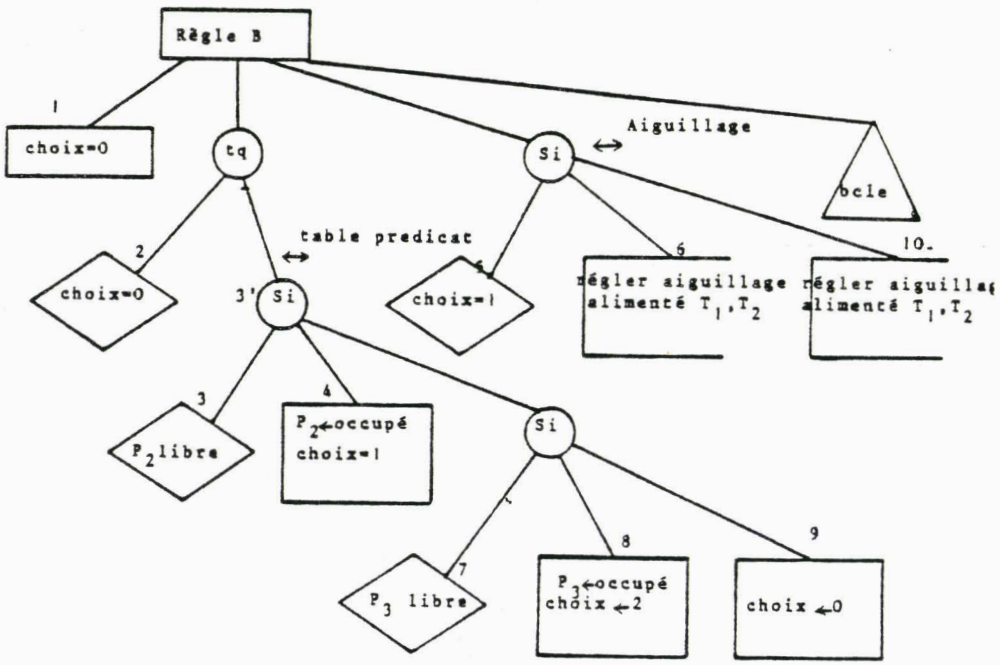
.. la fonction d'étiquetage l_2
la fonction d'étiquetage l_1 représente la restriction de l_2 aux noeuds 1,2,3,4,5,6, et par ailleurs :

$l_2(7) = r$, $l_2(8) = e$, $l_2(10) = f$

.. la liaison du schéma de processus structure A_2 est constituée par le partage de deux ressources T et A. Le partage de la ressource T se situe au noeud 3', et le partage de la ressource A au noeud 9'

Si le tronçon T_2 est choisi dans le cas où les deux tronçons T_2 et T_3 se trouvent simultanément libres, un interprétation possible du schéma de processus est la suivante :

$I\ (\ (A_2,\ \Sigma_2,\ 1_2)) =$

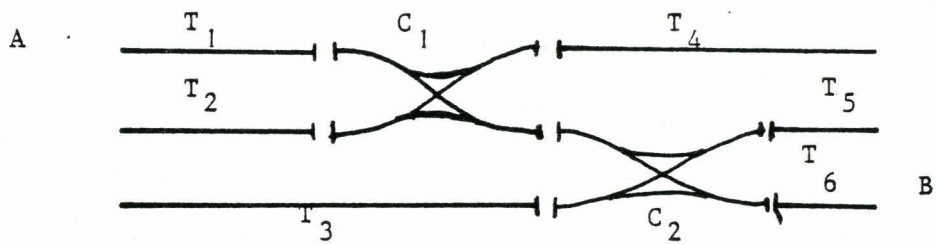


L'interprétation des noeuds 1 à 6 de l'arbre programmatique A_2 reste identique à la précédente et il nous a suffi de fournir la seule interprétation des nouveaux noeuds : 7,8,9 et 10. Cette facilité d'adaptation du modèle à de nouvelles consignes nous montre l'un des intérêts de la représentation des schémas de processus par les arbres programmatiques étiquetés.

