Un environnement et un langage graphique pour la spécification de processus parallèles communicants

Eqerem Arkaxhiu

THESE

présentée à

l' Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR DE 3ème CYCLE
"Informatique"

par

Eqerem ARKAXHIU

UN ENVIRONNEMENT ET UN LANGAGE GRAPHIQUE POUR LA SPECIFICATION DE PROCESSUS PARALLELES COMMUNICANTS

Thèse soutenue le 5 juillet 1984 devant la Commission d'Examen :

Monsieur	**Jacques MOSSIERE**	**: Président**
Messieurs	**Didier BERT**	**Examinateurs**
	Philippe JORRAND	
	Jean-Claude LATOMBE	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1982-1983

Président de l'Université : D. BLOCH

Vice-Président : René CARRE
Hervé CHERADAME
Marcel IVANES

PROFESSEURS DES UNIVERSITES :

ANCEAU François	E.N.S.I.M.A.G.
BARRAUD Alain	E.N.S.I.E.G.
BAUDELET Bernard	E.N.S.I.E.G.
BESSON Jean	E.N.S.E.E.G.
BLIMAN Samuel	E.N.S.E.R.G.
BLOCH Daniel	E.N.S.I.E.G.
BOIS Philippe	E.N.S.H.G.
BONNETAIN Lucien	E.N.S.E.E.G.
BONNIER Etienne	E.N.S.E.E.G.
BOUVARD Maurice	E.N.S.H.G.
BRISSONNEAU Pierre	E.N.S.I.E.G.
BUYLE BODIN Maurice	E.N.S.E.R.G.
CAVAIGNAC Jean-François	E.N.S.I.E.G.
CHARTIER Germain	E.N.S.I.E.G.
CHENEVIER Pierre	E.N.S.E.R.G.
CHERADAME Hervé	U.E.R.M.C.P.P.
CHERUY Arlette	E.N.S.I.E.G.
CHIAVERINA Jean	U.E.R.M.C.P.P.
COHEN Joseph	E.N.S.E.R.G.
COUMES André	E.N.S.E.R.G.
DURAND Francis	E.N.S.E.E.G.
DURAND Jean-Louis	E.N.S.I.E.G.
FELICI Noël	E.N.S.I.E.G.
FOULARD Claude	E.N.S.I.E.G.
GENTIL Pierre	E.N.S.E.R.G.
GUERIN Bernard	E.N.S.E.R.G.
GUYOT Pierre	E.N.S.E.E.G.
IVANES Marcel	E.N.S.I.E.G.
JAUSSAUD Pierre	E.N.S.I.E.G.
JOUBERT Jean-Claude	E.N.S.I.E.G.
JOURDAIN Geneviève	E.N.S.I.E.G.
LACOUME Jean-Louis	E.N.S.I.E.G.
LATOMBE Jean-Claude	E.N.S.I.M.A.G.

.../...

LESSIEUR Marcel	E.N.S.H.G.
LESPINARD Georges	E.N.S.H.G.
LONGEQUEUE Jean-Pierre	E.N.S.I.E.G.
MAZARE Guy	E.N.S.I.M.A.G.
MOREAU René	E.N.S.H.G.
MORET Roger	E.N.S.I.E.G.
MOSSIERE Jacques	E.N.S.I.M.A.G.
PARIAUD Jean-Charles	E.N.S.E.E.G.
PAUTHENET René	E.N.S.I.E.G.
PERRET René	E.N.S.I.E.G.
PERRET Robert	E.N.S.I.E.G.
PIAU Jean-Michel	E.N.S.H.G.
POLOUJADOFF Michel	E.N.S.I.E.G.
POUPOT Christian	E.N.S.E.R.G.
RAMEAU Jean-Jacques	E.N.S.E.E.G.
RENAUD Maurice	U.E.R.M.C.P.P.
ROBERT André	U.E.R.M.C.P.P.
ROBERT François	E.N.S.I.M.A.G.
SABONNADIERE Jean-Claude	E.N.S.I.E.G.
SAUCIER Gabrielle	E.N.S.I.M.A.G.
SCHLENKER Claire	E.N.S.I.E.G.
SCHLENKER Michel	E.N.S.I.E.G.
SERMET Pierre	E.N.S.E.R.G.
SILVY Jacques	U.E.R.M.C.P.P.
SOHM Jean-Claude	E.N.S.E.E.G.
SOUQUET Jean-Louis	E.N.S.E.E.G.
VEILLON Gérard	E.N.S.I.M.A.G.
ZADWORNY François	E.N.S.E.R.G.

PROFESSEURS ASSOCIES

BASTIN Georges	E.N.S.H.G.
BERRIL John	E.N.S.H.G.
CARREAU Pierre	E.N.S.H.G.
GANDINI Alessandro	U.E.R.M.C.P.P.
HAYASHI Hirashi	E.N.S.I.E.G.

PROFESSEURS UNIVERSITE DES SCIENCES SOCIALES (Grenoble II)

BOLLIET Louis
Chatelin Françoise

PROFESSEURS E.N.S. Mines de Saint-Etienne

RIEU Jean
SOUSTELLE Michel

CHERCHEURS DU C.N.R.S.

FRUCHART Robert	Directeur de Recherche
VACHAUD Georges	Directeur de Recherche

.../...

ALLIBERT Michel	Maître de Recherche
ANSARA Ibrahim	Maître de Recherche
ARMAND Michel	Maître de Recherche
BINDER Gilbert	
CARRE René	Maître de Recherche
DAVID René	Maître de Recherche
DEPORTES Jacques	
DRIOLE Jean	Maître de Recherche
GIGNOUX Damien	
GIVORD Dominique	
GUELIN Pierre	
HOPFINGER Emil	Maître de Recherche
JOUD Jean-Charles	Maître de Recherche
KAMARINOS Georges	Maître de Recherche
KLEITZ Michel	Maître de Recherche
LANDAU Ioan-Dore	Maître de Recherche
LASJAUNIAS J.C.	
MERMET Jean	Maître de Recherche
MUNIER Jacques	Maître de Recherche
PIAU Monique	
PORTESEIL Jean-Louis	
THOLENCE Jean-Louis	
VERDILLON André	

CHERCHEURS du MINISTERE de la RECHERCHE et de la TECHNOLOGIRE (Directeurs et Maîtres de Recherches, ENS Mines de St. Etienne)

LESBATS Pierre	Directeur de Recherche
BISCONDI Michel	Maître de Recherche
KOBYLANSKI André	Maître de Recherche
LE COZE Jean	Maître de Recherche
LALAUZE René	Maître de Recherche
LANCELOT Francis	Maître de Recherche
THEVENOT François	Maître de Recherche
TRAN MINH Canh	Maître de Recherche

PERSONNALITES HABILITEES à DIRIGER des TRAVAUX de RECHERCHE (Décision du Conseil Scientifique)

ALLIBERT Colette	E.N.S.E.E.G.
BERNARD Claude	E.N.S.E.E.G.
BONNET Rolland	E.N.S.E.E.G.
CAILLET Marcel	E.N.S.E.E.G.
CHATILLON Catherine	E.N.S.E.E.G.
CHATILLON Christian	E.N.S.E.E.G.
COULON Michel	E.N.S.E.E.G.
DIARD Jean-Paul	E.N.S.E.E.G.
EUSTAPOPOULOS Nicolas	E.N.S.E.E.G.
FOSTER Panayotis	E.N.S.E.E.G.

.../...

GALERIE Alain	E.N.S.E.E.G.
HAMMOU Abdelkader	E.N.S.E.E.G.
MALMEJAC Yves	E.N.S.E.E.G. (CENG)
MARTIN GARIN Régina	E.N.S.E.E.G.
NGUYEN TRUONG Bernadette	E.N.S.E.E.G.
RAVAINE Denis	E.N.S.E.E.G.
SAINFORT	E.N.S.E.E.G. (CENG)
SARRAZIN Pierre	E.N.S.E.E.G.
SIMON Jean-Paul	E.N.S.E.E.G.
TOUZAIN Philippe	E.N.S.E.E.G.
URBAIN Georges	E.N.S.E.E.G. (Laboratoire des ultra-réfractaires ODEILLON)
GUILHOT Bernard	E.N.S. Mines Saint Etienne
THOMAS Gérard	E.N.S. Mines Saint Etienne
DRIVER Julien	E.N.S. Mines Saint Etienne
BARIBAUD Michel	E.N.S.E.R.G.
BOREL Joseph	E.N.S.E.R.G.
CHOVET Alain	E.N.S.E.R.G.
CHEHIKIAN Alain	E.N.S.E.R.G.
DOLMAZON Jean-Marc	E.N.S.E.R.G.
HERAULT Jeanny	E.N.S.E.R.G.
MONLLOR Christian	E.N.S.E.R.G.
BORNARD Guy	E.N.S.I.E.G.
DESCHIZEAU Pierre	E.N.S.I.E.G.
GLANGEAUD François	E.N.S.I.E.G.
KOFMAN Walter	E.N.S.I.E.G.
LEJEUNE Gérard	E.N.S.I.E.G.
MAZUER Jean	E.N.S.I.E.G.
PERARD Jacques	E.N.S.I.E.G.
REINISCH Raymond	E.N.S.I.E.G.
ALEMANY Antoine	E.N.S.H.G.
BOIS Daniel	E.N.S.H.G.
DARVE Félix	E.N.S.H.G.
MICHEL Jean-Marie	E.N.S.H.G.
OBLED Charles	E.N.S.H.G.
ROWE Alain	E.N.S.H.G.
VAUCLIN Michel	E.N.S.H.G.
WACK Bernard	E.N.S.H.G.
BERT Didier	E.N.S.I.M.A.G.
CALMET Jacques	E.N.S.I.M.A.G.
COURTIN Jacques	E.N.S.I.M.A.G.
COURTOIS Bernard	E.N.S.I.M.A.G.
DELLA DORA Jean	E.N.S.I.M.A.G.
FONLUPT Jean	E.N.S.I.M.A.G.
SIFAKIS Joseph	E.N.S.I.M.A.G.
CHARUEL Robert	U.E.R.M.C.P.P.
CADET Jean	C.E.N.G.
COEURE Philippe	C.E.N.G. (LETI)

.../...

DELHAYE Jean-Marc	C.E.N.G. (STT)
DUPUY Michel	C.E.N.G. (LETI)
JOUVE Hubert	C.E.N.G. (LETI)
NICOLAU Yvan	C.E.N.G. (LETI)
NIFENECKER Hervé	C.E.N.G.
PERROUD Paul	C.E.N.G.
PEUZIN Jean-Claude	C.E.N.G. (LETI)
TAIEB Maurice	C.E.N.G.
VINCENDON Marc	C.E.N.G.

LABORATOIRES EXTERIEURS

DEMOULIN Eric	C.N.E.T.
DEVINE	C.N.E.T. (R.A.B.)
GERBER Roland	C.N.E.T.
MERCKEL Gérard	C.N.E.T.
PAULEAU Yves	C.N.E.T.
GAUBERT C.	I.N.S.A. Lyon

Remerciements

je tiens à remercier :

Jacques Mossiere, Professeur á l'ENSIMAG, qui m'a fait l'honneur de présider le jury de ma thèse;

Philippe Jorrand, Directeur de Recherche au CNRS, qui m'a reçu dans son équipe et pour le soutien qu'il m'a prodigué tout au long de ce travail;

je voudrais exprimer ma reconnaissance à Messieurs Didier Bert, Chargé de Recherche au CNRS, et Jean Claude Latombe, Professeur á l'ENSIMAG, d'avoir accepté de participer à ce jury.

Cette thèse a profité de l'experience et des observations pertinentes de mes collegues d'équipe J. M. Perreira, P. Lagnier, S. Rogé, M. Cisneros.

Je ne voudrais pas dissocier de ces remerciements H. Saya pour son aide sympathique.

Je remercie enfin le personnel du Service de Reprographie de l'IMAG pour le soin apporté á l'impression de cette thèse

Egerem Artaxiu.

TABLE DES MATIERES

IV - EVALUATION DE L'EXPRESSION

I - INTRODUCTION

L'objectif à long terme du projet SPARC (systèmes parallèles communicants) est d'intégrer dans un cadre cohérent les spécifications de fonctions, de types et de processus communicants et de fournir au concepteur de systèmes les outils correspondants d'analyse et de validation. Il s'agit avant tout d'un environnement pour l'aide à la conception des systèmes, qui peuvent être mis sous la forme de réseaux de processus interconnectés, qui communiquent par envoi et réception de messages à travers les portes.

Le langage FP2 (Functional Parallel Programming) est conçu pour répondre à ces besoins. La caractéristique fondamentale du FP2 est l'utilisation systématique des termes pour représenter les notions de base du langage. FP2 est un langage grâce auquel les systèmes peuvent être décrits et programmés en termes de réseaux de processus communicants. Les messages sont des valeurs, au sens habituel de la programmation et ont des types. Des paires choisies de portes peuvent être reliées par connecteurs orientés, permettant ainsi aux messages de circuler d'une porte à une autre. La description de chaque processus englobe à la fois les fonctions appliquées par le processus aux messages reçus afin de calculer les messages qu'il émet et l'ordonnancement des communications le long des connecteurs attachés à ce processus, avec séquentialité , non déterminisme et parallélisme.

Dans FP2 les valeurs ont leurs types et les types sont des algèbres de termes. Les valeurs sont représentées par des termes dans ces algèbres et les fonctions sont définies par les équations. Les processus sont des systèmes de transition non déterministes définis par des règles, qui décrivent sous quelles conditions un ensemble donné de communications peut avoir lieu. Lorsqu'une règle est appliquée, son ensemble de communications se déroule, reflétant ainsi le parallélisme et des conditions nouvelles sont établies. Les conditions sont représentées par des termes et l'applicabilité des règles est représentée par le "pattern matching".

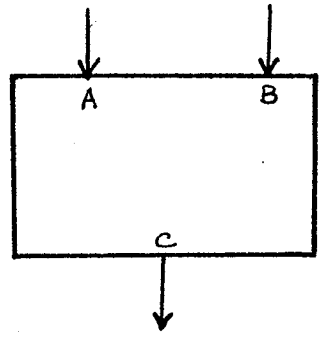
FP2 est ainsi un langage où la programmation fonctionnelle, le type algébrique , les processus communicants, le parallélisme et le non déterminisme cohabitent harmonieusement. On trouvera une description formelle de FP2 dans [JOR 84] [PER 84] [CIS 84].

Le composant élémentaires pour organiser le calcul parallèle en FP2 est le processus. Le processus a ses portes à travers lesquelles les messages peuvent circuler de l'environnement du processus vers le processus ou du processus vers son environnement. Les messages arrivent à la porte le long des connecteurs orientés. Le transport d'un message le long de connecteur est une communication.

On cherche donc d'abord à décrire un processus isolé en exprimant la façon dont il est perçu par son environnement potentiel, c'est-à-dire par tout réseau de processus auquel il sera éventuellement connecté pour échanger avec lui des messages. Décrire un processus, c'est décrire l'activité de communication le long des connecteurs de ses portes. Ensuite, c'est précisément la construction de réseaux de processus qu'il faut décrire avec en particulier, les opérations qui consistent à établir une connexion entre deux processus ou à rendre invisibles certains connecteurs.

Le comportement du processus englobe deux phénomènes :
(1) les séquences dans lesquelles les communications peuvent avoir lieu le long des différents connecteurs qui appartiennent à l'ensemble des séquences possibles pour le processus. De plus certaines communications étant en général simultanément possibles, les séquences intéressantes sont des séquences d'évènement où un évènement est un ensemble de communications.

(2) chaque message, qui est émis par un processus est le résultat du calcul effectué par le processus. Le résultat est en général une fonction des messages reçus en avant.
Par exemple, soit le processus qui admet deux entiers en entrée sur les portes A et B et rend en sortie le maximum des deux entiers sur la porte C.



Si la communication le long du connecteur ?.A est notée ?.A(m), la séquence des évènements ?.A, ?.B, C.? peut être :

?.A(3) ?.B(6) C.?(6)

Si σ dénote l'évènement initial, cette séquence sera représentée ainsi :

(C.?(σ(?.B(?.A(3,σ)))))

Cette construction provient de [JOR 83] , et l'idée générale est de pouvoir exprimer le processus de manière à ce qu'un fonctionnement légal de celui-ci soit un terme de la forme ci-dessus.

L'originalité de ce projet réside dans le fait que la spécification des dits réseaux se rapproche de la visualisation que peut s'en faire un utilisateur potentiel.

Pour manipuler ces objets, il est nécessaire de conserver dans une base de données les réseaux ainsi constitués.
SPARC propose un outil de comparaison entre deux processus permettant de dire s'il y a équivalence des deux comportements.

Enfin, le poste de travail ne peut être conçu sans la possibilité d'une exécution du processus, d'où l'implantation d'un interpréteur permettant de gérer, entre autres, le non déterminisme de fonctionnement.

Dans cette thèse nous présentons les résultats obtenus lors de la réalisation de quelques aspects du poste de travail. La thèse comporte 4 chapitres , le contenu de chaque chapitre est organisé comme suit :

Le chapitre 1 est un résumé des connaissances de définitions formelle de FP2, nécessaires à une bonne compréhension de la suite.

Le chapitre 2 est consacré à l'architecture de SPARC et à la syntaxe de description des processus.

Dans le chapitre 3 nous parlons du calcul de l'expression. Nous y présentons le problème de renommage des portes. L'optimisation du calcul et les

algorithmes correspondants sont présentés en détail.

Dans le chapitre 4, les idées de description graphique et de construction du réseau sont données. Quelques détails d'implantation sont aussi mentionnés.

La conception de cette partie du système est faite indépendamment des autres parties du projet SPARC, comme la recherche de méthodes d'analyse, de validation, de synthèse et de simulation du comportement.

La construction de ces parties et leur intégration, constituent donc le principal problème.

En conclusion, on peut dire que tout ce qu'on a présenté ne constitue qu'une initiation à la réalisation du poste de travail, qui ne pourra être complet, qu'après un travail commun théorique et appliqué.

II- DEFINITIONS

2 - DEFINITIONS

Nous présentons dans cette partie les notions que nous considérons comme indispensables pour comprendre la suite, ainsi que la terminologie spécifique et les principes sur lesquels sont basées les méthodes présentées dans les chapitres suivants. Toutes les définitions proviennent de [JOR 83] .

2.1. Présentation algébrique de processus

2.1.1. Définitions préliminaires

Soient A un ensemble de sortes
P un ensemble de noms de portes
X un ensemble de variables
F un ensemble de noms de fonctions

F et X sont des ensembles disjoints.

Soient les deux applications suivantes : arité notée α et sorte notée β définies sur F et X

$\alpha : F \rightarrow A^*$ $\quad$ $\beta : F \rightarrow A$

$\alpha : X \rightarrow \{\varepsilon\}$ $\quad$ $\beta : X \rightarrow A$

Les deux sous-ensembles suivants sont définis :

$F[s,a] = \{f \in F \mid \alpha(f) = 1, \ \beta(f) = a\}$ où $s = a_1 a_2 \ldots a_n$ et

$X[a] = \{x \in X \mid \beta(x) = a\}$

2.1.2. Termes fonctionnels

U [F,X,a] est l'ensemble des termes fonctionnels sur les noms de fonctions F, les noms de variables X et de sorte a et est défini inductivement par :

(1) $X[a] \subseteq U[F,X,a]$

(2) $F[\varepsilon,a] \subseteq U[F,X,a]$

(3) $\forall f \in F[a_1,a_2,\ldots,a_n,a]$, $\forall Ui \quad U[F,X,a_i]$, $i = 1,2,\ldots,n$
$f(U_1,U_2,U,\ldots,U_n) \in U[F,X,a]$

2.1.3. Termes d'état

Soit G l'ensemble des noms d'états (G,X,F étant disjoints) avec l'arité $\alpha : G \to A^*$. Les sous-ensembles suivants sont définis :

$G[s] = \{g \in G \mid \alpha(g) = s\}$ avec $s = a_1, a_2,\ldots,a_n$

$V[F,X,G]$ est l'ensemble des termes d'état sur les noms de fonctions F, les noms des variables X et les noms d'états G, et est défini inductivement par :

(1) $G[\varepsilon] \subseteq V[F,X,G]$

(2) $\forall g \in G[a_1,\ldots,a_n]$, $\forall ui \in U[F,X,a_i]$
alors $i = 1,\ldots,n$, $g(u_1,\ldots,u_n) \in V[F,X,G]$

2.1.4. Termes d'évènement

Soit $H[K]$ l'ensemble de noms d'évènements défini par $H[K] = 2^K$ où K est l'ensemble des connecteurs défini par :

$K = K_{int} \cup K_{ext}$

$K_{int} = P \times P$ (ensemble de connecteurs internes)

$K_{ext} = K_{in} \cup K_{out}$ (ensemble de connecteurs externes)

$K_{in} = \{?\} \times P$ (ensemble de connecteurs d'entrée)

$k_{out} = P \times \{?\}$ (ensemble de connecteurs de sortie)

Une application η sorte de message est définie sur K par

$\eta : K \to A$

$\forall k \in K$, $\eta(k)$ est la sorte des messages qui peuvent être communiqués le long de k.
Si l'application π est définie et représente une permutation arbitraire de $H[K] \rightarrow K^*$ et il est alors possible d'étendre la définition de η à $H[K]$ par :

$$\forall h \in H[K], \pi[K](h) = k_1, k_2, \ldots, k_n \Longrightarrow \eta(h) = \eta(k_1) \ldots \eta(k_n)$$

Les deux sous-ensembles de H[K] sont définis :

$$H[K,s] = \{h \in H[K] \mid \eta(h) = s\} \quad \text{si } s = a_1, \ldots, a_n$$

Soit enfin σ, le nom d'évènement initial :

$W[F,X,G,K]$ est l'ensemble des termes dévènement sur les noms de fonctions F, les noms de variables X, les noms d'états G, et les noms de connecteurs K, et est défini inductivement par :

(1) $\sigma \in W[F,X,G,K]$

(2) $\forall h \in H[K,a_1 \ldots a_n]$, $\forall u_i \in U[F,X,a_i]$ $i = 1,\ldots,n$
$\forall v \in V[F,X,G]$, $h(u_1,\ldots,u_n,v) \in W[F,X,G,K]$

(3) $\forall h \in H[K,a_1,\ldots,a_n]$, $\forall u_i \in U[F,X,a_i]$ $i = 1,\ldots,n$
$\forall w \in W[F,X,G,K]$, $h(u_1,\ldots,u_n,w) \in W[F,X,G,K]$

Un terme d'évènement construit par la règle (2) est appelé terme d'évènement simple.
Si u est un terme fonctionnel, terme d'état ou terme d'évènement var(u) désigne l'ensemble des variables apparaissant dans le terme u.

2.1.5. Termes de transition

Si $w = h(u_1,\ldots,u_n,v)$ est un terme d'évènement simple, soient les trois ensembles suivants :

$$X = var(v)$$
$$X_{in} = \bigcup var(u_i),\ k_i \in K_{in}$$
$$X_{out} = \bigcup var(u_I),\ k_i \in K_{out}$$

w est appelé terme de transition s'il possède la propriété

$$X_{out} \subseteq X \cup X_{in}$$

c'est-à-dire si la valeur des variables utilisées en sortie n'est fonction que des valeurs "connues" en entrée ou dans l'état précédent.

2.1.6. Présentation de processus

La donnée d'un ensemble fini de connecteurs K et d'un ensemble fini et non vide de noms d'états G permet la définition de l'ensemble

$\Sigma = K \cup G$ appelé signature de processus.

L'ensemble des termes syntaxiquement corrects, pour la signature Σ est donné par :

$$T[\Sigma] = V[F,\emptyset,G] \cup W[F,\emptyset,G,K]$$

Etant donné une signature Σ, une équation de processus est la donnée d'un couple $\langle w,v \rangle$, où $w \in W[F,X,G,K]$ et $v \in V[F,X,G]$. Une équation de processus est dite initiale, si $v \in V[F,\emptyset,G]$ et si $W = \sigma$, sinon elle est dite non initiale.

Enfin, une présentation de processus est la donnée d'une signature Σ, d'un ensemble non vide d'équations initiales sur Σ appelé I, et d'un ensemble d'équations non initiales sur Σ appelé E.

$$P = (\Sigma, I, E).$$

Exemple

Le processus MAX (1) reçoit un entier par A
(2) reçoit un entier par B
(3) rend le maximum par C

sera présenté par :

```
MAX = processus
   K = {?.A,?.B,C.?}
        η(?.A) = NAT
        η(?.B) = NAT
        η(C.?) = NAT
   G = {X,Y,Z}
       α(X) = ε
       α(Y) = NAT
       α(Z) = NAT
   I = {<σ,X >}
   E = {<{?.A}(m,X),Y(m)> ,
        <{?.B  (n,Y(m)),Z(MAX(m,n))> ,
        <{C.?} (m,Z(m)),X >}
fin.
```

Soient v_o et v_1 les termes d'état et $e = k_1(u_1),\ldots,k_n(u_n)$ un évènement sur $h = \{k_1,k_2,\ldots,k_n\}$. Soient X_0,X_1,X_{in},X_{out} les ensembles de variables définis par :

$$X_0 = var(v_0)$$
$$X_1 = var(v_1)$$
$$X_{in} = \bigcup var(U_i),\ k_i \in K_{in}$$
$$X_{out} = \bigcup var(U_i),\ k_i \in K_{out}$$

Si v_o, v_1 et e vérifient les conditions suivantes

$$X_1 \subseteq X_0 \cup X_{in}$$
$$X_{out} \subseteq X_0 \cup X_{in}$$

alors la notation

$$v_o \Longrightarrow e\ v_1$$

est appelée règle et dénote l'équation non initiale $\langle h(l,v_0),v_1\rangle$ où la liste l est une permutation de u_i.

