MASTER EN SCIENCES INFORMATIQUES

Analyse automatique d'un problème par tables de décision et équations logiques
Essai sur la cinématique des fichiers

de Cocquéau des Mottes, Emmanuel

Award date:
1973

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX - NAMUR

INSTITUT D'INFORMATIQUE

ANNEE ACADEMIQUE 1972-1973

Analyse automatique d'un problème par tables de décision et équations logiques

Essai sur la cinématique des fichiers

Emmanuel de Cocquéau des Mottes

Mémoire présenté en vue de l'obtention
du grade de Licencié et Maître en Informatique

JURY DU MEMOIRE :

M. J. BRUNIN

Il nous est particulièrement agréable de remercier ici toutes les personnes qui ont permis la réalisation de ce travail.

Notre reconnaissance s'adresse tout particulièrement à Monsieur le professeur J. BRUNIN dont l'érudite compétence autant que la bienveillante attention ont été les éléments moteurs de notre recherche.

Nous tenons à exprimer également toute notre gratitude :

à Messieurs les professeurs de la section d'Informatique pour leur enseignement dont ce mémoire est l'aboutissement.

à Monsieur le professeur CHARLES qui nous a très aimablement accueilli à l'université de Montpellier ainsi qu'à Madame LAVIT et Monsieur FILLIATRE à qui nous devons de précieux renseignements.

à Monsieur VERBRAEKEN qui par des conseils attentifs nous a permis d'entrer toujours plus profondément dans notre sujet.

à notre dactylo, Madame Wibin sans qui la réalisation concrète de ces pages n'aurait pas été possible.

Enfin, nous n'oublions pas ceux qui par leur patience et leur collaboration anonyme nous ont aidé à surmonter les difficultés rencontrées.

- - - - - - - -

TABLE DES MATIERES

IIe partie - ESSAI SUR LA CINEMATIQUE DES FICHIERS.

- - - - - - - -

Ière partie : SOFTWARE - TABLES DE DECISION ET EQUATIONS LOGIQUES

Introduction

Un grand nombre de problèmes rencontrés dans la pratique ont des données de nature bivalante et donc transposables immédiatement en table de décision binaire. D'autres problèmes dont les données sont exprimées en logique polyvalente pourraient aisément se ramener au cas classique binaire. L'énorme inconvénient de cette transposition réside dans le fait que l'on accroit considérablement le nombre des variables. Il serait donc intéressant de considérer directement le cas des tables de décision exprimées en logique polyvalente. D'autre part, certains problèmes, par leur énoncé, sont plus aisément transposables en équations logiques. On pourra donc, tout en gardant les avantages que procurent les tables de décision, éviter leurs désavantages, à savoir les difficultés de leur élaboration. Mais ces dernières deviennent difficilement utilisables lorsque le nombre des variables devient important. Les équations logiques évitent cet inconvénient en supprimant automatiquement un grand nombre de cas représentant les incompatibilités logiques soit entre variables (conditions) soit entre décisions. Les équations logiques permettront donc de construire une table la plus petite possible.

1.1. Mise en oeuvre d'un problème sous forme de table.

Une table de décision comporte toujours deux parties : une partie CONDITION et une partie DECISION. Les colonnes verticales de la table représentent les règles, c'est-à-dire une combinaison de conditions suivie des décisions correspondantes.

REGLES ⟶	1 2 3 .....
CONDITIONS	
DECISIONS	

Nous envisagerons dans ce qui suit, uniquement le cas des tables de décisions combinatoires, c'est-à-dire celles pour lesquelles l'ordre dans lequel les conditions sont examinées n'influe pas sur les décisions prises.

a) Table bivalente ou polyvalente.

Une table de décision peut s'exprimer sous différentes formes :

- la table T-1 est une table de nature binaire (bivalente)

"2" signifiant que la condition est présente
"1" signifiant que la condition est absente
"0" signifiant que la condition est indifférente, c'est-à-dire qu'elle n'influe pas sur les décisions à prendre pour la règle en question.

Pour une décision, "1" signifie qu'elle doit être exécutée, "0" qu'elle doit être ignorée.

REGLES ⟶	I	II	III	IV
E_1 = Police catégorie supér.	2	0	0	1
E_2 = N° accident ⟨A	2	1	2	0
E_3 = Ancienneté ⟩B	2	1	1	2
S_1 Prime	1	0	1	0
S_2 Ristourne 5 %	0	0	0	1
S_3 Ristourne 10 %	1	0	1	0

T-1

- la table T-2 est une table polyvalente, c'est-à-dire qu'une condition (une décision) peut prendre un certain nombre de valeurs comprises entre 1 et 9, 0 représentant la condition indifférent

REGLES ⟶	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV	
Type de livre (A) (1) Edition de luxe (2) Edition normal (3) Livre de poche	0	0	0	0	1	1	1	1	1	$\bar{1}$	2	2	3	3	AU
Catégorie de client (B) (1) Détail (2) Administration (3) Représentant édition luxe (4) Représentant édition normale poche (5) Etranger	1	1	2	2	3	3	3	5	5	5	4	4	4	4	
Montant de la commande en F (C) (1) Moins de 10,00 (2) Moins de 500,00 (3) Moins de 1000,00 (4) Plus de 1000,00	$\bar{4}$	4	$\bar{4}$	4	2	3	4	$\bar{4}$	4	1	$\bar{4}$	4	$\bar{4}$	4	
Réduction (R) (1) 0 % (2) 10 % (3) 20 %	1	2	2	3	2	3	3	2	3	2	2	3	1	2	Erre
Frais d'envoi (E) (1) Non (2) Oui	1	1	1	1	2	2	1	2	2	2	1	1	1	1	
Mode de paiement (P) (1) Comptant (2) 30 jours (3) 3 mois	1	2	3	3	3	3	3	2	3	2	2	2	2	2	

T-2

Dans cette table figurent des valeurs complexes. Par exemple C = $\overline{4}$ signifie que C peut prendre les valeurs 1, 2 ou 3.

b) Table limitée ou étendue.

- La table T-3 est dite "étendue" car pour décrire les conditions (ou le décisions), on trouve des opérateurs (signes, ...) et des opérande de valeurs décimales ou symboliques (lettres, ...). Cette dernière peut être traduite de telle sorte qu'elle ne contienne plus que des expressions standards binaires. Elle devient alors une table "limitée" dans l'expression du langage. Par rapport à la table T-3, elle se caractérise non seulement par une multiplication du nombre de lignes, mais également du nombre de colonnes.
Cependant, elle peut également être transposée en table polyvalente (dont la table binaire n'est qu'un cas particulier) sans pour autant accroître les dimensions de celle-ci.

Cond.	a	<5		$5 \leqslant 10$		$\geqslant 10$	
	b	<3	$\geqslant 3$	<6	$\geqslant 6$	<8	$\geqslant 8$
Dec.	P	P_1	P_1	P_2	P_2	P_1	P_3
	R	R_1	0	R_2	0	R_3	0

T-3

Comme nous l'avons vu, les tables de décision, à caractère combinatoire, utilisées pour l'analyse d'un software ne comportent rien de plus que les combinaisons associées "entrée-sortie" enregistrées dans un ordre quelconque.
Ces tables permettent, comme nous le verrons dans la suite par un exemple :

- de générer une programmation automatique de l'organigramme représenté par la table ;
- de présenter à l'analyste ou au responsable, une synthèse claire et précise du problème ;
- d'optimaliser le traitement en machine.

c) Table complète.

Il reste encore une distinction importante à faire :

Une table est complète lorsque le nombre de règles de la table corresp au nombre de cas envisagés. Ainsi, dans la table T-4, il y a 10 cas correspondant aux 10 règles envisagées.

REGLES ⟶	I	II	III	IV	V	VI	VII	VIII	IX	X	XI
C 1. (1) noir (2) galvanisé	1	1	1	1	1	2	2	2	2	2	F
C 2. (1) simple longueur (2) double longueur	1	1	1	2	2	1	1	1	1	2	F
C 3. (1) bout lisse (2) fileté (3) fileté avec raccord (4) fileté avec extrémité	1	2	3	3	3	1	2	3	3	4	F
C 4. (1) faible épaisseur (3) forte épaisseur (2) épaisseur normale	1	2	2	2	3	2	2	2	2	2	F
C 5. (1) non huilé (2) huilé	1	2	2	2	2	1	1	1	1	1	F
C 6. (1) soudé (2) semi-étiré (3) étiré	1	2	2	3	3	1	2	2	3	3	F
C 7. (1) 1" (2) 1 1/2" (3) 2" (4) 2 1/2" (5) 4"	3	4	4	5	5	1	1	2	2	3	F

T-4

d) Table complètement définie.

Une table est complètement définie lorsque, pour chaque cas, toutes les conditions et décisions sont parfaitement fixées.
La table binaire T-5 ci-dessous est complètement définie.

REGLES	I	II	III	IV
C 1	1	2	1	2
C 2	1	1	2	2

T-5

Lorsque la table est complètement définie et complète, le nombre de règles est égal au produit des valences de chaque condition.
Dans le cas idéal où les paramètres sont binaires et le nombre de conditions est n, le nombre de cas sera 2^n.

Voyons maintenant, à l'aide d'un exemple, quels sont les avantages résultant de la transposition d'un problème sous forme de table.

Le problème à traiter est défini par la table T-2.

Cette table se lit de la manière suivante : pour la règle III, quel que soit le type de livre, s'il s'agit d'un client faisant partie de l'administration et que le montant de sa commande ne dépasse pas 1.000 F, effectuer les opérations suivantes :

1) opérer une réduction de 10 %
2) ne pas tenir compte des frais d'envoi
3) le mode de paiement doit se faire endéans les 3 mois

Cette table comporte 14 règles. Cependant, compte tenu des valeurs indifférentes (0) et des valeurs complexes (4 par exemple), on peut considérer que le nombre total de cas posés n'est pas 14 mais 42.

La règle III regroupe par exemple 9 cas, qui sont :

1	2	3	1	2	3	1	2	3
2	2	2	2	2	2	2	2	2
1	1	1	2	2	2	3	3	3

Le nombre total de cas étant 3 x 5 x 4 = 60 cas (produits des valences), 18 cas sont oubliés et pourraient être soumis à l'analyse si nécessaire.

D'autre part, la table pourrait permettre de détecter :

- les cas redondants, c'est-à-dire les cas pour lesquels les conditions et les décisions sont identiques ;

- les cas impossibles pour lesquels on a des conditions identiques pour des décisions différentes.

Nous voyons donc que pour certains problèmes, les tables permett de simplifier et d'assurer plus de garanties à la solution.

A partir de la table amendée, un programme optimisé au mieu grâce à un algorithme sera généré. D'où la grande facilité de programmation pour le programmeur qui n'a plus qu'à écrire les conditions et les décisions, ne devant plus se soucier de l'organigramme du problème.

1.2. Mise en oeuvre d'un programme sous forme d'équations logiques.

Lorsque la constitution des tables devient difficile, on peut l'éviter en passant directement par la résolution algébrique du problème.

Considérons l'énoncé suivant pour lequel la procédure algébrique sera adoptée.

Après le calcul du montant brut d'une facture, une réduction éventuelle doit être calculée.

Les règles de ce calcul sont les suivantes :

- si le client est de catégorie 1 (C_1), il obtient une réduction A (R_A), s'il transporte les marchandises par ses propres moyens (p) ou si ses habitudes de paiement (h) sont bonnes ou si le montant brut de la facture est supérieur à 50.000 F (50). Si deux de ces conditions sont satisfaites, il reçoit en plus une réduction B (R_B). Si les trois conditions sont satisfaites, il reçoit les réductions A, B et C (R_C).

- si le client n'appartient pas à la catégorie 1 (C_1), il obtient une réduction A si le montant brut de la facture est supérieur à 60.000 F (60) et s'il a transporté les marchandises par ses propres moyens.

- si le client n'est pas de catégorie 1 et si le montant brut n'est pas supérieur à 50.000 F, la carte perforée contenant les données de base pour la facturation est erronée. Dans ce cas, la carte doit être déposée dans la case D.

L'énoncé peut être transcrit aisément en un système d'équations, soit :

$$R_A = C_1 (p + h + 50) + \overline{C_1} \cdot 60 \cdot p$$

$$R_B = C_1 \left[p (h + 50) + h (p + 50) + 50 (p + h) \right]$$

$$= C_1 \left[p (h + 50) + 50\, h \right]$$

$$R_C = C_1 \cdot p \cdot h \cdot 50$$

$$D = \overline{C_1} \cdot \overline{50}$$

Nous pouvons également extraire du problème une équation d'incompatibilité entre variables, soit :

$$I\,1 = \overline{50} \cdot 60$$

En effet, si le montant brut de la facture est inférieur à 50.000 F, il ne peut être supérieur à 60.000 F.

Si nous prenons l'inverse de I 1, soit $\overline{I\,1} = \overline{60} + 50$, nous obtenons l'ensemble des cas qui ne sont pas incompatibles. De plus, en multipliant chacune des équations R_A, R_B, R_C et D par $\overline{I\,1}$, nous écartons automatiquement l'ensemble des solutions incompatibles. (D'un point de vue ensembliste, cela revient à faire l'intersection de deux ensembles).

Nous obtenons alors :

$R_A = C_1 (p \cdot \overline{60} + h \cdot \overline{60} + 50) + \overline{C_1} \cdot 60 \cdot 50 \cdot p$

$R_B = C_1 \cdot p \cdot h \cdot (\overline{60} \cdot 50) + C_1 \cdot p \cdot 50 + C_1 \cdot 50 \cdot h$

$R_C = C_1 \cdot p \cdot h \cdot 50$

$D = \overline{C_1}\ \overline{60} . \overline{50}$

Une fois ces cas écartés, il serait intéressant de simplifier au maximum chacune des équations, ce qui permettrait d'emblée de rejeter les redondances.
Ceci ne nous permet pas encore de déceler les cas oubliés. Par "oubliés on pourrait entendre, soit des cas non repris dans l'énoncé du problème, soit des cas non perçus par l'analyste lors de la transposition en équation

Un procédé simple pour les trouver consiste à calculer $f = R_A + R_B + R_C$ et à prendre $\overline{f}$; f représentant l'ensemble des cas existants, $\overline{f}$ est bien la fonction recherchée.

Nous verrons également dans une étude ultérieure comment rechercher les incompatibilités entre équations logiques (entre décisions). Par exemple, on dira que R_A est incompatible avec D si lorsque pour une combinaison de condition, D ne peut prendre en même temps que R_A la valeur 1 et inversément. Cette recherche est importante pour la construction de la table. Elle se fera d'ailleurs à ce moment-là.

Notons pour cet exemple qu'au départ de 32 cas possibles, 8 cas étaient invalides ($\overline{50}$. 60) et ont été éliminés, 17 cas sont examinés dans les équations, 7 cas resteraient donc éventuellement à analyser.

Le table construite à partir de ces équations est la table T-6.

Cas	I	II	III	IV	V	VI	VII	VIII	IX	
C_1	1	1	2	2	2	2	2	2	2	
p	0	2	1	1	1	2	2	2	2	
h	0	0	1	2	2	1	1	2	2	
50	1	2	2	1	2	1	2	1	2	
60	1	2	0	1	0	1	0	1	0	
R_A	0	1	1	1	1	1	1	1	1	
R_B	0	0	0	0	1	0	1	1	1	
R_C	0	0	0	0	0	0	0	0	1	
D	1	0	0	0	0	0	0	0	0	
Nb. cas	4	2	2	1	2	1	2	1	2	1

CHAPITRE 1

APERCU GENERAL DU PROBLEME.

Comme nous l'avons vu dans l'introduction, un problème de décision peut se mettre sous deux formes : soit sous forme de tables de décision, soit sous forme d'équations logiques de décision. De plus, surchacune de ces représentation, l'étude se fait en deux étapes.

1. analyse automatique du problème transposé suivant le cas sous forme de tables de décision ou d'équationslogiques.
2. génération automatique d'un programme représentant la structure (l'organigramme) du problème.

I. 1. Analyse d'un problème sous forme de table de décision.

Les pages précédentes laissaient entrevoir la multiplicité de types de table. Le choix d'une représentation s'imposait donc.
Les principaux critères de ce dernierpeuvent s'énumérer comme suit :

- souplesse plus ou moins grande dans l'analyse automatique,
- facilité d'utilisation de ces tables,
- domaine d'application de celles-ci.

Ce dernier critère semble le plus significatif. En effet, jusqu'à présent, les traducteurs de tables de décision permettaient au prix d'un apprentissage rapide, à tous d'utiliser l'ordinateur, sans passer par l'intermédiaire d'un programmeur. Par contre, le but poursuivi ici n'est pas de supprimer le travail du programmeur, mais de facilite sa tâche en lui décrivant le problème sous forme d'un organigramme optimisé, dépourvu de toute redondance, de toute erreur logique. Nous avons donc choisi les tables polyvalentes qui permettent une grand facilité de mise en oeuvre et dont le domaine d'application est infini.

Supposons maintenant un problème qui aurait p our représentation tabulaire la table T.7

Règles ⟶	1	2	3	4	5	6	7	8	9
C1	0	0	0	0	1	1	1	1	1
C2	1	1	2	2	3	3	3	5	-4
C3	-4	4	-4	4	2	3	4	-4	4
D_1	1	2	2	3	2	3	3	2	3
D_2	1	1	1	2	2	2	1	2	2
D_3	1	2	3	3	3	3	3	2	3

T.7

Cette table comprend 3 conditions, 3 décisions et 9 règles.
Supposons que les conditions C_i et les décisions D_j soient celles définies dans la table T.2 La conditions C_1 peut donc prendre 3 valeurs (1, 2 ou 3), la valeur 0 signifiant que la condition est indifférente; de même C_2 peut prendre 5 valeurs et C_3 4 valeurs. Rappelons encore que $C_i = -k$ signifie que la condition C_i peut prendre toutes les valeurs possibles sauf la valeur k.

La table T.7 n'est pas complète puisque le nombre total (n) de cas pouvant exister est égal à 60 (produit des valences de chaque condition c'est-à-dire 3 x 5 x 4) alors que le nombre de règles est 9.

De plus, elle n'est pas complètement définie puisqu'elle comporte les conditions à valeurs nulles et à valeurs négatives.
Les programmes d'analyse de cette table procède comme suit :

I. 1.1. Vérification des valeurs attribuées aux conditions C_i.

Cela consiste simplement à voir si une valeur quelconque est en valeur absolue inférieure à la valence de la condition à laquelle elle se rapporte.

I. 1.2. Eclatement de la table en la table complètement définie correspondante.

Dans le cas de l'exemple précédent, cette opération donne la table T.8

Règles	111111111	222	333333333	444	5	6	7	888	9999
C1	123123123	123	123123123	123	1	1	1	111	1111
C2	111111111	111	222222222	222	3	3	3	555	1235
C3	111222333	444	111222333	444	2	3	4	123	4444
D1	111111111	222	222222222	333	2	3	3	222	3333
D2	111111111	111	111111111	222	2	2	1	222	2222
D3	111111111	222	333333333	333	3	3	3	222	3333

T.8

On remarque ainsi que la règle 1 est composée de 9 cas, la règle 2 de 3 cas, etc.....

I. 1.3. Analyse des redondances sur la table complètement définie.

On appelle "cas redondants" les cas pour lesquels les combinaisons des conditions et des décisions correspondantes sont identiques.
On dit que 2 règles sont redondantes lorsque leur intersection est non vide et que leurs décisions sont identiques.
En fait, l'existence d'une telle situation dans une table n'est pas une faute logique, mais la construction du programme correspondant à la table exige que les redondances soient supprimées (voir algorithme de génération dans le chapitre II).

Dans l'exemple de la table T.8, on remarque que la règle 4 et la règle 9 sont redondantes puisqu'elles ont en commun la combinaison des conditions (1, 2, 4) et que celle-ci correspond à la combinaison des décisions (3, 2, 3) identique pour les 2 règles.

I. 1.4. Analyse des incompatibilités sur la table complètement définie.

Rappelons que l'on appelle "cas incompatibles" des cas dont les combinaisons des conditions sont identiques, mais dont au moins une décision diffère.

On entend par règles incompatibles, des règles dont l'intersection comprend au moins 1 cas pour lequel les combinaisons des décisions dans chaque règle sont différentes.

Contrairement au cas précédent, une telle situation correspond à une faute grave puisque lors d'un e occurrence d'une combinaison de conditions correspondant à des cas incompatibles, il n'y aura aucune raison de choisir telle combinaison de décision plutôt que telle autre.

Dans l'exemple de la table T.8 on remarque que les règles 9 et 2 sont incompatibles puisqu'à la combinaison de conditions (1, 1, 4) correspond respectivement les combinaisons de décisions (3, 2, 3) et (2, 1, 2).
On constate qu'il en est de même pour les règles 9 et 7.

I. 1.5. Détection de tous les cas oubliés.

Nous avons vu que le nombre total de cas pour la table T.7 était 60. Or la table T.8 comprenant 34 colonnes, envisage 32 cas puisqu'il y avait une redondance et une incompatibilité.

60 - 32 = 28 cas pourraient donc encore être soumis à l'analyse si nécessaire. Il s'avère donc utile de fournir à l'analyste ces cas en lui demandant s'il s'agit d'un oubli volontaire ou involontaire.

I. 2. Génération du programme à partir de la table.

La génération du programme ne peut se faire qu'à partir d'une table ou il n'y a ni redondances, ni incompatiblités.
Supposons qu'après l'analyse de la table T.7, nous l'ayons corrigée pour obtenir la table T.9.
Celle-ci diffère de la précédente par la règle 9 ou l'on a remplacé $C_2 = -4$ par $C_2 = 4$.
Cette modification permet de supprimer la redondance des règles 9 et 4, et les incompatibilités des règles 9 et 2 et des règles 9 et 7.

Règles →	1	2	3	4	5	6	7	8	9
C_1	0	0	0	0	1	1	1	1	1
C_2	1	1	2	2	3	3	3	5	-4
C_3	-4	4	-4	4	2	3	4	-4	4
D_1	1	2	2	3	4	3	3	2	3
D_2	1	1	1	2	2	2	1	2	2
D_3	1	2	3	3	3	3	3	2	3

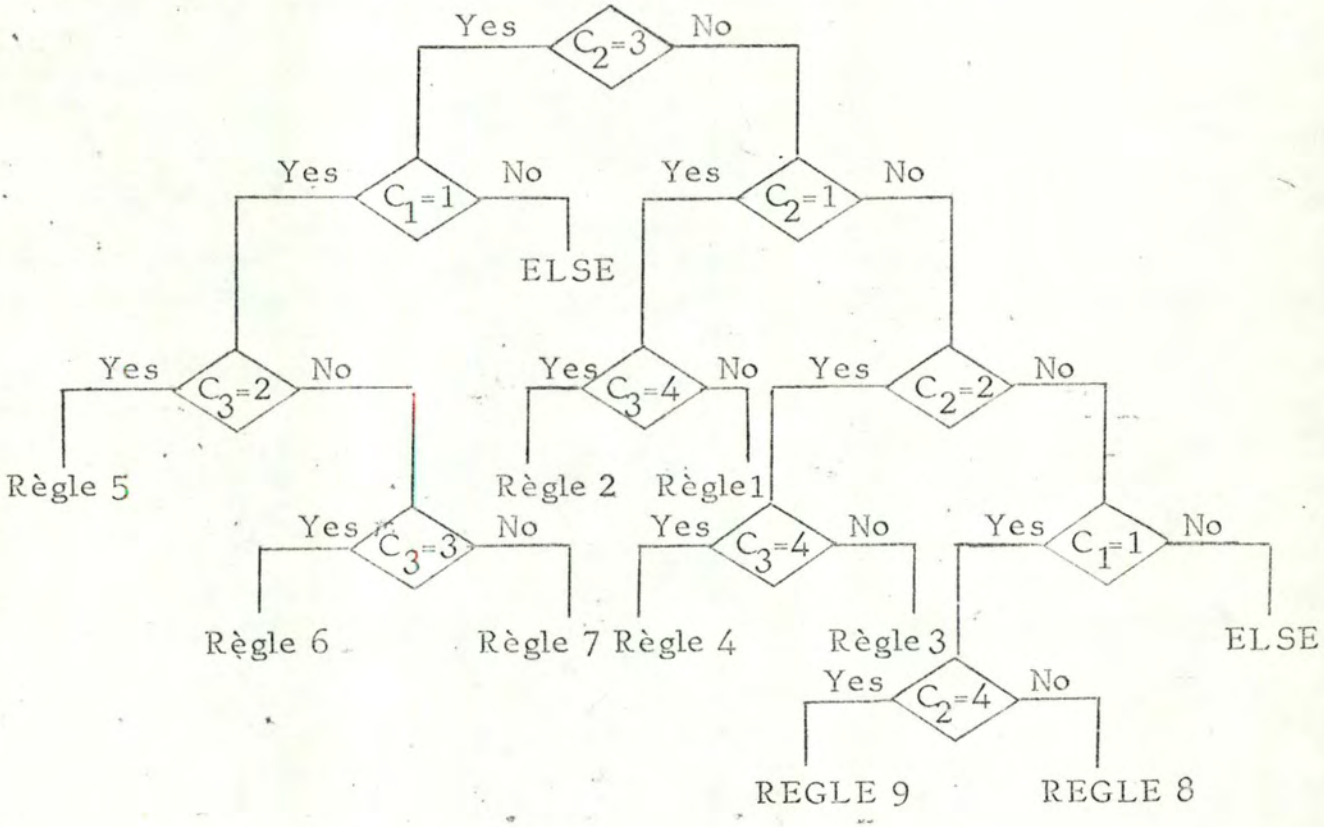
T.9

Le programme généré à partir de cette table est le suivant :

```
            CLI     C2, X'F3'
            BE      PARA 1
            CLI     C2, X'F1'
            BE      PARA 2
            CLI     C2, X'F2'
            BE      PARA3
            CLI     C1, X'F1'
            BE      PARA 4
            BC      15, ELSE

PARA1       EQU     *
            CLI     C1, X'F1'
            BE      PARA5
            BC      15,ELSE

PARA2       EQU     *
            CLI     C3,X'F4'
            BE      REGLE2
            BC      15,REGLE1

PARA3       EQU     *
            CLI     C3, X'F4'
            BE      REGLE4
            BC      15,REGLE3

PARA4       EQU     *
            CLI     C2,X'F4'
            BE      REGLE9
            BC      15,REGLE8

PARA5       EQU     *
            CLI     C3,X'F2'
            BE      REGLE5
            CLI     C3,X'F3'
            BE      REGLE6
            BC      15,REGLE7
```

```
REGLE1      EQU     *
            BAL     1,D11
            BAL     1,D21
            BAL     1,D31
            BC      15,FIN

REGLE2      EQU     *
            BAL     1,D12
            BAL     1,D21
            BAL     1,D32
            BC      15,FIN

REGLE3      EQU     *
            BAL     1,D12
            BAL     1,D21
            BAL     1,D33
            BC      15,FIN

REGLE4
              .
              .
              .

FIN         EOJ
```

L'organigramme correspondant au programme précédent.



I. 2.1. Nous voyons que le programme généré ne tient absolument pas compte de ce que représente les conditions et les décisions.

C'est au programmeur d'ajouter une série de routines "Dij" exécutant les décisions et de positionner avant, les conditions aux bonnes valeurs.
Ce qui est important, c'est de constater que le programmeur n'a plus à se soucier de la logique du problème.

I. 2.2. Cette programmation est basée sur l'organigramme (voir théorie dans le chapitre II). Elle consiste à générer un jeu d'instructions de façon itérative jusqu'au moment ou tous les cas ont été examiné De plus; il existe deux algorithmes d'optimisation de cet organigramme, le premier qui optimise le nombre de branchements, le second ou l'on choisit les aiguillages de telle sorte que pour une sélection donnée, les deux branches de l'alternative soient de dimensions semblables. Nous avons choisi le second, c'est-à-dire celui qui réduit le nombre moyen de tests, tout d'abord parce que le traitement par éliminations successives des décision constitue un cas d'espèce, ensuite, parce que la diminution du nombre d'opérations de test (d'ailleurs plus lente que celle de branchement)est plus intéressante que celle du nombre d'opérations de branchements.

I. 3. Analyse des équations logiques.

Nous avons déjà vu dans l'introduction comment un problème transpo sous forme d'équations logiques peut être résolu.

Voyons maintenant en détail, à l'aide d'un autre exemple, toutes les possibilités fournies par le programme d'analyse des équations logiques.

Il s'agit du problème classique de réservation d'une place d'avion dont l'énoncé est le suivant :

"Un voyageur se présentant au guichet peut formuler une demande de place de première classe. Si au moins une place de première est disponible pour le vol qu'il désire, un billet de première classe lui sera délivré. Si toutes les places de première sont réservées, il sera invité à accepter éventuellement une place en classe touriste si au moins une de ces places est libre.
Dans ce cas, on lui délivre un billet touriste. S'il n'accepte pas de changer de classe, ou si toutes les places de classe touriste sont réservées, sa demande sera enregistrée dans une liste d'atten de première classe.
S'il demande une place touriste, et si une telle place est disponible on lui délivre un billet touriste, sinon on lui demande s'il accepterait une place en première classe.

Si oui, et si une telle place est disponible, on lui délivre un billet de première. Dans le cas contraire, sa demande prendra place en file d'attente touriste".

Pour ce problème, le client doit donc fournir 2 renseignements :

1. le type de place (D_p = première; D_t = touriste) qu'il désire.
2. s'il est d'accord d'accepter une place dans l'autre catégorie. (A)

Il y a en plus 2 conditions internes aux problèmes qui sont :

1. d_p : disponibilité de la classe première
2. d_t : disponibilité de la classe touriste.

Quant aux conditions, elles sont aux nombres de quatre :

1. P_p : délivrer un billet première,
2. P_t : délivrer un billet touriste,
3. L_p : placer sur la liste d'attente "première",
4. L_t : placer sur la liste d'attente touriste.

I. 3.1. Equations de décision.

L'analyste doit maintenant transposer ce problème sous forme d'équations logiques. Nous obtenons :

$$P_p = D_p \cdot d_p A + D_t \bar{d}_t d_p A + D_p \bar{d}_p \bar{A}$$

$$P_t = D_t d_t A + D_p \bar{d}_p d_t A + D_t d_t \bar{A}$$

$$L_p = D_p \bar{d}_p \bar{A} + D_p \bar{d}_p \bar{d}_t A + D_p \bar{d}_p \bar{d}_t \bar{A}$$

$$L_t = D_t \bar{d}_t \bar{A} + D_t \bar{d}_p \bar{d}_t A + D_t \bar{d}_p \bar{d}_t \bar{A}$$

I. 3.2. Equations d'incompatibilité.

Ces équations étant posées, il s'agit maintenant de déceler dans le problème quelles seraient les combinaisons de conditions qui ne pourraient pas se produire ?

Dans cet exemple, les équations exprimant des incompatibilités entre variables sont :

$$E1 = \bar{D}_p \bar{D}_t + D_p D_t$$

(Il faut au moins que le client demande une place dans une catégorie et il ne peut en demander une dans les deux à la fois).

I. 3.3. Table des incompatibilités.

L'analyste doit pour terminer fournir au programme , les éventuelles incompatibilités entre certaines décisions.

On entend par là des décisions (équations de décisions) ne pouvant prendre la valeur "1" pour une même combinaison des conditions. Dans l'exemple de réservation des places d'avion, il est évident que P_p c'est-à-dire " délivrer un billet de première" ne peut se produire (prendre la valeur 1) pour une même occurrence des valeurs des conditions que P_t c'est-à-dire "délivrer un billet touriste" sans quoi dans certaines situations une personne se verrait délivrer les 2 billets en même temps.

Dans cette première version, ces incompatibilités sont mises sous forme de tables dans lesquelles une ligne correspond à une équation logique de décision.

Pour l'exemple précédent : P_p = 0 1 1 1 est une ligne de la table signifiant que si par suite d'une certaine occurrence des valeurs des variables "conditions" (D_p, D_t, d_p, d_t, A), la décision P_p valait "1" alors P_t, L_p et L_t doivent être à zéro (col. 2, 3 et 4 à 1).

Dans cet exemple, il est évident que toutes les décisions à prendre sont exclusives d'où la table complète d'incompatibilités suivante :

P_p : 0 1 1 1

P_t : 1 0 1 1

L_p : 1 1 0 1

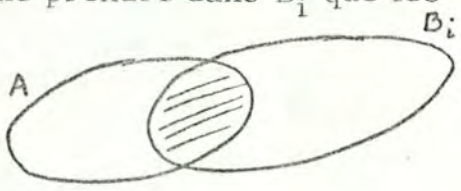
L_t : 1 1 1 0

Il peut se faire que dans un problème l'occurrence de la 2e variable de décision, par exemple, n'exclue aucune autre occurrence d'autres variables "décision". Dans ce cas, la deuxième ligne de la table peut être omise (elle contiendrait en fait toutes des valeurs nulles).

I. 3.4. Procédé d'analyse automatique.

Les étapes successives de l'analyse des équations de décision sont les suivantes :

1. simplification de chacune des équations logiques de décision (voir algorithme dans le chapitre III).
2. simplification et complémentation des équations logiques d'incompatibilités.
3. multiplication des résultats obtenus à l'étape 2 pour toutes les équations d'incompatibilités.
   On obtient alors le produit de l'inverse des incompatibilités.
4. multiplication de toutes les équations de décision par le résultat de l'étape 3. et simplification .
   On obtient alors les équations de décision ou l'on a supprimé tous les cas incompatibles.
   En effet : le résultat de l'étape 3 est un ensemble A représentant tous les cas compatibles.
   Considérons B_i = (solutions de la i^e équation de décision (résultat de l'étape 1).) L'étape 4 consiste à faire l'intersection entre A et B_i pour $\forall_i$ c'est-à-dire à ne prendre dans B_i que les



   cas qui sont compatibles.
   Il est évident que si nous obtenons une équation de décision pour laquelle $A \cap B_i = \emptyset$, cette équation est incompatible n'ayant aucune solution acceptable.
   Elle sera donc signalée à l'analyste.
   Il faut signaler aussi que le fait de simplifier les équations de décision supprime toutes les redondances.
5. Multiplication de toutes les équations de décision de l'étape 1 , complémentation du résultat et simplification.
   Cela nous donne sous la forme d'équation logique la plus condensée possible tous les cas non prévus.

6. Si aucune incompatibilité n'a été détectée, on construit la table de décision à partir des résultats obtenus à l'étape 4 et en tenant compte de la table des incompatibilités décrite en 3.3.
Nous obtenons une table de décision dans laquelle ont été supprimés tous les cas redondants et tous les cas incompatibles.
Ceci a l'avantage de ne pas devoir exprimer la table complètement définie qui risque d'être grande lorsque le nombre de variables est important.

Pour l'exemple précédent nous obtenons :

1. simplification des équations logiques de décision.

$$P_p = d_p \cdot A \cdot D_t \cdot d'_t + D_p \cdot d_p$$

$$P_t = D_p\, d'_p\, A\, d_t + D_t \cdot d_t$$

$$L_p = D_p \cdot d'_p \cdot A' + D_p \cdot d'_p \cdot d'_t$$

$$L_t = A'\, D_t\, d'_t + d'_p\, D_t\, d'_t$$

2. Simplification et complémentation des équations logiques d'incompatibilités.

$$E_1 = D_p \cdot D'_t + D'_p\, D_t$$

3. Produit de l'inverse des incompatibilités.

$$I = D_p \cdot D'_t + D'_p \cdot D_t$$

4. Multiplication des équations de décision par l'inverse des incompatibilités et simplification.

1. $P_p \cdot I = D_p\, d_p\, D'_t + D'_p\, d_p\, A\, D_t\, d'_t$

2. $P_t \cdot I = D_p \cdot d'_p\, A\, D'_t\, d_t + D'_p\, D_t\, d_t$

3. $L_p \cdot I = D_p\, d'p\, A'\, D'_t + D'_p \cdot d'_p\, D_t\, d'_t$

4. $L_t \cdot I = D'_p \cdot d'_p \cdot D_t\, d'_t + D'_p\, A'\, D_t\, d'_t$

5. Equation des cas oubliés.

Cas oubliés = $D'_p\, D'_t$

N.B. : Dans ce qui précède, une variable suivie du symbol représente la variable complémentée.

CHAPITRE II

ETUDE DES TABLES DE DECISION.

II. 1. Construction d'une table complètement définie.

Nous avons vu dans les exemples précédents qu'en général la table de décision relative à un problème déterminé n'était pas une table complètement définie. Il s'agissait donc de trouver une procédure permettant d'une part, de permettre de calculer tous les cas présents dans une règle particulière et d'autre part, de pouvoir connaître les cas non présents dans la table.

Supposons que nous ayons une table ayant comme entrée les conditions C_i / $i \in [1, MC]$ avec MC = nombre total des conditions.

