UNIVERSITE DE NANCY I

U. E. R. DE MATHEMATIQUES

traduction
des tables de décision

thèse

pour l'obtention du
doctorat de spécialité
informatique (3e cycle)

soutenue le 10 mars 1973

par

bernard aubry

jury

M. C.	PAIR	Président
M. M.	DEPAIX	Examinateurs
M. J. C.	DERNIAME	

UNIVERSITE DE NANCY I

U. E. R. DE MATHEMATIQUES

traduction des tables de décision

thèse

pour l'obtention du
doctorat de spécialité
informatique (3e cycle)

soutenue le 10 mars 1973

par

bernard aubry

jury

M. C.	PAIR	Président
M. M.	DEPAIX	Examinateurs
M. J. C.	DERNIAME	

Je tiens à exprimer toute ma reconnaissance à Monsieur le Professeur PAIR pour l'aide bienveillante et la formation qu'il m'a donnée.

Je remercie vivement Monsieur le Professeur DEPAIX pour l'honneur qu'il me fait en participant au jury.

Ce travail a été réalisé sous la direction de Monsieur DERNIAME à qui j'exprime ma profonde reconnaissance pour les conseils et les encouragements amicaux qu'il m'a prodigués tout au long de cette étude.

Je remercie vivement les membres du groupe "CIVA" pour leur participation à la définition de ce travail et pour leurs conseils amicaux.

Je remercie également le personnel de l'IUCA ainsi que Mademoiselle LANG et Madame DUCLOY qui m'ont apporté leur aide pour la réalisation matérielle de cette thèse.

Je voudrais enfin exprimer toute ma gratitude à mon épouse qui, en plus d'un soutien moral de tous les instants, a très activement participé à la réalisation matérielle de ce travail.

S O M M A I R E

CHAPITRE I

INTRODUCTION

1. 'CIVA'

L'utilisation de l'ordinateur nécessite une formulation précise et exhaustive des problèmes à résoudre.

Cette contrainte devient fondamentale pour les applications de gestion pour lesquelles la délimitation précise des objectifs et des méthodes est souvent très délicate.

L'analyse des problèmes est en général confiée à un personnel spécialisé qui, en collaboration avec les responsables des services intéressés, definit une chaine de traitement formée d'une suite de programmes, chaque programme étant chargé d'une tâche bien définie.

Les instructions fournies par l'analyste doivent être très claires, exhaustives et non contradictoires de façon à rendre possible le travail des programmeurs qui doivent les coder en un langage formel, totalement non ambigu, qui sera utilisable par la machine.

D'autre part, ces instructions fournies par l'analyse doivent être compréhensibles par les non spécialistes de l'informatique appartenant aux services demandeurs, elles doivent donc être exprimées dans un langage assez proche de la langue naturelle.

Ces objectifs nécessitent donc un langage normalisé et formalisé, la langue naturelle s'avérant très insuffisante sur ce dernier point.

Jusqu'à présent, le langage le plus couramment utilisé est celui des organigrammes et ordinogrammes, complété par des explications plus ou moins précises en langue naturelle ou par l'utilisation partielle des expressions du langage de programmation.

Cette technique entraine deux écueils de nature opposée bien que non incompatible lors de l'analyse d'une chaine.

. On peut obtenir une description floue incomplète, éventuellement contradictoire ou ambigue, souvent difficile à comprendre et sujette à interprétation.

. Au contraire, on peut obtenir une description trop détaillée, conduisant à un organigramme enchevétré avec un très grand nombre de figurines représentant des opérations élémentaires liées aux particularités du langage de programmation ou à la configuration du hardware.

Un tel ordinagramme peut devenir quasiment incompréhensible pour toutes les personnes autres que l'auteur et nécessiter un véritable décryptage de la part du programmeur à qui il est destiné. Il est alors inutilisable par les services demandeurs qui ne peuvent exercer aucun controle à ce niveau.

D'autre part, la nature même des organigrammes implique plus que la description logique d'un processus ; elle constitue une solution partielle au problème par la structure même qu'elle met en évidence.

Ceci entraine, en plus des inconvénients mentionnés ci-dessus, une grosse difficulté dans la maintenance des programmes.

En effet, toute modification peut entrainer des variations de l'algorithme et nécessiter une nouvelle analyse de l'ensemble du programme de façon à détecter les répercussions éventuelles d'une modification sur l'ensemble du réseau que représente l'organigramme.

Or, la possibilité de modification et d'évolution aisée des chaines est primordiale et même vitale pour les entreprises si l'on veut éviter que l'informatique n'apparaisse comme un élément bloquant et sclérosant de leur gestion.

On peut enfin reprocher à cette méthode de description de l'analyse de n'apporter aucun outil d'aide à la conception, aucun élément de méthodologie permettant une analyse plus facile et plus efficace.

Le projet 'Civa' se propose d'être un moyen d'expression global pour le travail informatique, permettant d'éviter ainsi les coupures 'analyse-programmation' et 'programmation-mise en exploitation' mais surtout de favoriser les travaux de maintenance.

Comme nouveau moyen d'expression de l'analyse, 'Civa' se propose de favoriser, dès le début des travaux, la description précise et formelle des tâches de façon à ne rien laisser à l'interprétation sémantique. Ceci permettra d'autre part une intervention plus précoce de la machine qui pourra directement prendre en charge certains résultats de l'analyse et assurer certaines tâches de routine (regroupement, entrées-sorties, etc)

Le mode d'expression 'Civa' devra être utilisé à tous les niveaux du processus d'étude et de réalisation d'un projet. Or l'organisation des informations et des opérations se fait progressivement depuis la phase de définition du projet jusqu'à la mise en exploitation ; il est donc souhaitable de pouvoir définir des 'objets' (opérations ou informations) qui seront progressivement décrits.

Ceci nous conduit à l'idée de modularité de l'analyse. La définition de module permettra de retarder au maximum la phase de description détaillée (proche de la machine) et elle favorisera la décomposition du problème en opérations élémentaires. La division se fera horizontalement (chaque module étant chargé d'une opération de nature particulière) et verticalement (chaque module pouvant utiliser pour sa description des modules de niveau inférieur, plus détaillés).

La modularité entrainera l'utilisation de tests de débranchement et de conditionnement nécessaires à la reconstitution des processus complexes traités. Là encore, il faudra s'efforcer d'éviter les contraintes 'technologiques' pour l'écriture des conditionnements tout en autorisant une bonne efficacité des programmes (En analyse classique, il s'agit surtout d'éviter les tests inutiles).

Dans ce cadre, les tables de décision fournissent un moyen synthétique d'expression des conditionnements complexes sans que l'auteur ait à définir la forme et l'ordre des tests élémentaires.

'Civa' fera largement appel aux tables de décision et le but de œ travail sera principalement de lui fournir un traducteur.

CHAPITRE II

LES TABLES DE DECISION

1. Définition.-

Concues à l'origine comme un moyen d'expression clair des conditionnements complexes régissant des actions à entreprendre (cf. communication of the A.C.M., Juin-Nov. 1961) et utilisées pour la communication entre analystes et programmeurs, le champ d'application des tablesde décision est maintenant beaucoup plus vaste.

D'une part, il s'agit d'un outil d'analyse qui fournit aux conditionnements une expression simple et indépendante des contraintes de langage ou de handware. L'utilisation des tables de décision permet un contrôle de la non-contradiction des règles se rapportant à un même problème ainsi qu'une vérification de l'exhaustivité de cet ensemble de règles. Cette utilisation est bien adaptée à une analyse modulaire très favorable à la facilité de programmation et à l'exmploitation des travaux.

D'autre part, le développement de plusieurs traducteurs de tables de décision vers divers langage de programmation (cf. Bibl. 23 - Bibl. 24) permet maintenant leur utilisation en tant qu'outil de programmation.

Dans "Civa", tel qu'il a été défini (cf. Réf. 25), les diverses possibilités des tables de décision sont utilisées aux différents niveaux, de l'analyse à l'exploitation.

Précisons cette notion.

Définition.

Une table de décision est une représentation synthétique d'un ensemble de règles, chacune d'elles exprimant les conditions suffisantes pour la réalisation d'un groupe d'actions.

EXEMPLE de table de décision

1 - Un représentant de commerce doit visiter un client en ville et il désire savoir s'il doit utiliser sa voiture ou aller à pied.

Sa décision sera fonction du temps qu'il fait, de la distance à parcourir et des difficultés de stationnement.

En adoptant les notations suivantes :

V - vrai

F - faux

X - action à entreprendre

La table suivante représentera ses règles de comportement :

Stationnement très difficile			V	F			
Distance < 500 m	V	V	F	F	F		
Distance > 1000 m			F	F	F	V	V
Il pleut	V	F	F	F	V	F	V
Prendre imperméable	X				X		X
Prendre voiture				X	X	X	X
Aller à pied	X	X	X				

Table II.1

2 - Utilisation des tables de décision pour l'automatisation des études de fabrication (cf. réf. 20)

Il s'agit d'apporter une aide au bureau d'études d'une entreprise fabriquant des appareils de mesure de grandeurs électriques.

Chaque type d'appareils à fabriquer est décomposé suivant un certain nombre de fonctions bien définies.

A chacune de ces fonctions sont associées des règles de contruction et d'utilisation qui permettent la détermination du produit à partir des spécifications du client.

Soit, par exemple, une table relative à la détermination de l'enroulement de la bobine d'un galvanomètre :

ble II.2

Dimension de l'échelle	4	4	4	8
Unité à mesurer	MV	MV	MA	MA
Plage de mesure	10-75	76-200	10-200	75-250
Nbre de tours enroulement	24	45	24	50
Nature du fil	Cu	Cu	Cu	Al
Diamètre du fil	16	8	16	16
Numéro de schéma	A1234	B5678	F9012	K7821
Description par table	10	17	10	7

2. DESCRIPTION DES TABLES DE DECISION

2.1. Représentation des tables

Les deux exemples précédents sont conformes à la représentation généralement admise, qui peut être schématisée par la figure suivante :

	SOUCHES	ENTREES
CONDITIONS	Souches des conditions	Entrées des conditions
ACTIONS	Souches des actions	Entrées des actions

A) la partie "souches des conditions"est l'énumération de toutes les conditions dont dépendent les décisions à prendre. Chaque condition occupe une ligne et toutes les conditions à traiter y sont décrites.

B) la partie "souches des actions" est l'énumération de toutes les actions qu'il est possible d'entreprendre dans le cadre du problème.

C) la partie "entrées des conditions" représente les diverses combinaisons de conditions pour lesquelles une décision d'action sera prise.

D) la partie "entrées des actions" représente les combinaisons d'actions qui sont susceptibles d'être exécutées.

E) la partie "entrées" des tables de décision est divisée en colonnes dans chacune desquelles figurent :

. en partie condition, une "valeur" de l'ensemble des actions.

. en partie action, la définition de l'ensemble des actions à entreprendre en cas de vérification de l'ensemble de conditions défini en partie condition.

Chacune de ces colonnes définit une REGLE de la table de décision.

Ces définitions nous ont précisé la structure générale des tables de décision, leur aspect syntaxique.

Il nous reste à décrire leur sémantique.

2.2. Lecture des tables de décision.

a) exemple :

Soit à lire la table II.2

Nous l'interpreterons de la manière suivante :

- si la dimension de l'échelle est 4 ET
- si l'unité à mesurer est le millivolt (MV) ET
- si la plage de mesure est 10-75

ALORS

- l'enroulement aura 24 tours ET
- le fil sera en cuivre ET
- son diamètre sera 16/10 mm ET
- le numéro de schéma sera A1234
- la description est poursuivie à la table 10

Lecture règle 1

OU

- si la dimension de l'échelle est 4 ET
- si l'unité à mesurer est le millivolt ET

.....

ALORS

- l'enroulement aura 45 tours ET

.....

Lecture règle 2

OU

.....

règle 3

OU

.....

règle 4

Cet exemple illustre les deux principes généralement admis de la sémantique des tables de décision décrites en b et c.

b) - dans une règle, les réalisations de condition sont liées par ET ainsi que les réalisations d'actions

On a le schéma de lecture suivant :

Règle 1

Si condition 1 ET condition 2 ET.....

ALORS

action 1 ET action 2 ET.....

c) - à chaque combinaison possible de condition, il correspond au plus une combinaison d'action (Une réalisation des conditions ne peut vérifier que l'entrée d'une seule règle)

On a le schéma de lecture suivant :

Table 1

On a règle 1 OU règle 2 OU.....
(OU exclusif, car une seule règle est possible pour une entrée donnée)

d) - remarques.

Les définitions précédentes (en général, celles de "communication of the ACM") sont les plus généralement admises, mais elles ne sont pas obligatoires et des variantes sont possibles.

Par exemple :

- table de décision avec présentation horizontale des règles.
- tables avec conditions liées par d'autres opérateurs que ET

  EX. Si condition 1 OU condition 2 OU condition 3
  ALORS action 1 ET action 2

- tables acceptant des règles non exclusives
  (ce type de table sera précisé par la suite en vue de son utilisation et de sa traduction)

3. TYPES DE TABLES DE DECISION.-

3.1. Les tables à entrées étendues

Les définitions précédentes, sans restriction de forme, définissent une forme très générale des tables de décision: les tables à entrées étendues ("extended entry decision table")

- les souches peuvent contenir une définition partielle de la condition ou de l'action.
- les entrées précisent les réalisations parmi celles qui sont possibles.

  (la souche étant imprécise, un très grand nombre de réalisations sont possibles, l'entrée definit celles qui concernent la table considérée).

  EX. la table II.2 est une table à entrées étendues.

3.2 Les tables à entrées limitées.

En apportant quelques restriction sur la partie "entrées" des tables précédentes, on crée un type de table plus maniable, plus facile à formaliser et dont la traduction sera automatisable : les tables de décision à entrées limitées ("limited entry decision table").

Des tables qui sont un cas particulier des tables à entrées étendues sont elles que :

- les couches définissent complètement les conditions et les actions.
- les seules réalisations possibles, figurent dans les entrées, sont oui ou non (vrai ou faux pour les conditions, faire ou ne pas faire pour les actions).

3.3. Problème de mise à jour de fichier.

Enr. [illegible] = enr. ancien	oui	oui	oui	non
Est-ce une création	oui	non	oui	non
Modifier enregistrement ancien	non	oui	non	non
Créer un enregistrement	non	non	oui	non
Anomalie création existant ou modif. non existant	oui	non	non	oui

REMARQUE

Cette notation (oui-non) pour les entrées est peu utilisée et bien qu'aucune normalisation ne soit définie, la méthode de notation est généralement l'une des suivantes

a) pour l'entrée des conditions, on trouve :

- 0	1	notation booléenne	
- N	Ø	NON	OUI
- N	Y	NO	YES
- F	V	FAUX	VRAI

La notation est généralement N, Y pour les publications de langue anglaise.

La notation française n'est pas fixée

Pour la suite de notre exposé, nous adopterons la notation V, F qui évite toutes les confusions possibles entre le 0 zéro et le 0 oui.

b) la notation pour l'entrée des actions est généralement la suivante:

- X l'action sera exécutée
- blanc l'action ne sera pas exécutée.

4. DESCRIPTION ET PROPRIETES DES TABLES A ENTREES LIMITEES.

4.1. Exemple de table de décision

Etude d'un problème d'assurance-vie.

Pour le calcul de prime d'assurance, on se propose les règles suivantes :

I) si le contractant à moins de 30 ans

1 - si il est en excellente santé et n'a jamais été accidenté :

. proposer un contrat de type A

2 - si il est en mauvaise santé et a déjà été accidenté :

. refuser tout contrat

3 - dans les autres cas :

. faire une enquête.

II) si le contractant a plus de 30 ans, même processus, sauf proposition d'un contrat de type B à la place du contrat de type A.

Ces règles peuvent être regroupées dans une table de décision à entrées limitées, avec les correspondances suivantes entres les règles de la table et les règles de décision proposées ci-dessus.

R1 - I1
R2, R3 - I3
R4 - I2
R5 - II1
R6, R7 - II3
R8 - II2

On obtient latable II.4

	R1	R2	R3	R4	R5	R6	R7	R8
Age $\leqslant$ 30 ans	V	V	V	V	F	F	F	F
bonne santé	V	V	F	F	V	V	F	F
déjà accidenté	F	V	F	V	F	V	F	V
Contrat A	X							
Contrat B					X			
Enquête		X	X			X	X	
Refus				X				X

Cet exemple illustrera la définition suivante :

4.2. Tables à entrées limitées complètes

Définition : Dans une table à entrées limitées, les seules entrées possibles étant V ou F (oui ou non), une table comportant n conditions pourra avoir 2^n règles au maximum.

Une table à entrées limitées à n condition et 2^n règles est dite complète.

EX. - la table II.3 est une table complète à 2 conditions et $2^2 = 4$ règles

- la table II.4 est une table complète à 3 conditions et 2^3 règles

Ce type de table est très utile pour vérifier l'exhaustivité d'une analyse (il suffit de compter les règles pour s'assurer que tous les cas possibles ont été envisagés)

Cependant, lorsque le nombre de conditions s'élève, le nombre de règles devient prohibitif alors qu'une grande partie d'entre elles sont superflues pour la logique du problème.

4.3. La règle E

En vue de réduire le nombre des règles à écrire, on regroupe en une seule règle l'ensemble des règles non définies par le problème et dont l'entrée correspond à une réalisation de condition sans intérêt pour lui.

Cette règle est dite "règle E" ("Else rule"). Elle figure à droite des autres règles dans la table de décision, les entrées de ses conditions étant laissées en blanc.

Remarque 1 -

Soit une table à entrée limitée à n conditions et dans laquelle r règles sont expréssement définies par l'auteur.

La règle E représente alors les 2^n-r règles non définies.

Remarque 2 -

La règle E est de nature différente des autres règles de la table.

Alors que les règles normales définissent de façon positive un ensemble d'actions à réaliser si les conditions sont vérifiées, la règle E intervient de façon négative.

Si les valeurs entrées ne vérifient aucune des autres règles alors les actions spécifiées par la règle E sont exécutées.

EXEMPLE.

Soient les décisions suivantes :

Si C1 et NON C2 et C3 et NON C4 et C5
faire A1

Si C1 et C2 et NON C3 et NON C4 et NON C5
faire A2

Si C1 et NON C2 et C3 et C4 et C5
faire A1 et A2

Dans les autres cas, faire AE

AE est l'action définie par la règle E

On obtient latable de décision suivante :

Table II.5

	R1	R2	R3	E
C1	V	V	F	
C2	F	V	F	
C3	V	F	V	
C4	F	F	V	
C5	V	F	V	
A1	X		X	
A2		X	X	
AE				X

- sur cette table, la règle E représente $2^5-3 = 29$ règles
- généralement l'action AE consistera à signaler qu'une entrée non prévue est survenue ; elle peut cependant être tout autre.

EXEMPLE.

Reprenons la table II.4. Nous l'avons définie par les 6 règles de décisions I1, I2, 13, II1, II2, II3. On aurait pu également la définir par les règles I1, I2, II1, II3 et les "autres cas" pour lesquels on décide de faire une enquête.

Ces règles auraient pu être représentées par la table suivante qui comporte explicitement les règles correspondant à I1, I2, II1, II2 ; la règle E regroupant les autres cas.

Table II.6

		R1	R2	R3	R4	E
C1	âge $\leqslant$ 30 ans	V	F	V	F	
C2	bonne santé	V	V	F	F	
C3	déjà accidenté	F	F	V	V	
Act 1	contrat A	X				
Act 2	contrat B		X			
Act 3	enquête					X
Act 4	refus de contrat			X	X	

NOTE.-

Dans de nombreux cas, où l'entrée de la règle E n'entraine pas d'action spécifique et indique seulement une sortie de la table, la règle E sera omise.

En cas de réalisation des conditions correspondant à une entrée de la règle E, l'action entreprise consistera à sortir de la table et à passer en séquence.

4.4. Equivalence.

Deux tables de décision seront dites équivalentes si les souches sont identiques et si, à une réalisation quelconque de l'ensemble des conditions correspond dans chacune des tables la même réalisation de l'ensemble des actions.

EXEMPLE.

Les tables II.4 et II.6 sont équivalentes (Elles sont ici interchangeables et, à ce niveau, différent seulement par le mode d'écriture).

4.5. Entrées indifférentes.

Deux règles d'une table à entrées limitées ne différant que par l'entrée d'une seule condition peuvent être regroupées en une seule règle, l'entrée de la condition concernée étant notée '-'.

Cette condition est dite condition indifférente pour la nouvelle règle.

EXEMPLE.

Soit la table II.4

Les règles R2 et R6
R3 et R7
R4 et R8
peuvent être regroupées.

Nous obtenons la table II.7 équivalente à la table d'origine

Table II.7

C1	V	-	-	F	-
C2	V	V	F	V	F
C3	F	V	F	F	V
Act 1	X				
Act 2				X	
Act 3		X	X		
Act 4					X

4.6 Ensemble d'actions

On applera ensemble d'actions d'une règle l'ensemble ordonné des actions désignées dans la partie entrée des actions de cette règle.

EXEMPLE.

a) soit R3 sur la table II.5
son ensemble d'actions est
(act A1, act A2) dans cet ordre

b) soit R1 sur la table II.6
son ensemble dactions est (act1)

Notation

Dans la suite, les actions seront notées act1, act2, ..., act_i, ... les ensembles d'actions étant notés A1, A2, ..., A_i, ...

Généralement l'indice i de l'ensemble d'action A_i sera celui de la règle (Ri) à laquelle il appartiendra.

EXEMPLE.

A l'aide de cette notation, la table II.7 peut s'écrire :

Table II.8

	R1	R2	R3	R4	R5
C1	V	-	-	F	-
C2	V	V	F	V	F
C3	F	V	F	F	V
	A1	A2	A3	A4	A5

Les ensembles d'actions étant :

A1 = (act1)

A2 = A3 = (act3)

A4 = (act2)

A5 = (act4)

REMARQUE

Cette notation sera également utilisée (sans définition des ensembles d'actions) pour définir, à titre d'exemple, les tables de décision pour lesquelles l'entrée des actions et leur nature est sans importance.

4.7. Simplification d'une table de décision

Il s'agit de rechercher dans l'ensemble des tables équivalentes à une table donnée, des tables de décision dont le nombre de règles explicites et/ou le nombre d'entrées indifférentes est supérieure

Ces tables seront dites "plus simples" que la table d'origine.

EXEMPLE

- II.8 est une simplification de II.4
- II.6 est également une simplification de II.4

REMARQUE

II.8 et II.4 sont équivalentes à II.8. Elles sont donc équivalentes entre elles. Les critères de simplification définis précédemment sont cependant insuffisants pour lier II.8 et II.6 (la relation "plus simple" peut être considérée comme un ordre partiel sur l'ensemble des tables équivalentes)

En appliquant simultanément à la table II.4 les procédés qui ont permis de définir II.6 et II.8, nous obtenons la table II.9 qui est plus simple que chacune des précédentes.

Table II.9

	R1	R2	R3	E
C1	V	F	-	
C2	V	V	F	
C3	F	F	V	
Act 1	X			
Act 2		X		
Act 3				X
Act 4			X	
	A1	A2	A3	A4

4.8 Utilisation de l'algèbre de Boole

Pour la simplification des tables à entrées limitées.

Adoptons les notations booléennes :

+ et × pour OU et ET

$\overline{Ci}$ pour l'entrée Ci = F

Ci pour l'entrée Ci = V

Dans une règle, chaque ensemble d'actions Ai est spécifié par l'entrée des conditions, entrée qui peut être représentée par un monome booléen.

EXEMPLE 1

Soit la règle R2 de la table II.9

L'entrée des actions est représentée par le monome :

$\overline{C1}\ \overline{C2}\ \overline{C3}$ qui spécifie l'ensemble d'action A2

On notera A2 = $\overline{C1}\ C2\ \overline{C3}$

EXEMPLE 2

Simplification de la table II.4

Les huit règles de la table s'écrivent :

R1 - A1 = $C1\ C2\ \overline{C3}$

R2 - A2 = $C1\ C2\ C3$

R3 - A3 = $C1\ \overline{C2}\ \overline{C3}$

R4 - A4 = $C1\ \overline{C2}\ C3$

R5 - A5 = $\overline{C1}\ C2\ C3$

R6 - A6 = $\overline{C1}\ C2\ C3$

R7 - A7 = $\overline{C1}\ \overline{C2}\ \overline{C3}$

R8 - A8 = $\overline{C1}\ \overline{C2}\ C3$

En fait, les ensembles d'actions A2, A3, A6, A7 sont identiques ainsi que A4 et A8.