Soit $K = [k_1,k_2,\ldots,k_m]$ l'ensemble des noms de connecteurs et $a_i \in A^*$ $i = 1,\ldots,n$.
Soient v_i $i = 1,\ldots,p$ les termes d'état en V [F,∅,G] . Soient v_{o_i} et v_{1_i} les termes d'état en V [F,G] et e_i un évènement en h_i $i = 1,\ldots,q$, pour $h_i \in H$ [K] avec l'ensemble de noms de variables en v_{o_i}, v_{1_i} et e_i telles que "$v_{o_i} \Longrightarrow e_i\ v_{1_i}$" peut être une règle.

La notation

processus
  Connecteurs : $k_1 : a_1$
  ⋮
  $k_m : a_m$
  Etats : $g_1 : s_1$
  ⋮
  $g_n : s_n$
  Init : v_1
  ⋮
  v_p
  Règles : $v_{0_1} \Longrightarrow e_1\ v_{1_1}$
  ⋮
  $v_{0_q} \Longrightarrow e_q\ v_{1_q}$

fin.

dénote la présentation de processus (Σ,I,E) telle que :

$$\Sigma = K \cup G \quad \text{avec } \forall\, k_i \in K,\ \eta(k_i) = a_i$$
$$\text{et } \forall\, g_i \in G\ ,\ \alpha(g_i) = s_i$$

$$I = \{<\sigma, v_i> \mid v = 1,\ldots,p\}$$
$$E = \{<h_i(l_i, v_{0_i}), v_{1_i}> \mid i = 1,\ldots,q\}$$

où l_i, $i = 1,\ldots,q$ est la permutation des termes en e_i de sorte $\eta(h_i)$
C'est une manière plus claire de représentation des processus.

MAX = Processus

Connecteurs : ?.A : NAT
?.B : NAT
C.? : NAT

Etats : X : ()
Y : NAT
Z : NAT

Init : X

Règles : $X \Rightarrow$?.A(m) Y(m)
$Y(m) \Rightarrow$?.B(n) Z(MAX(m,n))
$Z(m) \Rightarrow$ C.?(m) X

Fin.

Le processus R E G :

(1) initialement le registre contient 0.
(2) son contenu peut être lu par la porte M.
(3) son contenu peut être invalide, en envoyant un signal par la porte T
(4) un nouveau contenu peut être écrit à travers L

R E G : Processus

Connecteurs : ?.L : NAT
M.? : NAT
?.T : SIG

Etats : F : NAT
G : ()

Init : F(0)

Règles : $F(p) \Rightarrow M.?(p)\ F(p)$
$F(p) \Rightarrow ?.T(zz)\ G$
$G \Rightarrow L.?(p)\ F(p)$

fin.

2.2. Construction de réseau de processus

Cette partie donne la définition des trois opérations que l'on peut effectuer sur les processus et les connecteurs et qui sont :

l'union : notée "//"
la connexion : notée "+"
l'abstraction : notée "-"

2.2.1. Union

2.2.1.1. Produit de termes d'états :

Soient G1 et G2 deux ensembles de noms d'états avec leurs fonctions d'arité $\alpha 1$ et $\alpha 2$. Considérons l'ensemble des noms d'états G et sa fonction d'arité α définie :

$$\forall\, g_1 \in G_1,\ \forall\, g_2 \in G_2 \quad g = \psi(g_1, g_2) \in G \quad \text{et} \quad \alpha(g) = \psi\,(\alpha_1(g_1),\ \alpha_2(g_2))$$

où $\psi(s_1, s_2)$ est une fusion arbitraire de s_1 et s_2.

G est appelé produit de termes d'états et est noté G = G1.G2

Ex. Pour les processus MAX et REG

G1 = {X,Y,Z} et G2 = {F,G}

G = G1.G2 = {X-F, X-G, Y-F, Y-G, Z-F, Z-G}

$\alpha_1(X) = ()$ $\alpha_2(F) = NAT$ $\alpha(X-F) = NAT$

$\alpha_1(Y) = NAT$ $\alpha_2(F) = NAT$ $\alpha(Y-F) = NAT\ NAT$

Soient V1 = V[F,X,G1] et V2 = V [F,X,G2] deux ensembles de termes d'états, l'ensemble de termes d'états V est le produit de V1 et V2 noté V1.V2, ssi V = V [F,X,G1.G2] .

Ainsi, soit $v_1 = g1(m1) \in V1$ et $v_2 = g2(m2) \in V2$, alors il existe un élément $v \in V$ tel que v = (g1.g2)(ψ(m1,m2)) où v = v1.v2

Ex. v1 = Y(m) v2 = F(p) v = Y-F(m,p)

v1 = Z(MAX(m,n)) v2 = G v = Z-G(MAX(m,n))

2.2.1.2. Produit de termes d'évènement

Etant donnés deux ensembles de termes d'évènements W1 = W [F,X,G1,K1] et W2 = [F,X,G2,K2] , l'ensemble des termes d'évènements W est le produit de W1 et W2, noté W1.W2, ssi W = [F,X,G1.G2,K] où $K = K1 \cup K2$.

Ainsi, soient deux termes d'évènements simples $w1 = h1(m1,v1) \in W1$ et $w2 = h2(m2,v2) \in W2$ avec h1 et h2 disjoints, alors il existe un seul élément $w \in W$ tel que

$$w = (h1 \cup h2)(\psi[h1,h2,K]\ (m1,m2),v1.v2)$$

où $\psi[h1,h2,K\ ]\ (m1,m2)$ est une permutation de m1 et m2 compatible avec l'arité de message de $h1 \cup h2$. Cet élément est noté w1.w2

Ex. $w1 = Y(m) \Rightarrow ?.B(n)$ $\quad w2 = F(p) \Rightarrow M.?(p)$

$w = Y\text{-}F(m,p) \Rightarrow ?.B(n)\ M.?(p)$

2.2.1.3. Produit des équations

Soient $\Sigma 1 = K1 \cup G1$ et $\Sigma 2 = K2 \cup G2$ deux signatures de processus telles que K1 et K2 sont des ensembles disjoints de noms de connecteurs. La signature Σ est le produit de $\Sigma 1$ et $\Sigma 2$ noté $\Sigma 1.\Sigma 2$, ssi $\Sigma = (K1 \cup K2) \cup (G1.G2)$

Soit I1 l'ensemble des équations initiales en $\Sigma 1$ et I2 l'ensemble des équations initiales en $\Sigma 2$. L'ensemble I est le produit de I1 et I2, noté I1.I2, ssi I est l'ensemble des équations initiales en $\Sigma 1.\Sigma 2$ tel que

$$I = \{<\sigma, v1.V2> \mid <\sigma,v1> \in I1\ ,\ <\sigma,v2> \in I2\ \}$$

Etant donnés X1 et X2 deux ensembles disjoints de noms de variables, E1 l'ensemble des équations non initiales en $\Sigma 1$ et X1, E2 l'ensemble des équations non initiales en $\Sigma 2$ et X2, alors l'ensemble E est le produit de E1 et E2, noté E1.E2, ssi E est l'ensemble des équations non initiales en $\Sigma 1.\Sigma 2$ et $X1 \cup X2$ tel que

$$E = \{<w1.w2, v1.v2> \mid <w1,v1> \in E1\ ,\ <w2,v2> \in E2\}$$

Ex. e1 : $Y(m) \Rightarrow ?.B(n)\ \ Z(MAX(m,n))$

e2 : $F(p) \Rightarrow M.?(p)\ \ F(p)$

e = e1.e2 : $Y\text{-}F(m,p) \Rightarrow ?.B(n)\ M.?(p)\ \ Z\text{-}F(MAX(m,n),p)$

2.2.1.4. Union

Soient P1 = ($\Sigma 1$,I1,E1) et P2 = ($\Sigma 2$,I2,E2) deux processus avec $\Sigma 1 = K1 \cup G1$ et $\Sigma 2 = k2 \cup G2$. K1 et K2 sont disjoints. L'union de P1 et P2 noté P1 // P2 "met ensemble" deux processus ou réseaux de processus en ayant comme résultat un processus, qui décrit à la fois les comportements individuels et simultanés de ses opérandes. Ce nouveau processus s'écrit sous la forme :

$$P = P1 \;//\; P2 = (\Sigma, I, E)$$

où $\Sigma = \Sigma 1.\Sigma 2$

$I = I1.I2$

$E = E1.Z[G2] \cup E2.Z[G1] \cup E1.E2$

En effet, la construction des règles se décompose ainsi :

a : E1.Z[G2]

Pour toute règle $G \Rightarrow$ evt D de P

Pour tout terme d'état W' de P'

La règle $G\text{-}W' \Rightarrow$ evt D.W' est créée

b : E2.Z[G1]

Pour toute règle $G' \Rightarrow$ evt D' de P'

Pour tout terme d'état W de P

La règle $G'\_W \Rightarrow$ evt' D'_W est créée

c : E1.E2

Représente le produit des équations de P1 et P2 selon le modèle donné en 1.2.2.1.3.

Ex. D'après le schéma ci-dessus P1 = MAX P2 = REG

a : $X\text{-}F(p) \Rightarrow ?.A(m)$ $Y\text{-}F(m,p)$

$X\text{-}G \Rightarrow ?.A(m)$ $Y\text{-}G(m)$ pour la première règle de MAX

b : $X\text{-}F(p) \Rightarrow M.?(p)$ $X\text{-}F(p)$

$Y\text{-}F(m,p) \Rightarrow M.?(p)$ $Y\text{-}F(m,p)$

$Z\text{-}F(m,p) \Rightarrow M.?(p)$ $Z\text{-}F(m,p)$ pour la première règle de REG

c : $Y\text{-}F(m,p) \Rightarrow ?.B(n)$ $M.?(p)$ $Z\text{-}F(MAX(m,n),p)$ pour la deuxième règle de MAX et la première règle de REG.

2.2.2. Connexion

2.2.2.1. Substitution

Soit F l'ensemble des noms de fonctions et X l'ensemble des noms de variables avec l'arité α et de sorte β . L'application β est étendue en U[F,X]

$$\forall\ u = f(u_1,\ldots,u_n) \in U[F,X]\ ,\quad \beta(u) = \beta(f)$$

Une substitution est une application : $\rho : X \longrightarrow U[F,X]$ telle que $\beta(\rho(x)) = \beta(x)$

L'application d'une substitution ρ sur un terme t, donne un nouveau terme t' où toute occurrence de x dans t dans le domaine de ρ est remplacée par $\rho(t)$

Etant données deux substitutions ρ et ρ' dans le même domaine, ρ est plus générale que ρ , s'il existe une substitution ρ'' telle que $\rho = \rho'' \circ \rho$ où "$\circ$" dénote la composition des substitutions.

2.2.2.2. Unification

On dit que t1 et t2 sont unifiables s'il existe une substitution ρ telle que $\rho(t1) = \rho(t2)$, ρ est appelé l'unificateur de t1 et t2.

Parmi toutes les substitutions possibles, il en existe une qui est plus générale que les autres, on l'a nomme le plus général unificateur de t1 et t2 soit en abrégé p g u (t1,t2).

Ex.

Soient $t1 = f(g(y,h(x))$

et $t2 = f(g(h(u),z))$

où x,y,z,v sont des variables

f,g,h sont des fonctions

t1 et t2 sont unifiables par graphe G tel que :

Graphe $(\rho) = \{ \langle y,h(u)\rangle , \langle z,h(x)\rangle\}$

et $\rho(t1) = \rho(t2) = f(g(h(u),h(x)))$

L'application d'une substitution ρ sur un terme t1 donne un nouveau terme t'1 , tel que pour toute variable x de t1, si $\langle x,y\rangle \in$ Graphe (ρ) alors toutes les occurrences de x dans t1 sont remplacées par y dans t'1.

2.2.2.3. Connexion

Les échanges entre processus composants d'un réseau sont effectués par rendez-vous, c'est-à-dire que tous les processus impliqués dans l'échange doivent être dans un état où l'échange est possible pour que celui-ci puisse être effectué. Une communication entre M et N est possible uniquement quand les communications à travers M.? et ?.N sont simultanément possibles.

Soit P1 = (Σ1, I1, E1) une présentation de processus avec Σ1 = K1 ⊎ G1. Soient M et N deux noms de portes telles que $M.? \in K1$, $?.N \in K1$ et $\eta(M.?) = \eta(?,N)$. La connexion de M à N en P1, notée P1 + N.N est un processus P, qui peut se comporter comme le processus représenté par P1 et qui permet la communication le long M.N. de résultat de cette opération est le suivant :

$$P = P1 + M.N = (\Sigma, I, E)$$

où $\Sigma = K \cup G$, avec $K = K1 \cup \{M.N\}$ et $G = G1$

$I = I1$

$E = E \cup E'$

avec $E' = \{\langle \rho(w\text{-}M.?(t) - ?.N(u) + M.N(t)),\ \rho(v)\rangle \mid \langle u,v\rangle \in E1$ et $M.?(t) \in W$ et $?.N(u) \in W$ et $\rho = pgu(t,u)\}$.

Ex. Considérons la règle (c) de l'exemple en 1.2.2.1.4.

$$Y\text{-}F(m,p) \Longrightarrow\ ?.B(n)\ M.?(p)\ \ Z\text{-}F(MAX(m,n),p)$$

Cette règle appartient à l'ensemble des règles de processus P1 = MAX // REG.
Si on veut construire le processus P = P1 + B.M, cette règle sera représentée :

$$Y\text{-}F(m,n) \Longrightarrow B.M(n)\ \ Z\text{-}F(MAX(m,n),n).$$

Dans ce cas ρ = pgu(n,p) signifie qu'il faut remplacer toute occurrence de p par n.

Le fait de connecter M et N n'empêche nullement de les réutiliser plus tard pour d'autres connexions puisque l'on rajoute seulement dans les règles de fonctionnement, le cas de la communication directe entre ces deux portes.

2.2.3. Abstraction

Soit P1 = $(\Sigma 1, I1, E1)$ la description d'un processus avec $\Sigma 1 = K1 \cup G1$.
Soit k le nom d'un connecteur. Cacher k dans P1, noté par P1-k, c'est présenter un nouveau processus, qui peut se comporter comme le processus représenté par P1, $k \in K_{EXT}$, alors les communications le long de k sont maintenant invisibles. Le résultat de cet opérateur est :

$$P = P1-k = (\Sigma, I, E)$$

où $\Sigma = K \cup G$, avec $k = k1-\{k\}$ et $G = G1$

$I = I1$

$E = (E1-E') \cup E''$

avec $E' = \{\langle w,v\rangle \mid \langle w,v\rangle \in E1 \quad \text{et} \quad \exists u,\ k(u) \in W$

$E'' = \{\langle w-k(u),v\rangle \mid \langle w,v\rangle \in E1 \quad \text{et } k \in K_{INT} \text{ et } k(u) \in W\}$

Remarque

Tout au long de cette thèse nous utiliserons une autre notation qui correspond à celle de [JOR 81] , parce que c'est une notation plus claire et plus compacte. L'opérateur d'abstraction sera noté "[]" et entre les crochets on comprend un sous-ensemble des connecteurs. Dans ce cas l'opérateur d'abstraction préserve l'accès au sous-ensemble sélectionné {k1,.....,kp} .
On peut le commenter ainsi : supprimer les règles , qui se réfèrent aux connecteurs externes non continus dans [] et il peut produire certaines règles avec un évènement vide τ en cachant les communications internes non continues dans [].
En fait il n'y a aucune différence entre les deux notations, mais elles sont complémentaires, puisque dans la première on connaît les connecteurs, qu'on

doit enlever, alors que dans la deuxième on connaît les connecteurs, qui restent visibles.

Exemple : Pour l'opérateur d'abstraction, il est intéressant de montrer le cas $k \in K_{INT}$ et les communications le long de k sont invisibles. Considérons la règle dans l'exemple (1.2.2.2.3) et supposons que le connecteur B.M n'appartienne pas à l'ensemble des connecteurs {k1,...,kp} alors la règle deviendra Y-F(m,)$\Rightarrow\tau$ Z-F(MAX(m,n),n)

2.2.4. Expression

Les expressions apparaissent au niveau du système comme le moyen d'évaluer un réseau de processus. La première donnée est donc l'ensemble des processus élémentaires qui sont déjà rangés dans la base de données, la deuxième, les connecteurs qui appartiennent à l'ensemble des processus. Les opérateurs sont union, connexion et abstraction. La syntaxe de construction de l'expression est la suivante :

```
Expression ::= Expression // Expression/
               Expression + connecteur interne /
               Expression [liste de connecteurs]  /
               Nom de processus
Liste de connecteurs ::= connecteur /
                         liste de connecteurs, connecteur.
```

Pour la syntaxe du connecteur (cf. 2.2.1).

Exemples

(1) MAX

(2) MAX // REG

(3) MAX // REG + A.M

(4) MAX // REG + A.M [?.B,M.?]

Une expression peut être simple (1), c'est à dire être un processus, ou être composée uniquement d'opérateurs et d'opérandes (2), (3), (4). Les opérandes sont prédéfinis. Les opérateurs sont exécutés dans leur ordre d'apparition, c'est à dire qu'une expression est évaluée de gauche à droite. Avec ce principe nous avons supprimé les crochets.

Exemple

(P // Q + A.B) // R [C.?,?.D] est évalué

P // Q + A.B // R [C.?,?.D]

III - ARCHITECTURE DU SPARC

Le système a l'ambition de proposer une solution aux problèmes posés par l'introduction des techniques interactives c'est-à-dire d'offrir bien plus de souplesse et de facilité en étudiant les processus en parallèles communicants. L'objectif du système est donc de créer un environnement graphique qu'implique la multiplicité des problèmes conçus de façon indépendante.

Cette réalisation expérimentale a exclusivement pour but de montrer que les études théoriques sur un modèle du parallélisme et des processus communicants peuvent avoir une traduction concrète sous la forme d'un système d'aide à la conception d'applications réparties.

Pour cela, un environnement graphique est crée et on distingue deux dimensions dans l'implantation : l'architecture du système et le mécanisme du fonctionnement. Dans la suite de ce chapitre, j'indique d'abord quels sont, dans cette optique, les éléments principaux du système, les avantages pratiques immédiats que l'on peut attendre d'un tel système et puis je décris d'une manière plus détaillée le fonctionnement du système à travers le menu graphique.

3.1. Schéma fonctionnel de SPARC

Afin d'établir quelle est la place de chaque élément et de décrire son rôle, on commence par introduire le modèle général de fonctionnement de chaque composant. Les besoins exprimés à l'heure actuelle par les concepteurs d'applications réparties convergent généralement vers un système d'aide à la conception et à l'intégration permettant :

- de décrire les différentes parties de l'application avec l'association d'une représentation graphique à la description d'un processus et utilisation de cette représentation pour exprimer les constructions de réseaux

- d'archiver les descriptions en vue de leur réutilisation. Une base de données de description de processus est réalisée et l'accès à cette base de données.

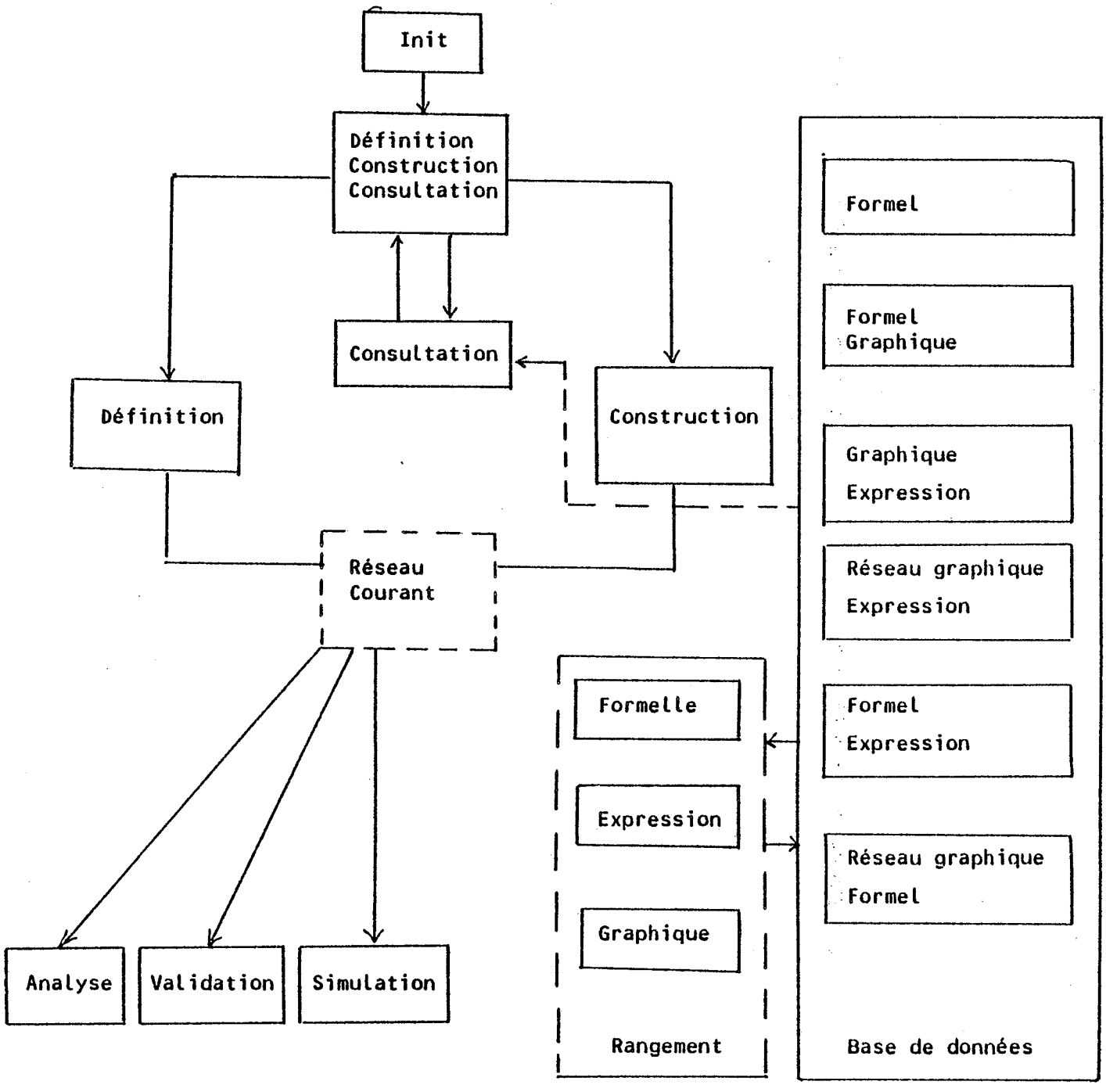
- de construire un réseau de processus

- d'analyser le comportement d'un système

- de valider la réalisation d'un processus par un réseau de processus plus élémentaires

- de simuler leur comportement avec usage de représentation graphique.

Schéma fonctionnel de SPARC



Les fonctions offertes par le poste de travail sont organisées autour de la notion de "Réseau Courant" (RC), qui est l'objet sur lequel chaque fonction opère.

Initialement, le poste de travail est dans un état "neutre" où l'écran propose un choix entre description de processus , construction de réseau et consultation. Les trois fonctions sont en effet les étapes initiales nécessaires pour constituer R.C. L'utilisateur commence donc par choisir l'une de ces trois fonctions.

3.2. Description

Un nouveau langage est conçu [JOR 84] pour pouvoir décrire tous les aspects des processus communicants. Ce langage non algorithmique et de nature simple, permet donc de décrire l'organisation statique de chaque processus et son comportement. La description de processus comprend deux aspects : formel et graphique.
Après le choix, par l'utilisateur, de la fonction de description du processus, l'écran propose le formulaire suivant :

NOM
CONNECTEURS
ETAT
INIT
REGLES

Les paragraphes qui suivent décrivent le contenu de chacune des fenêtres concernant la description des processus.

3.2.1. Connecteurs

3.2.1.1. Syntaxe

```
<IIdentificateur >  ::=  <lettre>  <lettre ou chiffre>
< lettre ou chiffre>  ::=  <lettre>  | <chiffre>
< nom de la  porte > ::=  <Identification> {"<"<chiffre> >}
< nom de la porte d'entrée > ::=  <nom de la porte>
< nom du connecteur > ::=   <nom du connecteur d'entrée> |
                            <nom du connecteur de sortie> |
                            <nom du connecteur interne>

< nom du connecteur d'entrée > ::=   ?. <nom de la porte d'entrée>
< nom du connecteur de sortie> ::=  <nom de la porte de sortie>.?
< nom du connecteur interne >  ::=  <nom de la porte de sortie> .
                                    <nom de la porte d'entrée>

< connecteur > ::=  <nom du connecteur>  : sort
< ensemble des connecteurs > ::=  <connecteur>  ,  {<connecteur>}  ;
```

3.2.1.2. Commentaires

Tout échange des messages s'effectue le long d'un connecteur, chaque connecteur reliant deux portes ou reliant une porte avec l'extérieur. Ainsi si A et B sont des noms de portes, A.B, B.A, A.?, B.?, ?.A, ?.B sont des noms de connecteurs.

