50376
1983
297

THÈSE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

pour obtenir le titre de

DOCTEUR DE 3ème CYCLE

Spécialité : Automatique

par

Ahmed NAIFI

Maître ès-Sciences

CONTRIBUTION A L'ETUDE DES SYSTEMES LOGIQUES INDUSTRIELS

Fonction combinatoire et Théorie de l'information. Sécurité et Simulation de la Partie Opérative.

Soutenue le 12 Septembre 1983 devant la Commission d'Examen

Membres du Jury : MM.	P.	VIDAL	Président
	J.M.	TOULOTTE	Rapporteur
	M.	STAROSWIECKI	Rapporteur
	J.	DEFRENNE	Examinateur
	J.P.	FRACHET	Invité

A la mémoire de mes Parents

A mes soeurs

A mes frères

Le travail de recherche présenté dans ce mémoire a été effectué au Laboratoire d'Automatique de l'Université des Sciences et Techniques de Lille 1.

Que Monsieur Pierre VIDAL, Professeur à l'Université de Lille 1, trouve ici l'expression de ma reconnaissance pour la confiance qu'il m'a accordé en m'accueillant dans son Laboratoire. Je suis très honoré qu'il ait accepté de présider mon jury de thèse.

Que Messieurs Jean-Marc TOULOTTE, Professeur et Marcel STAROSWIECKI, Maître-Assistant à l'Université de Lille 1 trouvent ici l'expression de mes plus vifs remerciements pour les conseils qu'ils m'ont prodigués tout le long de ce travail, ainsi que la confiance témoignée.

Que Messieurs Jean DEFRENNE, Maître-Assistant à l'Université de Lille 1 et J.P FRACHET, Professeur à l'INSCM - SAINT-OUEN, trouvent ici mes plus vifs remerciements pour m'avoir fait l'honneur d'examiner ce travail.

Je tiens à remercier tous les membres du Laboratoire d'Automatique, en particulier Bernard CEURSTEMONT dont l'aide a été si souvent sollicitée.

Enfin je tiens à exprimer mes plus vifs remerciements à Madame Michèle LELONG pour sa collaboration, sa compétence, sa gentillesse et sa patience lors de la frappe de ce texte.

S O M M A I R E G É N É R A L

PREMIÈRE PARTIE

FONCTION COMBINATOIRE ET THÉORIE DE L'INFORMATION

INTRODUCTION GÉNÉRALE

La fonction combinatoire joue un rôle très important dans la mise en oeuvre des systèmes logiques industriels. Elle intervient à tous les niveaux et de ce fait, son influence est très sensible sur les performances de ces systèmes. Cela explique l'intérêt croissant que les chercheurs dans ce domaine portent à l'évaluation de la fonction combinatoire.

Ils ont mis au point et cherchent à mettre au point différentes méthodes d'évaluation de celle-ci. Leur objectif principal est de faire une évaluation performante de la fonction combinatoire. Le critère de performance qui est généralement retenu, se traduit par :

. L'optimisation de la durée de l'évaluation
. L'optimisation de l'occupation mémoire des programmes engendrés

Dans le premier chapitre de cette étude, après avoir rappeler la définition de la fonction combinatoire et les diverses formes qu'elle peut revêtir, nous exposons quelques unes des méthodes d'évaluation de la fonction combinatoire qui ont été mises au point et qui résument les différentes orientations des recherches sur ce problème.

Nous présentons aussi les différentes techniques de matérialisation et d'implantation de la fonction combinatoire par les différents matériels technologiques disponibles sur le marché. Nous soulignons enfin les avantages et les inconvénients de ces techniques vis à vis des performances, de l'extension et de la maintenabilité des systèmes conçus.

Dans le deuxième et le troisième chapitre, nous proposons une nouvelle méthode d'évaluation de la fonction combinatoire. Elle permet de construire l'arbre de décision binaire représentant celle-ci.

Dans un premier temps et après avoir rappeler la définition de la structure de l'arbre de décision binaire et sa mise en oeuvre dans l'évaluation de la fonction combinatoire.

Nous étudions un critère de choix des tests à l'aide de la théorie de l'information, ainsi que sa mise en oeuvre pour la comparaison entre partitions d'un ensemble de données.

Ce critère de choix des tests sera utilisé pour associer à chaque noeud intermédiaire de l'arbre de décision binaire, le meilleur test.

Dans un deuxième temps, nous présentons l'algorithme construit autour de ce critère et qui permet d'obtenir l'arbre de décision binaire. Nous testons ensuite cet algorithme sur plusieurs exemples de fonctions combinatoires et nous discutons des résultats obtenus. Enfin, nous proposons une extension de cet algorithme à l'évaluation des fonctions combinatoires multi-valuées.

PREMIÈRE PARTIE FONCTION COMBINATOIRE ET THEORIE DE L'INFORMATION

Mots clés : Méthode logicielle, Fonction combinatoire, arbre de décision binaire, théorie de l'information, fonction de test; critère de choix des fonctions de test.

Résumé : L'étude présente une méthode logicielle sous-optimale générant un arbre de décision binaire représentant une fonction combinatoire.

Elle met en oeuvre un critère de choix des fonctions de test au niveau de chaque noeud intermédiaire de l'arbre de décision binaire généré.

Ce critère de choix des fonctions de test est basé sur la théorie de l'information. Il opère par une comparaison entre les informations apportées par les partitions créées par un ensemble de fonction de test sur la valeur d'assignation de la fonction combinatoire.

CHAPITRE I

ÉTUDE DE QUELQUES MÉTHODES D'ÉVALUATION DE LA FONCTION COMBINATOIRE

CHAPITRE I

ÉTUDE DE QUELQUES MÉTHODES D'ÉVALUATION DE LA FONCTION COMBINATOIRE

CHAPITRE I

ÉTUDE DE QUELQUES MÉTHODES D'ÉVALUATION DE LA FONCTION COMBINATOIRE

La fonction combinatoire est importante. Elle intervient à tous les niveaux dans la mise en oeuvre des automatismes logiques.

Suivant son utilisation, elle peut être plus ou moins complexe et ainsi revêtir diverses formes.

Dans ce chapitre, nous rappelons la définition de la fonction combinatoire, quelques unes des formes qu'elle peut revêtir. Nous présentons aussi quelques unes des nombreuses méthodes d'évaluation de la fonction combinatoire et nous discutons de l'implantation matérielle de celle-ci, par différents éléments technologiques. Les paramètres caractéristiques de l'évaluation (espace mémoire, vitesse de traitement) au niveau de l'implantation seront mis en évidence. Ils traduisent les performances de la méthode d'évaluation mise en oeuvre.

I.1 - QUELQUES ELEMENTS THEORIQUES

I.1.1. - Variables d'entrée - variables produit logique - modalités

Soit $X = X_1, X_2, \ldots X_n$ un ensemble de n variables booléennes $n = |X|$

Soit $B_2 = \{0,1\}$ l'algèbre de boole à 2 éléments. C'est l'ensemble des modalités.

- Variablesproduit logique

Soit $Y = B_2^n = \{0,1\}^n$ le produit cartésien de B_2 effectué n fois

On pose $Y = \{Y_1, Y_2, \ldots, Y_m\}$, avec $m = |B_2^n| = 2^n$

on a : $\forall Y_j$, $j = 1,2,\ldots,m$, $Y_j = X_1^{e_1} X_2^{e_2} X_3^{e_3} \ldots X_n^{e_n}$

avec $j = e_1\ e_2\ e_3 \ldots e_n$ (j est un indice, dont $e_1\ e_2 \ldots e_n$ est la représentation binaire, $0 < j < m$, $e_i \in B_2$) et $X_i^{e_i} \in B_2$ est la modalité de la variable booléenne $X_i \in X$ tel que :

$$X_i^{e_i} = \begin{cases} 1 & X_i = e_i \quad \text{dans ce cas on notera } X_i^1 = X_i \\ 0 & X_i \neq e_i \quad \text{dans ce cas on notera } X_i^0 = \overline{X_i} \\ & \quad (\text{complément de } X_i) \end{cases}$$

Y_j est appelé un minterme nous l'appelerons dans la suite de cette étude une variable produit logique .

I.1.2. - Définition de la fonction combinatoire

Partition caractéristique

Soit donc $X = X_1, X_2, \ldots, X_n$ l'ensemble de n - variables d'entrée définie sur l'algèbre de boole B_2.

On appelle une fonction combinatoire F, notée F(X), une application de l'ensemble Y sur l'ensemble B_2, autrement dit :

$$F : = Y \longrightarrow B_2 \qquad (I.1a)$$

$$\forall Y_j,\ Y_j \in Y,\ j = 1,2,\ldots,m \qquad F(Y_j) = \beta \in B_2$$

En d'autre terme une fonction combinatoire de n variables d'entrée est une fonction qui a toute combinaison binaire à n digit :
$e = e_1\ e_2 \ldots e_n$ fait correspondre la valeur $\beta \in B_2$.

La fonction combinatoire F(X) serait une partition de l'ensemble Y. Soit P_F cette partition, on a alors :

$$P_F(Y) = \{P_\beta, \beta \in B_2\} \text{ avec } P_\beta = \{Y_j \in Y / F(Y_j) = \beta \in B_2\} \qquad (I.1b)$$

P_β est une classe de la partition P_F.

P_F est la partition caractéristique de la fonction combinatoire F(X).

Nous considérons par la suite que F(X) est complètement définie autrement pour :

$$\beta, \alpha \in B_2 \quad \forall P_\beta, P_\alpha \in P_F \text{ on a :}$$

$$P_\beta \cap P_\alpha = \phi \text{ pour } \beta \neq \alpha$$

La partition caractéristique représente l'une des diverses formes que la fonction combinatoire F(X) peut prendre.

I.2 - DIVERSES FORMES DE LA FONCTION COMBINATOIRE

Sur l'ensemble X des variables booléennes d'entrée et l'ensemble B_2, une fonction combinatoire F(X) peut être construite par application des règles suivantes | GIV.70, LEE.76 | .

1/ Soit a une constante définie sur B_2 . F(X) = a et F(X) = X_i, i=1,..n sont des fonctions combinatoires. La première est appelée la fonction constante, la seconde la fonction projection.

b/ Si F(X) est une fonction combinatoire, alors $\overline{F(X)}$ est une fonction combinatoire. $\overline{F(X)}$ est la fonction complémentaire de F(X).

c/ Si $F_1(X)$ et $F_2(X)$ sont deux fonctions combinatoires : alors $F_1(X) + F_2(X)$ et $F_1(X) . F_2(X)$ (+, . représentent respectivement les opérations de réunion et d'intersection) sont des fonctions combinatoires.

d/ Seules les fonctions résultantes de l'application, en un certain nombre fini d'étapes des règles (a, b, c) sont des fonctions combinatoires.

Parmi les diverses formes algébrique résultantes de l'application des règles ci-dessus, les plus utilisées sont : la première forme canonique, l'écriture simplifiée en sommes d'intersections premières (implicants premières) et la forme avec mise en facteur. Ces formes sont illustrées par l'exemple de la figure I.1.

	X_1	X_2	X_3	F(X)
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

$$F(X) = X_1 \overline{X}_2 X_3 + X_1 X_2 \overline{X}_3 + \overline{X}_1 X_2 X_3 + X_1 X_2 X_3$$

$$F(X) = X_1 X_2 + X_1 X_3 + X_2 X_3$$

$$F(X) = X_1 (X_2 + X_3) + X_2 X_3$$

Partition caractéristique

$$P_F = \{P_1, P_0\} : P_1 = \{3,5,6,7\}$$

$$P_0 = \{0,1,2,4\}$$

Figure I.1 : Diverses formes de la fonction combinatoire

D'autres formes plus complexes sont évidemment possibles, mais quelque soit la forme de la fonction combinatoire . La mise en oeuvre de cette dernière dans les automatismes logiques, nécessite des méthodes d'évaluation et d'implantation de celle-ci.

I.3 - EVALUATION DE LA FONCTION COMBINATOIRE

L'évaluation de la fonction combinatoire se caractèrise par deux paramètres :

. *l'espace mémoire occupé par les programmes engendrés*
. *la durée moyenne de l'évaluation de la fonction combinatoire*

Dans l'évaluation de la fonction combinatoire, les chercheurs dans ce domaine, ont cherché et cherchent encore à mettre au point des méthodes d'évaluations qui permettent de minimiser ces deux paramètres.

I.3.1. - L'occupation mémoire

Suivant le matériel utilisé dans l'implantation de la fonction combinatoire, l'occupation mémoire peut être évaluée en octet (systèmes programmés) ou en nombre de modules dans d'autres technologies (circuit multiplexeurs, réseaux logique programmable ...).

L'occupation mémoire peut être réduite, par la description d'une représentation minimale des fonctions combinatoires. Le critère de la représentation minimale est souvent le suivant | GIV.70 | :

i c'est la forme qui contient le nombre le plus petit de littéraux

ii c'est la forme constituée par le plus petit nombre de termes (qui peut être l'une des deux formes normales ou non).

La minimisation d'une fonction combinatoire est accomplie pour obtenir une somme de produits (où produit de somme) minimale | GIV.70, LEE.76, ARE.78 | . Toutefois, cette forme n'est pas toujours la forme optimale pour l'exécution (temps de traitement).

I.3.2. - Durée d'évaluation d'une fonction combinatoire

La durée d'évaluation d'une fonction combinatoire correspond au temps écoulé entre l'instant d'apparition d'une combinaison des variables d'entrées et celui où la fonction combinatoire est assignée à la valeur correspondante.

Dans les systèmes programmés, elle est liée au temps d'exécution du programme qui est difficilement chiffrable dans les systèmes programmés universels, mais directement lié au temps de cycle dans les automates programmables qui sont plus appropriés à ce genre de traitement.

Dans les autres systèmes (réalisations spéciales), elle correspond au temps de réponse du circuit réalisant la fonction combinatoire.

Dans l'évaluation d'une fonction combinatoire, il est toujours souhaitable de minimiser le temps de traitement pour améliorer les performances du système dans lequel la fonction combinatoire intervient.

La façon la plus efficace pour élaborer la valeur d'une fonction combinatoire, c'est de la transformer en arbre de décision binaire | BRE.72, MAN.78, POL.71, SIL.78 | .

En effet, si "n" est le nombre de variables booléennes de la fonction combinatoire, le plus long chemin de l'arbre est la longueur "n" donc la valeur de la fonction combinatoire est obtenue au bout de "n" tests au maximum. Cependant, l'utilisation des arbres de décision binaire pose le problème de la définition d'une structure permettant une évaluation optimale de la fonction combinatoire (problème complexe d'optimisation | POL .71 |).

De nombreuses études ont été entreprises pour évaluer la fonction combinatoire. De ces études, il ressort deux types d'évaluations.

- Une évaluation de la fonction combinatoire sous forme d'arbre de décision binaire.
- Une évaluation séquentielle de la fonction combinatoire.

Nous présenterons ci-dessous quelques méthodes de ces différentes évaluations.

I.4 - QUELQUES METHODES D'EVALUATION DE LA FONCTION COMBINATOIRE

I.4.1. - Evaluation de la fonction combinatoire par arbre de décision binaire

L'idée consiste à construire un arbre de décision binaire (ADB) qui représente la fonction combinatoire et tel que l'évaluation de la fonction revient à suivre une suite d'aiguillages entre les chemins de l'ADB. Chaque pas est déterminé par l'état d'un test (vrai-faux) constitué par une variable booléenne où par une fonction de variables booléennes.

Cette approche est intéressante parce qu'elle offre d'une part une évaluation performante et d'autre part elle permet une implantation matérielle facile et pratique surtout sur calculateur universel.

Nous présentons ci-dessous deux procédures qui permettent de construire l'ADB qui représente une fonction combinatoire donnée.

I.4.1.1. - Méthode heuristique de construction de l'ADB

| SIL.78 |

La méthode heuristique proposée par l'auteur, comprend deux étapes :

- la détermination d'une couverture disjointe de la fonction avec un nombre minimum de monômes.
- l'obtention de l'ADB par la mise en oeuvre de la formule d'expansion de Shannon | AKE.78 | suivante :

Soit $X = \{ X_1, X_2, \ldots, X_n \}$ un ensemble de variables booléennes et $F(X)$ une fonction combinatoire définie sur X. On peut décomposer $F(X)$ en utilisant la formule :

Pour $\forall X_i \in X$, $i = 1,2,\ldots,n$

$$F(X) = F(X_1,X_2,\ldots,X_n) = X_i\, F(X_1,X_2,\ldots 1,\ldots,X_n) + \overline{X}_i\, F(X_1,X_2,\ldots,0,\ldots X_n) \qquad (I.2)$$

La construction de l'ADB représentant F(X) consiste donc à répéter l'application de la formule (I.2) suivant les étapes suivantes

i : Appliquer la formule (I.2) sur la variable la plus fréquente, rencontrée dans la forme algébrique de la fonction combinatoire F(X) (le choix est arbitraire si la fréquence des variables est la même).

ii : Appliquer la formule (I.2) sur les expressions des facteurs résultants de l'application précédente.

iii : Répéter (ii) jusqu'à ce que les facteurs résultants ne contiennent plus que des monômes comportants une seule variable booléenne, auquel cas : on obtient l'arbre de décision binaire associé à la fonction considérée.

EXEMPLE : Soit $F(X) = X_1 \overline{X}_3 X_4 + X_1 X_3 \overline{X}_4 + X_2 X_3 X_4 + X_2 \overline{X}_3 \overline{X}_4$

Appliquons l'algorithme décrit ci-dessus pour construire l'ADB associé à F(X).

Dans l'expression de F(X), on remarque que les variables X_3 et X_4 sont les plus fréquentes : les poids de X_3 et X_4 étant identiques, le choix sera alors arbitraire : on choisit la variable X_3 . Appliquons la formule (I.2); on trouve :

$$F(X) = X_3 \mid X_1 \overline{X}_4 + X_2 X_4 \mid + \overline{X}_3 \mid X_1 X_4 + X_2 \overline{X}_4 \mid$$

on pose

$$F_{\overline{X}_3} = X_1 X_4 + X_2 \overline{X}_4 \; ; \; F_{X_3} = X_1 \overline{X}_4 + X_2 X_4$$

2ème application de (I.2)

$$F_{\overline{X}_3} = X_1 X_4 + X_2 \overline{X}_4 = X_4 \mid X_1 \mid + \overline{X}_4 \mid X_2 \mid$$

$$F_{X_3} = X_1 \overline{X}_4 + X_2 X_4 = X_4 \mid X_2 \mid + \overline{X}_4 \mid X_1 \mid$$

d'où l'ADB représentant $F(X) = X_1 \overline{X}_3 X_4 + X_1 X_3 \overline{X}_4 + X_2 X_3 X_4 + X_2 \overline{X}_3 \overline{X}_4$:

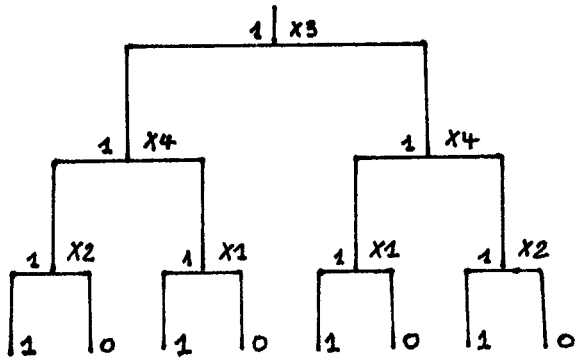


Figure I.2 : ADB associé à F(X)

Les résultats obtenus par cette méthode sont intéressants, bien qu'une représentation optimale à la fois par rapport à l'occupation mémoire et à la rapidité du traitement, n'est pas possible. La solution proposée offre un compromis possible entre ces deux paramètres. En effet l'application de la méthode peut donner une implantation tabulaire réduite en même temps que le nombre de test pour arriver à une décision reste convenable. De plus la méthode offre une mise en oeuvre facile et pratique.

I.4.1.2. - Une méthode de synthèse des arbres de décision binaires minimaux pour l'évaluation de la fonction combinatoire| MAN.79 |

La méthode proposée par l'auteur permet de faire l'évaluation d'une fonction combinatoire simple ou multiple (complètement ou incomplètement définie), |MAN.78, 79 |. Elle suggère la synthèse des fonctions combinatoires sous forme d'arbres minimaux de multiplexeurs lorsqu'il s'agit de conception de systèmes logiques câblées; où sous forme d'arbres de décision binaires lorsqu'il s'agit d'une implantation programmée.

Dans cette étude l'auteur exploite les propriétés des ensembles standards |DAV.77| pour décrire un arbre de décision binaire.

1/ Ensemble standard-propriétés

Considérons un arbre de multiplexeurs (fig.I.3), réalisant la fonction combinatoire $F(X_1, X_2, X_3, X_4)$.

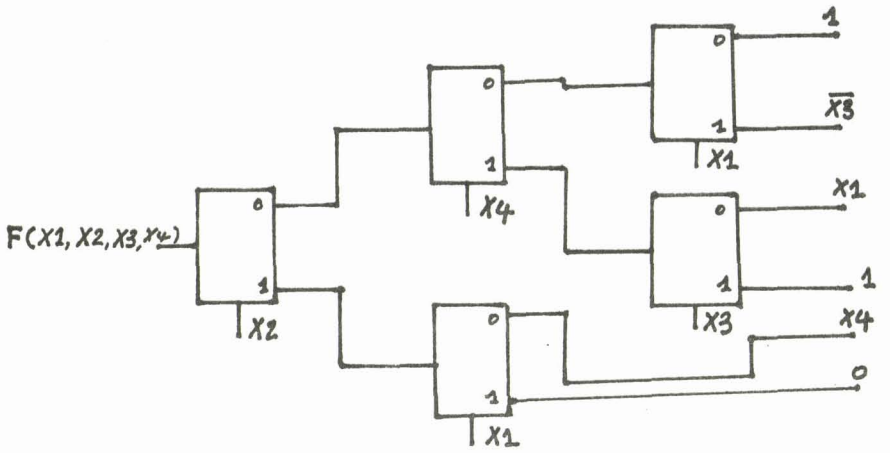


Figure I.3

Rappelons qu'un multiplexeur à une variable, est un système logique réalisant l'équation :

$$Z = \grave{a}\ K_0 + a\ K_1 \qquad \text{(I.3a)}$$

a : est une variable de commande, K_0, K_1 deux variables auxiliaires et Z une variable de sortie.

En appliquant la relation (I.3a), à chacun des multiplexeurs de la figure I.3, on obtient l'expression de la fonction combinatoire $F(X_1,X_2,X_3,X_4)$ réalisée par cet arbre :

$$F(X_1,X_2,X_3,X_4) = \overline{X}_2\,\overline{X}_4\,X_1 + X_1\,\overline{X}_2\,\overline{X}_3\,X_4 + X_1\,X_2\,X_3\,X_4 + \overline{X}_2\,X_3\,X_4 + X_2\,\overline{X}_1\,X_4$$

$$= \{\,0,2,3,5,7,8,9,11\}. \qquad \text{(I.3b)}$$

Etudions les propriétés du polynômes (I.3b) représentant la fonction combinatoire : $F(X_1, X_2, X_3, X_4)$. On remarque que :

a/ l'ensemble des impliquants de $F(X_1, X_2, X_3, X_4)$ recouvre les huits mintermes de cette fonction : c'est la propriété de couverture.

b/ l'ensemble des impliquants de $F(X_1, X_2, X_3, X_4)$ réalise une partition des huits mintermes de cette fonction : c'est la propriété de séparation. En d'autres termes cela veut dire que chaque minterme n'est couvert que par un seul impliquant.

c/ toute branche vive de l'arbre (fig.I.3) réalise un impliquant de $F(X_1, X_2, X_3, X_4)$. (rappelons qu'une branche est vive si le point d'entrée n'est pas à 0). C'est la propriété de compatibilité.

Tout ensemble d'impliquants qui satisfait les trois propriétés précédentes, est un ensemble standard. Un tel ensemble est toujours réalisable par un arbre et inversement, un arbre produit toujours un ensemble standard.

Il suffit donc, pour une fonction combinatoire donnée, de chercher l'ensemble standard qui est réalisable par un arbre minimal.

2/ Algorithme de recherches des arbres minimaux

Le but de la procédure est la recherche systématique des arbres minimaux d'une fonction combinatoire incomplètement définie. Elle est décrite par les étapes principales suivantes :

- *calcul de tous les impliquants de la borne supérieure de la fonction*
- *détermination des ensembles d'impliquants qui recouvre la borne inférieure de la fonction et qui définissent une partition des mintermes de celle -ci (propriétés de couverture et de séparation)*
- *choix parmi ces ensembles, ceux qui sont standards (compatibilité)*
- *sélection, parmi les ensembles standards de ceux qui sont réalisables par un arbre minimal.*

3/ Exemple : Pour illustrer cette procédure d'évaluation, on considère la fonction combinatoire incomplètement définie suivante:

$$F(X_1,X_2,X_3,X_4) = \Sigma\ 0,2,3,5,7,8,9 + \Phi\ 10,11,12,13,14,15 \qquad (I.4)$$

$$= X_1 + X_2\ X_4 + \overline{X}_2\ \overline{X}_4 + X_3\ X_4$$

La mise en oeuvre de l'algorithme décrit ci-dessus permet alors de :

- calculer tous les impliquants de la borne supérieure de la fonction combinatoire (I.4). On applique pour cela la méthode de Mc CLUSKEY| MC.4 | à $F_3(X_1,X_2,X_3,X_4) = \Sigma\ 0,2,3,5,7,8,9,10,\ldots,15$. On trouve trente deux impliquants qui recouvrent un ou plusieurs mintermes de la borne inférieure.

$$F_I(X_1,X_2,X_3,X_4) = \Sigma\ 0,2,3,5,7,8,9\ . \qquad (I.5)$$

Le tableau de la figure (I.4) représente ces impliquants, en indiquant pour chacun d'eux les mintermes de (I.5) qu'il recouvre.

On calcule ensuite un score qui est le nombre de couverture possible pour chaque minterme de (I.5).

On classe les mintermes de (I.5) selon l'ordre croissant du score soit :

mintermes →	Σ 0	5	2	3	7	9	8
score	4	4	6	6	6	8	10

(I.6)

On détermine les ensembles d'impliquants qui recouvrent la borne inférieure (I.5) et qui définissent une partition des mintermes de celle-ci.(couverture et séparation). Cette détermination se fait en utilisant l'arbre de couverture de Meisel |MEI | (fig.I.4b).

L'établissement de cet arbre commence par l'étude de la couverture du minterme qui a le plus petit score. Soit donc dans notre exemple le minterme 0 :

les couvertures possibles sont les impliquants : $\overline{X}_2\ \overline{X}_4$, $\overline{X}_1\ \overline{X}_2\ \overline{X}_4$, $\overline{X}_2\ \overline{X}_3\ \overline{X}_4$ et $\overline{X}_1\ \overline{X}_2\ \overline{X}_3\ \overline{X}_4$ (fig.I.4a) qui déterminent quatre branches distinctes de l'arbre de couverture (fig. I.4b).

L'examen de la couverture du minterme 5 fait apparaître quatre couvertures $X_2\ X_4$, $\overline{X}_1\ X_2\ X_4$, $X_2\ \overline{X}_3\ X_4$ et $\overline{X}_1\ X_2\ \overline{X}_3\ X_4$. Elles produisent seize branches distinctes; chacune d'elle couvre les mintermes 0 et 5 au minimum. Dans une branche aucun minterme n'est recouvert deux fois (séparation).

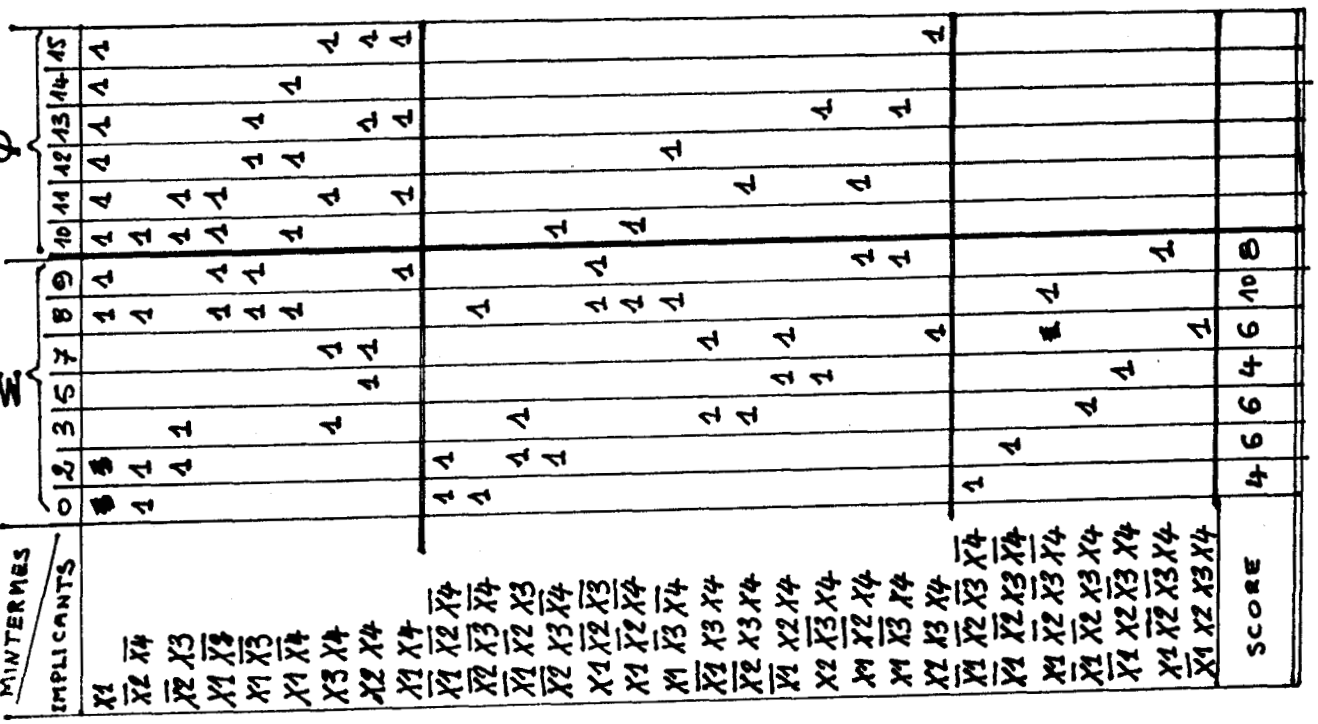


Figure I.4a : Table des implicants de F(X)

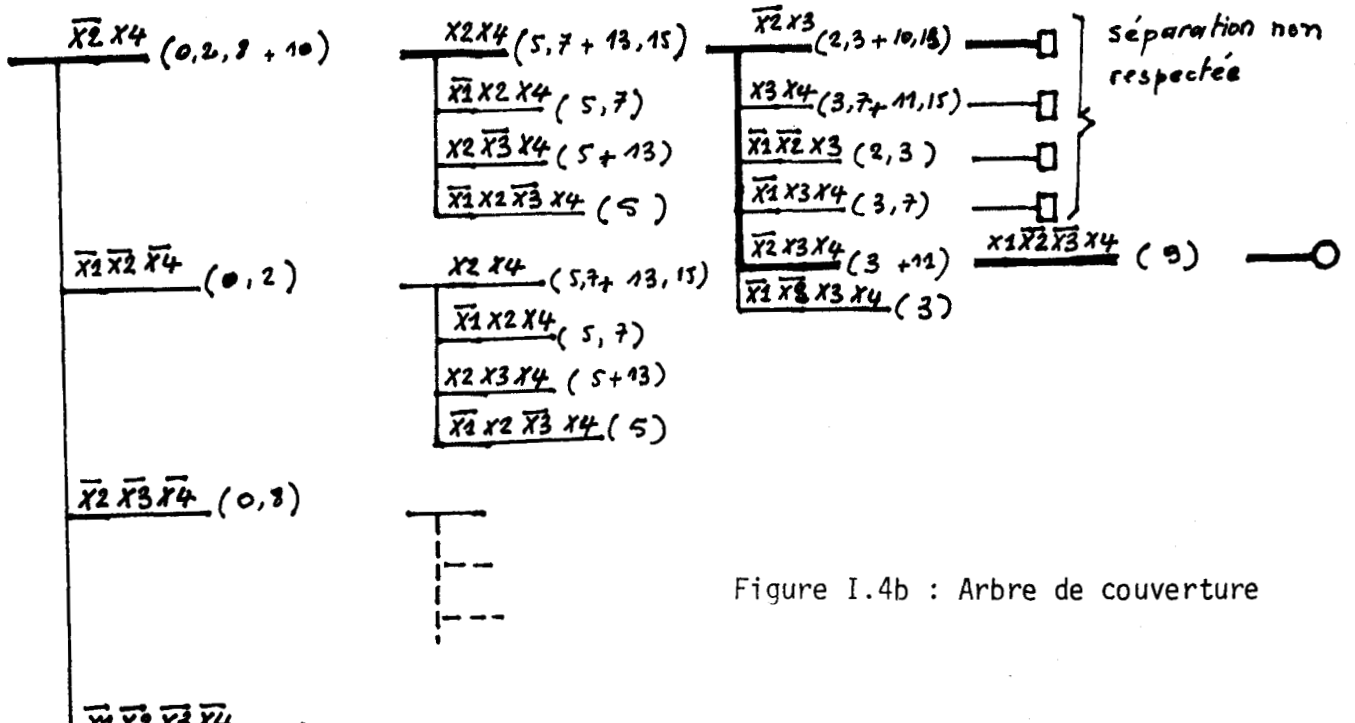


Figure I.4b : Arbre de couverture

Une branche de l'arbre de couverture, dans laquelle tous les mintermes de la borne inférieure de $F(X_1, X_2, X_3, X_4)$, (I.5) sont couvertes, correspond à un ensemble d'impliquants. C'est le cas de la branche ($\overline{X}_2\ \overline{X}_4, X_2\ X_4, \overline{X}_2\ X_3\ X_4, \overline{X}_1\ \overline{X}_2\ \overline{X}_3\ \overline{X}_4$) (fig.I.4b) soit donc

$$\{\ \overline{X}_2\ \overline{X}_4\ ,\ X_2\ X_4,\ \overline{X}_2\ X_3\ X_4,\ \overline{X}_1\ \overline{X}_2\ \overline{X}_3\ \overline{X}_4\ \}\quad \text{cet ensemble} \qquad (I.6)$$

Il vérifie les propriétés de (couverture et de séparation). Il reste à tester la propriété de compatibilité. Le fait que l'ensemble (I.6) admet les représentations bidimensionnelles (I.7a,b), vérifie la propriété de compatibilité et ainsi l'ensemble (I.6) est standard.

$$F(X_1,X_2,X_3,X_4) = \left|\begin{array}{l} \overline{X}_2 \rightarrow \left|\begin{array}{l} \overline{X}_4.1 \\ X_4 \rightarrow \left|\begin{array}{l} \overline{X}_3.X_1 \\ X_3.1 \end{array}\right. \end{array}\right. \\ X_2 \end{array}\right. \quad (a)$$

(fig.I.7)

$$F(X_1,X_2,X_3,X_4) = \left|\begin{array}{l} \overline{X}_4.\overline{X}_2 \\ X_4 \rightarrow \left|\begin{array}{l} \overline{X}_2 \rightarrow \left|\begin{array}{l} \overline{X}_3.X_4 \\ X_3.1 \end{array}\right. \\ X_2.1 \end{array}\right. \end{array}\right. \quad (b)$$

La représentation (I.7a) pourra-t-être réalisable par un arbre de multiplexeurs. La figure (I.8a) décrit cet arbre.

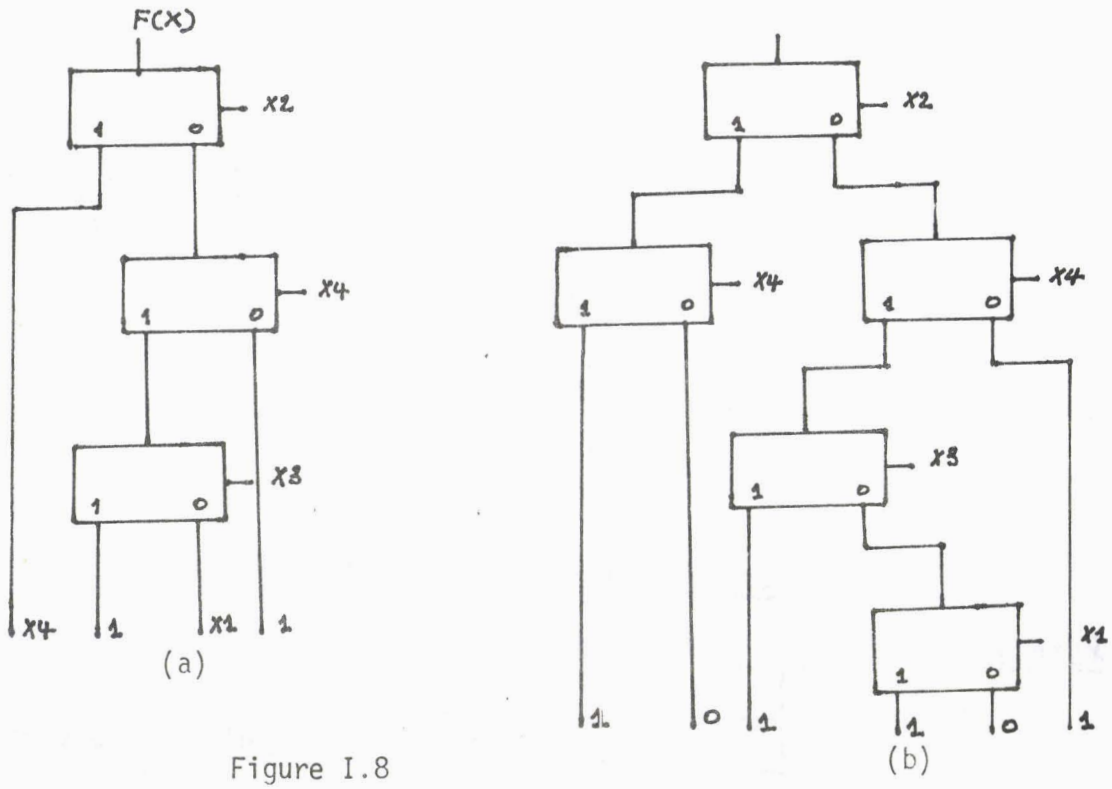


Figure I.8

Il reste à vérifier que l'arbre de la figure I.8a est minimal : on remarque que la relation : $k > p - 1$ (k = nombre de multiplexeurs, p = nombre d'impliquants) est vérifiée puisque l'ensemble standard (I.6), ainsi que l'ensemble des impliquants premiers ($p = 4$) comportent quatre éléments. L'arbre de la figure I.8a est donc minimal.

A ce stade de la recherche, l'évaluation de la fonction combinatoire $F(X_1,X_2,X_3,X_4)$ est donc terminée si une seule solution suffit.

Cette étude illustre une des nombreuses méthodes d'évaluation de la fonction combinatoire (complétement ou incomplétement définie) par un arbre de décision binaire. L'exploitation des propriétés des ensembles standard (couverture, séparation, compatibilité), sont à la base de cette méthode. Les arbres minimaux obtenus sont matérialisables directement par des multiplexeurs.

Des arbres de décision binaire sont facilement obtenus par des modifications simples des arbres minimaux, si le souhait d'une matérialisation par les systèmes programmés est émis.

I.4.1.3. - Conclusion

De nombreuses recherches sur l'évaluation de la fonction combinatoire sous forme d'arbres de décisions binaires, ont été menées. Elles sont justifiées par l'utilisation croissante de ces arbres | LEE.59, DAV.77, BOV.77 | .Les critères d'optimisation sont très divers. Dans la méthode que nous avons présenté,les auteurs ont cherché à minimiser le nombre des éléments (nombre de multiplexeurs) qui réalisent l'arbre d'évaluation de la fonction combinatoire. THAIZE et DAVID | DAV.77 | ont montré que la minimisation de la longueur maximum d'une branche de l'arbre (temps de traitement maximum) est aussi lié à la détermination des ensembles standards.

