UNIVERSITÉ D'ALGER

ÉCOLE NATIONALE POLYTECHNIQUE

DÉPARTEMENT ÉLECTRICITÉ

PROJET DE FIN D'ÉTUDES

SYNTHESE DES MACHINES SEQUENTIELLES SYNCHRONES PAR DES METHODES INFORMATIQUES

Propose par:
Mr C-Corruble

Etudie par
Mrs MADANI
YAKER

Juin 1975

SYNTHESE DES MACHINES SEQUENTIELLES SYNCHRONES PAR DES METHODES INFORMATIQUES

Propose par:

Mr C. Corruble

Etudie par

Mrs MADANI

YAKER

Juin 1975

ORDINATEUR UTILISE : IRIS 45 (C.I.I) DU CENTRE DE CHEQUES POSTAUX D'ALGER

LANGAGE :FORTRAN IV.

Nous prions les responsables du centre de calcul des cheques postaux, et particulièrement Mr.SID AHMED ZIAM d'accepter nos vifs remerciements pour toute l'aide qu'ils nous ont fournie.

X

PLAN D'ETU DE

IN TRODUCTION.

CHAPITRE 1.

DEFIN ITION S ET CARACTERISTIQUES PRINCIPALES DES SYSTEMES SEQUENTIELS SYNCHRONES.

CHAPITRE 2.

SYNTHESE DES TABLES : ETABLISSEMENT DE LA TABLE DES PHASES.

CHAPITRE 3.

REDUCTION DES TABLES.

3.1. RECHERCHE DES PAIRES COMPATIBLES.

3.2.RECHERCHE DES COMPATIBLES MAXIMAUX.

3.3. RECHERCH E D'UNE COUVERTURE MINIMALE ET FIN IE.

CHAPITRE 4.

ASSIGNEMENT (CODAGE) DES ETATS.

CONCLUSION.

IN TRODUCTION.

Le but de notre étude est de cerner le problème de l'automatisation du calcul des machines séquentielles.
Notre travail s'est limité à la recherche de méthodes de calcul facilement programmables, dans le cas des machines séquentielles synchrones.
En partant de la table des charges dela machine qu'on désire réaliser nous avons établi une série de programmes en langage FORTRAN qui nous permettent d'aboutir aux ensembles d'équations de sortie.
Dans la succession de ces différentes étapes nous nous sommes éfforcés d'établir des sous-programmes qui puissent facilement s'emboiter les uns aux autres: les données d'entrées de l'un sont les valeurs de sortie du sous- programme précédent.
En fonction de la capacité de mémoire de l'ordinateur utilisé, ces différents sous- programmes peuvent etrs utilisésindividuellement ou regroupés.
Certains d'entre eux sont spécifiques aux machines synchrones (synthèse dex tables, assignement des états), alorsque les autres peuvent etre aussi bien utilisés pour l'étude des machine asybchrones
A l'heure actuelle ou les composants électroniques (circuits intégrés) sont devenus très économique, on cherche en général à obtenir plutôt un systeme logique réalisable et exempt d'erreur , qu'un systeme s'approchant le plus près possible de l'optimum.
ceci permet de faciliter la programmation et d'économiser en temps de passage sur Ordinateur, sans pour autant accroitre sensiblement le coût de réalisation de la mackine.

CH APITRE1. DEFINITIONS ET CARACTERISTIQUES DES SYSTEMES SEQUENTIELS SYNCH RONES.

.1. PHASES DE FONCTIONNEMENT D UNE MACHINE SEQUENTIELLE.

A. SYSTEMES SYNCHRONES.

Toutes les opérations sont déclenchées par un signal périodique d'origine extérieure au réseau: c'est le signal d'horloge qui se présente sous forme d'impulsions.
Pour une machine synchrone les phases sont constantes en durée.

. MACHINES ASYNCHRONES.

Les phases sont de durée physique variable ;elles ne être définies que par :

Les entrées
Le fonctionnement interne du systeme.

.2. MODELE GENERAL.

Un réseau binaire est un modèle abstrait de réseau logique qui posséde lss propriétés suivantes:

.2.1. ENTREES

Le système possede m entées binaires ($m \geq 1$) x_i (x_i=o ou 1) qui forment un vecteur binaire

$$X= (x_1, x_2, \ldots, x_m)$$

L'entée X peut prendre au maximum 2^m valeurs codées en binaire.

.2.2. SORTIES.

Le systeme possede n sorties z_j formant un vecteur de sorties:

$$Z =(z_1, z2, \ldots, z_n)$$

La sortie peut prendre au maximum 2^m valeurs distinctes codées en binaire

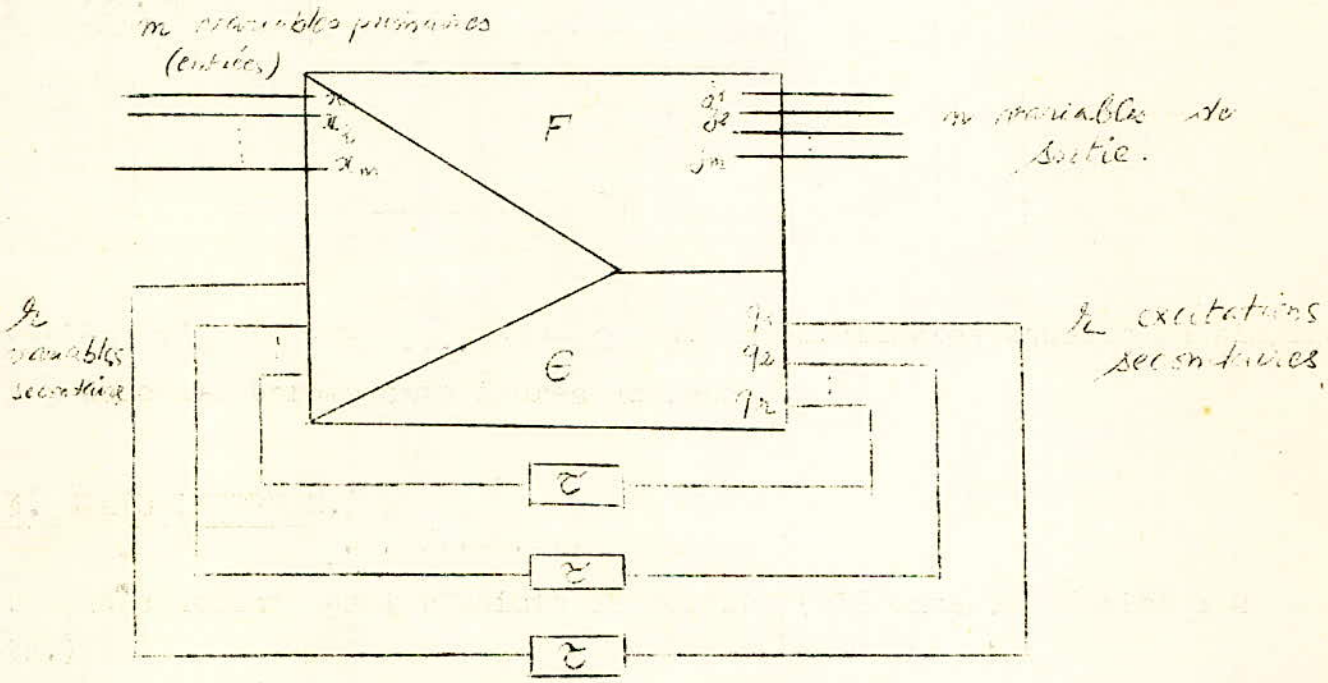
Il y a donc m+n bornes exterieures au reseau.

1.2.3. ETATS INTERNES.

Le reseau contient aussi r points de mémoire (éléments de mémoire à 2 etats.)

Le vecteur $Q = (q_1, q_2, \ldots, q_r)$ est appelé état de mémoire ou état interne du réseau séquentiel.

1.2.4. FONCTION NEMENT.



Les vecteurs entée , sortie, et état interne ne sontdéfinis que pendant des instants numérotés 0 ,1,2,...t,...

En fonction de X et Q à l'instant t(notés X_t,et Q_t) le réseau élabore un vecteur Q,qui sera affecté à la phase t+1 et noté Q_{t+1} .

En fonction de X_t et Q_t, il élabore aussi un vecteur Z_t qui est la sortie dans la phase t.

*Modèle de Huffman -Mealy,ou Mealy (pour les systèmes synchrones).

$Q_{t+1} = g (X_T, Q_t)$

$Z_t = F(X_t, Q_t)$

* Modèle de moore:

$$Q_{t+1} = G(X_t, Q_t) \quad \text{et} \quad Z_t = F(Q_t).$$

1.2.5. ETATS PRESENTS ET FUTURS. ET T TOTAL. TRANSITIONS.

Etats présents d'entrée , interne et de sortie: on utilise l'indice t

(X_t, Q_t, Z_t)

Etat interne futur: Q_{t+1}

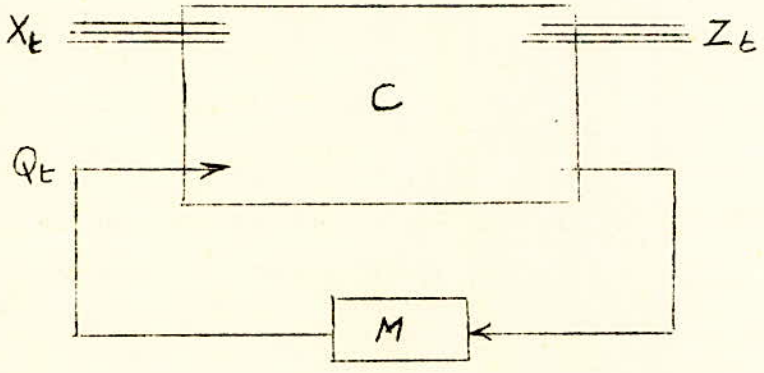
Etat total: le vecteur (X Q)formé de la juxtaposition de X et de Q (vecteur à n+r composantes) reçoit le nom d'état total.

$$Q_{t+1} = \varphi[(XQ)_t]$$
$$Z_t = \Gamma[(XQ)_t]$$

Ces relations traduisent :

- une évolution du réseau:à partir de l'état interne présent Qt et de l'entée présente X_t il détermine l'état futur. D'ou le calcul des tr transitions
- une évolution des sorties.

1.2.6. RESAUX SEQUENTIELS SYNCHRONES ET ASYNCHRONES.



Un réseau séquentiel se décompose en un réseau combinationnel et en une mémoire M.

A l'instant donné , les états de l'entrée et de la sortie de la mémoire M sont associés respectivement à l'état futur Q_{t+1} et l'état présent Q_T

1.2.6.1. SYSTEMES SEQUENTIELS SYNCHRONES.

MEMOIRE/ . Elle est constituée de bascules électroniques ou plus rarement de retards calibrés et égaux à une période d' horloge .On a r boucles de reaction traversant chacune une bascule (ou un retard).

RESEAU COMBINATIONNEL.

En général il est constitué de bascules (T,RS, JK,....etc.)

SYNCHRONISATION

Le réseau fonctionne en synchronisme avec les impulsons d'horloge.
Ce n'est qu'au cours de ces impulsions que le réseau combinationnel est actif et peut prendre en compte les variations d'entée,de sortie,et d'état interne.

On utilise plusieurs méthodes : par exemple on interpose des portes et "ET" sur les entrées x_i et q_i du réseau C.

Le plus souvent la synchronisation est introduite directement au niveau des portes qui forment le réseauC.

L'horloge peut être introduite suivant les cas comme une entée spéciale ou comme une entrée ordinaire.

Dans le premier cas l'horloge est le plus souvent distincte de l'alimentation ;mais dans certains cas on utilise directement celle-ci pour introduire le signal d'horloge (le circuit ne fonctionne que lorsque l'alimentation est haute par exemple)

Elle pourrait être introduite par exemple à l'entrée des bascules: donc les signaux existent seulement pendant les impulsions d'horloge.

* Emploi des retards :

Pour obtenir un retard d'une période d'horloge on peut utiliser des retard exactements calibrés à une période . Mais l'emploi des bascules est meilleur à cause du danger de dérive sous l'influence des conditions d'ambiance (chaleur par exemple)

De plus l'horloge peut varier et les retards se désaccordent par rapport à l'horloge. Par contre quand on utilise des bascules le retard s'ajuste de lui même.

* Signal d'horloge.

La sortie des bascules sert à leur commande d'entrée en se rebouclant à travers C.

dans le cas des bascules à commande par impulsion ,il faut que le signal de sortie des bascules n' arrivent à l'entrée de C que lorsque l'impulsion d'horloge s'est éteinte pour éviter plusieurs transitions par impulsion d'horloge.

Le risque de fuite de synchronisation est plus faible si on utilise le front descendant des impulsions d'attaques des bascules.

Autre condition: il faut un espacement entre les impulsions d'horloge au dela duquel on ne peut pas descendre . Il faut que la période soit suffisante pour que la transition déclenchée par une impulsion au temps t soit terminée avant que se répète l'impuulsion de rang t+1.

L'état permanant des bascules doit être atteint après un temps $t < T$

1.2.6.2.

SYSTEMES SEQUENTIELS ASYNCHRONES.

En général les boucles deréaction sont constituées par des retards.

1.3. REPRESENTATIONS.

1.3.1. EQUATIONS ET TABLE DEVERITE.

Notation:

Etat present	Etat futur
Q_t	Q_{t+1}

$$\begin{cases} Q_{t+} = G(X_t, Q_t) \\ Z_t = F(X_t, Q_t) \end{cases} \rightarrow \text{Mealy.}$$

$$\begin{cases} Q_{t+1} = G(X_t, Q_t) \\ Z_t = F(Q_t) \end{cases} \rightarrow \text{Moore.}$$

1.2.6.2.

1.3.2. REPRESENTATION STUCTURALE.

1.3.2.1. ALPHABETS.

L'ensemble E des entrées X possibles est appelé alphabet d'entrée
L'ensemble S des sorties Z possibles est appelé alphabet de sortie
On désigne par Q l'ensemble des etats internes;

1.3.2.2. MOTS . SEQUENCES. LONGUEUR D'UNE SEQUENCH.

La séquence ou le mot (d'entrée ou de sortie) est une suite finie ou infinie de lettres (d'entrées ou de sorties). Le nombre kde lettres dans la séquence en est la longueur .
Exemple: $X_1 X_2 X_3 \ldots X_k$ constitue une séquence de longeur k
(Elle est formée par la succession de k vecteurs X^0)

1.3.2.3. SUCCESSEURS.

Etant donné un état interne Q , l'état G(Q, X_i) est appelé successeur de Q par X_i.

1.3.2.4. TABLES DES TRANSIONS . TABLES DES SORTIES.

A. TABLE DES TRANSI TIONS . (TABLE DE FLUENCE.)

Exemple :(fig.3.)

Q \ X	X_1	X_2
Q_1	Q_2	Q_4
Q_2	Q_3	Q_6
Q_3	Q_5	Q_7
Q_4	Q_1	Q_2
Q_5	Q_5	Q_6
Q_6	Q_3	Q_4
Q_7	Q_3	Q_7
Q_8	Q_8	Q_7

figure 3

Le contenu d'une case (X, Q) est le successeur de Q (ligne) par X (colonne)

B. TABLE DES SORTIES 5(fig.4)

Exemple: fig.4.

Q \ X	x_1	x_2
Q_1	z_1	z_2
Q_2	z_2	z_1
Q_3	z_2	z_1
Q_4	z_1	z_2
Q_5	z_1	z_1
Q_6	z_1	z_2
Q_7	z_2	z_2
Q_8	z_2	z_2

figure 4

La case (x_i, Q_j) contient la sortie $F(x_i, Q_j)$

C. TABLE COMPOSITE DES TRANSITIONS ET SOR TIES.

Exemple: fig.5.

Q \ X	x_1	x_2
Q_1	2, 1	4, 2
Q_2	5, 2	6, 1
Q_3	3, 2	7, 1
Q_4	1, 1	2, 2
Q_5	5, 1	6, 1
Q_6	5, 1	6, 1
Q_7	3, 2	7, 2
Q_8	8, 2	8, 2

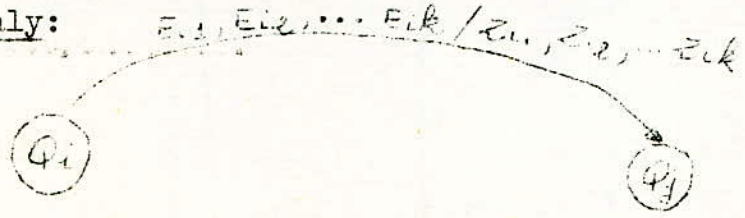
Cette table définit complètement le réseau séquentiel, compte tenu des conventions utilisées

1.3.2.5. GRAPHES D'UN RESEAU SEQUENTIEL.

Ce graphe se compose de :

- noeuds ou sommets représentés par des cercles
- d'arcs orientés , liants des paires de sommmets

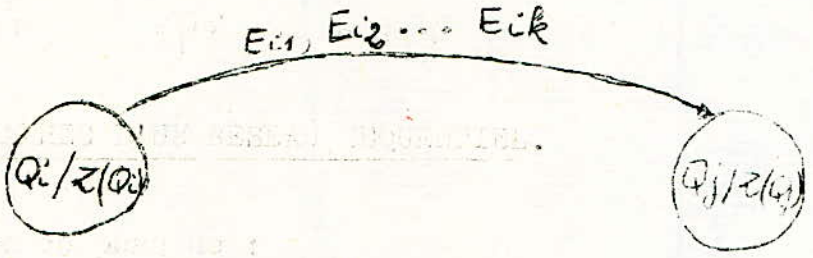
*Reseau de Mealy:



Le cercle contient l'indication d'un état Q_i

Sur l'arc $Q_i \longrightarrow Q_j$ on porte les entrées $E_{i1}, E_{i2}, E_{i3} \ldots E_{ik}$ puis les sorties correspondantes.

* * Reseau de MOORE/



Le cercle contient l'indication Q_i/ Z(Q_i)

Sur la transition $Qi \longrightarrow Qj$ on porte les entrées causant cette transiton.

CHAPITRE .2. SYNTHESE DES TABLES.

Cette partie consiste ,étant donné un certain nombre de spécifications ,d'établir la table de fluence (ou des phases) du systeme séquentiel.

2.1. GENERALITES.

2.1.1. RAPPEL SUR LES EQUATIONS GENERALES D'UN SYSTEME SEQUENTIEL

$$Z_n = F(X_n, Q_n)$$

$$Q_{n+1} = G(X_n, Q_n)$$

Xn = vecteur d'entrée à l'instant n.
Zn = vecteur de sortie à l'instant n.
Qn = vecteur état interne à l'instant n.

212. FORME NORMALE D'UN ENONCE.

DEFINITION. Les spécifications d'un systeme sequentiel sont sous forme normale si elles sont représentées par des correspondances entre un certain nombre de séquences temporelles de vecteurs d'entrée et le même nombre de vecteurs de sortie.

EXEMPLE.

Etant donné l'alphabet d'entrée X=(X_1, X_2, X_3) et l'alphabet de sortie (Z) = (Z1,Z2), des spécifications sous forme normale peuvent etre par exemple:

à la séquence d'entrée X1X2X1X3 correspond la sequence de sortie Z1Z2Z2Z1 (a).

à la séquence X1 X3 X2. CORRESPOND la séquence Z 1 Z1 Z2 (b)

définition: on appelle séquence d'entrée interdite une séquence d'entrée pour laquelle on ne définit aucune séquence de sortie. L'état initial d'une machine est celui ou elle se trouve lorsqu'on

va lui appliquer une séquence d'entrée.

