REDUCTION DES MACHINES SEQUENTIELLES NON DETERMINISTES

J. ZAHND

En partant de la notion classique de machine séquentielle incomplètement spécifiée, on introduit une généralisation de ce modèle, sous la forme de tables d'états qui associent à chaque état présent et à chaque signal d'entrée un ensemble d'etats futurs et un ensemble de signaux de sortie. On définit ensuite une fonction de réponse qui caractérise le comportement entrée-sortie d'une telle machine, ce qui permet de poser en termes exacts le problème de sa réduction. On passe en revue quelques applications possibles de ce modèle: synthèse des systèmes digitaux, réseaux de Petri, microprogrammes non déterministes. Ensuite on aborde le problème de la réduction de ces machines. On généralise la notion classique de recouvrement compatible et l'on montre qu'elle permet de réduire une machine non déterministe. On termine par l'énoncé des principaux problèmes ouverts qui peuvent faire l'objet d'une suite de ce travail.

1. INTRODUCTION

La théorie des machines séquentielles s'est développée en relation avec les méthodes de synthèse des systèmes logiques câblés, notamment l'usage de tables d'états pour représenter le fonctionnement des systèmes séquentiels. Malgré l'usage de plus en plus répandu de nouveaux modèles tels que réseaux de Petri, Grafcet, les notions fondamentales de cette théorie conservent une valeur certaine. En particulier ces notions s'appliquent bien à la représentation et la transformation des microprogrammes.

Néanmoins, il y a lieu de se demander à l'heure actuelle s'il ne convient pas de procéder à une extension de cette théorie dans le sens d'une généralisation de la notion de table d'états de façon à inclure comme cas particuliers les nouveaux modèles mentionnés ci-dessus. C'est dans cette ligne de recherche que se situe la présente communication. Son but est d'introduire le modèle de la machine séquentielle non deterministe, et d'aborder le problème de sa réduction. On ne présentera qu'une solution partielle de ce problème, la solution générale n'étant pas encore connue. Notre but est avant tout de présenter un thème de recherche qui paraît mériter une certaine attention.

2. LE MODELE DE LA MACHINE SEQUENTIELLE NON DETERMINISTE

2.1. LA MACHINE SÉQUENTIELLE CLASSIQUE

Nous partirons de la notion classique de machine séquentielle, au sens de machine de Mealy incomplètement spécifiée, la machine de Moore étant considérée comme un cas particulier. Nous supposons connue la théorie de la réduction de ces machines, telle qu'elle est exposée par exemple par Booth (ref. 1). Nous ne rappelons ici que les premières définitions.

On sait qu'une machine séquentielle est définie par une table d'états ou de façon equivalente par un graphe d'états. Un exemple simple est donné par la table 1 et la figure 2.

- J. Zahnd, Ecole polytechnique féderale - 16, Chemin de Bellerive - 1007 Lausanne, Suisse

- Article rebut el Gener de 1982.
- Aquest treball va ser presentat a les 1es. Jornades del Diseny Llogic a Barcelona del 15 al 17 de Juliol de 1981.

	a	b
1	- ; -	2 ; -
2	4 ; -	3 ; -
3	1 ; 0	- ; -
4	1 ; 1	- ; -

Table 1

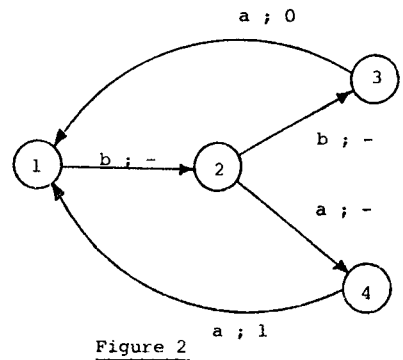

Figure 2

Mathématiquement, une machine R est caractérisée par les données suivantes:

L'ensemble X des signaux d'entrée, appelé -- aussi *l'alphabet d'entrée* de la machine. Dans l'exemple ci-dessus, X = {a,b}.

L'ensemble Y des *états* (ou états internes), à savoir Y = {1, 2, 3, 4} dans l'exemple.

L'ensemble Z des signaux de sortie, appelé -- aussi *l'alphabet de sortie*. Dans notre exem-- ple, Z = {0, 1} .

La fonction de transition δ , qui associe un état futur δ (y, x) à certains couples (y, x) où y est un état et x un signal d'entrée. Un tel couple est appelé un *état total*.

La *fonction de sortie* λ, qui associe un signal de cortie λ(y, x) à certains états totaux (y, x).

Dans la table 1, on a par exemple δ(2,a) = 4 λ(3, a) = 0, tandis que δ(1, a) et λ(2, a) ne sont pas définis, ce qui est noté par le ti-- ret "-".

Pour designer une machine R avec ses alpha--- bets X, Y, Z et ses fonctions de transition - et de sortie, on utilisera la notation ------ R(X, Y, Z, δ, λ). Pour la désigner avec ses alphabets seulement: R(X, Y, Z).

2.2. LE MODÈLE GÉNÉRALISÉ

Afin de généraliser le modèle classique rappelé ci-dessus, nous allons d'abord le présen-- ter d'une façon différente. Pour cela, nous - analysons la signification des "don't care -- conditions" représentées par les tirets de la table d'états.

On utilise une table d'états incomplètement spécifiée pour représenter un système logique qui se trouve encore à l'état de projet. La table représente en quelque sorte le cahier des charges d'un système qu'on se propose de construire. Nous supposons connu ici - le procédé selon lequel on réalise une table d'états incomplètement spécifiée par un système logique séquentiel synchronisé (ref. 2, chap. 6). Ce système logique sera représenté par une table d'états complètement spécifiée. Les "don"t care" de la table d'états initiale signifient simplement que le cahier des - charges donné admet plusieurs réalisations.

Considérons par exemple la machine R de la - table 1. La table d'états complètement spécifiée d'une réalisation quelconque de cette - machine peut être construite immédiatement: il suffit de remplacer chaque tiret de la table par un élément arbitraire y de Y (resp. un élément z de Z). La table obtenue définit une nouvelle machine R' complètement spéci-- fiée, que nous appellerons elle-même une réalisation de R. Ceci nous suggère d'interpréter une machine incomplètement spécifiée comme une représentation de l'*ensemble de ses - réalisations possibles*. Selon cette interpretation, on peut décrire la machine R de la - table 1 par la table 3, qui est construite - en remplaçant chaque état futur non spécifié par l'ensemble des états Y = {1, 2, 3, 4} ,- et chaque sortie non spécifiée par l'ensem-- ble des signaux de sortie Z = {0, 1} .

	a	b
1	1,2,3,4 ; 0,1	2 ; 0,1
2	4 ; 0,1	3 ; 0,1
3	1 ; 0	1,2,3,4 ; 0,1
4	1 ; 1	1,2,3,4 ; 0,1

Table 3

L'intérêt de cette nouvelle représentation - est qu'elle suggère immédiatement une généralisation de la notion de machine séquentielle classique. On voit en effet que la table 3 associe à chaque état et à chaque signal - un *ensemble* d'etats futurs et un *ensemble* de signaux de sortie. Ces ensembles comportent toujours ici: ou bien tous les élements de Y (resp. Z), ou bien un seul élément. Mais on peut imaginer que certains cahiers des charges se traduiront par des ensembles quelconques d'états futurs et de signaux de sortie. Ces ensembles ne seront cependant jamais vides, puisque quel que soit le système avec - lequel on réalisera un tel cahier des charges, ce système possédera un état futur et - un signal de sortie pour chaque état présent et chaque signal d'entrée. Ces réflexions -- nous conduisent à généraliser comme suit la notion de machine séquentielle.