Donc A4 est spécifiée par C1 $\overline{C2}$ C3 ou $\overline{C1}$ $\overline{C2}$ C3, ce qui s'écrit :

A4 = C1 $\overline{C2}$ C3 + $\overline{C1}$ $\overline{C2}$ C3

De même :

A2 = C1 C2 C3 + C1 $\overline{C2}$ $\overline{C3}$ + $\overline{C1}$ C2 C3 + $\overline{C1}$ $\overline{C2}$ $\overline{C3}$

En appliquant les règles booléennes de simplification,

on obtient :

A4 = $\overline{C2}$ C3(C1+$\overline{C1}$) = $\overline{C2}$ C3

A2 = C2 C3 + $\overline{C2}$ $\overline{C3}$

Donc après simplification, les huit expressions précédentes se réduisent à :

A1 = C1 C2 $\overline{C3}$

A2 = C2 C3 + $\overline{C2}$ $\overline{C3}$

A4 = $\overline{C2}$ C3

A5 = $\overline{C1}$ C2 $\overline{C3}$

Ces expressions définissant la table II.10

C1	V	-	-	-	F
C2	V	V	F	F	V
C3	F	V	F	V	F
	A1	A2	A2	A4	A5

REMARQUE 1 -

L'utilisation de l'algèbre de Boole formalise la simplification des tables de décision et donc la facilite dans une large mesure. Malheureusement cette méthode n'est pas automatisable, ce qui limite son intérêt pratique.

REMARQUE 2

La simplification d'une table est toujours souhaitable, mais elle peut parfois conduire à plusieurs tables équivalentes sans choix possible selon le critère de simplicité défini ici.

EXEMPLE.

Proposons nous la simplification de la table II.11

C1	V	F	F	
C2	F	F	F	
C3	F	F	V	
	A1	A1	A1	A2

On a :

$$A1 = C1\ \overline{C2}\ \overline{C3} + \overline{C1}\ \overline{C2}\ \overline{C3} + \overline{C1}\ \overline{C2}\ C3$$

d'où

$$A1 = \overline{C2}\ \overline{C3} + \overline{C1}\ \overline{C2}\ C3$$

mais aussi :

$$A1 = C1\ \overline{C2}\ \overline{C3} + \overline{C1}\ \overline{C2}$$

donc on obtient deux tables équivalentes à la table II.11

C1	-	F	
C2	F	F	
C3	F	V	
	A1	A1	A2

Table II.12

C1	V	F	
C2	F	F	
C3	F	-	
	A1	A1	A2

Table II.13

Le choix entre ces deux tables devra faire intervenir des considérations sémantiques telles que le coût des tests C1 et C3 et/ou la probabilité de réalisation des règles.

4.9. Ambiguités

Définition : Deux règles d'une table de décision seront dites ambigues si il existe une réalisation de l'ensemble des conditions satisfaisant à l'entrée des conditions de ces deux règles.

L'ambiguité peut avoir plusieurs aspects différents liés à la sémantique de la table.

a) ambiguité apparente ou Redondance

Deux règles Ri et Rj seront dites redondantes si elles sont ambigues et si les ensembles d'actions Ai et Aj sont identiques.

EXEMPLE 1

Proposons nous de définir une fonction à deux variables booléennes et valeurs réelles.

f(x, y) sera défini à l'aide de la table suivante :

Table II.14

	R1	R2	R3
x = 0	V	F	-
y = 0	V	-	F
f = 1	X	-	-
f = 2	-	X	X

La règle R2 représente les deux règles suivantes, en ne remplaçant que l'entrée indifférente de C2 :

F	F
V	F
-	-
X	X

La règle R3 représente les deux règles, en ne remplaçant que l'entrée indifférente de C1 :

F	V
F	F
-	-
X	X

L'ambiguité des règles R2 et R3 est due à cette double définition identique de la règle.

F
F
-
X

Cette ambiguité n'enlève rien à la validité de la table et il sera inutile de la remplacer par une table équivalente non ambigue, mais moins simple, du type de la table II.15.

x = 0	V	F	-
y = 0	V	V	F
f = 1	X	-	-
f = 2	-	X	X

Dans ce cas, la table redondante est plus simple que ses équivalentes non ambigues. L'inverse est également possible comme on peut le voir dans l'exemple 2

Exemple 2

Soit la table II.16

	R1	R2	R3
C1	V	F	F
C2	V	V	-
A	X	-	-
B	-	X	X
	A1	A2	A3

Il est possible de simplifier cette table

$A2 = \overline{C1}\ C2 + \overline{C1} = \overline{C1}$

On obtient la table II.17 équivalente à la table II.16 non ambigue et plus simple.

C1	V	F
C2	V	-
A	X	-
B	-	X

b) ambiguité réelle ou contradiction.

Deux règles Ri et Rj seront dites contradictoires, si elles sont ambigues et si les ensembles d'actions Ai et Aj sont différents.

b.1) ambiguité effective.

EXEMPLE 1

Proposons nous de définir f, fonction de variables booléennes à valeurs réelles, à l'aide de la table de décision suivante :

Table II.18

	R1	R2	R3
x = 0	V	F	-
y = 0	V	-	F
f = 1	X	X	-
f = 2	-	-	X

L'ambiguité nous conduit ici à deux définitions contradictoires de la règle d'entrée des conditions (F,F)

On a $f(0,0) = 1$ R1

$f(1,0) = F(1,1) = 1$ R2

$f(0,1) = f(1,1) = 2$ R3

La contradiction apparait ici au niveau de la définition de f

L'ambiguité de la table correspond ici à une erreur logique de définition lors de sa construction qui doit donc être supprimée.

Dans l'exemple 1, un choix est à faire.

Si, par exemple, on décide de prendre $f(1,1) = 1$
on pourra construire la table II.19 définissant la fonction F

x = 0	V	F	F	V
y = 0	V	V	F	F
f = 1	X	X	X	-
f = 2	-	-	-	X

f est ainsi complètement définie

f(0,0) = 1 f(1,1) = 1
f(1,0) = 1 f(0,1) = 2

La table II.19 peut être simplifiée, on obtiendra la table II.20, table non ambigue définissant complètement la fonction F

Table II.20

x = 0	-	F	V
y = 0	V	F	F
f = 1	X	X	-
f = 2	-	-	X

Dans l'exemple précédent, la contradiction apparait de façon effective entre les règles. Mais ceci n'est pas général et il est possible d'avoir des règles contradictoires selon notre définition et pourtant parfaitement compatibles entre elles dans la table.

b•2) ambiguité formelle

EXEMPLE.

Soit la table II.21

		R1	R2	R3
C1	Age <18	V	-	F
C2	Age >65	-	V	F
		A1	A2	A2

Cette table est dite contradictoire à cause des règles R1 et R2 (cf. définition contradiction)

Cependant, l'entrée (V,V) est impossible de part la nature même des conditions.

La contradiction est donc ici strictement formelle et elle peut donc être admise dans une table.

Il serait en effet possible de supprimer la contradition dans une table équivalente.

Par exemple, la table II.22 non ambigue est équivalente à la table II.21

Table II.22

	R1	R2	R3	E
C1	V	F	F	
C2	F	V	F	
	A1	A2	A2	Err

La règle E correspondant ici à la règle d'entrée (V,V)

En renonçant à détecter l'erreur éventuelle constituée par l'entrée (V,V), on peut obtenir une table équivalente non ambigue plus simple.

Par exemple, la table II.23

	R1	R2	R3
C1	V	-	F
C2	F	V	F
	A1	A2	A2

b.3) suppression des ambiguités formelles.
L'ambiguité formelle est due à une confusion sur le sens de la notation '-' dans l'entrée des conditions.

Par définition, une condition d'entrée notée '-' dans une règle est une condition indifférente pour cette règle et n'a donc pas à être testée.

Dans une table formellement contradictoire (cf. table II.21) l'entrée'-' ne correspond pas obligatoirement à une condition indifférente mais peut simplement indiquer une condition qu'il est inutile de tester, le résultat de ce test étant entrainé par le test d'une autre condition LIEE à la condition considérée.

EXEMPLE.

Considérons la table II.21

Les deux conditions C1 et C2 ne sont évidemment pas indépendantes ($C1 \Rightarrow \overline{C2}$ et $C2 \Rightarrow \overline{C1}$

L'entrée '-' de la condition C2 dans la règle R1 ne signifie pas que C2 est indifférent dans R1 mais simplement qu'il n'y a pas lieu de vérifier C2 = F puisqu'on a vérifié C1 = V

Pour la clarté d'écriture des tables et la facilité de leur traduction, il est utile d'éviter ce genre d'ambiguité purement fictive.

Deux solutions sont alors possibles :

1- suppression de l'ambiguité en recherchant une table équivalente non ambigue.

EXEMPLE.

La table II.23 non ambigue est équivalente à la table II.21

Cette méthode a l'inconvénient de remplacer une table par une table plus complexe (le nombre moyen de test par règle est plus élevé).

2- suppression de l'ambiguité par l'utilisation d'une notation mieux adaptée.

Nous proposons la notation suivante :

+ V en entrée d'une condition indique que :
  - la condition concernée n'a pas à etre testée,
  - sa valeur, résultant du test des autres conditions de la règle, est V

+ F en entrée d'une condition a les mêmes spécifications que + V, la valeur de la condition étant F

EXEMPLE.

En utilisant cette notation, la table II.21 s'écrit :

		R1	R2	R3	E
C1	Age $<$ 18	V	+F	F	
C2	Age $>$ 65	+F	V	F	
		A1	A2	A3	Err

Table II.24

Cette table n'est pas ambigue, elle permet une détection d'erreur et favorise une traduction efficace.

c) ambiguités complémentaires.

EXEMPLE.

Posons nous le problème suivant :
Nous nous proposons de sortir de notre appartement.
Plusieurs comportements sont alors possibles :
si il pleut, prendre un parapluie ;
si il fait froid, prendre un manteau.
Les deux situations sont évidemment simultanément possibles.

La table de décision suivante est une représentation possible de ce comportement.

Table II.25

	R1	R2	R3
Il pleut	V	F	V
Il fait froid	F	V	V
prendre un parapluie	X	-	X
prendre un manteau	-	X	X

Une façon beaucoup plus simple et naturelle de représenter ce comportement est possible pour la table II.26

	R1	R2
il pleut	V	-
il fait froid	-	V
prendre un parapluie	X	-
prendre un manteau	-	X

En décidant ici que si les deux conditions sont vraies (les deux règles ambigues étant alors sélectées), on exécute alors les deux ensembles d'actions (l'ordre étant quelconque).

Cette table de décision n'est pas conforme à la définition générale de la sémantique des tables de décision (p. II.8) qui suppose des règles mutuellement exclusives.

Ce type de table étant très intéressant à cause de la simplicité des tables obtenues et de leur facilité de création, nous proposons d'étendre la définition des tables de décision et d'accepter les règles ambigues lorsque les ensembles d'action sont compatibles.

Cette nouvelle définition implique, au moment de la création de la table, une étude de la nature des ensembles d'actions des règles ambigues pour s'assurer de leur non-contradiction.

EXEMPLE. (cf. Réf. 19)

On se propose d'utiliser une table de décision pour décrire les règles d'attribution de certaines primes, ces règles étant définies ci-dessous.

Attributions de :

- prime d'ancienneté (PA) si (âge $<$ 30 et ancienneté $\geq$ 5)
  si (âge $\geq$ 30 et ancienneté $\geq$ 3)
- prime spéciale (PS) si (non cadre et ancienneté $\geq$ 3)
- intéressement employés (IE)
  si (non cadre et ancienneté $\geq$ 5)

- intéressement cadre (IC) si (cadre et ancienneté $\geq$ 5)
- intéressement spécial (IS) si ancienneté $\geq$ 5
- prime jeune ménage (PJM) si (marié et âge $<$ 30)

L'utilisation des tables de décision sous leur forme classique conduit à la table II.27

Cette table très lourde peut être simplifiée de façon à obtenir la table II.28

Cette forme de table est difficile à construire et semble à peu près inutilisable en pratique dans le cas traité ici.

Table II.27

Age < 30	F	F	F	F	F	F	F
Ancienneté ≥ 3	V	V	V	V	V	V	V
Ancienneté ≥ 5	F	F	F	F	V	V	V
Cadre	F	F	V	V	F	F	V
Marié	F	V	F	V	F	V	F
Prime ancienneté	X	X	X	X	X	X	X
Prime spéciale	X	X	-	-	X	X	-
int. employés	-	-	-	-	X	X	-
int. cadres	-	-	-	-	-	-	X
int. spécial	-	-	-	-	X	X	X
prime jeune ménage	-	-	-	-	-	-	-

F	V	V	V	V	V	V	V	V	V	
V	F	F	V	V	V	V	V	V	V	
V	F	F	F	F	F	V	V	V	V	
V	F	V	F	F	V	F	F	V	V	
V	F	F	F	V	V	F	V	F	V	
X	-	-	-	-	-	X	X	X	X	-
-	-	-	-	-	-	X	X	-	-	-
-	-	-	-	-	-	X	X	-	-	-
X	-	-	-	-	-	-	-	X	X	-
X	-	-	X	X	-	X	X	X	X	-
-	X	X	-	X	X	-	X	-	X	-

Age<30	F	F	F	F	V	V	V	V	V	V	V	V	
Anc.≥3	V	V	+V	+V	F	V	V	V	+V	+V	+V	+V	
Anc≥5	F	F	V	V	F	F	F	F	V	V	V	V	
Cadre	F	V	F	V	-	F	F	V	F	F	V	V	
Marié	-	-	-	-	F	F	V	V	F	V	F	V	
P.A.	X	X	X	X	-	-	-	-	X	X	X	X	-
P.S.	X	-	X	-	-	-	-	-	X	X	-	-	-
I.E.	-	-	X	-	-	-	-	-	X	X	-	-	-
I.S.	-	-	X	X	-	X	X	-	X	X	X	X	-
I.C.	-	-	-	X	-	-	-	-	-	-	X	X	-
P.J.M.	-	-	-	-	X	-	X	X	-	X	-	X	-

Table II.28

L'utilisation d'une table de décision admettant des règles ambigues nous permettra de représenter les règles d'attribution de prime à l'aide de la table II.29

âge < 30	-	F	-	-	-	-	V
anc. ≥ 3	+V	V	V	+V	+V	+V	-
anc. ≥ 5	V	-	-	V	V	V	-
Cadre	-	-	F	F	V	-	-
Marié	-	-	-	-	-	-	V
P.A.	X	X	-	-	-	-	-
P.S.	-	-	X	-	-	-	-
I.E.	-	-	-	X	-	-	-
I.S.	-	-	-	-	-	X	-
I.C.	-	-	-	-	X	-	-
P.J.M.	-	-	-	-	-	-	X

Cette table est beaucoup plus simple que la table II.28

Elle a surtout l'avantage de la facilité de construction, chacune de ses règles se déterminant par une simple codification de l'énoncé en langue naturelle des décisions à prendre selon les cas.

Une simplification est possible en regroupant la première et la sixième règle qui ont des conditionnements identiques .

On obtient alors la table II.30.

âge < 30	-	F	-	-	-	V
anc. ≥ 3	+V	V	V	+V	+V	-
anc. ≥ 5	V	-	-	V	V	-
cadre	-	-	F	F	V	-
marié	-	-	-	-	-	V
P.A.	X	X	-	-	-	-
P.S.	-	-	X	-	-	-
I.E.	-	-	-	X	-	-
I.S.	X	-	-	-	-	-
I.C.	-	-	-	-	X	-
P.J.M.	-	-	-	-	-	X

CHAPITRE III

TRADUCTION DES TABLES

1. GENERALITES.-

La structure des tables de décision a conduit à l'étude d'algorithmes destinés à effectuer la traduction d'une forme normalisée de table de décision en un texte de programme écrit dans un langage admis dans un compilateur.

L'automatisation d'un tel algorithme réduit considérablement le rôle du programmeur qui se limite alors à la recopie, selon un format normalisé, des tables fournies par l'analyste.

D'autre part, l'automatisation permet de prévoir une optimisation du programme résultant sous ses aspects d'encombrement en mémoire ou de rapidité d'exécution.

Plusieurs algorithmes ont été définis pour réaliser la traduction des tables de décision. On se propose, en général, la traduction des tables de décision à entrées limitées, non ambigues, avec différents degrés de perfectionnement en vue de l'optimisation du programme résultant.

Les algorithmes de traduction peuvent se regrouper en plusieurs classes selon la méthode employée pour la traduction.

Nous citerons ici deux de ces méthodes de traduction : la traduction règle par règle et la traduction par analyse globale de la table.

2. TRADUCTION REGLE PAR REGLE.-

Toutes les conditions sont d'abord déterminées ; le résultat de ces tests fournit un "masque" qui est alors comparé à chacune des règles jusqu'à ce que la correspondance ait lieu, ce qui détermine l'ensemble d'actions à exécuter.

EXEMPLE. table III.1

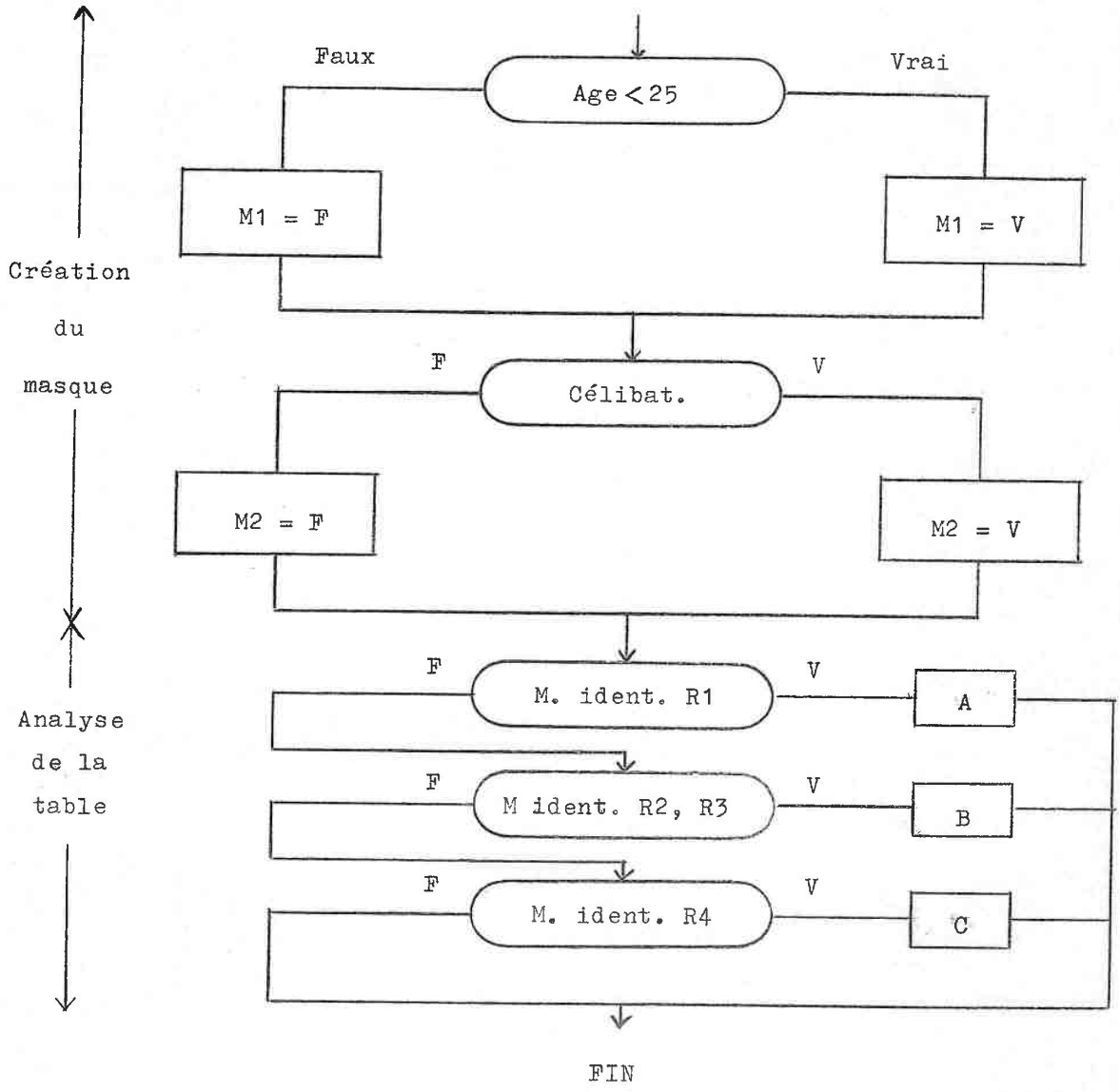
		R1	R2	R3	R4
C1	Age 25	V	V	F	F
C2	Célibataire	V	F	V	F
Tarif A		X	–	–	–
Tarif B		–	X	X	–
Tarif C		–	–	–	X

La traduction comporte deux phases :

a - création d'un masque M regroupant les résultats des tests des conditions (M_i=V si C_i est vraie ; =F sinon)

b - comparaison des entrées de chaque règle avec ce masque jusqu'à la correspondance, ce qui détermine l'action à exécuter.

Le programme équivalent aura la structure suivante (M étant le "masque"):



Cette méthode très simple nous fournit un programme résultant de structure simple mais avec un encombrement important, du fait du nombre élevé de tests, et un temps d'exécution relativement long.

Notons cependant que la durée moyenne d'exécution peut facilement être réduite en modifiant l'ordre des règles avec le masque.

EXEMPLE. Supposons que la règle R4 de la table III.1 est très fréquente.

Il est évident que dans le programme il faudra comparer d'abord R4 et M et seulement ensuite tester les autres règles.

Dans les cas où existent des entrées indifférentes, la méthode peut encore être améliorée en testant certaines règles avec un masque partiel correspondant aux entrées déterminées avant de compléter le masque pour tester les autres règles.

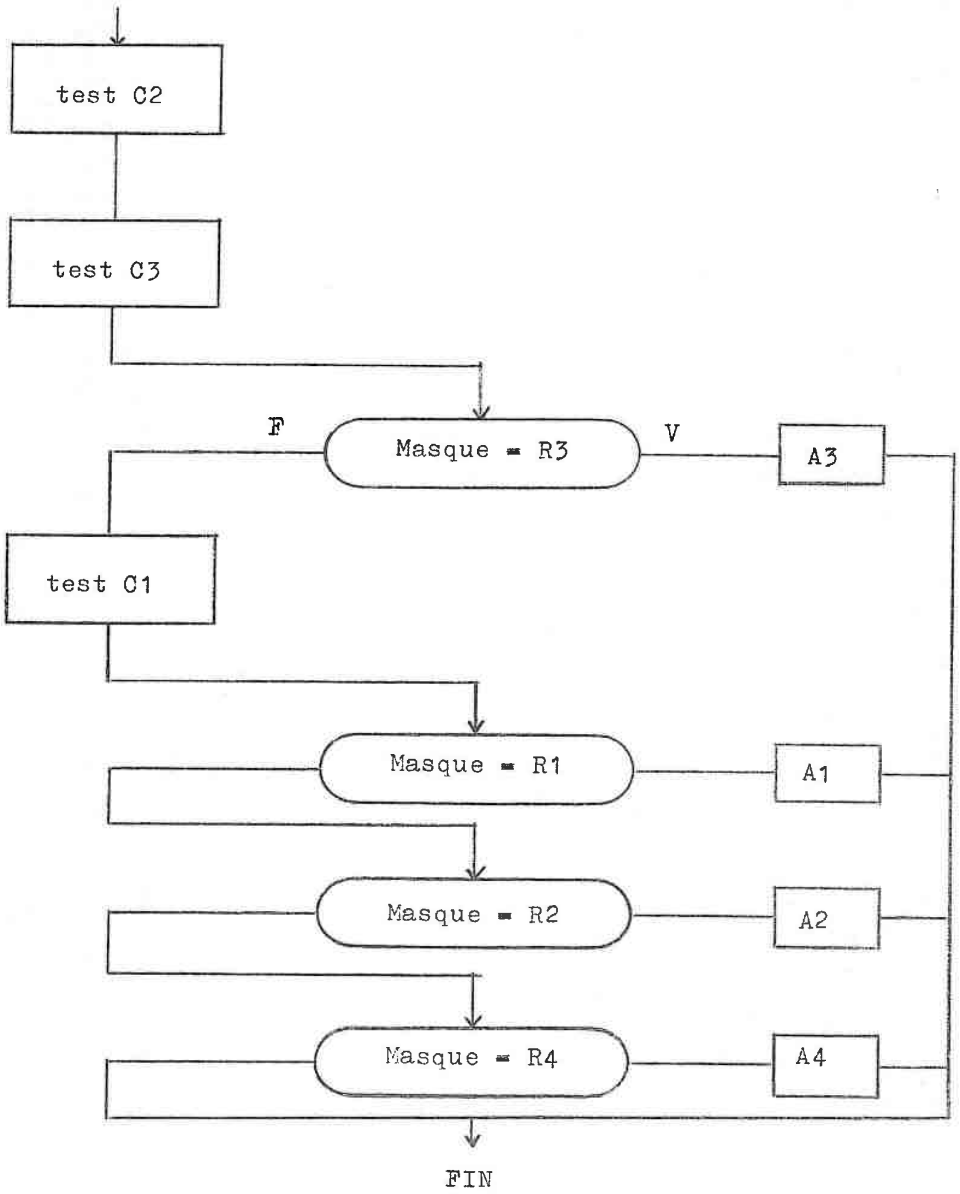
EXEMPLE. Table III.2

Supposons que :

- R3 est la règle la plus fréquente.
- le test de C1 est long

	R1	R2	R3	R4
C1	V	V	-	F
C2	V	-	F	-
C3	F	V	F	-
	A1	A2	A3	A4

Le programme résultant pourra être schématisé ainsi



3. TRADUCTION PAR ANALYSE

L'algorithme est basé sur la propriété suivante des tables de décision à entrées limitées.

