Langages analysables de gauche à droite, construction d'un analyseur pour langages LR (1)

Jacques Courtin

▶ **To cite this version:**

Jacques Courtin. Langages analysables de gauche à droite, construction d'un analyseur pour langages LR (1). Modélisation et simulation. Université Joseph-Fourier - Grenoble I, 1968. Français. NNT : . tel-00281024

HAL Id: tel-00281024
https://theses.hal.science/tel-00281024

Submitted on 20 May 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N° d'ordre

THESE

présentée à

LA FACULTE DES SCIENCES DE L'UNIVERSITE DE GRENOBLE

pour obtenir

LE GRADE DE DOCTEUR DE TROISIEME CYCLE
" MATHEMATIQUES APPLIQUEES "

par

Jacques Courtin

Langages analysables de gauche à droite : Construction d'un analyseur pour langages L R (1)

Thèse soutenue le 29 Juin 1968, devant la Commission d'examen :

MM. J. KUNTZMANN	Président
B. VAUQUOIS	
L. BOLLIET	Examinateurs
J.C. BOUSSARD	

N° d'ordre

THESE

présentée à

LA FACULTE DES SCIENCES DE L'UNIVERSITE DE GRENOBLE

pour obtenir

LE GRADE DE DOCTEUR DE TROISIEME CYCLE
" MATHEMATIQUES APPLIQUEES "

par

Jacques Courtin

Langages analysables de gauche à droite : Construction d'un analyseur pour langages L R (1)

Thèse soutenue le 29 Juin 1968, devant la Commission d'examen :

MM.	J. KUNTZMANN	Président
	B. VAUQUOIS	
	L. BOLLIET	Examinateurs
	J.C. BOUSSARD	

FACULTE DES SCIENCES

UNIVERSITE DE GRENOBLE

L I S T E D E S P R O F E S S E U R S

-:-:-:-:-

DOYEN HONORAIRE :

M. MORET

DOYEN :

M. BONNIER

PROFESSEURS TITULAIRES

MM. NEEL Louis	Chaire de Physique Expérimentale
HEILMANN René	Chaire de Chimie
KRAVTCHENKO Julien	Chaire de Mécanique Rationnelle
CHABAUTY Claude	Chaire de Calcul différentiel et Intégr
BENOIT Jean	Chaire de Radioélectricité
CHENE Marcel	Chaire de Chimie Papetière
FELICI Noël	Chaire d'Electrostatique
KUNTZMANN Jean	Chaire de Mathématiques Appliquées
BARBIER Reynold	Chaire de Géologie Appliquée
SANTON Lucien	Chaire de Mécanique des Fluides
OZENDA Paul	Chaire de Botanique
FALLOT Maurice	Chaire de Physique Industrielle
KOSZUL Jean-Louis	Chaire de Mathématiques
GALVANI O.	Mathématiques
MOUSSA André	Chaire de Chimie Nucléaire
TRAVNARD Philippe	Chaire de Chimie Générale
SOUTIF Michel	Chaire de Physique Générale
CRAYA Antoine	Chaire d'Hydrodynamique
REULOS R.	Théorie des Champs
BESSON Jean	Chaire de Chimie
AYANT Yves	Physique Approfondie

GALLISSOT	Mathématiques
Mlle LUTZ Elisabeth	Mathématiques
BLAMBERT Maurice	Chaire de Mathématiques
BOUCHEZ Robert	Physique Nucléaire
LLIBOUTRY Louis	Géophysique
MICHEL Robert	Chaire de Minéralogie et Pétrographie
BONNIER Etienne	Chaire d'Electrochimie et d'Electrométallurgie
DESSAUX Georges	Chaire de Physiologie Animale
PILLET E.	Chaire de Physique Industrielle et Electrotechnique
YOCCOZ Jean	Chaire de Physique Nucléaire Théorique
DEBELMAS Jacques	Chaire de Géologie Générale
GERBER R.	Mathématiques
PAUTHENET R.	Electrotechnique
VAUQUOIS B.	Chaire de calcul électronique
BARJON R.	Physique Nucléaire
BARBIER Jean-Claude	Chaire de Physique
SILBER R.	Mécanique des fluides
BUYLE-BODIN Maurice	Chaire d'Electronique
DREFUYS B.	Thermodynamique
KLEIN J.	Mathématiques
VAILLANT F.	Zoologie et Hydrobiologie
ARNAUD Paul	Chaire de Chimie
SENGEL P.	Chaire de Zoologie
BARNOUD F.	Chaire de Biosynthèse de la cellulose
BRISSONNEAU P.	Physique
GAGNAIRE	Chaire de Chimie Physique
Mme KOFLER L.	Botanique
DEGRANGE Charles	Zoologie
PEBAY-PEROULA J.C.	Physique
RASSAT A.	Chaire de Chimie Systématique
DUCROS P.	Chaire de Cristallographie Physique
DODU Jacques	Chaire de Mécanique Appliquée I.U.T.
ANGLES D'AURIAC P.	Mécanique des Fluides
LACAZE A.	Thermodynamique

PROFESSEURS SANS CHAIRE

MM.	GIDON P.	Géologie et Minéralogie
	GIRAUD P.	Géologie
	PERRET R.	Servomécanisme
Mme	BARBIER M.J.	Electrochimie
Mme	SOUTIF J.	Physique
	COHEN J.	Electrotechnique
	DEPASSEL R.	Mécanique des Fluides
	GASTINEL N.	Mathématiques Appliquées
	GLENAT R.	Chimie
	BARRA J.	Mathématiques Appliquées
	COUMES A.	Electronique
	PERRIAUX J.	Géologie et Minéralogie
	ROBERT A.	Chimie Papetière
	BIAREZ J.P.	Mécanique Physique
	BONNET G.	Electronique
	CAUQUIS G.	Chimie Générale
	BONNETAIN L.	Chimie Minérale
	DEPOMMIER P.	Etude Nucléaire et Chimie Atomique
	HACQUES Gérard	Calcul Numérique
	POLOUJADOFF M.	Electrotechnique

PROFESSEURS ASSOCIES

MM.	NAPP-ZINN	Botanique
	RODRIGUES Alexandre	Mathématiques Pures
	STANDING Kenneth	Physique Nucléaire

MAITRES DE CONFERENCES

MM.	LANCIA Roland	Physique Atomique
Mme	KAHANE J.	Physique
	DEPORTES C.	Chimie
Mme	BOUCHE L.	Mathématiques
	SARROT-REYNAUD	Géologie Propédeutique
Mme	BONNIER M.J.	Chimie
	KAHANE A.	Physique Générale
	DOLIQUE J.M.	Electronique

BRIERE G.	Physique M.P.C.
DESRE G.	Chimie S.P.C.N.
LAJZROWICZ J.	Physique M.P.C.
VALENTIN P.	Physique M.P.C.
BERTRANDIAS J.P.	Mathématiques Appliquées - TMP
LAURENT P.J.	Mathématiques Appliquées - TMP
CAUBET J.P.	Mathématiques Pures
PAYAN J.J.	Mathématiques
Mme BERTRANDIAS F.	Mathématiques Pures M.P.C.
LONGEQUEUE J.P.	Physique
NIVAT M.	Mathématiques Appliquées
SOHM J.C.	Electrochmie
ZADWORNY F.	Electronique
DURAND F.	Chimie Physique
CARLER G.	Biologie Végétale
AUBERT G.	Physique M.P.C.
DELPUECH J.J.	Chimie Organique
PFISTER J.C.	Physique C.P.E.M.
CHIBON P.	Biologie Animale
IDELMAN S.	Physiologie Animale
BOUVARD Maurice	Hydrologie
RICHARD Lucien	Botanique
PELMONT Jean	Physiologie Animale
BLOGH D.	Electrotechnique I.P.
BOUSSARD J.Claude	Mathématiques Appliquées I.P.
MOREAU René	Hydraulique I.P.
BRUGEL L.	Energétique I.U.T.
SIBILLE R.	Construction Mécanique I.U.T.
ARMAND Yves	Chimie I.U.T.
BOLLIET Louis	Infromatique I.U.T.
KUHN Gérard	Energétique I.U.T.
GERMAIN Jean-Pierre	Construction Mécanique I.U.T.
CONTE René	Thermodynamique
JOLY Jean-René	Mathématiques Pures
PIERY Yvette	Biologie Animale

MAITRE DE CONFERENCES ASSOCIES

SAWCZUK A.	Mécanique des Fluides
CHEEKE J.	Thermodynamique
YAMADA O.	Physique du Solide
NATR Lubomir	B.M.P.V.
NAYLOR Arch	Physique Industrielle
SILBER Léo	Radioélectricité
NOZAKI Akihiro	Mathématiques Appliquées
RUTLEDGE Joseph	Mathématiques Appliquées
DONOHO Paul	Physique Générale
EGGER Kurt	B.M.P.V.

INTRODUCTION

Les méthodes d'analyse syntaxique actuelles qui sont très générales ont l'inconvénient d'être très lentes car elles procèdent par essais successifs. Il nous a paru intéressant d'essayer de donner une méthode d'analyse qui soit très rapide, l'existence d'une telle méthode étant nécessaire à la réalisation effective d'un compilateur dirigé par la syntaxe [CO] .

La méthode que nous proposons a été formalisée par D.E. KNUTH [Kn-1] .

Dans un premier chapitre nous exposerons le principe de l'algorithme et les principaux résultats qui en découlent. En particulier, nous verrons que les langages définis par les grammaires LR(k) ont la puissance des langages déterministes au sens de Ginsburg et Greibach [GG] .

Les chapitres suivants seront consacrés aux applications pratiques, à la façon de programmer les divers algorithmes, et aux résultats que nous avons obtenus.

Je tiens à exprimer ma profonde reconnaissance à

Monsieur le Professeur KUNTZMANN, Directeur de l'Institut de Mathématiques Appliquées de Grenoble, qui a bien voulu me faire l'honneur de présider le Jury,

Monsieur le Professeur VAUQUOIS, Directeur du Centre d'Etudes pour la Traduction Automatique, qui a bien voulu faire partie du Jury,

Monsieur BOLLIET, Maître de Conférences à l'Institut Universitaire de Technologie de Grenoble, d'avoir eu la bienveillance de s'intéresser à ce travail,

Monsieur BOUSSARD, Maître de Conférences à l'Institut Polytechnique, qui a bien voulu m'aider et me guider dans mes travaux,

Monsieur COHEN, Professeur Assistant au Massachussets Institute of technology, qui dirigea mes premiers travaux.

Je tiens à remercier également toutes les personnes du Laboratoire qui m'ont apporté leur aide, en particulier J. LE PALMEC, A. COLMERAUER et Y. SIRET.

Enfin, je remercie toutes les personnes qui ont participé à la réalisation matérielle de cette thèse.

T A B L E DES M A T I E R E S

.-.-.-.-.

I 1. LANGAGES ANALYSABLES DE GAUCHE A DROITE

1.1. Quelques définitions sur les langages 1

1.2. Grammaire context-free 2

1.3. Arbre syntaxique associé à une phrase par une grammaire 4

1.4. Grammaire "non-ambigue" - Exemple récapitulatif 7

1.5. Langages analysables de gauche à droite - Grammaires LR(K). 8

1.6. Analyse des langages analysables de gauche à droite 13

II 2. CONSTRUCTION DE LA TABLE

2.1. Présentation des données 37

2.2. Détection des symboles terminaux. 40

2.3. Construction de la table. 41

2.4. Sortie des résultats. 51

2.5. Programme 51

2.6. Résultats 68

2.7. Utilisation de la table pour la mise au point des grammaires 131

III 3. ANALYSEUR LISP-ALGOL

IV 4. TRANSFORMATION DE LA TABLE

4.1. Ordre particulier pour la consultation de la table 151

4.2. Réalisation pratique 151

4.3. Transformation de la table 152

4.4. Programme 157

V 5. ANALYSEUR ALGOL

VI 6. CONCLUSION

6.1. Rapidité de l'analyseur 173

6.2. Construction de la table dans le cas général 174

6.3. Lien avec la génération 176

BIBLIOGRAPHIE 178

CHAPITRE I

LANGAGES ANALYSABLES DE GAUCHE A DROITE

I - 1 Quelques définitions sur les langages.

I - 1 - 1 Vocabulaire et chaînes.

- On dispose d'un ensemble V fini appelé vocabulaire dont les éléments sont appelés symboles.

- Une chaîne α sur V est une suite finie de symboles $X_1, X_2, \ldots, X_n \in V$.

On écrira $\alpha = X_1 \ldots X_n$

On notera V* l'ensemble de toutes les chaînes que l'on peut construire sur V.

I - 1 - 2 Langage.

- On appelle langage tout sous-ensemble de V*.

I - 1 - 3 Longueur d'une chaîne.

- Si $\alpha = X_1 X_2 \ldots X_n$, on dira que la longueur de la chaîne α, que l'on notera $\mathcal{L}(\alpha)$, est égale à n.

I - 1 - 4 Concaténation de deux chaînes.

- Soient deux chaînes α et β

$\alpha = X_1 X_2 \ldots X_n$

$\beta = Y_1 Y_2 \ldots Y_m$

La concaténation de ces deux chaînes, que l'on notera $\alpha\beta$, est par définition, une chaîne égale à

$$X_1 X_2 \ldots X_n Y_1 Y_2 \ldots Y_m$$

L'élément neutre de la concaténation est la chaîne vide que l'on notera Λ :

$$\Lambda\alpha = \alpha\Lambda = \alpha.$$

Il est clair que $\mathcal{L}(\alpha\beta) = \mathcal{L}(\alpha) + \mathcal{L}(\beta)$

On définit de même la concaténation de la chaîne α, n fois avec elle même :

$$\alpha^n = \{\text{si } n = 0 \text{ alors } \Lambda \text{ sinon } \alpha\alpha^{n-1}\}$$

I - 2 Grammaire Context-Free.

I - 2 - 1 Définition.

- On définit une grammaire Context-Free par
  - La donnée d'un vocabulaire terminal V_T
  - La donnée d'un vocabulaire non-terminal V_N tel que $V_T \cup V_N = V$, $V_T \cap V_N = \emptyset$
  - Un élément particulier S appelé Axiome avec $S \in V_N$
  - Un ensemble fini $\mathcal{P}$ de productions.

Par la suite, on notera $G = (V_T, V_N, S, \mathcal{P})$

I - 2 - 2 Production.

- Une production est un élément d'une relation finie notée $\rightarrow$ entre les éléments de V_N d'une part et les chaînes sur V d'autre part.
- On notera $A \rightarrow \theta$ la production signifiant que $\rightarrow$ est vraie pour le couple (A, θ), $A \in V_N$, $\theta \in V^*$.

I - 2 - 3 Conventions.

- On notera les éléments de V_T par des lettres minuscules et ceux de V_N par des lettres majuscules.

- On désignera par X, Y, Z les éléments de V, et par α, β, γ... les chaînes sur V (c'est-à-dire des éléments de V*).

I - 2 - 4 Relation $\rightarrow$ entre deux chaînes.

- On dira que $\alpha \rightarrow \beta$ si et seulement si il existe des chaînes φ, Ψ, θ et un non-terminal A tels que $\alpha = \varphi A \Psi$, $\beta = \varphi \theta \Psi$, $A \rightarrow \theta$ étant une production de G.

On appellera dérivation directe une telle relation.

I - 2 - 5 Relation transitive

- On dira que $\alpha \Rightarrow \beta$ s'il existe une suite finie de chaînes α_o, α_1, α_2 ..., α_n $(n > 0)$ telle que $\alpha_o \rightarrow \alpha_1 \rightarrow \alpha_2 \rightarrow \ldots \rightarrow \alpha_n$ avec $\alpha_o = \alpha$ et $\beta = \alpha_n$.

- On appellera dérivation une telle relation.

- Il y a lieu de remarquer que $\alpha \Rightarrow \alpha$ n'est pas toujours vrai. S'il en est ainsi, on dira que la grammaire est circulaire pour α.

I - 2 - 6 Relation $\Rrightarrow$

- On écrira $\alpha \Rrightarrow \beta$ pour $\alpha \Rightarrow \beta$ ou $\alpha = \beta$.

I - 2 - 7 Langage défini par une grammaire G.

- On appellera langage défini par G, que l'on notera L(G) le sous-ensemble de V_T^* tel que

$$L(G) = \{\alpha / S \Rightarrow \alpha,\ \alpha \in V_T^*\}$$

I - 2 - 8 Phrase.

- On appellera phrase n'importe quelle chaîne $\alpha \in V^*$ telle que

$$S \Rrightarrow \alpha$$

- On appellera phrase terminale les éléments de L(G).

I - 2 - 9 Grammaires équivalentes.

- Deux grammaires G et G' seront dites équivalentes si elles génèrent le même langage. $L(G) = L(G')$

I - 2 - 10 Restrictions habituelles sur une grammaire Context-Free.

1 - Elle ne comporte pas de cycle, c'est-à-dire qu'il n'existe pas de symbole non-terminal A tel que $A \Rightarrow A$

2 - Elle ne comporte pas de production de la forme $A \rightarrow \Lambda$

3 - Pour tout symbole $Z \in V$, il existe toujours des chaînes $\alpha, \beta \in V^*$ telles que $S \Rightarrow \alpha Z \beta$

4 - Pour tout symbole $Z \in V$, il existe toujours une chaîne $\alpha \in V_T^*$ telle que $Z \Rightarrow \alpha$

On peut toujours se ramener à ce cas. Il existe d'ailleurs des programmes qui permettent de transformer une grammaire G en une grammaire équivalente vérifiant ces 4 conditions [MA] , à condition que $\Lambda \notin L(G)$.

On appellera maintenant grammaire, une grammaire Context-Free satisfaisant aux quatre restrictions précédentes.

I - 3 Arbre syntaxique associé à une phrase par une grammaire.

Cette présentation est inspirée de [V V V] et de [T R]

I - 3 - 1 n-graphe.

- Etant donné un ensemble X fini et une application multivoque T de X dans X, le couple (X, T) constitue un graphe.

- On appelle Arc tout couple (x_i, x_j) tel que $x_j \in T\ x_i$

- Soient $T_1, T_2, \ldots T_n$ des applications multivoques de X dans X, on appelle n-graphe le (n+1)-uple : $(X, T_1, T_2, \ldots, T_n)$ considéré comme le graphe superposition des graphes (X, T_1), (X, T_2), ..., (X, T_n) dans lequel tous les arcs sont désignés selon leur appartenance à l'application d'où ils proviennent.

I - 3 - 2 Structure.

Soit un n-graphe $(X, T_1, T_2 \ldots T_n)$, V un ensemble de symboles (vocabulaire) et Δ une application univoque de X dans V.

- On appelle structure le (n+3)-uple : $(X, V, T_1, T_2, \ldots, T_n, \Delta)$.

En attribuant à chaque noeud x_i du n-graphe un élément de V au moyen de Δ, on obtient la structure du n-graphe.

Ainsi des noeuds distincts de la structure pourront être interprétés par le même élément de V.

I - 3 - 3 Structure arborescente.

- On appelle structure arborescente le 5-uple (X, V, T_1, T_2, Δ) associé au 2-graphe (X, T_1, T_2) tel que :

- Le graphe (X, T_1) soit une arborescence, c'est-à-dire

1 - Il existe un noeud $x_o \in X$ unique tel que $T_1^{-1}\ x_o = \emptyset$.

2 - Pour tout noeud $x \neq x_o$, il existe un noeud y unique tel que $T_1^{-1}\ x = y$, $y \in X$

3 - Le graphe est connexe.

- L'application T_1 étant donnée, l'application T_2 doit satisfaire aux conditions suivantes :

1 - A tout sous-ensemble T_1x correspond une composante connexe de (X, T_2)

2 - Pour chacun de ces sous-ensembles T_1x, la relation d'ordre associée à T_2 est totale. Ainsi dans tout sous-ensemble $X_i = T_1x_i$ il existe un $x_A \in X_i$ unique tel que $T_2^{-1}x_A = \emptyset$ et un $x_B \in X_i$ unique tel que $T_2x_B = \emptyset$.

La donnée du 2 graphe (X, T_1, T_2) constitue une arborescence munie d'une relation d'ordre restreinte.

On emploiera aussi le terme d'arbre pour désigner une structure arborescente.

Pratiquement, on représentera T_2 par l'écriture de gauche à droite des éléments de chaque composante connexe de (X, T_2).

I - 3 - 4 Arbre syntaxique associé à une phrase par une grammaire.

Soit une grammaire $G = (V_T, V_N, S, \mathcal{P})$. Considérons la structure arborescente (X, V, T_1, T_2, Δ) associée à G telle que

1 - $V = V_T \cup V_N$

2 - $T = \{x \in X / T_1x = \emptyset\}$

3 - $\Delta = \Delta_1 \cup \Delta_2$

Δ_1 étant une application univoque de T dans V_T, Δ_2 étant une application univoque de X - T dans V_N avec $\Delta_2(x_o) = S$

- Si $\Delta(x_i) = A_p$, $A_p \in V_N$

et si

- pour $y_1 \in X$, $T_1^{-1}y_1 = x_i$ et $T_2^{-1}y_1 = \emptyset$,
  $\Delta(y_1) = X_{p_1}$
- pour $y_j \in X$, $y_j = T_2y_{j-1}$, $\Delta(y_j) = X_{p_j}$ $\quad 2 < j < n_p$
- pour $y_{n_p} \in X$, $y_{n_p} = T_2y_{n_p-1}$ et $T_2y_{n_p} = \emptyset$
  $\Delta(y_{n_p}) = X_{p_{n_p}}$

alors il existe dans $\mathcal{P}$ la production

$$A_p \rightarrow X_{p_1} X_{p_2} \ldots X_{p_{n_p}}$$

I - 4 - Grammaire "non-ambiguë" - Exemple récapitulatif.

I - 4 - 1 Grammaire "non-ambiguë".

- Une grammaire sera dite "non-ambiguë", si pour chaque phrase, il n'existe qu'une seule structure.

I - 4 - 2 Exemple récapitulatif.

Illustrons les principales définitions précédentes par un exemple, tiré de celui à partir duquel D.E. KNUTH expose sa théorie.

Soit une grammaire $G = (V_T, V_N, S, \mathcal{P})$ avec $V_T = \{a, b, c, d, e\}$
$V_N = \{S, A, B, C, D, E\}$

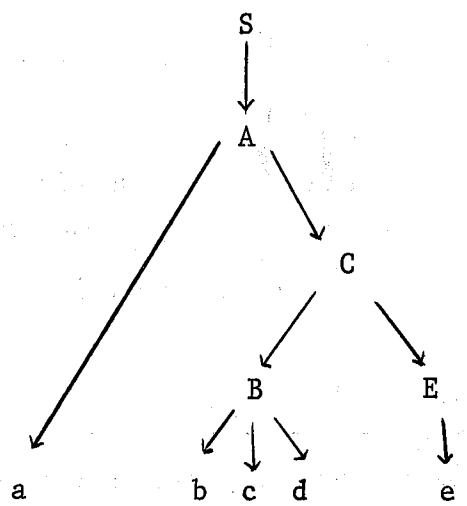
$\mathcal{P}$ ayant 5 productions :

1 - $S \rightarrow A$
2 - $A \rightarrow a\ C$
3 - $B \rightarrow b\ c\ d$
4 - $C \rightarrow B\ E$
5 - $E \rightarrow e$

$L(G) = \{a\ b\ c\ d\ e\}$

S, A, a C, a B E, a b c d E, a b c d e sont des phrases.

Construisons la structure associée à a b c d e



Pour cette grammaire il n'existe qu'un seul arbre bien qu'il y ait plusieurs manières de générer la phrase a b c d e.

En effet, en voici deux :

(1) $S \rightarrow A \rightarrow a\ C \rightarrow a\ B\ E \rightarrow a\ b\ c\ d\ E \rightarrow a\ b\ c\ d\ e$

(2) $S \rightarrow A \rightarrow a\ C \rightarrow a\ B\ E \rightarrow a\ B\ e \rightarrow a\ b\ c\ d\ e$

Pour éviter ce léger inconvénient, on peut convenir d'un ordre particulier dans l'application des productions.

On peut par exemple décider de substituer toujours le caractère non-terminal le plus à gauche (1) ou le plus à droite (2).

C'est ce que nous allons essayer de formaliser dans l'autre sens.

I - 5 Langages analysables de gauche à droite - Grammaires LR(K).

I - 5 - 1 Définition du Manche *

Soit $X_1\ X_2 \ldots X_t$ une phrase ($\forall i,\ X_i \in V$).

On cherche le plus petit n possible tel qu'il existe un j, tel que

$$Y \rightarrow X_j\ X_{j+1} \ldots X_n$$

soit une production de G et que

$$X_1\ X_2 \ldots X_{j-1}\ Y\ X_{n+1}\ X_{n+2} \ldots X_t$$

soit une phrase.

($1 \leq j \leq n,\ Y \in V_N,\ 1 \leq n \leq t$)

- On dira que le couple $(Y,\ X_j\ X_{j+1} \ldots X_n)$ est un manche de la phrase $X_1\ X_2 \ldots X_t$

- Si le manche est unique, on peut donner un algorithme simple pour remonter à l'axiome S. En effet, soit une phrase $X_1\ X_2 \ldots X_t$, soit $(Y,\ X_j\ X_{j+1} \ldots X_n)$ le manche de cette phrase, on obtient par définition une nouvelle phrase $X_1\ X_2 \ldots X_{j-1}\ Y\ X_{n+1} \ldots X_t$.

---

* handle pour D. KNUTH. Cette appellation est justifiée par la position du manche dans le dessin d'un arbre syntaxique.

Il suffit maintenant de chercher le manche de cette phrase et ainsi de suite jusqu'à l'axiome S.

Le problème que nous allons étudier est donc la détermination du manche d'une phrase.

- On peut remarquer que si on a commencé avec une phrase terminale, tous les symboles qui se trouvent à droite du deuxième élément du manche sont des symboles terminaux.

- En considérant (2) de la grammaire précédente où l'on substituait toujours le caractère non-terminal le plus à droite, on avait obtenu

(2) $S \rightarrow A \rightarrow a\ C \rightarrow a\ B\ E \rightarrow a\ B\ e \rightarrow a\ b\ c\ d\ e$.

Quel est le manche de a b c d e ?
On regarde d'abord s'il existe une production de la forme $Y \rightarrow a$ puis des productions $Y \rightarrow a\ b$ ou $Y \rightarrow b$ puis $Y \rightarrow a\ b\ c$, $Y \rightarrow b\ c$ ou $Y \rightarrow c$, etc ...
La première production qui convient est $B \rightarrow b\ c\ d$ donc (B, b c d) est le manche de a b c d e car a B e est bien une phrase.
On cherche maintenant le manche de a B e qui est (E, e) et ainsi de suite.
(2) est donc une autre manière de concevoir le manche.

Convention. Soient 1, 2, ..., s les numéros de productions de la grmmaire, $\alpha = X_1\ X_2 \ldots X_t$ une phrase. Supposons que $(Y, X_{r+1}\ X_{r+2} \ldots X_n)$ soit un manche de α et que $Y \rightarrow X_{r+1}\ X_{r+2} \ldots X_n$ soit la $p^{ième}$ production, alors on dira que le couple (n, p) représente ce manche.

On a donc caractérisé le manche par le rang de sa frontière droite et par le numéro de la production.

I - 5 - 2 Réduction.

On appelle réduction la relation inverse d'une production :

$$\theta \succ\!\!- A$$

Par extension on appellera aussi réduction le fait de remplacer dans une chaîne, la sous-chaîne θ par le symbole A.

I - 5 - 3 Définition d'une k-phrase.

Soit $k \geq 0$, " $\dashv$ " un nouveau caractère, $\dashv \notin V$.

- On appelle k-phrase, une phrase suivie de k caractères " $\dashv$ ". Si α est une phrase, $\alpha \dashv^k$ est la k-phrase associée à α.

I - 5 - 4 Définition d'une grammaire LR(k).

Soient deux k-phrases α et β

$\alpha = X_1 X_2 \ldots X_n X_{n+1} X_{n+2} \ldots X_{n+k} Y_1 Y_2 \ldots Y_u$

$\beta = X_1 X_2 \ldots X_n X_{n+1} X_{n+2} \ldots X_{n+k} Z_1 Z_2 \ldots Z_v$

avec $u \geq 0$, $v \geq 0$, $X_{n+1}, X_{n+2}, \ldots, X_{n+k}, Y_1, Y_2 \ldots Y_u, Z_1 Z_2 \ldots Z_v \in V_T \cup \{\dashv\}$.