Définissons en plus les valences V_i associées à chaque condition.
V_i est la valeur maximum que peut prendre la condition C_i.
Le domaine de valeurs de C_i est donc $\{ -V_i, -V_i+1, \ldots\ldots 1, 0, 1, \ldots\ldots V_i-1, V_i \}$

1e propriété :

Le nombre de cas total (NT) relatif à un ensemble de conditions est égal au produit des valeurs de ces conditions soit :

$$NT = \prod_{i=1}^{MC} V_i$$

Supposons une règle $R = (i_1, i_2, \ldots\ldots i_{MC}) \Longrightarrow$ occurence des valeurs d'une combinaison de conditions. On peut énoncer la

2e propriété :

le nombre de cas (NC) représenté par la règle R est égal au produit des valences des conditions dont la valeur dans R est nulle et des valences - 1 des conditions dont la valeur dans R est négative.

Autrement dit :

$$NC = \left(\prod_{i \in I_1} V_i \right) \cdot \left(\prod_{j \in I_2} V_j - 1 \right)$$

avec :

$I_1 = \{$ ens des indices $k \in [1, MC]$ / que $R(k) = i_k$ soit nulle $\}$

$I_2 = \{$ ens des indices $k \in [1, MC]$ / que $R(k) = i_k$ soit négati

Voyons maintenant quelles sont les formules générales permettant de générer les cas relatifs à une règle R.
Supposons que nous ayons déjà calculé le nombre NC de cas contenu dans cette règle R (Propriété 2)
soit encore :

V_i = valence de la iè ligne

$$W_i = \left(\prod_{j \in L_i} V_j \right) \left(\prod_{j \in L'_i} V_j - 1 \right)$$

avec $L_i = \{$ indices $1 \leq j \leq i$ / $R(j) = 0 \}$

$L'_i = \{$ indices $1 \leq j \leq i$ / $R(j) < 0 \}$

et $W_0 = 1$ par convention.

Considérons aussi la matrice A de dimension (MC, NC) représentant l'ensemble des cas de la règle R.

Par exemple : A (i,j) pour i = 1,MC représente le j^{em} cas de la règle R.

N.B. : D'après la définition des W_i, on a toujours que $W_{MC} = NC$.

3e propriété :

les éléments de la matrice A sont calculés de la manière suivante :

1. si $R(i) > 0$ alors $A(i,j) = R(i)$ pour $j = 1, \ldots, NC$
2. si $R(i) = 0$ alors $A(i,j) = L$

pour $j = 1 + \frac{(L-1) * NC}{W_{i-1} * V_i} + k * \frac{NC}{W_{i-1}}$, ...

$$\ldots\ldots\ldots, L * \frac{NC}{W_{i-1} * V_i} + k * \frac{NC}{W_{i-1}}$$

pour $L = 1, \ldots\ldots, V_i$

pour $k = 0, \ldots\ldots, W_{i-1} - 1$

3. si $R(i) < 0$ alors $A(i,j) = L_1$

$$\text{pour } j = 1 + (L_2 - 1) * \left[\frac{NC}{W_{i-1} * V_i} + 1\right] + k * \frac{NC}{W_{i-1}}$$

$$\ldots\ldots, L_2 * \left[\frac{NC}{W_{i-1} * V_i} + 1\right] + k * \frac{NC}{W_{i-1}}$$

$$\text{pour } L_2 = 1, \ldots\ldots, V_i - 1$$

$$\text{pour } k = 0, \ldots\ldots, (W_{i-1} - 1)$$

ou $L_1 = L_2$ pour $L_2 < |R(i)|$

$= L_2 + 1$ pour $L_2 \geqslant |R(i)|$

et ou $[\ \ ]$ signifie "la partie entière".

Exemple :

Considérons , $R = (0, 2, -4, 0)$ une règle pour laquelle on a les valences

$$V_1 = 3,\ V_2 = 2,\ V_3 = 4,\ V_4 = 2$$

on a que

$MC = 4$ (nombre total de conditions)

$NC = (V_1 * V_4) * (V_3 - 1)$

$= (3 * 2) * (3)$

$= 18$

Cela signifie que la règle R représente en réalité 18 cas. Essayons maintenant de construire la matrice A représentant tous ces cas.

A est de dimension (4, 18)

Le calcul des W_i donne :

$$W_0 = 1,\ W_1 = 3,\ W_2 = 3,\ W_3 = 9,\ W_4 = 18$$

$R(1) = 0 \Longrightarrow A(1,j) = L$

pour $j = 1 + (L-1) * 6 + k.18, \ldots, L.6 + k . 18$

pour $L = 1, 2, 3$

pour $k = 0$

$\Longrightarrow$ A (1,j) = 1 pour j = 1, ..., 6
= 2 pour j = 7, ..., 12
= 3 pour j = 13, ..., 18

R (2) = 2 $\Longrightarrow$ A (2,j) = 2 pour j = 1, ..., 18

R (3) = -4 $<$ 0 $\Longrightarrow$ A (3,j) = L1
pour j = 1 + (L_2-1) $\times$ 2 + k $\times$ 6, ..., $L_2 \times 2 + k \times$
pour L_2 = 1, 2, 3
pour k = 0, 1, 2

ici $L_2 = L_1$ pour L_2 = 1, 2, 3 car $L_2 < |-4|$
$\Longrightarrow$ A (3,j) = 1 pour j = 1, 2, 7, 8, 13, 14
$\Longrightarrow$ A (3,j) = 2 pour j = 3, 4, 9, 10, 15, 16
$\Longrightarrow$ A (3, j)= 3 pour j = 5, 6, 11, 12, 17, 18

R (4) = 0 $\Longrightarrow$ A (4,j) = L
pour j = 1 + (L-1)$\times$ 1 + k $\times$ 2, ..., L $\times$ 1 + k $\times$ 2
pour L = 1, 2
pour k = 0, 1, 2, 3, 4, 5, 6, 7, 8
$\Longrightarrow$ A (4,j)= 1 pour j = 1, 3, 5, 7, 9, 11, 13,15
A (4,j)= 2 pour j = 2, 4, 6, 8, 10, 12, 14, 16, 18

On a donc pour la règle R = (0, 2, -4, 0) l'ensemble des cas représenté par A, c'est-à-dire :

	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
	1	1	2	2	3	3	1	1	2	2	3	3	1	1	2	2	3	3
	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
Valeurs de F: →	4	28	10	34	16	40	5	29	11	35	17	41	6	30	12	36	18	42

II. 2. Méthode utilisée pour détecter les cas redondants, incompatibles et oubliés.

La méthode employée consiste à établir une fonction bijective entre les entiers compris dans l'intervalle y = [1, NT] et l'ensemble X des cas possibles correspondant à la table analysée.

On entend par cas possibles des cas dont la valeur pour chaque condition, prise en valeur absolue est comprise entre 1 et la valence de cette condition.

On obtient la fonction :

$$F : \mathbb{Z}^{MC}_{|X} \longrightarrow \mathbb{N}_{|Y}$$

$$(i_1, i_2 \ldots i_{MC}) \longrightarrow F(i_1, i_2, \ldots i_{MC}) =$$

$$1 + \sum_{j=1}^{MC} \left(\prod_{k=0}^{j-1} V_k \right) (i_j - 1)$$

$$\text{avec } V_0 = 1$$

Considérons les 18 cas de la règle R = (0, 2, -4, 0) que l'on a généré précédemment dans la matrice A.

Pour le 1er cas on a :

$$F(1, 2, 1, 1) = 1 + 0 + 1 \cdot (3) + 0 \cdot (3.2) + 0 (3.2.4) = 4$$

On vérifiera sans peine les valeurs de F correspondant aux différents cas de R et qui sont représentées en dessous de la matrice A (T.9).

On remarquera encore que la valeur de la fonction est maximum pour le cas ou toutes les valeurs des conditions sont égales à leur valence. Dans l'exemple précédent il s'agissait de F (3, 2, 4, 2) = 48 = NT.

II. 3. Programmation des tables de décision.

Dans ce qui suit, nous allons discuter les deux principales techniques de programmation des tables de décision à savoir la programmation basée sur l'organigramme et celle utilisant la technique du "rule mask". Nous examinerons ensuite des procédés d'optimisation de la technique de l'organigramme.

II. 3.1. Technique de l'organigramme.

Cette technique développée par Pollack dans le cas de table bivalente, transforme la table de décision en une arborescence

dont chaque sommet représente un test. A chaque feuille de cette arborescence est associée soit une des règles définies dans la table, soit la règle ELSE.

La règle ELSE est une règle qui correspond à une combinaison de condition ne se trouvant pas dans la table. Comme l'analyse de celle-ci est supposée avoir été faite précédemment, on suppose que cette règle ne peut correspondre qu'à une erreur. La décision correspondant à cette règle renverra donc à une routine d'erreur.

L'arborescence est un organigramme ayant la même logique que la table de décision et est construite en appliquant les étapes suivantes :

1. choisir la condition à tester dans la table initiale, soit C_i,

2. choisir la valeur k à tester pour cette condition C_i. Cette valeur sera choisie parmi l'ensemble $[\ 1, V_i\ ]$ ou V_i est la valence de C_i, autrement dit on teste sur l'ensemble des valeurs simples de la condition C_i.

3. Placer le test $C_i = k$ à la racine de l'arborescence.

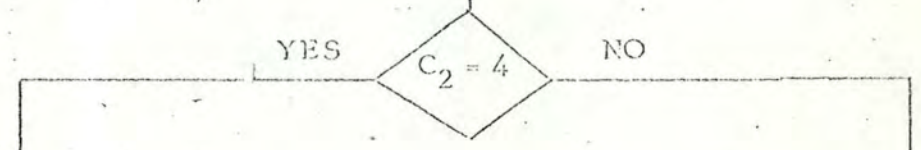
4. Connecter à la branche de l'arborescence correspondant à la sortie "YES" du test, la sous-table contenant dans sa partie "condition" toutes les conditions de la table initiale exceptée la condition testée c'est-à-dire C_i; Pour celles-ci, cette sous-table contiendra toutes les règles pour lesquelles la condition $C_i = k$ est vérifiée.

   On aura donc dans cette table toutes les règles pour lesquelles la ième condition vaut soit k, soit 0, soit -j (avec $j \neq k$)

   Connecter à la branche de l'arborescence correspondant à la sortie "NO" du test, la sous-table contenant dans sa partie "condition" toutes les conditions de la table initiale. Pour celles-ci, cette sous-table contiendra toutes les règles pour lesquelles C_i n'est pas égale à k, c'est-à-dire pour lesquelles la ième condition vaut soit 0, soit i (avec $i \neq k$) soit - k.

   La formation des deux sous-tables est illustrée par l'exemple suivant :

Règles →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	V_i
C_1	0	0	0	0	1	1	1	1	1	-1	2	2	3	3	3
C_2	1	1	2	2	3	3	3	5	5	5	4	4	4	4	5
C_3	-4	4	-4	4	2	3	4	-4	4	1	-4	4	-4	4	4



Règles →	11	12	13	14
C_1	2	2	3	3
C_3	-4	4	-4	4

Règles →	1	2	3	4	5	6	7	8	9	10
C_1	0	0	0	0	1	1	1	1	1	-1
C_2	1	1	2	2	3	3	3	5	5	5
C_3	-4	4	-4	4	2	3	4	-4	4	1

5. On applique ensuite la procédure décrite précédemment en prenant successivement chacune des deux sous-tables comme table initiale. On continue le processus pour toutes les sous-tables dérivées de ces dernières aux étages suivants jusqu'au moment où l'on tombe sur un des cas ci-dessous :

   a. une des sous-tables est vide c'est-à-dire qu'il n'y a plus de règles satisfaisant à une des alternatives d'un des tests. Dans ce cas on termine cette branche de l'arborescence par la règle ELSE.

   b. une des sous-tables ne contient plus qu'une colonne (règle). On termine alors cette branche de l'arborescence par la règle correspondant à cette colonne.

REMARQUE.

1. Rappelons que cet algorithme ne s'applique qu'à des tables ou les redondances ont été supprimées sans quoi il se pourrait que l'on n'arrive jamais à l'un des cas a et b.

Exemple général.

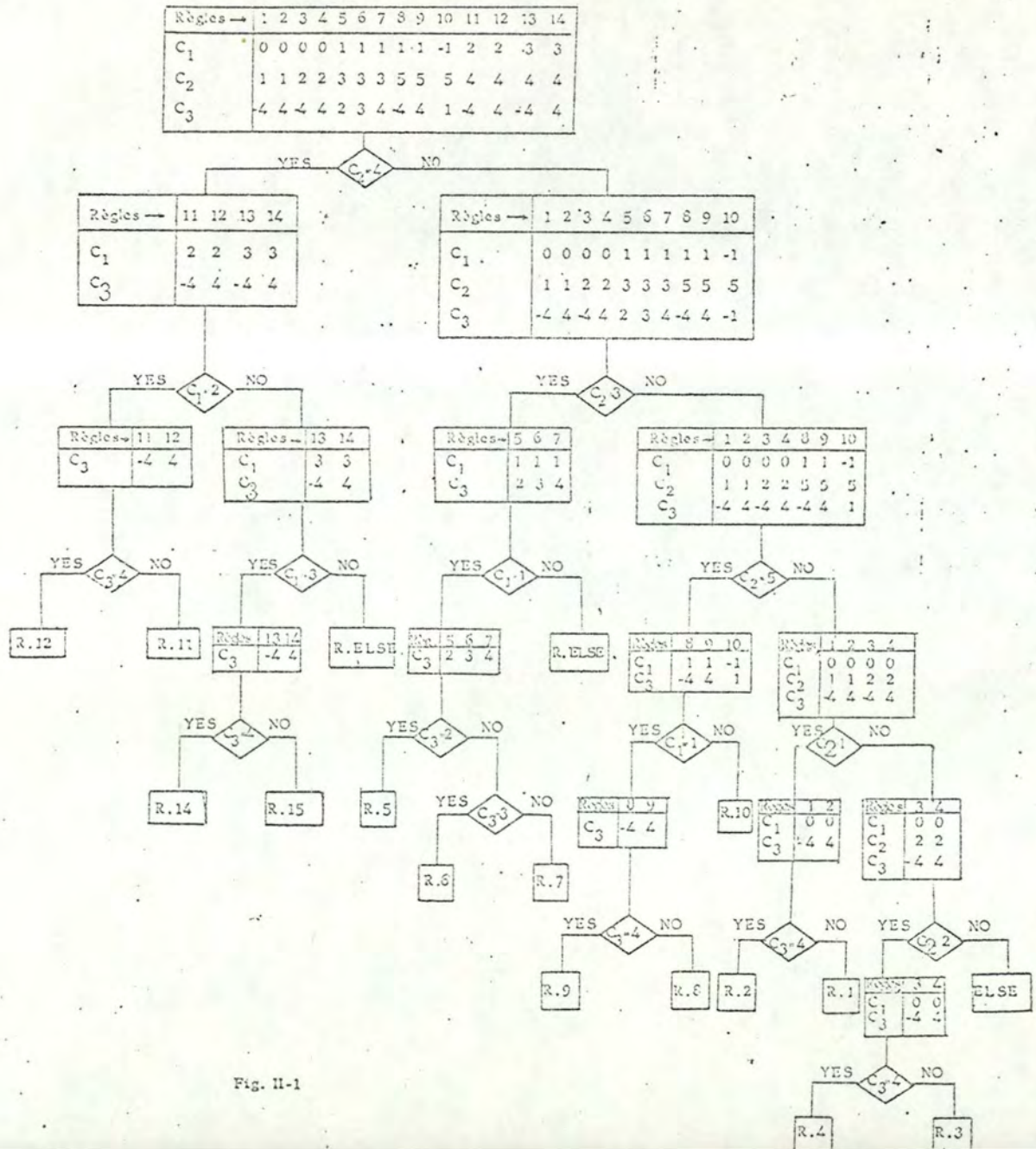


Fig. II-1

II. 3.2. Technique du "rule mask".

Signalons tout d'abord que cette technique telle qu'elle est présentée ici ne s'applique qu'à des tables où toutes les conditions sont de nature binaire.

Introduisons tout d'abord deux notions nouvelles, celle de matrice de conditions et celle de matrice de masques. Ces deux matrices ont pour origine la table de décision.

Dans la matrice de condition, les 1 remplacent les 2 de la Table de décision, c'est-à-dire les occurrences "oui" des conditions tandis que les 1 y remplacent les 1 et les 0, c'est-à-dire les occurrences "non" et "indifférence" des conditions.

D'autre part, dans la matrice de masque, les 1 remplacent les 2 et les 1 et les 0 de la table de décision sont repris dans cette dernière.

Dans l'exemple ci-dessous, la table T.10 a pour matrice de masque la table T.12 et pour matrice de condition la table T.11.

T.10

	I	II	III	IV	V
C_1	2	2	0	0	1
C_2	2	0	1	2	1
C_3	1	2	1	0	2
C_4	2	2	2	1	0

T.11

COND.

C_1	1	1	0	0	0
C_2	1	0	0	1	0
C_3	0	1	0	0	1
C_4	1	1	1	0	0

T.12

MAS

C_1	1	1	0	0	1
C_2	1	0	1	1	1
C_3	1	1	1	0	1
C_4	1	1	1	1	0

Il est bon de distinguer 2 phases.
La phase de conversion durant laquelle sont construits ces deux matrices et la phase exécution pendant laquelle on se sert de ces deux matrices comme opérateurs.
En plus de ces deux matrices, un vecteur donnée doit être prévu. Un vecteur donnée est une zone de mémoire réservée pour enregistrer durant la phase d'exécution, les réponses aux conditions au moyen des valeurs 0 (si $C_i = 1$) et 1 (si $C_i = 2$).

Durant la phase exécution, le programme procéde comme suit :

1. remplissage du vecteur donnée par le résultat des conditions.

   Exemple : supposons que le cas suivant se produise :

   - la réponse à C_1 est NON $\Longrightarrow C_1 = 1$
   - la réponse à C_2 est NON $\Longrightarrow C_2 = 1$
   - la réponse à C_3 est NON $\Longrightarrow C_3 = 1$
   - la réponse à C_4 est OUI $\Longrightarrow C_4 = 2$

   alors le vecteur donnée correspondant est

   (0 0 0 1)

2. Chaque élément du vecteur donnée est multiplié par l'élément correspondant de la 1ère colonne de la matrice masque, et le résultat est ensuite comparé à la 1ère colonne de la matrice condition. Si les deux vecteurs correspondent alors on applique la règle 1. Dans le cas contraire on multiplie le vecteur donnée à la seconde colonne de la matrice masque et on compare le résultat à la seconde colonne de la matrice condition; si les deux vecteurs correspondent alors on prend en considération la règle 2. On continue de cette manière jusqu'au moment ou :

   - soit l'on trouve la règle à appliquer,
   - soit l'on arrive au bout de la table. Dans ce dernier cas on applique la règle ELSE.

   Prenons comme exemple le vecteur donnée précédent à savoir

   (0 0 0 1)

   1ère étape :

$$\underset{D}{\begin{bmatrix}0\\0\\0\\1\end{bmatrix}} \times \underset{M_1}{\begin{bmatrix}1\\1\\1\\1\end{bmatrix}} = \underset{R}{\begin{bmatrix}0\\0\\0\\1\end{bmatrix}} \neq \underset{COND_1}{\begin{bmatrix}1\\1\\0\\1\end{bmatrix}}$$

   D = Donnée
   M_i = i^e vecteur colonne de la matrice masque
   $COND_i$ = i^e vecteur colonne de la matrice condition.
   R = résultat.

2ème étape :

$$\begin{bmatrix}0\\0\\0\\1\end{bmatrix} \times \begin{bmatrix}1\\0\\1\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix} \neq \begin{bmatrix}1\\0\\1\\1\end{bmatrix}$$

$D \qquad M_2 \qquad R \qquad COND_2$

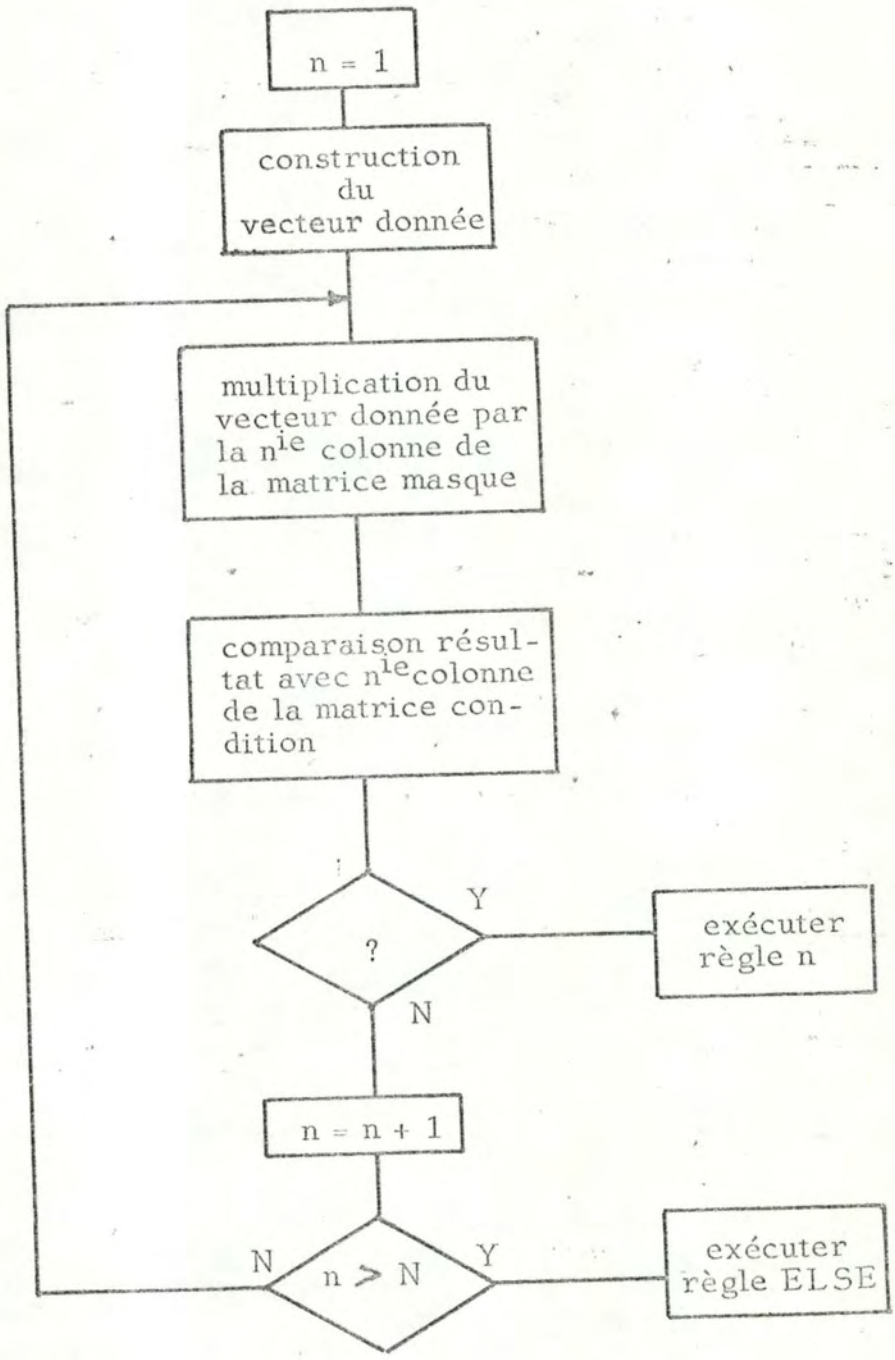
3ème étape :

$$\begin{bmatrix}0\\0\\0\\1\end{bmatrix} \times \begin{bmatrix}0\\1\\1\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

$D \qquad M_3 \qquad R \qquad COND_3$

On applique donc la règle 3.

On peut résumer la technique du "rule mask" par l'organigramme suivant :



II. 3.3. Nouvelle technique du "rule mask" pour les tables polyvalentes.

L'inconvénient de la méthode précédente est comme nous l'avons vu le fait qu'elle ne s'applique qu'à des tables de nature binaire. Il était donc intéressant de rechercher une méthode plus générale permettant de s'appliquer aux tables polyvalentes du type de celles employées pour la méthode de l'organigramme.

Considérons la table T.13.

Règles →	I	II	III	IV	Valence ↓
C_1	2	3	0	1	3
C_2	-2	0	1	2	2
C_3	-1	3	1	0	3

T.13

Cette table possède 3 conditions dont les valences, c'est-à-dire les valeurs maximales des conditions sont respectivement 3, 2, 3.

On associe alors a chaque ligne de condition C_i une matrice de masque M_i définie comme suit :

1. la matrice M_i est de dimension (V_i , N) ou V_i est la valence de la $i^{iè}$ condition et N le nombre de règles de la table des conditions R.
2. $M_i (1,k) = 0$ si $R (i, k) = x$ avec $x \neq 1$
   $= -1$

   $M_i (1,k) = 1$ si $R (i, k) = 1$
   $= 0$
   $= -x$ avec $x \neq 1$

Pour la table T.13, nous obtenons les 3 matrices suivantes relatives aux 3 lignes des conditions

$$M_1 \equiv \begin{pmatrix} 0\,0\,1\,1 \\ 1\,0\,1\,0 \\ 0\,1\,1\,0 \end{pmatrix} \quad M_2 \equiv \begin{pmatrix} 1\,1\,1\,0 \\ 0\,1\,0\,1 \end{pmatrix} \quad M_3 \equiv \begin{pmatrix} 0\,0\,1\,1 \\ 1\,0\,0\,1 \\ 1\,1\,0\,1 \end{pmatrix}$$

On peut encore définir les matrices M_i de la manière suivante :

- la $j^{è}$ colonne de la matrcie M_i représente l'ensemble des valeurs acceptables pour la i^{e} condition de la j^{e} règle.

Ainsi, par exemple, le 1^{er} vecteur colonne de M_3 c'est-à-dire (0 1 1) exprime que pour la règle 1, la condition C_3 peut prendre la valeur 2 ou 3.

En plus de ces matrices de masque nous aurons comme pour la technique précédente un vecteur donnée dont les éléments seront cette fois les valeurs des conditions elles-mêmes.

Durant la phase d'exécution, le programme procède comme suit :

1. remplissage du vecteur donnée par le résultat des conditions.
   Par exemple : D≡ (3 2 3)

2. D'après ce vecteur donnée on sélectionne une ligne de chaque matrice de masque et l'on multiplie toutes ces lignes entre elles.
   La sélection se fait de la manière suivante :
   - si D (i) = k, on sélectionne la $k^{ième}$ ligne de la $i^{è}$ matrice.

   Le résultat R de cette étape est donc défini par :

   $$R(i) = \overset{MC}{\underset{l=1}{**}} M_l\ (D(l), i) \text{ pour } i = 1, \ldots, N$$

   ou MC = nombre de conditions.

   Pour l'exemple précédent nous avons :

   $$R(1) = M_1\ (D(1), 1) * M_2\ (D(2), 1) * M_3\ (D(3), 1)$$
   $$= M_1\ (3,1) * M_2\ (2, 1) * M_3\ (3,1)$$
   $$= 0 * 0 * 1 = 0$$

   $$R(2) = M_1\ (D(1), 2) * M_2\ (D(2), 2) * M_3\ (D(3), 2)$$
   $$= M_1\ (3, 2) * M_2\ (2, 2) * M_3\ (3, 2)$$
   $$= 1 * 1 * 1 = 1$$

   $$R(3) = 1 * 0 * 0 = 0$$

   $$R(4) = 0 * 1 * 1 = 0$$

   Nous avons donc le vecteur résultat $R \equiv \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$

3. On applique la règle correspondant à la valeur R(i) = 1 du vecteur R.

   En effet, s'il n'y a pas de redondances et d'incompatibilités, le vecteur R contiendra au plus un seul "1".

   Ainsi, pour l'exemple de vecteur D = (3 2 3), il faut exécuter la règle 2.

   Si par contre le vecteur R est identiquement nul, on exécutera la règle ELSE.

II. 3.4. Optimisation du traitement.

La programmation en langage machine se base, comme nous l'avons vu précédemment, uniquement sur des opérations de tests (CLI) et de branchements(BE et BC) en cascade destinées à aboutir à la décision qui correspond aux conditions introduites. L'optimisation du traitement peut consister soit à minimiser l'espace mémoire soit à diminuer le temps d'exécution. Cependant, si les conditions spécifiées dans une table de décision sont telles que seulement une ou deux instructions en langage machine sont nécessaires pour tester chacune d'elles, l'espace mémoire ne sera pas influencée significativement par une modification de la structure de l'arbre. Si par contre, chaque condition requiert pour être testée une sous-routine, le programme peut être structuré de telle sorte que chaque sous-routine n'apparaisse qu'une seule fois.
Tester la condition à un point particulier de l'organigramme signifie alors se brancher à la sous-routine et revenir ensuite au point du test. On voit donc que même dans le cas ou le test prend beaucoup de place mémoire, l'on peut ramener celui-ci à un couple d'instructions en langage machine. Ce qui rend également le problème de minimisation de l'espace mémoire moins important.
La seule optimisation valable ici est donc la minimisation du temps d'exécution. Celle-ci est influencée par un certain nombre de facteurs qui sont :

1. le nombre moyen de branchements,
2. le nombre moyen de tests,
3. le temps nécessaire pour tester chaque condition,
4. la fréquence relative de chaque règle.

Il faut noter que la diminution du nombre moyen de tests provoque une augmentation du nombre de branchements.
Un choix devait donc être fait à ce niveau. Or, nous savons que le temps d'exécution d'un test est plus long que celui d'un branchement et que le test peut être précédé, nous venons de le voir, par une routine allant placer une valeur dans la condition à tester.
Si donc nous considérons le temps d'exécution d'un test comme

étant celui de la routine (éventuelle) précédent le test, plus celui du test proprement dit, ce temps peut être fort variable et relativement long. Nous avons donc choisi de limiter le nombre de tests à exécuter.

1. Limitation du nombre de tests:

Cet algorithme équilibre à chaque test les deux branches de l'arbre de façon à diminuer la longueur moyenne de celles-ci et donc conséquemment des tests à effectuer.
Cela peut se faire en choisissant de façon adéquate, et les conditions et les valeurs à tester dans ces dernières.

Soit R la matrice des conditions associée à une branche de l'arbre et supposons que la dimension de celle-ci soit (D_1, D_2) avec $D_2 > 1$ (sinon R est une feuille de l'arbre)
Le problème se résoudra en procédent dans l'ordre chronologique suivant :

a. déterminer pour chaque règle i ($1 \leq i \leq D_2$) les poids

$$q_i = \prod_{k \in I_1} V_k \quad \text{avec } I_1 = \{ k \ / \ R(k,i) = 0 \}$$

et V_k = valence de la $k^{ième}$ condition.

b. pour chaque ligne relative à chacune des conditions, effectuer les sommes S_k des poids correspondants aux 0, c'est-à-dire aux valeurs optionnelles.

$$S_k = \sum_{i \in I_2} q_i \quad \text{avec } I_2 = \{ i \ / \ R(k, i) = 0 \}$$

c. on calcule alors

$$S_r = \min_{1 \leq k \leq D_1} S_k$$

C_r sera la condition sélectionnée.

d. pour cette condition C_r on choisira la valeur k à tester de telle sorte que

$$\left| X_k - \sum_{j \neq k} X_j \right| = \min_{1 \leq i \leq V_r} \left| X_i - \sum_{j \neq i} X_j \right|$$

avec

$$X_i \quad \sum_{j \in L_i} q_j \quad \text{ou} \quad L_i = \{ j \ / \ R(r,j) = 0, \ i \text{ ou } -1 \ (1 \neq i) \}$$

Le choix de la condition C_r permet d'affirmer que le test sera le plus significatif étant donné que pour lui l'intersection des deux sous-tables résultantes contiendra un nombre minimum de règles.

Le choix de la valeur k pour cette condition équilibre les deux branches puisque l'on prend la valeur pour laquelle la différence entre le nombre de règles ayant cette valeur k pour la condition C_r et celles pour lesquelles la valeur est différente de k est minimum.

q	2 I	2 II	4 III	1 IV	1 V	2 VI	8 VII	4 VIII	S
C_1	1	1	1	2	2	2	0	2	8
C_2	1	1	2	2	2	2	3	1	0
C_3	0	0	0	1	1	2	0	0	16
C_4	1	2	0	1	2	0	0	0	14

T.14

Considérons l'exemple de la figure T.14.

On a : $S_2 = \min_{1 \leqslant i \leqslant 4} S_i$ et donc le test portera sur C_2

de plus on a $X_1 = 8 \quad X_2 = 8$ et $X_3 = 8$

On calcule alors $\left| X_k - \sum_{j \neq k} X_j \right| = \min_{1 \leqslant i \leqslant 3} \left| X_i - \sum_{j \neq i} X_j \right|$

$= \min \{ 8, \ 8, \ 8 \}$

Dans ce cas, k peut prendre les valeurs 1, 2 ou 3, le minimum étant atteint p our ces 3 valeurs.

Il est donc indifférent de commencer à sélectionner dichotomiquement (1, 2) de 3 ou (1, 3) de 2 ou (2, 3) de 1 à partir de C_2.

2. Fréquence relative de chaque règle.

Il est certain que l'ordre de sélection peut être modifié si les combinaisons des conditions déterminant les décisions sont examinées avec des fréquences d'importance différente.
Il suffira alors de revenir au cas général en tenant compte dans la détermination de la valeur de S_i, de la probabilité de traitement des combinaisons des conditions à remplir.

Appelons P_i la probabilité de la $i^{è}$ règle.
L'étape b devient :

$$S_k = \sum_{i \in I_2} q_i P_i \quad \text{avec } I_2 = \{ i / R(k, i) = 0 \}$$

tandis que pour l'étape d on remplace X_i par $X'_i = \sum_{j \in L_i} q_j P_j$

Cette modification permet non plus d'équilibrer l'arbre, mais la longueur des branches pondérée par la fréquence de passage dans celles-ci.

$100 * P_i$	10	15	25	20	30	
q_i	1	3	2	4	3	
Règles	1	2	3	4	5	S_i ↓
C_1	2	2	0	0	1	1,3
C_2	2	0	1	2	3	0,45 ←
C_3	1	2	1	0	2	0,8
C_4	2	3	2	1	0	0,9

T.15

Prenons l'exemple de la table T.15 ci-dessus pour laquelle les valences des conditions sont respectivement 2, 3, 2, 3.

On a, par exemple $q_4 = V_1 * V_3 = 2 * 2 = 4$

et $$S_1 = q_3 P_3 + q_4 P_4 = \frac{2 * 25 + 4 * 20}{100} = 1,3$$

le minimum des S_i à lieu pour i = 2, ce qui signifie que le test portera sur C_2.

Reste à choisir la valeur de C_2 sur laquelle portera le test.

Nous avons $X_1 = \sum_{j \in L_1} q_j P_j = \frac{3 * 15 + 2 * 25}{100} = 1,25$

$$X_2 = \frac{1 * 10 + 3 * 15 + 4 * 20}{100} = 1,65$$

$$X_3 = \frac{3 * 15 + 3 * 30}{100} = 1,65$$

on calcule $\left| X_k - \sum_{j \neq k} X_j \right| = \min_{1 \leq i \leq 3} \left| X_i - \sum_{j \neq i} X_j \right|$

$$= \min \quad (2.05 , 1.25 , 1.25)$$

On peut donc tester indifféremment sur la valeur 2 ou la valeur 3.

3. Temps nécessaire au test de chaque condition.

Ce temps, nous l'avons vu, peut parfois être long puisqu'il peut nécessiter dans certains cas l'exécution de toute une sous-routine. Il est donc important de trouver un moyen de faire intervenir un facteur de pondération pénalisant les conditions dont le temps de test est long.

Pour cela nous introduisons dans la table une colonne supplémentaire dans laquelle l'on indiquera le temps d'exécution du test (t_i)

Une nouvelle modification est apportée dans la détermination de S_k pour tenir compte de cette information supplémentaire.

$S_k = t_k \sum_{i \in I_2} q_i P_i$ avec $I_2 = \{ i / R(k,i) = 0 \}$

Illustrons maintenant cet algorithme sur un exemple simple, celui de la figure T.16.