D'autres travaux récents sur l'évaluation de la fonction combinatoire ont été publiés. Dans | PER.76 | les auteurs considèrent une fonction combinatoire décrite par un certain nombre de sous-fonctions. Chaque sous-fonction combinatoire est définie sur un ensemble de variables booléennes. Ces ensembles de variables sont disjoints entre eux, d'autre part chaque sous-fonction est représentée par un arbre de décision binaire. Les auteurs ont montré qu'il est possible de trouver l'arbre de décision binaire optimal de la fonction combinatoire considérée et celà en cherchant un meilleur arrangement des arbres des sous-fonctions données.

Dans le processus d'évaluation des fonctions combinatoires par arbre de décision binaire, il est indispensable d'attendre l'acquisition de l'ensemble des valeurs des variables booléennes d'entrée pour commencer l'évaluation. Il est possible d'opérer autrement; en effet, on peut considérer que l'évaluation de la fonction combinatoire peut se faire au fur et à mesure de l'apparition des variables booléennes d'entrée. L'évaluation est donc séquentielle. Cette approche constitue une deuxième orientation des recherches sur l'évaluation de la fonction combinatoire.

I.4.2. - Evaluation séquentielle de la fonction combinatoire

Dans l'approche traditionnelle d'évaluation de la fonction combinatoire, la phase d'évaluation ne commence qu'après avoir fait l'acquisition de toutes les valeurs des variables booléennes d'entrée.

Une autre approche peut être envisagée, elle concerne l'évaluation de la fonction combinatoire au fur et à mesure de l'apparition des variables d'entrée. L'assignation de la fonction combinatoire à une valeur se fait lors de l'apparition de la dernière variable d'entrée.

Dans ces conditions on parlera d'évaluation séquentielle de la fonction combinatoire.

Cette technique d'évaluation n'est pas très utilisée, bien qu'elle peut constituer une solution intéressante dans des situations particulières. Des recherches restent à faire dans cette direction pour améliorer les performances de cette nouvelle forme d'évaluation. Parmi les travaux rares qui ont été faits à notre connaissance, nous rappelons ci-dessous les principaux points d'une méthode d'évaluation séquentielle développée par PLAUSIC E DANIELSSON | PLA.79 |.

Dans leur étude, les auteurs en partant d'une fonction combinatoire sous forme de sommes de produits logiques, ont utilisé une forme paramètrique pour développer des algorithmes d'évaluation ainsi :

$$\text{soit } F(X) = F(X_1, X_2, X_3, X_4) =$$

$$= P_1 + P_2 + P_3$$

$$P_1 = \overline{X}_1 X_2 X_3 \; ; \; P_2 = X_2 X_3 \overline{X}_4 ; \; P_3 = X_1 \overline{X}_3 \qquad (I.8)$$

F(X) est mise sous la forme paramètrisée suivante :

$$F(X) = (X_1 \, Ra_{11}) \, (X_2 \, Ra_{12}) \, ... \, (X_n \, Ra_{1n})$$

$$+ (X_1 \, Ra_{21}) \, (X_2 \, Ra_{22}) \, ...(X_n \, Ra_{2n})$$

$$"$$

$$"$$

$$+ (X_1 \, Ra_{m1}) \, (X_2 \, Ra_{m2}) \, ...(X_m \, Ra_{mn}) \qquad (I.9)$$

où X_1, X_2, ... X_n représentent les variables d'entrée $X_i \in \{0,1\}$
a_{11}, a_{12}, ..., a_{mn} sont des paramètres qui définissent la fonction
$a_{ij} \in \{0,1,x\}$ (x = indifférent).

R est une relation (une application sur l'ensemble {0,1}).Elle est définie par la table de vérité :

x	a	x Ra
0	0	1
0	1	0
0	x	1
1	0	0
1	1	1
1	x	1

Autrement dit le vecteur d'entrée $X = X_1, X_2, X_3, \ldots X_n$ est comparé bit par bit selon la relation R avec m vecteurs :

$$A_i = a_{i1}, a_{i2}, \ldots, a_{in} \quad , i = 1, 2, \ldots m.$$

Ainsi les paramètres correspondants à la fonction décrite par (I.8) sont les suivants :

$$A = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & x \\ x & 1 & 1 & 0 \\ 1 & x & 0 & x \end{bmatrix}$$

Les résultats partiels nécessaires à l'évaluation de la fonction combinatoire peuvent être exprimés comme suit :

$$\text{si} \quad p_{ij} = x_j \; R \; a_{ij}$$

$$\begin{aligned} \text{Alors} \quad F(X) &= p_{11} \cdot p_{12} \cdots \cdots p_{1n} \\ &+ p_{21} \quad p_{22} \cdots \cdots p_{2n} \\ &\text{"} \\ &\text{"} \\ &+ p_{m1} \quad p_{m2} \cdots \cdots p_{mn} \end{aligned}$$

on pose

$$P_i = p_{i1} \quad p_{i2} \cdots \cdots p_{in}$$

$$F(X) = P_1 + P_2 + P_3 + \ldots . P_m$$

on peut écrire les P_i sous une forme itérative :

$$P_i = P_{ij} \cdot p_{i,j+1} \cdots . p_{in}$$

$$\text{Soit} \begin{cases} j = 1, 2, \ldots n \\ P_{i0} = 1 \\ P_{ij} = P_{i,j-1} \cdot p_{ij} \\ P_i = P_{in} \end{cases} \qquad (I.9)$$

P_{ij} est le résultat partiel obtenu à l'étape j (lors de l'apparition de la variable booléenne x_j).

On peut définir également sous la même forme itérative F(X); on a

$$F = F_i + P_{i+1} + \ldots + P_m$$

avec

$$\begin{cases} i = 1,2 \ldots m \\ F_0 = 0 \\ F_i = F_{i-1} + P_i \\ F = F_m \end{cases} \qquad (I.10)$$

L'exploitation des deux formes itératives représentées par (I.9) et (I.10) sont la base de l'évaluation séquentielle de la fonction combinatoire. Les auteurs ont montré les différents schémas de réalisations. Nous décrivons deux schémas qui montrent bien l'aspect séquentiel de l'évaluation.

Dans la réalisation décrite par la figure (I.8a) ; les variables d'entrée apparaissent en séquence. Cela permet d'évaluer partiellement F_i tel que :

$$F = F_i + P_i + \ldots P_m$$

mais aussi les fonctions d'évaluation partielle g_j définies par :

$$g_0 = P_{10} + P_{20} + \ldots + P_{m0} = 1$$

"

"

$$g_j = P_{1j} + P_{2j} + \ldots + P_{mj}$$

"

"

$$g_n = P_{1n} + P_{2n} + \ldots + P_{mn}$$

$$+ P_1 + P_2 + \ldots + P_m = F(X) \qquad (I.11)$$

Dans le schéma de la figure (I.8a) l'apparition d'une variable X_j conditionne l'apparition du vecteur de comparaison correspondant soit $\overline{A}_j = a_{1j}, a_{2j} \ldots a_{mj}$. Il résulte de cette comparaison le vecteur $p_{1j}, p_{2j}, \ldots, p_{mj}$. L'application d'un ET logique entre les éléments de ce vecteur et celui du vecteur $P_{1,j-1}, P_{2,j-1}, \ldots P_{m,j-1}$ résultant de l'apparition de la variable X_{j-1} permet de déterminer les produits partiels P_{1j}, $P_{2j} \ldots, P_{mj}$.

Cette procédure est répétée jusqu'à l'apparition de la dernière variable d'entrée X_n. L'évaluation de la fonction combinatoire considérée est alors terminée.

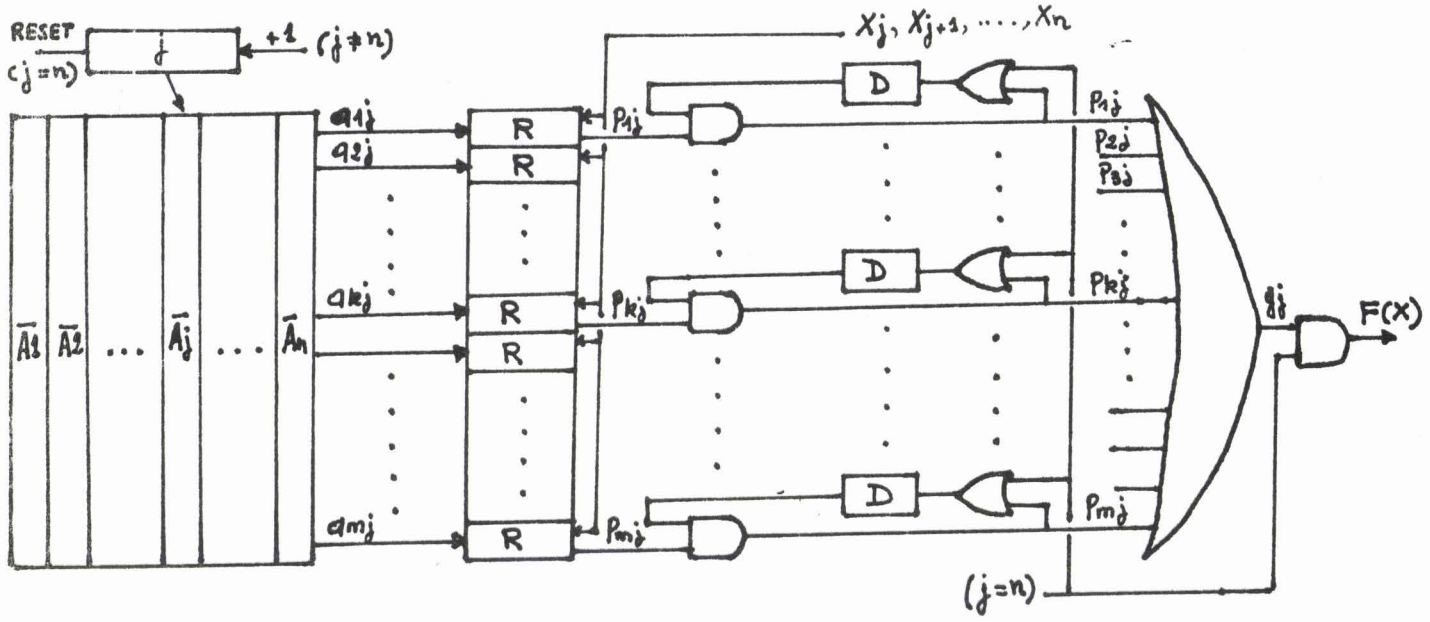


Figure I.8a

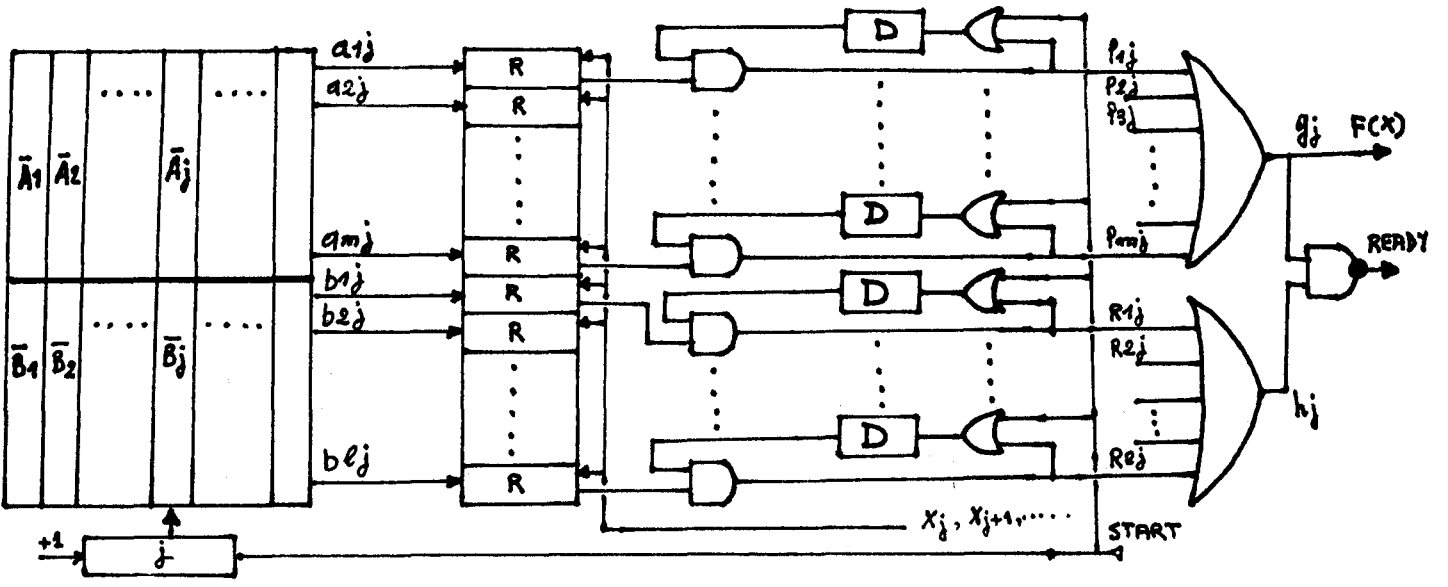


Figure I.8b

Une évaluation rapide peut être envisagée (figure I.8b) . Cela nécessite l'évaluation simultanée des fonctions F(X) et $\bar{F}(X)$ suivant la même procédure que celle décrite par la figure I.8a.

L'élaboration d'un signal (READY) à partir des évaluations partielles (g_j, h_j) obtenues à chaque apparition d'une variable X_j permet d'arrêter le processus d'évaluation sans attendre que toutes les variables d'entrée apparaissent. L'exemple de la figure (I.9) montre que l'évaluation de la fonction combinatoire décrite par (I.8) et son complément

$$\bar{F}(X) = R_1 + R_2 + R_3 \quad \text{tel que :}$$

$$R_1 = \bar{X}_1 \bar{X}_3 \; ; \; R_2 = \bar{X}_2 \bar{X}_3 \; ; \; R_3 = X_1 X_2 X_3$$

$$B = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix} = \begin{bmatrix} 0 & x & 0 & x \\ x & 0 & 1 & x \\ 1 & x & 1 & 1 \end{bmatrix}$$

j	0	1	2	3	4
X_j		0	0	1	0
g_j	1	1	0	0	0
h_j	1	1	1	1	1
READY	0	0	1	1	1

(a) F(X) = 0

j	0	1	2	3	4
X_j		0	1	1	0
g_j	1	1	1	1	1
h_j	1	1	1	0	0
READY	0	0	0	1	1

(b) F(X) = 1

Figure I.9

se termine lors de l'apparition de la variable X_2 (fig.I.9a) pour le vecteur d'entrée (0 0 1 0). Pour le vecteur d'entrée (0 1 1 0), (fig.I.9b), l'évaluation de F(X) se termine avec l'apparition de la variable d'entrée X_3.

On remarque dans cet exemple que la recherche de la rapidité du traitement de l'évaluation se fait au détriment d'un espace mémoire d'occupation beaucoup plus important. Cela illustre bien le problème de l'évaluation de la fonction combinatoire quand il est posé en terme d'optimisation des deux paramètres caractéristiques de l'évaluation à savoir (vitesse de traitement et espace mémoire).

Dans cette étude les auteurs ont proposé une procédure d'évaluation de la fonction combinatoire, dans laquelle cette dernière est représentée par une forme paramètrique.

L'ensemble des paramètres ainsi obtenus, permet par une comparaison avec les valeurs des variables d'entrées de déterminer l'assignation de la fonction combinatoire considérée. L'utilisation de la forme itérative permet de prendre en compte les variables d'entrées au fur et à mesure qu'elles apparaissent ce qui détermine des évaluations partielles et de ce fait l'assignation de la fonction combinatoire n'aura lieu qu'avec l'apparition de la dernière variable d'entrée.

Une version plus performante de la procédure proposée a été étudiée. Elle permet, si cela est possible d'assigner la fonction combinatoire avant l'apparition de toutes les variables d'entrée.

La procédure d'évaluation séquentielle proposée permet l'évaluation des fonctions combinatoires (complètement et incomplètement définies) et de fonctions multiples.

Le but recherché par les auteurs dans cette étude, est de présenter une nouvelle approche de l'évaluation de la fonction combinatoire. Cela peut justifier le peu d'intérêt qu'ils ont accordé à l'étude des performances de l'évaluation séquentielle proposée.

I.5 - MATERIALISATION INDUSTRIELLE DE LA FONCTION COMBINATOIRE

La mise en oeuvre de la fonction combinatoire dans un automatisme industriel nécessite le choix d'un matériel adéquat à son implantation. Les critères de ce choix peuvent être très divers. Vu la diversité des technologies disponibles qui a été rendu possible grâce aux progrès technologiques réalisées ces dernières années. Le choix est devenu très difficile à faire.

Toutefois, on peut remarquer qu'il y a deux grandes orientations qui concernent la matérialisation des fonctions combinatoires. Il s'agit de matérialisation par des systèmes programmés. L'autre matérialisation est faite par des systèmes câblés. Elle constitue une réalisation spéciale car les éléments technologiques mis en oeuvre sont spécifiques à ce genre d'application.

I.5.1. - Réalisation de la fonction combinatoire par les systèmes câblés

La matérialisation de la fonction combinatoire peut être faite par un assemblage de réseaux logiques particuliers. L'apparition des circuits tel que PLA, PAL, FPLA ..., où la mise en oeuvre des circuits de mémorisation tel que (ROM, RAM, PROM) ont énormément facilité l'élaboration de structures spéciales capables de traiter la fonction combinatoire.

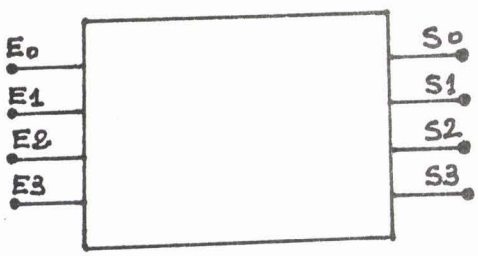
Suivant l'assemblage de ces éléments, on peut obtenir soit des structures capables d'implanter directement la fonction combinatoire par un câblage approprié ou éventuellement par une pré-programmation. Soit des structures séquentielles bien adaptées et permettant un traitement programmé de la fonction combinatoire.

I.5.1.1. - Implantation directe de la fonction combinatoire

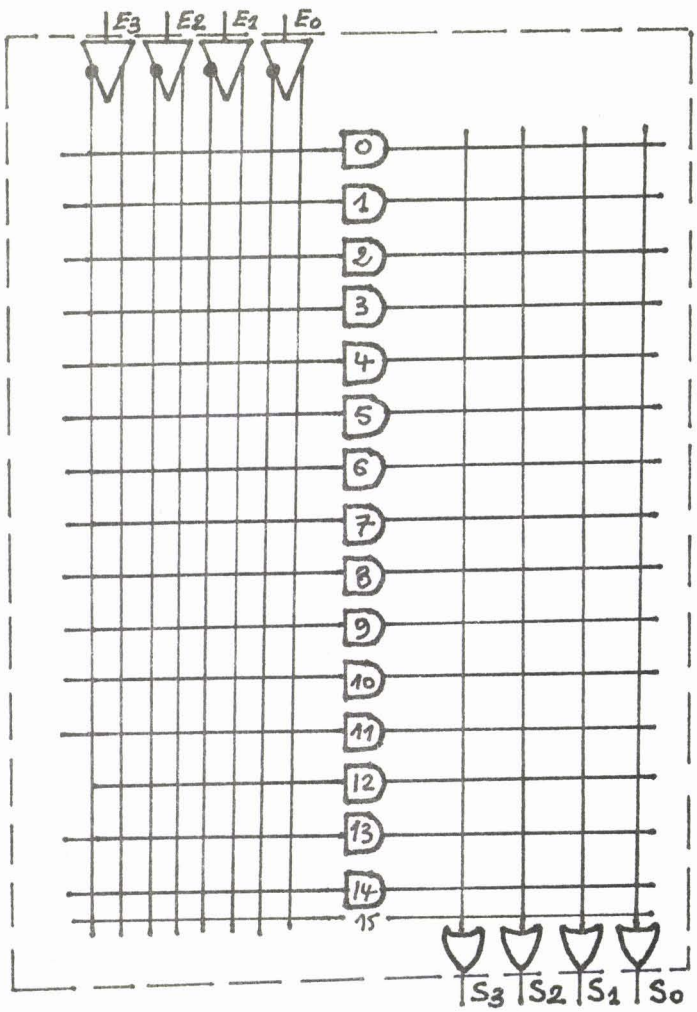
L'utilisation des circuits multiplexeurs, constitue une des implantations directes les plus courantes de la fonction combinatoire. Un exemple a été donné dans le paragraphe I.4.1 .

L'avènement de circuits complexes à haut degrés d'intégration tel que les réseaux logiques programmables (FPLA, PLA, PAL, ...) ont permis aussi un développement considérable des applications mettant en oeuvre des implantations directes de la fonction combinatoire. Leur emploi constitut une solution avantageuse qui se traduit par la facilité dans l'utilisation et par des performances au niveau du traitement assez intéressantes.

Pour illustrer l'utilisation de ces circuits dans l'implantation directe des fonctions combinatoires (pour plus de détail voir |TOU.80|),on considère le réseau logique représenté par la figure I.10a, sa structure interne est représentée par la figure I.10b.



(a)



(b)

Figure I.10

La mise en oeuvre de ce circuit permet de réaliser l'ensemble des fonctions définies sur 4 variables d'entrées, avec la possibilité d'évaluer plusieurs fonctions simultanées.

Ainsi par exemple les fonctions combinatoires :

$F_0 = (X_1,X_2,X_3,X_4) = X_1\,X_2 + X_1\,X_3 + X_2\,X_3$, $F_1(X) = X_1\,\overline{X}_4 + X_2 + \overline{X}_1\,\overline{X}_3$

$F_2 = (X) = \overline{X}_1\,\overline{X}_2 + X_3\,X_4$; $F_3\,(X) = X_1\,X_2\,X_3 + \overline{X}_2\,\overline{X}_4$

définies sur l'ensemble des variables $X = X_1, X_2, X_3, X_4$, peuvent être réalisées soit :

- par le schéma de la figure I.11a dans le cas de l'utilisation de circuits de mémorisation (POM, RAM, PROM).
- soit par le schéma de la figure I.11b dans le cas de l'utilisation d'un réseau logique programmable(PAL). Les réseaux logiques programmables tel que (PLA, FPLA) offrent encore plus de liberté dans la manoeuvre du fait des possibilités de programmation de leur circuits d'entrées et de sorties.

Lorsque des applications nécessitent un nombre important d'entrée/sortie, il est possible d'opérer des extensions par adjonction d'autres réseaux |TOU.80|

L'implantation directe des fonctions combinatoires constitue une solution intéressante si le nombre des entrées/sorties est peu important. Lorsque ce nombre devient relativement conséquent, l'implantation directe devient complexe. Dans ce cas une autre solution peut être envisagée. Il s'agit de l'élaboration de structures séquentielles à partir de ces circuits.

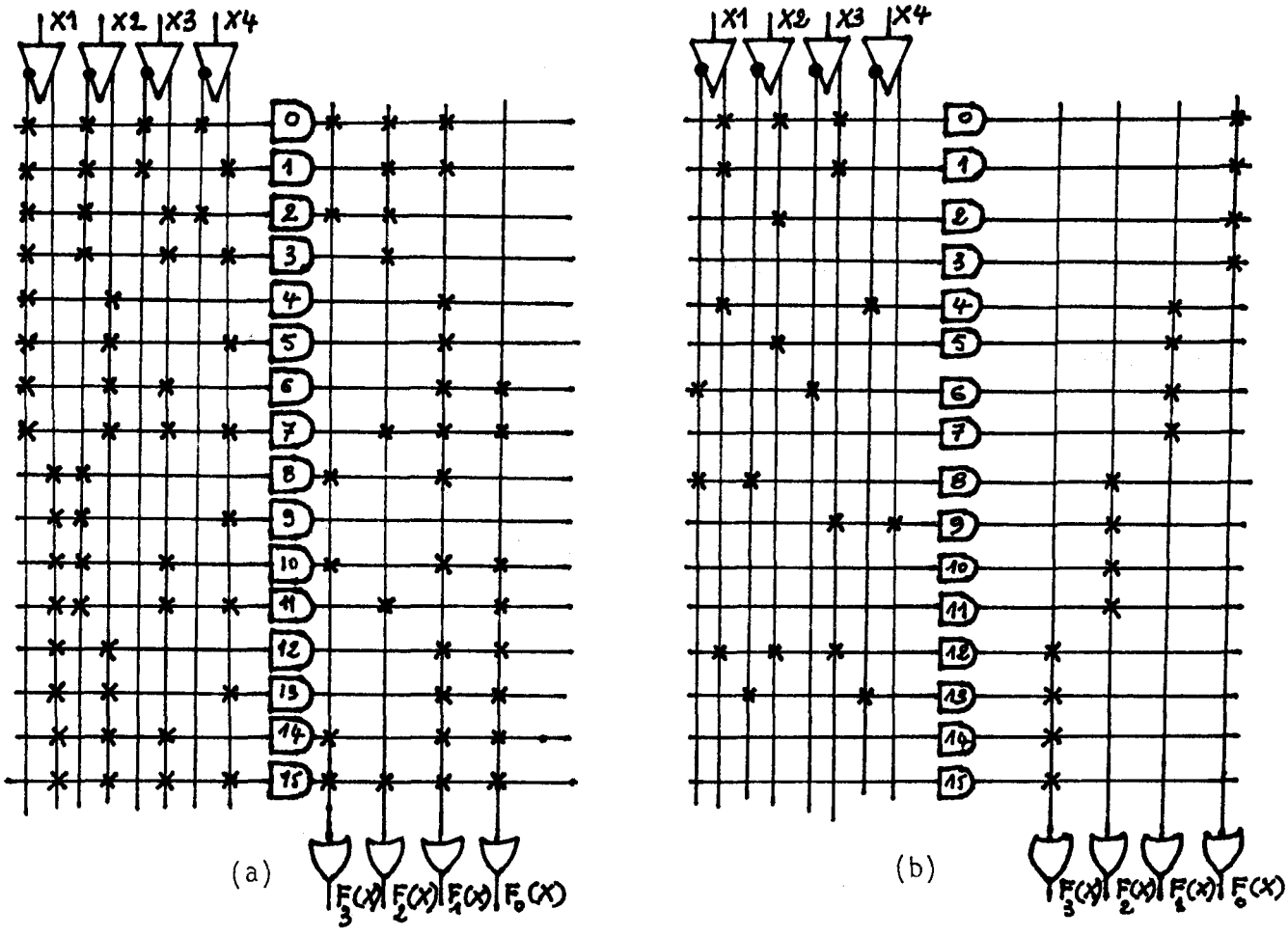


Figure I.11

I.5.1.2. - Construction de structures séquentielles

Nous avons vu au (I.4.1) que l'un des moyens d'évaluation de la fonction combinatoire est celui de construire l'arbre de décision binaire. Nous avons vu aussi que la matérialisation de cet arbre peut être envisagée par une implantation directe à base de circuit multiplexeur.

Il est possible aussi de construire une structure séquentielle à base des réseaux logiques où/et des circuits de mémorisation. La conception d'un processeur approprié à ce traitement sera fonction du choix du mode d'adressage et du jeu des instructions de la machine séquentielle ainsi élaborée.
(dans le cas de ce traitement deux instructions sont nécessaires : instruction de test et instruction de sortie).

La démarche à suivre pour établir cette structure séquentielle peut être décrite par les étapes suivantes.

- Recherche de l'arbre de décision binaire minimal d'évaluation
- Représentation tabulaire de l'arbre |POL.71|
- Codage du tableau
- Implantation de la machine séquentielle par la mise en oeuvre des circuits appropriés |TOU.80| |TOU.82|.

Le choix de la matérialisation de la fonction combinatoire par les systèmes câblés présente des avantages au niveau des performances (vitesse de traitement). Il présente aussi des inconvénients importants tel que : mauvaise adaptabilité à l'extension, une maintenabilité difficile. Cela fait, que ce type de matérialisation reste réserver à des applications spéciales.

Une matérialisation qui se développe de plus en plus dans le domaine des automatismes logiques car elle offre beaucoup plus de souplesse dans l'utilisation et l'exploitation de ces processus. Il s'agit de la matérialisation par des systèmes programmés.

I.5.2. - Réalisation de la fonction combinatoire par les systèmes programmés

Le développement des systèmes programmés et les avantages qu'ils offrent au niveau de leur exploitation, ont fait que les concepteurs de processus automatisés, s'orientent de plus en plus vers une synthèse matérielle programmée de ces processus.

La matérialisation de la fonction combinatoire, par l'importance qu'elle joue dans ces processus automatisés a bénéficié la première de cette utilisation des systèmes programmés.

Elle peut être implantée suivant deux procédures dans ces systèmes.

I.5.2.1. - Implantation "orientée donnée" de la fonction combinatoire

Par cette procédure, la fonction combinatoire est implantée sous forme d'une table de donnée qui est variable suivant la fonction combinatoire à traiter.

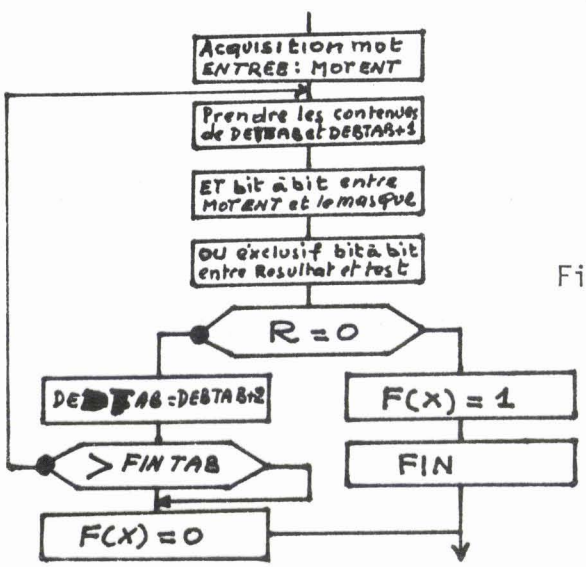
La gestion de cette table de donnée est assurée par un programme fixe et indépendant des données. Il est écrit dans le langage du système programmé choisi.

Cette procédure d'implantation offre beaucoup de souplesse dans l'utilisation : l'extension est facile (modification de la table de donnée); la maintenance est aisée. Parmi les inconvénients de cette procédure, il y a l'encombrement mémoire qui est relativement important. La vitesse de traitement est aussi relativement lente.

L'implantation de la procédure "orientée données" peut se faire, suivant plusieurs techniques :

- *le masquage*
- *l'addressage*
- *la simulation*

La figure I.12 représente l'implantation "orientée données" de la fonction combinatoire suivant la méthode par masquage.



. Organigramme de gestion

Figure I.12

DEBTAB	1	1	0	0	0	0	0	0
	1	1	0	0	0	0	0	0
	1	0	1	0	0	0	0	0
	1	0	1	0	0	0	0	0
	0	1	1	0	0	0	0	0
FINTAB	0	1	1	0	0	0	0	0

MASQUE1 (1 1 0 0 0 0 0 0)
TEST1 (1 1 0 0 0 0 0 0)

Table de données
de $F(X) = X_1X_2 + X_1X_3 + X_2X_3$

I.5.2.2. - Implantation "orientée programme" de la fonction combinatoire

Cette implantation est possible sur des systèmes programmés performants et disposant d'instruction de branchement. La fonction combinatoire est implantée sous forme d'un programme, l'espace mémoire nécessaire à l'implantation dépend de la fonction combinatoire à traiter. Toute extension ou modification de la fonction combinatoire passe par une refonte totale du programme.

Cette procédure d'implantation est performantecar elle offre un espace mémoire réduit et une vitesse de traitement rapide. La maintenance est plus difficile.

Malgrès l'intérêt qu'offre l'utilisation des systèmes programmés universels dans le traitement de la fonction combinatoire. Ceux-ci restent inadaptés à ce genre de traitement. Un besoin pressant s'est vite ressenti pour la conception de systèmes programmés et de langages appropriés, pour la mise en oeuvre de la fonction combinatoire en particulier et des systèmes logiques en général. Les automates programmables dès leur apparition sur le marché ont répondu à ce besoin et aux exigences dictées par les impératifs industriels dans le domaine de la logique industrielle.

I.6 - CONCLUSIONS

Les recherches sur l'évaluation de la fonction combinatoire sont très abondantes. Elles reflètent l'importance que celle-ci occupe dans la synthèse des automatismes industriels.

L'objectif de ces recherches est de développer des méthodes d'évaluation performantes. Il s'agit de minimiser l'espace mémoire occupé par la fonction combinatoire et le temps de traitement nécessaire à l'assignation de la fonction combinatoire.

Dans ce chapitre, nous avons présenté deux types de méthodes d'évaluation de la fonction combinatoire. Elles représentent les deux principales orientations des recherches dans ce domaine. Il s'agit :

. les méthodes d'évaluation séquentielles qui sont appelées à se développer dans le futur. Jusqu'à présent, les recherches entreprises dans cette direction, restent timides et insuffisantes.

. des méthodes d'évaluation par arbre de décision binaire. L'arbre de décision binaire est un moyen très efficace pour faire l'évaluation. Sa mise en oeuvre est facile et pratique. De nombreuses méthodes d'évaluation par arbre de décision binaire ont été mis au point. Elles utilisent différents critère de décision pour choisir les tests associés aux noeuds intermédiaires de l'arbre.

Dans le chapitre suivant nous étudierons une nouvelle approche pour construire l'arbre de décision binaire d'évaluation. En effet nous considérerons le problème de l'évaluation de la fonction combinatoire comme un problème de classification. Nous utiliserons la théorie de l'information pour développer un critère de décision entropique. Sa mise en oeuvre permettra de choisir le meilleur test au niveau de chaque noeud intermédiaire de l'arbre d'évaluation.

CHAPITRE II

DÉFINITION ET ÉTUDE D'UN CRITÈRE DE CHOIX DES FONCTIONS DE TEST

CHAPITRE II

DÉFINITION ET ÉTUDE D'UN CRITÈRE DE CHOIX DES FONCTIONS DE TEST

CHAPITRE II

DÉFINITION ET ÉTUDE D'UN CRITÈRE DE CHOIX DES FONCTIONS DE TEST

Nous avons présenté ci-dessus différentes méthodes d'évaluation de la fonction combinatoire dans lesquelles divers procédés ont été exploités (ensemble standard, forme paramètrique etc ...).

Dans l'étude qui va suivre, nous présenterons une nouvelle approche de l'évaluation de la fonction combinatoire. Elle consiste à considérer cette évaluation comme un problème de classification.

En effet, considérons une fonction combinatoire F(X) définie sur X l'ensemble des variables booléennes d'entrée par :

$$\begin{cases} Y = \{ y_1, y_2, \ldots, y_m \} \; ; \; m = |Y| \\ P_F(Y) = \{P_\beta, \; \beta \epsilon B_2\} \; , \; P_\beta = \{y \in Y / F(y) = \beta \epsilon B_2\} \end{cases} \qquad (II.1)$$

Dans ces conditions : assigner la fonction combinatoire F(X) à une valeur déterminée lors de l'apparition d'un élément $y \in Y$, reviendra à classifier l'élément $y \in Y$ dans l'une des classes P_β ($\beta \epsilon B_2$) de la partition $P_F(Y)$.

Cette procédure suppose l'existence d'une structure de classification appropriée et déterminée au préalable.

Nous choisirons comme structure de classification, l'arbre de décision binaire noté par la suite (ADB). Ce choix est motivé par les nombreux avantages qu'offre son utilisation au niveau de l'implantation matérielle.

La construction de cette structure de classification sera réalisée par la mise en oeuvre de la théorie de l'information de Claude E. SHANNON (1948), en utilisant la notion d'entropie.

Dans ce chapitre, après avoir rappelé quelques notions sur la théorie des probabilités, nous étudierons la structure de classification choisie et sa mise en oeuvre dans l'évaluation de la fonction combinatoire.

Nous rappelons aussi les notions essentielles du concept de la théorie de l'information. Nous les appliquerons ensuite pour définir et étudier un critère de choix des fonctions de test associées aux noeuds de l'ADB.

II.1 - ESTIMATION DES DISTRIBUTIONS DES PROBABILITES

Dans sa forme la plus générale [PARZ.60], une mesure de probabilité sur un ensemble Y fini est une fonction :

$$\begin{array}{ccc} P_r : = \mathcal{P}(Y) & \longrightarrow & R \\ P \in \mathcal{P}(Y) & & P_r(P) \in R \end{array} \qquad (II.2)$$

qui vérifie les propriétés suivantes :

i : $\forall P \in \mathcal{P}(Y) : P_r(P) > 0$

ii : Si $P_1, P_2, \ldots P_K$ sont K ensembles disjoints ($i \neq j : P_i \cap P_j = \emptyset$) on a :

$$P_r\left(\bigcup_{i=1}^{K} P_i\right) = \sum_{i=1}^{K} P_r(P_i)$$

iii : $P_r(Y) = 1$

Ainsi, si on reprend l'application (I.1a) définissant la fonction combinatoire F(X) soit :

$$F : = Y \longrightarrow B_2$$

on définit pour $\beta \in B_2$

$$P_r(F=\beta,\ \beta \in B_2) = P_r(\{y \in Y/F(y) = \beta \in B_2\}$$

Toujours pour Y fini, la définition (II.2) de probabilité, revient à donner un poids pj à chaque yjε Y, avec :

$$\forall_j;\ j = 1,2,\ldots,m \quad p_j > 0 \text{ et } \sum_{j=1}^{m} p_j = 1$$

On définit alors pour tout P $\varepsilon \mathcal{P}$ (Y)

$$P_r(P) = \sum_{j:yj\varepsilon P} p_j$$

En général, les variables yj ε Y ont toutes la même importance et sont de ce fait affectées de poids égaux, c'est-à-dire :

$$\forall j : pj = \frac{1}{|Y|} = \frac{1}{m} \text{ et ainsi}$$

$$P_r(P) = \sum_{j:yj\varepsilon P} p_j = \sum_{j:yj\varepsilon P} \frac{1}{m} = \frac{|P|}{m}$$

Cela constitue une estimation des probabilités par les fréquences relatives. Le fait de considérer les éléments yj ε Y équiprobables, à défaut d'une connaissance précise des probabilités d'occurence des yj ε Y constitue une hypothèse raisonnable, elle correspond au cas dans lequel la connaissance initiale que nous avons du système est la plus faible. Dans le cadre de notre étude, nous adapterons cette hypothèse. Par conséquent et en considérant la représentation de la fonction combinatoire F(X). (II.1)

$$\forall P_\beta, \beta\varepsilon B_2,\ P_\beta \in P_F : P_\beta = \{y \in Y/F(y) = \beta \in B_2\}$$

Comme par hypothèse on a :

$$\forall_j ,\ j=1,2,\ldots,m \quad pj = \frac{1}{m}$$

La probabilité pour qu'un élément yj ε Y appartienne à la classe $P_\beta \in P_F$ de Y, ou encore la probabilité pour que la fonction F(X) ait la modalité $\beta \varepsilon B_2$ sera :

$$p_\beta = P_r\ (P_\beta, \beta\varepsilon B_2) = \frac{|P_\beta|}{m}$$

II.2 - DESCRIPTION DE LA STRUCTURE DE CLASSIFICATION :

Arbre de décision binaire.

L'arbre de décision constitue l'un des moyens les plus utilisés dans l'évaluation de la fonction combinatoire | MAN.78, AKE.77 |.

Il est aussi utilisé dans beaucoup de domaine tel que la reconnaissance des formes | SET.82 |, le test et diagnostic des systèmes | VAR.82 | etc ...

Avant d'aborder la construction de l'arbre de décision binaire, nous discuterons auparavant :

- la description de la structure de l'arbre, nous rappelerons à cet effet quelques éléments théoriques sur la définition de l'ADB.
- sa mise en oeuvre dans l'évaluation de la fonction combinatoire.

II.2.1. - Description de la structure de classification

On considère le triplet $\Psi = (Y, N, T)$ dans lequel :

$Y = \{ y_1, y_2, \ldots, y_m \}$ représente l'ensemble des variables produits logiques défini sur X l'ensemble des variables booléennes.