Supposons que (n, p) soit un manche de α et que (m, q) soit un manche de β.

Une grammaire G sera dite LR(K) si dans un cas comme celui-ci $m=n$ et $p=q$.

En d'autres termes, une grammaire est LR(K) si et seulement si quelque soit le manche, il est uniquement déterminé par la chaîne qui se trouve à sa gauche et les k caractères terminaux qui se trouvent à sa droite. (Le cas k=0 correspond au cas très simple où l'on cherche le plus petit n tel qu'il existe une production $Y \rightarrow X_j X_{j+1} \ldots X_n$)

Une grammaire LR(K) est donc "non-ambiguë". En effet pour une phrase donnée, il n'existe qu'un seul manche, donc il n'y a qu'un seul arbre.

- Illustrons cette définition en fixant k à la valeur 1.

Soit une phrase

$X_1\ X_2 \ldots X_n\ X_{n+1}\alpha \dashv$ telle que

a) toutes les réductions possibles ont été effectuées sur $X_1\ X_2 \ldots X_n$

b) la queue $X_{n+1}\alpha \dashv$ représente la partie de la chaîne non encore examinée.

On veut savoir si, en regardant le prochain caractère (k=1) X_{n+1}, on peut trouver un r tel que $X_r\ X_{r+1} \ldots X_n$, soit le membre droit d'une production et que l'on puisse appliquer cette production dans le contexte X_{n+1}.

- Si la réponse est oui, on saura donc, d'après la définition d'une grammaire LR(1), que ce r et cette production sont uniques.

- Si la réponse est non, on examinera un nouveau caractère pour savoir s'il existe un manche ayant une frontière égale à n+1.

Montrons l'influence de k-caractères sur l'exemple suivant.

1 - S → C d	4 - A → a b
2 - S → a G	5 - F → c e
3 - C → A c	6 - G → b F

Cette grammaire génère le langage {a b c d, a b c e}.

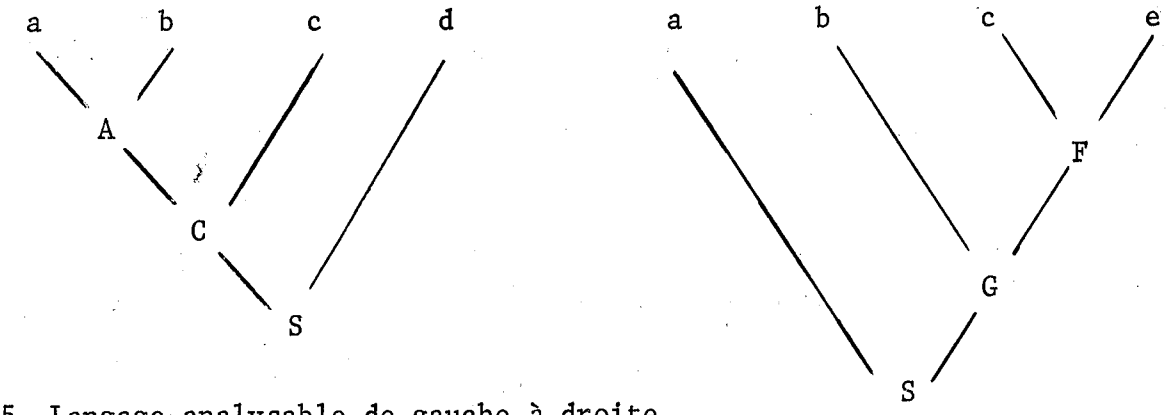
Le manche de a b c d est (A, a b), par contre celui de a b c e est (F, c e).

Supposons que nous cherchions le manche de a b c d.

Examinons d'abord le premier caractère de la phrase. Avec a on ne peut rien faire. On essaye donc avec a b. Il y a bien un manche (A, a b) à condition que l'on soit dans le cas de la phrase a b c d.
Il faut donc aller examiner 2 caractères à droite de b pour savoir si l'on a un "d" ou un "c" en fin de phrase.

Par contre, pour les manches suivants, il n'y aura pas de problème. On dira que la grammaire précédente est LR(2).

On voit d'ailleurs très bien le résultat sur les deux arbres suivants



I - 5 - 5 Langage analysable de gauche à droite.

On appellera langage analysable de gauche à droite un langage défini par une grammaire LR(K).

Donnons maintenant quelques théorèmes intéressants qui ont été démontrés par D.E. KNUTH. [KN.1].

I - 5 - 6 Quelques théorèmes sur les grammaires LR(K).

Théorème 1. L'existence d'un $k \geq 0$, tel que G soit une grammaire LR(K) est un problème indécidable.

Théorème 2. Pour un k donné, on sait décider si une grammaire est LR(K).

Théorème 3. Si G est LR(K) et si G définit le langage L, il existe un D.P.D.A * qui accepte le langage L.

Théorème 4. Si L est un langage déterministe, il existe une grammaire LR(1) qui définit L.

---

* D.P.D.A. : deterministic push-down automaton [G G].

Théorème 5. En conséquence, un langage L est défini par une grammaire LR(K)
- si et seulement si L est un langage déterministe
- si et seulement si L est défini par une grammaire LR(1).

Les langages définis par des grammaires LR(K) étant déterministes et réciproquement, il est donc intéressant d'essayer de donner
- une méthode pour vérifier qu'une grammaire est LR(K) pour un k donné
- une méthode d'analyse des langages analysables de gauche à droite.

I - 6 Analyse des langages analysables de gauche à droite.

I - 6 - 1 Définition de k : α

$k : \alpha = \{$si $\mathcal{L}(\alpha) \geq k$ alors les k premiers caractères de α sinon $\alpha\}$.

I - 6 - 2 Définition de $H_k(\varphi)$

$H_k(\varphi) = \{k : \psi / \psi \in V_T^* \text{ et } \varphi \Rightarrow \psi\}$ (φ chaîne sur V)

Exemple.

Soit la grammaire G

1 -	S → B C D	5 -	C → D
2 -	B → C E	6 -	C → D c
3 -	B → E	7 -	D → d
4 -	C → c	8 -	E → e

$$H_3(S) = \begin{Bmatrix} c\ e\ c, & d\ e\ c, & c\ e\ d, & d\ c\ e, \\ e\ c\ d, & e\ d\ d, & e\ d\ c, & d\ e\ d \end{Bmatrix}$$

Pour une grammaire G et un entier $k \geq 0$ fixé, nous allons donner la construction d'une grammaire $\mathcal{F}_K$ de telle manière que les phrases de $\mathcal{F}_K$ expliciteront toutes les configurations possibles du manche et de k caractères à sa droite.

Nous en déduirons une condition nécessaire et suffisante pour qu'une grammaire G soit LR(K) et nous pourrons ensuite donner une méthode d'analyse fondée sur la construction de $\mathcal{F}_K$.

I - 6 - 3 Construction de $\mathcal{F}_K$.

- α et β seront deux chaînes de longueur k construites sur $V_T \cup \{\dashv\}$.
- p sera un numéro de production de G ayant A_p en partie gauche.

Soit $G = (V_T, V_N, S, \mathcal{P})$, on construit
$\mathcal{F}_K = (V'_T, V'_N, S', \mathcal{P}')$ telle que

1 - $V'_T = V_T \cup V_N \cup \{\dashv\} \cup p$ p numéro de production

2 - $V'_N = \{[A_p ; \alpha]\}$ $A_p \in V_N$.

Les $[A_p ; \alpha]$ sont déterminés progressivement comme suit à partir de $[S ; \dashv^k]$.

3 - $S' = [S ; \dashv^k]$

4 - si $A_p \rightarrow X_{p_1} X_{p_2} \ldots X_{pn_p}$ est une production de G, les productions correspondantes de $\mathcal{F}_k$ seront :

4 - 1

$$[A_p ; \alpha] \rightarrow X_{p_1} X_{p_2} \ldots X_{p(j-1)} [X_{p_j} ; \beta]$$

pour tout j $(1 \leq j \leq n_p)$ tel que $X_{p_j} \in V_N$, et tout (α, β) tel que

- $[A_p ; \alpha] \in V'_N$
- $\beta \in H_K(X_{p(j+1)} X_{p(j+2)} \ldots X_{pn_p} \alpha)$

4 - 2

$$[A_p ; \alpha] \rightarrow X_{p_1} X_{p_2} \ldots X_{pn_p} \alpha p$$

On peut remarquer que l'on obtient un très grand nombre de productions (ce nombre étant néanmoins fini).

Pratiquement, on opère de la manière suivante : à partir des productions de G ayant S en partie gauche, on construit des éléments de V'_N, ou alors V_N ne contient que le seul élément S qui n'apparait jamais en partie droite. Puis, on reprend chaque élément de V'_N ainsi construit, qui eux-mêmes donneront de nouveaux éléments de V'_N. On continue ainsi jusqu'à épuisement de tous les éléments du nouveau vocabulaire non-terminal. On voit qu'en même temps, les chaînes α se constituent automatiquement.

Nous allons maintenant donner un lien entre $\mathcal{F}'_k$ et G en démontrant le lemme suivant.

I - 6 - 4 Lemme 1.

$S_i ; {}^{k} X_1 X_2 \ldots X_n X_{n+1} \ldots X_{n+k}$ dans $\mathcal{F}'_k$ si et seulement si il existe une k-phrase de G : $X_1\ X_2 \ldots X_n\ X_{n+1} \ldots X_{n+k}\ Y_1\ Y_2 \ldots Y_u$ dont le manche soit (n, p)

($X_{n+1}, X_{n+2}, \ldots, X_{n+k}, Y_1, Y_2, \ldots Y_u \in V_T \cup \{\dashv\}$, $X_1, X_2, \ldots X_n \in V$, $u \geq 0$)

Démonstration.

a) $[S ; \dashv^{k}] \Rightarrow X_1\ X_2 \ldots X_n\ X_{n+1} \ldots X_{n+k} p$

par définition de la relation $\Rightarrow$, il existe une suite finie de chaînes $\beta_0, \beta_1, \ldots, \beta_q$ telle que

$$[S ; \dashv^{k}] = \beta_0 \rightarrow \beta_1 \rightarrow \ldots \rightarrow \beta_{q-1} \rightarrow \beta_q = X_1\ X_2 \ldots X_{n+k} p$$

- q=1

$$[S ; \dashv^{k}] \rightarrow X_1\ X_2 \ldots X_n\ X_{n+1} \ldots X_{n+k} p$$

Par définition des productions de $\mathcal{F}_k$, on a $X_{n+1}\ X_{n+2} \ldots X_{n+k} = \dashv^{k}$ et il existe dans G la production numéro p :

$$S \rightarrow X_1\ X_2 \ldots X_n$$

Il existe donc bien dans G une k-phrase $X_1\ X_2 \ldots X_n \dashv^{k}$ avec manche (n, p)

- q > 1

Dans $\mathcal{G}_k$ on a :

- $\beta_t = X_1 X_2 \ldots X_{r_{t-1}} [X_{r_t} ; \alpha_t]$ $0 < t \leq q-1$

avec $\alpha_t \in H_k(X_{r_{t+1}} X_{r_{t+2}} \ldots X_{r_t+pr_t} \alpha_{(t-1)})$

- $\beta_q = X_1 X_2 \ldots X_n X_{n+1} \ldots X_{n+k} p.$

On passe de β_{q-1} à β_q par application d'une production de $\mathcal{G}_k$ de la forme

$[A_p ; \alpha] \rightarrow X_{p_1} X_{p_2} \ldots X_{pn_p} \alpha p.$ $(\alpha = \alpha_{q-1})$

Aux chaînes $\beta_1, \beta_2, \ldots, \beta_q$ de $\mathcal{G}_k$ on peut faire correspondre des chaînes $\gamma_1, \gamma_2, \ldots, \gamma_q$ de G avec

$$\gamma_t = X_1 X_2 \ldots X_{r_{t-1}} X_{r_t} X_{r_{t+1}} \ldots X_{r_{t+pr_t}} \ldots$$

$$0 < t \leq q$$

telles que

$$- S \dashv^k = \gamma_0 \rightarrow \gamma_1 \rightarrow \ldots \rightarrow \gamma_2 \rightarrow \ldots \rightarrow \gamma_3 \rightarrow \ldots \rightarrow \ldots \rightarrow \gamma_{q-1} \rightarrow \ldots \rightarrow \gamma_q$$

Avant de passer de γ_t à γ_{t+1}, on applique un certain nombre de productions de G telles qu'à droite de X_{r_t} il n'y ait que des symboles terminaux et telles que l'on puisse écrire

$$X_1 X_2 \ldots X_{r_{t-1}} X_{r_t} \alpha_t \ldots$$

On passera ainsi de γ_{q-1} à γ_q par application de la production numéro p :

$$A_p \rightarrow X_{p_1} X_{p_2} \ldots X_{pn_p}$$

telle que

$$\gamma_q = X_1 X_2 \ldots X_n X_{n+1} \ldots X_{n+k} Y_1 Y_2 \ldots Y_u$$

avec $X_1 X_2 \ldots X_n \in V$, $X_{n+1} X_{n+2} \ldots Y_u \in V_T \cup \{\dashv\}$

et $X_{p_1} X_{p_2} \ldots X_{pn_p} = X_j X_{j+1} \ldots X_n$

Il existe donc bien une k-phrase de G avec manche (n, p).

b) Soit une k-phrase de G :

$X_1\ X_2\ \ldots\ X_n\ X_{n+1}\ \ldots\ X_{n+k}\ Y_1\ Y_2\ \ldots\ Y_u$

avec manche (n, p).

Par définition, il existe donc une suite finie de chaînes $\gamma_0, \gamma_1, \ldots, \gamma_q$

telle que

$S \dashv^k = \gamma_0 \rightarrow \gamma_1 \rightarrow \ldots \rightarrow \gamma_2 \rightarrow \ldots \rightarrow \gamma_{q-1} \rightarrow \ldots \rightarrow \gamma_q$

avec $\gamma_q = X_1\ X_2\ \ldots\ X_n\ X_{n+1}\ \ldots\ X_{n+k}\ Y_1\ Y_2\ \ldots\ Y_u$

- q=1.

On a donc forcément u=0 et $X_{n+1}\ X_{n+2}\ \ldots\ X_{n+k} = \dashv^k$.

Donc $S \rightarrow X_1\ X_2\ \ldots\ X_n$ est la production numéro p et il existe dans $\mathfrak{F}_k$ la production $[S, \dashv^k] \rightarrow X_1\ X_2\ \ldots\ X_n \dashv^k p$

Le lemme est donc démontré pour q=1.

- q > 1

par le même raisonnement que précédemment on fait correspondre aux chaînes $\gamma_0, \gamma_1, \ldots, \gamma_q$ de G des chaînes $\beta_0, \beta_1, \ldots \beta_q$ de $\mathfrak{F}_k$ telles que

- si $\gamma_t = X_1\ X_2\ \ldots\ X_{r_{t-1}}\ X_{r_t}\ X_{r_{t+1}}\ \ldots\ X_{r_{t+pr_t}}\ \ldots$

(on suppose que $X_{r_{t+1}}, \ldots, X_{r_{t+pr_t}} \in V_T \cup \{\dashv\}$

sinon on applique un certain nombre de productions de G pour qu'il en soit ainsi).

alors $\beta_t = X_1\ X_2\ \ldots\ X_{r_{t-1}}\ [X_{r_t}, \alpha_t]$

α_t étant choisie telle que l'on puisse écrire

$\gamma_t = X_1\ X_2\ \ldots\ X_{r_{t-1}}\ X_{r_t}\ \alpha_t\ \ldots$

$- [S ; \dashv^k] = \beta_0 \rightarrow \beta_1 \rightarrow \ldots \rightarrow \beta_{q-1} \rightarrow \beta_q$

Le manche étant (n, p) on est donc passé de γ_{q-1} à γ_q par application de la production numéro p :

$A_p \rightarrow X_{p_1}\ X_{p_2}\ \ldots\ X_{pn_p} \quad (A_p = X_{r(q-1)})$

telle que $\gamma_{q-1} = X_1\ X_2 \ \dots\ X_{r(q-1)}\ X_{n+1}\ X_{n+2} \ \dots\ X_{n+k} \ \dots$

et $\gamma_q = X_1\ X_2 \ \dots\ X_{p_1}\ X_{p_2} \ \dots\ X_{pn_p}\ X_{n+1}\ X_{n+2} \ \dots\ Y_1 \ \dots\ Y_u$

à la production numéro p de G il correspond une production de $\mathcal{F}_k$ unique

$$[A_p\ ;\ \alpha] \rightarrow X_{p_1}\ X_{p_2} \ \dots\ X_{pn_p}\ \alpha p$$

avec $\alpha = X_{n+1}\ X_{n+2} \ \dots\ X_{n+k}$

et on passera de β_{q-1} à β_q par application de cette production.

On a donc bien

$$[S\ ;\ \dashv^k] \Rightarrow X_1\ X_2 \ \dots\ X_n\ X_{n+1} \ \dots\ X_{n+k}\ p.$$

I - 6 - 5 Exemple de construction de $\mathcal{F}_k$.

Soit G la grammaire suivante *

1 -	S → a A c	3 -	A → a S c
2 -	S → b	4 -	A → b

Cette grammaire génère le langage $a^n\ b\ c^n (n \geq 0)$. Nous allons voir que lorsque n est pair, le premier manche de $a^n\ b\ c^n$ est (S, b), par contre lorsque n est impair, le premier manche est (A, b).

Construisons $\mathcal{F}_1$

[S ; ⊣]	→ a [A ; c]		
[S ; ⊣]	→ a A c ⊣ 1	[A ; c]	→ b c 4
[S ; ⊣]	→ b ⊣ 2	[S ; c]	→ a [A ; c]
[A ; c]	→ a [S; c]	[S ; c]	→ a A c c 1
[A ; c]	→ a S c c 3	[S ; c]	→ b c 2

---

* c'est un exemple de D.E. KNUTH [K n-1] , on verra plus loin que cette grammaire est LR(0).

explicitons les phrases terminales de $\mathcal{F}_1$

(1) $[S\ ;\dashv] \rightarrow a\ [A\ ;\ c] \rightarrow a^2\ [S\ ;\ c] \rightarrow \ldots\ldots \rightarrow a^{2p}\ S\ ;\ c \rightarrow a^{2p+1}\ A\ c\ c\ 1$

(2) $[S\ ;\dashv] \rightarrow a\ [A\ ;\ c] \rightarrow \ldots\ldots\ldots \rightarrow a^{2p}\ [S\ ;\ c] \rightarrow a^{2p}\ b\ c\ 2$

(3) $[S\ ;\dashv] \rightarrow a\ [A\ ;\ c] \rightarrow \ldots\ldots\ldots \rightarrow a^{2p-1}\ [A\ ;\ c] \rightarrow a^{2p}\ S\ c\ c\ 3$

(4) $[S\ ;\dashv] \rightarrow a\ [A\ ;\ c] \rightarrow \ldots\ldots\ldots \rightarrow a^{2p+1}\ [A\ ;\ c] \rightarrow a^{2p+1}\ b\ c\ 4$

(5) $[S\ ;\dashv] \rightarrow b \dashv 2$ avec $p \geq 1$

(6) $[S\ ;\dashv] \rightarrow a\ A\ c \dashv 1$

Par exemple, on interprète (2) de la façon suivante : soit une phrase $a^{2p}\ b\ c^{2p}$, la première réduction que l'on pourra appliquer est $b \rangle\!\!- S$ à condition qu'après b il y ait un c, le numéro de la production étant égal à 2. On obtient donc une nouvelle phrase $a^{2p}\ S\ c^{2p}$.
de la même manière, avec (3), on peut dire que la réduction que l'on doit appliquer maintenant est $a\ S\ c \rangle\!\!- A$ (numéro 3 avec contexte c) on obtient $a^{2p-1}\ A\ c^{2p-1}$ etc ...

Toutes les phrases de $\mathcal{F}_1$ explicitent bien tous les manches possibles ainsi qu'un caractère à la droite de ces manches. Ici, on voit très bien, lorsque n est pair, que le premier manche est (S, b), par contre, lorsque n est impair on trouve (A, b).
Ensuite, les manches sont alternativement (S, a A c) et (A, a S c).

Donnons maintenant une condition nécessaire et suffisante sur les phrases de $\mathcal{F}_k$ pour qu'une grammaire G soit LR(K).

I - 6 - 6 Théorème 6.

> G est une grammaire LR(K) si et seulement si la grammaire $\mathcal{F}_k$ satisfait à la propriété suivante :
>
> (1) si $[S\ ;\dashv\ ^k] \Rightarrow^{\theta} p$ et
> si $[S\ ;\dashv\ ^k] \Rightarrow \theta\ \varphi\ q$ alors $\varphi = \Lambda$ et $p = q$

Démonstration.

D'après le lemme 1 $[S ; \dashv^{k}] \Rightarrow \theta\ p$ si et seulement si il existe une k-phrase de $G, X_1\ X_2 \ldots X_n\ X_{n+1} \ldots X_{n+k}\ Y_1\ Y_2 \ldots Y_u$ avec manche (n, p), en posant $\theta = X_1\ X_2 \ldots X_n\ X_{n+1} \ldots X_{n+k}$.

De même $[S ; \dashv^{k}] \Rightarrow \theta\ \varphi\ q$ si et seulement si il existe une k-phrase de G

$X_1\ X_2 \ldots X_n\ X_{n+1} \ldots X_{n+k}\ \varphi\ Z_1\ Z_2 \ldots Z_v$ avec manche (m, q).

1) $\mathcal{F}'_k$ satisfait à (1).

On a $n = \mathcal{L}(\theta) - k$,

et $m = \mathcal{L}(\theta\ \varphi) - k = \mathcal{L}(\theta) + \mathcal{L}(\varphi) - k$.

- $\varphi = \Lambda$ entraîne que $m = n$.
- $p = q$, les deux manches sont donc identiques.

Par définition, G est LR(K).

2) G est LR(K), $[S ; \dashv^{k}] \Rightarrow \theta\ p$ et $[S ; \dashv^{k}] \Rightarrow \theta\ \varphi\ p$

Par définition, les deux manches (n, p) et (m, q) doivent être identiques, donc $p = q$ et $m = n = \mathcal{L}(\theta) - k = \mathcal{L}(\theta) + \mathcal{L}(\varphi) - k$.

Il s'ensuit que $\varphi = \Lambda$. La condition (1) est bien vérifiée.

I - 6 - 7 Théorème 7.

Une grammaire LR(K) est LR(K+1).

Démonstration.

Montrons que si G n'est pas LR(K+1), G n'est pas LR(K).

D'après le théorème 6, si G n'est pas LR(K+1), on peut trouver des chaînes θ et φ, et des entiers p et q tels que

$[S ; \dashv^{k+1}] \Rightarrow \theta\ p$, $[S ; \dashv^{k+1}] \Rightarrow \theta\ \varphi\ q$

avec 1) $\varphi = \Lambda$, $p \neq q$

ou 2) $\varphi \neq \Lambda$, $p = q$

ou 3) $\varphi \neq \Lambda$, $p \neq q$

1) $\varphi = \Lambda$, $p \neq q$

Par définition de $\mathcal{F}_{k+1}$ et de $\mathcal{F}_k$ on peut donc écrire

$[S ; \dashv^k] \Rightarrow \theta' p \quad [S ; \dashv^k] \Rightarrow \theta' q$

avec $\theta = \theta' a \quad a \in V_T \cup \{\dashv\}$

p étant différent de q, G n'est pas LR(K).

2) $\varphi \neq \Lambda$, $p = q$

Dans $\mathcal{F}_k$ on peut écrire

$[S ; \dashv^k] \Rightarrow \theta' p \quad [S ; \dashv^k] \Rightarrow \theta' \varphi' p$

avec $\theta = \theta' a$, $\varphi' = a \varphi''$

$\varphi = \varphi'' b$ avec $\mathcal{L}(\varphi'') \geq 0$

$(a, b \in V_T \cup \{\dashv\})$

φ' est $\neq \Lambda$, G n'est pas LR(K).

3) $\varphi \neq \Lambda$, $p \neq q$

évident d'après 1) et 2).

Nous pensons que le résultat est évident d'après la définition du manche : en effet, si k caractères suffisent à déterminer le manche, il sera encore uniquement déterminé si on ajoute un caractère à la chaîne de longueur k qui sert à prendre la décision. Il a été tout de même intéressant de montrer qu'avec $\mathcal{F}_k$, on retrouve un résultat convenable.

I - 6 - 8 Exemples d'applications du théorème 6.

Nous allons nous servir du théorème 6 pour deux exemples : Une grammaire qui est LR(0), une grammaire qui définit un langage non déterministe.

1 - Reprenons l'exemple (I - 6 - 5)
Pour simplifier prenons k=0.

Convention. On écrira $[A_p]$ pour $[A_p ; \Lambda]$.

[S] → a [A]	[A] → a [S]
[S] → a A c 1	[A] → a S c 3
[S] → b 2	[A] → b 4

On peut remarquer que dans le cas $\mathcal{F}_0$, le nombre de productions de $\mathcal{F}_0$ est égal au nombre de productions de G plus le nombre de symboles non-terminaux apparaissent en partie droite des productions de G.

Explicitons les phrases terminales de $\mathcal{F}_0$.

$$[S] \to a\ [A] \to a^2\ [S] \to \ldots \to a^{2p}\ [S] \to a^{2p+1}\ A\ c\ 1$$
$$[S] \to a\ [A] \to a^2\ [S] \to \ldots \to a^{2p+1}\ [A] \to a^{2p+2}\ S\ c\ 3$$
$$[S] \to \ldots\ldots\ldots\ldots\ldots\ldots \to a^{2p}\ [S] \to a^{2p}\ b\ 2$$
$$[S] \to \ldots\ldots\ldots\ldots\ldots\ldots \to a^{2p+1}\ [A] \to a^{2p+1}\ b\ 4$$

avec $p \geq 0$

La condition (1) du théorème 6 est bien vérifiée, la grammaire est donc LR(0).

2 - Donnons un exemple de langage non-déterministe.

Prenons le langage miroir défini par la grammaire suivante :

1 - S → a S a	3 - S → a a
2 - S → b S b	4 - S → b b

Cette grammaire génère le langage

$$x\,x^{-}$$

où x est une chaîne quelconque construite sur {a, b} et x^{-} est la chaîne inverse de x.

Construction de $\mathcal{T}'_1$

[S ;⊣] → a [S ; a]	[S ; a] → b S b a 2
[S ;⊣] → a S a ⊣ 1	[S ; a] → a a a 3
[S ;⊣] → b [S ; b]	[S ; a] → b b a 4
[S ;⊣] → b S b⊣ 2	[S ; b] → a [S ; a]
[S ;⊣] → a a⊣ 3	[S ; b] → a S a b 1
[S ;⊣] → b b ⊣ 4	[S ; b] → b [S ; b]
[S ; a] → a [S ; a]	[S ; b] → b S b b 2
[S ; a] → a S a a 1	[S ; b] → a a b 3
[S ; a] → b [S ; b]	[S ; b] → b b b 4

Vérifions que la grammaire n'est pas LR(1).

Par exemple :

[S ;⊣] → a [S ; a] → a b [S ; b] → a b b b b 4 = θ 4

[S ;⊣] → a [S ; a] → a b [S ; b] → a b b [S ; b]

→ a b b b b b 4 = θ b 4

- ici on est donc dans le cas où $\varphi \neq \Lambda$ et p = q

On peut également trouver des cas où $\varphi \neq \Lambda$ et p ≠ q

par exemple :

[S ;⊣] → a [S ; a] → a b [S ; b] → a b b [S ; b]

→ ..... → a b^n [S ; b] → a b^n a [S ; a]

→ a b^n a a a a 3

donc avec les phrases a b^m 4 ($m \leq n$) on est dans le cas où $\varphi \neq \Lambda$ et $p \neq q$.

Vu la construction de $\mathcal{F}_k$ nous pensons que cette grammaire n'est pas LR(k) pour tout $k \geq 0$. (L(G) n'étant pas déterministe, G n'est pas LR(K). th. 5).

Nous allons maintenant donner une méthode d'analyse des langages analysables de gauche à droite. Cette méthode consiste en la construction d'une table d'états dont le principe est tout à fait semblable à la construction de $\mathcal{F}_k$. Ensuite, cette table, nous permettra, lorsque la grammaire sera LR(K), d'analyser des chaînes quelconques.