DEFINITION 1. Une machine séquentielle ----- R(X, Y, Z) est caractérisée par:

Une fonction de transition qui associe à chaque état total (y,x) un sous-ensemble non vide de Y. Ce sous-ensemble sera noté $(y.x)_R$ ou simplement y.x.

Une fonction de sortie qui associe à chaque état total (y,x) un sous-ensemble non vide de Z. Ce sous-ensemble sera noté --- $(y/x)_R$ ou simplement y/x.

Par exemple pour la machine de la table 3, on a

1.a = {1,2,3,4} ; 2.a = {4}
1/a = {0,1} ; 3/a = {0}

Sauf indication contraire, le terme de machine sera pris dorénavant au sens de la définition 1 ci-dessus. Une telle machine est *non - déterministe* lorsqu'elle possède au moins un état total (y,x) pour lequel l'un des ensembles y.x, y/x possède plus d'un élément. Une machine incomplètement spécifiée au sens --- classique (table 1) devient ainsi une machine non déterministe (table 3), mais il s'agitlà d'une forme particulière de non déterminisme.

Il est légitime de se demander ici si le modèle général défini cidessus présente un intéret pratique. Nous traiterons cette question brièvement dans la section 3. On verra que ce modèle se présente naturellement dans diverses situations, qui suffisent à notre sens à justifier son étude théorique.

2.3. FONCTIONS DE RÉPONSE

2.3.1.

Pour une machine R(X,Y,Z), une sequence d'entrée de longueur n (n entier $\geqslant$ 1), notée --- x(1,n), est une suite finie x(1,n) = ------ = x(1) x (2)...x (n) de n signaux d'entrée - x(i). On définit de même une séquence d'etats y(1,n) et une séquence de sortie z(1,n).

Lorsqu'on veut poser de façon tout-à-fait générale le problème de la réduction d'une machine, il faut introduire une description de son comportement "entrée-sortie". On entend par là une description qui ne se réfère ---- qu'aux alphabets X et Z. Nous admettons en - effet que l'environnement de la machine ne - voit que son entrée et sa sortie, et qu'il - peut tout au plus la mettre dans un état initial. Ceci est représenté schématiquement -- dans la figure 4.

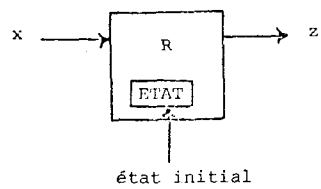

Figure 4

la machine contient un registre d'état, qui n'est pas observé par l'environnement, mais auquel on peut donner au besoin une valeur - initiale. Réduire la machine R(X,Y,Z) signifie alors: remplacer R par une autre machine R'(X,U,Z) qui aura les mêmes alphabets d'entrée et de sortie que R, qui aura moins d'états que R, et dont le comportement entrée-- sortie, comparé à celui de R, sera acceptable par l'environnement. C'est cette dernière condition qu'il importe de préciser, et qui nécessite une définition exacte de la notion de comportement entrée-sortie.

Cette définition peut être donnée de plusieurs manières, et le choix de la définition influence la notion de réduction. Pour bien montrer ce point, nous allons brièvement rappeler deux définitions différentes rencontrées dans la théorie classique des machines incomplètement spécifiées.

2.3.2. Fonction de réponse de Ginsburg

Considérons une machine incomplètement spécifiée R(X,Y,Z,δ,λ) au sens classique, un état initial p et une séquence d'entrée x(1,n). Selon Ginsburg (Réf. 3, l'un des premiers ouvrages consacrés à la théorie des machines séquentielles), la réponse de la machine n'est définie que si *chacun des termes suivants est défini* par la table d'états:

$$\left.\begin{array}{l} y(1) = p \\ \qquad z(1) = \lambda(y(1), x(1)) \\ y(2) = \delta(y(1), x(1)) \\ \qquad z(2) = \lambda(y(2), x(2)) \\ \ldots \\ y(n) = \delta(y(n-1), x(n-1)) \\ \qquad z(n) = \lambda(y(n), x(n)) \end{array}\right\} \quad (1)$$

Si chacun de ces termes est défini, la *réponse de la machine pour l'état initial* p *et la séquence d'entrée* x(1,n) est la *séquence* z(1,n) définie par (1). Cette réponse est une fonction de p et de x(1,n) que nous notons ρ(p, x(1,n)) et appelons *fonction de réponse de Ginsburg*. Si l'un quelconque des termes z(1), y(2),..., y(n), z(n) n'est pas défini ("don't care condition"), alors ρ(p,x(1,n)) n'est pas défini, et la séquence d'entrée x(1,n) est dite *inapplicable* à l'état p.

Pour qu'une machine R'(X,Y',Z,δ',λ') dans un état initial q puisse remplacer R dans l'état initial p, il faut et il suffit selon Ginsburg que: chaque fois que la réponse ρ(p, x(1,n)) est définie, la réponse ρ'(q, x(1,n)) soit définie et égale à la première.

Cette définition, la première du genre donnée historiquement, présente un inconvénient pratique. En effet, la notion de réduction qui en découle est telle que l'on peut réduire la machine R de la table 1 en supprimant simplement les états 1 et 2, car aucune séquence d'entrée n'est applicable à ces états au sens ci-dessus. On obtient ainsi la table 5.

	a	b
3	- ; 0	- ; -
4	- ; 1	- ; -

Table 5

On remarque aussi que la seule séquence applicable aux états 3 et 4 dans les deux machines est la séquence x = a, de longueur 1. Or, cette réduction (table 5) n'est pas conforme au cahier des charges à partir duquel on a construit la table 1, et qui est le suivant: supposons que les symboles 0 et 1 soient codés respectivement par les séquences b b a et b a a. On veut construire une machine de décodage. Cette machine ne recevra que des séquences d'entrées formées de blocs b b a ou b a a, et devra donner tous les trois instants le signal de sortie 0 ou 1 correspondant au dernier bloc reçu:

entrée	b	b	a	b	b	a	b	a	a
sortie	-	-	0	-	-	0	-	-	1

Le signal de sortie aux autres instants peut être quelconque. On voit immédiatement que toute réalisation de la table 1 satisfait ce cahier des charges, mais que ce n'est pas le cas pour la table 5. La notion de séquence applicable de Ginsburg est visiblement trop restrictive, et elle n'a pas été reprise par d'autres auteurs.