$N = \{n_1, n_2, \ldots, n_k\}$ représente l'ensemble des noeuds intermédiaires de Ψ

On définit un noeud intermédiaire $n_\lambda \in N$ par la paire :

$$n_\lambda = (f_\lambda, \Pi_\lambda) \text{ ou } \lambda = 1,2,\ldots,k$$

$$\Pi_\lambda = \{y \in Y\} \; ; \; \Pi_\lambda \subseteq Y$$

f_λ : la fonction de test définit sur un sous ensemble de X et associée à $n_\lambda \in N$.

REMARQUE : la fonction f_λ opère une partition de l'ensemble Π_λ notée :

$$P_{f_\lambda}(\Pi_\lambda) = \{P_\alpha, \alpha \in B_2\} \; ; \; P_\alpha = \{y \in \Pi_\lambda \subseteq Y / f_\lambda(y) = \alpha \in B_2\}$$

$T = \{t_1, t_2, \ldots, t_l\}$ représente l'ensemble des noeuds terminaux de Ψ.
On définit un noeud terminal $t_i \in T$ par la paire :

$$\forall t_i \in T; \; t_i = (\beta, \Pi_i) \; , \; i=1,2,\ldots,l$$

dans laquelle $\beta \in B_2$ représente l'assignation de la fonction F(X) tel que :

$$\forall \Pi_i, \; i=1,2,\ldots,l \quad \Pi_i = \{y \in Y / F(y) = \beta \in B_2\}$$

On a aussi :

$$* \; \Pi_i \subset P_\beta \in P_F \; , \; \beta \in B_2$$

$$* \; \bigcup_{i=1}^{l} \Pi_i = Y \; , \; \forall \Pi_i, \; i=1,2,\ldots,l \; ; \; |\Pi_i| \geq 1$$

Le triplet $\Psi = (Y,N,T)$ est un arbre de décision binaire si les conditions suivantes sont remplies :

i : il existe un noeud $n_r \in N$ et un seul qui n'a pas de noeuds ascendants. n_r est le noeud racine de l'arbre Ψ

ii : soit n_λ un noeud intermédiaire de Ψ tel que :

$$\forall n_\lambda, \; \lambda = 1,2,\ldots,k, n_\lambda \in N / n_\lambda = (f_\lambda, \Pi_\lambda)$$

si il existe $n_{\lambda 1}, n_{\lambda 2} \in N$ (ou $t_i, t_j \in T$)

$$n_{\lambda 1} = (f_{\lambda 1}, \Pi_{\lambda 1}) \text{ et } n_{\lambda 2} = (f_{\lambda 2}, \Pi_{\lambda 2}) \text{ tel que :}$$

$$* \; \bigcup_{i=1}^{2} (\Pi_{\lambda i}) = \Pi_\lambda \text{ ou encore } \Pi_{\lambda 1}, \Pi_{\lambda 2} \in P_{f_\lambda}$$

Alors les noeuds $n_{\lambda 1}, n_{\lambda 2}$ sont les noeuds successeurs de $n_\lambda \in N$ ou encore $n_\lambda \in N$ est le noeud ascendant des noeuds $n_{\lambda 1}, n_{\lambda 2}$.

iii : si $\nexists\, t_i \in T$: * t_i n'a pas de noeuds successeurs

* t_i est un point d'assignation de la fonction combinatoire.

REMARQUE

Si Ψ est un arbre de décision binaire, alors on a la relation

$$l = k + 1 \quad \text{avec } l = |T| \text{ et } k = |N|$$

II.2.2. - Mise en oeuvre de la structure de classification dans l'évaluation de F(X)

On suppose que l'arbre de décision binaire Ψ défini ci-dessus a été construit. On considère un élément $y^* \in Y$.

L'unique séquence de test pour l'élément y^* dans l'arbre Ψ, symbolisée par $\mathcal{J}_\Psi(y^*)$ est un ensemble ordonné d'éléments de $\Psi = (Y,N,T)$. Elle est décrite par les étapes suivantes :

1) le premier élément de $\mathcal{J}_\Psi(y^*)$ est le noeud racine n_r de Ψ.

2) soit $n_j = (f_j,\Pi_j)$; $y^* \in \Pi_j$ le $j^{\text{ième}}$ élément de $\mathcal{J}_\Psi(y^*)$ et soit $n_\lambda = (f_\lambda, \Pi_\lambda)$ un élément descendant de n_j et qui est aussi un élément de Ψ. Alors n_λ est choisi comme le $j+1^{\text{ième}}$ élément de $\mathcal{J}_\Psi(y^*)$ si f_j affecte y^* dans $\Pi_\lambda \subset \Pi_j$ et que $\lambda \notin \{1,2,\ldots,l\}$ sinon la séquence de classification $\mathcal{J}_\Psi(y^*)$ de l'élément $y^* \in Y$ est terminée au noeud n_j qui est alors un noeud terminal. La fonction combinatoire F(X) est alors assignée à la valeur caractéristique du noeud.

II.2.3. - Construction de la structure de classification

Construire l'arbre de décision binaire de classification, c'est choisir la fonction de test au niveau de chaque noeud $n_\lambda \in N$ de Ψ en commençant par le noeud racine de Ψ. Dans ce choix des fonctions de test, on cherchera à minimiser le nombre de tests qui ménera à l'assignation de la fonction combinatoire. Cela constitue l'objectif que nous nous sommes fixés pour améliorer les performances de l'évaluation de la fonction combinatoire.

Il reste à construire l'ensemble des fonctions de test qui sera associé à l'ADB Ψ d'une part, d'autre part, il faut opérer le choix de la meilleure fonction de test parmi cet ensemble au niveau de chaque noeud intermédiaire. Ce choix est indispensable pour aboutir à l'objectif qu'on s'est fixé. Le critère de ce choix sera discuté plus loin, en attendant, nous étudions la construction de l'ensemble des fonctions de test qui sera associé à l'ADB Ψ.

II.2.3.1. - Définition et construction des fonctions de test

Soit $X = \{ X_1, X_2, \ldots X_n \}$ l'ensemble des variables booléennes d'entrée. Pour construire l'ensemble des fonctions de test associé à l'arbre Ψ, on doit choisir :

- le nombre w des variables booléennes qui constituent le test
- et construire l'ensemble des fonctions défini pour les w variables booléennes.

On considère alors :

Z un sous-ensemble de X de w éléments ($w = |Z| < n$)

$Z \in \mathcal{Z} = \{Z_1, Z_2, \ldots, Z_M\}$; Z est un sous-ensemble de variables booléennes, constitué d'une combinaison w parmi les n variables booléennes d'entrée de l'ensemble X.

$$M = |\mathcal{Z}| = C_w^n = \frac{n\ !}{w!\ (n-w)!}$$

Sur l'ensemble $Z \in \mathcal{Z}$, on définit une fonction de test par l'application :

$$f : = B_2^W \longrightarrow B_2 \qquad ; |B_2| = 2$$

$$\hat{y} \qquad\qquad f(\hat{y}) = \alpha \in B_2 \qquad\qquad (II.3)$$

$\hat{y}$ représente la restriction de la variable $y \in Y$ au seuls digits correspondants aux variables de l'ensemble $Z \in \mathcal{Z}$.

Il existe 2^{2^W} applications f | LEE.76 | définies sur tout ensemble $Z \in \mathcal{Z}$. Sur l'ensemble $\mathcal{Z}$ on construit donc l'ensemble des fonctions de test :

$$\mathcal{F} = \{ f_1, f_2, \ldots, f_\Omega \} \; ; \; \Omega = M.2^{2^W}$$

Toutes les fonctions de test de l'ensemble $\mathcal{F}$ ne seront pas utilisées pour construire l'arbre de décision Ψ.

On choisira seulement un sous-ensemble $\mathcal{F}^* \subset \mathcal{F}$. Toute fonction de test $f \in \mathcal{F}^*$ doit avoir la propriété suivante :

- soit un noeud intermédiaire $n_\lambda \in N$, $\lambda = 1,2,\ldots, K$

  $n_\lambda = (f_\lambda, \Pi_\lambda)$, f_λ est la fonction de test choisie et associée à n_λ lors de la construction de Ψ ; $f_\lambda \in \mathcal{F}^*$

Sur l'ensemble Π_λ toute fonction $f \in \mathcal{F}^*$ doit opérer une partition distincte de cet ensemble autrement dit :

$$\forall f_i, f_j \; , \; i \neq j \; ; \; f_i, f_j \in \mathcal{F}^* \text{ alors}$$

$$p_{f_i}(\Pi_\lambda) \neq P_{f_j}(\Pi_\lambda) \quad \forall_i, \forall_j \; . \; i \neq j \qquad (II.4)$$

Le choix de l'ensemble $\mathcal{F}^*$ des fonctions de test, permet de réduire le temps de traitement lors de la construction de l'arbre de décision Ψ. L'ensemble $\mathcal{F}^*$ est construit par :

- le choix de w
- la mise en oeuvre de la propriété (II.4).

prenons, un exemple pour illustrer ce qui vient d'être dit.

EXEMPLE 1 on veut construire un ensemble de fonctions de test $\mathcal{F}^*$ dont lequel chaque test est effectué sur une seule variable booléenne. Alors w = 1.

Dans ce cas, on a :

$$\mathcal{Z} = \{ X_1, X_2, X_3, \ldots, X_n \} \ , \ \forall Z \in \mathcal{Z}, \ Z = \{X_i\} \ , \ i=1,2,\ldots$$

$$\mathcal{Z} = X.$$

dénombrons les 2^{2^w} (4) applications f (II.3), représentées par la figure II.1.

X_i \ f	0	$\overline{X_i}$	X_i	1
0	0	1	0	1
1	0	0	1	1

Figure II.1

, parmi les 2^{2^w} applications, on élimine les 2 applications triviales 0 et 1. Il reste : $f(Z) = X_i$ et $f(Z) = \overline{X}_i$

Autrement l'ensemble $\mathcal{F}^*$ des fonctions de test est constitué pour w = 1 par :

$$\mathcal{F} = \{ X_1, \overline{X_1}, X_2 , \overline{X_2}, \ldots, \overline{X}_n , X_n \}$$

Cet ensemble peut être réduit par la relation (II.4), on obtient alors l'ensemble $\mathcal{F}^*$ de fonctions de test qui sera associé à l'arbre Ψ.

$$\mathcal{F}^* = \{ X_1, X_2, \ldots, X_n \}$$

EXEMPLE 2 on veut construire des fonctions de test sur deux variables booléennes prises parmi l'ensemble X.

On a alors : w = 2 et Z = { X_i, X_j } ; i,j = 1,2,...,n ; $i \neq j$; j > i

et l'ensemble $\mathcal{Z}$ = { $\{X_1, X_2\}$, $\{X_1, X_3\}$, ..., $\{X_{n-1}, X_n\}$ }

Sur w = 2, il y a 16 applications f (II.3), elles sont représentées par le tableau de la figure II.2 .

X_i X_j \ f	$\overline{X_i}.\overline{X_j}$	$\overline{X_i}X_j$	$X_i\overline{X_j}$	X_iX_j	$\overline{X_i}$	$\overline{X_j}$	$X_i \oplus X_j$	$X_i \odot X_j$	X_i	X_j	$\overline{X_i}+\overline{X_j}$	X_i+X_j	$\overline{X_i}+X_j$	$X_i+\overline{X_j}$	0	1
0 0	1	0	0	0	1	1	1	0	0	0	1	0	1	1	0	1
0 1	0	1	0	0	1	0	0	1	1	0	1	1	1	0	0	1
1 0	0	0	1	0	0	1	0	1	0	1	1	1	0	1	0	1
1 1	0	0	0	1	0	0	1	0	1	1	0	1	1	1	0	1

Figure II.2

L'ensemble des fonctions de test est :

$$\mathcal{F} = \{ \bigcup_{i,j;j>i} \{\overline{X}_i\overline{X}_j, \overline{X}_iX_j, X_i\overline{X}_j, X_iX_j, X_i \oplus X_j, X_i \odot X_j, \overline{X}_i+\overline{X}_j, \overline{X}_i+X_j, X_i+X_j, X_i+\overline{X}_j\} \}$$

et l'application de la règle (II.4) , nous permet de déterminer l'ensemble des fonctions de test associé à l'arbre Ψ.

$$\mathcal{F}^* = \{ \bigcup_{i,j;j>i} \{\overline{X}_i X_j, X_i \overline{X}_j, \overline{X}_i \overline{X}_j, X_i X_j, X_i \oplus X_j\} \} \; ; \; X_i, \; X_j \in X$$

REMARQUE

La détermination de l'ensemble $\mathcal{F}^*$, devient très vite difficile à faire, avec le choix de w le nombre des variables de test. Ainsi pour w = 3 il y a 256 applications f et c'est déjà beaucoup.

Après la construction de l'ensemble $\mathcal{F}^*$ des fonctions de test, il reste à associer à chaque noeud intermédiaire $n_\lambda \in N$ de l'arbre Ψ, une fonction $f \in \mathcal{F}^*$.

Le choix de la fonction de test $f_\lambda \in \mathcal{F}^*$ associée à n_λ, sera réalisé par comparaison des différentes partitions de Π_λ opérées par les fonctions de test de l'ensemble $\mathcal{F}^*$ et la mise en oeuvre d'un critère de choix développé par application du concept de la théorie de l'information.

II.3 - CHOIX DES FONCTIONS DE TEST : critère entropique de choix

II.3.1. - Concept de la théorie de l'information

La théorie de l'information est née en 1948 avec les travaux de Claude E. SHANNON et de Norbert WIENER. C'était une réponse puissante à un problème très concrêt : comment mesurer l'information et en assurer la transmission entre deux points (un émetteur et un récepteur à travers un canal soumis au bruit | ASH.65 |. Aujourd'hui, cette théorie, après un développement rapide et fécond, est utilisée dans différents domaines (communications, mathématiques, analyse de systèmes).

II.3.1.1. - Définition formelle de l'entropie

Soit l'ensemble Y défini auparavant, et Φ une variable définie par l'application :

$$\Phi := Y \longrightarrow B \text{ ; B ensemble des modalités de } \Phi$$
$$y \in Y \quad \Phi(y) = \beta \in B \qquad \text{(II.5)}$$

Du fait que chaque variable ϕ définie sur Y, donne lieu à une partition de Y, l'entropie peut être définie de deux manières équivalentes :

- Comme une fonction qui s'applique aux variables définies sur Y par l'application (II.5).
- Comme une fonction qui s'applique aux partitions de Y.

a/ Entropie d'une variable

On définit l'entropie de la variable Φ par :

$$H(\Phi) = - \sum_{\beta \in B} P_r(\Phi=\beta) \, \text{Log}_2 \, P_r(\Phi=\beta) \text{ bit} \quad \text{(II.6)}$$

ou $P_r(\Phi=\beta) = P_r(\{y \in Y / \Phi(y) = \beta \in B\})$, on pose $P_r(\Phi=\beta) = p\beta$

Avec $H(\Phi) = 0$ (par définition $0 \, \text{Log}_0 = 0$. Cela provient du fait que $\lim_{p \to 0^+} p \log p = 0$)

L'entropie est mesurée en bit. Un bit correspond à l'entropie d'une variable qui prendrait 2 modalités, chacune avec la probabilité 1/2.

b/ Entropie d'une partition de Y

On peut remarquer que l'entropie d'une variable Φ ne dépend pas explicitement des modalités qu'elle prend, mais seulement des probabilités avec lesquelles cela est fait. Autrement dit, elle dépend du nombre de $y \in Y$ pour lesquels $\Phi(y) = \beta$ et ceci pour chaque $\beta \in B$. Cela est justement la partition P_Φ de Y.

Dans ce cas l'entropie est définie comme la fonction :

$$H := \mathcal{P}(Y) \longrightarrow R \qquad (II.7)$$

$$P_\Phi = \{P_\beta, \beta\in B\} \qquad H(P_\Phi) = - \sum_{\beta\in B} P_r(P_\beta) \text{ Log } P_r(P_\beta)$$

$$P_\beta = \{ y \in Y / \Phi(y) = \beta\in B \}$$

Qu'il s'agisse donc de l'entropie de la variable Φ ou de la partition P_Φ de Y associée à Φ, cela est tout à fait équivalent.

Dans la suite de cet exposé, nous utilisons l'entropie d'une partition, nous considérons donc la définition (II.7).

II.3.1.2. - Quelques propriétés de l'entropie

Dans ce paragraphe, nous nous limitons à énoncer et à interpréter les propriétés de l'entropie pour toute démonstration voir bibliographie et notamment | ASH.65, GIU ..77 - TOR.82 | .

i : $\forall P \in \mathcal{P}(Y)$; $H(P) > 0$:

De plus $H(P) = 0$ implique que la variable associée à la partition P est une fonction constante.

Interprétation : l'information qu'une variable associée à une partition P est toujours non négative. Les seules variables qui ne fournissent aucune information sont les variables constantes 'elles prennent toujours la même modalité).

ii : soit une partition P de Y et soient p_β , $\beta\in B$ les probabilités des classes $P_\beta \in P$. Alors

$$H(P) = - \sum_{\beta\in B} p_\beta \text{ Log } p_\beta < \text{Log } |B| \text{ : avec égalité}$$

$$\forall_{\beta\in B}\, p_\beta = \frac{1}{|B|}$$

Interprétation : l'entropie d'une partition est maximale lorsque toutes ses classes sont équiprobables.

iii : soient P_{Φ_1} et P_{Φ_2} deux partitions de Y. Alors

$$H(P_{\Phi_1} \cup P_{\Phi_2}) \leq H(P_{\Phi_1}) + H(P_{\Phi_2})$$

avec égalité ssi P_{Φ_1} et P_{Φ_2} sont statistiquement indépendants.

iiii : soient P_{ϕ_1} une partition de K_1 classes et P_{Φ_2} une partition de K_2 classes.

Si $P_{\Phi_1} < P_{\Phi_2}$ alors $H(P_{\Phi_1}) > H(P_{\Phi_2})$.

Le fait que $P_{\Phi_1} < P_{\Phi_2}$ implique que chaque classe de P_{Φ_1} est contenue dans une classe de P_{Φ_2}. Chaque classe de P_{Φ_2} est justement l'union de classes de P_{Φ_1} qu'elle contient.

II.3.1.3. - Lois conjointes

Considérons deux partitions de l'ensemble Y.

La partition P_A (dont les classes sont notées : $A_1, A_2, \ldots A_{K1}$) et la partition P_B (dont les classes sont notées : $B_1, B_2, \ldots B_{K2}$).

On notera aussi les probabilités correspondants à chaque une des classes :

$$p_i = P_r(A_i) \quad , \; i = 1,2,\ldots, K_1$$

$$p_j = P_r(B_j) \quad ; \; j = 1,2,\ldots, K_2$$

$$p_{ij} = P_r(A_i \cap B_j)$$

L'information moyenne apportée par la partition P_A vaut :

$$H(P_A) = - \sum_{i=1}^{K_1} p_i \text{ Log } p_i$$

de même

$$H(P_B) = - \sum_{j=1}^{K_2} p_j \text{ Log } p_j$$

L'information moyenne relative à la connaissance simultanée de P_A et P_B vaut :

$$H(P_A,P_B) = - \sum_{i,j} p_{ij} \text{ Log } p_{ij} \qquad (II.8)$$

et elle est appelée : information mutuelle ou (entropie mutuelle)

Si l'information telle qu'on la définit en mathématique, correspond bien à l'idée qu'on ce fait de la notion d'information, on doit savoir :

$$H(P_A,P_B) < H(P_A) + H(P_B) \qquad (II.9)$$

En effet, mathématiquement, on a bien l'inégalité ci-dessus, que les propriétés de concavité de la fonction X Log X permettent de démontrer | SHA.48 | , | GIU .71 |.

La quantité

$$H(P_B/P_A) = H(P_A,P_B) - H(P_A) \qquad (II.10)$$

est appelée information de la partition P_B conditionnée par la partition P_A ou information supplémentaire apportée par P_B quand P_A est réalisée.

REMARQUE : d'après la relation (II.10), on a :

$$H(P_B/P_A) < H(P_B) \qquad (II.11)$$

Vérifions que cette définition (II.10) est cohérente avec la définition des probabilités conditionnelles :

$$P_r(B_j/A_j) = P_r(B_j \cap A_j) \;/\; P_r(A_i) = p_{ij}/p_i$$

dont

$$H(P_B/A_i) = - \sum_j (p_{ij}/p_i) \text{ Log } (p_{ij}/p_i) \qquad \text{(II.12)}$$

L'information supplémentaire conditionnelle apportée par P_B à partir de P_A, n'est autre que l'espérance mathématique de la relation (II.12). En effet :

$$\sum_i p_i \, H(P_B/A_i) = - \sum_i \sum_j p_{ij} \text{ Log } (p_{ij}/p_i)$$

$$= - \sum_i \sum_j p_{ij} \mid \text{Log } p_{ij} - \text{Log } p_i \mid$$

$$= - \sum_i \sum_j p_{ij} \text{ Log } p_{ij} + \sum_i p_i \text{ Log } p_i$$

$$= H(P_B, P_A) - H(P_A)$$

dont $H(P_B/P_A) = - \sum_i \sum_j p_{ij} \text{ Log } (p_{ij}/p_i)$ (II.13)

Nous allons appliquer ces différentes quantités, pour étudier un critère de choix des fonctions de test associées aux noeuds intermédiaires de l'arbre de décision Ψ d'évaluation de la fonction combinatoire.

II.3.2. - Information conditionnelle appliquée à la comparaison entre partitions : critère entropique de choix des fonctions de test

Nous avons représenté ci-dessus (II.1), la fonction combinatoire par :

$$\begin{cases} Y = \{ y_1, y_2, \ldots, y_m \} \\ P_F = \{ P_\beta \,, \beta \in B_2 \} \; ; \; P_\beta = \{ y \in Y / F(y) = \beta \in B_2 \} \end{cases}$$

Sur l'ensemble X des variables booléennes d'entrée, et par le choix de la valeur de w; nombre des variables constituants le test et par la mise en oeuvre de la propriété (II.4), nous avons déterminé et construit l'ensemble des fonctions de test qui sera associé à l'arbre de décision Ψ.

Alors pour $\Psi = (Y, N, T)$ et

$$\forall\, n_\lambda \in N,\ \lambda = 1,2,\ldots k \quad , \quad n_\lambda = (f_\lambda\ ,\ \Pi_\lambda\ )$$

avec $\forall\ f_\lambda \in \mathcal{F}^*$

f_λ : fonction de test associée à n_λ et déterminée à postériori. Justement, il reste à déterminer la fonction $f_\lambda \in \mathcal{F}^*$ qu'il faut associée à $n_\lambda \in N$.

Pour faire le choix de f_λ , il faut évaluer l'information apportée par la partition de l'ensemble Π_λ (restriction de l'ensemble Y et caractéristique du noeud n_λ), sur la partition $P_F(Y)$ (caractéristique de la fonction combinatoire à évaluer) et cela pour chaque fonction de test $f \in \mathcal{F}^*$.

II.3.2.1. - Evaluation de l'information apportée par la partition opérée par une fonction de test, sur la valeur à donner à F(X)

On considère donc une fonction de test $f \in \mathcal{F}^*$ et un noeud

$n_\lambda \in N$ tel que : $n_\lambda = (f_\lambda, \Pi_\lambda)$ dans ces conditions :

1) $f \in \mathcal{F}^*$ opère une partition de Π_λ . Soit :

$$P_f = \{P_\alpha, \alpha \in B_2\} \ ; \ P_\alpha = \{y \in \Pi_\lambda \ c\ Y / f(y) = \alpha \in B_2\}$$

2) F(X) opère une partition de Π_λ , notée $P_{F,\lambda}$, soit :

$$P_{F,\lambda} = \{P_{\beta,\lambda},\ \beta \in B_2\} \ ;\ P_{\beta,\lambda} = \{y \in \Pi_\lambda \subset Y / F(y) = \beta \in B_2\}$$

on a les relations suivantes :

$$\forall_{\beta \in B_2} \ ; \forall_\lambda \ , \ \lambda = 1,2,\ldots,k \ ; \forall P_{\beta,\lambda} \in P_{F,\lambda} : \forall P_{\beta\varepsilon} P_F$$

$$P_{\beta,\lambda} \subset P_\beta .$$

$$P^\lambda_{\beta\alpha} = P_\alpha \cap P_{\beta,\lambda} \quad \forall_{\alpha,\beta \in B_2} \ , \forall_\lambda \quad \lambda = 1,2,\ldots,k$$

et tel que

$$\bigcup_{\beta \in B_2} P^\lambda_{\beta\alpha} = P_\alpha$$

3) Suivant l'hypothèse sur les probabilités des éléments $y \in Y$, on considère les estimations des probabilités suivantes :

$$p\alpha = P_r\ (P\alpha, \alpha \in B_2) = \frac{|P\alpha|}{|\Pi_\lambda|} \qquad (a)$$

$$p\beta\alpha = P_r\ (P_\alpha \cap P_{\beta,\lambda}) = \frac{|P^\lambda_{\beta\alpha}|}{|\Pi_\lambda|} \qquad (b) \qquad (II.14)$$

$$p^\lambda_\beta = p_\beta = P_r(P_{\beta,\lambda} \quad \beta \in B_2) = \frac{|P_{\beta,\lambda}|}{|\Pi_\lambda|} \qquad (c)$$

Il est évident que l'on a les relations :

$$p\alpha = \sum_{\beta \in B_2} p_{\beta\alpha} \qquad (a)$$

$$p\beta = \sum_{\alpha \in B_2} p_{\beta\alpha} \qquad (b) \qquad (II.15)$$

$$p_{\beta,\alpha} = \frac{p_{\beta\alpha}}{p\alpha} \qquad (c)$$

et $\sum_{\alpha}\sum_{\beta} P_{\beta\alpha} = 1$ (d)

Dans ces conditions, on définit l'information apportée par P_f de Π_λ sur la connaissance de la partition $P_{F,\lambda}$ de Π_λ par :

$$H(P_{F,\lambda}/P_f) = - \sum_{\alpha\in B_2} \sum_{\beta\alpha B_2} p_{\beta\alpha} \text{ Log } (p_{\beta\alpha}/P_\alpha) \qquad (II.15)$$

La quantité (II.15) évalue la quantité d'information restant pour donner une valeur $\beta\in B_2$ à F(X), après avoir réalisé le test $f \in \mathfrak{F}^*$
En effet :

1) Si par l'application d'une fonction de test $f \in \mathfrak{F}^*$ au noeud $n_\lambda \in N$ chaque classe de la partition P_f de Π_λ opérée par f, affecte une valeur $\beta\in B_2$ à F(X). Autrement si :

$f \in \mathfrak{F}^*$ tel que :

$$P_f = P_{F,\lambda} \text{ soit } \forall P_\alpha \in P_f \text{ et } \forall P_{\beta,\lambda} \in P_{F,\lambda}$$

on a $P^\lambda_{\beta\alpha} = (P_\alpha \cap P_{\beta,\lambda}) = P_{\beta,\lambda} = P_\alpha$ Alors :

$$* \; H(P_{F,\lambda}/P_f) = 0$$

La démonstration est triviale puisqu'il suffit de remplacer dans l'expression (II.15) la quantité :

$$p_{\beta\alpha} \text{ par } p_\alpha \text{ puisque } P^\lambda_{\beta\alpha} = P_\alpha \quad \forall_{\alpha,\beta\in B_2} .$$

2) Si $\forall P_\alpha \in P_f$, $\alpha\in B_2$ et $\forall P_{\beta,\lambda} \in P_{F,\lambda}$ on a :

$$P^\lambda_{\beta\alpha} \subset P_\alpha \text{ alors :}$$

$$* \; H(P_{F,\lambda}/P_f) > 0$$

ceci veut dire, qu'il subsiste une incertitude sur la valeur à donner

F(X) après avoir appliqué la fonction de test $f \in \mathcal{F}^*$ au noeud n_λ. Démontrons cette nouvelle proposition et évaluons la quantité d'incertitude qui reste à lever pour donner une valeur à F(X).

$$H(P_{F,\lambda}/P_f) = - \sum_{\alpha \in B_2} \sum_{\beta \in B_2} p_{\beta\alpha} \text{ Log } (p_{\beta\alpha}/p_\alpha)$$

montrer que $H(P_{F,\lambda}/P_f) > 0$ ce qui est évident puisque l'on a :

$$p_{\beta\alpha} < p_\alpha \quad \forall_{\alpha,\beta \in B_2} \qquad (II.16)$$

Dans ce cas, la fonction de test $f \in \mathcal{F}^*$ est insuffisant pour permettre l'assignation de F(X) si elle est affectée au noeud n_λ ; par conséquent d'autres tests sont nécessaires pour arriver à cette fin.

Après avoir évalué la quantité d'information apportée par l'application de chaque fonction de test au noeud $n_\lambda \in N$, il est donc nécessaire de choisir la fonction de test f_λ qui sera associée à n_λ.

Ce choix sera réalisé par la mise en oeuvre d'un critère entropique basée sur la quantité (II.15).

II.3.2.2. - Critère entropique de choix des fonctions de test

Il s'agit de choisir la meilleure fonction de test parmi l'ensemble $\mathcal{F}^*$ qu'il faut associer à chaque noeud de l'arbre de décision Ψ. Le meilleur choix sera défini par la minimisation de la quantité (II.15). Ainsi

pour tout n_λ , $n_\lambda \in N$, $\lambda = 1,2,\ldots,k$.

$\forall f \in \mathcal{F}^*$ $\quad P_f$ de Π_λ et $P_{F,\lambda}$ de Π_λ.

On calcule $H(P_{F,\lambda}/P_f)$.

Alors on choisira $f_\lambda \in \mathcal{F}^*$ qui vérifie le critère suivant.

$$H(P_{F,\lambda}/P_{f_\lambda}) = \underset{f \in \mathcal{F}^*}{\text{Min}} H(P_{F,\lambda}/P_f) \qquad (II.17)$$

La mise en oeuvre du critère (II.17) nous permettra de construire l'arbre de décision binaire d'évaluation de la fonction combinatoire.

Dans le chapitre suivant, nous étudierons, l'algorithme construit autour de ce critère qui permettra la réalisation de l'ADB d'évaluation de la fonction combinatoire.

CHAPITRE III

CONSTRUCTION D'UN ARBRE DE DÉCISION BINAIRE REPRÉSENTANT UNE FONCTION COMBINATOIRE

CHAPITRE III

CONSTRUCTION D'UN ARBRE DE DÉCISION BINAIRE REPRÉSENTANT UNE FONCTION COMBINATOIRE

CHAPITRE III

CONSTRUCTION D'UN ARBRE DE DÉCISION BINAIRE REPRÉSENTANT UNE FONCTION COMBINATOIRE

Le but de l'utilisation d'une représentation sous forme d'un arbre de décision (A D B) étant une évaluation performante des fonctions combinatoires représentées, il est donc intéressant lors de la construction d'un A D B d'en choisir un qui minimise le critère de performance. Les deux éléments qui interviennent le plus souvent dans ce critère sont :

a/ la durée moyenne de l'évaluation

b/ l'occupation mémoire des programmes engendrés

Notre objectif dans cette étude est la minimisation de la durée maximum d'évaluation d'une fonction combinatoire.

Si on suppose que tous les tests sont équidurables, la minimisation de la durée maximum d'évaluation revient à minimiser le plus long chemin sur l'ADB représentant la fonction.

En ce qui concerne l'occupation mémoire nous avons constaté que la minimisation du plus long chemin donne presque toujours des ADB avec nombre de noeuds près du minimum.

Dans ce chapitre,nous présentons une méthode d'évaluation de la fonction combinatoire. Elle consiste à construire un arbre de décision binaire qui représente une fonction combinatoire donnée. Cette méthode est basée sur la mise en oeuvre du critère entropique de choix des tests étudié dans le chapitre précédent.

Nous étudions dans un premier temps un algorithme décrivant cette méthode. Nous nous intéressons en suite à l'implantation et au test de cette méthode. A cet effet, plusieurs exemples de fonctions combinatoires seront traités et les résultats obtenus commentés.

Nous discutons enfin de l'extension de la méthode à l'évaluation des fonctions combinatoires multi-valuées qui connaissent actuellement un développement considérable.

III.1 - DESCRIPTION DE LA METHODE D'EVALUATION PROPOSEE

La méthode d'évaluation que nous proposons est une méthode exhaustive. La mise en oeuvre de l'algorithme qui sera décrit ci-dessous permet d'obtenir un ADB représentant la fonction combinatoire considérée. Il procède par une analyse descendante : la construction de l'ADB commence par la construction du noeud racine. Les autres noeuds sont construits par l'algorithme suivant l'ordre décrit par le graphe de la figure III.1 .

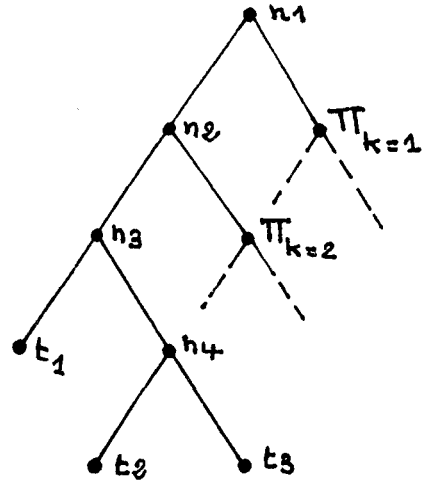


Figure III.1

Mise en oeuvre de l'algorithme (construction des noeuds dans l'ordre :

$n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow t_1 \rightarrow n_4 \rightarrow t_2 \rightarrow t_3 \ldots$)

Pour construire un arbre de décision binaire représentant une fonction combinatoire donnée, l'algorithme doit disposer :

a/ des données relatives à la fonction combinatoire F(X) à évaluer, à savoir :

- l'ensemble $X = \{X_1, X_2, \ldots, X_n\}$, $n = |X|$: de variables booléennes d'entrée.
- l'ensemble $B = \{ 0,1 \}$ des modalités.

. l'ensemble $Y = \{ y_1, y_2, \ldots y_n \}$; $m = |Y|$, des variables produits logiques.

. la partition caractéristique de F(X) notée :

$$P_F = \{ P_\beta, \beta \in B \} \; ; \; P_\beta = \{ y \in Y \; / \; F(y) = \beta \in B \}$$

b/ des données relatives aux fonctions de test à savoir :

. w, le nombre des variables booléennes d'entrée qui constituent les tests.

. l'ensemble des fonctions de test : $\mathcal{F}^* = \{ f_1^*, f_2^*, \ldots, f_L^* \}$

on a, $\forall f^* \in \mathcal{F}^*$ et $\forall y \in Y : f^*(\hat{y}) \in B$

$\hat{y}$ restriction de la variable produit logique $y \in Y$ aux seuls w digits correspondants aux variables booléennes constituantes de $f^* \in \mathcal{F}^*$.

c/ du critère en tropique de choix des fonctions de test (II.17) dont la mise en oeuvre permet d'associer à chaque noeud intermédiaire n_i, une fonction de test $f^* \in \mathcal{F}^*$.

Dans ces conditions la construction d'un ADB $\psi = (Y, N, T)$ représentant une fonction combinatoire F(X) donnée, consiste à construire chaque noeud intermédiaire:

$$n_i \in N \; , \; i = 1,2,\ldots k \quad n_i = (f_i, \Pi_i) \; f_i \in \mathcal{F}^*, \; \Pi_i \subseteq Y$$

et chaque noeud terminal :

$$t_\lambda \in T, \; \lambda = 1,2,\ldots, \mathcal{I} : \quad t_\lambda = (\beta \, , \Pi_\lambda) : \beta \in B, \; \Pi_\lambda \subset P_\beta \in P_F$$

III.1.1. - Algorithme de construction de l'ADB

L'algorithme est constitué de deux phases distinctes :

- *la phase d'acquisition des données nécessaires à la construction de l'ADB*
- *la phase traitement qui concerne la construction de l'ADB. Les étapes suivantes permettent la description de cet algorithme.*

A/ Phase d'acquisition des données

A.1 - Introduire l'ensemble $X = \{X_1, X_2, \ldots X_n\}$ des variables booléennes et n : $|X|$;

A.2 - Introduire l'ensemble des modalités : $B = \{0,1\}$

A.3 - Introduire la fonction combinatoire F(X);

a) générer l'ensemble $Y = \{y_1, y_2, /// y_m\}$; $m = |Y|$;

b) générer la partition caractéristique de la fonction combinatoire considérée :

$$P_F = \{P_\beta,\ \beta \in B\} \ ; \ P_\beta = \{ y \in Y / F(y) = \beta \in B\}$$

A.4 - Construction de l'ensemble des fonctions de test :

a) introduire w nombre des variables des tests

b) construire l'ensemble $\mathcal{F}^* = \{ f_1^*, f_2^*, \ldots, f_2^* \}$; $L = |\mathcal{F}^*|$;

B/ Phase de construction de l'ADB ψ

B.1 - Initialisation : i=1; $\Pi_i \leftarrow Y$; K=1 ; $\lambda = 0$

B.2 Pour toute fonction $f_j^* \in \mathcal{F}^*$, j = 1,2,..., L

a) construire les partitions $P_{f_j^*}$ et $P_{F,i}$ de Π_i :

$$P_{f_j^*} = \{P_\alpha, \alpha \varepsilon B\} \ ; \ P_\alpha = \{y \in \Pi_i \subseteq Y / f_j^* (\hat{y}) = \alpha \varepsilon B\}$$

$$P_{F,i} = \{P_{\beta,i}, \beta \varepsilon B\} \ ; \ P_{\beta,i} = \{y \in \Pi_i \subseteq Y / F(y) = \beta \varepsilon B\}$$

b) détermination des probabilités des sous classes $P^{i}_{\beta\alpha} = P_{\alpha} \cap P_{\beta,i}$

$$p_{\beta\alpha} = \frac{|P_{\alpha} \cap P_{\beta,i}|}{|\Pi_i|} \; ; \; p_{\alpha} = \frac{|P_{\alpha}|}{|\Pi_i|} \qquad \forall \beta, \alpha \in B$$

c) évaluation de la quantité d'information apportée par la partition

$P_{f^*_j}$ sur la valeur à donner à F(X) : soit

$$H_j \, (P_{F,i}/P_{f^*_j})$$

d) choix de la fonction de test associée au noeud $n_i \in N$: on calcule :

* $H_i \, (P_{F,i}/P_{f_j}) = \underset{j=1,2,..L}{\text{Min}} \; H_j \, (P_{F,i}/P_{f^*_j}) : f_j \in \mathcal{F}^*$

* $f_i = f_j$

B.3 - Recherche des noeuds terminaux : Pour $H_i \, (P_{F,i}/P_{f_j}) = 0$

a) $\forall \alpha, \beta \in B : \lambda = \lambda + 1$

$$P_{\alpha} \subset P_{\beta} \in P_F \; : \; t_{\lambda} = (\beta, P_{\alpha})$$

b) $K = K - 1$

c) Pour K = 0 : la construction de l'ADB Ψ est terminée : Edition des résultats : FIN

d) Pour K > 0 : i = i+1 : $\Pi_i = \Pi_K$: Aller à (B2)

B.4 - Pour $H_i \, (P_{F,i}/P_{f_j}) > 0$

a) Pour $P_{\alpha=1} \subset P_\beta \in P_F$, $\beta \in B$: $\lambda = \lambda+1$: $t_\lambda = (\beta, P_{\alpha=1})$: i=i+1:

$\Pi_i = P_{\alpha=0}$: Aller à (B2)

b) Pour $P_{\alpha=0} \subset P_\beta \in P_F$, $\beta \in B$: $\lambda = \lambda + 1$: $t_\lambda = (\beta, P_{\alpha=0})$: i=i+1

$\Pi_i = P_{\alpha=1}$: Aller à (B2)

c) Pour tout $\alpha, \beta \in B$ $P^i_{\beta\alpha}$ c $P_\alpha \in P_{f_j}$: i=i+1: $\Pi_i = P_{\alpha=1}$:

$\Pi_K = P_{\alpha=0}$: K = K+1 : Aller à (B2)

III.2 - IMPLANTATION ET TEST DE LA METHODE D'EVALUATION PROPOSEE

Pour vérifier les performances de la méthode d'évaluation proposée, nous avons implanté l'algorithme décrit ci-dessus sur un micro-ordinateur. Nous avons aussi testé la méthode en traitant plusieurs exemples de fonctions combinatoires.

Pour chaque fonction l'algorithme a permi de construire l'ADB associé. Il est donné sous forme d'une table de décision fig.III.2 .

