SUR UNE MÉTHODE ALGORITHMIQUE DE SYNTHÈSE D'UNE MACHINE SÉQUENTIELLE

par

Fernando SAEZ VACAS**
Ingénieur des Télécommunications (Madrid)
Maître ès sciences aéronautiques

Eric DACLIN*
Ingénieur E.N.S.E.E.H.T.
Maître ès sciences

INTRODUCTION

Dans le problème de la synthèse d'une machine séquentielle, on peut, grosso modo, distinguer trois étapes : le passage des données du système à construire à la représentation tabulée (table de fluence ou table des phases, dans le cas d'un système asynchrone); le codage des états internes de cette table (de façon, dans le cas d'une machine synchrone, à obtenir si possible une décomposition en sous-machines, et, dans le cas d'une machine asynchrone, de façon à éviter les phénomènes de courses et d'aléas); enfin, l'écriture des équations et le dessin du schéma logique de la machine (en essayant par exemple de minimaliser le nombre de composants nécessaires à la réalisation du circuit).

L'article que nous présentons ici traite de la première étape de la synthèse, dite « synthèse abstraite » dans la littérature. Plus modestement, d'ailleurs, nous avons abordé un cas particulier de ce stade, correspondant à une présentation bien particulière des données.

Celles-ci peuvent, en effet, être fournies sous trois formes :

— ou bien une série de correspondances entre séquences d'entrée et séquences de sortie de longueur finie ou infinie à partir d'un instant initial;
— ou bien l'exigence de la réalisation d'un nombre fini de cycles;
— ou bien l'exigence de la réalisation d'un nombre infini de cycles.

La première forme est plutôt d'usage théorique. La seconde forme correspond à la reconnaissance d'un certain nombre de séquences avec retour à l'état initial. La troisième forme semble plus adaptée aux problèmes utilisant des systèmes du type compteur ou registre à décalage. Dans le domaine industriel, on peut dire, en simplifiant, que la deuxième forme correspond à des problèmes du type « machine transfert » et la troisième à des problèmes du type « téléphone » ou « ascenseur ».

La méthode exposée dans les pages qui suivent et due, pour son esprit, à V.M. Gloushkov [2], [3]*, est applicable à tous les problèmes présentés sous la première et deuxième formes. C'est-à-dire qu'elle permet de résoudre les problèmes à un nombre fini de séquences entrée-sortie possibles, pratiquement quel que soit le nombre de ces entrées et de ces sorties. En effet, cette méthode est algorithmique et, donc, automatisable.

Le présent travail comprend quatre parties. La première (très brève) est un rappel des propriétés générales des automates finis (alias machines séquentielles) envisagés sous l'aspect de systèmes qui réalisent une correspondance entre des séquences d'entrée et des séquences de sortie. Y figure également une présentation intuitive du langage des expressions régulières. La deuxième donne rapidement l'esprit de la méthode de synthèse due à Gloushkov. La troisième est l'exposé d'un certain nombre de modifications que nous avons apporté, pour des fins de simplification à l'algorithme original de Gloushkov; l'accent y est mis en particulier sur le cas où les machines étudiées sont des machines asynchrones. La quatrième partie contient des exemples d'application. On n'a pas mentionné ici, pour ne pas alourdir l'exposé, l'organigramme d'un programme mis au point sur l'ordinateur IBM 360-44 du Centre d'Etudes et de la Recherche en Automatisme. On pourra trouver en [9] cet organigramme.

En annexe, enfin, nous avons exposé les règles de l'algèbre des expressions régulières, afin de ne pas surcharger inutilement le corps de cet article.

I - RAPPELS

Le comportement d'un automate fini (c'est-à-dire, pour simplifier, d'un système séquentiel) avec des alphabets d'entrée et de sortie X et Z ($x_j \in X$), $j = 1$,

(*) Ingénieur au C.E.R.A.
(**) Actuellement au laboratoire d'automatique de l'Ecole des Télécommunications de Madrid.

(*) Les chiffres entre crochets se réfèrent à la bibliographie qui figure en fin de l'article.

... m, ($Z_i \in Z$), i = 1 ... n peut être caractérisé en spécifiant *tous* les ensembles de séquences d'entrée qui induisent un signal de sortie Z_i, $\forall z_i$ E z. De tels ensembles qui peuvent être infinis sont décrits à l'aide d'une formule appelée *expression régulière* [1] - [6].

Il existe des méthodes qui permettent d'obtenir des tables d'états à partir des expressions régulières. Nous nous intéressons à la méthode algorithmique de Gloushkov [2] - [4], et en particulier nous fixons notre attention sur le traitement des expressions régulières itérées.

Nous allons établir un certain nombre des propriétés à l'égard de l'approche de Gloushkov.

1.1. Préliminaires. Machines séquentielles

1.1.1. Notations

Considérons un automate fini déterministe synchrone stationnaire à état initial comme une structure mathématique A

$$A = (X, Z, Q, g, f, q_0)$$

où

$X = \{ x \}$ ensemble fini d'entrées
$Z = \{ z \}$ — de sorties
$Q = \{ q \}$ — d'états

$g : Q \times X \to Q$ fonction séquentielle de transition
$f : Q \to z$ fonction de sortie au sens de Moore
q_0 : .état initial,

g et f sont manipulées le plus souvent sous forme de tables ou de graphes. Un premier pas de la synthèse est de trouver Q, g et f, connaissant la transformation $X(K) \to Z(K)$ pour tout K appartenant à l'intervalle d'observation.

1.1.2. Entrées - Sorties

Définissons par un X surmonté d'une barre ($\overline{X}$) une séquence des vecteurs d'entrée. La même notation est prise pour les séquences d'états et celles de sorties.

On aura par exemple :

pour

$$\overline{X} = X(1)\,X(2) \ldots X(k) \to \begin{cases} q(k) = g(q_0; \overline{X}) \\ \overline{Z} = f(q_0)\, f[q(1)] \ldots f[q(k)] \end{cases}$$

Autrement dit :

$$\overline{Z}(q_0; \overline{X}(k)) = f(q_0)\, f[q(1)] \ldots f[q(k)]$$

En général, on aura :

$\overline{Z}[q_j; \overline{X}(k)] = f(q_j)\, f[q(1)] \ldots f[q(k)]$ du fait de la propriété séquentielle de la transformation g.

Nous dirons que : $\overline{X}' < \overline{X}$, $\exists\, \overline{X}'' \longrightarrow \overline{X}'\,\overline{X}'' = \overline{X}$

$$\overline{X}'\,\overline{X}'' = \overline{X} \longrightarrow \overline{Z}'\,\overline{Z}'' = Z$$

avec $\overline{Z}' = \overline{Z}[q_0; \overline{X}'(k_1)]$

$\overline{Z}'' = Z[q(k_1) ; \overline{X}''(k_1, k_2)]$

$\overline{Z} = \overline{Z}[q_0; \overline{X}(k_2)]$

1.1.3. Autre écriture

D'après ceci, on peut donc formuler autrement la structure mathématique de la machine. Appelons $\overline{X}$ et $\overline{Z}$ les ensembles des séquences d'entrée et de sortie propres à cette machine.

$A = (\overline{X}, \overline{Z}, \overline{Q}, \overline{G}, \overline{F}, q_0)$
$= \{ \overline{X} \}$ ensemble des sequences d'entrée
$= \{ \overline{Z} \}$ — de sortie
$\overline{Q} = \{ \overline{q} \}$ — d'états
$G : \overline{X} \to Q$
$F : \overline{Q} \to \overline{Z}$

Dans la section suivante nous allons envisager la synthèse comme un problème où l'on connaît $\overline{X}$, $\overline{Z}$ et la transformation F.G et l'on veut trouver $\overline{Q}$, G et F.