2.3.3. Fonction de réponse λ

La fonction de réponse adoptée généralement aujourd'hui pour les machines incomplètement spécifiées (ref. 1, 4) est la suivante. Considérant les relations (1), on dit maintenant que la séquence x(1,n) est *applicable* à l'état p si et seulement si chacun des éléments y(2), y(3),...,y(n), z(n) est défini. Dans ce cas, la *réponse de la machine pour l'état initial* p *et pour la séquence d'entrée* x(1,n) est par définition *l'élément* z(n). Cette réponse est définie même si les z(i) pour i = 1,...,n-1 ne sont pas définis. La réponse z(n) est notée λ (p,x(1,n)), ce qui étend simplement le domaine de la fonction de sortie λ.

La théorie de la réduction qui découle de ce choix d'une fonction de réponse ne permet plus de réduire la table 1 sous la forme de

la table 5.

2.3.4
Fonction de réponse du modèle généralisé

Les deux paragraphes qui précèdent montrent bien qu'en général un modèle de machine non déterministe admet plusieurs interprétations et qu'il convient de choisir celle qui corres pond aux intentions des utilisateurs du modèle.

Nous sommes maintenant confrontés à ce problème pour le modèle de machine généralisé (definition 1). Considérons par exemple la machine R(X,Y,Z) donnée par la table 6, où X = {a,b} Y = {1,2,3,4}, Z = {s,t,u,v}, et 1.a = {1,2}, 1/a = {s,t}, etc.

	a	b
1	1,2 ; s,t	1 ; u
2	3,4 ; u,v	1 ; v
3	1,2,3,4 ; s,t,u,v	1 ; s
4	1,2,3,4 ; s,t,u,v	1 ; s

Table 6

Soient p un état initial et x(1,n) une séquence d'entrée. Vu son caractère non déterministe, il est clair que la machine R admet plusieurs réponses, et nous allons définir ce qu'il convient d'appeler une *réponse admise* par R pour la séquence d'entrée et l'état initial donnés.

Une première idée est de considérer R comme une représentation de l'ensemble de ses réalisations (Cf. sect. 2.2), une réalisation de R étant simplement une table d'états complètement spécifiée tirée de la table 6 en remplaçant chaque ensemble d'états futurs y.x par un élément arbitraire de cet ensemble, et de même pour chaque ensemble de signaux de sortie y/z. Lorsqu'on place l'une quelconque de ces réalisations dans l'état initial p et qu'on lui applique la séquence d'entrée x(1,n), on obtient la séquence d'états y(1,n) et la sequence de sortie z(1,n) calculées par les relations (1), où δ et λ sont les fonctions de transition et de sortie de cette réalisation particulière. Vis-à-vis de la table 6, ces sequences vérifieront les relations suivantes:

$$\begin{cases} y(1) = p & (2) \\ y(k+1) \in y(k).x(k) \quad \text{pour } k = 1,\ldots, n-1 & (3) \\ z(k) \in y(k)/x(k) \quad \text{pour } k = 1,\ldots, n & (4) \end{cases}$$

Mais inversément, des séquences y(1,n) et z(1,n) qui vérifient ces relations ne sont pas nécessairement engendrées par une réalisation de R. Par exemple, pour p = 1 et x(1,3) = aab, les séquences y(1,3) = 112 et z(1,3) = stv satisfont les relations (2), (3) (4), ce qu'on peut exprimer en disant que le diagrame

$$1 \xrightarrow{a\ ;\ s} 1 \xrightarrow{a\ ;\ t} 2 \xrightarrow{b\ ;\ v}$$

est compatible avec la table 6. Mais il est clair que ce diagramme n'est compatible avec aucune réalisation de R, puisqu'on ne saurait avoir à la fois $\delta(1,a) = 1$ et $\delta(1,a) = 2$, ni $\lambda(1,a) = s$ et $\lambda(1,a) = t$, dans une même réalisation. On peut même constater qu'aucune des réalisations possibles de R ne peut donner la réponse z(1,3) = stv pour cet état initial et cette séquence d'entrée.

Néanmoins, nous dirons que la séquence z(1.3) = = stv est bien une réponse admise par la machine R. Nous posons donc la définition qui suit.

DEFINITION 2. Soient R(X,Y,Z) une machine séquentielle au sens de la définition 1, p un état initial, et x(1,n) une séquence d'entrée. Nous dirons qu'une séquence de sortie z(1,n) est une *réponse admise par R pour l'état initial p et la séquence d'entrée* x(1,n) s'il existe une séquence d'états y(1,n) telle que les relations (2), (3), (4) soient vérifiées. On désignera par $R_p(x(1,n))$ l'ensemble des réponses ainsi admises. La fonction qui à chaque état p et à chaque séquence x(1,n) associe l'ensemble $R_p(x(1,n))$ sera la *fonction de réponse* de R.

Le diagramme de la figure 7 illustre cette définition. Il représente l'ensemble R_2 (aab) pour la machine de la table 6. Cet ensemble se compose de toutes les séquences de sortie telles que stv, pouvant être "lues le long d'une branche" de cet arbre.

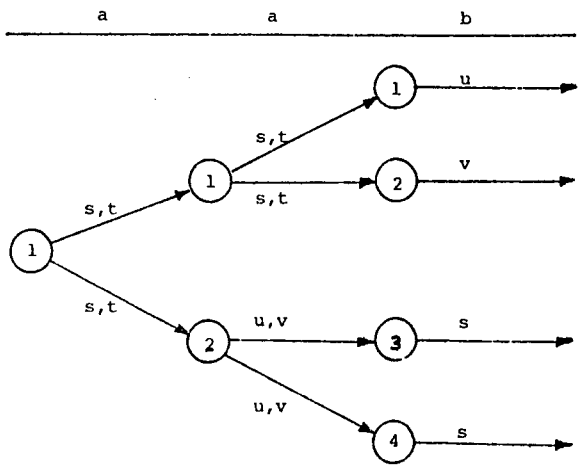

Figure 7

Une autre façon d'illustrer la définition 2 est de considérer l'ensemble des réalisations de R. Une realisation particulière peut être représentée par le schéma de la figure 8, où S est une machine combinatoire qui calcule - l'état futur $\delta(y,x)$ et le signal de sortie $\lambda(y,x)$, et D est un registre d'état.