A deux sequences ne pourra correspondre le meme état initial qu'à la condition suivante : si elles commencent toutes deux par le meme vecteur d'entrée ,il faut qu'à ce vecteur corresponde le meme vecteur de sortie.

DEFINITION. On appelle séquence d'entrée itérée ,une séquence qui consiste en l'application à la machine zéro fois ,une fois,.... p fois ..., d'une même suite d'entrée. Notation: $\{X\ 1\ X\ 2\ X\ 1\}$

Ceci signifie qu'on peut appliquer à une machine M :

- ou bien rien du tout (séquence de longeur nulle notée λ)
- ou bien X1 X2 X1
- ou bien X1 X2 X1, X1 X2 X1 .... X1 X2 X1 (p fois)
- ou bien X1 X2 X1... ... X1 X2 X1 ...

A une séquence périodique d'entrée doit correspondre une séquence périodique de sortie.

On peut avoir des séquences :

- entierement périodiquss: exp. $\{X1\ X2\ X1\}$
- a fin périodique : exp. X2 X1 X3$\{X3\ X2\}$
- a début périodique: exp.$\{X2\ X3\}$ X1 X2 X3

Remarque: une séquence à début périodique telle que:

$$\{x_i\ x_e \dots x_k\}\ x_i\ x_j$$
$$\{z_i\ z_e \dots z_k\}\ z_m z_j$$

n'à aucun sens, puisque à x_k succede dans le cadre de la séquence périodique le couple (X_i, Z_i)et, dans la séquence non périodique le couple (X_i, Z_m). Une telle séquence à début périodique n'aura de sens que si la séquence périodique et la séquence non périodique sont compatibles (il faudrait dans l'exemple avoir Zm =Zi)

De façon générale ,on dira qu'un enoncé sous forme normaleest non contradictoire lorsque , parmi ses parties , il n'en existe pas deux telles que les séquences d'entrée soient complétement confondues alors que les sequences de sortie ne le sont pas .

A partir de cet énoncé sous forme normale il existe plusieurs méthodes qui permettent d'établir la table des phases.

Methodes naturelles:

1. méthode de GINSBURG.
2. méthode d'AIZERMAN.
3. cas des machines asynchrones: méthode de MOISIL-IOANIN

Methode algébrique: utilisant les expressions regulieres:

-- méthode de GLOUSKHOV.
-- méthode de BROZOZOWSKI.

2.2. ETUDE DE LA METHODE DE GINSBURG.

2.2.1. ENONCE.

Lorsque l'on doit trouver la table de fluence d'une machine séquentielle qui réalise une correspondance entre p séquences d'entrées et p séquences de sorties (S1 à Sp) dont m ne sont pas périodiques et (m - p) sont à fin périodiques, la méthode s'énonce de la façon suivante:

Soit les séquences données/

$X_1^i \dots X_{k(i)}^i$
$Z_1^i \dots Z_{k(i)}^i$ } avec $1 \leq i \leq m$ non périodiques de longueur $k(i)$.

$X_1^j \dots X_{\ell(j)}^j \{ X_{\ell+1}^j \dots X_{q(j)}^j \}$
$Z_1^j \dots Z_{\ell(j)}^j \{ Z_{\ell+1}^j \dots Z_{q(j)}^j \}$ } avec $m+1 \leq j \leq p$ non périodiques de longueur $\ell(j)$ mais à fin périodique de période $q - \ell$.

A chaque paire (X_b^a Z_b^a) telle que $1 \leq a \leq p$ et $1 \leq b \leq k$ ou q on associe un état Q_b^a et on définit la sortie F et la transition G ainsi :

$$F(Q_b^a, X_b^a) = Z_b^a$$
$$G(Q_b^a, X_b^a) = Q_{b+1}^a \quad \text{si} \quad 1 \leq a \leq m$$
$$G(Q_q^a, X_q^a) = Q_{\ell+1}^a \quad \text{si} \quad m+1 \leq a \leq p$$

Ceci revient à dire que la machine change d'états après l'application de chaque vecteur d'entrée .

Pour chaque séquence Sj la machine est prise à l'état initiale Q_0^j ; par l'application du premier vecteur d'entrée X_1, la machine passe à l'état Q_1^j , par l'application du deuxieme vecteur X_2^j elle passe à l'état Q_2^j,... et ainsi de suite.

2.2.2. EXEMPLE.

Soit la séquence suivante : $\begin{cases} x_1 & x_2 & x_1 & x_3 \\ z_1 & z_1 & z_2 & z_2 \end{cases}$

Initialement la machine est à l'état Q_0 . En appliquant X_1, elle passe à l'état Q_1 et la sortie est Z_1 . En appliquant X_2, elle passe à l'état Q_2 avec la sortie Z_1. DE proche en proche on aboutit à la table composite des états et des sorties de la fig. 2.1.

2)° exemple: séquence à fin itérée (périodique)

Soit la séquence à fin périodique suivante :

$$S \begin{cases} x_1 & x_2 & x_3 & \{ x_1 & x_3 \} \\ z_1 & z_1 & z_2 & \{ z_2 & z_1 \} \end{cases}$$

Initialement la machine est à l'état Q_0.

X1 mène à l'état Q1, X2 à Q2, X3 à Q3, X1 à Q4.

Mais il faut que $G(X_3, Q_4)$ soit spécifié pour permettre le bouclage de l'itération. De façon à réiterer la séquence $X_1 X_3$ on doit avoir

$$G(X_3, Q_4) = Q_3 .$$

Dans ce cas on a bien : $G(X_1, G(X_3, Q_4)) = Q_4$ et

$$F(X_1, G(X_3, Q_4)) = Z_2$$

D'ou la table de fluence representée par le tableau 2.2.

On remarque que cette méthode est facilement programmable, mais elle conduit à un encombrement de la mémoire de la calculatrice, par le nombre très élevé d'états internes qu'elle introduit.

Figure 2.1

φ \ X	X_1	X_2	X_3
φ_0	φ_1, z_1	-, -	-, -
φ_1	-, -	φ_2, z_1	-, -
φ_2	φ_3, z_2	-, -	-, -
φ_3	-, -	-, -	-, z_2

Tableau 2.2

φ \ X	X_1	X_2	X_3
φ_0	φ_1, z_1	-, -	-, -
φ_1	-, -	φ_2, z_1	-, -
φ_2	-, -	-, -	φ_3, z_2
φ_3	φ_4, z_2	-, -	-, -
φ_4	-, -	-, -	φ_3, z_1

Tableau 2.3

φ \ X	X_1	X_2	X_3
φ_0	φ_0, z_1	φ_1, z_1	-, z_2
φ_1	φ_0, z_2	-, -	-, -

2.3.METHODE D'AIZERMAN.

Cette méthode aboutit directement à une table de fluence réduite comportant le moins d'états possibles . Elle a cependant l'inconvénient d'être difficilement programmable.

2.3.1. EXEMPLE

Soit la séquence suivante : $\begin{cases} X_1 & X_2 & X_1 & X_3 \\ Z_1 & Z_1 & Z_2 & Z_2 \end{cases}$

On suppose la machine dans l'état initial Q_0. Rien ne nous empeche de prendre $G(X_1, Q_0) = Q_0$ ce qui impose $F(X_2, Q_0) = Z_1$
Puis on applique X_2, on essaye de prendre encore Qo comme successeur de lui-même par application de l'entrée X_2. On pose:
$G(X_2, Q_0) = Q_0$
Mais alors pour l'entrée suivante X1 on aurait:

$$F(X_1, G(X_2, Q_0)) = F(X_1, Q_0) = Z_1$$

Or on voit ,d'après la séquence à réaliser , qu'on doit avoir

$$F(X_1, G(X_2, Q_0)) = Z_2.$$

On ne peut pas donc prendre $G(X_2, Q_0) = Q_0$
On introduit un nouvel état Q_1 défini par : $G(X_2, Q_0) = Q_1$ et $F(X1, Q1) = Z2$.
Comme F(X3,Qo) n'est pas encore défini on peut prendre G(X1,Q_1) égal à Qo
D'après cet exemple on voit que la machine reste toujours au même état tant qu'on n'aboutit pas à une contradiction .
La table de fluence (tableau 2.3) est ici très réduite.
Si une contradiction apparait dans le remplissage de la table de fluence et de la table de sortie, il ne suffit pas toujours de supprimer le dernier état introduit, il peut être nécessaire de remonter plus haut dans le remplissage. Ceci rend cette méthode difficilement programmable.

2.4. METHODE UTULISEE

Nous avons utilisé une méthode de synthèse des tables personnelle qui s'inspire à la fois de la méthode de GINSBURG et de le méthode d'AIZERMAN. La méthode de GINSGURGest certes très facile à programmer ,mais elle conduit à un nombre d'états internes très élevé, et par suite à un grand encombrement de la mémoire de l'ordinateur La méthode d'AIZERMAN conduit directement à une table des phases réduite, avec un minimum de nombres d'états ,mais elle est difficile à programmer et le temps de programmation serait élevé. La méthode que nous avons utilisé est un compromis entre ces deux méthodes: le nombre d'états internes ,sans être minimal, est assez réduit.

2.4.1. EXPOSE DE LA METHODE.

Soit une machine séquentielle synchrone dont l'énoncé est sous forme normale. La machine est definie par p séquences (d'entrés et de sortie) dont m sont non periodiques et p-m à fin périodique
On commencera d'abord à traiter les séquences non périodiques, puis les les séquences à fin périodique.
On essaie d'établir la table des sorties F(I,J) et la table des transitions G(I,J) de façon à éviter toute contradiction .
Pour ces tables ,la ligne I indique l'état interne de la machine et la colonne J l'entrée (X1,X2,ou X3)
Initialement nous considérons la machine à l'état 1.
Si la machine à l'instant T est dans un certain état interne I elle reste dans le meme état interne I à l'instant T+1 si la sortie F(I,J) n'est pas encore spécifiée.
Autrement ,si l'entrée J à l'instant T+1 est déja apparue alors que la machine se trouvait à l'état I(la sortie F(I,J) serait dans ce cas spécifiée) la machine passe à l'état suivant I+1.
Quand on passe d'une séquence à une autre la machine change d'état elle passe à l'état suivant.

Cas d'une séquence à fin périodique de longueur n, ou la séquence non périodique a une longueur k:

t	1	...	k	k+1	... N
X	X^1	...	X^k	$\{X^{k+1}$	... $X^n\}$

L'entrée X^k à l'instant k met la machine à l'état Q. Il faut qu'à l'instant n, l'entrée X^n méne la machine au même etat Q. pour réaliser le bouclage de l'itération. On garde donc pour ce genre de séquence l'état Q en mémoire de façon à s'en servir pour la fin de la séquence.

Exemple.

$$S_1 \begin{cases} X_1 & X_2 & X_3 & X_1 & X_3 & X_1 \\ Z_1 & Z_2 & Z_1 & Z_2 & Z_2 & Z_1 \end{cases}$$

$$\begin{matrix} X_1 & X_2^* & \{X_3 & X_2 & X_1^*\} \\ Z_1 & Z_2 & \{Z_1 & Z_2 & Z_2\} \end{matrix}$$

Initialement la machine est à l'état 1: d'ou F(1,X1) =Z1
Pour l'état interne 1 l'entrée X2 apparait pour la premiere fois (La sortie F(1,X2) n'est pas encore spécifiée) donc la machine reste au même état.

G(1,X1) = 1 et F(1,X2) = Z2

La sortie F(1,X3) n'est pas encore spécifiée ,la machine reste dans le meme etat 1 pour l'entrée X2: G(1,X2)= 1 et

F(1,X3)= Z1.

La sortie F(1,X1) est déjà spécifiée ;donc pour l'entrée X1 la machine change d'état:

G(1,X3) = 2 et F(2,X1) = Z2. et de même

L'entrée X1 est déjà apparue pour l'état 2 (F(2,X1)) déjà spécifiée) la machine passe à l'état 3

G(2,X3)= 3 et F(3, X1)= Z1.

SYNTHESE DES TABLES

TABLEAU 2.4

I \ J	1	2	3
1	1	1	2
2	2	-	3
3	-	-	-
4	4	4	5
5	4	5	-

TABLE des Transitions

TABLEAU 2.5

I \ J	1	2	3
1	z_1	z_2	z_1
2	z_2	-	z_2
3	z_1	-	-
4	z_1	z_2	z_1
5	z_1	z_2	-

TABLE des sorties

Pour la séquence à fin périodique on fait le même travail et on a en fin de séquence :

$$G(5, X1^*) = G(4, X2^*) = 4.$$

D'ou la table des transitons (tableau 2.4.) et la table des sorties. (tableau 2.5.)

Pour connaitre G(I,J) on teste si pour l'entrée suivante XL la case F(I,L) est déjà remplie:

Si cette case est déjà remplie la machine passe à l'état suivant

$$G(I,J) = I + 1 .$$

Si cette case n'est pas remplie la machine reste au même état

$$G(I,J) = I .$$

2.4.2. CONVEN TIONS.

Pour indiquer qu'un état interne ,ou une sortie n'est pas spécifiée nous utilisons comme symbole le chiffre zéro.

Les vecteurs d'entrées X1,X2,... Xj ... indiquent la colonne de la table des transitions et de la table des sorties (colonne1, 2, ...j).

Les vecteurs de sortie Z1,Z2,...seront inscrits dans la table des sorties tout simplement :1,2 ...

2.5. ORGANIGRAMME

2.5.1. SIGNIFICATION DES TERMES UTILISES.

X(K,T) : tableau des vecteurs d'entrée - K indique la séquence
- T indique l'ordre dans la séquence.

Z(K,T) tableau des vecteurs de sortie.

A(K) indique la longueur de la séquence .

-- Pour les séquences non périodiques A(K) indique le longueur de la séquence.

-- Pour les séquences à fin périodiques A(K) indique la longueur de la partie de la séquence nonpériodique.

-- B(N) indique la longueur totale de la séquence à fin périodique.

F(I,J) : indique la sortie correspondant à l'état I et à l'entrée J.

G(I,J) : indique la transition (l'état futur) pour l'état interne I et l'entrée J.

2.5.2. PROCEDE.

Exemple choisi: (voir tableaux 2 6)

Nous avons choisi un exemple avec 4 séquences dont deux non périodiques et deux à fin périodique.
Le nombre de vecteurs d'entrée est trois :X1 , X2, X3 .
Le nombre de vecteurs desortie est deux: Z1, Z2.
Pour chaque état interne nouvellement intoduit nous posons:
F(I,J) = 0 et G(I,J) = 0 quelque soit J; J=1,... NE avec :
NE= nombre de vecteurs d'entrée.
L'organigramme permet de trouver la table des états internes G(I,J) et la table des sorties F(I,J) dans le cas de C séquences dont D sont non périodiques ,et dont C-D sont à fin périodiques.
Le nombre de vecteurs d'entrée est NE.

2.6. RESULTATS.

Le résultat fournit par l'ordinateurc pour l'exemple choisi est le suivant : (voir tableau 2.7.)

Le tableau obtenu représente la table composite des états,
(G(I,J) indiqué par le premier chiffre) et des sorties .
(F(I,J) indiqué par le 2° chiffre.)
Le chiffre zéro indique un état ou une sortie non spécifiée.

2.7. CONCLUSION.

La méthode que nous avons utilisé nous fournit une table non réduite. Il faut alors utiliser d'autres programmes (Chap.3) pour aboutir à une table plus réduite . Il est à remarquer qu'en général la table qu'on obtient est une table incomplètement spécifiée
D'autres méthodes peuvent etre utilisées por la synthèse des tables
Ainsi la méthode de GLOUSHKOV utilisant les expressions régulières a déjà été programmée:
En france un programme de cette méthode a été mis au point par MF. SAES VACAS au centre d'études et de recherches en automatisme.
IL en est de meme en URSS à l'institut de Cybernétique de l'académie des sciences de la RSS d'UKRAINE.

2.5.2. Pro-cede - Exemple

$k = 1$

T	1	2	3	4
X	x_1	x_2	x_1	x_3
Z	z_1	z_1	z_2	z_2

$k = 2$

T	1	2	3	4
X	x_1	x_2	x_3	x_2
Z	z_1	z_1	z_2	z_2

$k = 3$

T	1	2	3	4	5
X	x_1	x_2	x_3	$\{x_1$	$x_3\}$
Z	z_1	z_1	z_2	$\{z_2$	$z_1\}$

$k = 4$

T	1	2	3	4
X	x_1	x_3	$\{x_1$	$x_2\}$
Z	z_1	z_2	$\{z_2$	$z_1\}$

Tableau 2.6

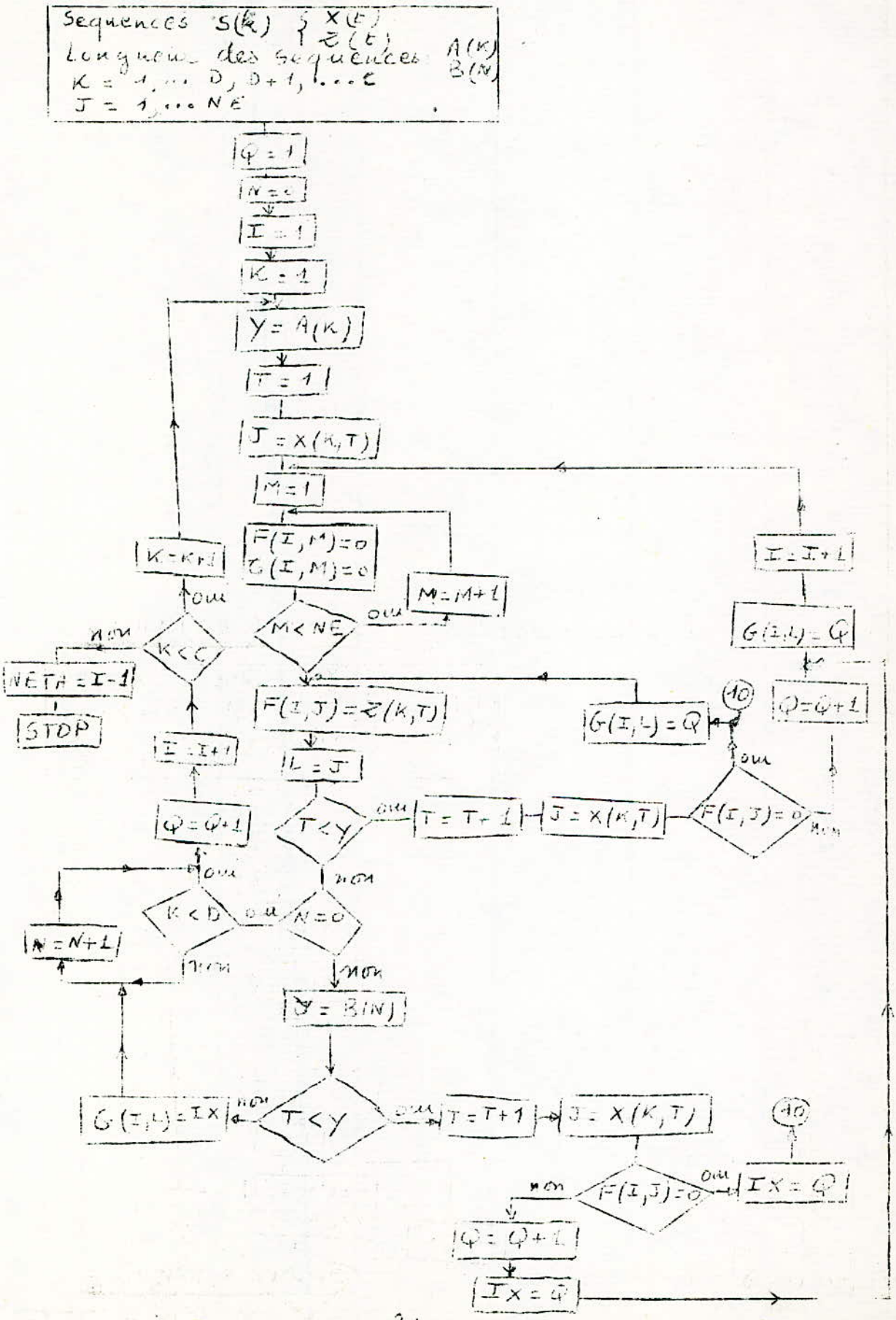
TABLE DES ETATS ET DES SORTIES

TABLEAU : 2.7

	x_1	x_2	x_3
1	1, 1	2, 1	0, 0
2	2, 2	0, 0	0, 2
3	3, 1	3, 1	4, 2
4	0, 0	0, 2	0, 0
5	5, 1	5, 1	6, 2
6	6, 2	0, 0	6, 1
7	7, 1	0, 0	8, 2
8	8, 2	8, 1	0, 0

NOMBRE D ETATS : 8

ORGANIGRAMME : Synthese des tables



```
      Q = 1
      N = 0
      I = 1
      DO 18 K = 1, C
      LONG = A(K)
      T = 1
      J = X(K,T)
7     DO 1 M = 1, NE
      F(I,M) = 0
1     G(I,M) = 0
6     F(I,J) = Z(K,T)
      L = J
      IF (T - LONG) 2, 3, 3
2     T = T + 1
      J = X(K,T)
      IF (F(I,J)) 4, 5, 4
      GOTO 6
5     G(I,L) = G(I,L) + Q
      GOTO 6
4     Q = Q + 1
20    G(I,L) = G(I,L) + Q
      I = I + 1
      GOTO 7
3     IF (N) 2, 9, 8
8     LONG = B(N)
      IF (T - LONG) 13, 14, 14
13    T = T + 1
      J = X(K,T)
      IF (F(I,J)) 15, 16, 15
16    IX = Q
      GOTO 5
15    Q = Q + 1
      IX = Q
      GOTO 20
14    G(I,L) = IX
```

IF(K-D) 10, 11, 11

25

```
11    N=N+1
10    Q=Q+1
      I=I+1
12    CONTINUE
      NETA=I-1
```

CHAPITRE 3. REDUCTION DU NOMBRE DES ETATS D'UNE TABLE.