I - 6 - 9 Construction d'une table d'états.

On modifie légèrement la grammaire G en lui ajoutant une production numéro zéro : $S_0 \rightarrow S \dashv^k$. Cette production a pour but de transformer les phrases de G en k-phrases. On prendra S_0 comme nouvel axiome.

On se propose de construire une table d'états contenant le maximum d'information relative à la façon dont se font les dérivations dans G. On part de S_0, et on parcourt toutes les productions de G de la même façon que pour la construction de $\mathcal{F}_k$.

Avant de donner la construction de la table, expliquons brièvement la méthode d'analyse.

Soit une chaîne $X_1\ X_2\ \ldots\ X_n$.
et une pile dont la configuration initiale est :

$$Q_0 \mid X_1\ X_2\ \ldots\ X_n \dashv^k$$

à gauche du trait vertical, on a l'état initial Q_0, à droite la chaîne à analyser suivie de k-caractères "$\dashv$".
Au départ, on translate X_1 à gauche du trait et on se trouve dans un état Q_1
$Q_0\ X_1\ Q_1 \mid X_2\ \ldots\ X_n \dashv^k$.

Maintenant, on veut savoir si en examinant $X_2\ X_3\ \ldots\ X_{k+1}$ il existe un manche (A_p, X_1).
Si c'est oui, on appliquera la réduction correspondante et on se retrouvera (dans ce cas) dans l'état Q_0, si c'est non, on translatera X_2 à gauche et on se trouvera dans un état Q_2, et ainsi de suite.
Il est donc intéressant de construire un ensemble $Z_p(Q_j)$ dont les éléments sont des chaînes de longueur k sur $V_T \cup \{\dashv\}$ qu'il suffira de consulter pour savoir s'il existe un manche ou non.
D'autre part, lorsqu'on translate X_{j+1}, il sera également intéressant de savoir si la chaîne $X_1\ X_2\ \ldots\ X_{j+1}$ est telle qu'il existe une k-phrase de G commençant par

$$X_1\ X_2\ \ldots\ X_{j+1}\ X_{j+2}\ \ldots\ X_{j+k}$$

Donc on constuira également un ensemble $Z(Q_j)$ formé de chaînes de longueur k sur $V_T \cup \{\dashv\}$ qui soit tel que $X_{j+1}\ X_{j+2}\ \ldots\ X_{j+k} \in Z(Q_j)$ s'il existe une k-phrase de G

$$X_1\ X_2\ \ldots\ X_j\ X_{j+1}\ \ldots\ X_{j+k}\ \ldots \dashv^k$$

Enfin, on construira les états $Q_0, Q_1\ \ldots\ Q_n$.

Au cours de la construction de la table, on donnera un moyen de vérifier que la grammaire est LR(K).

L'algorithme d'analyse, dont nous avons donné rapidement un aperçu, est simple et rapide.

La description de cet algorithme et son application à des cas réels constituent la majeure partie de cet ouvrage.

1 - Construction de la table.

Comme nous l'avons dit, cette table sera constituée d'états (en réalité d'ensembles d'états) et des ensembles Z et Z_p.

α) Définition d'un état.

Soit s le nombre de productions de la grammaire G.

Nous appellerons état le triplet $[p, j ; \alpha]$ avec

- p numéro de production. $0 \leq p \leq s$
- j indice de position dans une production.
- α une chaîne de longueur k construite sur $V_T \cup \{\dashv\}$.

Remarque sur l'indice de position.

$j = 0$ représente la partie gauche de la production

$j = 1, 2, \ldots n_p$ repère les éléments de la partie droite de la production, n_p étant la longueur de la $p^{ième}$ production.

Par définition, on sera dans l'état $[p, j ; \alpha]$, si la chaîne analysée a la forme $\beta X_{p_1} X_{p_2} \ldots X_{p_j} \ldots$ et si la grammaire G peut générer une phrase de la forme $\beta A_p \alpha \ldots\ldots$ avec $A_p \rightarrow X_{p_1} X_{p_2} \ldots X_{p_j} \ldots X_{pn_p}$

On a donc trouvé les j premiers caractères de la production numéro p ayant une partie gauche égale à A_p, α est une chaîne de longueur k qui peut suivre la production en question.
Cette définition ne s'applique ni à l'état initial ni aux états intermédiaires que nous allons définir.

β) Construction des ensembles d'états.

β1) On pose $Q_0 = [0, 0 ; \dashv^k]$ qui correspond à l'état initial.

β2) Définitions.

On appelle état intermédiaire le triplet $[q, 0 ; \beta]$. Un tel état correspond à la partie gauche de la production numéro q de G.

On appelle ensemble d'états S un ensemble formé d'états $(p, j ; \alpha)$ avec $1 \leq j \leq n_p$.

Les ensembles d'états S_i se construisent récursivement à partir de l'état initial Q_0.

Supposons que nous ayons construit un certain nombre d'ensembles S_1, S_2, ... S_n, ..., et que nous voulions construire de nouveaux ensembles S_i $(i > n)$ avec l'ensemble S_n.

β3) Construction des états intermédiaires.

On construit un ensemble S'_n de la manière suivante :

$$S'_n = S_n \cup \{[q, 0 ; \beta] \mid \exists [p, j ; \alpha] \in S'_n$$

avec $0 \leq j \leq n_p$, $X_{p(j+1)} = A_q$ et

$$\beta \in H_k(X_{p(j+2)} \ldots X_{pn_p} \alpha)\}$$

S'_n est donc l'union de S_n et de l'ensemble des états intermédiaires qui lui correspondent, la construction étant rendue récursive par l'apparition de S'_n dans sa propre définition.

Supposons que S_n ne contienne qu'un seul état $[p, j ; \alpha]$ qui soit tel que $X_{p(j+1)} \in V_N$ et $j < n_p$, on cherche dans G toutes les productions ayant une partie gauche A_q égale à $X_{p(j+1)}$.

On note le numéro de production q, la position j(j=0), et β est une chaîne de longueur k qui peut suivre cette production quand elle est appliquée à $X_{p(j+1)}$ dans la production p. Ensuite, si $X_{q_1} \in V_N$, on recommence avec cet élément et ainsi de suite.

Si S_n contient plusieurs états, la même construction est réalisée pour tous les états de S_n.

β4) Construction de $\ldots(S_n)$.

$$\ldots(S_n) = \{[p, j+1 ; \alpha] \mid \exists [p, j ; \alpha] \in S'_n\}$$

β5) Constructions des ensembles d'états.

Si jusqu'ici on a créé d ensembles d'états, alors avec S_n on construit S_{d+1}, S_{d+2}, ..., S_{d+r} tels que :

$$S_{d+1} \cup S_{d+2} \cup \ldots S_{d+r} = R(S_n)$$

$$S_{d+i} \cap S_{d+j} = \emptyset \text{ pour tout } i \neq j$$

$$(i=1, 2, \ldots \nu,\ j=1, 2, \ldots \nu)$$

Les S_{d+i} étant formés d'états $[p, j ; \alpha]$ appartenant à $R(S_n)$ tels que tous les X_{p_j} correspondant à un ensemble S_{d+i} sont identiques.

§6) Construction de Z_n, Z_n^0, Z_n^1, ... Z_n^p ... Z_n^s.

§6 - 1)

$$Z_n = \{\beta \mid \exists [p, j ; \alpha] \in S'_n,\ j < n_p,\ \beta \in H_k(X_{p(j+1)} \ldots X_{pn_p} \alpha)\}$$

Z_n est constitué de toutes les chaînes de longueur k construites sur $V_T \cup \{\dashv\}$ qui peuvent suivre X_{p_j} et la présence d'une telle chaîne après X_{p_j} dans la pile d'analyse indique qu'il n'y a pas encore de manche (puisque $j < n_p$).

§6 - 2)

$$Z_n^p = \{\alpha \mid \exists [p, n_p ; \alpha] \in S'_n\}$$

Z_n^p est formé de toutes les chaînes de longueur k construites sur $V_T \cup \{\dashv\}$ qui peuvent suivre X_{pn_p}, et la présence d'une telle chaîne après X_{pn_p} dans la pile d'analyse indique qu'il y a un manche (n_p : on est en fin de production), donc une réduction à appliquer avec la production numéro p.

On a alors immédiatement :

Théorème 8.

Pour que la grammaire soit LR(K), il faut et il suffit que Z_n, Z_n^0, Z_n^1, ... Z_n^p ... Z_n^s soient disjoints deux à deux pour tout n.

Cas particulier k=0.

Les ensembles Z_n, Z_n^0, ... Z_n^s contiennent tous la chaîne vide.

Pour que la grammaire soit LR(0), il faut et il suffit que S'_n contienne soit un seul état de la forme $[p, n_p]$ *, soit plusieurs états de la forme $[p, j]$ * avec $j < n_p$.

L'état final est $[0, 1 ; \dashv^k]$ qui correspond à $S\ (0 : S_0 \rightarrow S \dashv^k)$ et on aura par convention $S_1 = \{[0, 1 ; \dashv^k]\}$.

I - 6 - 10 Analyse d'une chaîne quand G est LR(K).

Soit une chaîne $X_1\ X_2 \ldots X_n \ldots$
Au départ on dispose d'une pile d'analyse

$$Q_0 | X_1\ X_2 \ldots X_n \ldots \dashv^k$$

à gauche du trait vertical on met l'état initial, à droite la chaîne à analyser suivie de k caractères " $\dashv$ ".

Supposons que nous ayons analysé n caractères de la chaîne d'entrée et admettons que la pile d'analyse ait la forme suivante :

$$Q_0\ X^1\ S^1\ X^2\ S^2\ X^3 \ldots X^m\ S^m | Y_1\ Y_2 \ldots Y_k\ \omega \quad (1)$$

tout ce qui se trouve à gauche du trait vertical est formé alternativement d'ensembles d'états S^i et de caractères X^i. A droite, on a mis en évidence k caractères $Y_1\ Y_2 \ldots Y_k$ qui seront utilisés pour diriger l'analyse, ω représente la partie de la chaîne non encore examinée. On suppose que si on a translaté n caractères de la chaîne d'entrée on a effectué toutes les réductions possibles sur $X_1\ X_2 \ldots X_n$.

---

* On écrit $[p, j]$ pour $[p, j ; \Lambda]$

C'est pourquoi nous les avons distingués par $X^1\ X^2\ \ldots\ X^m$. En effet $X^i \in V$ alors que $X_i \in V_T$.

On se trouve dans l'ensemble d'états S^m égal à S_i (un ensemble d'états de la table) 3 possibilités peuvent se présenter.

α) $Y_1\ Y_2\ \ldots\ Y_k \in Z_i$.

D'après la construction de Z_i, il n'y a donc pas encore de manche à gauche du trait, on translate donc Y_1 à gauche.

La pile devient : $Q_0\ X^1\ S^1\ \ldots\ X^m\ S^m\ Y_1 | Y_2\ \ldots\ Y_k\ \omega$
Y_1 devient X^{m+1} et Y_{j+1} devient Y_j pour $j=1\ \ldots\ k-1$.
Le premier caractère de la chaîne ω devenant Y_k d'où

$$Q_0\ X^1\ S^1\ \ldots\ X^m\ S^m\ X^{m+1} | Y_1\ Y_2\ \ldots\ Y_k\ \omega'$$

Au vu de X^{m+1} on passe alors dans l'ensemble d'états S^{m+1} défini par :

$$S^{m+1} = \{[p,\ j\ ;\ \alpha] | (X_{p_j} = X^{m+1}) \cap [p,\ j\ ;\ \alpha] \in R(S^m),\ S^m = S_i\}$$

La pile devient

$$Q_0\ X^1\ S^1\ \ldots\ X^m\ S^m\ X^{m+1}\ S^{m+1} | Y_1\ Y_2\ \ldots\ Y_k\ \omega'$$

et on est ramené à la forme (1).

β) Supposons maintenant que $Y_1\ Y_2\ \ldots\ Y_k \in Z_i^p$. Il y a donc un manche (m, p) à gauche du trait vertical, on peut donc appliquer une réduction avec la $p^{\text{ième}}$ production.
Cette production ayant une longueur n_p, on pose $r = m - n_p$.
La pile contient donc immédiatement à gauche du trait (abstraction faite des ensembles d'états) la chaîne $X^{r+1}\ X^{r+2}\ \ldots\ X^{r+n_p}$ qui est telle que

$$A_p \rightarrow X^{r+1}\ X^{r+2}\ \ldots\ X^{r+n_p}$$

On remplace cette chaîne par A_p. La pile devient

$$Q_0\ X^1\ S^1\ \ldots\ X^r\ S^r\ A_p \mid Y_1\ Y_2\ \ldots\ Y_k\ \omega$$

on pose m=r, A_p devient X^{m+1}, et à partir de l'ensemble d'états S^m et au vu de X^{m+1} on passe dans l'ensemble d'états S^{m+1}.

La pile devient $Q_0\ X^1\ S^1\ \ldots\ X^{m+1}\ S^{m+1} \mid Y_1\ Y_2\ \ldots\ Y_k\ \omega$ et on est ramené à la forme (1).

$$\gamma)\ Y_1\ Y_2\ \ldots\ Y_k \notin Z_i \cup Z_i^p \quad (p=0,\ 1,\ \ldots\ s)$$

la chaîne est donc incorrecte, elle n'appartient pas à L(G) et l'analyse est terminée.

Dans le cas d'une phrase, l'analyse sera terminée lorsque l'on aura $Y_1\ Y_2\ \ldots\ Y_k = \dashv^k$ et que l'on sera dans l'ensemble d'états $S^1 = S_1 = \{[0,\ 1, \dashv^k]\}$. En effet si on est dans l'ensemble S_1, cela veut dire que l'on est remonté jusqu'à l'axiome S.

I - 6 - 11 Exemple de construction de la table.

Prenons l'exemple donné par D.E. KNUTH [Kn1].

Soit G_1 la grammaire suivante :

0	S → B⊣		
1	L → a	5	R → a
2	L → L N b	6	R → b N R
3	B → a	7	N → a
4	B → L R	8	N → b N N b

On construit la table pour k=1. Dans le cas particulier des grammaires LR(1), on prendra la convention suivante : si $[p,\ j\ ;\ a_1]$, $[p,\ j\ ;\ a_2]$, ..., $[p,\ j\ ;\ a_i]$ sont des états, on regroupe ces états en un seul :

$$[p,\ j\ ;\ a_1\ a_2\ \ldots\ a_i] \quad (a_j \in V_T \cup \{\dashv\},\ j=1\ \ldots\ i)$$

La table est :

n	S_n	$S'_n - S_n$	Z_n	p	Z_n^p	ACTION	X_{n+1}	$R(S_n)$
0	0 0 ⊣	30 ⊣, 40 ⊣ 10 a b, 20 a b	a			TRANSLATER	B a L	0 1 ⊣ 31 ⊣ , 11 a b 41 ⊣ , 21 a b
1	0 1 ⊣			0		STOP		
2	31 ⊣ , 11a b			3 1	 a, b	REDUIRE 3 REDUIRE 1	R	42
3	41 , 21 a b	50 , 60 70 b, 80 b	a, b			TRANSLATER	N a b	22 a b 51 , 71 b 61 , 81 b
4	42 ⊣			4	⊣	REDUIRE 4		
5	22 a b		b			TRANSLATER	b	23 a b
6	51 ⊣ , 71 b			5 7	 b	REDUIRE 5 REDUIRE 7	N	62 , 82 b
7	61 ⊣ , 81 b	70 a b, 80 a b	a, b			TRANSLATER	a b	71 a b 81 a b
8	23 a b			2	a, b	REDUIRE 2		
9	62 ⊣ , 82 b	50 ⊣ , 60 ⊣ 70 b, 80 b	a, b			TRANSLATER	R N a b	63 83 b 51 , 71 b 61 , 81 b
10	71 a b			7	a, b	REDUIRE 7		
11	81 a b	70 a b, 80 a b	a, b			TRANSLATER	N a b	82 a b 71 a b 81 a b
12	63 ⊣			6		REDUIRE 6		
13	83 b		b			TRANSLATER	b	84 b
14	82 a b	70 b, 80 b	a, b			TRANSLATER	N a b	83 a b 71 b 81 b
15	84 b			8	b	REDUIRE 8		
16	83 a b		b			TRANSLATER	b	84 a b
17	84 a b			8	a, b	REDUIRE 8		

Pour chaque ensemble d'états S_n, les ensembles Z_n, Z_n^0, ... Z_n^p ... Z_n^s sont disjoints deux à deux, la grammaire est donc LR(1).
On peut voir également sur la table que la grammaire n'est pas LR(0). En effet, regardons par exemple ce qui se passe pour l'ensemble d'états $S_2 = \{31 \dashv , 11\ a\ b\}$.
Nous avons $Z_2^1 = \{a, b\}$, $Z_2^3 = \{\dashv\}$.
Dans le cas LR(0) ces deux ensembles seraient vides et S'_n contiendrait deux états $[p, n_p]$ et $[r, n_r]$. On aurait donc le choix entre deux réductions.
Il nous faut donc bien un caractère pour prendre la décision. Le même choix se retrouverait pour l'ensemble d'états S_6.

Remarque sur les ensembles d'états.

Nous avons obtenu les ensembles $S_{11} = \{81\ a\ b\}$ et $S_7 = \{61 \dashv , 81\ b\}$.
Ces deux ensembles sont tels que $X_{81} = X_{61} = b$.
On peut regrouper ces deux ensembles en un seul : $S = \{61 \dashv , 81\ a\ b\}$. En effet avec cet ensemble on crée $\mathcal{R}(S)$ qui est tel que

$\mathcal{R}(S_7) \subseteq \mathcal{R}(S)$ et $\mathcal{R}(S_{11}) \subseteq \mathcal{R}(S)$.

On fait de même avec

$S_{10} = \{71\ a\ b\}$ et $S_6 = \{51 \dashv , 71\ b\}$

en construisant $\{51 \dashv , 71\ a\ b\}$.

Il y a un petit inconvénient à faire ces regroupements : dans certain cas, si on analyse une chaîne qui n'appartient pas à L(G) l'erreur n'est pas détectée dès son apparition du fait que l'on admet un contexte plus grand.

Nous allons donner maintenant la table en regroupant les ensembles d'états, puis nous analyserons deux chaînes : la première appartenant à L(G), la deuxième n'appartenant pas à L(G). Sur la dernière, nous verrons la détection de l'erreur dans le cas du non regroupement et dans celui du regroupement.

I - 6 - 12 Table réduite.

n	S_n	$S'_n - S_n$	Z_n	p	Z_n^p	ACTION		$\Omega(S_n)$
0	0 0 ⊣	30 ⊣ , 40 ⊣ 10 a b, 20 a b	a			TRANSLATER	a L B	31 ⊣ , 11 a b 41 ⊣ , 21 a b 0 1 ⊣
1	0 1 ⊣			0	⊣	STOP		
2	31 ⊣ , 11 a b			3 1	⊣ a, b	REDUIRE 3 REDUIRE 1		
3	41 ⊣ , 21 a b	50 ⊣ , 60 ⊣ 70 b, 80 b	a, b			TRANSLATER	R N a b	42 ⊣ 22 a b 51 ⊣ , 71 b 61 ⊣ , 81 b
4	42 ⊣			4	⊣	REDUIRE 4		
5	22 a b		b			TRANSLATER	b	23 a b
6	51 ⊣ , 71 a b			5 7	⊣ a, b	REDUIRE 5 REDUIRE 7		
7	61, 81 a b	70 a b, 80 a b	a, b			TRANSLATER	N a b	62 ⊣ , 82 a b 71 a b 81 a b
8	23 a b			2	a, b	REDUIRE 2		
9	62 ⊣ , 82 a b	50 ⊣ , 60 ⊣ 70 b, 80 b	a, b			TRANSLATER	R N a b	63 ⊣ 83 a b 51 ⊣ , 71 b 51 ⊣ , 81 b
10	63 ⊣			6	⊣	REDUIRE 6		
11	83 a b		b			TRANSLATER	b	84 a b
12	84 a b			8	a, b	REDUIRE 8		

I - 6 - 13 Analyse de deux chaînes.

α) Soit la chaîne a b a a. Vérifions que c'est une phrase terminale avec la table non réduite.

$Q_0 \mid a\ b\ a\ a \dashv$	$a \in Z_0$ donc on translate
$Q_0\ a \mid b\ a\ a \dashv$	au vu de a, on trouve $S^1 = S_2 = \{31 \dashv , 11\ a\ b\}$
$Q_0\ a\ S_2 \mid b\ a\ a \dashv$	$b \in Z_2^1$ donc appliquer $a \succ\!\!- L$
$Q_0\ L \mid b\ a\ a \dashv$	avec L recherche dans $\mathcal{R}(Q_0)$ de S^1 $= S_3 = \{41 \dashv , 21\ a\ b\}$
$Q_0\ L\ S_3 \mid b\ a\ a \dashv$	$b \in Z_3$ translation et $S^2 = S_7 = \{61 \dashv , 81\ b\}$
$Q_0\ L\ S_3\ b\ S_7 \mid a\ a \dashv$	$a \in Z_7$ translation et $S^3 = S_{10}$
$Q_0\ L\ S_3\ b\ S_7\ a\ S_{10} \mid a \dashv$	$a \in Z_{10}^7$ donc appliquer $a \succ\!\!- N$
$Q_0\ L\ S_3\ b\ S_7\ N \mid a \dashv$	avec N, recherche dans $\mathcal{R}(S_7)$ de $S^3 = S_9 = \{62 \dashv , 82\ b\}$
$Q_0\ L\ S_3\ b\ S_7\ N\ S_9 \mid a \dashv$	$a \in Z_9$ translation et $S^4 = S_6 = \{51 \dashv , 71\ b\}$
$Q_0\ L\ S_3\ b\ S_7\ N\ S_9\ a\ S_6 \mid \dashv$	$\dashv \in Z_6^5$ donc appliquer $a \succ\!\!- R$
$Q_0\ L\ S_3\ b\ S_7\ N\ S_9\ R \mid \dashv$	avec R recherche dans $\mathcal{R}(S_9)$ de $S^4 = S_{12} = \{63 \dashv\}$
$Q_0\ L\ S_3\ b\ S_7\ N\ S_9\ R\ S_{12} \mid \dashv$	$\dashv \in Z_{12}^6$ donc appliquer $b\ N\ R \succ\!\!- R$
$Q_0\ L\ S_3\ R \mid \dashv$	avec R recherche dans $\mathcal{R}(S_3)$ de $S^2 = S_4$
$Q_0\ L\ S_3\ R\ S_4 \mid \dashv$	$\dashv \in Z_4^4$ donc appliquer $LR \succ\!\!- B$
$Q_0\ B \mid \dashv$	avec B recherche dans $\mathcal{R}(Q_0)$ de $S^1 = S_1$
$Q_0\ B\ S_1 \mid \dashv$	$\dashv \in Z_1^0$ l'analyse est terminée.

a b a a est donc une phrase terminale.

β) Soit la chaîne a b a a b a. Vérifions qu'elle n'appartient pas à L(G).

L'analyse est la même que précedemment jusqu'à :

Q_0 L S_3 b S_7 N S_9 a S_6|b a⊣ b ∈ Z_6^7 donc
appliquer a⟩— N

Q_0 L S_3 b S_7 N S_9 N|b a⊣ avec N recherche dans $\mathcal{R}v(S_9)$ de
$S_4 = S_{13}$

Q_0 L S_3 b S_7 N S_9 N S_{13}|b a⊣ b ∈ Z_{13} donc translation
et $S^5 = S_{15}$

Q_0 L S_3 b S_7 N S_9 N S_{13} b S_{15}|a⊣ pour S_{15} il faudrait trouver un "b", or il y a un "a" donc erreur.

- avec la table réduite même analyse que précedemment jusqu'à

Q_0 L S^1 b S^2 N S^3 N S^4 b S^5|a⊣ avec $S^1 = S_3$,
$S^2 = S_7$, $S^3 = S_9$,
$S^4 = S_{11}$, $S^5 = S_{12}$.

pour S_{12}, a ∈ Z_{12}^8 donc appliquer b N N b⟩— N

Q_0 L S_3 N|a⊣ avec N recherche dans $\mathcal{R}(S_3)$ de
$S^2 = S_5 = \{22$ a b$\}$

Q_0 L S_3 N S_5|a⊣ a ∉ Z_5 donc erreur.

L'erreur a été détectée un peu plus loin que précédemment.

CHAPITRE II

CONSTRUCTION DE LA TABLE

La construction de la table dans le cas des grammaires LR(k) avec k > 1 risquant d'entraîner un très grand nombre de calculs, nous nous sommes intéressés au cas particulier des grammaires LR(1). Ce cas particulier est des plus intéressant puisque d'après le théorème 5 si une grammaire est LR(k) on peut trouver une grammaire équivalente qui soit LR(1).

Les programmes que nous allons présenter ont été écrits en lisp-algol ; ce langage est décrit dans la thèse de J. COHEN [Co].

II - 1 Présentation des données.

Nous avons mis les productions de la grammaire sous forme de liste de la manière suivante :

On écrira $(A_p\ X_{p_1}\ X_{p_2}\ \ldots\ X_{pn_p})$ pour une production de la forme

$$A_p \rightarrow X_{p_1}\ X_{p_2}\ \ldots\ X_{pn_p}.$$

Les productions seront lues à l'aide de la procédure NOMMER, le pointeur étant un élément du tableau L. L[I] sera un pointeur sur la liste représentant la production numéro I.

II - 1 - 1 Conventions d'écriture des productions, codage à la main.

a) Symboles terminaux.

Ils doivent avoir une longueur égale à un (par exemple A, B, C, ...) ou doivent commencer par l'un des symboles suivants : ', ., =, +, -, X, /, :

Les parenthèses servent en lisp-Algol de symboles séparateurs ; ces caractères ne sont donc pas utilisables dans une chaîne de caractères représentant un symbole du vocabulaire. Nous avons donc pris les conventions suivantes : 'L' et 'J' respectivement pour parenthèse gauche et parenthèse droite.

Lorsqu'il y a le choix entre plusieurs représentations du même symbole de base, nous avons pris la représentation la plus courte : 'A' pour 'jusqua', 'N=' pour 'noneg' etc ...

b) Symboles non terminaux.

Afin de gagner de la place en mémoire, il faut que les symboles non terminaux (dont la longueur est supérieure ou égale à deux) aient la plus petite longueur possible. Par exemple, pour la grammaire Algol, les symboles tels que expression-arithmétique, expression-booléenne ont été écrits EA, EB.

Ce travail de préparation étant assez pénible, nous avons pensé qu'il était utile de donner un programme réalisant un certain codage et présentant les productions sous forme de liste.

II - 1 - 2 Programme Backus.

Nous nous sommes servis du programme Backus écrit par F. MARTIN. Ce programme ne respecte pas les conventions précédentes puisqu'il traite une grammaire écrite sous forme normale de Backus et qu'il code par des entiers les symboles terminaux de 1 à NBTERMINAUX et les symboles non terminaux de NBTERMINAUX+1 à NBSYMBOLES.

Il range également les productions l'une après l'autre dans un tableau grammaire.

Pour chaque production il range en grammaire [I] le code de la partie gauche, en grammaire [I + 1] la longueur de la partie droite et de grammaire [I + 2] à grammaire [I + grammaire [I + 1] + 1] le code des éléments de la partie droite.

Le tableau se termine par un zéro pour indiquer la fin des productions.

Exemple. Soit la grammaire

S → a S b S → c

GRAMMAIRE [I]	I		
code de S	1	partie gauche	première production
3	2	nombre d'éléments partie droite	
code de a	3	éléments codés partie droite	
code de S	4		
code de b	5		
code de S	6	partie gauche	deuxième production
1	7	nombre d'éléments partie droite	
code de c	8	élément codé partie droite	
0	9	fin des productions	

Les codes de a, b, c, S seront respectivement 1, 2, 3, 4.