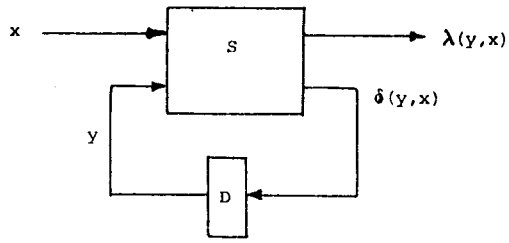

Figure 8

Imaginons que l'on ait construit ainsi toutes les réalisations possibles de R, et que leurs systèmes combinatoires respectifs soient S_1, $S_2,\ldots,S_k$. La définition 2 revient à dire -- que les réponses admises par R sont les réponses que peut fournir le système de la figure 9, dans lequel on s'autorise à manipuler de - façon arbitraire les variables de commande -- des multiplexeurs (en synchronisme avec l'horlogue du système).

Les figures 7 et 9 peuvent suffire à justi--- fier la proposition suivante qui découle de - la définition 2.

Proposition 1: Soient R(X,Y,Z) une machine et p un état quelconque.

Pour toute séquence d'entrée x(1) de longueur 1, on a

$$R_p(x(1)) = p/x(1). \qquad (5)$$

Pour toute séquence d'entrée x(1,n) de lon-- gueur n > 1, on a

$$R_p(x(1,n)) = R_p(x(1)) \bigcup_{q\in p.x(1)} R_q(x(2,n)) \qquad (6)$$

La démonstration rigoureuse de la formule de récurrence (6) peut être trouvée dans la réf. 5 (chap 2). Les ensembles $R_p(x(1))$, $R_q(x(2,n))$ sont des langages sur l'alphabet Z; la reunion et le produit dans la formule (6) sont les opérations régulières usuelles (réf. 5, chap.1).

Le choix de la définition 2 se justifie essentiellement par le fait qu'il entraîne essen-- tiellement les mêmes principes de réduction - que la théorie classique pour les machines -- particulières de la forme illustrée par la table 3, forme correspondant aux machines classiques incomplètement spécifiées (table 1).

Il faut voir cependant que ce n'est pas le -- seul choix possible. En effet, lorsqu'on ---- applique une séquence d'entrée x(1,n) à un -- état p, on peut s'intéresser seulement aux -- derniers éléments z(n) des réponses admises - z(1,n). Par exemple dans la figure 7, où p=1 et x(1,3) = aab, ces éléments sont u,v,s. On pourrait considérer cet ensemble {u,v,s} --- comme la réponse de la machine R. On définirait ainsi une autre fonction de réponse, -- qu'il est naturel de noter p/x(1,n). Dans -- l'exemple cité: 1/aab = {u,v,s} .En ce qui -

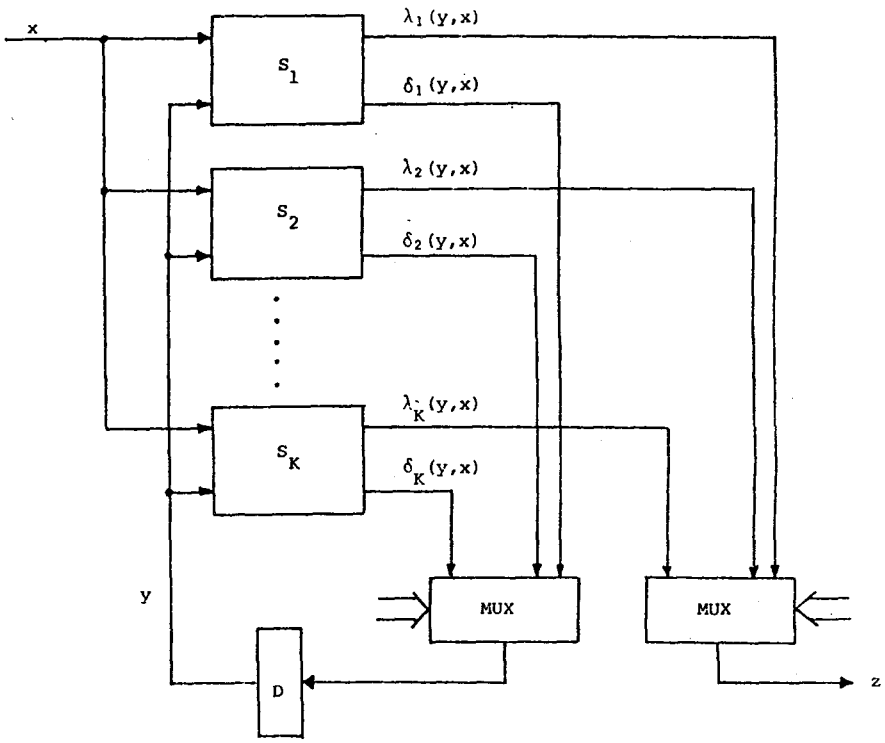

Figure 9

concerne la réduction, le choix de l'une ou l'autre des fonctions de réponse $R_p(x(1,n))$, p/x(1,n) n'est pas indifférent pour les machines non déterministes en général. Tout ce qu'on peut dire est que le langage $R_p(x(1,n))$ est contenu dans le produit des langages p/x(1), p/x(1,2),...,p/x(1,n).

3. APPLICATIONS DU MODELE GENERAL

On passe en revue dans cette section quelques applications possibles du modèle de machine - général introduit dans la section 1.2.

3.1. SYNTHÈSE DES SYSTÈMES DIGITAUX

3.1.1.

Le cas de non déterminisme le plus simple se rencontre couramment dans les problèmes séquentiels usuels. Il se trouve par exemple - dans la table d'états incomplètement spécifiée qu'est la table 10.

	1	2	3	4
a	b ; --	e ; 0-	e ; 00	b ; -1
b	c ; 00	d ; --	- ; -1	- ; -0
c	- ; -1	e ; -1	a ; 00	b ; 01
d	a ; -0	b ; 1-	- ; --	- ; --
e	b ; 0-	- ; -0	- ; -0	d ; 11

Table 10

Dans cette machine R(X,Y,Z) on a X = 1,2,3,4, Y = a,...,e et l'alphabet de sortie est l'ensemble des couples (,0,0), (0,1), (1,0),(1,1) notés simplement 00, 01, 10, 11. En d'autres termes Z est le produit cartésien $\{0,1\}^2$. Les notations --, -0, 0-, ... représentent des sous-ensembles de Z, à savoir:

-- représente l'ensemble Z lui-même
-0 " " {00,10}
0- " " {00,01}
etc.

Ce genre de table d'états, où Z est un produit cartésien et où pour chaque état total (y,x) l'ensemble y/x est un sousproduit cartésien de Z n'est pas traité dans les ouvrages théoriques classiques, qui se bornent toujours au cas où y/x est toujours: soit un seul élément de Z, soit tout l'ensemble Z. La raison de ceci est que la notion classique de séquence applicable perd sa clarté lorsqu'un signal de sortie peut être non seulement défini ou non défini, mais encore partiellement défini, comme c'est le cas dans cet exemple.