1.1.4. Notion d'états équivalents

Deux états $q_i \in Q$ et $q_j \in Q$ sont équivalents lorsque, pour des séquences d'entrée identiques, les séquences de sortie sont identiques ([7], [8]).

Ceci s'exprime par :

$$q_i \simeq q_j \longleftarrow \forall\, \overline{X}, \overline{Z}\,(_iq\ ;\ \overline{X}) = Z\,(q_j{}'\overline{X})$$

1.2. Expressions régulières. Présentation intuitive

1.2.1. Exemple introductif

Considérons une machine logique qui réalise entre l'alphabet d'entrée (X_1, X_2, X_3) et l'alphabet de sortie (Z_1, Z_2) les correspondances :

$X_1\,X_2\,X_3\,\{ X_1\,X_3 \} \longrightarrow Z_1\,Z_2\,Z_2\,\{ Z_2, Z_1 \}$ (C1)
$X_1\,X_2\,X_2\,X_3 \longrightarrow Z_1\,Z_2\,Z_1\,{}_2Z$ (C2)

Cherchons à quelles conditions la sortie Z_1 se produira. Intéressons-nous d'abord à la corretpondance C2.

A partir de l'état initial, on voit que la machine fournira une sortie Z_1 :
— *ou bien* si elle reçoit la séquence de longueur unité X_1,
— *ou bien* si elle reçoit la séquence de longueur trois $X_1\,X_2\,X_2$.

La correspondance C1 nous permet de dire que,

(1) La notation $\{ X_i \ldots X_j \}$ signifie que la machine reçoit zéro fois, une fois, ..., p fois, ..., consécutivement la séquence d'entrée $X_i \ldots X_j$.

à partir du même état initial, la machine fournira une sortie Z_1 :

— *ou bien* si elle reçoit la séquence de longueur unité X_1,
— *ou bien* si elle reçoit la séquence de longueur cinq $X_1 X_2 X_3 X_1 X_3$,
— *ou bien* si elle reçoit la séquence de longueur sept $X_1 X_2 X_3 X_1 X_3 X_1 X_3$,
— *ou bien* si elle reçoit la séquence de longueur $(5 + 2n)$ $X_1 X_2 X_3 X_1 X_3 \ldots X_1 X_3$, n fois,

quel que soit n.

Pour résumer ce que nous venons de dire et en utilisant le signe $+$ comme celui qui signifie « ou », on peut écrire une expression $R(Z_1)$ (signifiant : « Z_1 est réalisée ») sous la forme :

$$R(Z_1) = X_1 + X_1 X_2 X_2 + X_1 + X_1 X_2 X_3 X_1 X_3 + X_1 X_2 X_3 X_1 X_3 X_1 X_3 + \ldots + + X_1 X_2 X_3 X_1 X_3 X_1 X_3 \ldots X_1 X_3 +$$

$R(Z_1)$ s'appelle *l'expression régulière* de Z_1. De même, chacun des termes séparés par un signe $+$, dans $R(Z_1)$, est une expression régulière.

On remarque, toutefois, dans $R(Z_1)$ deux caractéristiques qui semblent assez gênantes :

— la première (mais il est évident que nous pourrons très vite y remédier) est que X_1 apparaît deux fois tout seul. Est-il logique d'éliminer cette redondance?
— la seconde, beaucoup plus ennuyeuse, est que nous faisons appel à des expressions régulières infinies, ce qui complique singulièrement la tâche et alourdit quelque peu l'écriture.

Nous allons voir maintenant comment l'algèbre des expressions régulières permet, dans une certaine mesure, la simplification de $R(Z_1)$.

1.2.2. Algèbre des expressions régulières

a) Avant tout, insistons ici sur le fait qu'il s'agit pour nous *d'expliquer rapidement* et de façon aussi *concrète* que possible le mécanisme des expressions régulières. La rigueur mathématique sera donc parfois laissée de côté en cours de route, mais le résultat auquel nous arriverons sera le bon. Simplement, nous prendrons un chemin plus intuitif et moins rigoureux pour y arriver.

b) Commençons donc par essayer de répondre à la première question que nous venons de poser : peut-on éliminer, comme redondant, un des deux termes X_1 qui apparaît, isolé, dans l'expression $R(Z_1)$ que nous venons d'établir?

La réponse est, bien sûr, oui. En effet, que signifie l'écriture :

$$P = X + X \ ?$$

Tout simplement que l'événement P est réalisé à condition que la machine qui le produit reçoive, à partir de son état initial, X *ou bien* X. Donc on a :

$$X + X = X.$$

A propos de cette opération « $+$ », notons aussi une propriété importante. Cette propriété (sous-jacente dans notre explication) est que

$$X + Y = Y + X.$$

En effet, si $P = X + Y$, cela veut dire que P est réalisé par application de X *ou bien* par application de Y, à partir d'un état initial donné, c'est-à-dire, aussi, par application de Y *ou bien* par application de X.

c) Nous devrions maintenant répondre à la deuxième question que nous avons posée à la fin du § 1.2.1. Mais nous avons besoin, pour cela, d'une notion complémentaire que nous allons donner maintenant. Cette notion correspond, à peu près, à celle de « mise en facteur » dans l'algèbre classique.

Reprenons, par exemple, les deux premiers termes de $R(Z_1)$; soit

$$P(Z_1) = X_1 + X_1 X_2 X_2.$$

La signification de cette écriture a été expliquée plus haut. Ne serait-il pas possible de « mettre X_1 en facteur » dans les deux termes de P? Autrement dit, d'interpréter P, cette fois-ci, en disant : P est réalisé, à partir de l'état initial, si on reçoit X_1 suivi de « rien du tout » *ou bien* suivi de $X_2 X_2$?

Ceci impose que nous introduisions une expression régulière particulière qui explicite ce « rien du tout ». Appelons-le λ. Ce λ, suivant n'importe quelle expression régulière ne modifiera pas celle-ci (c'est un *élément neutre* pour la juxtaposition ou plutôt *concaténation* de deux expressions régulières). Et nous pouvons écrire $P(Z_1) = X_1 (\lambda + X_2 X_2)$, c'est-à-dire de façon plus générale :

$$X = X\lambda,$$

on a coutume de désigner λ sous le nom de « séquence de longueur nulle ».