333.2 Exemple simple de conflit (détection et résolution)

a - définition du problème

Considérons le circuit de la figure suivante, formé de deux croisements C_1 et C_2 et de six tronçons T_1, T_2, T_3 T_4, T_5 et T_6.



Le train A se trouve sur le tronçon T_1 et doit se rendre sur le tronçon T_5, le second train B se trouve sur le tronçon T_6 et doit se rendre

(1) dans le premier cas sur le tronçon T_3

(2) dans le second cas sur le tronçon T_2

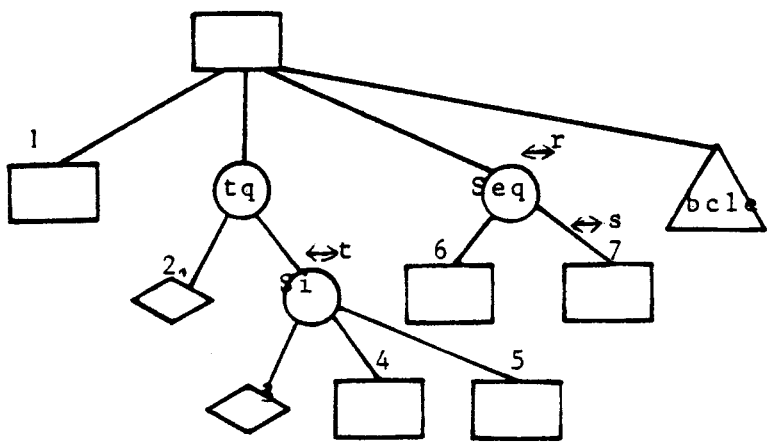
Remarque : pour ce circuit les ressources partagées sont au nombre de trois : le croisement C_1, le croisement C_2 et la table contenant l'état des différents tronçons (un tronçon étant soit occupé, soit libre).

b - étude du choix (1)

b.1 Schéma de processus associé au train A et interprétation

Le schéma de processus structuré $\mathcal{P}_3$ chargé de la surveillance du train A sera défini par l'arbre programmatique étiqueté (A_3, Σ_3, l_3), suivant :

.. A_3



.. $\Sigma_3 = \mathcal{A}_3 \cup \mathcal{P}_3$ avec $\mathcal{A}_3 = \{a,b,c,d,e\}$

$\mathcal{P}_3 = \{p,q\}$

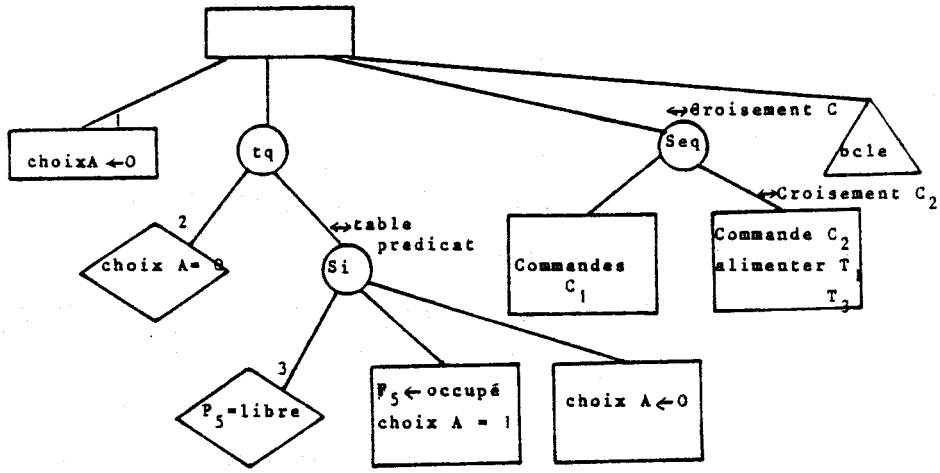
.. la fonction d'étiquetage l_3 est telle que :

$l_3(1) = a$, $l_3(2) = p$, $l_3(3) = q$, $l_3(4) = b$, $l_3(5) = c$, $l_3(6) = d$