3.1.2.
On remarque que dans l'exemple qui précède l'ensemble y/x est toujours un sous-ensemble de Z de la forme U x V où U et V sont deux sous-ensembles de {0,1}. C'est ce que nous avons appelé un "sous-produit cartésien" de Z. On exprime aussi cette propriété en disant que dans le signal de sortie (z_1,z_2) les variables z_1,z_2 sont indépendantes.

On recontre cependant des machines où les ensembles y/x ne sont pas de cette forme. Considérant par exemple l'alphabet Z ci-dessus (table 10), on pourrait avoir y/x = {01, 10} pour un certain état total (y,x). Il s'agirait alors d'une machine à deux variables de sortie z_1,z_2 qui auraient ce qu'on appelle des "don't care" liés.

Cette situation se présente souvent lorsqu'un système est décomposé a priori et que la machine R(X,Y,Z) est une composante du système, dont les autres composantes sont fixées au départ. Un exemple est donné dans la figure 11, où R possède 6 variables de sortie S_0,S_1,S_2,S_3,M,C_n qui commandent une unité arithmétique et logique 74181. Cette unité effectue des opérations arithmétiques et logiques sur des mots de 4 bits A et B, le résultat étant donné sous la forme d'un mot de 4 bits F et d'un bit de retenue C_{n+1} (carry). L'alphabet de sortie Z de la machine R est ici le produit cartésien $\{0,1\}^6$. Cette machine a pour rôle de commander des séquences d'opérations de l'ALU, selon un certain cahier des charges.

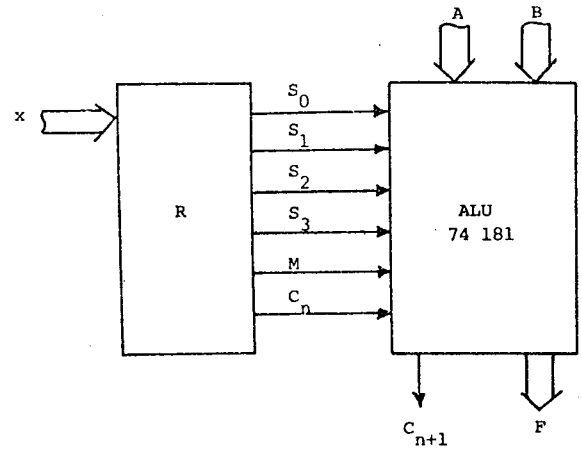

Figure 11

Supposons que dans l'état y et pour le signal x, la machine R doive commander l'opération F = A + B (somme arithmétique). Selon les specifications de l'ALU, il y a trois signaux de sortie qui permettent de commander cette opération, à savoir les trois vecteurs z_1,z_2,z_3 suivants:

	S_3	S_2	S_1	S_0	M	C_n
z_1 :	1	1	1	1	0	1
z_2 :	1	1	1	1	0	0
z_3 :	0	0	0	1	0	0

En composant la table d'états de R, on pourra donc écrire y/x = $\{z_1,z_2,z_3\}$, car il n'y a aucune raison de choisir a priori l'un des vecteurs z_1,z_2,z_3: cela pourrait limiter les possibilités de réduction de la table.

3.2 RÉSEAUX DE PETRI: MICROPROGRAMMES NON DÉTERMINISTES.

Les exemples qui précèdent montrent qu'il est tout à fait courant de recontrer des machines R(X,Y,Z) dans lesquelles la fonction de sortie y/x est de type non déterministe. Le cas où la fonction de transition y.x n'est pas déterministe est plus rarement enviasagé à l'heure actuelle, mais les exemples qui suivent suggèrent qu'il y a lieu de s'y intére-

ser.

3.2.1
Réseaux de Petri

Nous ne pouvons pas faire ici un exposé détaillé sur l'application des réseaux de Petri à la synthèse des systhèmes logiques et nous renvoyons le lecteur à l'ouvrage de Daclin et Blanchard (réf. 6). Rappelons seulement qu'on se limite dans cette application aux réseaux de Petri *sauf*, c'est-à-dire dont les places ne contiennent jamais plus d'un marqueur. On peut ainsi associer à chaque place une variable logique Y qui vaut 1 si la place contient un marqueur et 0 lorsqu'elle est vide. S'il y a en tout n places, donc n variables de marquage $Y_1,\ldots,Y_n$ le vecteur de marquage $(Y_1,\ldots,Y_n)$ définit l'état du réseau.

Pour représenter avec un tel réseau un système logique dont les variables d'entrée sont $X_1,\ldots,X_m$ et les variables de sortie $Z_1,\ldots,Z_p$, on associe à chaque transition du réseau une condition logique $E(X_1,\ldots,X_m)$, fonction booléenne des variables d'entrée (fig. 12).

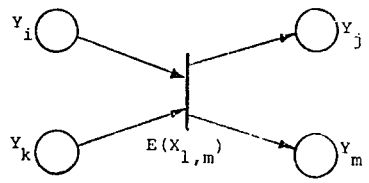

Figure 12

Lorsque le système se trouve dans un état $y = (Y_1,\ldots,Y_n)$, il y a pour un vecteur d'entrée $x = (X_1,\ldots,X_m)$ plusieurs transitions franchissables en général, donc plusieurs états (marquages) futurs possibles, encore que certains auteurs introduisent ici des restrictions qui nous paraissent inutiles. Le modèle général du réseau de Petri admet que dans un ensemble de transitions franchissables, une seule d'entre elles est franchie, celle-ci pouvant être choisie arbitrairement. Si l'on définit les variables de sortie Z_i comme fonctions (non déterministes) des variables X_i, Y_i on voit qu'un réseau de Petri ainsi interprété n'est autre qu'une machine séquentielle R(X,Y,Z) dont les fonctions de transition y.x et de sortie y/x sont non déterministes.