P_i	10	30	5	20	25		
q_i	1	2	2	1	6	t_i	S_i
Règles	1	2	3	4	5	↓	↓
C_1	1	1	0	2	2	60	6
C_2	2	1	3	-2	0	5	7,5
C_3	1	0	1	2	0	20	42
C_4	3	-2	1	1	2	11	0 ←

T.16

Les valences des conditions sont respectivement 2, 3, 2, 3 .
Après le calcul des q_i et des S_i, on remarque que le test doit porter sur C_4 .
Pour cette condition nous avons :

$X_1 = 0,9$

$X_2 = 1,5$

$X_3 = 0,7$

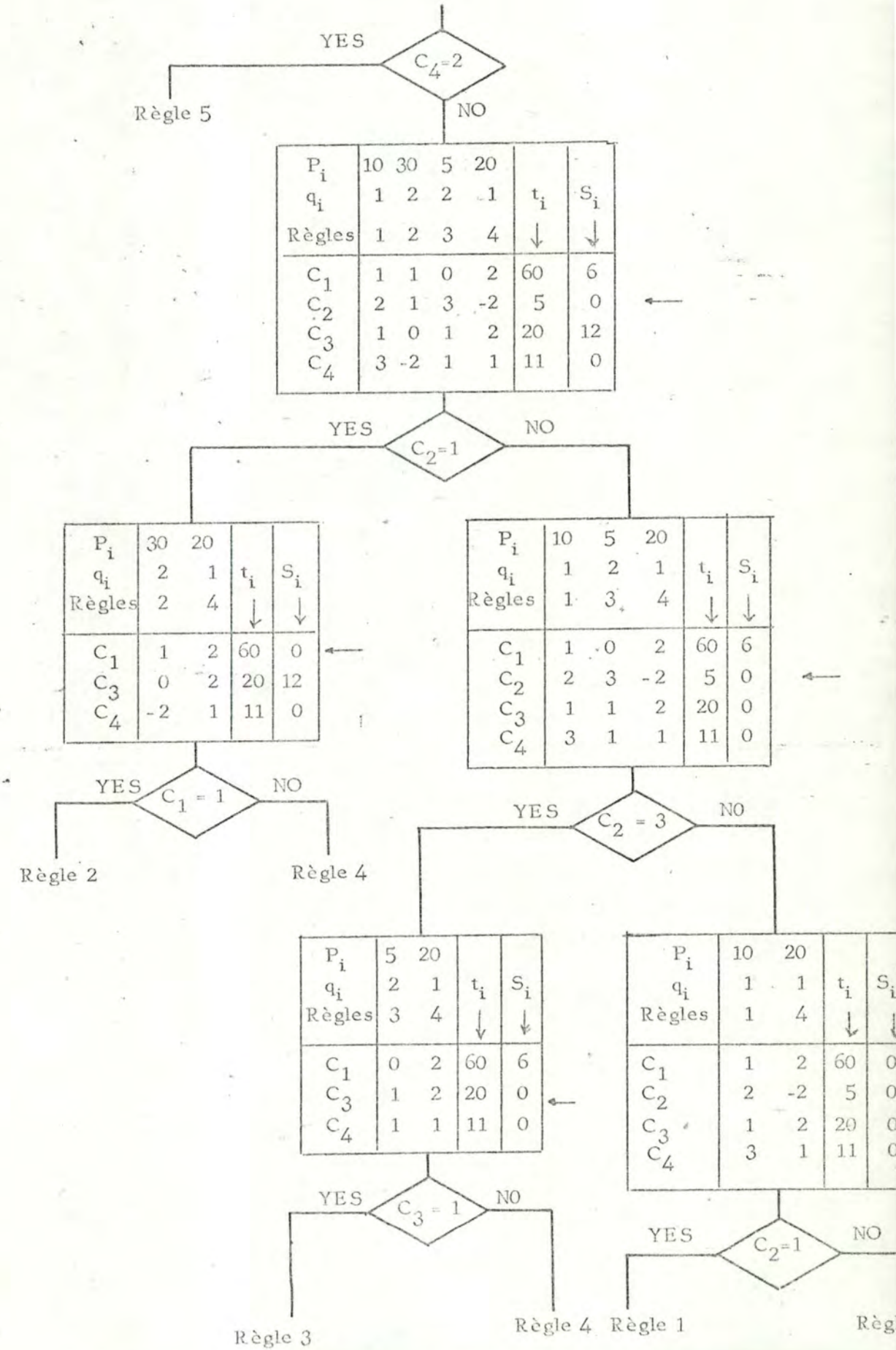
En appliquant la formule, on s'aperçoit que la valeur à prendre pour la condition C_4 est 2.

On exécute le test et on continue ainsi sur chacune des branches jusqu moment ou l'on tombe sur une table vide ou ayant une seule colonne (voir figure II.2).

REMARQUES :

1. La méthode précédente suppose que les cas impossibles ne se produisent jamais. Dans le cas contraire, il faut prendre un certain nombre de précautions supplémentaires.
   En effet, dans l'exemple de la figure II.2, la combinaison des conditions (2, 2, 1, 1) qui représente un cas non prévu dans la table renverrait à la règle 4.

2. Nous verrons aussi que la valeur choisie pour le test d'une condition n'est pas toujours acceptable (voir chapitre IV - page 10 - remarque 2) et qu'il faudra donc adjoindre à l'algorithme un certain nombre de règles supplémentaires.


YES
$C_4=2$
NO
Règle 5
P_i 10 30 5 20
q_i 1 2 2 1
Règles 1 2 3 4
t_i ↓
S_i ↓
C_1 1 1 0 2 60 6
C_2 2 1 3 -2 5 0
C_3 1 0 1 2 20 12
C_4 3 -2 1 1 11 0
YES
$C_2=1$
NO
P_i 30 20
q_i 2 1
Règles 2 4
t_i ↓
S_i ↓
C_1 1 2 60 0
C_3 0 2 20 12
C_4 -2 1 11 0
P_i 10 5 20
q_i 1 2 1
Règles 1 3 4
t_i ↓
S_i ↓
C_1 1 0 2 60 6
C_2 2 3 -2 5 0
C_3 1 1 2 20 0
C_4 3 1 1 11 0
YES
$C_1 = 1$
NO
Règle 2
Règle 4
YES
$C_2 = 3$
NO
P_i 5 20
q_i 2 1
Règles 3 4
t_i ↓
S_i ↓
C_1 0 2 60 6
C_3 1 2 20 0
C_4 1 1 11 0
P_i 10 20
q_i 1 1
Règles 1 4
t_i ↓
S_i ↓
C_1 1 2 60 0
C_2 2 -2 5 0
C_3 1 2 20 0
C_4 3 1 11 0
YES
$C_3 = 1$
NO
YES
$C_2=1$
NO
Règle 3
Règle 4
Règle 1
Règ

CHAPITRE III

ETUDES DES EQUATIONS LOGIQUES.

III. 1. Treillis de Boole et polynômes booléens.

III. 1.1. Définition des Treillis.

Sup-demi-treillis : on appelle ainsi tout ensemble ordonné dans lequel deux éléments quelconques admettent une borne supérieure.

Inf.-demi-treillis : notion duale de la précédente.

Treillis : c'est un ensemble ordonné qui est à la fois un sup.-demi-treillis et un inf.-demi-treillis.

III. 1.2. Polynômes booléens.

Définition : soit un ensemble $An = \{a_1, a_2, \ldots a_n\} \cup \{0,1\}$ à $n+2$ générateurs. On appelle polynôme booléen toute expression formée à partir des éléments de An et de symboles $+, ., '$.

Définissons sur $\mathcal{P}(An)$ une série d'axiomes.

1. Axiomes d'addition.

(1) $p + p = p \quad \forall p \in \mathcal{P}(An)$

(2) $p + q = q + p \quad \forall p, q \in \mathcal{P}(An)$

(3) $(p + q) + r = (p + q) + r \quad \forall p, q, r \in \mathcal{P}(An)$

(4) $p + 0 = p \quad \forall p \in \mathcal{P}(An)$

(5) $p + 1 = 1 \quad \forall p \in \mathcal{P}(An)$

2. Axiomes du produit.

(1') $p.p = p \quad \forall p \in \mathcal{P}(An)$

(2') $p.q = q.p \quad \forall p, q \in \mathcal{P}(An)$

(3') $(p.q).r = p(q.r) \quad \forall p, q, r \in \mathcal{P}(An)$

(4') $p.1 = p \quad \forall p \in \mathcal{P}(An)$

(5') $p.0 = 0 \quad \forall p \in \mathcal{P}(An)$

On remarque que les axiomes 1'-5' sont les axiomes duaux de 1-5.

3. Axiomes d'absorption.

(1") $p + pq = p$ $\quad \forall p, q \in \mathcal{P}(An)$

(2") $p(p+q) = p$ $\quad \forall p, q \in \mathcal{P}(An)$

4. Axiomes de complémentarité.

(1"') $q + q' = 1$ $\quad \forall q \in \mathcal{P}(An)$

(2"') $qq' = 0$ $\quad \forall q \in \mathcal{P}(An)$

(3"') $1' = 0$

5. Axiomes de distributivité.

(1"") $q(p + r) = qp + qr$ $\quad \forall p, q, r \in \mathcal{P}(An)$

Sur l'ensemble infini $\mathcal{P}(An)$ on peut maintenant définir une relation d'équivalence ($\equiv$)

$p \equiv q \overset{\text{déf.}}{\Longleftrightarrow}$ on peut passer de p à q par une suite finie de transformations utilisant chacune l'un des axiomes précédents ou une de leurs conséquences.

Il est évident que ....

Cette relation d'équivalence est compatible avec les opérations de somme, produit et complément.

III. 1.3. Algèbre de boole libre à n générateurs.

Définition : si on ne suppose aucune relation entre les générateurs l'ensemble quotient $L(An) = \mathcal{P}(An) / \equiv$ est appelé algèbre de boole libre à n générateurs.

Théorème 1. l'ensemble L (An) muni de la loi de composition interne $+ : L(An) \times L(An) \longrightarrow L(An) : (p, q) \rightsquigarrow p + q$ répondant aux axiomes de l'addition est un sup.-demi-treillis possédant un élément universel (élément maximum) 1 pour la relation d'ordre

$$R_1(p, q) \Longleftrightarrow p + q = q$$

Démonstration.

Il faut voir que la relation R_1 de L(An) est bien une relation d'ordre :

a. R_1 est réflexive : $R_1(p, p) \quad \forall p \in L(An)$

$\Longrightarrow p + p = p \quad \forall p \in L(An)$ (axiome 1)

b. R_1 est antisymétrique :

$R_1(p,q)$ et $R_1(q,p) \Longrightarrow p = q$

en effet $p + q = q$ et $p + q = p \Longrightarrow p = q$

car l'axiome 2 de l'addition donne $p + q = q + p$.

c. R_1 est transitive :

$R_1(p,q)$ et $R_1(q, r) \Longrightarrow R_1(p,r)$

en effet $p + q = q$ et $q + r = r \Longrightarrow$

$p + r = p + (q + r) = (p + q) + r = q + r = r$

en vertu de l'axiome 3.

De plus, on a $R_1(p, 1) \quad \forall p \in L(An)$ puisque l'axiome 5 donne $p + 1 = 1 \quad \forall p \in L(An)$

N.B. : Pour cette relation d'ordre R_1, on voit que

$\sup(q, p) = p + q$

en effet : on a $R_1(p, q + p)$ car $p + q + p = q + p$ (vrai)

$\Longrightarrow q + p$ majore p

de même on a $R_1(q, q + p) \Longrightarrow q + p$ majore q.

Il faut voir encore que si l'on a $R_1(p,r)$ et $R_1(q,r)$ alors $r / R_1(r, p+q)$

En d'autres mots que $p + q$ est le plus petit polynôme majorant p et q.

En effet : $\left.\begin{array}{l} R_1(p,r) \Longleftrightarrow p + r = r \\ R_1(q,r) \Longleftrightarrow q + r = r \end{array}\right\} \Longrightarrow p + (q + r) = r$

$\Longrightarrow (p + q) + r = r$

$\Longrightarrow R_1(p + q, r)$

ce qui contredit l'hypothèse.

REMARQUE : $\mathcal{P}(An)$ muni de la loi + n est pas un sup.demi-treilli pour la relation R_1, car celle-ci n'est qu'un pré-ordre sur $\mathcal{P}(An)$.

Théorème 2 : l'ensemble L (An) muni de la loi de composition interne : $L(An) \times L(An) \longrightarrow L(An) : (p,q) \rightsquigarrow p.q$ vérifiant les axiomes du produit est un Inf.-demi-treillis possédant un élément minimum 0 pour la relation d'ordre

$$R_2(p,q) \Longleftrightarrow pq = p$$

Démonstration. La relation R_2 étant la relation duale de R_1, la démonstration est identique à celle du théorème 1 en prenant cette fois les axiomes du produit.

N.B. : Pour cette relation d'ordre R_2 on voit aussi que

$$\text{Inf}(q, p) = p.q$$

Théorème 3.: L'ensemble L (An) muni des 3 lois de composition $+, \cdot, '$, (complémentation) et obéissant aux séries d'axiomes 1-5 est un treillis de Boole complémenté pour la relation d'ordre

$$R(q, p) \Longleftrightarrow q \leqslant p \overset{\text{def.}}{\Longleftrightarrow} q + p = p \overset{\text{def.}}{\Longleftrightarrow} qp = q$$

Démonstration.

1. C'est un treillis.

   Pour cela, il faudrait voir que les relations d'ordre R_1 et R_2 coïncident et sont égales à R.

   a. supposons que l'on a $R_1(p,q)$ et montrons que dans ce cas l'on a $R_2(p,q)$

   $R_1(p,q) \Longrightarrow p + q = q$ (1)

   si $R_2(p,q)$ est faux $\Longrightarrow pq \neq p$

   en remplaçant p par pq dans (1) on obtient alors $pq + q \neq q$

   ce qui contredit l'axiome 1"

   b. On a de même $R_2(p,q) \Longrightarrow R_1(p, q)$

   Conclusion $R_2 = R_1 = R$

2. C'est un treillis de Boole.

   On appelle treillis de Boole un treillis distributif. Cela est évident grâce à l'axiome 1"" de distributivité.

3. C'est un treillis de Boole complémenté.

   Les axiomes de complémentarité sont la définition même d'un treillis complémenté.

   - De plus, on a les relations suivantes :

$$p.q \leq p \leq p + q \qquad \forall\, p, q \in L\,(An)$$
$$p.q \leq q \leq p + q \qquad \forall\, p, q \in L\,(An)$$

REMARQUES :

1. Dans ce qui précéde, on a indiqué par p, q, r .. tantôt les polynômes appartenant à $\mathcal{P}$ (An), tantôt la classe de polynômes appartenant à L (An).
   Lorsque l'on écrit $p \in L\,(An)$, le lecteur comprendra qu'il s'agit de " la classe des polynômes représentée par le polynôme $p \in \mathcal{P}\,(An)$".

2. On dit souvent que L (An) est une algèbre de Boole libre à n générateurs. Le qualificatif "libre" traduit le fait qu'aucune condition booléenne autre que celles déductibles des axiomes n'a été supposée entre les générateurs.

III. 1.4. Monômes et co-monômes dans L (An).

Définitions :

1. On appelle monôme un élément de L (An) de la forme

   $\tilde{a}_{i1} \cdot \tilde{a}_{i2} \cdot \ldots \cdot \tilde{a}_{ik}$ avec $\tilde{a}_{ij} = a_{ij}$ ou a'_{ij} et

   $a_{ij} \in An$ et $k \leq n$

   N.B. : Nous supposerons que chaque générateur ne figure qu'une seule fois dans le monôme et que (O, 1) n'y figure pas. Ceci n'est évidemment pas une restriction puisque tout polynôme (et en particulier tout monôme) de $\mathcal{P}$ (An) simplifié par des transformations découlant des axiomes se trouve dans la même classe de L (An).

2. On appelle monôme canonique un monôme s'écrivant avec toutes les lettres de $An \setminus \{0, 1\}$

3. Co-monôme : élément de L (An) dont une expression est

   $\tilde{a}_{i1} + \tilde{a}_{i2} \cdots + \tilde{a}_{ik}$

   C'est la notion duale de monôme.

4. Co-monôme canonique :

Notion duale de monôme canonique.

Convention : 1 est un monôme
0 est un co-monôme

Exemple : soit $A_3 = \{a, b, c\}$

l'ensemble des monômes canoniques de A_3 est :
$\{abc, abc', ab'c, a'bc, a'bc', a'b'c, a'b'c', ab'c'\}$

D'une manière générale, pour n générateurs le nombre des monômes canoniques est 2^n.

Théorème 4 : tout polynôme booléen peut être décomposé en une somme de monômes canoniques ou de co-monômes canoniques.

Démonstration :

1. Cela est vrai pour un monôme m_1.
En effet, supposons que la lettre k soit absente dans le monôme, on remplace alors m_1 par $m_1k + m_1k'$.
Cette somme est égale à m_1 car $k + k' = 1$ (axiome 1''').
On remplace ensuite m_1k par m_2 et
m_1k' par m_3.
et on recommence la même opération pour ces 2 monômes.

Puisque le nombre des générateurs est fini, on arrivera nécessairement à une somme de monômes contenant chacun toutes les lettres de $An \setminus \{0,1\}$.

2. C'est vrai pour un polynôme quelconque.
En effet, le polynôme est une somme de monômes.
Il suffit d'appliquer le (1) pour chaque monôme.
On opère ensuite une simplification qui consiste à remplacer tous les monômes canoniques identiques par un seul monôme canonique (cela se base sur le fait que $p + p = p$).

Définition :

On dit que le monôme m_1 est multiple de m_2 si $m_1 \leq m_2$ c.à.d. si l'on a $m_1 m_2 = m_1$
ou $m_1 + m_2 = m_2$

(on dit encore que m_2 divise m_1).

Théorème 5 :

$m_1 \leqslant m_2$ si tous les générateurs figurant dans m_2 figurent dans m_1 sous la même forme.

Démonstration :

Considérons le polynôme $m_1 . m_2$.

Puisque tous les générateurs de m_2 figurent dans m_1 et que l'on a les axiomes du produit, on peut ramener $m_1 . m_2$ à m_1 après un nombre fini de transformation.

Conclusion : $m_1 . m_2$ et m_1 appartiennent à la même classe dans L(An)

$\Longrightarrow m_1 . m_2 = m_1$

$\Longrightarrow R(m_1, m_2)$

$\Longrightarrow m_1 \leqslant m_2$

Inversément : $m_1 \leqslant m_2 \Longrightarrow m_1 . m_2 = m_1$

$\Longrightarrow$ on ne peut trouver dans m_2 de générateur ne figurant pas sous la même forme dans m_1

En effet : 1. supposons $\tilde{a}_p \in m_2$ et $\notin m_1$

on a alors $m_1 . m_2 \tilde{a}'_p = m_1 \tilde{a}'_p$

or $\tilde{a}'_p \tilde{a}_p = 0 \Longrightarrow m_1 \tilde{a}'_p = 0$ impossible.

2. supposons $\tilde{a}_p \in m_2$ et $\tilde{a}'_p \in m_1$

$\Longrightarrow 0 = m_1$ pour la même raison.

Exemple : supposons $m_1 = a\ b\ c'\ d$

$m_2 = a\ c'$

$m_1 \leqslant m_2$ car $m_1\ m_2 = a\ b\ c'\ d = m_1$

III. 1.5. Fonctions booléennes.

a. Définitions.

- Une variable booléenne simple est une variable prenant ses valeurs dans $B = \{0, 1 / 0 < 1\}$.

- Par extension une variable booléenne à n dimensions est variable prenant ses valeurs dans B^n .

- Une fonction booléenne à n variables est une fonction $f = \{0,1\}^n \longrightarrow \{0,1\}$ représentée par une infinité de polynômes booléens algébriquement égaux de $\mathcal{P}(An)$, faisant donc partie de la même classe de L (An).

b. Monômes maximaux d'une fonction booléenne.

- On appelle monôme d'une fonction booléenne f tout monôme m_1 de L(An) majoré par f c'est-à-dire tel que $m_1 \leq f$ ou encore que $m_1 + f = f$.

  Exemple : soit f = a b c + a b d c + ad

  Chacun des termes de la somme est un monôme de la fonction booléen f puisque par exemple :

  a b c + f = f.

- m_1 est un monôme maximal de f si en plus on a la condition

  $\forall m\ (m_1 \leq m \leq f \Longrightarrow m = m_1)$

  Exemple : supposons m_1 = a b c et $m_1 \leq f$.

  m_2 = abcd' et $m_2 \leq f$

  m_2 étant multiple de m_1 ($m_2 \leq m_1$), m_2 ne saurait être un monome maximal de f.

  m_1 le sera si $\nexists m \leq f$ qui soit multiple de m_1.

- Base d'une fonction booléenne f. est l'ensemble des monômes de f dont la somme est égale à f.

- Base canonique de f : ensemble des monômes canoniques dont la somme est égale à f .

- Forme canonique de f : est la somme (à une permutation près) des monômes canoniques de la base canonique de f .

III. 1.6. Consensus d'un ensemble de monômes.

On dit qu'une variable est monoforme dans une expression si elle n'apparaît dans celle-ci que sous une forme (soit directe, soit complémentée). Dans le cas contraire elle est dite Biforme.

Considérons une fonction f : on peut la représenter (après suppression éventuelle de paranthèses) sous la forme :

$$f = \sum_{i=1}^{P} m_i \quad (m_i = \text{monômes formant la base de } f)$$

Chaque m_i peut s'écrire sous la forme $\alpha_i \cdot \beta_i$

où α_i = ensemble des variables monoformes de f.

β_i = ensemble des variables biformes de f.

Exemple : $f = ab + cb'$

$$m_1 = \alpha_1 \beta_1 = (a)(b)$$

$$m_2 = \alpha_2 \beta_2 = (c)(b') .$$

on peut donc écrire

$$f = \sum_{i=1}^{P} m_i = \sum_{i=1}^{P} \alpha_i \beta_i .$$

Théorème 6 : le produit des variables monoformes de $f = \sum_{i=1}^{P} m_i$ est multiple de f ssi la somme de ses variables biformes vaut 1.

$$\alpha = \prod_{i=1}^{P} \alpha_i \leq \sum_{i=1}^{P} m_i \Longleftrightarrow \sum_{i=1}^{P} \beta_i = 1$$

Démonstration:

1. $\Longrightarrow$

$$\alpha \leq \sum_{i=1}^{P} m_i \longrightarrow \alpha \left(\sum_{i=1}^{P} m_i \right) = \alpha$$

$$\longrightarrow \prod_{i=1}^{P} \alpha_i \left(\sum_{i=1}^{P} \alpha_i \beta_i \right) = \alpha$$

$$\alpha \cdot \sum_{i=1}^{P} \beta_i = \alpha \Longrightarrow \alpha \leq \sum_{i=1}^{P} \beta_i$$

Si on donne aux variables booléennes de α des valeurs telles que : $\alpha = 1$ on obtient $1 \leq \sum_{i=1}^{P} \beta_i$ puisque $\alpha \cap \sum_{i=1}^{P} \beta_i = \emptyset$

Puisque 1 est un élément universel :

$$\Longrightarrow \sum_{i=1}^{P} \beta_i = 1$$

2. $$\Longleftarrow \sum_{i=1}^{P} \beta_i = 1 \Longrightarrow \alpha \cdot \left(\sum_{i=1}^{P} m_i\right) = \alpha \left(\sum_{i=1}^{P} \alpha_i \beta_i\right)$$

$$= \alpha \left(\sum_{i=1}^{P} \beta_i\right)$$

$$= \alpha$$

$$\Longrightarrow \alpha \leq \sum_{i=1}^{P} m_i$$

Reprenons l'exemple précédent :

$f = ab + cb'$

on a ici $b + b' = 1 \Longrightarrow \alpha = ac \leq f$

Théorème 7.

Pour que l'inégalité $\alpha \leq f$ soit irréductible c'est-à-dire telle que si on supprime un des monômes de f, alors f ne majore plus α, il faut et il suffit que :

$$\sum_{i=1}^{P} \beta_i = 1$$

soit irréductible (c'est-à-dire qu'on ne peut supprimer un des β_i sans rompre l'égalité).

Démonstration : par l'absurde.

1. Supposons $\alpha \leq f$ réductible et montrons que cela implique que :

$\sum_{i=1}^{P} \beta_i = 1$ est réductible,

soit $\beta = \sum_{i=1}^{P} m_i$ et supposons que m_1 puisse être supprimée de β sans changer l'inégalité $\alpha \leq \beta$,

on a $\alpha \leq \sum_{i=2}^{P} m_i$

$$\Longrightarrow \alpha \cdot \sum_{i=2}^{P} m_i = \alpha$$

or $\alpha = \prod_{i=1}^{P} \alpha_i$ (produit des variables monoformes)

$$\Longrightarrow (\alpha_1 . \alpha_2 \ldots . \alpha_p)(\alpha_2 \beta_2 + \ldots + \alpha_p \beta_p) = \alpha .$$

$$\Longrightarrow \alpha \cdot \sum_{i=2}^{P} \beta_i = \alpha .$$

$$\Longrightarrow \alpha \leq \sum_{i=2}^{P} \beta_i$$

On peut choisir les variables de α de sorte que $\alpha = 1$; l'intersection entre α et $\sum_{i=2}^{P} \beta_i$ étant vide on obtient alors $\sum_{i=2}^{P} \beta_i = 1$

or $\sum_{i=1}^{P} \beta_i = 1 \Longrightarrow$ l'inégalité est réductible.

2. De même si $\sum_{i=1}^{P} \beta_i = 1$ est réductible, montrons alors que $\alpha \leq \beta$ est réductible,

supposons β_1 le terme que l'on peut simplifier

on obtient $\sum_{i=2}^{P} \beta_i = 1$

$$\Longrightarrow \alpha_i \sum_{i=2}^{P} m_i = \alpha \cdot \sum_{i=2}^{P} \beta_i = \alpha$$

$\Longrightarrow$ l'inégalité est réductible.

Définition : si la somme $\beta = \sum_{i=1}^{P} \beta_i$ des variables biformes d'un ensemble $M = \{ m_1, \ldots m_p \}$ de monômes est égale à 1 et est irréductible, on dit alors que le produit des variables monoformes des monômes de M est le consensus de M et on écrit :

$$\alpha = C(M)$$

Exemple : $f = abc + ab' + ac'$

$$\beta = \sum_{i=1}^{3} \beta_i = bc + b' + c' = 1 \text{ irréductible}$$

$\Longrightarrow \alpha = a$ est le consensus de $\{ abc, ab', ac' \}$.

Théorème 8.

soit $f = \sum_{i=1}^{P} m_i = \sum_{i=1}^{P} \alpha_i \beta_i$

si $\exists$ m tel que $m \leq f$ est irréductible alors $m \leq \alpha \leq f$,

ou α est le consensus de M et monome maximal de f.

Démonstration :

$$m \leq f \Longrightarrow m.f = m$$

$$\Longrightarrow \sum_{i=1}^{P} m\alpha_i \beta_i = m$$

supposons $\exists\, a \in \alpha_i \quad \forall i \in I \subset [1, P]$

et que $a \notin m$

alors en posant $a = 0$ on obtient

$\sum_{i \notin I} m\alpha_i \beta_i = m$ et l'inégalité est réductible (contraire à l'hypothèse) $\longrightarrow$ toutes les variables monoformes de f doivent se trouver dans m

$$\Longrightarrow m \leq \alpha = \prod_{i=1}^{P} \alpha_i$$

$$m \leq f \Longrightarrow \sum_{i=1}^{P} \quad m\beta_i = m \quad (\text{car } m.\alpha = m)$$

supposons alors que $\sum_{i=1}^{P} \beta_i$ soit réductible et que l'on puisse supprimer β_1

$$\Longrightarrow m \sum_{i=2}^{P} \beta_i = m$$

$\Longrightarrow m \sum_{i=2}^{P} \alpha_i \beta_i = m \Longrightarrow m \leq \gamma$ est réductible

contraire à l'hypothèse

$\Longrightarrow m \leq \sum_{i=1}^{P} \beta_i$ et $\sum_{i=1}^{P} \beta_i$ est irréductible.

De plus, l'intersection des variables de $\sum_{i=1}^{P} \beta_i$ et de m est vide sinon $m \leq \gamma$ serait réductible

$\Longrightarrow$ puisqu'on peut choisir des valeurs aux variables de m tel que $m = 1$

on a donc $\sum_{i=1}^{P} \beta_i = 1$ est irréductible.

$\Longrightarrow m \leq \alpha \leq \gamma$ et α est consensus (définition du Consensus).

Reste à voir que α est monôme maximal de γ

c.à.d. si $\forall\ m' / (\alpha \leq m' \leq \gamma)$ alors $\alpha = m'$

On vient de voir que si $m \leq \gamma$ était irréductible alors on avait $m \leq \alpha \leq \gamma$ et $\alpha \leq \gamma$ était irréductible.

De même ici on sait que $\alpha \leq \gamma$ est irréductible, cela implique que $m' \leq \gamma$ est irréductible.

Appliquons alors le théorème à m' et l'on obtient

$m' \leq \alpha \leq f$

$\Longrightarrow m' = \alpha$

C.Q.F.D.

Corollaire : si m est un monôme maximal de $\sum_{i=1}^{P} m_i$, alors $\exists\ P \subset E$ tel que $m = \mathcal{C}(P)$.

Démonstration :

1. si $m \leq \sum_{i=1}^{P} m_i$ est irréductible alors d'après le théorème précédent $m = \mathcal{C}(E)$

2. si $m \leq \sum_{i=1}^{P} m_i$ est réductible alors $\exists\ I \subset [1, P]$ tel que $m \leq \sum_{i \in I} m_i$ est irréductible et on se retrouve dans le cas (1).

III. 2. Les algorithmes de simplification de fonction.

Le corollaire précédent permet grâce à la recherche des consensus des parties de $E = \{ m_1 \dots m_p \}$ (s ils existent) de trouver tous les monômes maximaux de la fonction $f = \sum_{i=1}^{P} m_i$. En effet, tout monôme maximal étant égal au consensus d'une partie de l'ensemble E, il suffit de calculer le consensus de toutes les parties de E (s'ils existent) et de prendre uniquement ceux qui sont monômes maximaux.
Cependant, lorsque p est grand, cette recherche peut être fort longue, ce qui a nécessité la recherche d'algorithmes plus performants.

Théorème 9 :

Supposons $P = \{ m_i / i \in [1, p-1] \}$ avec $C(P)$ existe et supposons que m_p sont multiple de $m_i \quad \forall i \in [1, p-1]$ alors :

$m_p \leq C(P_1)$ avec $P_1 \subset P$

Démonstration :

On sait que $C(P) \leq \sum_{i=1}^{P-1} m_i$

or $m_p \cdot \sum_{i=1}^{P-1} m_i = m_p$ par hypothèse

$$\Longrightarrow m_p \leq \sum_{i=1}^{P-1} m_i \quad (1)$$

Supprimons dans $\sum_{i=1}^{P-1} m_i$ des monômes de façon à ce que (1) soit irréductible. Soit P_1 l'ensemble des monômes restants on a en fonction du théorème 8 $m_p \leq C(P_1)$

Conclusion :

On peut supprimer dans l'ensemble des monômes ceux qui sont multiples des autres, sans perdre aucun monôme maximal de la fonction de départ.

1ère amélioration pour la recherche des monômes maximaux.

Lors de la recherche des consensus on peut à tout moment supprimer dans l'ensemble des monômes de f et l'ensemble des consensus ceux qui sont multiples des autres puisque cela ne fait perdre aucun monôme maximal (Th. 9) et que cela ne modifie pas la fonction f par définition des multiples.

Cette opération s'appelle suppression de multiples.

2ème amélioration .

Remarquons encore que si l'on range les monômes de $\mathcal{f}$ par ordre d'importance du nombre de variables , pour voir si un monôme (m) est multiple d'un autre, il suffit de voir :

1. si m est divisible par les monômes m_i ayant au plus autant de lettres que lui,

2. si m divise les monômes m_i ayant strictement plus de lettres que lui.

Remarques :

1. Il est intéressant de constater que le consensus d'un ensemble de monômes est multiple des monômes canoniques de cet ensemble car dans $\mathcal{C}$ (M) ne figure que des variables monoformes (évidemment de même type que dans l'ensemble des monômes). Ceux-ci peuvent donc toujours être supprimés.

2. Si $m = \{ m_1, \ldots, m_p \}$ ne contient pas de variables biformes, il n'existe des consensus que pour les éléments de $\mathcal{P}$ (M), formé d'un seul monôme.

En effet, dans ce cas $\beta_i = 1$ pour $\forall i \in [1, P]$

et donc $\sum_{i=1}^{P} \beta_i = 1$ n'est irréductible que pour $P > 1$.

En vertu de la définition du Concensus, il n'existe pas de consensus pour $P > 1$.

Exemple de recherche des monômes maximaux.

Soit $\mathcal{f} = abc + ab' + a'bc + ac'$

considérons $E = \{ abc, ab', a'bc, ac' \}$.

1. la 1ère suppression de multiples laisse $\mathcal{f}$ inchangée.
2. recherche des consensus de tous les sous-ensembles de $\mathcal{f}$ comprenant au moins 2 éléments puisque le consensus d'un monôme est le monôme lui-même. (voir tableau)

3. On effectue alors une suppression de multiple sur

   $E_1 = E \cup \{ ac, bc, ab, a \}$

   On obtient $\mathcal{f} = a + bc$ (monômes maximaux de $\mathcal{f}$).

Elément de $\mathcal{P}(E)$	Variables biformes	$\leq \beta_i$	$\alpha = \prod_{i=1}^{P} \alpha_i$
$\{abc, ab'\}$	b	$b + b' = 1$ irréductible	ac
$\{abc, a'bc\}$	a	$a + a' = 1$ "	bc
$\{abc, ac'\}$	c	$c + c' = 1$ "	ab
$\{ab', a'bc\}$	a, b	$ab' + a'b \neq 1$	Néant
$\{ab', ac'\}$		$1 + 1 = 1$ réductible	Néant
$\{a'bc, ac'\}$	a, c	$a'c + ac' \neq 1$	Néant
$\{abc, ab', a'bc\}$	a, b	$ab + ab' + a'b \neq 1$	Néant
$\{abc, ab', ac'\}$	b, c	$bc + b' + c' = 1$ irréd.	a
$\{abc, a'bc, ac'\}$	a, c	$ac + a'c + ac' \neq 1$	Néant
$\{ab', a'bc, ac'\}$	a, b, c	$ab' + a'bc + ac' \neq 1$	Néant
$\{abc, ab', a'bc, ac'\}$	a, b, c	$abc + ab' + a'bc + ac' \neq 1$	Néant

Cet exemple montre qu'en ajoutant à la fonction de départ f, la somme des consensus des parties de f, on peut réduire f à a + bc par suppression de multiples.

N.B. : L'adjonction à f de la somme (P) des consensus ne change rien à f puisque $P \leq f \xrightarrow{\text{déf}} P + f = f$.

Définition :

On appelle base principale complète de f, la somme de tous les monômes maximaux de f.

III. 2.1. Recherche de la base principale d'une fonction f par la méthode linéaire.

soit $f = \sum_{i=1}^{P} m_i$

1. Ranger les monômes par ordre croissant du nombre de lettres.
2. Effectuer une première suppression de multiples.
   C.à.d. supprimer tous les monômes m_i / $\exists$ $j \neq i$ avec $m_i \leq mj$
3. I = 2
   m_2 = monôme pivot.
4. Calcul du consensus de $M_{ki} = \{ m_k , m_i \}$
   mettre le consensus dans la liste des monômes.
5. Effectuer une suppression de multiples sur la liste (le consensus lui-même peut-être supprimer au cas où il est multiple d'un monôme de la liste).
6. Faire I = I + 1
   Aller en 4.

On s'arrête lorsque les 2 derniers monômes de la liste ne donnent plus de nouveau monôme par consensus.

Justification de l'algorithme.

voir page 257 dans [F L 2] .

III. 2.2. Recherche des monômes maximaux par double dualisation.

Supposons $f = \sum_{i=1}^{P} m_i$

1. Calculer $f^{*} = \prod_{i=1}^{P} M_i$

   ou M_i est le co-monôme correspondant à m_i, c'est-à-dire la somme des variables directs ou complémentées apparaissant dans m_i .

2. Opérer une suppression de multiple.
3. Calculer $f^{**} = (\prod_{i=1}^{P} M_i)^{*} = \sum_{i=1}^{P} M_i^{*}$

   On obtient alors tous les monômes maximaux de f après une nouvelle suppression de multiple.

Justification. voir page 263 dans [FL2] .

Les deux algorithmes précédents permettaient de trouver l ensemble des monômes maximaux d'une fonction booléenne.
Cet ensemble est appelé base principale complète de f.
La fonction f représentée par cette somme n'est cependant pas encore la forme minimale de f.
Il reste encore à rechercher parmi ces monômes maximaux une base principale irréductible, c'est-à-dire une base ou l'on ne peut supprimer un monôme sans changer la valeur de la somme.

III. 2.3. Fonctions de présence.

Considérons l'ensemble $M_p = \{ m_1, \ldots m_p \}$ comme la base principale complète de f obtenue par un des deux algorithmes précédents.
Soit g une fonction booléenne.

On définit la fonction de présence de g comme étant une fonction

$$\Lambda g = \sum_{k=1}^{R} (\tilde{1}\ \tilde{2} \ldots \tilde{p})_k$$

ou - R représente le nombre d'éléments de $\mathcal{P}(M_p)$ c'est-à-dire de l'ensemble des parties de M_p, qui sont parties majorantes de g.

$E = \{ m_{i1}, \ldots m_{ik} \} \subset M_p$ est partie majorante de g

ssi $g \leq \sum_{r=1}^{k} m_{ir}$

- $(\tilde{1}\ \tilde{2} \ldots \tilde{p})_j$ représente le monôme canonique de l'algèbre de boole libre L (1, ... p) correspondant à la j^e partie majorante $E \subset M_p$ de la fonction g.
- $\tilde{k} = k$ si le monôme maximal $m_k \in M_p$ est présent dans la j^e partie majorante.
- $\tilde{k} = k'$ s'il est absent.