Remarquons ici que l'on a, dans le cas général : $XY \neq YX$ (revenir à la signification intuitive de XY et YX). Mais : $\lambda X = X\lambda$. En effet, toute expression régulière précédée ou suivie de la séquence de longueur nulle n'est pas modifiée.

d) Les préliminaires étant posés, considérons de nouveau $R(Z_1)$. On peut l'écrire :

$$R(Z_1) = X_1 (\lambda + X_2 X_2) + X_1 X_2 X_3 X_1 X_3 (\lambda + X_1 X_3 + \ldots + X_1 X_3 \ldots X_1 X_3 + \ldots) = P(Z_1) + Q(Z_1) \cdot S(Z_1),$$

Quelle est la signification de $S(Z_1)$? C'est :

— *ou bien* on n'envoie rien à la machine (séquence de longueur nulle),
— *ou bien* on envoie $X_1 X_3$,
— *ou bien* on envoie $X_1 X_3 X_1 X_3$,
— *ou bien* on envoie $X_1 X_3 X_1 X_3 \ldots X_1 X_3$

⋮ p fois ⋮

— *ou bien* on envoie $X_1 X_3 X_1 X_3 \ldots X_1 X_3 \ldots X_1 X_3$

On dira que $S(Z_1)$ est l'*itération* de $X_1\ X_3$ et on notera

$$S(Z_1) = \{ X_1\ X_3 \} \text{ ou } < X_1\ X_3 > \text{ ou } (X_1\ X_3)^* \quad (1)$$

Nous pouvons donc écrire, finalement :

$$R(Z_1) = X_1 (\lambda + X_2 (X_2 + X_3\ X_1\ X_3 \{ X_1\ X_3 \}))$$

De façon analogue, on aurait trouvé :

$$R(Z_2) = X_1\ X_2 (\lambda + X_2\ X_3 + X_3 (\lambda + X_1 \{ X_3\ X_1 \})).$$

1.2.3. Formalisation. Propriétés diverses

Résumant de façon un peu plus abstraite ce que nous venons de voir, on peut, pour certaines classes importantes d'événements infinis, introduire dans la théorie des automates l'algèbre des expressions régulières avec trois opérations :

a - Union, notée PVQ ou $P + Q$,
b - Concaténation ou produit, notée $P \cdot Q$ ou PQ,
c - Itération, notée P^* ou $\{P\}$ ou $< P >$ [2] [3] [10].

Toute expression régulière peut être obtenue par un nombre fini d'applications des trois opérations précédentes, en tenant compte que :

1 - X_j pour $j = 1, 2, \ldots m$ est une expression régulière,
2 - λ, séquence de longueur nulle est une expression régulière,
3 - Φ, ensemble vide des séquences est aussi une expression régulière,
4 - P et Q sont des expressions régulières.

On peut trouver les principales règles de cette algèbre dans les références [1], [5].

En ce qui concerne les expressions régulières itérées, qui nous intéressent plus particulièrement, nous avons :

$$\bar{P}^* = \bigcup_{k=0}^{\infty} P^k$$

où $\bar{P}^2 = \bar{P}\bar{P}$: $P^3 = \bar{P}\bar{P}\bar{P}$ etc... et $\bar{P}^0 = \bar{\lambda}$.

On note sous barre l'ensemble représenté par l'expression régulière contenue.

Si l'on fait $P = i^*$, où $i = (X_1 + X_2 + \ldots + X_m)$ on a l'expression régulière tout à fait particulière i^*, qui représente l'ensemble de toutes les séquences possibles ou *dictionnaire d'entrée* (ou *événement universel*).

Il est important de souligner que l'expression régulière que représente un événement dans un automate n'est pas unique.

Deux expressions régulières P et Q sont équivalentes, $P = Q$ à la condition nécessaire et suffisante qu'elles représentent le même ensemble de séquences. Une expression régulière P est contenue dans une autre Q, $P < Q$ à la condition nécessaire et suffisante que l'ensemble représenté par P soit un sous-ensemble de celui représenté par Q.

(1) On trouve les deux premières notations dans la littérature russe et la troisième dans la littérature anglo-saxonne.

Les expressions suivantes sont équivalentes :

$P^{**} = P^*$	parce que	$\bar{P}^{**} = P^*$
$P^* = \lambda + PP^*$	—	
$PP^* = P^*P$	—	
$P^*P^* = P^*$	—	
$P^* + P = P^*$	—	

On voit donc que toutes les expressions régulières itérées sont équivalentes à P^* ou PP^*.

Nous proposons d'écrire autrement l'expression PP^*; les raisons pour le faire seront exposées plus bas.

Nous écrivons $P^{1*} = PP^*$
$P^{*1} = P^*P$

L'expression $P_1{}^*$ représente un événement cyclique.

2 - MÉTHODE DE SYNTHÈSE DE GLOUSHKOV ([2], [3], [4], [9])

La spécification d'une transformation par un automate entre les mots ou séquences formés dans les alphabets X et Z est équivalente à la spécification des événements $R(Z_1)$, $R(Z_2)$, ... $R(Z_n)$ représentés dans la dite transformation par les signaux de sortie $Z_1, Z_2, \ldots Z_n$.

Gloushkov a mis au point un procédé systématique de synthèse à partir des expressions régulières dont nous allons exposer ci-dessous les grandes lignes ([2], [4]).

2.1. Principe de la méthode

Première étape. On identifie à l'aide des nombres naturels tous les endroits qu'il y a avant et après tous les symboles de chaque expression régulière en appliquant des règles qu'on trouvera un peu plus loin. On appelle cette opération *indexation*.

Deuxième étape. On construit une *table de succession des endroits* (1). Les colonnes de cette table sont repérées par les lettres de l'alphabet X et les lignes par des ensembles d'indices obtenus précédemment. Au croisement de la colonne X_j et de la ligne correspondante à l'ensemble (p, r, w ...) on écrit l'ensemble d'indices qui suivent X_j dans les expressions régulières lorsque X_j est précédée par p ou r ou w ou ....

Troisième étape. Afin de passer à la table de fluence (type Moore) du système, on définit comme état interne de la machine chaque sous-ensemble de l'ensemble total des indices formé dans le stade antérieur. A chaque état présent q_s on associe l'ensemble des lettres de sortie pour lesquelles au moins un des indices compris dans q_s se trouve dans l'ensemble d'indices de l'endroit final de l'expression régulière correspondante.

2.2. Règles d'indexation de Gloushkov

Sans perte de généralité, on peut supposer $R(Z_j)$, $\forall Z_j \in Z$ entre des parenthèses ordinaires. On appelle

(1) Appelée aussi table par classes d'indices ou table ébauche.

endroit initial (noté 0) l'endroit qui précède la parenthèse, *endroit final* celui qui suit cette parenthèse, *endroits antéfondamentaux* et *fondamentaux* ceux qui précèdent et suivent respectivement une lettre X dans l'expression régulière.

Les endroits initiaux de toutes les expressions $R(Z_j)$ sont les mêmes. Tous les endroits fondamentaux sont indicés avec des nombres naturels différents dont l'effet s'étend à d'autres endroits, suivant l'ordre des actions de l'algèbre des expressions régulières [3].

2.2.1. Règles d'extension des indices

R.1. Les indices des endroits précédant une parenthèse quelconque (ordinaire ou d'itération) s'étendent à tous les endroits initiaux de cette parenthèse.

R.2. Les indices des endroits finals d'une parenthèse quelconque s'étendent à l'endroit qui suit immédiatement cette parenthèse.

R.3. Les indices des endroits qui précèdent immédiatement une parenthèse d'itération (ou le mot vide λ) s'étendent à tous les endroits qui suivent immédiatement cette parenthèse (ou le mot vide).

R.4. Les indices des endroits qui suivent immédiatement une parenthèse d'itération s'étendent à tous les endroits qui précèdent cette parenthèse.

2.2.2. Règles d'identification des endroits

Ces quatre règles sont d'un emploi facile et sans équivoque. Or, dans la pratique, il est intéressant d'affecter à certains endroits fondamentaux les mêmes indices. Cette identification peut se faire dans les deux cas suivants. On identifie :

I.1. Les endroits qui sont subordonnés à des ensembles identiques d'indices antéfondamentaux et finals.

I.2. Les endroits des différentes expressions $R(Z_1)$, $R(Z_2)$ ... ou des différents termes appartenant à la même paire de parenthèses auxquels amènent des chemins identiques (ensembles de mots) en partant de l'endroit initial ou de l'endroit qui précède la parenthèse.