3.1. IN TROCDUCTION.

Il s'agit d'exposer les règles et les méthodes qui permettent de minimaliser le nombre des états d'une table.
Nous étudierons le cas des tables de fluence complètes ,puis le cas des tables de fluence incomplètes.
En premier lieu nous avons établi un programme en FORTRAN permettant de rechercher toutes les paires compatibles d'une table de fluence donnée.
Une fois les paires compatibles trouvées , il faut rechercher les compatibles maximaux. Nous avons donc ecrit un programme donnant les compatibles maximaux.
En dernier lieu; nous avons écrit un programme permettant de trouver une couverture minimale de la table de fluence.
Nous commencerons par donner quelques définitions importantes.

3.2. DEFINITIONS.

3.2.1. EQUIVALENCE DES ETATS.

Définitio n 1:on apelle complète, une table de fluence dans laquelle ""état suivant "" et ""sortie"" sont définis en tout leurs points, c'est à dire quels que soient l'état interne initial et l'entrée considérée.

Definition 2: deux états internes d'une machine séquentielle sont équivalents si, pour toute séquence d'entrée qui leur est appliquée ils conduisent à des sorties identiques.

Théorème: deux états internes q_i et q_j d'une machine séquentielle sont équivallents si, pour toute entrée X_k, les sorties sont égales et les états suivants respectifs sont eux-mêmes équivalents ou égaux.

3.2.2. REDUCTION DES TABLES COMPLETES.

A. Séquence d'entrée de longueur 1.

Pour construire la table des paires équivalentes, on opèrera de la façon suivante:

-* dans la premiere colonne on mettra toutes les paires d'états vérifiant la condition nécessaire sur les sorties.

-* Chacune des entrées possibles X_k du systeme sert ensuite à reperer une colonne dans laquelle on écrira la paire $G(X_k, Q_i)$ dont l'équivalence est impliquée par celle de Q_i ET Q_j

Le tableau de la figure 3.2. présente ainsi , dans la premiere colonne , toutes les paires d'états qui pourraient constituer des paires équivalentes si les conditions qu'elles supposent(et quis sont inscrites dans les colonnes correspondant aux diverses entrées du systeme) sont vérifiées.

Sur cette table P1 (fig. 3.2.) on entoure les paires d'états de la premiere colonne tels que, dans la ligne qui leur correspond,figurent des paires qui ne sont pas dans cette premiere colonne.

B. Séquence d'entrée de longueur 2.

A partir du tableau P1 on construit le tableau P2 (fig. 3.3) qui s'obtient en entourant dans la premiere colonne toutes les paires dans la ligne desquelles se trouvent une ou des paires encerclées dans la construction de P1.

C. Séquence de longueur supérieure.

On construit de la même façon P3 à partir de P2, P4 à partir de P3, etc...., jusqu'à ce que les lignes qui correspondent aux paires restantes, (non encerclées) dans la premiere colonne ne soient remplies que de paires non encerclées de la premiere colonne.

Le processus est nécessairement limité et l'on aura au maximum (n-1) partitions à faire si la machine compte n états.

	X_1	X_2	X_3
1	5/0	1/1	3/1
2	8/1	6/0	5/0
3	10/1	8/0	7/0
4	9/0	1/1	10/1
5	4/1	6/0	2/0
6	2/0	6/1	8/1
7	6/1	8/0	5/0
8	7/0	8/1	6/1
9	8/1	6/0	3/0
10	2/0	1/1	4/1

Fig. 3.1.
Table de fluence dsnnée.

	x_1	x_2	x_3
1,4	5,9	1,1	3,10
1,6	2,5	1,6	3,8
1,8	5,7	1,8	3,6
1,10	2,5	1,1	3,4
2,30	8,10	6,8	5,7
2,5	4,8	6,6	2,5
2,7	6,8	6,8	5,5
2,9	8,8	6,6	3,5
3,5	4,10	6,8	2,7
3,7	6,10	8,8	5,7
3,9	8,10	6,8	3,7
4,6	2,9	1,6	8,10
4,8	7,9	1,8	6,10
4,10	2,9	1,1	4,10
5,7	4,6	6,8	2,5
5,9	4,8	6,6	2,3
6,8	2,7	6,8	6,8
6,10	2,2	1,6	4,8
7,9	6,8	6,8	3,5
8,10	2,7	1,8	4,6

Fig. 3.2.

Table des Paires - Partition P_1.

	X1	X2	X3
(1,4)	5,9	1,1	3,10
(1,6)	2,5	1,6	3,8
(1,8)	5,7	1,8	3,6
(1,10)	2,5	1,1	3,4
(2,3)	8,10	6,8	5 7
(2,5)	4,8	6,6	2,5
2,7	6,8	6,8	5,5
2,9	8,8	6,6	3,5
3,5	4,10	6,8	2,7
(3,7)	6,10	8,8	5,7
(3,9)	8,10	6,8	3,7
(4,6)	2,9	1,6	8,10
(4,8)	1,9	1,8	6,10
4,10	2,9	1,1	4,10
(5,7)	4,6	6,8	2,5
(5,9)	4,8	6,6	2,3
6,8	2,7	6,8	6,8
(6,10)	2,2	1,6	4,8
7,9	6,8	6,8	3,5
(8,10)	2,7	1,8	4,6

Fig. 3.3.

3.3. REDUCTION DES TABLES INCOMPLETES.

Les tables defluence incomplètes sont des tables pour lesquelles certaines fonctions (fonction sortie ou fonction état suivant) ne sont pas définis par les spécifications.
C'est le cas que l'on rencontre le plus souvent lorsqu'on a à traiter un probleme de synthese d'un systeme sequentiel.
La reduction de ces tables est extremement complexe .
Cependant ,nous pouvons etendre la méthode que nous avons exposé pour le cas des tables complètement spécifiés.
Donnons quelques définitions fondamentales.

DEFINITION 1.

Deux états sont dits compatibles si, pour toute séquence d'entrée applicable , les séquences des sorties correspondantes ont la propriété suivante:

""toutes les fonctions de sortie définies à la fois pour les deux états sont égales.""

THEOREME :

La condition nécessaire et suffisante pour que deux états soient compatibles est que pour toute entrée ou les sorties relatives à ces états sont définies ,les sorties soient égales et que, pour toute entrée ou ils sont tous deux définis ,les états suivants relatifs aux deux états initiaux soient compatibles ou egaux.

Remarquons que la définition de la compatibilité met en évidence que cette relation n'est pad transitive.

3.3.1. DETERMINATION DES PAIRES COMPATIBLES.

Le processus de réduction est analogue à celui que nous venons d'exposer . Nous allons l'illustrer en considérant le tableau de la fig. 3.4. qui représente une table de fluence incomplète.
Le processus de la recherche des paires compatibles nous conduit au tableau de la fig.3.5.

Il ne reste comme paires d'états équivalents que:
(2,7), (2,9) ,(3,5),(4,10),(6,8),(7,9).

IL s'agit maintenant de chercher les classes d'états équivalents par réunion d'états.

--si un état n'est compris dans aucune paire figurant (defaçon non encerclée) dans la table finale des paires ,il n'est équivalent à aucun autre ,et devra figurer dans la table réduite.

-- si une paire de la premiere colonne de la table finale contient deux états qui ne figurent dans aucune autre paire de cette colonne ces deux états forment à eux seuls une classe d'états équivalents. On peut condenser les deux lignes de la table de fluence correspondant à ces états en unee seule.

sur notre exemple, les paires (3,5),(6,8),(4,10),constituent 3classes d'équivalence.

--si une paire de la premiere colonne de la table des paires contient des états qui figurent dans d'autre paires ces états formeront une classe d'équivalence et l'on pourra fusionner en une seule les lignes de la table de fluence qui leur correspondent.

	X1	X2	X3
1	2,–	1,–	5,1
2	4,–	2,0	–,–
3	1,0	3,1	–,1
4	4,0	4,0	3,–
5	6,1	–,1	4,–
6	5,–	1,–	2,–
7	5,–	7,0	1,0

Fig 3.4.

8

	X1	X2	X3
12	2,4	1,2	5,–
13	1,2	1,3	5,–
(14)	2,4	1,4	3,5
(15)	2,6	1,–	4,5
(16)	2,5	1,1	2,5
24	4,4	2,4	–,3
(26)	4,5	1,2	–,2
(27)	4,5	2,7	–,1
(36)	1,5	1,3	–,2
(46)	4,5	1,4	2,3
(47)	4,5	4,7	1,3
56	5,6	1,–	2,4
(67)	5,5	1,7	1,2

Fig. 3.5.

ORGAN IGRAMME:

Procédé:

Nous avons donc programmé la recherche des paires compatibles à partir d'une table de fluence non completement spécifié.

N ous avons adopté les conventions suivantes:

*** un état suivant ou une sortie non spécifiée est représenté par le symbole "0"(zero).

*** Dans la premiere colonne on a representé les couples d'états susceptibles d'etre compatibles ,c'ets -à-dire, possedant la meme sortie ,quand elles sont toutes deux spécifiéss.

On a adopté la convention suivante : si les deux états successeurs sont égaux,ou si l'un ou l'autre n'est pas spécifié,nous les representons par le symbole 0 (zero).

On a choisi de placer les couples(Q_i, Q_j) tels que : $Q_i < Q_j$

Tableaux T1(A,B),T2(A,B): T1(A,B) correspond au premier état et T2(A,B) correspond au deuxieme état.

G(I,J): tableau des états suivants.

F(I,J): tableau des sorties.

NPC. nombre de paires compatibles

Résultats:

Nous avons obtenu les paires compatibles suivantes:

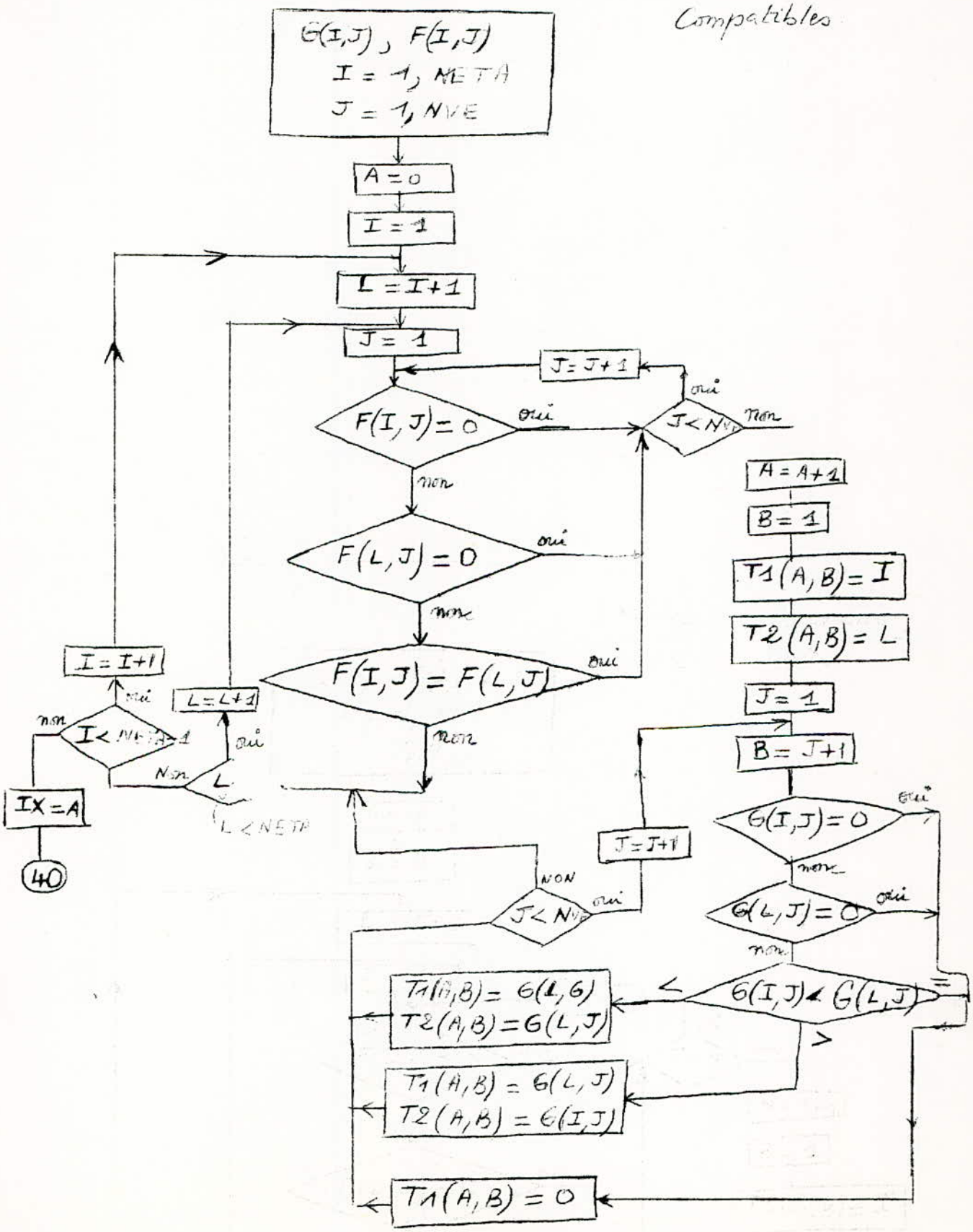
NPC=4. (1,2), (1,3),(2,4);(5,6).

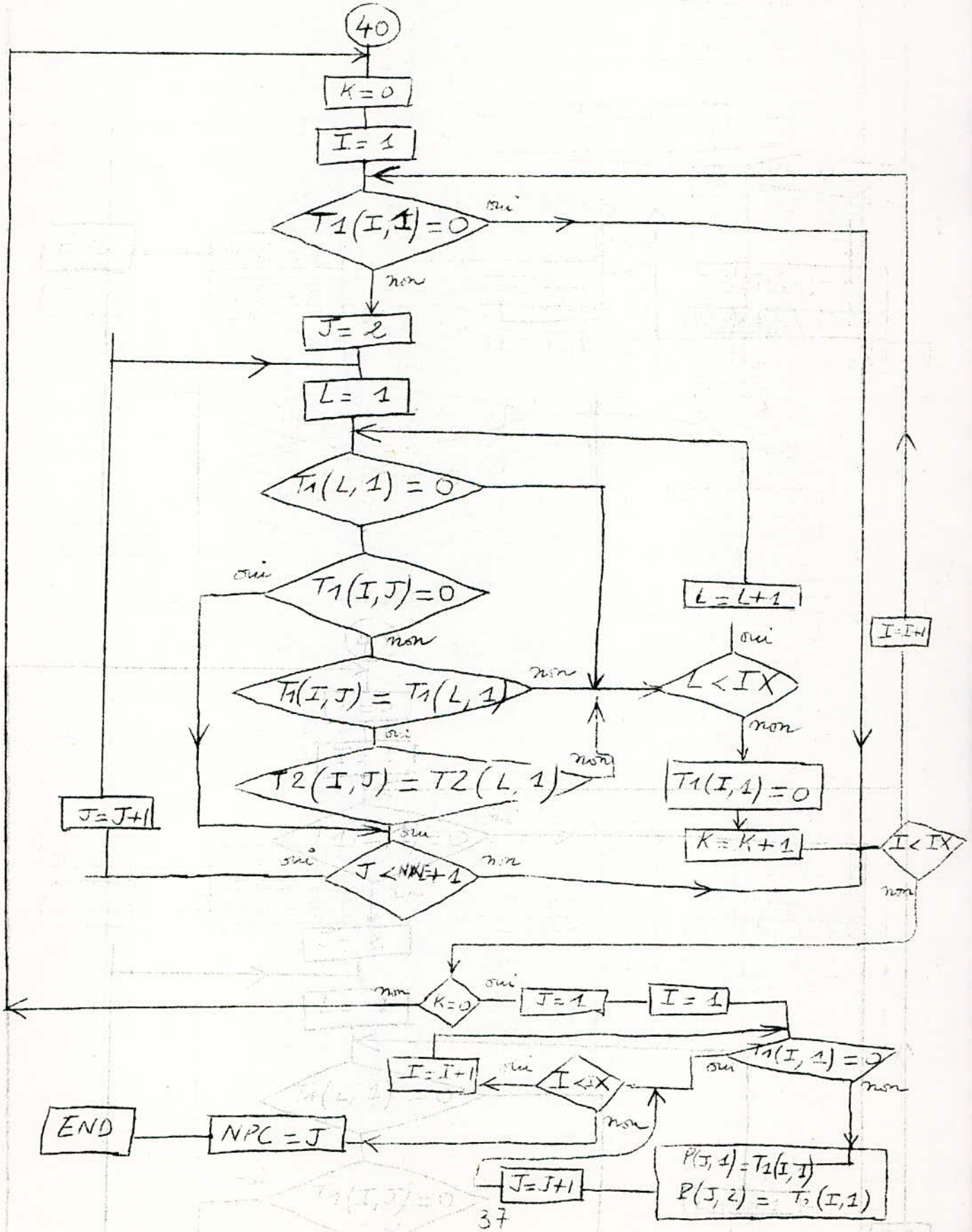
CONCLUSION.

LE PROGRAMME QUE NOUS AVONS ÉTABLI PERMET DONC D'OBTENIR LES PAIRES compatibles d'une table de fluence donnée;

Ce programme reste valable pour le cas des machines séquentielles asynchrones.