Exemple :
soit $M_3 = \{ m_1, m_2, m_3 \} = \{ ab, b'c, ac \}$ et $g = ab'c + abc'$.
Parmi les 2^3 parties de M_3, il y en a 3 qui sont des parties majorantes de g à savoir :

$$\Lambda_1 = \{ ab, b'c \}$$

$$\Lambda_2 = \{ ab, ac \}$$

$$\Lambda_3 = \{ ab, b'c, ac \}$$

Puisque, par exemple, $g \leqslant ab + b'c$.

Donc R = 3 et l'on a $\lambda g = \overset{3}{\underset{k=1}{\leqslant}} (\hat{1}\ \tilde{2}\ \tilde{3})_k$

Puisque la partie majorante A_1 n'est formée que des monômes m_1 et m_2 de M_3 on a que $(\hat{1}\ \hat{2}\ \tilde{3})_1 = 123'$

et ainsi de suite

l'on obtient finalement que $\lambda g = 123' + 12'3 + 123$.

Remarques :

1. si $1\,2 \ldots k\,k+1' \ldots p'$ est un monôme de λg, cela signifie que $\{ m_1, \ldots m_k \}$ est une partie majorante de g.

   Cela peut s'exprimer simplement par $1\,2 \ldots k$.

   On peut donc supprimer dans les monômes de λ g les variables qui sont complémentées sans changer λ g.

2. si $1\,2 \ldots k$ est un monôme de λ g alors $1\,2 \ldots k\ \tilde{k+1} \ldots \tilde{p}$ est un monôme de λ g puisque si P est une partie majorante de g alors $\forall X \ / \ P \subset X \subset M_p$
   X est une partie majorante de g.

3. De la remarque 1 découle immédiatement le fait que les monômes maximaux de λ g ne contiennent aucune variable complémentée.

Théorème 10.

Les parties majorantes irréductibles correspondent aux monômes maximaux de λ g et inversément.

Démonstration.

1. Supposons une partie majorante irréducitble $\{ m_1 \ldots . m_k \}$
   A cette partie correspond le monôme $1\,2 \ldots k$ de λ g.
   Supposons que m_k puisse être supprimé tout en gardant le fait que :
   $$g \leq \overset{k-1}{\underset{i=1}{\leqslant}} m_i$$

$\Longrightarrow$ 1 2 ... k - 1 $\in \lambda$ g.

or 1 2 ... k $\leq$ 1 2 ... k-1

$\Longrightarrow$ 1 2 ... k n'est pas monôme maximal.

2. $\{ m_1, \ldots m_k \}$ étant une partie majorante irréductible le monôme associé 1 ... k est maximal puisqu'on ne peut trouver un monôme de λ g > 1 .. k sinon cela signifierait que la partie majorante correspondant à ce monôme serait inclue dans $\{ m_1, \ldots . mk \}$.

Nous allons voir maintenant comment déterminer $\lambda \mathcal{F}$.
Cela nous permettra grâce à la recherche des monômes maximaux de $\lambda \mathcal{F}$ de trouver toutes les bases principales irréductibles de $\mathcal{F}$.

III. 2.4. Calcul des fonctions de présence des monômes maximaux.

Définissons d'abord la fonction de consensus CONS associé à la base principale complète M_p de $\mathcal{F}$

$$CONS = \sum_{i=1}^{P} i\, m_i$$

Thèorème 11.

$\{ {}^1m_1, \ldots {}^Pm_p \}$ est une base principale irréductible de CONS.

Démonstration : puisque 1, ... P sont des variables monoformes de toutes les parties de $\{ {}^1m_1, \ldots {}^Pm_p \}$, les nouveaux monômes obtenus par consensus en appliquant un des 2 algorithmes précédents contiennent au moins deux variables de présence . $\Longrightarrow$ aucun monôme de la base initiale n est multiple de ceux-ci.

$\Longrightarrow$ les monômes de la base initiale sont maximaux.
De plus, aucun d'eux ne peut être majoré par la somme des autres car ils ont tous une variable $\neq$ entre eux (1, 2, ... P)

Théorème 12 :

Si nous calculons la base principale complète de CONS et que nous mettons en facteur les monômes m_i partout ou ils figurent, nous obtiendrons

$$CONS = \sum_{i \in I} m_i \, \lambda_{m_i}$$

ou λ_{m_i} est la fonction de présence de m_i

Démonstration :

Voir théorème 1 p. 289 [FL2.]

En résumé :

Pour obtenir les fonctions de présence des monômes maximaux il faut :

1. déterminer la base principale complète de f.
2. déterminer CONS à l'aide de la base principale complète de f.
3. déterminer la base principale complète de CONS.
4. mettre chaque m_i en facteur.

On trouve ainsi en facteur avec chaque monôme maximal de f sa fonction de présence

Exemple :

- $f = ab' + ac + bc + a'bd$
  ou les monômes forment la base principale complète B.
- CONS = $ab'1 + ac2 + bc3 + a'bd4$
  ou $\{ ab'1, ac2, bc3, a'bd4 \}$ est une base principale irréductible de CONS (théorème 11).
- base principale complète de CONS est :
  $\{ ab'1, ac(2 + 13), bc3, a'bd4, bcd24 \}$

et on a que $\lambda_{ab'} = 1$

$\lambda_{ac} = 2 + 13 \Rightarrow$ cela signifie qu'il y a 2 sous-ensembles de B = $\{ ab', ac, bc, a'bd \}$ majoré par ac qui sont respectivement

ac et ab' + bc

$\lambda_{bc} = 3$

$\lambda_{a'bd} = 4$

Il peut arriver comme dans cet exemple que l'on obtienne dans la base principale complète de CONS des monômes qui ne soient pas monômes maximaux de f, c'est le cas de bcd.

On ne peut donc pas dire pour lui que $\lambda_{bcd} = 24$.

III. 2.5. Algorithme de recherche des bases principales irréductibles par majoration des monômes maximaux.

1. Partir du résultat obtenu par un des 2 algorithmes de recherche de tous les monômes maximaux de f soit :

$$\{ m_1, \ldots m_p \}$$

2. Calculer à l'aide de la fonction CONS $= \sum_{i=1}^{P} i\, m_i$

les fonctions de présence λ_{m_i} pour chacun des monômes m_i.

3. Calculer $\lambda_f = \prod_{i=1}^{P} \lambda_{m_i}$

4. Opérer des simplifications grâce aux axiomes

$x(x + y) = x$

$x + xy = x$

et en faisant éclater

$x + yz$ en $(x + y)(x + z)$

Exemple :

Reprenons l'exemple précédent,

on avait $\lambda_{ab'} = 1$

$\lambda_{ac} = 2 + 13$

$\lambda_{bc} = 3$

$\lambda_{a'bd} = 4$

on obtient donc $\lambda f = 1\,(2 + 13)\,34$

$= 1\,(2 + 1)\,(2 + 3)\,34$

$= 1\,3\,4$

donc la seule base principale irréductible est $\{ ab', bc, a'bd \}$

Justification :

Il reste pour démontrer la validité de l'algorithme à voir que

$$\lambda f = \prod_{i=1}^{P} \lambda m_i$$

on a $f = \sum_{i=1}^{P} m_i$

Tout d'abord, chaque fonction de présence λ_{m_i} est définie puisque $m_i \leq f$

1. $\lambda f \leq \prod_{i=1}^{P} \lambda m_i$

En effet tout monôme $1, \ldots j$ de λf détermine $\{ m_1, \ldots m_j \}$ partie majorante de f qui à fortiori est la partie majorante de m_i puisque $m_i \leq f$.

Donc $\forall i$ on a $1 \ldots j \leq \lambda m_i$

$$\Longrightarrow 1 \ldots j \leq \prod_{i=1}^{P} \lambda m_i \qquad \forall\, 1 \ldots j \in \lambda f$$

$$\Longrightarrow \lambda f \leq \prod_{i=1}^{P} \lambda m_i$$

2. $\lambda f \geq \prod_{i=1}^{P} \lambda m_i$

Prenons P monômes appartenant respectivement à $\lambda m_1 \ldots \lambda m_p$ chacun de ces monômes détermine une partie majorante de m_i ; le produit de ces monômes déterminera la réunions de ces p parties majorantes $\Longrightarrow$ ce sera certainement une partie majorante de f (car si $m_1 \leq g'$ et $m_2 \leq g''$ alors $m_1 + m_2 \leq g' + g''$) et donc le monôme formé appartient à λf.

C.Q.F.D.

CHAPITRE IV. - EXPLICATION DETAILLEE DU GENERATEUR

§ 1 : Programme de liaison.

Ce programme exécute soit à partir d'une table de décision polyvalente, soit à partir d'un système d'équations logiques, une analyse approfondie permettant de déceler des cas redondants, incompatibles et oubliés. Dans le cas où l'analyse ne décèle ni redondance, ni incompatibilité, il génère alors un programme "assembler" correspondant à la logique du problème. Les routines permettant d'exécuter ces opérations sont au nombre de 6.

Il s'agit de PROB 1 : normalisation des équations logiques.
PROB 2 : transformation des équations normalisées.
PROB 3 : simplification, complémentation et produit des équations logiques.
PROB 4 : transformation des équations logiques en table de décisi
PROB 5 : génération des instructions en Assembler correspondant à la table de décision.
PROB 6 : analyse de la table de décision.

Le programme chargé d'appeler ces différentes routines et de décider l'emplacement de chargement de celles-ci en mémoire, est géré par le programme EQUALOG.

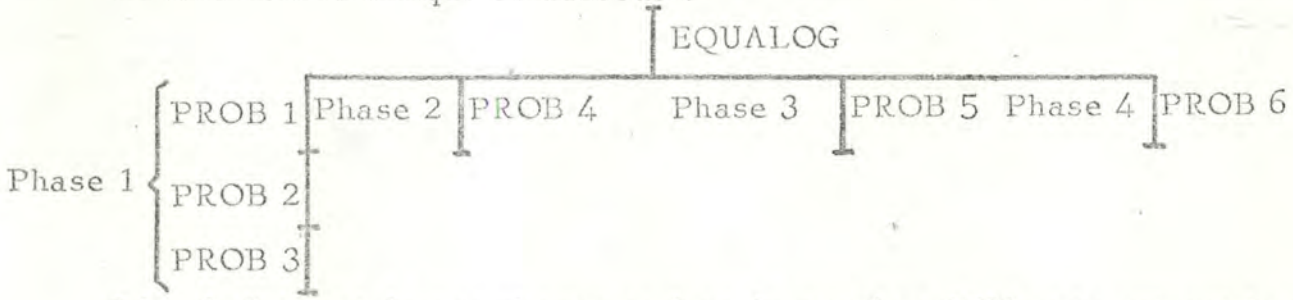
La seule routine commune aux deux problèmes est PROB 4.

Le générateur peut être décomposé en quatre parties indépendantes, qui sont :

I. Analyse des équations logiques (PROB 1, PROB 2, PROB 3).
II. Transposition des équations logiques en table (PROB 4).
III. Génération des instructions en Assembler (PROB 5).
IV. Analyse de la table de décision (PROB 6).

La figure IV-1 montre la conception générale du programme de liaison.

Seul le programme EQUALOG sera constamment en mémoire centrale, les autres programmes seront en overlay. L'endroit de chargement de ceux-ci se fera comme indiqué ci-dessous :



L'enchaînement de ces phases se fera de manière différente suivant le cas : lorsque l'on part des équations logiques, nous aurons les phases successives 1, 2, 3, tandis qu'au départ des tables de décision, nous aurons l'enchaînemer 4, 3. Il faut ajouter encore que le programme peut être interrompu au niveau de chaque phase lorsque le résultat de celle-ci ne permet pas d'assurer la validité de la phase suivante.

PROGRAMME DE LIAISON

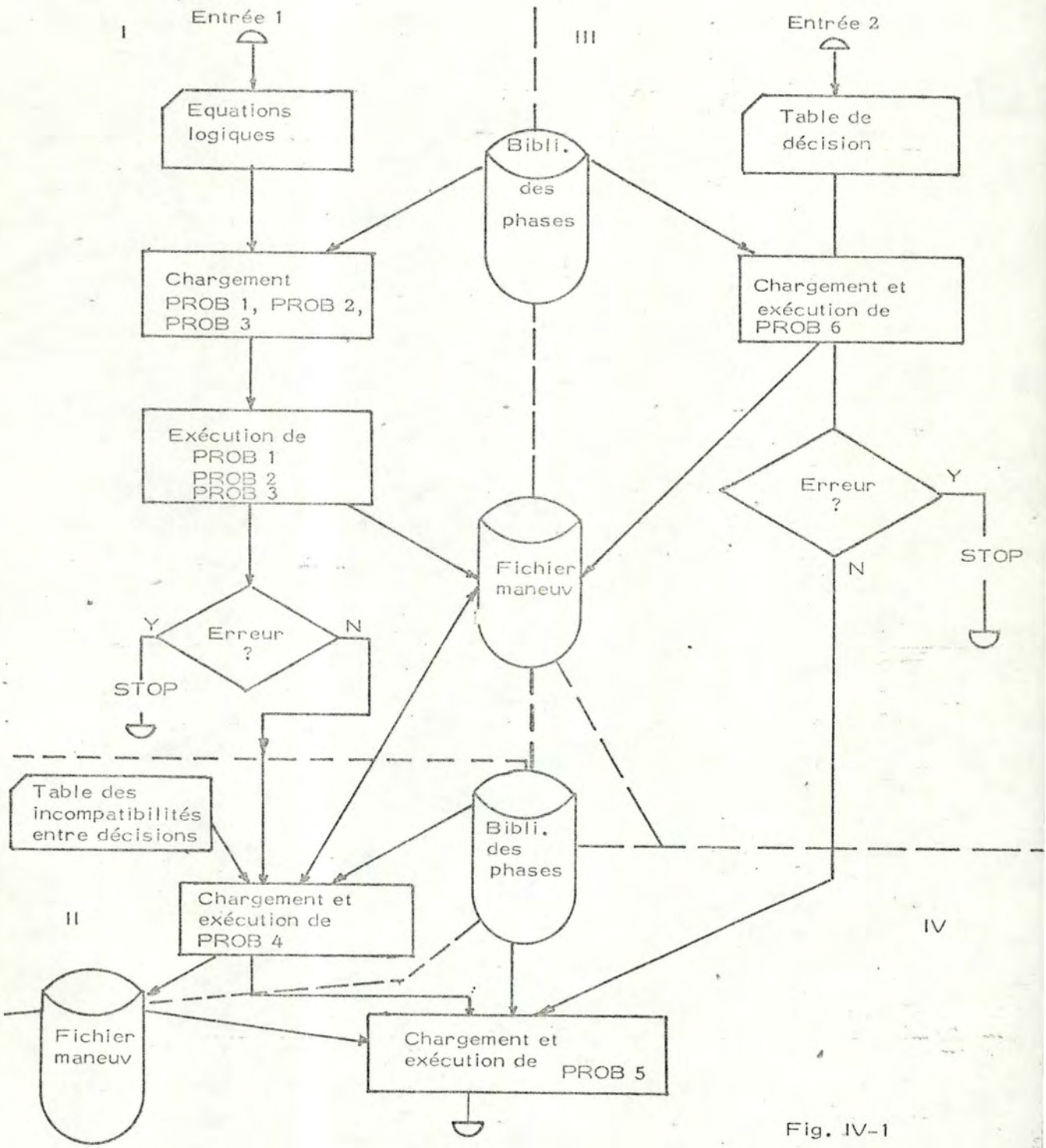


Fig. IV-1

------: représente les parties indépendantes.

§ 2 : Cas d'un problème transposé en table de décision.

Les informations contenues dans les tables de décision sont rentrées à l'aide de cartes perforées (annexes 1 et 2). Nous avons vu que dans ce cas le programme se divisait en deux parties : analyse de la table et génération des instructions. La génération des instructions n'aura lieu qu'au cas où l'analyse ne détectera ni redondance, ni incompatiblité.

2.1. PROB 6 - Analyse de la table.

Cette phase est elle-même divisée en un certain nombre de parties :

a) Contrôle de syntaxe des cartes.

Lors de la lecture des cartes, un certain nombre d'erreurs peuvent être détectées, qui occasionnent un arrêt du traitement. Ce sont :

stop 1 : - col 3 de la carte d'en-tête est différente de blanc au zéro.
- valeur inférieure à zéro dans les colonnes 6 et 7.

stop 2 : - col 1 et 2 des cartes "condition" et "décision" ont une valeur différente des col 1 et 2 de la carte d'en-tête.

stop 3 : - erreur de séquence pour les cartes "condition" et "décisi (col 4 et 5).

stop 4 : - pas de carte "condition" ou "décision" (col 3).

stop 5 : - erreur de séquence (carte C après carte D).

stop 6 : - la zone "condition" (col 11 et suivante) contient une valeur supérieure (en valeur absolue) à la valence (col 6 et 7) de cette condition.

stop 7 : - erreur de séquence (carte D avant carte C).

stop 8 : - détection de fin de fichier (/*) avant qu'une carte D soit lu

stop 9 : - nombre de possibilités (col 6 et 7) inférieur à zéro.

b) Eclatement de la matrice des "conditions".

Cette matrice (R) fut garnie, lors de la lecture des cartes par la zone "conditions" (col 11 et suivante), des cartes "condition". Pour exécuter l'éclatement de celles-ci, on calcule d'abord le nombre de cas NC représentés par chaque règle (chaque colonne de R). On applique ensuite les formules vues au chapitre II , § 1, pour éclater chacune des règles. Nous obtenons finalement une matrice A représentant l'ensemble des cas (et non plus des règles) de la table initial

c) Détection des cas redondants et incompatibles.

Au départ de la matrice donnant l'ensemble des cas envisagés, on calcule pour chacun d'eux (chacune des colonnes de R), la valeur de la fonction correspondante (voir chap. II, § 2). Lorsque pour deux colonnes (cas) on trouve la même valeur par la fonction, c'est qu'il y a redondance ou incompatibilité pour ce cas. On regarde alors les décisions, ce qui permet de décerner dans laquelle des deux situations l'on se trouve.

d) Détection des cas oubliés.

Ceux-ci peuvent être trouvés aisément en regardant quelles sont les valeurs de la fonction qui ne se sont pas produites durant l'examen des cas redondants et incompatibles. Rappelons que l'ensemble des valeurs prises par la fonction est l'ensemble des entiers compris entre 1 et NT où NT est le nombre total de cas pouvant exister pour cette table.

2.2. PROB 5 - Génération des instructions.

La méthode adoptée ici est la technique de l'organigramme développée au chapitre II (§ 3).

Elle revient en fait à construire l'arbre correspondant à la table (voir figure II-1). Or, le traitement ne pouvant se faire que séquentiellement, on est obligé lorsque l'on descend sur une branche, de mémoriser à chaque sommet une partie de l'information, à savoir une des deux sous-tables construites à partir du test. Comme toutes ces sous-tables ne sont que des sous-matrices de la matrice des conditions R, il suffit pour les mémoriser de connaître deux informations :

1. Un vecteur CONDIT dont la ième composante indique si la ième condition de la table initiable est présente (1) ou non (0) dans la sous-table. Il possède donc MC composantes (MC : nombre de conditions de la table).

2. Un vecteur BASE 1 dont la ième composante indique si la ième règle de la table initiale est présente (1) ou non (0) dans la sous-table.

Etudions la manière de procéder sur un exemple simple :

Règles →	1	2	3	4
C_1	1	2	1	1
C_2	2	0	-2	3
C_3	0	1	1	2

et supposons que les valences des conditions soient $V_1 = 2$, $V_2 = 3$, V_3

a) Initialisation.

On construit les matrices M_i associées à chaque condition C_i. (Il s'agit des matrices Mi définies au chap. II, § 3.3 pour la nouvelle technique du "rule mask".

On obtient $M_1 \equiv \begin{vmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{vmatrix}$ $M_2 \equiv \begin{vmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{vmatrix}$ $M_3 \equiv \begin{vmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{vmatrix}$

au départ, on a aussi CONDIT ≡ (1 1 1)

BASE 1 ≡ (1 1 1 1)

b) Construction de l'arbre.

Définissons M_i (l) comme étant la lème ligne de la matrice M_i et supposons qu'il faille d'abord tester $C_1 = 1$.

Les opérations suivantes sont exécutées :

1. $\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

BASE 1 $M_1(1)$ BASE 2

On regarde alors si BASE 2 possède plus d'une composante à 1. Si tel est le cas, la sous-table correspondant à la branche YES du test $C_1 = 1$ contient plus d'une règle. On met alors à zéro la composante de CONDIT correspondant à la condition testée (1) et on stocke dans un stack le numéro de cette condition, les vecteurs CONDIT et BASE 2.

STACK ⟶ | 1 | 1 0 1 1 | 0 1 1 |

2. Calculer BASE 2 = $\underset{1 \neq 1}{\textcircled{\Sigma}}$ $M_1(1)$ où $\textcircled{\Sigma}$ signifie "somme modul

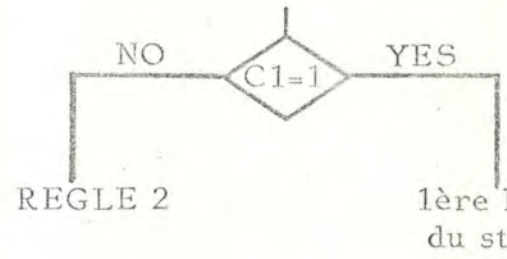
On obtient BASE 2 ≡ (0 1 0 0)

On exécute ensuite $\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$

BASE 1 BASE 2 BASE 1

3. On regarde alors si BASE 1 possède plus d'une composante à 1. Comme ce n'est pas le cas, on regarde si ce vecteur possède une seule composante à 1. Nous sommes dans cette situation, ce qui signifie que la sous-table associée à la branche NO du test $C_1 = 1$ est réduite à une seule règle, celle correspondant à la composante non nulle. L'algorithme du chap. II (§ 3) nous dit alors que l'on est arrivé à l'une des feuilles de l'arbre.

Nous avons donc après l'étape 3 la situation suivante :



Le programme généré associé s'écrit :

```
CLI   C1, X'F1'
BE    PARA 1
B     REGLE 2
```

4. La branche NO du test précédent étant terminée, on transfert dans CONDIT et BASE 1, la première ligne du sommet du stack. On a donc CONDIT ≡ (0 1 1) et BASE 1 ≡ (1 0 1 1). Ces deux informations définissent la sous-table correspondant à la branche YES, c'est-à-dire :

REGLES ⟶	1	3	4
C_2	2	-2	3
C_3	0	1	2

Supposons que le nouveau test doive porter sur $C_2 = 2$.

5. Exécuter $\begin{bmatrix}1\\0\\1\\1\end{bmatrix} \times \begin{bmatrix}1\\1\\0\\0\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$

BASE 1 $M_2(2)$ BASE 2

Comme BASE 2 possède une seule composante à 1, on adjoint à la branche YES du test $C_2 = 2$, la règle correspondant à la composante non nulle, c'est-à-dire règle 1.
Cette branche est donc terminée.

6. Calculer BASE 2 = $\underset{1 \neq 2}{\Sigma}$ $M_2(1)$

BASE 2 ≡ $\begin{bmatrix}0\\1\\1\\0\end{bmatrix} \oplus \begin{bmatrix}0\\1\\1\\1\end{bmatrix} = \begin{bmatrix}0\\1\\1\\1\end{bmatrix}$

$M_2(1)$ $M_2(3)$

On multiplie ensuite BASE 1 et BASE 2

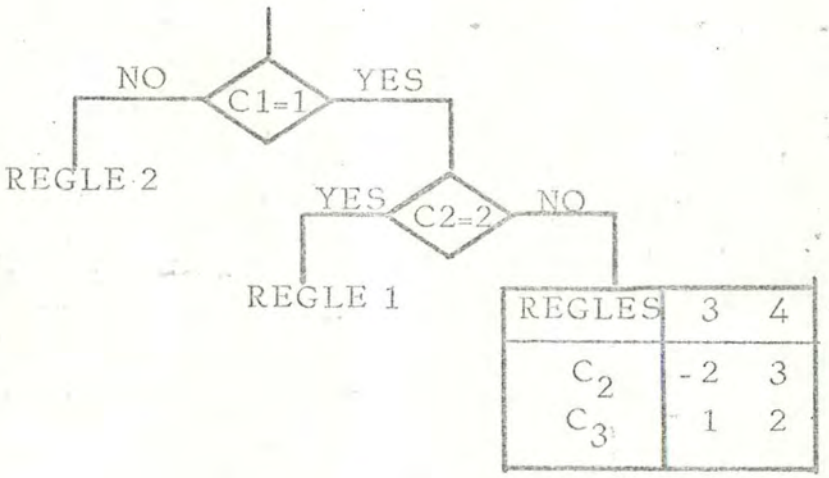
$$\begin{bmatrix}1\\0\\1\\1\end{bmatrix} \times \begin{bmatrix}0\\1\\1\\1\end{bmatrix} = \begin{bmatrix}0\\0\\1\\1\end{bmatrix}$$

BASE 1 BASE 2 BASE 1

Comme BASE 1 possède plus d'une composante à 1, on continue en prenant la sous-table définie par CONDIT et BASE 1, à savoir

REGLES	3	4
C_2	- 2	3
C_3	1	2

L'organigramme actuel est :



REGLES	3	4
C_2	- 2	3
C_3	1	2

Les instructions sont :

```
        CLI     C1, X' F1'
        BE      PARA 1
        B       REGLE 2
PARA 1  EQU     *          ——→ généré au moment où l'on retire
                                une ligne du stack
        CLI     C2, X' F2'
        BE      REGLE 1
        ..................→ branche NO du test C2 = 2
        ..................
```

Supposons que le nouveau test soit $C_2 = 3$.

7. Exécuter $\begin{bmatrix}0\\0\\1\\1\end{bmatrix} \times \begin{bmatrix}0\\1\\1\\1\end{bmatrix} = \begin{bmatrix}0\\0\\1\\1\end{bmatrix}$

BASE 1 $M_2(3)$ BASE 2

Comme BASE 2 possède plus d'une composante différente de zéro, on met la composante de CONDIT correspondant à la condition testée (C_2) à zéro et on stocke dans le stack cette condition et les deux valeurs CONDIT et BASE 2.

STACK → | 2 | 0 0 1 1 | 0 0 1 |

8. Calculer BASE 2 = (2) $M_2(1)$
$1 \neq 3$

On obtient BASE 2 ≡ (1 1 1 0)

On exécute ensuite : $\begin{bmatrix}0\\0\\1\\1\end{bmatrix} \times \begin{bmatrix}1\\1\\1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$

BASE 1 BASE 2 BASE 1

BASE 1 a une seule composante à 1 ; cela implique que la branche correspondant au résultat NO du test est une feuille à laquelle on associe "règle 3".

9. On retire la 1ère ligne du sommet du stack.
On a CONDIT ≡ (0 0 1) BASE 1 ≡ (0 0 1 1)

La sous-table correspondante est

REGLES →	3	4
C_3	1	2

Supposons que le test suivant porte sur $C_3 = 1$.

10. On exécute $\begin{bmatrix}0\\0\\1\\1\end{bmatrix} \times \begin{bmatrix}1\\1\\1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$

BASE 1 $M_3(1)$ BASE 2

BASE 2 ne contient qu'une composante à 1. La branche YES se termine donc par règle 3.

11. Calculer BASE 2 = $\underset{1 \neq 1}{\Sigma}$ $M_3(1)$

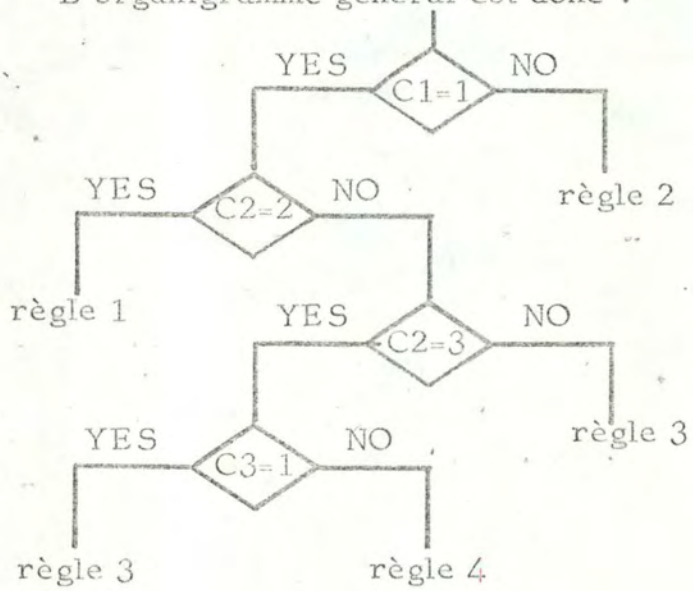
BASE 2 = (1 0 0 1)

On exécute ensuite $\begin{bmatrix}0\\0\\1\\1\end{bmatrix} \times \begin{bmatrix}1\\0\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

BASE 1 BASE 2

Cela signifie que la branche NO correspond à la règle 4.

L'organigramme général est donc :



Les instructions générées sont :

```
        CLI     C1, X'F1'
        BE      PARA 1
        B       REGLE 2
PARA 1  EQU     *
        CLI     C2, X'F2'
        BE      REGLE 1
        CLI     C2, X'F3'
        BE      PARA 2
        B       REGLE 3
PARA 2  EQU     *
        CLI     C3, X'F1'
        BE      REGLE 3
        B       REGLE 4
```

Remarques :

1. Il peut arriver qu'à une étape, les vecteurs BASE 1 ou BASE 2 aient toutes leurs composantes nulles. On arrive alors à une feuille de l'arbre à laquelle on associe la règle ELSE.

2. Le choix de la valeur à tester pour une condition doit se faire en tenant compte non seulement des critères d'optimisation (chap. II, § 3.4), mais aussi de l'état de la sous-table sur laquelle doit porter le test. Ainsi, par exemple, si lors d'une étape ou avait la sous-table

REGLES	3	4
C_2	- 2	2
C_3	1	2

et que le critère d'optimisation adopté pour le choix des valeurs, nous imposait de prendre pour la condition C_2, la valeur 1,
ce choix nous donnerait pour BASE 2 le vecteur (0 0 1 0) et pour BASE 1 le vecteur (0 0 0 1).
Les instructions générées seraient donc :

```
CLI   C2, X'F1'
BE    REGLE 3
B     REGLE 4
```

Ce qui ne correspond pas à la logique de la table puisque, pour $C_2 = 3$, par exemple, on exécuterait la règle 4 au lieu de la règle 3. On a donc pris comme règle supplémentaire pour le choix de la valeur de la condition à tester que celle-ci figure dans la ligne de cette condition que l'on teste, soit sous forme directe, soit sous forme complexe. Pour l'exemple ci-dessus, cela voudrait dire qu'on ne pourrait prendre pour la condition 2 que la valeur 2 pour le test, ce qui ne poserait plus de problème.

§ 3 : Cas d'un problème transposé en équations logiques.

Ce problème comprend 3 parties :

- analyse des équations logiques
- transposition des équations en table
- génération des instructions correspondant à la table

Cette dernière partie est commune aux deux problèmes et a déjà été explicitée au paragraphe précédent.

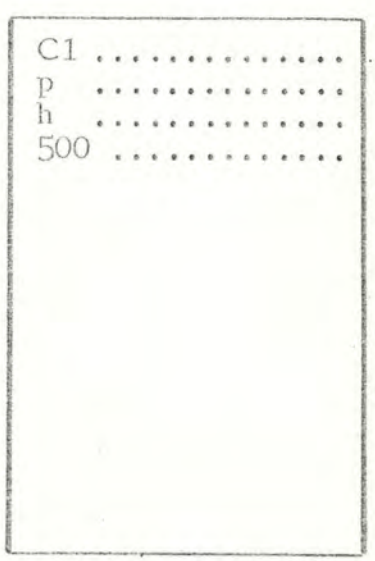
3.1. Analyse des équations logiques.

Comme pour le cas des tables, tout problème, une fois traduit en équations logiques, est ensuite introduit dans la machine à l'aide de cartes perforées dont le dessin se trouve en annexes 3 et 4.

La phase analyse comprend trois modules, qui sont :

a) PROB 1 - Normalisation des équations logiques.

Ce programme transforme les équations logiques de décision et d'incompatibilité sous une forme acceptable par le programme de simplification et de complémentation de fonction. Il transforme chaque variable différente rencontrée en un caractère alphanumérique en commençant par les premières lettres de l'alphabet et supprime l'opérateur multiplication (x) devenu inutile (puisque la longueur des variables devient une longueur implicite égale à 1). Au fur et à mesure de cette transformation, il maintient à jour une table des variables, commune à l'ensemble des équations. Prenons comme exemple l'équation D1 = C1 x p x h + C1 x p x 500 + C1' x h x p. La fonction normalisée est D1 = ABC + ABD + A'CB, tandis que la table des variables contient :



b) PROB 2 - Transformation des équations normalisées.

Ce programme exécute la transformation inverse de PROB 1, au moyen de la table constituée par ce dernier.

c) PROB 3 - Simplification, complémentation et produit de fonctions.

Ce programme permet :

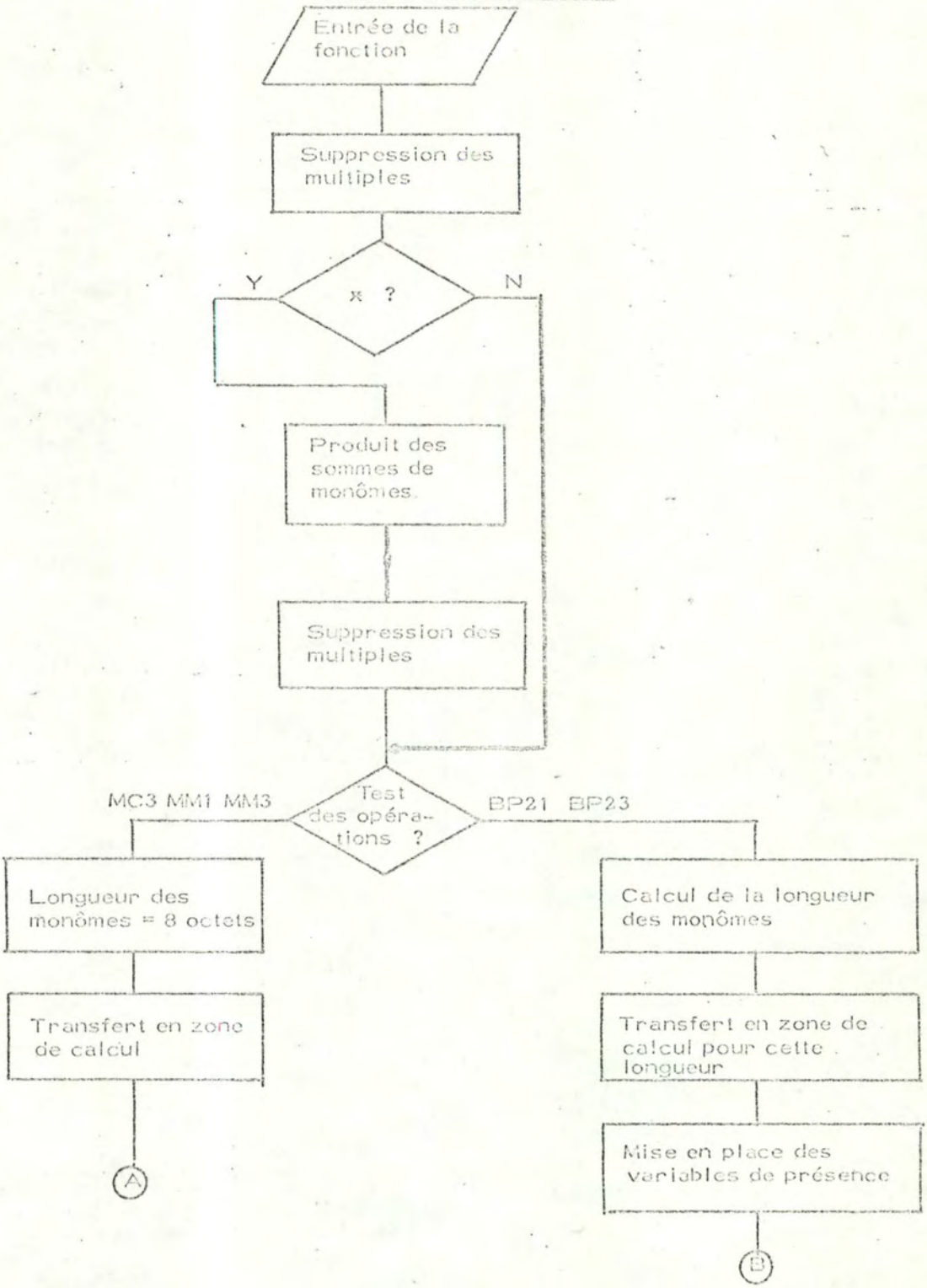
- de multiplier entre elles un certain nombre d'équations logiques ne contenant pas de parenthèses ;
- de complémenter une équation logique ;
- de simplifier cette équation en deux étapes :
  a) recherche des monômes maximaux (base principal complète) ;
  b) recherche des bases principales irréductibles. (voir théorie dans le chapitre III).