Dans cette fenêtre, on décrit chaque connecteur en indiquant son nom et le type des messsages qu'il peut transporter. Par exemple, un opérateur binaire sur des entiers naturels, qui reçoit ses opérandes par les portes A et B et qui renvoie son résultat par la porte C aura les connecteurs suivants

```
NOM : MAX ;
-------------------------------------
CONNECT :          ?.A : NAT,
                   ?.B : NAT,
                   C.? : NAT,
```

3.2.2. Symboles d'états

3.2.2.1. Syntaxe

<nom de variable> ::= <Identificateur en lettres minuscules>
<nom de fonction> ::= <Identificateur en lettres majuscules>
<sort de variables> ::= < à définir >
<sort de fonctions> ::= < à définir >
<état> ::= <nom d'état> : <sort>
<nom d'état> ::= <Identificateur en lettres majuscules>
<ensemble d'états> ::= <état> , {<état>} ;

3.2.2.2. Commentaire

La fenêtre "Etats" sert à définir les symboles, dits "symboles d'états".
Par exemple, si pour décrire le comportement de l'opérateur binaire évoqué plus haut on a besoin de trois symboles d'état, "X" d'arité nul, "Y" et "Z" d'arité 1 avec un argument de type "NAT", on complètera la description de cet opérateur de la façon suivante :

```
NOM : MAX ;
CONNECT :  ?.A : NAT,
           ?.B : NAT,
           C.? : NAT ;
Etats   :  X : ()  ,
           Y : NAT ,
           Z : NAT ;
```

Cette partie là s'appelle signature (Σ), c'est à dire

<signature> ::= <ensemble des connecteurs> <ensemble des états> .

3.2.3. Comportement

3.2.3.1. Syntaxe

```
        <règle>  ::=  <terme d'évènement> <terme d'état>
<terme d'état>   ::=  <nom d'état> |
                      <nom d'état> (<terme fonctionnel> , {<terme fonctionnel>})
<terme fonctionnel>::=  <nom de variable> |
                        <nom de fonction> |
                        <nom de fonction> ( <terme fonctionnel> ,
                                            {<terme fonctionnel >}).
<terme d'évènement>   ::=  <σ>  |  <terme d'état>⟹ <évènement>
<évènement>   ::=  <communication> |
                   <communication>  <évènement>
<communication>  ::=  <connecteur> ( <terme fonctionnel> )
<ensemble des règles>  ::=  <règle>  ; { <règle> }
<règle initiale>  ::=  <nom d'état> |
                       <nom d'état> ( <valeur initiale> ,  <valeur initiale>} )
<valeur initiale>  ::=  <constante>
```

3.2.3.2. Commentaire

Afin de présenter l'exemple, on introduit d'abord la définition du nom de fonction à travers le type abstrait. Un type abstrait est la donnée d'une signature et d'un ensemble d'équations sur les termes, qui définissent la sémantique des opérateurs, en créant les termes d'équivalence. Les opérateurs sont séparés en deux classes : les constructeurs et les opérateurs . Un exemple est le type "NAT" des entiers naturels (on accepte que le type Bool des Booleans est défini avec les contructeurs "true" et "false", et les opérations "or", "and" et "not").

```
Type    NAT
        Cons    0 :→ NAT
                succ : NAT→ NAT
        opns    add, mul : (NAT,NAT) → NAT
                eq, leq : (NAT,NAT) → Bool
                1 : → NAT
        Vars
                m,n : NAT
                add(o,n)              ==    n
                add(succ(m),n)        ==    succ(add(m,n))
                mul(o,n)              ==    o
                mul(succ(m),n)        ==    add(mul(m,n),n)
                leq(o,m)              ==    true
                leq(succ(m),o)        ==    false
                leq(succ(m),succ(n))  ==    leq(m,n)
                eq(m,n)               ==    and(leq(m,n),leq(n,m))
                          1           ==    succ(o)
        fin
```

Lorsqu'on a défini des types abstraits avec leurs opérations primitives, il est souvent nécessaire, pour une application particulière, de déclarer de nouveaux opérateurs. Par exemple, les nouvelles opérations en NAT peuvent introduire :

```
Op
   opns   min,max  :   (NAT,NAT)  → NAT
   vars   m,n  :  NAT
   eqns   min(m,n)  == cond(leq(m,n),n,m)
          max(m,n)  == cond(leq(m,n),m,n)
end
```

En utilisant la fonction max (notée MAX ici) on va pouvoir décrire l'opérateur binaire évoqué plus haut :

```
NOM : MAX
-------------------------------------------------
CONNECT :  ?.A : NAT,
           ?.B : NAT,
           C.? : NAT ;
-------------------------------------------------
TERMES : X : () ,
         Y : NAT,
         Z : NAT ;
-------------------------------------------------
INIT : X ;
-------------------------------------------------
REGLES :    X ==> ?.A(m)  Y(m) ;
          Y(m)  ==> ?.B(n) Z(MAX(m,n));
          Z(m)  ==> C.?(m) X #
```

Ce sera la manière de faire entrer un processus et qui correspond à cette représentation interne :

{{ NOM }} ∂C∂ ..... ∂T∂ .....
∂I∂ ..... ∂R∂ ..... }}

où ∂C∂ , ∂T∂ , ∂I∂ , ∂R∂ sont des délimiteurs et
∂C∂ signifie les connecteurs
∂T∂ signifie les symboles d'état
∂I∂ les règles initiales
∂R∂ les règles

Pour l'opérateur binaire

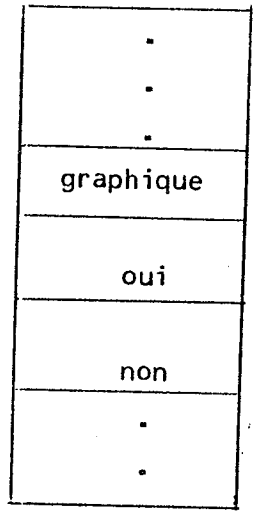
{{ MAX }{ ∂C∂ ?.A : NAT , ?.B : NAT, C.? : NAT ∂T∂ X : (), Y:NAT, Z:NAT
∂I∂ X ∂R∂ X ==> ?.A(m) Y(m) ; Y(m) ==> ?.B(n) Z(MAX(m,n)) ;
Z(m) ==> C.?(m) X }}

Alors cela entraine la définition de processus

<Processus> ::= <signature> <règle initiale> <ensemble des règles>

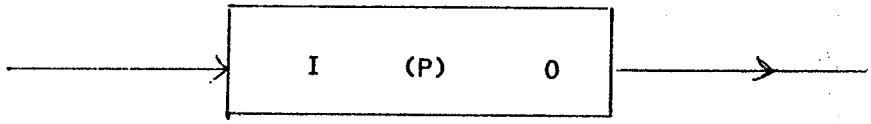
3.2.4. Description graphique

Après avoir fini la description en termes de type abstrait le menu se transforme d'après le schéma suivant :



Si l'utilisateur choisi "oui" alors le système passe dans l'environnement de la description graphique du processus.

L'objet de base est donc la boite et la communication. Voici donc un processus P, ayant deux portes de communication, une porte d'entrée I, et une porte de sortie O ; les fils attachés aux portes sont "en l'air", on considère qu'ils sont reliés à des portes du milieu extérieur que l'on notera '?'



La notion de connecteur tient compte du sens de la communication. Le type du connecteur est défini par rapport à la description formelle. Lorsque la description graphique se fait après la description formelle, les noms des portes et leurs types sont déjà définis. Dans ce cas il est suffisant d'analyser la description formelle et de demander à l'utilisateur de choisir les éléments graphiques. Cette manière d'agir d'une part préserve la cohérence des deux formes de description par rapport aux noms et aux types des portes et d'autre part minimise les erreurs de l'utilisateur.

Plus précisément on va parler de ces éléments de description en 4.7.

3.3. Construction de réseau

3.3.1. Evaluation de l'expression

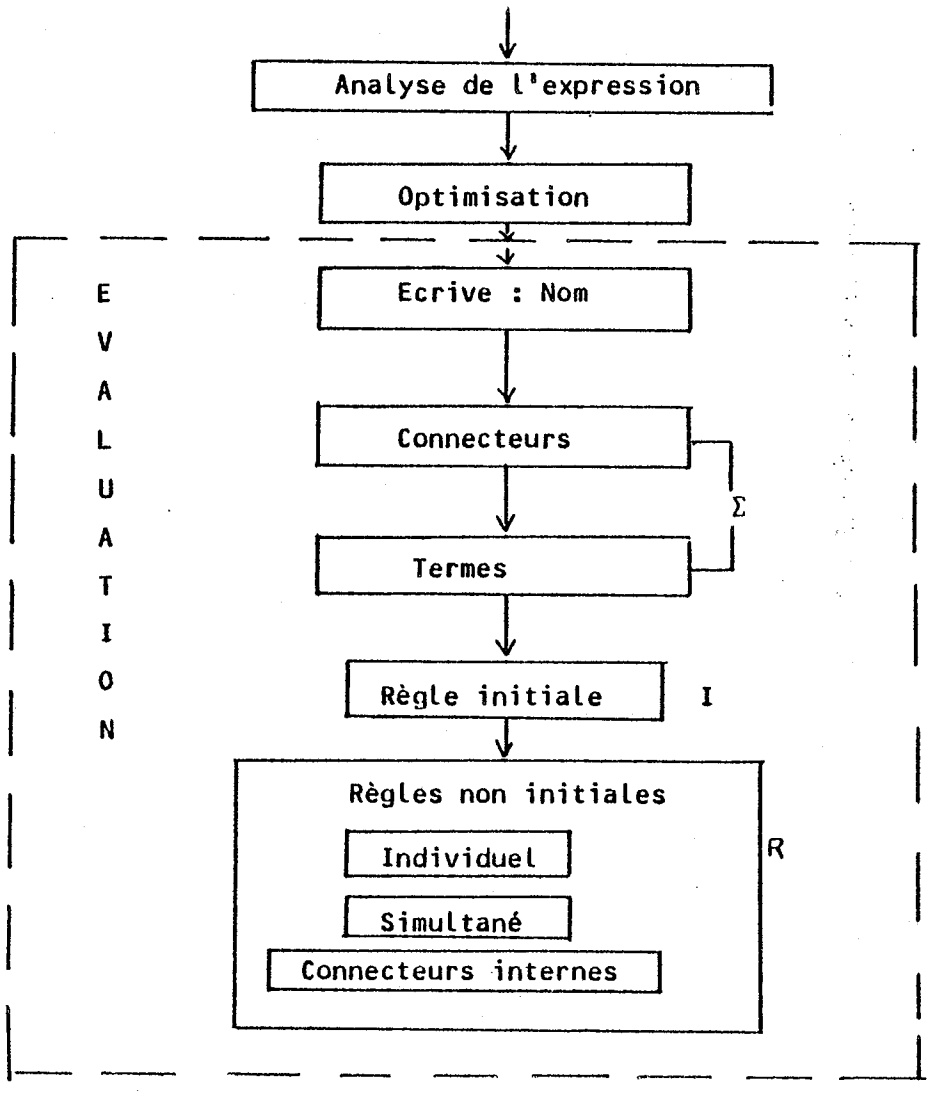
L'expression peut être rangée dans la base de données de deux manières :

- soit cela sera créé de manière graphique
- soit on la fait entrer par clavier

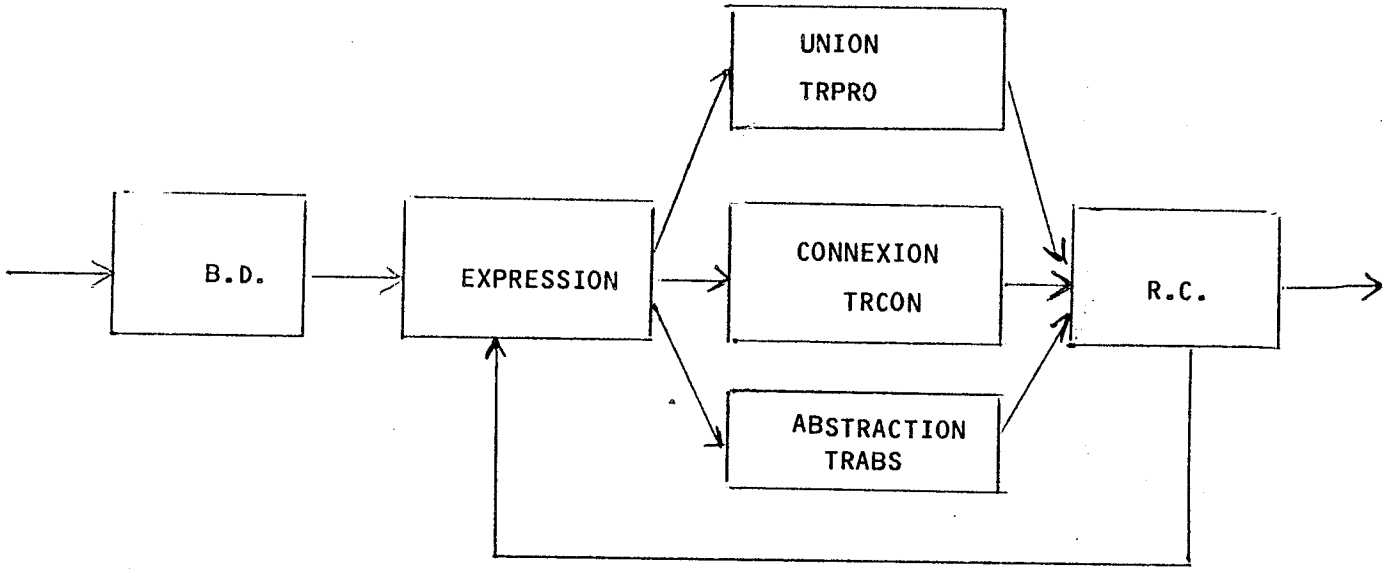
Dans l'évaluation de l'expression on distingue trois parties :

- l'analyse de l'expression
- l'algorithme d'optimisation
- l'évaluation du réseau courant

Le schéma suivant donne une idée générale.



Etant donnée une expression la fonction d'analyse prépare tous les vecteurs nécessaires qu'on utilise par la suite. L'algorithme d'analyse se traduit d'après le schéma :



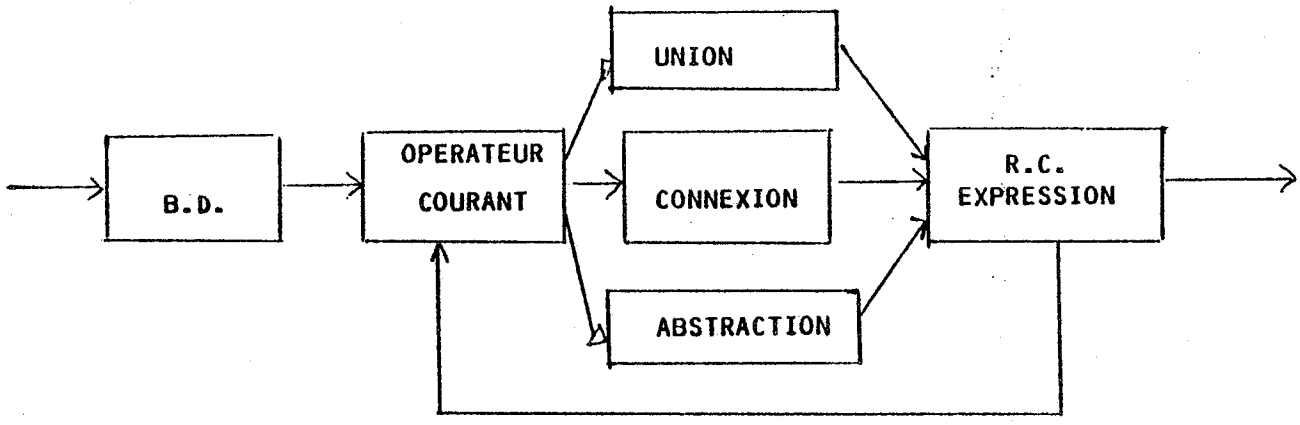
TRPRO, TRCON, TRABS sont les noms des fonctions que nous avons utilisées pour les procédures correspondantes. En présence d'un opérateur l'une des fonctions mentionnées sera activée. En ce qui concerne l'optimisation on en parlera plus précisément dans le chapitre III. Pour la partie évaluation, chaque boite correspond à une fonction de calcul, sauf la dernière qui correspond à trois fonctions. La définition algébrique des opérateurs laisse toujours le résultat de calcul dans la syntaxe présentée en (2.2.). Les fonctions NOM, CONNECTEURS, TERMES, INITIAL, COMSEP, COMSIM, CONSIM sont créées. Les trois dernières correspondent : au comportement individuel, au comportement simultané et aux communications entre les processus. Cela justifie la représentation sur l'écran où les équations sont ordonnées : les premières sont les équations qui décrivent le comportement individuel, ensuite viennent celles qui décrivent le comportement simultané et enfin les équations qui décrivent les communications soit visibles, soit invisibles.

3.3.2. Réseau graphique de processus

Le langage graphique permet au concepteur du système de décrire et d'organiser les éléments de son application. Les supports graphiques d'information constituent un cadre pour exprimer de façon claire

la construction des réseaux de processus et la simulation du comportement.

Etant choisi le modèle descriptif graphique, chaque opérateur de construction du réseau de processus a sa "naturelle" interprétation graphique. La réalisation a en effet un double rôle : elle donne une image très claire du réseau des processus et elle attache à ce réseau une expression équivalente à ceci :



Ayant affiché à la fois les menus des processus et les opérateurs, l'utilisateur peut obtenir le R.C. et l'expression. On voit alors qu'une évaluation de cette expression est possible dans un deuxième temps. Tous les détails sur cette étape sont donnés au chapitre IV.

3.4. Rangement

Pour manipuler les objets, et en poursuivant sur l'idée de schéma, il apparaît nécessaire de les conserver dans une base de données. Dès que l'utilisateur a terminé la description d'un processus, c'est le processus élémentaire que l'on range dans la base de données. Quand l'utilisateur a terminé la conception d'un réseau de processus communicants c'est le réseau que l'on range dans la base des données. Aussi l'aspect graphique des bases des données figure dans le projet SPARC.

Le fonctionnement du système nécessite la création de différents éléments de la base de données et plus précisément :

- la description formelle du processus
- la description graphique du processus
- l'expression formelle du réseau
- la description graphique du réseau

Un processus ou un réseau de processus est rangé dans la base de données par la représentation formelle , par la représentation graphique ou par les deux. Cela dépend du choix de l'utilisateur du système.

3.5. Consultation

A partir de la base de données, il convient de fournir à l'utilisateur les informations nécessaires pour la conduite de l'organisation de son travail. Compte tenu des particularités du système, il est pratique de faire la consultation à plusieurs niveaux, qui dans notre cas se fonde strictement sur le traitement des informations provenant de la partie rangement du système. Pour le travail de consultation, on a besoin d'une grande variété d'informations - c'est-à-dire de connaître aux différents niveaux et de leur corrélation , à savoir :

- le processus élémentaire dans son langage formel de description
- la description formelle et la description graphique
- un réseau de processus et son expression correspondante
- l'expression formelle d'un réseau de processus et sa représentation graphique comme un processus élémentaire
- l'expression formelle d'un réseau de processus et la représensation graphique de ce réseau

Toutes les informations ci-dessus sont schématisée dans le schéma fonctionnel.

Un processus d'ouverture pour le réseau graphique de processus est réalisé. Le processus d'ouverture se fait ici, par trois fonctions, qui s'appellent autant de fois qu'eux apparaissent dans l'expression graphique du réseau. Nous ne faisons pas l'affichage immédiat de l'image, mais on préserve les informations en analysant l'expression jusqu'au bout.

L'organisation de la consultation laisse, en fait, l'utilisateur maitre de sa mise en oeuvre et lui permet, de bien connaître tout ce qu'on a rangé dans la base de données.

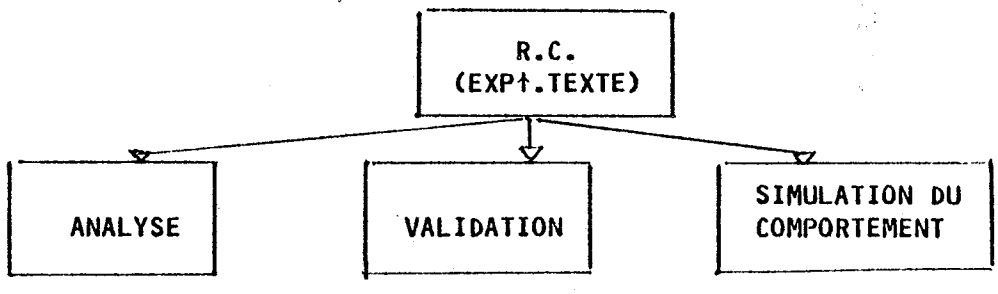
3.6. Phase d'étude du poste de travail

Dans cette partie, plutôt que de développer le fonctionnement du système pour les problèmes évoqués, on en présente les principaux aspects qui seront développés et auxquels ce modèle servira de base. Les fonctions offertes par le poste de travail sont organisées autour de la notion de "Réseau courant" (R.C.) , qui est l'objet sur lequel chaque fonction opère. Jusqu'à maintenant nous avons vu que :

- quand on décrit un processus, c'est R.C. que l'on décrit
- quand on construit un réseau de processus, c'est le R.C. que l'on construit
- c'est à R.C. que l'on associe une représentation graphique.

D'après le schéma fonctionnel on voit bien que toutes les fonctions de la phase d'étude se sont branchées vers R.C. Pour développer le système, il est donc suffisant de connaître le R.C.

Le variable pointeur EXP↑.TEXTE avec des variables auxiliaires TINT et TETEEXP, pour le manipuler, forment dans le système le R.C. Pour ceux, qui travailleront sur les aspects que nous allons mentionner ci-dessous, il est nécessaire de faire la liaison :



Nous avons fait connaître la structure interne du R.C. en 2.2.3.2.

3.6.1. Analyse

Le poste de travail devrait permettre l'analyse statique des propriétés du comportement dynamique. Le langage de SPARC est non algorithmique et on décrit l'ensemble des suites possibles d'évènements qui peuvent avoir lieu dans ce système. A partir de là, on fera l'expression des propriétés à analyser par des formules de la logique temporelle.
En SPARC , cette question sera traitée d'une façon très générale, en introduisant la possibilité de décrire des processus communicants génériques.

3.6.2. Validation

SPARC offrira une fonction de validation. Il est essentiel, pendant la conception d'un système, de pouvoir confronter la description de ce que l'on est en train de construire à une description préalable du système, que l'on veut réaliser. La spécification et la réalisation en cours de construction seront exprimées dans le même formalisme ce qui devrait rendre cette tâche plus naturelle. La validation se distingue alors de l'analyse par les notions de comparaison de processus pour la première et de vérification de propriétés pour la seconde.

3.6.3. Simulation du fonctionnement

En SPARC le fonctionnement du processus sera en général le suivant : en partant de l'état courant (l'état initial étant le premier), tentative de l'unification de celui-ci avec une partie gauche de règle. On effectue les communications en passant dans un état nouveau et ceci se répète. La simulation du fonctionnement d'un processus ou d'un réseau de processus sera liée à l'usage des représentations graphiques, qui va donner une image de la suite des évènements. Par exemple en clignotant l'ensemble des connecteurs , qui participent dans l'évènement courant.

3.6.4. Synthèse

A partir de l'ensemble des processus disponibles dans la base de données et d'une spécification fournie par l'utilisateur du

système l'objet de la fonction de synthèse sera double.

1- déterminer, s'il est possible, de construire un ou plusieurs réseaux réalisant correctement la spécification (appel à la fonction de validation)
2- parmi tous les réseaux satisfaisants la spécification, fournir une ou plusieurs "meilleures solutions".

3.7. Programme d'exploitation

Le but de ce programme est d'employer toutes les commandes, qui sont construites pour décrire les processus, créer les réseaux et les expressions formelles qui correspondent à ces réseaux. Le mécanisme de fonctionnement, le menu, change par rapport à l'environnement où se trouve le système. Le principe de fonctionnement : lors d'un passage du système d'un état à un autre, l'utilisateur devrait choisir sur un ensemble minimal des opérations.

3.7.1. Organisation du système

L'essentiel de la conception du programme d'exploitation se définit ainsi :

- le programme de description est d'abord traduit en un modèle conservant la totalité de l'information
- le programme de création des réseaux graphiques dans le modèle ainsi généré
- l'expression correspondante (synonyme) du réseau de processus est créée
- l'évaluation de l'expression est faite

3.7.2. Gestion et consultation des catalogues

Cette fonction, concernant les catalogues, vise un double objectif :

- permettre la consultation d'un catalogue et la sélection de l'un des objets qu'il contient,
- permettre sa gestion, son enrichissement.