Après constitution de tout le tableau, l'adjonction de la séquence

```
pour I := 2, I + grammaire [I] + 2
tant que grammaire [I - 1] ≠ 0 faire
        PERFORATION(grammaire, I) ;
```

nous a permis de perforer sous forme de liste les productions de la grammaire.

```
procédure PERFORATION(A, I) ; entier tableau A ; entier I ;
    début aiguillage MOD := E2, E3, E4, E5, E6, E7 ;
        SORTIE(7, PERFO, 1, A[I-1], A[I], A[I+1]) ;
        PERFO : allera MOD[A[I]] ;
        E2 : MODELE("(1X, 1H(, 1X, 2(I3, 1X), 1H))") ;
        E3 : MODELE("(1X, 1H(, 1X, 3(I3, 1X), 1H))") ;
        E4 : MODELE("(1X, 1H(, 1X, 4(I3, 1X), 1H))") ;
        E5 : MODELE("(1X, 1H(, 1X, 5(I3, 1X), 1H))") ;
        E6 : MODELE("(1X, 1H(, 1X, 6(I3, 1X), 1H))") ;
        E7 : MODELE("(1X, 1H(, 1X, 7(I3, 1X), 1H))")
    fin PERFORATION ;
```

Nous nous sommes limités à la longueur de la partie droite égale à 6.

Exemple. Pour la grammaire précédente, on aurait le résultat suivant :

```
(4   1   4   2)
(4   3)
```

On peut également sortir les résultats autrement que sur cartes, pour cela, il suffit de changer le numéro de l'élément d'entrée-sortie.

II - 2 Détection des symboles terminaux.

Si on ne se sert pas du programme BACKUS, la grammaire écrite à la main avec les conventions d'écriture indiquées en II - I - I est lue avec le transcodage de caractères défini par l'appel suivant de la procédure CODER :

CODER(" ␣ () A B ... Z 0 1 ... 9 =, '.+ - X / : ")

Le code d'un caractère est son rang dans la chaîne paramètre effectif de coder (soit 1 pour ␣ , 2 pour (, ..., 40 pour = ..., 48 pour :)

La procédure TERMINAL permet alors de détecter les symboles terminaux :

```
booléen procédure TERMINAL(L) ; valeur L ; entier L ;
        TERMINAL := si CADR(L) ≥ 40 ou CDDR(L) = nil
                    alors vrai sinon faux ;
```

Par contre lorsque la grammaire a été préalablement traitée par le programme BACKUS on emploie l'instruction suivante pour déterminer si un symbole est terminal

```
TERMINAL := si L ≤ nbterminaux alors
              vrai sinon faux ;
```

nbterminaux est l'adresse du dernier terminal dans la liste d'atomes des symboles du vocabulaire. Il faut que cette liste commence par les symboles terminaux et que ces symboles ne soient pas mélangés aux autres.
Pour ce faire, juste après l'instruction PREPARER on utilise l'instruction suivante :

```
pour I := 1 pas 1 jusqua nbterminaux faire
        T := entier en liste(I) ;
        nbterminaux := T ;
```

entier en liste est une entier procédure qui met un entier sous forme de liste. Etant donné que les atomes ne sont créés qu'une seule fois et ont ainsi une adresse fixe, on sait que de 1 à NBTERMINAUX on trouve les adresses des entiers mis sous forme de liste. Toutes les adresses des atomes et des listes construites après cette instruction auront une adresse supérieure à nbterminaux à moins qu'il ne s'agisse d'un entier représentant un symbole terminal ou de nil qui est initialisé par PREPARER.

II - 3 Construction de la table.

L'étude théorique a montré le rôle essentiel du contexte pour

- Construire un état : $[p, j ; \alpha]$

- Constuire les ensembles Z_n, Z_n^0, Z_n^1 ... Z_n^s. Ces derniers nous servent pendant la construction de la table pour vérifier si la condition LR(1) est satisfaite et au cours de l'analyse pour décider si l'on doit continuer à lire ou si l'on doit appliquer une réduction.

Pour la recherche du contexte nous aurions pu utiliser l'algorithme de Warshall pour déterminer la matrice SUCCESSEUR [CO]. Cette matrice étant en général très creuse, il est intéressant de la ranger sous forme de liste.

D'autre part, cet algorithme n'étant plus utilisable pour $k > 1$, nous avons donc essayé de programmer cette recherche de la manière suivante :

II - 3 - 1 Construction de liste context et de liste droite.

α) Définitions.

Pour un symbole A appartenant à V_N nous avons construit l'ensemble $H_1(A)$ (définition de $H_K(\varphi)$: I - 6 2). Cette information est rangée dans liste context de la manière suivante : Soient A_1, A_2, ... A_q les éléments de V_N, $H_1(A_1)$, $H_1(A_2)$, ... $H_1(A_q)$ les ensembles correspondants.
Pour chaque i, $H_1(A_i)$ est mis sous forme de liste :

$(a_{i_1}\ a_{i_2}\ \ldots\ a_{ir_i})$ [$a_{i_j} \in H_1(A_i)$, $1 \leq j \leq r_i$]

On pose alors par définition :

Liste context = $((A_1)\ (a_{11}\ a_{12}\ \ldots\ a_{1r_1})\ (A_2)$

$(a_{21}\ a_{22}\ \ldots\ a_{2r_2})\ \ldots\ (A_q)\ (a_{q_1}\ a_{q_2}\ \ldots\ a_{qr_q}))$

A l'aide de liste context, on peut construire la liste de tous les symboles terminaux qui peuvent suivre un élément A_i de V_N.

Soient B_1, B_2, ... B_ℓ des éléments de V_N tels qu'il existe une production de la forme

$$A_{p_j} \rightarrow \ldots A_i\ B_j \ldots \quad 1 \leq j \leq \ell$$

Soient b_1, b_2, ... b_k des éléments de $V_T \cup \{-|\}$ tels qu'il existe une production de la forme

$$A_{p_j} \rightarrow \ldots A_i\ b_j \ldots \quad 1 \leq j \leq k$$

On construit

$$H'_1(A_i) = \bigcup_{m=1}^{\ell} H_1(B_m) \bigcup_{j=1}^{k} \{b_j\}$$

$H'_1(A_i)$ est mis en liste comme $H_1(A_i)$.

On définit liste droite de la même façon que pour liste context :

liste droite = $((A_1)\ (H'_1(A_1)) \ldots (A_q)\ (H'_1\ (A_q)))$

β) Réalisations pratiques.

Dans un premier temps, nous avions construit liste context récursivement en parcourant toutes les productions rangées dans un ordre quelconque. Nous nous sommes aperçus que pour une grammaire assez importante on saturait la mémoire et la construction était fort longue. Afin d'éviter cette saturation et d'accélérer la construction de cette liste, nous avons limité la récursivité en ordonnant les productions.

β - 1 - Toutes les productions ayant même partie gauche V sont rangées les unes à la suite des autres. Désignons par A(V) cet ensemble

β - 2 - On dira que $A(V) < A(W)$ si toutes les productions

$$V \rightarrow X_{i_1}\ X_{i_2} \ldots X_{i_p}$$

se trouvent placées avant toutes les productions

$$W \rightarrow X_{j_1}\ X_{j_2} \ldots X_{j_q}$$

β - 3 - Nous avons partitionné les ensembles A(V) en trois sous-ensembles $A_1(V)$, $A_2(V)$, $A_3(V)$ tels que :

- $A_1(V) < A_2(V) < A_3(V)$
- $A_1(V)$ est l'ensemble des productions telles que X_{i_1} V_T
- $A_2(V)$ est l'ensemble des productions telles que $A(X_{i_1}) < A(V)$
- $A_3(V)$ est l'ensemble des productions telles que $X_{i_1} = V$

Nous interdisons donc les croisements : c'est-à-dire des productions de la forme : $V_1 \rightarrow V_2\ X_1$, $V_2 \rightarrow V_3\ X_2$, ..., $V_{n-1} \rightarrow V_N\ X_{n-1}$, $V_N \rightarrow V_1\ X_n$. X_i étant une chaîne sur V.

Il est très rare de rencontrer de tels croisements. Si l'on en rencontre, ils sont en principe de la forme $V_1 \rightarrow V_2\ X_1$, $V_2 \rightarrow V_1\ X_2$.

En général, on peut toujours les éviter en faisant $V_1 \rightarrow V_1\ X_2\ X_1$.

Ainsi, en parcourant toutes les productions de 0 à s (nombre de productions), quand on trouve une production de la forme A → B X on sait que $H_1(B)$ se trouve déjà dans liste context, ce qui permet de construire celle-ci de façon efficace.

Exemple.

Soit la grammaire G1 :

0	S0 → B1		
1	L1 → A	5	R1 → A
2	L1 → L1 N1 B	6	R1 → B N1 R1
3	B1 → A	7	N1 → B N1 N1 B
4	B1 → L1 R1	8	N1 → A

Liste context = ((N1) (A B) (R1) (A B) (B1) (A) (L1) (A))

Liste droite = ((N1) (A B) (L1) (A B) (B1) ( ))

II - 3 - 2 Construction des états.

Rappelons que dans le cas LR(1) nous avons pris la convention suivante : si $[p, j ; a_1]$, $[p, j ; a_2]$, ... $[p, j ; a_i]$ sont des états, on écrit $[p, j ; a_1 a_2 \dots a_i]$ l'état correspondant. $a_j \in V_T \cup \{\dashv\}$ $1 \leq j \leq i$

Tous les ensembles d'états S_i (par abus de langage nous dirons états) que nous construirons seront mis sous forme de liste et seront repérés par un élément du tableau ETA :

$$ETA[I] = (X_{p_j} (p_1\ j_1\ \alpha_1)\ (p_2\ j_2\ \alpha_2) \dots (p_n\ j_n\ \alpha_n))$$

Nous avons introduit dans cet ensemble X_{p_j} qui caractérise l'état. Rappelons que

$$X_{p_j} = X_{p_\ell\ j_\ell} \qquad 1 \leq \ell \leq n$$

Expliquons brièvement l'algorithme de construction.
Avec l'état initial ETA[0] = (S0 (00⊣)).
On construit n_0 états repérés par ETA[1], ETA[2], ... ETA[n_0].
Puis avec ETA[1] on construit n_1 états repérés par ETA[$n_0 + 1$], ETA[$n_0 + 2$], ... ETA[$n_0 + n_1$]. On continue ainsi de suite en prenant dans l'ordre croissant de l'indice tous les états les uns à la suite des autres jusqu'à épuisement de tous les états. Afin d'éviter de tourner en rond il faudra éliminer les états identiques. D'autre part, afin d'obtenir une table réduite nous ferons certains groupements, cette question fera l'objet du paragraphe suivant.

Pour la construction des états, on emploie une procédure

CONSETA(A, R, C, I2, J1) à 5 paramètres formels : 2 paramètres donnés A et I2, 3 paramètres résultats R, C, J1.

- A est une sous-liste (p j α)
- R est un tableau de pointeurs dans lequel seront stockés les états intermédiaires (q 0 θ)

- C est un tableau de pointeurs sur les états définitifs (X_{p_j} (p_1 j_1 α_1) ... (p_n j_n α_n))
- I est un entier dont la valeur est l'indice du dernier état construit.
- J1 est un entier qui est augmenté de 1 chaque fois que l'on construit un nouvel état.

Pour un état ETA [I] , on appelle la procédure Conseta pour toutes les sous-listes (p_k j_k α_k) de cet état.

Pour une sous-liste (p j α) conseta construit

1°) Les états intermédiaires (q 0 β)

2°) Les états définitifs en groupant ceux qui ont la même tête X_{p_j}

3°) Les ensembles Z_i, Z_i^0, ... Z_i^s.

4°) Cette procédure est récursive, elle est rappelée pour tous les états intermédiaires (q 0 β).

Pour la construction des ensembles Z_i, Z_i^0, ... Z_i^s nous avons défini deux entiers "globaux" Z et ZP. Z pointera sur une liste représentant Z_i, ZP pointera sur une liste représentant les ensembles Z_i^0, Z_i^1 ... Z_i^s.

Pour Z, la liste sera toujours de la forme

(A B C ...) (1)

Pour ZP, la liste contiendra des sous-listes.

Par exemple, ((3 A B) (6 C D X)) (2)

Les numéros indiquent un numéro de production. L'exemple précédent correspondrait aux ensembles Z_i^3 et Z_i^6.

Pour un i donné, à la fin de la construction de ces deux listes, il faut vérifier

- que ces deux listes n'ont pas d'éléments terminaux en commun
- pour ZP, il faut s'assurer que les sous-listes ont une intersection vide.

Ces deux vérifications sont faites à l'aide de deux fonctions : TESTZP(X, Y) et F(X, Y, Z).

X a toujours la forme (2), Y a toujours la forme (1).
TESTZP(X, Y) prend la valeur vrai si X et Y n'ont pas d'éléments terminaux en commun.

Pour vérifier que la grammaire est LR(1), il suffit de prendre un booléen "test" qui sera égal à TESTZP(ZP, Z).
Pendant la construction de ZP, avant d'y ajouter une sous-liste T, on vérifie que T et les sous-listes de TEZP n'ont pas d'éléments terminaux en commun en posant

TEST := TESTZP(ZP, CDR(T)).

Exemple.

Soient les deux fonctions TESTZP et F

```
booléen procédure TESTZP(X, Y) ; valeur X, Y ; entier X, Y ;
      TESTZP := si X = nil alors vrai
                sinon F(X, CDAR(X), Y) ;
booléen procédure F(X, Y, Z) ; valeur X, Y, Z ; entier X, Y, Z ;
     F := si member(CAR(Y), Z) alors faux
          sinon si CDR(Y) ≠ nil alors F(X, CDR(Y), Z)
                sinon TESTZP(CDR(X), Z) ;
```

Prenons test := TESTZP(((3 A B) (4 C)), (E F))

X ≠ nil donc F(((3 A B) (4 C)), (A B), (E F)) CAR((A B)) = A or A ∉ (E F), de plus (B) ≠ nil donc F(((3 A B) (4 C)), (B), (E F)). Même chose B ∉ (E F), par contre CDR(B) = nil donc TESTZP(((4 C)), (E F)). La première liste n'étant pas égale à nil, on appelle

F(((4 C)), (C), (E F))

C ∉ (E F) donc TESTZP(( ), (E F)) cette fois-ci la première liste est égale à nil et TESTZP prend la valeur vrai.

Pour garder ces deux listes on a défini deux tableaux globaux TEZ et TEZP, tels que TEZ [I] = Z et TEZP [I] = ZP.

II - 3 - 3 Groupement d'états.

Soient A et B deux états

$A = (X_{p_j} \ (p_1\ j_1\ \alpha_1)\ (p_2\ j_2\ \alpha_2) \ \ldots \ (p_n\ j_n\ \alpha_n))$

$B = (X_{q_k} \ (q_1\ k_1\ \beta_1)\ (q_2\ k_2\ \beta_2) \ \ldots \ (q_m\ k_m\ \beta_m))$

Définitions.

1) Nous dirons que $A \leq B$ si

- $X_{p_j} = X_{q_k}$
- $n \leq m$
- pour toutes les sous-listes $(p_i\ j_i\ \alpha_i)$ de A, il existe un j tel que $(q_j\ k_j\ \beta_j)$ soit une sous-liste de B avec $p_i = q_j$, $j_i = k_j$ et $\alpha_i \subseteq \beta_j$

2) Nous dirons que $A \supset B$ si

- $X_{p_j} = X_{q_k}$
- $n > m$
- pour toutes les sous-listes $(q_j\ k_j\ \beta_j)$ de B, il existe un i tel que $(p_i\ j_i\ \alpha_i)$ soit une sous-liste de A avec $p_i = q_j$, $j_i = k_j$ et $\beta_j \subseteq \alpha_i$

3) Nous dirons que $A \sim B$ si

- $X_{p_j} = X_{q_k}$
- si $n \leq m$, pour toutes les sous-listes $(p_i\ j_i\ \alpha_i)$ de A il existe un j tel que $(q_j\ k_j\ \beta_j)$ soit une sous-liste de B avec $p_i = q_j$, $k_j = j_i$.
- si $n > m$, pour toutes les sous-listes $(q_j\ k_j\ \beta_j)$ de B il existe un i tel que $(p_i\ j_i\ \alpha_i)$ soit une sous-liste de A avec $p_i = q_j$, $k_j = j_i$.

Pour un I donné, après avoir construit les états ETA[I2 + 1] ... ETA[I2 + J1] qui appartiennent à $\mathcal{R}$(ETA[I]), on examine tous les états numérotés de 1 à I2 pour déterminer s'il existe un J ($1 < J \leq I2$) tel que B = ETA[J] et un k ($I2 + 1 \leq k \leq I2 + J1$) tel que A = ETA[k] avec $A \subseteq B$ ou $A \supset B$ ou $A \sim B$.

I Il existe un B tel que $A \subseteq B$ ou $A \supset B$ un groupement est possible et sera fait

1°) $A \subseteq B$

Il est inutile de garder A. En effet si on recommençait tous les calculs avec A : création d'états, ensembles Z, ZP. Tous ces résultats ont déjà été ou seront obtenus (suivant que $I \geq J$ ou $I < J$) avec B.

On supprime A. C'est-à-dire que l'on fait un retassement en posant ETA[L] := ETA[L + 1] ($k \leq L < j_1$).
A la fin du retassement j_1 prend la nouvelle valeur $j_1 - 1$.

Pour l'analyse, il faut garder la trace de A. On a donc défini un tableau REP qui pour chaque I pointera sur une liste contenant les numéros des états correspondant aux états supprimés. Dans le cas ci-dessus, on mettra comme numéro, le numéro de B c'est-à-dire J.

2°) $A \supset B$

- B n'a pas encore été utilisé pour la construction d'autres états ($J > I$).

On remplace B par A, on supprime A tout en sauvegardant le numéro de B comme précédemment.

- B a déjà été utilisé ($J \leq I$), on ne fera rien car il faudrait reconstruire tous les états que B a construit et ceci récursivement.

II Il n'existe pas de B tel que $A \subseteq B$ ou $A \supset B$, mais il existe B tel que $A \sim B$ un groupement est possible et sera fait dans certaines conditions.

1°) A $\sim$ B et B n'a pas été utilisé (J > I)

On remplace B par C union des listes A et B. La liste C est construite de la manière suivante :

$C = (X_{r_\ell} (r_1\ \ell_1\ \gamma_1) \ldots (r_u\ \ell_u\ \gamma_u))$

- $X_{r_\ell} = X_{p_j} = X_{q_k}$
- supposons que $m \geq n$ (le cas $n > m$ est symétrique)
- $s = m$. C aura donc autant de sous-listes que B : $r_i = q_i$, $\ell_i = k_i$ $(1 \leq i \leq m)$
- γ_i sera égale à
- β_i s'il n'existe pas de sous-liste $(p_k\ j_k\ \alpha_k)$ de A telle que $p_k = q_i$ et $j_k = k_i$
- l'union de α_k et de β_i s'il existe une telle sous-liste.

2°) A $\sim$ B, B a déjà été utilisé (J $\leq$ I).

On ne fera un groupement que si A et B ne possèdent qu'une seule sous-liste (c'est-à-dire que A et B correspondent au même point de la même production mais qu'ils n'ont pas le même contexte) et dans les deux cas particuliers suivants :

1er cas.

B a bien été utilisé pour la construction d'autres états mais en fait il n'en a pas créé, car cet état correspondait à une fin de production. TEZ [J] pointe donc sur une liste vide et TEZP [J] pointe sur une liste ne contenant qu'une seule sous-liste. On remplace B par C union de A et de B et on ajoute à TEZP [J] le contexte de A qui ne se trouvait pas dans B.

2ème cas.

On impose 3 conditions

- B n'a créé qu'un seul état D
- D ne possède qu'une seule sous-liste (c'est-à-dire que dans la production on est arrivé à une métavariable suivie d'un terminal)
- D n'a pas encore été utilisé pour la construction d'autres états.

On ajoute à B et D le contexte de A qui ne se trouvait pas dans B et D.

Bien entendu, pour chaque groupement, on élimine A tout en sauvegardant le numéro de B (I - 1°)).

Nous n'avons pas effectué tous les groupements possibles. Nous nous sommes bornés à quelques cas particuliers assez fréquents afin d'éviter une perte de place et de temps assez importantes.

Lorsque pour un I donné, tous les groupements que nous avons indiqués ont été effectués, J1 a pris une nouvelle valeur égale au nombre d'états qui n'ont pas été supprimés. I2 prend la nouvelle valeur I2+J1, cette valeur est stockée dans le tableau RTABLE (RTABLE[I]:=I2).

II - 4 Sortie des résultats.

Lorsque tous les états ont été créés, on fait sortir les résultats sur cartes (ou sur bande). Pour les états on ne garde que l'élément X_{p_j} qui le caractérise, tout le reste est inutile pour l'analyse. Les résultats sont ensuite utilisés par l'analyseur. Cette utilisation fera l'objet des chapîtres suivants.
On trouvera sur les pages suivantes un listage de ce programme.