Recherche des paires Compatibles




40
K=0
I=1
T1(I,1)=0
oui
non
J=2
L=1
T1(L,1)=0
T1(I,J)=0
oui
non
T1(I,J)=T1(L,1)
non
oui
T2(I,J)=T2(L,1)
non
L=L+1
oui
L<IX
non
T1(I,1)=0
K=K+1
I<IX
I=I+1
J=J+1
oui
J<NBVE+1
non
K=0
J=1
I=1
T1(I,1)=0
I=I+1
I<IX
P(J,1)=T1(I,I)
P(J,2)=T1(I,1)
J=J+1
END
NPC=J

PROGRAMME FORTRAN

Recherche des paires compatibles

```
1       7
        A = 0
        IN = NETA - 1
        DO 3 I = 1, IN
        M = I + 1
        DO 2 L = M, NETA
        DO 5 J = 1, NVE
        IF(F(I,J)) 4, 5, 4
4       IF(F(L,J)) 6, 5, 6
6       IF(F(I,J) - F(L,J)) 2, 5, 2
5       CONTINUE
        A = A + 1
        B = 1
        T1(A,B) = I
        T2(A,B) = L
        DO 7 J = 1, NVE
        B = J + 1
        IF(G(I,J)) 8, 9, 8 ----> 8 IF(G(L,J)) 10, 9, 10
10      IF(G(I,J) - G(L,J)) 11, 9, 12
11      T1(A,B) = G(I,J)
        T2(A,B) = G(L,J)
        GOTO 7
12      T1(A,B) = G(L,J)
        T2(A,B) = G(I,J)
        GOTO 7
9       T1(A,B) = 0
7       CONTINUE
2       CONTINVE
3       CONTINVE
        IX = A
C  RE   CHERCHE DES INCOMPATIBLES
1       K = 0
        DO 13 I = 1, IX
        IF(T1(I,1)) 28, 13, 28
28      DO 14 J = 2, NVE+1
        DO 15 L = 1, IX
```

```
      IF (T1(L,1))16,15,16
16    IF(T1(I,J))18,14,18
18    IF(T1(I,J)-T1(L,1)) 15,21,15
21    IF(T2(I,J)-T2(L,1)) 15,14,15
15    CONTINUE
      T1(I,1)=0
      K=K+1
      GOTO 13
14    CONTINUE
13    CONTINUE
      IF(K) 1,24,1
C     ECRITURE DES PAIRES COMPATIBLES
24    J=1
      DO 25 I=1,IX
      IF(T1(I,1))26,25,26
26    P(J,1)=T1(I,1)
      P(J,2)=T2(I,1)
      J=J+1
25    CONTINUE
      NPC=J-1
      STOP
```

3.4. RECHERCHE DES COMPATIBLES MAXIMAUX.

3.4.1. INTRODUCTION.

POur la réduction du nombre d'états d'une table, après avoir rechercher les paires compatibles ,il faut rechercher les compatibles d'ordre supérieur à deux de façon à former les compatibles maximaux.
POur une table complète ment spécifiée la relation est transitive par exemple: si 1et 2 sont compatibles,de meme 1et 3 ,alors 2 et 3 sont aussi compatibles et on peut former le compatible d'ordre 3:1.2,3
Par contre ,pour une table incomplètement spécifiée ,la compatibilité n'est pas une relation transitive:
Exemple: si on a les paires compatibles 1,2 et 1,3 ,il faut aussi avoir la paire compatible 2,3 pour pouvoir former le compatible d'ordre 3; 1,2,3.
Compte tenu de ce fait, le programme de la recherche des compatibles maximaux ne s'impose que dans le cas général d'une table incompletement spécifiée.

3.4.2. PROPRIETES DES COMPATIBLES.

Tout sous-ensemble d'un compatible est lui même un compatible:
--Etant donné un compatible ($q_1 q_2 q_3 \ldots q_k$)et un état $q_i$qui ne lui appartient pas ,por que ($q_1 \ldots q_k, q_i$) soit aussi un compatible, il faut et il suffit que $q_i$soit compqtible avec q_1, q_2,q_3,.... q_k.
Compatible maximal:
C'est un compatible tel que si on ajoute un autre état et qui ne lui appartient pas ,le nouvel ensemble n'est pas un un compatible, quel que soit cet état. En d'autres termes, il n'existe aucun état,distinct de ceux du compatible, pour lequel la proprieté ci-dessus soit vraie.
--Borne superieure sur le nombre d'états d'une table minimale.
L'ensemble E de tous les comp atibles maximaux est un ensemble fermé
Tout successeur d'un compatible deE est un compatible maximal.
Comme E contient tous ceux-ci ,on voit ainsi que pour tout compatible de E ,ses successeurs sont inclus dans d'autres compatibles de E.
Il fournit donc certainement nu recouvre ment de table initiale.
Le nombre de compatibles maximaux constitue une borne superieure du nombre d'états que possedera une table minimale.

3.4.3. METHODES DE RECHERCHE DES COMPATIBLES MAXIMAUX.

Parmi plusieurs methodes telles que :la methode dePAUL ET UNGER la méthode de Ch. IOAN IN, la méthode deMARCUS....etc.,nous avons choisi la methode de Marcus qui se prete facilement à une programmation en FORTRAN.

3.4.3.1.:EXPOSE DE LA METHODE DE MARCUS.

Cette methode peut etre formulée comme suit:

1. Pour toute paire incompatible(i,j)former la somme booleennei+j
2. Reunir toutes les sommes ainsi obtenues par le signe"." du produit "ET".
3. Developper le produit sous forme d'une somme de produits.
4. Eliminer les termes redondants, par application de la loi d'absorption.
5. Pour tout monome X de la somme ainsi obtenue ,en écrire un nouveau X* dont les facteurs sont tous ceux qui ne figurent pas dans X.

3.4.3.2. EXEMPLE.

Soit une table de fluence possedant 7 etats internes pour laquelle on a obtenu au prealable les paires compatibles suivantes:
(1,2),(1,3),(2,3),(2,4),(2,5),(2,6),(2,7),(3,4),(3,5),(3,6),(3,7),
(4,5),(4,6),(5,6).
Les paires incompatibles sont alors:
(1,4),(1,5),(1,6),(1,7),(4,7),(5,7),(6,7).
La methode deMarcus consiste à developper l'expressionP suivante dans laquelle les paires d'états incompatibles sont sous forme d'une somme logique "OU",et ou on a combiné tous ces temes au moyen de l'opération ET.

$P = (1+4)(1+5)(1+6)(1+7)(4+7)(5+7)(6+7)$

On peut simplifier cette expression en éliminant les termes redondants conformement aux règles de l'algèbre de boole.

$A.A = A$ (1)

$A + AB = A(1+B) = A$ (2)

$(A+B)(A+C)(A+D) = A.A.A + AAB + AAC + AAD + ABC + ABD + ACD + BCD$

D'après (1) et (2) on a :

(A+B)(A+C)(A+D) = A+ BCD.
L'expression P se simplifie donc :
P = (1+ 4567)(456 +7)
En éliminant les temes redondants on obtient finalement :
P = 17 + 4567 + 1456
D'ou l'ensemble des compatibles maximaux obtenus en prenant pour chaque produit de P tous les états qui n'y figurent pas:
((23456)(237)(123))

3.4.3.DEMONSTRATION DE LA METHODE DE MARCUS.

Soit m et n une paire d'incompatible . Le produit de la somme P comprendra alors le terme (m + n)
P ± (...+ ...)(...,+...)(m + N)(...+;;;)
La resolution de cette expression donne des produitsqui comportent chacun un "m" ou un "n". Aucun ensemble d'etats internes manquants ne peut contenir à la fois m et n.
Si p et q constituent une paire de compatibles le produit de sommes ne peut inclure le terme (p+ q). La resolution de l'expression logique P doit donner au moins un produit resultant ne contenant ni p ni q.
Exemple: (m + p)(n + q) =mn +mq + np + pq.
Donc aucun ensemble d'états internes manquants ne peut contenir une paire d'incompatibles, et chaque paire de compatibles appartient au moins à l'un de ces ensembles.

3.4.4. ORGANIGRAMME.

3.4.4.1. PROCEDE.

Connaissant les paires compatibles on recherche pour chaque etat interne I tous les états qui lui sont incompatibles J, K, L ..., et on constitue la somme boolenne (I + JKL ...).On ne s'occupe que des des états supérieurs à I.

Pour une facilité de programmation, les données sont mises sous forme d'un tableau à trois dimensions NC(K,I,J)

Exemple: P = (1 + 4567)(4 + 7)(5 + 7)(6 + 7)

Signification des indices.

-- K indique la somme considerée dans l'expression:

K = 1 (1+4567)

K = 2 (4+7) ...etc.

-- L'indice I indique le terme de la somme considérée:

Pour K=1: I=1 → 1

I=2 → 4567

-- L'indice J indique l'état interne représenté dans chacun des termes des differentes sommes. J= 1,... NETA.

Avec NETA = nombre d'états internes

Quand l'état est present :NC(K,I,J)=1

Quand l'état est absent : NC(K,I,J)=0

On aura ainsi pour l'exemple choisi le tableau 3.4.1.

Autres termes utilisés:

NETA= nombre d'états internes

NINC ≠ nombre d'incompatibles). Il represente le nombre de sommes de l'expression logique P.

NLIG(K) = nombre de lignes I existantes pour cette valeur de K

Au depart NLIG(K)=2 quelque soit K, $1 \leq K \leq$ NINC

LG(K,I) = Pour la ligne K,I il represente le nombre d'états internes représentés dans chacun des termes des differentes sommes.

MC(I,J): ce tableau permet de representer les compatibles maximaux obtenus.

Exemple: MC(1,1)=2

MC(1,2)=3

MC(1,3)=7

Soit donc le le compatible maximal 237

NCM= nombre de compatibles maximaux obtenus.

Remarque: Cet organigramme ne nous fournit que les compatibles d'ordre≥2 . Pour avoir l'ensemble des compatibles maximaux,

il faut ajouter à la liste obtenue l'ensemble des états internes qui ne sont éventuellement compatibles avec aucun des autres états et qui ne sont alors représentés dans aucun des compatibles obtenus.

TABLEAU 341

K	I \ J	1	2	3	4	5	6	7	TERME
1	1	1	0	0	0	0	0	0	1
	2	0	0	0	1	1	1	1	4567
2	1	0	0	0	1	0	0	0	4
	2	0	0	0	0	0	0	1	7
3	1	0	0	0	0	1	0	0	5
	2	0	0	0	0	0	0	1	7
4	1	0	0	0	0	1	0	0	6
	2	0	0	0	0	0	0	1	7

TABLEAU des données

$P = (1 + 4567)(6 + 7)(5 + 7)(6 + 7)$

3.4.4.2. PROCEDE.

Exemple: P = (1 + 4567)(4 + 7)(5+7)(6+7)
Nous effectuons d'abord le premier produit .
Puis avant d'effectuer l'autre produit nous eliminons d'abord les termes redondants.
Pour eliminer les termes redondants ,on prend chacun des termes obtenus qu'on compare succéssivement à tous les autres termes.
Dans le cas ou sa longueur (LG(K,I)) est inferieure ou égale à celle du é 2° terme avec laquel on le compare, on vérifie si tous les états representés dans le premier sont aussi représentés dans le second. Dans le cas affirmatif,il faut supprimer le second terme.
En procedant ainsi de proche en proche on aboutit à une expression reduite de P.

P = (14+17+4567+4567)(5+7)(6+7)
P = (14+17+4567)(5+7)(6+7)
P = (145+147+157+17+4567+4567)(6+7)
P = (145+17 +4567)(6+7)
P = (1456+1457+167+17+4567+4567)
P = 1456+17+4567

Dans l'organigramme pour indiquer qu'un terme est éliminé ,nous posons: LG(K,I)=0 de façon à ne plus en tenir compte.
Finalement pour obtenir les compatiblesmaximaux ,il suffit de prendre les états qui ne sont pas representer dans les differentes produits de la somme finale:
(237,23456,123)

RESULTATS OBTENUS.

Pour l'exemple choisi :
P= (1+4567) (4+7)(5+7)(6+7)
L'ordinateur nous a fourni le resultat suivant:

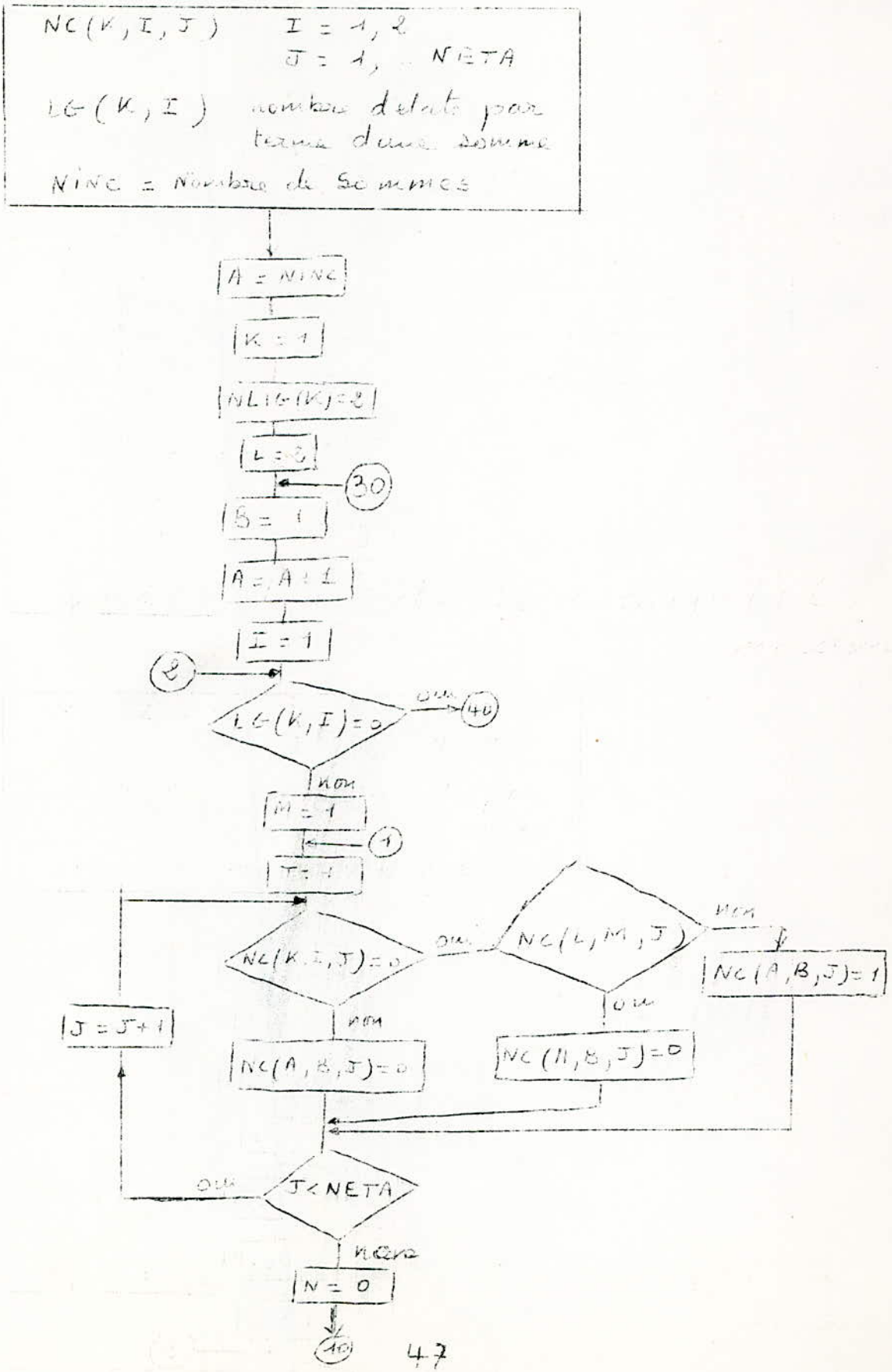
COMPATIBLES MAXIMAUX: 0 1 1 0 0 0 1
0 1 1 1 1 1 0
1 1 1 0 0 0 0

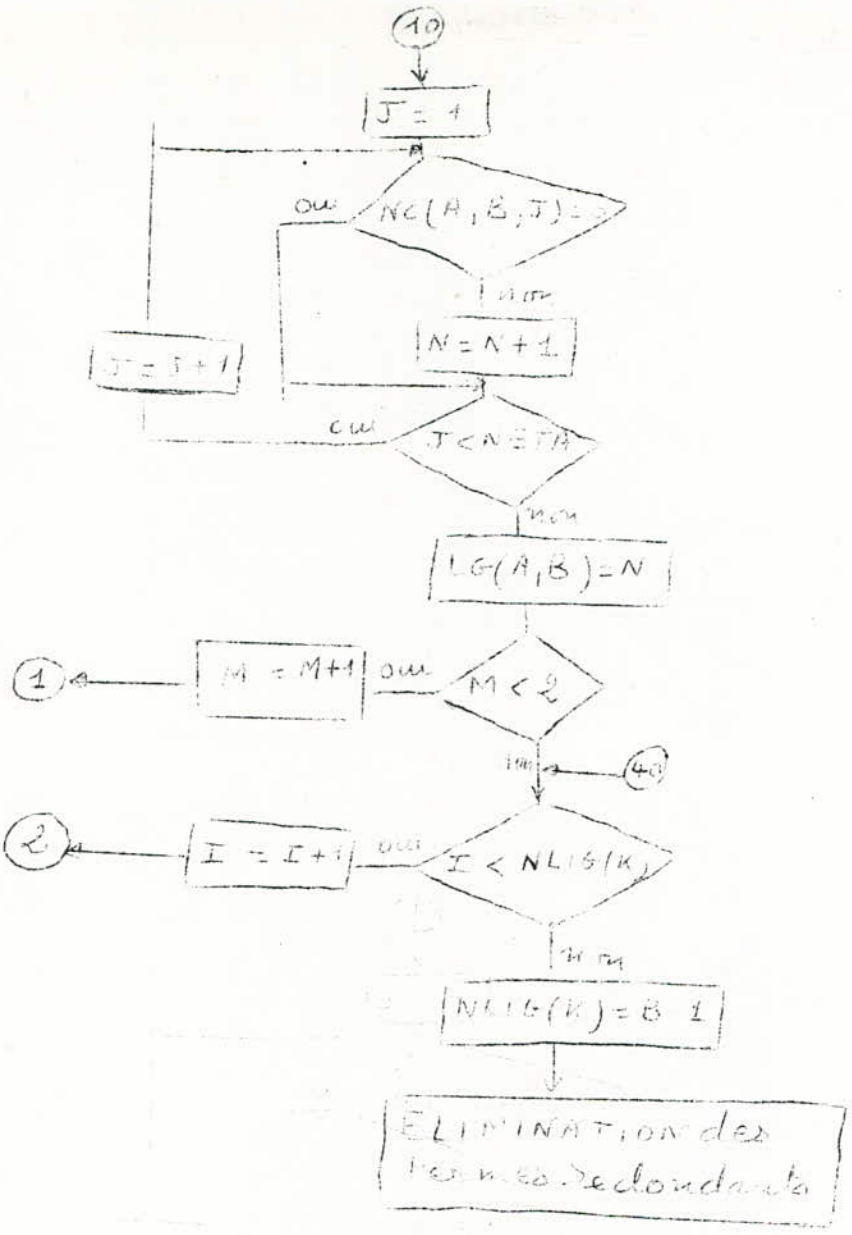
Nombre de compatibles maximaux :3
Soit donc les compatibles maximaux:237, 23456, 123.