Nous avons deux paramètres d'entrée à savoir : l'adresse de l'équation logique et celle d'une zone où sont indiquées les opérations à exécuter :

MC3	complémentation de la fonction
MM1	recherche des monômes maximaux par la méthode linéaire
MM3	recherche des monômes maximaux par la méthode de double dualisation
BP21	recherche des bases principales irréductibles avec utilisation de MM1
BP23	recherche des bases principales irréductibles avec utilisation de MM3

Le paramètre de sortie contient l'adresse de la zone "résultat".
L'organigramme général de PROB 3 se trouve en figure IV-1.

Organigramme général



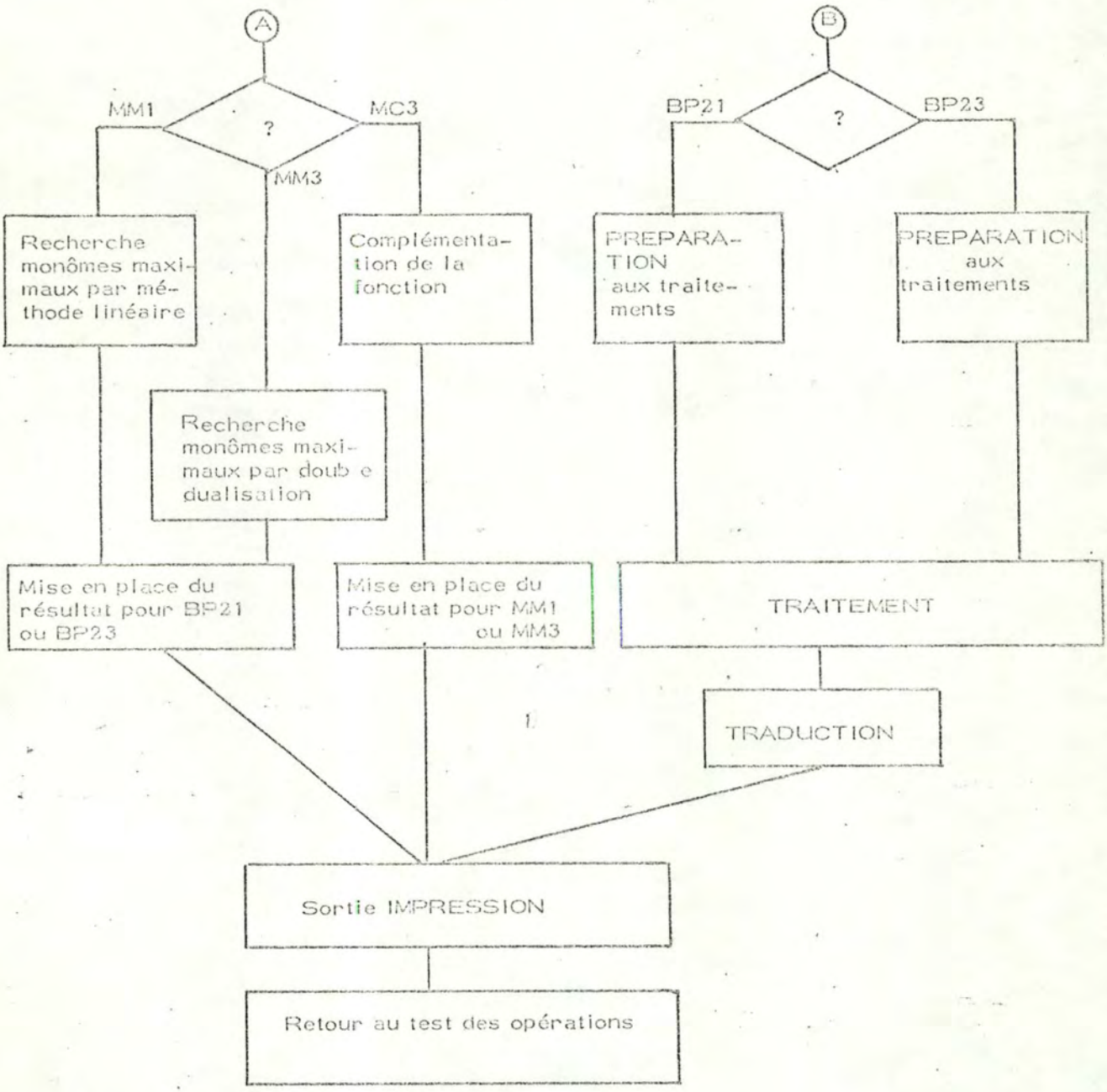

A
MM1
?
MC3
MM3
Recherche monômes maximaux par méthode linéaire
Complémentation de la fonction
Recherche monômes maximaux par doub e dualisation
Mise en place du résultat pour BP21 ou BP23
Mise en place du résultat pour MM1 ou MM3
B
BP21
?
BP23
PREPARATION aux traitements
PREPARATION aux traitements
TRAITEMENT
TRADUCTION
Sortie IMPRESSION
Retour au test des opérations

Fig. IV-1

Exemples :

1. Supposons que nous voulions simplifier la fonction normalisée
   f = AE + ABC + ACE + XY.
   La fin de la fonction est détectée par le 1er blanc rencontré.
   Dans la zone opération, l'on place MM1 BP21. Le résultat obtenu sera
   f = AE + ABC + XY.

2. Si avant de simplifier la fonction, on avait voulu la complémenter, on aurait placé dans la zone "opération" MC3 MM1 BP21.

3. Il est possible également de simplifier et/ou de complémenter un produit de fonctions.
   f = A'D'E + A'BD' + BD'F x A'BE' + AB + CD x AB'C.
   Le programme commencera par exécuter le produit des deux premières sommes de monômes, simplifiera ensuite le résultat pour le multiplier avec la 3ème et éventuelle complémenter le résultat.

Remarques :

1. Les algorithmes MM3 et BP23 ne sont pas utilisés dans le générateur. Cependant, une partie importante de MM3 est reprise dans l'algorithme de complémentation de fonction (MC3), tandis que BP23 est identique à BP21 à la différence près qu'il exploite respectivement les résultats de MM3 et de MM1.

2. MC3 doit toujours se trouver en tête de la zone "opération".

Représentation interne des monômes de la fonction booleenne dans PROB 3.

1. Cas des algorithmes MC3 MM1 MM3.

Chaque monôme est représenté en mémoire par deux mots, soit 64 Bits.
Le 1er mot représente les variables sous forme directe.
Le 2ème mot représente les variables sous forme complémentée.
Chaque bit des mots correspond à une variable.
Le 1er bit de gauche correspond à la variable A.
Le dernier bit de droite correspond à la variable 7.
1 signifie que la variable est présente et 0 qu'elle est absente.

Exemple : AE'F76'

1er mot	1 0 0 0 0 1 0 ..... 0 0 1
2ème mot	0 0 0 0 1 0 0 ..... 0 1 0

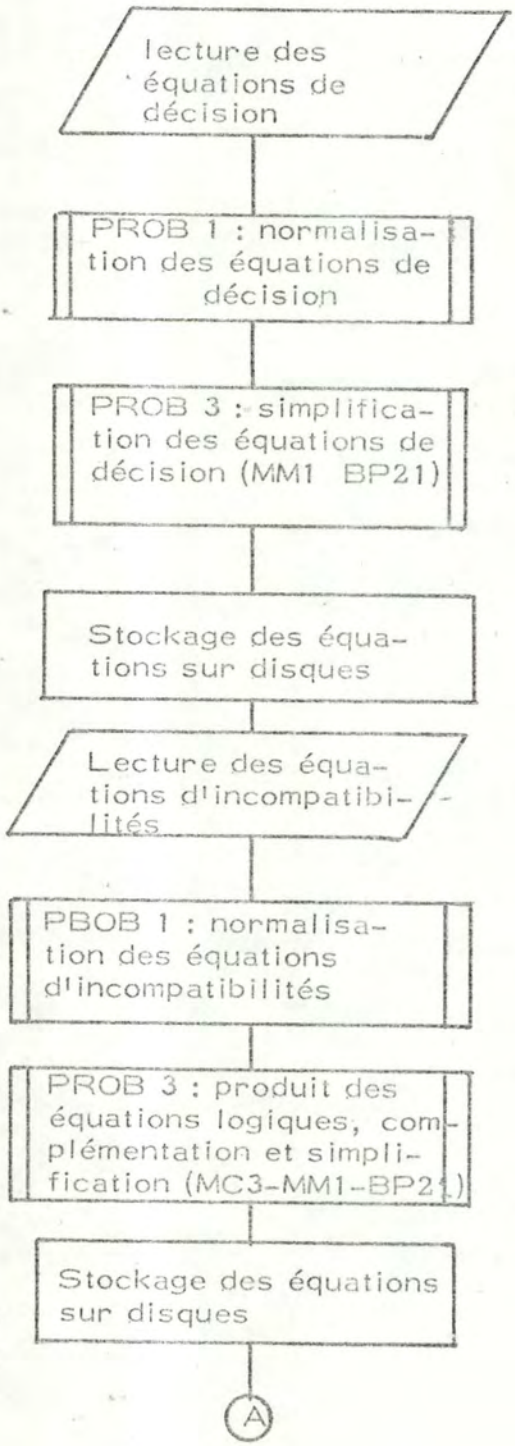
Les 2 mots sont placés consécutivement en mémoire. Chaque monôme occu
donc 8 octets.

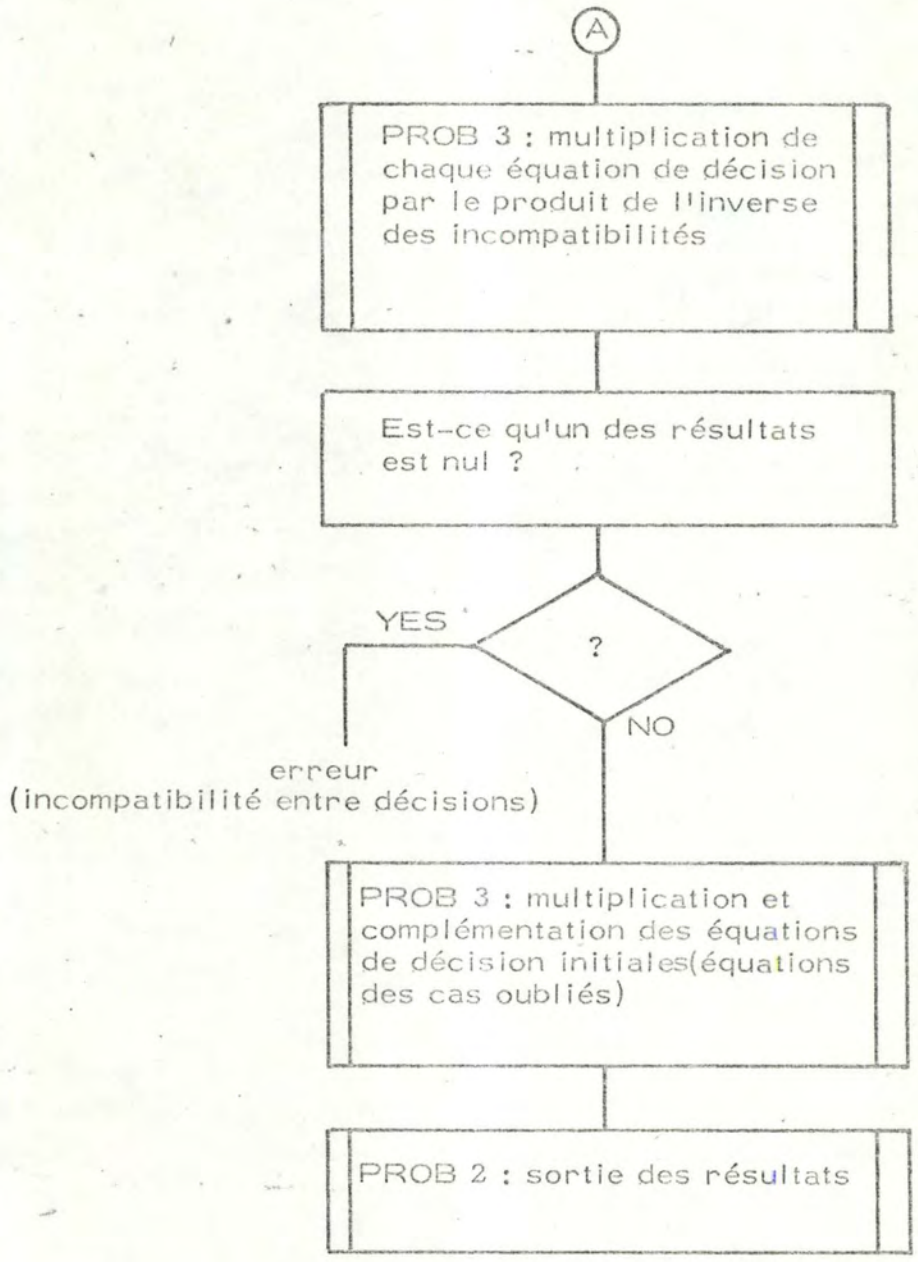
2. Cas des algorithmes BP21 et BP23.

Pour ces 2 algorithmes, il faut ajouter des variables de présence.
Ce problème est résolu en plaçant à la suite de la représentation du monôme du type précédent autant de mots supplémentaires que cela est nécessaire.
Ainsi, si l'on a moins de 32 monômes, un troisième mot suffit.
Pour le 1er monôme, on met à 1 le premier bit de gauche du 3ème mot et les autres à zéro.
Pour le nème monôme, on met à 1 le nème bit de gauche du 3ème mot et les autres à zéro.
Dans le cas où il y a plus de 32 monômes et moins de 64, on ajoute un 4ème mot et ainsi de suite.
Ainsi, pour le 34ème monôme, c'est le bit du 4ème mot qui est à 1 et le restant à zéro.
La longueur de chaque monôme en mémoire dépend donc du nombre de monômes obtenus lors d'un des algorithmes MM1 ou MM3.

d) Description de la phase d'analyse des équations logiques.

L'organigramme ci-dessous explique comment est réalisée la liaison entre les trois programmes précédents, pour permettre la suppression des redondances, la détection des cas incompatibles et des cas oubliés.




A
PROB 3 : multiplication de chaque équation de décision par le produit de l'inverse des incompatibilités
Est-ce qu'un des résultats est nul ?
YES
?
NO
erreur
(incompatibilité entre décisions)
PROB 3 : multiplication et complémentation des équations de décision initiales(équations des cas oubliés)
PROB 2 : sortie des résultats

3.2. PROB 4 - Transposition des équations logiques en table.

Ce programme construit une table de décision à partir des équations de décision multipliées par l'inverse des équations d'incompatibilités. Pour exécuter cette opération, il tient compte des informations se trouvant dans une table et qui représentent les incompatibilités entre les décision (voir chap. 1. § 3.3.).
Cette table est entrée par cartes perforées (voir annexe 5 et 6).

3.3. EXEMPLES

A. ANALYSE DES TABLES DE DECISION

Honeywell Bull

DENOMINATION DE L'IMPRIME : ANALYSE D'UNE TABLE DE DECISION et GENERATION DES INSTRUCTIONS

APPLICATION

REDACTEUR | CLICHE | PROGRAMME / UT N° | BANDE PILOTE N° | DATE DE CREATION | PAGE

VOICI LA TABLE DE DECISION INITIALE

```
                1  2  3  4   ( numero des règles)
      NC
C  1   3   1  2  1  1
   2   2   0 -1  1  1   } conditions
   3   4   1  4 -2  1
   4   4   1  1  1  1
D  1       1  2  1  0   } décisions
   2       2  2  2  0
```

nombre de possibilités par ligne "condition"

numero de ligne.

F1 SIGNIFIE-CAS INCOMPATIBLES CAD CONDITIONS IDENTIQUES ET DECISIONS DIFFERENTES

F2 SIGNIFIE-CAS REDONDANTS CAD CONDITIONS ET DECISIONS IDENTIQUES

F2 SE PRODUIT POUR LES COLONNES 1 ET 3 POUR LE CAS: 1 1 1 1 ——→ représente le cas commun aux 2 3 et 4

IL N Y A PAS DE CAS REDONDANTS ET INCOMPATIBLES ——→ signifie que la table est correcte.

CAS OUBLIES

```
2  1  1  1
3  1  1  1
2  2  1  1
3  2  1  1
1  1  2  1
:  :  :  :
1  :  :  :
```

——→ cas non envisagé dans l'ensemble des règles de la table initiale.

IL N Y A PAS DE CAS OUBLIES

GENERATION DES INSTRUCTIONS EN ASSEMBLER CORRESPONDANT A LA TABLE DE DECISION

```
        CLI   C 2,X'F 3'
        BE    PARA 1
        BC    15,REGLE 4
PARA 1  EQU   *
        :
        :
        :
```

——→ ensemble des instructions correspondant à la logique de la table.

GENERATION DES INSTRUCTIONS NON EXECUTABLES CAR LA TABLE CONTIENT DES INCOH. OU DES REDONDANCES

EXEMPLE 1

VOICI LA TABLE DE DECISION INITIALE

```
          1  2  3  4
     NC
C  1    2  1  2  1  2
   2    2  1  1  2  2

D  1       1  0  1  0
   2       1  1  0  0
```

F1 SIGNIFIE-CAS INCOMPATIBLES CAD CONDITIONS IDENTIQUES ET DECISIONS DIFFERENTES

F2 SIGNIFIE -CAS REDONDANTS CAD CONDITIONS ET DECISIONS IDENTIQUES

IL N Y A PAS DE CAS REDONDANTS ET INCOMPATIBLES

CAS OUBLIES

IL N Y A PAS DE CAS OUBLIES

GENERATION DES INSTRUCTIONS EN ASSEMBLER CORRESPONDANT A LA TABLE DE DECISION

```
          CLI   C 1,X'F 1'
          BE    PARA 1
          CLI   C 1,X'F 2'
          BE    PARA 2
          BC    15,ELSE
PARA 1    EQU   *
          CLI   C 2,X'F 1'
          BE    REGLE 1
          BC    15,REGLE 3
PARA 2    EQU   *
          CLI   C 2,X'F 1'
          BE    REGLE 2
          BC    15,REGLE 4
REGLE 1   EQU   *
          BAL   1,D11
          BAL   1,D21
          BC    15,FIN
REGLE 2   EQU   *
          BAL   1,D21
          BC    15,FIN
REGLE 3   EQU   *
          BAL   1,D11
          BC    15,FIN
REGLE 4   EQU   *
          BC    15,FIN
ELSE      BAL   1,ERREUR
FIN       EOJ
```

EXEMPLE 2

VOICI LA TABLE DE DECISION INITIALE

		NC	1	2	3	4	5
C	1	2	2	0	0	1	0
	2	2	2	2	1	2	1
	3	2	1	2	1	1	2
D	1		1	0	0	0	0
	2		0	0	0	1	0
	3		0	1	1	0	0
	4		0	0	0	0	1

F1 SIGNIFIE-CAS INCOMPATIBLES CAD CONDITIONS IDENTIQUES ET DECISIONS DIFFERENTES

F2 SIGNIFIE -CAS REDONDANTS CAD CONDITIONS ET DECISIONS IDENTIQUES

IL N Y A PAS DE CAS REDONDANTS ET INCOMPATIBLES

CAS OUBLIES

IL N Y A PAS DE CAS OUBLIES

GENERATION DES INSTRUCTIONS EN ASSEMBLER CORRESPONDANT A LA TABLE DE DECISION

```
          CLI   C 2,X'F 2'
          BE    PARA 1
          CLI   C 2,X'F 1'
          BE    PARA 2
          BC    15,ELSE
PARA 1    EQU   *
          CLI   C 3,X'F 1'
          BE    PARA 3
          BC    15,REGLE 2
PARA 2    EQU   *
          CLI   C 3,X'F 1'
          BE    REGLE 3
          BC    15,REGLE 5
PARA 3    EQU   *
          CLI   C 1,X'F 1'
          BE    REGLE 4
          BC    15,REGLE 1
REGLE 1   EQU   *
          BAL   1,D11
          BC    15,FIN
REGLE 2   EQU   *
          BAL   1,D31
          BC    15,FIN
REGLE 3   EQU   *
```

```
          BAL    1,D31
          BC     15,FIN
REGLE 4   EQU    *
          BAL    1,D21
          BC     15,FIN
REGLE 5   EQU    *
          BAL    1,D41
          BC     15,FIN
ELSE      BAL    1,ERREUR
FIN       EOJ
```

EXEMPLE 3

VOICI LA TABLE DE DECISION INITIALE

```
              1  2  3  4
        NC

C  1     3    1  2  1  0
   2     2    0 -1  1  0
   3     4    1  4 -2  0
   4     4    1  1  1  0

D  1          1  2  1  0
   2          2  2  2  0
```

F1 SIGNIFIE-CAS INCOMPATIBLES CAD CONDITIONS IDENTIQUES ET DECISIONS DIFFERENTE

F2 SIGNIFIE -CAS REDONDANTS CAD CONDITIONS ET DECISIONS IDENTIQUES

```
F2 SE PRODUIT POUR LES COLONNES  1 AND  3 POUR LECAS:   1  1  1  1
F1 SE PRODUIT POUR LES COLONNES  1 AND  4 POUR LECAS:   1  1  1  1
F1 SE PRODUIT POUR LES COLONNES  3 AND  4 POUR LECAS:   1  1  3  1
F1 SE PRODUIT POUR LES COLONNES  3 AND  4 POUR LECAS:   1  1  4  1
F1 SE PRODUIT POUR LES COLONNES  1 AND  4 POUR LECAS:   1  2  1  1
F1 SE PRODUIT POUR LES COLONNES  2 AND  4 POUR LECAS:   2  2  4  1
```

CAS OUBLIES

IL N Y A PAS DE CAS OUBLIES

GENERATION DES INSTRUCTIONS NON EXECUTEES CAR LA TABLE CONTIENT DES INCOM. OU I

**FORTRAN ** STOP

EXEMPLE 4

VOICI LA TABLE DE DECISION INITIALE

			1	2	3	4	5	6	7	8	9	10	11	12	13	14
		NC														
C	1	3	0	0	0	0	1	1	1	1	1	-1	2	2	3	3
	2	5	1	1	2	2	3	3	3	5	5	5	4	4	4	4
	3	4	-4	4	-4	4	2	3	4	-4	4	1	-4	4	-4	4
			123													
D	1		1	2	3	3	3	3	2	3	2	2	2	2	2	3
	2		1	1	1	2	2	0	2	2	2	1	1	1	1	1
	3		1	2	2	2	3	3	2	3	2	2	3	1	2	1

F1 SIGNIFIE-CAS INCOMPATIBLES CAD CONDITIONS IDENTIQUES ET DECISIONS DIFFERENT

F2 SIGNIFIE -CAS REDONDANTS CAD CONDITIONS ET DECISIONS IDENTIQUES

IL N Y A PAS DE CAS REDONDANTS ET INCOMPATIBLES

CAS OUBLIES

1	3	1
2	3	1
3	3	1
1	4	1
2	3	2
3	3	2
1	4	2
2	5	2
3	5	2
2	3	3
3	3	3
1	4	3
2	5	3
3	5	3
2	3	4
3	3	4
1	4	4
2	5	4
3	5	4

GENERATION DES INSTRUCTIONS EN ASSEMBLER CORRESPONDANT A LA TABLE DE DECISION

cond 2 = 4

```
         CLI   C 2,X'F 4'
         BE    PARA 1
         CLI   C 2,X'F 3'
         BE    PARA 2
         CLI   C 2,X'F 5'
         BE    PARA 3
         CLI   C 2,X'F 1'
         BE    PARA 4
         CLI   C 2,X'F 2'
         BE    PARA 5
         BC    15,ELSE
PARA 1   EQU   *
         CLI   C 1,X'F 2'
         BE    PARA 6
         CLI   C 1,X'F 3'
         BE    PARA 7
         BC    15,ELSE
PARA 2   EQU   *
         CLI   C 1,X'F 1'
         BE    PARA 8
         BC    15,ELSE
PARA 3   EQU   *
         CLI   C 1,X'F 1'
         BE    PARA 9
         BC    15,REGLE10
PARA 4   EQU   *
         CLI   C 3,X'F 4'
         BE    REGLE 2
         BC    15,REGLE 1
PARA 5   EQU   *
         CLI   C 3,X'F 4'
         BE    REGLE 4
         BC    15,REGLE 3
PARA 6   EQU   *
         CLI   C 3,X'F 4'
         BE    REGLE12
         BC    15,REGLE11
PARA 7   EQU   *
         CLI   C 3,X'F 4'
         BE    REGLE14
         BC    15,REGLE13
PARA 8   EQU   *
         CLI   C 3,X'F 2'
         BE    REGLE 5
         CLI   C 3,X'F 3'
         BE    REGLE 6
         BC    15,REGLE 7
PARA 9   EQU   *
         CLI   C 3,X'F 4'
         BE    REGLE 9
         BC    15,REGLE 8
REGLE 1  EQU   *
         BAL   1,D11
         BAL   1,D21
         BAL   1,D31
         BC    15,FIN
REGLE 2  EQU   *
         BAL   1,D12
         BAL   1,D21
         BAL   1,D32
         BC    15,FIN
```

```
REGLE 3   EQU   *
          BAL   1,D13
          BAL   1,D21
          BAL   1,D32
          BC    15,FIN
REGLE 4   EQU   *
          BAL   1,D13
          BAL   1,D22
          BAL   1,D32
          BC    15,FIN
REGLE 5   EQU   *
          BAL   1,D13
          BAL   1,D22
          BAL   1,D33
          BC    15,FIN
REGLE 6   EQU   *
          BAL   1,D13
          BAL   1,D33
          BC    15,FIN
REGLE 7   EQU   *
          BAL   1,D12
          BAL   1,D22
          BAL   1,D32
          BC    15,FIN
REGLE 8   EQU   *
          BAL   1,D13
          BAL   1,D22
          BAL   1,D33
          BC    15,FIN
REGLE 9   EQU   *
          BAL   1,D12
          BAL   1,D22
          BAL   1,D32
          BC    15,FIN
REGLE10   EQU   *
          BAL   1,D12
          BAL   1,D21
          BAL   1,D32
          BC    15,FIN
REGLE11   EQU   *
          BAL   1,D12
          BAL   1,D21
          BAL   1,D33
          BC    15,FIN
REGLE12   EQU   *
          BAL   1,D12
          BAL   1,D21
          BAL   1,D31
          BC    15,FIN
REGLE13   EQU   *
          BAL   1,D12
          BAL   1,D21
          BAL   1,D32
          BC    15,FIN
REGLE14   EQU   *
          BAL   1,D13
          BAL   1,D21
          BAL   1,D31
          BC    15,FIN
ELSE      BAL   1,ERREUR
FIN       EOJ
```

EXEMPLE 5

VOICI LA TABLE DE DECISION INITIALE

		NC	1	2	3	4	5	6	7	8	9	10	11	12	13	14
C	1	3	0	0	0	0	1	1	1	1	1	-1	2	2	3	3
	2	5	1	1	2	2	3	3	3	-5	5	5	-4	4	4	4
	3	4	-4	4	-4	4	2	3	4	-4	4	1	-4	4	-4	4
D	1		1	2	3	3	3	3	2	3	2	2	2	2	2	3
	2		1	1	1	2	2	0	2	2	2	1	1	1	1	1
	3		1	2	2	2	3	3	2	3	2	2	3	1	2	1

F1 SIGNIFIE-CAS INCOMPATIBLES CAD CONDITIONS IDENTIQUES ET DECISIONS DIFFERENTE

F2 SIGNIFIE -CAS REDONDANTS CAD CONDITIONS ET DECISIONS IDENTIQUES

F1 SE PRODUIT POUR LES COLONNES 1 AND 8 POUR LECAS: 1 1 1
F1 SE PRODUIT POUR LES COLONNES 1 AND 8 POUR LECAS: 1 1 2
F1 SE PRODUIT POUR LES COLONNES 1 AND 8 POUR LECAS: 1 1 3
F1 SE PRODUIT POUR LES COLONNES 3 AND 8 POUR LECAS: 1 2 1
F1 SE PRODUIT POUR LES COLONNES 3 AND 8 POUR LECAS: 1 2 2
F1 SE PRODUIT POUR LES COLONNES 3 AND 8 POUR LECAS: 1 2 3
F2 SE PRODUIT POUR LES COLONNES 5 AND 8 POUR LECAS: 1 3 2
F1 SE PRODUIT POUR LES COLONNES 6 AND 8 POUR LECAS: 1 3 3
F1 SE PRODUIT POUR LES COLONNES 1 AND 11 POUR LECAS: 2 1 1
F1 SE PRODUIT POUR LES COLONNES 1 AND 11 POUR LECAS: 2 1 2
F1 SE PRODUIT POUR LES COLONNES 1 AND 11 POUR LECAS: 2 1 3
F1 SE PRODUIT POUR LES COLONNES 3 AND 11 POUR LECAS: 2 2 1
F1 SE PRODUIT POUR LES COLONNES 3 AND 11 POUR LECAS: 2 2 2
F1 SE PRODUIT POUR LES COLONNES 3 AND 11 POUR LECAS: 2 2 3
F1 SE PRODUIT POUR LES COLONNES 10 AND 11 POUR LECAS: 2 5 1

CAS OUBLIES

3 3 1
2 4 1
1 5 1
3 3 2
2 4 2
1 5 2
3 5 2
3 3 3
2 4 3
1 5 3
3 5 3
2 3 4
3 3 4
1 4 4
2 5 4
3 5 4

GENERATION DES INSTRUCTIONS NON EXECUTEES CAR LA TABLE CONTIENT DES INCOM. OU D

EXEMPLE 6

VOICI LA TABLE DE DECISION INITIALE

		NC	1	2	3	4	5	6	7	8	9	10
C	1	3	0	0	0	0	1	1	1	1	1	-1
	2	5	1	1	2	2	3	3	3	5	5	5
	3	4	-4	4	-4	4	2	3	4	-4	4	1
D	1		1	2	3	3	3	3	2	3	2	2
	2		1	1	1	2	2	0	2	2	2	1
	3		1	2	2	2	3	3	2	3	2	2

F1 SIGNIFIE-CAS INCOMPATIBLES CAD CONDITIONS IDENTIQUES ET DECISIONS DIFFEREN

F2 SIGNIFIE -CAS REDONDANTS CAD CONDITIONS ET DECISIONS IDENTIQUES

IL N Y A PAS DE CAS REDONDANTS ET INCOMPATIBLES

CAS OUBLIES

1	3	1
2	3	1
3	3	1
1	4	1
2	4	1
3	4	1
2	3	2
3	3	2
1	4	2
2	4	2
3	4	2
2	5	2
3	5	2
2	3	3
3	3	3
1	4	3
2	4	3
3	4	3
2	5	3
3	5	3
2	3	4
3	3	4
1	4	4
2	4	4
3	4	4
2	5	4
3	5	4

GENERATION DES INSTRUCTIONS EN ASSEMBLER CORRESPONDANT A LA TABLE DE DECISIO

```
          CLI   C 2,X'F 3'
          BE    PARA 1
          CLI   C 2,X'F 5'
          BE    PARA 2
          CLI   C 2,X'F 1'
          BE    PARA 3
          CLI   C 2,X'F 2'
          BE    PARA 4
          BC    15,ELSE
PARA 1    EQU   *
          CLI   C 1,X'F 1'
          BE    PARA 5
          BC    15,ELSE
PARA 2    EQU   *
          CLI   C 1,X'F 1'
          BE    PARA 6
          BC    15,REGLE10
PARA 3    EQU   *
          CLI   C 3,X'F 4'
          BE    REGLE 2
          BC    15,REGLE 1
PARA 4    EQU   *
          CLI   C 3,X'F 4'
          BE    REGLE 4
          BC    15,REGLE 3
PARA 5    EQU   *
          CLI   C 3,X'F 2'
          BE    REGLE 5
          CLI   C 3,X'F 3'
          BE    REGLE 6
          BC    15,REGLE 7
PARA 6    EQU   *
          CLI   C 3,X'F 4'
          BE    REGLE 9
          BC    15,REGLE 8
REGLE 1   EQU   *
          BAL   1,D11
          BAL   1,D21
          BAL   1,D31
          BC    15,FIN
REGLE 2   EQU   *
          BAL   1,D12
          BAL   1,D21
          BAL   1,D32
          BC    15,FIN
REGLE 3   EQU   *
          BAL   1,D13
          BAL   1,D21
          BAL   1,D32
          BC    15,FIN
REGLE 4   EQU   *
          BAL   1,D13
          BAL   1,D22
          BAL   1,D32
          BC    15,FIN
REGLE 5   EQU   *
          BAL   1,D13
          BAL   1,D22
          BAL   1,D33
          BC    15,FIN
REGLE 6   EQU   *
```

```
                BAL     1,D13
                BAL     1,D33
                BC      15,FIN
REGLE 7         EQU     *
                BAL     1,D12
                BAL     1,D22
                BAL     1,D32
                BC      15,FIN
REGLE 8         EQU     *
                BAL     1,D13
                BAL     1,D22
                BAL     1,D33
                BC      15,FIN
REGLE 9         EQU     *
                BAL     1,D12
                BAL     1,D22
                BAL     1,D32
                BC      15,FIN
REGLE10         EQU     *
                BAL     1,D12
                BAL     1,D21
                BAL     1,D32
                BC      15,FIN
ELSE            BAL     1,ERREUR
FIN             EOJ
```

EXEMPLE 7

VOICI LA TABLE DE DECISION INITIALE

		NC	1	2	3	4	5	6	7	8	9
C	1	3	0	0	0	0	1	1	1	1	1
	2	5	1	1	2	2	3	3	3	5	4
	3	4	-4	4	-4	4	2	3	4	-4	4
D	1		1	2	2	3	2	3	3	2	3
	2		1	1	1	2	2	2	1	2	2
	3		1	2	3	3	3	3	3	2	3

F1 SIGNIFIE-CAS INCOMPATIBLES CAD CONDITIONS IDENTIQUES ET DECISIONS DIFFERENT

F2 SIGNIFIE -CAS REDONDANTS CAD CONDITIONS ET DECISIONS IDENTIQUES

IL N Y A PAS DE CAS REDONDANTS ET INCOMPATIBLES

CAS OUBLIES

1	3	1
2	3	1
3	3	1
1	4	1
2	4	1
3	4	1
2	5	1
3	5	1
2	3	2
3	3	2
1	4	2
2	4	2
3	4	2
2	5	2
3	5	2
2	3	3
3	3	3
1	4	3
2	4	3
3	4	3
2	5	3
3	5	3
2	3	4
3	3	4
2	4	4
3	4	4
1	5	4
2	5	4
3	5	4

GENERATION DES INSTRUCTIONS EN ASSEMBLER CORRESPONDANT A LA TABLE DE DECISIO

```
         CLI   C 2,X'F 3'
         BE    PARA 1
         CLI   C 2,X'F 1'
         BE    PARA 2
         CLI   C 2,X'F 2'
         BE    PARA 3
         CLI   C 1,X'F 1'
         BE    PARA 4
         BC    15,ELSE
PARA 1   EQU   *
         CLI   C 1,X'F 1'
         BE    PARA 5
         BC    15,ELSE
PARA 2   EQU   *
         CLI   C 3,X'F 4'
         BE    REGLE 2
         BC    15,REGLE 1
PARA 3   EQU   *
         CLI   C 3,X'F 4'
         BE    REGLE 4
         BC    15,REGLE 3
PARA 4   EQU   *
         CLI   C 2,X'F 4'
         BE    REGLE 9
         BC    15,REGLE 8
PARA 5   EQU   *
         CLI   C 3,X'F 2'
         BE    REGLE 5
         CLI   C 3,X'F 3'
         BE    REGLE 6
         BC    15,REGLE 7
REGLE 1  EQU   *
         BAL   1,D11
         BAL   1,D21
         BAL   1,D31
         BC    15,FIN
REGLE 2  EQU   *
         BAL   1,D12
         BAL   1,D21
         BAL   1,D32
         BC    15,FIN
REGLE 3  EQU   *
         BAL   1,D12
         BAL   1,D21
         BAL   1,D33
         BC    15,FIN
REGLE 4  EQU   *
         BAL   1,D13
         BAL   1,D22
         BAL   1,D33
         BC    15,FIN
REGLE 5  EQU   *
         BAL   1,D12
         BAL   1,D22
         BAL   1,D33
         BC    15,FIN
REGLE 6  EQU   *
         BAL   1,D13
         BAL   1,D22
         BAL   1,D33
         BC    15,FIN
```

```
REGLE 7   EQU    *
          BAL    1,D13
          BAL    1,D21
          BAL    1,D33
          BC     15,FIN
REGLE 8   EQU    *
          BAL    1,D12
          BAL    1,D22
          BAL    1,D32
          BC     15,FIN
REGLE 9   EQU    *
          BAL    1,D13
          BAL    1,D22
          BAL    1,D33
          BC     15,FIN
ELSE      BAL    1,ERREUR
FIN       EOJ

**FORTRAN ** STOP
```

EXEMPLE 8

VOICI LA TABLE DE DECISION INITIALE

	NC	1	2	3	4	5	6	7	8	9
C	1	3	0	0	0	0	1	1	1	1
	2	5	1	1	2	2	3	3	5	-4
	3	4	-4	4	-4	4	2	3	4	-4	4
D	1		1	2	2	3	2	3	3	2	3
	2		1	1	1	2	2	2	1	2	2
	3		1	2	3	3	3	3	3	2	3

F1 SIGNIFIE-CAS INCOMPATIBLES CAD CONDITIONS IDENTIQUES ET DECISIONS DIFFERENTES

F2 SIGNIFIE -CAS REDONDANTS CAD CONDITIONS ET DECISIONS IDENTIQUES

F1 SE PRODUIT POUR LES COLONNES 2 AND 9 POUR LECAS: 1 1 4
F2 SE PRODUIT POUR LES COLONNES 4 AND 9 POUR LECAS: 1 2 4
F1 SE PRODUIT POUR LES COLONNES 7 AND 9 POUR LECAS: 1 3 4

CAS OUBLIES

1 3 1
2 3 1
3 3 1
1 4 1
2 4 1
3 4 1
2 5 1
3 5 1
2 3 2
3 3 2
1 4 2
2 4 2
3 4 2
2 5 2
3 5 2
2 3 3
3 3 3
1 4 3
2 4 3
3 4 3
2 5 3
3 5 3
2 3 4
3 3 4
1 4 4
2 4 4
3 4 4
2 5 4
3 5 4

GENERATION DES INSTRUCTIONS NON EXECUTEES CAR LA TABLE CONTIENT DES INCOM. OU DES

B. ANALYSE DES EQUATIONS LOGIQUES

Honeywell Bull

DENOMINATION DE L'IMPRIME : Analyse à partir des équations logiques

APPLICATION

REDACTEUR	CLICHE	PROGRAMME / UT N°	BANDE PILOTE N°	DATE DE CREATION	PAGE

VOICI LE SYSTEME D EQUATIONS LOGIQUES

DO1 =
C1*P*H*S0+H*P) 1re équation logique de décision.

LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
ABCD+) 1re équation logique normalisée (sortie de PROB1)
BC+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
BC+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
BC+
DO1 =) 1re équation logique de décision normalisée et simplifiée (sortie de PROB3)
BC

DO2 =) 2nd équation logique de décision
. . . . . .
. . . . . .

VOICI LE SYSTEME D EQUATIONS DONNANT LES INCOMPATI.
IO1 =) 1re équation logique d'incompatibilité.
S0'*H0

LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
D'E) 1re équation logique d'incompatibilité normalisée (sortie de PROB1)
COMPLEMENTATION DE LA FONCTION
E'+) équation logique représentant l'inverse de la 1re équation logique d'incompatibilité normalisée.
D+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
. . . . . .
. . . . . .

IO1 =) inverse de la 1re équation logique d'incompatibilité normalisée.
E'+D simplifié
IO2 =) 2nd équation logique d'incompatibilité.
. . . . . .
. . . . . .

PRODUIT DE L INVERSE DES INCOMPATIBILITÉS
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
E'+) inverse simplifié de la 1re équation logique d'incompatibilité
D+
XXXXXXXXXXXXXXXXX
. . . .) inverse simplifié de la 2nd équation logique d'incompatibilité.
. . . . .
RESULTAT DU PRODUIT DE SOMMES DE NOMBRES
E'+) produit de toutes les équations d'incompatibilités complémentées.
D+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
. . . . . .
. . . . . .

Honeywell Bull

DENOMINATION DE L'IMPRIME				APPLICATION
Analyse à partir des équations logiques				
RÉDACTEUR	CLICHE	PROGRAMME UT N°	BANDE PILOTE N	DATE DE CREATION	PAGE

EQUATIONS DE DECISION MULTIPLIEES PAR LE PRODUIT DE L INVERSE DES INCOMPATI.
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
E'+)
D+) ——> produit simplifié de toutes les équations d'incompatibilité complémentées.
XXXXXXXXXXXXXXXXX
BC+) ——> 1e équation logique de décision simplifiée.
XXXXXXXXXXXXXXXXX
RESULTAT DU PRODUIT DE SOMMES DE MONOMES ou LE PRODUIT DE SOMMES DE MONOMES EST NUL
. . . . . .
. . . . . .
. . . . . .
DEC =
BE'C+BDC) ——> 1e équation logique de décision dans laquelle on a retiré les cas incompatibles.

LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXX
. . . . . .
. . . . . .

EQUATIONS DES CAS OUBLIÉS
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXX
BC+
AB+) ——> somme de toutes les équations logiques de décision minimisées et simplifiées.
. . . .
. . . .
. . .
COMPLEMENTATION DE LA FONCTION
. . . . . .
. . . . . .
BASE PRINCIPALE EXPLOITE METHODE LIBRAIRE
. . . . . .
. . . . . .
CO =
A'D'B'A'DE) ——> equation logique représentant les cas oubliés.

GENERATION DES INSTRUCTIONS NON DISCUTEES CAR INCOMPATIBILITÉ DE CERTAINES EQUATIONS ——> message imprimé logique
résultat du produit d'une équation de décision par le produit de l'inverse des compatibilités est

DUREE = 31,96 SECONDES ——> temps d'exécution.

Exemple 1. - Calcul de réduction dans la facturation.

L'exemple est celui défini au § 1.2 de l'introduction.

3 cas sont examinés :

A. Analyse des équations de décision uniquement.
Il s'agit de :

D01 = C1 * P + C1 * H + C1 * 50 + C1' * 60 * P

D02 = C1 * P * H + C1 * P * 50 + C1 * H * P +

C1 * H * 50 + C1 * 50 * P + C1 * 50 * H

D03 = C1 * P * H * 50

D04 = C1' * 50' * 60 + C1' * 50 * 60'

B. Introduction de deux équations d'incompatibilité :

I01 = 50' * 60

I02 = C1 * P * H * 50

⟹ erreur : la décision D03 est nulle lorsqu'on lui supprime les cas incompatibles.

C. Suppression de l'équation d'incompatibilité I02.

La table donnant la correspondance entre les variables d'entrée et les variables normalisées et la suivante :

Variables d'entrée	Variables normaliées
C1	A
P	B
H	C
50	D
60	E

EXEMPLE 1 A

```
VOICI LE SYSTEME D EQUATIONS LOGIQUES
D01 =
C1*P+C1*H+C1*50+C1'*60*P

LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
AB+
AC+
AD+
A'BE+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
AB+
AC+
AD+
BE+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
AB+
AC+
AD+
BE+
D01 =
AB+AC+AD+BE



D02 =
C1*P*H+C1*P*50+C1*H*P+C1*H*50+C1*50*P+C1*50*H

LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
ABC+
ABD+
ABC+
ACD+
ABD+
ACD+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
ACD+
ABD+
ABC+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
ACD+
ABD+
ABC+
D02 =
ACD+ABD+ABC
```

```
D03 =
C1*P*H*50


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
ABCD+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
ABCD+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
ABCD+
D03 =
ABCD



D04 =
C1'*50'*60+C1'*50'*60'


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
A'D'E+
A'D'E'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'D'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'D'+
D04 =
A'D'
```

```
EQUATIONS DES CAS OUBLIES
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
AB+
AC+
AD+
BE+
ACD+
ABD+
ABC+
ABCD+
A'D'+
COMPLEMENTATION DE LA FONCTION
A'B'D+
A'DE'+
AB'C'D'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'B'D+
A'DE'+
AB'C'D'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'B'D+
A'DE'+
AB'C'D'+
CO  =
A'B'D+A'DE'+AB'C'D'


                DUREE =      13,06 SECONDES
```

EXEMPLE 1 B

```
VOICI LE SYSTEME D EQUATIONS LOGIQUES
D01 =
C1*P+C1*H+C1*50+C1'*60*P


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
AB+
AC+
AD+
A'BE+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
AB+
AC+
AD+
BE+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
AB+
AC+
AD+
BE+
D01 =
AB+AC+AD+BE



D02 =
C1*P*H+C1*P*50+C1*H*P+C1*H*50+C1*50*P+C1*50*H


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
ABC+
ABD+
ABC+
ACD+
ABD+
ACD+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
ACD+
ABD+
ABC+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
ACD+
ABD+
ABC+
D02 =
ACD+ABD+ABC
```

```
D03 =
C1*P*H*50


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
ABCD+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
ABCD+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
ABCD+
D03 =
ABCD



D04 =
C1'*50'*60+C1'*50'*60'


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
A'D'E+
A'D'E'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'D'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'D'+
D04 =
A'D'



VOICI LE SYSTEME D EQUATIONS DONNANT LES INCOMPATI.
I01 =
50'*60


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
D'E+
COMPLEMENTATION DE LA FONCTION
E'+
D+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
E'+
D+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
E'+
D+
I01 =
E'+D
```

```
I02 =
C1*P*H*50


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
ABCD+
COMPLEMENTATION DE LA FONCTION
A'+
B'+
C'+
D'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'+
B'+
C'+
D'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'+
B'+
C'+
D'+
I02 =
A'+B'+C'+D'


PRODUIT DE L INVERSE DES INCOMPATIBILITES
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
E'+
D+
XXXXXXXXXXXXXXXXX
A'+
B'+
C'+
D'+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
A'E'+
B'E'+
C'E'+
D'E'+
A'D+
B'D+
C'D+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'E'+
B'E'+
C'E'+
D'E'+
A'D+
B'D+
C'D+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
D'E'+
A'D+
B'D+
C'D+
```

EQUATIONS DE DECISION MULTIPLIEES PAR LE PRODUIT DE L INVERSE DES INCOMPATI.
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
D'E'+
A'D+
B'D+
C'D+
XXXXXXXXXXXXXXXXX
AB+
AC+
AD+
BE+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
ABD'E'+
ACD'E'+
A'BDE+
BC'DE+
AB'D+
AC'D+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
ABD'E'+
ACD'E'+
A'BDE+
BC'DE+
AB'D+
AC'D+
AB'CE'+
ABC'E'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
ABD'E'+
A'BDE+
BC'DE+
AB'D+
AB'CE'+
ABC'E'+
BASE PRINCIPALE IRREDUCTIBLE
ACD'E'+
A'BDE+
AB'D+
AC'D+
ABC'E'+
BASE PRINCIPALE IRREDUCTIBLE
ABD'E'+
ACD'E'+
A'BDE+
AB'D+
AC'D+
BASE PRINCIPALE IRREDUCTIBLE
ABD'E'+
A'BDE+
AB'D+
AC'D+
AB'CE'+
BASE PRINCIPALE IRREDUCTIBLE
ACD'E'+
A'BDE+
BC'DE+
AB'D+
ABC'E'+
DEC=
ACD'E'+A'BDE+BC'DE+AB'D+ABC'E'

```
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
D'E'+
A'D+
B'D+
C'D+
XXXXXXXXXXXXXXXXX
ACD+
ABD+
ABC+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
ABCD'E'+
AB'CD+
ABC'D+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
ABCD'E'+
AB'CD+
ABC'D+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
ABCD'E'+
AB'CD+
ABC'D+
DEC=
ABCD'E'+AB'CD+ABC'D


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
D'E'+
A'D+
B'D+
C'D+
XXXXXXXXXXXXXXXXX
ABCD+
LE PRODUIT DE SOMMES DE MONOMES EST NUL
DEC=


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
D'E'+
A'D+
B'D+
C'D+
XXXXXXXXXXXXXXXXX
A'D'+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
A'D'E'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'D'E'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'D'E'+
DEC=
A'D'E'
```

EXEMPLE 1 6

```
VOICI LE SYSTEME D EQUATIONS LOGIQUES
D01 =
C1*P+C1*H+C1*50+C1'*60*P


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
AB+
AC+
AD+
A'BE+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
AB+
AC+
AD+
BE+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
AB+
AC+
AD+
BE+
D01 =
AB+AC+AD+BE



D02 =
C1*P*H+C1*P*50+C1*H*P+C1*H*50+C1*50*P+C1*50*H


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
ABC+
ABD+
ABC+
ACD+
ABD+
ACD+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
ACD+
ABD+
ABC+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
ACD+
ABD+
ABC+
D02 =
ACD+ABD+ABC
```

```
EQUATIONS DES CAS OUBLIES
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
AB+
AC+
AD+
BE+
ACD+
ABD+
ABC+
ABCD+
A'D'+
COMPLEMENTATION DE LA FONCTION
A'B'D+
A'DE'+
AB'C'D'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'B'D+
A'DE'+
AB'C'D'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'B'D+
A'DE'+
AB'C'D'+
CO  =
A'B'D+A'DE'+AB'C'D'

GENERATION DES INSTRUCTIONS NON EXCECUTEE CAR INCOMPATIBILITE DE CERTAINES EQUATI

                DUREE =      31,96 SECONDES
```

```
D03 =
C1*P*H*50


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
ABCD+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
ABCD+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
ABCD+
D03 =
ABCD



D04 =
C1'*50'*60+C1'*50'*60'


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
A'D'E+
A'D'E'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'D'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'D'+
D04 =
A'D'



VOICI LE SYSTEME D EQUATIONS DONNANT LES INCOMPATI.
I01 =
50'*60


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
D'E+
COMPLEMENTATION DE LA FONCTION
E'+
D+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
E'+
D+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
E'+
D+
I01 =
E'+D
```

PRODUIT DE L INVERSE DES INCOMPATIBILITES
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
E'+
D+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
E'+
D+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
E'+
D+

EQUATIONS DE DECISION MULTIPLIEES PAR LE PRODUIT DE L INVERSE DES INCOMPATI.
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
E'+
D+
XXXXXXXXXXXXXXXXX
AB+
AC+
AD+
BE+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
ABE'+
ACE'+
BDE+
AD+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
ABE'+
ACE'+
BDE+
AD+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
ABE'+
ACE'+
BDE+
AD+
DEC=
ABE'+ACE'+BDE+AD

```
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
E'+
D+
XXXXXXXXXXXXXXXXX
ACD+
ABD+
ABC+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
ABD+
ACD+
ABCE'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
ABD+
ACD+
ABCE'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
ABD+
ACD+
ABCE'+
DEC=
ABD+ACD+ABCE'
```

```
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
E'+
D+
XXXXXXXXXXXXXXXXX
ABCD+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
ABCD+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
ABCD+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
ABCD+
DEC=
ABCD
```

```
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
E'+
D+
XXXXXXXXXXXXXXXXX
A'D'+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
A'D'E'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'D'E'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'D'E'+
DEC=
A'D'E'
```

```
EQUATIONS DES CAS OUBLIES
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
AB+
AC+
AD+
BE+
ACD+
ABD+
ABC+
ABCD+
A'D'+
COMPLEMENTATION DE LA FONCTION
A'B'D+
A'DE'+
AB'C'D'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'B'D+
A'DE'+
AB'C'D'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'B'D+
A'DE'+
AB'C'D'+
CO  =
A'B'D+A'DE'+AB'C'D'


                DUREE =     24,50 SECONDES
```

Exemple 2. - Réservation de place d'avion.

Partons de l'exemple énoncé au § 1.3. du chapitre I.

2 cas sont envisagés :

A. Equations de décision sans incompatibilité.

L'énoncé du problème a permis d'écrire le système d'équations de décision :

D01 = DP * D1P * A + DT * D1T' * D1P * A + DP * D1P * A'

D02 = DT * D1T * A + DP * D1P' * D1T * A + DT * D1T * A'

D03 = DP * D1P' * A' + DP * D1P' * D1T' * A + DP * D1P' * D1T' * A

D04 = DT * D1T' * A' + DT' * D1P' * D1T' * A + DT' * D1P' * D1T' * A

N.B. :

Les variables sont celles définies au chapitre I à la restriction près que d est remplacé par D1.

B. Adjonction de l'équation d'incompatibilité.

IO1 = DP' * DT' + DP * DT.

Pour cet exemple, la table de correspondance entre les variables est :

Variables d'entrée	Variables normaliées
DP	A
D1P	B
A	C
DT	D
D1T	E

Remarque :

Dans l'équation de décision D04 de cet exemple, nous avons volontairement introduit une faute.

En effet, d'après l'énoncé, l'équation D04 correspondant à une mise en attente d'une place touriste était :

D04 = DT * D1T' * A' + DT * D1P' * D1T' * A + DT * D1P' * D1T' * A

(dans les deux derniers monômes on a mis DT au lieu de DT').

Les conséquences de cette erreur sont les suivantes :

1. l'équation des cas oubliés nous signale que

   DP' * D1P' * A * DT * D1T'

   a été omis.

2. lors de la transposition de ces équations en table, une erreur sera détectée venant du fait que la combinaison des conditions (2 1 0 1 1) correspondant au monôme DP * D1P' * DT * D1T' est commune à D03 et D04.

   En effet, la table des incompatibilités définie au Chap. I. page 10, interdit que ces deux équations se produisent simultanément.

EXEMPLE 2 A

```
VOICI LE SYSTEME D EQUATIONS LOGIQUES
D01 =
DP*D1P*A+DT*D1T'*D1P*A+DP*D1P*A'


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
ABC+
BCDE'+
ABC'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
BCDE'+
AB+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
BCDE'+
AB+
D01 =
BCDE'+AB



D02 =
DT*D1T*A+DP*D1P'*D1T*A+DT*D1T*A'


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
CDE+
AB'CE+
C'DE+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
AB'CE+
DE+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
AB'CE+
DE+
D02 =
AB'CE+DE



D03 =
DP*D1P'*A'+DP*D1P'*D1T'*A+DP*D1P'*D1T'*A'


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
AB'C'+
AB'CE'+
AB'C'E'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
AB'C'+
AB'E'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
AB'C'+
AB'E'+
D03 =
AB'C'+AB'E'
```

```
D04 =
DT*D1T'*A'+DT'*D1P'*D1T'*A+DT'*D1P'*D1T'*A'


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
C'DE'+
B'CD'E'+
B'C'D'E'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
C'DE'+
B'C'E'+
B'D'E'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
C'DE'+
B'D'E'+
D04 =
C'DE'+B'D'E'



EQUATIONS DES CAS OUBLIES
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
BCDE'+
AB+
AB'CE+
DE+
AB'C'+
AB'E'+
C'DE'+
B'D'E'+
COMPLEMENTATION DE LA FONCTION
A'BD'+
A'D'E+
A'B'CDE'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'BD'+
A'D'E+
A'B'CDE'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'BD'+
A'D'E+
A'B'CDE'+
C0  =
A'BD'+A'D'E+A'B'CDE'


                DUREE =     12,66 SECONDES
```

EXEMPLE 2 B

```
VOICI LE SYSTEME D EQUATIONS LOGIQUES
D01 =
DP*D1P*A+DT*D1T'*D1P*A+DP*D1P*A'


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
ABC+
BCDE'+
ABC'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
BCDE'+
AB+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
BCDE'+
AB+
D01 =
BCDE'+AB



D02 =
DT*D1T*A+DP*D1P'*D1T*A+DT*D1T*A'


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
CDE+
AB'CE+
C'DE+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
AB'CE+
DE+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
AB'CE+
DE+
D02 =
AB'CE+DE



D03 =
DP*D1P'*A'+DP*D1P'*D1T'*A+DP*D1P'*D1T'*A'


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
AB'C'+
AB'CE'+
AB'C'E'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
AB'C'+
AB'E'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
AB'C'+
AB'E'+
D03 =
AB'C'+AB'E'
```

```
D04 =
DT*D1T'*A'+DT'*D1P'*D1T'*A+DT'*D1P'*D1T'*A'


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
C'DE'+
B'CD'E'+
B'C'D'E'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
C'DE'+
B'C'E'+
B'D'E'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
C'DE'+
B'D'E'+
D04 =
C'DE'+B'D'E'



VOICI LE SYSTEME D EQUATIONS DONNANT LES INCOMPATI.
I01 =
DP'*DT'+DP*DT


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
A'D'+
AD+
COMPLEMENTATION DE LA FONCTION
AD'+
A'D+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
AD'+
A'D+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
AD'+
A'D+
I01 =
AD'+A'D



PRODUIT DE L INVERSE DES INCOMPATIBILITES
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
AD'+
A'D+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
AD'+
A'D+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
AD'+
A'D+
```

```
EQUATIONS DE DECISION MULTIPLIEES PAR LE PRODUIT DE L INVERSE DES INCOMPATI.
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
AD'+
A'D+
XXXXXXXXXXXXXXXXX
BCDE'+
AB+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
ABD'+
A'BCDE'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
ABD'+
A'BCDE'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
ABD'+
A'BCDE'+
DEC=
ABD'+A'BCDE'


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
AD'+
A'D+
XXXXXXXXXXXXXXXXX
AB'CE+
DE+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
AB'CD'E+
A'DE+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
AB'CD'E+
A'DE+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
AB'CD'E+
A'DE+
DEC=
AB'CD'E+A'DE


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
AD'+
A'D+
XXXXXXXXXXXXXXXXX
AB'C'+
AB'E'+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
AB'C'D'+
AB'D'E'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
AB'C'D'+
AB'D'E'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
AB'C'D'+
AB'D'E'+
DEC=
AB'C'D'+AB'D'E'
```

```
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
AD'+
A'D+
XXXXXXXXXXXXXXXXX
C'DE'+
B'D'E'+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
AB'D'E'+
A'C'DE'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
AB'D'E'+
A'C'DE'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
AB'D'E'+
A'C'DE'+
DEC=
AB'D'E'+A'C'DE'



EQUATIONS DES CAS OUBLIES
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
BCDE'+
AB+
AB'CE+
DE+
AB'C'+
AB'E'+
C'DE'+
B'D'E'+
COMPLEMENTATION DE LA FONCTION
A'BD'+
A'D'E+
A'B'CDE'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'BD'+
A'D'E+
A'B'CDE'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'BD'+
A'D'E+
A'B'CDE'+
CO  =
A'BD'+A'D'E+A'B'CDE'


                DUREE =     24,60 SECONDES
```

Exemple 3. - Calcul de primes.

"Dans une entreprise 4 types de primes peuvent être octroyés aux employés s'ils remplissent les conditions suivantes :

1. Prime d'ancienneté.

Si l'employé possède au moins 3 années de service (A3) et si son absence au bureau est inférieure ou égale ou moyenne à 1 mois ou 4 semaines(B4), la prime (D01) s'élève à 15 % du salaire mensuel (S_m).

2. Prime de présence.

Si l'employé à une ancienneté d'au moins un an (A1) et si son absence est limitée à 2 semaines par an (B2), la prime (D02) s'élève à 20 % de S_m diminuée de 8 % par semaine d'absence.

Si l'employé a une ancienneté inférieure à un an ($\overline{A1}$) , il recevra une prime de présence (D03) de 6 % du S_m pour autant que son absence moyenne soit inférieure à 2 jours (B0).

3. Prime de productivité.

Si l'employé a une ancienneté d'au moins 3 ans (A3) et s'il a été absent moins de 2 semaines (B2) ou si dans le cas contraire, ayant été absent moins d'un mois (B4), son signalement (N) est supérieur à 15, la prime (D04) vaudra $\frac{N}{20}$ * 20 % du S_m.

Si l'employé a une ancienneté inférieure à 1 an ($\overline{A1}$) , son absence est limitée à 2 jours (B0) et si son signalement est supérieur à 15 (N), sa prime D06 s'élèvera à (N-15) x 2 % du S_m.

4. Prime pour service exceptionnel.

Si l'employé a le minimum d'absence (B0) et possède une cote signalement supérieure à 15 (N), la prime (D07) s'élève à (N-15) * 4 % du S_m.

x

x x

L'énoncé du problème permet d'écrire immédiatement le système d'équations logiques :

D01 : A3 * B4

D02 : A1 * B2

D03 = $\overline{A1}$ * B0

D04 = A3 * B2 + A3 * B4 * N

D05 = A1 * $\overline{A3}$ * B2

D06 : $\overline{A1}$ * B0 * N

D07 = B0 * N

Donc ces relations Ai et Bj valent respectivement 1 lorsque le nombre d'années d'ancienneté est supérieur à i et lorsque le nombre de semaines d'absence est inférieur à j.

Du même coup, certaines incompatibilités relatives aux variables A et B peuvent être relevées. Elles s'expriment par :

I01 : $A3 * \overline{A1}$

I02 = $\overline{B4} * B2 + \overline{B4} * B0 + \overline{B2} * B0$.

Table de correspondance entre les variables :

Variables d'entrée	Variables normalisées
A3	A
B4	B
N	C
BO	D
B2	E
A1	F

EXEMPLE 3

```
VOICI LE SYSTEME D EQUATIONS LOGIQUES
D01 =
A3*B4+A3*B4*N+A3*B0*B4*B2'


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
AB+
ABC+
ABDE'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
AB+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
AB+
D01 =
AB



D02 =
A1*B2


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
EF+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
EF+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
EF+
D02 =
EF



D03 =
A1'*B0+A3*A1'*B2*B0


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
DF'+
ADEF'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
DF'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
DF'+
D03 =
DF'
```

```
D04 =
A3*B2+A3*B4*N+A3*B2*N


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
AE+
ABC+
ACE+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
AE+
ABC+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
AE+
ABC+
D04 =
AE+ABC



D05 =
A1*A3'*B2


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
A'EF+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'EF+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'EF+
D05 =
A'EF



D06 =
A1'*B0*N


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
CDF'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
CDF'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
CDF'+
D06 =
CDF'
```

```
D07 =
B0*N


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
CD+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
CD+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
CD+
D07 =
CD


VOICI LE SYSTEME D EQUATIONS DONNANT LES INCOMPATI.
I01 =
A3*A1'*B4+A3*A1'


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
ABF'+
AF'+
COMPLEMENTATION DE LA FONCTION
A'+
F+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'+
F+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'+
F+
I01 =
A'+F


I02 =
B4'*B2+B4'*B0+B2*B0


LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
B'E+
B'D+
DE+
COMPLEMENTATION DE LA FONCTION
D'E'+
BD'+
BE'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
D'E'+
BD'+
BE'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
D'E'+
BD'+
BE'+
I02 =
D'E'+BD'+BE'
```

```
PRODUIT DE L INVERSE DES INCOMPATIBILITES
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
A'+
F+
XXXXXXXXXXXXXXXXX
D'E'+
BD'+
BE'+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
A'D'E'+
A'BD'+
A'BE'+
D'E'F+
BD'F+
BE'F+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'D'E'+
A'BD'+
A'BE'+
D'E'F+
BD'F+
BE'F+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'D'E'+
A'BD'+
A'BE'+
D'E'F+
BD'F+
BE'F+
```

```
EQUATIONS DE DECISION MULTIPLIEES PAR LE PRODUIT DE L INVERSE DES INCOMPATI.
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
A'D'E'+
A'BD'+
A'BE'+
D'E'F+
BD'F+
BE'F+
XXXXXXXXXXXXXXXXX
AB+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
ABE'F+
ABD'F+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
ABE'F+
ABD'F+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
ABE'F+
ABD'F+
DEC=
ABE'F+ABD'F
```

```
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXX
A'D'E'+
A'BD'+
A'BE'+
D'E'F+
BD'F+
BE'F+
XXXXXXXXXXXXXXXX
EF+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
BD'EF+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
BD'EF+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
BD'EF+
DEC=
BD'EF
```

```
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXX
A'D'E'+
A'BD'+
A'BE'+
D'E'F+
BD'F+
BE'F+
XXXXXXXXXXXXXXXX
DF'+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
A'BDE'F'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'BDE'F'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'BDE'F'+
DEC=
A'BDE'F'
```

```
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
A'D'E'+
A'BD'+
A'BE'+
D'E'F+
BD'F+
BE'F+
XXXXXXXXXXXXXXXXX
AE+
ABC+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
ABCE'F+
ABD'EF+
ABCD'F+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
ABCE'F+
ABD'EF+
ABCD'F+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
ABCE'F+
ABD'EF+
DEC=
ABCE'F+ABD'EF
```

```
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
A'D'E'+
A'BD'+
A'BE'+
D'E'F+
BD'F+
BE'F+
XXXXXXXXXXXXXXXXX
A'EF+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
A'BD'EF+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'BD'EF+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'BD'EF+
DEC=
A'BD'EF
```

```
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
A'D'E'+
A'BD'+
A'BE'+
D'E'F+
BD'F+
BE'F+
XXXXXXXXXXXXXXXXX
CDF'+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
A'BCDE'F'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'BCDE'F'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'BCDE'F'+
DEC=
A'BCDE'F'
```

```
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
A'D'E'+
A'BD'+
A'BE'+
D'E'F+
BD'F+
BE'F+
XXXXXXXXXXXXXXXXX
CD+
RESULTAT DU PRODUIT DE SOMMES DE MONOMES
A'BCDE'+
BCDE'F+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
A'BCDE'+
BCDE'F+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
A'BCDE'+
BCDE'F+
DEC=
A'BCDE'+BCDE'F
```

```
EQUATIONS DES CAS OUBLIES
LA FONCTION EST DONNEE PAR
XXXXXXXXXXXXXXXXX
AB+
EF+
DF'+
AE+
ABC+
A'EF+
CDF'+
CD+
COMPLEMENTATION DE LA FONCTION
B'D'E'+
A'D'E'+
A'C'E'F+
B'C'E'F+
A'D'F'+
BASE PRINCIPALE COMPLETE METHODE LINEAIRE
B'D'E'+
A'D'E'+
A'C'E'F+
B'C'E'F+
A'D'F'+
BASES PRINCIPALES IRREDUCTIBLES METHODE LINEAIRE
BASE PRINCIPALE IRREDUCTIBLE
B'D'E'+
A'D'E'+
A'C'E'F+
B'C'E'F+
A'D'F'+
CO  =
B'D'E'+A'D'E'+A'C'E'F+B'C'E'F+A'D'F'


                DUREE =      39,80 SECONDES
```

Conclusion.

Le but poursuivi par cette étude était l'analyse automatique d'un problème transposé soit sous forme de table, soit sous forme d'équations logiques et la construction de l'organigramme correspondant à sa logique. Jamais dans ce qui précède, nous ne nous sommes attachés à ce que représentait le test d'une condition ou l'exécution d'une décision. Le développement de ces deux parties aurait considérablement limité le champ d'application des tables et des équations et aurait en échange rendu très peu de service au programmeur.

De plus, les équations logiques, faciles à manier, permettent outre l'analyse du problème, la correction automatique de certaines erreurs, en particulier les redondances grâce au programme de simplification des fonctions. Elles ont d'autre part l'avantage d'occuper beaucoup moins de place en mémoire et donc de permettre un plus grand nombre de variables (32 variables binaires contre 14 au maximum pour les tables). Une fois l'analyse terminée, les équations sont transposées en table de façon à permettre la génération du programme correspondant à la logique du problème. Dans ce cas aussi, les équations logiques ont un avantage de plus, celui de permettre un compactage maximum de la table. Ceci a pour conséquence une meilleureoptimisation du programme généré et uneplus grande clarté de la table.

Notons que ce n'est que durant l'analyse et non la génération du programme que le nombre de variables est limité à 14.

Pour la génération, le nombre maximum de variables pouvant être utilisé n'est pas prévisible. Il dépend de la logique du problème et de la grandeur du stack.

Ces considérations sur les équations logiques nous pousse à aller plus loin.

Il pourrait en effet être interessant de transposer les tables en équations logiques de façon à les faire profiter de l'avantage de ces derniers (correction automatique, compactage des règles, plus grand nombre de variables). Cela nécessite toutefois certaines précautions dans la simplification des fonctions lorsque l'on a des tables polyvalentes.

Annexe 1.

10.99.002 F

INPUT POUR PI

PROBLEME N° _____ TABLE DE DECISION. PAGE _____ SUR _____

1re CARTE

TA 0.....

CODE

NOMBRE DE CARTES

CARTE d' EN-TÊTE

CODE

NOMBRE DE CONDITIONS

NOMBRE DE SEQUENCES

CONDITIONS

CARTES "CONDITIONS"

CODE

NOMBRE DE DECISIONS

NOMBRE DE SEQUENCES

DECISIONS

CARTES "DECISION"

Annexe 2.

Description des cartes "données" pour le cas des tables de décision.

CODE : numéro compris entre 00 et 99. Il peut représenter, par exemple, un numéro de table.

Nombre de règles : représente le nombre de colonnes de la table initiale.

Code carte : C pour les cartes "condition"
D pour les cartes "décision"

Numéro de séquence :
- commence à 01 pour la 1ère ligne des décisions et la 1ère ligne des conditions
- est incrémenté de 1 pour chaque ligne "condition" ou "décision"
- doit être en séquence et continu

Nombre de possibilités : représente la valence V_i de chaque condition C_i, c'est-à-dire le nombre de valeurs que peut prendre la condition C_i (V_i doit être inférieure à 10).

Conditions :
- donnent les valeurs des conditions pour chaque règle.
- on peut avoir comme valeurs :

soit 0 (condition indifférente)
soit - k
soit k
} avec $1 \leq k \leq$ nombre de possibilités

Décisions :
- donnent les valeurs des décisions pour chaque règle.
- on peut avoir des nombres compris entre 0 et 99.

10.90.002 F

Annexe 3.

DESSIN DES CARTES — EQUATIONS LOGIQUES.

PROBLEME N° — PAGE — SUR

CARTE D' EN-TÊTE

EQV.....

CODE CARTE | NUMERO DE SEQUENCE | CONTINUATION

EQUATION DE DECISION

CODE CARTE | NUMERO DE SEQUENCE | CONTINUATION

EQUATION D' INCOMPATIBILITÉ

DERNIÈRE CARTE

FIN.....

Annexe 4.

Description des cartes "données" pour le cas des équations logiques.

- CODE CARTE : D s'il s'agit d'une équation logique de décision ;
  I s'il s'agit d'une équation logique représentant des incompatibilités entre variables.

- NUMERO DE SEQUENCE : - commence à 01 pour la 1ère équation de décision et la 1ère équation d'incompatibilité ;
  - est incrémenté de 1 pour chaque équation de décision et d'incompatibilité.

- CONTINUATION : / signifie que la carte suivante est une carte de continuation ;
  ␣ signifie "pas de carte de continuation".

- EQUATION : les équations peuvent s'écrire de la colonne 7 à 80.
  Il ne peut y avoir de blanc.
  Les opérateurs sont + (addition), x (multiplication), ' (complémentation).
  Les variables peuvent avoir au maximum 6 caractères quelconques à l'exception de , +, x, '.

- La dernière carte doit se terminer par FIN.

Annexe 5 .

TABLE DES INCOMPATIBILITES

PROG 4 .....

PROBLEME N° .....

PAGE ..... SUR .....

10.00.002 F

DESSIN DES CARTES

DERNIERE CARTE

STOP

Annexe 6.

Description des cartes "données" pour PROB 4.

CODE CARTE : doit contenir la lettre D.

NUMERO DE LIGNE : donne le numéro de l'équation de décision à laquelle se rapporte cette ligne de la table.

Col 7 et suivantes : un "1" dans la ième colonne ($i \geq 7$) de la carte signifie que la décision (i-6)ème est incompatible avec celle correspondant au numéro de ligne.
un "0" ou un " " signifie qu'elle n'est pas incompatible.

Remarque :

Les cartes représentant les incompatibilités entre différentes décisions peuvent ne pas exister. Ainsi, par exemple, si la décision 3 n'est incompatible avec aucune autre décision, on ne doit pas spécifier cette ligne dans la table.

BIBLIOGRAPHIE

[FL1] LAPSCHER, F. Ensembles ordonnés et algèbresde Boole Université des sciences et techniques du languedoc, UER de mathématiques - Tome 1 - juin 1972.

[FL2] LAPSCHER, F. Ensembles ordonnés et algèbres de Boole, Université des sciences et techniques du languedoc, UER de mathématiques - tome 2 - juin 1972.

[VW] VERBRAEKEN, W. De omzetting van beslissingstabellen in computerprogramma's Katholieke Universiteit Leuven, Instituut voor toegepaste economische weten schappen - 1972.

[VM] VERHELST, M. Décision tables : new developments and implementations Institute of Applied Economic Sciences, University of Louvain (Belgium) - decembre 1971.

[CB] CHARLES, B. Structures algébriques - Ensembles ordonné monoïdes alèbres de Boole Université des sciences et techniques du languedoc - 1971.

[BK] BAGLIN, G. et KLEE, J. Les tables de décisions. Initiation pratique - L'informatiqque - 1970.

[PK] KING, P.J.H. Conversion of decision tables to computer programs by rule mask techniques - communications of the ACM 9, 11 - november 1966.

[SP] POLLACK, S.L. Conversion of limited - entry decision tables to computer programs, communicatior of the ACM 8, 11 - november 1965.

[RS] REINWALD, L.T. and SOLAND, R.H. Conversion of limited - entry decision table to optimal computer programs I : minimum average processing time - Journal of the AC 13, 3 - juli 1966.

[MV] VERHELST, M.R. The conversion of limited - entry decision tables to optimal and near-optimal flowcharts two new algorithms - communications of the ACM 15

[PS] POLLACK, S.L. Analysis of the decision rules in decision tables - the Rand corporation - Santa Moni California - may 1963.