II - 5 Programme.

```
DEBUT
    COMMENTAIRE DECLARATION ET AFFECTATION DE NPFER(NB DE PAGES EN M
    R )ET NMAX (NB DE PAGES TOTAL) NB DE DOUBLETS MAX=230*NMAX , PLACE
    PRISE PAR LA ZONE EN LISTE DANS LA MR :2*NPFER*230 ;
    ENTIER NMAX,NPFER;
    NMAX:=EDONNEE;
    NPFER:=EDONNEE;

    DEBUT
        COMMENTAIRE :BLOC UN :DECLARATION DES VARIABLES GLOBALES ET
        DES PROCEDURES ALGOL-LISP ;
        ENTIER MAUX,BCAR,INDICE,ADVAL,RHS,AIPD;
        COMMENTAIRE NEPASDEPLACERLACARTEPRECEDENTE;
        ENTIER TABLEAU TCAR,TCDR[0:232*NPFER],TE[1:11,1:4];

        ENTIER NIL,IPD,IPDMAX,LISTEDATOMES,BOITE;
        ENTIER N,I,EMPILE,EMPILE2,LISTECONTEXT,COMPT,I2,J1,UN,X,K4,K
        I1,Y,Z,ZP,LISTEDROITE,B,I5,NBTERMINAUX,LONGTAB,TQ,H,H1,K1,T,
        T1,T2,A,J,PR,TEMPS1,ZERO,NO;
        BOOLEEN BOOL1,BOOL2,BOOL,E,TEST,PERFO;
        LIRE(N);
        LIRE(LONGTAB);
        DEBUT
            ENTIER TABLEAU L[0:N],ETA,REP,TEZ,TEZP[0:LONGTAB],
            RTABLE[-1:LONGTAB],R[1:100];

            COMMENTAIRE :LES N REGLES DE GRAMMAIRE SONT RANGEES SOUS
            FORME DE LISTE DANS LE TABLEAU L . ETA[I] REPRESENTE
            L'ETAT NUMERO I . REP[I] , TEZ[I] , TEZP[I]
            REPRESENTENT RESPECTIVEMENT POUR L'ETAT NUMERO I , LA
            LISTE DES ETATS CONSTRUITS , L'ENSEMBLE Z , L'ENSEMBLE ZP
            . RTABLE[I] :NOMBRE D'ETATS CONSTRUITS AVEC LES ETATS
            NUMERO 0,1,.....I . LE NOMBRE TOTAL D'ETATS ETANT EGAL A
            I2 ;


            PROCEDURE ECRIRE BARATIN(CANAL,CHAINE) ;
            VALEUR CANAL;
            ENTIER CANAL;
            CHAINE CHAINE;
            DEBUT
                ALALIGNE(1);
                ECRIRECHAINE(CANAL,CHAINE);
            FIN ;


            BOOLEEN PROCEDURE EQUAL(X,Y);
            VALEUR X,Y;
            ENTIER X,Y;

            COMMENTAIRE :EQUAL PREND LA VALEUR VRAI SI LES DEUX
            LISTES X ET Y SONT IDENTIQUES , FAUX DANS LE CAS CONTRAIRE
            ;

            EQUAL:= SI X=Y ALORS VRAI
```

```
SINON
SI ATOM(X) ET ATOM(Y) ALORS EQ(X,Y)
SINON
SI NON (ATOM(X) IDENT ATOM(Y)) ALORS FAUX
SINON
SI EQUAL(CAR(X),CAR(Y)) ALORS EQUAL(CDR(X),CDR(Y))
SINON FAUX ;


BOOLEEN PROCEDURE MEMBER(X,Y) ;
VALEUR X,Y;
ENTIER X,Y;

COMMENTAIRE :MEMBER PREND LA VALEUR VRAI SI X APPARTIENT
A LA LISTE Y ;

MEMBER:= SI NON ATOM(X) ET ATOM(Y) ALORS FAUX
SINON
SI ATOM(X) ET ATOM(Y) ALORS EQ(X,Y)
SINON
SI EQUAL(X,CAR(Y)) ALORS VRAI
SINON MEMBER(X,CDR(Y));


ENTIER PROCEDURE UNION(M,N);
VALEUR M,N;
ENTIER M,N;

COMMENTAIRE :UNION DES DEUX LISTES M ET N ;

UNION:= SI ATOM(M) ALORS (
SI MEMBER(M,N) ALORS N
SINON CONS(M,N))
SINON
SI MEMBER(CAR(M),N) ALORS UNION(CDR(M),N)
SINON CONS(CAR(M),UNION(CDR(M),N));


ENTIER PROCEDURE IEME ( L,I ) ;
VALEUR L,I;
ENTIER L,I;

COMMENTAIRE :DONNE LE IEME ELEMENT DE LA LISTE L ;

IEME:= SI I=1 ALORS CAR(L)
SINON IEME(CDR(L),I-1);


ENTIER PROCEDURE NUMERIQUE(X) ;
VALEUR X;
ENTIER X;

COMMENTAIRE :VALEUR NUMERIQUE DE L'ENTIER MIS SOUS FORME
DE LISTE ;
```

```
DEBUT
     ENTIER M1;
     X:=CDR(X);
     M1:=CAR(X)-30;
     POUR X:=CDR(X) TANTQUE  X ≠ NIL FAIRE
     M1:=M1*10     (X)-30;
     NUMERIQUE:=M1;
FIN PROCEDURE NUMERIQUE ;


ENTIER PROCEDURE ENTIER EN LISTE(T);
VALEUR T;
ENTIER T;

COMMENTAIRE :L'ENTIER T EST MIS SOUS FORME DE LISTE ;

DEBUT
     ENTIER I,J;
     J:= SI T=0 ALORS CONS(30,NIL)
     SINON NIL;
     POUR I:=T-(T ./. 10)*10 TANTQUE  T ≠ 0 FAIRE
     DEBUT
          J:=CONS(I+30,J);
          T:=T ./. 10
     FIN ;
     J:=CONS(-1,J);
     ENTIERENLISTE:=COMPARERATOME(J)
FIN ENTIER EN LISTE ;


BOOLEEN PROCEDURE TERMINAL(L) ;
VALEUR L;
ENTIER L;

COMMENTAIRE :TERMINAL PREND LA VALEUR VRAI SI L
REPRESENTE UN ELEMENT DU VOCABULAIRE TERMINAL , FAUX DANS
LA CAS CONTRAIRE ;

TERMINAL:= SI CAR(CDR(L)) ≥ 40 OU CDDR(L)=NIL ALORS VRAI
SINON FAUX ;


ENTIER PROCEDURE NOMBRE(L) ;
VALEUR L;
ENTIER L;

COMMENTAIRE :NOMBRE D'ELEMENTS DE LA LISTE L ;

DEBUT
     ENTIER I,X;
     X:=L;
     I:=1;
     POUR X:=CDR(X) TANTQUE  X ≠ NIL FAIRE
     I:=I+1;
     NOMBRE:=I;
```

```
FIN NOMBRE ;


ENTIER PROCEDURE CHERCHREGLE(K) ;
VALEUR K;
ENTIER K;

COMMENTAIRE :DONNE LE NUMERO DE LA PREMIERE REGLE DE
GRAMMAIRE AYANT UNE PARTIE GAUCHE EGALE A K ;

DEBUT
    ENTIER I;
    I:=0;
    E:= VRAI ;
    POUR I:=I+1 TANTQUE  I ≤ N ET E FAIRE
    SI CAR(L[I])=K ALORS
    DEBUT
        CHERCHREGLE:=I;
        E:= FAUX ;
    FIN
FIN CHERCHREGLE ;


COMMENTAIRE :LES DEUX PROCEDURES SUIVANTES VERIFIENT QUE
CHAQUE SOUS-LISTE DE X N'A PAS D'ELEMENTS EN COMMUN AVEC
LA LISTE Y ;

BOOLEEN PROCEDURE F(X,Y,Z) ;
VALEUR X,Y,Z;
ENTIER X,Y,Z;
F:= SI MEMBER(CAR(Y),Z) ALORS FAUX
SINON
SI CDR(Y) ≠ NIL ALORS F(X,CDR(Y),Z)
SINON TESTZP(CDR(X),Z);

BOOLEEN PROCEDURE TESTZP(X,Y) ;
VALEUR X,Y;
ENTIER X,Y;
TESTZP:= SI X=NIL ALORS VRAI
SINON F(X,CDAR(X),Y);


ENTIER PROCEDURE TERMINAUX(J,K) ;
VALEUR J,K;
ENTIER J,K;

COMMENTAIRE :POUR UN SYMBOLE K DONNE ET UN NUMERO DE
REGLE J , ON RECHERCHE TOUS LES SYMBOLES TERMINAUX DERIVES
DE K , DE PLUS EMPILE REPRESENTE LA LISTE DE TOUS LES
SYMBOLES NON TERMINAUX DERIVES DE K ;

DEBUT
    ENTIER M,I;
    SI J > N ALORS TERMINAUX:=NIL
    SINON
```

```
    SI TERMINAL(CADR(L[J])) ALORS
    DEBUT
        COMMENTAIRE :C'EST UN SYMBOLE TERMINAL ;
        TERMINAUX:= SI J=N ALORS UNION(CADR(L[J]),
        TERMINAUX(J+1,K))
        SINON
        SI CAR(L[J+1])=K ALORS UNION(CADR(L[J]),
        TERMINAUX(J+1,K))
        SINON UNION(CADR(L[J]),TERMINAUX(N+1,K))
    FIN
    SINON
    SI CAR(L[J])=CADR(L[J]) ALORS TERMINAUX:=
    TERMINAUX(
    SI CAR(L[J+1])=K ALORS J+1
    SINON N+1,CADR(L[J]))
    SINON
    DEBUT
        COMMENTAIRE :CE N'ETAIT PAS UN SYMBOLE TERMINAL
        DONC CONSTRUCTION DE LA LISTE EMPILE ;
        TERMINAUX:=NIL;
        M:=J;
        I:=CADR(L[J]);
        EMPILE:=UNION(I,EMPILE);
        POUR M:=M+1 TANTQUE  CAR(L[J])=CAR(L[M]) ET M
        < N FAIRE
        SI CAR(L[M]) ≠ CADR(L[M]) ALORS EMPILE:=
        UNION(CADR(L[M]),EMPILE);
    FIN ;
FIN TERMINAUX ;


ENTIER PROCEDURE CONTEXT(K,L) ;
VALEUR K,L;
ENTIER K,L;

COMMENTAIRE :CONSTRUCTION DE LISTECONTEXT OU RECHERCHE
DANS LISTECONTEXT DU CONTEXTE D'UN SYMBOLE K DONNE .
LORSQUE CETTE PROCEDURE EST APPELEE LORS DE LA
CONSTRUCTION DES ETATS L REPRESENTE UN ETAT ;

DEBUT
    ENTIER I,J,T1,T2;
    EMPILE:=NIL;
    EMPILE2:=CONS(K,NIL);
    SI NULL(K) ALORS CONTEXT:=CDDR(L)
    SINON
    SI TERMINAL(K) ALORS CONTEXT:=CONS(K,NIL)
    SINON
    DEBUT
        COMMENTAIRE :RECHERCHE DANS LISTECONTEXT DU
        CONTEXTE DE K ;
        E:= VRAI ;
        SI LISTECONTEXT ≠ NIL ALORS
        DEBUT
            POUR J:=LISTECONTEXT,CDDR(J) TANTQUE  J ≠ NIL
```

```
                    ET E FAIRE
                    SI K=CAAR(J) ALORS
                    DEBUT
                        E:= FAUX ;
                        CONTEXT:=CADR(J)
                    FIN ;
                FIN ;
                SI E ALORS
                DEBUT
                    T1:=I5+1;
                    COMMENTAIRE :ON NE L'A PAS TROUVE DONC
                    PREPARATION DE LA CONSTRUCTION DE LISTECONTEXT
                    ;
                    T1:=TERMINAUX(T1,K);
'ASFINI:            SI EMPILE ≠ NIL ALORS
                    DEBUT
                        I:=CAR(EMPILE);
                        EMPILE:=CDR(EMPILE);
                        E:= VRAI ;
                        POUR J:=LISTECONTEXT,CDDR(J) TANTQUE  J ≠
                        NIL ET E FAIRE
                        SI I=CAAR(J) ALORS
                        DEBUT
                            E:= FAUX ;
                             T2:=CADR(J);
                        FIN ;
                        SI E ALORS
                        DEBUT
                            T2:=CHERCHREGLE(I);
                            T2:=TERMINAUX(T2,I);
                        FIN ;
                        T1:= SI NOMBRE(T1) ≥ NOMBRE(T2) ALORS
                        UNION(T2,T1)
                        SINON UNION(T1,T2);
                        ALLERA  PASFINI
                    FIN ;
                    COMMENTAIRE :CONSTRUCTION DE LISTECONTEXT ;
                    LISTECONTEXT:=CONS(EMPILE? CO  'T1
                    LISTECONTEXT));
                    CONTEXT:=T1;
                FIN
            FIN
        FIN CONTEXT ;


        ENTIER PROCEDURE SUITE(K);
        VALEUR K;
        ENTIER K;

        COMMENTAIRE :RECHERCHE DANS LISTEDROITE DES SYMBOLES QUI
        PEUVENT SUIVRE LE SYMBOLE K ;

        DEBUT
            ENTIER J;
            E:= VRAI ;
```

```
        POUR J:=LISTEDROITE,CDDR(J) TANTQUE  J ≠ NIL ET E
        FAIRE
        SI K=CAAR(J) ALORS
        DEBUT
            E:= FAUX ;
            SUITE:=CADR(J);
        FIN ;
        SI E ALORS SUITE:=NIL;
FIN SUITE ;


PROCEDURE CONSTEST(A,Z,ZP,B,I,J) ;
VALEUR A,I,J;
ENTIER A,I,J,Z,ZP;
BOOLEEN B;

COMMENTAIRE :CONSTRUCTION DE ZP OU DE I SUIVANT QUE B EST
VRAI OU NON ;

DEBUT
        ENTIER T;
        SI B ALORS
        DEBUT
            T:=CONS(CAR(A),CDDR(A));
            COMMENTAIRE :VERIFICATION QUE LA CONVELLE
            SOUS-LISTE DE ZP EST A INTERSECTION VIDE AVEC LES
            AUTRES SOUS-LISTES DE ZP;
            SI TESTZP(ZP,CDR(T)) ALORS ZP:=CONS(T,ZP)
            SINON TEST:= FAUX
        FIN
        SINON
        SI Z=NIL ALORS Z:=CONTEXT(IEME(LI,J+2),A)
        SINON
        POUR Y:=CONTEXT(IEME(LI,J+2),A),CDR(Y) TANTQUE  Y
        ≠ NIL FAIRE
        SI NON MEMBER(CAR(Y),Z) ALORS Z:=CONS(CAR(Y),Z);
FIN CONSTEST ;


BOOLEEN PROCEDURE COMPARAISON(A,B,BOOL1,BOOL2) ;
VALEUR A,B;
ENTIER A,B;
BOOLEEN BOOL1,BOOL2;

COMMENTAIRE :COMPARAISON DES DEUX LISTES A ET B. B SURCO
A C'EST A DIRE QUE B POSSEDE PLUS OU AUTANT DE SOUS-LISTE
QUE A. SI POUR UNE SOUS-LISTE DONNEE DE A ON TROUVE DANS B
UNE SOUS-LISTE AYANT LE MEME NUMERO DE CELLE EN TETE
COMPARAISON PREND LA VALEUR VRAI. BOOL1 ET BOOL2 SONT DEUX
INDICATEURS POUR SAVOIR SI TOUS LES SYMBOLES D'UNE
SOUS-LISTE SE RETROUVENT DANS L'AUTRE SOUS-LISTE. EN MEME
TEMPS R REPRESENTE L'UNION DE CES DEUX SOUS-LISTES. CETTE
PROCEDURE EST APPELLEE RECURSIVEMENT POUR TOUTES LES
SOUS-LISTES DE A ;
```

```
DEBUT
    ENTIER I,J,K,L,M,T;
    BOOLEEN E;
    SI A=NIL ALORS COMPARAISON:= VRAI
    SINON
    DEBUT
        T:=CAR(A);
        I1:=I1+1;
        I:=CAR(T);
        J:=CADR(T);
        K:=CDDR(T);
        E:= VRAI ;
        POUR L:=B,CDR(L) TANTQUE  L ≠ NIL ET E FAIRE
        DEBUT
            M:=CAR(L);
            SI I=CAR(M) ET J=CADR(M) ALORS
            DEBUT
                M:=CDDR(M);
                SI NOMBRE(M) ≥ NOMBRE(K) ALORS
                DEBUT
                    I:=UNION(K,M);
                    BOOL2:=EQUAL(M,I);
                FIN
                SINON
                DEBUT
                    I:=UNION(M,K);
                    BOOL1:=EQUAL(K,I);
                FIN ;
                E:= FAUX ;
                R[I1]:=I;
            FIN ;
        FIN ;
        COMPARAISON:= SI E ALORS FAUX
        SINON COMPARAISON(CDR(A),B,BOOL1,BOOL2);
    FIN ;
FIN COMPARAISON ;


PROCEDURE CONSETA(A,R,C,I2,J1);
VALEUR A,I2;
ENTIER A,I2,J1;
ENTIER TABLEAU R,C;

COMMENTAIRE :CONSTRUCTION DES ETATS QUE L'ON PEUT
CONSTRUIRE A PARTIR DE LA SOUS-LISTE A , J1 INDIQUE LE
NOMBRE D'ETATS QUE L'ON VA CONSTRUIRE , I2 INDIQUE LE
NUMERO DU DERNIER ETAT QUE L'ON A CONSTRUIT AVANT
D'ACTIVER LA PROCEDURE , ON RANGE LES NOUVEAUX ETATS DANS
LE TABLEAU C , LE TABLEAU R CONTENANT LES ETATS
INTERMEDIAIRES DE S' - S ;

DEBUT
    ENTIER P,I,J,M,TP,MP;
    COMMENTAIRE :NUMERO DE REGLE ;
    I:=NUMERIQUE(CAR(A));
```

```
                COMMENTAIRE :POSITION DANS LA REGLE ;
                J:=NUMERIQUE(CADR(A));
ETATFINAL:      SI I=0 ET J=1 ALORS ALLERA  FINIA;
                SI J+1=NOMBRE(L[I]) ALORS
                DEBUT
                    COMMENTAIRE :CONSTRUCTION DE ZP ;
                    CONSTEST(A,Z,ZP, VRAI ,I,J);
                    ALLERA  FINIA;
                FIN ;
                SI K4=1 ALORS
                DEBUT
                    COMMENTAIRE :CONSTRUCTION DU PREMIER ETAT ;
                    K:=K+1;
                    K4:=0;
                    TP:=CDDR(A);
                    C[I2+K]:=CONS(CAR(A),CONS(ENTIERENLISTE(J+1),
                    TP));
                    C[I2+K]:=CONS(IEME(L[I],J+2),CONS(C[I2+K],
                    NIL));
                    P:=0;
                    E:= VRAI ;
                    COMMENTAIRE :REGROUPEMENT D'ETATS ;
                    POUR P:=P+1 TANTQUE  E ET P ≤ K-1 FAIRE
                    DEBUT
                        M:=CAR(C[I2+K]);
                        SI M=CAR(C[I2+P]) ALORS
                        DEBUT
                            C[P+I2]:=CONS(CAR(C[P+I2]),
                            CONS(CADR(C[I2+K]),CDR(C[P+I2])));
                            K:=K-1;
                            E:= FAUX ;
                        FIN ;
                    FIN ;
                    COMMENTAIRE :CONSTRUCTION DE
                    CONSTEST(A,Z,ZP, FAUX ,I,J);
                FIN ;
                M:=IEME(L[I],J+2);
                COMMENTAIRE :SYMBOLE TERMINAL DONC PAS DE
                CONSTRUCTION ;
                SI TERMINAL(M) ALORS ALLERA  FINIA;
                TQ:=SUITE(M);
                P:=CHERCHREGLE(M)-1;
                SI NOMBRE(L[I])=J+2 ALORS
                DEBUT
                    J:=CDDR(A);
                    TQ:= SI NOMBRE(J) ≥ NOMBRE(TQ) ALORS UNION(TQ,J)
                    SINON UNION(J,TQ);
                FIN ;
                MP:=CONS(UN,TQ);
                POUR P:=P+1 TANTQUE  CAR(L[P])=M FAIRE
                DEBUT
                    K1:=K1+1;
                    K:=K+1;
                    TP:=ENTIERENLISTE(P);
                    POUR I:=2 PAS 1 JUSQUA K1-1 FAIRE
```

```
                    SI CAR(R[I])=TP ALORS
                    DEBUT
                        COMMENTAIRE :ETAT INTERMEDIAIRE DEJA
                        CONSTRUIT ;
                        K:=K-1;
                        K1:=K1-1;
                        ALLERA  END;
                    FIN ;
                    E:= VRAI ;
                    COMMENTAIRE :CONSTRUCTION D'UN ETAT INTERMEDIAIRE
                    ;
                    R[K1]:=CONS(TP,CONS(ZERO,TQ));
                    COMMENTAIRE :CONSTRUCTION DE Z ;
                    CONSTEST(R[K1],Z,ZP, FAUX ,P,0);
                    B:=CONS(TP,MP);
                    H:=IEME(L[P],2);
                    C[I2+K]:=CONS(H,NIL);
                    E:= VRAI ;
                    I:=0;
ITER:               I:=I+1;
                    SI H=CAR(C[I+I2]) ALORS
                    DEBUT
                        COMMENTAIRE :CONSTRUCTION D'UN NOUVEL ETAT ET
                        REGROUPEMENT ;
                        C[I2+I]:=CONS(H,CONS(B,CDR(C[I2+I])));
                        C[I2+K]:=NIL;
                        K1:=K1-1;
                        K:=K-1;
                        E:= FAUX ;
                    FIN
                    SINON
                    SI I < K-1 ET E ALORS ALLERA  ITER;
                    COMMENTAIRE :CONSTRUCTION D'UN NOUVEL ETAT SI NON
                    REGROUPEMENT ;
                    SI E ALORS C[I2+K]:=CONS(CAR(C[I2+K]),CONS(B,
                    CDR(C[I2+K])));
END:                SI P=N ALORS ALLERA  FINIA;
                FIN ;
FINIA:          I1:=I1+1;
                J1:=K;
                SI K1=1 ALORS ALLERA  FINICONSETA;
                COMMENTAIRE :CONSTRUCTION DE NOUVEAUX ETATS A PARTIR
                DES ETATS INTERMEDIAIRES ;
                SI I1 ≤ K1 ALORS CONSETA(R[I1],R,C,I2,J1);
FINICONSETA: FIN CONSETA ;


MAINPROGRAM:

            PREPARER;


            K:=0;
            TEMPS1:=TEMPS;
            POUR I:=0 PAS 1 JUSQUA N FAIRE
```

```
                DEBUT
LECTURE:        NOMMER(2, " " ,L[I]);
                I1:=NOMBRE(L[I]);
                SI I1 ≥ 3 ALORS
                DEBUT
                    K:=K+1;
                    I5:=K;
                    ETA[K]:=CDR(L[I]);
                    COMMENTAIRE :DETERMINATION POUR UN SYMBOLE DE
                    TOUS LES SYMBOLES QUI PEUVENT LE SUIVRE DANS LES
                    REGLES DE GRAMMAIRE ;
ITERATION:          SI NOMBRE(ETA[K]) ≥ 3 ALORS
                    DEBUT
                        ETA[K+1]:=CDR(ETA[K]);
                        ETA[K]:=CONS(CAR(ETA[K]),CAR(ETA[K+1]));
                        K:=K+1;
                        ALLERA  ITERATION;
                    FIN ;
                    POUR I1:=I5,I1+1 TANTQUE  I1 ≤ K FAIRE
                    POUR J1:=1 PAS 1 JUSQUA I1-1 FAIRE
                    SI CAR(ETA[J1])=CAR(ETA[I1]) ALORS
                    DEBUT
                        ETA[J1]:=CONS(CAR(ETA[J1]),
                        UNION(CDR(ETA[I1]),CDR(ETA[J1])));
                        POUR Z:=I1 PAS 1 JUSQUA K-1 FAIRE
                        ETA[Z]:=ETA[Z+1];
                        K:=K-1;
                        I1:=I1-1;
                    FIN ;
                FIN ;
            FIN ;
            NOMMER(0, " ( SO ( 0 0 // ) ) " ,ETA[0]);
            ZERO:=ENTIERENLISTE(0);
            UN:=ENTIERENLISTE(1);
            LISTEDROITE:=NIL;
            LISTECONTEXT:=NIL;
            POUR I5:=0 PAS 1 JUSQUA N-1 FAIRE
            DEBUT
                SI CAR(L[I5]) ≠ CAR(L[I5+1]) ALORS
                COMMENTAIRE :CONSTRUCTION DE LISTECONTEXT ;
                PR:=CONTEXT(CAR(L[I5+1]),ETA[0]);
            FIN ;
            POUR I1:=1 PAS 1 JUSQUA K FAIRE
            SI NON TERMINAL(CAR(ETA[I1])) ALORS
            DEBUT
                I5:=CONS(CAR(ETA[I1]),NIL);
                ETA[I1]:=CDR(ETA[I1]);
                J1:=CONTEXT(CAR(ETA[I1]),NIL);
TOURNE:         SI CDR(ETA[I1]) ≠ NIL ALORS
                DEBUT
                    ETA[I1]:=CDR(ETA[I1]);
                    Z:=CONTEXT(CAR(ETA[I1]),NIL);
                    J1:= SI NOMBRE(Z) ≥ NOMBRE(J1) ALORS UNION(J1,Z)
                    SINON UNION(Z,J1);
                    ALLERA  TOURNE;
```

```
                FIN ;
                COMMENTAIRE :CONSTRUCTION DE LISTEDROITE ;
                LISTEDROITE:=CONS(I5,CONS(J1,LISTEDROITE));
            FIN ;
            ALALIGNE(4);
            ECRIREBARATIN(1,
            "                                                    " );
            ECRIRECHAINE(1, " ************************** " );
            ECRIREBARATIN(1,
            "                                                    " );
            ECRIRECHAINE(1, " *CONSTRUCTION DE LA TABLE* " );
            ECRIREBARATIN(1,
            "                                                    " );
            ECRIRECHAINE(1, " ************************** " );
            ALALIGNE(5);
            I2:=0;
            PERFO:= VRAI ;
            POUR COMPT:=0 PAS 1 JUSQUA I2 FAIRE
            DEBUT
INITIALISATION: X:=ETA[COMPT];
                Z:=NIL;
                ZP:=NIL;
                K:=0;
                K1:=1;
                REP[COMPT]:=NIL;
                I1:=2;
                TEST:= VRAI ;
                POUR X:=CDR(X) TANTQUE  X ≠ NIL FAIRE
                DEBUT
                    K4:=1;
                    I1:=I1-1;
                    COMMENTAIRE :CONSTRUCTION D'ETATS ;
                    CONSETA(CAR(X),R,ETA,I2,J1);
                FIN ;
                TEST:=TEST ET TESTZP(ZP,Z);
                ECRIRELISTE(1,ETA[COMPT]);
                ECRIREBARATIN(1, " CET ETAT QUI EST LE " );
                ECRIREATOME(1,CDR(ENTIERENLISTE(COMPT)));
                I:=I2;
                SI J1=0 ALORS
                DEBUT
                    ECRIRECHAINE(1,
                    " IEME N A PAS PERMIS DE CONSTRUIRE D ETATS " );
                    ECRIRECHAINE(1,
                    " CAR ON ETAIT ARRIVE EN FIN DE REGLE " );
                    ALLERA  ECRITURE;
                FIN ;
DANKE:          I:=I+1;
                COMMENTAIRE :COMPARAISON DES ETATS ;
                H:=CAR(ETA[I]);
                BOOL:= VRAI ;
                J:=1;
                COMMENTAIRE :RECHERCHE DU MEME SYMBOLE DE TETE ;
                POUR J:=J+1 TANTQUE  J ≤ I2 ET BOOL FAIRE
                SI H=CAR(ETA[J]) ALORS
                DEBUT
                    T:=CDR(ETA[J]);
                    Q:=CDR(ETA[I]);
```

```
T1:=NOMBRE(T);
T2:=NOMBRE(Q);
COMMENTAIRE :RECHERCHE DU PLUS GRAND DES DEUX
ETATS ;
SI T1 ≥ T2 ALORS
DEBUT
    A:=ETA[I];
    B:=ETA[J];
FIN
SINON
DEBUT
    SI J < COMPT ALORS ALLERA  TROLONG;
    A:=ETA[J];
    B:=ETA[I];
FIN ;
X:=A;
I1:=0;
T1:=NIL;
BOOL2:=BOOL1:= FAUX ;
E:= FAUX ;
SI COMPARAISON(CDR(A),CDR(B),BOOL1,BOOL2) ALORS
DEBUT
    COMMENTAIRE :REGROUPEMENT D'ETATS DANS DES
    CAS PARTICULIERS ;
    SI BOOL2 ET NON BOOL1 ALORS
    DEBUT
        E:= VRAI ;
        ETA[J]:=B;
    FIN
    SINON
    SI J > COMPT OU J < COMPT ET T1=1 ET
    T2=1 ET TEZ[J]=NIL OU J < COMPT ET
    RTABLE[J]=RTABLE[J-1]+1 ET T1=1 ET T2=1 ET
    RTABLE[J] > COMPT ET REP[J]=NIL ET
    CDDR(ETA[RTABLE[J]])=NIL ALORS
    DEBUT
        E:= VRAI ;
        ETA[J]:=NIL;
        I1:=0;
        POUR X:=CDR(X) TANTQUE  X ≠ NIL FAIRE
        DEBUT
            T:=CAR(X);
            I1:=I1+1;
            Q:=CONS(CAR(T),CONS(CADR(T),R[I1]));
            T:=CONS(CAR(T),CADR(T));
            T1:=CONS(T,T1);
            ETA[J]:=CONS(Q,ETA[J]);
        FIN ;
        X:=B;
        POUR X:=CDR(X) TANTQUE  X ≠ NIL FAIRE
        DEBUT
            T:=CAR(X);
            Q:=CONS(CAR(T),CADR(T));
            SI NON MEMBER(Q,T1) ALORS ETA[J]:=
            CONS(CONS(CAR(T),CONS(CADR(T),
```

```
                    CDDR(T))),ETA[J]);
                FIN ;
                SI TEZ[J]=NIL ALORS
                DEBUT
                    TEZP[J]:=CONS(CONS(CAR(T),R[I1]),
                    NIL);
                FIN
                SINON
                SI RTABLE[J]=RTABLE[J-1]+1 ALORS
                DEBUT
                    T:=CADR(ETA[RTABLE[J]]);
                    Q:=CAR(ETA[RTABLE[J]]);
                    ETA[RTABLE[J]]:=CONS(CAR(T),
                    CONS(CADR(T),R[I1]));
                    ETA[RTABLE[J]]:=CONS(Q,
                    CONS(ETA[RTABLE[J]],NIL));
                FIN ;
                ETA[J]:=CONS(H,ETA[J]);
            FIN ;
            SI E ALORS
            DEBUT
                COMMENTAIRE :CONSTRUCTION DE LA LISTE
                REPRESENTANT LES ETATS DEJA CONSTRUITS ;
                REP[COMPT]:=UNION(ENTIERENLISTE(J),
                REP[COMPT]);
                J1:=J1-1;
                COMMENTAIRE :SUPPRESSION DE L'ETAT DEJA
                CONSTRUIT ;
                POUR K:=I PAS 1 JUSQUA I2+J1 FAIRE
                ETA[K]:=ETA[K+1];
                BOOL:= FAUX ;
                I:=I-1;
            FIN ;
        FIN ;
TROLONG:    FIN ;
        SI I < I2+J1 ALORS ALLERA  DANKE;
        SI J1=0 ALORS
        DEBUT
            ECRIRECHAINE(1,
            " IEME A PERMIS DE CONSTRUIRE DES ETATS " );
            ECRIRECHAINE(1,
            " QUI AVAIENT DEJA ETE CONSTRUITS " );
        FIN
        SINON
        SI J1=1 ALORS
        DEBUT
            ECRIRECHAINE(1,
            " IEME A PERMIS DE CONSTRUIRE L ETAT NUMERO " );
            ECRIREATOME(1,CDR(ENTIERENLISTE(I2+1)))
        FIN
        SINON
        DEBUT
            ECRIRECHAINE(1,
            " IEME A PERMIS DE CONSTRUIRE LES ETATS "
            ECRIRECHAINE(1, " SUIVANTS QUI SONT NUMEROTES DE
```

```
                    );
                    ECRIREATOME(1,CDR(ENTIERENLISTE(I2+1)));
                    ECRIRECHAINE(1, " A " );
                    ECRIREATOME(1,CDR(ENTIERENLISTE(I2+J1)));
                FIN ;
                ALALIGNE(1);
                POUR I:=I2+1 PAS 1 JUSQUA I2+J1 FAIRE
                ECRIRELISTE(1,ETA[I]);
ECRITURE:SAUTLIGNE;
                TEZ[COMPT]:=Z;
                TEZP[COMPT]:=ZP;
                SI Z=NIL ALORS ECRIREBARATIN(1, " Z EST VIDE " )
                SINON
                DEBUT
                    ECRIRELISTE(1,TEZ[COMPT]);
                    ECRIRECHAINE(1, "  REPRESENTE Z " );
                FIN ;
                SI ZP=NIL ALORS ECRIREBARATIN(1, " ZP EST VIDE " )
                SINON
                DEBUT
                    ECRIRELISTE(1,TEZP[COMPT]);
                    ECRIRECHAINE(1, "  REPRESENTE ZP " );
                FIN ;
                SI TEST ALORS
                DEBUT
                    ECRIREBARATIN(1,
                    " LES DEUX LISTES Z ET ZP SONT A " );
                    ECRIRECHAINE(1, " INTERSECTION VIDE " );
                FIN
                SINON
                DEBUT
                    ECRIREBARATIN(1,
                    " LES DEUX LISTES Z ET ZP N ETANT PAS " );
                    ECRIRECHAINE(1, " A INTERSECTION VIDE " );
                    ECRIRECHAINE(1, " LA GRAMMAIRE N EST PAS LR1 " );
                    PERFO:= FAUX ;
                FIN ;
                I2:=I2+J1;
                RTABLE[COMPT]:=I2;
                SI REP[COMPT] ≠ NIL ALORS
                DEBUT
                    ECRIREBARATIN(1,
                    " LA LISTE SUIVANTE CONTIENT LE NUMERO " );
                    ECRIRECHAINE(1,
                    " DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS " );
                    ECRIRELISTE(1,REP[COMPT]);
                FIN ;
                ALALIGNE(4);
            FIN ;
            TEMPS1:=TEMPS-TEMPS1;
            ALALIGNE(3);
            ECRIREBARATIN(1, "                    LA TABLE EST TERMINEE. "
            );
            ECRIRECHAINE(1, " LA CONSTRUCTION DE LA TABLE A DUREE " );
            ECRIREATOME(1,CDR(ENTIERENLISTE(TEMPS1)));
```

```
        ECRIRECHAINE(1, " DIXIEMES DE SECONDE " );
        SI PERFO ALORS
        DEBUT
            COMMENTAIRE :PERFORATION DE LA TABLE SI LA GRAMMAIRE
            EST LR1 ;
            POUR I:=0 PAS 1 JUSQUA I2 FAIRE
            DEBUT
                ECRIRELISTE(2,TEZP[I]);
                ECRIRELISTE(2,TEZ[I]);
                ETA[I]:=CAR(ETA[I]);
                ECRIRECHAINE(2, " " );
                ECRIRELISTE(2,ETA[I]);
                ECRIRELISTE(2,REP[I]);
                ECRIRELISTE(2,ENTIERENLISTE(RTABLE[I]));
            FIN ;
            ECRIREBARATIN(1,
            "                                                  " );
            ECRIRECHAINE(1, " LA GRAMMAIRE EST LR1 " );
        FIN
        SINON
        DEBUT
            ECRIREBARATIN(1,
            "                                                  " );
            ECRIRECHAINE(1, " LA GRAMMAIRE N EST PAS LR1 " );
        FIN ;
        ECRIRECHAINE(1, " NOMBRE D ETATS " );
        ECRIREATOME(1,CDR(ENTIERENLISTE(I2)));
        ECRIREBARATIN(1, "       TOUTES LES LISTES ONT OCCUPEES "
        );
        ECRIREATOME(1,CDR(ENTIERENLISTE(2*IPD)));
        ECRIRECHAINE(1,
        " MEMOIRES , IL Y AVAIT 15 PAGES EN MEMOIRE RAPIDE ET "
        );
        I:=(IPD-15*232) ./. 232+1;
        ECRIREATOME(1,CDR(ENTIERENLISTE(I)));
        ECRIRECHAINE(1, " SUR LES DISQUES " );
        SAUTPAGE;

    FIN ;
FIN BLOC UN ;
FIN PROGRAMME LISP ;
```

II - 6 Résultats.