3.2.2
Microprogrammes non déterministes

Dans un article sur les systèmes séquentiels microprogrammés, Mange et Sanchez /7/ présentent une manière de réaliser un système séquentiel, donné par une table d'états classique, au moyen d'un microprogramme qui ne comporte que deux types d'instruction: le test d'une variable d'entrée, et l'assignation d'une valeur aux variables de sortie. Ils considèrent par exemple la table d'états 13, à deux états A et B, deux variables d'entrée x_1 et x_2, et une variable de sortie z qui peut prendre les valeurs 0,2,3,4,6.

	x_1x_2 0 0	0 1	1 1	1 0
A	A;0	B;-	A;2	A;3
B	A;-	B;0	B;4	B;6

Table 13

Une façon de passer de cette table à un microprogramme du type mentionné ci-dessus consiste à construire un programme séparé pour chacune des lignes de la table d'états, puis à relier ces programmes entre eux (fig. 14). La construction séparée des programmes A et B est évidente. Le point qui nous intéresse ici est la façon de les relier. En effet lorsqu'on sort de l'instruction Nº 3 du programme A pour $x_2 = 1$, on est renvoyé au programme B, conformément au $\delta(A,01) = B$ de la table 13. Mais comme x_1 et x_2 viennent d'être testés (avec le résultat $x_1x_2 = 01$), on peut passer indifféremment à l'une quelconque des instructions Nº 7,9, ou 12 du programme B. Ceci est indiqué par B(7,9,12) à la sortie du programme A. De même en sortant du programme B, on peut passer indifféremment à l'une quelconque des instructions Nº 1,3,6 du programme A. Le programme résultant peut être considéré comme une machine séquentielle non déterministe à 12 états (un état par instruction), avec l'alphabet de sortie Z' = {0,2,3,4,6,N}. Les éléments 0,2,3,4,6 de Z' représentent respectivement les opérations z: = 0,...,z: = 6, et N représente l'opération neutre (NOP). La table de cette machine est la table 15, où l'on remarque les ensembles d'états futurs {7,9,12}

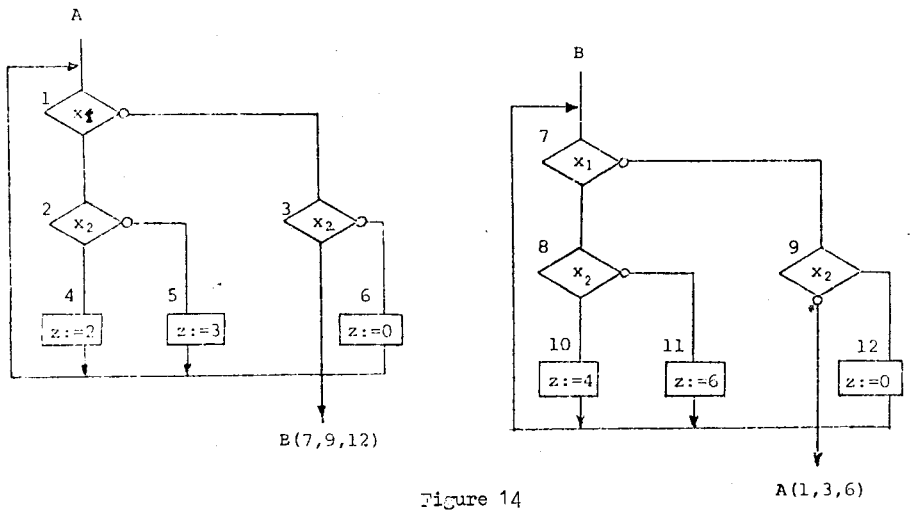

Figure 14

et {1,3,6} dans les lignes 3 et 9. Il est clair que ces deux lignes peuvent fusionner si l'on remplace {7,9,12} par 12 et {1,3,6} par 6. C'est un exemple élémentaire de réduction de table d'états non déterministe.

	0 0	0 1	1 1	1 0
1	3 ; N	3 ; N	2 ; N	2 ; N
2	5 ; N	4 ; N	4 ; N	5 ; N
3	6 ; N	7,9,12 ; N	7,9,12 ; N	6 ; N
4	1 ; 2	1 ; 2	1 ; 2	1 ; 2
5	1 ; 3	1 ; 3	1 ; 3	1 ; 3
6	1 ; 0	1 ; 0	1 ; 0	1 ; 0
7	9 ; N	9 ; N	8 ; N	8 ; N
8	11 ; N	10 ; N	10 ; N	11 ; N
9	1,3,6 ; N	12 ; N	12 ; N	1,3,6 ; N
10	7 ; 4	7 ; 4	7 ; 4	7 ; 4
11	7 ; 6	7 ; 6	7 ; 6	7 ; 6
12	7 ; 0	7 ; 0	7 ; 0	7 ; 0

Table 15

Nous ne pouvons pas exposer ici de façon plus complète la relation qui existe entre les notions de microprogramme et de machine séquentielle. Nous renvoyons le lecteur au texte de Davio /réf.8/, basé sur les travaux de Glushkov /réf.9/. En considérant les programmes -- comme des machines séquentielles, on peut leur appliquer les méthodes de transformation et de réduction de ces machines. L'exemple ci-dessus donne à penser que de façon générale lorsqu'un programme est conçu par segments, la combinaison de ces segments entre eux peut se traduire par un programme non déterministe.

4. THEORIE DE LA REDUCTION

Il n'est pas possible de faire ici un exposé complet de cette théorie, d'autant plus qu'elle présente encore des problèmes non résolus. Nous voulons seulement en donner les bases et suggérer les travaux qui pourraient faire suite à ce bref exposé.

4.1 LE PROBLÈME DE LA RÉDUCTION

Comme nous l'avons dit au paragraphe 2.3.1, - réduire une machine R(X,Y,Z), c'est trouver - une machine R'(X,U,Z) ayant les mêmes alpha-- bets d'entrée et de sortie que R, moins d'é-- tats que R, et dont le comportement entrée- - sortie soit acceptable par l'environnement de R. Nous voulons préciser ces termes en par--- tant des définitions 1 (sect. 3.2) et 2 (pa-- ragr. 2.3.4).

DEFINITION 3. Soient R(X,Y,Z) et R'(X,U,Z) - deux machines, y un état de R et u un état de R'. On dira que l'état u *simule* (covers) l'état y si pour toute séquence d'entrée --- x(1,n) on a

$$R'_u(x(1,n)) \subset R_y(x(1,n)). \qquad (7)$$

La relation (7) signifie que toute réponse - z(1,n) admise par la machine R' pour la sé-- quence x et l'état initial u est admise par la machine R pour l'état initial y. C'est -- pourquoi l'on peut remplacer la machine R -- dans l'état initial y par la machine R' dans l'état initial u.

DEFINITION 4. On dit que la machine R'(X,U,Z) *simule* la machine R(X,Y,Z) si pour chaque -- état y de R il existe un état u de R' qui simule y au sens de la définition précédente. La machine R' est une *réduction* de R si elle simule R et si le nombre d'éléments de U est inférieur ou égal celui de Y. Enfin R' est - une *réduction minimale* de R si R' est une réduction de R et s'il n'existe pas de réduc-- tion R" de R ayant moins d'états que R'.

Il est évident que la relation R' *simule* R, tout comme la relation R' *est une réduction de* R, est réflexive et transitive. On dira - que R est *réduite* lorsque R est une réduction minimale d'elle-même.