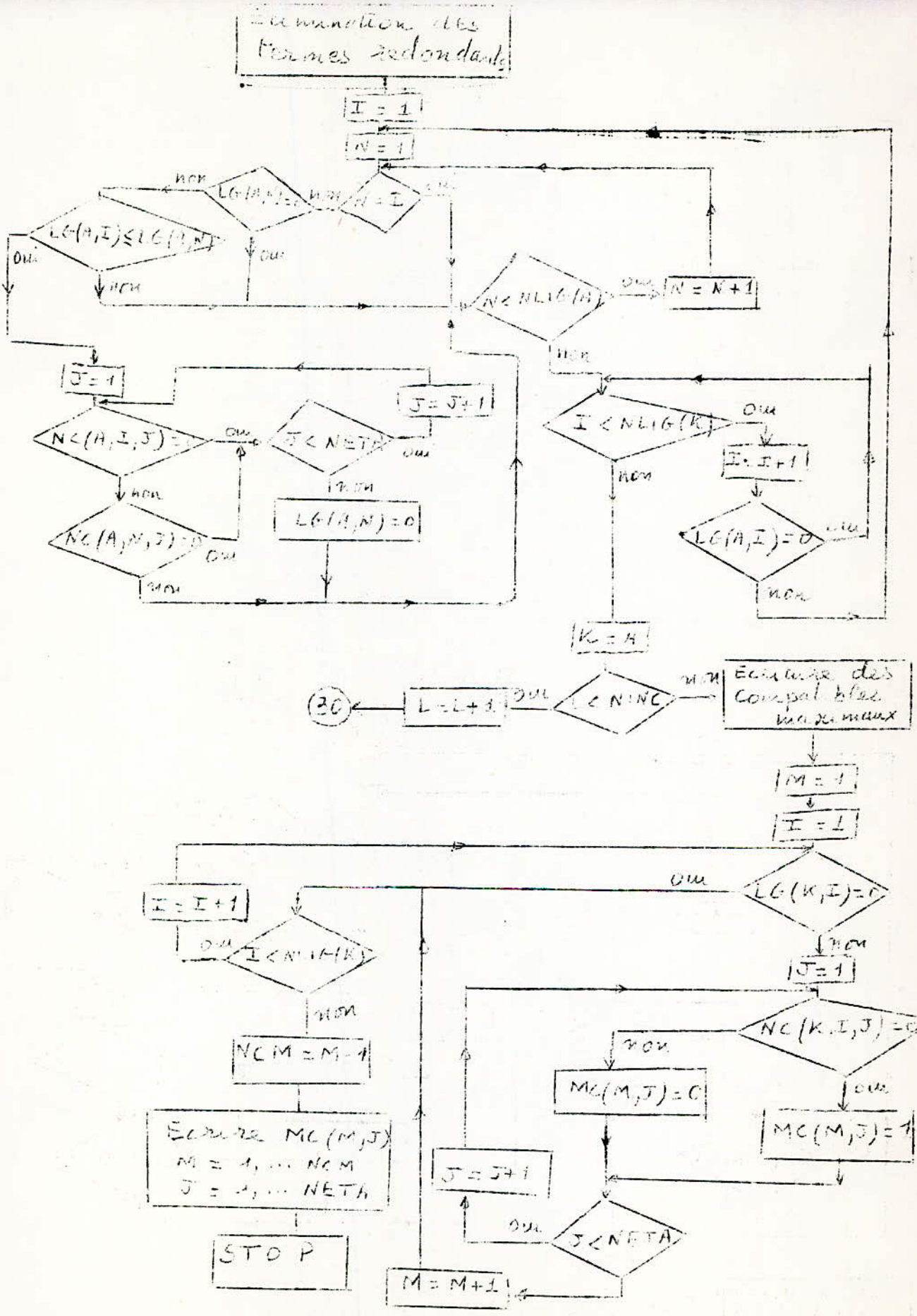
CONCLUSION:

Les methodes de PAUL ET UNGAR, et deCH. IOANIN sont aussi programmables.
La methode de marcus , basée sur les regles de la logique binaire doit se preter mieux à une programmation dans un autre langaga que le fortran,qui est un langage plûtôt arithmétique .

Organigramme de la recherche des compatibles maximaux




10
J = 1
OUI
NC(A,B,J) =
NON
N = N + 1
J = J + 1
OUI
J < NETA
NON
LG(A,B) = N
1
M = M+1
OUI
M < 2
NON
4@
2
I = I+1
OUI
I < NLIG(K)
NON
NLIG(K) = B 1
ELIMINATION des termes redondants


Elimination des Termes redondants
I = 1
N = 1
LG(A,N) = 0
N = I
LG(A,I) ≤ LG(A,N)
N < NLIG(A)
N = N+1
J = 1
J = J+1
N < (A,I,J) = 0
J < NETA
NC(A,N,J) = 0
LG(A,N) = 0
I < NLIG(K)
I = I+1
LG(A,I) = 0
K = A
L = L+1
(30)
I < NINC
Ecriture des Complexes maximaux
M = 1
I = 1
I = I+1
I < NINC
LG(K,I) = 0
J = 1
NC(K,I,J) = 0
MC(M,J) = 0
MC(M,J) = 1
J = J+1
J < NETA
M = M+1
NCM = M-1
Ecrire MC(M,J)
M = 1, ... NCM
J = 1, ... NETA
STOP
oui
non

PROGAMME FORTRAN

Recherche des paires compatibles

```
1                7
                 A = NINC
                 K = 1
                 NLIG(K) = 2
                 DO 1 L = 2, NINC
                 B = 1
                 A = A + 1
                 DO 2 I = 1, NLIG(K)
                 IF (LG(K,I)) 3,2,3
3                DO 4 M = 1,2
                 Y = NETA
                 DO 5 J = 1, Y
                 IF (NC(K,I,J)) 6,7,6
7                IF (NC(L,M,J)) 6,8,6
6                NC(A,B,J) = 1
                 GO TO 5
8                NC(A,B,J) = 0
5                CONTINUE
                 N = 0
                 DO 9 J = 1,Y
                 IF (NC(A,B,J)) 10,9,10
10               N = N - 1
9                CONTINUE
                 LG(A,B) = N
4                CONTINUE
2                CONTINUE
                 NLIG(A) = B - 1
C   ELIMINATION DES TERMES REDONDANTS
                 I = 1
                 NK = NLIG(A)
26               DO 12 N = 1, NK
                 IF (N-1) 13, 12, 13
13               IF (LG(A,N)) 14, 12, 14
14               IF (LG(A,I) - LG(A,N)) 15, 15, 12
15               J = 1
20               IF (NC(A,I,J)) 16, 17, 16
```

```
1       7

16      IF(NC(A,N,J)) 17, 12, 17
17      IF(J-Y) 12, 19, 19
18      J = J+1
        GOTO 20
19      LG(A,N) = 0
12      CONTINUE
25      IF(I - NLIG(A)) 21, 22, 22
21      I = I+1
        IF(LG(A,I)) 26, 25, 26
22      K = A
1       CONTINUE
C Ecriture des compatibles maximaux
        M = 1
        NK = NLIG(K)
        DO 25 I = 1, NK
        IF(LG(K,I)) 28, 27, 28
28      DO 32 J = 1, Y
        IF(NC(K,I,J)) 30, 31, 30
30      MC(M,J) = 0
        GOTO 32
31      MC(M,J) = 1
32      CONTINUE
        M = M+1
27      CONTINUE
        NCM = M-1
```

3.5. RECHERCH E D'UNE COUVERTURE MINIMALE ET FINIE.

1. NITRODUCTION.

Ce programme permet de choisir parmi les compatibles maximaux trouvés précédemment ceux qui vont permettre une couverture de la table initiale, qui soit à la fois minimale(nécessitant un nombre réduit de compatibles) et finie (il faut que tous les états de la table initiale y soient representes.)

Le choix de ces compatibles maximaux permet de trouver une table T' minimale recouvrant la table initiale T.

2. RECOUVREMENT D'UNE TABLE T PAR UNE TABLE T'.

T est recouverte par T', si pour tout état q_i de T,il existe un état q'_i de T', tel que q_i soit recouvert par q'_i.

Pour qu'une table T' recouvre une table T,il faut:

1. Qu'il existe un recouvrement des ensembles Q des états de T dont les ensembles soient associés aux états de T'.
2. Que les ensembles du recouvrement soient des compatibles.
3. Que le recouvrement soit fermé. Il faut que le successeur de chacun des compatibles ,pour toute entrée X ,ne se fractionne pas en parties appartenant à des compatibles différents.
4. Pour que la table T' soit minimale il faut que le nombre de compatibles de recouvrement soit minimal.

BORNE SUPERIEURE SUR LE NOMBRE D'UNE TABLE MINIMALE.

Le nombre de compatibles maximaux constitue une borne superieure du nombre d'états que possedera une table minimale. En effet tout successeur d'un compatible est aussi un compatiblemaximale L'ensemble des compatibles maximaux est un ensemble fermé et fournit certainement un recouvrement de lat table initiale

BORNE INFERIEURE SUR LE NOMBRE D'ETATS.

Le nombre d'états du plus grand incompatible maximale est une borne inférieure du nombre d'états de la table minimale.

3. METHODE UTILISEE.

Les données de ce programme sont les sorties du programme sur la recherche des compatibles maximaux. Les données sont sous forme d'un tableau MC(I,J) comportant les compatibles maximaux d'ordre supérieur ou égale à2

Procédé.

Exemple : tableau 54.

On part du tableau MC(I,J) contenant l'ensemble des compatibles maximaux d'ordre supérieure ou égale à 2.

J = 1, ... nombre d'états (nétat)

I = 1, ... nombre de compatibles maximaux (NCM)

On commence par regarder les états qui ne sont représentés dans aucun des compatibles et les états qui y sont représentés une seule fois.

Si l'état J n'appartient à aucun des compatibles, on est obligé de le choisir pour une couverture de façon à respecter la condition 2. (couverture finie)

Si l'état J n'appartient qu'à un seul des compatibles, on choisit ce compatible pour notre couverture . Tous les autres états représentés dans ce compatible seront barrés. Dans la suite de la recherche on ne s'occupera que des états non encore choisis

Dans le cas ou tous les états J ne sont pas choisis, les états non choisis appartiennent à deux compatibles ou plus. Pour chaque "état J" non encore choisi ,on choisi parmi tous les compatibles ou il est represenété ,le compatible possédant la longueur la plus grande,c'est-à-dire représentant le maximum d'états.

Pour l'exemple choisi:

--- l'état 10 n'est représenté dans aucun des compatibles, on le choisit.

--- l'état 1 n'est représenté qu' une fois: on choisit le compatible 124. DE meme por l'*état 3: on choisit le compatible 38.

--- Tous les autres états : 5,6,7 et9 sont représentés chacun dans deux compatibles différents. Pour l'état 5 on choisit le compatible 5679. qui possede la longueur (4) la plus grande.

Tous les états sont représentés,d'ou la couverture :

(10), (124), (38), et (5679)

Recherche d'une couverture minimale et finie

Tableau 54

E	1	2	3	4	5	6	7	8	9	10	compatibles
1	1	1	0	1	0	0	0	0	0	0	1 2 4
2	0	1	0	0	0	0	1	0	1	0	2 7 9
3	0	0	1	0	0	0	0	1	0	0	3 8
4	0	0	0	0	1	1	1	0	1	0	5 6 7 9
5	0	0	0	0	1	1	0	1	0	0	5 6 8

Exemple : Tableau Mc (I, J)

4. ORGANIGRAMME.

Conventions et termes utilisés:

- MC(I,J) : tableau des données donnant l'ensemble des compatibles maximaux d'ordre supérieur ou egale à 2.

- NETA : nombre d'états.

- NCM : nombre de compatibles maximaux

- JC(M) : tableau qui indique si un état est choisi ou non.
  Initialement on pose : JC(M)= 0 pour M=1,...NETA.
  Chaque fois qu'un état M est choisi: JC(M)=1.

- LN(J,K) : tableau qui fournit l'ensemble des compatibles ou l'état J est représenté.

- NX(J) : nombre de compatibles ou l'état J est représenté .

- Y(I,L) : tableau qui permet de recueillir l'ensemble des compatibles choisis.

- A(I) : tableau qui nous fournit la longueur (nombre d'états représentés) de chaque compatible.

Resultat obtenu.

Pour l'exemple choisi (tableau 4) le resultat fournit par l'ordinateur est le suivant/

COMPATIBLES CHOISIS: 10
124
38
5679

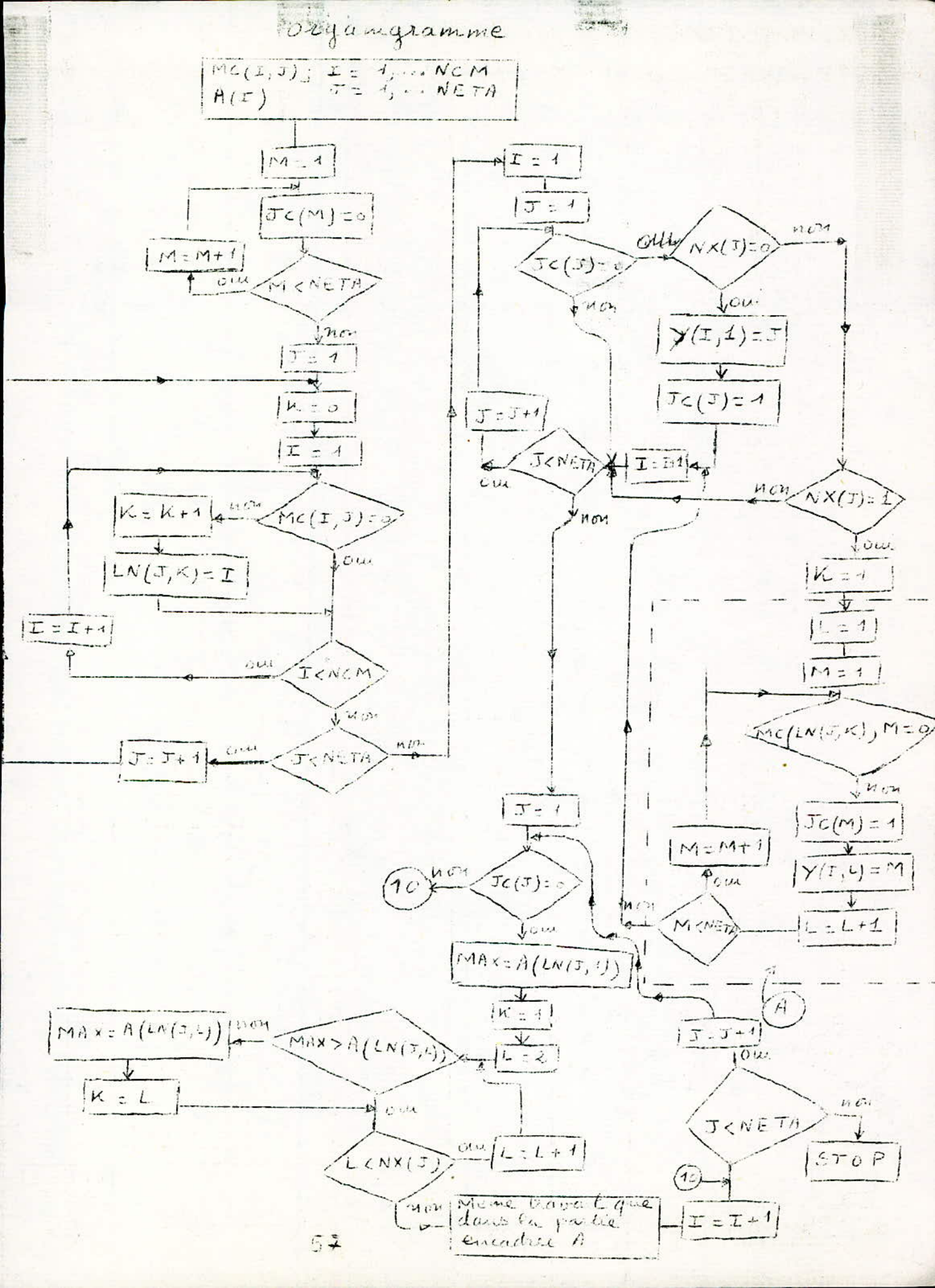
CONCLUSION.

Cet organigramme nous permet donc de choisir parmi les compatibles maximaux ceux qui fournissent une couverture à la fois minimale et finie.

Ayant fait ce choix ,il suffit alors de réecrire la nouvelle table minimale T' à partir de la table initiale T, en remplaçant chacun des états de T par le compatible qui le recouvre.

Pour ce programme l'encombrement de la mémoire est faible même pour une machine possedant un nombre élevé d'états.

Organigramme



Recherche d'une couverture minimale et pure

PROGRAMME FORTRAN

```
      DO 1 M = 1, NETA
1     JC(M) = 0
      DO 2 J = 1, NETA
      K = 0
      DO 3 I = 1, NCM
      IF (MC(I, J)) 4, 3, 4
4     K = K + 1
      LN(J, K) = I
3     continue
2     NX(J) = K
C     Représentes une (ou zero) fois
      I = 1
      DO 5 J = 1, NETA
      IF (JC(J)) 5, 6, 5
6     IF (NX(J)) 7, 8, 7
8     Y(I, 1) = J
      JC(J) = 1
      GOTO 11
7     IF (NX(J) - 1) 5, 10, 5
10    K = 1
      L = 1
      DO 15 M = 1, NETA
      LA = LN(J, K)
      IF (MC(LA, M)) 16, 15, 16
16    JC(M) = 1
      Y(I, L) = M
      L = L + 1
15    continue
11    I = I + 1
5     continue
      DO 9 J = 1, NETA
      IF (JC(J)) 9, 12, 9
12    MAX = A(LN(J, 1))
      K = 1
      NL = NX(J)
```

```
      DO 13 L = 2, NL
      IF (MAX - A(LN(J,L))) 14, 13, 13
14    MAX = A(LN(J,L))
      K = L
13    CONTINUE

      L = 1
      DO 18 M = 1, NETA
      LA = LN(J,K)
      IF (MC(LA,M)) 19, 12, 19
19    JC(M) = 1
      Y(I,L) = M
      L = L + 1
18    CONTINUE
      I = I + 1
9     CONTINUE
      STOP
```

CHAPITRE 4. CODAGE DES ETATS.

4.1. INTRODUCTION.

4.1.1. GENERALITES.

Une etape fondamentale du cheminement vers la réalisation pratique de la machine consiste à "coder" les états précédemment définis c'est-à-dire, leur associer des variables binaires (dites secondaires) qui seront attachées par la suite à des organes technologiques Cette étape doit donc permettre de dresser la matrice d'excitation ou de transition.
Le problème consiste à affecter à chaque ligne de la table (c'est-à-dire en fait, à chaque état) un nombre binaire comprenant un certain nombre de digits, de façon à distinguer chacune de ses lignes(donc les états internes correspondants.)
Pour les systèmes séquentiels asynchrones, on s'efforce de satisfaire les adjacences entre états afin d'éviter les problèmes de courses.
Pour les systèmes séquentiels synchrones , libérés de la contrainte d'adjacence et de transitions directs , nous allons nous attacher à choisir parmi les nombreux codes possibles celui ou ceux qui peuvent présenter des avantages de réalisation. L'assignement (ou codage) choisi a une incidence considérable sur la quantité de matériel requis pour la réalisation finale du système.
On peut être amené à des décompositions de systèmes séquentiels en ensemble de systèmes de moindre volume.