		Nœuds de test				Nœuds de Sortie	
				Nº Nœud suivant			
Nº Nœud	Nature des Nœuds	Variables du test	Valeur du test	test VRAI	test FAUX	Nº NOEUD SUIVANT	VALEUR DE SORTIE
1	TEST						
	SORTIE					1	
						1	

Figure III.2 : Table de décision associée à l'ADB construit

L'algorithme offre la possibilité de choisir la longueur des tests (w) associés aux noeuds intermédiaires de l'ADB. Ainsi pour chacun des exemples de fonctions combinatoires traitées, on peut obtenir des ADB différents suivant le choix de la longueur des tests.

Nous donnons dans les figures III.3,4,5,6 les différents ADB obtenus par la mise en oeuvre de l'algorithme présenté ci-dessus.

III.2.1. - Performances de la méthode d'évaluation proposée

Nous nous sommes fixés comme objectif de réduire la durée maximale d'évaluation de la fonction combinatoire.

A ce titre, nous avons développé un critère entropique de choix des tests qu'il faut associer aux noeuds de l'ADB représentant une fonction combinatoire donnée.

La mise en oeuvre de ce critère entropique de choix des tests permet à priori de construire un ADB minimum.

En étudiant attentivement les ADB d'évaluations et qui représentent les fonctions combinatoires des exemples traités, on remarque que les ADB obtenus qui représentent les exemples 1,2,4 (fig.III.3,4,6) sont minimaux.

Par contre l'ADB de la figure III.5 qui représente la fonction combinatoire $F(X_1,X_2,X_3,X_4) = X_1\overline{X}_3X_4 + X_1X_3\overline{X}_4 + X_2X_3X_4 + X_2\overline{X}_3\overline{X}_4$ n'est pas minimal. L'ADB de la figure III.7 représente la solution minimale pour cette fonction.

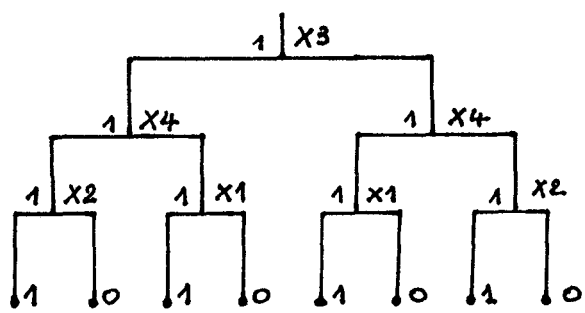
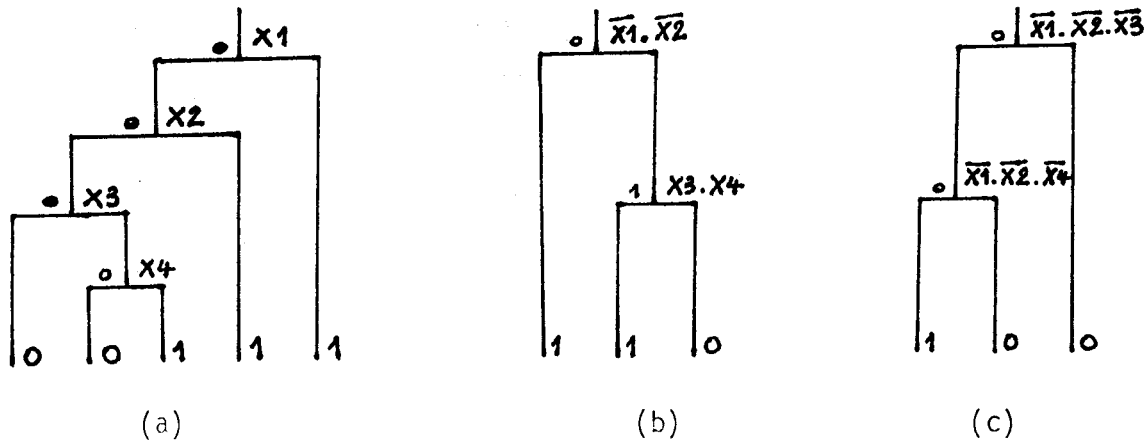


Figure III.7 : ADB minimal représentant F(X)

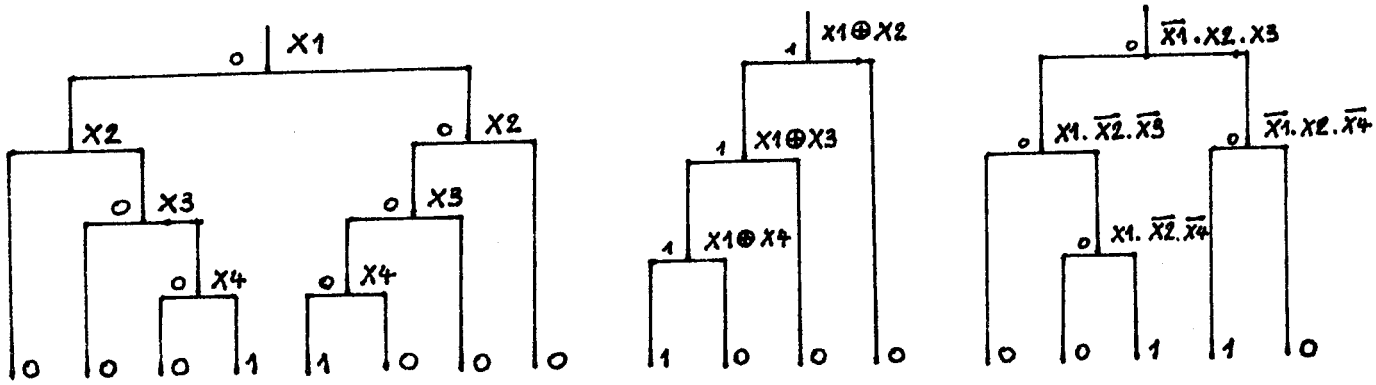
Exemple 1 : $F_1(X) = X_1 + X_2 + X_3 X_4$



(a) (b) (c)

Figure III.3 : ADB de $F_1(X)$

Exemple 2 : $F_2(X) = X_1 \bar{X}_2 \bar{X}_3 \bar{X}_4 + \bar{X}_1 X_2 X_3 X_4$



(a) (b) (c-

Figure III.4 : ADB de $F_2(X)$

Exemple 3 : $F_3(X) = X_1\ X_3\ \overline{X}_4 + X_1\ \overline{X}_3\ y_4 + X_2\ X_3\ X_4 + X_2\ \overline{X}_3\ \overline{X}_4$

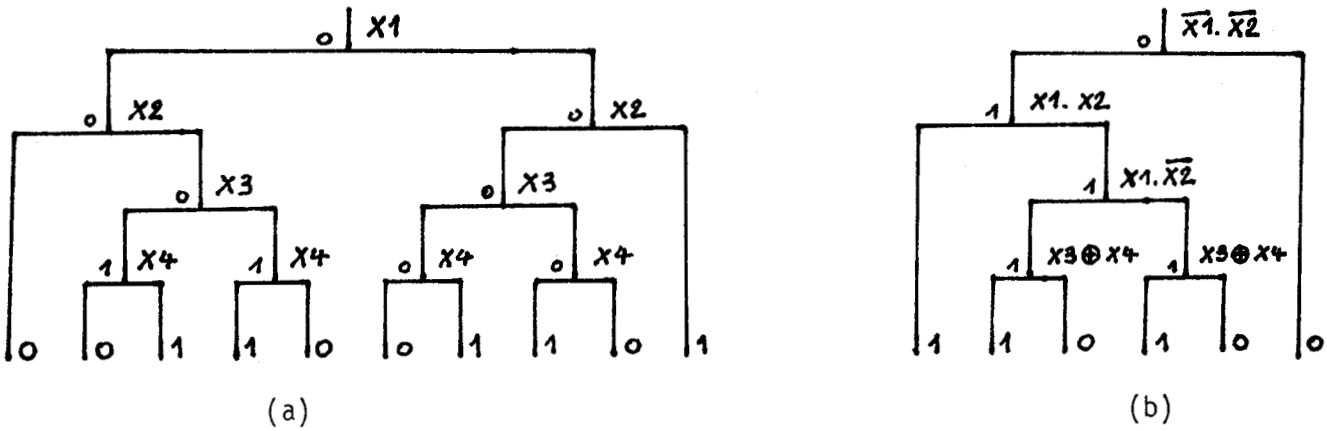


Figure III.5 : ADB de $F_3(X)$

Exemple 4 : $F_4(X) = X_1\ X_2 + X_2\ X_3 + X_1\ X_3$

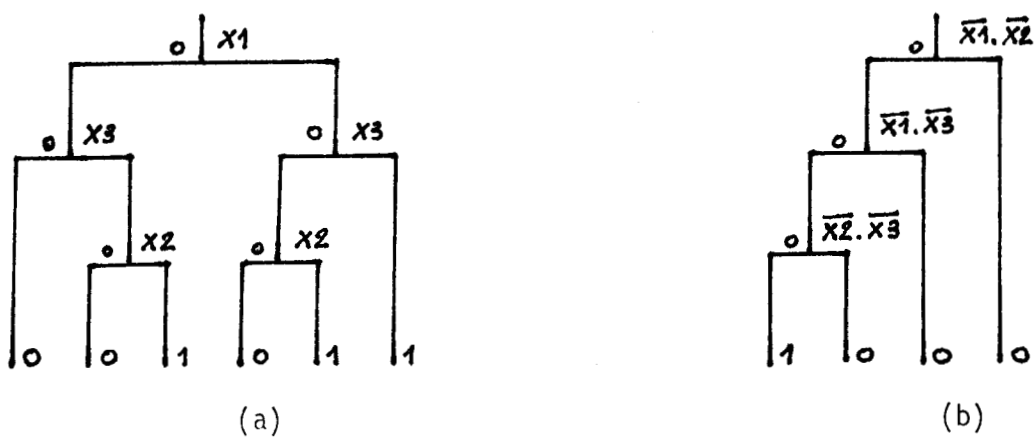


Figure III.6 : ADB de $F_4(X)$

Le traitement de l'exemple 3 montre donc que la méthode d'évaluation proposée n'est pas optimale.

On constate néanmoins que cette méthode donne des résultats intéressants lorsque les variables booléennes ont la même fréquence d'apparition (sous forme normale ou complémentée) dans la forme algébrique représentant la fonction combinatoire (voir exemple 1,2,4).

Les résultats sont moins satisfaisants lorsque la forme algébrique de la fonction combinatoire comporte des variables booléennes d'entrées dont la fréquence d'apparition est plus importante que celle d'autres variables les ADB de l'exemple 4 illustrent ce cas.

Nous pensons que l'introduction dans le critère entropique de choix des tests, d'une information qui traduit la fréquence d'apparition des variables booléennes dans la forme algébrique de la fonction combinatoire, peut améliorer considérablement les performances de la méthode. Nous avons construit aussi pour les exemples traités des ADB dans lesquels les tests sont des fonctions d'un sous-ensemble de variables booléennes d'entrée (fig.III.3b,3c,4b,4c,5b,6b).

Ce choix a permi de minimiser l'occupation mémoire des programmes engendrés. On constate dans ce cas que l'espace mémoire est approximativement réduit de moitié.

La durée moyenne d'évaluation est aussi réduite. Toutefois cette réduction se fait au détriment d'une complexité accrue des tests associés aux noeuds de l'ADB ,cela dégrade les performances temporelles de l'évaluation, si on adopte l'une des techniques de matérialisations étudiées dans le paragraphe (I.5). La recherche d'une structure matérielle capable d'implanter directement les ADB tel que ceux des figures III.3b, 3c, 4b, 4c, 5b,6b. peut rendre l'évaluation des fonctions combinatoires encore plus performante .

L'algorithme décrivant la méthode d'évaluation proposée a été conçu pour construire l'ADB représentant une fonction combinatoire bivaluée.
(B = { 0,1 }.)

Il peut être facilement modifié pour permettre la constructions d'un arbre de décision représentant une fonction combinatoire multi-valuée (B > 2). Dans le paragraphe suivant, nous rappelons quelques définitions relatives à la fonction combinatoire multi-valuée. Nous parlerons aussi des modifications à apporter à l'algorithme décrit dans (III.2) de façon à l'adapter à l'évaluation de ce type de fonction combinatoire.

III.3 - EVALUATION DE LA FONCTION COMBINATOIRE MULTI-VALUEE

Lorsqu'une fonction combinatoire F(X) définie sur X l'ensemble des variables d'entrée par l'application :

$$F : Y \longrightarrow B \; ; \; |B| > 2 \; , \; Y = B^n, \; n = |X|$$

prend ses valeurs dans l'ensemble B (B : algébre multi-valuée : $|B| > 2$) on dira que F(X) est une fonction combinatoire multi-valuée. | LEE.78 |.

Comme pour une fonction combinatoire bi-valuée, la fonction multi-valuée peut être décrite par une table de vérité ou représentée par une partition de l'ensemble Y.

Illustrons cela par un exemple : soit une fonction combinatoire multi-valuée F(X) définie sur $X = \{ X_1, X_2, X_3 \}$ et prenons ses valeurs dans l'ensemble $B = \{ 0,1,2 \}$. F(X) est définie par la table de vérité de la fig.III.8a.

Cette représentation met en évidence les classes de la partition de l'ensemble Y caractéristique de F(X). fig.III.8b:

$$\begin{cases} P_F(Y) = \{ P_0, P_1, P_2 \} \\ P_1 = \{ 1,4,5,6,11,12,13,14,19,24 \} \\ P_2 = \{ 2,3,7,17,18,21,25,26 \} \\ P_0 = \{ 0,8,9,10,15,16,20,22,23 \} \end{cases}$$

Figure III.8b : Partition caractéristique de F(X).

	X1	X2	X3	F(X)
0	0	0	0	0
1	0	0	1	1
2	0	0	2	2
3	0	1	0	2
4	0	1	1	1
5	0	1	2	1
6	0	2	0	1
7	0	2	1	2
8	0	2	2	0
9	1	0	0	0
10	1	0	1	0
11	1	0	2	1
12	1	1	0	1
13	1	1	1	1
14	1	1	2	1
15	1	2	0	0
16	1	2	1	0
17	1	2	2	2
18	2	0	0	2
19	2	0	1	1
20	2	0	2	0
21	2	1	0	2
22	2	1	1	0
23	2	1	2	0
24	2	2	0	1
25	2	2	1	2
26	2	2	2	2

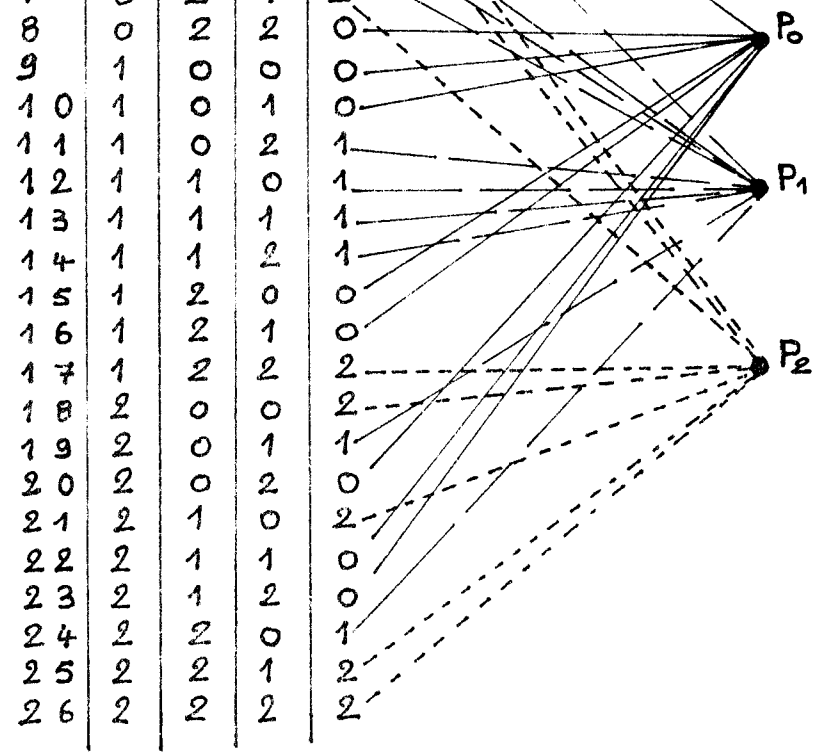


Figure III.8a : Table de vérité de F(X)

Comme pour la fonction combinatoire bi-valuée, il existe :

$$[\,|B|\,]^n$$

$[\,|B|\,]$ fonctions combinatoires multi-valuées définies sur les ensembles B et X.

Ce nombre augmente rapidement avec l'accroissement de l'ensemble B des modalités. Ainsi par exemple sur l'ensemble $X = \{ X_1, X_2 \}$ on trouve pour :

$B = \{0,1\} \rightarrow 2^{2^2} = 16$ fonctions bi-valuées.

$B = \{0,1,2\} \rightarrow 3^{3^2} = 19\,683$ fonctions combinatoires 3-valuées.

En plus de ce nombre important des fonctions combinatoires multi-valuées, ces dernières présentent une structure algébrique très complexe et de ce fait leur analyse est très difficile à faire dans le cas général mais faisable lorsque la fonction multi-valuée présente une symétrie par rapport aux variables d'entrées.

Dans le paragraphe suivant, nous généralisons la méthode d'évaluation proposée aux fonctions multi-valuées.

III.3.1. - Construction d'un arbre de décision multi-valué (ADM)

Nous avons vu que le moyen le plus efficace pour faire l'évaluation de la fonction combinatoire était de construire l'ADB qui représente cette fonction.

La construction de l'ADM représentant une fonction combinatoire multi-valuée peut aussi constituer une solution intéressante au problème de l'évaluation de ce type de fonction. Dans ce cas, ce choix sera lié à l'existence de structures matérielles capables d'implanter un ADM comme c'est le cas pour les ADB (voir I.5).

La méthode que nous avons développée précédemment pourra donc être généralisée (l'algorithme décrit dans III.3.1.1. traduit cette généralisation) à l'évaluation de toute fonction combinatoire.

La construction de l'arbre de décision représentant la fonction combinatoire dépendra alors de l'ensemble B des modalités prises par les variables d'entrée et la fonction combinatoire considérée.

Nous avons testé la méthode sur l'évaluation des fonctions combinatoires bi-valuées ($|B| = 2$). Sa mise en oeuvre pour l'évaluation des fonctions multi-valuées ($|B| > 2$) permet la construction de l'ADM représentant cette fonction.

L'exemple de la fig.III.10.a,b illustre cette application.

	X_1	X_2	$F(X_1,X_2)$
0	0	0	1
1	0	1	1
2	0	2	1
3	1	0	0
4	1	1	0
5	1	2	2
6	2	0	2
7	2	1	2
8	2	2	1

Fig.III.10.a :
Table de vérité de $F(X_1,X_2)$

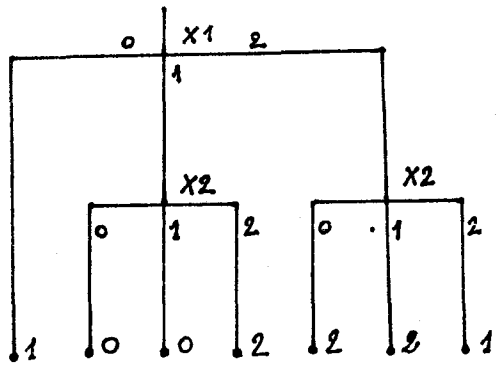


Figure IV.10b : ADM minimum associé à $F(X_1X_2)$

III.3.1.1. - Construction de l'arbre de décision représentant une Fc : Algorithme général

Cet algorithme permet de construire l'arbre de décision représentant une fonction combinatoire quelconque ($\forall\ |B| \geq 2$).

. Acquisition des données

1 : introduire l'ensemble $X = \{X_1, X_2, \ldots, X_n\}$; $n = |X|$ des variables d'entrées

2 : introduire l'ensemble B des modalités.

3 : introduire la fonction combinatoire à évaluer $F(X)$

3.a : génération de l'ensemble $Y = \{y_1, y_2, \ldots, y_m\}$; $m = |Y|$

3.b : génération de la partition P_F de Y :

$$P_F = \{P_\beta,\ \beta \in B\}\ ;\ P_\beta = \{y \in Y / F(y) = \beta \in B\}$$

4 : construction de l'ensemble $\mathcal{F}^*$ des fonctions de test.

4.a : introduire w le nombre des variables qui constituent les tests.

4.b : construction de l'ensemble $\mathcal{F}^* = \{f_1^*, f_2^*, \ldots, f_L^*\}$; $L = |\mathcal{F}^*|$:

$\forall\ y \in Y\ ,\ \forall\ f^* \in \mathcal{F}^*$: $f^*(\hat{y}) = \alpha \in B$.

. Construction de l'arbre de décision

5 : initialisation : $i=1$; $\Pi_i = Y$, $\lambda = 0$, $K = 0$

6 : pour toute fonction $f_j^* \in \mathcal{F}^*$; $j = 1,2,\ldots, L$:

6.a construire les partitions $P_{f_j^*}$ et $P_{F,i}$ de Π_i.

$$P_{f_j^*} = \{P_\alpha, \alpha \in B\} \; ; \; P_\alpha = \{y \in \Pi_i \subseteq Y / f_j^*(\hat{y}) = \alpha \in B\}$$

$$P_{F,i} = \{P_{\beta,i}, \beta \in B\} \; ; \; P_{\beta,i} = \{y \in \Pi_i \subseteq Y / F(y) = \beta \in B\}.$$

6.b : déterminer les probabilités des sous-classes $P^i_{\beta\alpha} = P_\alpha \cap P_{\beta,i}$

$$p_{\beta\alpha} = \frac{|P_\alpha \cap P_{\beta,i}|}{|\Pi_i|} \; ; \; p_\alpha = \frac{|P_\alpha|}{|\Pi_i|} \; ; \; \forall \beta,\alpha \in B$$

6.c : évaluer la quantité d'information apportée par la partition $P_{f_j^*}$ sur la valeur à donner à $F(X)$ soit :

$$H_j\,(P_{F,i}/P_{f_j^*})$$

6.d : choix de la fonction de test associée au noeud $n_i \in N$: on calcule :

* $H_i\,(P_{F,i}/P_{f_j}) = \underset{j=1,2,..,L}{\text{Min}} \; H_j(P_{F,i}/P_{f_j^*}) : f_j, f_j^* \in \mathcal{F}^*$

* $f_i = f_j$

. Recherche des noeuds terminaux

7 : pour $H_i\,(P_{F,i}/P_{f_j}) = 0$:

7.a : $\forall \alpha,\beta \in B$: $\lambda=\lambda+1$: si $P_\alpha \subset P_\beta \in P_F$ alors $t^i_\lambda = (\beta, P_\alpha)$

7.b : pour K=0: FIN CONSTRUCTION ARBRE : EDITION RESULTATS

7.c : pour $K > 0$: $i=i+1$: $\Pi_i = \Sigma_K$: $K=K-1$: Aller à (6)

8 : pour H_i $(P_{F,i}/P_{f_j}) > 0$:

8.a : pour tout $\alpha,\beta \in B$: si P_α c P_β : $\lambda=\lambda+1$: $t^i_\lambda = (\beta, P_\alpha)$

8.b : pour tout $\alpha,\beta \in B$: si $P^i_{\beta\alpha}$ c P_α : $K=K+1$: $\Sigma_K = P_\alpha$: Aller à (7c)

III.4 - CONCLUSIONS

Nous avons présenté au cours de cette étude, une nouvelle méthode d'évaluation de la fonction combinatoire. Elle procède par classification pour construire l'arbre de décision.

Elle met en oeuvre un critère de choix des tests basé sur la théorie de l'information. Ce critère permet d'associer à chaque noeud de l'arbre de décision, le test le meilleur relativement à ce noeud. Il est évident que l'optimalité n'est ici que locale et que le nombre de test maximum nécessaire à l'assignation de la fonction considérée, n'est pas forcément minimisé. L'heuristique utilisée permet toutefois d'espérer l'obtention de bons résultats.

Nous avons décrit la méthode d'évaluation proposée sous forme d'un algorithme. Sa mise en oeuvre permet à priori, par une analyse exhaustive et descendante, la construction d'un arbre de décision minimum associée à la fonction combinatoire considérée.

L'application et le test de l'algorithme sur plusieurs exemples de fonctions combinatoires, a montré qu'on n'obtient pas toujours la solution minimale.

Nous avons constaté lors de ce test que le résultat obtenu, dépend beaucoup de la fréquence d'apparition de variables d'entrées dans l'expression algébrique de la fonction combinatoire. Nous pensons que l'introduction d'une information relative à la fréquence d'apparition des variables d'entrées, dans le critère de choix des tests, peut améliorer considérablement les performances de cette méthode.

Si jusqu'à présent cette nouvelle méthode ne donne pas toujours de meilleurs résultats, nous pensons qu'elle constitue néanmoins une voie intéressante dans l'évaluation de la fonction combinatoire.

Parmi les avantages qu'elle offre :

. la méthode est générale. Elle peut être appliquée à l'évaluation de différents types de fonctions combinatoires.

. la mise en oeuvre de la méthode se caractérise par une utilisation facile et une implantation pratique sur tout système informatique.

Parmi ses inconvénients :

. la méthode est exhaustive comme la pluplart des méthodes d'évaluation déjà existantes.

Des développements de la méthode que nous avons présentés restent à faire, il s'agit :

- d'étudier la modification du critère de choix tester pour y inclure une information relative à la fréquence d'apparition des variables d'entrées dans l'expression algébrique de la fonction combinatoire considérée.

- d'étudier l'extension de la méthode à l'évaluation des fonctions combinatoires incomplétement définies.

Nos travaux futurs seront orientés dans ce sens.

Deuxième Partie

SÉCURITÉ ET SIMULATION DE LA PARTIE OPÉRATIVE

I N T R O D U C T I O N

L'évolution de la technologie électronique (mémoires, RAM, ROM etc ... microprocesseurs, tableaux logiques programmables) a entrainé un développement considérable des systèmes automatisés qui permettent de réaliser des procédés industriels de plus en plus complexes.

La complexité de ses réalisations a vite mis en évidence le besoin,

. d'une part de nouvelles méthodes d'analyse et de représentation de grands systèmes | VAL.76 | ,|DAC.76 | ,|BLA.80 |.

. d'autre part des techniques de détection et de diagnostic qui doivent répondre à l'une des préoccupations majeurs des constructeurs de systèmes automatisés à savoir la sûreté de fonctionnement et plus particulièrement la sécurité des systèmes qui devient une donnée de base dans la conception des systèmes automatisés.

Dans ce mémoire, nous proposons d'améliorer la sécurité d'un système logique, en opérant une surveillance accrue et en temps réel de la partie opérative du système.

La méthode de test et de diagnostic mise en oeuvre pour remplir cette surveillance sera basée sur la description et la simulation de la partie opérative.

Dans le premier chapitre, après avoir rappelé la définition de la sûreté de fonctionnement d'un système et les notions qu'elle englobe (fiabilité, disponibilité, sécurité ...), nous définirons deux approches complémentaires qui permettent d'appréhender correctement les problèmes de sécurité à savoir :

- l'intolérance aux fautes
- la tolérance aux fautes

Nous commenterons aussi les différents moyens de réaliser des systèmes tolérants ou intolérants aux fautes.

Dans le cadre des travaux antérieurs relatifs à la détection et à la localisation des défauts dans les systèmes logiques, nous étudierons :

. des méthodes de détection utilisant un code K parmi n

. des méthodes de détection utilisant le test en ligne

Nous entreprendrons au deuxième chapitre, l'étude d'une méthode de description et de décomposition de la partie opérative. Seules les informations contenues dans cette dernière seront exploitées.

Nous définirons l'élément de base de cette description : l'action élémentaire et nous établirons des règles de décomposition à partir desquelles un algorithme de décomposition sera développé. L'utilisation de cet algorithme permettra la décomposition de toute action associée à un actionneur en actions élémentaires.

Le chapitre III traitera de l'étude de la procédure de test et de diagnostic organisée à partir de la description et de la simulation de la partie opérative.

Nous étudierons ensuite les différents défauts dans la partie opérative ainsi que le comportement de la procédure de test et de diagnostic vis à vis de leur détection.

Nous discuterons aussi de l'implantation de cette procédure sur un dispositif de surveillance. On insistera sur :

- la structuration des données relatifs à la description de la partie opérative et de sa gestion.

- les problèmes de synchronisation des échanges entre le système et le dispositif de surveillance.

- la condition de réalisation de la surveillance, elle concerne le temps de cycle de traitement du dispositif de surveillance. Elle sera décrite par une contrainte temporelle sur ce temps de cycle.

Nous terminerons cette étude par la validation des performances de la procédure de test et de diagnostic. Nous présentons le système réalisé qui a été utilisé pour mener à bien cette validation.

DEUXIÈME PARTIE

SECURITE ET SIMULATION DE LA PARTIE OPERATIVE SYSTEME-LOGIQUE

Mots clés : Sécurité, partie opérative, simulation, action élémentaire test et diagnostic, détection des défauts.

Résumé : L'étude rentre dans le souci constant d'améliorer la sécurité des systèmes logiques industriels . Ces améliorations impliquent la détection et la localisation des défauts dans ces systèmes.

La procédure proposée permet le test et le diagnostic des capteurs et actionneurs de la partie opérative et ses liaisons avec la partie commande.

La procédure est basée sur la simulation de la partie opérative. Le test et le diagnostic est opéré suivant une vérification de la cohérence des comptes-rendus issus de la partie opérative et ceux issus de la simulation de celle-ci.

Pour simuler la partie opérative, il est nécessaire de décrire cette dernière. Une méthodologie de description de la partie opérative est proposée. Elle est basée sur la définition d'une action élémentaire.

CHAPITRE I

AMÉLIORATION DE LA SÛRETÉ DE FONCTIONNEMENT D'UN SYSTÈME

CHAPITRE I

AMÉLIORATION DE LA SÛRETÉ DE FONCTIONNEMENT D'UN SYSTÈME

CHAPITRE I

AMÉLIORATION DE LA SURETÉ DE FONCTIONNEMENT D'UN SYSTÈME

Dans ce chapitre après avoir rappeler les notions essentielles qui caractérisent la sûreté de fonctionnement d'un système, nous traiterons les différentes techniques de redondance qui améliorent la sûreté de fonctionnement.

Nous nous intéresserons ensuite à l'aspect sécurité dans les systèmes logiques industriels. Nous étudierons à cet effet quelques méthodes de détection et localisation des pannes qui affectent les composants de ces systèmes. Ces méthodes reflétent les principales directions des recherches entreprises dans ce domaine.

I.1 - SURETE DE FONCTIONNEMENT D'UN SYSTEME

D'un point de vue général, la sûreté de fonctionnement peut être définie comme l'aptitude d'un système et de son environnement à minimiser la probabilité d'apparition de défaillances et/ou à minimiser leurs effets.

La sûreté de fonctionnement est caractérisée par plusieurs notions :

I.1.1. - Grandeurs caractéristiques de la sûreté de fonctionnement

|LAP.75|

La sûreté de fonctionnement est composée principalement par quatre grandeurs caractéristiques : Fiabilité, Disponibilité, Maintenabilité et sécurité :

LA FIABILITE :

Exprime l'aptitude d'un système à suivre un déroulement conforme à celui prévu dans des conditions d'utilisation bien spécifiées. On la définie par la probabilité de non apparition de défaillance entre un instant donné et l'instant courant.

LA DISPONIBILITE :

C'est la fraction de temps pendant laquelle le système est apte à fonctionner et la faculté qu'il offre à exécuter les tâches qui lui sont soumises. Elle prend en compte la faculté d'adaptation lorsqu'interviennent des pannes | KER 76 |.

La disponibilité est définie par la probabilité que le système soit en état de bon fonctionnement à un instant donné.

LA MAINTENABILITE :

Elle traduit soit le maintien de bon fonctionnement, soit la remise du système dans cet état après une défaillance. Elle est caractérisée par la probabilité de restaurer l'intégrité du système à un instant donné sachant qu'il devient défaillant à tout instant.

LA SECURITE :

Elle envisage toute circonstance normale ou anormale susceptible de conduire à une situation dangereuse pour le personnel ou le matériel | LIE 76 |. Elle est définie par la probabilité de non apparition d'une panne aux conséquences catastrophiques dans un intervalle de temps donné.

Toutes ces notions sont bien entendu très liées. Elles recouvrent des points communs : ainsi la disponibilité tient en compte la fiabilité des composants matériels; la sécurité d'un système peut être augmentée en accroissant la fiabilité de ses éléments.

Elles peuvent aussi entrer en conflit | LIE 76 |,| BAC 80 |.

En tenant compte des paramètres cités ci-dessus, on trouve deux approches permettant de réaliser des systèmes sûrs de fonctionnement.

I.1.2. - Conception des systèmes logiques tolérants

Un système intolérant aux fautes réagit dès l'apparition d'une anomalie. Pour obtenir une sécurité acceptable, il faut donc éliminer les causes d'erreurs. Ceci entraîne :

- l'élimination des erreurs de conception
- l'utilisation des éléments de matériels et logiciel éprouvés
- une politique de maintenance préventive
- une adaptation du système à son environnement

Un système tolérant aux fautes doit être capable d'assurer un service suffisant en présence d'erreur.

Il doit également détecter les anomalies afin d'éviter leur accumulation.

Il est certain que l'on doit s'orienter vers la mise en oeuvre de systèmes tolérants. Ceci intervient à différents stades de la conception.

I.1.2.1. - Système réceptif - Système sensible et sécurité

L'état total d'un système logique est défini par :

- son état des variables d'entrée
- son état des variables de sortie
- son état interne

Rappelons ici quelques définitions relatives au comportement d'un système vis à vis des modifications de ses états internes, des variables d'entrée et des variables de sortie.

Le système est supposé représenter et décrit par un grafcet | BLA 80 |.

- SITUATION

On appelle une situation d'un grafcet, l'ensemble des étapes actives à un instant donné.

On dira aussi que tout état est instable si la situation varie pour l'état des variables d'entrée invariant.

- CONTEXTE D'UN GRAFCET

On appelle contexte d'un grafcet, une relation binaire entre l'activité, l'inactivité d'un sous-ensemble d'étapes du grafcet et des variables d'entrée.

Dans ces conditions, on peut définir les systèmes réceptifs, les systèmes sensibles par :

A/ Réceptivité d'un système

Un système représenté par un grafcet et placé dans une situation donnée est dit réceptif à une variation d'un contexte donné, si cette variation provoque le passage du système à une autre situation.

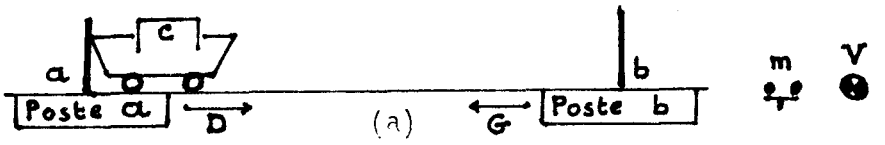
B/ Sensibilité d'un système

Un système représenté par un grafcet et placé dans une situation donnée est dit sensible à une variation d'un certain contexte si cette variation provoque la modification des variables de sortie du système sans la modification de la situation de celui-ci.

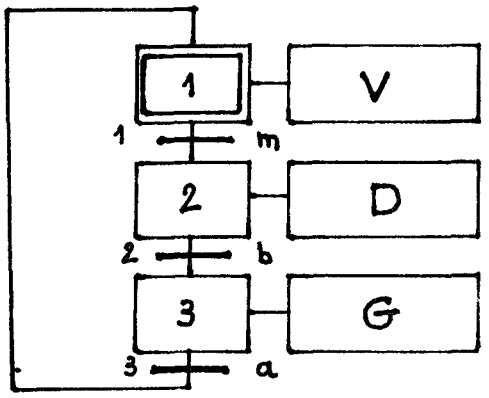
C/ Indifférence d'un système

Un système représenté par un grafcet et placé dans une situation donnée est dit indifférent à un certain contexte si le système n'est ni réceptif, ni sensible à la variation de ce contexte.

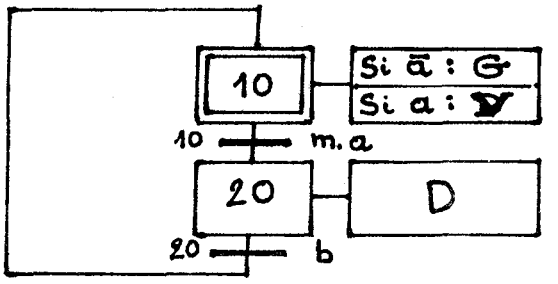
EXEMPLE : On considère le processus représenté par la figure Ia et dont le fonctionnement est décrit par les deux grafcets équivalents des figures Ib et Ic. | BLA 80 |.



Processus



(b) : Grafcet



(c) Grafcet

Figure 1

- Le système dont le fonctionnement est représenté par la figure Ib est un système réceptif : ainsi par exemple lorsque le système sera dans la situation ((3) : étape (3) active), il sera réceptif au contexte C_o = ā dont la variation (C_o = a) provoque l'évolution du système de la situation (3) à la situation (1).

- Le système dont le fonctionnement est décrit par le grafcet de la figure Ic est un système sensible. En effet, lorsque le système sera dans la situation (10). Il sera sensible au contexte C_o = a dont la variation (C_o = ā) provoque la modification des variables de sorties (V = 0, G = 1) tout en restant dans la situation (10).

Il serait intéressant d'étudier le comportement des systèmes réceptifs et des systèmes sensibles vis à vis des défauts.

Il est évident que les systèmes sensibles sont plus exposés aux défauts mêmes les plus fugitifs. En effet l'apparition d'un défaut dans un système sensible peut activer des sorties non attendues et de ce fait cette situation risque d'engendrer des conséquences très dangereuses pour le matériel et le personnel. Par conséquent, les systèmes sensibles offrent une mauvaise sécurité et il est donc préférable de ne pas les utiliser.

Les systèmes réceptifs offrent relativement une sécurité meilleure car les risques engendrés par l'apparition de défauts dans ces systèmes sont plus limités.

Il est donc préférable, lors de l'analyse d'un processus de concevoir des systèmes globalement réceptifs ou à défaut des systèmes "minimum sensibles" (nombre de situations où le système est sensible est minimum). Il est préférable aussi que les systèmes conçus soient tolérants aux défauts.

Cette tolérance aux défauts nécessitera des systèmes redondants qui permetteront une reconnaissance des défauts et si possible une marche dégradée.

I.1.3. - Différentes redondances

Un dispositif redondant est un élément du système qui lui permet de tolérer certaines fautes. L'introduction de tel dispositif peut être envisagée au niveau de l'ensemble du système ou seulement au niveau de certains de ses éléments.

Suivant le fonctionnement d'un dispositif redondant dans un système, on peut définir plusieurs types de redondances :

. REDONDANCE DYNAMIQUE

Le dispositif redondant ne prend part active à la réalisation des tâches qu'après détection et réaction à la faute.

. REDONDANCE SELECTIVE

L'élèment redondant remplace l'élèment défaillant après détection et réaction à la faute.

. REDONDANCE TEMPORELLE

Ce type de redondance a pour fonction de mémoriser des informations avant détection de la faute (point de reprise). Ces informations ne sont utilisées qu'après détection.

. REDONDANCE STATIQUE

Où l'élément redondant participe à la réalisation des tâches avant même la détection d'une erreur. Un cas particulier de la redondance statique, c'est la redondance massive. Dans ce cas l'élément redondant est constitué par des éléments semblables à l'élément primaire qui effectuent le même traitement. La réalisation des tâches est obtenue par une procédure de vote.

L'étude de ces redondances montre que (si on tient seulement compte de l'aspect sécurité)les redondances dynamiques offrent une plus grande sécurité contrairement aux redondances statiques dans lesquelles une défaillance de l'élément redondant est généralement fatale au système.

Dans notre étude, nous ne nous intéressons qu'à l'aspect sécurité. Notre objectif sera d'améliorer la sécurité d'un système logique.

I.2 - SECURITE D'UN SYSTEME LOGIQUE

La sécurité d'un système logique concerne tous ses éléments constituants ainsi que l'environnement extérieur avec lequel il dialogue. Il est donc impératif que chaque composant de cet ensemble (système - environnement) apporte sa contribution pour garantir une meilleure sécurité.