Le système offre à l'utilisateur la possibilité de réutiliser des entités précédemment définies, ceci grâce à quatre catalogues contenant respectivement

- les processus élémentaires
- les expressions
- la description graphique
- les réseaux graphiques

Toutes les entités créées doivent être nommées par l'utilisateur et sont, par conséquent, accessibles grâce à leur nom.

La base de données est composée de plusieurs contextes (partition de la base de données). L'utilité des contextes est qu'ils permettent de ne pas avoir de recherches trop longues à effectuer dans une base de données très grande.

3.7.3. Eléments base du dialogue

Nous rappelons brièvement ici, les règles du dialogue imposées par les objectifs visés (cf. 2.1.). La caractéristique essentielle est que l'utilisateur ne communique avec le système qu'à l'aide de données graphiques, de menus ou de noms.

3.7.3.1. Les fonctions de base du dialogue

Les fonctions de base du dialogue proposées, sont au nombre de trois :

- la collecte des coordonnées
- l'introduction d'un nom
- le menu

3.7.3.1.1. La collecte de coordonnées

Cette opération fondamentale est effectuée dans tous les cas à l'aide d'une tablette, sauf dispositif offrant à l'utilisateur la précision et la facilité d'emploi. Lorsque le système attend une introduction des coordonnées le rôle qui incombe à l'utilisateur est de marquer le point sur tablette.

3.7.3.1.2. Introduction d'un nom

Cette fonction, effectuée à l'aide d'un clavier alphanumérique, permet d'associer un nom quelconque aux différentes entités en vue de les rendre accessibles. Le même nom peut être affecté à deux entités de nature différentes.

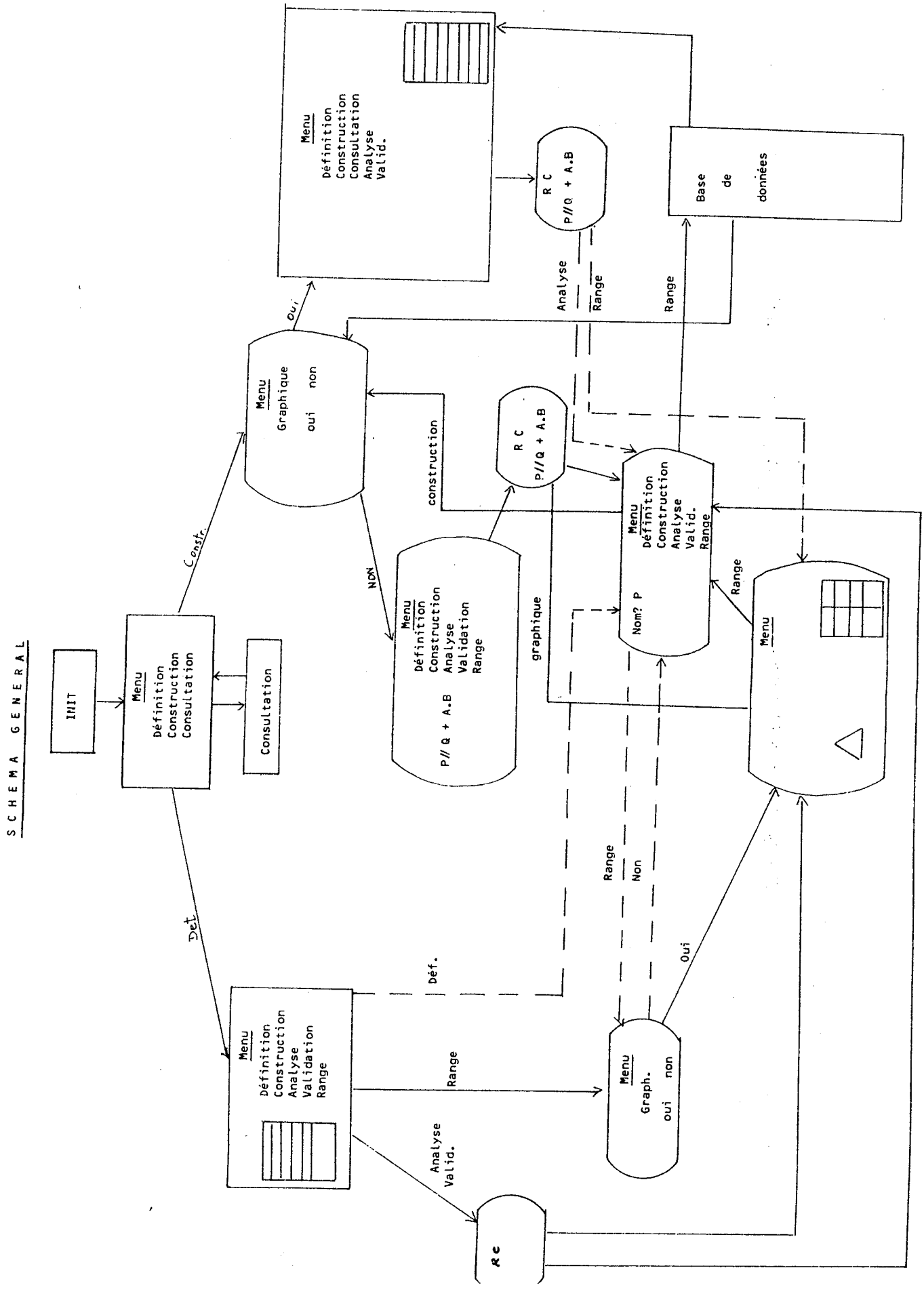
3.7.3.1.3. Menu

Cette fonction comporte deux opération successives :

- le système présente la liste des objets aux commandes disponibles,
- l'utilisateur désigne les objets qu'il désire sélectionner

Au début le système fait apparaître les trois fonctions DESCRIPTION, CONSULTATION, CONSTRUCTION. La commande choisie fait apparaître un nouveau menu et ainsi de suite. L'évolution du menu avec le temps est représentée dans le schéma détaillé suivant.

SCHEMA GENERAL



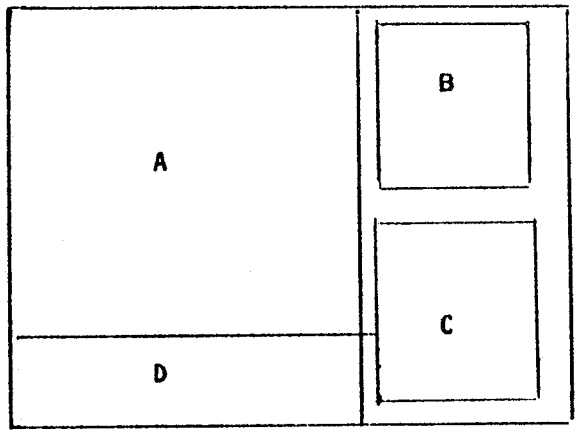
3.7.3.2. Choix de l'utilisateur

Ce choix peut, également, revêtir plusieurs aspects selon la configuration :

- désignation directe de l'objet sur l'écran à l'aide du dispositif de désignation disponible (photostyle, reticule, etc...)
- Appui sur les touches des fonctions
- Désignation sur la tablette, d'une zone réservée à cet effet et associé à l'objet désiré. L'utilisateur peut, dans ce cas, disposer sur la tablette, une feuille de papier sur laquelle sont représentées, dans leurs zones correspondantes, les objets à sélectionner.

3.7.3.3. Gestion de l'écran

La gestion de l'écran est représentée par la figure suivante :



L'écran est séparé en quatre espaces :

- l'espace de travail (A)
- le menu qui contient la liste d'actions (B)
- le menu de description ou le menu de la liste des objets contenus dans les catalogues (C)
- l'espace de dialogue alphanumérique (D)

IV - EVALUATION DE L'EXPRESSION

Le calcul de l'expression est l'un des problèmes centraux en SPARC. Etant donné l'ensemble des processus élémentaires et l'expression on manipule ces processus pour obtenir d'autres processus. Deux problèmes sont posés :

- renommage des connecteurs
- simplification et optimisation du calcul

Dans ce chapitre nous présentons le calcul de l'expression à travers ces deux problèmes. La fonction de renommage est définie en remplaçant d'autres approches proposées dans la littérature pour un renommage automatique en cas de conflit des noms. Le principe d'optimisation est présenté en détail avec l'algorithme correspondant.

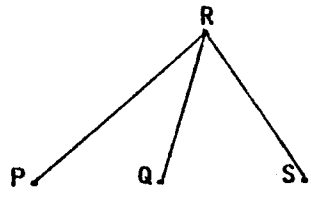
4.1. Renommage

Une application est décrite de façon hiérarchisée selon une structure arborescente dont les feuilles sont des processus élémentaires ou composés.

Par exemple :

R = P // Q + A.D [B.?, A.D, ?.C] // S [B.?, ?.E] (1)

est une expression décrivant une combinaison R des processus P,Q et S.



Les connecteurs externes des processus P,Q,S sont :

pour P : ?.A B.?
pour Q : ?.C D.?
pour S : ?.E B.?

Pendant l'évaluation du premier opérateur d'abstraction, il est clair que le connecteur B.? appartient au processus P, mais dans le deuxième opérateur il faut savoir si B.? appartient au processus P ou au processus S. Ceci nous a conduit à une méthode pour traiter automatiquement le problème du renommage des portes de processus.

4.1.1. Principe

Dans [MIL 80] , [CAN 83] et dans d'autres travaux sur les processus communicants en parallèle, on utilise un opérateur explicite de renommage [S]. Ainsi chez Milner l'opérateur est exprimé [nom1/nom2]. On choisit d'abord une forme particulière pour les identificateurs de portes :

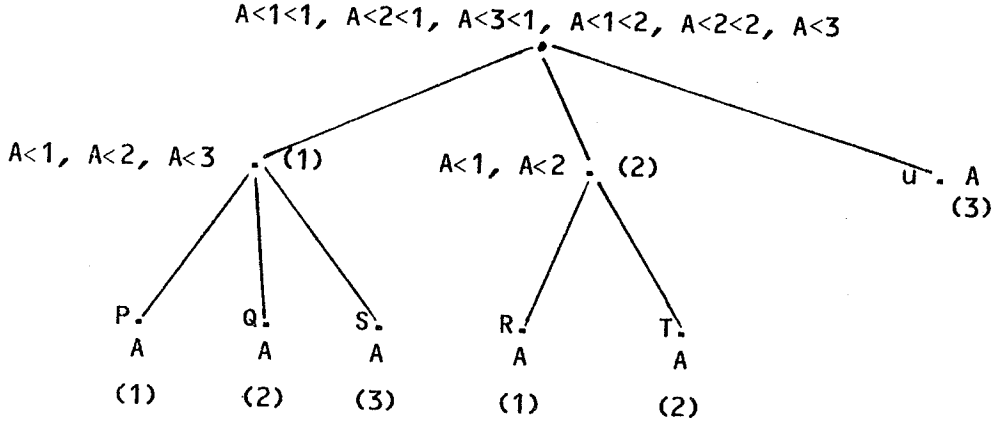
<identificateur> ::= <lettre> , {<lettre> ou <chiffre>}

<identificateur du renommage> ::= <identificateur> { <séparateur> <chiffres

<séparateur> ::= "<"

Exemple : AB1 < 1 < 3 est un identificateur de renommage

On a remplacé l'opérateur de renommage par une fonction de renommage selon le principe suivant : dans une expression donnée s'il y a des portes avec le même nom, l'identificateur de renommage se forme en ajoutant à l'identificateur précédent un séparateur et des entiers qui expriment la position du processus dans l'expression à évaluer. L'exemple ci-dessous illustre ce principe :



On comprend qu'une porte de nom A existe dans tous les processus élémentaires.

4.2.1. Fonction de renommage

Nous introduisons la fonction appelée "nom" de la manière suivante :

nom(identificateur de renommage) = identificateur

Exemples

nom(A) = A
nom(A<1) = A
nom(A<1<2) = A

On accepte uniquement les processus dont les portes ont des noms distincts. Des identificateurs répétés appartiennent donc à des processus distincts.
Une fonction de renommage est construite : soient P1, P2, ..., Pn les processus qui participent à la construction du réseau P, et soient $\lambda 1, \lambda 2, \ldots, \lambda n$ les ensembles identificateurs correspondants aux processus.
On va noter :

$$\lambda_\Sigma = \lambda 1 \cup \lambda 2 \cup \ldots \cup \lambda n$$

$$\lambda_I = \bigcup \lambda i \cap \lambda j \quad i \neq j \quad i,j = 1,\ldots n$$

Nous définissons la fonction de renommage :

$$F : (\lambda_\Sigma , NAT) \rightarrow \lambda$$

$$F(id,ordre) = \begin{cases} id & \text{si } id \notin \lambda_I \\ id < ordre & \text{si } id \in \lambda_I \end{cases}$$

λ : ensemble des identificateurs de renommage
ordre : ordre de l'apparition du processus où se trouve la porte nommée id.
Pour l'expression (1)
λ_1 : ?.A, B.?
λ_2 : ?.C, D.?
λ_3 : ?.E, B.?

λ_Σ : ?.A, B.?, ?.C, D.?, ?.E, B.?

λ_I : B.?

F(?.A,1) = ?.A F(?.C,2) = ?.C F(?.E,3) = ?.E
F(B.?,1) = B<1.? F(D.?) = D.? F(B.?,3) = B<3.?

Le réseau R sera :

R = P//Q + A.D [B<1.?, A.D, ?.E] // S [B<3.?, ?.E]

où B<1.? ∈ P, B<3 .? ∈ S

Si nous considérons l'expression suivante :

R = P// Q + A.D // S (2)

On ne peut rien dire sur le problème de renommage. Ceci nous a conduit vers un traitement du problème de renommage en deux points.

4.1.3. Renommage de l'expression

L'expression est créée soit par le langage graphique (cf. ch. 4), soit par l'utilisateur.

4.1.3.1. Graphique

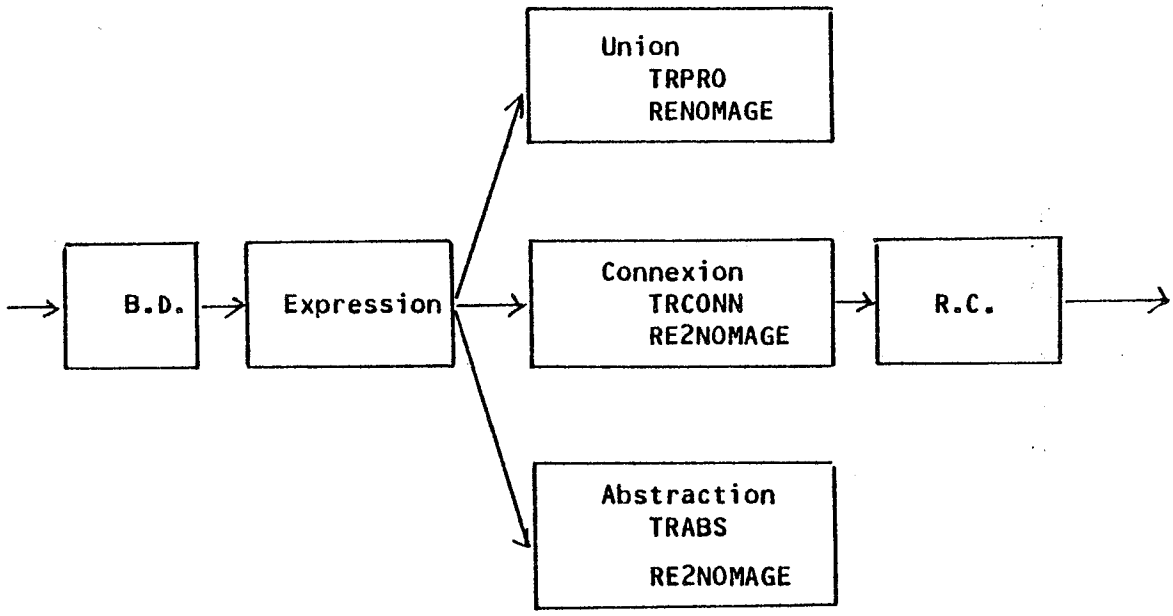
Le renommage de l'expression est confié au logiciel correspondant (Cf. 5.4.1.)

4.1.3.2. Utilisateur

Dans ce cas, c'est l'utilisateur qui doit écrire l'expression en renommant les portes selon le principe 4.1.1. Les portes qui appartiennent aux opérateurs de connexion et d'abstraction sont renommées. La procédure RE2NOMAGE met en correspondance les portes avec les processus correspondants, pendant le calcul de l'expression (cf. Schéma 4.1.4).

4.1.4. Renommage des processus

L'exemple (2) suggère l'idée de renommage des portes de processus. Le schéma de l'analyse de l'expression est modifié :



Ceci signifie que le processus de renommage est activé chaque fois que l'opérateur d'union est présent. Nous rappelons que les fonctions TRPRO, TRCONN, TRABS sont définies en 3.3.1.

4.2. Optimisation

Dans le chapitre I nous avons défini les opérateurs sur les processus. Le calcul de l'expression est l'un des problèmes fondamentaux du projet SPARC, qui pose des problèmes à cause de l'opérateur d'union, qui multiplie la quantité des règles. Si on considère P = P1//P2, soit :

e' = card(E1)
e" = card(E2)
g' = card(G1) , G1 $\in$ Σ1
g" = card(G2) , G2 $\in$ Σ2

la cardinalité de l'ensemble des règles de P est alors :

e = card(E) = e'.g"+e".g'+e'.e"
e'.g" - tiennent compte du fonctionnement de P1
e".g' - tiennent compte du fonctionnement de P2
e'.e" - tiennent compte du fonctionnement simultané de P1 et P2.

L'union de deux processus n'est donc pas seulement le produit cartésien de leurs actions en parallèle, mais aussi le produit cartésien de leur inactivité possible avec les actions de l'autre. L'opérateur de connexion ne rajoute qu'une règle et l'opérateur d'abstraction ne fait que la suppression de règles.

Sipour le réseau RES = MAX//REG + M.A [B.?, ?.C] le résultat a 7 règles, par contre dans l'opérateur d'union seul nous aurions 24 règles. Lorsque l'on se trouve en présence d'une expression, on peut savoir calculer l'expression telle qu'elle est, soit en rechercher les propriétés. En particulier, on peut essayer de montrer que deux expressions ont le même résultat de calcul. Nous cherchons à obtenir des résultats indépendants de l'interprétation de l'expression. L'idée qui va nous guider est celle-ci : nous voulons remplacer une expression par une autre qui ait le même résultat, mais qui occupe le minimum de mémoire. On se pose alors deux problèmes :

- élimination de la redondance
- équivalence des résultats de calcul

4.2.1. Elimination de la redondance

D'après ce que nous avons dit dans le paragraphe précédent, la redondance provient de l'opérateur d'union et c'est l'opérateur d'abstraction qui l'élimine. A partir de là, le principe d'élimination de la redondance est : l'opérateur d'abstraction doit être exécuté le plus tôt possible. L'expression sera alors transformée. On peut alors noter

que : tous les problèmes que l'on va étudier ne tiennent compte que des résultats finals de l'expression, mais pas des résultats intermédiaires, qui seront tout à fait différents.

4.2.1.1. Propriété sur les opérateurs.

Afin d'étudier la transformation de l'expression, il est utile de préciser ici certaines propriétés portant sur les opérateurs.

Union : l'opérateur d'union est commutatif : P1//P2 = P2//P1

associatif : (P1//P2)//P3 = P1//(P2//P3)

On omettra en général les parenthèses : P1//P2//P3.

Abstraction : le résultat du calcul d'une expression se terminant par une abstraction est égal au résultat du calcul d'une nouvelle expression obtenue à partir de la première en supprimant tous les opérateurs d'abstraction sauf le dernier.

$$P1//P2 + Ci.Cj\ [\overset{1}{..}]\ //P3 + C_{i1}.C_{j1}\ [\overset{2}{..}] = P1//P2 + Ci.Cj//P3 + C_{i1}.C_{j1}[\overset{2}{..}]$$

Ceci comprend tous les cas, c'est-à-dire si l'expression a la forme :

$$P1//P2 + Ci.Cj\ [\overset{1}{..}]\ //P3 + C_{i1}.C_{j1}$$

On imagine alors un deuxième opérateur d'abstraction :

$$[\overset{2}{..}] = [\overset{1}{..}] \quad U \quad (\text{ensemble des connecteurs du P3}) \ U \ (C_{i1}.C_{j1})$$

Dans ce cas :

$$P1//P2 + Ci.Cj\ [\overset{1}{..}]\ //P3 + C_{i1}.C_{j1} = P1//P2 + Ci.Cj//P3 + C_{i1}.C_{j1}\ [\overset{2}{..}]$$

Connexion : L'opérateur de connexion est exécutable à condition que les connecteurs externes participants ne soient pas supprimés :
Exemple : MAX//REG + M.A [?.B,M.?] : la connexion est exécutable
MAX//REG [?.B,M.?] + M.A : la connexion n'est pas exécutable, parce que le connecteur ?.A a été supprimé.

Les opérateurs de connexion et d'abstraction ne sont pas commutatifs : dans une expression on ne peut pas toujours changer l'ordre d'évaluation des opérateurs de connexion et d'abstraction. L'exemple précédent montre bien cette propriété.
A partir de là nous pouvons dire que les opérateurs de connexion et d'abstraction sont commutatifs à condition que les connecteurs externes qui forment le connecteur interne de la connexion appartiennent à l'opérateur d'abstraction. Dans ce cas l'opérateur de connexion sera exécutable, mais l'équivalence de résultat n'est pas assurée.

4.2.1.2. Transformation de l'expression

La conclusion que l'on tire du paragraphe précédent est la suivante : quand on change la place des opérateurs de connexion et d'abstraction, on perd les informations nécessaires pour évaluer l'opérateur de connexion et l'équivalence du résultat n'est pas assurée. Pour résoudre ces problèmes, nous allons distinguer deux cas :

Premier cas : les connecteurs externes correspondant à l'opérateur de connection appartiennent à l'opérateur final d'abstraction.
On va noter : C_A l'ensemble des connecteurs internes
A_f l'ensemble des connecteurs de l'opérateur final d'abstraction
Le résultat du calcul est obtenue en faisant d'abord l'union de tous les processus soumis individuellement à l'opérateur final d'abstraction, puis l'ensemble des connexions C_A, et enfin le résultat soumis à l'opérateur final d'abstraction : :

$$P = ((P1\ [A_f]\ )//\ (P2\ [A_f])\ \ldots\ (Pn[A_f]\ ) + \{C_A\}\ )\ [A_f]$$

où par $\{C_A\}$, on comprend la répétition de l'opérateur de connexion, autant de fois que la cardinalité de $\{C_A\}$.

Deuxième cas : Les connecteurs externes correspondant à l'opérateur de connection n'appartiennent pas à l'opérateur final d'abstraction ou lui appartiennent partiellement.
On va noter : A_c l'ensemble des connecteurs externes qui participent à la construction de C_A mais qui n'appartiennent pas à A_f .
Le résultat final sera alors :

$$P = ((P1\ [A_f \cup A_c])//(P2[A_f \cup A_c]) \ldots (Pn[A_f \cup A_c] + \{C_A\})\ [A_f]$$

4.2.1.3. Equivalence

Soit une expression ayant la forme ci-dessous :

$$P1//P2 + \{Ci.Cj\}\ [f_1]\ //P3 + \{C_{i1}.C_{j1}\}\ [f_2]$$

où $\{Ci.Cj\}$, $\{C_{i1}.C_{j1}\}$ sont des ensembles de connecteurs internes et f_1, f_2 des ensembles de connecteurs internes ou externes.
Toute expression peut se ramener à cette forme. En effet, on peut remplacer l'un quelconque des Pi (i = 1,2,3) par une expression de la même forme, et ceci récursivement.
Nous allons assurer l'équivalence de résultat des deux expressions suivantes :

(a) $P1//P2 + \{Ci.Cj\}\ //P3 + \{C_{i1}.C_{j1}\}\ [f_2]$

(b) $((P1[(f_2 \cup A_c) \cap \Lambda_1])//(P2\ [(f_2 \cup A_c) \cap \Lambda_2]) + \{Ci.Cj\}\ //$

$$(P3[(f_2 \cup A_c) \cap \Lambda_3]) + \{C_{i1}.C_{j1}\})\ [f_2]$$

où Λ_1, Λ_2, Λ_3 sont l'ensemble des connecteurs de P1, P2, P3.