4.2. CODES DISTINCTS . CODES VALIDES.

Lorsque l'on veut coder une table de fluence il suffit, donc, d'affecter à chaque ligne un ensemble binaire, ou"mot". Il est bien clair que ce codage ne sera valable que s'il permet de caractériser les lignes et, par conséquent, les états, c'est-à-dire qu'un mot

4.3. EXEMPLE DE CODAGE D'UNE TABLE.

Considérons le système représenté par sa table de fluence fig. 4.1. On effectue son codage au moyen du code Gray à 3 digits. On aboutit donc aux tableaux de la fig.4.2. Les trois tableaux correspondent aux trois variables.

	$\bar{x}$	x
1	5	1
2	8	1
3	6	3
4	7	5
5	1	5
6	3	6
7	4	3
8	2	6

Fig. 4.1

x \ y_2y_3	00	01	11	10	00	01	11	10	00	01	11	10
y_1												
0	1	1	1	1	1	0	1	0	0	0	1	1
0	0	0	0	0	0	1	1	0	1	0	1	0
1	1	0	1	1	1	1	1	1	1	1	1	0
1	0	0	0	1	0	0	1	1	0	0	1	0

Fig 4.2.

Le tableau de la fig 4.2. nous permet de déduire les equations du système :

$$Y_1 = \bar{X}\bar{y}_1 + X(y_2\bar{y}_3 + y_1 y_2 + y_1\bar{y}_3)$$

$$Y_2 = \bar{X}\,\bar{y}_1\,\bar{y}_2\,\bar{y}_3 + (y_1 + y_2)(X + y_2)$$

$$Y_3 = X y_1 y_3 + \bar{X}\,\bar{y}_1\,y_2 + y_2 \cdot y_3 + y_1\bar{y}_2\,\bar{y}_3$$

	00	01	11	10
00	1	2	3	4
01	8	7	6	5
11	8	7	6	5
10	1	2	3	4

Etats présents.

Fig. 4.3

	00	01	11	10
00	5	8	6	7
01	2	4	3	1
11	6	3	6	5
10	1	1	3	5

Etats Suivants.

Fig. 4. 4

REMARQUES:

1. Dans les tables de KARN AUGH que nous venons de représenter, une case donnée correspond à un ensemble y1 y2 y3 X, soit encore à un état présent et à l'entrée qu'on lui applique ; quant au contenu de la case ,c'est la valeur que prend la variable considéree dans le mot associé à l'état suivant.
2. Considérons la variable y1; nous voyons qu'elle prend dans le code choisi la valeur 0 pour les états 1,2,3,4 et la valeur 1 pour les états 5,6,7, et 8.
   On dira qu'elle réalise une partition en deux classes que l'on écrira sous la forme : (1 2 3 4 ,5 6 7 8)

4.4. CODAGE PAR ETUDE DES ADJACENCES (W. HUMPHREY).

4.4.1. NOTIONS GENERALES SUR LES PARTITIONS.

Une partition d'un ensemble E est une famille de sous ensembles deux à deux disjoints tels que leur union soit l'ensemble E lui meme
Le nombre de classes est compris entre 1 (partition I) et n, nombre d'états de l'ensemble ,c'est-à-dire de la machine considérée (partition 0)
Une partition u possède la propriété de substitution par rapport à un système S si, pour toute paire d'états q_i, q_j appartenant à la même classe de u et pour chaque entrée de X_k les états suivants q_{ik} et q_{jk} sont également contenus dans une même classe de u
On utilisera pour ces partitions l'abréviation "p.p.s" = partition à propriété de substitution.

4.4.2. RELATION D'ORDRE, SOMMES , PRODUITS.

Deux partitions u1 et u2 sont égales si chaque classe de l'une est identique à une classe del'autre.
Une partition u1 est plus grande qu'une partition u2 si, chaque classe de u2 est contenue dans une classe de u1.
On appelle somme de 2 partitions la partition obtenus en effectuant l'union des classes qui les composent .
On appelle produit de deux partitions ,la partition obtenue en effectuant l'intersection des classes qui les composent.

4.4.3. UTILISATION DES PARTITIONS p.s.POUR LE CODAGE.

Définitions:

les éléments fondamentaux d'une partition sont :

* * le nombre de classes que nous notons K(u)
* * le nombre d'états contenus dans la plus grande classe: L(u)

On pose

$$K = (\log_2 K(u)) \text{ et } l = (\log_2 L(u))$$

k variables sont nécessaires pour coder les différentes classes de u qui se comportent comme des états associés .

l variables sont nécessaires pour coder les états de la plus grande classe.

Les k premieres variables permettent de caractériser la classe dans laquelle se trouve un étatt donné.

Les l variables suivantes permettent de caractériser cet état à l'intérieur de la classe définie par les k premieres.

Le nombre minimal de variables est So définie par :

$$So = (\log_2 n)$$

Une condition pour avoir un codage présentant un nombre minimal de variables est donc :

$$\boxed{So = k + l}$$

4.4.4. PROCESSUS DE RECHERCHE DES PAIRES SUBSTITUTIVES.

Considérons la table 4.5.

ETATS PRESEN TS	ETATS FUTURS			
	0 0	0 1	1 1	1 0
1	2	4	2	4
2	7	5	7	5
4	5	5	5	5
5	6	6	6	6
6	1	1	1	1
7	9	6	9	6
9	1	1	1	1

Les partitions substitutives d'une machine peuvent etre calculées en considérant ,à tour de rôle , chaque couple d'états internes, dans la table des états ,et en déterminant les couples d'états futurs pour tous les états d'entrée possibles.
On poursuit le calcul en cherchant les paires des états futurs pour les couples précédemment déterminés et ainsi de suite, jusqu'à ce que une liste des couples d'états soit obtenue. On examine ensuite la liste finale et on combine les couples appropriés pour former la plus petite partition substitutive, en exploitant le fait que la condition de substitution est transitive.
Considérons la table des états représentés par le tableau 4.5.
En comparant le couple d'états (1,5) , on obtient la table de fluence donnée par le tableau 4.6. Les coples d'états n'ont besoin d'être inclus qu'une seule fois dans la partition.
On obtient finalement :

$$P1 = (1,5)(2,6)(4,6)(7,1)(2,9)$$

Remarquons que tous les états internes sont inclus dans la partition. Les couples (1,5) et (7,1) ainsi que les couples (2,6), (4,6) et (2,9) peuvent etre combinés selon la relation de transitivité pour donner la plus petite partition substitutive :

$$P1 = (1,5,7)(2,4,6,9)$$

Pour cet exemple particulier ,il n'ya que deux partitions non triviales, qui jouissent de la propriété de substitution .

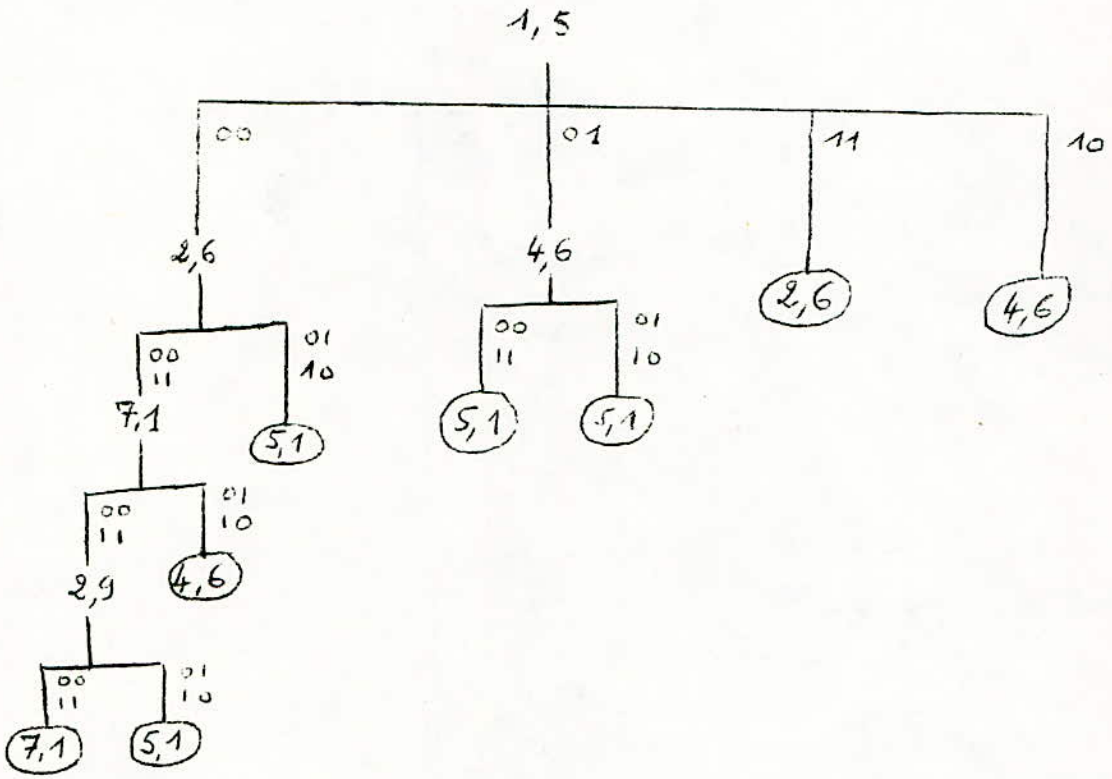
$$P1 \neq (1,5,7)(2,4,6,9)$$

P2 = (1)(2,4)(7,5)(6,9)

toutes les autres sont triviales ,en ce sens que tous les états sont contenus dans une partition :

P3 = (1,2,4,5;6,7,9)

Tableau 4.6. Extraction des p.s.



Comment utiliser maintenant les partition substitutives pour le codage des états ?

Une fois que l'on a obtenu l'ensemble complet de toutes les partitions substitutives non triviales , il faut sélectionner celles qui contiennent le moins de blocs (ou d'éléments)

La premiere étape consiste donc à sélectionner une partition adéquate
Supposons que l'on ait choisi :

$$P1 = (1)(2,4)(5,7)(6,9)$$

Cette partition contient 4 blocs contenant chacun ,au plus, deux éléments.
On sait que pour coder une table à 7 états il faut trois bits.
Deux de ces bits peuvent servir pour distinguer les différents blocs , et le troisieme bit pour distinguer les deux éléments d'un même bloc.
Il ya trois façons distinctes pour effectuer ce codage. On choisira alors celle qui aboutira à des équations simples .

CONCLUSION:

Il resort de cet e xposé que nous venons de faire que la méthode servant à obtenir une partition substitutive est un procédé par tatônnements , basée sur la définition de la propriété de substitution. De plus , pour n'importe quelle machine séquentielle, il peut exister plus d'une partition substitutive .
Il n'ya pas de procédé simple pour obtenir partition substitutive unique, qui donne un codage minimal.
Notons enfin , que pour une machine complétement spécifiée(c'est-à dire dont la table de fluence est complétement spécifiée) , les procédés de minimisation et de codage peuvent etre combinés en un seul.

4.5. CODAGE DES ETATS PAR LA METHODE DE DOLOTTA -Mc. CLUSKEY.

C'est cette méthode que nous avons programmé. Nous avons établi deux programmes :

- l'un nous donne les différentes colonnes codables
- l'autre nous donne l'évaluation de ces colonnes dans un systeme de numération . octal.

Cette méthode est basée sur l'idée d'une colonne codable.
Pour une table des états à r lignes , une colonne codable est une colonne comportant des "0" et des "1" et qui a les caractéristiques suivantes:

a). Elle est de longueur r
b). Elle a un "0" en 1° position.
c). Elle n'a pas plus de 2^{n-1} "0".
d). Elle n'a pas plus de 2^{n-1} "1".

n désigne le nombre de variables d'états requises , définis par:

$$2^{n-1} < r \leqslant 2^n$$

Pour des valeurs de n et de r données ,il faut calculer le nombre de colonnes codables . Ce nombre est dans tous les cas ,inférieur au nombre de codes distincts Nd donné par le tableau 1.
En définissant des critères pour le choix d'un sous-ensemble approprié de colonnes, l'importance du calcul nécessaire pour aboutir à un bon codage sera considérablement réduit.
Pour une table des états donnée ,les colonnes codables peuvent etre représentées sous la forme d'une matrice de base à r lignes et C(r) colonnes.
De plus , on peut identifier chaque état interne de la table à une ligne de la matrice de base selon une correspondance bijective.
On peut alors représenter, de la même maniere , les états futurs pour chaque état d'entrée.

Pour la commodité du calcul , nous avons représenté les colonnes codables dans le système de numération octal. Le chiffre le moins significatif est placé au bas de la colonne ; on partage la colonne en groupes de trois bits équivalents à un chiffre octal. Les colonnes qui ont un "1" en tête , sont représentéss sous la forme complémentée, qui est obtenue en inversant la colonne en question. (c'est-à-dire en remplaçant les "1" par des "0" et vice-versa) et en exprimant le résultat comme un nombre octal négatif.
Le nombre de colonnes codables pour une table à r lignes est:

$$C(r) = A_{r-1}^{MAX} + A_{r-1}^{MAX-1} + \dots + A_{r-1}^{MIN}$$

avec:

$$A_{r-1}^{MAX} = \frac{(r-1)!}{(MAX)!\ (\ (r-1)-MAX\ )!}$$

$$MAX = 2^{n-1} - 1$$
$$MIN = (r-1) - 2^{n-1}$$

Le premier travail a consisté donc à établir un organigramme permettant de trouver toutes les colonnes codables d'une table donnée. On obtient alors la matrice de base à r lignes et C(r) colonnes .
Le deuxieme travail a consisté à évaluer les différentes colonnes de la matrice de base ainsi que des matrices des états futurs , dans le système de numération octal.
Cette évaluation va nous permettre de définir les critères de choix des colonnes qui vont servir pour le codage de la table des états .
Ainsi pour chaque ligne du tableau nous comparons la donnée de la matrice de base avec les données des états futurs et attribuons une évaluation ,conformément aux critères établis par DOLOTTA et Mc CLUSKEY:

1. Toute donnée composée exclusivement de "0" vaut 20 points.
2. Toute donnée composée exclusivement de" 1 " vaut 10 points.
3. Des données identiques dans des colonnes adjacentes (différentes d'une seule valeur des variables x) valent 20 points.
4. Une donnée identique à celle de la matrice de base vaut 4 points .
5. Une donnée identique au complément d'une donnée de la matrice de base vaut 4 points.

Les évaluations sont exprimées dans le système de numération octal. On compare donc les différentes évaluations , et on choisit le vecteur de base dont l'évaluation est la plus élevée , comme premiere colonne codable .

Le choix de cette premiere colonne a pour conséquence la modification de la procédure d'évaluation nécessitée par la recherche de la deuxieme colonne .

a. L'existence d'une donnée correspondant au vecteur de base d'une colonne choisie vaut 4 points .
b. L'existence de données similaires à celles qui apparaissent dans la colonne déjà choisie vaut 20 points.

Une fois les colonnes choisies , il faut vérifier qu'elles constituent un code valide .

Pour valider un code , les colonnes codables doivent satisfaire les conditions suivantes :

a. Pour chaque paire de colonnes , C1 etC2 d'un ensemble
   (C1 C2 ... Cn)
   le nombre de "1" obtenus en effectuant l'opération "ET" entre toutes les combinaisons des colonnes ($\bar{C}1\bar{C}2$, $C1\bar{C}2$, $\bar{C}1C2$, $C1C2$) doit etre inférieur ou égal à 2^{n-2} dans chaque cas.
b. Pour valider un code complet , il faut effectuer la même opération sur toutes les colonnes , dans toutes les combinaisons.
   Pour trois colonnes nous aurons:
   ($\bar{C}1\bar{C}2\bar{C}3$, $\bar{C}1\bar{C}2C3$, $\bar{C}1C2\bar{C}3$, $\bar{C}1C2C3$, $C1\bar{C}2\bar{C}3$, $C1C2C3$; $C1\bar{C}2C3$, $C1C2\bar{C}3$)
   et la somme des "1" dans chaque cas , doit être inférieure ou égale à : 2^{n-K}
   avec:
   — n = nombre de variables d' états .
   — K = nombre de colonnes.

ORGANIGRAMMES.

PROCEDE.

Soit une table possedant NE états. Le premier programme nous permet de trouver l'ensemble des colonnes codables .

PREMIER PROGRAMME.

D'après la définition des colonnes codables ,toute colonne commence par un zero ; les NE-1 autres positions sont constituées par des combinaisons de "1" et de "0".

Le nombre maximal de 1 dans une colonne est donnée par :

$$MAX = 2^{N-1}$$

N étant le nombre de variables necessaires pour coder les NE états.
(NE inferieur ou egal à 2^N)

Le nombre minimal de 1 dans une colonne est :

$$MIN = NE - MAX$$

Pour chaque nombre de 1 (NU avec $MIN \leqslant NU \leqslant MAX$) il faut trouver l'ensemble des combinaisons possibles de ces N U "1" parmi les N E-1 positions .

Pour chaque valeur de N U on cherche les différentes positions de NC(I,J) des chiffres 1 parmi les NE-1 positions .

NC(I,J) : J= 1,... NU

I indique la colonnne codable.

Connaissant les différentes positions NC(I,J) des chiffres 1 pour chaque colonnd I , on peut alors écrire la colonne codable I donnée par C(K,I)

avec: I = 1,... NCC . NCC = nombre de colonnes codables

K = 1,...NE.

SYMBOLES UTILISES.

N E = nombre d'états internes de la table à coder.

NVAR = nombre de variables nécessaires au codage des NE états.
(NE inferieur ou egal à 2^{NVAR})

MAX= nombre de chiffres 1 dans une colonne .

MIN = nombre minimal de chiffres 1 à classer dans une colonne.

N U = nombre de chiffres 1 à classer parmi les NE-1 positions

B(I) : nombre de chiffres 1 dans la colonne codable I.

NC (I,J) : indique la position des chiffres 1 dans la colonne codable I.

$1 \leq NC(I,J) \leq NE-1$

I indique la colonne codable.

J indique les différents chiffres 1; J = 1,...NU.

C(I,J) : tableau qui donne les différentes colonnes codables.

l'indice J indique la colonne codable .

l'indice I indique chacune des positions de la colonne.

NCC : nombre de colonnes codables.

2°. PROGRAMME.

Connaissant les colonnes codables , ce programme nous permet de calculer la valeur octale de chaque colonne de la matrice de base et des matrices des entrées.

Le chiffre le moins significatif est placé en dernier (soit en bas de la colonne). On a partage la colonne en groupe de trois bits équivalents à un chiffre octal.

Les colonnes qui ont un 1 en tête sont représentées sous la forme complémentaire : on inverse la colonne en remplaçant les 1 par des 0 et vice-versa. Le résultat dans ce cas est exprimé comme un nombre octal négatif .

On identifie chaque état interne de la table à une ligne de la matrice de base .

Pour chacune des entrées X de la machine séquentielle, on associe une matrice MAT(I,J) dont chaque ligne I (correspondant à l'état futur) est constituée par la ligne K de la matrice de base tel que I soit le successeur de l'état K pour l' entrée X.

SYMBOLES UTILISES.

C(I,J) : tableau qui nous fournit la matrice de base .
I = 1,... NE
J = 1,...NCC

G(I,J) : table des transitions ou de fluence.
I = 1, ... NE
J = 1,... NVE (avec NVE = nombre de vecteurs d'entrée

MAT(I,J) : tableau qui ni indiquent les différentes matrices: matrice de base et matrice correspondant à chacune des entrées.

Y(L) : tableau qui permet de receuillir chacun des chiffres "S" du nombre octal.

NVO(J,N) : tableau qui indique la valeur octale de chacune des des colonnes J.
J = 1,... NCC
N = 1,... NVE+1.

RESULTATS OBTENUS.

Pour le premier programme, nous avons choisi une table de fluence à 7 états internes. Les colonnes codables obtenues sont représentées dans le tableau 4.7.