Remarque 1

Les règles R.1 et R.4 sont systématiques et programmables sur calculateur. Les règles I.1 et I.2 ne sont pas systématiques mais au contraire nécessitent, dans un cas général, un certain degré de raisonnement de la part de l'ingénieur.

Exemple :

$$R(Z_1) = \underset{0}{|} X_1 \underset{1}{|} X_2 \underset{2}{|} + \underset{0}{|} X_1 \underset{1}{|} X_2 \underset{2}{|} X_2 \underset{4}{|} X_1 \underset{5}{|}$$

$$+ \underset{0}{|} X_2 \underset{6}{|} X_3 \underset{7}{|} X_2 \underset{8}{|} X_1 \underset{5}{|}$$

Par application de la règle I.1. le même indice est associé aux deux endroits représentés par 5.

Par application de la règle I.2. on associe les mêmes indices aux paires représentées par 1 et 2.

Remarque 2

On peut substituer aux règles I.1 et I.2 une transposition directe du principe d'équivalence d'états (I.1.4), où q_i et q_j seraient des ensembles d'indices. Il faut souligner la simplicité d'application de ce principe du fait que ces états sont chiffrés.

Donc, dans l'exemple ci-dessus on peut prendre :

$$4 \simeq 8$$

et :

$$R(Z_1) = \underset{0}{|} X_1 \underset{1}{|} X_2 \underset{2}{|} + \underset{0}{|} X_1 \underset{1}{|} X_2 \underset{2}{|} X_2 \underset{4}{|} X_1 \underset{5}{|}$$

$$+ \underset{0}{|} X_2 \underset{6}{|} X_3 \underset{7}{|} 2_2 \underset{4}{|} X_1 \underset{5}{|}$$

Parmi les méthodes de synthèse qui travaillent à partir de l'algèbre des expressions régulières, celle de Gloushkov (qui est programmable sur calculateur numérique) nous paraît assez bonne, même appliquée à des cas réels avec des alphabets d'entrée et de sortie à un nombre important de lettres. Cependant, on peut essayer, dans le cas particulier où l'on a à traiter des expressions régulières comportant des itérations, de la simplifier quelque peu. Ces simplifications font l'objet du paragraphe qui suit.

3 - NOUVEL ALGORITHME POUR LES EXPRESSIONS 1* - ITÉRÉES [9]

Les idées que nous développons dans cette section optimisent la méthode de Gloushkov au point de vue temps, dimension et efficacité des résultats directs. Elles sont fondamentalement basées sur le principe d'équivalence d'états (I.1.4). L'introduction d'une nouvelle écriture pour les événements cycliques (cf. § I.2.3.), nous permet d'en profiter au maximum. Le cheminement de notre exposé sera le suivant :

— on considère un seul événement régulier. Les résultats sont généralisables à plusieurs (1).
— recherche des états équivalents (nous ne faisons pour le moment aucune hypothèse sur les expressions P et F).
— représentation graphique.
— conséquences et propositions au niveau de l'indexation et de passage à la table par classes d'indices.

3.1. Evénement $R = i^* F$

Théorème : Les différents ensembles d'indices à l'intérieur de l'expression i* représentent des états équivalents (Démonstration : par application du théorème (I.1.4)).

On peut donc écrire :

$$R = \underset{q_1}{|} i^* \underset{q_1}{|} F \quad (q_1 \text{ ensemble d'indices} \simeq \text{état})$$

(1) Sauf en ce qui concerne l'indexation en sens inverse (cf. ci-dessous § 3-5, Remarque 2).

(voir fig. 1.a). Ceci se traduit, au niveau de l'algorithme de synthèse par la proposition suivante.

Proposition 1. *Pour l'indexation* : Faire $q_i = 1$, par exemple.

Pour le passage à la table : Seul l'état q_1 est représenté par un ensemble d'un seul élément : l'indice 1. Donc, écrire un 1 dans toutes les cases qui ne correspondant pas aux séquences

$$\{ X : \bar{X} \leqslant \bar{X}' \in \bar{F} \}.$$

Tous les autres états sont des ensembles d'indices dont un élément est toujours le 1 et dont le reste des éléments est obtenu par application de l'algorithme de Gloushkov à l'expression F.

3.2. Evénement R = P + i* F

C'est la même chose que le cas précédent, à la différence près qu'il faut prévoir un état supplémentaire pour tenir compte aussi bien d'un événement initial (représenté par P) que d'un événement non initial (i*F).

On pourra écrire :

$$R = \underset{q_0}{|\ P\ |} + \underset{q_1}{|\ i^*\ |}\ \underset{q_1}{|\ F\ |}$$

L'interprétation de cette indexation est un peu spécieuse (fig. 1.b), voire arbitraire, puisque, en fait, l'indexation ci-dessus n'est valable qu'à partir de l'instant t = 1. On en déduit la proposition 2.

Proposition 2. *Pour l'indexation* : : faire $q_0 = 0$ et $q_1 = 1$. L'assignation d'indices pour P et F est tout à fait normale.

Pour le passage à la table : seuls les états $q_0 = 0$ et $q_1 = 1$ sont constitués par un seul indice. Tous les autres états sont des ensembles d'indices dont un élément est toujours le 1 et dont le reste des éléments est obtenu par application de l'algorithme de Gloushkov aux expressions P et F. En particulier, les éléments des q_i sont obtenus par application de l'algorithme de Gloushkov en supposant pendant le premier instant l'événement universel indexé avec un 0.

Voyons à l'aide d'un exemple simple comment seraient remplies les deux premières lignes :

Soit l'expression : $R_1 = X_1 X_2 + i^* (X_1 X_3 + X_2 X_1)$ définie sur les alphabets

$$x = \{ X_1 ; X_2 ; X_3 \}$$
$$Z = \{ Z_1 ; Z_2 \}$$

L'indexation conduit à :

$$_1 = \underset{0}{|\ X_1\ |}\ \underset{2}{|\ X_2\ |} + \underset{3}{|\ i^*\ |}\ (\ \underset{1}{|\ X_1\ |}\ \underset{4}{|\ X_2\ |} + \underset{1}{|\ X_2\ |}\ \underset{5}{|\ X_1\ |}\ )$$

La table est celle de la figure 2 a.

Et le diagramme celui de la figure 2 b (comparer avec le diagramme de la fig. 1 b).

Dans le paragraphe 3.1 nous avons vu que l'événement universel (vis-à-vis du comportement d'une machine séquentielle) peut être représenté par un seul état où la machine laisse « passer le temps » tant qu'elle ne reçoit pas l'une des séquences à reconnaître.