$l_3(7) = e$

Une interprétation, I_3, possible pour ce schéma de processus structuré est la suivante :

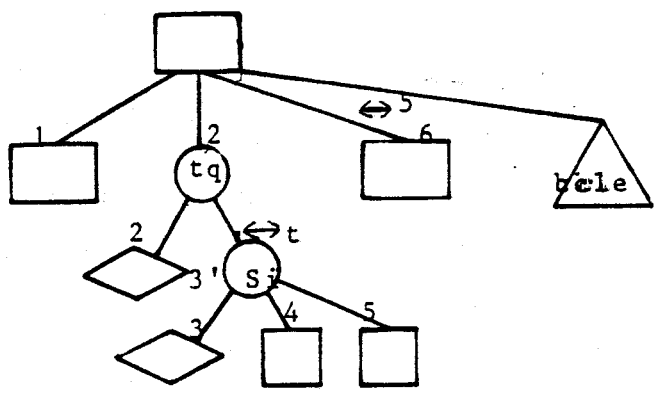
$I_3(\mathcal{P}_3)$:



b.2 Schéma de processus associé au train B et interprétation

Le schéma de processus structure $\mathcal{P}_4$, associé au train B sera défini par l'arbre programmatique étiqueté (A_4, Σ_4, l_4) :

.. A_4



.. $\Sigma_4 = \mathcal{a}_4 \cup \mathcal{P}_4$ avec $\mathcal{a}_4 = \{a', b', c', d'\}$
$\mathcal{P}_4 = \{p', q'\}$

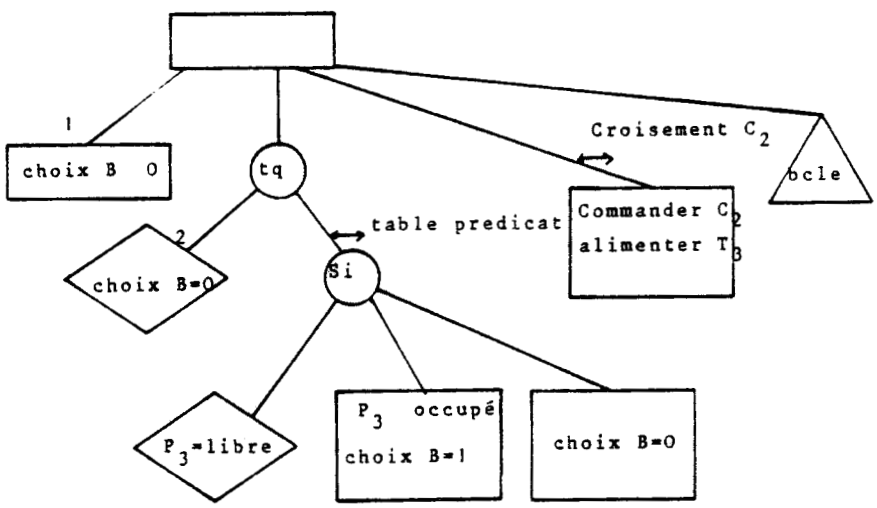
.. fonction d'étiquetage l_4

$l_4(1) = a'$, $l_4(2) = p'$, $l_4(3) = q'$, $l_4(4) = b'$, $l_4(5) = c'$
$l_4(6) = d'$.

.. les relations de ce schéma deprocessus $\mathcal{P}_4$ avec le schéma de processus $\mathcal{P}_3$ sont consitutées du partage des ressources s, t, aux niveaux des noeuds (6) et (3').

Une interprétation I_4 de ce schéma de processus structuré est la suivante :

I_4 ($\mathcal{P}_4$) :



L'étude des langages de ressources de ces deux schémas de processus nous montre l'absence de conflit entre $\mathcal{P}_3$ et $\mathcal{P}_4$.

$L_R(\mathcal{P}_3) = \{t, rs\}$

$L_R(\mathcal{P}_4) = \{t, s\}$

c - étude du choix (2)

Le schéma deprocessus du convoi A et son interprétation reste identique à la précédente.