Pour le deuxieme programme ,qui donne l'évaluation des différentes colonnes dans le système de numèration octal ,nous avons choisi l'exemple suivant :

Table de fluence:

ETATS PRESENTS	ETATS FUTURS	
	00	01
1	2	3
2	6	4
3	4	4
4	5	5
5	1	1
6	7	5
7	1	1

La matrice de base des colonnes codables, correspondant à cette table de fluence ,est donnée par le tableau 4.8.
L'estimation (c'est-à-dire l'évaluation)des différentes colonnes de la matrice de base ainsi que des matrices de transitions,dans le système de numération octal ,est donnée par le tableau 4.9.
Finallement, la table codée est représentée par le tableau 4.10

CONCLUSION.

De toutes les methodes concernant le codage (étude des adjacences, paires substitutives, couple de paires substitutives), la méthode de DOLOTTA-Mc.CLUSKEY est la plus systématique et s'adapte mieux à la programmation sur calculateur. Bien qu'elle n'aboutit pas à la solution optimale, elle donne des résultats très satisfaisants.

Cependant,si la table possède un grand nombre d'états internes (plus de 20) l'encombrement mémoire de l'ordinateur est important.

COLONNES codables

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1

0 0 0 0 1 1 1 1 1 1 0 0 0 1 0 0 1 0 1 1 0 1 1 1 1 0 0 1 0 1 1 0 1 1 1

0 1 1 1 0 0 1 0 1 1 0 0 1 0 0 1 0 1 0 1 1 0 1 1 1 0 1 0 1 0 1 1 0 1 1

1 0 1 1 0 1 1 1 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 0 0 1 1 0 1 1 0 1

1 1 0 1 1 0 0 1 1 0 1 0 0 0 1 1 1 0 0 0 1 1 1 0 1 1 1 1 0 0 0 1 1 1 0

1 1 1 0 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0

NOMBRE : 35

TABLEAU 4.7. Resultats fournis par l'ordinateur pour une table possedant 7 états internes

TABLEAU 4.8.

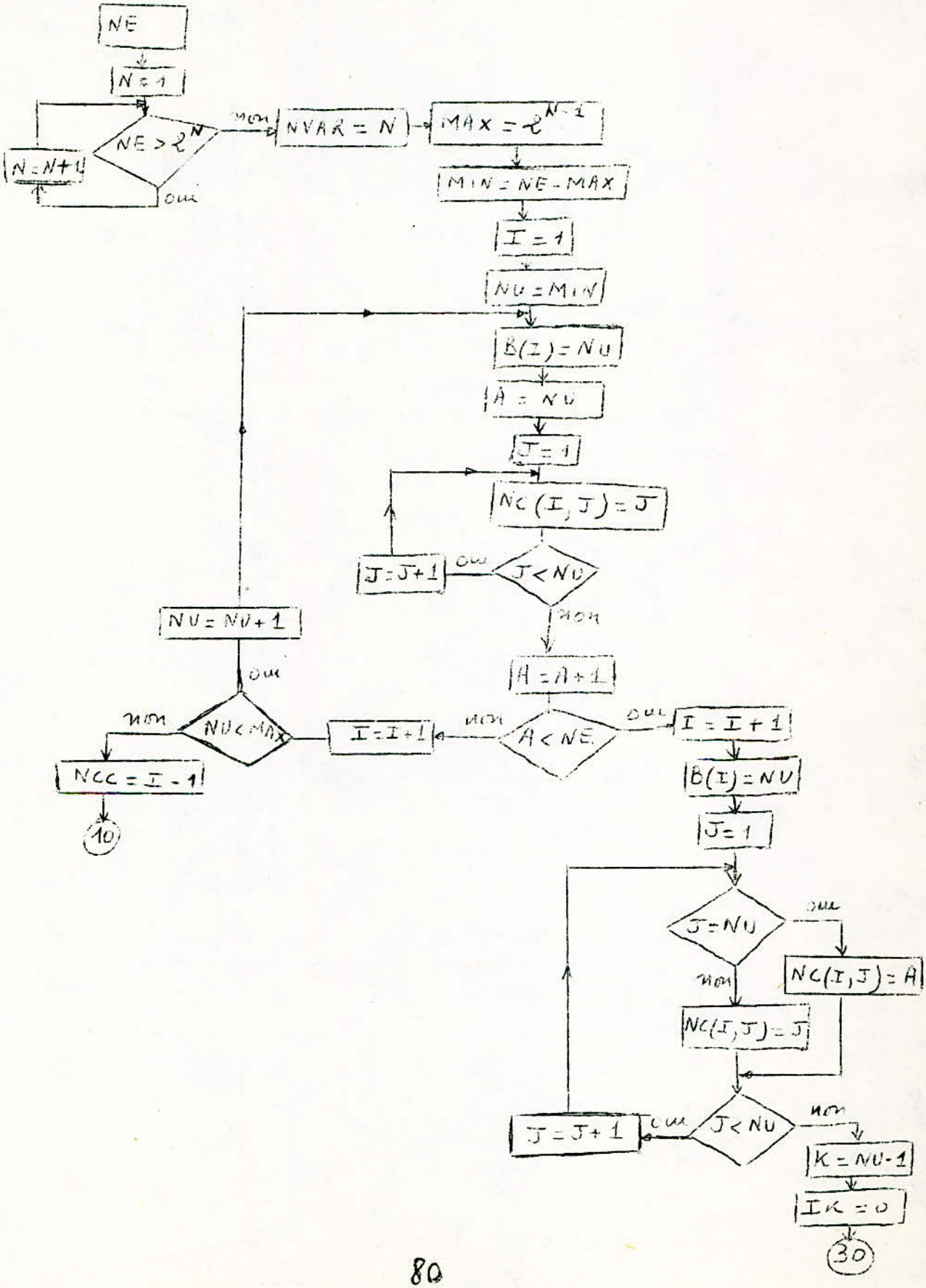
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1	0	0	0	0	0	0	0	0	0	0	0	0	0	00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
4	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1
5	1	0	1	1	1	0	1	1	1	0	0	0	1	1	1	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0	1
6	1	1	0	1	1	1	0	1	1	0	1	1	0	0	1	1	0	1	1	0	1	1	0	0	1	0	1	1	0	0	1	0	0	1	0
7	1	1	1	0	1	1	1	0	1	1	0	1	0	1	0	1	1	0	1	1	0	1	0	1	0	1	0	1	0	1	0	0	1	0	0

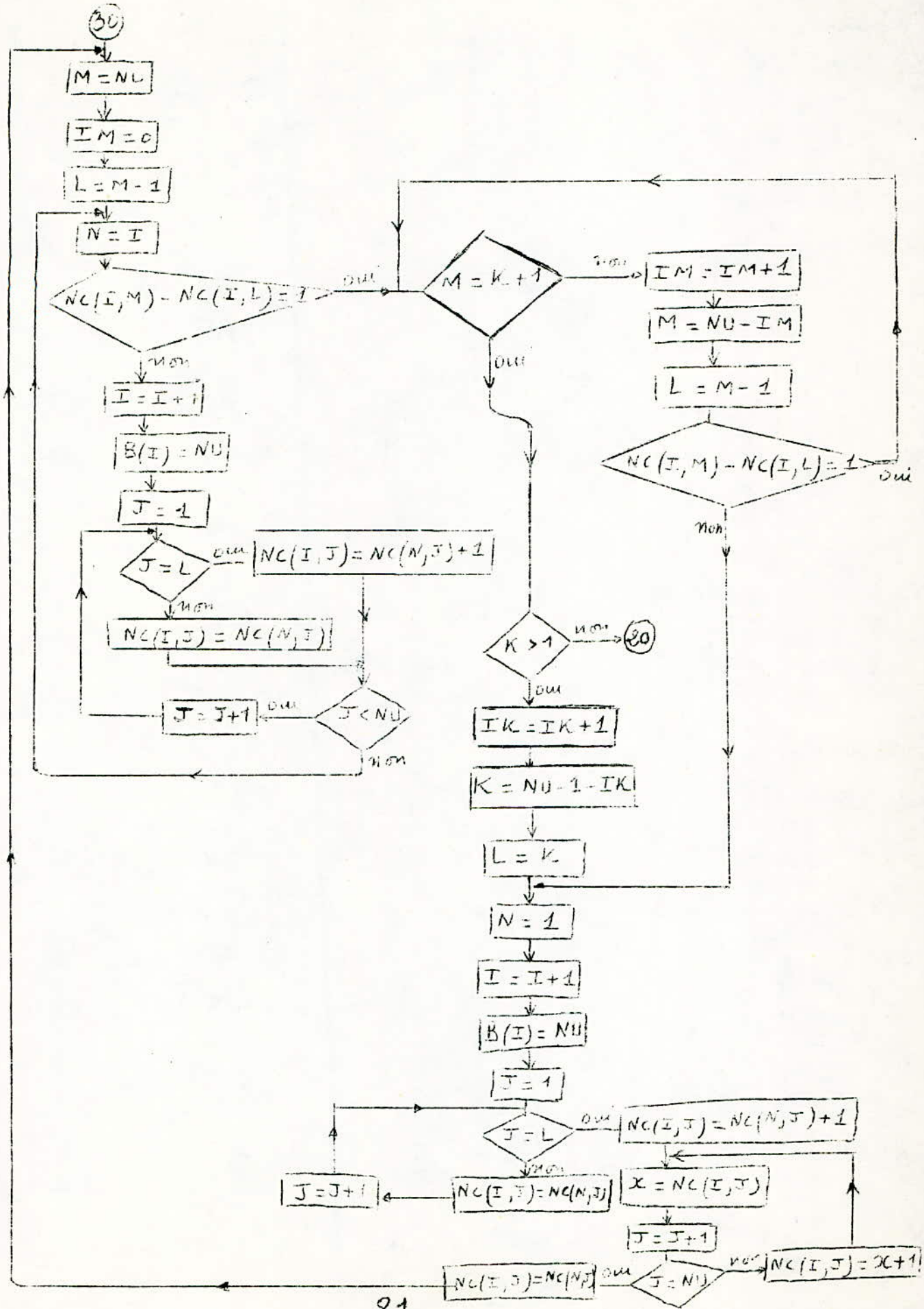
77

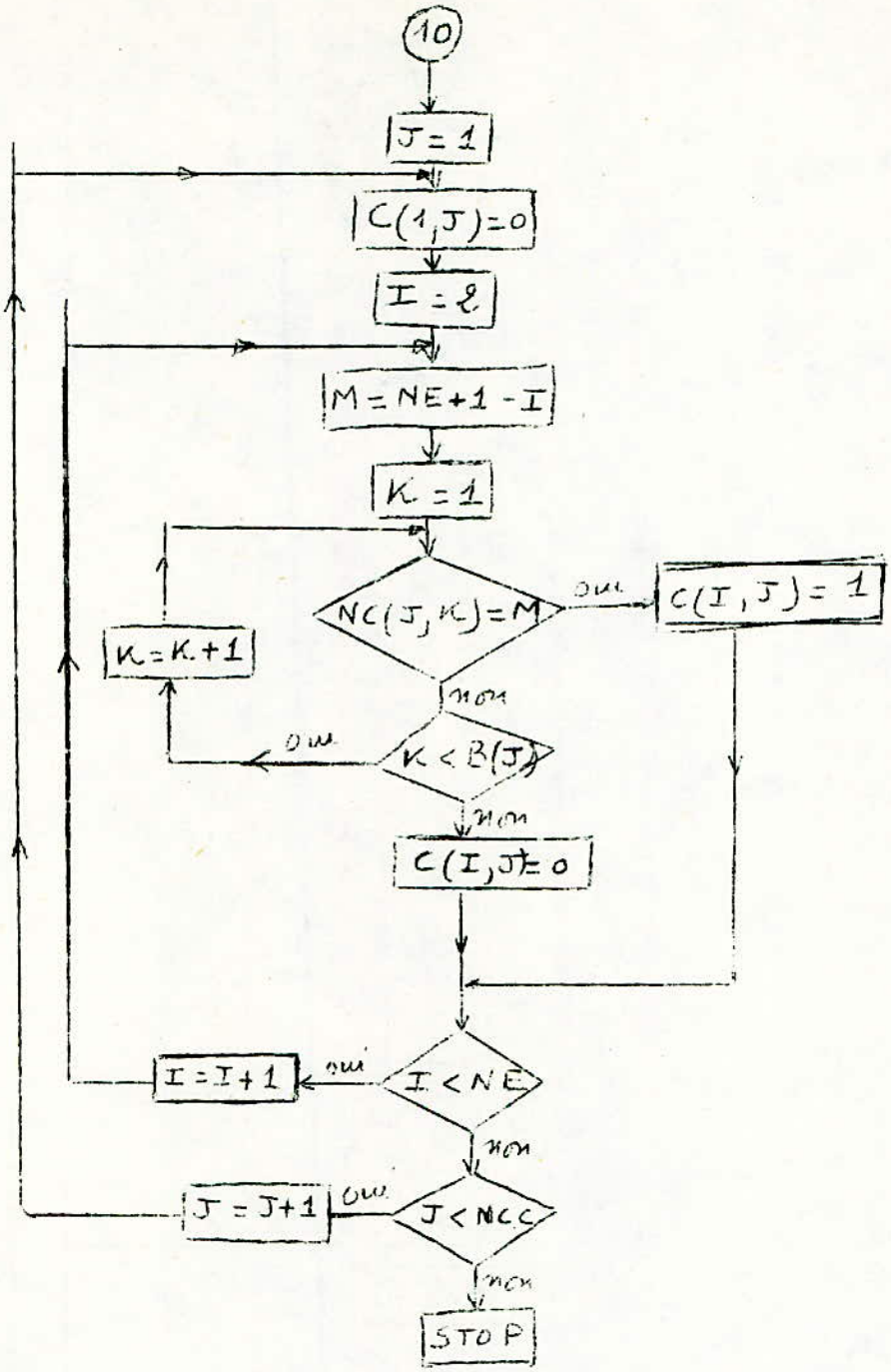
Tableau 4.10. (table codée)

	Etats présents			Etats futurs	
	y_1	y_2	y_3	00	01
1	0	0	0	101	100
2	1	0	1	011	010
3	1	0	0	010	010
4	0	1	0	110	110
5	1	1	0	000	000
6	0	1	1	111	110
7	1	1	1	000	000