IIe PARTIE - ESSAI SUR LA CINEMATIQUE DES FICHIERS

INTRODUCTION.

Dans les pages qui suivent, nous avons essayé de voir quelles seraient les paramètres nécessaires à la génération automatique des mouvements entre un fichier maître et des fichiers esclaves. Pourquoi cette génération automatique ?
Dans la pratique, un grand nombre de problèmes traitent des mouvements entre plusieurs fichiers dont les enregistrements comprennent un ou plusieurs indicatifs de comparaison. Malheureusement, un même indicatif sur des enregistrements de fichiers différents se trouve soit sous forme différente (packé, dépacké, sans signe, avec signe), soit à des endroits différents. Cela implique à chaque coup une série d'instructions dans le programme pour convertir les indicatifs de façon à ce qu'ils soient comparables. De plus, les traitements occasionnés par la comparaison d'indicatifs sur les fichiers se fait à des moments différents suivant le programmeur; tantôt à la rupture des indicatifs, c'est à dire lorsqu'on a déjà lu l'article suivant, tantôt directement lors de la détection des articles sur lesquels il faut opérer un traitement. La génération automatique permettra de systématiser tout cela moyennant un certain nombre d'informations que l'analyste pourra fournir dans une table. De plus, il s'agit comme pour les tables de décision de prévoir une analyse permettant de déceler un maximum d'erreurs. Une différence fondamentale réside cependant dans le fait qu'ici cette analyse ne pourra se faire qu'au moment de l'exécution puisque le contenu des fichiers ne peut être connu avant. A l'aide des paramètres que nous aurons défini, nous pourrons dire à tout instant de l'exécution quelle sera la situation (l'article est dans le fichier maître et pas dans le fichier esclave, déclassement dans le fichier esclave etc....). Ces paramètres devront en plus servir à déterminer quelles sont les fichiers qui devront progresser lors de la phase suivante.

- - - - - - - - - -

CHAPITRE I

CONFIGURATION DE BASE ET PARAMETRES DE CINEMATIQUE.

I. 1. Définition de la configuration de base.

La cinématique des fichiers sera discutée sur la base de la configuration de la figure I-1 qui comprend :

- 3 fichiers : A, B et C que nous appellerons respectivement :
  - primaire - ancien (ou principal, ou maître),
  - secondaire (ou mouvement)
  - primaire - nouveau.
- 1 fichier : D utilisé lorsqu'il s'agira d'enregistrer des messages d'erreur résultant des comparaisons successives des indicatifs de A et de B.

Le but que nous poursuivons est la systématisation d'une procédure de comparaison des indicatifs, de détection et de traitement des opérations à exécuter ainsi que des anaomalies ou des cas moins fondamentaux tels que les déclassements dans B et accidentellement dans A.

Pour cela nous adoptons un schéma standard à savoir celui de la figure I.1 où pour chaque fichier d'entrée nous réservons deux zones :

- l'une pour la lecture Z_L
- l'autre comme zone de travail WORK-NAME,

et où de même pour chaque fichier de sortie nous réservons deux zones :

- l'une p our l'écriture Z_E
- l'autre comme zone de travail WORK-NAME.

I. 2. Définition des paramètres de cinématique :

Afin de permettre la détection et l'analyse des particularités de traitement de ces fichiers, 4 paramètres de cinématique sont définis.
Il s'agit de :

IND1 = $(I_A, I_B)_P$: résultat de la comparaison de l'indicatif du fichier A (I_A) et de celui du fichier B (I_B) à l'instant présent.

IND2 = $(I_A, I_B)_P$: résultat de la comparaison des indicatifs de A et de B à l'instant précédent.

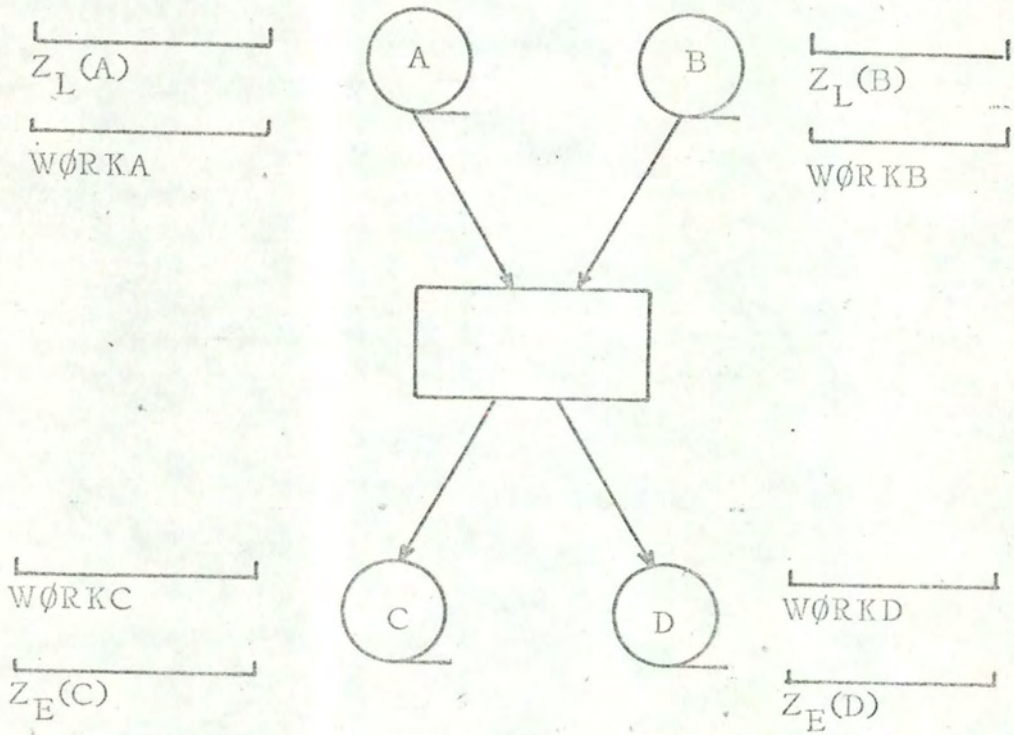


Fig. I. 1

IND3 = (I_B^N , I_B^A) : résultat de la comparaison des indicatifs du fichier B nouveau (à l'instant présent) et ancien (à l'instant précédent).

IND4 = (I_A^N, I_A^A) : même chose que IND3 pour le fichier A.

Ces résultats de comparaison ont pour valeur 1, 3, 2 selon que le premier indicatif est respectivement inférieur, égal ou supérieur au second. Nous supposerons tout d'abord que le fichier A a été trié préalablement et qu'il est donc sans déclassement. Ceci permet de ne s'intéresser qu'aux 3 premiers paramètres.

De plus, nous avons choisi de faire progresser le fichier B lorsque la valeur de IND1 = 3 c'est-à-dire lorsqu'à l'instant présent les indicatifs des deux fichiers sont égaux.

Considérons l'exemple suivant :

- un fichier maître A est composé d'une série d'enregistrements dont les indicatifs rangés en ordre croissant sont (1, 2, 4, 7, 10, 11).
- un fichier mouvement B est formé d'une série d'enregistrements dont les indicatifs sont dans l'ordre (3, 4, 4, 4, 3, 5, 7, 8, 8, 9, 11).

Le tableau T.I.1 résume les valeurs prises par les indicatifs en cours d'exécution.

Dans ce dernier, une flèche indique lequel des deux fichiers progresse lors du passage d'une phase à l'autre de l'exécution.

numéro de phase	fichier A I_A	fichier B I_B	IND1 $(I_A, I_B)_P$	IND2 $(I_A, I_B)_P$	IND3 (I_B^N, I_B^A)	Opérations
1	1	3	1	-	-	
2	2	3	1	1	3	$\bar{B}A$ (1)
3	4	3	2	1	3	$\bar{B}A$ (2)
4	4	4	3	2	2	$B\bar{A}$ (3)
5	4	4	3	3	3	BB (4)
6	4	4	3	3	3	BB (4)
7	4	3	2	3	1	déclass. dans B
8	4	5	1	(2) 3	2	BA(4)SWITCH→
9	7	5	2	1	3	SWITCH→(
10	7	7	3	2	2	$B\bar{A}$ (5)
11	7	8	1	3	2	BA(7)SWITCH→
12	10	8	2	1	3	SWITCH→(
13	10	8	2	2	3	BB (8)
14	10	9	2	2	2	$B\bar{A}$ (8)
15	10	11	1	2	2	$B\bar{A}$ (9)
16	11	11	3	1	3	$\bar{B}A$ (10)

T.I. 1

$\bar{B}A$ (1): l'article dont l'indicatif est 1 se trouve dans A mais pas dans B.

I. 3. Détection des opérations à exécuter.

Les valeurs des paramètres indiquées dans les colonnes correspondantes de T.I. 1 nous permettent de relever les cas suivants :

1. $B\bar{A}$: l'article se trouve dans le fichier B et pas dans le fichier A. Ce cas se produit pour les valeurs (-, 2, 2) des paramètres pris dans l'ordre IND1, IND2 et IND3 et ou "-" signifie valeur quelconque

2. $\bar{B}A$: l'article se trouve dans le fichier A et pas dans le fichier B. Ce cas d'opération correspond aux valeurs (-, 1, 3) moyennant une information supplémentaire (SWITCH).

3. BA : l'article se trouve dans A et dans B (1, 3, 2).

4. BB : Il y a deux articles ayant même indicatif dans B.
Cela a lieu pour les combinaisons de valeurs (3, 3, 3) et (2, 2, 3).

5. Déclassement dans B (1, 2, 3) et (3, 2, 3).

Remarquons que le cas $\overline{BA}$ (l'article ne se trouve ni dans A, ni dans B) ne peut évidemment être détecté dans la séquence.
Notons aussi que la détection des cas se fait à la rupture des indicatifs.
Les valeurs attribuées aux différents cas s'expliquent par les considérations suivantes :

1. pour voir réaliser $B\overline{A}$, il faut et il suffit que :

   - le paramètre IND1 soit 1 ou 2 ou3. En effet, ce cas ne sera détecté qu'après avoir lu un nouvel article de B et donc peut donner lieu a une valeur quelconque de ce paramètre.

   - le second paramètre soit 2 qui est le signe que l'article de B ne pourrait pas être présent dans A à la lecture précédente.

   - le troisième paramètre soit 2 caractérisant la rupture normale pour aboutir à ce cas.

   La détection de $B\overline{A}$ dans T.I. 1 à lieu lors des phases 4, 10, 14 et 15

2. pour voir réaliser $\overline{B}A$, il faut et il suffit que :

   - le premier paramètre soit 1 ou 2 ou 3 donc sans importance. Ce cas apparaîtra après avoir lu un nouvel article de A et lui aussi peut donner lieu à une valeur quelconque de ce paramètre.

   - le second paramètre soit 1 signifiant que l'article précédent de A ne se trouve pas dans B.

   - le troisième paramètre soit normalement 3 puisqu'il n'y aura pas eu lecture de B.

   Il y a toutefois une restriction importante à cette procédure. En effet, la configuration de valeur (2, 1, 3) ainsi que (1 1 3) non seulement peut provoquer l'enregistrement d'un $\overline{B}A$ mais également être l'indice de la fin d'un BA. Dans ce cas, il ne faut pas enregistrer un $\overline{B}A$ et pour qu'il en soit ainsi, il faudra mémoriser l'opération BA par un aiguillage.
   Lorsqu'on tombe sur la configuration (-, 1, 3) on regardera si le SWITCH = 1 auquel cas l'on n'enregistrera pas $\overline{B}A$ et on remettra le SWITCH à zéro.

La détection de $\overline{BA}$ dans T.I. 1 a lieu lors des phases 2, 3, 16.

3. pour voir réaliser BA, il faut et il suffit que :

- le premier paramètre soit 1 puisque l'égalité sera détectée à la rupture et donc pour l'indicatif de A plus petit que l'indicatif de B.
- le second paramètre soit 3 puisqu'avant la rupture on avait $I_A = I_B$.
- le troisième paramètre soit 2. Il ne peut normalement en être qu'ainsi puisque les indicatifs sont normalement en ordre croissant.

La détection de BA dans T.I.1. a lieu lors des phases 8, 11.

4. pour voir réaliser BB, il faut et il suffit que :

- le premier paramètre soit identique au second puisque la comparaison a l'instant présent et à l'instant précédent doit être la même du fait que l'on a 2 fois le même indicatif dans B.
- le second paramètre soit 2 ou 3 puique la détection du cas ne peut se faire qu'après avoir fait progresser B. Ce qui n'a lieu que si $I_A \geqslant I_B$ à l'instant précédent.
- le troisième paramètre soit 3 caractérisant un double dans B.

La détection de BB dans T.I. 1 à lieu lors des phases 5, 6 , 13.

5. pour voir réaliser "déclassement dans B", il faut et il suffit que :

- le premier paramètre soit 2 indiquant que $I_A > I_B$. Cela vient du fait que l'on vient de faire progresser le fichier B (pour détecter le déclassement).
- le second paramètre soit 3 ou 2 puisque pour que B progresse il fallait qu'à l'instant précédent l'indicatif du fichier A soit $\geqslant$ à l'indicatif du fichier B.
- le troisième paramètre soit 1 caractérisant le déclassement dans B

La détection "déclassement dans B" a lieu lors de la phase 7.

Remarquons encore que parmi les 27 combinaisons possibles des paramètres (3^3), 11 d'entre eux ont des valeurs d'informations, les autres cas ne peuvent normalement jamais se produire.

Prenons, par exemple, le cas (3, 1, 1).

IND2 = 1 signifie que précédemment I_A était inférieur à I_B.

Or, maintenant $I_A = I_B$ (IND1 = 3). Cela signifie donc que :

- ou bien l'on a fait progresser B, mais alors dans ce cas il faudrait IND3 = 2.
- ou bien l'on a fait progresser A, mais alors dans ce cas on devrait avoir IND3 = 3.

Il s'agit donc d'une situation impossible.

La figure I.2 résume l'ensemble des cas correspondants aux valeurs des indicatifs.

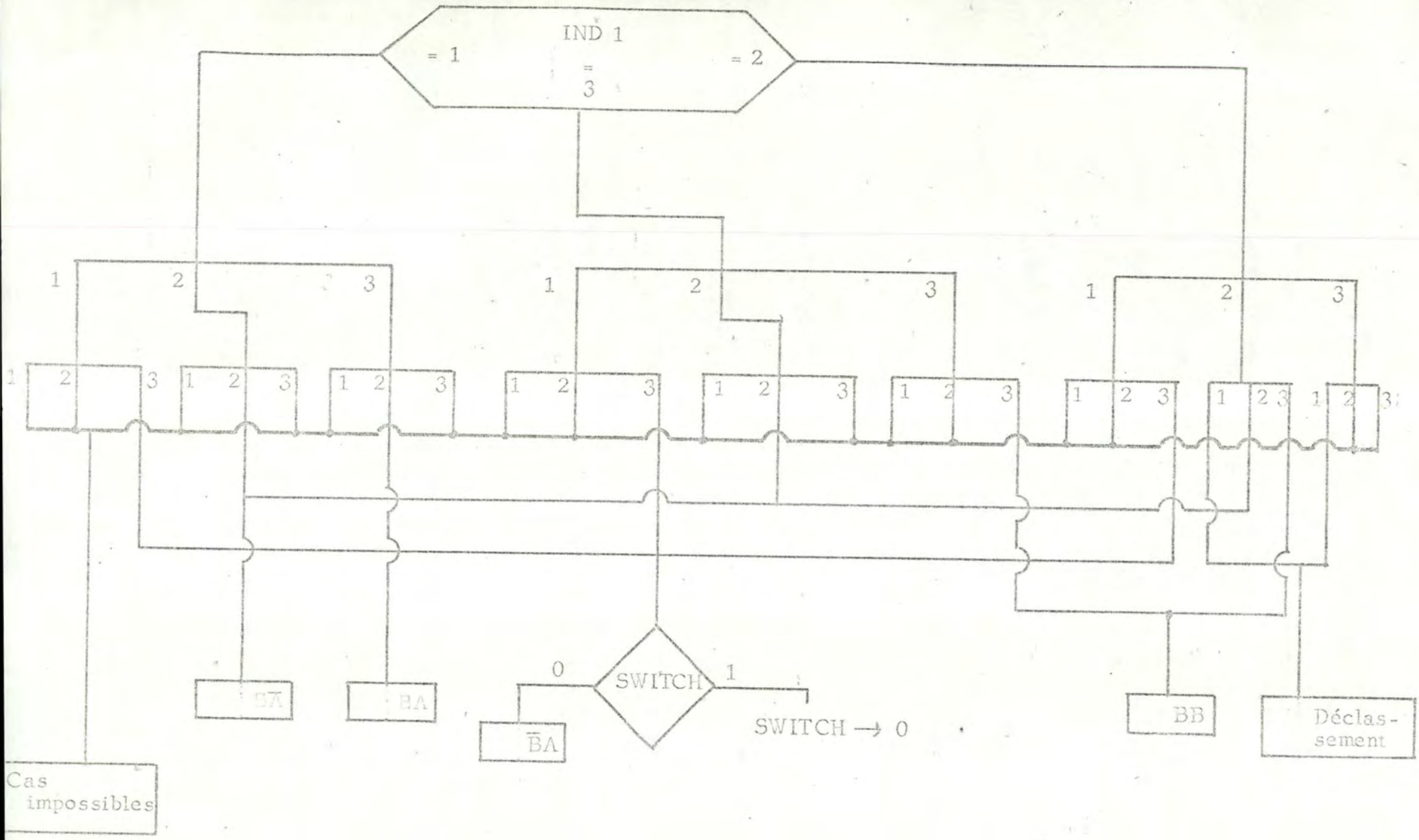

IND 1
= 1
= 2
= 3
1 2 3
SA
BA
0
SWITCH
1
BA
SWITCH → 0
BB
Déclas-
sement
Cas
impossibles

Fig. I. 2

CHAPITRE II

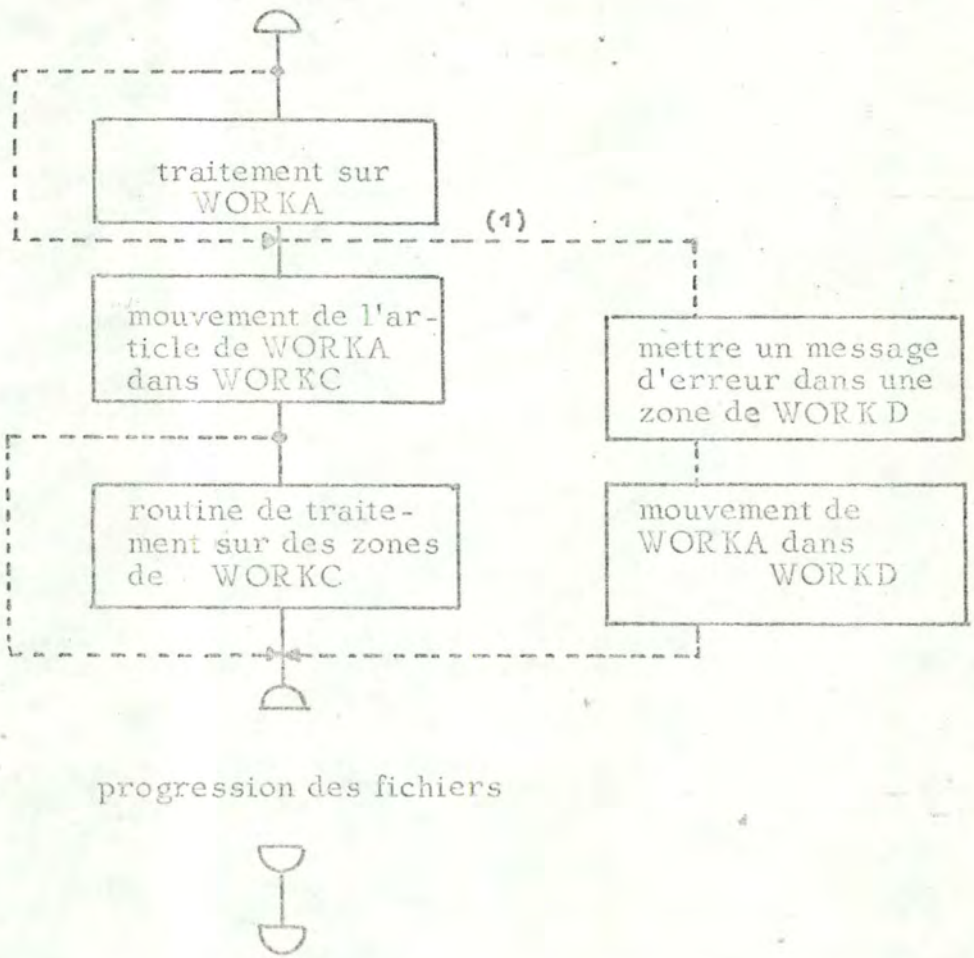
ETUDE DES MOUVEMENTS DE FICHIERS EN FONCTION DES CAS D'OPERATIONS.

Nous allons voir maintenant comment de manière encore rudimentaire vont se faire les mouvements d'articles des fichiers dans les différentes zones WORK-NAME, Z_I et Z_E de façon à chaque fois revenir après le traitement à une configuration qui permette de continuer de la même manière que précédemment.

II. 1. $\overline{BA}$: l'article se trouve dans A mais pas dans B.

Cette opération rappelons-le sera détectée lorsque les paramètres de cinématique auront les valeurs (-, 1, 3) avec le SWITCH = 0. De plus, comme la détection se fait à la rupture sur le numéro d'indicatif, l'on vient de lire un nouvel article du fichier A pour le mettre dans Z_L (A). L'article pour lequel il y a eu la détection de $\overline{BA}$ est donc dans WORKA.

L'organigramme correspondant à ce traitement aura la structure suivante



Dans la plupart des cas, la détection de $\overline{B}A$ déclenchera la recopie de l'article se trouvant dans WORKA moyennant éventuellement un traitement.

Les arcs de l'organigramme en trait discontinu permettent de court-circuiter certains traitements suivant les applications.
Le chemin (1) représente le traitement $\overline{B}A$ d'une application ou le fait de ne pas avoir d'article correspondant dans le fichier B est une erreur.

Reprenons l'exemple du tableau T.I. 1 et supposons que nous soyons à la phase 3.. La configuration au moment où l'on détecte $\overline{B}A$ (2) c'est-à-dire l'absence d'article d'indicatif 2 dans B est la suivante:

$Z_L(A)$	4	$Z_L(B)$	3
WORKA	2	WORKB	-

A la sortie de la routine soit à la phase suivante, nous aurons :

$Z_L(A)$	4	$Z_L(B)$	4
WORKA	-	WORKB	3

car $\overline{B}A$ a été détecté par (2, 1, 3)

II. 2. $B\overline{A}$: l'article se trouve dans B mais pas dans A.

Ce cas exige dans la pratique un traitement plus long que le précédent.
$B\overline{A}$ sera détecté à la rencontre de la combinaison des valeurs (-, 2, 2) et donc lorsqu'on a fait progresser B.
On peut constater que dans tous les cas au moment d'entrer dans cette routine, la zone WORKA est toujours libre. Elle a en effet déjà dû être traitée puisque pour détecter $B\overline{A}$ il faut nécessairement que l'indicatif de l'article B soit plus petit que celui de A.

De plus, comme pour le cas $\overline{B}A$, l'article concerné est dans WORKB puisque l'on vient de lire un nouvel article du fichier B dans $Z_L(B)$.
Ce cas d'opération provoque soit la création automatique d'un nouvel article soit la création suivant une valeur déterminée d'un code condition.

Description de l'organigramme.

Le test :

(1) est court-circuité dans le cas d'une création automatique. Par contre, si la création nécessite la présence d'une valeur particulière d'une code condition pour la distinguer de la modification et l'annulation, un message d'erreur sera détecté dans ces deux derniers cas puisqu'on ne peut annuler et modifier un article inexistant.

(2) représente une routine pouvant être court-circuitée et qui traite les zones de WORKB exigeant un calcul. Le résultat sera transféré par celle-ci dans WORKA.

(3) branchement à une routine de contrôle qui détectera la présence de toutes les zones de WORKA. Cela est nécessaire dans le cas ou plusieurs articles du fichier B sont nécessaires à la création d'un article du fichier A. Au cas où toutes les zones ne seraient pas présentes on renvoit à un cas d'anomalies. Ce contrôle n'est pas obligatoire et peut être sauté (chemin 4).

(5) pour certaines applications, il peut se faire que la détection de $B\bar{A}$ doit être considéréecomme une anomalie.

Prenons l'exemple de la phase 4 du tableau T.I. 1 pour laquelle on détecte $B\bar{A}$ (3) c'est-à-dire l'absence d'article d'indicatif 3 dans A.

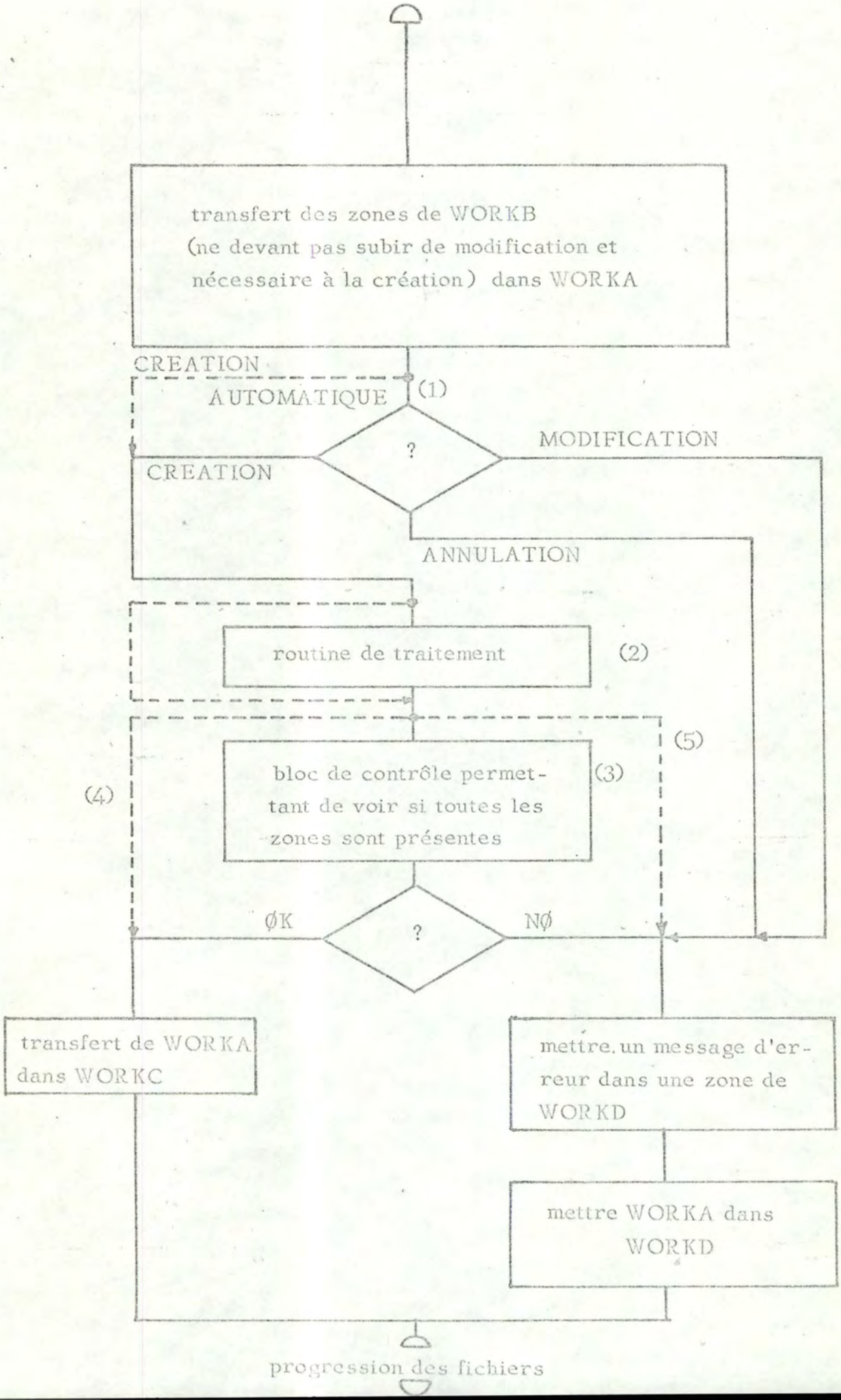
Le contenu des zones d'entrée est alors le suivant :

$Z_L(A)$	4	$Z_L(B)$	4
WORKA		WORKB	3

Après l'exécution de la routine nous aurons :

$Z_L(A)$	4	$Z_L(B)$	4
WORKA		WORKB	4

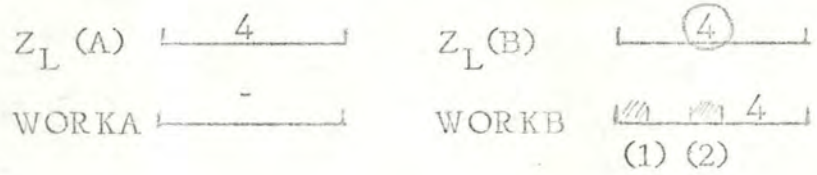
car la détection de $B\bar{A}$ s'est faite par la combinaison de valeurs (3, 2, 2), IND1 = 3 indiquant que c'est le fichier B qui doit progresser.


transfert des zones de WORKB
(ne devant pas subir de modification et
nécessaire à la création) dans WORKA
CREATION
AUTOMATIQUE
(1)
?
MODIFICATION
CREATION
ANNULATION
routine de traitement
(2)
(5)
bloc de contrôle permet-
tant de voir si toutes les
zones sont présentes
(3)
(4)
ØK
?
NØ
transfert de WORKA
dans WORKC
mettre un message d'er-
reur dans une zone de
WORKD
mettre WORKA dans
WORKD
progression des fichiers

II. 3. BA : l'article se trouve dans A et dans B.

Au moment de la détection de ce cas, l'article concerné se trouve, pour le fichier A dans Z_L (A). Par contre, pour le fichier B, l'article en correspondance avec celui de Z_L (A) se trouve dans WORKB à la condition toutefois qu'il n'y ait qu'un seul enregistrement par indicatif dans B. Dans le cas ou le fichier B possède plusieurs enregistrements pour l'indicatif en question, tous ceux-ci, sauf le dernier, auront déjà été traité (avec transfert éventuellement dans certaines zones de WORKA) '.

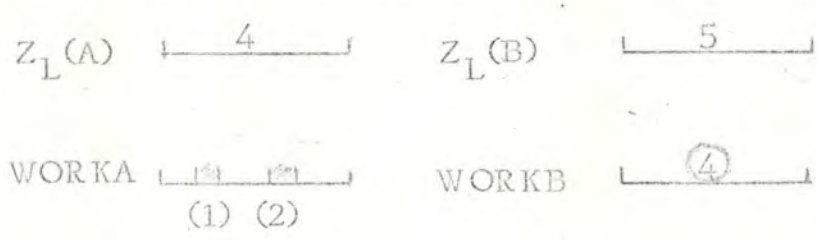
Exemple : supposons que l'on soit à la phase 5 du tableau T.I. 1.
A ce moment la nous avons la configuration :



On a entouré d'un rond l'un des deux indicatifs 4 des enregistrements du fichier B pour pouvoir les distinguer.
A ce moment a lieu la détection du cas BB (4) c'est-à-dire la présence d'au moins deux articles ayant le même indicatif (4) dans le fichier B.

La routine BB se charge alors de celui des articles 4 qui se trouve dans WORKB. Ce traitement peut, par exemple, transférer certaines zones de WORKB dans WORKA. La sortie de BB libère automatiquement la zone WORKB et déclenche la lecture d'un nouvel article sur B.

Supposons alors que l'article suivant à lire dans B soit l'indicatif numéro 5. Nous aurons alors.



C'est à ce moment qu'aura lieu la détection de AB (4) c'est-à-dire la présence d'un article 4 dans A et dans B.

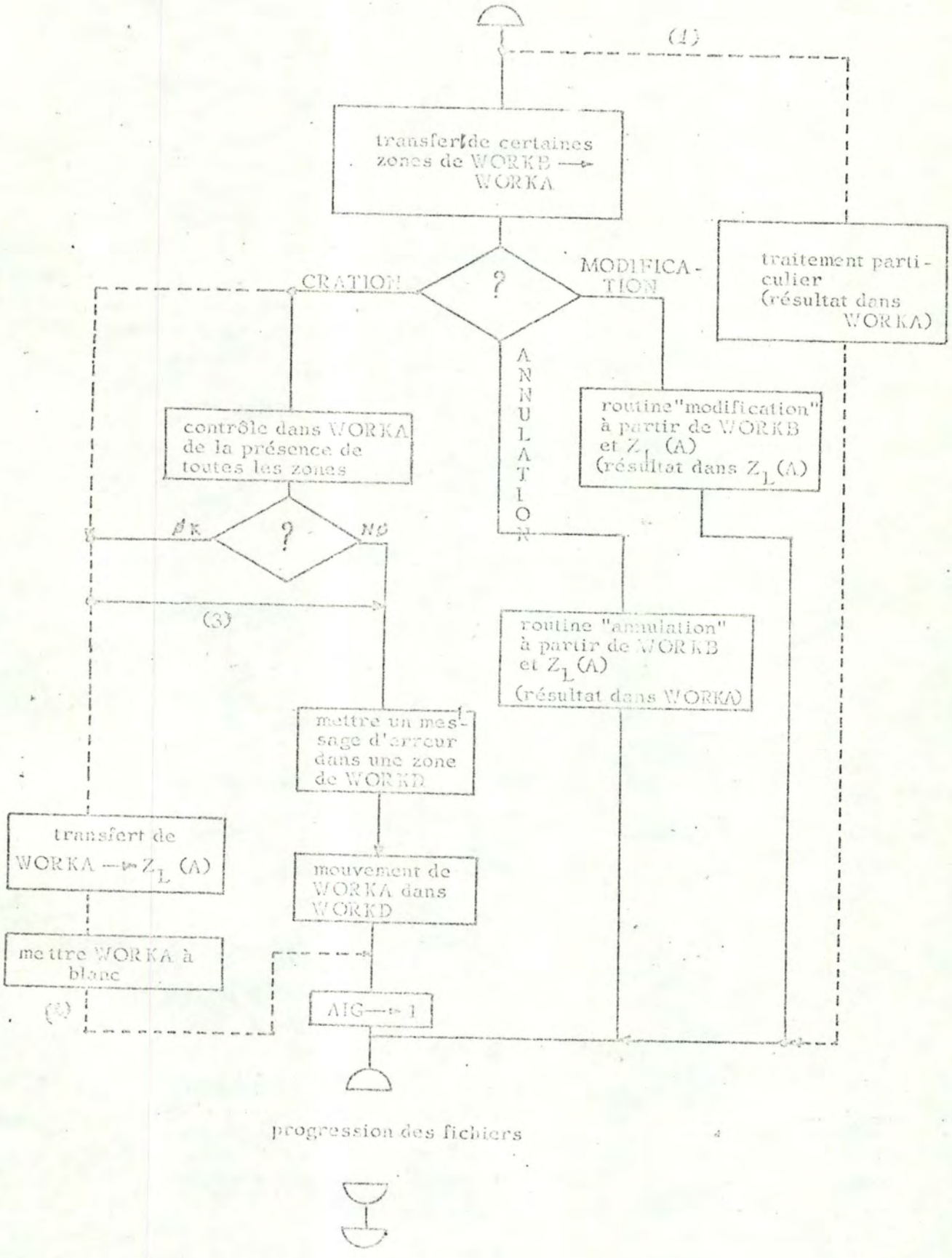
Dans le cas, par exemple, d'une mise à jour, le traitement précédent (BB) aura déjà transféré dans WORKA, les nouvelles zones correspondant aux enregistrements précédents ayant un indicatif 4.

Explication de l'organigramme.

Le chemin :

(1) correspond à un traitement particulier tel que, par exemple, une addition cumulative d'une même zone de tous les enregistrements de B ayant le même indicatif.

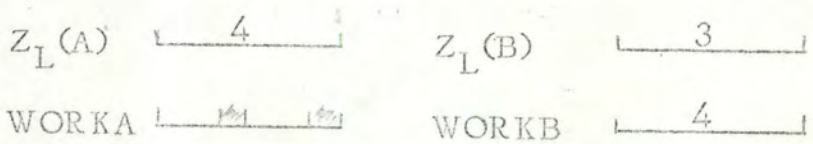
Lors d'une création, nous avons dans WORKA l'article que l'on vient de créer et dans Z_L (A) l'ancien article.
On peut à ce moment là prendre la décision soit de conserver l'ancien article et mettre le nouveau sur un fichier d'erreur (3), soit de supprimer l'ancien et garder le nouveau (2). De toute façon, il faut que l'article que l'on garde se trouve à la sortie de la routine dans Z_L (A) de façon à pouvoir continuer le traitement normalement.