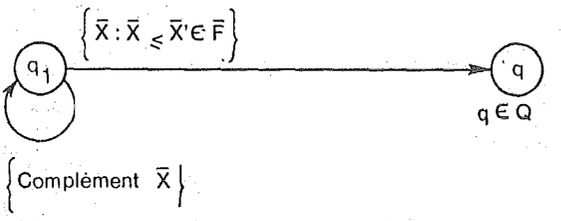

Fig. 1a — Diagramme de fluence partiel pour R = i*F

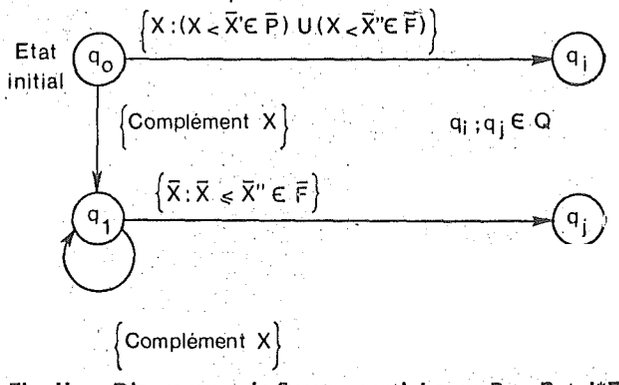

Fig. 1b — Diagramme de fluence partiel pour R = P + i*F

Fig. 2a

	X_1	X_2	X_3
0	1, 2, 4	1, 5	1
1	1, 4	1, 5	1

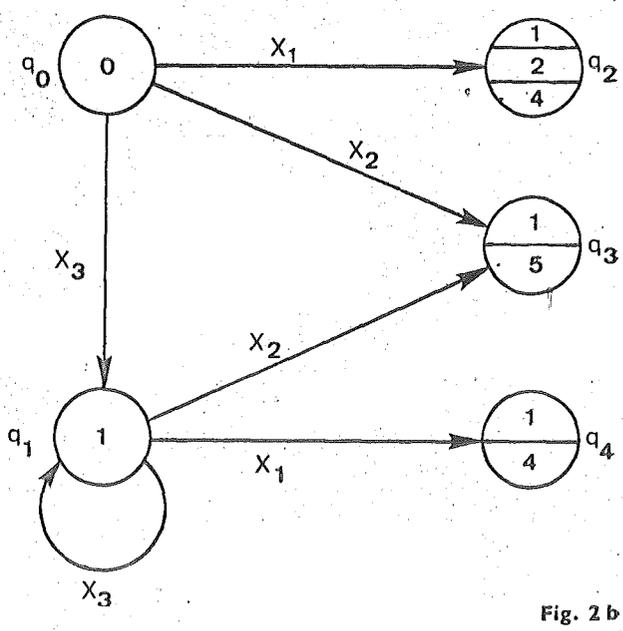

Fig. 2b

On a pu caractériser i* par un seul indice. D'où la proposition suivante :

Proposition 3. L'expression x*, $\forall x \in X$, peut être caractérisée par un seul indice ($\underset{\ }{|\ x^*\ |}$), à l'intérieur de toute expression régulière. Aucun algorithme spécial pour le passage à la table.

N.B. La proposition 3 est un corollaire de la proposition 2.

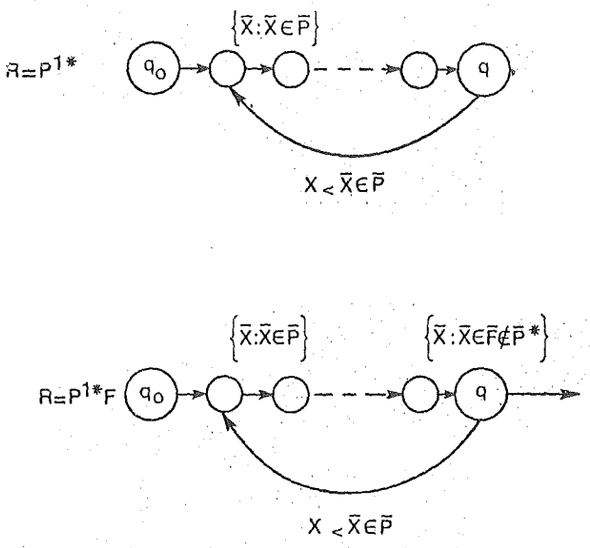

Fig. 3 — Diagrammes de fluence partiels pour R = P1* et R = P1* F

3.3. Evénement R = Pi*

Nous allons démontrer qu'il existe un algorithme d'indexation sur P1* qui est équivalent à l'algorithme de Gloushkov sur P < P > et plus souple que celui-ci.

Supposons indexée par la méthode de Gloushkov une expression de la forme :

$$R = \underset{\alpha_i}{|} \underset{\alpha_p}{P} | < P > \underset{\alpha''_p}{|}$$

(nous considérons cette expression isolée, mais en général elle fait partie d'une autre plus compliquée).

Par les définitions mêmes d'expression itérée et d'équivalence des états, ceux qui sont représentés par les ensembles d'indices α_p et α''_p sont équivalents. Donc $\alpha''_p = \alpha_p$. Mais, d'après les règles d'indexation, $\alpha''_p = \{\alpha'_p, \alpha_p\}$. Donc $\alpha'_p = \alpha_p$.

Ceci étant, par des applications successives du théorème d'équivalence, on démontre que les états de la machine pour des séquences $\bar{X}'$ représentées dans < P > sont équivalents aux états pour $\bar{X}'$ représentées dans P, $\forall X' \leqslant \bar{X} \in P$.

Ceci est symbolisé par le diagramme de fluence de la figure 3, où, par souci de simplicité, on a considéré une expression P constituée d'une seule séquence. Remarquons que nous avons déjà écrit P1* sur la figure. C'est parce que, même en indexant, on pourra toujours l'écrire comme ceci.

D'après les propriétés que nous venons de voir nous croyons pouvoir faire la proposition suivante :

Proposition 4. On écrit P1* au lieu de P < P > ou P P* ou P { P }. On indexe P1* comme s'il s'agissait de P. On boucle en étendant l'ensemble des indices finals de P1* aux endroits initiaux. La méthode de passage à la table reste la même.

3.4. Variables asynchrones

Les variables asynchrones peuvent être représentées par des expressions XX* [ou X < X > ou X { X }]. (Cf. [4] et [10]).

Considérées comme faisant partie d'une séquence et manipulées par un algorithme du type Gloushkov, on obtiendrait des ensembles d'indices pour tous les endroits verticaux de l'expression :

$$\underset{i}{|} \underset{i+1}{X} \underset{i+1}{|} < \underset{i+1}{|} \underset{i+2}{X} \underset{i+2}{|} > \underset{i+1}{|}$$

Or X < X > est un cas particulier de P < P >.

Donc :

Proposition 5. Nous voyons deux possibilités :

Possibilité 5.1 On écrit X1* et on indexe $\underset{i}{|} \underset{i+1}{X^{1*}} |$

Pour le passage à la table, il faut tenir compte de la signification implicite symbolisée par le diagramme de la figure 4.

Possibilité 5.2. On écrit X1* et on indexe $\underset{i}{|} \underset{i+1}{X^{1*}} \underset{i+1}{|}$

Dans ce cas, le passage à la table reste tel que le préconise Gloushkov.

La proposition 5.1. est avantageuse en ce sens que l'indexation d'une séquence de variables asynchrones devient la même chose que l'indexation d'une séquence de variables synchrones. Ceci est vrai à un niveau plus élevé pour deux expressions régulières formellement identiques, l'une pour des variables asynchrones, l'autre pour des variables synchrones. En plus nous nous trouvons avec un seul indice par endroit ce qui est très agréable en cas de programmation sur calculateur numérique. Personnellement nous retenons cette possibilité.

Par contre, la possibilité 5.2. est cohérente avec la proposition 4. Or, au point de vue pratique, ceci n'est pas tellement important car de toutes façons il y aura toujours une différence même symbolique entre une variable 1*-itérée et une expression 1*-itérée (celle-ci sera repérée par ( )1*).

Remarque : Il est évident que toutes les expressions sions P et F que nous avons considérées jusqu'ici peuvent représenter des ensembles de séquences à variables asynchrones. Supposons, par exemple, que ce soit le cas dans l'expression R = P1* F de la figure 3. Le graphe correspondant devient celui de la figure 5.