La détermination d'une des conditions de la table permet de la diviser en deux sous-tables dans lesquelles la condition considérée ne figure pas. Une de ces tables est obtenue par regroupement des règles dont l'entrée est F ou '-' dans la table d'origine ; l'autre par regroupement des règles dont l'entrée est V ou '-'.

La répétition de ce processus pour toutes les sous-tables successivement obtenues et tant qu'il existe des conditions à tester conduit à une arborescence, représentation logique du programme résultant.

Cette méthode étant celle retenue pour la réalisation de notre traducteur, nous l'analyserons en détail. (La génération d'un programme à partir de l'arborescence obtenue sera traitée dans un chapitre distinct)

EXEMPLE. Soit la table III.3

	R1	R2	R3
C1	F	-	V
C2	F	V	-
C3	-	V	F
C4	-	-	F
	A1	A2	A3

Décidons par exemple de tester C1

L'application de la règle de division entraîne :

III.4 correspondant à l'entrée F de C1

III.5 correspondant à l'entrée V de C1

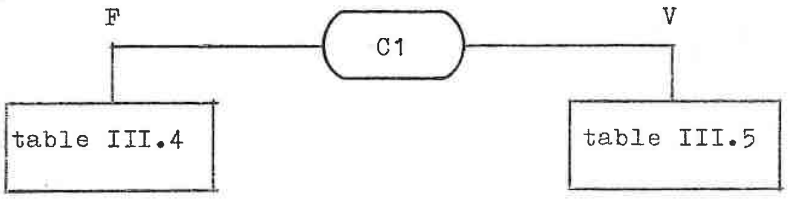
Table III.4

	R1	R2
C2	F	V
C3	-	V
C4	-	-
	A1	A2

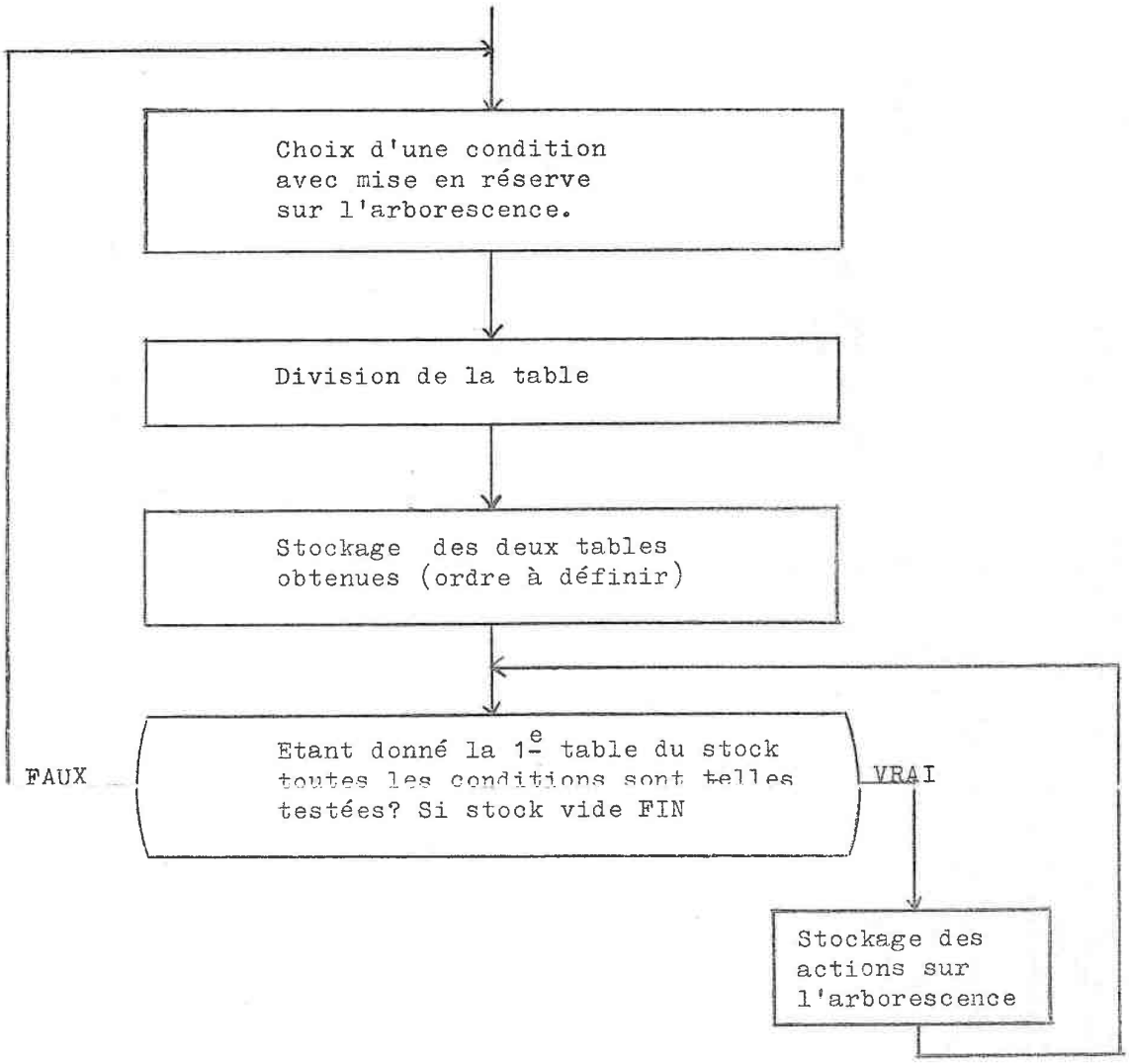
Table III.5

	R2	R3
C2	V	-
C3	V	F
C4	-	F
	A2	A3

Cette décomposition peut se représenter schématiquement ainsi :



L'algorithme de traduction basé sur cette décomposition sera schématiquement représenté ainsi :



3.1. Traduction simple avec stockage sur une pile

EXEMPLE. Table III.6

Proposons nous de traduire cette table, c'est-à-dire construire l'arborescence décriant un programme équivalent

	R1	R2	R3	R4
C1	V	V	F	F
C2	V	F	V	F
	A1	A2	A3	A4

PHASES DE LA TRADUCTION

1ère itération

(traitement de la table III.5)

étape 1 - choix de C1 comme première condition à tester, mise en réserve sur l'arborescence

- C1

étape 2 - découpage en deux tables

III.7 pour les entrées F

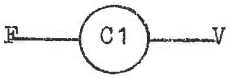
	R3	R4
C2	V	F
	A3	A4

III.8 pour les entrées V

	R1	R2
C2	V	F
	A1	A2

CREATION DE L'ARBORESCENCE

(organigramme + notation préfixée)



étape 3 - stockage de tables dans l'ordre III.7 - III.8

étape 4 - soit III.7 1ère table du stock. C2 n'est pas testée. Faire une nouvelle itération.

2ème itération

(traitement de la table III.7)

étape 1 - choix de la condition C2, mise en réserve

- C1 C2

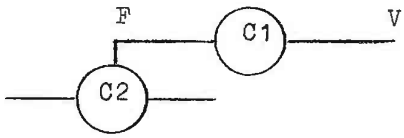
étape 2 - découpage en deux tables

III.9 pour les entrées F

R4
A4

III.10 pour les entrées V

R3
A3



étape 3 - stockage
stock = III.9 - III.10 - III.8

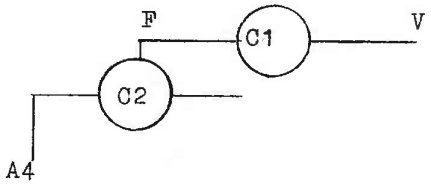
étape 4 - soit III.9 1ère table de stock.
Toutes les conditions sont testées.

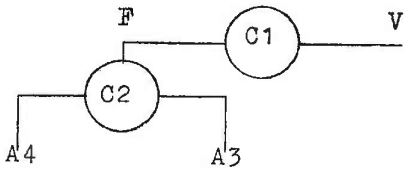
étape 5 - rangement de l'action correspondante sur l'arborescence, soit A4
- C1 C2 A4

étape 4b - soit III.10 1ère table du stock.
toutes les conditions sont testées.

étape 5b - rangement de l'action A3
- C1 C2 A4 A3

étape 4c - soit III.8, 1ère table du stock
Il reste des conditions à tester, nouvelle itération

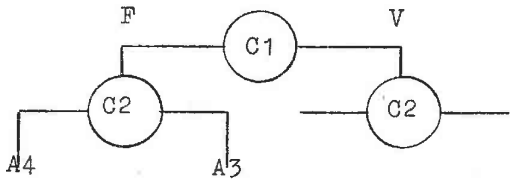




3ème itération

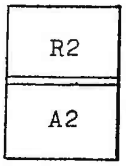
(traitement de la table III.8)

étape 1 - choix de la condition C2. Mise en réserve de C2
- C1 C2 A4 A3 C2

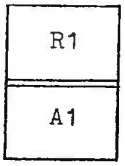


étape 2 - découpage en deux tables.

III.11 pour les entrées F



III.12 pour les entrées V



étape 3 - stockage

stock = III.11 - III.12

étape 4 - soit III.11 1ère table du stock

Toutes les conditions ont été testées.

étape 5 - rangement de l'action correspondante A2

- C1 C2 A4 A3 C2 A2

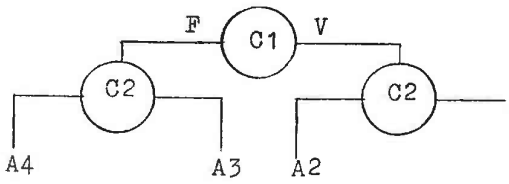
étape 4b - soit III.12 1ère table de stock

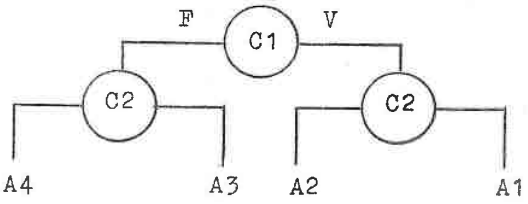
Toutes les conditions ont été testées.

étape 5b - rangement de l'action correspondante A1

- C1 C2 A4 A3 C2 A2 A1

étape 4c - stock vide FIN





L'exemple précédent, en même temps qu'il illustre le fonctionnement de l'algorithme décrit par le schéma, met en évidence quelques ambiguités de ce schéma.

Réexaminons-le "pavé par pavé".

- Choix de la condition à tester.

A chaque itération, le choix de la condition à tester est libre (ici, nous avons testé la première condition de la table).

Tous les critères de choix sont possibles (choix de la condition la plus difficile à définir ou de la plus facile....)

C'est à ce niveau du choix de la condition à tester que portent les différents procédés d'optimisation sur lesquels nous reviendrons dans le cadre de l'optimisation du traducteur.

Signalons cependant que,dans la version de base du traducteur réalisée ici, le choix de la condition se porte sur la condition ayant le plus d'entrée V ou F (le moins d'entrées indifférentes). En cas d'ambiguité, la première est choisie.

- Division en deux tables.

Le processus a été décrit précédemment.

Notons cependant qu'il est possible qu'une des deux tables résultantes soit vide.

- Stockage des tables.

Dans le schéma descriptif, l'ordre de stockage n'était pas défini. Lors de l'exemple précédent, le stockage a été fait sur une pile. Les deux tables résultantes se placent dans l'ordre suivant :

- table correspondant aux entrées V ou '-'
  puis
- table correspondant aux entrées F ou '-'

Ce choix que nous conserverons lors de la réalisation de notre traducteur est lié à la mise en mémoire, sous forme préfixée, de l'arborescence en cours de création ; ce qui implique également une création branche par branche de l'arborescence.

D'autres choix sont évidemment possibles ; particulièrement le stockage des tables obtenues par division, sur une pile, ce qui permet une création de l'arborescence niveau par niveau. Cependant, ces méthodes entraînent des complications pour la mémorisation de l'arborescence en cours de création et sa conversion ultérieure en un texte de programme.

3.2 Traduction avec entrées '-' et stockage sur une pile

EXEMPLE. Soit à traduire la table III.13

L'arborescence sera représentée en notation préfixée (notée ARB)
Le stock de table sera noté STO

	R1	R2	R3
C1	F	-	V
C2	F	V	-
C3	-	V	F
C4	-	-	F
	A1	A2	A3

1) Proposons-nous de traduire cette table à l'aide de l'algorithme utilisé pour le premier exemple.

- STO est une pile
- la division de chaque table se fait à partir de la 1ère condition.

1ère itération

traitement de III.13

Division à partir de C1. Donc ARB = C1

et STO = III.14 - III.15

ou III.14 table obtenue à partir des entrées F ou '-'

III.15 table obtenue à partir des entrées V ou '-'

Table III.14

	R1	R2
C2	F	V
C3	-	V
C4	-	-
	A1	A2

Table III.15

	R2	R3
C2	V	-
C3	V	F
C4	-	F
	A2	A3

(pour simplifier la notation de STO, les tables y seront notées tn au lieu de III.n)

t14 est en tête de STO. D'où :

2ème itération

traitement de III.14⇒ STO = t15

Division à partir de C2 d'où :

ARB = C1 C2

STO = t16-t17-t15

ou III.16 obtenue à partir des entrées F ou '-'

III.17 obtenue à partir des entrées V ou '-'

III.16

	R1
C3	-
C4	-
	A1

III.17

	R2
C3	V
C4	-
	A2

III.16 est en tête de STO d'où :

traitement de III.16 ⟹ STO = t17-t15

toutes les conditions utiles sont déjà définies

d'où stockage de l'action correspondante A1

ARB = C1 C2 A1

III.17 est maintenant en tête de STO d'où :

3ème itération

traitement de III.17 ⇒ STO = t15

Division à partir de C3 ARB = C1 C2 A1 C3

et STO = t18-t19-t15

ou III.18 obtenue à partir des entrées F ou '-'

III.19 obtenue à partir des entrées V ou '-'

III.18

VIDE

III.19

	R2
C4	-
	A2

III.18 se trouve en tête de STO d'où :

traitement de III.18 ⇒ STO = t19-t15

La table est vide, on décide dans ce cas qui correspond à une entrée non prévue de la table de prendre une action standard notée E (qui correspond à l'action de la règle E lorsque celle-ci est définie)

d'où ARB = C1 C2 A1 C3 E

III.19 se trouve en tête de STO d'où :

traitement de III.19 ⇒ STO = t15

Toutes les conditions utiles ont été définies

Stockage de l'action correspondante A2

ARB = C1 C2 A1 C3 E A2

III.15 se trouve en tête de STO d'où :

4ème itération

traitement de III.15 ⇒ STO = VIDE

Division à partir de C2 ARB = C1 C1 A1 C3 E A2 C2

et STO = t20-t21

ou III.20 obtenue à partir des entrées F ou '-'

III.21 obtenue à partir des entrées V ou '-'

III.20

	R3
C3	F
C4	F
	A3

III.21

	R2	R3
C3	V	F
C4	-	F
	A2	A3

III.20 se trouve maintenant en tête de STO d'où :

5ème itération

traitement de III.20 ⇒ STO = t21

Division à partir de C3 ARB = C1 C2 A1 C3 E A2 C2 C3

et STO = t22-t23-t21

ou III.22 obtenue à partir des entrées F ou '-'

III.23 obtenue à partir des entrées V ou '-'

III.22

	R3
C1	F
	A3

III.23

VIDE

III.22 se trouve en tête de STO d'où :

6ème itération

traitement de III.22 ⇒ STO = t23-t21

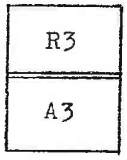
Division à partir de C4 ARB = C1 C2 A1 C3 E A2 C2 C3 C4

et STO = t24-t25-t23-t21

ou III.24 obtenue à partir des entrées F ou '-'

III.25 obtenue à partir des entrées V ou '-'

III.24 III.25



VIDE

III.24 se trouve en tête de STO d'où :

traitement de III.24 ⇒ STO = t25-t23-t21

Toutes les conditions utiles sont définies

d'où stockage de l'action correspondante A3

ARB = C1 C2 A1 C3 E A2 C2 C3 C4 A3

III.25 se trouve en tête de STO d'où :

traitement III.25 ⇒ STO = t23-t21

la table est vide, on effectue l'action standard E

ARB = C1 C2 A1 C3 E A2 C2 C3 C4 A3 E

III.23 se trouve en tête de STO d'où :

traitement de III.23 ⇒ STO = t21

la table est vide action standard E

ARB = C1 C2 A1 C3 E A2 C2 C3 C4 A3 E E

III.21 se trouve en tête de STO d'où :

7ème itération

traitement de III.21 ⇒ STO = vide

division à partir de C3

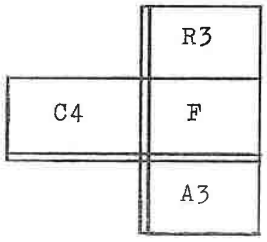
ARB = C1 C2 A1 C3 E A2 C2 C3 C4 A3 E E C3

etSTO = t26-t27

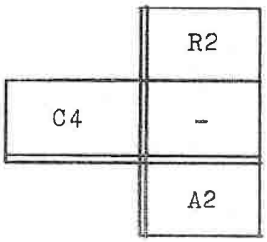
ou III.26 obtenue à partir des entrées F ou '-'

III.27 obtenue à partir des entrées V ou '-'

III.26



III.27



III.26 se trouve en tête de STO d'où :

8ème itération

traitement de III.26

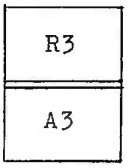
Division à partir de C4

ARB = C1 C2 A1 C3 E A2 C2 C3 C4 A3 E E C3 C4

et STO = t28-t29-t27

ou III.28 obtenue à partir des entrées F ou '-'
III.29 obtenue à partir des entrées V ou '-'

III.28 III.29



VIDE

III.28 se trouve en tête de STO d'où :

traitement de III.28 ⇒ STO = t29-t27
toutes les conditions utiles sont définies
d'où stockage de l'action correspondante A3
ARB = C1 C2 A1 C3 E A2 C2 C3 C4 A3 E E C3 C4 A3

III.29 se trouve en tête de STO d'où :

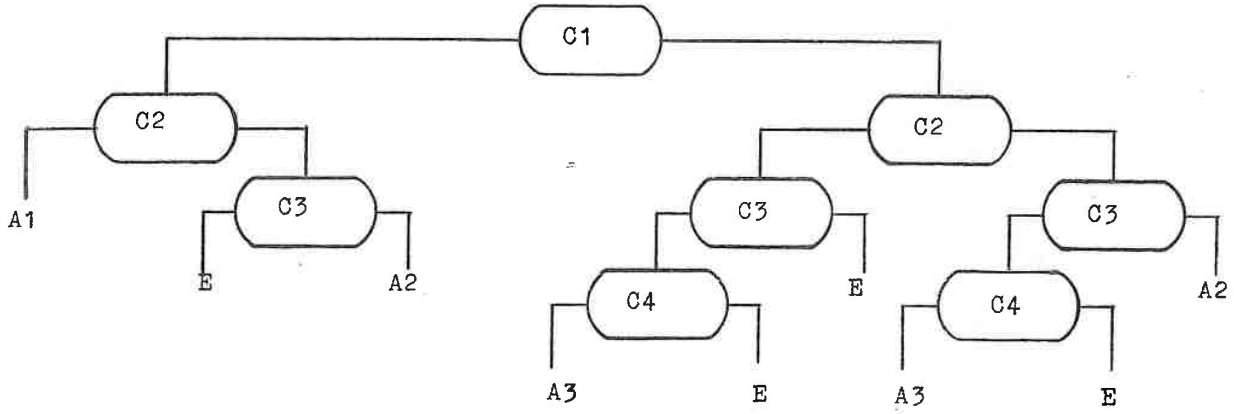
traitement de III.29 ⇒ STO = t27
la table est vide, on effectue l'action E
ARB = C1 C2 A1 C3 E A2 C2 C3 C4 A3 E E C3 C4 A3 E

III.27 se trouve en tête de STO d'où :

traitement de III.27 ⇒ STO = VIDE
toutes les conditions utiles sont définies
d'où stockage de l'action correspondante A2
ARB = C1 C2 A1 C3 E A2 C2 C3 C4 A3 E E C3 C4 A3 E A2

STO vide fin de la traduction

ARB représente l'arborescence suivante :



Ce résultat pourrait également être obtenu en effectuant le stockage des tables sur une file.

3.3. Traduction avec entrées '-'. Stockage sur une file

EXEMPLE. Reprenons la table III.13

- STO est une file
- la division des tables se fait à partir de la 1ère condition
- ARB représentatation schématique de l'arborescence
- les tables seront celles définies dans l'exemple précédent.