(1)
transfert de certaines zones de WORKB → WORKA
?
CRATION
MODIFICA-TION
ANNULATION
traitement particulier (résultat dans WORKA)
contrôle dans WORKA de la présence de toutes les zones
routine "modification" à partir de WORKB et Z_L (A) (résultat dans Z_L (A)
ØK
?
KO
(3)
routine "annulation" à partir de WORKB et Z_L (A) (résultat dans WORKA)
mettre un message d'erreur dans une zone de WORKD
transfert de WORKA → Z_L (A)
mouvement de WORKA dans WORKD
mettre WORKA à blanc
(2)
AIG → 1
progression des fichiers

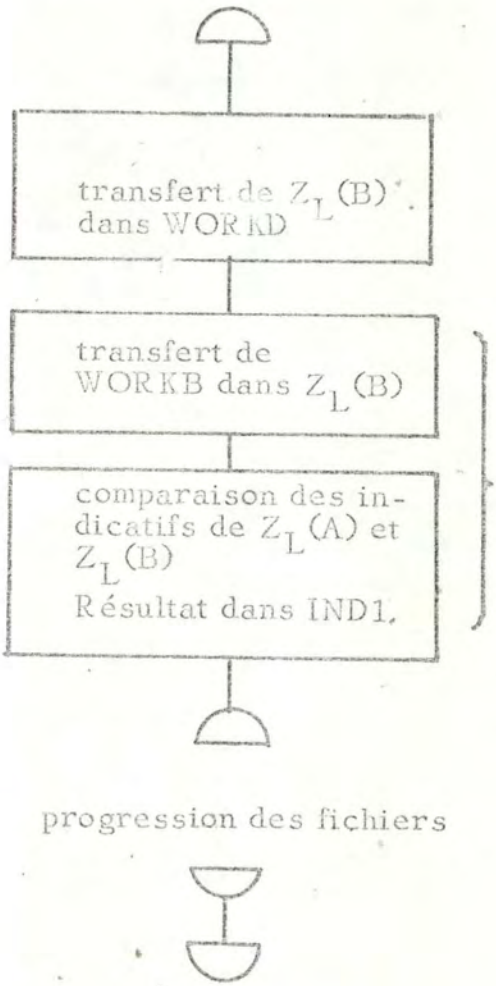
II. 4. "Cas de déclassement dans B".

Ce cas fait exception et est détecté directement et non lors de la lecture de l'article suivant. Cela vient du fait que celui-ci est tout simplement envoyé sur le fichier des anomalies (D) avec un message d'erreur.

Au moment de la détection du cas de déclassement (Phase 7 du tableau T.I. 1) la configuration des zones d'entrée est la suivante :



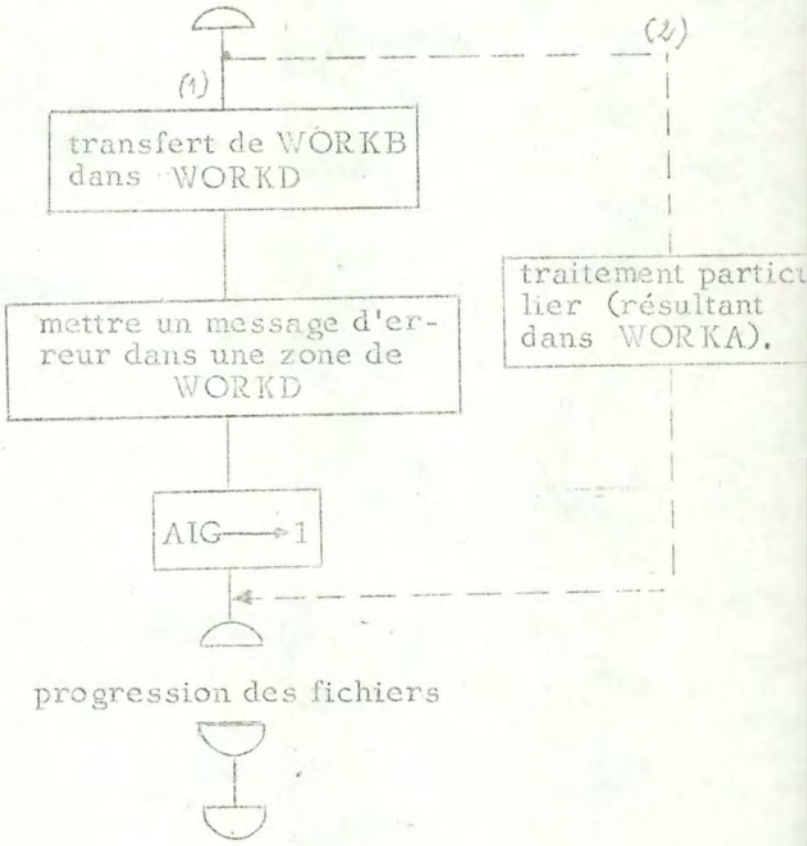
L'article déclassé (3) est donc dans $Z_L(B)$ au moment de l'entrée dans la routine.
Après l'envoi de ce cas sur le fichier des anomalies, il faut rétablir la situation qu'il y avait avant et rétablir les paramètres de cinématique aux bonnes valeurs.



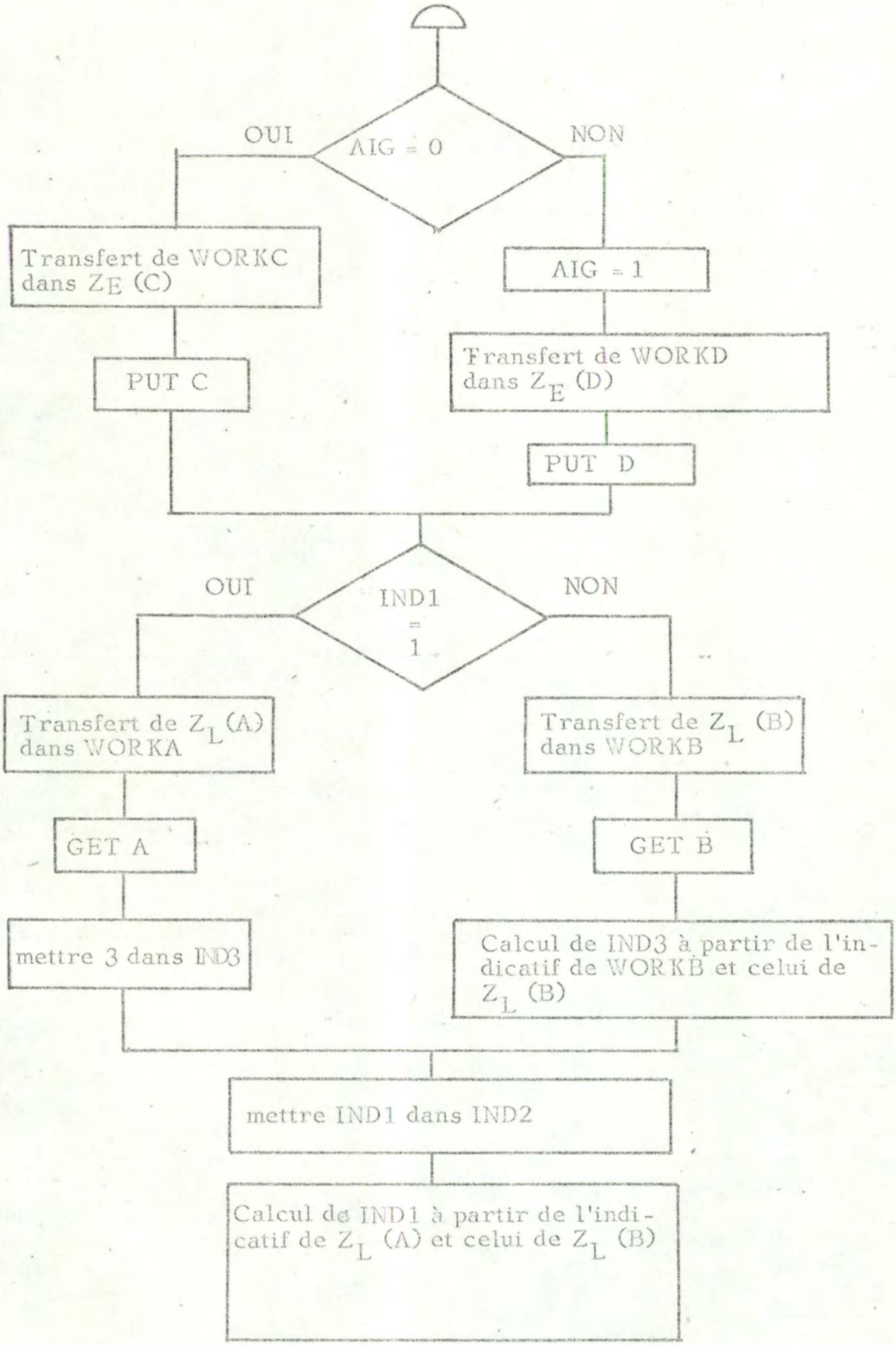
II. 5. BB : deux articles de B ont le même indicatif.

Au moment de la détection d'un double dans B, la zone WORKA est libre. Les deux enregistrements de même indicatif sont dans Z_L(B) et WORKB.

Les traitements seront différents selon que les doubles sont (2) ou ne sont pas (1) permis dans B. Dans ce dernier cas, les traitements seront assez variés. Disons simplement que les modifications qui sont à faire dans le fichier maître doivent avoir lieu dans la zone WORKA, à partir de l'information de WORKB. Il ne peut être question dans cette routine de toucher aux informations se trouvant dans Z_L (A) puisque c'est seulement à la rupture sur le numéro d'indicatif de B que l'on saura à quel cas se rapportent ces doubles (voir l'exemple du cas AB au § 3).



II. 6. Cas - 13 avec SWITCH = 1.


AIG = 0
OUI
NON
Transfert de WORKC dans Z_E (C)
PUT C
AIG = 1
Transfert de WORKD dans Z_E (D)
PUT D
IND1 = 1
OUI
NON
Transfert de Z_L (A) dans WORKA
GET A
mettre 3 dans IND3
Transfert de Z_L (B) dans WORKB
GET B
Calcul de IND3 à partir de l'indicatif de WORKB et celui de Z_L (B)
mettre IND1 dans IND2
Calcul de IND1 à partir de l'indicatif de Z_L (A) et celui de Z_L (B)

II. 8. Application :

Le problème posé est illustré par la figure II. 1 dans laquelle on distingue les fichiers maîtres nouveau (A) et ancien (C) et le fichier (B) secondaire. Ce dernier permet la "mise à jour" par cartes.

Pour un même enregistrement de A, on peut avoir dans B, 3 types d'enregistrements correspondants (de même indicatif).

Ces types sont distingués par un code carte (C_c)

1 pour le NOM

2 pour l'ADRESSE

3 pour la PROFESSION.

De même un code opération (C_o) précise le genre d'opération à effecteur :

A - p our ANNULATION

C - pour CREATION

M - pour MODIFICATION.

Le fichier B est classé en ordre croissant sur :

- le numéro de compte (en majeur)
- le code-opération (en inter)
- le code-carte (en mineur).

Le jeu d'essai pour le fichier A est (2, 4, 5, 8, 10) tandis que pour le ficher B on a les enregistrements (1C1, 1C2, 1C3, 2M1, 2M3, 3C1, 3C2, 3C3, 4A1, 4A2, 1M3, 6M2, 6M3, 7A3, 9C1, 9C2, 9C3, 9M1, 10M1).

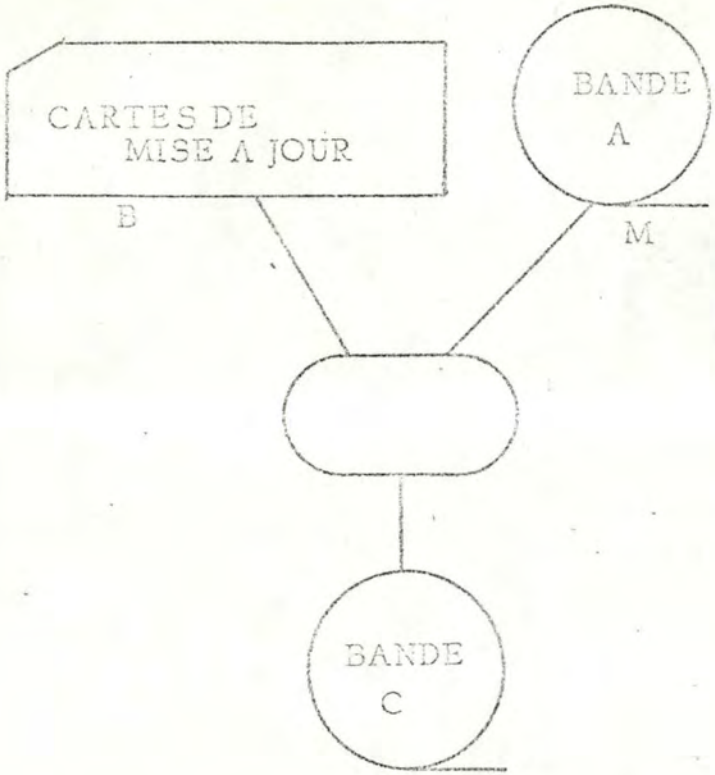
Pour ces derniers, le 1e caractère représente le numéro d'indicatif (I_A),

le 2d caractère représente le code opération (C_0),

et le 3e caractère représente le code carte (C_c).

Le tableau T.II-1 reprend les différentes phases d'exécution. Dans le fichier de sortie C, nous aurons les enregistrements (1, 2, 3, 4, 5, 8, 9) et dans le fichier des anomalies D nous aurons (6, 7).

Pour 6 nous avons une modification sur inexistant.
Pour 7 nous avons une annulation sur inexistant.



BANDE A/C

NUMERO de COMPTE	NOM	ADRESSE	PROFESSION
1 — 10	11 — 78	79 — 146	147 — 214

CODE OPERATION

CODE CARTE

NUMERO de COMPTE			
1 — 10	11	12 — 79	80
NUMERO de COMPTE	1	NOM	Code r.
NUMERO de COMPTE	2	ADRESSE	Code r.
NUMERO de COMPTE	3	PROFESSION	Code r.

A = ANNULATION

C = CREATION

M = MODIFICATION

CARTES B

REMARQUES :

1. Pour l'indicatif 9 nous avons une création suivie immédiatement d'une modification. Pour pouvoir réaliser cela il faudra détecter dans la routine BB la fin de la création et d'autre part, prévoir dans la partie annulation de la routine BA un traitement spécial de modification sur un article se trouvant dans WORKA.

2. Le dernier enregistrement (10) sera traité lors de la détection de fin de fichier.

EXEMPLE GENERAL.

Numéro de phase	I_A	I_B	C_0	C_c	IND_1	IND_2	IND_3	$Z_L(A)$	WORKA	$Z_L(B)$	WORKB	Opération
1	2	1	C	1	2	-	-	2	2	1C1	1C1	
2	2	1	C	2	2	2	3	2	-	1C2	1C1	BB (1)
3	2	1	C	3	2	2	3	2	1C1	1C3	1C2	BB (1)
4	2	2	M	1	3	2	2	2	1C1/1C2	2M1	1C3	$B\bar{A}$ (1)
5	2	2	M	3	3	3	3	2	-	2M3	2M1	BB (2)
6	2	3	C	1	1	3	2	2	2M1	3C0	2M3	BA (2)(SWITCH=1)
7	4	3	C	1	2	1	3	4	2 (modifié)	3C1	-	SWITCH $\rightarrow$ 0
8	4	3	C	2	2	2	3	4	-	3C2	3C1	BB (3)
9	4	3	C	3	2	2	3	4	3C1	3C3	3C2	BB (3)
10	4	4	A	1	3	2	2	4	3C1/3C2	4A1	3C3	$B\bar{A}$ (3)
11	4	4	A	2	3	3	3	4	-	4A2	4A1	BB (4)
12	4	1	M	3	2	3	1	4	4A1	1M3	4A2	déclassement
13	4	6	M	2	1	3	2	4	4A1	6M2	4A2	BA (4) (SWITCH=1)
14	5	6	M	2	1	1	3	5	4 (modifié)	6M2	-	SWITCH $\rightarrow$ 0
15	8	6	M	2	2	1	3	8	5	6M2	-	$\bar{B}A$ (5)
16	8	6	M	3	2	2	3	8	-	6M3	6M2	BB (6)
17	8	7	A	3	2	2	2	8	6M2	7A3	6M3	$B\bar{A}$ (6)
18	8	9	C	1	1	2	2	8	-	9C1	7A3	$B\bar{A}$ (7)
19	10	9	C	1	2	1	3	10	8	9C1	-	$\bar{B}A$ (8)
20	10	9	C	2	2	2	3	10	-	9C2	9C1	BB (9)
21	10	9	C	3	2	2	3	10	9C1	9C3	9C2	BB (9)
22	10	9	M	1	2	2	3	10	9C1/9C2	9M1	9C2	BB (9)
23	10	10	M	1	3	2	2	10	9C1/9C2/9C3	10M1	9M1	$B\bar{A}$ (9)

Numéro de phase	Mouvements des articles dans les zones d'E/S.
1	mettre à blanc la zone WORKA.
2	mettre dans WORKA le contenu de WORKB.
3	mettre dans WORKA le contenu de WORKB.
4	mettre dans WORKA le contenu de WORKB et transférer WORKA dans WORKC après contrôle.
5	mettre WORKB dans WORKA.
6	mettre WORKB dans WORKA. Modification de Z_L(A) avec WORKA - résultat dans Z_L (A).
7	transfert de WORKA dans WORKC.
8	mettre WORKB dans WORKA.
9	mettre WORKB dans WORKA.
10	mettre dans WORKA le contenu de WORKB et transférer WORKA dans WORKC après contrôle.
11	mettre WORKB dans WORKA.
12	mettre Z_L(B) dans WORKC et WORKB dans Z_L(B).
13	mettre WORKB dans WORKA. Annuler dans Z_L(A) les zones renseignées dans WORKA et mettre le résultat dans Z_L(A).
14	transfert de WORKA dans WORKC.
15	transfert de WORKA dans WORKC.
16	mettre WORKB dans WORKA.
17	mettre WORKB dans WORKA et transférer WORKA dans WORKD avec un message d'erreur (AIG $\longrightarrow$1).
18	idem
19	mettre WORKA dans WORKC.
20	mettre WORKB dans WORKA.
21	mettre WORKB dans WORKA.
22	mettre WORKB dans WORKA. Toutes les zones sont présentes; un nouvel article est créé dans WORKA.
23	modifier avec WORKB une zone de WORKA et transférer WORKA dans WORKC.

CHAPITRE III

GENERATION DU BLOC COMPARAISON.

III.1. Table d'indicatif.

Jusqu'à présent nous ne nous sommes pas intéressés à la manière dont on comparait les indicatifs. On a juste dit que les paramètres de cinématique contenaient le résultat de la comparaison des indicatifs.

Or, il se fait qu'en général cette comparaison ne porte pas nécessairement sur un seul indicatif et que de plus un indicatif peut être éclaté en plusieurs zones dans l'enregistrement.
Il nous est apparu intéressant de demander à l'analyste lors de l'élaboration du problème de regrouper toutes les informations concernant les indicatifs pour chaque fichier. La table T.III. 1 est un projet de table d'indicatif.

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

Nom fichier		N° indicatif	adresse relative indicatif	long. zone origine	TYPE ORIG	TYPE DEFI	long. zone définitive
A		1	1	4			
			20	1			
		2	10	4			
B		1	1	4			
			10	2	D	P	1
		2	20	2			
			24	2			

T. III. 1

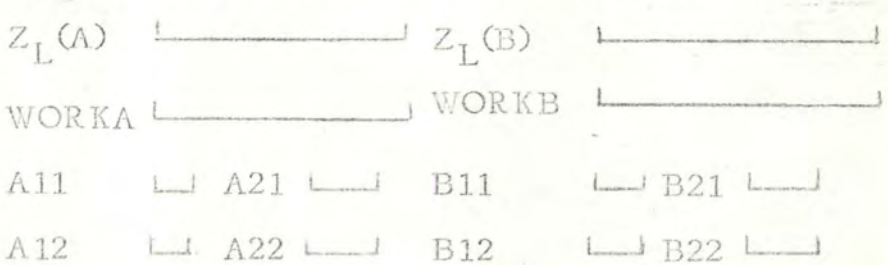
Cette table comprend :

- le nom du fichier (col. 4 à 11).
- la description des indicatifs du fichier par ordre de priorité (col. 15 à 27).
  - col. 15 - numéro de priorité de l'indicatif.
  - col. 16 à 19 : adresse relative par rapport au début de l'enregistrement d'une zone où se trouve une partie ou la totalité de l'indicatif.
  - col. 20 à 22 : longueur de la zone de l'enregistrement
  - col. 23 : type de cette zone (dépacké, packé).
  - col. 24 : type dans lequel on doit convertir la zone.
  - col. 25 à 27 : longueur de la zone dans laquelle on doit mettre le résultat de la conversion.

REMARQUE :

Il peut se faire que l'on n'ait pas besoin de convertir la zone d'indicatif. Dans ce cas on laissera à blanc les colonnes 23 à 27.

Pour chaque fichier d'entrée, nous avons pris l'option de réserver par indicatif, 2 zones correspondant à la longueur de celui-ci de fa à pouvoir faire aisément les comparaisons.
La longueur de ces zones est calculée en faisant la somme des longueurs des zones origines(dans le cas ou il n'y a pas de conversion de type) ou des zones après conversion (col. 25 à 27).
Pour l'exemple repris dans la table, nous aurons :



A 21 donne le nom de la 2e zone du 1er indicatif.

Pour l'exemple de la Table T.III-1 :

A11 et A21 auront comme longueur 5 (4+1)

A12 et A22 auront comme longueur 4.

III. 2. Génération des zones d'E/S.

Nous avons vu précédemment que nous avions besoin de deux zones d'une longueur égale à celle des enregistrements du fichier (Z_L et WORK). Cela provient de ce que nous ne détectons les cas d'opérations qu'à la rupture c'est-à-dire après avoir lu l'article suivant dans le fichier.

Les macros logiques de lecture et d'écriture existant dans le software de l'ordinateur sont de deux types.

- lecture dans (écriture à partir de) un tampon d'un enregistrement du fichier.

  ({ GET / PUT } nom du tampon)

- lecture dans (écriture à partir de) un tampon avec transfert automatique de l'article dans (à partir de) une zone de travail.

  ({ GET / PUT } nom du tampon, nom zone de travail)

Définissons maintenant pour chaque type de fichier laquelle de ces macros de lect./écriture est utilisée et quelles sont les zones Z_L et WORK.
Convenons d'appeler IOAREA le nom du tampon d'E/S.

a. Les fichiers "carte".

Nous emploierons pour ces fichiers le 1er type de macros d'E/S avec comme IOAEREA la zone Z_L.

Une lecture comprendra donc :

1. le transfert de Z_L dans WORK
2. GET Z_L .

b. Les fichiers "IMPRIMANTE".

Pour ces fichiers seront utilisé le 2d type de macros d'E/S avec comme IOAREA la zone Z_L et comme zone de travail la zone WORK.

Une écriture s'exprimera par PUT ZL, WORK

c. Les fichiers "BANDE" en lecture.

Dans ce cas deux options sont possibles :

- simple buffering :

  On utilise le 2d type de macros d'E/S avec comme zone de travail la zone Z_L.

  1. transfert de Z_L dans WORK
  2. GET IOAREA, ZL.

- double buffering :

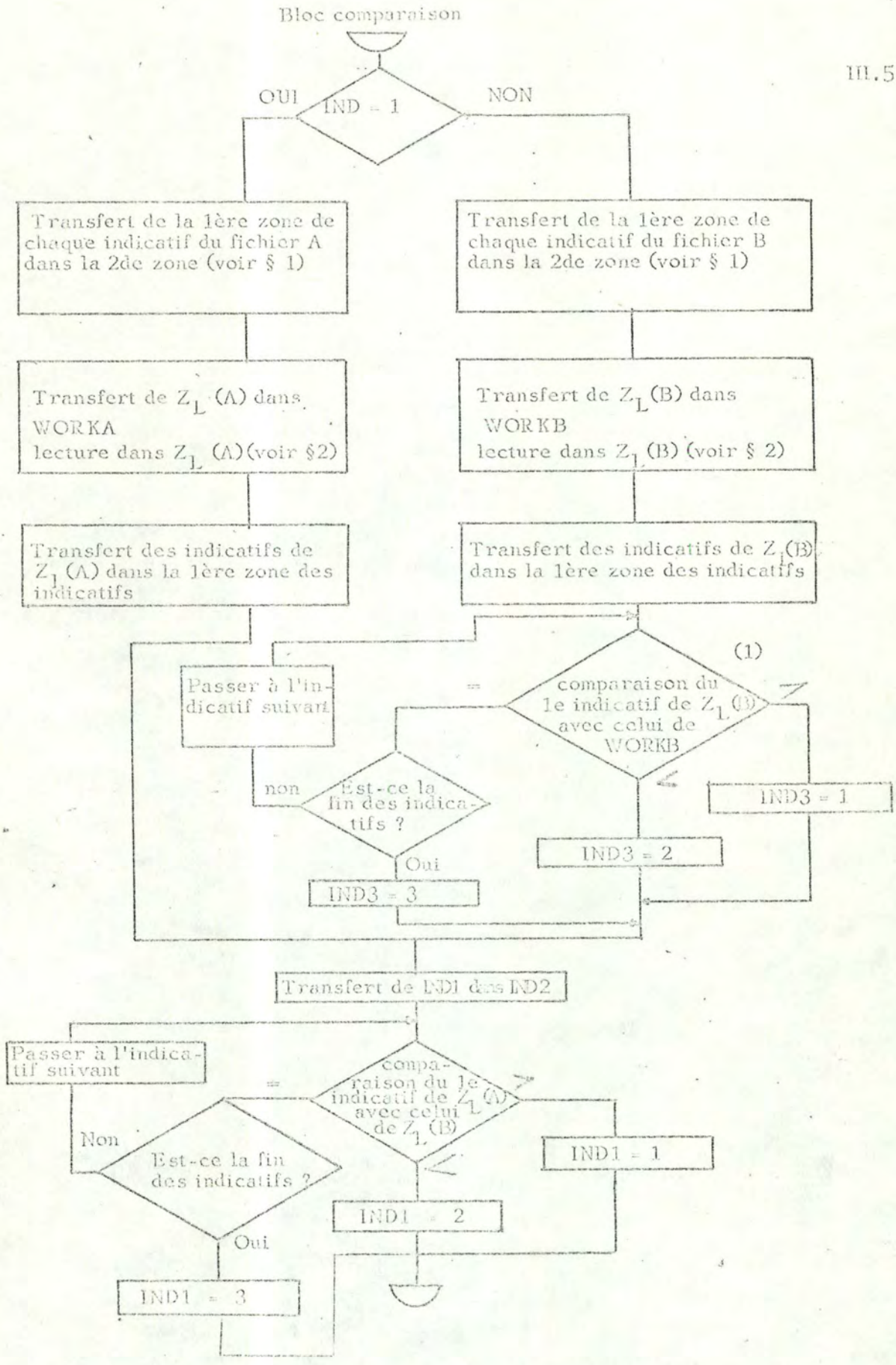
  même chose que le simple buffering mais avec l'emploi de deux IOAREA en bascule.

III. 3. Bloc comparaison.

Ce bloc
- regarde lequel des fichiers et à lire,
- exécute le transfert de la zone d'entrée Z_L avant l'opération de lecture,
- lance l'opération de lecture,
- regroupe, converti et transfert dans les zones réservées les indicatifs.
- compare ces derniers et garni les paramètres de cinématique.

Les comparaisons (1) de l'organigramme se font entre les zones des indicatifs : la 1er zone du i^e indicatif contient le i^e indicatif de Z_L tandis que la 2d zone du i^e indicatif contient le i^e indicatif de WORK etc....

Une procédure spéciale d'initialisation doit être prévue lors de la 1ère lecture.


Bloc comparaison
OUI
IND = 1
NON
Transfert de la 1ère zone de chaque indicatif du fichier A dans la 2de zone (voir § 1)
Transfert de la 1ère zone de chaque indicatif du fichier B dans la 2de zone (voir § 1)
Transfert de Z_L (A) dans WORKA lecture dans Z_L (A) (voir §2)
Transfert de Z_L (B) dans WORKB lecture dans Z_1 (B) (voir § 2)
Transfert des indicatifs de Z_1 (A) dans la 1ère zone des indicatifs
Transfert des indicatifs de Z_1 (B) dans la 1ère zone des indicatifs
(1)
comparaison du 1e indicatif de Z_1 (B) avec celui de WORKB
=
Passer à l'indicatif suivant
non
Est-ce la fin des indicatifs ?
Oui
IND3 = 1
IND3 = 2
IND3 = 3
Transfert de IND1 dans IND2
Passer à l'indicatif suivant
=
comparaison du 1e indicatif de Z_1 (A) avec celui de Z_1 (B)
Non
Est-ce la fin des indicatifs ?
Oui
IND1 = 1
IND1 = 2
IND1 = 3

CHAPITRE IV

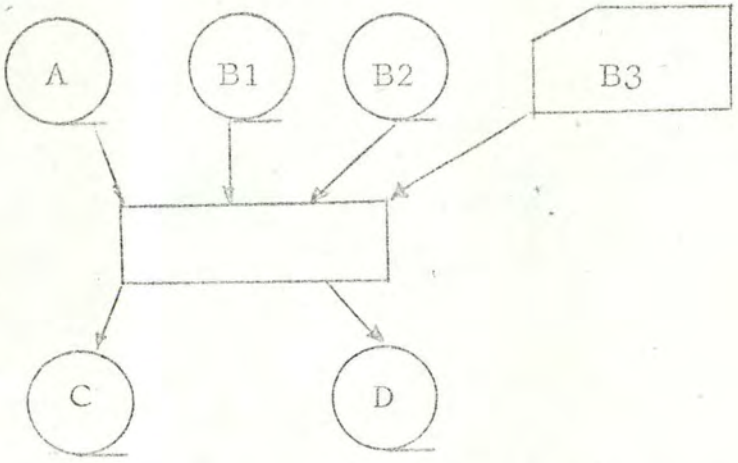
PLUSIEURS FICHIERS SECONDAIRES.

Jusqu'à présent nous n'avons considéré que des problèmes pour lesquels nous avions un seul fichier primaire et un seul fichier secondaire. Il arrivera souvent dans la pratique d'avoir plus de deux fichiers en entrée.

Considérons l'exemple suivant dans lequel nous avons 4 fichiers en entrée (un maître A et 3 esclaves B1, B2, B3) et 2 fichiers en sortie.

Les comparaisons des indicatifs se feront cette fois entre le fichier maître et chacun des fichiers secondaires. Il y aura donc 3 paramètres de cinématique (IND1 J, IND2 J, IND3 J) par fichier secondaire J.

Ainsi par exemple, IND2 3 contiendra le résultat de la comparaison des indicatifs du fichier maître A et du fichier esclave B3 à l'instant précédent.



La progression des fichiers se fera de la manière suivante :

lecture de A si $IND11 = IND12 = IND13 = \ldots\ldots = 1$

lecture de B_i si $\min_{j \in B} IND1j = IND1i$ avec $B = \{ j \,/\, IND1j \neq 1 \}$

et B_i = le $i^{è}$ fichier secondaire

Considérons maintenant le jeu d'essai du tableau T.IV. 1.
Les cas d'opérations sont détectés à chaque phase uniquement sur le fichier qui progresse sauf dans le cas de lecture du fichier maître, ou l'on analyse l'ensemble des paramètres de cinématique.

L'analyse des cas d'opérations nous permet de conclure que pour l'indicatif :

111 nous avons A B1 $\overline{B2}$ $\overline{B3}$

112 nous avons A B1 B2 B3

121 nous avons A $\overline{B1}$ $\overline{B2}$ B3

122 nous avons A $\overline{B1}$ B2 $\overline{B3}$

123 nous avons $\overline{A}$ B1 B2 $\overline{B3}$

211 nous avons B1 B1

REMARQUES :

1. En fait, le cas $\overline{A}\overline{B}_3$ de l'indicatif 1 2 3 n'a évidemment pas été détecté mais est très facile à déduire.

2. Nous voyons que pour retenir le cas AB_j un SWITCH par fichier secondaire est nécessaire.

3. Une procédure de fin de fichier est à prévoir pour les derniers indicatifs.

Phase d'exécution	fichier A			fichier B1			fichier B2			fichier B3			AB1			AB2			AB3			Cas d'opérations		
	I_1	I_2	I_3	I_1	I_2	I_3	I_1	I_2	I_3	I_1	I_2	I_3	IND11	IND21	IND31	IND12	IND22	IND32	IND13	IND23	IND33			
1	1	1	1	1	1	1	1	1	2	1	1	2	3	-	-	1	-	-	1	-	-			
2	1	1	1	1	1	2	1	1	2	1	1	2	1	3	2	1	1	3	1	1	3	AB1 (111) SWITCH1 → 1		
3	1	1	2	1	1	2	1	1	2	1	1	2	3	1	3	3	1	3	3	1	3	SWITCH1 → 0	$A\bar{B}2(111)$	$A\bar{B}_3$ (111)
4	1	1	2	1	2	3	1	1	2	1	1	2	1	3	2	3	3	3	3	3	3	AB1 (112) SWITCH1 → 1		
5	1	1	2	1	2	3	1	2	2	1	1	2	1	1	3	1	3	2	3	3	3		AB2 (112) SWITCH2 → 1	
6	1	1	2	1	2	3	1	2	2	1	2	1	1	1	3	1	1	3	1	3	2			AB3 (112) SWITCH3 → 1
7	1	2	1	1	2	3	1	2	2	1	2	1	1	1	3	1	1	3	3	1	3	SWITCH1 → 0	SWITCH2 → 0	SWITCH3 → 0
8	1	2	1	1	2	3	1	2	2	2	1	1	1	1	3	1	1	3	1	3	2			AB3 (121) SWITCH3 → 1
9	1	2	2	1	2	3	1	2	2	2	1	1	1	1	3	3	1	3	1	1	3	$A\bar{B}1$ (121)	$A\bar{B}2$ (121)	SWITCH3 → 0
10	1	2	2	1	2	3	1	2	3	2	1	1	1	1	3	1	3	2	1	1	3		AB2 (122) SWITCH2 → 1	
11	2	1	1	1	2	3	1	2	3	2	1	1	2	1	3	2	1	3	3	1	3	$A\bar{B}1$ (122)	SWITCH2 → 0	$A\bar{B}3$ (122)
12	2	1	1	2	1	1	1	2	3	2	1	1	3	2	2	2	2	3	3	3	3	$\bar{A}B1$ (123)		
13	2	1	1	2	1	1	2	1	1	2	1	1	3	3	3	3	2	2	3	3	3		$\bar{A}B2$ (123)	
14	2	1	1	2	1	1	2	1	1	2	1	1	3	3	3	3	3	3	3	3	3	B_1B_1 (211)		

T. IV. 1.

CONCLUSION.

Nous avons vu dans ce qui précède un moyen de générer automatiquement une cinématique des fichiers dont les particularités sont :

a. 1 fichier maître A et 1 ou plusieurs fichiers mouvements B_j.

b. le fichier maître A est trié et supposé sans déclassement.

c. les fichiers B_j sont normalement triés et les anomalies à ce sujet sont détectées.

d. les fichiers A et B_j peuvent posséder plusieurs indicatifs par enregistrement.

e. les fichiers B_j peuvent contenir plusieurs enregistrements ayant les mêmes valeurs d'indicatif (1).

Trois paramètres seulement IND1j, IND2j et IND3j par fichier secondaire Bj permettent le traitement automatique de cette cinématique.
Nous n'avons pas considéré le paramètre IND4 = (I^N_A , I^A_A) .
Celui-ci devra être introduit dans le cas ou le fichier maître A possède plusieurs indicatifs par enregistrement. De plus, ce paramètre permettra de dépister les déclassements dans A. Il ne sera toutefois testé que pour les cas ou IND1j sera égale à 1 pour tout j car c'est seulement dans ce cas que l'on fera progresser A.

Certains problèmes restent encore à examiner de plus près.

Il s'agit de :

a. la première et la dernière phase d'exécution.

b. la génération automatique des procédures de traitement des opérations.

Le cas a. pourrait être résolu dans la majeure partie des cas en mettant au fur et à mesure de l'exécution à une valeur maximum les indicatifs des fichiers pour lesquelles on détecte la "fin de fichier" et en s'arrêtant à la phase suivant celle pour laquelle on a détecté la "fin de fichier" pour tous les fichiers d'entrée.

Le cas b. pourrait se résoudre grâce à un certain nombre d'informations contenu dans des tables de description de fichier et d'opérations à exécuter.

BIBLIOGRAPHIE

- IBM 360, générateur automatique de programmes - n° 103065-3.

- CSC, coyent II DOS user's manuel, volume 1, computer sciences. corporation (applications systems decision), 1968-1969.

- ARIANE , Générateur de programmes COBOL, GAMMA (groupe Bossard) 1971.