Avant de définir ces contributions, nous allons tout d'abord rappeler la structure d'un système logique industriel, pour dégager les différents éléments qui seront appeler à jouer un rôle pour améliorer la sécurité.

I.2.1. - Structure d'un système logique industriel

D'une façon générale, un système logique industriel peut se décomposer en deux parties qui coopérent | BLA 80 | | RAP 77 | . L'une est dite partie commande, l'autre est la partie opérative. (fig.I.1).

La partie opérative constitue le processus à automatiser. Elle effectue des tâches précises, décrites par un cahier des charges, lorsque l'ordre lui en est donné par la partie commande. Elle établit des comptes-rendus, image de son évolution, grâce auxquels cette dernière est tenue informée de l'état d'avancement des tâches ordonnées.

La partie commande, c'est un automatisme qui élabore des ordres destinés à la partie opérative, suivant les comptes-rendus reçus de celle-ci. Outre ce dialogue par ordres et comptes-rendus avec la partie opérative, la partie commande échange des informations avec l'environnement extérieur qui peuvent être des consignes et des signalisations.

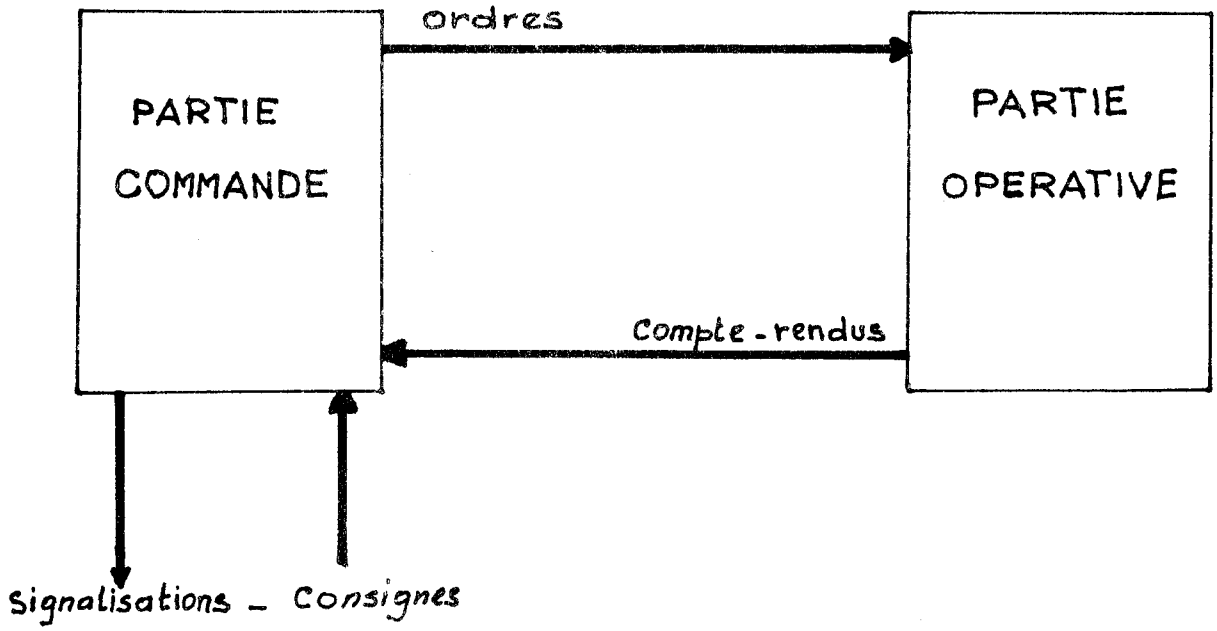


Figure I.1 - Structure d'un système logique

I.2.2. - La sécurité des machines séquentielles placées dans leur environnement

a) La sécurité propre au matériel

La partie opérative doit assurer la sécurité des personnels et matériel. Cette préoccupation doit être présente au niveau de l'étude ergonomique du poste de travail et dans l'implantation de la machine sur le site. Un certain nombre de décrets et de recommandations réglementent ou régissent les réalisations en ce domaine.

Les choix technologiques au niveau des interfaces entre commande et partie opérative doivent aussi être faits en respectant les critères de sécurité dans tous les modes de marche.

b) La sécurité propre au logiciel

La commande doit à son tour garantir la sécurité même en cas de défaillance matérielle, logicielle et humaine. Des représentations particulières, *GRAFCET*, *GEMMA* associées à des traducteurs automatiques, limitent les risques d'erreurs de logiciel. | BLA 80 | | ADE 80 | .

Une période de validation du logiciel attentive, reste toujours indispensable.

I.2.3. - Sécurité vis à vis des défaillances matérielles

Dans le paragraphe suivant, nous présenterons quelques méthodes de détection et de localisation des pannes. Elles portent selon des cas :

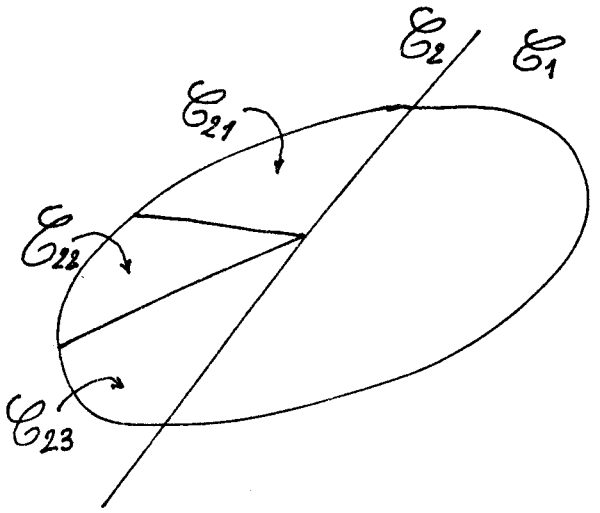
- sur un contrôle de la commande
- sur une redondance matérielle des capteurs
- sur une surveillance accrue des capteurs et actionneurs

Notre objectif est de détecter les défaillances de la partie opérative et de l'ensemble des organes de liaison avec la partie commande. Une vérification temps réel des relations causales existents entre les ordres et les comptes-rendus d'une part et les procédures de dialogue, d'autre part doit permettre de détecter certaines défaillances.

Appelons $\mathcal{E}$ l'ensemble des entrées de la partie commande. Il peut être décomposé en deux classes :

- l'ensemble $\mathcal{E}_1$ des variables de contrôle qui permettent le suivi de l'évolution de la partie opérative. Elles interviennent dans la relation causale action/compte-rendu.

- l'ensemble $\mathcal{E}_2$ des variables d'échange qui interviennent dans les relations homme/machine ou machine/machine.



Cet ensemble $\mathcal{E}_2$ peut lui-même être partitionné. En effet quand il y a dialogue avec l'environnement, on trouve dans $\mathcal{E}_2$:

- l'ensemble $\mathcal{E}_{21}$ des appels reçus de l'extérieur
- l'ensemble $\mathcal{E}_{22}$ des réponses aux appels émis par la commande
- les entrées correspondantes à une procédure d'échange du type " monologue ", forment l'ensemble $\mathcal{E}_{23}$.

Les éléments de $\mathcal{C}_1$ et $\mathcal{C}_{22}$ traduisent les effets relatifs aux ordres générés par la commande. Ils seront disponibles pour un test de vraisemblance permettant une détection de défaut. Les éléments de $\mathcal{C}_{21}$ seront exploités dans le même esprit par les machines constituants l'environnement.

Seuls les éléments de $\mathcal{C}_{23}$ ne permettent aucun test de vraisemblance. La sécurité pour ces éléments passe alors par précaution entre (cablage de sécurité ...).

I.3 - QUELQUES METHODES DE DETECTION ET DE LOCALISATION DES PANNES DANS LES SYSTEMES LOGIQUES

I.3.1. - Méthodes de détection des pannes par codage " K parmin "

I.3.1.1. - Méthode de TOHMA | TOH 71 |

TOHMA propose une réalisation d'une machine séquentielle sûre en présence de pannes. Cette réalisation est basée sur un codage (K parmin) des états de la machine.

Dans cette réalisation, l'auteur a supposé les entrées et le circuit combinatoire de sortie sans faute . Les pannes détectées sont des pannes simples de collage à zéro ou à un d'une connexion entre les modules.

Il a défini aussi les conditions de sûreté d'une machine séquentielle en présence de fautes :

- Si S est l'ensemble des états normaux de la machine et S_R l'ensemble des états obtenus quand il y a panne agissante, il faut que les ensembles S et S_R soient disjoints : $S \cap S_R = \Phi$
- L'état suivant d'un état appartenant à S_R doit lui-même appartenir à S_R quelque soit l'entrée appliquée.

Cette deuxième condition si elle est vérifiée, exclue le masquage (obtention d'une sortie correcte après une séquence non correcte dûe à une panne).

L'utilisation d'un code K parmi n pour le codage de l'ensemble S des états normaux de la machine, apporte les avantages suivants :

- La distance de HAMMING entre deux mots du code est égale à deux. La sûreté en présence de fautes ne peut pas être réalisée si deux mots du code ont une distance égale à 1 car dans ce cas un collage simple peut se traduire par une inversion de ces deux mots.

. Il n'existe pas dans le code K parmi n un mot qui inclut un autre.

Rappelons qu'un mot $M_1 = \alpha_1 \alpha_2 .. \alpha_n$ inclut le mot $M_2 = \beta_1 \beta_2 ... \beta_n$ si $\forall \alpha_i \in M_1$, $\forall \beta_i \in M_2$ on a $\alpha_i > \beta_i$.

Ces considérations ont conduit à une réalisation monotone de la fonction d'état suivant de la machine.

Les circuits d'excitation sont réalisés par des formes " ONSET " ou " OFFSET " qui s'écrivent :

" ONSET " $Y_i^+ = \sum_j I_j \mid \sum_h (y_{h_1} . y_{h_2} \ldots y_{h_K})$ (a)

" OFFSET" $\bar{Y}_i^+ = \sum_j I_j \mid \sum_h (\bar{y}_{h_1} . \bar{y}_{h_2} \ldots . \bar{y}_{h_{n-K}})$ (b)

Dans ces expressions : y_i^+ est l'état suivant de la variable y_i, I_j est un vecteur d'entrée, K est le nombre des variables d'états au niveau 1 (code K parmin), y_{h_1}, y_{h_2}, ... sont les K variables d'état ayant la valeur 1 dans un présent état q_h, qui transfert à un état suivant où $y_i^+ = 1$ sous I_j dans la forme (a).

L'auteur a étudié le masquage des pannes sur ces deux formes de réalisation. Cette étude a conduit à des machines non sûres en présence de pannes.

Pour remédier à cette défaillance de la réalisation,TOHMA a proposé une solution qui consiste à combiner les deux formes (a) et (b) et à opérer une partition de l'ensemble des n états de la machine en deux blocs B1 et B2 tel que :

- Les K premières variables $y_1, y_2, \ldots y_K$ du bloc B1 sont réalisées par la forme "OFFSET" (b).
- Les (n-K) variables du bloc B2 sont réalisées par la forme "ONSET" (a).

Les mots $y_1 = y_2 = \ldots = y_K = 1$

$y_{K+1} = \ldots = y_n = \phi$ sont interdits

En fonctionnement normal, il y a donc au moins une variable à l'état zéro parmi les K premières et une variable à l'état 1 parmi les n-K dernières.

Dans le cas de collage à 1, la partie "OFFSET" atteint la valeur finale 1 1 1 1 ... et il y a au moins une variable "ONSET" au niveau 1, le mot obtenu a un poids supérieur à K. La panne est ainsi détectée.

Dans le cas de collage à zéro, on déduit par dualité que le poids final est inférieur à K et la panne est détectée.

La machine séquentielle ainsi réalisée est donc sûre en présence de défauts ce qui est l'objectif de l'auteur.

Enfin TOHMA, montre que sa méthode reste parfois valable même si les mots du code K parmi n sont tous employés, toutefois l'analyse dans ce cas devient très longue et difficile.

Une autre faiblesse de la méthode, c'est qu'elle suppose les entrées et le circuit combinatoire de sortie sans défauts.

Pour remédier à cette faiblesse, DIAZ a proposé une méthode [DIA.74] permettant le test des entrées et des sorties et en employant que la forme "ONSET".

I.3.1.2. - Méthode de DIAZ [DIA.74]

Nous avons vu plus haut que l'emploi combiné des formes "ONSET" et "OFFSET" permet d'éviter le masquage et de réaliser une machine sûre.

Il est possible de n'employer que la forme "ONSET" si on s'assure au préalable que le masquage ne pourra pas avoir lieu dans le cas du collage à 1.

DIAZ a proposé un algorithme de codage qui permet d'éviter cette situation.

CIRCUIT COMBINATOIRE D'ENTREE

Le circuit d'excitation est réalisé sous la forme "ONSET". Plusieurs cas de pannes des entrées peuvent être envisagées. Ils sont tous détectées sauf le cas où la panne conduit à une inversion des vecteurs d'entrée.

Cette situation peut permettre un masquage et la panne ne peut pas être détectée; en effet :

soit X_i une variable d'entrée I_1, I_2 deux vecteurs d'entrée issus du circuit combinatoire d'entrée . I_1 et I_2 peuvent être mis sous la forme :

$$I_1 = A_1 X_i + B_1 \bar{X}_i$$

$$I_2 = A_2 X_i + B_2 \bar{X}_i$$

A_1, A_2, B_1, B_2 sont indépendants de X_i; un seul de ces vecteurs peut être à la fois au niveau 1, ce qui conduit à la condition

$$A_1.A_2 = B_1.B_2 = \phi$$

Le collage de X_i à zéro peut conduire à une inversion des vecteurs d'entrée et la panne est masquée.

Exemple : $A_1 = 1$, $A_2 = \phi$; $B_1 = \phi$, $B_2 = 1$

Dans ce cas le vecteur normalement prévu est I_1. La panne conduit à l'apparition de I_2, il y a inversion des vecteurs d'entrée I_1 et I_2. La condition d'inversion s'écrit :

$$A_1.B_2 = 1 \text{ ou } A_2.B_1 = 1$$

Pour éviter le masquage, il faut vérifier les conditions de non inversion pour tous les couples de vecteurs d'entrée I_K, I_L et pour toutes les X_i :

$$A_K.B_L = \phi$$

Une solution consiste à rendre redondantes les entrées qui sont dédoublées en X_i et X'_i appartenant au code (1 parmi 2). Ceci s'appelle : Réalisation DOUBLE RAIL.

Les équations du circuit combinatoire d'entrée s'écrivent alors :

$$I_K = A_K.X_i + A'_K\ X'_i$$

Les termes B_K et B'_K sont nuls, ce qui vérifie la condition de non inversion.

CIRCUIT COMBINATOIRE DE SORTIE

Il est réalisé suivant la même méthode : les sorties sont dédoublées comme les entrées ce qui permet une réalisation double rail.

L'ensemble ainsi conçu est donc sûr en présence de fautes.

AUTOTESTABILITE

Un circuit combinatoire est autotestable pour un ensemble de fautes si, pour toute faute de cet ensemble il existe une entrée qui conduit à une sortie erronée observable. Cette définition a été étendue à un circuit séquentiel. Dans le cas où un circuit est sûr et autotestable on dira qu'il est : totalement autotestable.

Après avoir rappeler les conditions de testabilité,DIAZ propose un algorithme permettant de vérifier ces conditions et montre aussi comment une machine séquentielle complète peut être réalisée autotestable.

1.3.1.3. - Conclusion

Les méthodes que nous avons résumé ci-dessus, sont basées sur la synthèse des machines séquentielles à partir des tables d'état. Un codage des états de la machine est alors réalisé pour établir les équations combinatoires qui traduisent le fonctionnement de la machine.

D'autres auteurs | MAR.74 |, |DEF.79|, après avoir présenter d'autres approches de descriptions d'une machine séquentielle, proposent des méthodes de détection des pannes en effectuant des tests en ligne.

1.3.2. - Méthodes de détection des pannes par test en ligne

Il n'existe pas de méthode générale, applicable systèmatiquement. Chaque processus constitue un cas d'espèce. Il est donc nécessaire d'exploiter toute information susceptible d'être vérifiée.

On présente ici deux méthodes de test en ligne. Elles mettent en oeuvre des informations issues d'une description de processus par réseau de Petri ou Grafcet.

1.3.2.1. - Méthode de test en ligne utilisant le réseau de Petri |MAR.74|

Le cahier des charges du processus est décrit par un réseau de Petri |BLA.76 | |THI.78|. La pondération du réseau de Petri sera la base de la méthode de test en ligne proposée.

Cette pondération du réseau de Petri est obtenue en attribuant à chaque place P_i un nombre entier strictement positif p_i.p_i est appelé poids de la place P_i.

La valeur du poids p_i est choisie de façon tel que : pour toute transition t_j du réseau de Petri, la somme des poids des places antécédantes soit égale à la somme des poids des places subséquentes.

Plusieurs approches de pondération d'un réseau de Petri |MAR.74| sont possibles.

Si on défini le poids total marqué d'un réseau de Petri par la quantité :

$\sum_i m_i p_i$: m_i est le nombre de marqueur dans la place P_i ($m_i = 1$, $r_i \rightarrow$ RdP. sauf).

On vérifiera que pour tout réseau de Petri pondéré, le poids total marqué reste constant à partir du marquage initial pour toute séquence de franchissement des transitions.

Dans la réalisation de test en ligne proposée par l'auteur, une valeur différente de la valeur normale du poids total marqué, est imposée en cas de présence d'une panne.

La méthode n'est pas applicable pour les circuits combinatoires d'entrée et de sortie.

Pour permettre un test en ligne de l'ensemble du système, les circuits combinatoires des entrées et des sorties, sont réalisées totalement autotestables conformément à la méthode de DIAZ. |DIA.75 |

Le test en ligne, en vue de la détection des pannes dans un système logique peut être envisagé d'une autre manière, notamment celle de la surveillance des entrées du système. DEFRENNE a proposé une méthode de test en ligne des capteurs et des actionneurs. Elle est basée sur la variation du niveau des variables d'entrée.

I.3.2.2. - Méthode de test en ligne des capteurs et actionneurs par surveillance du niveau des variables d'entrée |DEF.79| |DEF.80|

La fiabilité des composants électroniques utilisés pour la réalisation de la partie commande d'un système logique industriel, est nettement supérieure à celle de la technologie des capteurs et actionneurs, éléments qui constituent la partie opérative du système.

Les pannes pour la plus part sont dûes donc, soit à des défauts sur ces éléments eux-mêmes, soit sur les liaisons qui les relient à la partie commande.

L'auteur, après avoir introduit la notion de variation de niveau des variables d'entrée dans la description du cahier des charges |RAP.77| d'un système logique, a développé une procédure de test en ligne. La mise en oeuvre de cette procédure permet de détecter et de localiser des pannes affectant les éléments de la partie opérative. Les informations nécessaires à ce test sont obtenues à partir des comptes-rendus issus de la partie opérative.

a) Représentation affinée du cahier des charges

Dans cette représentation le fonctionnement du processus est décrit en prenant en compte les variations non orientées des variables d'entrée. La variation non orientée d'une variable X_i sera représentée par la variable dérivée $\dot{X}_i$. Elle permettra de mémoriser la variation de niveau de X_i jusqu'à son exploitation. La figure (fig.I.3) montre la mise en oeuvre de cette notion.

EXEMPLE : On suppose, que dans un cycle nous avons une action A qui provoque le déplacement d'une canne devant les capteurs C_1, C_2, C_3 (fig.I.2). Appelons X_1, X_2, X_3 les entrées correspondantes de la partie commande.

Figure I.2 : Processus

On suppose qu'avant l'exécution de l'action A, le capteur C1 est excité ($X_1 = 1$). Le déplacement de la canne va provoquer des changements d'états logiques sur les variables X_1, X_2, X_3, la représentation du fonctionnement de ce système est donné par le grafcet (fig.I.3). La variation non orientée des variables est mis en évidence.vis à vis de la détection des pannes, cette notion de surveillance des niveaux des variables d'entrée, s'avère insuffisante pour la détection de quelques défauts (blocage des actionneurs, pannes sur le dernier capteur accessible par une action). Une autre notion de temporisation en " CHIEN DE GARDE " a été introduite pour remédier à cette insuffisance. Elle consiste à déclencher une temporisation en même temps que l'action. La durée de cette temporisation est réglée légèrement supérieure à celle de l'action.

L'exploitation de ces deux notions (variation de niveau de variable, temporisation en "chien de garde "), permet de définir trois perceptes |DEF.80| suivants lesquels, une description affinée du fonctionnement d'un système logique sera la base d'une procédure de test en ligne des capteurs et actionneurs.

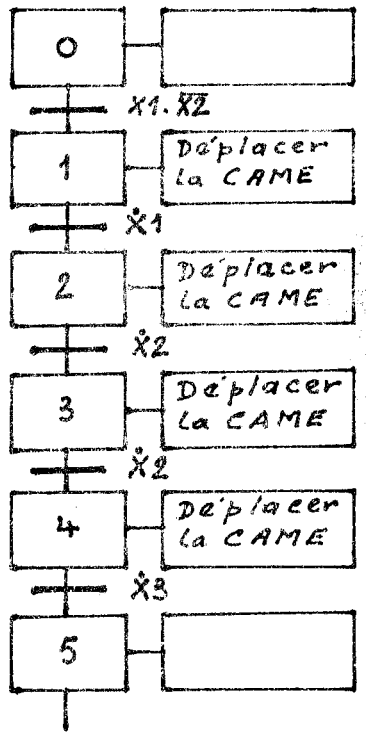
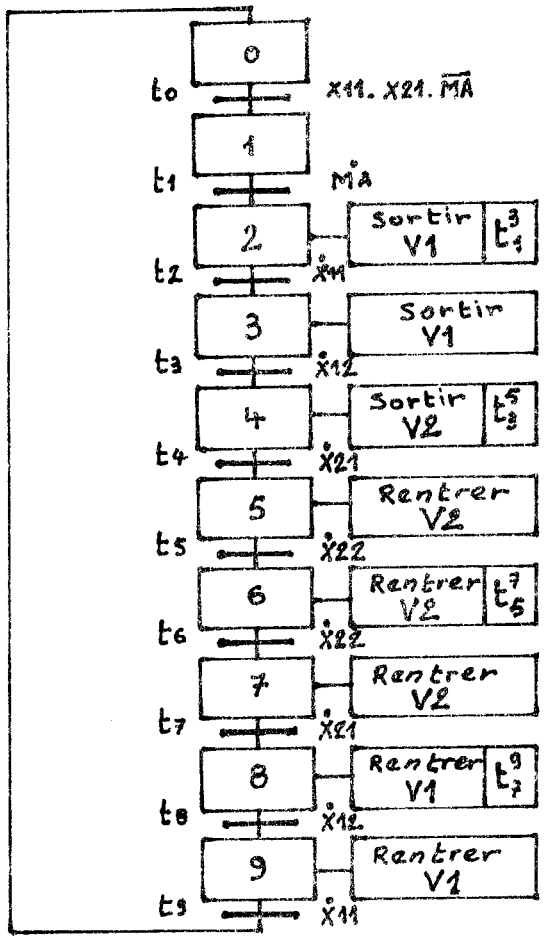


Figure I.3

EXEMPLE : Soit deux vérins V1 et V2 qui exécutent un cycle en L (fig.I.4). Aux capteurs VR1 et VS1 (vérin 1 rentrée et vérin 1 sortie)les deux variables X11 et X12, de même pour les capteurs VR2 et VS2, on associe X21 et X22. La description affinée de ce système est donnée par la (fig.I.5).



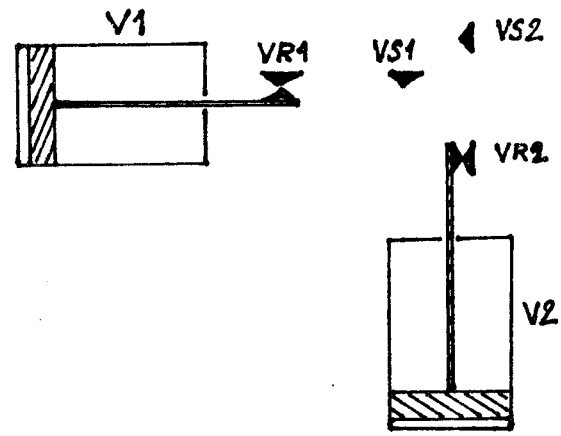


Figure I.4 : Processus

Figure I.5: Description affinée du système

b) Détection et localisation d'un défaut sur un capteur ou actionneur

De la description de fonctionnement introduite ci-dessus, nous sommes en mesure de détecter les anomalies de fonctionnement dans un système logique et cela à partir de la surveillance des comptes-rendus issus de la partie opérative du système.

b.1 - Détection d'une panne

Pour chaque état représenté par le grafcet (fig.I.5), il est possible de connaitre l'ensemble des entrées pour lesquelles, une modification de niveau est acceptable voir attendue :
un défaut entraînant une modification intempestive sur une entrée X_i s'exprime alors par :

$$M_i = \dot{X}_i \cdot \overline{\sum_{v_j \varepsilon v_i} v_j}$$

Pour l'ensemble du système et en tenant compte des temporisations " chien de garde ", la fonction globale de défaut peut s'écrire :

$$F = \sum_{i=1}^{N} M_i + \sum_{k=1}^{P} FT_K$$

N : est le nombre des transitions du garfcet

P : est le nombre des temporisations de " chien de garde "

FT_K représente la sortie de la temporisation k

v_j représente la fonction de validation qui prend la valeur 1 si la transition t_j est validée (dans le cas où il est affecté à chaque étape k une variable interne E_k : $v_j = \Pi_k E_k$, π : ET logique entre toutes les étapes qui participent à la validation de t_j).

v_i représente l'ensemble des fonctions de validation des transitions dont l'évènement est associé à $\dot{X}_i$

Pour l'exemple précédent (fig.4) les équations de défauts sont :

$$M_{11} = \dot{X}_{11} \cdot (\overline{E_2 + E_9}) + Ft_1^3,\ M_{12} = \dot{X}_{12} \cdot (\overline{E_3 + E_8}) + Ft_7^0$$

$$M_{21} = \dot{X}_{21} \cdot (\overline{E_4 + E_7}) + Ft_3^5,\ M_{22} = \dot{X}_{22} \cdot (\overline{E_2 + E_6}) + Ft_5^7$$

La fonction globale de détection de défauts dans le système (fig.I.4) s'écrit :

$$F = M_{11} + M_{12} + M_{21} + M_{22}$$

La mise à 1 de cette fonction F met en évidence la présence d'un défaut dans le système. Ce défaut est détecté par :

- soit la modification normalement improbable du niveau d'une ou plusieurs variables d'entrée.
- soit la temporisation qui permet de détecter des collages de fin de course, ou de blocage d'actionneurs.

Tous les défauts ne sont pas détectés par la méthode, notamment :

- tout défaut qui génère un compte-rendu conforme au fonctionnement normal dans un temps inférieur ou égal à la valeur habituelle, n'est pas détecté.
- tout défaut sur une variable qui peut évoluer en permanence,est également indécelable

b.2 - Localisation d'un défaut

Pour permettre une intervention efficace sur le système devenu défectueux à la suite d'une panne, la localisation des pannes est plus que nécessaire.

On sait par l'étape de détection, qu'un défaut sur une entrée X_i se manifeste par :

- *une modification non attendue de* $\dot{X}_i$
- *une variation de l'état d'un capteur accessible après passage sur celui-ci par maintien de l'action en cours.*
- *le positionnement de l'indicateur de fin de temporisation.*

Il est donc possible d'établir de la description affinée du cahier des charges du système, des fonctions combinatoires de localisation d'un défaut possible sur variable d'entrée X_i . Pour illustrer cette étape de localisation, considérons l'exemple de la (fig.I.5) et supposons un défaut sur l'entrée X_{11}. Il peut se manifester par :

- la modification de $\dot{X}_{11}$ alors que l'étape 2 ou 3 ne sont pas actives
- la modification de $\dot{X}_{12}$ alors que l'étape 2 est active
- le positionnement de la temporisation Ft_7^3 si l'étape 9 est active

Ceci peut se résumer par la fonction combinatoire de localisation :

$$D_{11} = \dot{X}_{11}\,(\overline{E_2 + E_9}) + \dot{X}_{12} \cdot E_2 + Ft_7^9 \cdot E_9$$

La mise à 1 de D_{11} conditionne l'émission d'un message de localisation du défaut : " vérifier le capteur VR1 et sa ligne " de même lorsque : $Ft_7^9 \cdot E_9 = 1$ il fait émettre en plus le message de localisation " la tige de V1 ne sort plus ".

Cette phase de localisation ne peut être validée que si elle est précéder de la phase de détection.

Le traitement de la fonction F de détection est en permanence validée, et sa mise à UN, signe de la présence d'une défaillance déclenche la procédure de localisation qui se matérialise par le traitement des fonctions combinatoires de localisation. La méthode de DEFRENNE rappelée ci-dessus permet le test en ligne des capteurs et actionneurs de la partie opérative d'un système logique. Ce test en ligne se matérialise par la détection puis la localisation de défauts affectant les capteurs ou actionneurs ou leur liaison avec la partie commande. Il se réalise par la surveillance de la variation du niveau des variables des comptes-rendus.

La mise en oeuvre de la méthode présente néanmoins, un handicap pour des systèmes logiques de dimension industrielle où le nombre des entrées est important. Ce handicap peut se justifier par les points suivants :

. l'intégration des considérations liées à la sécurité dans la description fonctionnelle, augmente la difficulté de l'analyse.

. le traitement des fonctions combinatoires de détection et de localisation nécessite un temps important pouvant être prohibitif. Toutefois seule la détection des défauts doit être exécutée en temps réel. Le traitement de la localisation s'effectuera après arrêt de la machine.

Des méthodes d'implantation sur machines programmées (automate spécialisé) permettent un traitement limité aux transitions susceptibles d'être franchises [SUA.78] . Il sera parfois indispensable d'y recourir pour réduire le temps de traitement.

I.3.2.3. - Conclusions

Les méthodes que nous avons résumées ci-dessus, se trouvent dépendantes de l'outil de synthèse des systèmes logiques. Ainsi les méthodes de DIAZ et TOHMA supposent une synthèse à partir d'une table d'état. Chaque état étant codé, (code K parmi n) pour établir les équations combinatoire de fonctionnement.

La méthode de MARIN part d'une description par réseau de Petri et celle de DEFRENNE part d'une description par réseau de Petri ou par Grafcet.

Notre objectif dans la suite du mémoire est de proposer une méthodologie de test et de diagnostic d'un système logique qui soit indépendant de la description de la partie commande.

Le choix de la description de la partie opérative comme la base de cette méthodologie de test et de diagnostic, est motivée par les considérations suivantes :

. l'indépendance de la méthode de l'outil de synthèse de la partie commande

. la fréquence des pannes dans les éléments de la partie opérative est plus importante que celle des pannes dans les éléments de la partie commande, qui sont en général plus fiables. Mais à cause aussi et surtout de l'environnement dans lequel les éléments de la partie opérative évoluent. C'est un environnement qui favorise les défauts (poussières, usures, échauffement contraintes mécaniques, électriques, parasites).

Cette deuxième considération nous amène donc à opérer une surveillance accrue des éléments de la partie opérative en exploitant les informations données par une description de la partie opérative.

CHAPITRE II

DESCRIPTION ET REPRÉSENTATION DE LA PARTIE OPÉRATIVE D'UN SYSTÈME LOGIQUE

CHAPITRE II

DESCRIPTION ET REPRÉSENTATION DE LA PARTIE OPÉRATIVE D'UN SYSTÈME LOGIQUE

CHAPITRE II

DESCRIPTION ET REPRÉSENTATION DE LA PARTIE OPÉRATIVE D'UN SYSTÈME LOGIQUE

La simulation de la partie opérative en temps réel constitue une redondance active à partir de laquelle une méthodologie de test et de diagnostic sera définie. Cette dernière permettra une détection et une localisation rapide de toute défaillance dans les éléments de la partie opérative d'un système logique.

Cette simulation nécessite la description de la partie opérative. Dans ce chapitre, nous étudions une description de la partie opérative. L'élément de base de cette description sera la définition d'une action élémentaire.

Nous établissons aussi des règles de décomposition permettant de décomposer toute action associée à un actionneur en actions élémentaires.

A la suite de ces règles, nous présentons une procédure globale de décomposition de la partie opérative en actions élémentaires. Enfin, nous terminons cette étude par l'énoncé des règles d'exploitation des éléments de cette description.

II.1 - CLASSIFICATION DES ELEMENTS D'UN SYSTEME LOGIQUE INDUSTRIEL

Nous avons vu au chapitre précédent (I.3.1) que le système logique industriel peut être décomposé en deux éléments :

- la partie commande qui gère le processus
- la partie opérative qui effectue les tâches

L'ensemble du système est placé dans un environnement qui inclut d'autres machines et des opérateurs.

II.1.1. - Partie opérative/Partie commande

Dans la pratique, cette classification n'est pas évidente car les grandeurs appliquées aux actionneurs sont élaborées à plusieurs niveaux :

a) le premier niveau correspond aux sorties de l'organe de traitement (automate) qui pilote la partie opérative.

b) le second niveau comprend les préactionneurs qui effectuent une adaptation de l'information en vue de son exploitation par les actionneurs.

Un traitement est fréquent au niveau des préactionneurs. Il peut être combinatoire dans le cas des ordres "sécuritaires" (arrêt d'urgence, détection phase coupée ..., par exemple ou même séquentiel (interfaces types bistables tel que distributeurs, contacteur autoentretenu ....).

Pour que la simulation soit valable, il faut que toutes les grandeurs de traitement soient prises en compte. Ceci conduit à des schémas différents, illustrés par les figures (II.1a,b).

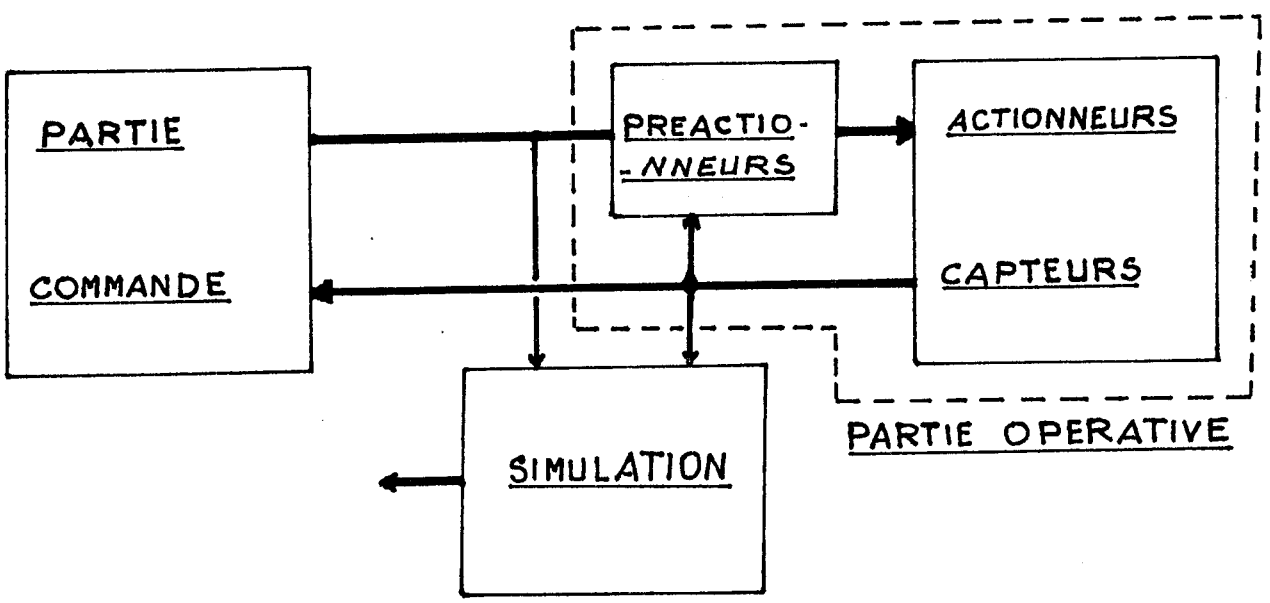


Figure II.1a

Partie opérative inclue les préactionneurs.

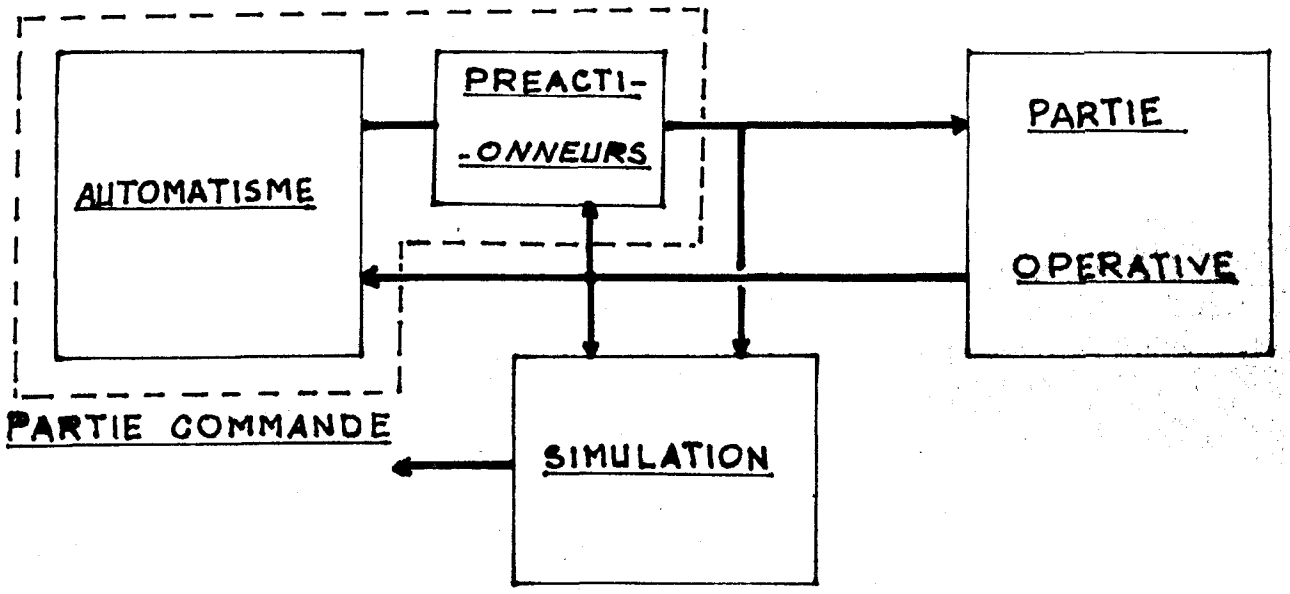


Figure II.1b : Partie commande inclue les préactionneurs

Une étude du même type peut être menée au niveau de la prise en compte de l'environnement.

II.1.2. - Dialogue homme/machine ou machine/machine

Les interventions de l'environnement (opérateur, autre machine) peuvent agir au niveau de l'automate de commande ou au niveau de préactionneurs (ou les deux).

La simulation de la partie opérative est acceptable si elle utilise les grandeurs directement appliquées aux actionneurs.

Par contre si on traite les ordres issus de l'automate de commande, il faut prendre en compte les grandeurs directement appliquées aux préactionneurs ainsi que la nature du traitement que ces derniers effectuent.

Nous avons également vu au paragraphe I.3.3. que les entrées correspondantes à une procédure d'échange unidirectionnelle (monologue) ne permettent pas l'introduction de test de vraissemblance.

Nous proposons de créer le plus souvent possible un dialogue de type appel/réponse entre l'opérateur et la machine. Cela permet une amélioration de la sûreté en plusieurs points. Ce dialogue permet :

- une vérification de type causale donc une vérification du vecteur de communication.
- un contrôle de la vigilance de l'opérateur (temps de réponse contrôlé)
- une limitation des erreurs de conduite puisque l'opérateur répond à la machine dans un contexte imposé.

Dans ce chapitre, nous allons s'intéresser exclusivement à la partie opérative tel qu'elle sera décrite ci-dessous.

II.2 - CONSTITUTION DE LA PARTIE OPERATIVE

La partie opérative est le processus physique à automatiser. Elle est constituée notamment de deux ensembles .