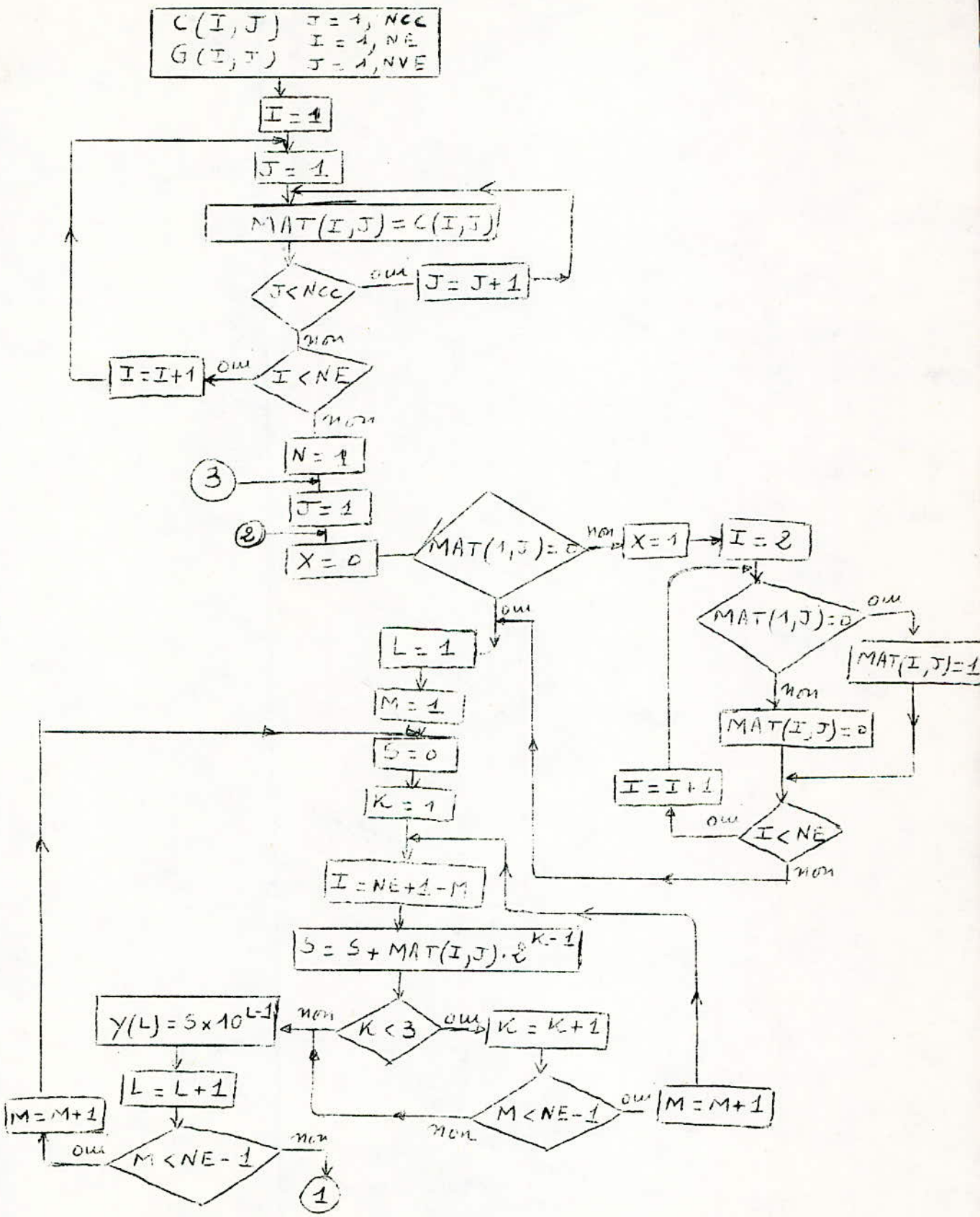
Organigramme : Codage 1ère partie

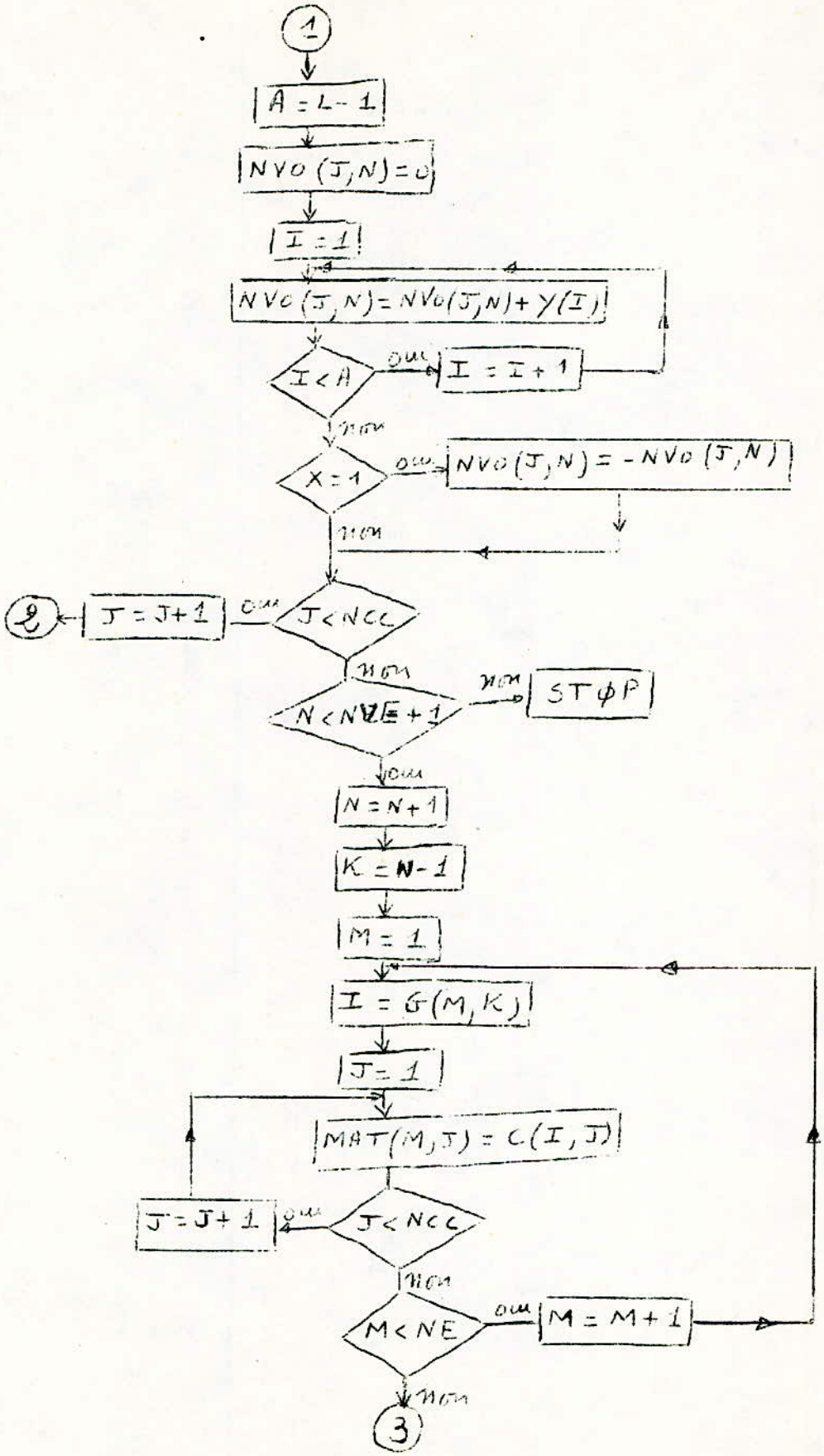



30
M = NU
IM = 0
L = M - 1
N = I
NC(I,M) - NC(I,L) = 1
oui
non
M = K + 1
oui
non
IM = IM + 1
M = NU - IM
L = M - 1
NC(I,M) - NC(I,L) = 1
oui
non
I = I + 1
B(I) = NU
J = 1
J = L
oui
NC(I,J) = NC(N,J) + 1
non
NC(I,J) = NC(N,J)
J = J+1
oui
J < NU
non
K > 1
non
20
oui
IK = IK + 1
K = NU - 1 - IK
L = K
N = 1
I = I + 1
B(I) = NU
J = 1
J = L
oui
NC(I,J) = NC(N,J) + 1
non
NC(I,J) = NC(N,J)
J = J+1
X = NC(I,J)
J = J+1
J = NU
oui
NC(I,J) = NC(N,J)
non
NC(I,J) = X+1


10
J = 1
C(1,J) = 0
I = 2
M = NE+1 - I
K = 1
NC(J,K) = M
oui
C(I,J) = 1
non
K = K+1
ou
K < B(J)
non
C(I,J) = 0
I = I+1
oui
I < NE
non
J = J+1
ou
J < NCC
non
STOP

Organigramme : Codage 2e partie




1
A = L - 1
NVO (J,N) = 0
I = 1
NVO (J,N) = NVO(J,N) + Y(I)
I < A
oui
I = I + 1
non
X = 1
oui
NVO (J,N) = - NVO (J,N)
non
2
J = J + 1
oui
J < NCC
non
N < NVE + 1
non
STOP
oui
N = N + 1
K = N - 1
M = 1
I = G(M,K)
J = 1
MAT(M,J) = C(I,J)
J = J + 1
oui
J < NCC
non
M < NE
oui
M = M + 1
non
3

PROGRAMME FORTRAN

Codage des états 1ère partie

```
      NE =
      N = 1
3     IF (NE - 2**N) 1, 1, 2
2     N = N+1
      GOTO 3
1     NVAR = N
      MAX = 2**(N-1)
      MIN = NE - MAX
      I = 1
      DO 4 NU = MIN, MAX
      B(I) = NU
      A = NU
      DO 5 J = 1, NU
5     NC (I, J) = J
28    A = A + 1.
      IF (A - NE) 6, 7, 7
6     I = I + 1
      B(I) = NU
      DO 8 J = 1, NU
      IF (J - NU) 9, 10, 8
9     NC (I, J) = J
      GOTO 8
10    NC(I,J) = A
8     CONTINUE
C COMBINAISONS pour cette valeur de A
      K = NU - 1
      IK = 0
27    M = NU
      IM = 0
      L = M - 1
17    N = I
      IF (NC(I,M) - (NC (I,L) + 1)) 12, 13, 12
12    I = I + 1
      B(I) = NU
      DO 14 J = 1, NU
      IF (J - 1) 13, 16, 15
15    NC (I,J) = NC (N,J)
```

```
      GOTO 14
16    NC(I,J)=NC(N,J)+1
14    CONTINUE
      GOTO 17
13    IF(M-(K+1))18,19,18
18    IM=IM+1
      M=NU-IM
      L=M-1
      IF(NC(I,M)-(NC(I,L)+1))20,13,20
20    N=I
      I=I+1
      B(I)=NU
      J=1
23    IF(J-L)21,22,21
21    NC(I,J)=NC(N,J)
      J=J+1
      GOTO 23
22    NC(I,J)=NC(N,J)+1
26    X=NC(I,J)
      J=J+1
      IF(J-NU)24,25,24
24    NC(I,J)=X+1
      GOTO 26
25    NC(I,J)=NC(N,J)
      GOTO 27
19    IF(K-1)28,28,29
29    IK=IK+1
      K=(NU-1)-IK
      L=K
      GOTO 20
7     I=I+1
4     CONTINUE
      NBC=I-1
C     ECRITURE DES COLONNES
      DO 30 J=1,NBC
      C(I,J)=0
      DO 31 I=2,NE
```

```
      M = NE + 1 - I
      K = 1
36    IF (NC(J,K) - M) 33, 32, 33
33    IF (K - B(J)) 34, 35, 35
34    K = K + 1
      GOTO 36
35    C(I,J) = 0
      GOTO 31
32    C(I,J) = 1
31      CONTINUE
30      CONTINUE
      WRITE (102, 50) ((C(I,J), J = 1, NBC), I = 1, NE)
50    FORMAT (18H COLONNES CODEBLES /7(2X, 35I2,
      STOP
      END
```

PROGRAMME FORTRAN : Codage 2e partie

```
1       7
        DIMENSION MAT(7,35), NVØ(35,5)
        INTEGER C(7,35), Y(2), G(7,2), X, S, A
        NE = 7
        NCC = 35
        NVE = 2
        READ(105,40)((G(I,J),J=1,NVE),I=1,NE)
40      FØRMAT(2X,14I2)
        READ(105,50)(C(I,J),J=1,NCC),I=1,NE)
50      FØRMAT(2X,35I2)
        DØ 1 I=1,NE
        DØ 1 J=1,NCC
1       MAT(I,J)=C(I,J)
        N=1
21      DØ 2 J=1,NCC
        X=0
        IF(MAT(1,J))3,4,3
3       X=1
        DØ 5 I=2,NE
        IF(MAT(I,J))6,7,6
6       MAT(I,J)=0
        GØTØ 5
7       MAT(I,J)=1
5       CØNTINUE
4       L=1
        M=1
14      S=0
        K=1
11      I=(NE+1)-M
        S=S+MAT(I,J)*2**(K-1)
        IF(K-3)8,9,9
8       K=K+1
        IF(M-(NE-1))10,9,9
10      M=M+1
        GØTØ 11
9       Y(L)=S*10**(L-1)
```

```
1       7
        L=L+1
        IF(M-(NE-1)) 12, 13, 13
12      M = M+1
        GOTO 14
13      A = L-1
        NVO(J,N) = 0
        DO 15 I = 1, A
15      NVO(J,N) = NVO(J,N) + Y(I)
        IF(X) 16, 2, 16
16      NVO(J,N) = - NVO(J,N)
2       CONTINUE
        IF(N-(NVE+1)) 17, 18, 18
17      N = N+1
        K = N-1
        DO 19 M = 1, NE
        I = G(M,K)
        DO 20 J = 1, NCC
20      MAT(M,J) = C(I,J)
19      CONTINUE
        GOTO 21
18      NA = NVE+1
        WRITE(108,60)(J,(NVO(J,N),N=1,NA),J=1,NCC)
60      FORMAT(14H VALEUR OCTALE//4(5X,I4))
        STOP
        END
```

CONCLUSION.

1. STRUCTURES DES ETAPES.

Nous avons etabli six programmes qui nous permettent à partir d'un énoncé de probleme séquentiel synchrone ,d'aboutir à une table de fluence réduite et codée de la machine.

Le 1^{e} programme concerne la synthese des tables: à partir des caractéristiques physiques de la machine séquentielle qu'on désire réaliser, on traduit l'énoncé sous forme de correspondance entre séquences d'entrée et séquences de sortie. Le programme nous permet d'établir la table des transitions et la table des sorties.

La 2° étape consiste à réduire les tables ainsi trouvées. Cette réduction est obtenue à l'aide de trois programmes .

- Le premier programme concerne la recherche des paires d'états compatibles.
- Le second programme concerne la recherche des compatibles maximaux. Ce programme n'est nécessaire que dans le cas des tables incomplètements spécifiées.
- Ayant les différents compatibles maximaux,le troisième programme permet de rechercher une couverture minimale et finie de la table initiale.

A partir de la table initiale T et connaissant les compatibles choisis de façon à constituer une couverture minimale et finie , l'étape suivante consiste à réécrire la table réduite T'. Cette phase n'a pas été programmée ,du fait qu'elle se fait très facilement à la main.

Ces quatre premiers programmes peuvent etre regroupés ensemble.

Le 5° et Le 6° programme que nous avons établis concernent le codage des états . A partir de la table réduite ,on recherche un assignement des états qui génère un code valable et qui permet d'aboutir à des équations de sortie simples et réduites . Ceci nous permet alors d'écrire la table codée de la machine séquentielle .

Ayant obtenu cette table codée ,il suffit alors d'en tirer les équations logiques des excitations secondaires et des sorties. Le reste du travail qui consiste à minimiser ces équations logiques est un problème de logique combinatoire . A ce stade de la synthèse de la machine séquentielle synchrone la théorie actuelle sur l'automatisation des calculs est très développée.

Les étapes à suivre sont alors les suivantes:

- A partir de la table codée ,tirer les équations logiques des excitations secondaires et des sorties sous forme d'expressions numériques (ex. F=R(0,2,5,6,7)).
- Calculer l'ensemble des implicants premiers par la méthode de Quine-Mc. Cluskey (on trouvera un algorithme de Mc. Cluskey -Harrison dans la revue Automatisme n° 5 de mai 1974,page 297)
- Réduire aux formes minimales en choisissant les sous-ensembles des implicants premiers .
- Matérialiser l'ensemble des équations de sorties en tenant compte du type de logique et des portes disponibles (Nand,NOr ...)

La succession des différentes étapes de la synthèse des machines séquentielles synchrones peut être résumée par l'organigramme de la fig.1.

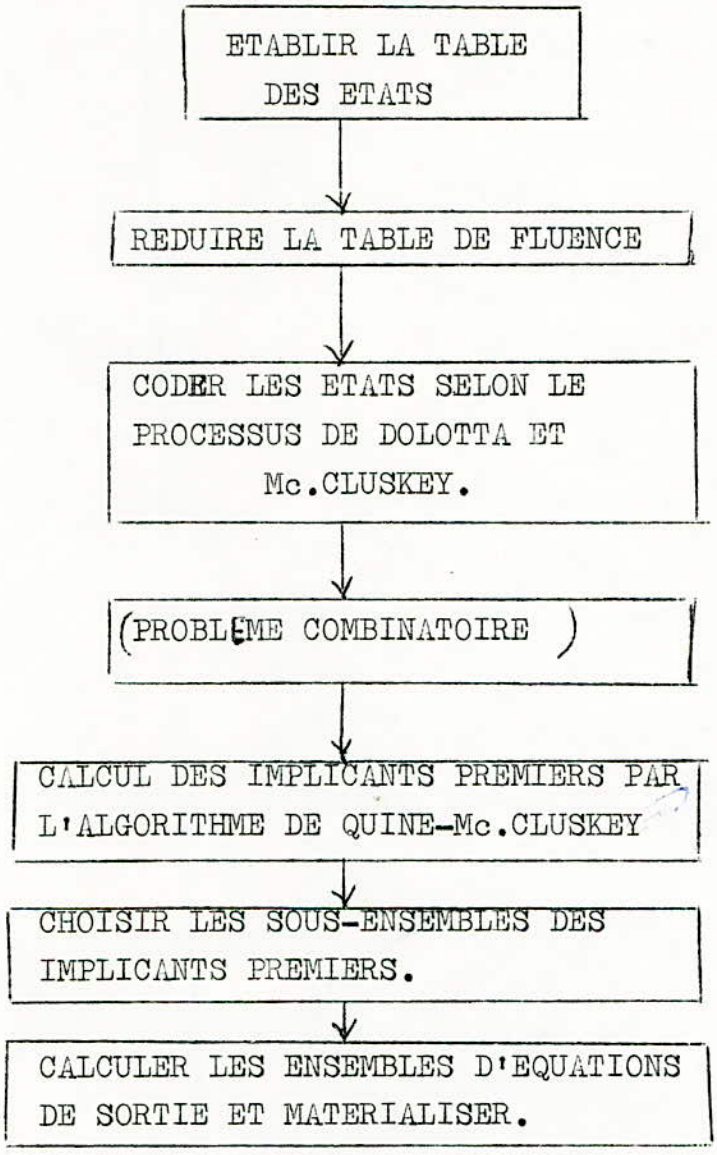


Fig. 1

2. PROBLEMES DE L'AUTOMATISATION DU CALCUL DES SYSTEMES LOGIQUES

Actuellement le probleme du calcul automatique des systemes de logique séquentielle ,est au stade de la recherche .

Les simplifications des systemes logiques requiert beaucoup de calculs. Pour une machine séquentielle possédant un assez grand nombre d'états internes nous sommes obligés de fractionner le travail en plusieurs étapes successives.

Sans cela le temps de passage sur ordinateur deviendrait prohibitif ,et nous risquons d'atteindre rapidement un encombrement de mémoire. Le problème de l'assignement des états requiert à lui seul un programme . assez long et un emplacement mémoire élevé.Dans le cas d'une machine possédant plus d'une vingtaine d'états intenes le risque de dépassement de la capacité de mémoire de l'ordinateur est grand.

Langage de programmation.

Nous avons utilisé le FORTRAN;l'inconvénient majeur de ce langage est qu'il se prête beaucoup mieux au calcul numérique qu'au calcul binaire.

D'ailleurs le langage machine de la plupart des ordinateurs actuels n'est pas adapté au problème du calcul logique .Il faudrait disposer d'une machine organisée selon le système de base 2 et permettant une manipulation directe des bits selon les règles du calcul binaire.

Des machines spéciales permettant un calcul automatique plus approprié des systèmes logiques ont été construites. Ainsi en Belgique ,à l'université de Bruxelles, J. Florine a réalisé une calculatrice spécialisée pour la simplification d'une fonction de huit variables au plus (selon la méthode de Mc. Cluskey)

Pour une machine séquentielle complexe ,il n'ya aucune raison qui empêche de décomposer la synthèse en un certain nombre d'étapes distinctes telles que la synthèse des tables ,la réduction des états ,le codage des états, la simplification des équations logiques,etc... L'enchainement des différentes étapes se fait alors sous un contrôle permanent . Des calculs intermédiaires peuvent être réalisés à la main ,ce qui permet un gain en temps très apprécié.

3. MODES DE REALISATION DES MACHINES SEQUENTIELLES.

Notre étude a porté sur la synthèse des machines séquentielles .PAr une succession d'étapes nous avons abouti à des équations de sorties simples et techniquement faciles à réaliser . Cette réalisation peut se faire sous forme cablée ou programmée.

3.1. MACHINE CABL2E.

Ayant obtenu les différentes équations de sortie, la matérialisation technique se fait en fonction des éléments disponibles: portes (Nand,Nor, ET, OU, OU exclusif,...) à circuits intégrés, à transistors, à relais électromagnétique...

3.2. MACHINE PROGRAMMEE.

Il deux sortes de réalisation:

-- l'automate à micro-ordinateur.

-- l'automate programmable.

3.2.1. L'AUTOMATE A MICRO -ORDINATEUR.

Le problème séquentiel est mis sous forme de programme qu'on charge dans le processeur d'un micro ordinateur.

On peut parler ici par exemple ,d'un séquenceur à logique programmable MICRALSPA construit par la société Alspa . C'est un produit conçu autour du micro ordinateur français Micra (R2E)

3.2.2. L'AUTOMATE PROGRAMMABLE.

C'est une armoire modulaire ,programmable, dont la mise en oeuvre peut être assurée par des automaticiens.

Caractéristiques principales/

Mise en mémoire du programme d'automatisation sous forme de séquences.

Fonctionnement général:

L'automate programmable a pour fonction de donner des ordres ,émettre des commandes en tenant compte , d'une part de l'état de l'installation à automatiser et ,d'autre part, du programme enregistré.

Tout ce dont l'utilisateur a ~~besoin~~ doit se soucier c'est d'établir le programme et connecter les entrées et sorties L'automate programmable fait lui même les traductions nécessaires à son fonctionnement.

principaux avantages:

- la programmation se fait en langage clair d'automaticien (on n'utilise pas que des opérations logiques :ET,OU,NON)
- l'introduction et la modification eventuelle du programme est instantannée.
- L'équipement standard prévoit des sorties pour supervision par ordinateur.
- Le coût de revient dans la cas d'une grande instalation est assez faible.

Ces différents avantages font que l'automate programmable est la solution d'avenir; il tend à remplacer les grands ensembles d'automatisation séquentielle à relais électromagnétique ou à circuits logiques transistorisés.

3.2.3. LANGAGES DE PROGRAMMATION.

Pour la réalisation d'une machine programmée on utilise des langages d'automatismes industriels destinés à des automaticiens non informaticiens.

Parmi les différents langages élaborés par différentes sociétés nous pouvons citer:

-le FORTRAN en temps réel

-les langages dérivés du PL1

-le PROCOL:langage évolué établi par DGRST en France.

-PROSEL, PROSPRO: développé par I B M

-BICEPS :développé par Genéral Electric

-AUTOL : développé par SOdeteg 6 TAI;

BIBLIOGRAPHIE.

J. CHINAL. TECHNIQUES BOOLEENNES ET CALCULATEURS ARITHMETIQUES.

J.P. PERRIN. ; M. DENOUETTE; E. DACLIN;
SYSTEMES LOGIQUES. TOME 2.

D. LEWIN, L. MARET. SYSTEMES LOGIQUES.

P. NASLIN. CIRCUITS LOGIQUES ET AUTOMATISMES A SEQUENCES.
(DUNOD.)

J. KUNTZMANN, P. NASLIN. ALGEBRE DE BOOLE ET MACHINES LOGIQUES.

J.FLORINE. LA SYNTHESE DES MACHINES LOGIQUES .

AUTOMATISME . N° 5 MAI 1974 (PAGE 297)
(ALGORITH ME DE MAX CLUSKEY -HARISSON.)