Nous donnons les résultats de la construction de la table pour deux grammaires.

II - 6 - 1 Résultats grammaire G1.

On trouvera sur les pages 69 - 73 les résultats de la construction de la table pour la grammaire G1 du chapître 1 dont nous rappelons les productions (mises sous forme de liste).

0 -	(S0 B1 //)		
1 -	(L1 A)	5 -	(R1 A)
2 -	(L1 L1 N1 B)	6 -	(R1 B N1 R1)
3 -	(B1 A)	7 -	(N1 B N1 N1 B)
4 -	(B1 L1 R1)	8 -	(N1 A)

" convention pour représenter ⊣ .

II - 6 - 2 Grammaire d'un sous-ensemble d'Algol.

La deuxième table a été construite pour un sous ensemble d'un Algol édité. Nous donnons successivement la grammaire utilisée écrite sous forme normale de Backus, la grammaire correspondante sous forme de liste, les résultats du programme.

II - 6 - 2 - 1 Productions de la grammaire.

<S0> ::= <programme> ⊣

<bloc ou instruction composée> ::= début fin |
    début <intérieur de bloc ou d'instruction composée> fin |
    étiquette : <bloc ou instruction composée>

<programme> ::= <bloc ou instruction composée>

<déclaration> ::= procédure | procédure <instruction> |

****
CONSTRUCTION DE LA TABLE
****

(SC (0 0 //))
 CE 0 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 1 A 3

(B1 (0 1 //))
(A (1 1 A B) (3 1 //))
(L1 (2 1 A B) (4 1 //))

(A) REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(B1 (0 1 //))
 CE 1 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(A (1 1 A B) (3 1 //))
 CE 2 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
((3 //) (1 A B)) REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(L1 (2 1 A B) (4 1 //))
CE 3 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 4 A 7

(N1 (2 2 A B))
(A (5 1 //) (7 1 A B))
(B (6 1 //) (8 1 A B))
(R1 (4 2 //))

(A B) REPRESENTE Z
ZP EST VIDE
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(N1 (2 2 A B))
CE 4 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 8

(B (2 3 A B))

(B) REPRESENTE Z
ZP EST VIDE
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(A (5 1 //) (7 1 A B))
CE 5 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((7 A B) (5 //)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(B (6 1 //) (8 1 A B))
CE 6 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 9

(N1 (8 2 A B) (6 2 //))

(A B) REPRESENTE Z
ZP EST VIDE
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(6 5)

(R1 (4 2 //))
CE 7 IEME ETAT N A PAS PERMIS DE CONSTRUIRE 0 ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((4 //)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(B (2 3 A B))
CE 8 IEME ETAT N A PAS PERMIS DE CONSTRUIRE 0 ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((2 A B)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(N1 (8 2 A B) (6 2 //))
CE 9 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 10 A 11

(N1 (8 3 A B))
(R1 (6 3 //))

(A B) REPRESENTE Z
ZP EST VIDE
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(6 5)

(N1 (8 3 A B))
CE 10 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 12

(B (8 4 A B))

(B) REPRESENTE Z
ZP EST VIDE
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(R1 (6 3 //))
CE 11 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((6 //)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(B (8 4 A B))
CE 12 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((8 A 8)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

LA GRAMMAIRE EST LR1 NOMBRE D'ETATS 12

aiguillage <identificateur d'aiguillage> :=
<liste d'aiguillages> | tableau
<identificateur de tableau> crochet-gauche
<liste de paires de bornes> crochet-droit

<suite de déclarations> ::= <déclaration>|
<suite de déclarations> point-virgule <déclaration>

<début de début d'instruction pour> ::= pour <variable arithmétique>

<début d'instruction pour> ::= <début de début d'instruction pour> := <liste de pour>

<instruction pour> ::= étiquette : <instruction pour>|
<début d'instruction pour> faire|
<début d'instruction pour> faire <instruction>

<condition> ::= si <expression booléenne>

<début d'instruction conditionnelle> ::= <condition> alors <instruction inconditionnelle>
|<condition> alors

<instruction conditionnelle simplifiée> ::= <début d'instruction conditionnelle>|
<condition> alors <instruction pour>|

<instruction conditionnelle> ::= étiquette : <instruction conditionnelle>
|<instruction conditionnelle simplifiée>|
<début d'instruction conditionnelle> sinon|
<début d'instruction conditionnelle> sinon
<instruction>

<variable booléenne> ::= booléen|
identificateur-de-tableau-booléen crochet-gauche
<liste d'expressions arithmétiques> crochet-droit|
identificateur-de-procédure-booléenne

<variable arithmétique> ::= réel | identificateur-de-procédure-réelle|
entier|
identificateur-de-tableau-réel crochet-gauche
<liste d'expressions arithmétiques> crochet-droit|
identificateur-de-tableau-entier crochet-gauche
<liste d'expressions arithmétiques> crochet-droit|
identificateur-de-procédure-entière

<liste de parties gauches booléennes> ::= <variable booléenne>|
<liste de parties gauches booléennes>
:= <variable booléenne>

<liste de parties gauches arithmétiques> ::= <variable arithmétique>|
<liste de parties gauches arithmétiques> := <variable arithmétique>

<instruction de base> ::= allera <expression de désignation>|
étiquette : |
étiquette : <instruction de base>|
identificateur-de-procédure|
identificateur-de-procédure
(<liste de paramètres effectifs>)|
<liste de parties gauches arithmétiques>
:= <expression arithmétique>|
<liste de parties gauches booléennes>
:= <expression booléenne>

<instruction inconditionnelle> ::= <bloc ou instruction composée>|
<instruction de base>

<instruction> ::= <instruction inconditionnelle>|
<instruction conditionnelle> | <instruction-pour>

<intérieur de bloc ou d'instruction composée> ::= point-virgule|
point-virgule <instruction>|
<suite de déclarations> point-virgule |
<suite de déclarations>
point-virgule <instruction>|
<instruction>|
<intérieur de bloc ou d'instr.comp.>
point-virgule <instruction>|
<intérieur de bloc ou d'instr.comp.>
point-virgule

<début d'expression arithmétique> ::= <condition> alors
<expression-arithmétique-simple>

<début d'expression booléenne> ::= <condition> alors
<expression booléenne simple>

<début d'expression de désignation> ::= <condition> alors
<expression de désignation simple>

<expression de désignation simple> ::= (<expression de désignation>)|
étiquette|
identificateur-d'aiguillage
crochet-gauche <expression-arithmétique>
crochet-droit

<expression de désignation> ::= <expression de désignation simple>|
<début d'expression de désignation> sinon
<expression de désignation>

<expression arithmétique> ::= expression-arithmétique-simple|
<début d'expression arithmétique> sinon
<expression arithmétique>

<élément de liste de pour> ::= <expression arithmétique>|
<expression arithmétique> tant que
<expression booléenne>|
<expression arithmétique> pas
<expression arithmétique> jusqua
<expression arithmétique>

<liste de pour> ::= <élément de liste de pour>|
<liste de pour>, <élément de liste de pour>

<expression booléenne> ::= expression-booléenne-simple|
<début d'expression booléenne> sinon
<expression booléenne>

<liste d'expressions arithmétiques> ::= <expression arithmétique>|
<liste d'expressions arithmétiques>,
<expression arithmétique>

<paramètre effectif> ::= chaîne | identificateur-d'aiguillage|
identificateur-de-procédure-booléenne|
identificateur-de-procédure|
identificateur-de-procédure-entière|
identificateur-de-procédure-réelle|
identificateur-de-tableau-booléen|

identificateur-de-tableau-entier|
identificateur-de-tableau-réel|
<expression arithmétique>|
<expression booléenne>|
<expression de désignation>

<identificateur de tableau> ::= identificateur-de-tableau-réel|
identificateur-de-tableau-entier|
identificateur-de-tableau-booléen

<liste d'aiguillages> ::= <expression de désignation>|
<liste d'aiguillages>, <expression de désignation>

<paire de bornes> ::= <expression arithmétique> : <expression arithmétique>

<liste de paires de bornes> ::= <paire de bornes>|
<liste de paires de bornes>, <paire de bornes>

<liste de paramètres effectifs> ::= <paramètre effectif>|
<liste de paramètres effectifs>, <paramètre effectif>

II - 6 - 2 - 2 Productions de la grammaire sous forme de liste.

Nous avons utilisé le codage à la main :

```
0   (SO PR //)
1   (BIC 'début' 'fin')
2   (BIC 'début' IBC 'fin')
3   (BIC E : BIC)
4   (PR BIC)
5   (DE 'procédure')
6   (DE 'procédure IN)
7   (DE 'aiguillage' A := LD)
8   (DE 'tableau' IT .'L' LP 'J'.)
9   (SD DE)
10  (SD SD :: DE)
11  (DDIP 'pour' VA)
12  (DIP DDIP := LIP)
13  (IP E : IP)
14  (IP DIP 'faire')
15  (IP DIP 'faire' IN)
16  (CN 'si' EB)
17  (DIC CN 'alors' II)
18  (DIC CN 'alors')
19  (ICS DIC)
20  (ICS CN 'alors' IP)
21  (IC E : IC)
22  (IC ICS)
23  (IC DIC 'sinon')
24  (IC DIC 'sinon IN)
25  (VB L)
26  (VB K .'L' LEA 'J'.)
27  (VB G)
28  (VA R)
29  (VA F)
30  (VA I)
31  (VA M .'L' LEA 'J'.)
```

32 (VA J . 'L' LEA 'J' .)
33 (VA H)
34 (LPGB VB)
35 (LPGB LPGB := VB)
36 (LPGA VA)
37 (LPGA LPGA := VA)
38 (IB 'allera' ED)
39 (IB E :)
40 (IB E : IB)
41 (IB P)
42 (IB P 'L' LPE 'J')
43 (IB LPGA := EA)
44 (IB LPGB := EB)
45 (II BIC)
46 (II IB)
47 (IN II)
48 (IN IC)
49 (IN IP)
50 (IBC ::)
51 (IBC :: IN)
52 (IBC SD ::)
53 (IBC SD :: IN)
54 (IBC IN)
55 (IBC IBC :: IN)
56 (IBC IBC ::)
57 (DEA CN 'alors' S)
58 (DEB CN 'alors' B)
59 (DED CN 'alors' EDS)
60 (EDS 'L' ED 'J')
61 (EDS E)
62 (EDS A . 'L' EA 'J' .)
63 (ED EDS)
64 (ED DED 'sinon' ED)
65 (EA S)
66 (EA DEA 'sinon' EA)
67 (ELP EA)
68 (ELP EA 'tant que' EB)
69 (ELP EA 'pas' EA 'A' EA)
70 (LIP ELP)
71 (LIP LIP, ELP)
72 (EB B)
73 (EB DEB 'sinon' EB)
74 (LEA EA)
75 (LEA LEA, EA)
76 (PE C)
77 (PE A)
78 (PE G)
79 (PE P)
80 (PE H)
81 (PE F)
82 (PE K)
83 (PE J)
84 (PE M)
85 (PE EA)
86 (PE EB)
87 (PE ED)
88 (IT M)
89 (IT J)
90 (IT K)
91 (LD ED)
92 (LD LD, ED)
93 (PB EA : EA)
94 (LP PB)
95 (LP LP, PB)
96 (LPE PE)
97 (LPE LPE, PE)

II. 6 - 2 - 3 Résultats.

Sur les pages suivantes, on trouvera les résultats de la construction de la table pour la grammaire précédente.

```
**************************
*CONSTRUCTION DE LA TABLE*
**************************


( SO ( C C // ) )
 CE C IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 1 A 4

( PR ( C 1 // ) )
( BIC ( 4 1 // ) )
( 'DEBUT' ( 2 1 // 'SINON' :: 'FIN' ) ( 1 1 // 'SINON' :: 'FIN' ) )
( E ( 3 1 // 'SINON' :: 'FIN' ) )

( 'DEBUT' E )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE


( PR ( C 1 // ) )
 CE 1 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D'ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE


( BIC ( 4 1 // ) )
 CE 2 IEME ETAT N A PAS PERMIS DE CONSTRUIRE C ETATS CAR ON ETAIT EN FIN DE REGLE
```

```
 Z EST VIDE
( ( 4 // ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( 'DEBUT' ( 2 1 // 'SINON' :: 'FIN' ) ( 1 1 // 'SINON' :: 'FIN' ) )
 CE 3 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 5 A 40

( IBC ( 56 1 :: 'FIN' ) ( 55 1 :: 'FIN' ) ( 2 2 // 'SINON' :: 'FIN' ) )
( :: ( 51 1 :: 'FIN' ) ( 50 1 :: 'FIN' ) )
( SD ( 10 1 :: ) ( 53 1 :: 'FIN' ) ( 52 1 :: 'FIN' ) )
( IN ( 54 1 :: 'FIN' ) )
( DE ( 9 1 :: ) )
( II ( 47 1 :: 'FIN' ) )
( IC ( 48 1 :: 'FIN' ) )
( IP ( 49 1 :: 'FIN' ) )
( 'PROCEDURE' ( 6 1 :: ) ( 5 1 :: ) )
( 'AIGUILLAGE' ( 7 1 :: ) )
( 'TABLEAU' ( 8 1 :: ) )
( BIC ( 45 1 :: 'FIN' ) )
( IB ( 46 1 :: 'FIN' ) )
( ICS ( 22 1 :: 'FIN' ) )
( DIC ( 19 1 :: 'FIN' ) ( 24 1 :: 'FIN' ) ( 23 1 :: 'FIN' ) )
( DIP ( 15 1 :: 'FIN' ) ( 14 1 :: 'FIN' ) )
( 'ALLERA' ( 38 1 'SINON' :: 'FIN' ) )
( P ( 42 1 'SINON' :: 'FIN' ) ( 41 1 'SINON' :: 'FIN' ) )
( LPGA ( 37 1 := ) ( 43 1 'SINON' :: 'FIN' ) )
( LPGB ( 35 1 := ) ( 44 1 'SINON' :: 'FIN' ) )
( CN ( 18 1 'SINON' ) ( 17 1 'SINON' ) ( 20 1 :: 'FIN' ) )
( DDIP ( 12 1 'FAIRE' ) )
( VA ( 36 1 := ) )
( VB ( 34 1 := ) )
( 'SI' ( 16 1 'ALORS' ) )
( 'POUR' ( 11 1 := ) )
( R ( 28 1 := ) )
( F ( 29 1 := ) )
```

```
( I ( 30 1 := ) )
( M ( 31 1 := ) )
( J ( 32 1 := ) )
( H ( 33 1 := ) )
( L ( 25 1 := ) )
( K ( 26 1 := ) )
( G ( 27 1 := ) )
( 'FIN' ( 1 2 // 'SINON' :: 'FIN' ) )

( 'FIN' 'PROCEDURE' 'AIGUILLAGE' 'TABLEAU' :: 'SI' 'POUR' 'DEBUT' R F I M J H L K
G 'ALLERA' E P )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 3 4 )



( E ( 40 1 'SINON' :: 'FIN' ) ( 39 1 'SINON' :: 'FIN' ) ( 3 1 // 'SINON' :: 'FIN' )
( 13 1 :: 'FIN' ) ( 21 1 :: 'FIN' ) )
 CE 4 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 41

( : ( 21 2 :: 'FIN' ) ( 13 2 :: 'FIN' ) ( 3 2 // 'SINON' :: 'FIN' )
( 39 2 'SINON' :: 'FIN' ) ( 40 2 'SINON' :: 'FIN' ) )

( : )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( IBC ( 56 1 :: 'FIN' ) ( 55 1 :: 'FIN' ) ( 2 2 // 'SINON' :: 'FIN' ) )
 CE 5 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 42 A 43

( :: ( 55 2 :: 'FIN' ) ( 56 2 :: 'FIN' ) )
( 'FIN' ( 2 3 // 'SINON' :: 'FIN' ) )

( 'FIN' :: )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
```

```
( :: ( 51 1 :: 'FIN' ) ( 50 1 :: 'FIN' ) )
 CE 6 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 44

( IN ( 51 2 :: 'FIN' ) )

( 'SI' 'POUR' 'DEBUT' R F I M J H L K G 'ALLER' E P )   REPRESENTE Z
( ( 50 :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 3 20 19 18 4 17 16 12 11 10 )



( SD ( 10 1 :: ) ( 53 1 :: 'FIN' ) ( 52 1 :: 'FIN' ) )
 CE 7 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 45

( :: ( 52 2 :: 'FIN' ) ( 53 2 :: 'FIN' ) ( 10 2 :: ) )

( :: )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( IN ( 54 1 :: 'FIN' ) )
 CE 8 IEME ETAT N A PAS PERMIS DE CONSTRUIRE 0 ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 54 :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
```

(DE (9 1 ::))
CE 9 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((9 ::)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(II (47 1 :: 'FIN'))
CE 10 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((47 :: 'FIN')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(IC (48 1 :: 'FIN'))
CE 11 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((48 :: 'FIN')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(IP (49 1 :: 'FIN'))
CE 12 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

```
 Z EST VIDE
( ( 49 :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( 'PROCEDURE' ( 6 1 :: ) ( 5 1 :: ) )
 CE 13 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 46

( IN ( 6 2 :: ) )

( 'SI' 'POUR' 'DEBUT' R F I M J H L K G 'ALLERA' E P )   REPRESENTE Z
( ( 5 :: ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 3 20 19 18 4 17 16 12 11 10 )



( 'AIGUILLAGE' ( 7 1 :: ) )
 CE 14 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 47

( A ( 7 2 :: ) )

( A )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( 'TABLEAU' ( 8 1 :: ) )
 CE 15 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 48 A 51

( IT ( 8 2 :: ) )
( M ( 88 1 .'L' ) )
```

```
( J ( ES 1 .'L' ) )
( K ( SC 1 .'L' ) )

( M J K )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( BIC ( 45 1 :: 'FIN' ) )
 CE 16 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 45 :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( IB ( 46 1 :: 'FIN' ) )
 CE 17 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 46 :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( ICS ( 22 1 :: 'FIN' ) )
 CE 18 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 22 :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
```

```
( DIC ( 19 1 :: 'FIN' ) ( 24 1 :: 'FIN' ) ( 23 1 :: 'FIN' ) )
 CE 19 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 52

( 'SINON' ( 23 2 :: 'FIN' ) ( 24 2 :: 'FIN' ) )

( 'SINON' )   REPRESENTE Z
( ( 19 :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( DIP ( 15 1 :: 'FIN' ) ( 14 1 :: 'FIN' ) )
 CE 20 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 53

( 'FAIRE' ( 14 2 :: 'FIN' ) ( 15 2 :: 'FIN' ) )

( 'FAIRE' )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( 'ALLERA' ( 38 1 'SINON' :: 'FIN' ) )
 CE 21 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 54 A 60

( ED ( 38 2 'SINON' :: 'FIN' ) )
( EDS ( 63 1 'J' 'SINON' :: 'FIN' ) )
( DED ( 64 1 'J' 'SINON' :: 'FIN' ) )
( 'L' ( 60 1 'J' 'SINON' :: 'FIN' ) )
( E ( 61 1 'J' 'SINON' :: 'FIN' ) )
( A ( 62 1 'J' 'SINON' :: 'FIN' ) )
( CN ( 59 1 'SINON' ) )

( 'SI' 'L' E A )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 29 )
```

```
( P ( 42 1 'SINON' :: 'FIN' ) ( 41 1 'SINON' :: 'FIN' ) )
 CE 22 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 61

( 'L' ( 42 2 'SINON' :: 'FIN' ) )

( 'L' )   REPRESENTE Z
( ( 41 'SINON' :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( LPGA ( 37 1 := ) ( 43 1 'SINON' :: 'FIN' ) )
 CE 23 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 62

( := ( 43 2 'SINON' :: 'FIN' ) ( 37 2 := ) )

( := )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( LPGB ( 35 1 := ) ( 44 1 'SINON' :: 'FIN' ) )
 CE 24 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 63

( := ( 44 2 'SINON' :: 'FIN' ) ( 35 2 := ) )

( := )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
```

(CN (18 1 'SINON') (17 1 'SINON') (20 1 :: 'FIN'))
 CE 25 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 64

('ALORS' (20 2 :: 'FIN') (17 2 'SINON') (18 2 'SINON'))

('ALORS') REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(DDIP (12 1 'FAIRE'))
 CE 26 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 65

(:= (12 2 'FAIRE'))

(:=) REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(VA (36 1 :=))
 CE 27 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
((36 :=)) REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

```
( VB ( 34 1 := ) )
 CE 28 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 34 := ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( 'SI' ( 16 1 'ALORS' ) )
 CE 29 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 66 A 69

( EB ( 16 2 'ALORS' ) )
( B ( 72 1 'ALORS' ) )
( DEB ( 73 1 'ALORS' ) )
( CN ( 58 1 'SINON' ) )

( 'SI' B )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 29 )



( 'POUR' ( 11 1 := ) )
 CE 30 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 70

( VA ( 11 2 := ) )

( R F I M J H )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 36 35 34 33 32 31 )
```

(R (28 1 :=))
CE 31 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((28 :=)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(F (29 1 :=))
CE 32 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((29 :=)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(I (30 1 :=))
CE 33 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((30 :=)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(M (31 1 :=))
CE 34 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 71

(.'L' (31 2 :=))

(.°L°) REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(J (32 1 :=))
 CE 35 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 72

(.°L° (32 2 :=))

(.°L°) REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(H (33 1 :=))
 CE 36 IEME ETAT N A PAS PERMIS DE CONSTRUIRE C ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
((33 :=)) REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(L (25 1 :=))
 CE 37 IEME ETAT N A PAS PERMIS DE CONSTRUIRE C ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
((25 :=)) REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(K (26 1 :=))
 CE 3E IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 73

(.'L' (26 2 :=))

(.'L') REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(G (27 1 :=))
 CE 39 IEME ETAT N A PAS PERMIS DE CONSTRUIRE C ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
((27 :=)) REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

('FIN' (1 2 // 'SINON' :: 'FIN'))
 CE 4C IEME ETAT N A PAS PERMIS DE CONSTRUIRE C ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
((1 // 'SINON' :: 'FIN')) REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

```
( : ( 21 2 :: 'FIN' ) ( 13 2 :: 'FIN' ) ( 3 2 // 'SINON' :: 'FIN' )
( 39 2 'SINON' :: 'FIN' ) ( 40 2 'SINON' :: 'FIN' ) )
 CE 41 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 74 A 77

( IC ( 21 3 :: 'FIN' ) )
( IP ( 13 3 :: 'FIN' ) )
( BIC ( 3 3 // 'SINON' :: 'FIN' ) )
( IB ( 40 3 'SINON' :: 'FIN' ) )

( P 'ALLERA' G K L H J M I F R 'DEBUT' 'POUR' 'SI' E )   REPRESENTE Z
( ( 39 'SINON' :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 39 38 37 36 35 34 33 32 31 28 27 24 23 22 21 3 30 26 20 29 25 19 18 4 )



( :: ( 55 2 :: 'FIN' ) ( 56 2 :: 'FIN' ) )
 CE 42 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 78

( IN ( 55 3 :: 'FIN' ) )

( 'SI' 'POUR' 'DEBUT' R F I M J H L K G 'ALLERA' E P )   REPRESENTE Z
( ( 56 :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 3 20 19 18 4 17 16 12 11 10 )



( 'FIN' ( 2 3 // 'SINON' :: 'FIN' ) )
 CE 43 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 2 // 'SINON' :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
```

```
( IN ( 51 2 :: 'FIN' ) )
 CE 44 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 51 :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( :: ( 52 2 :: 'FIN' ) ( 53 2 :: 'FIN' ) ( 10 2 :: ) )
 DE 45 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 79 A 80

( IN ( 53 3 :: 'FIN' ) )
( DE ( 10 3 :: ) )

( 'TABLEAU' 'AIGUILLAGE' 'PROCEDURE' 'SI' 'POUR' 'DEBUT' R F I M J H L K G 'ALLERA' E P )
 REPRESENTE Z
( ( 52 :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 15 14 13 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 3 20 19 18 4 17 16 12 11
10 )



( IN ( 6 2 :: ) )
 CE 46 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 6 :: ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
```

```
( A ( 7 2 :: ) )
 CE 47 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 81

( := ( 7 3 :: ) )

( := )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( IT ( E 2 :: ) )
 CE 48 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 82

( .°L° ( 8 3 :: ) )

( .°L° )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( M ( EE 1 .°L° ) )
 CE 49 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( EE .°L° ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
```

```
( J ( ES 1 .'L' ) )
 CE SC IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( ES .'L' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( K ( SC 1 .'L' ) )
 CE 51 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( SC .'L' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( 'SINON' ( 23 2 :: 'FIN' ) ( 24 2 :: 'FIN' ) )
 CE 52 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 83

( IN ( 24 3 :: 'FIN' ) )

( 'SI' 'POUR' 'DEBUT' R F I M J H L K G 'ALLERA' E P )   REPRESENTE Z
( ( 23 :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 3 20 19 18 4 17 16 12 11 1 )
```

```
( 'FAIRE' ( 14 2 :: 'FIN' ) ( 15 2 :: 'FIN' ) )
 CE 53 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 84

( IN ( 15 3 :: 'FIN' ) )

( 'SI' 'POUR' 'DEBUT' R F I N J H L K G 'ALLER' E P )   REPRESENTE Z
( ( 14 :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 3 20 19 18 4 17 16 12 11 10 )



( ED ( 38 2 'SINON' :: 'FIN' ) )
 CE 54 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 38 'SINON' :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( EDS ( 63 1 'J' 'SINON' :: 'FIN' ) )
 CE 55 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 63 'J' 'SINON' :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
```

```
( DED ( 64 1 'J' 'SINON' :: 'FIN' ) )
 CE 56 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 85

( 'SINON' ( 64 2 'J' 'SINON' :: 'FIN' ) )

( 'SINON' )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( 'L' ( 60 1 'J' 'SINON' :: 'FIN' ) )
 CE 57 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 86

( ED ( 60 2 'J' 'SINON' :: 'FIN' ) )

( 'SI' 'L' E A )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 29 60 59 58 57 56 55 )



( E ( 61 1 'J' 'SINON' :: 'FIN' ) )
 CE 58 IEME ETAT N A PAS PERMIS DE CONSTRUIRE C ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 61 'J' 'SINON' :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
```

```
( A ( 62 1 'J' 'SINON' :: 'FIN' ) )
 CE 59 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 87

( .'L' ( 62 2 'J' 'SINON' :: 'FIN' ) )

( .'L' )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( ON ( 59 1 'SINON' ) )
 CE 60 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 88

( 'ALORS' ( 59 2 'SINON' ) )

( 'ALORS' )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( 'L' ( 42 2 'SINON' :: 'FIN' ) )
 CE 61 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 89 A 115

( LPE ( 97 1 , 'J' ) ( 42 3 'SINON' :: 'FIN' ) )
( PE ( 96 1 , 'J' ) )
( C ( 76 1 , 'J' ) )
( A ( 62 1 , 'J' ) ( 77 1 , 'J' ) )
( G ( 78 1 , 'J' ) )
( P ( 79 1 , 'J' ) )
( H ( 60 1 , 'J' ) )
( F ( 61 1 , 'J' ) )
( K ( 62 1 , 'J' ) )
( J ( 63 1 , 'J' ) )
( M ( 64 1 , 'J' ) )
```

```
( EA ( 85 1 , 'J' ) )
( EB ( 86 1 , 'J' ) )
( ED ( 87 1 , 'J' ) )
( S ( 65 1 , 'J' 'J' . 'TANTQUE' 'PAS' : 'A' ) )
( DEA ( 66 1 , 'J' 'J' . 'TANTQUE' 'PAS' : 'A' ) )
( 'L' ( 6C 1 , 'J' ) )

( S B 'SI' 'L' E C A G P H F K J M )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 29 56 69 56 55 68 67 )



( := ( 43 2 'SINON' :: 'FIN' ) ( 37 2 := ) )
 CE 62 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 106 A 107

( EA ( 43 3 'SINON' :: 'FIN' ) )
( VA ( 37 3 := ) )

( H J M I F R 'SI' S )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 36 35 34 33 32 31 29 69 104 103 )



( := ( 44 2 'SINON' :: 'FIN' ) ( 35 2 := ) )
 CE 63 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 108 A 109

( EB ( 44 3 'SINON' :: 'FIN' ) )
( VB ( 35 3 := ) )

( G K L 'SI' B )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 39 38 37 29 69 68 67 )
```