Ensemble $\mathscr{C}$ des capteurs

Ce sont des éléments de saisie de l'information. Par leur disposition spaciale, ils définissent l'espace d'évolution du processus. Par une configuration de leur état, ils traduisent une évolution spaciau-temporelle de celui-ci.

Cette évolution est le résultat d'une sollicitation physique d'un ou plusieurs éléments moteurs et dynamiques appartenant à l'ensemble des actionneurs.

Ensemble $\mathscr{A}$ des actionneurs

Les actionneurs sont des éléments de puissance. Leur fonction dans la partie opérative d'un système est définie avec précision dans un cahier des charges établi à l'avance. Elle consiste à exécuter des actions physiques (fig.II.2) selon les ordres reçus de la partie commande.

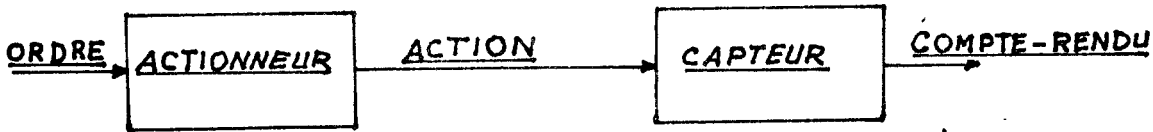


figure II.2 : Définition d'une action

REMARQUE : Nous prendrons en compte l'ensemble des grandeurs qui agissent sur les actionneurs. Nous appelerons ordre la grandeur qui provoque de façon certaine une action.
Dans la suite de ce chapitre, les grandeurs ordre et action seront confondues.

II.3 - DECOMPOSITION DE LA PARTIE OPERATIVE

Dans la partie opérative, chaque actionneur est sensible à un certain nombre de variables binaires. Chaque variable binaire correspond à un ordre élémentaire émanant de la partie commande. A partir des ordres élémentaires, il est possible de définir un ensemble de fonctions booléennes qui forment les commandes relatives à cet actionneur.

Soit $a \in \mathcal{A}$, C_a l'ensemble des commandes relatives à l'actionneur a.

L'actionneur $a \in \mathcal{A}$ soumis à une commande appartenant à C_a, réalise une action caractérisée par une alternance, d'états contrôlés, d'états intermédiaires et une vitesse.

ETAT CONTROLE : Un état est dit contrôlé s 'il existe une fonction booléenne, définie à partir d'un certain nombre de capteurs qui soit caractéristique de cet état.

Pour qu'un contrôle soit possible, il doit exister une bijection entre l'état contrôlé et la fonction booléenne associée.
Nous noterons $\mathcal{E}_a^i$ la fonction booléenne associée à l'état contrôlé i de l'actionneur $a \in \mathcal{A}$.

ACTION ELEMENTAIRE : Lorsque l'actionneur $a \in \mathcal{A}$, occupant le $i^{ème}$ état contrôlé ($\mathcal{E}_a^i$), soumis à une commande appartenant à C_a, passera par un état intermédiaire pour atteindre le $(i+1)^{ème}$ état contrôlé ($\mathcal{E}_a^{i+1}$), s'il existe.

Dans ces conditions, nous dirons que l'actionneur a $\epsilon \mathcal{A}$ réalise une action élémentaire.

Le $i^{ème}$ état contrôlé sera appelé état contrôlé initial de l'action élémentaire considérée.

Le $(i+1)^{ème}$ état contrôlé sera appelé état contrôlé final de cette action élémentaire.

Nous noterons alors pour toute action élémentaire N associée à un actionneur a $\epsilon \mathcal{A}$

. l'état contrôlé initial de N par : $\mathcal{E}_N^I \epsilon \mathcal{E}_a$

. l'état contrôlé final de N par : $\mathcal{E}_N^f \epsilon \mathcal{E}_a$

. la commande de N par : $C_N \epsilon C_a$

L'action élémentaire N peut être considérée comme une application définie par :

$$N : = \mathcal{E}_a \times C_a \longrightarrow \mathcal{E}_a$$

La généralisation à l'ensemble de la machine donne pour

$$\mathcal{E} = \bigcup_{a \epsilon \mathcal{A}} \mathcal{E}_a \; ; \; C = \bigcup_{a \epsilon \mathcal{A}} C_a$$

$$N : = \mathcal{E} \times C \longrightarrow \mathcal{E}$$

N est une action élémentaire quelconque.

Appelons $\mathcal{N}$ l'ensemble de toutes les actions élémentaires qui définissent par leur réalisation, l'évolution de la partie opérative. Cette dernière peut être décrite par le triplet :

$$\mathcal{P} = (\mathcal{A}, \mathcal{E}, \mathcal{N})$$

II.2.1. - Règles de décomposition

Une action p générée par un actionneur est décomposée en actions élémentaires, définies par :

. une commande $\underline{G}_{pk}$ émanant de la partie commande et/ou de l'environnement.

. un état contrôlé initial $\mathcal{C}_p^I \in \mathcal{C}_p$

. un état contrôlé final $\mathcal{C}_p^f \in \mathcal{C}_p$

Trois règles régissent la décomposition d'une action générée par un actionneur, en actions élémentaires.

R1 ** — Une action générée par un actionneur et contrôlée par un sous-ensemble de capteurs, est décomposée en actions élémentaires de façon à ce que toute modification des comptes-rendus relatifs à cette action, autre que celle qui traduit la disparition de l'état initial, est considérée comme état contrôlé final.

R2 ** — Lorsque l'état initial et l'état final ne sont pas distinguables, la disparition de l'état initial sera pris en compte pour créer deux actions élémentaires.

R3 ** — La disparition de l'état initial sera également pris en compte lorsque l'état final n'est pas contrôlable.

II.2.2. - Procédure de décomposition de la partie opérative en actions élémentaires

Pour chaque action p, il est pratiqué une décomposition en actions élémentaires comme suit :

1ère phase : a) Pour tout état contrôlé initial $\mathcal{E}_p^I$ de $\mathcal{E}_p$, il est recherché les ordres caractéristiques C_{pK} pris dans C_p qui permettent de quitter $\mathcal{E}_p^I$.

b) Pour chaque couple $\mathcal{E}_p^I$, C_{pK} ainsi trouvé, il est recherché les états contrôlés finaux $\mathcal{E}_p^f$

c) Pour tout couple $\mathcal{E}_p^I$, C_{pK} qui n'a pas d'état final, l'absence de l'état initial $\mathcal{E}_p^I$ sera pris comme état final (règle R3).

d) Pour tout couple $\mathcal{E}_p^I$, C_{pK} qui a l'état final égal à l'état initial $\mathcal{E}_p^I$, nous créerons deux actions élémentaires pseudo-contrôlées (règle R2) en prenant la disparition de l'état initial comme état final de l'une et comme état initial de l'autre.

2ème phase : a) Pour tout état contrôlé final $\mathcal{E}_p^f$, il y a lieu de chercher les ordres caractéristiques C_{pK} qui permettent de l'atteindre.

b) Pour chaque couple $\mathcal{E}_p^f$, C_{pK} ainsi obtenu, il y a lieu de rechercher l'état initial contrôlé $\mathcal{E}_p^I$.

c) Pour tout couple $\mathcal{E}_p^f$, C_{pK} à qui $\mathcal{E}_1^I$ n'existe pas une nouvelle action élémentaire est créée (R3).
Pour tout couple $\mathcal{E}_p^f$, C_{pK} a qui $\mathcal{E}_f^I$ existe l'action élémentaire a déjà été trouvée dans la 1ère phase.

A la fin de la 2ème phase, on obtient l'ensemble des actions élémentaires, chaque action élémentaire est représentée par un triplet :

$$(\mathcal{E}_p^I , C_{pK} , \mathcal{E}_p^f)$$

A chaque couple $\mathcal{E}_p^I$, C_{pK}, il correspond en général un seul état final $\mathcal{E}_p^f$. Toutefois, s'il existe plusieurs états finaux, il est impossible de les discriminer par avance (car autrement les éléments $\mathcal{E}_p^I$, C_{pK} devraient en tenir compte).

Dans ce cas, nous grouperons ces actions élémentaires en une seule en prenant comme état final la réunion des états finaux de chacune. Ceci est bien sûr à éviter car la sécurité est diminuée. Ce cas conduit à la troisième phase de la décomposition.

3ème phase : Toutes les actions élémentaires qui ne différent que par leur état final, sont regroupées en une action élémentaire représentée par la triplet :

$$(\mathcal{E}_p^I , C_{pK} , \mathcal{E}'^f_p) \quad \text{avec}$$

$$\mathcal{E}'^f_p = \bigcup_f \mathcal{E}_p^f$$

II.4 - REPRESENTATION D'UNE ACTION ELEMENTAIRE

Une action élémentaire $N_j \in \mathcal{N}$ est par définition associée à un actionneur $a \in \mathcal{A}$. Elle est caractérisée par :

a' Une fonction de commande C_{N_j} définie sur les ordres élémentaires qui participent à l'activation de l'action élémentaire N_j.

b' Une fonction initiale $\mathcal{E}^I_{N_j}$ définie sur les éléments de $\mathcal{E}_a$ qui contrôlent l'état contrôlé initial de N_j.

c/ Une fonction finale $\mathcal{C}^f_{N_j}$ définie sur les éléments de $\mathcal{E}_a$ qui contrôlent le ou les états finaux de l'action élémentaire.

d/ un scalaire T_{N_j} qui correspond au temps moyen de réalisation de l'action élémentaire N_j

e/ un scalaire ΔT_{N_j} qui traduit la tolérance admissible sur le temps de réalisation T_{N_j} de l'action élémentaire N_j.

Les éléments cités ci-dessus, permettent de décrire avec précision l'action élémentaire N_j. Dorénavant, celle-ci sera représentée par le quintriplet:

$$N_j = (\mathcal{C}^I_{N_j} , G_{N_j} , \mathcal{C}^f_{N_j} , T_{N_j} , \Delta T_{N_j}) \qquad (II.1)$$

$$N_j \in \mathcal{N}$$

La définition qui vient d'être faite de l'action élémentaire sera la base de la description de la partie opérative d'un système logique.

Pour que cette description soit possible sous la forme (II.1), une analyse et une étude de la partie commande sera supposée préalablement effectuée, car c'est à partir d'elle que la fonction de commande de chaque action élémentaire sera déterminée et ceci indépendamment de l'outil de description de la partie commande.

Quand aux paramètres T et ΔT de chaque action élémentaire, nous discuterons plus loin des méthodes permettant leur évaluation. En attendant, nous supposerons que leurs valeurs sont connues.

Nous allons illustrer la description de la partie opérative d'un système logique par un exemple. Chaque action élémentaire sera décrite par la forme (II.1).

EXEMPLE : Soit le cahier des charges suivant | BOS.79 | . Une tête de perçage coulissante sur un bâti métallique (fig. II.3). Le cycle de perçage se fait avec débourrage. Il est décrit par la (fig.II.4):

- la broche tourne en permanence
- l'opérateur ayant fixé la pièce, donne alors l'information départ de cycle (DY).
- après une approche à grande vitesse (DGV) jusqu'à la position (b1). Le perçage s'effectue à petite vitesse (DPV) jusqu'à (b2), suivie d'une montée à grande vitesse (MGV) jusqu'à (b1) et enfin descente à petite vitesse (DPV) jusqu'à (b3). La remontée de la broche se fait à grande vitesse (MGV) jusqu'à (h).

Le grafcet de la(fig.II.5) représente ce cahier des charges.

L'analyse de la partie opérative de ce système (fig.II.3) met en évidence les actions élémentaires présentées dans la table suivante (fig.II.6).

Chaque action élémentaire est décrite suivant le modèle (II.1).

Identification action élément	Vecteur $\mathcal{E}_N^I$ état initial	Vecteur commande $\mathcal{C}_N$	Vecteur $\mathcal{E}_N^f$ état final	Temps moyen de réalisation T_N	Tolérance sur T_N : ΔT
N1	h	DGV	b1	T1	ΔT1
N2	b1	DPV	b2	T2	ΔT2
N3	b2	MGV	b1	T3	ΔT3
N4	b2	DPV	b3	T4	ΔT4
N5	b3	MGV	b2	T5	ΔT5
N6	b1	MGV	h	T6	ΔT6

figure II.6 - Description de la partie opérative

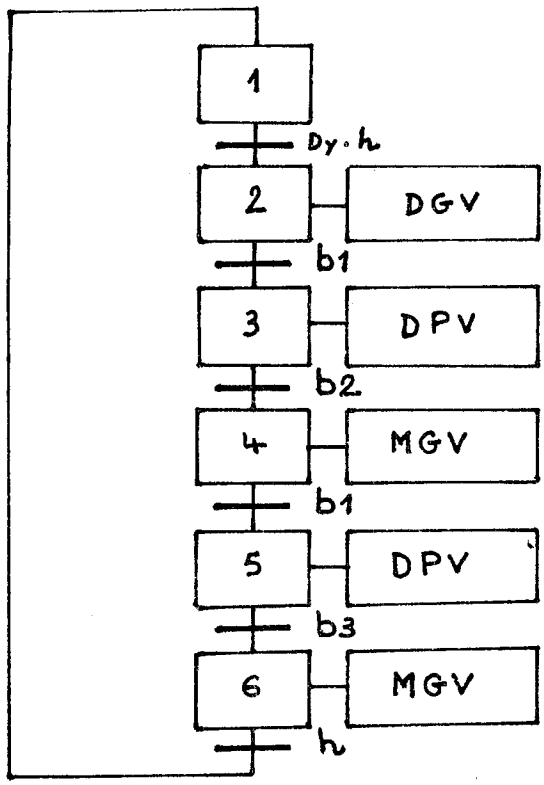


Figure II.5 : Grafcet de commande

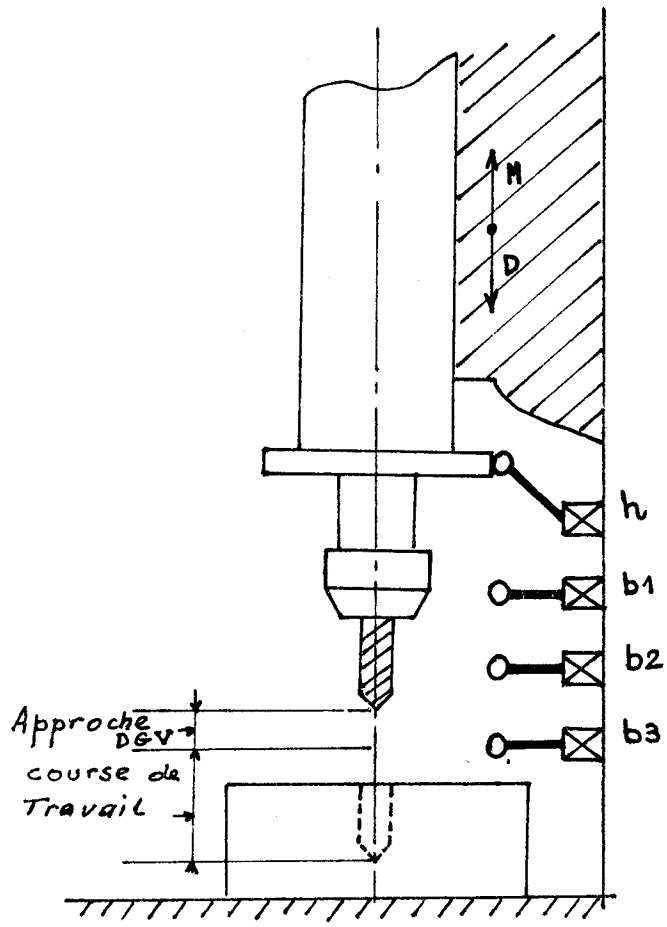


Figure II.3 : Processus

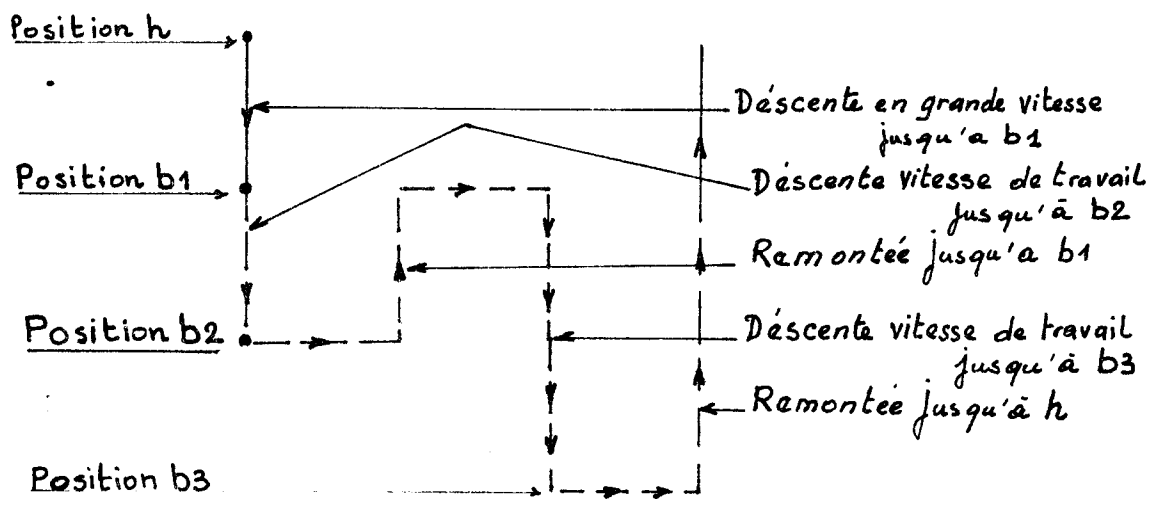


Figure II.4 : Cycle de perçage

REMARQUE : Dans cet exemple, toutes les actions élémentaires décrites sont contrôlées par un état initial et un état final. Il n'en est pas toujours ainsi dans la réalité. Il existe des actions élémentaires qui sont soient partiellement contrôlées, soient sans commandes. Elles constituent des cas particuliers, nous allons les étudier en mettant en oeuvre l'application des règles de décomposition R2 et R3.

II.5 - ETUDE DE QUELQUES CAS PARTICULIERS D'ACTIONS ELEMENTAIRES

Jusqu'à présent, nous nous sommes intéressés à des actions élémentaires contrôlées, c'est-à-dire en terme de la description par le modèle (II.1) :

. la fonction initiale $\mathcal{E}^I_{N_j} \neq 0$

. la fonction finale $\mathcal{E}^f_{N_j} \neq 0$

Dans la réalité industrielle, on peut trouver des situations dans lesquelles une ou plusieurs actions élémentaires peuvent être décrites suivant le modèle (II.1) mais avec soit :

. la fonction $\mathcal{E}^I_{N_j} = 0$

. la fonction $\mathcal{E}^f_{N_j} = 0$

. la commande implicite $C_{N_j} = 0$

Nous allons étudier ces trois cas et voir dans quelle mesure le modèle (II.1) peut leur être adapté.

Cette étude se fera dans l'hypothèse suivante :

$$N_j \in \mathcal{N} \qquad |\mathcal{E}^I_{N_j}| = |\mathcal{E}^f_{N_j}| = |C_{N_j}| = 1 \qquad \text{(II.2)}$$

Dans le cas général, les transformations qui seront opérées ne sont pas valables car chaque vecteur a plusieurs composantes.

Des exemples choisis guideront et illustreront cette étude.

II.5.1. - Contrôle de l'état initial ou final d'une action élémentaire non définie

Toute action élémentaire représentant ces deux situations, est décrite soit par :

$$a/\quad N_j = (\bullet, C_{N_j}, \mathcal{E}^f_{N_j}, T_{N_j}, \Delta T_{N_j})$$

ou

$$b/\quad N_j = (\mathcal{E}^I_{N_j}, C_{N_j}, \bullet, T_{N_j}, \Delta T_{N_j})$$

A priori, cette description permet d'explorer l'état des éléments physiques (capteur final et actionneur pour le cas a; capteur initial et actionneur pour le cas b) qui participent à sa réalisation. Toutefois, nous adoptons pour ces cas la même description que celle décrite par le modèle (II.1). Autrement toute action élémentaire de type (a) ou (b) sera décrite par :

$$N'_j = (\mathcal{E}^I_{N'_j}, C_{N_j}, \mathcal{E}^f_{N_j}, T_{N_j}, \Delta T_{N_j}) \text{ avec} \qquad (II.3)$$

- pour le cas a : $\mathcal{E}^I_{N'_j} = \overline{\mathcal{E}^f_{N_j}}$

- pour le cas b : $\mathcal{E}^f_{N'_j} = \overline{\mathcal{E}^I_{N_j}}$

On appellera toute action élémentaire de type (II.3) une action élémentaire pseudo-contrôlée. Cette solution apporte les avantages suivants :

- la description de la partie opérative sera uniforme car toute action élémentaire est représentée suivant le même modèle (II.1) . L'exploitation de cette description sera de ce fait pratique et aisée.

- le fait de ramener le non contrôle de l'état initial (final) d'une action élémentaire à un contrôle explicite utilisant l'état de l'élément final (initial) permet une meilleure surveillance et un contrôle efficace des éléments physiques de l'action élémentaire considérée.

EXEMPLE : Soit une came C entrainée en rotation par un moto-réducteur R (fig.II.7). Cette came doit effectuée un tour et un seul à partir de la position d à chaque fois que l'ordre lui en est donné.

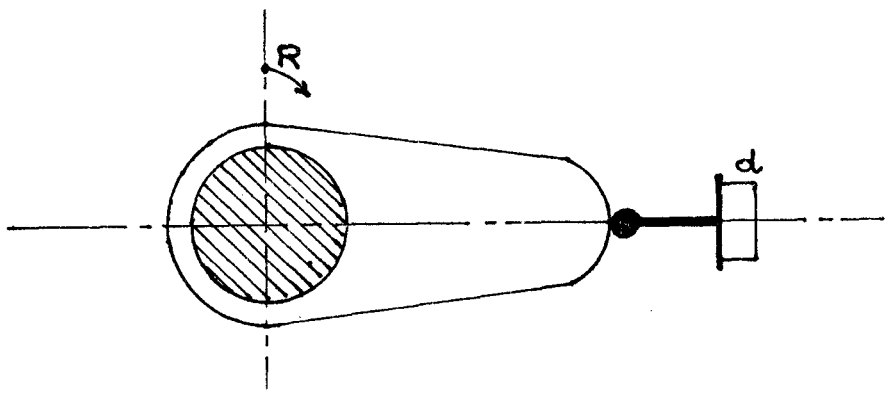


Figure II.7 : Rotation d'une came

La partie opérative de ce système peut être décrite par une seule action élémentaire N_j tel que :

$$N_j = (d, R, d, T_{N_j}, \Delta T_{N_j})$$

L'ambiguité de cette description vient du fait que :

$$\mathcal{E}^{I}_{N_j} = \mathcal{E}^{f}_{N_j}$$

l'état initial $\mathcal{E}^{I}_{N_j}$ et l'état final sont indistinguables par conséquent nous appliquons la règle de décomposition R2 à l'action élémentaire N_j. On trouve deux actions élémentaires N'_j et N''_j tel que :

$$N'_j = (\mathcal{E}^{I}_{N'_j}, C_{N_j}, \bullet, T_{N'_j}, \Delta T_{N'_j})$$

avec

$$\mathcal{E}^{I}_{N'_j} = \mathcal{E}^{I}_{N_j} = d$$

$$N''_j = (\bullet, C, \mathcal{E}^{f}_{N''_j}, T_{N''_j}, \Delta T_{N''_j})$$

avec

$$\mathcal{E}^{f}_{N''_j} = \mathcal{E}^{f}_{N_j} = d$$

On remarque que ces deux actions élémentaires N'_j et N''_j sont partiellement contrôlées par conséquent, pour les rendre totalement contrôlées, nous appliquons la transformation (II.3). Cela donne :

$$N'_j = (d, R, \bar{d}, T_{N'_j}, \Delta T_{N'_j})$$

et

$$N''_j = (\bar{d}, R, d, T_{N''_j}, \Delta T_{N''_j})$$

Ces deux actions élémentaires N' et N" constituent donc la description de la partie opérative du système (fig.II.7). Elles sont conformes au modèle décrit par (II.1).

REMARQUE : Au terme de la décomposition de l'action N_j en deux actions élémentaires N'_j et N''_j, les temps de réalisations de ces actions doivent vérifier la relation temporelle :

$$T_{N_j} \approx T_{N'_j} + T_{N''_j} \text{ avec } T_{N''_j} >> T_{N'_j}$$

En regardant la description de l'action élémentaire N'_j, on remarquera que son temps de réalisation $T_{N'_j}$ correspondra approximativement au temps de réponse du capteur de l'état initial. C'est un temps en général très court et de ce fait la surveillance de l'action élémentaire N'_j sera très difficile. Elle reste néanmoins possible si le temps de traitement (T_D) du dispositif de surveillance vérifie la contrainte temporelle suivante :

$$T_{N'_j} - \Delta T_{N'_j} > T_D \qquad \text{(II.4)}$$

Nous discuterons dans le chapitre suivant l'importance de cette contrainte (II.4) et son impact sur les performances du dispositif de surveillance.

II.5.2. - Action élémentaire à commande implicite $G_{N_j} = 0$

Dans ce cas, la commande est en général générée par un dispositif de rappel interne à l'actionneur (actionneur monostable, ...) . La génération de cette commande est donc conditionnelle à la disparition d'une fonction de commande antérieur du même actionneur. Par conséquent, on peut considérer que l'action élémentaire dont il est question est à commande implicite. La description d'une telle action est la suivante :

$$N_j = (\mathcal{E}^I_{N_j} , \bullet , \mathcal{E}^f_{N_j} , T_{N_j} , \Delta T_{N_j})$$

La transformation qui sera opérée sur cette description, consiste à rendre la commande explicite. On obtient ceci, en considérant la disparition du vecteur de commande $C_{N'_j}$ d'une action élémentaire N'_j antérieure, générée par l'actionneur. Dans ces conditions, la description de N_j devient :

$$N_j = (\mathcal{E}^I_{N_j} , C_{N_j} , \mathcal{E}^f_{N_j} , T_{N_j} , \Delta T_{N_j}) \qquad (II.5)$$

$$C_{N_j} = \overline{C}_{N'_j}$$

EXEMPLE : On considère le système décrit par la (fig.II.8) et où l'actionneur est un monostable.

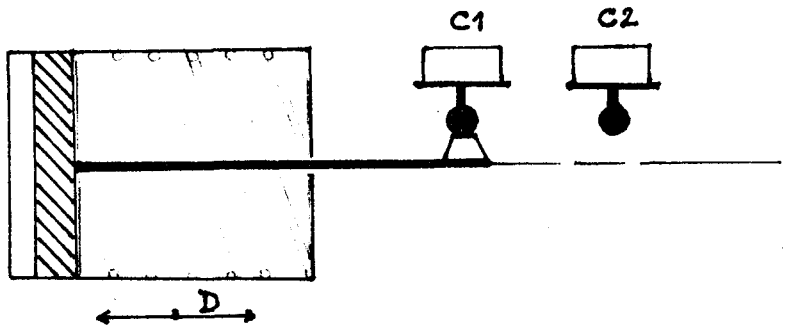


Figure II.8 : Actionneur monostable

La description de la partie opérative de ce système comporte deux actions élémentaires :

$$N_1 = (C1, D, C2, T_{N_1}, \Delta T_{N_1})$$

$$N_2 = (C2, \bullet, C1, T_{N_2}, \Delta T_{N_2})$$

N_2 est une action élémentaire à commande implicite. En effet la disparition de la commande D conditionne la génération de l'action élémentaire N_2. Sa description sera donc conforme à la transformation (II.5) soit :

$$N_2 = (C2, \overline{D}, C1, T_{N_2}, \Delta T_{N_2})$$

Grâce aux transformations (II.3) et (II.5), la partie opérative peut être représentée par une description uniforme des actions élémentaires. Cette uniformité apporte une meilleure exploitation de la partie opérative surtout au niveau du traitement de la surveillance des actions élémentaires.

Toutefois, le traitement de la surveillance des actions élémentaires pseudo-contrôlées (issues d'une action élémentaire dont l'état initial n'est pas défini), peut poser des problèmes dans des situations particulières. Pour remédier à ces cas de figures, un traitement spécifique est envisagé au niveau du traitement de la phase de validation des actions élémentaires.

Dans la description de la partie opérative qu'on vient de présenter, nous avons supposé que les paramètres temporels $(T,\Delta T)$ des actions élémentaires sont connus.

Nous discutons ci-après deux approches de leur évaluation.

II.6 - EVALUATION DES PARAMETRES TEMPORELS DES ACTIONS ELEMENTAIRES

Pour achever la description de chaque action élémentaire, il faut déterminer les valeurs de T et ΔT.

Pour se faire,une campagne de mesure des temps de réalisation s'avère nécessaire. Si la répartition des temps de réalisation mesurés pour chaque action élémentaire, suit une loi Gaussienne, nous pourrons prendre le temps moyen comme valeur du paramètre T et évaluer le paramètre ΔT par l'expression :

$$\Delta T = \alpha\ \sigma$$

σ représente l'écart type et α un coefficient de confiance choisi.
Le choix du coefficient de confiance α, influence les performances de la surveillance, ainsi :

. pour α très grand la sécurité du système se trouve diminuée.

. pour α très petit, la disponibilité du système est affectée, car de fausses pannes peuvent être détectées. Il faut donc chercher et choisir une valeur de α qui réalise un bon compromis sécurité-disponibilité.

La détermination des valeurs des paramètres temporels (T, ΔT) dans les conditions précédentes, suppose que l'état final ($\mathscr{E}_N^f$) de chaque action élémentaire, soit unique. Dans le cas où l'état final est multiple, il n'est plus possible d'évaluer les paramètres temporels comme précédemment.

L'acquisition des mesures des temps de réalisation constitue la phase la plus importante dans cette évaluation. Nous proposons deux méthodes d'acquisition des mesures. On appelera l'une : Méthode d'acquisition "HORS LIGNE" et l'autre méthode d'acquisition " EN LIGNE ".

a) Méthode d'acquisition "HORS LIGNE"

Dans cette méthode l'acquisition des mesures est faite hors ligne (le système n'est pas en fonctionnement normal).

Chaque action élémentaire est réalisée. Une temporisation est lancée pour évaluer la durée de sa réalisation.

Le nombre de mesures (n) est fixé à l'avance, l'acquisition sera supposée terminée lorsqu'on aura n mesures pour chaque action élémentaire. L'organigramme représenté par la (fig.II.9a) décrit les différentes phases de cette méthode.

Dans cette méthode, on considère que :

$$\forall_{N_j}, N_j \in \mathscr{N}_1 \quad N_j = (\mathscr{E}^I_{N_j}, C_{N_j}, \mathscr{E}^f_{N_j}, T_{N_j}, \Delta T_{N_j}) \; ; \; |\mathscr{N}| = M$$

et

$$\forall_{N_j}, N_j \in \mathscr{N} \quad \tau_j = \{ \tau_{j1}, \tau_{j2}, \cdots \tau_{jN} \} \; ; \; |\tau_j| = n$$

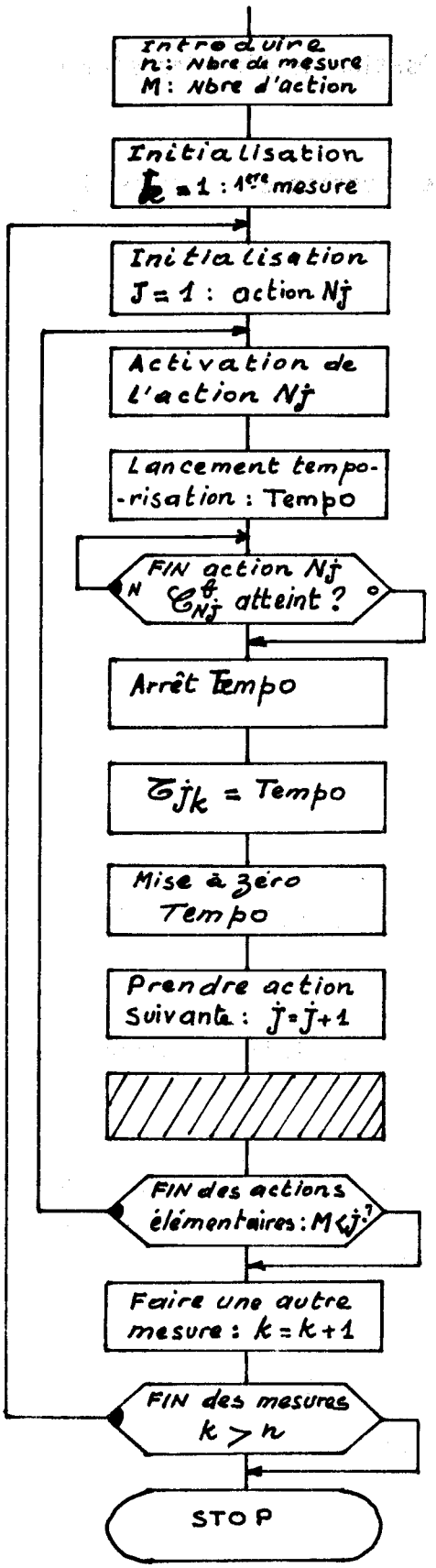


Figure II.9a : Acquisition des mesures hors ligne

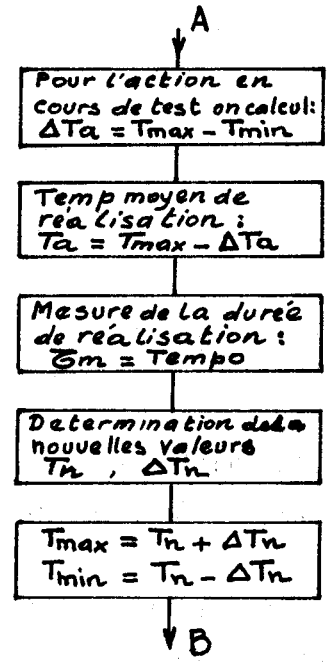


Figure II.9b : Acquisition des mesures en ligne

$\mathcal{N}$ = ensemble des actions élémentaires

τ_j = ensemble des mesures du temps de réalisation de l'action élémentaire N_j

τ_{jk} = $k^{ème}$ mesure du temps de réalisation de l'action élémentaire N_j

Dans ces conditions :

$$\forall N_j \, , \, N_j \in \mathcal{N} \quad , \quad T_j = \frac{\sum_{k=1}^{k=n} \tau_{jk}}{n}$$

et la tolérance sur T_j sera évaluée par :

$$\Delta T_j = \alpha \, \sigma_j$$

avec

$$\sigma_j = \left(\frac{1}{n} \sum_{k=1}^{N} \frac{1}{n} (\tau_{jk} - T_j)^2 \right)^{1/2}$$

b) Méthode d'acquisition " EN LIGNE "

L'actualisation des valeurs des paramètres temporels des actions élémentaires se fait au cours de la surveillance.

A chaque fois qu'une action élémentaire est réalisée normalement, ses paramètres temporels sont réajustés.

Ce réajustement (fig.II.8b) consiste à remplacer les anciennes valeurs des paramètres temporels (T_a, ΔT_a) de l'action élémentaire considérée par de nouvelles valeurs (T_{N_j}, ΔT_{N_j}).

(T_{N_j}, ΔT_{N_j}) sont déterminées à partir de (T_a, ΔT_a) et de la mesure (T_m) de la durée de réalisation de cette action élémentaire.

La détermination des nouvelles valeurs (T_{N_j} , ΔT_{N_j}) est obtenue par la mise en oeuvre de l'algorithme décrit dans l'annexe B. La mise en oeuvre de la méthode d'acquisition en ligne constitue un traitement supplémentaire pour le dispositif de surveillance.

L'évaluation des paramètres temporels des actions élémentaires est d'une importance capitale dans la surveillance de la partie opérative. La précision de cette évaluation contribue largement à améliorer la sécurité et la disponibilité du système.

Cette précision dépend de la méthode d'acquisition des mesures du temps. Dans la méthode d'acquisition, hors ligne, l'évaluation des paramètres temporels des actions élémentaires est faite une fois pour toute avant la mise en exploitation du système. La dégradation des caractéristiques des éléments physiques affecte la précision initiale de l'évaluation. Toutefois, la mise en oeuvre de la méthode est facile, mais assez longue. Dans la méthode d'acquisition en ligne, les paramètres temporels sont réajustés en permanence en cours d'exploitation. La précision d'évaluation est meilleure du fait de la prise en compte du vieillissement du matériel. La mise en oeuvre est plus difficile et les performances de la surveillance sont affectées du fait de l'augmentation relative du temps de traitement du dispositif de surveillance.

II.7 - CONDITIONS DE VALIDATION ET DE REALISATION DES ACTIONS ELEMENTAIRES

La surveillance de la partie opérative en vue de la détection de défaillance dans ses éléments physiques, passe par la surveillance de chaque action élémentaire qui la constitue.

Cette surveillance s'opère si les deux conditions suivantes sont remplies.

- L'action élémentaire est validée.
- L'action élémentaire est entrain de se réaliser si et seulement si elle est validée.

II.7.1. - Conditions de validation des actions élémentaires

Soit l'action élémentaire $N_j \in \mathcal{N}$

$$N_j = (\mathcal{E}^I_{N_j}, C_{N_j}, \mathcal{E}^f_{N_j}, T_{N_j}, \Delta T)$$

Pour que l'action élémentaire N_j commence à se réaliser, deux conditions sont nécessaires :

-la fonction de commande C_{N_j} devient active et soit appliqué à l'actionneur qui génère N_j.

- l'actionneur qui génère N_j occupe l'état initial décrite par la fonction $\mathcal{E}^I_{N_j}$ représentant les capteurs initiaux.

Ces deux conditions peuvent se résumer par l'équation combinatoire

$$V_{N_j} = C_{N_j} \cdot \mathcal{E}^I_{N_j} \quad (\ .\ \text{ET LOGIQUE}) \qquad (II.6)$$

V_{N_j} est appelé : condition de validation de l'action élémentaire N_j. L'action élémentaire N_j sera validée si : $V_{N_j} = 1$

Une action élémentaire validée est en cours de réalisation.

II.7.2. - Condition de réalisation d'une action élémentaire

En absence de pannes, une action élémentaire N_j validée suivant les conditions ci-dessus, est considérée comme réalisée si :

- l'actionneur $a_{N_j} \in \mathcal{A}$ générant N_j, sous l'effet de la fonction de commande C_{N_j}, ayant quitté l'état initial ($\mathcal{E}^I_{N_j}$) à l'instant to ; atteint l'état final ($\mathcal{E}^f_N$) à l'instant τ tel que :

$$to + T_{N_j} - \Delta T_{N_j} < \tau < to + T_{N_j} + \Delta T_{N_j} \qquad (II.7)$$

L'exploitation des deux conditions (II.6) et (II.7), au niveau de chaque action élémentaire, constitue une base de données à partir de laquelle la surveillance sera organisée. Nous étudierons dans le prochain chapitre la structure de cette base de données.

II.8 - CONCLUSION

Pour exploiter les informations contenues dans la partie opérative, nous nous sommes livrés tout le long de ce chapitre à une exploration de ces informations pour parvenir à une certaine méthodologie de description et de décomposition de la partie opérative.

Nous avons pris comme unité de cette description, l'action élémentaire définie à partir d'états contrôlés (définis par des capteurs) et de déplacement des actionneurs (ordre, temps) dans cet espace.

La description de la partie opérative ainsi opérée sera la base d'une simulation de celle-ci.

Cette simulation constituera une redondance logicielle active à partir de laquelle une méthode de test et de diagnostic des éléments de la partie opérative, sera proposée.

La mise en oeuvre de cette méthode permettra la détection et la localisation de tous les défauts affectants ces éléments. L'étude de la méthode de test et de diagnostic, ses performances vis à vis de la détection et de la localisation des défauts et de son implantation sur un dispositif de surveillance, fera l'objet du chapitre suivant .