1ère itération

traitement de III.13 ⟹ STO = VIDE

division à partir de C1

ARB : C1

STO = t14-t15

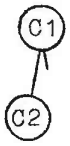
III.14 en tête de file d'où :

2ème itération

traitement de III.14 ⟹ STO = t15

division à partir de C2

ARB :



STO = t15-t16-t17

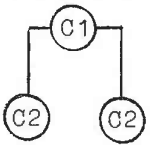
III.15 en tête de file d'où

3ème itération

traitement de III.15 ⟹ STO = t16-t17

division à partir de C2

ARB :



STO = t16-t17-t20-t21

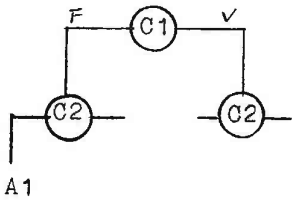
III.16 en tête de file d'où :

traitement de III.16 ⟹ STO = t17-t20-t21

toutes les conditions utiles sont déjà testées

d'où stockage de l'action correspondante A1

ARB



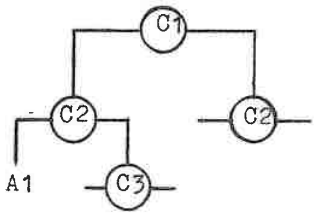
III.17 se trouve en tête de file d'où :

4ème itération

traitement de III.17 $\Longrightarrow$ STO = t20-t21

division à partir de C3

ARB



STO = t20-t21-t18-t19

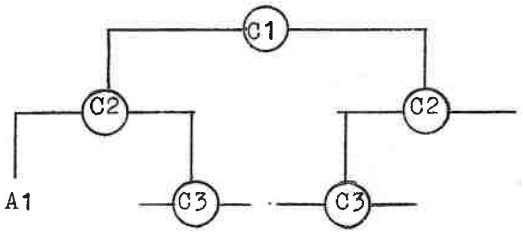
III.20 se trouve en tête de file d'où :

5ème itération

traitement de III.20 ⟹ STO = t21-t18-t19

division à partir de C3

ARB :



STO = t21-t18-t19-t22-t23

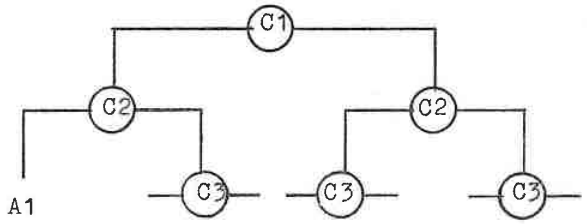
III.21 en tête de file d'où :

6ème itération

traitement de III.21 ⟹ STO = t18-t19-t22-t23

division à partir de C3

ARB :



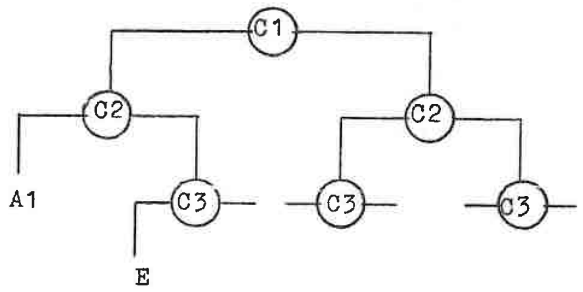
STO = t18-t19-t22-t23-t26-t27

III.18 en tête de file d'où :

traitement de III.18 ⇒ STO = t19-t22-t23-t26-t27

III.18 est vide. On prend l'action standard E

ARB :



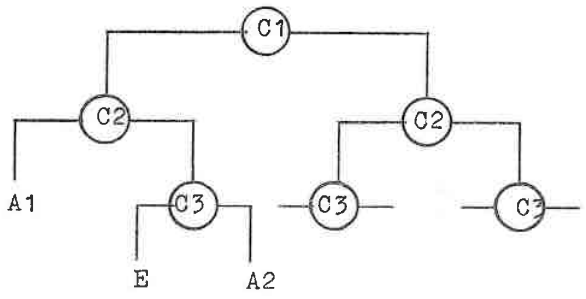
III.19 se trouve en tête de file d'où :

traitement de III.19 ⇒ STO = t22-t23-t26-t27

toutes les conditions utiles sont déjà traitées

d'où stockage de l'action correspondante A2

ARB :



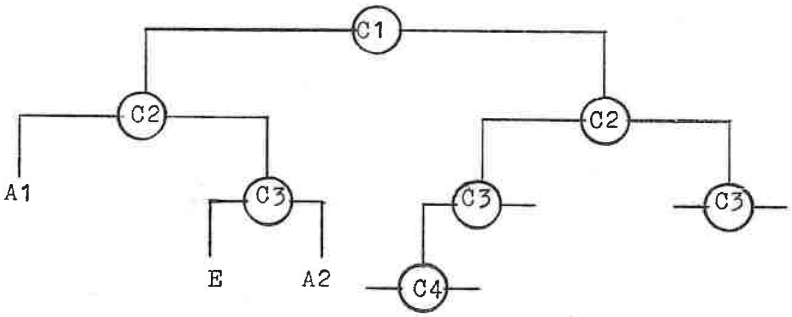
III.22 en tête de file d'où :

7ème itération

traitement de III.22 ⟹ STO = t23-t26-t27

division à partir de C4

ARB :



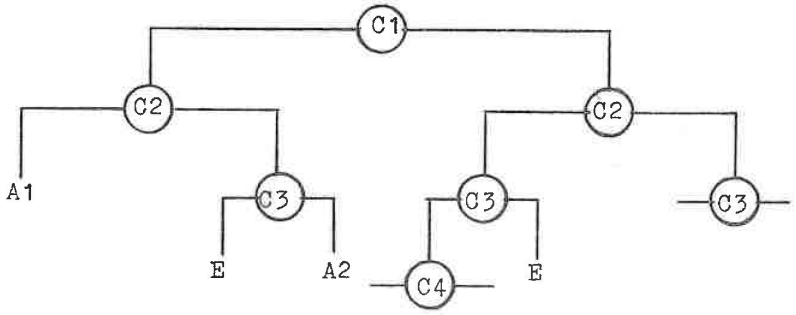
STO = t23-t26-t27-t24-t25

III.23 en tête de file d'où :

traitement de III.23 ⟹ STO = t26-t27-t24-t25

III.23 vide d'où stockage de l'action E

ARB :



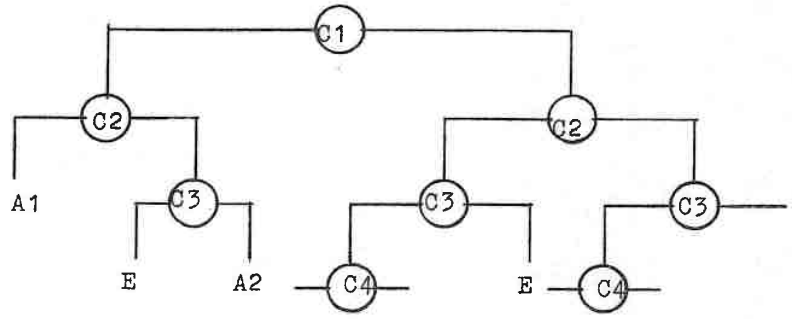
III.26 se trouve en tête de file d'où :

8ème itération

traitement de III.26 ⟹ STO = t27-t24-t25

division à partir de C4

ARB :



STO = t27-t24-t25-t28-t29

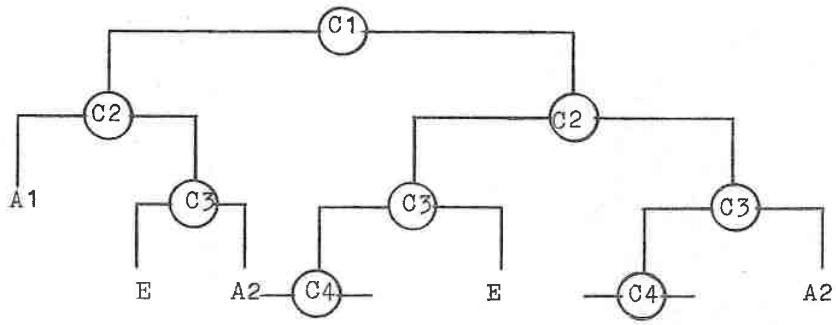
III.27 se trouve en tête de file d'où :

traitement de III.27 ⟹ STO = t24-t25-t28-t29

toutes les conditions sont définies

stockage de l'action correspondante A2

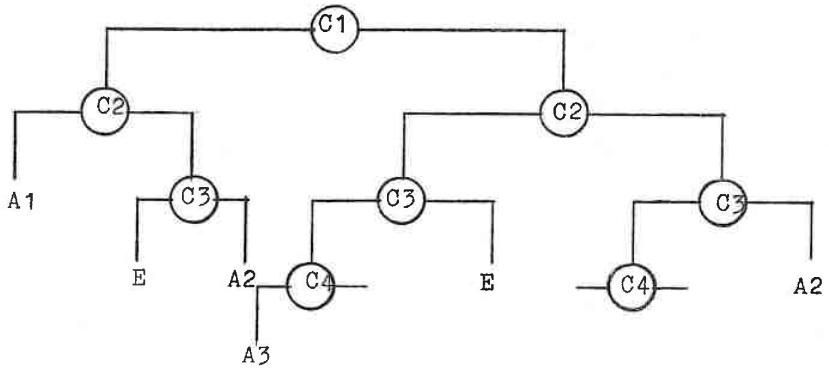
ARB :



III.24 se trouve en tête de file d'où :

traitement de III.24 ⟹ STO = t25-t28-t29
toutes les conditions utiles sont définies
stockage de l'action correspondante A3

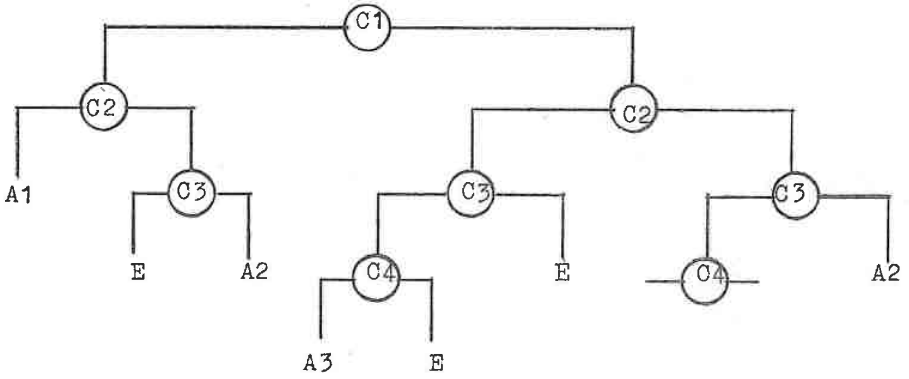
ARB :



III.25 se trouve en tête de file d'où :

traitement de III.25 ⟹ STO = t28-t29
III.25 vide
action standard E

ARB :



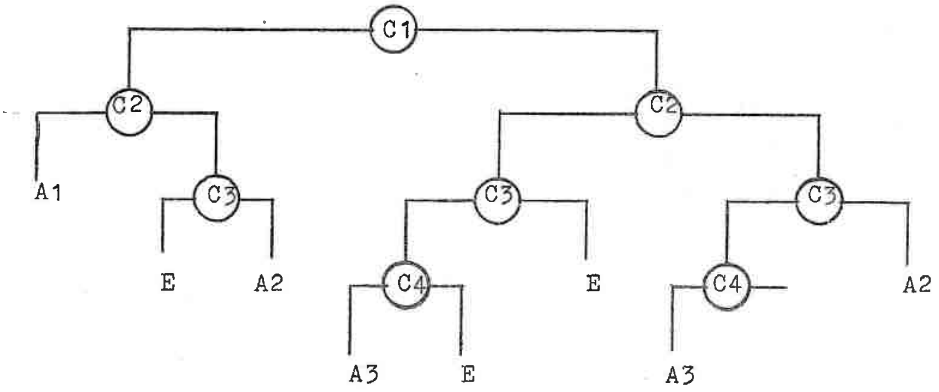
III.28 en tête de file d'où :

traitement de III.28 $\Longrightarrow$ STO = t29

toutes les conditions utiles sont définies

stockage de l'action A3

ARB :



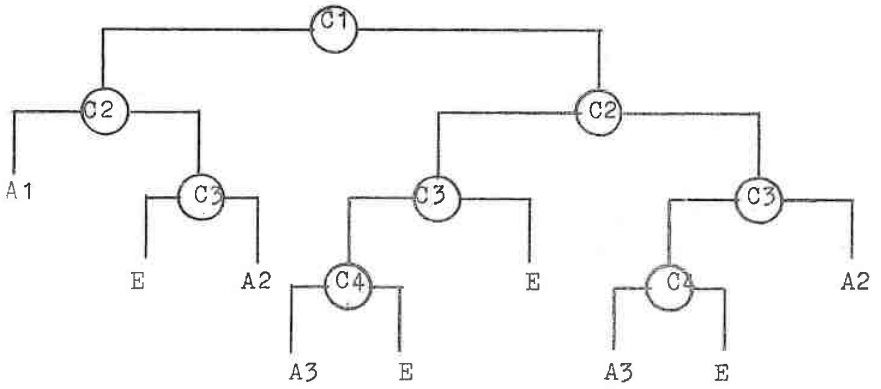
III.29 en tête de file d'où :

traitement de III.29 $\Longrightarrow$ STO = vide

III.29 vide

stockage de l'action standard E

ARB :



STO VIDE

FIN de traduction

Le résultat est évidemment semblable à celui obtenu par la méthode précédente.

Cet exemple illustre bien la création, niveau par niveau, de l'arborescence résultante.

3.4 Traduction avec sélection de la condition à tester

Le choix d'une nouvelle règle pour la détermination de la première condition à tester dans chaque table produirait une arborescence différente.

Reprenons cet exemple en décidant de choisir la condition qui a le moins d'entrées '-'.

La table de départ en encore III.13

	R1	R2	R3
C1	F	-	V
C2	F	V	-
C3	-	V	F
C4	-	-	F
	A1	A2	A3

STO sera une pile

ARB notation préfixée de l'arborescence en construction

1-traitement de III.13

C1 1ère des conditions a une une seule entrée '-'

ARB = C1

STO = t14-t15

III.14

	R1	R2
C2	F	V
C3	-	V
C4	-	-
	A1	A2

III.15

	R2	R3
C2	V	-
C3	V	F
C4	-	F
	A2	A3

2-traitement de III.14

division à partir de C2

ARB = C1 C2

STO = t16-t17-t15

III.16

	R1
C3	-
C4	-
	A1

III.17

	R2
C3	V
C4	-
	A2

3-traitement de III.16

plus de condition à tester ⇒ action A1

ARB = C1 C2 A1

STO = t17-t15

4-traitement de III.17

division à partir de C3

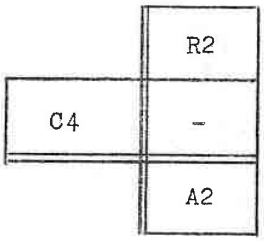
ARB = C1 C2 A1 C3

STO = t18-t19-t15

III.18

VIDE

III.19



5-traitement de III.18

III.18 vide ⇒ action E

ARB = C1 C2 A1 C3 E

STO = t19-t15

6-traitement de III.19

plus de condition à tester ⇒ action A2

ARB = C1 C2 A1 C3 E A2

STO = t15

7-traitement de III.15

division à partir de C3 qui n'a pas d'entrée '-'

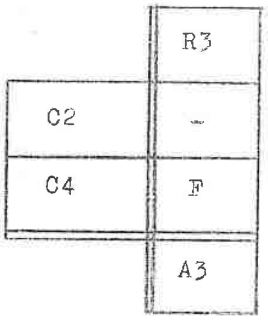
ARB = C1 C2 A1 C3 E A2 C3

STO = t30-t31

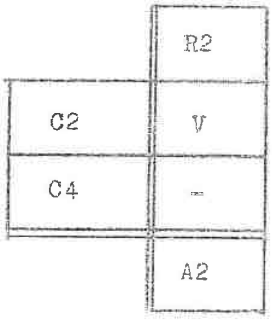
III.30 obtenue à partir des entrées F ou '-'

III.31 obtenue à partir des entrées V ou '-'

III.30



III.31



8-traitement de III.30

division à partir de C4

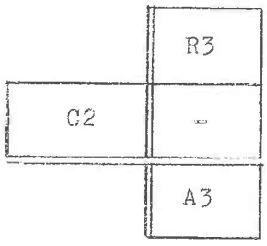
ARB = C1 C2 A1 C3 E A2 C3 C4

STO = t32-t33-t31

III.32 obtenue à partir des entrées F ou '-'

III.33 obtenue à partir des entrées V ou '-'

III.32



III.33

VIDE

9 -traitement de III.32

Plus de condition à tester ⇒ action A3

ARB = C1 C2 A1 C3 E A2 C3 C4 A3

STO = t33-t31

10-traitement de III.33

III.33 vide ⇒ action E

ARB = C1 C2 A1 C3 E A2 C3 C4 A3 E

STO = t31

11-traitement de III.31

division à partir de C2

ARB = C1 C2 A1 C3 E A2 C3 C4 A3 E C2

STO = t34-t35

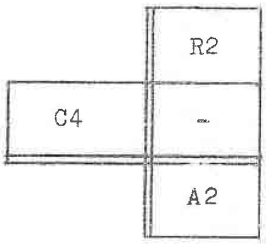
III.34 obtenue à partir des entrées F ou '-'

III.35 obtenue à partir des entrées V ou '-'

III.34

VIDE

III.35



12-traitement de III.34

III.34 vide ⇒ action E

ARB = C1 C2 A1 C3 E A2 C3 C4 A3 E C2 E

STO = t35

13-traitement de III.35

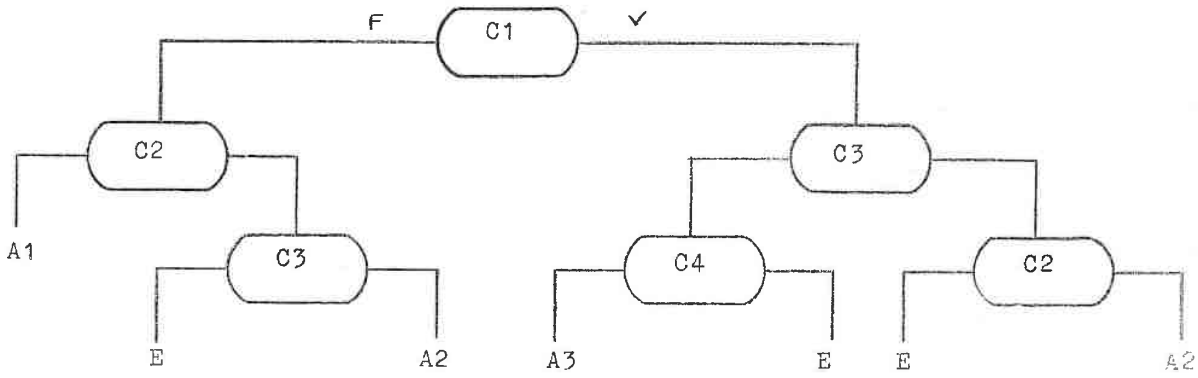
plus de condition à tester ⇒ action A2

ARB = C1 C2 A1 C3 E A2 C3 C4 A3 E C2 E A2

STO VIDE

fin de traduction

ARB représentant l'arborescence suivante :



L'algorithme utilisé pour traiter cet exemple correspond à celui qui sera utilisé pour la réalisation du traducteur, avec cependant deux extensions :

- possibilité de traiter des tables ambigues
- possibilité de traiter des tables avec corditions liées

3.5 Traduction de tables ambigues

Le processus est exactement le même que dans le cas simple en admettant que la table qui, toutes ses conditions étant testées, détermine une action (donc comprend une seule règle) puisse en fait déterminer plusieurs actions (donc comprendre encore plusieurs règles).

EXEMPLE. Soit la table III.36

	R1	R2	R3
C1	V	-	V
C2	-	V	-
C3	F	F	V
	A1	A2	A3

R1 et R2 sont ambigues

1ère itération

choix de C3

division

ARB = C3

STO = t37-t38

III.37

C1	V	-
C2	-	V
	1	2

III.38

C1	V
C2	-
	3

2ème itération

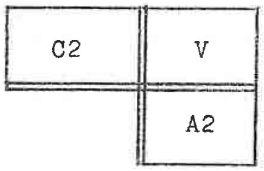
choix de C1

division

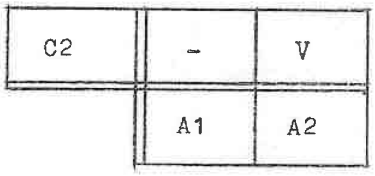
ARB = C3 C1

STO = t39-t40-t38

III.39



III.40



3ème itération

choix de C2

division en deux tables

ARB = C3 C1 C2

STO = t40-t38

VIDE

d'où ARB = C3 C1 C2 AE A2 C2

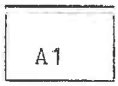
4ème itération

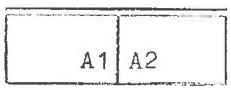
choix de C2

ARB = C3 C1 C2 AE A2 C2

division en deux tables

STO = t38





d'où ARB = C3 C1 C2 AE A2 C2 A1 A1,2

A1,2 étant la réunion des ensembles d'actions

A1 et A2 sans ordre entre A1 et A2

5ème itération (III.38)

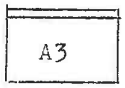
choix de C1

ARB = C3 C1 C2 AE A2 C2 A1 A1,2 C1

division en deux tables

STO = VIDE

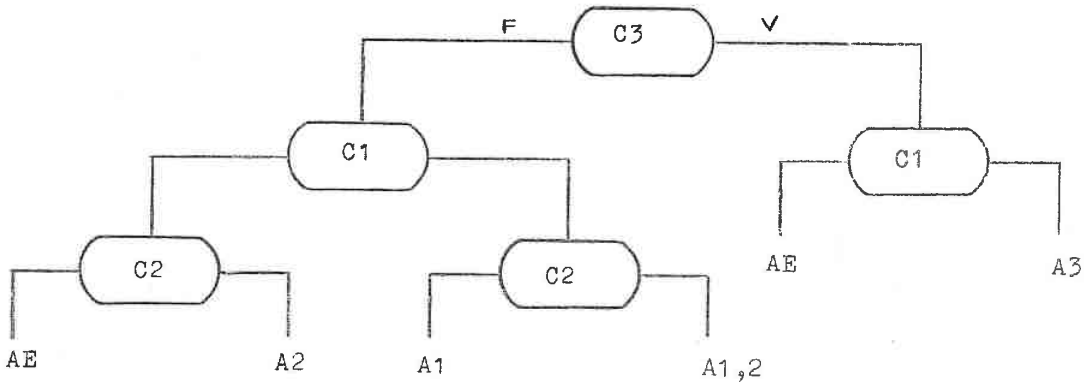
VIDE



d'où ARB = C3 C1 C2 AE A2 C2 A1 A1,2 C1 AE A3

FIN de traduction

ARB représente l'arborescence suivante :



REMARQUE.-

Lorsque plusieurs règles sont ambigues, nous sommes amenés à réaliser simultanément leurs ensembles d'actions (EX. A1,2) Il est impossible ici de spécifier un ordre d'exécution entre A1 et A2

3.6 Traduction des tables avec conditions liées

Le processus de traduction est encore le même avec la règle suivante.

- lors du choix de la condition à tester, les entrées de condition marquées + sont assimilées aux entrées '-'
- au moment de la division en deux tables, les entrées +F sont assimilées aux entrées F, les entrées +V sont assimilées aux entrées V

EXEMPLE. Soit la table III.41

On a ici la liaison

$C1 \longrightarrow \overline{C2}$

donc réciproquement

$C2 \longrightarrow \overline{C1}$

	R1	R2	R3
C1	V	+F	V
C2	+F	V	+F
C3	F	F	V
	A1	A2	A3

1ère itération

choix de C3

ARB = C3

division en deux

STO = t42-t43

III.42

	R1	R2
C1	V	+F
C2	+F	V
	A1	A2

III.43

	R3
C1	V
C2	+F
	A3

2ème itération (III.42)

choix de C1

ARB = C3 C1

Division en deux

STO = t44-t45-t42

III.44

	R2
C2	V
	A2

III.45

	R1
C2	+F
	A1

3ème itération (III.44)

choix de C2

ARB = C3 C1 C2

division en deux tables

VIDE

A2

d'où ARB = C3 C1 C2 AE A2

4ème itération (III.45)

plus de condition à tester ⟹ action 1

ARB = C3 C1 C2 AE A2 A1

5ème itération (III.42)

choix de C2

ARB = C3 C1 C2 AE A2 A1 C2

division en deux tables

STO = t46-t47

III.46

VIDE

III.47

	R3
C2	+F
	A3

6ème itération (III.46)

table vide

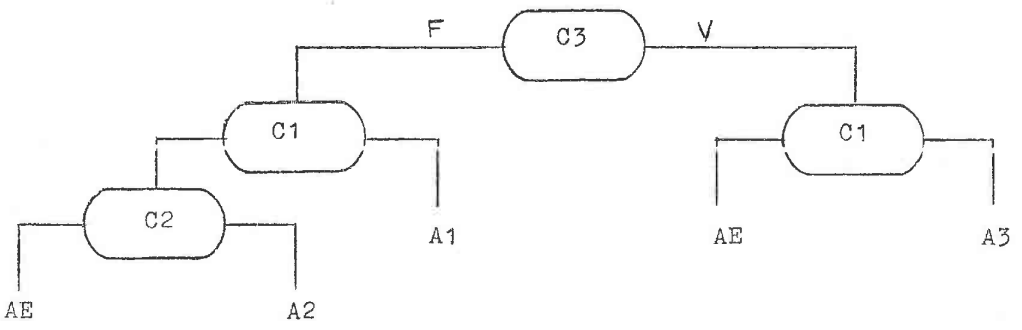
ARB = C3 C1 C2 AE A2 A1 C2 AE

7ème itération (III.47)

plus de condition à évaluer ⟹ action A3

ARB = C3 C1 C2 AE A2 A1 C2 AE A3

FIN de traduction



CHAPITRE IV

REALISATION DU TRADUCTEUR

1. GENERALITES.-

1.1 L'algorithme de traduction.

Il s'agit, dans ses grandes lignes, de l'algorithme présenté page III.6

On procède par analyse globale de la table avec division en sous-tables, jusqu'à la résolution complète.

- le stockage des tables au cours de la traduction est fait sur une pile,
- l'arborescence résultante est créée sous une forme préfixée,
- pour la division d'une table, on choisit la condition qui présente le moins d'entrées neutres (entrées -, +F, +V)
- la division d'une table fournit deux sous-tables :
  - une table correspondant aux entrées F ou '-' de la condition choisie,
  - une table correspondant aux entrées V ou '-'.

Cet algorithme permet la traduction des tables à entrées limitées, même ambigues ou présentant des conditions non indépendantes.