3.5. Indexation en sens inverse

Soient A, B, C, ... etc., des expressions faisant partie d'une expression régulière R et séparées à l'intérieur de celle-ci par le symbole de disjonction (+). Soient A = A'a, B = B'b, C = C'c, ... ; A', B', C' ... etc., étant des expressions qui représentent n'importe quels ensembles de séquences finis ou infinis (λ à la limite); a, b, c, ... représentant chacune une séquence finie (λ à la limite).

Considérons un automate A décrit par l'expression R. On va réaliser l'opération suivante : prenons deux à deux les expressions a, b, c, ... ; b et c par exemple. Supposons $b \neq \lambda$ et $c \neq \lambda$. Ceci implique que $b = b'X_b$; $c = c'X_c$. Soit q_b, l'état de l'automate lorsqu'il a reçu

une séquence de l'ensemble représenté par l'expression B' concatenée par b'. Soit q_c, l'état atteint par action de l'expression C' concatenée par c'.

Si $X_b = X_c$, on peut identifier les endroits fondamentaux qui suivent X_c. Si, on a aussi $b' = b''X'_b$ et $c' = c''X'_c$ avec $X'_b = X'_c$, on peut encore faire la même opération.

Maintenant, plaçons-nous dans la situation inverse : nous voulons faire la synthèse d'un automate qui réalise l'événement décrit par R et nous le désirons muni d'un ensemble d'états le plus réduit possible. D'après ce qui vient d'être dit, énonçons une nouvelle proposition :

Proposition 6. Pour toutes les expressions A, B, C... dans R avec les conditions exposées dans les lignes ci-dessus chercher systématiquement de la droite vers la gauche (en commençant par l'endroit final auquel on associe un même indice), en comparant lettre par lettre, les endroits (états) équivalents, auxquels on peut associer les mêmes indices.

Si les séquences a, b, c, ... sont constituées de variables asynchrones X^{1*} auxquelles on applique la proposition 5.1. seulement on peut associer les mêmes indices aux endroits situés immédiatement à droite des lettres identiques.

Remarques. 1. La proposition 6 n'est qu'une application pratique généralisée à plusieurs expressions A, B, C, ... de la propriété de l'algèbre des expressions régulières $A + B = PS + QS = (P + Q)S$ avec S représentant dans ce cas une séquence finie.

Exemple :

$$R = \underset{i}{P} \mid \underset{i}{(} \mid \underset{j}{X_2} \mid \underset{k}{X_3} \mid + \mid \underset{i}{X_3} \mid \underset{l}{X_1} \mid \underset{n}{X_3} \mid \underset{k}{)} \mid \underset{k}{F}$$

2. Lorsque l'on a affaire à plusieurs expressions régulières simultanées, il importe de vérifier que cette façon de procéder n'introduit pas d'ambiguïté sur les sorties dans le cas où l'on indexe simultanément plusieurs expressions régulières.

3.6. Conclusion

Dans le cadre de la synthèse des automates à partir des expressions régulières, on a fait une recherche des états équivalents pour les automates décrits par une de ces expressions. Les résultats de cette recherche nous ont permis d'établir quelques propositions qui allègent l'écriture et le traitement au moyen de l'algorithme de Gloushkov d'une classe assez large des événements réguliers, surtout des événements cycliques et en particulier les machines asynchrones.

Toutes les propositions sont algorithmiques et systématiques, donc programmables sur calculateur. Leur conséquence immédiate est la construction de tables d'état très réduites, très souvent minimales (1).

(1) Une vérification simple consiste à appliquer ce qui vient d'être dit à un certain nombre de cas qui dans la littérature sont résolus au moyen d'un algorithme de dérivées (par exemple, dans Harrizon [5], pp. 328-330, dans Kuznetsov [10], pp. 1077-1078) ou même par la méthode originelle de Gloushkov (Perrin et al. [4], t. 2, pp. 40-43, 47-52).

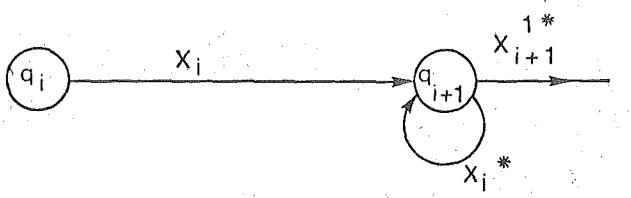

Fig. 4 — Diagramme de fluence (passage de q_1 à q_{i+1} par application d'une variable asynchrone x_i)

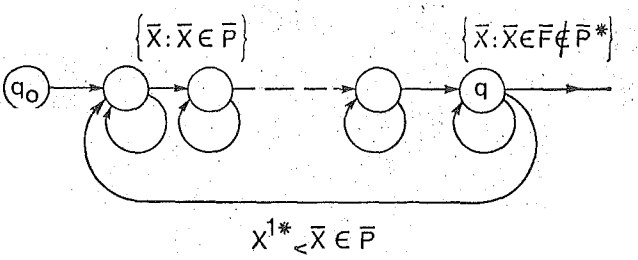

Fig. 5 — Diagramme de fluence de $R = P^{1*}F$ avec P et F constitué par des variables asynchrones

4 - EXEMPLES

Nous avons considéré [9] la synthèse automatique de la classe de machines séquentielles type « machine transfert », représentée par l'expression régulière :

$$R = i^* P = i^* \sum_{j=1}^{k} P_j$$

où P_j signifie une séquence finie de variables asynchrones. L'alphabet d'entrée, la longueur et le nombre des séquences sont quelconques. L'alphabet de sortie est à deux éléments, oui ou non. L'expression R ci-dessus est définie pour la sortie oui (ou 1).

La construction d'une table d'états est faite en appliquant le procédé de Gloushkov, amélioré par les propositions 1 pour i* et la proposition 5.1. pour p_j. Le procédé a été programmé en langage Fortran pour I.B.M. 360/44 au C.E.R.A.

4.1. Exemple 1

Construire la table d'états d'une machine abstraite asynchrone avec un alphabet d'entrée $\{X_1 ; X_2 ; X_3\}$, un alphabet de sortie $\{Z_1 , Z_2\}$ et dont le comportement soit décrit par l'expression régulière :

$$R_1 = i^* (X_3^{1*} X_1^{1*} X_2^{1*} X_1^{1*} + X_3^{1*} X_2^{1*} X_3^{1*} X_2^{1*} X_1^{1})$$

Pour l'indexation on associe un 1 aux premier et deuxième endroits à cause de la proposition 1, un 2 au quatrième et neuvième endroits à cause de 12 (Gloushkov), un 5 aux treizième et septième endroits à cause de la proposition 5.1. et 11 (Gloushkov).

Indexation :

$$R_1 = \underset{1}{|} i \underset{1}{|} \underset{1}{(} \mid \underset{2}{X_3^{1*}} \mid \underset{3}{X_1^{1*}} \mid \underset{4}{X_2^{1*}} \mid \underset{5}{X_1^{1*}} \mid$$

$$+ \mid \underset{1}{X_3^{1*}} \mid \underset{2}{X_2^{1*}} \mid \underset{6}{X_3^{1*}} \mid \underset{7}{X_2^{2}} \mid \underset{4}{X_1^{1*}} \mid \underset{5}{)}$$

Par application directe, on obtient la table par classes d'indices (ou table ébauche) de la figure 6.