CHAPITRE III

TEST ET DIAGNOSTIC DE LA PARTIE OPÉRATIVE ET DÉTECTION DES DÉFAUTS

CHAPITRE III

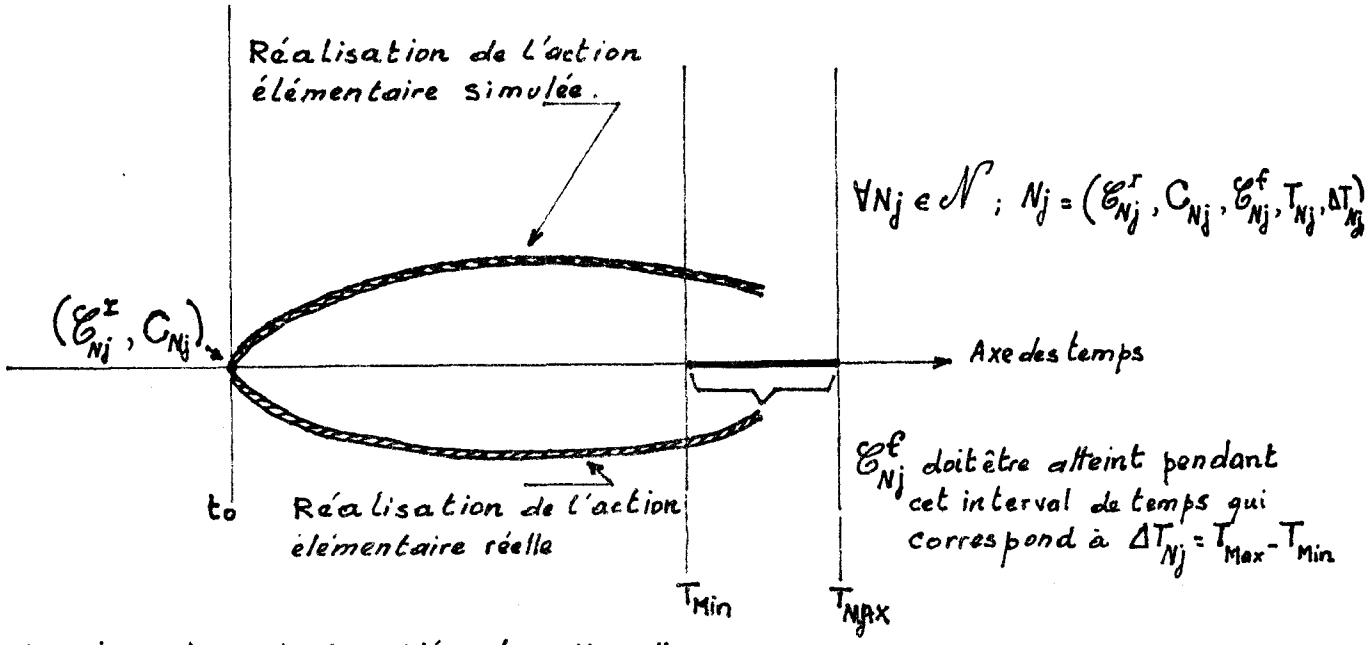
TEST ET DIAGNOSTIC DE LA PARTIE OPÉRATIVE ET DÉTECTION DES DÉFAUTS

TEST ET DIAGNOSTIC DE LA PARTIE OPÉRATIVE ET DÉTECTION DES DÉFAUTS

La méthode de test et de diagnostic que nous allons proposer a pour but la détection et la loca lisation rapide de toute défaillance affectant les éléments de la partie opérative d'un système logique.

Elle met en oeuvre la simulation de la partie opérative. Cette simulation a été rendue possible grâce à la description de la partie opérative, présentée dans le chapitre précédent.

L'exploitation de cette simulation de la partie opérative constitue une redondance active à partir de laquelle le test et le diagnostic seront organisées. En effet :
lors de la réalisation d'une action élémentaire réelle par le système, le dispositif de surveillance lance la réalisation de l'action élémentaire simulée correspondante comme cela est représenté sur la (fig.III.1).



to : Instant ou l'action élémentaire Nj reelle commence à se réaliser.

Figure III.1

La réalisation de chaque action élémentaire simulée par le dispositif de surveillance, s'opère à partir d'une implantation de la description de la partie opérative et de l'acquisition de l'état d'évolution, en cours (image des entrées/sorties) de celle-ci.

III.1 - PROCEDURE DE TEST ET DE DIAGNOSTIC DES ACTIONS ELEMENTAIRES

La partie commande élabore des ordres élémentaires à partir d'un compte-rendu émanant de la partie opérative. Ces ordres élémentaires une fois affectée, détermine une évolution de cette dernière.

Toute action élémentaire réelle qui commence à se réaliser, valide l'action élémentaire simulée correspondante qui se réalisera simultanément par le dispositif de surveillance.

Le test et le diagnostic de cette action consiste alors à vérifier que :

a) l'état contrôlé final de cette action n'est pas atteint avant un temps minimum (T_{MIN}) paramètre de l'action élémentaire simulée correspondante.

b) l'état contrôlé final de cette action élémentaire en cours de réalisation est atteint avant un temps maximum (T_{MAX}) paramètre de l'action élémentaire simulée.

c) l'état contrôlé final n'est pas atteint après le temps maximum.

d) l'état contrôlé final n'est pas atteint sans validation de l'action élémentaire simulée.

dans tous les cas où l'un des points (a,b,c,d) n'est pas vérifié au cours d'un cycle de traitement du dispositif de surveillance .

Un élément physique de l'action élémentaire est alors le siège d'une défaillance. C'est la phase de détection.

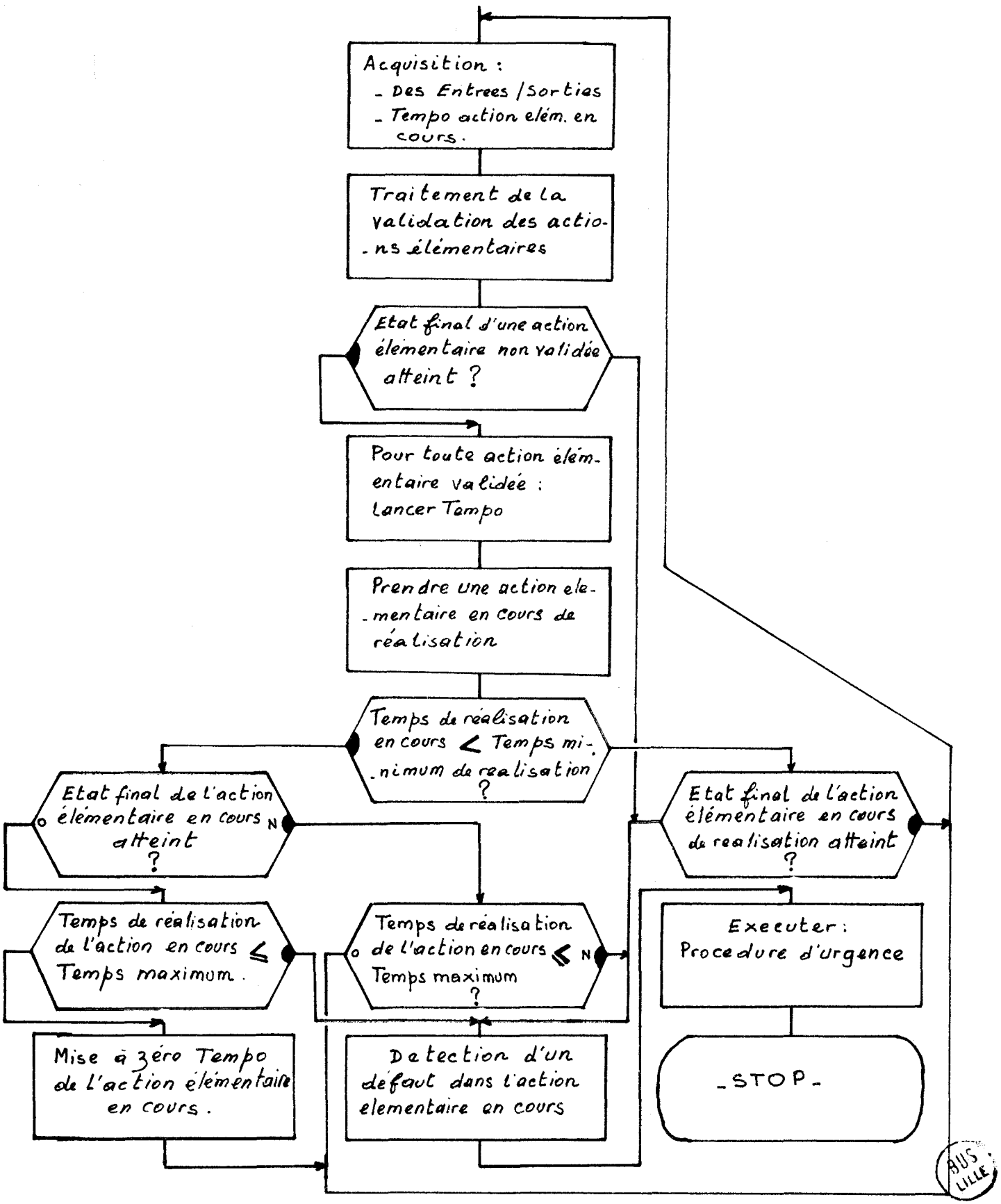

Acquisition :
- Des Entrees / Sorties
- Tempo action elém. en cours.
Traitement de la validation des actions élémentaires
Etat final d'une action élémentaire non validée atteint ?
Pour toute action élémentaire validée : lancer Tempo
Prendre une action elementaire en cours de réalisation
Temps de réalisation en cours < Temps minimum de realisation ?
Etat final de l'action élémentaire en cours atteint ?
O
N
Etat final de l'action élémentaire en cours de realisation atteint ?
Temps de réalisation de l'action en cours ≤ Temps maximum.
Temps de réalisation de l'action en cours ≪ Temps maximum ?
O
N
Executer : Procedure d'urgence
Mise à zéro Tempo de l'action élémentaire en cours.
Detection d'un défaut dans l'action elementaire en cours
-STOP-

Figure III.2

A la suite de cette détection, un message de localisation est émis par le dispositif de surveillance après avoir engagé une procédure d'urgence qui permet de mettre le système dans un état de sécurité.

L'organigramme représenté par la (fig.III.2) décrit les différentes étapes de test et de diagnostic d'une action élémentaire en cours de réalisation.

III.2 - TEST ET DIAGNOSTIC D'UNE ACTION ELEMENTAIRE ET DETECTION DES DEFAUTS

L'objectif à atteindre par la mise en oeuvre de la méthode de test et de diagnostic présentée ci-dessus, est de détecter tout défaut simple affectant soit :

- *l'un des capteurs de la partie opérative*
- *l'un des actionneurs de la partie opérative*
- *l'une des liaisons de la partie commande avec la partie opérative.*

Nous allons étudier les différentes pannes qui peuvent apparaître dans l'un de ces éléments ainsi que le comportement du dispositif de surveillance vis à vis d'elles.

a) Défauts dans les capteurs

Un défaut dans un capteur se manifeste par soit :

- *un collage à zéro*
- *un collage à Un*

La détection de ce défaut aura lieu lorsque ce capteur représente l'état final d'une action élémentaire et que cette action élémentaire se réalise. Suivant l'instant (t_f) où le capteur final tombe en panne, le comportement du dispositif de surveillance est différent. La (fig.III.3) représente et décrit les différentes situations.

DEFAUT DANS LE CAPTEUR	INSTANT D'APPARITION DU DEFAUT : t_f	DETECTION DU DEFAUT	
		Etat du capteur final à zéro	Etat du capteur final à un
COLLAGE A ZERO	$t_f < t_o$ ou $t_o < t_f < T_{MIN}$	ETAT FINAL ATTEINT AVANT T_{MIN}	ETAT FINAL NON ATTEINT APRES T_{MAX}
	$T_{MIN} < t_f < T_{MAX}$		ETAT FINAL ATTEINT APRES T_{MAX}
COLLAGE A UN	$t_f < t_o$ ou $t_o < t_f < T_{MIN}$	ETAT FINAL NON ATTEINT APRES T_{MAX}	ETAT FINAL ATTEINT AVANT T_{MIN}
	$T_{MIN} < t_f < T_{MAX}$	ETAT FINAL ATTEINT APRES T_{MAX}	

Figure III.3 : Détection des défauts dans les capteurs

b) Défauts dans les actionneurs

Un défaut dans un actionneur peut se manifester par un blocage de celui-ci.

Ce défaut est détecté par une non atteinte de l'état final de l'action élémentaire correspondante (qui est en cours de réalisation) après l'écoulement d'une durée de réalisation supérieure à T_{MAX} de cette action.

Tout défaut qui agit sur la vitesse de l'actionneur peut être aussi détecté, ainsi :

- si à la suite d'une défaillance dans l'actionneur, sa vitesse devienne supérieure à sa vitesse normale. La détection se matérialise par l'atteinte de l'état final en un temps inférieur à T_{MIN}.
- pour sa vitesse inférieure, la détection se matérialise par l'atteinte de l'état final en un temps supérieur à T_{MAX}.

c) Défauts dans les liaisons (partie commande, partie opérative)

c.1 - Défauts dans les liaisons capteurs - partie commande

Tout défaut affectant une telle liaison se manifeste par un collage à zéro (ou Un). La détection de ces défauts est semblable à celle de la détection des défauts dans les capteurs (fig.III.3).

c.2 - Défauts dans les liaisons partie commande - actionneurs

La détection des pannes dans les liaisons (partie commande - actionneurs) dépend d'où le dispositif de surveillance puise ses données.

En effet, deux configurations peuvent être envisagées dans l'intégration du dispositif de surveillance dans le système.

Dans la première configuration représentée par la (fig.III.4a), le dispositif de surveillance puise ses données dans l'image mémoire des entrées - sorties de la partie commande. Cette structure a le mérite d'inclure les éléments d'interfaces dans la partie opérative; par conséquent tout défaut les affectants sera détecté par la surveillance.

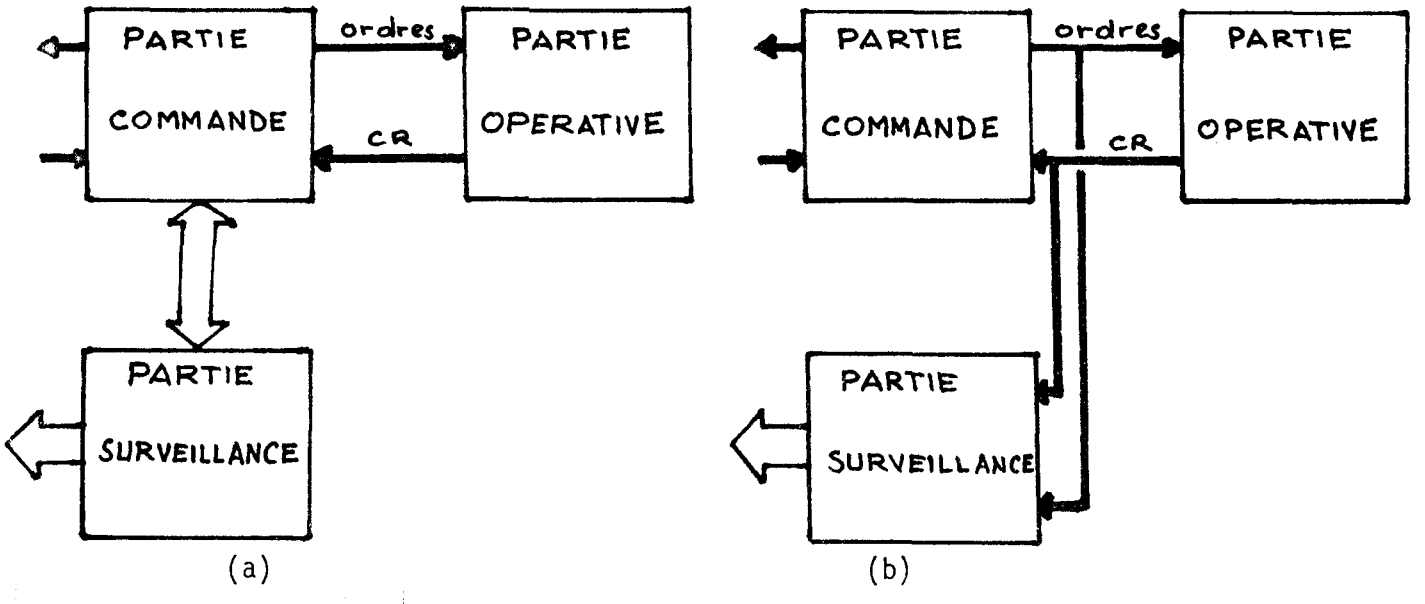


Figure III.4

Dans la deuxième configuration représentée par la (fig.III.4b), la prise des données nécessaires à la surveillance est faite au niveau des liaisons elles-mêmes.

L'étude des défauts dans les liaisons dans les deux structures et leur détection, montre qu'il y a des cas de masquage (fig.III.5).

STRUCTURE	DEFAUT DANS LA LIAISON	INSTANT D'APPARITION DU DEFAUT: t_f	DETECTION
FIG III.4a	collage à zéro	$t_o < t_f < T_{MAX}$	Etat final non atteint
		$t_f < t_o$	Etat final non atteint.
	collage à Un		Masquage
FIG III.4b	collage à zéro	$t_o < t_f < T_{MAX}$	Etat final non atteint.
		$t_f < t_o$	Masquage
	collage à Un		Masquage

Figure III.5 : Détection des défauts dans les liaisons

Etude des cas de masquages

a/ Le masquage apparaît lors d'un défaut de collage à Un d'une liaison. Il se manifeste :

- dans la structure (fig.III.4a), lors de la réalisation de l'action élémentaire défectueuse, par l'atteinte de l'état final sans que l'action élémentaire simulée correspondante, soit validée. De ce fait, le test et le diagnostic de cette action élémentaire, ne peut avoir lieu et le défaut est alors masqué. Pour remédier à ce masquage, un traitement supplémentaire a été envisagé (fig.III.2).

- dans la structure (fig.III.4b) : le masquage du défaut se manifeste par la réalisation et le test de l'action élémentaire défectueuse; bien que la réalisation de celle-ci n'était pas attendue. On peut éliminer le masquage en envisageant un traitement spécifique par le dispositif de surveillance.

b/ Le masquage apparaît lors d'un défaut de collage à zéro d'une liaison dans la structure (fig.III.4b). Il est la conséquence de la non réalisation de l'action élémentaire défectueuse. On ne peut remédier à ce masquage; toutefois on peut remarquer qu'il est sans incidence sur la sécurité du système.

A travers l'étude des défauts et leur détection par la mise en oeuvre de la méthode de test et de diagnostic décrite par la (fig.III.2), on voit que l'intégration du dispositif de surveillance suivant la structure (fig.III.4a) parait plus avantageuse. Elle permet en effet de détecter tous les défauts qui peuvent affecter les éléments de la partie opérative, ses liaisons avec la partie commande et aussi les interfaces.

Le seul inconvénient de cette structure est la complexité des éléments matériels, nécessaires à la réalisation de la connexion (partie commande - dispositif de surveillance).

La structure (fig.III.4b) fait apparaitre des cas de masquage de défauts. La fiabilité de la partie opérative se trouve diminuer du fait de l'adjonction des éléments matériels nécessaires à la réalisation de la connexion.

III.3 - CONDITIONS DE REALISATION DE LA SURVEILLANCE

L'intégration du dispositif de surveillance dans le système logique, doit permettre un meilleur test et diagnostic des actions élémentaires et donc une détection et une localisation efficace et rapide des défauts; cet objectif sera atteint à deux conditions :

- *une synchronisation des traitements relatifs aux dispositifs de surveillance et de commande*
- *le temps moyen de traitement de la surveillance vérifie une contrainte temporelle, définie par rapport aux paramètres temporels des actions élémentaires.*

III.3.1. - Synchronisation des traitements

Le dispositif de surveillance et la partie commande du système logique échangent des informations. Ils peuvent avoir des vitesses de traitement différentes. Pour garantir des échanges conformes à ceux décrits dans le fonctionnement normal, une synchronisation des traitements des deux dispositifs devient nécessaire et indispensable.

Le grafcet (fig.III.6) décrit la procédure de synchronisation des deux traitements effectués par la partie commande et le dispositif de surveillance.

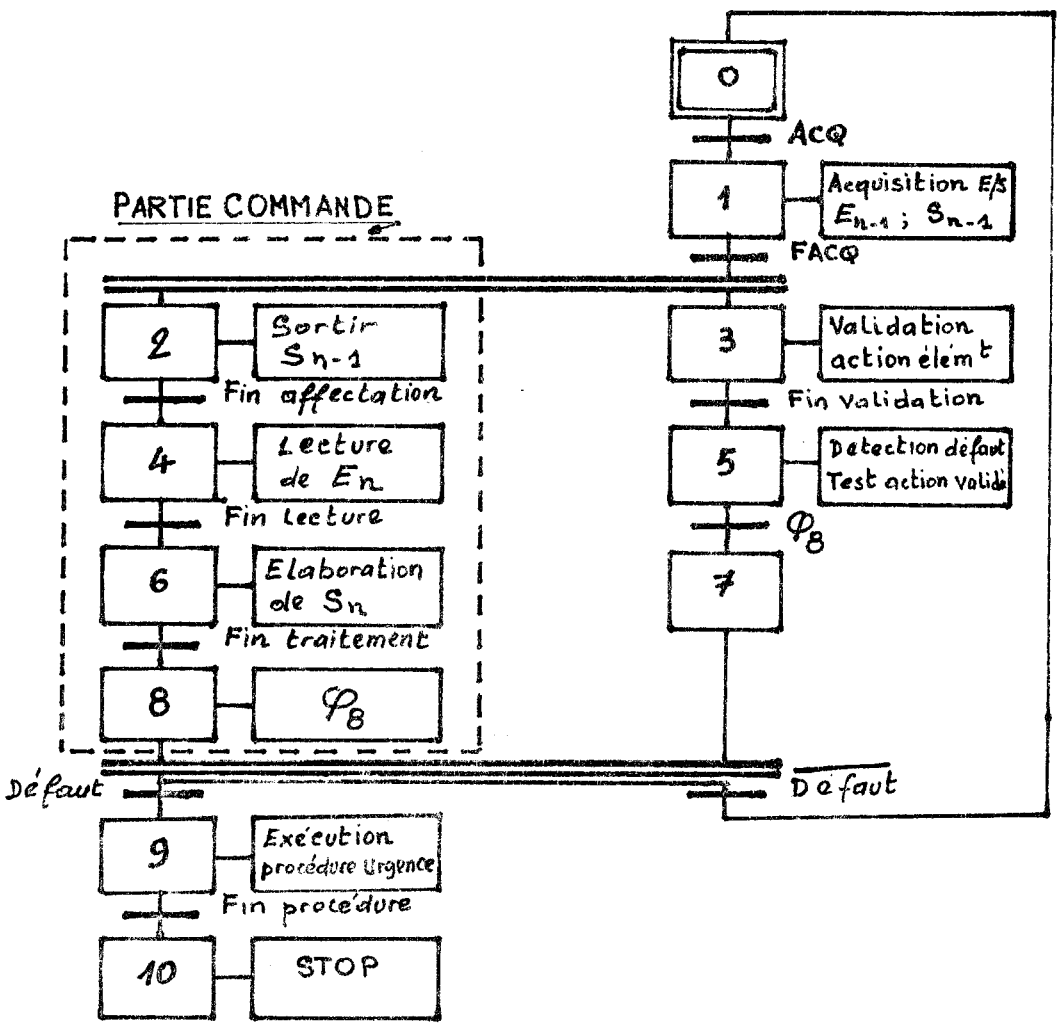


Figure III.6 : Synchronisation des traitements

On remarque dans la (fig.III.6), que l'affectation des sorties (S_{n-1}) élaborées par la partie commande, n'aura lieu que lorsque le dispositif de surveillance a terminé la phase d'acquisition des E/S.

Ainsi aucune évolution significative de la partie opérative ne pourra être masquée. Ceci garantira une meilleure surveillance.

III.3.2. - Contrainte temporelle sur le temps de traitement du dispositif de surveillance

Les performances de la surveillance dépendent largement du temps moyen de traitement du dispositif de surveillance. Plus le temps est court, plus la détection des défauts dans les éléments de la partie opérative est meilleure et efficace.

Nous proposons dans cette étude de déterminer la limite supérieure de ce temps de traitement. Elle constitue une contrainte temporelle qui garantira une sécurité et une disponibilité meilleure du système. La détermination de cette contrainte temporelle est liée à la description de la partie opérative et notamment aux paramètres temporels des actions élémentaires.

Considérons une action élémentaire $N_j \in \mathcal{N}$ ensemble des actions élémentaires.

$$\forall N_j \in \mathcal{N}\ ;\ j = 1,2,\ldots,M\ ;\ M = |\mathcal{N}|\ ;\ N_j = (\mathcal{E}^I_{N_j}, C_{N_j}, \mathcal{E}^f_{N_j}, T_{N_j}, \Delta T_{N_j})$$

on pose $T_{Min} = T_{N_j} - \Delta T_{N_j}$

$T_{Max} = T_{N_j} + \Delta T_{N_j}$

t_o = instant où l'action élémentaire N_j commence à se réaliser

t_f = instant d'apparition du défaut

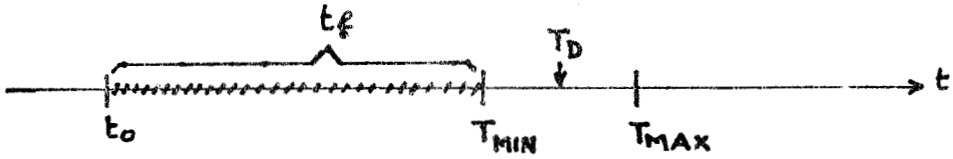
T_D = temps moyen de traitement du dispositif de surveillance

λ = instant d'acquisition $\lambda = 1,2,3 \ldots$

Suivant la valeur de T_D par rapport à T_{Min} et ΔT_{N_j}, des cas de masquage de défauts sont mis en évidence; en effet :

a) supposons que :

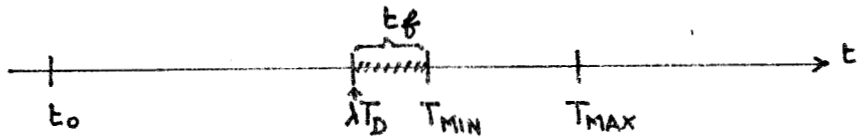
$$T_{Max} > T_D > T_{Min}$$



Dans cette situation on voit que tout défaut qui se traduit par l'atteinte de l'état final ($\mathcal{E}^f_{N_j}$) de N_j avant T_{Min} est masqué.

$$t_o < t_f < T_{Min}$$

b) soit $T_D < T_{Min}$ et $T_D > \Delta T_{N_j}$



Pour $T_{Min} - \lambda T_D < T_D$; $\lambda = 1,2 \ldots$

Il y a aussi masquage de tout défaut qui apparaitre à l'instant t_f et qui se traduit par l'atteinte de l'état final ($\mathcal{E}^f_{N_j}$) de N_j avant T_{Min}.

Un autre cas peut se produire sous cette hypothèse sur T_D :

Dans cette situation, l'état final de l'action élémentaire N_j en cours de test est atteinte normalement, mais du fait du retard de l'acquisition (instant λT_D). Le dispositif de surveillance verra une atteinte de l'état final en un temps supérieur à T_{Max}. Il détecte alors une fausse panne et de ce fait la disponibilité du système se trouve diminuée.

c) Prenons maintenant $T_D < \Delta T_{N_j}$

Dans ce cas les risques de masquage et de détection de fausses pannes sont limités à leurs minimums.

En considérant l'ensemble des actions élémentaires, la sécurité et la disponibilité du système seront garanties grâce à une meilleure surveillance par le dispositif de surveillance auquel on impose une contrainte temporelle donnée par la relation suivante :

$$T_D < \Delta T_{Min}$$

$$\Delta T_{Min} = Min \{ \Delta T_1, \Delta T_2, \ldots, \Delta T_j, \ldots \Delta T_M \} ; M = |\mathcal{N}|$$

III.4 - IMPLANTATION DE LA METHODE DE TEST ET DE DIAGNOSTIC DE LA PARTIE OPERATIVE

L'implantation de la méthode de test et de diagnostic dans le dispositif de surveillance nécessite une organisation des données relatifs à la description de la partie opérative.

Suivant la structure de ses données, la gestion sur la base de l'organigramme de la (fig.III.2) est différente.

Nous étudions ci-dessous deux structures des données de la description de la partie opérative et leur gestion.

III.4.1. - Structures des données

Dans le chapitre précédent, nous avons étudié la description et la décomposition de la partie opérative (fig.II.5) en actions élémentaires. L'implantation de cette description en mémoire du dispositif de surveillance sera faite suivant deux structures de données (fig.III.7) , (fig.III.8).

Dans la structure représentée par la (fig.III.7), on trouve deux tables de données :

a/ Table de validation des actions élémentaires (fig.III.7a). Elle contient des expressions booléennes décrivant les conditions de validations des actions élémentaires.

b/ Table de test des actions élémentaires (fig.III.7b). Elle contient les éléments caractéristiques de la réalisation de chaque action élémentaire (voir II.6.2).

INDICE ACTION ELEMENTAIRE	EXPRESSION BOOLEENNE DE VALIDATION DE L'ACTION

Figure III.7a : table de validation des actions élémentaires

Indice action élémentaire	Adresse état courant de l'état final de l'action	Valeur de l'état final (Ef) de l'action	$T_{MIN} = T - \Delta T$	$T_{MAX} = T - \Delta T$

Figure III.7a : table de test des actions élémentaires

Une deuxième organisation des données peut être envisagée. Elle est représentée par la (fig.III.8). Cette organisation est opérée autour des états contrôlés initiaux : on trouve dans cette structure deux tables :

a/ Tables des états contrôlées (fig.III.8a). Chaque élément de cette table représente l'expression booléenne associée à l'état contrôlé.

b/ Table de test des actions élémentaires (fig.III.8b). Il s'agit dans cette représentation de regrouper toutes les actions élémentaires réalisables à partir d'un état contrôlé initial.

INDICE ETAT CONTROLE	EXPRESSION BOOLEENNE ASSOCIEE A L'ETAT CONTROLE

Figure III.8a : table des états contrôlés

Indice état contrôlé	Ordre élémt de l'action élém.t	Increment à ajouter à l'indice de l'état contrôlé	Temps moyen de réalisation de l'action	Tolérance sur le temps moyen	Indicateur de validation de l'action élémt	Mesure du temps de réalisation de l'action élémt
	Ordre de l'action 1					
	Ordre de l'action 2					

Figure III.8b : table de tests des actions élémentaires

Exemple : Pour illustrer ces deux organisations des données, nous reprenons l'exemple de la (fig.II.5) qui représente la table de données descriptive de la partie opérative du processus décrit par la (fig.II.2) suivant les deux structures de données étudiées, on obtient :

Indice action élémentaire	Adresse état courant de l'état final $\mathcal{E}_f$	valeur état final $\mathcal{E}_f$	Temps minim. um de réalisation	Temps max de réalisation
1	< b1 >	1	$T_1 - \Delta T_1$	$T_1 + \Delta T_1$
2	< b2 >	1	$T_2 - \Delta T_2$	$T_2 + \Delta T_2$
3	< b1 >	1	$T_3 - \Delta T_3$	$T_3 + \Delta T_3$
4	< b3 >	1	$T_4 - \Delta T_4$	$T_4 + \Delta T_4$
5	< b2 >	1	$T_5 - \Delta T_5$	$T_5 + \Delta T_5$
6	< h >	1	$T_6 - \Delta T_6$	$T_6 + \Delta T_6$

Table de tests des actions élémentaires de la structure des données (fig.III.7)

Indice action élémt	Condition de validation
1	h . DGV
2	b1 . DPV
3	b2 . MGV
4	b2 . DPV
5	b3 . MGV
6	b1 . MGV

Table de validation des actions élémentaires

Pour la structure de données (fig.III.8) on obtient l'implantation suivante :

Indice état contrôlé	Expression associée à l'état contrôlé
4	b3
3	b2
2	b1
1	h

Table des états contrôlés

Indice état contrôlé Initial	Ordre élémt de l'action élémentaire	Increment à ajouter à l'indice des état contrôlés	Temps moyen de réalisation des actions élémt	Tolérance sur le temps moyen	Indicateur de validation	compteur temps de réalisation
1	DGV	+ 1	T_1	ΔT_1	0	$\mathcal{C}_1$
2	DPV	+ 1	T_2	ΔT_2	0	$\mathcal{C}_2$
	MGV	- 1	T_6	ΔT_6	0	$\mathcal{C}_6$
3	DPV	+ 1	T_4	ΔT_4	0	$\mathcal{C}_4$
	MGV	- 1	T_3	ΔT_3	0	$\mathcal{C}_3$
4	MGV	- 1	T_5	ΔT_5	0	$\mathcal{C}_5$

Table de tests des actions élémentaires - structure de données (fig.III.8)

III.4.2. - Gestion des structures de données étudiées et surveillance des actions élémentaires

III.4.2.1. - Gestion de la structure de données (fig.III.7)

La gestion de cette structure de données est représentée par l'organigramme (fig.III.9). Elle comprend :

- Une phase d'acquisition qui concerne les entrées-sorties de la partie commande. Ils sont nécessaires à la validation et au test des actions élémentaires. Elle concerne aussi les signaux de synchronisation.

- Une phase validation qui en fonction des états courants des entrées et sorties, détermine les actions élémentaires validées. Toute action élémentaire validée est alors inscrite dans la liste (TP).

- une phase de test et de surveillance qui comprend les traitements suivants :

  . le lancement des actions élémentaires validées et appartenant à la liste TP

  . la surveillance des actions élémentaires lancées (appartenant à TJ). Elle se manifeste par le contrôle du temps de réalisation de l'action élémentaire en cours.

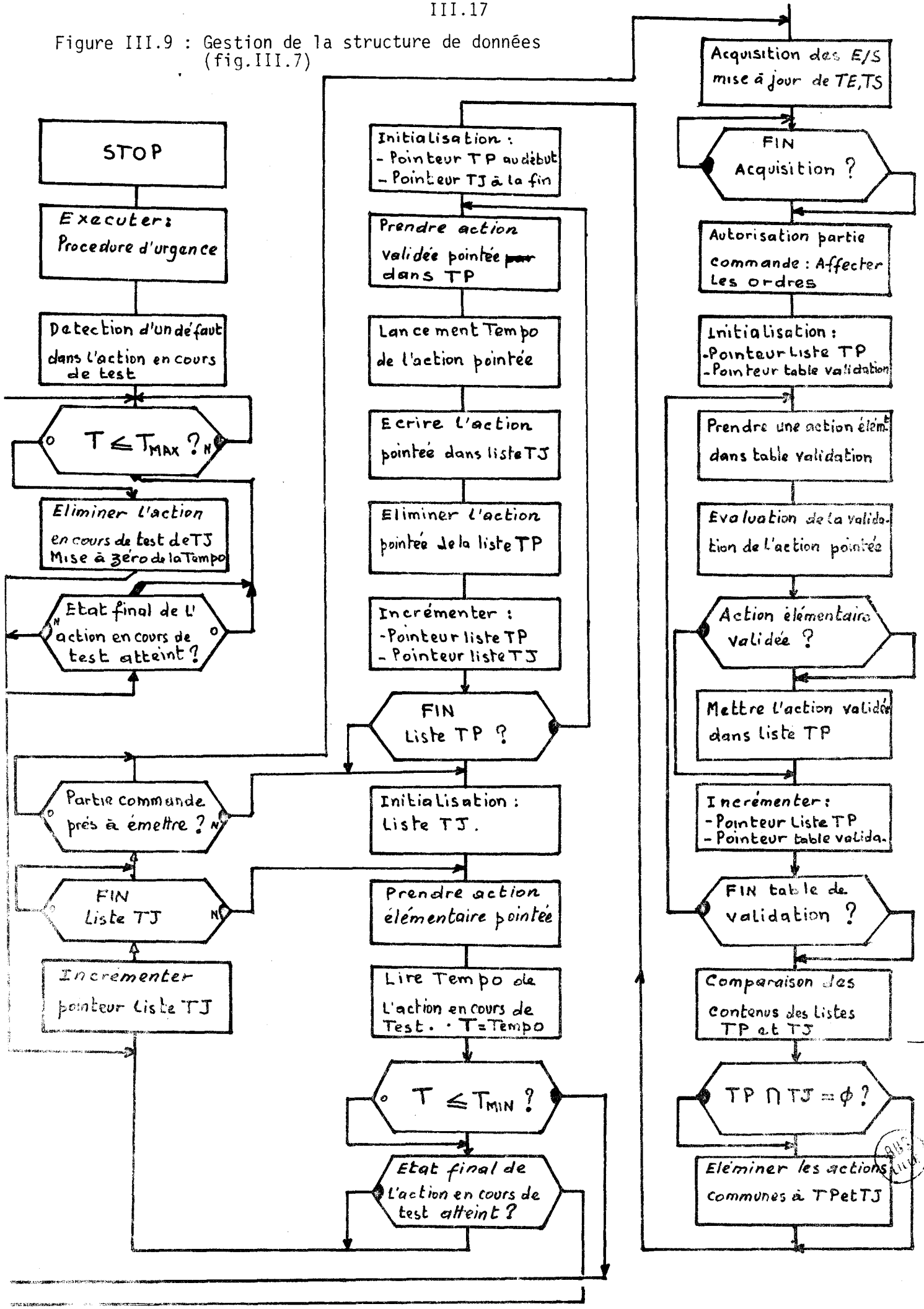
  . toute action élémentaire normalement réalisée est éliminée de la liste TJ.

  . toute action élémentaire détectée en défaillance, déclenche une procédure d'urgence qui met le système dans un état de sécurité. La détection est signalée par une alarme et un message de localisation du défaut est émis par le dispositif de surveillance.

III.4.2.2. - Gestion de la structure de données (fig.III.8)

La gestion de cette structure est décrite par la (fig.III.10). Elle comprend plusieurs phases :

Figure III.9 : Gestion de la structure de données (fig.III.7)



- Phase d'initialisation . Elle se traduit par la mise à jour de la liste des états contrôlés occupés (ECO). Tout état contrôlé occupé par un actionneur est écrit dans la liste ECO.

  La surveillance et le test se limiteront aux seules actions élémentaires réalisables à partir des états contrôlés de cette liste ECO.

- Phase acquisition . Elle concerne l'acquisition de l'état courant des capteurs et des ordres élémentaires nécessaires à la surveillance et celle des signaux de synchronisations.

- Phase validation et surveillance des actions élémentaires

  . La validation consiste à tester l'ordre élémentaire de chaque action élémentaire non valide et à positionner son indicateur de validation (IV=1) si l'ordre élémentaire est actif.

  . La surveillance concerne toutes les actions élémentaires dont l'indicateur de validation est positionné (IV=1) et aussi toutes les actions élémentaires qui ne sont pas valides.