1.2. Description schématique de l'algorithme.

- Précisons tout d'abord quelques notations qui seront également utilisées pour la description détaillée du programme.

Soient :

TABLE : la table à traduire,
STO : la pile gérée par le traducteur,
TA : étant la table figurant au sommet de STO,
la division de TA donnera TAV et TAF
ARB : l'arborescence descriptive.

. l'algorithme pourra se décrire ainsi :
- initialisation

TABLE est placée sur STO

d'où

TA = TABLE

TA est une table vide (pas de règle)	V	V	+F	+F	F
TA est une table neutre (toutes les conditions utiles sont testées)	+F	+F	V	V	F
TA est la dernière table de STO	V	F	V	F	-
Placer l'action E sur ARB	X	X	-	-	-
Placer l'action concernée sur ARB	-	-	X	X	-
Rechercher condition Division TA en TAF et TAV TAF remplace TA TAV placé en tête de STO	-	-	-	-	X
Boucler la table	-	X	-	X	X

. en utilisant l'organigramme, nous décrirons l'algorithme par le schéma suivant (les notations étant identiques)

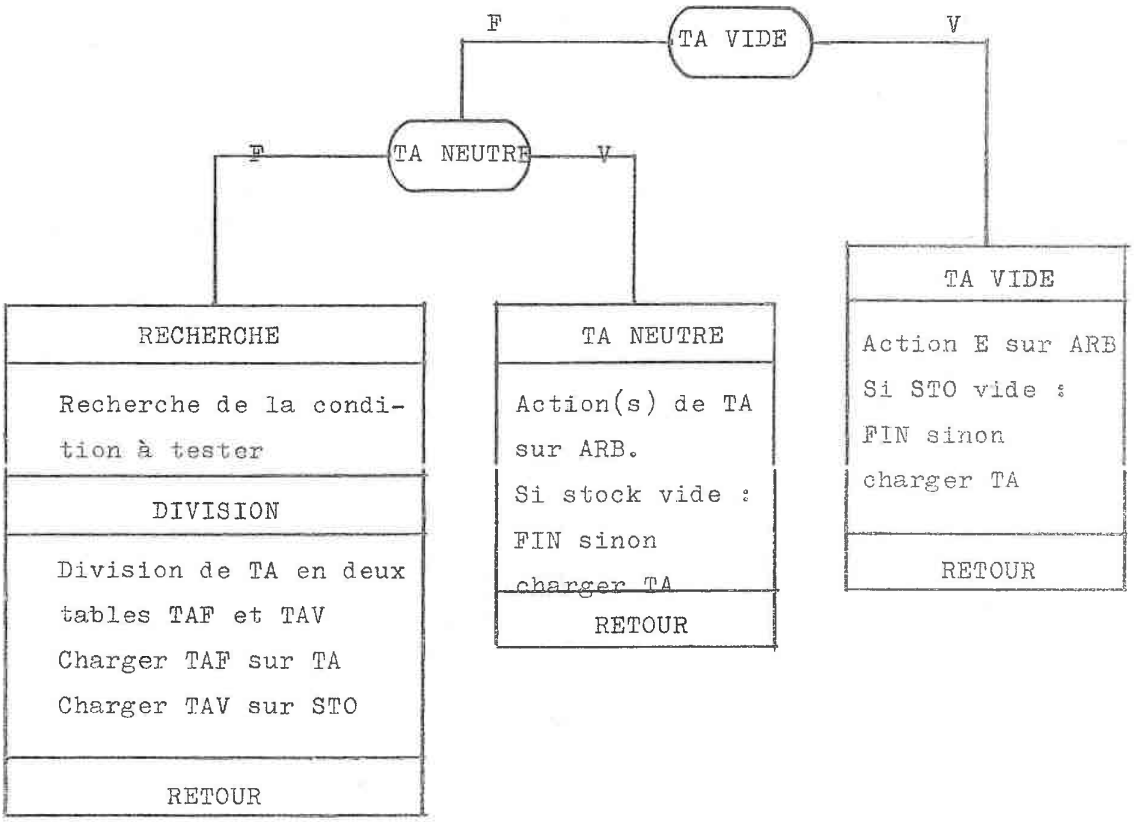
REMARQUE.-

Cet organigramme représente une des traductions possibles de la table de division précédente mais d'autres sont évidemment possibles,

PHASE PRELIMINAIRE

Standardisation - transposition Séparation des actions - initialisation

PHASES ITERATIVES



I.3. Les entrées

1.3.1) format d'écriture des tables.

- les tables admises par le traducteur sont des tables à entrées limitées qui peuvent être ambigues ou à conditions liées.
- les souches conditions seront des expressions booléennes, les souches actions étant des instructions ou des appels de modules.
- les entrées des conditions seront :

  F : faux
  V : vrai
  - : indifférent
  +V : vrai, impliqué par les réalisations d'autres conditions de la règle.
  +F : faux, impliqué par les réalisations d'autres conditions de la règle.
  blanc : entrées pour la règle E

- les entrées des actions seront :

  X : action à réaliser,
  0,1, 0,2, ..., n, ... : action à réaliser dans l'ordre spécifié (chaque numéro étant écrit sur deux positions)
  - : action ignorée.

- la règle E (facultative)
  si elle est présente, l'entrée des conditions sera blanc, l'entrée des actions étant normale.

1.3.2.) <u>acquisition des tables</u>

L'acquisition de la table par le traducteur se fait ligne par ligne, c'est-à-dire que la table est entrée sous forme d'une chaine de caractères représentant la table ligne par ligne de haut en bas.

L'entrée comportera en plus trois valeurs fournies par l'auteur de la table (ou le codifieur) :

. le nombre de conditions
. le nombre d'actions
. le nombre de règles (E incluse éventuellement)

EXEMPLE : soit la table IV.1

Age < 25	V	F	+F
Age > 60	+F	F	V
Tarif = A	X	-	-
Tarif = B	-	X	-
Tarif = C	-	-	X

En entrée du traducteur, on aura alors :

1) NCOND = 2, NACTI = 3, NREGL = 3
2) une chaine de caractères :

```
//AGE < 25//VF+F//AGE > 60//+FFV//
  TARIF=A//X--//TARIF=B//-X-//
  TARIF=C//--X//
```

REMARQUES.-

Cette chaine aurait pu être :

```
//AGE < 25//␣V␣F+F//AGE > 60//+F␣F␣V//
  TARIF=A//␣X␣-␣-//TARIF=B//␣-␣X␣-//
  TARIF=C//␣-␣-␣X//
```

Cette chaine étant obtenue à partir de la tableIV.1 en attribuant deux caractères à chaque entrée.

Les souches (conditions et actions) n'étant pas utiles lors de la traduction de la table, nous procéderons d'abord à un découpage de la chaine entrée : les souches seront gardées en mémoire en vue de leur utilisation lors de la génération du programme résultant.

Une nouvelle chaine est obtenue par suppression des éléments de souches et des séparateurs, les lignes étant suffisamment différenciées par leur position et leur longueur.

EXEMPLE.-

La table IV.1. sera donc transformée pour être présentée sous la forme suivante :

1 - 3 valeurs : NCOND = 2, NACTI = 3, NREGL = 3

2 - une chaine, réserve des souches :
AGE < 25//AGE > 60//TARIF=A//TARIF=B//TARIF=C//
la présence de blancs est possible dans cette chaine.

3 - une chaine représentant "les entrées" de la table à traduire :

a) VF+F+FFVX---X---X

ou

b)V␣F+F+F␣F␣V␣X␣-␣-␣-␣-␣X␣-␣-␣-␣-␣X

Cette diversité possible dans l'écriture des entrées justifie la décision de représenter par '-' l'entrée des actions à ignorer qui est généralement représentée par blanc.

En effet, ceci permet des blancs dans la chaine de caractères; ces blancs étant considérés comme caractères non significatifs seront facilement éliminés (la partie entrée des conditions d'une éventuelle règle E qui est normalement blanche ne figurera pas sur cette chaine mais seulement la partie entrée des actions selon la même syntaxe que les autres règles).

La décision d'écrire les numéros d'ordre daction sur deux positions à également été prise à cause de cette liberté d'écriture de la chaine d'entrée.

2. PHASE PRELIMINAIRE DE LA TRADUCTION.

2.1. Standardisation de la chaine entrée.

A partir de la chaine fournie, nous désirons obtenir une chaine normalisée, les caractères admis étant :

V, F	:	entrées Vrai et Faux
W et G	:	entrées +V et +F
-	:	entrée de condition indifférente
blanc ou -	:	entrée d'action à ignorer
X	:	entrée daction à exécuter
numérique	:	entrée d'action à exécuter (numéro d'ordre)
blanc	:	entrée des conditions de la règle E

Chaque caractère représentera donc une entrée de condition et d'action et toutes les entrées (même celles de la règle E) seront représentées.

Il s'agit donc à partir de la chaine fournie

- d'éliminer les blancs et autres caractères non significatifs.
- de regrouper sur un seul caractère les entrées +V, +F et numériques (nn)

  les conversions étant :

  +V ⟶ W
  +F ⟶ G
  NN ⟶ binaire sur un octet.

- détecter des éventuelles erreurs de syntaxe.

Cette phase de "compactage" peut être représentée par le schéma suivant :

TESTCAR 1 - (test d'un caractère)

LEC lecture d'un caractère (C) de la chaine

C = blanc	C = V,F,-,X.	C = +, num .	autre
Retour à LEC	recopie sur la nouvelle chaine. Retour LEC	Appel module TESTCAR 2	anomalie Retour à LEC

TESTCAR 2 - (vérification de la validité du couple CC'

- stockage C
- lecture de C' suivant C

C = +	V	F	
C' = V,F	V	+F	
C' numérique	+F	V	
conversion C' recopie	X	-	-
conversion CC' recopie	-	X	-
RETOUR	X	X	02
Anomalie	-	-	01

D'autre part, si il existe une règle E (RE = 1) à la fin de chaque ligne condition, il sera inséré un caractère ␣ entrée de la règle E.

- conversion C' si C'=V(CC'=+V) alors écriture N
  si C'=F(CC'=+F) alors écriture G

- conversion CC' CC' est un nombre à deux chiffres converti en binaire sur un octet.

Nous sommes maintenant en mesure de passer à la programmation.

2.1.1) Déclarations et réservations

a) entrée de la table à traiter avec ses paramètres. Ici on entre la table IV.1 sous la forme décrite page IV.7

- DONNEES.

```
CH    DATA   C' V F'
             C'+F+F'
             C' F V'
             C' X -'
             C' - -'          Chaine entrée
             C' X -'
             C' - -'
             C' X  '

NA    DATA   3                Nombres d'actions
NC    DATA   2                Nombres de conditions
NR    DATA   3                Nombres de règles
RE    DATA   0                Règle E
```

b) déclaration des constantes utilisées dans le programme.

CONSTANTES

```
BL      DATA    C' 000'
TI      DATA    C'-000'
F       DATA    C'F000'
PF      DATA    C'G000'
PV      DATA    C'W000'
V       DATA    C'V000'
F9      DATA    C'9000'
FO      DATA    C'0000'
PL      DATA    C'+000'
X       DATA    C'X000'
```

c) Réservation de variables de travail

VARIABLES

```
ST      RES     1
CT      RES     1
NL      RES     1
ER1     RES     1
ER2     RES     1
ER3     RES     1
ER      RES     1
NEC     RES     1
A       RES     8
```

Nombre effectif de règles (sansE)

Nombre de lignes dans la table

Chaine en sortie (après la phase compactage)

NEC = nombre d'entrées de condition avec celles de la règle E
= 0 si il n'y a pas de règle E (si RE = 0)

2.1.2. Programme.

STANDARDISATION

a)-détermination de CT, NEC, N^L

-calcul du nombre d'entrées de la chaine CH
stockage sur le registre 1

-initialisation des registres 5 et 3
respectivement index sur les chaines CH et A

INITIALISATION

```
LW, 1       NR
SW,1        RE              Calcul de CT
STW,1       CT

LW,1        NC
MW,1        NR              Calcul de NEC
STW,1       NEC

LW,1        NA
AW,1        NC              Calcul de NL
STW,1       NL

MW,1        NR
LW,2        NC
MW,2        RE              (1) = nombres d'entrées
SW,1        2

LI,5        0
LI,3        0
```

b) Prise en charge du caractère indexé par 5 sur CH. Test de ce caractère.

TEST CAR.1

```
LEC     LB,6    CH,5    lecture
        AI,5    1       (prise en charge)

        CW,6    BL      si C = BL
        BE      EL1     aller à EL1

        CW,6    V
        BE      EL2     Si C = V, F, -, X
        CW,6    F       aller à EL2
        BE      EL2
        CW,6    TI
        BE      EL2
        CW,6    X
        BE      EL2

        CW,6    PL
        BE      EL3     Si C = + ou C numérique
        CW,6    F0      aller à EL3 sinon aller à EL4
        BL      EL4
        CW,6    F9
        BL      EL3
```

```
EL4     LI,2      1        Erreur syntaxique
        STW,2     ER1      (caractère non significatif)
        STW,2     ER       ER1 et ER positionnés à 1
        B         EL1

EL2     STB,6     A,3      recopie sur A du
        AI,3      1        caractère lu
        B         EL5
```

c) test de caractère suivant C
Dans les cas où celui-ci est '+' ou numérique
conversion C' et CC'

TEST CAR.2

```
EL3     STB,6     4        Stockage de C
        LB,6      CH,5     prise en charge de C',
        AI,5      1        caractère suivant C

        CW,4      PL       Si C est numérique
        BNE       EL6      aller à EL6

        CW,6      V        Si C' n'est pas 'V'
        BNE       EL7      aller à EL7

        LB,2      PV       nous sommes dans le cas CC'=+V
        STB,2     A,3
        AI,3      1        Recopie de W sur A
        B         EL5
```

```
EL7     CW,6    F       Si C' n'est pas 'F'
        BNE     EL8     aller à EL8

        LB,2    PF      Nous sommes dans le cas
        STB,2   A,3     CC'=+F
        AI,3    1       Recopie de G sur A
        B       EL5

EL8     LI,2    1       Erreur syntaxe
        STW,2   ER2     + non suivi de F ou V
        STW,2   ER      ER2 et ER positionnés à 1
        B       EL5

EL6     CW,6    F0      Cas où C est numérique
        BL      EL9     Si C' n'est pas numérique
        CW,6    F9      aller à EL9
        BL      ELA

EL9     LI,2    1       Erreur syntaxique
        STW,2   ER3     Numérique non groupé
        STW,2   ER      par 2
        B       EL5     ER3 et ER positionnés à 1

ELA     SW,6    F0      Conversion des 2 caractères
        SW,4    F0      numériquess en binaire
        MW,4    DIX     Recopie sur A
        AW,4    6
        STB,6   A,3
        AI,3    1
        B       EL5
```

d) Retour à lecture

Dans le cas où il existe une règle E, un blanc sera ajouté à la fin de chaque ligne condition de façon à recréer l'entrée des conditions de la règle E

RETOUR

```
EL1     AI,1     1       Incrémentation de (1) de
                         façon à annuler l'effet de BDR, dans
                         le cas où le caractère qui vient
                         d'être lu ne correspond pas à une
                         entrée.

EL5     CW,3     NEC     Si (3)⩾ nombres d'entrées de con-
        BGE      ELB     ditions, aller à ELB.
        CW,3     CT      Si (3) ne correspond pas à une
        BNE      ELB     fin de ligne condition,aller
        LW,2     CT      à ELB.
        AW,2     NR      Incrémentation de CT
        STW,2    CT

        LW,2     BL      Génération de blanc sur A
        STB,2    A,3     comme entrée de condition
        AI,3     1       de la règle E

ELB     BDR,1    LEC     Décrémentation de (1)
                         Retour à LEC si (1)⩾ 0
```

EXEMPLE 1 Soit la chaine

␣V␣F+F+F␣F␣V␣X␣-␣-␣-␣X␣-␣-␣-␣X

représentant la table IV.1 (écrite avec 2 caractères par case)

On aura le processus suivant :

TESCAR 1	C=C1=␣			retour à lecture
"	C=C2=V		recopie A=V	"
"	C=C3=␣			"
"	C=C4=F		recopie A=VF	"
"	C=C5=+			
TESCAR 2	C'=C6=F	C' devient G	recopie A=VFG	retour à lecture
" 1	C=C7=+			
" 2	C'=C8=F	C' devient G	recopie A=VFGG	retour à lecture
" 1	C=C9=␣			"
" 1	C=C10=F		recopie A=VFGGF	"

le processus se poursuit par itération sur TESCAR 2

on obtient : A = VFGGFVX---X---X

EXEMPLE 2 Soit la table IV.2

Cond. 1	V	F	+F	
Cond. 2	F	-	F	
Cond. 3	+F	F	V	
Act. 1	01	-	-	-
Act. 2	02	X	X	X
Act. 3	-	-	-	X

L'entrée du traducteur sera alors :

NR = 4, NA = 3, NC = 3, RE = 1

CH = VF+FF-F+FFVOA---O2XXX---X

ce pourra être également :

CH = ␣V␣F+F␣F␣-␣F+F␣F␣VO1␣-␣-␣-

␣␣␣O2␣X␣X␣X␣-␣␣-␣-␣X

Dans tous les cas, le compactage donnera :

A = VFG␣G-G␣GFV␣1---2XXX---X

2.2 Transposition de la table entrée.

L'entrée de la table ligne par ligne est très naturelle à partir d'un texte écrit, mais la traduction nécessite un traitement règle par règle; nous devons donc transposer la table de façon à obtenir une représentation sous forme d'une suite de règle.

Soit A la chaine représentant la table (après compactage)
Soit B la chaine résultante (sous forme d'une suite de règles)
Soient NR et NL nombre de règles et nombre de lignes

La première règle est obtenue avec les caractères et ordre :
0, NA, 2×NR, 3×NR, ...(NL-1)×NR de A
La deuxième règle avec les caractères :
1, NR+1, 2×NR+1, ...(NL-1)×NR+1
et ainsi de suite pour les NR règles

Notons .IA l'indice des caractères de A
variant de 0 à NL×P-1
.IB l'indice des caractères de B
variant de 0 à NL×P-1

(On passe de A à B par un réarrangement de leurs éléments)
.CR numéro de la règle en cours de construction
variant de 0 à NR-1

Le processus de transposition pourra se schématiser ainsi :

```
IB = 0

pour CR de 0 à NR-1
    création de la règle CR
    pour IA de CR à CR+NR×(NL-1) pas NR.
        faire B(IB) = A(IA)
        IB = IB + 1
```

La partie programmation de cette phase de transposition sera placée après la partie 'compactage' qui fournira une chaine B standardisée.

Les déclarations sont celles de la partie précédente plus

DECLARATIONS

B	REG	8	(les valeurs numériques seront celles qui proviennent de l'exemple entrée en CH)

TRANSPOSITIONS

LW,2	NR	
LI,3	0	(3) indice sur B
LI,4	0	

```
ETI2        LW,5        4           (5) indice sur A
            LW,1        NL

ETI1        LB,6        A,5         A,5 → B,3
            STB,6       B,3
            AI,3        1             incrémentation de (3) pas 1
            AW,5        NR            Incrémentation de (5) pas NR
            BDR,1       ETI1
            AI,4        1
            BDR,2       ETI2
```

EXEMPLE 1 La chaine

A = VFGGFVX---X---X représentant la table IV.1

deviendra après transposition :

B = VGX--FF-X-GV--X

EXEMPLE 2 La chaine

A = VFG␣F-F␣GFV␣1---2XXXX---X représentant la table IV.2

deviendra

B = VFG12-F-F-X-GFV-X-␣␣␣-XX

2.3 SEPARATION DES ENTRES "ACTIONS"

La chaine de caractères obtenue après transposition représente la table sous forme d'une suite de règles. Pour la traduction de la table (création de la structure équivalente) l'entrée des actions est inutile, elle n'interviendra qu'au niveau de la génération du texte objet.

Nous diviserons la table de façon à obtenir deux chaines de caractères :

- une chaine représentant l'entrée des conditions, la partie action étant remplacée par un numéro d'ordre. (la règle E éventuelle ne figurera pas dans cette chaine)
- une chaine représentant les entrées des actions, la partie condition étant remplacée par le numéro d'ordre correspondant. (La règle E éventuelle -toujours en fin de chaine- figurera avec le premier numéro d'ordre sous-correspondant dans la chaine précédente).

EXEMPLE A partir d'une chaine :

B = | R1 | R2 | R3 | R4 |

Nous nous proposons de créer :

BA | 1 | A1 | 2 | A2 | 3 | A3 | 4 | A4 | 5 |

suite des actions avec leurs numéros

BC | C1 | 1 | C2 | 2 | C3 | 3 | C4 | 4 |

suite des conditions avec leurs numéros

si R5 était une règle E
BC serait identique mais A5 serait ajouté à B5

BC | 1 | A1 | 2 | A2 | 3 | A3 | 4 | A4 | 5 | A5 |

NR étant le nombre total de règles
RE étant le nombres de règles E (1 ou 0)

La création des chaines BA et BC se fait
règle par règle pour les NR-RE premières règles
BA étant complétée si RE = 1

La fin de BC sera marquée EL
La fin de BA sera marquée L

Le programme peut être représenté par le schéma
suivant :