En notant $P'i = Pi\ [(f_2 \cup A_c) \cap \Lambda i]$, l'expression (b) se transforme en :

(b') $P'1//P'2 + \{Ci.Cj\}\ //P'3 + \{C_{i1}.C_{j1}\}\ [f_2]$

D'après la définition de l'opérateur d'abstraction :

$$P'i \subset Pi$$

Pour les expression (a) et (b'), les connecteurs internes sont semblables. D'après la définition de l'opérateur d'abstraction :

$$P'1//P'2 +\{Ci.Cj\} //P'3 +\{C_{i1}.C_{j1}\} \subset P1//P2 +\{Ci.Cj\} //P3 + \{C_{i1}.C_{j1}\}$$

Après l'évaluation de l'opérateur d'abstraction :

$$P'1 //P'2 +\{Ci.Cj\} //P'3 + \{C_{i1}.C_{j1}\} [f_2] = P1//P2 +\{Ci.Cj\} //P'3 +\{C_{i1}.C_{j1}\}[f_2]$$

Prouvons maintenant l'égalité. Supposons qu'une règle $r_i \in$ (a) et $r_i \notin$ (b')
On va noter :

CIN les règles du comportement individuel
CSI les règles du comportement simultané
CCM les règles de communications

si $\tau_i \in$ CIN(a) alors $r_i \in$ CIN(b'), parce que c'est ce que nous avons préservé
$\tau_i \in$ CSI(a) alors $r_i \in$ CSI(b'), parce que CIN(a) = CIN(b')
$r_i \in$ CCM(a) alors $r_i \in$ CCM(b') parce que CCM(a) = CCM(b')

Ceci montre que toutes les règles de (b') sont dans (a) et réciproquement on trouve que, celles de (a) sont dans (b'). Les résultats sont donc égaux, et le principe d'optimisation choisi est donc correct.

Remarque : du point de vue de l'implémentation nous faisons le calcul de Pi $[f_2 \cap \Lambda_i]$ pour éliminer l'évaluation du dernier opérateur d'abstraction. Pour l'ensemble $A_c \cap \Lambda_i$ le logiciel d'application est capable de reconnaître les règles nécessaires pour l'évaluation de l'opérateur de connexion, parce que nous faisons le calcul des règles du résultat.

4.2.2. Modélisation numérique des processus

Un processus (cf. ch. I) est représenté dans sa forme algébrique par le triplet (Σ,I,E). Pour représenter les calculs, nous avons une modélisation numérique des processus. D'après la définition $\Sigma = C \cup G$ (C : ensemble des connecteurs, G : ensemble des symboles d'état), $C = \{c_i\}$ $i = 1...k$, $G = \{g_j\}$ $j = 1...l$. Pour chaque $c_i \in C$ nous avons fait la

correspondance c_i <--> i. De même que chaque $g_j \in G$ nous avons fait la correspondance g_j<--> j. Pour l'ensemble des règles $\{(w_i,v_i)\}$ où $w_i = v_j, \pi_i)$ $(v_i,v_j$: termes d'état, π_i : évènement), chaque terme d'état correspond à un symbole d'état v_j <--> R_j, v_i <--> R_i, $1 \leq R_j, R_i \leq l$. Les connecteurs, qui appartiennent à π_i constituent un sous-ensemble de C. Pour ce sous-ensemble, on trouve le sous-ensemble des naturels $K1 \subset K$, $K = \{1,...,k\}$. Pour une règle quelconque r_i nous aurons alors :

$$(w_i,v_i) \quad <---> R_j \quad ==> K1_i \quad R_i$$

Exemples

Pour les processus MAX et REG (cf. Ch.I)

```
Processus  MAX
Connect    3
Etats      3
Init       1
Règles     1  ==> 1  2
           2  ==> 2  3
           3  ==> 3  1

Processus  REG
Connecteur 3
Etats      2
Init       1
Règles     1  ==> 2  1
           1  ==> 3  2
           2  ==> 1  1
```

Pour l'évènement τ , le sous-ensemble des connecteurs est $K1_i = \emptyset$

4.2.3. Analyse et modélisation de l'expression

L'analyse de l'expression se fait de gauche à droite (cf.2.2.2.4). Chaque fois que l'opérateur d'union apparaît, la fonction TRPROC est appelée. Cette fonction est chargée de faire la modélisation numérique du processus. Pour cela quatre tableaux sont créés :

TABIN [1..N] contient l'index du processus dans la base de données.

TAB2IN[1..N] nombre de symboles d'état

TABP2[1..M,1..3] numéro des symboles d'état gauche et droite des règles.

TAB2[1..M,1..P] pour chaque règle le sous-ensemble $K1_i$.

Exemple : Considérons l'expression

MAX//REG + M.A [?.B,M.A,M.?]

Soient MAX, le processus qui est rangé le premier dans la base de données et REG le deuxième alors nous avons les tableaux suivants :

TABIN

1
2

TAB2IN

3
2

TABP2

P	TG	TD
1	1	2
1	2	3
1	3	1
2	1	1
2	1	2
2	2	1

TAB2

P	CONNECT
1	1
1	2
1	3
2	2
2	3
2	1

En présence d'opérateurs de connexion, la fonction TRCONN détermine les connecteurs externes participant à la connexion.
Le tableau TABPORT[1..T,1..4] contient les informations suivantes :

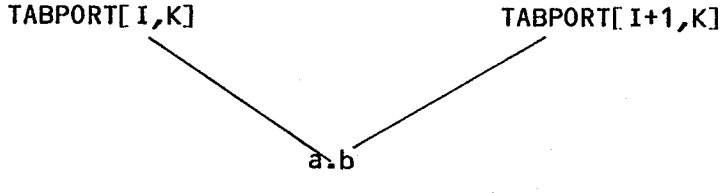
TABPORT[I,1] processus auquel appartient le connecteur

TABPORT[I,2] ordre du connecteur dans le processus

TABPORT[I,3] ordre de la règle à laquelle appartient le connecteur

TABPORT[I,4] symbole de renommage

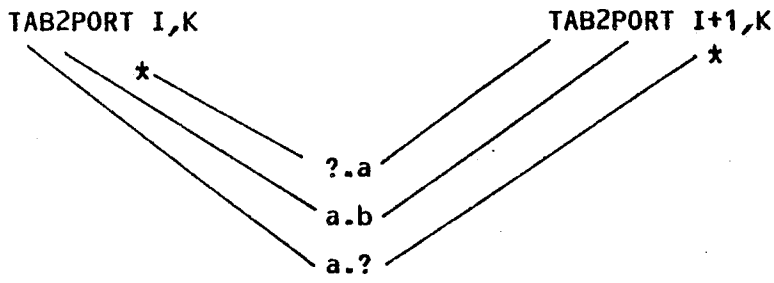
Un connecteur interne est représenté sur deux entrées de TABPORT :



Exemple : pour le connecteur interne M.A de l'exemple précédent

M :	2	2	1	0
A :	1	1	1	0

En présence de l'opérateur d'abstraction, la fonction TRABS détermine les connecteurs participants à cet opérateur. Cette fonction prépare des informations, rangées dans le tableau TAB2PORT[1..S,1..4], qui ont la même structure que celles rangées dans TABPORT. Mais ici, on distingue par une marque les connecteurs externes.
Plus précisément :



Cette marque sert à distinguer les trois sortes de connecteurs. Chaque fois que l'on se trouve en présence d'opérateur d'abstraction, le tableau TAB2PORT[1..S,1..4] est réinitialisé. Ceci exprime le fait que l'on exécute que le dernier opérateur d'abstraction.
Les connecteurs internes sont, soit visibles, soit invisibles (cf.2.2.2.2.). Visibles seront les connecteurs qui existent à la fois dans TABPORT et dans TAB2PORT et invisibles ceux qui n'existent, que dans TABPORT.

4.3. Evaluation de l'expression

Etant donnée une expression, on veut obtenir le résultat sous la forme d'un processus élémentaire, c'est-à-dire , que le résultat est la spécification du réseau associé à l'expression. La forme de représentation est (Σ,I,E) où $\Sigma = C \cup G$.

4.3.1. Connecteurs

L'ensemble des connecteurs qui existe après le calcul, se trouve dans le dernier opérateur d'abstraction. Deux cas se présentent :

- les connecteurs externes
- les connecteurs internes

Pour les premiers on a toutes les informations nécessaires dans la base de données. On les enlève de là pour les mettre dans le résultat du calcul. La fonction de renommage est activée à cette occasion.
Pour les autres on doit d'abord générer le nom et le type, selon le schéma :

<nom du connecteur interne> ::= <nom de la porte d'entrée>.
<nom de la porte de sortie>.
<type de message> ::= <type de message de la porte de sortie>

Il faut mettre en évidence qu'il y a des cas, où les connecteurs internes ou externes appartenant au dernier opérateur d'abstraction, cependant, à cause de la simultanéité, ils n'appartiennent pas aux règles qui décrivent le comportement du processus résultat.
Considérons le processus BEEP qui envoie un signal "zz" de sa porte S, chaque fois qu'une valeur de type NAT passe par sa porte K :

```
BEEP = processus
       connecteurs ?.K : NAT
                   K.? : NAT
                   S.? : SIG
       Etats       Q : ()
       Init        Q(0)
       Règles      Q ⟹ ?.k(q) K.?(q) S.?(zz) Q
fin
```

Pour l'expression RES = MAX//BEEP+K.A [?.B,K.A,C.?]
le connecteur K.A existe dans le dernier opérateur d'abstraction, mais à cause de la simultanéité la règle qui contient le connecteur K.B disparaît . Le résultat du calcul est :

```
RES = Processus
      Connecteurs ?.B : NAT
                  C.? : NAT

      Etats       X-Q : ()
                  Y-Q : NAT
                  Z-Q : NAT

      Init        X-Q
      Règles      Y-Q(m) ⇒ ?.B(n)  Z-Q(MAX(m,n))
                  Z-Q(m) ⇒ C.?(m) X-Q
      fin
```

4.3.2. Symboles d'état

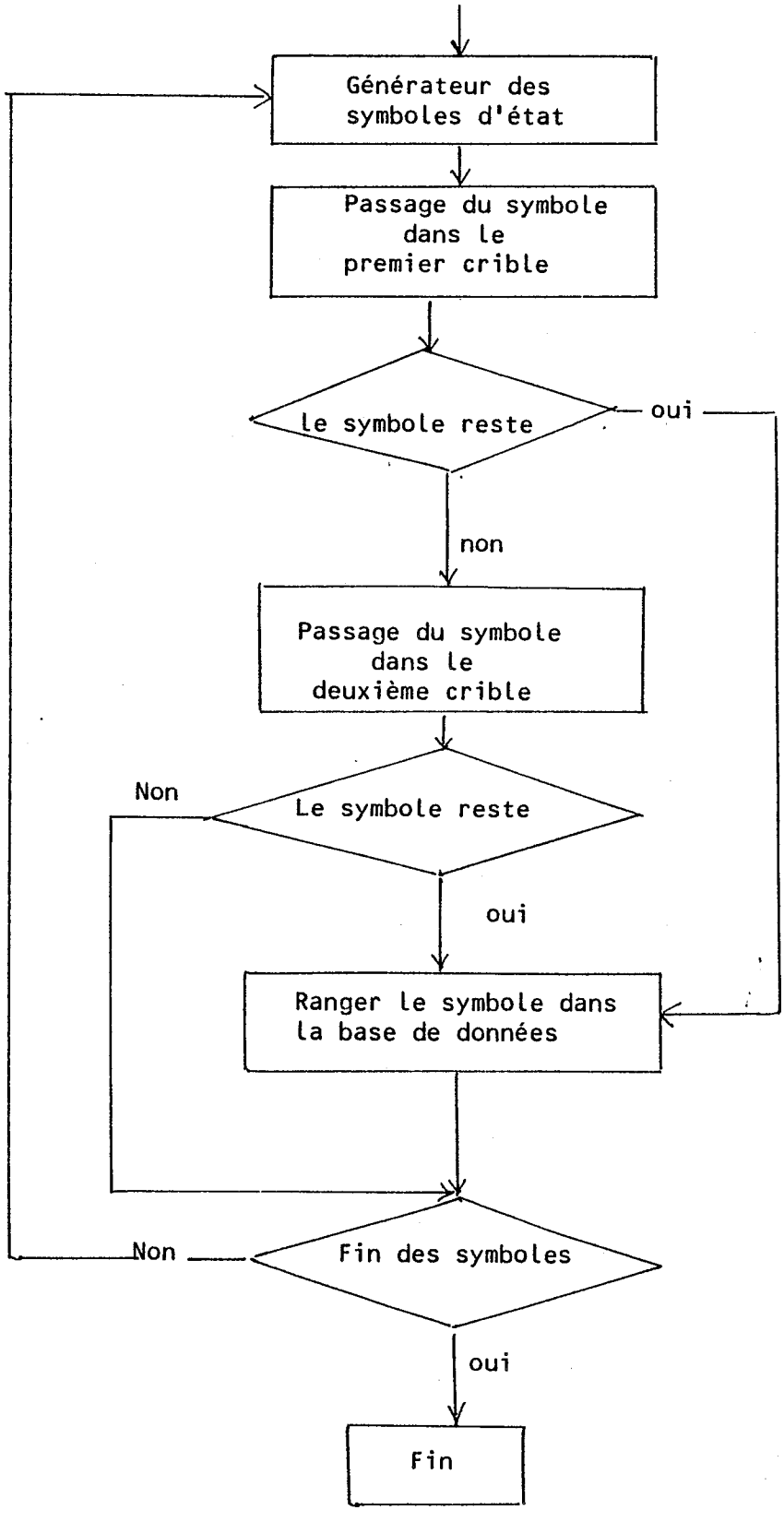
D'après la définition (cf.2.2.2.1.) pour un ensemble de processus $P_1, \ldots, P_n$, où la cardinalité des symboles d'état pour chacun est $g_1,\ldots,g_n$, le nombre des symboles d'état sera $g_1 * g_2 * \ldots * g_n$. En fait, après le calcul, tous les symboles d'état engendrés ne sont pas employés. Certains d'entre eux disparaissent en effet à cause de l'opérateur d'abstraction. Par exemple, pour l'expression :

MAX//REG + M.B [?.L,C.?]

Dans l'ensemble des règles, cinq symboles d'état sont utilisés. Mais pour MAX//REG on va générer six symboles d'état. Ceci crée un problème de simplification des symboles d'état : il suffit en effet de conserver les symboles d'état qui appartiennent aux règles du résultat.

4.3.2.1. Simplification

Le processus de simplification passe par trois phases. Schématiquement l'idée générale est la suivante :



4.3.2.1.1. Génération des symboles d'état

Pour faciliter la compréhension des algorithmes, nous allons présenter ici l'idée d'un générateur, qui est utilisé partout dans ce logiciel. Le problème est le suivant :

Ayant un vecteur $(N_1,N_2,...N_k)$ où N_k sont des naturels et $N_k \geqslant 1$, il faut générer l'ensemble des vecteurs $(V_1,V_2,...,V_k)$ où chaque V_i va parcourir toutes les valeurs $1 \leqslant V_i \leqslant N_i$

Exemple

Si le vecteur initial est (2,1,2) alors on aura :

(1,1,1) (1,1,2) (2,1,1) (2,1,2)

Ceci nous conduit vers une procédure récursive.

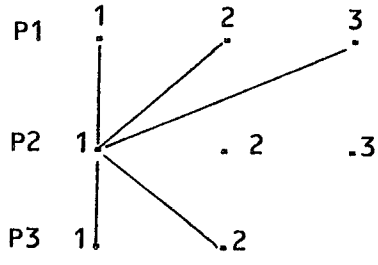
Soient $P_1, P_2,...,P_n$ les processus participant à la construction du réseau et $g_1, g_2,...,g_n$ le nombre de symboles d'état pour chaque processus. Le générateur fonctionne avec l'ensemble $G = g_1,g_2,...,g_n$ en donnant comme résultat toutes les fusions possibles entre les symboles d'état. L'ordre de produit des symboles d'état est l'ordre d'apparition des processus dans l'expression. L'ensemble des équations du résultat contient :

- les règles du comportement individuel du processus
- les règles du comportement simultané
- les règles qui décrivent les communications entre les processus.

Pour obtenir tous les états possibles il est suffisant de ne contrôler que le comportement individuel et les communications.
Pourquoi ne tient on pas compte du comportement simultané ?
Une règle du comportement simultané est issue du produit d'un ensemble des règles du comportement individuel et cette règle existe ssi les règles du comportement individuel existent après le calcul. Alors l'ensemble des symboles d'état des règles du comportement simultané est un sous-ensemble de l'ensemble des symboles d'état des règles du comportement individuel (cf.2.2.2.1.1.).

4.3.2.1.2. Premier crible

Le générateur donne tous les symboles d'état possibles. Si dans le vecteur $V = (v_1, v_2, \ldots, v_n)$ au moins une des composantes appartient aux symboles d'état des règles des processus élémentaires qui décrivent le comportement individuel du réseau , alors ce symbole d'état reste pour la spécification du réseau (cf.2.2.2.1.1.1.) .On dira qu'il passe le crible .
Soient P1,P2,P3 trois processus et G = {3,3,2}



L'une des règles du comportement individuel du processus P2 contient le symbole d'état 1. D'après la définition les fusions

(1,1,1) (2,1,1) (3,1,1) (1,1,2) (2,1,2) (3,1,2)

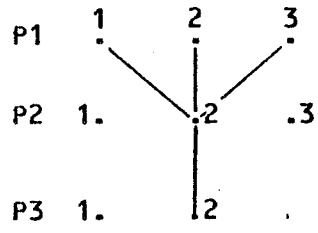
seront présentes dans le résultat.

4.3.2.1.3. Deuxième crible

Une règle de communication peut produire de nouveaux symboles d'état? Ceci vient du fait, que pour les règles de communication, même si les règles du comportement individuel, qui les produisent, disparaissent, elles restent dans le résultat du calcul. Les connecteurs sont, soit visibles, soit invisibles. On dira qu'un symbole d'état passe le deuxième crible s'il n'a pas passé le premier et si c'est un symbole d'état d'une des règles de communication.

Tout symbole d'état qui aura passé l'un des deux cribles sera conservé dans la spécification du réseau.

Soient les trois processus P1,P2,P3 du paragraphe précédent :



Supposons que $2 \notin P2$ et $2 \in P3$ appartiennent aux règles qui produisent une règle de communication quelconque.
La fusion (2,2,2) par exemple ne passe pas le premier crible, cependant elle passe le deuxième et sera donc présente dans la spécification du réseau.

4.3.2.2. La syntaxe des symboles d'état

Nous avons élargi la syntaxe des symboles d'état donnée en 3.2.2.

<symbole d'état> ::= <symbole d'état> - {symbole d'état}
<sorte> ::= <sorte> U {<sorte>}
<état> ::= <symbole d'état> : <sorte>

4.3.2. Calcul des règles

L'ensemble des règles d'un réseau de processus se compose des trois sortes de règles suivantes :

- les règles qui décrivent le comportement individuel
- les règles qui décrivent le comportement simultané
- les règles de communications

4.3.2.1. Comportement individuel

D'après la définition de l'opérateur d'union (cf.2.2.2.1.), le modèle choisi est un modèle synchrone. C'est une question fondamentale dans le calcul de l'expression. Le deuxième problème, qui se pose, est de reconnaître les règles qui seront présentes après l'exécution de l'opérateur d'abstraction.

4.3.2.1.1. Choix des règles du comportement individuel

Les règles qui existent sont celles qui ont tous les connecteurs dans le dernier opérateur d'abstraction. TAB2(cf.4.2.3.) contient pour chaque règle un vecteur représentant l'ensemble des connecteurs de cette règle.

Soit v_{ki} le vecteur correspondant à $r_{ki} \in P_k$, soit v'_{ki} le vecteur suivant :

$$e_j = \begin{cases} 1 & \text{si } (c_j \in v_{ki}) \text{ et } (c_j \in \text{TAB2PORT}) \\ 0 & \text{sinon} \end{cases}$$

pour tous les $c_j \in v_{ki}$, e_j est un composant quelconque de $V'k_i$

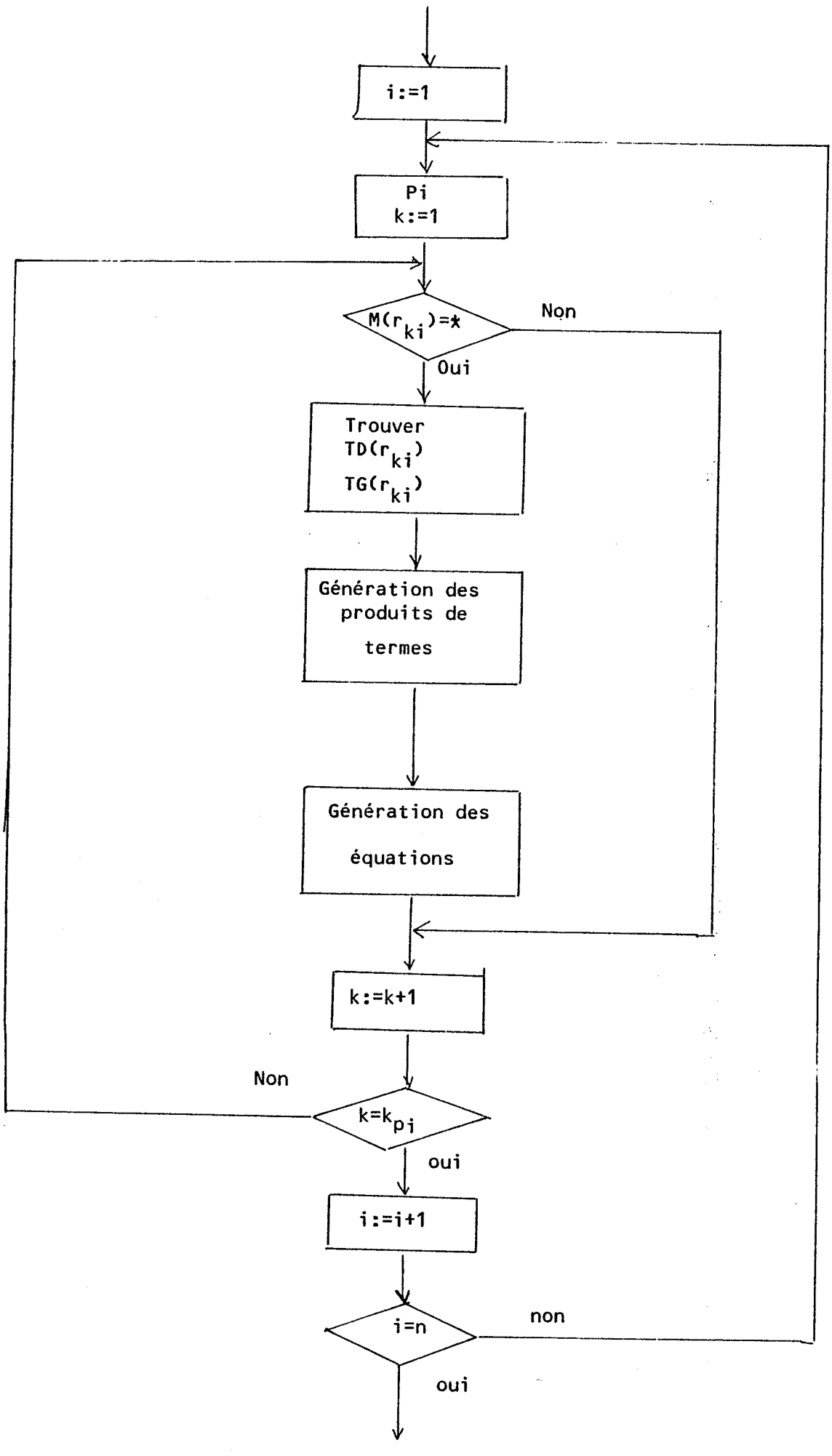
TAB2PORT est l'ensemble des connecteurs de l'opérateur d'abstraction. Si tous les composants de v'_{ki} sont égaux à 1 alors la règle existe sinon elle disparaît. Pour les règles qui restent nous avons choisi une marque pour les distinguer des autres :

$$v_{ki} = (C_1, \ldots, C_k, *)$$

4.3.2.1.2. Synchronisation

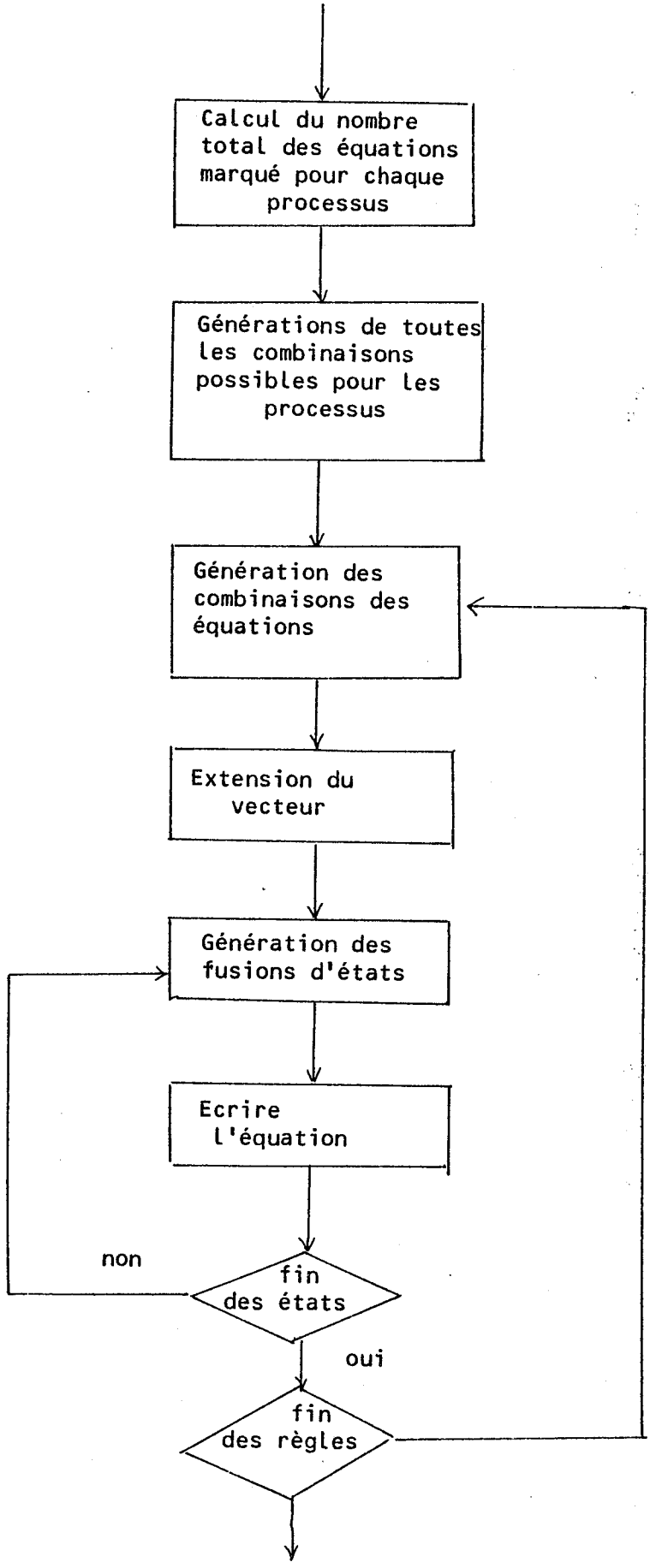
La synchronisation, c'est le problème de la création des termes d'état pour le terme d'évènement et le terme d'état. Nous avons utilisé les symboles TD pour le premier et TG pour le deuxième où TD, TG $\in r_{ki}$ et $r_{ki} \in P_k$.