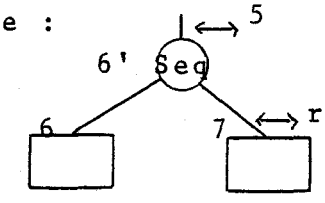
c.1 Schéma de processus associé au train B et interprétation

Le schéma de processus structuré chargé de piloter le train B est défini par l'arbre programmatique étiqueté (A_5, Σ_5, l_5) se déduisant del'arbre (A_4, Σ_4, l_4) de la manière suivante :

.. $\Sigma_5 = a_5 \cup \wp_5$ avec $a_5 = a_4 \cup \{e'\}$

$\wp_5 = \wp_4$

.. l'arbre programmatique A_5 se dérive de l'A.P. A_4 se substituant au noeud 6 la structure :

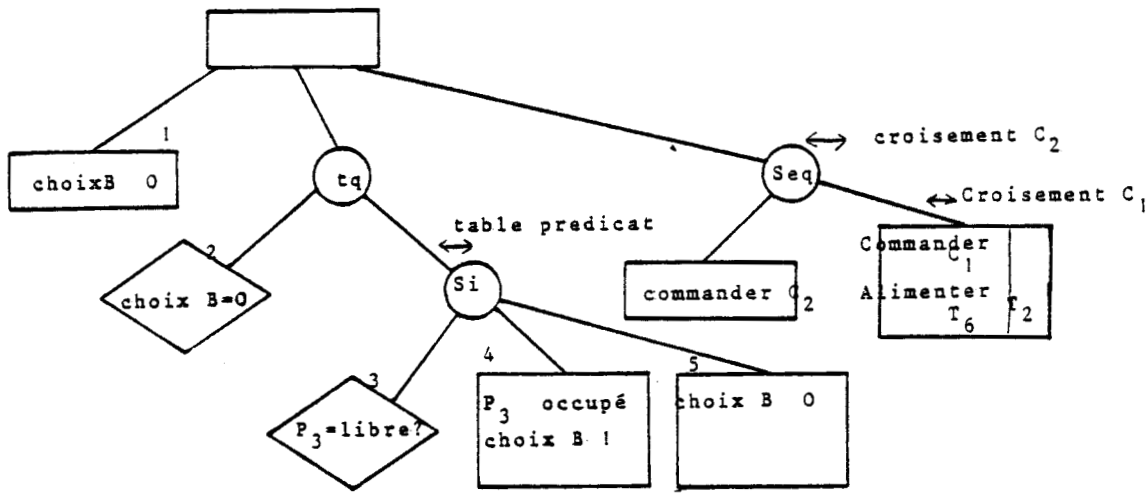


.. la fonction d'étiquetage l_5
la fonction d'étiquetage l_4 représente la restriction de l_5 aux noeuds 1 à 6 et par ailleurs $l_5(7) = e'$

.. la liaison du schéma de processus structure A_5 est constitué par trois relations de partage de ressources r,s,t. Le partage des ressources r,s,t s'effectue respectivement aux noeuds (7), (6'), (3').

Une interprétation I_5 de ce schéma de processus structure est la suivante :

$I_5(\mathcal{P}_5)$



or la liaison des deux schémas de processus entraîne le blocage des deux schémas de processus comme nous le montre l'examen deleur langage de ressources respectifs. Les langages de ressources $L_R(\mathcal{P}_3)$ et $L_R(\mathcal{P}_5)$ sont les suivants :

$$L_R(\mathcal{P}_3) = \{t, \underline{rs}\}$$

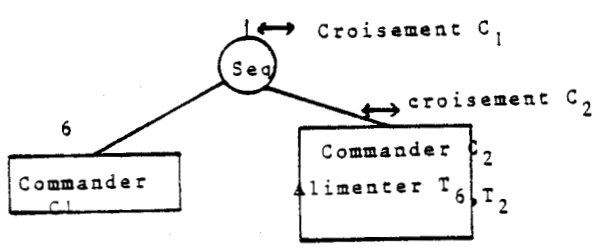
$$L_R(\mathcal{P}_5) = \{t, sr\}$$

Le lemme du met en évidence un état de blocage entre les deux schémas de processus.