4.2 MACHINES QUOTIENTS

4.2.1. Recouvrements compatibles

On sait que dans la théorie classique des machines incomplètement spécifiées le problème de la réduction d'une machine R(X,Y,Z) se ramène à celui de trouver un famille de sous-ensembles de Y qui recouvre Y, et qui jouisse - de certaines propriétés vis-à-vis des fonc--- tions de transition et de sortie δ,λ de R. -- Nous voulons généraliser cette notion dans le cas d'une machine non déterministe quelconque.

A cet effet, nous partons de la théorie classique en considérant la machine de la table - 16, qui reproduit la table 1. Soit $U = \{\alpha,\beta\}$ un ensemble à deux éléments,et associons à -- chaque élément u de U un sous-ensemble $\sigma(u)$ - de Y, selon la table 17. On dit qu'on a défini ainsi un *recouvrement* $\sigma: U \to Y$, parce que chaque élément de Y appartient à l'un au --- moins des ensembles $\sigma(u)$ appelés les *classes* du recouvrement.

	a	b
1	- ; -	2 ; -
2	4 ; -	3 ; -
3	1 ; 0	- ; -
4	1 ; 1	- ; -

Table 16

u	$\sigma(u)$
α	{1,2,3}
β	{1,2,4}

Table 17

Le recouvrement σ (table 17) jouit des deux - propriétés suivantes vis-à-vis de la machine considérée (table 16).

Première propriété (8)

Pour une classe quelconque $\sigma(u)$ et un signal d'entrée x quelconque, il existe une classe $\sigma(v)$ telle que $\delta(\sigma(u), x) \subset \sigma(v)$. La notation $\delta(\sigma(u),x)$ désigne l'ensemble des $\delta(y,x)$ spécifiés correspondant aux éléments y de $\sigma(u)$. On a par exemple:

$$\delta(\sigma(\alpha),a) = \delta(\{1,2,3\},a) = \{1,4\} \subset \sigma(\beta).$$

Deuxième propriété

Pour une classe quelconque $\sigma(u)$ et un signal d'entrée x quelconque, l'ensemble $\lambda(\sigma(u),x)$ possède au plus un élément. L'ensemble ----- $\lambda(\sigma(u), x)$ est par définition l'ensemble des $\lambda(y,x)$ spécifiés pour les éléments y de $\sigma(u)$. On a par exemple:

$$\lambda(\sigma(\alpha),a) = \lambda(\{1,2,3\},a) = \{0\}$$
$$\lambda(\sigma(\alpha),b) = \lambda(\{1,2,3\},b) = \emptyset \quad \text{(ensemble vide)}.$$

La première (resp. deuxième) propriété s'ex-- prime en disant que le recouvrement σ est *compatible* avec la fonction de transition (resp. de sortie) de la machine.

Nous voulons maintenant exprimer ces deux propriétés au moyen des fonctions y.x et y/x définies par la table 3 qui représente cette machine selon la définition 1. Pour cela il suf

fit de remarquer que

$$y.x = \begin{cases} \{\delta(y,x)\} & \text{si } \delta(y,x) \text{ est défini} \\ Y & \text{si } \delta(y,x) \text{ n'est pas défini} \end{cases} \quad (10)$$

$$y/x = \begin{cases} \{\lambda(y,x)\} & \text{si } \lambda(y,x) \text{ est défini} \\ Z & \text{si } \lambda(y,x) \text{ n'est pas défini.} \end{cases} \quad (11)$$

On voit alors que la relation $\delta(\sigma(u),x) \subset \sigma(v)$ peut s'exprimer en disant que pour tout $y \in \sigma(u)$ on a $y.x \cap \sigma(v) \neq \emptyset$. Par suite, la propriété (8) s'énonce:

Première propriété (12)

Pour tout $u \in U$ et pout tout $x \in X$ il existe un $v \in U$ tel que

$y.x \cap \sigma(v) \neq \emptyset$ quel que soit $y \in \sigma(u)$.

Secondement, dire que l'ensemble $\lambda(\sigma(u),x)$ possède au plus un élément, c'est aussi dire que l'intersection des ensembles y/x tels que $y \in \sigma(u)$ n'est pas vide. Par suite la propriété (9) s'énonce:

Deuxième propriété (13)

Pour tout $u \in U$ et pour tout $x \in X$ on a

$$\bigcap_{y \in \sigma(u)} y/x \neq \emptyset$$

Nous proposons maintenant de prendre les propriétés (12) et (13) comme *définition* d'un recouvrement compatible avec une machine R(X,Y,Z), dans le cas général d'une machine de forme quelconque.

4.2.2
Machines quotients

Supposons donnés une machine R(X,Y,Z) et un recouvrement $\sigma: U \to Y$ compatible avec R, où U est un ensemble fini $\{\alpha,\beta,\ldots\}$. En vertu de la propriété (12) de σ, on peut associer à chaque couple d'éléments $u \in U$, $x \in X$ un sous-ensemble *non vide* de U, que l'on notera u.x, et tel que

$$\forall v \in u.x, \ \forall y \in \sigma(u): \quad y.x \cap \sigma(v) \neq \emptyset \quad (14)$$

De même, en vertu de la propriété (13) de σ, on peut associer à chaque couple d'eléments $u \in U$, $x \in X$ un sous-ensemble *non vide* de Z, que l'on notera u/x, et tel que

$$\forall y \in \sigma(u): \quad u/x \subset y/x. \quad (15)$$

Autrement dit on peut construire une machine R'(X,U,Z) dont les fonctions de transition et de sortie u.x, u/x vérifient (14) et (15). Une telle machine est appelée une *machine quotient* de la machine R par le recouvrement σ. Si dans cette construction l'on choisit pour chaque couple u,x les plus grands ensembles u.x et u/x vérifiant (14) et (15), on dira que la machine R'est *la* machine quotient de R par σ. Cette terminologie est tirée de la /réf.5/.

Au cas où R a la forme particulière d'une machine classique, c'est-à-dire où les fonctions y.x et y/x sont dédinies par (10) et (11), les propriétés (14) et (15) d'une machine quotient s'énoncent aussi.

$$\forall v \in u.x: \quad \delta(\sigma(u), x) \subset \sigma(v) \quad (16)$$

$$\forall y \in \sigma(u): \quad \text{si } \lambda(y,x) \text{ est défini, alors } \{\lambda(y,x)\} = u/x \quad (17)$$

Par exemple, étant donnés la machine R(X,Y,Z) de la table 16 et le recouvrement $\sigma: U \to Y$ de la table 17, on peut construire la machine quotient R'(X,U,Z) de la table 18.

	a	b
α	β ; 0	α ; -
β	β ; 1	α ; -

Table 18

On sait par la théorie classique que la machine quotient ainsi construite est une réduction de R. Nous allons généraliser ceci dans le théorème suivant, qui constitue le principal résultat actuel sur la réduction des machines non déterministes.