('ALORS' (20 2 :: 'FIN') (17 2 'SINON') (18 2 'SINON'))
 CE 64 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 110 A 111

(IP (20 3 :: 'FIN'))
(II (17 3 'SINON'))

(P 'ALLERA' G K L H J M I F R 'DEBUT' 'POUR' E) REPRESENTE Z
((18 'SINON')) REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(39 38 37 36 35 34 33 32 31 28 27 24 23 22 21 3 17 16 30 26 20 4)

(:= (12 2 'FAIRE'))
 CE 65 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 112 A 114

(LIP (71 1 , 'FAIRE') (12 3 'FAIRE'))
(ELP (70 1 , 'FAIRE'))
(EA (69 1 , 'FAIRE') (68 1 , 'FAIRE') (67 1 , 'FAIRE'))

('SI' S) REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(29 69 104 103)

(EB (16 2 'ALORS'))
 CE 66 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
((16 'ALORS')) REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

```
( B ( 72 1 'SINON' :: 'FIN' 'ALORS' , 'J' ) )
 CE 67 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 72 'SINON' :: 'FIN' 'ALORS' , 'J' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( DEB ( 73 1 'SINON' :: 'FIN' 'ALORS' , 'J' ) )
 CE 68 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 115

( 'SINON' ( 73 2 'SINON' :: 'FIN' 'ALORS' , 'J' ) )

( 'SINON' )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( CN ( 59 1 'SINON' ) ( 58 1 'SINON' ) ( 57 1 'SINON' ) )
 CE 69 IEME ETAT A PERMIS DE CONSTRUIRE DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS


( 'ALORS' )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 68 )
```

```
( VA ( 11 2 := ) )
 CE 70 IEME ETAT N A PAS PERMIS DE CONSTRUIRE 0 ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 11 := ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( .'L' ( 31 2 := ) )
 CE 71 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 116 A 117

( LEA ( 75 1 , 'J'. ) ( 31 3 := ) )
( EA ( 74 1 , 'J'. ) )

( 'SI' S )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 29 69 104 103 )



( .'L' ( 32 2 := ) )
 CE 72 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 118

( LEA ( 75 1 , 'J'. ) ( 32 3 := ) )

( 'SI' S )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 29 69 104 103 117 )
```

```
( .'L' ( 26 2 := ) )
 CE 73 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 119

( LEA ( 75 1 , 'J'. ) ( 26 3 := ) )

( 'SI' S )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 29 69 104 103 117 )



( IC ( 21 3 :: 'FIN' ) )
 CE 74 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 21 :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( IP ( 13 3 :: 'FIN' ) )
 CE 75 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 13 :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
```

```
( BIC ( 3 3 // 'SINON' :: 'FIN' ) )
 CE 76 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 3 // 'SINON' :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( IB ( 40 3 'SINON' :: 'FIN' ) )
 CE 77 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 40 'SINON' :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( IN ( 55 3 :: 'FIN' ) )
 CE 78 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 55 :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( IN ( 53 3 :: 'FIN' ) )
 CE 79 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE
```

```
 Z EST VIDE
( ( 53 :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( DE ( 10 3 :: ) )
 CE 80 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 10 :: ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( := ( 7 3 :: ) )
 CE 81 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 120 A 121

( LD ( 92 1 , :: ) ( 7 4 :: ) )
( ED ( 91 1 , :: ) )

( 'SI' 'L' E A )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 29 60 59 58 105 56 55 )



( .'L' ( 8 3 :: ) )
 CE 82 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 122 A 124

( LP ( 95 1 , 'J'. ) ( 8 4 :: ) )
( PB ( 94 1 , 'J'. ) )
```

(EA (93 1 , 'J'.))

('SI' S) REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(29 69 104 103)

(IN (24 3 :: 'FIN'))
 CE 83 IEME ETAT N A PAS PERMIS DE CONSTRUIRE E ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
((24 :: 'FIN')) REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(IN (15 3 :: 'FIN'))
 CE 84 IEME ETAT N A PAS PERMIS DE CONSTRUIRE E ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
((15 :: 'FIN')) REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

('SINON' (64 2 , 'J' 'SINON' :: 'FIN'))
 CE 85 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 125

(ED (64 3 , 'J' 'SINON' :: 'FIN'))

('SI' 'L' E A) REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(29 60 59 58 105 56 55)

(ED (60 2 'J' 'SINON' :: 'FIN'))
CE 66 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 126

('J' (60 3 'J' 'SINON' :: 'FIN'))

('J') REPRESENTE Z
ZP EST VIDE
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

('L' (62 2 'J' 'SINON' :: 'FIN'))
CE 67 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 127

(EA (62 3 'J' 'SINON' :: 'FIN'))

('SI' S) REPRESENTE Z
ZP EST VIDE
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(29 65 104 103)

('ALORS' (57 2 'SINON') (58 2 'SINON') (59 2 'SINON'))
CE 66 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 126 A 130

(S (57 3 'SINON'))
(B (58 3 'SINON'))
(EDS (59 3 'SINON'))

(A E 'L' B S) REPRESENTE Z
ZP EST VIDE
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(59 58 57)

(LPE (S7 1 , 'J') (42 3 'SINCN' :: 'FIN'))
 CE ES IEME ETAT A PERMIS DE CCNSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 131 A 132

(, (S7 2 , 'J'))
('J' (42 4 'SINCN' :: 'FIN'))

('J' ,) REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(PE (S6 1 , 'J'))
 CE SC IEME ETAT N A PAS PERMIS DE CCNSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
((S6 , 'J')) REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(C (76 1 , 'J'))
 CE S1 IEME ETAT N A PAS PERMIS DE CCNSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
((76 , 'J')) REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(A (62 1 , 'J') (77 1 , 'J'))
CE 92 IEME ETAT A PERMIS DE CONSTRUIRE DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS

(. 'L') REPRESENTE Z
((77 , 'J')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(87)

(G (78 1 , 'J'))
CE 93 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((78 , 'J')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(P (79 1 , 'J'))
CE 94 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((79 , 'J')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(H (80 1 , 'J'))
CE 95 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

```
 Z EST VIDE
( ( EC , 'J' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( F ( E1 1 , 'J' ) )
 CE 96 IEME ETAT N A PAS PERMIS DE CONSTRUIRE C ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( E1 , 'J' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( K ( E2 1 , 'J' ) )
 CE 97 IEME ETAT N A PAS PERMIS DE CONSTRUIRE C ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( E2 , 'J' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( J ( E3 1 , 'J' ) )
 CE 98 IEME ETAT N A PAS PERMIS DE CONSTRUIRE C ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( E3 , 'J' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
```

(M (84 1 , 'J'))
CE 99 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((84 , 'J')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(EA (85 1 , 'J'))
CE 100 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((85 , 'J')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(EB (86 1 , 'J'))
CE 101 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((86 , 'J')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(ED (87 1 , 'J'))
CE 102 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

```
 Z EST VIDE
( ( 67 , 'J' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( S ( 65 1 'FAIRE' , 'J' 'SINON' :: 'FIN' 'J'. 'TANTQUE' 'PAS' : 'A' ) )
 CE 103 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 65 'FAIRE' , 'J' 'SINON' :: 'FIN' 'J'. 'TANTQUE' 'PAS' : 'A' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( DEA ( 66 1 'FAIRE' , 'J' 'SINON' :: 'FIN' 'J'. 'TANTQUE' 'PAS' : 'A' ) )
 CE 104 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 133

( 'SINON' ( 66 2 'FAIRE' , 'J' 'SINON' :: 'FIN' 'J'. 'TANTQUE' 'PAS' : 'A' ) )

( 'SINON' )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( 'L' ( 60 1 , 'J' 'SINON' :: 'FIN' ) )
 CE 105 IEME ETAT A PERMIS DE CONSTRUIRE DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS


( 'SI' 'L' E A )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 29 60 59 58 57 56 55 86 )
```

(EA (43 3 'SINON' :: 'FIN'))
CE 1CE IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((43 'SINON' :: 'FIN')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(VA (37 3 :=))
CE 1C7 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((37 :=)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(EB (44 3 'SINON' :: 'FIN'))
CE 1CE IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((44 'SINON' :: 'FIN')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(VB (35 3 :=))
CE 1C5 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

```
 Z EST VIDE
( ( 35 := ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( IP ( 2C 3 :: 'FIN' ) )
 CE 11C IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 2C :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( II ( 17 3 'SINON' ) )
 CE 111 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 17 'SINON' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( LIP ( 71 1 , 'FAIRE' ) ( 12 3 'FAIRE' ) )
 CE 112 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 134

( , ( 71 2 , 'FAIRE' ) )

( , )   REPRESENTE Z
( ( 12 'FAIRE' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
```

(ELP (70 1 , 'FAIRE'))
CE 113 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((70 , 'FAIRE')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(EA (69 1 , 'FAIRE') (68 1 , 'FAIRE') (67 1 , 'FAIRE'))
CE 114 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 135 A 136

('PAS' (69 2 , 'FAIRE'))
('TANTQUE' (68 2 , 'FAIRE'))

('TANTQUE' 'PAS') REPRESENTE Z
((67 , 'FAIRE')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

('SINON' (73 2 'SINON' :: 'FIN' 'ALORS' , 'J'))
CE 115 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 137

(EB (73 3 'SINON' :: 'FIN' 'ALORS' , 'J'))

('SI' B) REPRESENTE Z
ZP EST VIDE
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(29 69 68 67)

(LEA (75 1 , 'J'.) (31 3 :=))
 CE 116 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 138 A 139

(, (75 2 , 'J'.))
('J'. (31 4 :=))

('J'. ,) REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(EA (74 1 , 'J'.))
 CE 117 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
((74 , 'J'.)) REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(LEA (75 1 , 'J'.) (32 3 :=))
 CE 118 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 140

('J'. (32 4 :=))

('J'. ,) REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(138)

(LEA (75 1 , 'J'.) (26 3 :=))
 CE 115 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 141

('J'. (26 4 :=))

('J'. ,)   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(136)

(LD (S2 1 , ::) (7 4 ::))
 CE 120 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 142

(, (S2 2 , ::))

(,)   REPRESENTE Z
((7 ::))   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(ED (S1 1 , ::))
 CE 121 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
((S1 , ::))   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(LP ($5 1 , °J°.) (8 4 ::))
 CE 122 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 143 A 144

(, ($5 2 , °J°.))
(°J°. (8 5 ::))

(°J°. ,)   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(PB ($4 1 , °J°.))
 CE 123 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
(($4 , °J°.))   REPRESENTE ZF
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(EA ($3 1 , °J°.))
 CE 124 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 145

(: ($3 2 , °J°.))

(:)   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(ED (64 3 , 'J' 'SINON' :: 'FIN'))
CE 125 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((64 , 'J' 'SINON' :: 'FIN')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

('J' (6C 3 , 'J' 'SINON' :: 'FIN'))
CE 126 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((6C , 'J' 'SINON' :: 'FIN')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(EA (62 3 , 'J' 'SINON' :: 'FIN'))
CE 127 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 146

('J'. (62 4 , 'J' 'SINON' :: 'FIN'))

('J'.) REPRESENTE Z
ZP EST VIDE
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(S (57 3 'SINON'))
CE 128 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((57 'SINON')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(B (58 3 'SINON'))
CE 129 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((58 'SINON')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(EOS (59 3 'SINON'))
CE 130 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((59 'SINON')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(, (97 2 , 'J'))
CE 131 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 147

(PE (97 3 , 'J'))

```
( S B 'SI' 'L' E C A G P H F K J M )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 29 58 105 69 56 55 68 67 104 103 102 101 100 99 98 97 96 95 94 93 92 91 )



( 'J' ( 42 4 'SINON' :: 'FIN' ) )
 CE 132 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 42 'SINON' :: 'FIN' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( 'SINON' ( 66 2 'FAIRE' , 'J' 'SINON' :: 'FIN' 'J'. 'TANTQUE' 'PAS' : 'A' ) )
 CE 133 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 148

( EA ( 66 3 'FAIRE' , 'J' 'SINON' :: 'FIN' 'J'. 'TANTQUE' 'PAS' : 'A' ) )

( 'SI' S )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 29 69 104 103 )



( , ( 71 2 , 'FAIRE' ) )
 CE 134 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 149

( ELP ( 71 3 , 'FAIRE' ) )

( 'SI' S )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 29 69 104 103 114 )
```

```
( 'PAS' ( 69 2 , 'FAIRE' ) )
 CE 135 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 150

( EA ( 69 3 , 'FAIRE' ) )

( 'SI' S )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 29 69 104 103 )



( 'TANTQUE' ( 68 2 , 'FAIRE' ) )
 CE 136 IEME ETAT A PERMIS DE CONSTRUIRE LES ETATS SUIVANTS NUMEROTES DE 151 A 152

( EB ( 68 3 , 'FAIRE' ) )
( DEB ( 73 1 , 'FAIRE' ) )

( 'SI' B )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
( 29 69 67 )



( EB ( 73 3 'SINON' :: 'FIN' 'ALORS' , 'J' ) )
 CE 137 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 73 'SINON' :: 'FIN' 'ALORS' , 'J' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
```

(, (75 2 , 'J'.))
CE 138 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 153

(EA (75 3 , 'J'.))

('SI' S) REPRESENTE Z
ZP EST VIDE
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(29 69 104 103)

('J'. (31 4 :=))
CE 139 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((31 :=)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

('J'. (32 4 :=))
CE 140 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((32 :=)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

('J'. (26 4 :=))
CE 141 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((26 :=)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(, (52 2 , ::))
CE 142 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 154

(ED (52 3 , ::))

('SI' 'L' E A) REPRESENTE Z
ZP EST VIDE
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(29 60 59 58 105 56 55)

(, (55 2 , 'J'.))
CE 143 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 155

(PB (55 3 , 'J'.))

('SI' S) REPRESENTE Z
ZP EST VIDE
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(29 69 104 103 124)

('J'. (8 5 ::))
CE 144 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((E ::)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(: (93 2 , 'J'.))
CE 145 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 156

(EA (93 3 , 'J'.))

('SI' S) REPRESENTE Z
ZP EST VIDE
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(29 69 104 103)

('J'. (62 4 , 'J' 'SINON' :: 'FIN'))
CE 146 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((62 , 'J' 'SINON' :: 'FIN')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(PE (97 3 , 'J'))
CE 147 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((97 , 'J')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(EA (66 3 'FAIRE' , 'J' 'SINON' :: 'FIN' 'J' . 'TANTQUE' 'PAS' : 'A'))
CE 148 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((66 'FAIRE' , 'J' 'SINON' :: 'FIN' 'J' . 'TANTQUE' 'PAS' : 'A')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(ELP (71 3 , 'FAIRE'))
CE 149 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((71 , 'FAIRE')) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(EA (69 3 , 'FAIRE'))
CE 150 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 157

('A' (69 4 , 'FAIRE'))

```
( 'A' )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( EB ( 68 3 , 'FAIRE' ) )
 CE 151 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 68 , 'FAIRE' ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( DEB ( 73 1 , 'FAIRE' ) )
 CE 152 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 158

( 'SINON' ( 73 2 , 'FAIRE' ) )

( 'SINON' )   REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE



( EA ( 75 3 , 'J'. ) )
 CE 153 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
( ( 75 , 'J'. ) )   REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
```

(ED (92 3 , ::))
CE 154 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((92 , ::)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(PB (95 3 , 'J'.))
CE 155 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((95 , 'J'.)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

(EA (93 3 , 'J'.))
CE 156 IEME ETAT N A PAS PERMIS DE CONSTRUIRE D ETATS CAR ON ETAIT EN FIN DE REGLE

Z EST VIDE
((93 , 'J'.)) REPRESENTE ZP
LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

('A' (69 4 , 'FAIRE'))
CE 157 IEME ETAT A PERMIS DE CONSTRUIRE L ETAT NUMERO 159

(EA (69 5 , 'FAIRE'))

(°SI° S) REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(29 69 104 103)

(°SINON° (73 2 , °FAIRE°))
 CE 158 IEME ETAT A PERMIS DE CONSTRUIRE DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS

(°SI° B) REPRESENTE Z
 ZP EST VIDE
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE
 LA LISTE SUIVANTE CONTIENT LE NUMERO DES ETATS QUI AVAIENT DEJA ETE CONSTRUITS
(29 69 152 67 137)

(EA (69 S , °FAIRE°))
 CE 159 IEME ETAT N A PAS PERMIS DE CONSTRUIRE 0 ETATS CAR ON ETAIT EN FIN DE REGLE

 Z EST VIDE
((69 , °FAIRE°)) REPRESENTE ZP
 LES DEUX LISTES Z ET ZP SONT A INTERSECTION VIDE

LA GRAMMAIRE EST LR1 NOMBRE D°ETATS 159

Nous avons également essayé de faire la construction de la table pour d'autres grammaires.

a) un sous-ensemble de fortran qui n'est pas LR(1) a cause des ambiguités entre variable-arithmétique et variable-booléenne.

b) la méta-grammaire d'Algol 67 [VW] qui n'est pas LR(1) à cause du "and" qui se trouve dans de nombreuses productions.

c) les deux grammaires données par L. TRILLING [TR] :

- la première sur son langage de génération qui n'est pas LR(1)
- la deuxième sur un langage dérivé d'algol X qui est LR(1) après quelques modifications que nous allons donner.

II - 7 Utilisation de la table pour la mise au point des grammaires.

Si une grammaire G n'est pas LR(1), on peut essayer de trouver une grammaire G' équivalente qui le soit. On pourra ainsi

1°) affirmer que G' n'est pas ambiguë

2°) donner une méthode d'analyse très rapide.

Il est à remarquer que G pouvait très bien ne pas être ambiguë.

Lorsque la condition LR(1) n'est pas vérifiée, on détermine très facilement en examinant les résultats, quelles sont les productions et le contexte qui ne satisfont pas à la condition. On peut donc essayer en modifiant ces productions d'obtenir une grammaire équivalente qui soit LR(1).

Exemple.

Soit la grammaire dérivé d'algol X de L. TRILLING [TR] dont on trouvera un listage sur les pages suivantes.

La table que nous avons construite avec cette grammaire nous a indiqué que la grammaire n'est pas LR(1) pour les productions 13 et 14 à cause du symbole point virgule (;).
En effet soient les productions 1, 13 et 14

1 <bloc> ::= debut <declaration> ; <corps> fin
13 <declaration> ::= <declaration unique>|
14 <declaration unique> ; <declaration>

Dans la production numéro 1 <declaration> est suivie du symbole ";", donc on effectuera une réduction avec la production numéro 13 lorsque l'on trouvera un ";" après <declaration unique>.
Si on regarde la production numéro 14, après <declaration unique> il y a également un ";".

On a donc le choix entre effectuer une réduction avec 13 et ne pas effectuer de réduction mais continuer pour terminer la production numéro 14.

Pour supprimer cette alternative il suffit de remplacer les productions 1, 13 et 14 par :

1' <bloc> ::= début <déclaration> <corps> fin
13' <déclaration> ::= <déclaration unique> ;|
14' <déclaration unique> ; <déclaration>

Etant donné que H_1(corps) et H_1(déclaration) sont des ensembles disjoints, lorsque l'on aura obtenu "<déclaration unique>;", si le symbole qui suit appartient à H_1(corps

on appliquera une réduction avec la production numéro 13' sinon on translatera ce symbole et on attendra d'avoir obtenu <déclaration> pour appliquer une réduction avec 14'.
Dans ce cas, la condition LR(1) est vérifiée.

Cette condition n'était également pas vérifiée pour les productions numéro 47 et 61 à cause de la parenthèse droite. En effet :

47 <facteur> ::= (<expression arithmétique>)
61 <expression simple> ::= <expression arithmétique>|
62 <primaire logique>
63 <relation> ::= <expression simple> = <expression simple>

Etant donné que <expression simple> se réécrit <expression arithmétique> et que cette métavariable peut être suivie d'une parenthèse droite (production numéro 47), il en sera de même pour <expression simple>.
De ce fait, lorsque l'on trouve <expression arithmétique> suivie du symbole ")", on ne sait pas si l'on doit effectuer une réduction avec la production numéro 61 ou si l'on doit translater cette parenthèse pour ensuite effectuer une réduction avec la production numéro 47.

Nous avons donc supprimé la métavariable <expression simple> en supprimant les productions numéros 61 et 62 et en modifiant la production numéro 63 de la manière suivante :

<relation> ::= <expression arithmétique> = <expression arithmétique>
| <primaire logique> = <primaire logique>

Ainsi la présence du symbole "=" nous permet de décider avec la présence de la parenthèse droite de la production numéro 47.

- La table permet également de détecter les productions inutiles. Au cours de nos expériences nous avons trouvé des grammaires ayant des productions de la forme suivante :

α <expression arithmétique> ::= <primaire arithmétique>|
β <variable>
γ <primaire arithmétique> ::= <variable>

La table nous indique que la grammaire n'est pas LR(1) pour les productions β et γ. En effet pour le même contexte, on a deux réductions à appliquer.

Dans ce cas là, il suffit de supprimer la production β.
Il est à noter que ce genre de productions existent en algol 68.

Conclusion.

Il n'y a aucune méthode générale pour rendre une grammaire LR(1). La table nous indique les numéros de production et le contexte qui ne satisfont pas à la condition ; il faut essayer empiriquement au vu de ces renseignements et de la grammaire de modifier un certain nombre de productions.

Grammaire dérivé d'algol X de L. TRILLING.

<S0> ::= <programme> ⊣
<bloc> ::= début <déclaration> ; <corps> fin
<déclaration simple> ::= entier variable-entière|
entier composant-entier|
logique variable-logique|
logique composant-logique|
(nouveau-type) variable-nouvelle|
(nouveau-type) composant-nouveau|
libre variable-libre|
libre composant-libre|
référence <déclaration simple>

<déclaration unique> ::= <déclaration simple>|
(<déclaration de composants>) nouveau-type

<déclaration> ::= <déclaration unique>|
<déclaration unique> ; <déclaration>

<déclaration de composants> ::= <déclaration simple>|
<déclaration simple>, <déclaration de composants>

<instruction composée> ::= début <corps> fin

<programme> ::= <bloc>|<instruction composée>

<expression nouvelle 1> ::= variable-nouvelle|composant-nouveau
de <expression nouvelle 1>
' valeur <expression nouvelle 1>

<expression libre> ::= variable-libre|composant-libre de
<expression nouvelle 1>

<proposition si> ::= si <expression logique> alors

<variable entière 1> ::= variable-entière|composant-entier
de <expression nouvelle 1>|
valeur <variable entière 1>

<expression non libre> ::= <expression nouvelle 1>|
<variable entière 1>|
<variable logique>

<expression 1> ::= <expression non libre>|<expression libre>|
toin

<instruction> ::= <expression 1> := <expression>|
étiquette : <instruction>|rien|
allera étiquette|<proposition si>
<instruction> sinon <instruction>|
<bloc>|<instruction composée>

<corps> ::= <instruction>|<instruction>; <corps>

<facteur> ::=(<expression arithmétique>)|<variable entière 1>|
nombre-entier

<terme> ::= <facteur>|<terme> mul <facteur>

<expression arithmétique> ::= <terme>|
<expression arithmétique> + <terme>

<primaire logique> ::= (<expression logique>)|
<variable logique 1>|valeur-logique

<secondaire logique> ::= non <primaire logique>|
<primaire logique>

<facteur logique> ::= <facteur logique> et <secondaire logique>

<expression simple> ::= <expression arithmétique>
|<primaire logique>

<relation> ::= <expression simple>= <expression simple>|
<expression 1> est <expression 1>|
<expression non libre> .= <expression libre>|
<expression arithmétique> inf <expression arithmétique>|
<expression arithmétique> sup <expression arithmétique>

<expression nouvelle> ::= <expression nouvelle 1>|nouveau-type
(<liste de composants>)

<expression logique> ::= <expression logique> ou bien <facteur logique>|
<facteur logique>|<relation>

<expression> ::= <expression arithmétique>|<expression logique>|
<expression nouvelle>|<expression libre>|toin

<liste de composants> ::= <expression>|
<expression>, <liste de composants>

<variable logique 1> ::= variable-logique|composant-logique de
<expression nouvelle 1>|
valeur <variable logique 1>

II - 8 Utilisation des disques.

II - 8 - 1 Principe.

Pour des grammaires relativement importantes nous avons été amenés à utiliser les disques pour mémoriser les listes.
Le programme de gestion de disques que nous avons utilisé a été écrit par Y. Siret suivant un schéma défini par J. COHEN [CO] . La zone de listes a été divisée en pages. Chaque page contenant 2 fois 230 mots. Les premiers correspondant à TCAR, les seconds à TCDR.
Naturellement, on commence par remplir les pages qui se trouvent en mémoire rapide. Lorsque celle-ci est pleine, on envoie sur disques la page ayant eu la plus longue période d'inactivité et on appelle en mémoire rapide la page contenant l'information demandée. L'avantage des disques est que l'on peut stocker jusqu'à 500.000 mots. L'inconvénient en est la lenteur ; en effet le temps moyen d'accès des disques est environ 10^4 fois plus grand que celui de la mémoire rapide. *

II - 8 - 2 Application à notre problème.

La première fois que nous créons un état, il y a de fortes chances pour que la liste correspondant à cet état se trouve sur une page ou dans le cas le plus défavorable sur deux pages. Par contre, lorsque nous modifions cet état par augmentation du contexte par exemple, il va se trouver sur plusieurs pages. D'autre part, n'oublions pas que lorsque le programme a créé un état, il va chercher si parmi tous les états déjà créés, il n'en existe pas un qui soit identique ou qui le contienne. Il en résulte, lorsque la grammaire est assez importante et que les productions sont récursives (comme dans le cas de <expression arithmétique>) que l'on va appeler un nombre considérable de pages. Au bout d'un certain temps, on peut être sûr que

---

* ordinateur I.B.M. 7044 de l'Université de GRENOBLE.

l'ordinateur utilise la plus grande partie de son temps à appeler des pages.

Pour le sous-ensemble d'algol (II - 6 - 2) qui comprenait 97 productions, nous avons utilisé 36 pages dont 15 en mémoire rapide. Le temps d'exécution s'est élevé à 29 minutes.