	X_1	X_2	X_3	
1	1	1	1,2	
1,2	1,3	1,6	1,2	
1,3	1,3	1,4	1,2	
1,6	1	1,6	1,2,7	
1,4	1,5	1,4	1,2	
1,2,7	1,3	1,6,4	1,2,7	
1,5	1,5	1	1,2	R_1
1,6,4	1,5	1,6,4	1,2,7	

Fig. 6 — Table par classes d'indices

On passe à la table de fluence très facilement par exemple en posant :

1 → a	1,5 → e
1,2 → b	1,6 → f
1,3 → c	1,2,7 → g
1,4 → d	1,6,4 → h

(Cf. Annexe 2 où l'on peut trouver le même exemple résolu par la méthode originale de Gloushkov)

4.2. Exemple 2

Nous extrayons cet exemple du livre de M. Cl. Polgar [11]. L'énoncé du problème est le suivant. Un moteur Z entraîne un plateau P qu'il fait tourner. Il porte sur son arbre une came, qui définit la position de repos du système lorsqu'elle touche une fin de course C (on a alors C = 1). D'autre part, un moteur synchrone X entraîne, par l'intermédiaire d'un réducteur, une came qui agit sur une autre fin de course A. Ce deuxième ensemble jour le rôle d'horloge (ou minuterie) (fig. 7). Construire le système logique commandant l'ensemble du système de sorte que le plateau P asse un tour et un seul chaque fois que la minuterie établit le contact A.

On a visiblement ici un problème avec cycle puisque chaque passage en A de la minuterie entraîne la même séquence d'ordres. La conception du système logique de commande va donc être la suivante :

— établissement des séquences d'entrée-sortie possibles du système correspondant à un cycle (ce qui suppose le choix d'un état initial).
— passage aux expressions régulières.
— indexation de celles-ci.
— table de fluence.
— table des phases, réduction, codage.

4.2.1. Etablissement des séquences d'entrée

Nous pouvons prendre a priori comme état initial n'importe lequel des états stables du cycle. Prenons, par exemple, celui où le système étant au repos, il vient de recevoir un ordre A, qui a pour effet de mettre immédiatement Z en marche. La sortie ne pouvant prendre que les valeurs 0 ou 1, nous allons donc chercher l'expression régulière R (Z) correspondant à l'extinction de Z à la fin du cycle. Les combinaisons 00, 01, 10 et 11 des entrées C et A (dans l'ordre) seront repérées par X_0, X_1, X_2, X_3 (afin de simplifier les notations).

1 - Séquence d'entrée 1 (S_1) :

X_1 (rupture de C avant rupture de A)	Z = 1
puis X_0 (rupture de A)	Z = 1
puis X_2 (établissement de C)	Z = 0
puis X_3 (établissement de A)	Z = 1

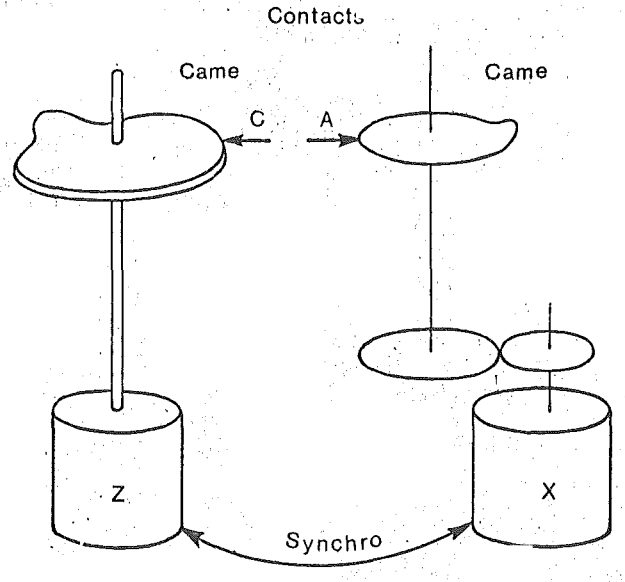

Fig. 7

2 - Séquence d'entrée 2 (S_2) :

X_2 (rupture de A avant rupture de C)	Z = 1
puis X_0 (rupture de C)	Z = 1
puis X_2 (établissement de C)	Z = 0
puis X_3 (établissement de A)	Z = 1

3 - Séquence d'entrée 3 (S_3)

X_1 (rupture de C avant rupture de A)	Z = 1
puis X_3 (établissement de C avant rupture de A)	Z = 0
puis X_2 (rupture de A)	Z = 0
puis X_3 (établissement de A)	Z = 1

4.2.2. Calcul des expressions régulières

Le lecteur pourra vérifier que Z et $\overline{Z}$ étant complémentaires, on peut se contenter d'une seule expression régulière pour définir-notre système ([2], [3], [4]). Alors S_1, S_2, S_3 engendrent chacune une expression régulière ($R_1(\overline{Z})$, $R_2(\overline{Z})$, $R_3(\overline{Z})$). On aura alors :

$$R(\overline{Z}) = R_1(\overline{Z}) + R_2(\overline{Z}) + R_3(\overline{Z}),$$

avec $R_1(\overline{Z}) = i^* X_1{}^{1*} X_0{}^{1*} X_2{}^{1*}$

$R_2(\overline{Z}) = i^* X_2{}^{1*} X_0{}^{1*} X_2{}^{1*}$

$R_3(\overline{Z}) = i^* (X_1{}^{1*} X_3{}^{1*} + X_1{}^{1*} X_3{}^{1*} X_2{}^{1*})$

D'où $R(\overline{Z}) = i^* (X_1{}^{1*} X_0{}^{1*} X_2{}^{1*} + X_2{}^{1*} X_0{}^{1*} X_2{}^{1*} + X_1{}^{1*} X_3{}^{1*} + X_1{}^{1*} X_3{}^{1*} X_2{}^{1*})$

4.2.3. Indexation

Pour alléger l'écriture et puisqu'aucune confusion n'est possible, nous écrivons $X_i{}^{1*} = X_i$.

L'expression R $(\overline{Z})$ devient alors, une fois indexée par application des règles énoncées plus haut :

$$R(Z) = |\,I^*\,|\,(\,|\,X_1\,|\,X_0\,|\,X_2\,|\,+\,|\,X_2\,|\,X_0\,|\,X_2\,|$$
$$1\quad 1\;1\quad 2\quad 3\quad 6\quad 1\quad 4\quad 3\quad 6$$

$$+\,|\,X_1\,|\,X_3\,|\,+\,|\,X_1\,|\,X_3\,|\,X_2\,|\,)\,|$$
$$1\quad 2\quad 5\quad 1\quad 2\quad 5\quad 6\;5$$
$$6$$

4.2.4. *Passage à la table des phases*

Le lecteur vérifiera que la table ébauche correspondante est celle de la figure 8 a qui donne la table des phases de la figure 8 b.

	X_0	X_1	X_3	X_2	Z
1	1	1,2	1	1,4	1
1,2	1,3	1,2	1,2	1,4	1
1,4	1,3	1,2	1	1,4	1
1,3	1,3	1,2	1	1,4,6	1
1,5	1	1,2	1,5	1,4,6	0
1,4,6	1,3	1,2	1	1,4,6	0

Fig. 8 a

CA \	00	01	11	10	Z
	1	2	①	3	1
	4	②	5	3	
	4	2	1	③	
	④	2	1	6	1
	1	2	⑤	6	0
	4	2	1	⑥	0

Fig. 8 b

Mais la table primitive de la figure 8 b n'est pas la plus simple possible. En effet, tous les états instables j situés dans la ligne où se trouve un état stable i n'ont d'existence réelle que si le changement de commande pour passer de la colonne où se trouve l'état stable à celle où se trouve l'état instable n'est provoquée que par le changement de valeur que d'une seule commande (afin d'éviter des courses sur les variables primaires). En appliquant ce raisonnement, on aboutit à la table primitive de la figure 9 qui se réduit en la table des phases de la figure 10 a et la table des sorties de la figure 10 b (il y a, en fait, d'autres réductions possibles, bien entendu).