IA, IB, IC étant les indices sur les chaines
BA, B et BC

B Chaine en entrée

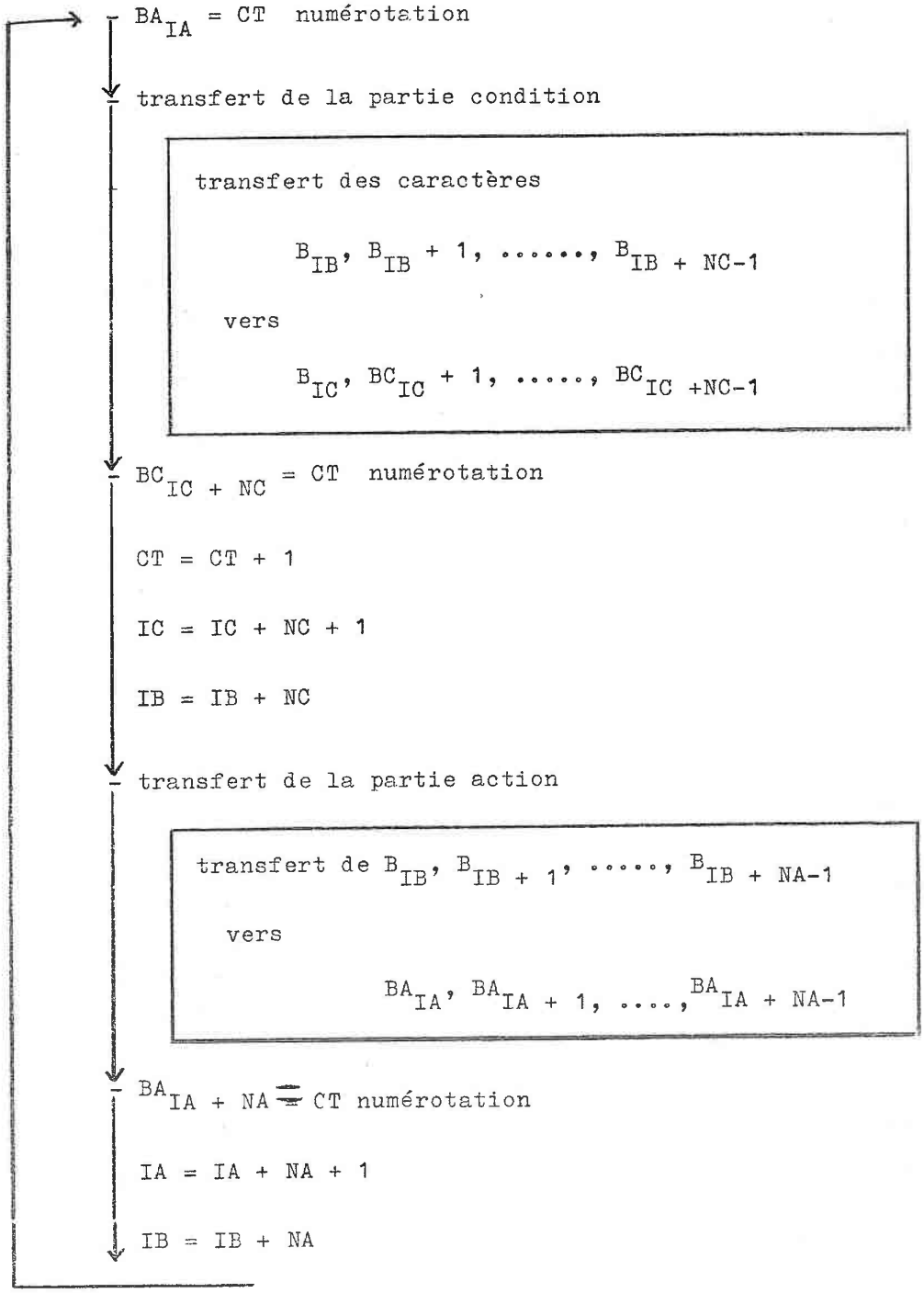
BA Chaine des actions

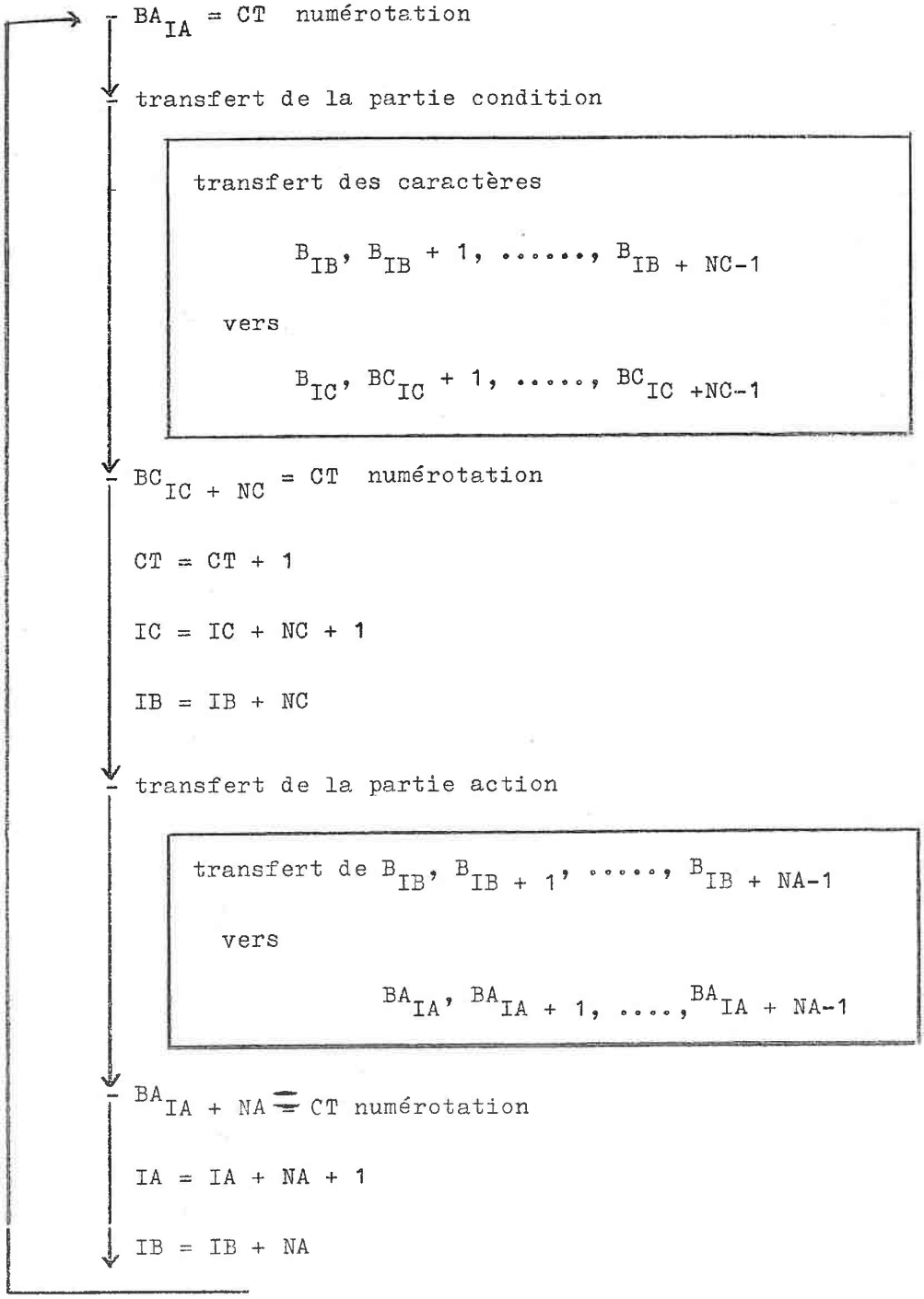
BC Chaine des conditions

CT compteur servant à la numérotation

CT = 1 IA, IB, IC = 0

faire (NR-RE) fois


BA_{IA} = CT numérotation
transfert de la partie condition
transfert des caractères
B_{IB}, B_{IB} + 1, ..., B_{IB} + NC-1
vers
B_{IC}, BC_{IC} + 1, ..., BC_{IC} +NC-1
$BC_{IC + NC}$ = CT numérotation
CT = CT + 1
IC = IC + NC + 1
IB = IB + NC
transfert de la partie action
transfert de B_{IB}, $B_{IB + 1}$, ..., $B_{IB + NA-1}$
vers
BA_{IA}, $BA_{IA + 1}$, ..., $BA_{IA + NA-1}$
$BA_{IA + NA}$ = CT numérotation
IA = IA + NA + 1
IB = IB + NA


BA_{IA} = CT numérotation
transfert de la partie condition
transfert des caractères
B_{IB}, B_{IB} + 1, ..., B_{IB} + NC-1
vers
B_{IC}, BC_{IC} + 1, ..., BC_{IC} +NC-1
$BC_{IC + NC}$ = CT numérotation
CT = CT + 1
IC = IC + NC + 1
IB = IB + NC
transfert de la partie action
transfert de B_{IB}, $B_{IB + 1}$, ..., $B_{IB + NA-1}$
vers
BA_{IA}, $BA_{IA + 1}$, ..., $BA_{IA + NA-1}$
$BA_{IA + NA}$ = CT numérotation
IA = IA + NA + 1
IB = IB + NA

fin d'itération

- marquage fin de BC

BC_{IC} = 'E', $BC_{IC + 1}$ = 'L'

si RE = 1 tranfert des actions de E

transfert de B_{IB}, $B_{IB + 1}$, ..., $B_{IB + NA - 1}$

vers

BA_{IA}, $BA_{IA + 1}$, ..., $BA_{IA + NA - 1}$

IA = IA + NA

- marquage fin de BA

BA_{IA} = 'L'

FIN de l'étape

REMARQUES sur le programme

BA et BC seront placés dans des zônes de mémoires BAX et BCX

BCX par la suite contiendra la pile des tables générées au cours de la traduction (BC en constitue l'initialisation)

Le remplissage de BCX se fera avec cadrage à droite ; pour simplifier les traitements, la position du premier élément non libre de BC sera mémorisée en ADBCX.

Déclarations et réservations.

DECLARATIONS.

Celles déjà faites aux étapes précédentes plus :

BAX	RES	6	Chaine des actions
BCX	RES	12	Chaine des conditions
REA	DATA	48	Longœur en octet de BCX
FIN	DATA	C'E000'	Marqueurs de fin de chaine
LIM	DATA	C'L000'	
ADBCX	RES	1	Adresse relative sur BCX
NUN	RES	1	NC + 1

Programme - SEPARATION

a) initialisation

- le registre (1) réservé à la numérotation est initialisé 1
  le premier numéro est alors placé sur BAX

- les registres (2) et (4), index conditions et actions sur la chaine B
  (2) initialisé BA(B) (adresse octet de B)
  (4) initialisé BA(B) + NC (adresse octet de la première action sur B)

- détermination de ADBCX
  ADBCX = longueur de BCX (longœur de la chaine condition à créer).

- les registres (3) et (5) index sur BCX et BAX
  (3) initialisé BA(BCX) + ADBCX
  (5) initialisé BA(BAX)

INITIALISATION.

```
LI,1        1           (1) initialisé à 1
STB,1       BAX         Première numérctation sur BAX

LI,2        BA(B)       (2) Adr. octet 1ère condition de B
LW,4        NC
AI,4        BA(B)       (4) Adr. octet 1ère action de B

LW,3        NC
AI,3        1           NUN = NC + 1
STW,3       NUN

LW,8        NR          (8) = NR - RE
SW,8        RE

LW,5        8           (5) = (NR - RE) × (NC + 1) + 2
MW,5        3           Longueur de BC

AI,5        2

LW,3        REA
SW,3        5
STN,3       ADBCX

AI,5        BA(BCX)     (5) Adr. octet sur BCX
LI,5        BA(BAX)
AI,5        1           (5) adr. octet sur BAX
```

LW,6 NC

LW,7 NA

b) transfert vers les chaines actions et conditions

Phase itérative.

On a :

- changement des registres (3) et (5) (en vue des MBS)
- transfert de la partie condition d'une règle.
- numérotation, incrémentation (1)
- transfert de la partie action d'une règle
- numérotation.

Si (1) $\leqslant$ (3) = nombre de règles (autres que E)
pousuite de l'itération.

TRANSFERTS

ETI 3	STB,6	3	
	STB,7	5	
	MBS,2	0	Transfert partie condition
	STB,1	0,3	Numérotation
	AI,3	1	
	AI,1	1	INcrémentation de (1)

```
        MB,4        0           transfert partie action
        STB,1       0,5         Numérotation
        AI,5        1

        AW,2        NA
        AW,4        NC

        CW,3        1           Retour si il reste des
        BLE         ETI3        règles à traiter.
```

c) <u>marquages de fin et transfert de la règle E</u>

- la chaine BC est bornée par E puis L
- si la règle E existe, transfert de la partie action puis marquage de fin par L.

MARQUAGE

```
        LB,2        FIN
        STB,2       0,3         Marqués 'E'
        AI,3        1           et 'I'
        LB,2        LIM
        STB,2       0,3

        LW,1        RE
        CW,1        ZER
        BE          ESI2
        MBS,4       0           transfert action de la règle E
ESI2    STB,2       0.5         marque 'L'
```

EXEMPLE 1

La chaine

B = VGX--FF-X-GV--X

représentant la table IV.1 subit le processus décrit page IV.23

Au départ BA, BC sont vides

1ère itération Règle 1	BC = VG	Transfert conditions 1ère règle
	BC = VG 1	Numérotation
	BA = 1X--	Transfert actions 1ère règle
	BA = 1X--2	Numérotation
2ème itération Règle 2	BC = VG1FF	
	BC = VG1FF2	
	BA = 1X--2-X-	
	BA = 1X--2-X-3	
3ème itération Règle 3	BC = VG1FF2GV	
	BC = VG1FF2GV3	
	BA = 1X--2-X-3--X	
	BA = 1X--2-X-3--4	
Marquage de fin.	BC = VG1FF2GV3EL	
	BA = 1X--2-X-3--X4L	

EXEMPLE 2

La chaine

B = VFG12-F-F-X-GFV-X-□□□-XX

représentant la table IV.2 donnera

BA = 112-2-X-3-X-4-XX

BC = VFG1F-F2GFV3

3. PHASES ITERATIVES DE LA TRADUCTION.

La phase précédente constituait une initialisation de la procédure itérative de traduction.

Nous avons vu que l'algorithme procède par division de la table en sous-tables qui sont stockées sur une pile (BCX étant la zone réservée à cette pile et BC sa valeur initiale)

Chaque itération comportera deux phases importantes :

- partant de la table placée en sommet de pile, rechercher la condition sur laquelle se fera la division.
- division de la table en deux sous-tables et la replacer au sommet de la pile par ces deux sous-tables.

La construction de l'arborescence résultante se fait par stockage des conditions après la phase de recherche et par stockage des actions au moment où les tables correspondantes, complètement résolues, apparaissent en sommet de pile.

3.1. Recherche de la condition à tester.

- une phase préliminaire de prise en charge de la première table de la pile.
- il faut ensuite s'assurer que la table a encore des conditions à déterminer (les autres cas seront traités ensuite)
- on passe ensuite à la phase de recherche de la condition adéquate en choisissant la condition qui a le moins d'entrées neutres (-, X, +F, +V). La condition ainsi trouvée est placée sur la chaine descriptive de l'arborescence

Le processus pourra être représenté par le schéma suivant :

Prise en charge de la première table

. Initialisation

Pour chaque caractère de BC jusqu'à la marque FIN

Comptage des caractères (cumul) Recherche des entrées V ou F (positionnement d'une variable TEST)

Le cumul donne LO longueur de la table en octet

d'où LON = LO/(NC + 1) longueur en nombre de règle.

. Table vide

Si LO = 0 table vide traitement ultérieur.

• Table neutre

si TEST = 0 pas d'entrées V ou F
table complètement résolue
traitement ultérieur

↓

recherche de la condition optimale

pour chaque condition, recherche le nombre d'entrées neutres.

Recherche condition

(pour IC variant de 0 à NC - 1)

Parmi les LON caractères de la ligne condition IC

BC_{IC}, $BC_{IC} + (NC + 1)$, ..., $BC_{IC} + (NC + 1) \times (LON - 1)$

soit C le nombre d'entrées neutres

si C est inférieur aux précédents
conserver C et l'indice IC

↓

stockage de la condition trouvée (de numéro IC + 1) sur la chaine descriptive DESTAB

Déclarations et réservations

Toutes celle déjà faites plus :

LO	RES	1	Longueur en octet de la table traitée.

```
LON       RES    1      Longueur en nombre de règle
                        de la table traitée.
K         RES    1
C         RES    1      Zones de travail, recherche
                        du nombre minimum d'entrée neutres
CI        RES    1
I         RES    1
J         RES    1
DESTAB    RES    8      Chaine de description de l'ar-
                        borescence
ID        DATA   0      Indice sur DESTAB initialisé à
                        zéro
```

Programme

RECHERCHE CONDITION

Initialisation

```
DEPIT     LI,1   0      Début de l'itération
          LI,2   1      (1) compte des caractères
          LW,3   ADBCX  (3) adresse du début de la table
          AI,3   BA(BCX)

ETI7      LB,4   0,3
          CB,4   F
          BE     ETI4
          CB,4   V
          BNE    ETI5

ETI4      LI,2   0      (2) positionné à zéro si on
                        trouve des entrées V ou F
                        (TEST)
```

```
ETI5        CB,4        FIN
            BE          ETI6        Si le caractère considéré
            AI,1        1           n'est pas la marques FIN
            AI,3        1           reprendre à ETI7
```

TABLE VIDE

```
ETI6        CW,1        ZERO        Table vide
            BE          ETI8        Traitement à ETI8
```

TABLE NEUTRE

```
            CW,2        ZERO        Table dont toutes les conditions
            BNE         ETI9        ont déjà été testées
                                    Traitement à ETI9
```

RECHERCHE CONDITION

```
            LW,5        NUN
            STW,1       7           Détermination de LO et LON
            DW,7        5
            STW,1       LO
            STW,7       LON
            STW,7       K           Initialisation de K par LON
            STW,2       CI          Initialisation de I et CI par 0
            STW,2       I
```

```
ETI14   LI,7    0       Début de la phase de recherche
        STW,7   C       de la condition
        STW,7   J

ETI12   LI,3    BA(BCX)
        AW,3    ADBCX
        AW,3    I       Calcul indice des caractères de
        LW,5    NUN     la ligne I (LON éléments)
        MW,5    J
        AW,3    5

        LB,6    0,3     Si caractère 'X', condition déjà
        CB,6    XM      testée voir ligne suivante
        BE      ETI13
        CB,6    TI      Si caractère '-', aller à ESI9
        BE,6    ESI9
        CB,6    PF      Si caractère '+F' aller à ESI9
        BE      ESI9
        CB,6    PV      Si caractère '+V' aller à ESI9
        BNE     ETI11

ESI9    MTW,1   C       Cumul des entrées neutres

ETI11   MTW,1   J
        LW,6    J       Si J < LON, retour à ETI12
        CW,6    LON     Si J ≥ LON, fin traitement d'une ligne
        BL      ETI12
        LW,6    C
        CW,6    K
```

```
                BGE         ETI13       Si C < K alors :
                STW,6       K           K=C (K = inf. des C déjà déterminés
                LW,6        I           CI=I (stockage du numéro de la con-
                STW,6       CI                dition correspondant à K)

ETI13           MIW,1       I           Retour à ETI14
                LW,6        I           pour traitement de la ligne
                CW,6        NC          suivante si I+1  NC
                BL          ETI14
                                        Fin de la phase de recherche
                LW,6        CI          CI indice de la condition recherchée
                AI,6        1
                LW,4        ID          Indice sur DESTAB
                STB,6       DESTAB4     Stockage du numéro de condition
                AI,4        1           trouvé, sur DESTAB
                STW,4       ID
```

EXEMPLE

Soit en entrée, la chaine BC représentant la table IV.2

BC = VFG1F-F2GFV3EL

on aura :

INITIALISATION

LO = 12 — BC a 12 caractères utiles

TEST = 1 — il existe des entrées V ou F sur BC

LON = 3

TABLE VIDE

Non, LO ≠ 0

TABLE NEUTRE
Non, TEST $\neq$ 0

1 - RECHERCHE CONDITION
1ère ligne condition
(caractère 0, 4, 8)
1 entrée neutre d'où C = 1 et IC = 0 (indice de la ligne)

2 - RECHERCHE CONDITION
2ème ligne condition
(caractères 1, 5, 9)
1 entrée neutre (pas inférieure à C)

3 - RECHERCHE CONDITION
3ème ligne condition
(caractères 2, 6, 10)
1 entrée neutre (pas inférieure à C)

Donc

STOCKAGE de la condition numéro 1 (= IC +1)
On aura DESTAB = 1

3.2. Division endeux sous-tables

Partant d'une table extraite du sommet de la pile BCX (lors de la première itération, il s'agit de la table entrée dans le traducteur), nous avons déterminé la condition par laquelle devait se poursuivre le processus d'analyse.

Il s'agit maintenant de diviser cette table en deux tables plus simples qui la remplaceront sur la pile.

La première sous-rable est créée par regroupement des règles dont l'entrée pour la condition testée est 'F', 'G', '-'.

La seconde est obtenue à partir des règles dont cette entrée est 'V', 'W', '-'.

Dans ces tables, la condition testée pour la division n'a plus à figurer ; afin de conserver le format des règles, les entrées ne seront pas supprimées mais marquées 'X'.

Soient

- BCX1, BCX2 les zones où seront créées les deux sous-tables résultant de la division
- LO1, LO2 longueurs en nombre de caractères de ces deux tables.

CI étant la condition à tester

RES zone de stockage 1 caractère.

Le processus de division-création peut alors être schématisé ainsi : LO1, LO2 = 0

Pour chaque règle de la table d'origine
(pour J variant de 0 à LON)

POSITIONNEMENT

- Recherche de l'entrée de la condition CI dans la règle concernée

  Stockage dans RES
  $(RES = BCX_{CI+J\times(NC+1)})$

- l'entrée trouvée est marquée 'X'
  $(BCX_{CI+J\times(NC+1)} = 'X')$

- détermination de la division

DIVISION

RES = V ou W	V	+F	F
RES = F ou G	+F	V	F
Recopie dela règle surBCX2-A1	X	-	X
Recopie dela règle surBCX2-A2	-	X	X
J = J + 1	X	X	X

STOCKAGE

- les deux tables obtenues sont stockées sur la pile BCX
  après suppression de la table d'origine.
  . retrait de la table d'origine
    (l'adresse du sommet de la pile est décalé de LO, longueur
    de la table traitée ; BCX désigne le sommet de cette pile)
  . marquage de fin de table (E) sur BCX1
    et transfert de BCX1 sur la pile BCX
    pour I de 0 à LO1

transfert de $BCX1_I$ sur BCX_I

  . transfert de BCX2 sur lapile BCX
    pour I de 0 à LO2-1

transfert de $BCX2_I$ sur BCX_{LO1+I}

Action A1

Transfert de la règle J sur BCX1
LO1 longueur de BCX1 = indice du 1er caractère libre sur BCX1
pour I de 0 à NC

transfert du caractère $BCX_{I+J\times(NC+1)}$ sur $BCX1_{LO1+I}$

LO1 = LO1+NC+1

Action A2

Transfert de la règle J sur BCX2

LO2 longueur de BCX2 = indice du 1er caractère libre sur BCX2

pour I, de 0 à NC

Transfert du caractère $BCX_{I+J\times(NC+1)}$ sur $BCX2_{LO2+I}$

LO2 = LO2+NC+1

REMARQUE.-

La règle 3 de la table précédente correspond bien au cas où RES = '-' car les seules valeurs possibles de RES sont F, G, V, W, -.

L'entrée des conditions aurait peut être correspondant à la règle E

On aurait eu :

RES = V ou W	V	+F	
RES = F ou G	+F	V	
Action A1	X	-	X
Action A2	-	X	X
J = J + 1	X	X	X

DECLARATIONS ET RESERVATIONS

```
BCX1        RES         6
BCX2        RES         6
LO1         RES         1
LO2         RES         1
```

Programme - DIVISION

INITIALISATION

```
            LI,6        0
            STW,6       J
            STW,6       LO1
            STW,6       LO2
```

POSITIONNEMENT

```
ETI18       LW,3        NUN
            MW,3        J           Calcul de l'indice du
            AI,3        BA(BCX)     caractère à considérer (RES)
            AW,3        ADBCX
            LW,7        CI
            AW,7        3
            LB,6        0,7         (6) stockage de RES
            LB,5        XM
            STB,5       0,7         Marquage de la position à
                                    'X' sur la chaine
```

DIVISION

```
        CB,6        F
        BE          ESI7        Si (6) = F ou +F aller à ESI7
        CB,6        PF
        BNE         ETI15
```

A1 - TRANSFERT SUR BCX1 (Cas d'entrées F ou +F)

```
ESI7    LW,4        3
        LI,5        BA(BCx1)
        AW,5        LO1         Préparation du transfert
        LW,6        NUN
        STB,6       5
        LW,6        LO1         Calcul de LO1 après transfert
        AW,6        NUN
        STW,6       LO1
        MBS,4       0           Transfert d'une règle sur BCX1
        B           ETI16       (règle d'indice J)

ETI15   CB,6        V           Si (6) = V ou +V aller à ESI8
        BE          ESI8
        CB,6        PV
        BNE         ETI17
```

A2 - TRANSFERT SUR BCX2 (Cas d'entrées V ou +V)

```
ESI8    LW,4        3
        LI,5        BA(BCX2)    Préparation du transfert
        AW,5        LO2
```

```
                LW,6        NUN
                STB,6       5
                LW,6        LO2
                AW,6        NUN         Progression de LO2
                STW,6       LO2
                MBS,4       0
                B           ESTI16
```

A1-A2 - TRANSFERT SUR BCX1 et BCX2 (Cas d'entrée '-')

```
ETI17           LW,4        3
                LI,5        BA(BCX1)    RES = '-'
                AW,5        LO1
                LW,6        NUN         Préparation du transfert
                STB,6       5
                LW,6        LO1         Programmation de LO1
                AW,6        NUN
                STW,6       LO1
                MBS,4       0           Transfert sur BCX1
                LW,4        3
                LI,5        BA(BCX2)
                AW,5        LO2
                LW,6        NUN         Préparation du transfert
                STB,6       5
                LW,6        LO2         Progression de LO2
                AW,6        NUN
                STW,6       LO2
                MBS,4       0           Transfert sur BCX2
```

TEST FIN DE BOUCLE

```
ETI16   MTW,1    J       Si (4) = indice sur BCX
        CW,4     LON     inférieur à LON, retour
        BL       ETI18   à ETI18
```

STOCKAGE

```
        LI,5     BA(BCX1)
        AW,5     LO1
        LB,6     FIN     M arquage de fin de BCX1
        STB,6    0,5
        LW,5     ADBCX   Calcul de l'indice du sommet
        AW,5     LO      de pile BCX en supprimant la
        SW,5     LO2     table d'origine
        AI,5     BA(BCX)
        LW,6     LO2     Préparation du transfert
        STB,6    5
        LI,4     BA(BCX2)
        MBS,4    0       Transfert de la sous-table 2
                         (entrées V, +V) sur BCX

        SW,5     LO2
        SW,5     LO1
        SW,5     UN
        CI,5     BA(BCX)
        BL       EIT20

        LW,6     LO1
        AI,6     1       Préparation du transfert
        STB,6    5
        LI,4     BA(BCX1)
        MBS,4    0       transfert de la sous-table 1
                         (entrées F, +F) en tête de BCX
```

```
SW,5        LO1
SW,5        UN          Calcul de ADBCX
LI,6        BA(BCX)     adresse du sommet de BCX
SW,5        6
STW,5       ADBCX
B           DEPIT       Retour au début de la phase
                        itérative.
```

EXEMPLE

Reprenons la chaine BCX représentant la table IV.2

BCX = VFG1F-F2GFV3EL

La condition choisie est la condition 1 (cf pas IV.)