Pour corriger ce blocage, il suffit de modifier la liaison du processus $\mathcal{P}_5$ et son interprétation.

.. les partages de ressources s et r sont maintenant respectivement associés aux noeuds (7) et (6,7). Le langage de ressource dans $L_R(\mathcal{P}_5)$ = {t,rs } et donc l'état de blocage disparaît.

.. L'interprétation I'_5 des noeuds (6) et (7) devient :



La détection aisée des états de blocages met en évidence sur cet exemple, l'intérêt complémentaire de la description d'un schéma de processus par un arbre programmatique étiqueté.

BIBLIOGRAPHIE DU CHAPITRE 3

(BEK 68) Bekey G.A. et Karplus W.J.
"Hybrid Computation"
Wiley Ind. Ed. 1968

(BOU) Boussin J.L.
"Synthèse et analyse des automatismes logiques"
Document Interne EDF

(COR 78) Corbeel D. et Gentina J.C.
"Analyse et synthèse des systèmes de commande des processus par réseaux de Petri"
Mini and Micro Computers and their application
Juin 1978 ZURICH (Suisse)

(DAC 76) Daclin E. et Blanchard M.
"Synthèse des systèmes logiques"
Edition Cepadues 1976

(DAL 79) Dalemagne D. Corbeel D. Gentina J.C.
"Digital control of process by microprocessors) Application to machine tool
Informatica 79 - Oct. 1979 - Ljvbljana (Yougoslavie).

(DAU 79) Dauphin G.
"Sur un logiciel de simulation hybride"
Mémoire de DEA - Juin 1979 - Lille

(DUM) Dumas J.M. et Prunet F.
"Un modèle d'analyse et de synthèse des systèmes de contrôle".
Extrait de AFC.1

(GIR 75) Girard P. et Naslin P.
"Construction des machines séquentielles industrielles"
Dunod 1975

(MAR 76) Marquette M. et Gentina J.C.
"Exemple d'une gestion par minicalculateur d'un ensemble de simulation hybride".
Mini and Microcomputers and their application
Juin 1976 - Zürich (Suisse)

(MAR 77) Marquette M.
"Sur un système de simulation hybride"
Thèse docteur Ingénieur - Juin 1977 - Lille

CONCLUSION GENERALE

Le formalisme du schéma de cablage constitue un outil bien adapté à la simulation de modèle mathématique sur une calculatrice de type II. Il a trois avantages essentiels :

(i) il décrit le modèle sans tenir compte de l'interprétation des opérateurs,

(ii) Il résoud le problème de la modélisation hybride grâce à l'introduction du magmoïde typé,

(iii) Il offre la possibilité d'étudier les transformations de cablage.

Le formalisme du schéma de contrôle permet de représenter et d'étudier les liaisons de divers processus industriels coopérants. Ce formalisme s'accompagne d'une méthodologie proche de la programmation structurée. Une telle démarche permet, à priori, de construire un réseau de Pétri vérifiant des conditions de bon fonctionnement.

Nos travaux futurs, dans ces domaines, se porteront dans quatre voies.

(i) Notre prochaine étude sur les schéma de cablage concerne le problème relatif au cablage automatique : l'utilisateur propose un modèle mathématique et le cablage est pris en charge par une calculatrice numérique. Cette étude n'est qu'amorcée mais l'algorithme de factorisation d'une expression formelle permettra la minimalisation du nombre d'opérateur nécessaire pour le cablage d'un modèle donné.

(ii) La détermination du caractère vivant d'un système de processus structurés où les liaisons seront quelconques et non d'un type donné.

(iii) La conception d'une nouvelle structure matérielle de type multiprocesseurs, qui devrait améliorer l'implémentation de la logique de commande.

(iv) L'étude de la sécurité dans la représentation d'un système de gestion de processus.