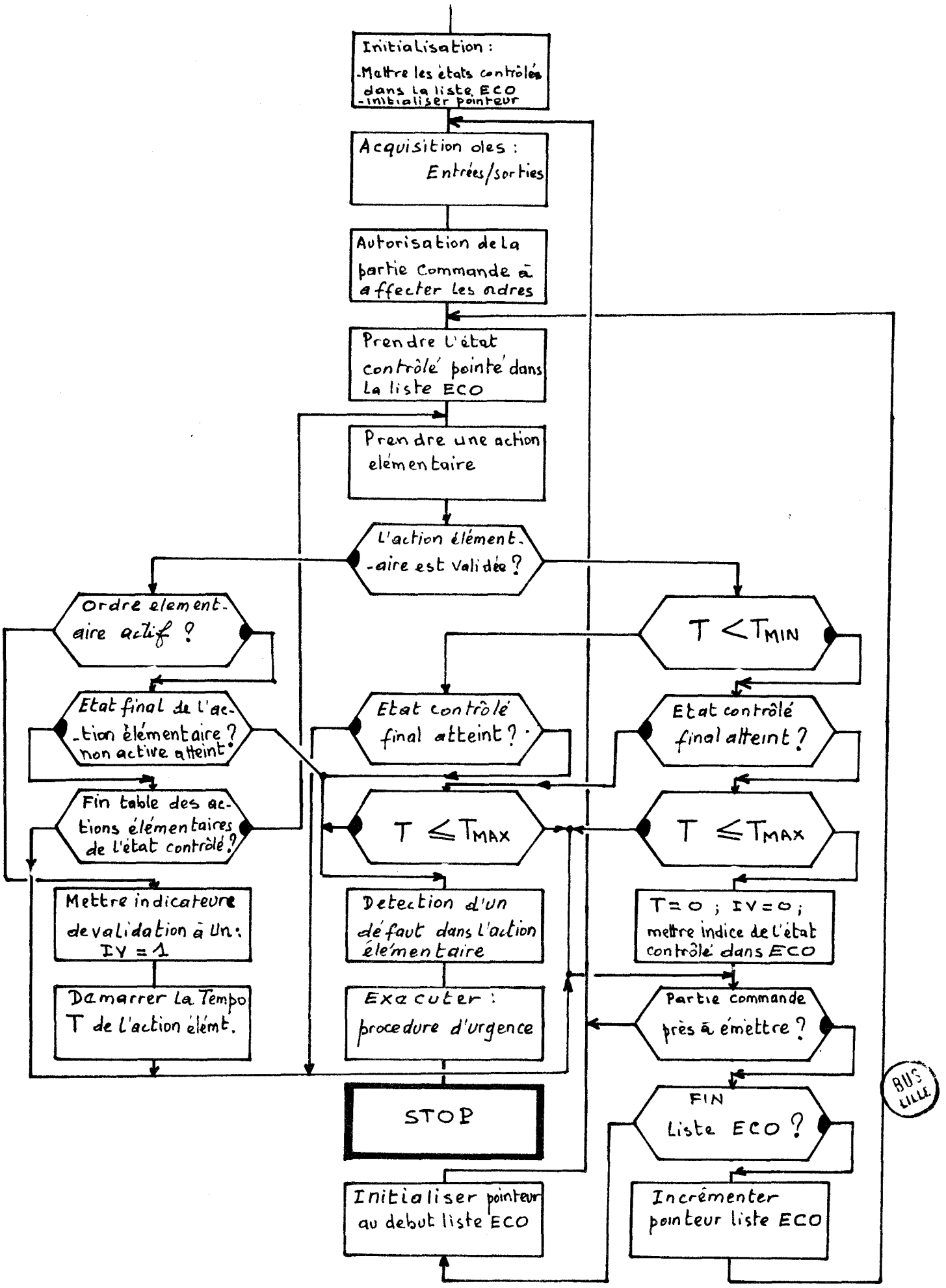
  Dans ces conditions et pour toute action élémentaire réalisée normalement, on écrit l'indice de l'état contrôlé final atteint dans la liste ECO, l'indicateur de validation est remis à zéro ainsi que la temporisation.

- En cas de défaut dans une action élémentaire, une procédure d'urgence est enclenchée par le dispositif de surveillance qui met le système dans un état de sécurité.

III.4.3. - Comparaisons des deux structures de données

L'implantation de la méthode de test et de diagnostic par la mise en oeuvre de la structure de données (fig.III.7) peut se justifier par :

Figure III.10 : Gestion de la structure de données (fig.III.6)



- Une mise en oeuvre facile, car l'obtention des structures de données (table de validation, table de test) à partir de la description de la partie opérative est immédiate.

- Une gestion facile, mais un temps de traitement dépendant surtout du temps de scrutation de la table de validation.
  Les performances de la surveillance sont donc liées à ce temps de traitement et son accroissement peut affecter l'efficacité de la surveillance comme on l'a vu (III.3.1.).

  Cet inconvénient constitue avec la surveillance des actions élémentaires, dont l'état initial est pseudo-contrôlé (dans des situations particulières), une faiblesse dans la mise en oeuvre de cette structure de données.

Tous ces inconvénients sont éliminés lors de la mise en oeuvre de la structure de données (fig.III.8). En effet :

Au niveau du temps de validation, il est limité aux seules actions élémentaires qui sont susceptibles d'être validées.

Au niveau de la surveillance des actions élémentaires pseudo-contrôlées, la mise en oeuvre de cette structure permet de surveiller ces actions élémentaires dans tous les cas.

La structure est performante, malgré une gestion complexe et une mise en oeuvre difficile.

III.5 - VALIDATION DE LA METHODE DE TEST ET DE DIAGNOSTIC

Pour valider la méthode de test et de diagnostic proposée, nous avons réalisé un système (voir annexe A), dans lequel la partie opérative est simulée sur un micro ordinateur (CBM). La partie commande est constituée d'un automate programmable (PB 100) et le dispositif de surveillance est un micro-ordinateur (SILEX).

L'implantation de la méthode de test et de diagnostic dans le dispositif de surveillance est faite suivant la structure de données (fig.III.7).

Après avoir simulé les différents défauts (III.2) dans la partie opérative, l'étude du comportement de la surveillance aux différents défauts a été conforme à celle décrite dans les tableaux (fig.III.3 et 5).

III.6 - CONCLUSION

Dans ce chapitre, nous avons présenté une méthode de test et de diagnostic de la partie opérative.

La base de cette méthode est la simulation de cette dernière.

Cette simulation a été possible grâce à une description et une décomposition de la partie opérative en actions élémentaires. La mise en oeuvre de la méthode de test et de diagnostic proposée a nécessité une structuration des données. Nous avons proposé deux structures de données et leur gestion, et selon lesquelles la méthode peut être implantée sur un dispositif de surveillance.

L'étude des performances et leurs validations montrent que la détection des défauts dans les éléments de la partie opérative est efficace et rapide et cela grâce à une surveillance accrue et en temps réel de l'évolution de la partie opérative, rendue possible par l'intégration du dispositif de surveillance étudié dans le système.

Tous les défauts simples sont détectés, aucun masquage de défauts n'est possible.

La partie commande est supposée sans défaut.

CONCLUSIONS GÉNÉRALES

L'objectif de l'étude proposée est d'amélioré la sécurité d'un système logique.

Dans cette perspective, nous nous sommes intéressés exclusivement à la détection et à la localisation des défauts dans les éléments physiques de la partie opérative d'un système logique, la partie commande étant supposée sans défaut.

Nous rappelons ci-dessous les principaux points du travail présenté dans ce mémoire.

. Nous avons présenté une méthode de description et de simulation de la partie opérative. Elle permet une décomposition de celle-ci en actions élémentaires.

. A partir de cette simulation, une procédure de test et de diagnostic. Elle opère une surveillance accrue et en temps réel des éléments de celle-ci.

. L'étude des performances de cette procédure de test et de diagnostic a montré que la majorité des défauts dans les éléments de la partie opérative sont détectés et localisés. Le risque de masquage des défauts est réduit au minimum.

. Au niveau de la réalisation et de l'implantation de la procédure de test et de diagnostic, nous avons insisté sur deux points importants:

  . La synchronisation des échanges entre le dispositif de surveillance et la partie commande du système.

  . La contrainte temporelle qui fixe la limite supérieure du temps de cycle de traitement du dispositif de surveillance pour garantir un bon pouvoir de détection des défauts.

Il reste à développer :

- La définition d'un langage de description de la partie opérative, associée à la mise en oeuvre d'un logiciel générant les structures de données relatives aux actions élémentaires, doit faciliter l'exploitation en milieu industriel de la procédure de test et de diagnostic proposée.

- La définition d'un dispositif de vérification de concordance au niveau des comptes-rendus et des ordres élémentaires.

  Le rôle de ce dispositif sera de détecter tout défaut dans la partie commande, qui affecte la conformité des comptes-rendus et des ordres élémentaires.

L'association de ce dispositif à la procédure de test et de diagnostic de la partie opérative proposée, peut accroitre encore d'avantage la sécurité du système logique surveillé.

B I B L I O G R A P H I E

AKE.78 S.B AKERS
" Binary decision diagram "
I.E.E.E. Trans. on Computers - Vol.C.27 n°6 - June 1978

ASH.65 R.ASH
" Information Theory "
John Wiley and Sons, 1965

ARM.79 W.W. ARMSTRONG AN K GECSEI
" Adaptation algorithm for biray tree networks "
I.E.E.E. Trans. on S.M.C. - Vol.SMC9, n°5 - Mai 1979

ARG.82 P.ARGENTIERO - R.CHIN - P.BEAUDET
" An automated approach to the design of decision tree classifiers "
I.E.E.E. Trans. on Pattern analysis and machine intelligence
Vol.PAMI-4, n°1 - January 1982

BRE.72 M.A. BREUER
" Design automation of digital systems "
Prence - Hall 1972

BAR.76 A.BARAK - E.SHAMIR
" On parallel evaluation of booleen Expressions "
SIAM J. Computers - Vol.5 n°4 - December 1976

BER.70 C.BERGE
" Graphes et hypergraphes "
Dunod, Paris 1970

CER.76 E.CERNY
" Comment on "Equational Logic" "
I.E.E.E. Trans. on computers - January 1976

DAC.76 E.DACLIN - M.BLANCHARD
" Synthèse des systèmes logiques "
Super Aéro - 1976

DAV.77 M.DAVIO - A.THAYSE
" Sequential evaluation of booleen functions "
MBLE - Res Labo - Brussel Rap.R.321 - Mars 1977

GIV.70 D.D.GIVONE
" Introduction to switching circuit theory "
Mc Graw Hill - 1970

GIU.77 S.GIUASU
" Information theory with applications "
Mc Graw Hill - Condon 1977

HAN.77 M.Z. HANANI
" An optimal evaluation of booleen expression in on line Query systems "
Communication of the ACM - Vol.20 n°5 - Mai 1977

HAR.82 C.R.P. HARTMANN - P.K. VARSHNEY - K.G. MEHROTRA - C.L GERBERICH
" Application of information theory to the construction of efficient decision trees "
I.E.E.E. Trans. on information theory Vol.IT28 n°4 - July 1982

LEE.59 C.Y. LEE
" Representation of Switching systems by binary decisions programs "

LEE.76 S.C LEE
" Digital circuits and logic systems "
Prence - Hall - 1976

LOP.77 R.LOPEZ DE MANDARAS BADIA
" Auto apprentissage d'une partition, application au classement itératif de données multidimentionnelles "
Thèse Doctorat de Spécialité - Univ.Paul Sabatier - Toulouse 1977

MAN.78 D.MANGE - E.SANCHEZ
" Synthèse des fonctions logiques avec des multiplexeurs "
Digital process n°1 Vol.4 - 1978

MAN.79 D.MANGE - E. CERNY - E. SANCHEZ
" Synthesis of minimal binary decision trees "
I.E.E.E. Trans. on Computers - Vol.C.28 - n°7 - July 1979

MEI.73 W.S. MEISEL - D.A MICHALOPOULOS
" A partitioning algorithm with application in pattern classification and the optimization of decision trees "
I.E.E.E. Trans. on Computers Vol.C.22 n°1 - Janvier 1973

MEI.67 W.S MEISEL
" A note on internal state minimization in incomplementely specified sequential networks "
I.E.E.E. Trans. on Computers, 16 - 1967

PARZ.60 E. PARZEN
" Modern probability theory and its application "
John Wiley and Sons - New-York.1960

PAY.77 H.J. PAYNE - N.S. MEISEL
" Algorithm for constructing optimal binary decision trees "
I.E.E.E. Trans. on Computers - Vol.C.26, n°3 - September 1977

PER.76 Y. PERL - Y. BREITBARTH
" Optimal sequential arrangement of evaluation trees for booleen functions "
Information Sciences - 11, 1-12 - 1976

PLA.79 V.M. PLAVSIC - PE DANIELSON
" Sequential evaluation of booleen functions "
I.E.E.E. Trans. on Computers. Vol.C.28 n°12 - December 1979

POK.78 J.L. POKOSKI
" Software analyses for combinatorial logic "
Computer Design / June 1978

POL.71 S.L POLLACK
" Decision tables : theory and practics "
Wiley Interscience - 1971

RAG.81 S.RAGUPATHI - G.VENKATESWARDU - M.P. SINGH
" A new classification of logic fonction of three variables "
Proceeding of the I.E.E.E. - Vol.69 - n°12 - December 1981

SAM.80 M. SAMI MOHAMED
" Conception et réalisation d'un automate programmable par schéma à réseau de Petri "
Thèse de Docteur-Ingénieur - Univ.Paul Sabatier - Toulouse 1980

SHAN.48 C.E SHANNON
" A mathematical theory of communication "
Bell system technical journal, Vol.27 - 1948

SIL.78 M. SILVA . SUAREZ
" Contribution à la synthèse programmable des automatismes logiques "
Thèse de Docteur-Ingénieur - Grenoble - 1978

TOR.82 V.M TORO - CORDOBA
" Contribution à l'analyse structurale des systèmes complexes à l'aide de l'entropie et ses généralisations "
Thèse 3ème Cycle - "Automatique" - Univ. Lille 1 - 1982

TOU.80 S.THELLIEZ - J.M TOULOTTE
" Grafcet et logiques industrielles programmées "
Edition Eyrolles - 1980

TOU.82 S.THELLIEZ - J.M TOULOTTE
" Applications industrielles du grafcet "
Edition Eyrolles - 1982

VAR.82 P.K. VARSHNEY - C.R.P HARTMANN - J.M DEFARIA,JR
" Application of information theory to sequential fault diagnostic "
I.E.E.E. Trans. on Computers - Vol.C.31 - n^2 - February 1982

B I B L I O G R A P H I E

AFC.80 AFCET. Groupe de travail " sécurité et disponibilité des systèmes informatiques " ."Sûreté de fonctionnement des systèmes informatiques "

AUT.82 " Problème de sûreté : le contrôbloc "
Revue " le Nouvel automatisme " Mai 1982

ADE.80 GEMMA
Publication ADEPA

BAC.80 J.P BACONNET - B. GIRARD - S. NATKIN
" Introduction à la sûreté de fonctionnement des systèmes informatiques "

BLA.80 M. BLANCHARD
" Comprendre, maitriser et appliquer le Grafcet "
Editions CEPADUES - 1980

DAC.76 E. DACLIN - M. BLANCHARD
" Synthèse des systèmes logiques "
Edition CEPADUES coll. SUP'AERO, 1976

DIA.74 M. DIAZ
" Conception de systèmes totalement autotestables et à pannes non dangereuses "
Thèse de Docteur ès Sciences
UNIVERSITE PAUL SABATIER - TOULOUSE - 1974

DUB.80 B. DUBUISSON - P. LAVISON
" Surveillance of an Nuclear reactor by USE of a Pattern Recognition méthodologie "
I3E on système man and cybernetic vol.SMC-10, n°10, October 1980

DEF.79 J. DEFRENNE
" Implantation de réseaux de Petri sur automate microprocesseur à haute sûreté de fonctionnement "
Thèse 3ème cycle "Automatique", Université de Lille 1, 1979

DEF.81 J. DEFRENNE - J.M TOULOTTE
" Diagnostic en ligne des capteurs et actionneurs "
Journée AFCET, 23 Mars 1981

DEF.82 J. DEFRENNE - J.M TOULOTTE
" Sur l'accroissement de la sécurité des systèmes à commande logique "
4ème journée scientifique et techniques de la production automatisée
8,9,10 Juin 1983

ITI.82 P. ITISCOHN
" Sécurité et automates programmables "
Revue Electronique Industrielle n°35 / 1.6.82

KUB.78 C. KUBIAK - L. ETESSE - A. PRINGENT - J.L RAINARD
" La sûreté de fonctionnement dans les systèmes complexes "
Note technique NT/RCI/PLC/3Juin 1978

KER.76 KERAN GUEYEN
" Etude sur la disponibilité des systèmes informatiques "
IRISA Publication interne n° 46 Juillet 1976

LAP.75 J.C LAPRIE
" Prévision de la sûreté de fonctionnement et architecture de structures numériques temps réel réparables "
Thèse de Docteur ès Sciences, Univ. Paul Sabatier Toulouse, 1975

LAN.77 C. LANDRAULT
" Prévision de la sûreté de fonctionnement des systèmes numériques réparables "
Thèse Docteur ès Sciences - INP de Toulouse - 1977

LIE.76 LIEVENS
" Sécurité des systèmes "
Editions CEPADUES - 1976

MAR.75 J. MARIN
" Sur le test en ligne des machines séquentielles réalisées à partir des réseaux de Petri "
Thèse de Docteur 3ème Cycle, Nice, Décembre 1975

OSS.80 B.E. OSSFELD AND I. JONSON
" Recovy and diagnostics in the central control of the AXE swiching systems "
IEEE Trans. on computer - vol. C-29, n°6 - June 1980

PB.100 MERLIN GERIN
Notice d'utilisation des automates programmables PB/100

RAP.77 AFCET
Rapport de la commission systèmes logiques présenté par M. BLANCHARD
" Normalisation du cahier des charges d'un automatisme logique "
Revue Automatisme tome XXIII n°3-4 Mars/Avril 1978

SUA.78 M. SILVA SUAREZ
" Contribution à la synthèse programmée des automates logiques "
Thèse INPG - Grenoble 1978

SAV.80 J. SAVIR
" Detection of single intermittent faults in sequentials circuits "
IEEE Trans. on computers vol. C-29 n°7 July 1980

SILEX Notice d'utilisation du micro ordinateur "Silex"

TOU.81 S. THELLIEZ - J.M TOULOTTE
" Grafcet et logique industrielle programmée "
Editions Eyrolles - 1981

TOU.78 J.M TOULOTTE
" Réseaux de Petri et automates programmables "
Revue automatisme, tome XXIII n_6-7 - Juillet/Août 1978

THI.78 S. THELLIEZ
" Pratique séquentielle et réseaux de Petri "
Editions Eyrolles, coll.EEA.1978

TOH.71 Y. TOHMA - Y. OHYAMA - R. SAKAI
" Réalisation of fail safe sequential machines by using a K - out - of - n code "
IEEE Trans. on computers vol. C-20 n°11 - Novembre 1971

VAL.76 R. VALETTE
" Sur la description, l'analyse et la validation des systèmes de commande parallèles "
Thèse doctorat ès Sciences, Univ. Paul Sabatier, Toulouse, Novembre 1976

BOS.78 J.C BOSSY - P. BRARD - P. FAUGERE - C. MERLAUD
" Le Grafcet sa pratique et ses applications "
Educalive

ANNEXE A

RÉALISATION D'UN SYSTÈME LOGIQUE

Pour tester et valider la méthode de test et diagnostic exposée dans le chapitre III, nous avons étudié le comportement de celle-ci en présence de différentes pannes simulées sur les différents éléments de la partie opérative du système logique réalisé (fig.A.1).

Dans cette annexe, nous allons décrire les différents composants de cette réalisation en évoquant les problèmes posés par leur interconnexion et les solutions proposées pour les résoudre.

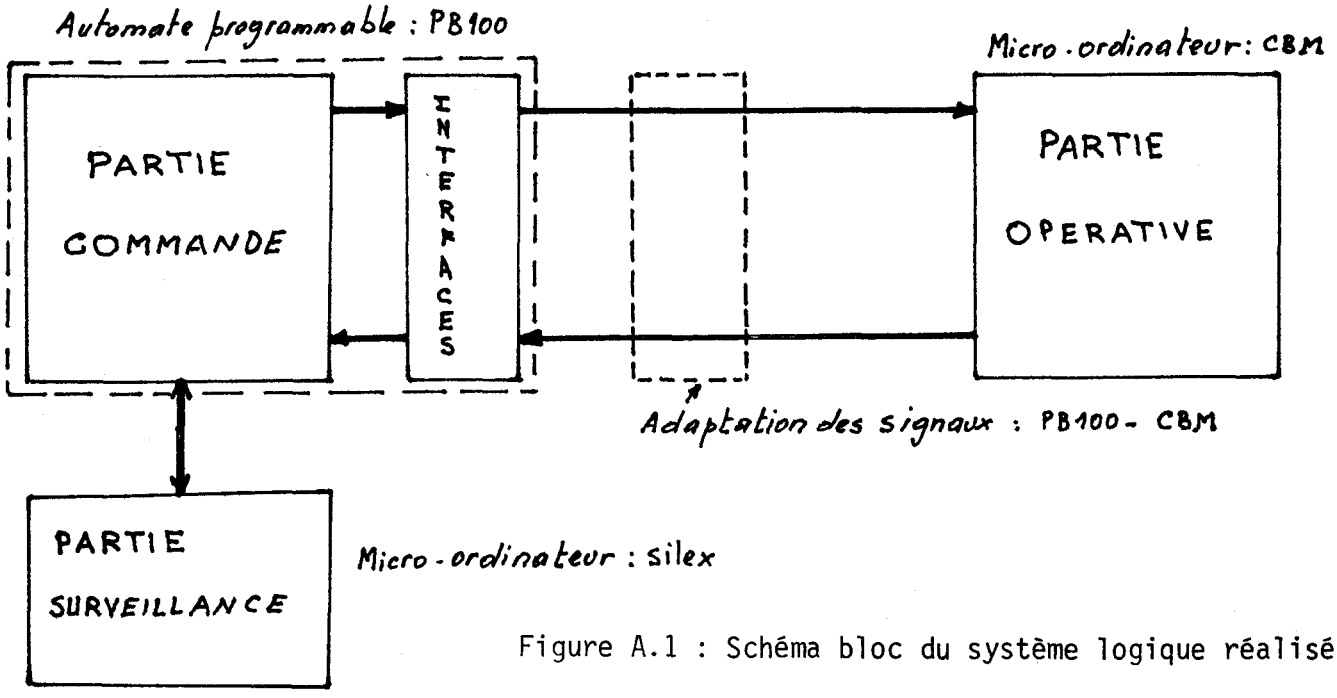


Figure A.1 : Schéma bloc du système logique réalisé

Dans le système logique décrit par la figure A.1, la partie opérative est matérialisée par un micro ordinateur CBM sur lequel on a simulé un processus. La partie commande est assurée par l'automate programmable PB100 de chez SIEMENS et la partie surveillance est remplie par un micro-ordinateur : silex.

A.1 - Description du processus

A défaut d'un processus réel, nous avons choisi le processus représenté par la figure A2.a. Le grafcet de la figure A2.b décrit son fonctionnement.

Nous avons simulé ce processus sur le micro-ordinateur (CBM). Le programme de simulation est écrit en langage BASIC. L'accès aux actionneurs et aux capteurs du processus est fait respectivement par les ports J2 et I3E du CBM.

La commande de ce processus simulé est assurée par l'automate programmable PB100. La connexion entre les deux machines a nécessité la réalisation d'un interface (fig.A2.c) pour résoudre le problème d'incompatibilité des signaux ($5^{V} \longrightarrow 48^{V}$) délivrés respectivement par le CBM et le PB100.

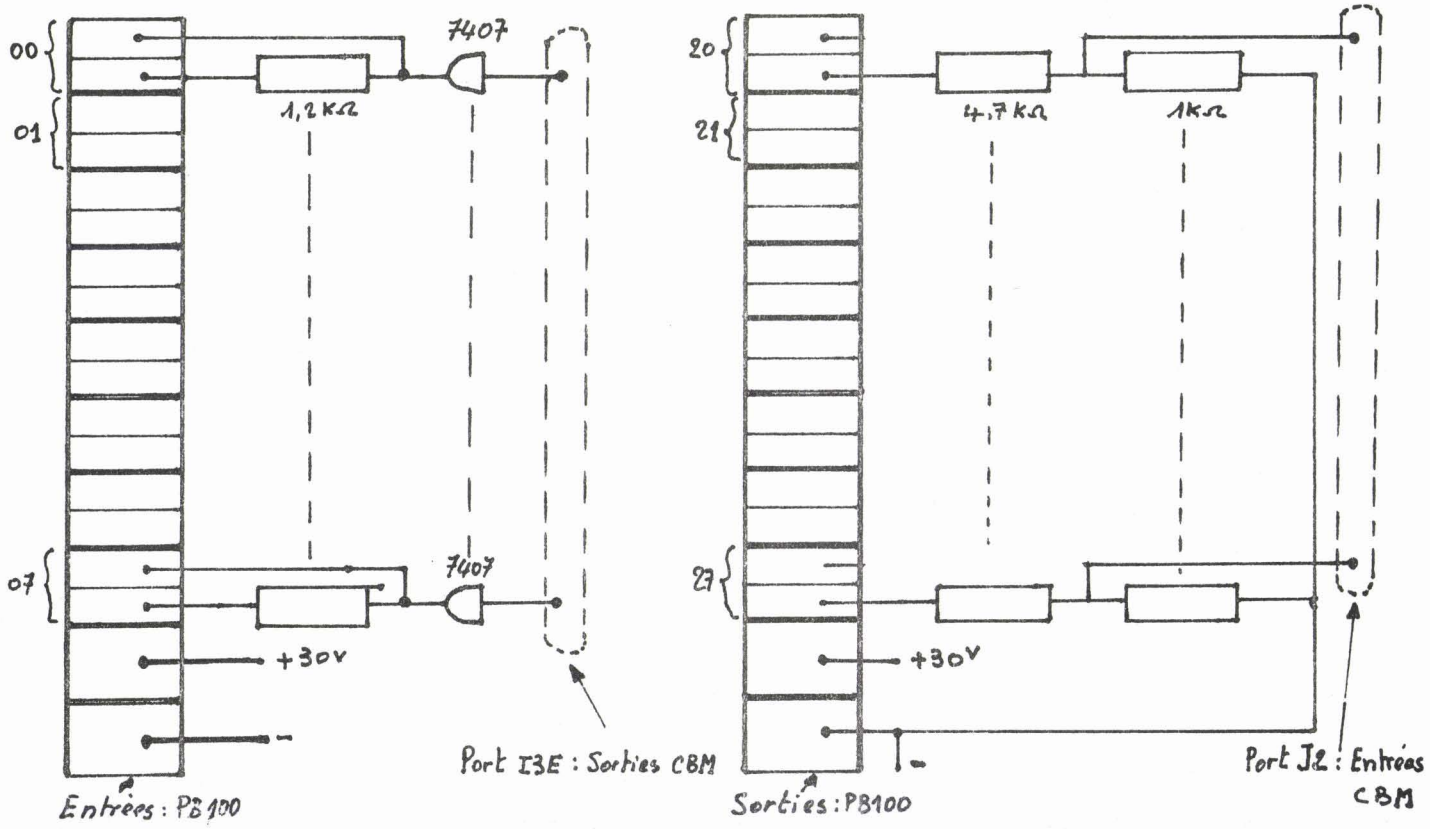


Figure A2.c : Adaptation signaux PB100 ⟶ CBM

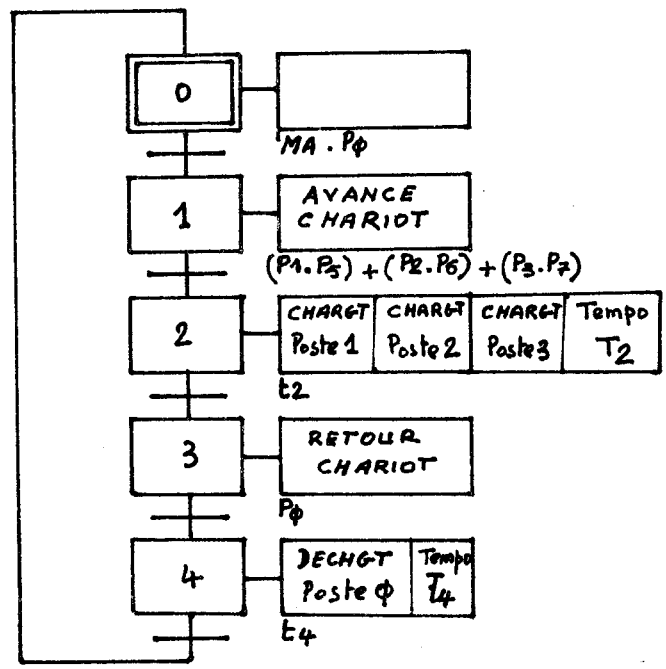


Fig. A.1.b : Grafcet de fonctionnement

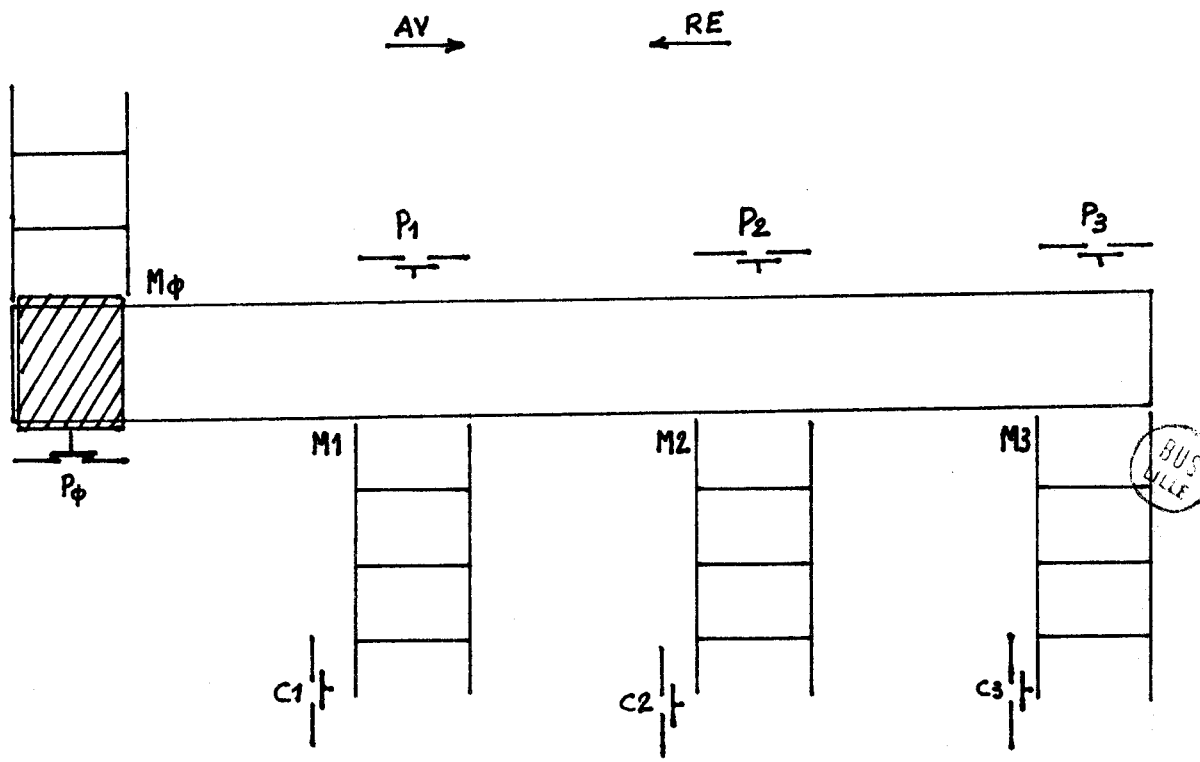


Fig. A.1.D : Processus

A.2 - Description de la partie commande

La commande du processus représenté par la figure A2.a et simulé par le CBM est assurée par l'automate programmable PB100.

Son rôle dans le système est de faire l'acquisition des comptes-rendus issus de la partie opérative (état du port I3E-CBM) et de fournir les ordres appropriés aux actionneurs de celle-ci (état du port J2 du CBM) après un traitement préalable.

Pour remplir cette fonction le grafcet de la fig. A2.b a été implanté sur le PB100

A.3 - Description de la partie surveillance

Le rôle de la partie surveillance dans le système représenté par la figure A1 est de suivre en temps réel l'évolution du processus décri ci-dessus. Elle doit assurer une surveillance accrue de la partie opérative suivant la méthode de test de diagnostic développée dans le chapitre III, pour détecter toute défaillance dans les éléments de celle-ci (actionneurs, capteurs, liaisons).

Cette surveillance est assurée par un micro-ordinateur (silex) sur lequel la méthode de test et de diagnostic a été implanté suivant l'algorithme décrit par l'organigramme de la figure III.9. (chapitre III)

Pour remplir cette fonction, la partie surveillance doit dialoguer avec la partie commande (automate PB100). Ce dialogue concerne la synchronisation des traitements et l'acquisition des données nécessaires à la surveillance. Ce dialogue est assuré par un sous-programme écrit en assembleur et implanté sur le (silex).

Pour rendre ce dialogue possible un interface entre le silex et le PB100 a été étudié. Il permet de transformer les impulsions 0 - 12^{V} du silex en une boucle de courant de 20 mA (PB100) ou vis versa. Le schéma de la figure A3 représente cet interface.

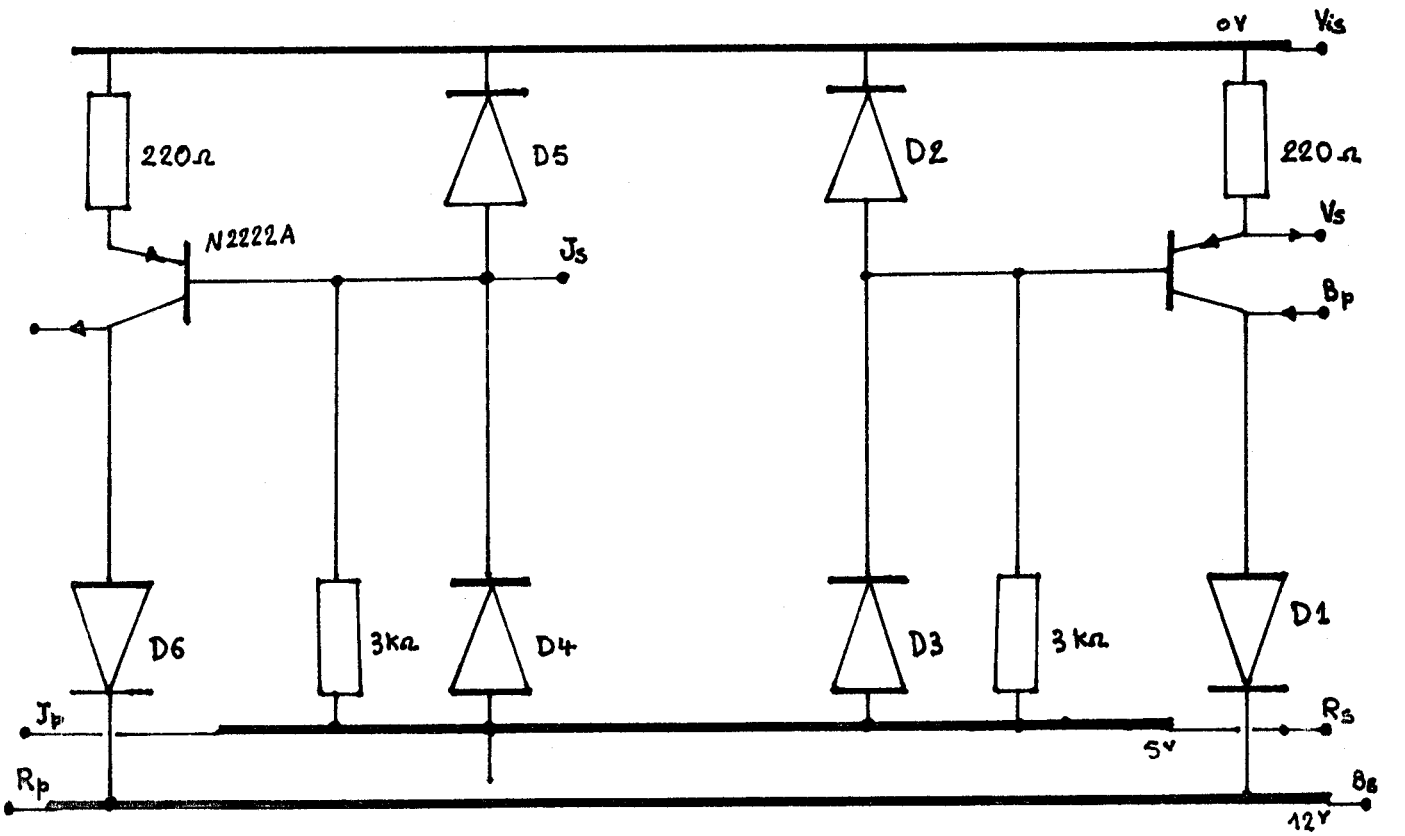


Fig. A.3 : Adaptation des signaux PB100 Silex.

ANNEXE B

ESTIMATION DES PARAMÈTRES TEMPORELS

Dans cette annexe, nous présentons un algorithme qui permet l'estimation et l'ajustement des paramètres temporels d'une action élémentaire lors de la mise en oeuvre de la méthode d'acquisition " En ligne ".

1) Ajustement du temps de réalisation d'une action élémentaire

On considère une action élémentaire $N_i \in \mathcal{N}$ dont les paramètres temporels sont T_{N_i} et ΔT_{N_i}

On a :

$T_{N_i} = \overline{\tau_{ij}} = \dfrac{\sum_{k=1}^{j} \tau_{ik}}{j}$ ou τ_{ij} la $j^{ème}$ mesure du temps de réalisation de N_i

$\overline{\tau_{ij}}$: la valeur moyenne

On pose $P_j = \dfrac{1}{j}$ (B.1)

$$Q_j = \sum_{k=1}^{j} \tau_{ik} \quad (B.2)$$

et $T_{N_i} = \overline{\tau_{ij}} = P_j \cdot Q_j$

Considérons maintenant la mesure $j+1^{ème}$ du temps de réalisation de $N_i \in \mathcal{N}$ soit $\tau_{i,j+1}$

On a :

$$P_{j+1} = \frac{1}{j+1} \qquad P_{j+1} = \frac{P_j}{P_j+1}$$

de même $Q_{k+1} = Q_k + \tau_{i,j+1}$

$$\overline{\tau}_{i,j+1} = P_{j+1} \cdot Q_{j+1} = \frac{P_j \cdot Q_j}{1 + P_j} + \tau_{i,j+1} \cdot \frac{P_j}{1 + P_j}$$

$$\overline{\tau}_{i,j+1} = \frac{1}{1 + P_j} \left| \overline{\tau}_{ij} + P_j \tau_{i,j+1} \right|$$

d'où le réajustement de T_{N_i} par l'arrivée de la $j+1^{ème}$ mesure $\tau_{i,j+1}$ du temps de réalisation soit :

$$T_{N_i} = \overline{\tau}_{i,j+1} = \frac{1}{1 + P_j} \left| \overline{\tau}_{ij} + P_j \cdot \tau_{i,j+1} \right| \qquad (B.3)$$

2) Réajustement de ΔT_{N_i} : nous avons défini

$$\Delta T_{N_i} = \alpha \cdot \sigma_{ij} \quad ; \sigma_{ij} \text{ écart type de } T_{N_i}$$

σ_{ij} avant l'arrivée de la $j+1^{ème}$ mesure $\tau_{i,j+1}$ est évalué par :

$$\sigma^2_{ij} = \frac{1}{j} \sum_{k=1}^{j} \tau^2_{ik} - \overline{\tau}^2_{ij}$$

à la j + 1 mesure $\sigma_{i,j+1}$ tel que

$$\sigma^2_{i,j+1} = \frac{1}{j+1} \sum_{k=1}^{j+1} \tau^2_{ik} - \overline{\tau}^2_{i,j+1}$$

on pose

$$\beta_j = \frac{1}{j} \sum_{k=1}^{k} \tau^2_{ik} \qquad \beta_{j+1} = \frac{1}{j+1} \cdot \sum_{k=1}^{j+1} \tau^2_{ik}$$

en utilisant les relations (B.1) et (B.2)

$$\beta_{j+1} = \frac{P_j}{1 + P_j} \left| Q_j + \tau^2_{i,j+1} \right|$$

$$\beta_{j+1} = \frac{\beta_j}{1 + P_j} + \frac{P_j}{1 + P_j} \tau^2_{i,j+1}$$

$$\sigma^2_{i,j+1} = \frac{1}{1 + P_j} \beta_j + \frac{P_j}{1 + P_j} \tau^2_{i,j+1} - \bar{\tau}^2_{i,j+1}$$

$$\sigma^2_{i\,j} = \beta_j - \bar{\tau}^2_{ij}$$

$$\sigma^2_{i,j+1} = \frac{1}{1 + P_j} \beta_j - \frac{1}{1 + P_j} \tau^2_{ij} + \frac{1}{1 + P_j} \tau^2_{ij} + \frac{P_j}{1 + P_j} \tau^2_{i,j+1} - \bar{\tau}^2_{i,j+1}$$

$$\sigma^2_{i,j+1} = \frac{1}{1 + P_j} \sigma^2_{ij} + \frac{P_j}{1 + P_j} \tau^2_{i,j+1} + \frac{\bar{\tau}^2_{ij}}{1 + P_j} - \bar{\tau}^2_{i,j+1} \quad (B.4)$$

En définitive avec chaque réalisation d'une action élémentaire ses paramètres sont réajustés par la mise en oeuvre des expressions (B.3) et (B.4).