Soient $g_1, g_2, \ldots, g_{k-1}, g_{k+1}, \ldots, g_n$ les cardinalités des symboles d'état pour $P1, P2, \ldots P_{k-1}, P_{k+1}, \ldots, Pn$. Alors les fusions $g_1, g_2, \ldots, g_{k-1}, TD, g_{k+1}, \ldots, g_n$ et $g_1, g_2, \ldots, g_{k-1}, TG, g_{k+1}, \ldots, g_n$ donnent tous les symboles d'état pour la règle r_{ki} (cf. 4.3.2.1.1.). Cela sera associé aux termes fonctionnels correspondant . Le schéma fonctionnel pour le comportement individuel est le suivant :


i:=1
Pi
k:=1
M(r_ki)=*
Non
Oui
Trouver
TD(r_ki)
TG(r_ki)
Génération des
produits de
termes
Génération des
équations
k:=k+1
Non
k=k_pi
oui
i:=i+1
i=n
non
oui

4.3.2.2. Comportement simultané

Le comportement simultané représente (cf.2.2.2.1.) le produit des règles de processus participant à un réseau de processus. Dans les conditions de l'optimisation du calcul, il faut tenir compte de tous les processus à la fois. Ceci nous a mené vers le schéma suivant :


Calcul du nombre total des équations marqué pour chaque processus
Générations de toutes les combinaisons possibles pour les processus
Génération des combinaisons des équations
Extension du vecteur
Génération des fusions d'états
Ecrire l'équation
non
fin des états
oui
fin des règles

4.3.2.2.1. Nombre total de règles

Dans l'ensemble de règles de chaque processus, on fait le calcul du nombre de règles, qui disposent de la marque :

$$S_{P_k} = \Sigma * \quad k = 1,\ldots,n$$

C'est-à-dire que, si les règles $r_{k,i1}, r_{k,i2},\ldots, r_{k,ij} \in P_k$ sont marquées, alors il y a une correspondance biunivoque :

$$i_j \quad <---> \quad j$$

où i_j est l'ordre de l'apparition de la règle r_{k,i_j} à la spécification du processus P_k et j l'ordre de l'apparition de la règle r_{k,i_j} à l'ensemble de règles qui disposent la marque.

4.3.2.2.2. Génération de toutes les combinaisons possibles pour les processus.

Si on se place d'un point de vue purement récursif , on voit que le comportement simultané représente toutes les combinaisons possibles entre les processus. En accord avec cette idée, nous avons introduit le générateur de toutes les combinaisons possibles pour les processus.

Si $V = (S_{P_{n1}}, S_{P_{n2}},\ldots, S_{P_{n_k}})$ est le vecteur , où $S_{P_{nj}} \neq 0 \quad j=1\ldots,k$,

alors on va générer tous les vecteurs

$$\{V_i\}, \quad i = 2,\ldots,k$$

et i représente le nombre de composants différents de 0 qui appartiennent à V_i.

Exemple

Soit V = (2,2,1,3)

$\{V_2\}$: (2,2,0,0) (2,0,1,0) (2,0,0,3) (0,2,1,0) (0,2,0,3) (0,0,1,3)

$\{V_3\}$: (2,2,1,0) (2,2,0,3) (2,0,1,3) (0,2,1,3)

$\{V_4\}$: (2,2,1,3)

4.3.2.2.3. Génération des combinaisons de règles

Dans un deuxième temps on fait la génération de toutes les combinaisons possibles pour l'ensemble de règles des processus appartenant à V_i $i = 2,\ldots,k$

Le générateur fonctionne d'après le schéma (cf.4.3.2.1.1.) en ayant les composants de V_i comme valeur. Le générateur fait le calcul de l'ensemble des vecteurs : $\{V'_i\}$ $i = 2,\ldots,k$

$\{V_2\}$:	(2,2,0,0)	(2,0,1,0)	(2,0,0,3)	(0,2,1,0)	(0,2,0,3)	(0,0,1,3)
$\{V'_2\}$:	(1,1,0,0)	(1,0,1,0)	(1,0,0,1)	(0,1,1,0)	(0,1,0,1)	(0,0,1,1)
	(1,2,0,0)	(2,0,1,0)	(1,0,0,2)	(0,2,1,0)	(0,1,0,2)	(0,0,1,2)
	(2,1,0,0)		(1,0,0,3)		(0,1,0,3)	(0,0,1,3)
	(2,2,0,0)		(2,0,0,1)		(0,2,0,1)	
			(2,0,0,2)		(0,2,0,2)	
			(2,0,0,3)		(0,2,0,3)	

Remarque

En fait le générateur fonctionne en éliminant les 0 intermédiaires. Par exemple :

(2,2,0,0) -→ (2,2)

(0,2,0,3) -→ (2,3)

pour cette raison nous avons développé le paragraphe suivant .

4.3.2.2.4. Extension du vecteur

A l'issue du paragraphe précédent nous avons obtenu un vecteur qui a, soit autant de composantes que les processus participant au calcul du comportement simultané i = n, soit une combinaison quelconque 2 <i< n L'extension est d'étendre l'ensemble des vecteurs $\{V'_i\}$ à un vecteur $T = (t_1, t_2, \ldots, t_n)$ de cette manière

$$t_l = \begin{cases} i_j & l=n_j \quad j=1,\ldots,n \\ g_l & l \neq n_j \end{cases}$$

où g_l est le nombre de symboles d'état du processus P_l et i_j représente la transformation $j \rightarrow i_j$ qui est l'inverse du 4.3.2.2.1.

4.3.2.2.5. Synchronisation

C'est le même problème que 4.3.2.1.3. La différence réside dans le fait qu'ici nous avons un ensemble pour le TD et TG

4.3.2.3. Les équations de communication

4.3.2.3.1. Choix des équations

Etant donné , l'ensemble des vecteurs qui représente l'ensemble des connecteurs de chaque règle TAB2 et le vecteur représentant l'ensemble des connecteurs appartenant à l'opérateur de connexion TABPORT, on va définir l'ensemble des règles qui participe à la création des règles de communication. Soit $v = (c_1, \ldots, c_k)$ et $v \in$ TAB2, si au moins l'un des c_i TABPORT alors cette règle sera marquée $v = (c_1, \ldots, c_k, *)$
Après cela tout se passe comme aux 4.2.2.2.2. ÷ 4.2.2.2.5.

4.3.2.3.2. Vérification d'un ensemble de règles

Ayant une combinaison quelconque des règles marquées, on va vérifier si cette combinaison représente un ensemble de règles de communications qui échangent les messages entre eux.

Soit R l'une des règles et v = $(C_1,\ldots,C_k,*)$ l'ensemble de connecteurs. L'algorithme de vérification fonctionne d'après le schéma suivant :

```
Pour tout c_i ∈ v   faire
                    si  c_i ∈ TAB2PORT  alors   BOL:=VRAI ;
                    si  c_i ∈ TABPORT
                    alors  si  les connecteurs appartiennent à
                               l'ensemble des règles de communication.
                           alors  activer l'algorithme d'unification
                           sinon  la combinaison ne vaut pas
                    sinon si  BOL = FALSE alors la combinaison ne vaut pas

fin
```

La variable BOL assure le fait que tous les connecteurs de la règle, qui est en train d'être examiné, sont, soit les connecteurs externes, qui appartiennent à l'opérateur d'abstraction, soit ils sont reliés en donnant des connecteurs internes, soit les deux.

4.3.2.3.7. Unification

Les messages qui circulent le long des connecteurs sont décrit par les termes fonctionnels. Soient A la porte de sortie et B la porte d'entrée, $f_a(t_1,\ldots,t_n)$ et $f_b(t'_1,\ldots,t'_n)$ les termes fonctionnels, qui décrivent les messages qui circulent le long des connecteurs A.? et ?.B. Alors , si $f_a = f_b$, pour trouver l'unificateur on résoud le système d'équations [MON 82] suivant :

$$\begin{aligned} t_1 &= t'_1 \\ &\vdots \\ t_n &= t'_n \end{aligned}$$

La solution de ce système d'équations donne l'unificateur ρ tel que :

$$\rho(f_a(t_1,\ldots,t_n)) \quad = \quad \rho(f_b(t'_1,\ldots,t'_n))$$

Puisque nous faisons le calcul global de toutes les communications pour un ensemble de processus, le problème s'élargit à l'unification d'un ensemble de termes fonctionnels $f_1, f_2, \ldots, f_i$ et $f'_1, f'_2, \ldots, f'_i$, c'est-à-dire qu'on va trouver d'après le modèle ci-dessus l'ensemble des unificateurs $\rho_1, \rho_2, \ldots, \rho_i$ tel que

$$\rho_1(f_1) = \rho_1(f'_1), \ldots, \rho_i(f_i) = \rho_i(f'_i)$$

Plus précisément :

$$\left.\begin{array}{l} t_{11} = t'_{11} \\ \vdots \\ t_{1j_1} = t'_{1j_1} \end{array}\right\} \longrightarrow \rho_1$$

$$\left.\begin{array}{l} t_{21} = t'_{21} \\ \vdots \\ t_{2j_2} = t'_{2j_2} \end{array}\right\} \longrightarrow \rho_2$$

$$\left.\begin{array}{l} t_{i1} = t'_{i1} \\ \vdots \\ t_{ij} = t'_{ij_i} \end{array}\right\} \longrightarrow \rho_i$$

Si $\langle w, v \rangle$ est la règle en cours de calcul alors les substitutions

$\rho_1(\ldots(\rho_i(w))\ldots)$ et

$\rho_1(\ldots(\rho_i(v))\ldots)$ auront lieu

V - FONCTIONS GRAPHIQUES DU SPARC

5.1. Introduction

Dans cette partie nous présentons les aspects principaux à travers une description graphique du poste de travail pour lequel ce modèle [JOR 83] a servi de base. Ce langage est un langage de description graphique de systèmes parallèles communicants en termes d'ensemble de tâches communicant par échange de messages. On introduit d'abord la notion de description graphique du processus et une interprétation graphique des opérateurs de construction des réseaux de processus communicants, puis l'expression formelle comme synonyme de ce réseau. Enfin on précisera quelques éléments relatifs à la nature de réalisation expérimentale.

5.2. Langage de description

La façon la plus "naturelle" pour décrire les réseaux de processus communicants est de dessiner des assemblages de "boites" et de "flèches" où les boîtes représentent les processus et les flèches leurs interconnections.

Le processus est défini uniquement par sa forme, ses dimensions et ses couleurs interne et externe, et un connecteur externe est défini uniquement par ses coordonnées et sa couleur. La forme choisie pour les connecteurs externes est le triangle équilatéral. La syntaxe des connecteurs est la suivante :

<connecteur externe d'entrée> ::= <triangle orienté vers l'intérieur de l'image du processus>

<connecteur externe de sortie> ::= <triangle orienté vers l'extérieur de l'image du processus>

<connecteur externe d'entrée-sortie> ::= <connecteur externe d'entrée> <connecteur externe de sortie>

Plus précisément un processus dans représentation graphique est présenté par les informations stockées dans une table de 7 éléments. On y trouve les renseignements suivants attachés à un processus :

Le nom du processus
La forme
La couleur extérieure (du contour)
La couleur intérieure
Les dimensions de la figure
Le pointeur vers le premier connecteur du processus
Le pointeur vers le dernier connecteur du processus

Un connecteur est représenté par les informations suivantes :

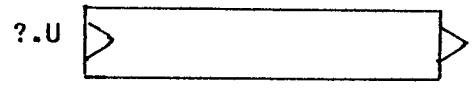
Le nom de la porte
Le type du connecteur (Entrée, Sortie, Entrée-Sortie)
La couleur du connecteur
Les coordonnées relatives du connecteur par rapport à la figure.
L'angle de rotation de concordance
Le code du côté (voir 5.7.1.1.)

Exemple

Description de la pile de capacité 1.

Nom : pile ;
Connect : ?.U,V.? : NAT ;
Etats : Q, R :() ;
Init : Q ;
Règles : Q = ?.U(n) R ; R = V.?(n) Q ;

(1)

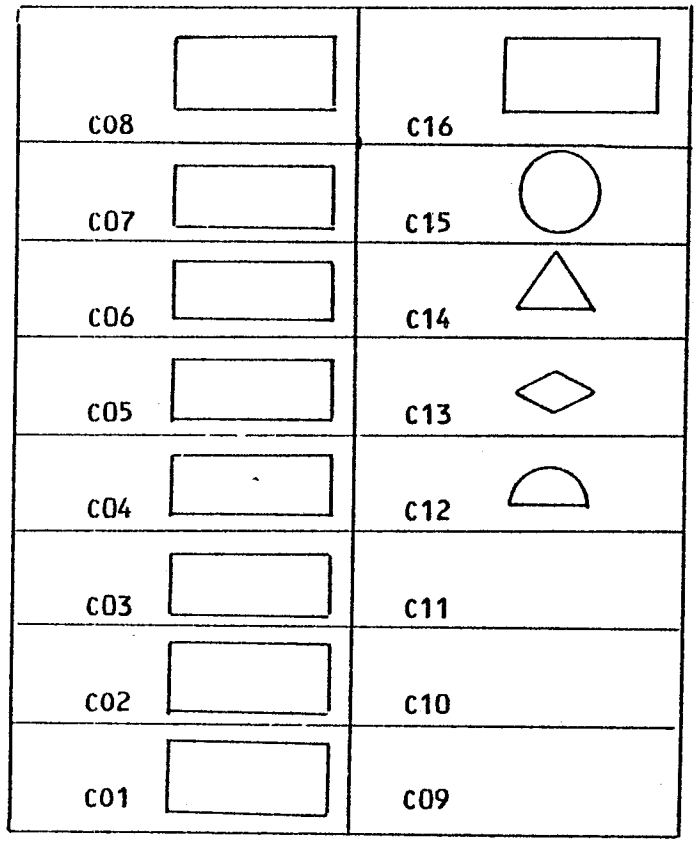


(2)

Dans la représentation ci-dessus (1) et (2) sont deux formes de description d'un processus. Nous avons remplacé toute partie de comportement par un rectangle. La différence entre les deux est que (2) n'exprime que les communications avec un environnement , ce qui nous intéresse pour notre étude.

5.2.1. Le menu de description graphique du processus

La description du processus se matérialise par le menu graphique. Nous avons symbolisé les éléments de la description selon le menu suivant :



Celui-ci est destiné à définir les éléments de la description graphique du processus. 1-8 sont les couleurs, 12-16 les formes de figures. Tous les éléments de description sont définis en mode de dialogue. C'est le système, qui demande à l'utilisateur l'élément de description à définir. C'est l'utilisateur qui doit déplacer le curseur sur l'élément désiré. Les éléments de description sont représentés graphiquement.

5.3. Construction du réseau de processus

Dans cette partie nous allons avoir une vue graphique pour les opérateurs sur les descriptions de processus communicants. Pour construire un réseau, on se donne l'ensemble des opérateurs sur les descriptions

de processus, qui appliquées à des processus ou à des réseaux déjà construits permet d'en obtenir de nouveaux. La première donnée est donc un menu d'opérateurs. Dès qu'on entre dans cet environnement un tel menu est affiché sur l'écran.

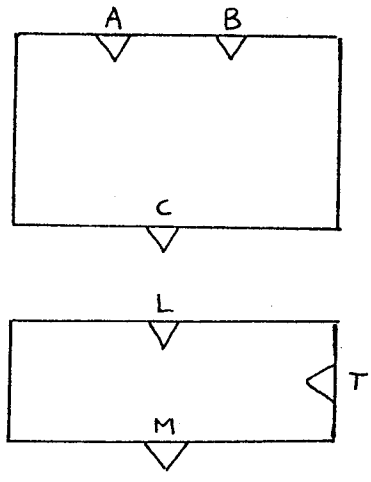
5.3.1. Opérateur d'union

L'opérateur d'union "met ensemble" deux processus ou réseaux de processus et le résultat est une description , qui décrit à la fois le comportement individuel et simultané de ses opérandes. L'union est un opérateur qui permet de construire de nouveaux réseaux à partir des processus ou des réseaux déjà rangés dans la base de données.

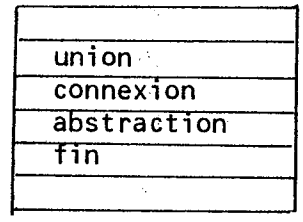
5.3.1.1. Menu de processus

Il s'agit d'un menu graphique et alphanumérique à la fois. Les processus ou les réseaux sont représentés dans ce menu avec ses noms et ses descriptions graphiques réduites. Tout l'ensemble de processus rangé dans la base de données est affiché sur l'écran. C'est un menu paginé. La quantité de pages dépend du nombre des processus existants. Ainsi, nous pouvons choisir l'opérande désiré sur l'ensemble des processus affichés.

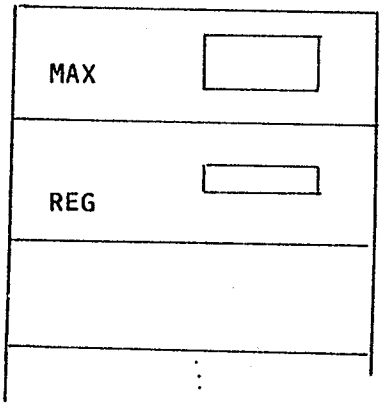
Exemple



Opérateurs



Opérandes

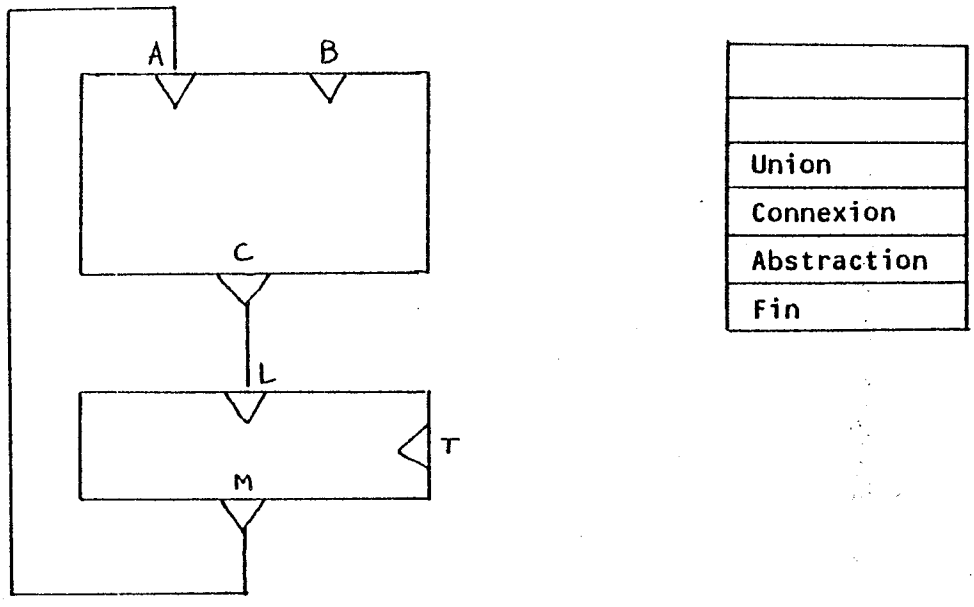


On indique l'opérateur d'union et les processus MAX et REG. Ce réseau est équivalent à MAX//REG.

5.3.2. Opérateur de connexion

L'opérateur de connexion décrit les échanges entre les processus composants d'un réseau. Tous les processus impliqués dans l'échange doivent être dans un état où l'échange est possible pour que celui-ci puisse être effectué. Une communication entre A.? et ?.B est possible uniquement quand les communications entre A.? et ?.B sont simultanément possibles. Etant donné un ensemble de processus formant un réseau, l'opérateur de connexion permet, après avoir choisi les connecteurs externes, d'obtenir la communication entre les processus communicants. Ceci est représenté comme une file entre les processus émetteurs et les processus récepteurs.

Exemple



Ayant indiqué (deux fois) l'opérateur de connexion et les portes C et L pour la première fois et M et A pour la deuxième fois nous obtenons le R.C. ci-dessus et qui est équivalent :

MAX//REG + C.L + M.A

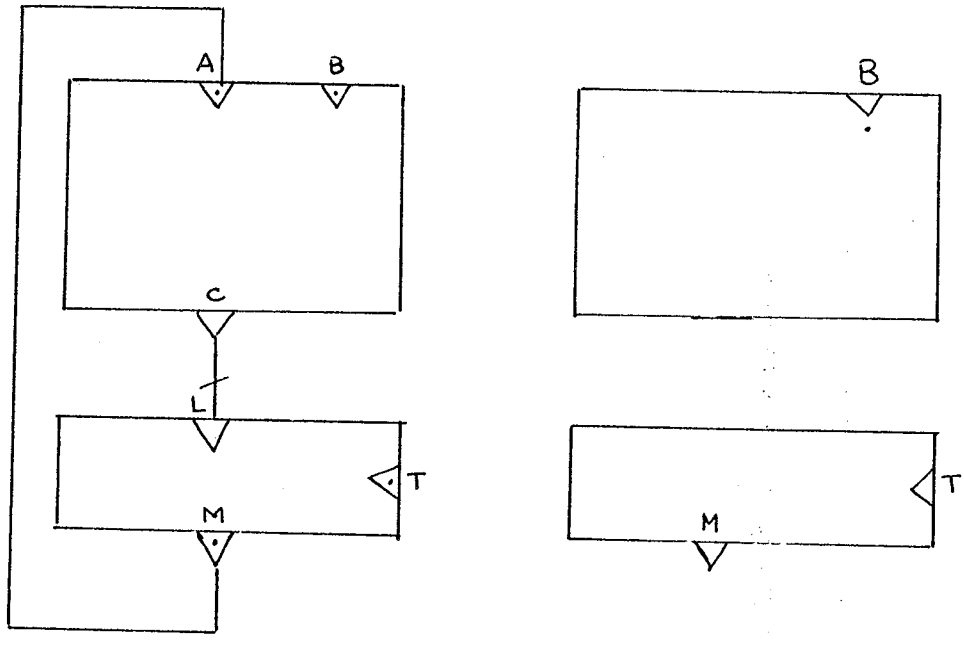
5.3.3. Opérateur d'abstraction

L'opérateur d'abstraction a pour opérande l'ensemble des connecteurs $K = K_{in} \cup K_{ext}$. Ayant un réseau de processus, on peut utiliser l'opérateur d'abstraction de cacher une partie des connecteurs, alors que préserve l'accès sur le sous-ensemble sélectionné $\{k_1, k_2, \ldots, k_p\}$. Certains de ces connecteurs sont marqués, pour différencier les connecteurs représentant les communications des connecteurs non utilisés.

Il est effectué de la façon suivante :

- On va donner une marque à certains connecteurs (point pour les connecteurs externes, trait pour les connecteurs internes)
- Tous les connecteurs non marqués sont supprimés
- L'ensemble des connecteurs marqués reste visible

Exemple



L'ensemble des connecteurs {?.A,?.B,C.?,?.L,?.T;M.?,C.L,M.A}

Le sous ensemble {?.B,M.?,?.T,C.L}

Le réseau est équivalent à l'expression

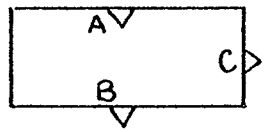
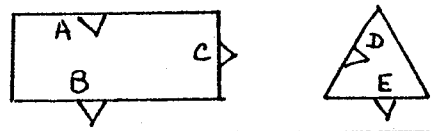
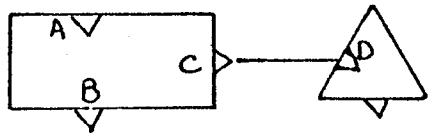
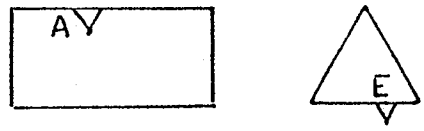
MAX//REG + C.L + M.A [?.B,M.?,?.T,C.L]

5.4. L'expression formelle comme synonyme de réseau

Dans cette partie nous avons présenté la technique de mise en oeuvre dans le système SPARC pour l'obtention d'un réseau de processus décrit en langage graphique. Ceci est mené en deux phases :
-d'abord la création de l'image graphique du réseau
-ensuite la construction d'une expression formée des noms de processus, les connecteurs et les trois opérateurs.