CA	00	01	11	10	Z
	—	2	①	3	1
	4	②	5	—	1
	4	—	1	③	1
	④	2	—	6	1
	—	2	5	6	0
	4	—	1	⑥	0

Fig. 9

CA	00	01	11	10
	4	2	①	③
	4	②	⑤	6
	④	2	1	⑥

Fig. 10 a

CA	00	01	11	10
	—	—	1	1
	—	1	0	—
	1	—	—	0

Fig. 10 b

Remarque : Cette simplification de la table par création de vides dus à des transitions impossibles peut être encore poussée plus loin. Par exemple, puisque Z = 0, la transition ⑥ → 4 → ④ est physiquement impossible (le contact C ne peut être relâché lorsque le moteur Z est arrêté). Il en est de même pour la transition ⑤ → 6 → ⑥.

4.2.3. *Codage*

On voit que, bien que chacune des lignes de la table doive être adjacente au deux autres, la présence d'un seul état stable dans les colonnes 00 et 01 permet d'éviter les phénomènes de course. Le codage se fait donc de façon très simple. On laisse au lecteur le soin de le faire et d'obtenir les circuits logiques associés.

ANNEXE

Voici pour comparaison la synthèse d'une table d'états correspondante à un automate décrit par l'expression régulière de l'exemple proposé à la section IV, traités avec la notation et l'algorithme originaux de Gloushkov.

$$R_1 = |<|X_1|+|X_2|+|X_3|<|(|X_3|<|X_3|$$
$$0\;0\;1\;0\;2\;0\;3\;0\;0\;4\;4\;5$$
$$1\quad 1\quad 1\quad 1\;1\quad 5$$
$$2\quad 2\quad 2\quad 2\;2$$
$$3\quad 3\quad 3\quad 3\;3$$

$$<|X_1|<|X_1|<|X_2|<|X_2|<|X_1|<|X_1|<|$$
$$4\;6\;6\;7\;6\;8\;8\;9\;8\;10\;10\;11\;10$$
$$5\quad 7\quad 7\quad 9\quad 9\quad 11\quad 11$$

$$+|X_3|<|X_3|<|X_2|<|X_2|<|X_3|<|X_3|$$
$$0\;4\;4\;5\;4\;12\;12\;13\;12\;14\;14\;15$$
$$1\quad 5\quad 5\quad 13\quad 13\quad 15$$
$$2$$
$$3$$

$$<|X_2|<|X_2|<|X_1|<|X_1|<|)|$$
$$14\;16\;16\;17\;16\;18\;18\;19\;18\;10$$
$$15\quad 17\quad 17\quad 19\quad 19\;11$$
$$18$$
$$19$$

et la table de la figure 11.

	X_1	X_2	X_3	
0	1	2	3,4	
1	1	2	3,4	
2	1	2	3,4	
3,4	1,6	2,12	3,4,5	
3,4,5	1,6	2,12	3,4,5	
2,12	1	2,13	3,4,14	
1,6	1,7	2,8	3,4	
2,13	1	2,13	3,4,14	
3,4,14	1,6	2,12,16	3,4,5,15	
1,7	1,7	2,8	3,4	
2,8	1,10	2,9	3,4	
2,12,16	1,18	2,13,17	3,4,14	
3,4,5;15	1,6	2,12,16	3,4,5;15	
1,10	1,11	2	3,4	R_1
2,9	1,10	2,9	3,4	
2,13,17	1,18	2,13,17	3,4,14	
1,18	1,19	2	3,4	R_1
1,11	1,11	2	3,4	R_1
1,19	1,19	2	3,4	R_1

Fig. 11

Par simple examen de la table on voit les équivalences suivantes :

$0 \simeq 1 \simeq 2 \rightarrow a$
$3, 4 \simeq 3, 4, 5 \rightarrow b$
$1, 6 \simeq 1, 7 \rightarrow c$
$2, 8 \simeq 2, 9 \rightarrow d$
$1, 10 \simeq 1, 11 \rightarrow e$ (associé à la sortie)
$2, 12 \simeq 2, 13 \rightarrow f$
$3, 4, 14 \simeq 3, 4, 5, 15 \rightarrow g$
$2, 12, 16 \simeq 2, 13, 17 \rightarrow h$
$1, 18 \simeq 1, 19 \rightarrow i$ (associé à la sortie)

Ce qui nous amène à la table plus réduite de la figure 12.

	X_1	X_2	X_3	
a	a	a	b	
b	c	f	b	
c	c	d	g	
d	e	d	b	
e	e	a	b	R_1
f	a	f	g	
g	c	h	g	
h	i	h	g	
i	i	a	b	R_i

Figure 12

Finalement on remarque que les états e et i sont équivalents. Par conséquent on pourra écrire :

$$e \simeq i \rightarrow e$$

Donc la table finale est celle de la figure 13.

	X_1	X_2	X_3	
a	a	a	b	
b	c	f	b	
c	c	d	g	
d	e	d	b	
e	e	a	b	R_1
f	a	f	g	
g	c	h	g	
h	e	h	g	

Fig. 13

qui est bien la même table que celle obtenue directement à la section 4.

BIBLIOGRAPHIE

[1] S.C. KLEENE : « Representation of Events in Nerve Nets and Finite Automata ». *Automata Studies*. C.E. Shannon, J. McCarthy Eds. Princeton University Press, pp. 3-41, 1956.

[2] V.M. GLOUSHKOV : *Synthèse des automates digitaux*. Physmathgiz, Moscou, 1962 (en russe).

[3] V.M. GLOUSHKOV : *Introduction à la cybernétique*. Editions de l'Académie des Sciences de la R.S.S. d'Ukraine, Kiev, 1964 (en russe). Traduction anglaise : « Introduction to cybernetics » Academic Press, 1966, pp. 91-116.

[4] J.P. PERRIN, M. DENOUETTE, E. DACLIN : *Systèmes logiques*. Dunod, 1967, pp. 28-53, tome 2.

[5] M.A. HARRISON : *Introduction to Switching and Automata Theory*. McGraw Hill, 1965, pp. 321-324.

[6] J.A. BRZOZOWSKI : « Canonical Regular Expressions and Minimal State graphs for definite events ». *Proceedings of the Symposium on MathematicalTheory of Automata*. Polytechnic Press of the P.I.B., 1963.

[7] E.F. MOORE : « Gedanken - experiments on sequential machines ». Automata Studies, p. 136.

[8] L.A. ZADEH, C.A. DESOER : *Linear System Theory : The State Space Approach*. McGraw Hill, 1963, pp. 70-71.

[9] F. SAEZ VACAS : *Programmation de l'algorithme de synthèse de Gloushkovpour les tables de fluence des systèmes séquentiels*. Thèse à l'Ecole Nationale Supérieure de l'Aéronautique, Paris, août 1966.

[10] O.P. KUZNETSOV : « Representation of Regular Events in Asynchronous Automate ». *Automation and Remote Control* Vol. 26, N° 6, 1965.

[11] Cl. POLGAR : *Techniques de l'emploi des relais dans les machines automatiques*., pp. 229-237. Eyrolles, Paris 1961.