Effectuons maintenant le processus de division en deux tables

1ère règle

POSITIONNEMENT

RES = V entrée de C1 dans la 1ère règle

V est remplacé par X sur BCX

BCX = XFG1F-F2GFV3EL

DIVISION

RES = V entraine

BCX2 = XFG1

LO2 = 4

2ème règle

POSITIONNEMENT

RES = F entrée de C1 dans la 2ème règle

F est remplacé par X sur BCX

BCX = XFG1X-F2GFV3EL

DIVISION

RES = F entraine

BCX1 = X-F2

LO1 = 4

3ème règle

POSITIONNEMENT

RES = G entrée C1 dans la 3ème règle

G remplacé par X sur BCX

BCX = XFG1X-F2XFV3EL

DIVISION

RES = G entraine

BCX1 = X-F2XFV3

LO1 = 8

STOCKAGE

Suppression de la table d'origine sur BCX d'où

BCX = EL

Marquage FIN sur BCX

BCX1 = X-F2XFV3E

Transfert de BCX2 sur BCX

BCX = XFG1EL

Transfert de BCX1 sur BCX

BCX = X-F2XFV3EXFG1EL

L'itération portera sur la table placée en tête de pile soit : X-F2XFV3

REMARQUE.-

BCX1 et BCX2 résultant de la division de la table IV.2 à partir de C1 représentent les tables.

C1	X	X
C2	-	F
C3	F	V
	A2	A3

C1	X
C2	F
C3	+F
	A1

3.3. Traitement dans le cas d'itération sur une table vide

Nous sommes dans le cas où la table placée en sommet de pile est vide (le premier caractère rencontré à la prise en charge est la marque fin de table 'E'.

Une table vide correspond à une entrée non prévue dans la table ; on exécutera alors l'action de la règle E si elle existe, sinon une action standard.

EXEMPLE.- Soit à diviser, à partir de la règle C1 la table IV.3

C1	V	V
C2	V	F
	1	2

Placée en sommet de pile
sous la forme VV1VF2E....
on a DESTAB = ...1
la division donne :
BCX1 = E
BCX2 = XV1XF2
d'où le sommet de pile BCX
EXV1XF2.....

La table en sommet de pile est vide ; l'action à placer sur DESTAB sera l'action correspondant au cas C1 = F
donc l'action E sera notée FF
d'où DESTAB = ...1FF
puis la table vide sera retirée de la pile, l'itération ultérieure portant sur
XVIXF2

On doit donc effectuer les opérations suivantes :
RE = 1 si il existe une règle E
= 0 sinon
si RE = 1
placer sur DESTAB le nombre NC (NC = nombre de règles de la table en entrée, donc égal numéro de l'ensemble action sur la règle E)
si RE = 0
placer sur DESTAB le nombre 'FF' réservé à l'action standard en cas d'entrée non prévue.

REMARQUE Deux sortes de codes figureront sur la chaine DESTAB :

1 - des nombres correspondant à des numéros de conditions

2 - des nombres correspondant à des numéros de règles (numéros d'ensembles d'actions)

De façon à les distinguer, tout en limitant leur longueur à 1 octet, on adoptera la convention suivante :

- les nombres de 0 à 127 représentent des numéros de conditions
- les nombres de 128 à 254 représentant des numéros d'actions (sous la forme : numéro réel + 128)

  Le nombre 255 ('FF') étant réservé à l'action standard en cas d'entrées non prévues.

REMARQUE.

Les caractères définis ici limitent la taille de la table entrée à 128 conditions et 128 règles. Ce qui semble pratiquement sans inconvénient.

DECLARATIONS ET RESERVATIONS

En plus de celles déjà effectuées, nous aurons :

```
COMP    DATA    X'000000080'
                nombre à ajouter aux numéros d'action
                avant stockage.

AE      DATA    X'FF000000'
                 numéro de l'action standard pour
                entrées non prévues.
```

TABLE VIDE

```
ETI8    LW,6    RE
        CW,6    UN          Si RE = 1
        BNE     ESI5        (6) = NC + 80(16)
        LW,6    NC
        AW,6    COMP
        B       ESI6
```

```
ESI5    LB,6     AE          (6) = AE

ESI6    LW,4     ID
        STB,6    DESTAB4     Stockage de (6) sur DESTAB
        AI,4     1
        STW,4    ID
        LW,6     ADBCX       Effacement de la table vide
        AI,6     1           sur BCX pour déplacement du
        STW,6    ADBCX       sommet de pile
        B        ETI19

ETI19   LI,6     BA(BCX)     Si la pile n'est pas vide
        AW,6     ADBCX       (caractère au sommet ≠ L)
        LB,4     0,6         retour au début de la phase
        CB,4     LIM         itérative.
        BNE      DEPIT
```

3.4. Itération sur une table complètement testée
(table ne comportant pas d'entrée V ou F)

Nous sommes dans le cas décrit page IV.33 où la table placée en tête de pile n'est pas vide mais où toutes les entrées sont neutres (LO ≠ 0, TEST = 0)

EXEMPLE Reprenons l'exemple de la page IV.48 table IV.3)

Après la phase de traitement de la table vide, nous avons :

DESTAB = ...01FF

pile BCX = XV1XF2E....

L'itération suivante donnera :

Division à partir de la condition 2

d'où DESTAB = ...01FF02

BCX = XX2EXXIE...

Nous arrivons donc à l'itération prenant en charge XX2
Toutes les conditions sont testées donc :

- stockage de l'action concernée (2)
- sur DESTAB
  Retrait de XX2 de la pile
- poursuite de la traduction
  traitement de XX1 à l'itération suivante

d'où ceci : DESTAB = ...01FF0282
BCX = XXIE...

REMARQUE.

Si la table considérée comporte encore plusieurs règles (règles ambigues), tous les ensembles d'actions concernés sont placés sur DESTAB séparés par '-' et un libellé AMBIGUITES ENTRE XX..., est édité.

Le traitement pourra être représenté schématiquement ainsi :

- stockage dernier ensemble d'action de la table
- si la table comporte plusieurs règles alors :
- stockage de l'ensemble d'actions sur la zone d'édition
- si adresse de l'action précédente supérieure à adresse sommet de la pile, sortie boucle
- stockage de '-' sur DESTAB
  stockage de l'ensemble action sur DESTAB

édition du libellé
suppression de la table sur la pile
retour au début de la phase itérative si il reste des tables sur la pile.

DECLARATIONS ET RESERVATIONS

Aux précédentes, on ajoute :

```
ET          TEST        C' AMBIGUITE ENTREE
ERR         RES         2      zone d'édition
ERI         DATA        X'FOFOFOFO'
FCOD        DATA        X'000000FO'
COMP        DATA        X'00000080'
```

PROGRAMME - TABLE NEUTRE

```
ETI9        LW,5        ERI
            STD,5       ERR
            SW,3        UN
            LB,5        0,3
            AW,5        COMP        Stockage de l'action
            LW,4        ID
            STB,5       DESTAB,4
            AI,4        1
            STW,4       ID          Si longueur de la table
            CW,1        NUN         NC + 1 aller à ESI3
            BLE         ESI3
            LI,5        0

ESI4        LB,4        0,3         Stockage action sur
            AW,4        FCOD        zone édition
            STB,4       ERR,5
            SW,3        NUN         (3) division de NCH
                                    adresse du numéro action précédent
```

```
            AI,5        1
            LI,2        BA(BCX)
            AW,2        ADBCX       Si (3) adresse du sommet de
            CW,3        2           pile, aller à EVI1
            BL          EVI1

            LB,2        TI
            LW,4        ID          Stockage
            STB,2       DESTAB,4    puis numérotion d'action
            AI,4        1           sur DESTAB
            LB,6        0,3
            AW,6        COMP
            STB,6       DESTAB,4
            AI,4        1
            STW,4       ID
            B           ESI4

EVI1        M:WRITE     M:LO, (BUF,ET), (SIZE,24)
                          édition du libellé ambiguité

ESI3        AW,1        ADBCX       Modification adresse sommet de pile
            AI,1        1           (Suppression de la table)
            STW,1       ADBCX

ETI19       LI,6        BA(BCX)
            AW,6        ADBCX       si le sommet de pile est ≠ L
            LB,4        0,6         pousuite de la traduction
            CB,4        LIM
            BNE         DEFIT
```

4. GENERATION DU PROGRAMME OBJET.

La phase principale de la traduction est terminée avec la génération d'une arborescence décrite en notation préfixée par la chaine de caractère DESTAB.

La dernière étape, beaucoup moins complexe, est constituée par la génération d'un texte à programme accepté par l'ordinateur.

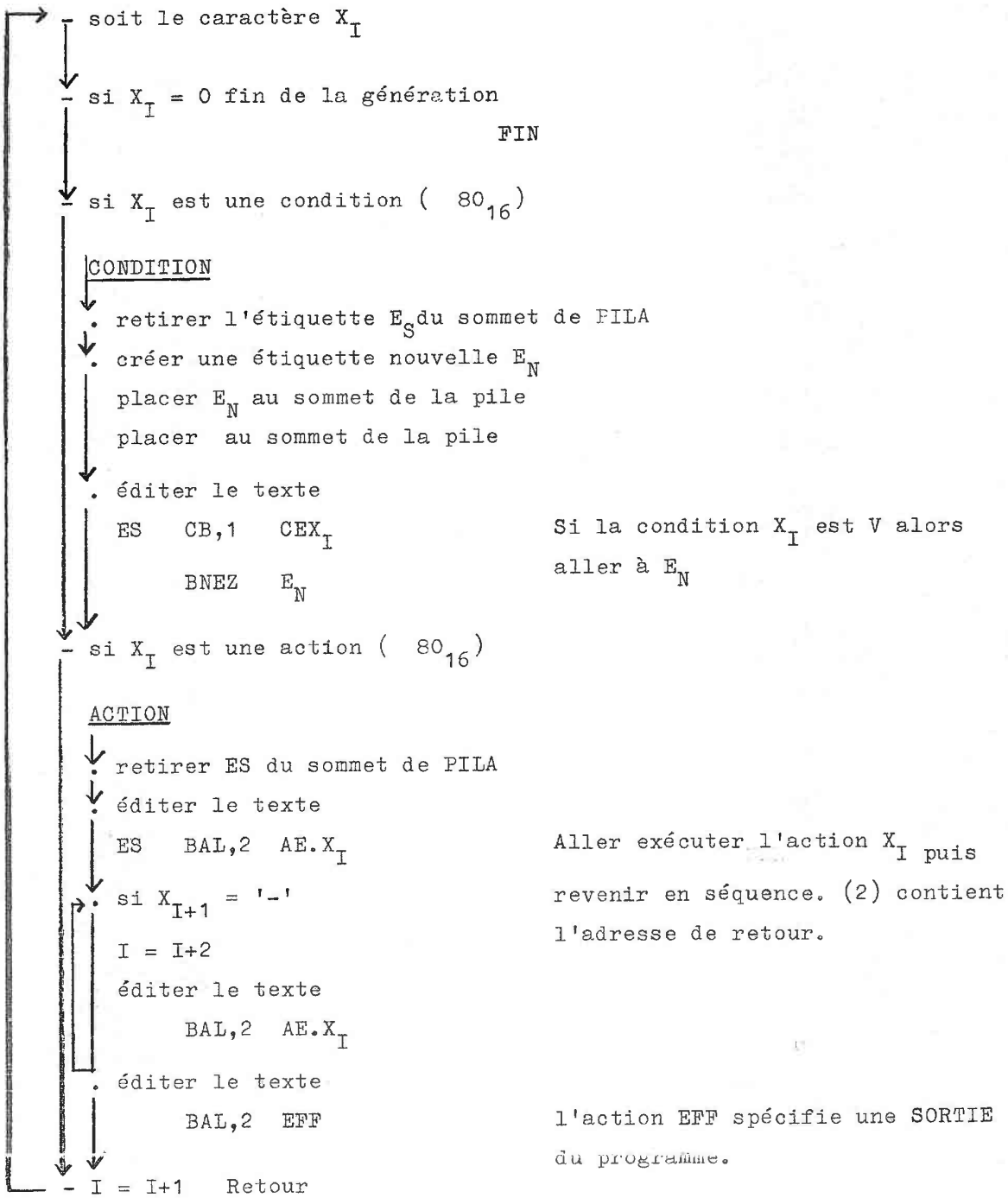
Nous décrirons ici un algorithme et un programme qui, à partir de l'arborescence DESTAB, nous fournira un texte de programme écrit en langage 'Symbol CII 10070'

L'algorithme, basé sur la gestion d'une pile d'étiquettes aura la structure suivante :

Soient :

- PILA la pile d'étiquettes utilisées : initialement
  PILA =

- I indice sur DESTAB
  X_I caractère d'indice I sur DESTAB (I initialisé à 0)


soit le caractère X_I
si X_I = 0 fin de la génération
FIN
si X_I est une condition (80_{16})
CONDITION
retirer l'étiquette E_S du sommet de PILA
créer une étiquette nouvelle E_N
placer E_N au sommet de la pile
placer au sommet de la pile
éditer le texte
ES CB,1 CEX_I
BNEZ E_N
Si la condition X_I est V alors aller à E_N
si X_I est une action (80_{16})
ACTION
retirer ES du sommet de PILA
éditer le texte
ES BAL,2 $AE.X_I$
Aller exécuter l'action X_I puis revenir en séquence. (2) contient l'adresse de retour.
si X_{I+1} = '-'
I = I+2
éditer le texte
BAL,2 $AE.X_I$
éditer le texte
BAL,2 EFF
l'action EFF spécifie une SORTIE du programme.
I = I+1 Retour

DECLARATIONS ET RESERVATIONS

A celles déjà faites, nous ajouterons :

GEN	TEXT	C'	Longueur 5 mots zone réservée à l'édition
ET	TEXT	C'EE00'	Préfixe pour la création d'étiquettes
CT	TEXT	C'CE00'	Préfixe pour l'édition de condition
AT	DATA	X'C1C5F070'	Préfixe pour l'édition d'action sans retrait de 80_{16}
BN	TEXT	C'BNEL'	texte à éditer
BR	TEXT	C'BAL,1	
REG	TEXT	C'2 '	
CB	TEXT	C'CB,1'	
PILA	DATA	$\not{0}$ + 2	
	DATA	X'000A0000'	Description et réservation de la pile
	RES	10	
TIR	DATA	X'00000060'	
EFIN	TEXT	C'EFFL'	Etiquette de sortie du programme
BLM	TEXT	C' '	

GENERATION

LW,4	ZERO	
LW,5	UN	
LW,6	BLM	
PSW,6	PILA	Placer blanc en tête de pile

```
ETI30     LB,6        DESTAB,4
          CB,6        ZERO
          BE          ETI31       Si (6) = 0  FIN
          CW,6        COMP
          BGE         ETI32       Si (6) ⩾ 80₁₆ → ETI32
```

CONDITION

```
          AW,6        CT
          STW,6       GEN+4
          PLW,6       PILA        Préparation de la 1ère ligne
          STW,6       GEN         d'édition.
          LW,6        CB
          STW,6       GEN+2
          LW,6        BLM
          STW,6       GEN+1
          STW,6       GEN+3
          M:WRITE, M:LO,(BUF,GEN),(SIZE,20)  édition

          STW,6       GEN
          LW,6        BN
          STW,6       GEN+2       Préparation de la 2ème ligne
          LW,6        ET          d'édition.
          AW,6        5
          STW,6       GEN+4
          M:WRITE, M:LO,(BUF,GEN),(SIZE,20)  édition

          PSW,6       PILA        Etiquette sur PILA
          LW,6        BLM
          PSW,6       PILA          sur PILA
          AI,5        1
          AI,4        1
          B           ETI30
```

ACTION

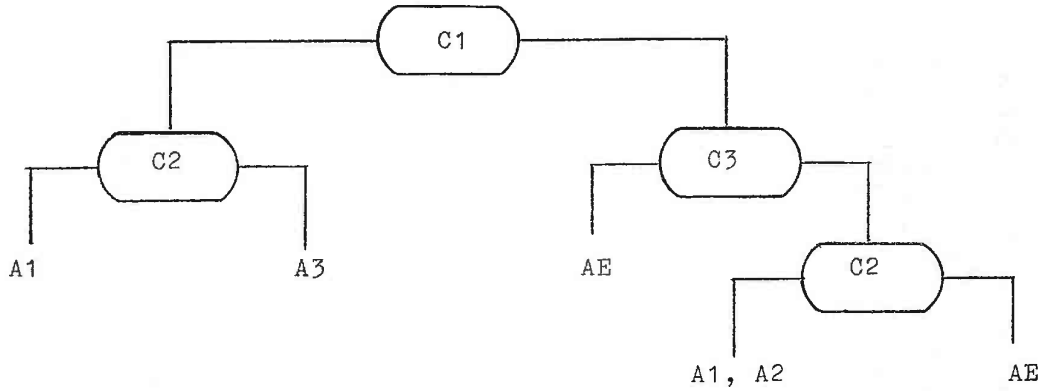
```
ETI32   AW,6        AT
        STW,6       GEN+4
        LW,6        BR          Préparation de la ligne d'édition
        STW,6       GEN+2
        LW,6        REG2
        STW,6       GEN+3
        PLW,6       PILA
        STW,6       GEN
        M:WRITE M:LO,(BUF,GEN),(SIZE,20)  édition
        AI,4        1

ETI33   LB,6        DESTAB,4
        CW,6        TIR
        BNE         ETI34       si caractère '-', préparation
        AI,4        1           d'édition
        LB,6        DESTAB,4
        AW,6        AT
        STW,6       GEN+4
        LW,6        BLM
        STW,6       GEN
        M:WRITE M:LO,(BUF,GEN),(SIZE,20)  édition
        AI,4        1
        B           ETI33

ETI34   LW,6        BLM
        STW,6       GEN
        LW,6        EFIN
        STW,6       GEN+4
        M:WRITE M:LO,(BUF,GEN),(SIZE,20)
        B           ETI30
```

EXEMPLE.

Soit DESTAB = 0102818303FF02816082FF00 (notation Hexadécimale)
qui sera plus clairement écrit :

DESTAB = C1C2A1A3C3AEC2A1-A2AE0

et qui représente l'arborescence suivante :



Le processus de génération sera le suivant :

CONDITION

X1 = C1

Pile initialisée ␣
Retrait de ␣
Création PILA = E1
Création PILA = E1␣
Edition
CB,1 C1
BNEZ E1

```
CONDITION

   X2 = C2                    Retrait de ␣
                              Création PILA = E1E2
                              Création PILA = E1E2␣

                Edition                 CB,1  C2
                                        BNEZ  E2

ACTION

   X3 = A1                    retrait de ␣

                Edition                 B     A1
                                        B     FIN

ACTION

   X4 = A3                    retrait E2

                Edition    E2           B     A3
                                        B     FIN

CONDITION

   X5 = C3                    retrait E1
                              création E3  PILA = E3
                                           PILA = E3␣

                Edition    E1           CB,1  C3
                                        BNEZ  E3
```

```
ACTION

   X6 = AE              Retrait de ␣

        Edition           B      AE
                          B      FIN

CONDITION

   X7 = C2              retrait E3
                        création E4  PILA = E4␣

        Edition   E3      CB,1   C2
                          BNEZ   E4

ACTION

   X8 = A1              retrait ␣

        Edition           B      A1

   X9 = -

   X10 = A2   Edition     B      A2

   X11  = -   Edition     B      FIN

ACTION

   X11 = AE             retrait E4

        Edition   E4      B      AE
                          B      FIN
```

D'où, en regroupant les termes, le texte suivant :

```
          CB,1        CEO1
          BNE2        EEO1
          CB,1        CEO2
          BNE2        EEO2
          BAL,2       AEO1
          BAL,2       EFF

EEO2      BAL,2       AEO3
          BAL,2       EFF

EEO1      CB,1        CEO3
          BNEZ        EEO3
          BAL,2       AEFF
          BAL,2       EFF

EEO3      CB,1        CEO2
          BNEZ        EEO4
          BAL,2       AEO1
          BAL,2       AEO2
          BAL,2       EFF

EEO4      BAL,2       AEFF
          BAL,2       EFF
```

CHAPITRE V

OPTIMISATION

1. GENERALITES

. Au cours des chapitres précédents, nous avons étudié quelques méthodes de traduction puis décrit la réalisation d'un traducteur.

L'objectif principal était alors d'obtenir un texte objet de façon aussi simple que possible par une traduction rapide et facile à mettre en oeuvre. Cependant, la rapidité du traducteur et sa complexité relative ne sont pas en général des éléments déterminants et il faut surtout définir un traducteur qui fournisse un programme performant, c'est-à-dire un programme peu encombrant et rapide.

Nous allons d'abord essayer de définir les qualités et les limites d'un programme obtenu à partir d'une table de décision, puis nous proposerons quelques algorithmes permettant une traduction efficace.

2. ETUDE QUANTITATIVE DES TABLES A ENTREES LIMITEES NON AMBIGUES SANS REGLE E.

2.1. Nombre moyen de tests.

Soit une table de décision T à n conditions et p règles
Soit NTp le nombre moyen de tests nécessaires à l'analyse complète de la table pour une traduction déterminée de la table, décrite par un programme P.

A partir de la table donnée T, nous allons essayer de définir des limites logiques pour NT.

a) - T a n conditions : on en déduit immédiatement que le nombre maximum de tests nécessaires à l'analyse complète de la table est n

Donc $NT_P \leqslant n \quad , \forall P$

b) - Dans de nombreux cas, il existe des règles pour lesquelles il est inutile de tester certaines conditions (entrées -,+F,+V)

Soit $\delta_{ij} = 1$ si l'entrée de la condition i dans la règle j est V ou F, = 0 sinon

Le nombre minimum de tests nécessaires pour sélecter la règle R_j sera alors :

$$NTM_j = \sum_{i=1}^{n} \delta_{ij}$$

Si f_j fréquence relative à la règle R_j

La moyenne du nombre minimum de tests nécessaires à l'analyse de la table sera alors :

$$NTM = \sum_{j=1}^{p} NTM_j$$

$$NTM = \sum_{j=1}^{p} \left(\sum_{i=1}^{n} \delta_{ij} \right) f_j$$

qui s'écrit également :

$$NTM = \sum_{\substack{j=1 \text{ à } p \\ i=1 \text{ à } n}} \delta_{ij} f_j$$

D'autre part,

$$NTM \leqslant NT_P \leqslant n \quad , \forall P$$

REMARQUE 1

NTM constitue une borne inférieure pour NT mais pas toujours une limite inférieure.

REMARQUE 2

$\sum_{j=1}^{p} \delta_{ij} f_j$ = probabilité de test de la condition i

$\sum_{j=1}^{p} \delta_{ij} f_j = P_i$

d'où NTM = $\sum_{i=1}^{n} P_i \leqslant NT_P \leqslant n$, $\forall P$

2.2. Mesure d'efficacité

Il est relativement facile de déterminer NTM pour une table donnée. Ceci donnera une référence pour définir l'efficacité d'une traduction.