4.2.3

THEOREME. Soient R(X,Y,Z) une machine quelconque, $\sigma: U \to Y$ un recouvrement de Y compatible avec R, et R'(X,U,Z) une machine quotient de R par σ. Tout état u de R' simule chacun des états y de R tels que $y \in \sigma(u)$. Par suite R' simule R.

DEMONSTRATION

Conformément à la définition 3 (sect. 4.1), nous devons montrer que la relation

$$y \in \sigma(u) \Rightarrow R'_u(x(1,n)) \subset R_y(x(1,n)) \quad (18)$$

est vraie quels que soient $y \in Y$, $u \in U$, et la séquence d'entrée $x(1,n)$. Nous procédons par récurrence sur la longueur n de x. Si n = 1, on a $x(1,n) = x \in X$, et la relation (18) est vraie en vertu de la formule (5) et de la propriété (15) d'une machine quotient. Supposons que la relation (18) soit vraie pour n, quels que soient y et u (hypothèse de récurrence). Considérons une séquence $x(1, n+1)$, et supposons que $y \in \sigma(u)$. En appliquant la formule (6) on peut écrire

$$R'_u(x(1,n+1)) = R'_u(x(1)) \bigcup_{v \in u.x(1)} R'_v(x(2,n+1)) \qquad (19)$$

$$R_y(x(1,n+1)) = R_y(x(1)) \bigcup_{q \in y.x(1)} R_q(x(2,n+1)). \qquad (20)$$

Comme on a supposé que $y \in \sigma(u)$, on a $R'_u(x(1)) \subset R_y(x(1))$ puisque $x(1)$ est une sequence de longueur 1.

D'autre part, en vertu de la propriété (14) d'une machine quotient, on voit que pour chaque terme $R'_v(x(2,n+1))$ dans (19) il existe un terme $R_q(x(2,n+1))$ dans (20) tel que $q \in \sigma(v)$, donc tel que

$R'_v(x(2,n+1)) \subset R_q(x(2,n+1))$, puisque $x(2,n+1)$ est de longueur n. On en déduit que $R'_u(x(1,n+1)) \subset R_y(x(1,n+1))$. Ainsi par récurrence, la relation (18) est vraie quel que soit n.

4.2.4.
Problèmes ouverts

Le théorème qui précède indique une méthode de réduction valable pour une machine non déterministe quelconque, méthode qui peut se résumer comme suit: 1) trouver un recouvrement compatible avec R et 2) construire une machine quotient de R par ce recouvrement. L'étape 1) de cette méthode reste à élaborer dans le cas général. Mais le problème majeur qui se pose d'abord est celui de l'universalité de la méthode, à savoir la question suivante: est-ce que toute réduction R'(X,U,Z) d'une machine R(X,Y,Z) peut être trouvée par cette méthode. Plus précisément, étant données une machine R(X,Y,Z) et une reduction R'(X,U,Z) de R au sens de la définition 4, est-ce qu'il existe un recouvrement $\sigma: U \to Y$ tel que R' soit une machine quotient de R par σ ?

On peut répondre par l'affirmative, à quelque nuance près, dans le cas ou R est définie par une table d'etats classique incomplètement spécifiée /réf.5/. Plus exactement on montre dans ce cas: si R'(X,U,Z) simule R et si chaque état u de R' simule au moins un état de R, alors le recouvrement $\sigma: U \to Y$ dont chaque classe $\sigma(u)$ est formée de l'ensemble des états y de R qui sont simulés par u, est un recouvrement compatible avec R, et R' est une machine quotient de R par ce recouvrement. Nous n'avons pas pu étendre ce résultat pour des machines quelconques, et le problème est ouvert à notre connaissance.

En attendant la réponse à cette question, on peut élaborer en détail la méthode de réduction indiquée ci-dessus, qui consiste essentiellement à trouver les recouvrements compatibles avec une machine R(X,Y,Z) donnée. Ce problème, plus facile semble-t-il que le précédent, est bien résolu pour les machines classiques. Mais les méthodes applicables dans ce cas ne se transposent pas immédiatement au cas général. Par exemple il n'est pas possible d'affirmer dans le cas général qu'un sous-ensemble C de Y est une classe de compatibilité sitôt que chaque sous-ensemble {p,q} de C est une classe de compatibilité. Cette affirmation peut déja être fausse pour une machine R dont seule la fonction de sortie y/x est non déterministe, du type mentionné au paragr. 3.1.2.

Pour terminer, nous donnons un exemple d'une machine R non déterministe (table 19) et d'un recouvrement σ compatible avec R (table 20), c'est-à-dire jouissant des propriétés (12) et (13). On a par exemple:

$$\forall y \in \sigma(\alpha) : y.0 \cap \sigma(\gamma) \neq \emptyset$$

La machine quotient de R par σ est donnée par la table 21.

	0	1
a	a,b ; 0,2	c,e ; 0,1,2,3
b	d ; 1	a,b,c,d,e ; 0,3
c	b ; 2	a,b,c ; 3
d	a,b,c,d,e ; 1,3	c ; 0,2
e	a,d ; 2,3	e ; 1,2

Table 19

u	σ(u)
α	{a,c}
β	{a,e}
γ	{b,d}

Table 20

	0	1
α	γ ; 2	α,β ; 3
β	α,β,γ ; 2	β ; 1,2
γ	γ ; 1	α ; 0

Table 21

5. BIBLIOGRAPHIE

/1/ T.L. BOOTH, "Sequential Machines and --- Automata Theory", Wiley, New York, 1967.

/2/ D. MANGUE, "Analyse et synthèse des systèmes logiques", Editions Georgi, St-Saphorin, 1978.

/3/ S. GINSBURG, "An Introduction to Mathematical Machine Theory", Addison-Wesley, - Reading, 1962.

/4/ A. GINZBURG, "Algebraic Theory of Automata", Academic Press, New York, 1968.

/5/ J. ZAHND, "Machines séquentielles", Editions Georgi,St-Saphorin, 1980.

/6/ E. DACLIN, M. BLANCHARD, "Synthèse des - systèmes logiques", Cepadues Editions, - Toulouse, 1976.

/7/ D. MANGUE, E. SANCHEZ, "Simplicité ou structuration? Deux méthodes pour la réalisation microprogrammée d'automatismes séquentiels", Le Nouvel Automatisme, --- t.XXVI (1981), nº 26, pp. 43-49

/8/ M. DAVIO, "Hardware implementation of - algorithmic computations", Philips internal Report R333, July 1976.

/9/ V.M. GLUSHKOV, "Automaton theory and formal microprogram transformation", Kibernetica, 1, pp. 1-9, 1965 (Engl. Transl. -- Cybernetics, pp. 1-8, Jan. 1968).