Le but de ce travail est d'obtenir un modèle sur lequel nous pouvons construire le réseau correspondant à partir du langage formel de description de processus. Au cours de la génération du réseau les noms des connecteurs sont renommés si cela est nécessaire (cf. 5.5)

On donne ici sous forme d'exemple les règles utilisées pour la traduction des instructions :

Op. graphique	Représentation graphique	Expression
		P1;
Union		P1//P2 ;
Connexion		P1//P2+C.D
Abstraction		P1//P2+C.D[?.A,E.?]

Les règles de transformations sont donc :

- les identificateurs apparaissent dans le même ordre dans l'expression que les objets correspondants dans leur identification graphique
- les opérateurs apparaissent dans l'ordre d'utilisation.

L'algorithme peut donc être décrit par une analyse des opérations graphiques :

- en créant un réseau de processus il est obligatoire de commencer par l'opérateur de l'union
- nous désignons le premier processus, qui correspond à l'écriture du nom dans l'expression
- en présence d'un opérateur on écrit l'opérateur
- en présence d'une désignation on écrit l'identificateur de l'objet désigné,

pour les connecteurs on fait intervenir la fonction de renommage,

- la fin d'une expression est déterminée par le point virgule.

L'expression créée doit être analysée de gauche à droite. Nous avons évité la nécessité du paranthésage, ce qui est compatible avec la syntaxe ci-dessus tout en évitant l'ambiguïté.

5.4. Renommage

Au chapitre précédent (cf.4.1) nous avons présenté le problème du renommage des portes et la manière d'agir de l'utilisateur. Notre propos consiste à trouver l'algorithme, qui est un moyen effectif d'obtenir l'identificateur de renommage lorsque l'on connait l'identificateur précédent. Les deux fonctions impliquées pour la génération des identificateurs sont décrites dans l'évaluation de l'expression. La génération des identificateurs repose sur l'évaluation (pour chaque identificateur) de la fonction

Nom(Identification de renommage)

ce qui fournit l'identificateur de base et de la fonction de renommage

$F(\Lambda_{\Sigma}, NAT) \rightarrow \Lambda$ où

$$F(Id,ordre) = \begin{cases} Id & \text{si} \quad Id \notin \Lambda_I \\ Id < ordre & \text{si} \quad Id \in \Lambda_I \end{cases}$$

5.4.1.1. L'algorithme de la génération des identificateurs

Nous avons conçu un générateur d'identificateurs. Compte tenu de ce logiciel l'expression formelle sera tout à fait en concordance avec le principe de renommage.

Les éléments principaux sont les suivants :

- pas de renommage quand les connecteurs ont des noms différents
- pas de renommage quand les connecteurs ont les mêmes noms mais transportent des messages de types différents, c'est le type qui fait la distinction entre les connecteurs.
- le problème de renommage se pose quand les deux portes ont le même type et le même nom. La fonction F(Id,ordre) est activée et l'identificateur de renommage est créé. La réalisation se fait d'après le schéma suivant :

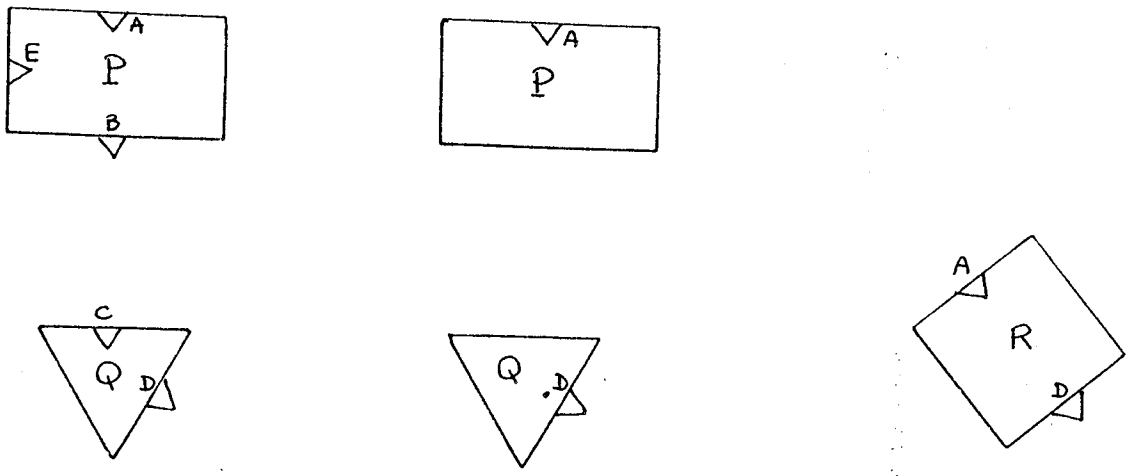
```
Pour tout Id ∈ Λ_Σ
    si Id ∈ Λ_I

        Alors début
                    Calcul (Id,ordre)
                    Idren := Id < ordre
              fin

        Sinon    pas de renommage
fin
```

5.5. Description graphique du réseau des processus

Dès que l'utilisateur a fini la conception d'un réseau il va le considérer à la fois comme un processus élémentaire et comme un réseau de processus, c'est-à-dire que le résultat final a deux formes de représentation :

- une forme égale à ce que nous avons décrit dans 5.2.
- une forme de représentation graphique du réseau

L'exemple suivant met en évidence les deux aspects, qu'on a évoqué ci-dessus :



Les processus Le Réseau R Le Réseau R

5.5.1. Description du réseau comme processus élémentaire

Nous avons donné en 5.2. la définition de la description graphique du processus élémentaire. Dans le cas des réseaux de processus pour opérer de même, nous avons choisi une représentation identique à

condition qu'il y ait une correspondance bijective entre les connecteurs externes du réseau et ceux de sa représentation sous forme de processus élémentaire. On préserve dans ce cas les informations nécessaires à la communication du réseau avec l'extérieur.

Nous avons réalisé le logiciel correspondant, qui montre la correspondance au moment de la description.
Deux avantages sont alors à considérer :
- nous avons le moyen d'utiliser le réseau lors de la création de réseaux plus complexes ;
- nous avons la possibilité de connaître la structure interne du réseau de processus (cf. 5.5.3).

5.5.2. Description généralisée du réseau

On se propose d'introduire une expression graphique, qui puisse produire l'image graphique du réseau. Ceci est mené en parallèle avec 5.4. L'expression est obtenue à partir des noms de processus et de connecteurs et des opérateurs de construction du réseau. Cependant il faut ajouter qu'il existe en plus quelques informations graphiques pour différents éléments du dessin. Autrement dit cette expression graphique est la description graphique du réseau. Les renseignements sont plus riches que la description comme processus élémentaire.

Tout d'abord on connaît tous les connecteurs externes et internes, ensuite on perçoit bien la structure interne du processus. Dans ce but, nous avons associé aux réseaux de processus la notion d'ouverture.

5.5.3. Ouverture

Ce processus permet à l'utilisateur de connaître le réseau sous une forme de processus élémentaire. C'est-à-dire le résultat est la spécification du réseau associé à l'expression graphique passée en paramètre de cette procédure. En fait c'est une sorte d'évaluation

d'expression, mais l'expression n'est plus formelle. Le principe d'optimisation tel qu'il est décrit en 4.2. est mis en oeuvre dans cette partie. Nous jugeons plus raisonnable de ne pas le décrire dans cette partie.
Le processus d'ouverture est décrit ici de façon algorithmique, par trois fonctions, qui sont appelées autant de fois que les opérateurs correspondants apparaissent dans l'expression graphique de description. Ce sont les fonctions INVPROC, INVCONN, INVAB. Nous ne faisons pas l'affichage immédiat de l'image, mais nous préservons les informations en analysant l'expression jusqu'au bout. L'objet de l'analyse est celui-ci :

La fonction INVAB crée une image totale de chaque processus en gardant tous les éléments de description.
La fonction INVCONN est activée lorsqu'un connecteur interne apparaît.
La fonction INVAB fait intervenir l'opérateur d'abstraction sur les deux premiers. A ce moment le processus résultat est affiché.

5.6. Quelques techniques graphiques

Nous citerons ici deux techniques de base employées pour la description du processus et la construction du réseau. On distingue en fait deux classes de techniques : la définition des différents éléments et la désignation d'un élément dans un ensemble donné.

5.6.1. Menu

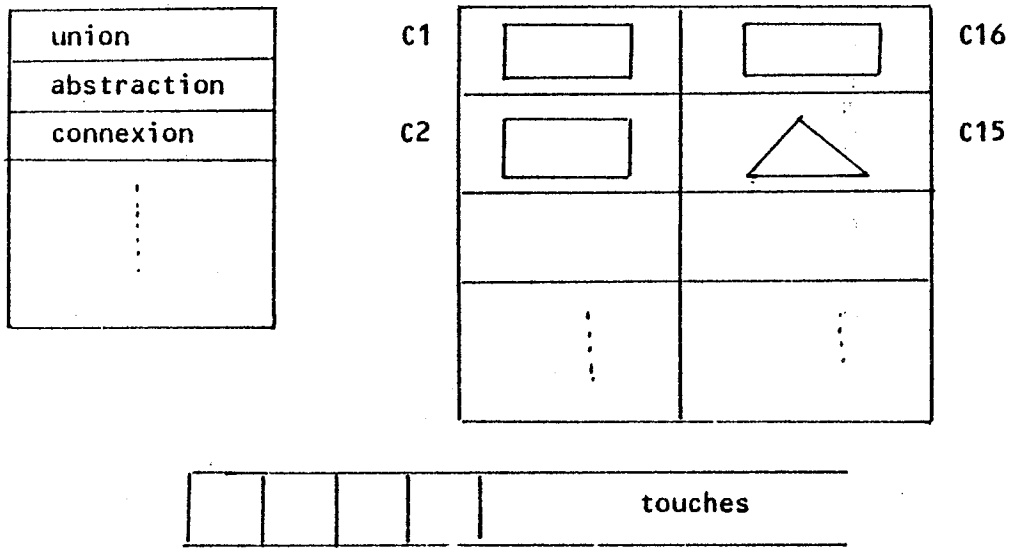
Nous nommerons menu [MAR 77] une primitive d'interaction dans laquelle le système présente sur l'écran :

- une liste de noms d'actions
- une liste d'objets

L'utilisateur désigne :

- l'action qu'il désire exécuter
- un ensemble d'objets parmi ceux présentés par le système.

Pour chaque nom d'action et objet de la liste, nous avons mis en correspondance une touche de fonction. Cette forme éclaircit l'utilisation de touches. L'utilisateur trouve sur le menu le nom d'action ou l'objet qu'il veut désigner et le numéro de la touche correspondante. RAMTEK possède 16 touches mais 32 possibilités.



5.6.2. Les menus graphiques

La liste des objets peut être représentée de deux manières différentes :

- présentation des noms affectés aux objets
- présentation des objets sous leur forme graphique

Nous avons construit le menu de deuxième type ou le menu qui a deux formes à la fois. Ceci dépend de l'environnement dans lequel se trouve le système.

Les menus graphiques sont construits en deux formes :

- présentation des objets sous leur forme graphique et par leur nom. C'est un menu variable, qui change dès que l'utilisateur a consulté une page. Ce sont les éléments graphiques complexes. Nous réalisons

cette forme de représentation en employant le principe de réduction de l'image.

- présentation des objets sous leur forme graphique. Cela se fait pour les éléments graphiques qui sont fixés. Nous utiliserons cette sorte de menu pour les éléments graphiques de description de processus.

5.6.2. Identification des connecteurs

La question principale est de savoir comment, à partir d'une information affichée, on peut retrouver la figure représentée. Il faudra donc identifier un élément sur l'écran et lui associer son homologue dans la structure de données concernée. Pour cela on conserve en mémoire les coordonnées du point affiché, permettant au logiciel de base de faire les transformations inverses et d'identifier la figure.

Les éléments que nous avons besoin d'identifier sont les connecteurs, soit externes, soit internes. Le principe de fonctionnement de l'algorithme est le suivant :

```
Debut
        Identification du point (x,y)
        Si  (x,y)  désigne un connecteur externe
            alors   succès
            Sinon
Debut
        Identification du point (x1,y1)
        Si   {(x,y),(x1,y1)}  désigne le connecteur interne
            alors   succès
            sinon   error
        fin
Fin
```

L'identification des connecteurs internes est réalisée par deux points en créant une petite ligne coupant le connecteur, qu'on veut identifier. Nous avons réalisé l'identification en utilisant la méthode MINIMAX [GIL 78] :

Module de l'algorithme :

```
COUP := FAUX ;  I := 1 ;
TANT QUE  COUP = FAUX  FAIRE

Debut
   Utilisation de la méthode MINIMAX pour le Iime segement
   Si les segments se coupent
   alors  COUP := VRAI
   I := I + 1
Fin
```

L'algorithme parcourt la liste des connecteurs, jusqu'au moment où on trouve le connecteur identifié.

5.7. Logiciel d'application

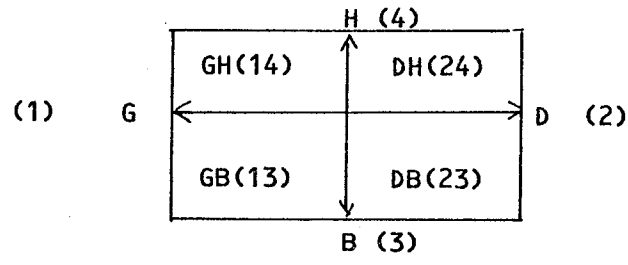
L'objet de cette partie est de donner la méthode concrète de description de processus et de création d'un réseau, décrite en termes d'éléments graphiques de description. Ce modèle permet au concepteur du système de décrire et d'organiser les éléments de son application.

Etant choisi, le modèle descriptif graphique, chaque opérateur de construction du réseau du processus a son interprétation graphique "naturelle".

5.7.1. Description

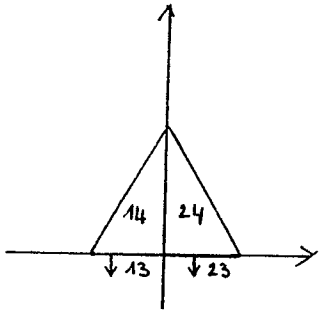
5.7.1.1. Codage des figures

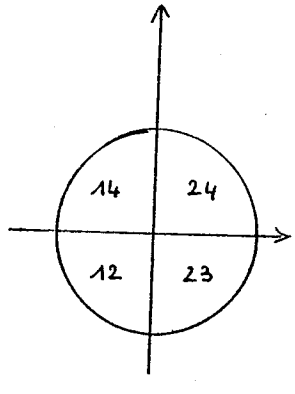
Nous proposons pour les quarts du plan la numérotation de la figure :

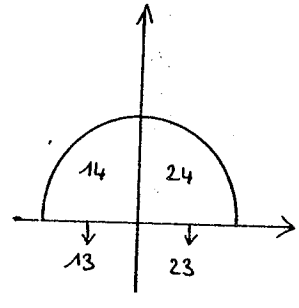


Le quart 2 4 est défini par l'ensemble des points (X,Y) tels que : $x \geqslant 0$ et $y \geqslant 0$. On identifie d'une manière analogue les quarts 2 3, 1 3, 1 4.

La procédure COTE, qui calcule pour chaque figure les relations spatiales avec la position de la porte marquée par curseur, donne les renseignements suivants :

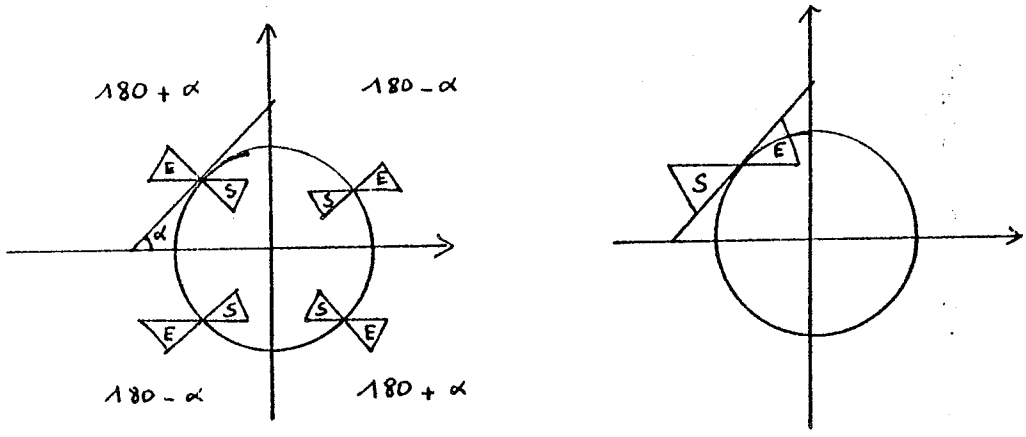






5.7.1.2. Définition des portes et calcul de l'angle de concordance

Nous définissons les portes sur chaque processus d'une manière standard. Le principe de l'algorithme repose sur l'idée de rotation des triangles jusqu'à la concordance avec la tangente. On va illustrer l'idée avec un cercle en donnant l'idée générale. Au point de départ on aura les relations suivantes :



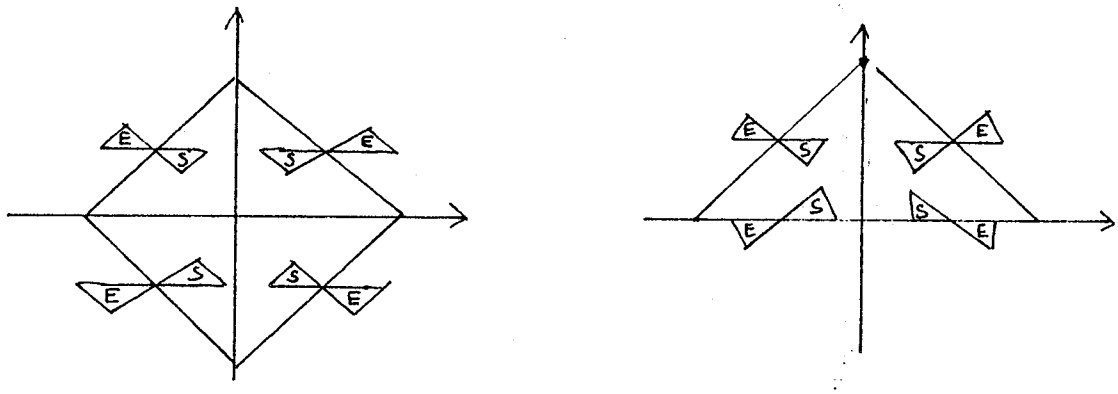
La séquence des opérations est déterminée par l'algorithme suivant :

- définition des triangles E (entrée) et S (sortie) en concordance avec la codification
- calcul de l'angle de la tangente pour le point de référence R

$$\alpha = \text{arctg} \left| \frac{X_R}{Y_R} \right|$$

- rotation Rot = 180 ± α autour du point de référence

En appliquant cet algorithme nous avons la situation suivante :



Les intérêts de cette méthode de représentation sont :

- l'identification des portes est définie uniquement pour tous les processus.
- cette méthode est moins coûteuse que la méthode fondée sur le principe de calcul mathématique.

L'algorithme de la définition des portes se divise en 3 parties :

- désignation du point de référence et son passage sur contour
- détermination du quart
- calcul de l'angle de la rotation

5.7.1.3. Description du processus

Lorsque l'utilisateur désire créer un nouveau processus le menu de description du processus est affiché. Après l'introduction du nom, le système fait appel au fichier de description de processus.

La description du processus s'effectue en deux temps :

1°- le choix des éléments graphiques de description (forme, couleur, dimensions) et leur affichage.

2°- le choix des éléments graphiques des portes et leur affichage symbolique.

L'affichage de processus se fait pour deux raisons :

- l'utilisateur peut consulter ce qu'il a décidé
- il est nécessaire de pouvoir choisir la position des portes.

Après les opérations ci-dessus, un nouveau processus est rangé dans la base de données.

5.7.2. Logiciel d'application de la création du réseau

Lorsque l'utilisateur désire créer un nouveau réseau de processus le système fait appel successivement :

- au menu des opérateurs
- aux fichiers de description de processus

5.7.2.1. Union

A. Consultation des catalogues

Cette fonction, concernant le catalogue de l'image du processus vise un double objetif :

- permettre la consultation
- permettre la sélection de l'un des objets qu'il contient

B. Edition

L'objet désigné par l'utilisateur est communiqué à un sous programme d'édition qui permet de le visualiser, de faire une translation, ou une rotation. Chaque fois que l'utilisateur désigne l'opération d'union sur le menu, il doit choisit un processus pour le visualiser. L'édition s'effectue en deux temps :

- l'édition du processus sans connecteurs
- l'édition des connecteurs en concordance avec la première édition.

5.7.2.2. Connexion

A. Choix des connecteurs externes

Ce choix est également effectué par désignation directe de l'objet sur l'écran, à l'aide du dispositif de désignation disponible et notamment du réticule. Par sa définition un connecteur interne relie un connecteur externe de sortie avec un connecteur externe d'entrée.

B. Edition

Le problème d'édition est étroitement lié au problème de liaison de deux portes à la condition qu'on doive respecter la syntaxe du connecteur interne

Cette forme de représentation signifie que les connecteurs A.? et ?.B échangent les messages entre eux. L'algorithme fonctionne selon le schéma suivant :

- désigner la première porte (sortie)
- désigner la deuxième porte (entrée)
- tracer le connecteur en mode dialogue

5.7.2.3. Abstraction

A. Choix des connecteurs

Ce choix est également effectué par désignation directe de l'objet sur l'écran. Alors qu'avec l'opération de connexion nous désignons seulement les connecteurs externes, avec l'opérateur d'abstraction nous désignons à la fois les connecteurs externes et internes.

B. Edition

Le problème d'édition est lié étroitement au processus d'effacement. L'algorithme d'édition fonctionne selon le schéma suivant :

- désignation des connecteurs qui appartiennent au sous-ensemble K1 (K-connecteurs d'abstraction)
- effacement des connecteurs qui appartiennent au sous-ensemble K2 = K - K1

VI - ANNEXE : DEROULEMENT D'UNE SESSION

DEROULEMENT D'UNE SESSION

1. Entrée dans le système

L'entrée dans le système se fait par la commande :

RUN MAIN

Sous le systèmes les commandes possibles sont :

DEFINITION
CONSTRUCTION
CONSULTATION
CONTEXTE *

Au début il est obligatoire de choisir la commande contexte (*), et le système fait afficher sur l'écran le message :

DONNEZ LE NUMERO DU CONTEXTE (DE 00 A 99) * *

et le message suivant est affiché :

CONTEXTE UTILISE POUR LA PREMIERE FOIS [Y/N]

la réponse est : N, si le contexte existe déjà
Y, si on initialise un nouveau contexte.