En effet, NT_p étant le nombre moyen de tests pour l'analyse d'une table donnée par un programme P, l'efficacité de P sera définie par le rapport :

$$E_P = \frac{NTM}{NT_P}$$

EXEMPLE soit latable V.1

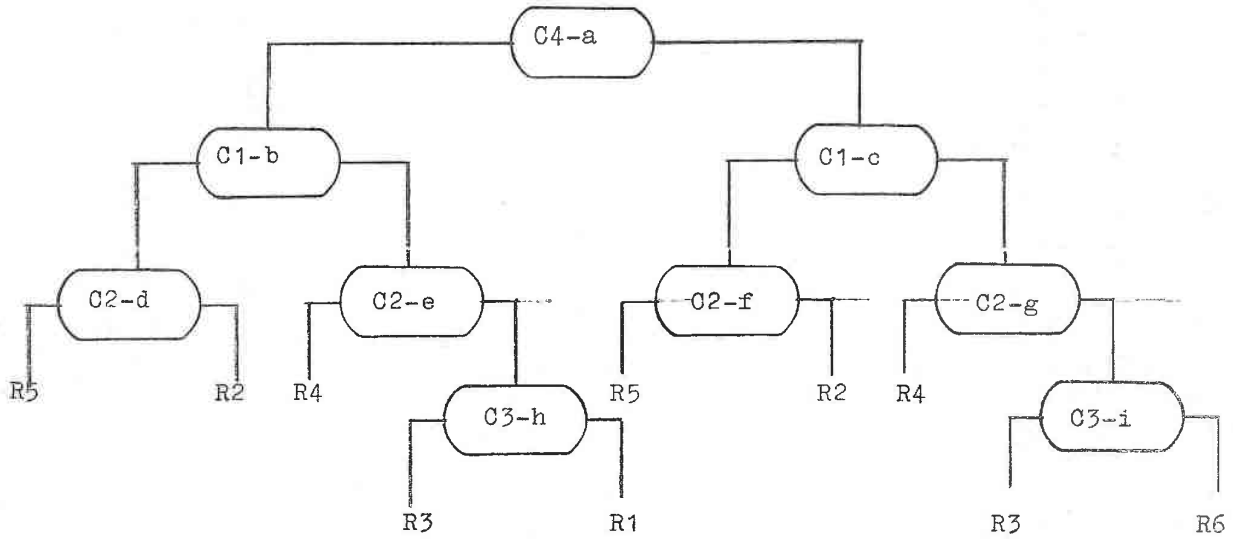
	R1	R2	R3	R4	R5	R6	P_i =
f_j =	1/16	4/16	2/16	3/16	5/16	1/16	
C1	V	F	V	V	F	V	1
C2	V	V	V	F	F	V	1
C3	V	-	F	-	-	V	4/16
C4	F	-	-	-	-	V	2/16

Pour cette table, on a fixé à priori une probabilité de réalisation de chacune des règles, puis calculé P_i pour chaque condition C_i

On a donc ici :

$$NTM = \sum_{i=1}^{4} P_i = 2,375$$

Considérons maintenant l'organigramme suivant qui représente une traduction possible de la table I.1



Pour déterminer le nombre moyen de tests de ce programme, nous allons cumuler les probabilités de chacune d'eux :

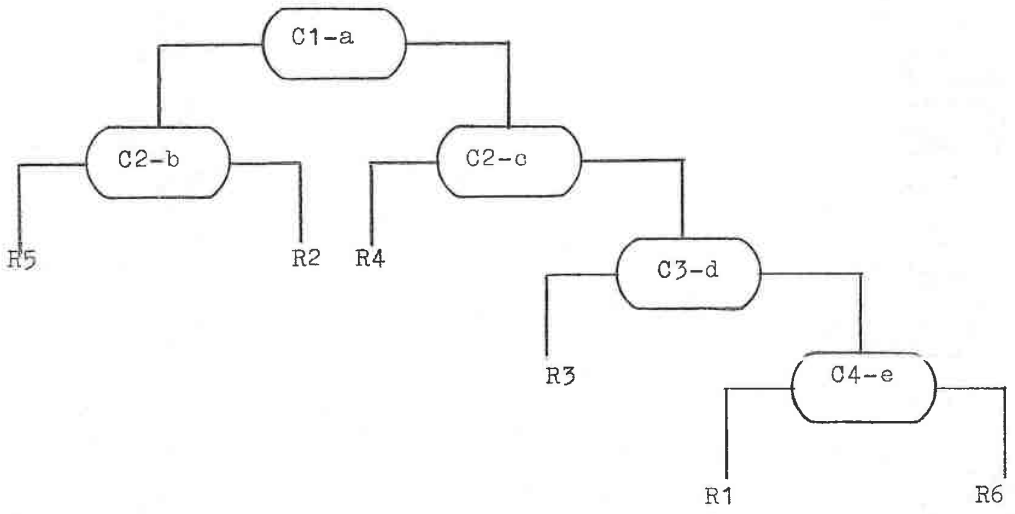
test	a	Prob.	= 1
-	b	-	= 1/2
-	c	-	= 1/2
-	d	-	= 9/32
-	e	-	= 7/32
-	f	-	= 9/32
-	g	-	= 7/32
-	h	-	= 1/8
-	i	-	= 1/8

$$NT = \sum P_i = 3,250$$

Efficacité de cette traduction :

$$\frac{NTM}{NT} = \frac{2,375}{3,250} = 73\ \%$$

Une autre traduction possible de cette table est représentée par le echéma suivant :



La possibilité de chaque test étant :

a - 1
b - 9/16
c - 7/16
d - 1/4
e - 1/8

NT = 2,375

on a alors : $\frac{NTM}{NT} = \frac{2,375}{2,375} = 1 = 100\ \%$

2.3. Durée moyenne d'analyse

Nous venons d'étudier les tables de décision sur le plan du nombre de tests nécessaires à l'analyse.

Il est possible d'affiner cette étude sur le plan de la durée d'analyse d'une table de décision.

En effet :

Si t_i est la durée d'estimations d'une condition donnée C_i

D_P étant la durée moyenne d'analyse d'une table donnée par un programme donné P

nous pouvons également essayer de déterminer des limites logiques pour D_P

a) on a évidemment :

$$D_P \leqslant \sum_{i=1}^{n} t_i \quad (= \text{durée de test de toutes les conditions})$$

b) reprenons les notations et le procédé suivi dans le cas des nombres moyens de tests.

La durée moyenne d'estimation d'une règle R_j sera au minimum :

$$DM_j = \sum_{i=1}^{n} \delta_{ij} t_i$$

d'où une durée moyenne minimum pour l'analyse de la table

$$DM = \sum_{j=1}^{p} DM_j$$

$$DM = \sum_{j=1}^{p} \left(\sum_{i=1}^{n} \delta_{ij} t_i \right) f_j$$

$$= \sum_{j,i} \delta_{ij} t_i f_j$$

$$= \sum_{i=1}^{n} t_i \sum_{j=1}^{p} \delta_{ij} f_j$$

et sachant que $P_i = \sum_{j=1}^{p} \delta_{ij} f_j$

$$DM = \sum_{i=1}^{n} P_i t_i$$

D'autre part, on a :

$$DM \leqslant D_P \leqslant \sum t_i, \forall P$$

Ceci permet donc également une mesure de l'efficacité d'une traduction par rapport à la référence que constitue

$$DM = E_P = \frac{DM}{D_P}$$

3. CAS DES TABLES AVEC REGLE E

Si T est une table possédant une règle E dont la probabilité de réalisation est non négligeable, les évaluations définies en 2.1. deviennent insuffisantes .

EXEMPLE Soit la table V.2

$f_{j=}$	0,25	0,25	0,50
C1	V	F	
C2	-	V	
	A1	A2	A3

P_i=
0,5
0,25

L'application des principes définis en 2.1. fournit :

$$NTM = \sum_{1,2} P_i = 0,75$$

Ce résultat est évidemment insuffisant et sans intérêt pour une éventuelle mesure defficacité.

Or, E représente un regroupement des règles non définies et n'a pas d'entrées propres. Les seuls tests effectués pour définir E sont en fait ceux qui sont faits pour la détermination des règles explicites.

NOTE

En reprenant les notations du chapitre 2.1., nous pouvons définir PC_{ij}

$$PC_{ij} \quad \delta_{ij} f_j$$

d'où avec cette notation :

$$P_i = \sum_{j=1}^{p} \delta_{ij} f_j$$

$$P_i = \sum_{j=1}^{p} PC_{ij}$$

Il est alors possible de généraliser cette notion à la règle E

Soit $\mathcal{E}$ l'ensemble de règles composant la règle E

pour $\epsilon \mathcal{E}$ on a $PC_{ie} = \delta_{ie} f_e$

et pour E globalement :

$$PC_{iE} = \sum_{e \in \{\mathcal{E}\}} PC_{ie}$$

$$= \sum_{e \in \{\mathcal{E}\}} \delta_{ie} f_e$$

Mais f_e n'est pas facile à déterminer et seul $f_E = \sum_{e \in \{\mathcal{E}\}} f_e$ est facilement mesurable.

Nous supposons donc que :

- les composantes d'une règle E sont équi probables

D'où $f_e = \frac{1}{2^n - r} f_E$ (r nombre de règless (sansE))
(n nombre de conditions)

D'où $PC_{iE} = \sum_{e \in \{\mathcal{E}\}} \delta_{ie} f_E \frac{1}{2^n - r}$

$$PC_{iE} = f_E \frac{1}{2^n - r} \sum_{e \in \{\mathcal{E}\}} \delta_{ie}$$

Mais la règle E ne spécifie aucun test.

Le nombre d'entrées d'une condition donnée C_i dans l'ensemble de règles composant E sera donc au maximum égal au nombre d'entrées de C_i dans l'ensemble des règles explicites D

Mais $\sum_{e \in \{\mathcal{E}\}} \delta_{ie}$ est évidemment majoré par le nombre de règles définissant E

Nous choisirons donc comme valeur :

$$\sum_{e \in \{\mathcal{E}\}} \delta_{ie} = \inf(2^n - r, \sum_{i \in \{D\}} \delta_{ij})$$

ce qui entraine :

$$\delta_{iE} = \inf\left(1, \sum_{i \in \{D\}} \delta_{ij} / 2^n - r\right)$$

NB. $\mathcal{E}$ et D sont des ensembles de règles complètes (sans entrées -)

EXEMPLE

Reprenons la table V.2 en définissant ainsi δ_{iE}

Nous obtenons un tableau des δ_{ij}

	0,25	0,25	0,5
δ_{ij}	1	2	E
1	1	1	1
2	0	1	1

on a en effet :

δ_{1E} = inf(1,2) =1

δ_{2E} = inf(1,1) =1

(ici E représente 1 règle)

Ceci donne :

P1 = 0,25×1+0,25×1+0,5×1=1

P2 = 0,25×0+0,25×1+0,5×1=0,75

D'où NTM = 1,75

REMARQUE 1

Ce résultat a seulement une valeur approximative en raison des conventions définies pour sa détermination.

REMARQUE 2

Si 2^n-r devient relativement grand (en pratique si n est assez grand) l'influence de la règle E devient négligeable dans le calcul de NTM même si sa probabilité est significative.

Nous pouvons le constater par l'exemple suivant :

Soit la table V.3

	0,3	0,3	0,4	δ_{iE}
C1	V	F		2/30
C2	F	F		2/30
C3	F	F		2/30
C4	V	F		2/30
C5	V	V		2/30

D'où

$$P_i = 0,3\times 1 + 0,3\times 1 + 0,4\times(2/30) \qquad = 1,\ldots,5$$

$$NTM = \sum_i P_i = 1,5+1,5+4/30$$

$$NTM = 3+0,1333 = 3,1333$$

Le même calcul sans tenir compte de E donnait pour chaque i

$$P_i = 1\times 0,3 + 1\times 0,3 = 0,6$$

$$NTM = \sum_i P_i = 3$$

REMARQUE 3

Une étude semblable est possible sur le plan de la durée moyenne d'analyse. Il faut alors faire intervenir les coûts de tests de chaque condition.

On aurait alors :

(en définissant $\Delta_{ij} = \delta_{ij} t_i$)

$$\Delta_{iE} = \inf\left(t_i, \sum_{i \in \{D\}} \delta_{ij} t_i \;/\; 2^n - r \right)$$

Le coût de la condition C_i pour la règle E intervient donc en tant que moyenne de plusieurs coûts. Ceci limite donc l'intérêt pratique d'une telle étude.

4. RECHERCHE D'UN ALGORITHME DE TRADUCTION

Nous nous proposons l'optimisation de la traduction en restant dans le cadre de l'algorithme de traduction défini au chapitre III (c'est-à-dire, traduction par analyse, en procédent par division successive des tables).

L'optimisation portera donc sur le choix de la condition à tester pour la division de la table.

4.1. Première règle de choix de la condition

La condition choisie sera la première du programme résultant ; elle sera donc testée à chaque analyse de la table.

Pour minimiser le nombre de conditions testées, il faut choisir la condition qui a la plus grande probabilité d'être testée.

La division fournit deux sous-tables qui peuvent à leur tour être divisées suivant cette règle qui pourra se définir ainsi :

Règle 1

Pour chaque division de la table, choisir la condition dont le test est le plus probable.

EXEMPLE 1 Table V.4

	R1	R2	R3	R4
f_j	3/10	4/10	2/10	1/10
C1	V	V	F	-
C2	V	-	-	F
C3	F	V	V	F

P_i
9/10
4/10
10/10

On a ici NTM = $\sum P_i = 2,3$

La règle adoptée pour la division est donc C3 et on obtient les deux sous-tables :

Table V.5

F_j	3/4	1/4
C1	V	-
C2	V	F

P_i
3/4
1

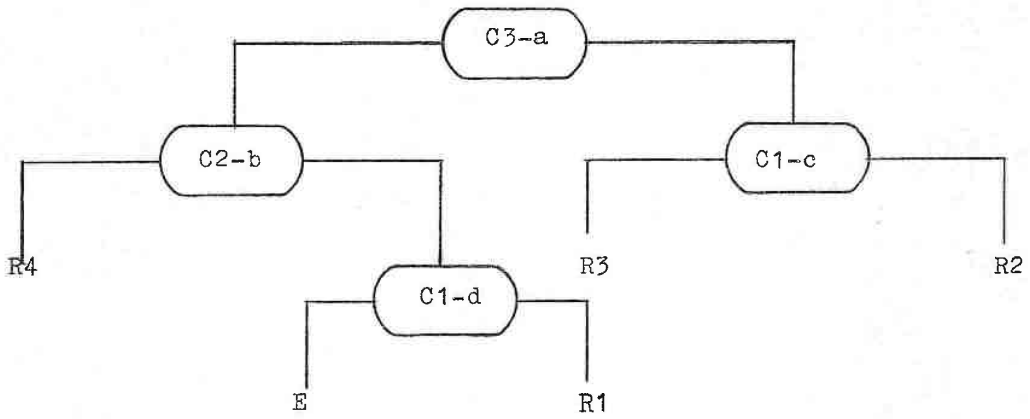
Table V.6.

F_j	2/3	1/3
C1	V	F
C2	-	-

P_i
1
0

Pour la table V.5, on choisit ensuite C2 puis C1.
Dans la table V.6, il reste C1 à tester.

Nous obtenons donc une traduction représentée par la figure suivante :



La probabilité de chaque test étant :

a)	prob.	1
b)	-	0,4
c)	-	0,6
d)	-	0,3

Cet organigramme définit un nombre moyen de test = 23 ; il traduit donc la table V.4. avec une efficacité de 100 %.

4.2. Deuxième règle de choix de la condition

Cette première règle peut s'avréer insuffisante pour choisir une seule condition. Ainsi, si l'on voulait traduire la table V.1 l'application de la règle 1 seule nous laisse hésiter entre les conditions C1 et C2. Une seconde règle de choix est nécessaire. Elles'appuie sur l'efficacité d'un test. Nous dirons que le test le plus efficace est celui dont le résultat est le moins prévisible

donc celui pour lequel $P(C_i=Vrai)$ est la plus proche de 1/2

D'où la seconde règle de choix:

Règle 2

En cas d'ambiguité après application de la règle 1, on décide de choisir la condition dont le test est le plus efficace (parmi celles qui sont possibles d'après la règle 1)

EXEMPLE

Nous nous proposons de traduire la table V.1

	R1	R2	R3	R4	R5	R6	P_i
f_j	1/16	4/16	2/16	3/16	5/16	1/16	
C1	V	F	V	V	F	V	1
C2	V	V	V	F	F	V	1
C3	V	-	F	-	-	V	4/16
C4	F	-	-	-	-	V	2/16

La règle 1 laisse une ambiguité entre C1 et C2 ($P_1=P_2$)

Appliquons la règle 2 :

prob (C1=V)=7/16

prob (C2=V)=1/2

Le test de C2 est plus efficace, la division se fera donc à partir de C2.

Nous obtenons les tables

V.7. correspondant aux entrées F

V.8. correspondant aux entrées V

Table V.7

	R4	R5	
f_j	3/8	5/8	P_i
C1	V	F	1
C3	-	-	0
C4	-	-	0

Table V.10

	R1	R2	R3	R6	
f_j	1/8	4/8	2/8	1/8	P_i
C1	V	F	V	V	1
C3	V	-	F	V	1/2
C4	F	-	-	V	1/4

- La traduction de la table V.7. se termine par le test de C1 qui détermine les règles R4 et R5.

- La règle 1 s'applique à la table V.8

Le test de C1 (d'après la règle 1) entraine une division en deux tables V.9 et V.10 (respectivement entrées F et V)

Table V.9

Table vide - sélection de la règle R2

Table V.10

	R1	R3	R6
f_j	1/4	2/4	1/4
C3	V	F	V
C4	F	-	V

P_i
1
1/2

C3 est chosie pour la division en :

- une table vide, sélection de R3
- la table V.11 correspondant aux entrées V

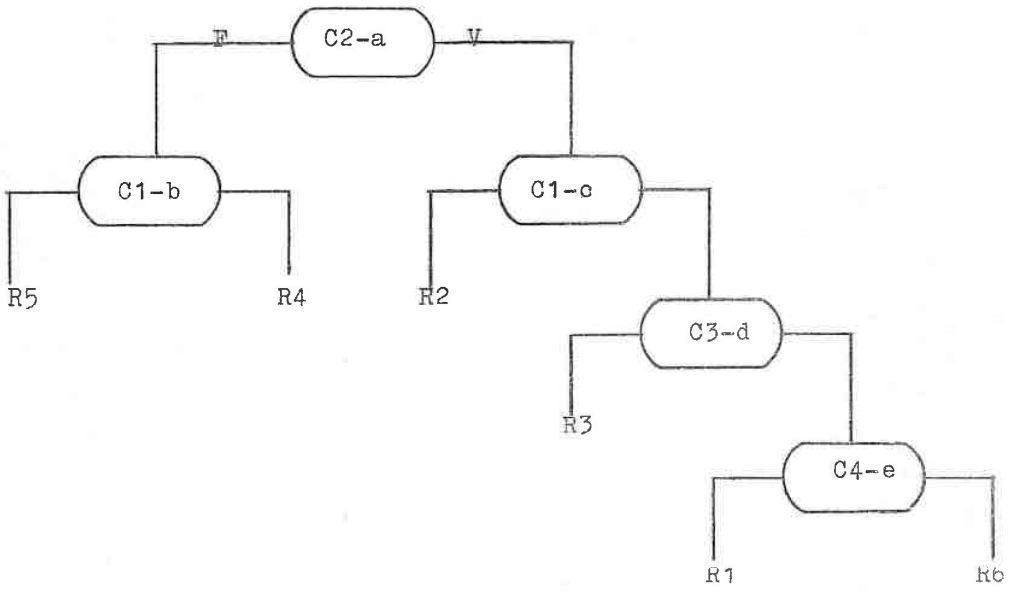
Table V.11

	R1	R6
f	1/2	1/2
C4	F	V

P
1

Enfin le test de C4 nous permet la sélection des règles R1 et R6.

Nous avons ici défini l'arborescence ci-dessous :



Calculons l'efficacité de cette traduction :

P(a) = 1
P(b) = 1/2
P(c) = 1/2
P(d) = 1/4
P(e) = 1/8

NT = 2,375 d'où efficacité $= \frac{NT1}{NT} = \frac{2,375}{2,375} = 100\ \%$

4.3. Cas de règles équiprobables

Supposons les règles équiprobables ($f_j = k \quad j = 1,\ldots,p$)

On a $P_i = \sum_j \alpha_{ij} f_j = k \sum_j \delta_{ij}$

$\sum_j \alpha_{ij}$ = nombre d'entrées actives de la condition C_i (entrées V ou F)

La règle 1 devient alors :

Pour chaque division de la table choisir la condition dont le nombre d'entrées actives est le plus grand.

Nous retrouvons ici la règle de choix de la condition de division que nous avons retenue pour la réalisation du traducteur.

4.4. Utilisation de l'optimisation

Les méthodes simples que nous venons de décrire nécessitent de nombreuses données statistiques ; nous pensons donc qu'il est toujours souhaitable d'effectuer une première traduction simple et, seulement dans le cas où elle ne donne pas satisfaction, de décider une phase d'étude statistique en vue de l'application des procédés d'optimisation faisant intervenir la probabilité de réalisation des règles, puis, si nécessaire, des évaluations du coût de chaque condition.

C O N C L U S I O N

Le traducteur que nous venons de décrire et qui sera utilisé pour la version de base de "Civa" est susceptible de nombreuses extensions et améliorations:

a) Vers une plus grande généralité des tables admises à l'entrée (aiguillages, tables à entrées étendues,...). Cependant, il semble souvent préférable de s'orienter vers des traducteurs spécialisés plus performants et plus efficaces sur le plan de la qualité des programmes générés.

b) vers une optimisation des programmes générés que nous venons d'esquiser au chapitre précédent.

Ce problème de l'optimisation de la traduction n'est cependant pas encore complétement résolu et il n'existe pas d'algorithme de recherche d'une traduction optimale (les algorithmes que nous avons présentés permettent une "bonne" traduction mais nous ne serons pas, en général, sûr d'avoir obtenu la meilleure traduction possible).

Des études approfondies sur ce sujet sont souhaitables mais il sera certainement indispensable de définir au préalable une stricte formalisation des tables de décision de façon à déterminer complètement leurs caractéristiques et leurs propriétés.

B I B L I O G R A P H I E

Informatique et Gestion

1 - Picard - N° 3-4-5
Les tables de décision

Communication of the ACM

2 - Kirk H.W - Vol.8/N°1/1965, p.41-43
Use of Decision Tables in computer Programming

3 - Press L.I - Vol.8/N°6/1965, p.385-390
Conversion of Decision Tables to Computer Programs

4 - Pollack S.L - Vol.8/N°11/1965, p.677-682
Conversion of Limited Entry Decision Tables

5 - Fisher D.L - Vol.9/N°1/1966, p.26-31
Data, documentation and Decision Tables

6 - Veinott, Syrill G. - Vol.9/N°1/1966, p.31-35
Programming Decision Tables in FORTAN COBOL,
or ALGOL

7 - King P.J.H. - Vol.9/N°11/1966, p.796-801
Conversion of Decision Table to Computer Programs
by Rule Mask Techniques

8 - Callahan M.D and Chapman A.E - Vol.10/N°7/1967, p.441-446
Description of Basic Algorithm in DETAB/65 préprocessor

9 - Chapin N - Vol.10/N°8/1967, p.507-512
Parsing of Decision Tables

10 - King P.J.H. - Vol.11/N°10/1968, p.680-684
Ambiguity in Limited Entry Decision Tables

11 - Muthukrishnan C.R and Rajaraman V - Vol.13/N°6/1970, p.347-351
On the conversion of Decision Tables to Computer programs

12 - Shwayder K - Vol.14/N°2/1971, p.69-73
Conversion of Limited Entry Decision Tables to Computer Programs - A proposed Modification to Pollack's Algorithm

13 - Dathe G - Vol.15/N°10/1972, p.906-909
Conversion of Decision Tables by Rule Mask Methode without Rule Mask

Journal of the ACM

14 - Reinwald L.T and Soland R.M - Vol.13/N°3/July 1966, p.339-358
Conversion of Limited Entry Decision Tables to Optimal Computer Programs - 1) Minimum Average Processing Time

15 - Reinwald L.T and Soland R.M - Vol.14/N°4/Oct.1967, p.742-755
Conversion of Limited Entry Decision Tables to Optimal Computer Programs - 2) Minimum Storage Requirement

The Computer Journal

16 - King P.J.H - Vol.10/Aug.1967, p.135-142
Decision Tables

17 - Application of Decision Tables
Edited by Herman Mc Daniel
Brandon/Systems Press, Inc.

18 - Decision Table Software
Hermann Mc Daniel
Brandon/Systems Press, Inc.

0.1 Informatique (management)

19 - Markia A. - N°1/1971, p.23-28
"Mea Culpa, Mea Maxima Culpa"

Documentation I.B.M.

20 - I.B.M. TF2 0007 - 11/8/1965
Application des tables de décision

21 - I.B.M. TF2 0014 - 30/3/1966
Vers l'automatisation des études de production

22 - I.B.M. TF2 0036 - 6/2/1968
Le point sur les tables de décisions

23 - I.B.M. H20 0492 - 1 - Oct.1968
System/360 Decision logic Translator (360A-CX-32X)
Application Description Manual

Documentation I.C.L.

24 - I.C.L. 1 900 SERIES - Jan. 1969
Decision Tables with Cobol

25 - Université NANCY 1 - 1973
Derniame J.C. - 'Civa' - Thèse d'Etat.

NOM DE L'ETUDIANT : AUBRY Bernard

Nature de la thèse : Doctorat de Spécialité en Informatique

Vu, Approuvé

et permis d'imprimer

NANCY, le 28.2.73

Le Président du Conseil de l'Université de NANCY I

J.R. HELLUY