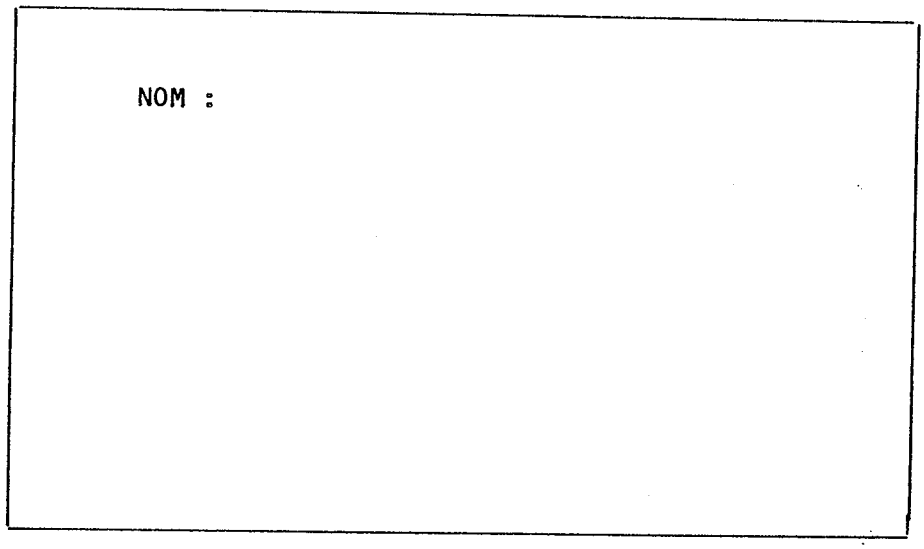
Les commandes

DEFINITION *
CONSTRUCTION
CONSULTATION

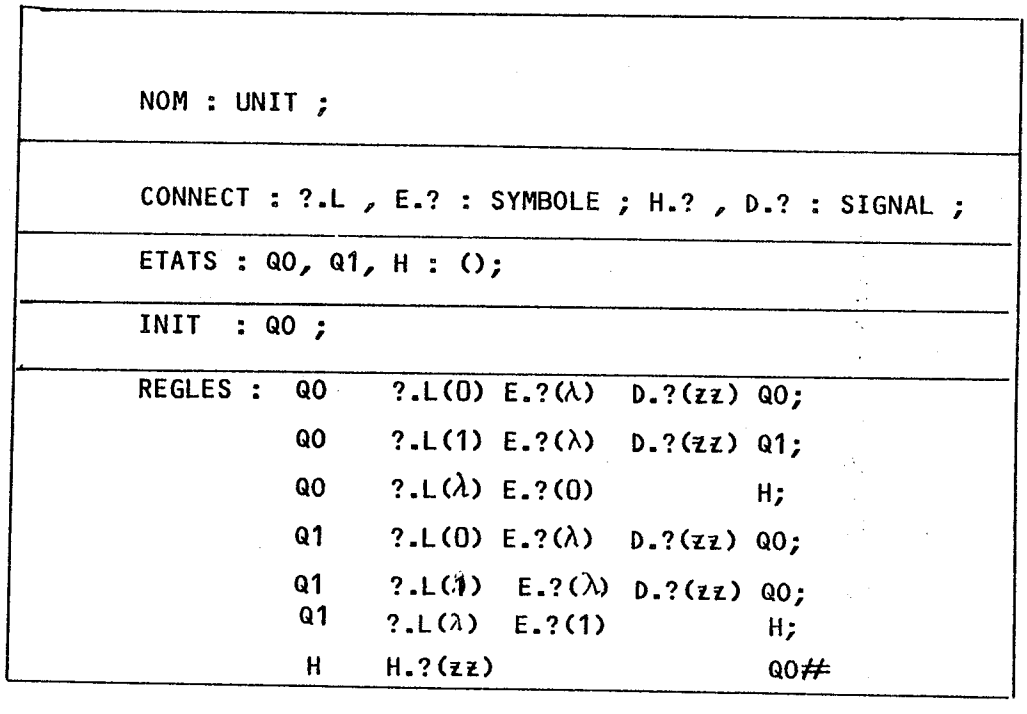
sont toujours sur l'écran.

2. Définition

Après avoir choisi la commande DEFINITION le formulaire suivant sera affiché :



Après avoir écrit le nom du processus, le système crée automatiquement les fenêtres CONNECTEURS, ETATS, INIT, REGLES. Pour ceci il faut respecter la syntaxe de spécification. Par exemple l'unité de contrôle pour la machine de Turing est [PER 84]



Le symbole # signifie la fin de la spécification du processus, ensuite le menu se transforme :

DESCRIPTION GRAPHIQUE

OUI *

NON

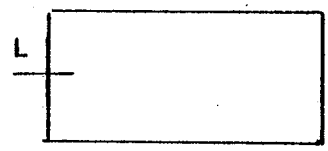
Choisissant OUI, le menu de description graphique est affiché et la description se fait en mode de dialogue .

DONNEZ LA FORME DU UNIT
L'utilisateur appuie sur CTRCL 16 (rectangle)
DONNEZ LES DIMENSIONS
L'utilisateur écrit 80 40
COULEUR EXTERNE
L'utilisateur appuie sur CTRCL 08 (blanc)
COULEUR INTERNE
L'utilisateur appuie sur CTRL 05 (bleu)

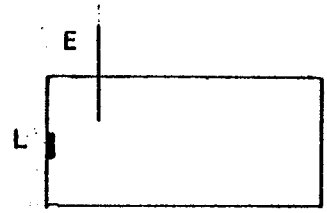
A ce moment le rectangle est affiché avec les caractéristiques ci-dessus

C08 BLANC	C16
C07 BLEU CIEL	C15
C06 VIOLET	C14
C05 BLEU	C13
C04 JAUNE	C12
C03 VERT	C11
C02 ROUGE	C10
C01 NOIRE	C09

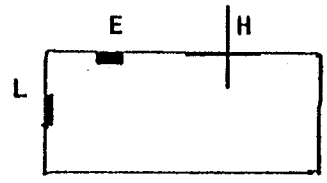
DONNEZ LA POSITION DE LA PORTE L
UTILISATEUR DEPLACE LE CURSEUR SUR UN POINT DU CONTOUR
DONNEZ LA COULEUR DE LA PORTE
CTRL



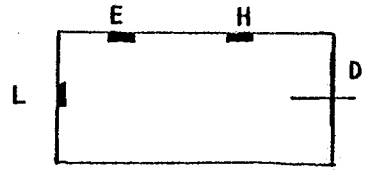
DONNEZ LA POSITION DE LA PORTE E
UTILISATEUR DEPLACE LE CURSEUR SUR UN POINT DU CONTOUR
DONNEZ LA COULEUR DE LA PORTE
CTRL



DONNEZ LA POSITION DE LA PORTE H
UTILISATEUR DEPLACE LE CURSEUR SUR UN POINT DU CONTOUR
DONNEZ LA COULEUR DE LA PORTE
CTRL



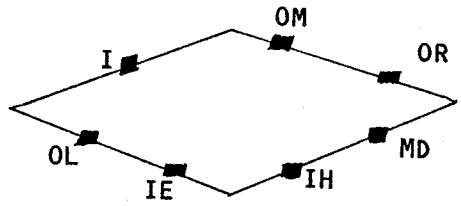
DONNEZ LA POSITION DE LA PORTE D
UTILISATEUR DEPLACE LE CURSEUR SUR POINT DU CONTOUR
DONNEZ LA COULEUR DE LA PORTE
CTRL



Choisissant pour la deuxième fois DEFINITION pour le processus BAND , nous aurons :

NOM : BAND ;
CONNECT : ?.I , ON.? , OR.? : SUIT DE SYMBOLES, OL.? , ?.IE : SYMBOLE , ?.IH , ?.MD : SIGNAL ;
ETATS : I :(), A : SUITE DE SYMBOLES SUITE DE SYMBOLES ;
INIT : I ;
I ?.I(x) A(λ,x) A(Y,CONS(x,b)) OL.?(b) ?.IE(c) A(y,CONS(x,c)); A(Y,λ) OL.?(b) ?.IE(c) A(y,CONS(λ,c)); A(y,CONS(x,b) OL.?(b) ?.IE(c) ?.MO(zz) A(CONS(y,c),x); A(y,λ) OL.?(b) ?.IE(c) ?.MO(zz) A(CONS(y,c),x); A(y,x) ?.IH(zz) OR.?(y) ON.?(x) I ≠≠

et la description graphique :



Après avoir terminé la description les commandes suivantes sont affichées :

DEFINITION
CONSTRUCTION
CONSULTATION *
CONTEXTE
ANALYSE
VALIDATION EVALUATION FIN DE SESSION

3. Construction

Choisissant la commande CONSTRUCTION le menu suivant est affiché

CONSTRUCTION GRAPHIQUE

OUI

NON

Choisissant NON l'utilisateur fait rentrer l'expression à travers le clavier alphanumérique d'après la syntaxe

NOM DU PROCESSUS : Expression ;

Exemple : SYS : MAX//REG + M.B [?.A,M.?]

Choisissant OUI le menu d'opérateurs sur les processus est affiché avec le message :

CHOISISSEZ L'OPERATEUR

L'utilisateur met le curseur sur l'opérateur d'union 04

Le menu de processus est affiché

CHOISISSEZ LE PROCESSUS SINON METTEZ LE COURSEUR SUR ORIGINE

L'utilisateur met le curseur sur UNIT

CTRL 01

DONNEZ LE POINT D'AFFICHAGE

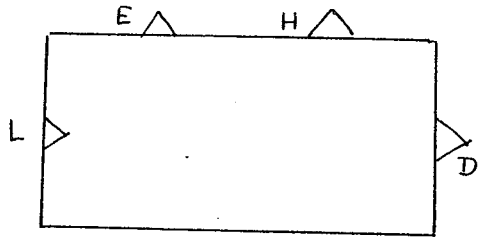
L'utilisateur déplace le curseur sur un point D'ESPACE DE TRAVAIL

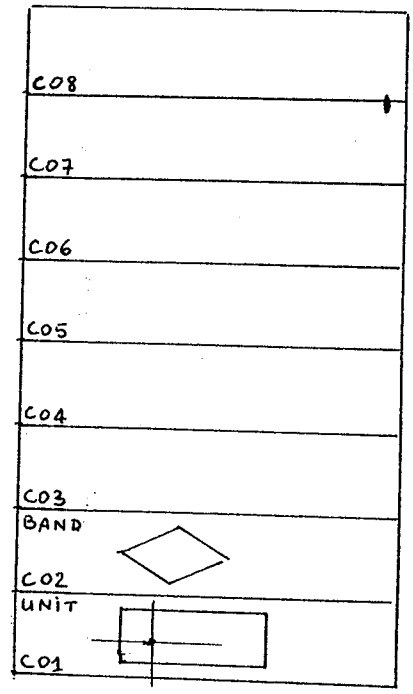
DONNEZ L'ANGLE DE LA ROTATION

L'utilisateur tape sur la valeur 0 (par exemple)

A ce moment le processus UNIT est affiché.

UNION *
CONNEXION
ABSTRACTION
FIN





CHOISISSEZ LE PROCESSUS SINON METTEZ LE CURSEUR
SUR ORIGINE

L'utilisateur met le curseur sur BAND

CTRL 02

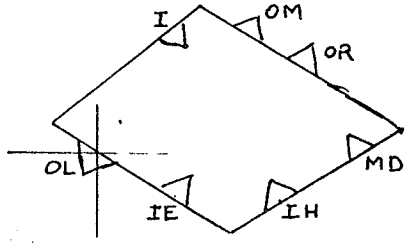
DONNEZ LE POINT D'AFFICHAGE

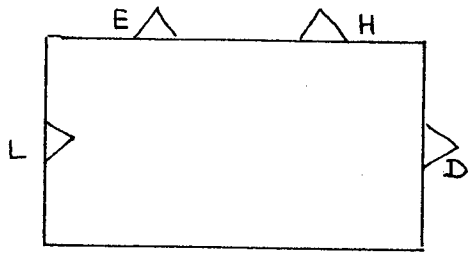
L'utilisateur déplace le curseur sur un point
D'ESPACE DE TRAVAIL

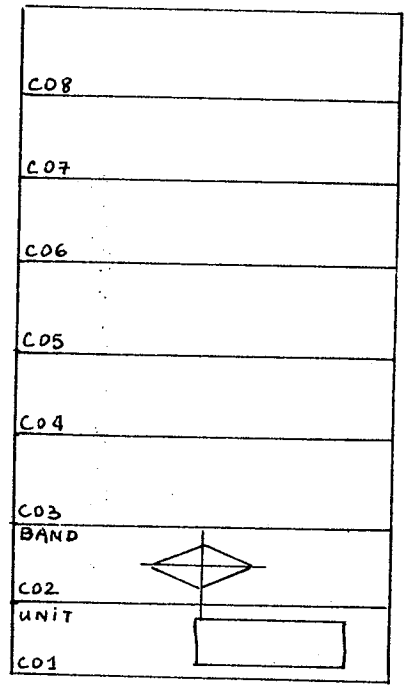
DONNEZ L'ANGLE DE ROTATION

L'utilisateur tape la valeur 0

A ce moment le processus BAND est affiché







Le menu des opérations est toujours celui, qui contient les opérateurs.

Après l'affichage du processus BAND le message suivant s'affiche

CHOISISSEZ LE PROCESSUS SINON METTEZ LE CURSEUR SUR ORIGINE

L'utilisateur appuie sur la touche

HOME

CHOISISSEZ L'OPERATEUR SUIVANT

L'utilisateur met le curseur sur l'opérateur de connexion

03

UNION
CONNEXION *
ABSTRACTION
FIN

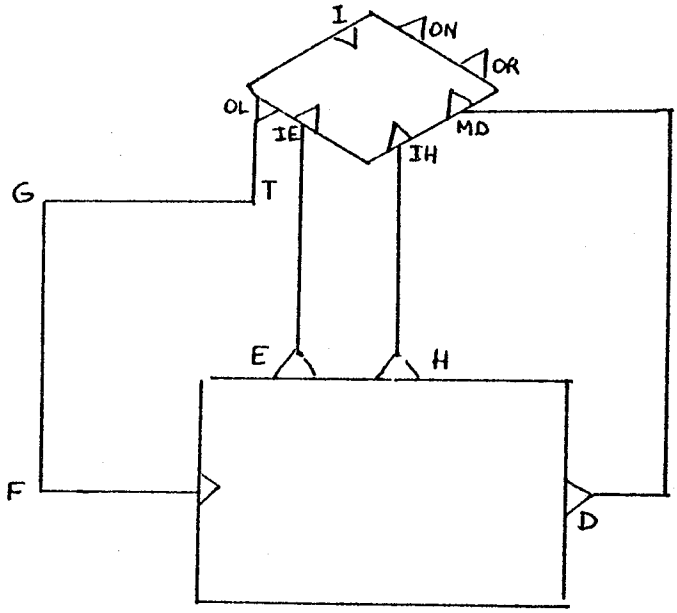
Le message suivant est affiché

METTEZ LE CURSEUR SUR LA PREMIERE PORTE (SORTIE)

L'utilisateur met le curseur sur la première porte OL

METTEZ LE CURSEUR SUR LA DEUXIEME PORTE (ENTREE)

L'utilisateur met le curseur sur la porte L.



CHOISISSEZ LE POINT SUIVANT

L'utilisateur met le curseur sur le point F

CHOISISSEZ LE POINT SUIVANT

L'utilisateur met le curseur sur le point G.

CHOISISSEZ LE POINT SUIVANT

L'utilisateur met le curseur sur le point T

CHOISISSEZ LE POINT SUIVANT

L'utilisateur met le curseur sur la porte OL

Le connecteur interne OL.L s'est fait. Le message suivant est affiché

CHOISISSEZ L'OPERATEUR SUIVANT

Choisissant 4 fois l'opérateur de connexion l'utilisateur construit 4 connecteurs internes OL.L, E.IE, H.IH, D.MD et il revient sur le message

CHOISISSEZ L'OPERATEUR SUIVANT

L'utilisateur met le curseur sur l'opérateur d'abstraction 02

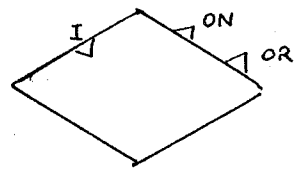
METTEZ LE CURSEUR SUR LE CONNECTEUR EXTERNE OU LE PREMIER POINT DU CONNECTEUR INTERNE

L'utilisateur met le curseur sur le connecteur externe ?.I

Choisissant 3 fois l'opérateur d'abstraction et les connecteurs externes ?.I , OR.? et ON.? le message suivant sera affiché.

CHOISISSEZ L'OPERATEUR SUIVANT

L'utilisateur met le curseur sur l'opérateur FIN et le réseau graphique suivant est affiché :



L'expression compilée est

BAND//UNIT + OL.L + E.IE + H.IH + D.MD [?.I , ON.? , OR.?]

Le menu des opérations revient à son état précédent.

DEFINITION
CONSTRUCTION
CONSULTATION
CONTEXTE
ANALYSE
VALIDATION
RANGEMENT *
EVALUATION
FIN DE SESSION

4. Rangement

Choisissant la commande RANGEMENT le message suivant est affiché :

VOULEZ VOUS RANGEZ L'EXPRESSION FORMELLE [Y/N]

Appuyant la touche Y : le réseau graphique est rangé dans la base de données enfin le message a la forme :

VOULEZ VOUS RANGEZ LE RESEAU COMME UN PROCESSUS ELEMENTAIRE [Y/N]

Appuyant la touche Y; tout se passe comme la commande DEFINITION
Le menu des opérations est toujours :

DEFINITION
CONSTRUCTION
CONSULTATION
CONTEXTE
ANALYSE
VALIDATION
RANGEMENT
EVALUATION *
FIN DE SESSION

5. Evaluation

Après avoir choisi la commande EVALUATION, le système demande à l'utilisateur

DONNEZ LE NOM DE L'EXPRESSION

L'utilisateur fait rentrer par le clavier le nom de l'expression p.e. BUN. Après le calcul, le résultat est affiché.

NOM : BUN

CONNECT : ?.I : SUITE DE SYMBOLES,
ON.? : SUITE DE SYMBOLES,
OR.? : SUITE DE SYMBOLES ;

ETATS : A-Q0 : SUITE DE SYMBOLES SUITE DE SYMBOLES,
A-Q1 : SUITE DE SYMBOLES SUITE DE SYMBOLES,
I-Q0 : (),
A-H : SUITE DE SYMBOLES SUITE DE SYMBOLES ;

INIT : I-Q0 ;

REGLES :

I-Q0	$\Longrightarrow$	?.I(x)	A-Q0(λ,x);
I-Q1	$\Longrightarrow$	?.I(x)	A-Q1(λ,x)
I-H	$\Longrightarrow$	?.I(x)	A-H(λ,x);
A-Q0(y,CONS(x,b))	$\Longrightarrow$	τ	A-H(y,CONS(x,0));
A-Q1(y,CONS(x,b))	$\Longrightarrow$	τ	A-H(y,CONS(x,1));
A-Q0(y,λ)	$\Longrightarrow$	τ	A-H(y,CONS(λ,0));
A-Q1(y,λ)	$\Longrightarrow$	τ	A-H(y,CONS(λ,1));
A-Q0(y,CONS(x,0))	$\Longrightarrow$	τ	A-Q1(CONS(y,b),x);
A-Q0(y,CONS(x,1))	$\Longrightarrow$	τ	A-Q1(CONS(y,b),x);
A-Q1(y,CONS(x,0))	$\Longrightarrow$	τ	A-Q1(CONS(y,b),x);
A-Q1(y,CONS(x,1))	$\Longrightarrow$	τ	A-Q0(CONS(y,b),x);
A-H(y,x)	$\Longrightarrow$	OR.?(y) ON.?(x)	I-Q0;

Après l'affichage le message suivant s'affiche :

VOULEZ VOUS RANGER LE RESEAU [Y/N]

Appuyant sur la touche Y : le réseau est rangé dans la base de données
Appuaynt sur la touche N : le réseau n'est pas rangé dans la base de données.

6. Consultation

Après avoir choisi la commande CONSULTATION, le système demande à l'utilisateur :

VOULEZ VOUS LA LISTE DE PROCESSUS [Y/N]

Appuyant sur la touche Y la liste de processus est affichée.

VOULEZ VOUS LA LISTE DE RESEAUX [Y/N]
Appuyant sur la touche Y la liste de réseaux est affichée.

VOULEZ VOUS CONSULTER UN PROCESSUS [Y/N]

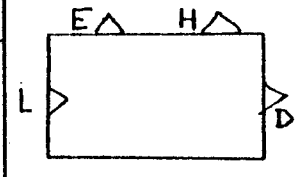
Appuyant sur la touche Y le système demande

DONNEZ LE NOM DE PROCESSUS

L'utilisateur fait renter le nom du processus par exemple
UNIT

La description graphique et formelle du processus est affichée.

NOM : UNIT ;
CONNECT : ?.L, E.? : SYMBOLE, H.?, D.? : SIGNAL;
ETATS : Q0, Q1, H : () ;
INIT : Q0 ;
REGLES : Q0 $\Rightarrow$?.L(0) E.?(λ) D.?(zz) Q0; Q0 $\Rightarrow$?.L(1) E.?(λ) D.?(zz) Q1; Q0 $\Rightarrow$?.L(λ) E.?(0) H; Q1 $\Rightarrow$?.L(0) E.?(λ) D.?(zz) Q1; Q1 $\Rightarrow$?.L(1) E.?(λ) D.?(zz) Q0; Q1 $\Rightarrow$?.L(λ) E.?(1) H; H $\Rightarrow$ H.?(zz) Q0;



Si le processus n'est pas élémentaire alors il y aura en plus l'expression formelle correspondante et le réseau graphique.

7. Fin de session

Cette commande sert pour sortir du système.

VII - CONCLUSION

Dans cette thèse nous avons présenté les résultats obtenus lors de la réalisation de quelques aspects du poste de travail. Les principaux aspects viennent d'être présentés, certains avec beaucoup de détails, d'autres de façon plus schématique, selon l'avancement de la participation aux travaux s'y rapportant. Les points importants peuvent être divisés en trois :

Une première partie est la création d'un environnement graphique pour l'ensemble du projet SPARC.

La deuxième partie développe les idées que nous avons introduites et donne des détails sur l'implantation réalisée. Le renommage des portes en cas de conflit des noms a été étudié. Avec la fonction de renommage nous avons remplacé l'opérateur de renommage, en donnant la possibilité de faire le renommage automatique des portes. L'optimisation et la simplification sont traitées d'un point de vue général. On a remplacé le calcul récursif par le calcul global de l'expression. Ceci nous a mené vers la recherche des propriétés sur les opérateurs.

En ce qui concerne l'interprétation graphique du SPARC, nous avons conçu le langage graphique de description du processus avec l'interprétation graphique des opérateurs sur la construction du réseau de processus.

Dans l'état actuel de ce qui est fait dans ce domaine, il me semble que l'on peut envisager l'intégration de FP2 avec LPG. En ce qui concerne l'interprétation graphique du projet SPARC, il reste à faire l'intégration de l'aspect graphique et de la simulation de la partie fonctionnelle de SPARC. Evidemment cet aspect sera traité parallèlement à la réalisation de la simulation du comportement en SPARC.

La conception de ce système est faite indépendamment des autres parties du projet SPARC. Ce système comporte quelques restrictions. La première est due au fait que nous avons travaillé sur un micro-ordinateur est la place disponible est 64 K y compris le système RT-11 .. La deuxième vient de l'algorithme d'unification, qui n'est pas réalisé pour le cas général.

Cependant, il ne faut pas perdre de vue le contexte général dans lequel s'inscrit ce travail. Il s'agit, d'une part, de recherches sur la méthodologie d'analyse, de validation, de synthèse et, d'autre part, des travaux sur l'implantation de ces méthodes.

VIII - BIBLIOGRAPHIE

[ARK 83] ARKAXHIU E.
"Fonctions graphiques de SPARC"
Rapport de DEA, 1983.

[BER 83] BERT D.
"Manuel de référence de LPG. Version 1.2"
R.R. n° 408, Grenoble, Décembre 1983.

[CAF 82] CAFERA R.
"Abstraction, partage de structure et retour arrière non aveugle dans la méthode de réduction matricielle en démonstration automatique de théorèmes"
Thèse de 3ème cycle, USMG, Novembre 1982.

[CAN 83] CANSELL D.
"Syntaxe et interprétation de programmes CSP applicatif"
LRIM 8309, Université de Metz, Décembre 1983.

[CIS 84] CISNEROS M.
"FP2 : un langage applicatif pour les processus communicants"
Thèse de 3ème cycle, à paraître.

[FOL 82] FOLEY J. - VANDAM A.
"Fundamentals of interactive computer graphics"
Addison Welsey Publishing Company, 1982.

[GIL 79] GILOI W.K.
"Computer graphics : Data Structures, Algorithms, Languages"
Prentice Hall, 1978.

[JOR 82] JORRAND P.
"Description and composition of communicating processes. Problems of analysis and correctness"
Rapport de recherche IMAG, R.R. n° 290, Février 1982.

[JOR 83] JORRAND P.
"Projet SPARC
IMAG, Décembre 1983

[JOR 84] JORRAND P.
"FP2 : functional parallel programming based on term substitution"
May 1984.

[LAG 83] LAGNIER P.
"Etude sur le parallélisme"
Rapport de DEA, 1983.

[LED 77] LEDUC LE BALLEUR A.
"Conception et réalisation d'un logiciel graphique interactif indépendant du contexte d'utilisation de logiciel GRIGRI"
Thèse de 3ème cycle, IMAG, 1977.

[LUC 82] LUCAS M.
"La réalisation des logiciels graphiques interactifs"
Eyrolles, Paris 1982.

[MAR 77] MARTINEZ F.
"Etude des problèmes de conception et de réalisation d'animation. Le système SAFRAN"
Thèse de 3ème cycle, IMAG, 1977.

[MIL 80] MILNEZ R.
"A calculus of communicating systems"
Lectures notes in computer sciences n° 92, Springer-Verlag, 1980.

[MON 82] MONTANARI U. - MARTELLI A.
"An efficient unification algorithm"
A.C.M. transaction on programming languages and systems.
Vol. 4, n° 2, April 1982.

[PER 84] PERREIRA H.M.
"Processus communicants : un langage formel et ses modèles. Problèmes d'analyse"
Thèse de 3ème cycle, IMAG, Juin 1984.

AUTORISATION de SOUTENANCE

VU les dispositions de l'article 3 du 16 avril 1974,

VU le rapport de présentation de Monsieur PH. JORRAND, Directeur de recherche

Monsieur ARKAXHIU Eqerem

est autorisé à présenter une thèse en soutenance pour l'obtention du titre de DOCTEUR de TROISIEME CYCLE, spécialité "Informatique".

Fait à Grenoble, le 15 juin 1984

Le Président de l'I.N.P.-G

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

P.O. le Vice-Président,