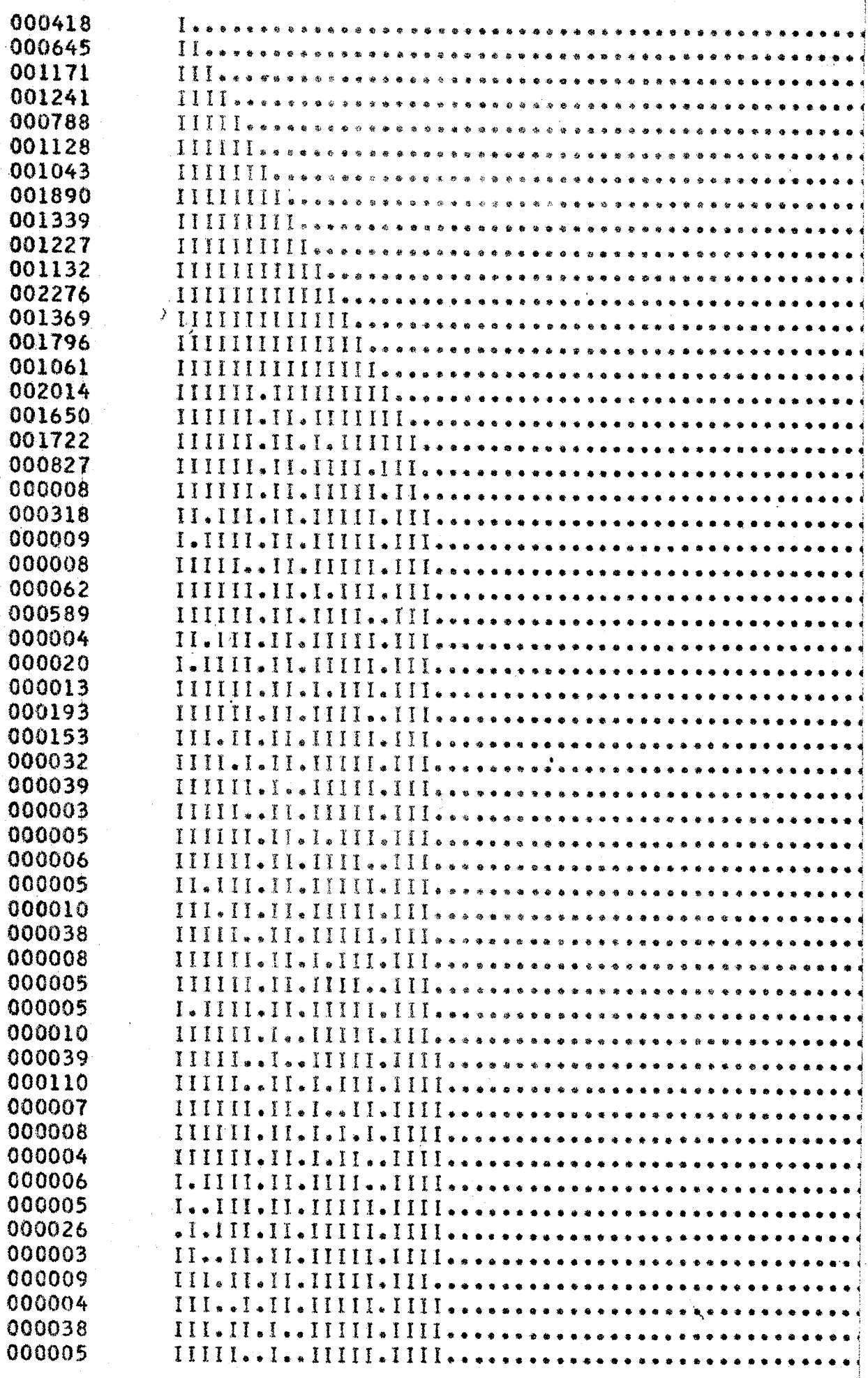
Par contre pour le même sous-ensemble d'algol auquel on avait ajouté la définition des expressions arithmétiques et booléennes (137 productions), au bout de 3 heures on avait utilisé 77 pages. A ce moment là on avait créé 230 états dont 186 avaient été examinés. Nous pensons qu'il faudrait une dizaine d'heures pour construire la table complète *

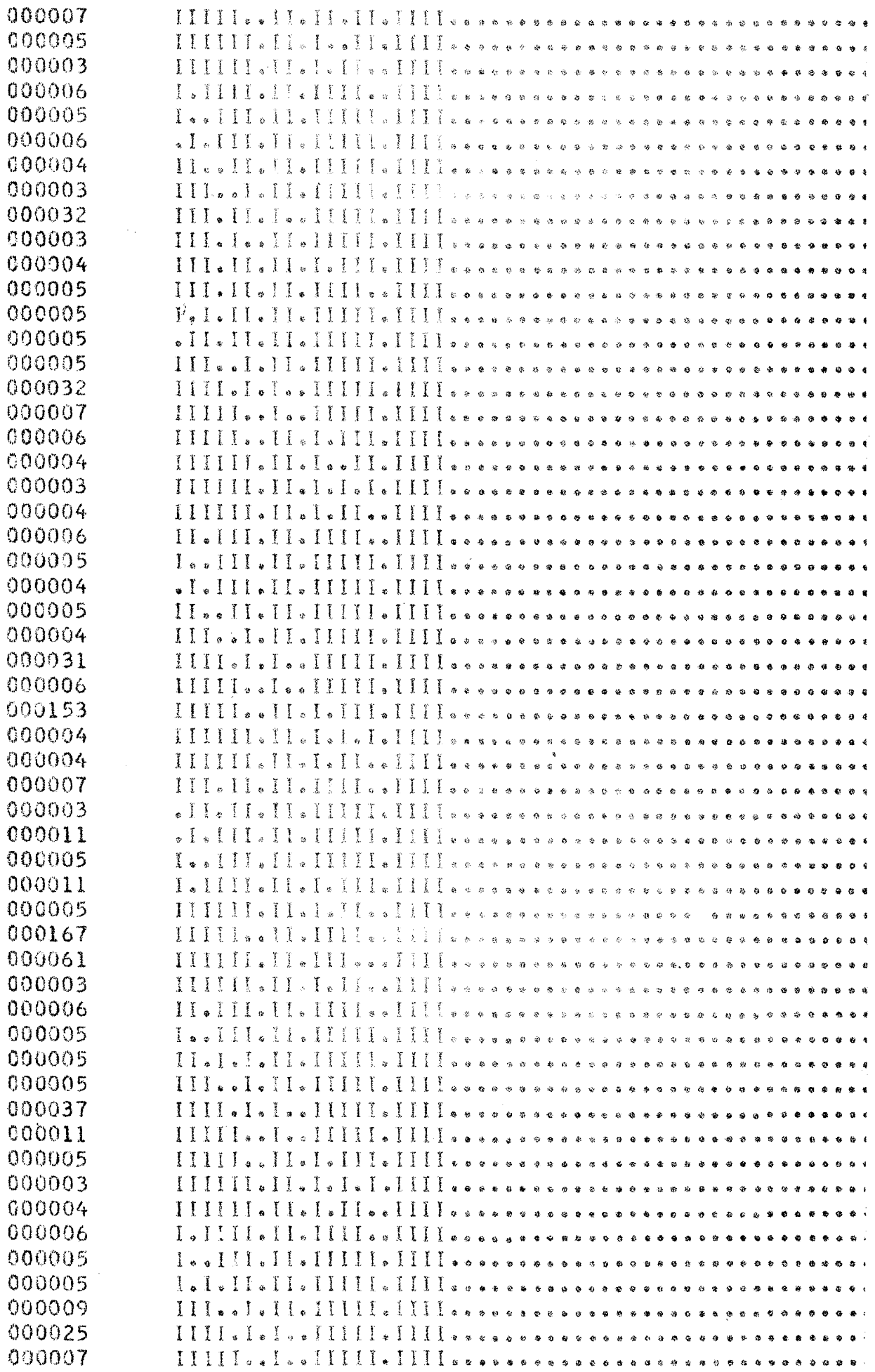
On trouvera sur les pages suivantes la distribution des pages en mémoire rapide pour le sous-ensemble d'algol (97 productions).
Les abscisses représentent les pages numérotées de 1 à 55. Chaque ligne indique qu'une page a été appelée. Les I correspondent aux pages en mémoire rapide, les . aux pages sur les disques. Les ordonnées représentent le temps (en 1/60 seconde) que la configuration précédente est restée en mémoire rapide.

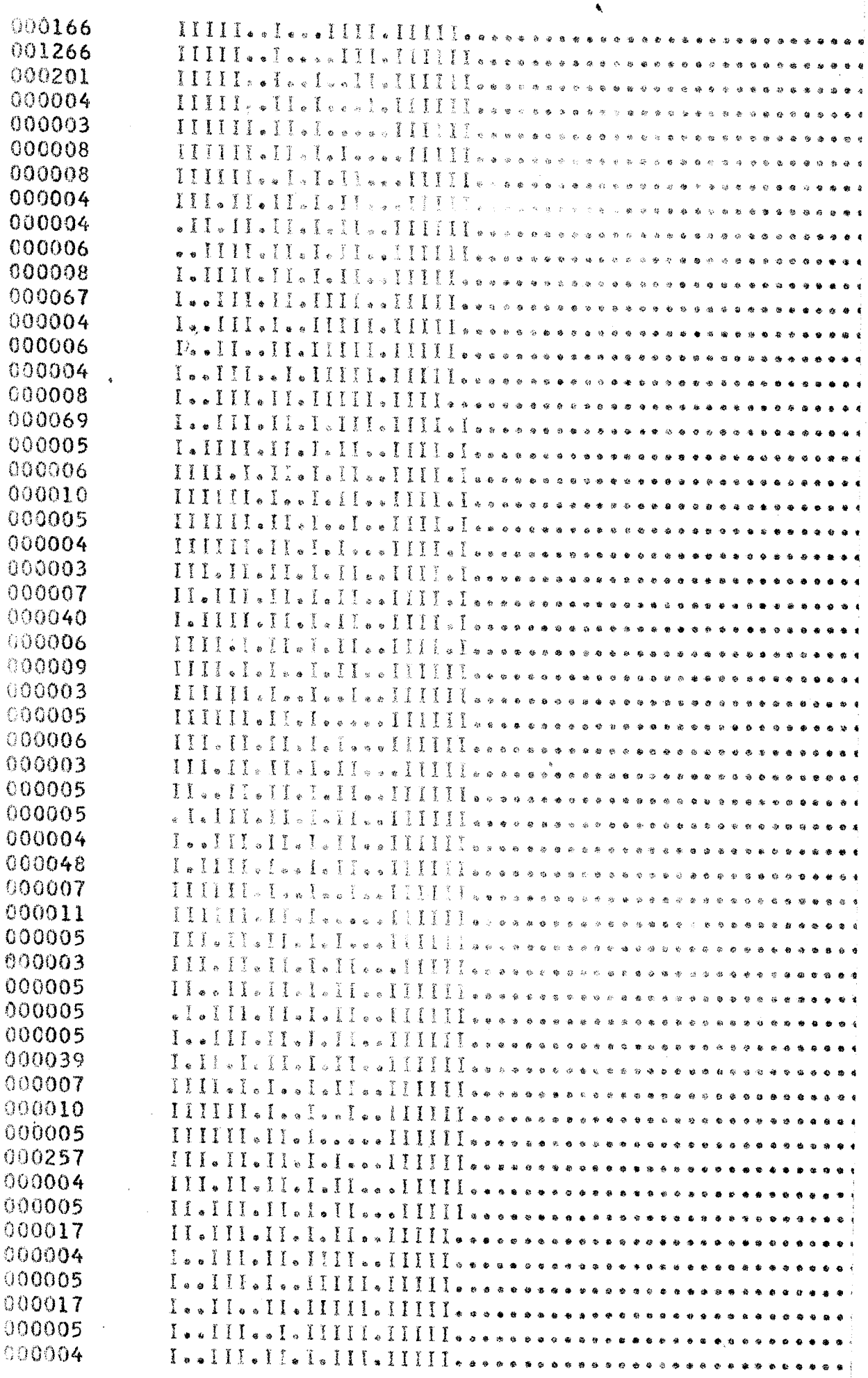
Les trois premières pages de résultats correspondent au début de l'exécution du programme, les trois suivantes correspondent à la fin de l'exécution du programme.

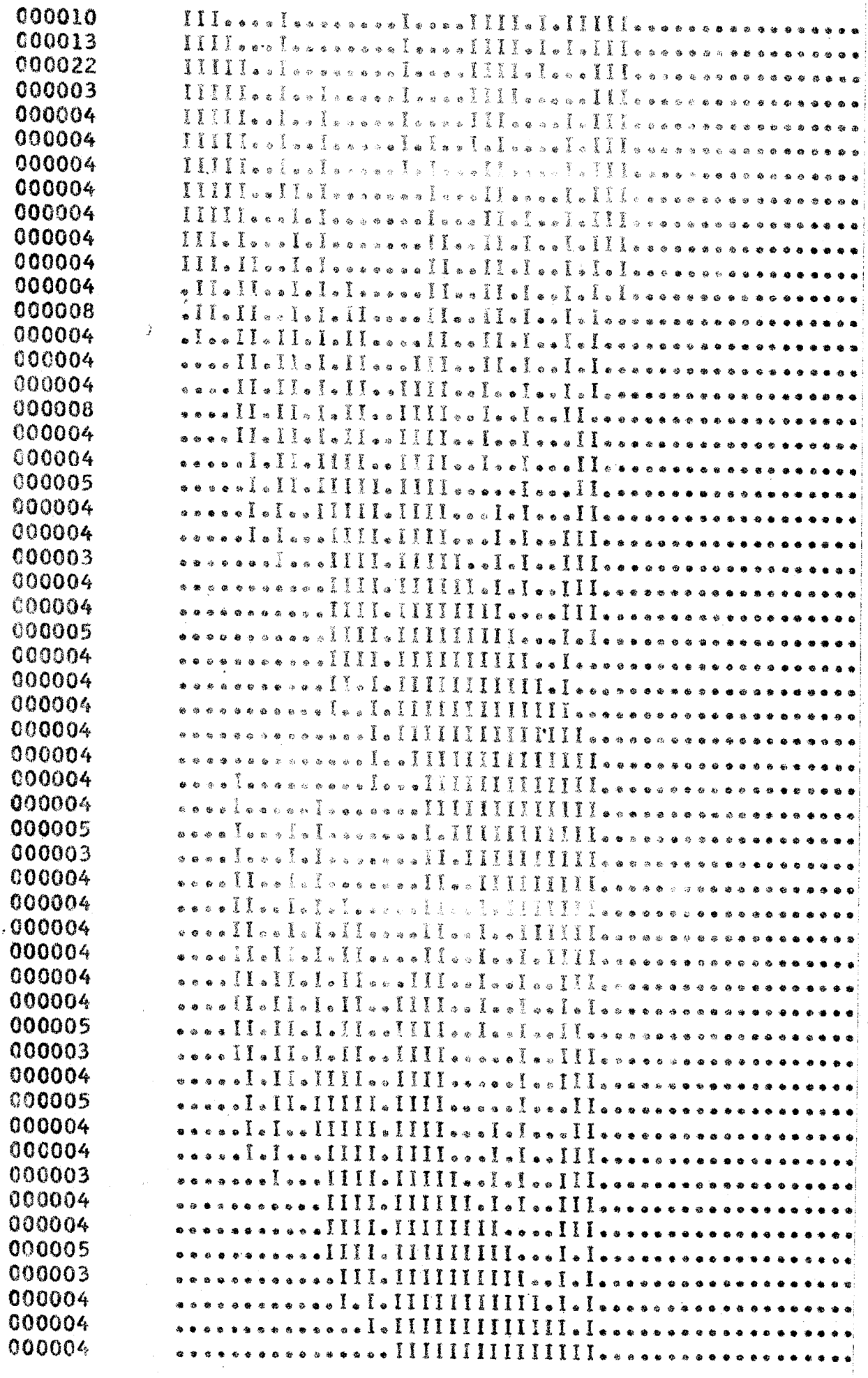
---

* en employant lisp-algol.

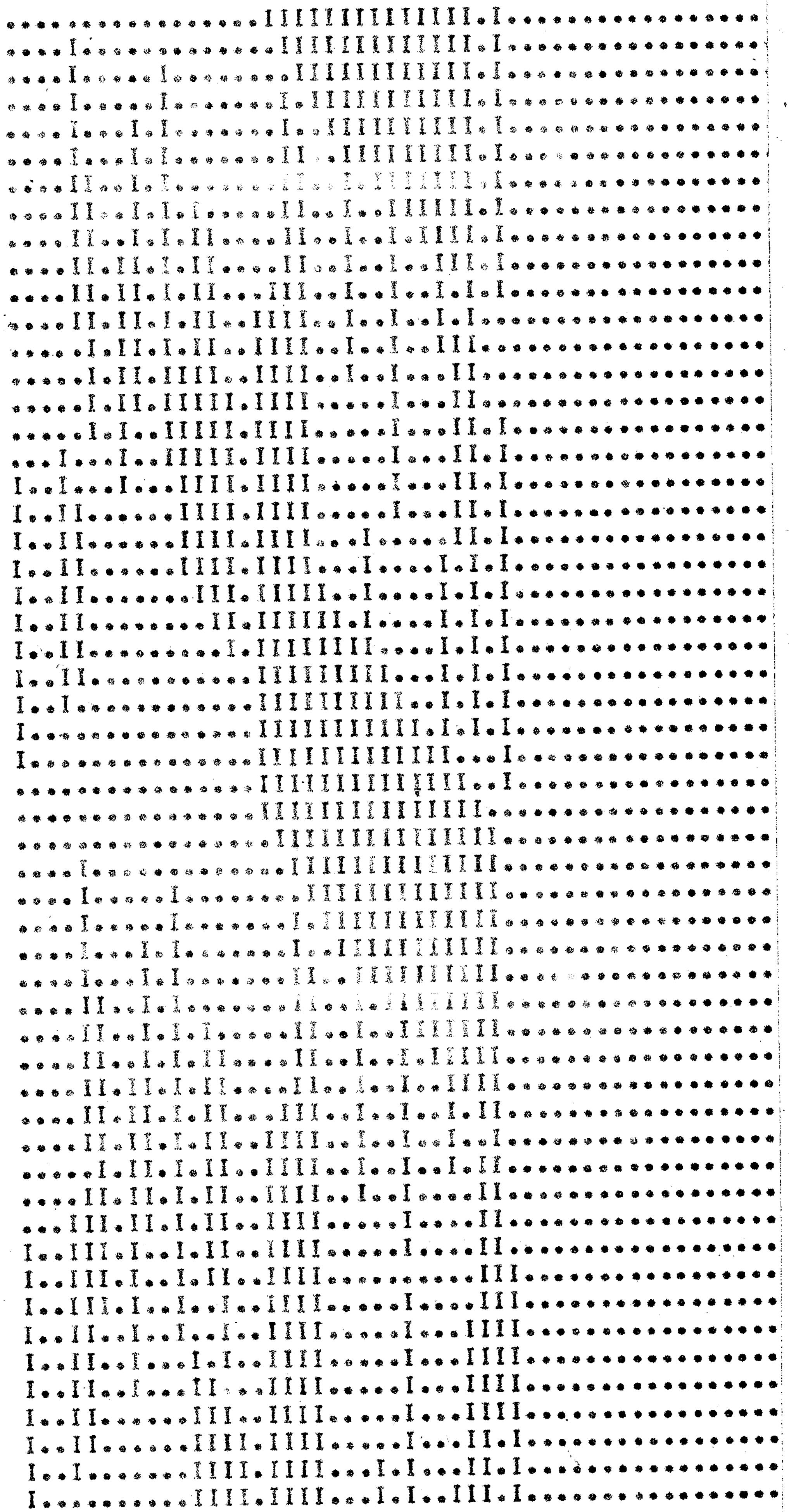

000418
000645
001171
001241
000788
001128
001043
001890
001339
001227
001132
002276
001369
001796
001061
002014
001650
001722
000827
000008
000318
000009
000008
000062
000589
000004
000020
000013
000193
000153
000032
000039
000003
000005
000006
000005
000010
000038
000008
000005
000005
000010
000039
000110
000007
000008
000004
000006
000005
000026
000003
000009
000004
000038
000005

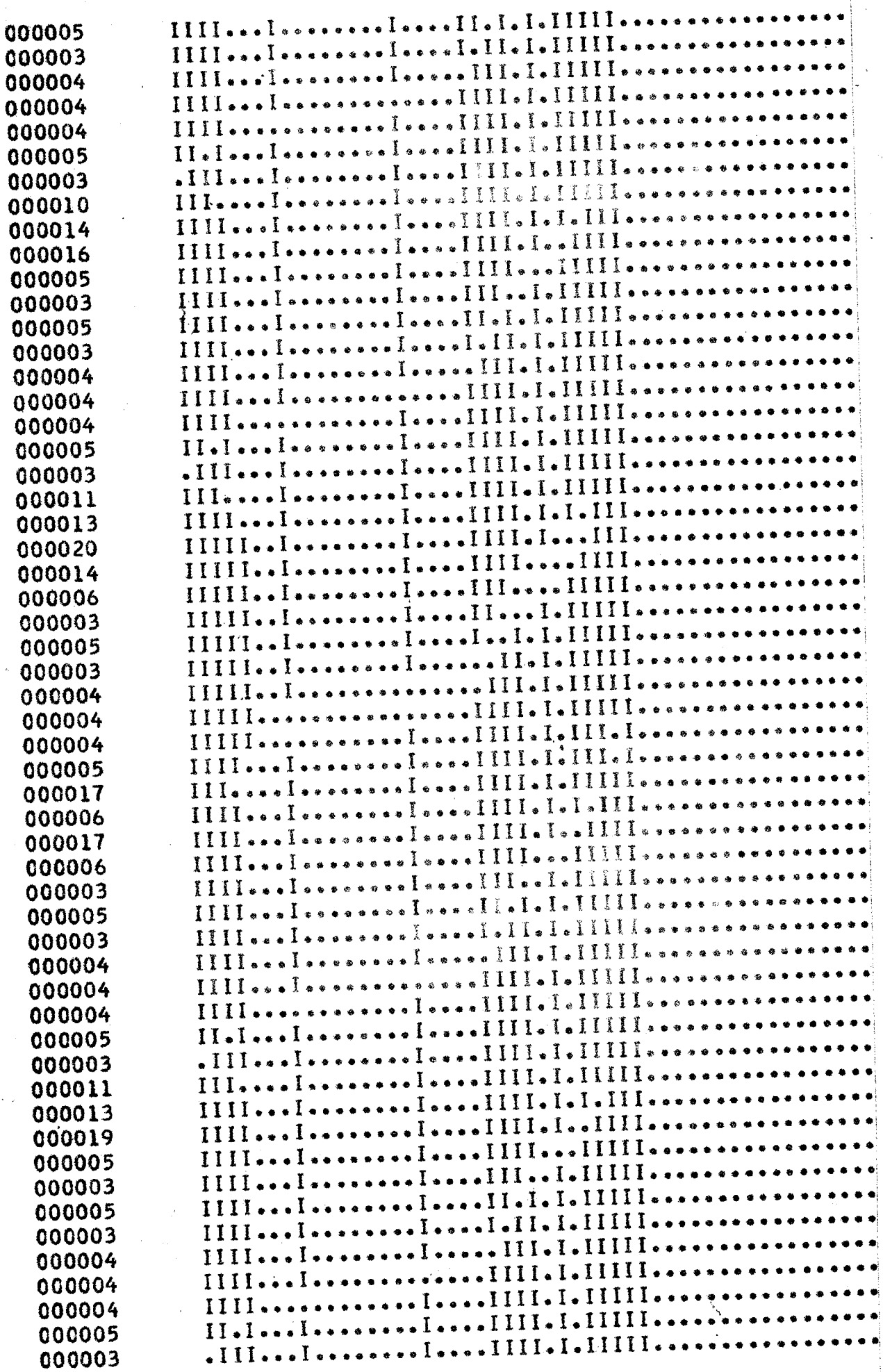






000005
000003
000004
000006
000004
000004
000004
000004
000004
000005
000003
000005
000004
000004
000004
000004
000005
000003
000008
000012
000005
000003
000004
000004
000005
000003
000004
000004
000004
000008
000004
000004
000004
000005
000003
000004
000004
000004
000004
000005
000003
000004
000005
000004
000003
000005
000083
000005
000004
000004
000004
000004
000005
000004
000004





CHAPITRE III

ANALYSEUR LISP-ALGOL

Nous avons vu que la table était constituée de cinq tableaux : ETA, TEZ et TEZP qui servent à diriger l'analyse, RTABLE qui est tel que RTABLE [I] - RTABLE [I-1] est égal au nombre d'états que l'état numéro I a effectivement crées, REP qui donne la liste des états qui avaient déjà été créés.

On range la chaîne à analyser dans PILE [1] , PILE [2] étant utilisé comme pile d'analyse. Au départ PILE [2] est égal à "0". Nous avons écrit quatre procédures Lisp-Algol : ANALYSE, ETASUIVANT , TRANSLATE et REDUIRE dont on trouvera un listage sur les pages suivantes.

Ce programme a été testé avec la table de la grammaire G1 de l'exemple théorique. Nous avons analysé plusieurs chaînes de longueur variable, ce qui nous a permis de vérifier que le temps d'analyse était proportionnel à la longueur de la chaîne.

Performance.

En une seconde, on n'analyse que 34 unités syntaxiques. Ce temps étant évidemment totalement incompatible avec une exploitation réelle de cette méthode d'analyse, nous avons, dans une seconde étape, réécrit l'analyseur en Algol, comme il est décrit au chapître V.

La table étant initialement sous forme de liste, nous avons été amenés à la transformer en une table numérique : c'est cette transformation qui est exposée au chapître suivant.

Après le programme, nous donnons le résultat de l'analyse d'une chaîne comprenant 369 unités syntaxiques.

```
DEBUT
    COMMENTAIRE :BLOC ZERO :DECLARATION DE L'ENTIER DIMENSION
    UTILISEE COMME BORNE SUPERIEURE DES TABLEAUX TCAR ET TCDR ;
    ENTIER DIMENSION;
    LIRE(DIMENSION);

    DEBUT
        COMMENTAIRE :BLOC UN :DECLARATION DES VARIABLES GLOBALES ET
        DES PROCEDURES ALGOL-LISP ;
        ENTIER TABLEAU TCAR,TCDR[1:DIMENSION],TE[1:11,1:4];
        ENTIER NIL,IPD,IPDMAX,LISTEDATOMES,BOITE;
        ENTIER N,I,S1,I2,TEMPS1,I1;
        BOOLEEN E;
        LIRE(N);
        LIRE(I2);
        DEBUT
            ENTIER TABLEAU L[0:N],ETA,REP,TEZ,TEZP[0:I2],
            RTABLE[-1:I2],PILE[1:2];


            BOOLEEN PROCEDURE APPARTENIR(X,Y);
            VALEUR X,Y;
            ENTIER X,Y;

            COMMENTAIRE :X EST UN ATOME , Y UNE LISTE NE POSSEDANT
            AUCUNE SOUS-LISTE . APPARTENIR PREND LA VALEUR VRAI SI X
            EST UN ELEMENT DE LA LISTE Y ;

            APPARTENIR:= SI Y=NIL ALORS FAUX
            SINON
            SI X=TCAR[Y] ALORS VRAI
            SINON APPARTENIR(X,TCDR[Y]);


            PROCEDURE ANALYSE(X,Y,Z) ;
            ENTIER X,Y,Z;

            COMMENTAIRE :LA LISTE Y CONTIENT LA CHAINE D'ENTREE , X
            SERT DE PILE DE MANOEUVRE , Z REPRESENTE L'ETAT
            CORRESPONDANT ;

            DEBUT
RETOURNE:       SI TCDR[Y]=NIL ET Z=1 ALORS
                DEBUT
                    TEMPS1:=TEMPS-TEMPS1;
                    ALALIGNE(1);
                    ECRIRECHAINE(1, "                    " );
                    ECRIRECHAINE(1,
                    " LA CHAINE PRECEDENTE EST CORRECTE " );
                    ECRIRECHAINE(1, " LE TEMPS D ANALYSE EST DE " );
                    ECRIREATOME(1,TCDR[ENTIERENLISTE(TEMPS1)]);
                    ECRIRECHAINE(1, " DIXIEMES DE SECONDE " );
```

```
        ALALIGNE(2);
        ECRIRECHAINE(1, "                          " );
        ECRIRECHAINE(1,
        " LE NOMBRE D UNITES SYNTAXIQUES EST DE " );
        ECRIREATOME(1,TCDR[ENTIERENLISTE(I1)]);
        ECRIRECHAINE(1, " DONC " );
        ECRIREATOME(1,TCDR[ENTIERENLISTE(CONVRE(I1*10/
        TEMPS1))]);
        ECRIRECHAINE(1, " UNITES SYNTAXIQUES PAR SECONDE "
        );
        SAUTPAGE;
    FIN
    SINON
    DEBUT
        SI TEZ[Z] ≠ NIL ALORS TRANSLATE(X,Y,Z)
        SINON
        SI TEZP[Z] ≠ NIL ALORS X:=REDUIRE(X,TCAR[Y],
        TEZP[Z]);
        SI X=NIL ALORS
        DEBUT
            ECRIRELISTE(1,Y);
            ECRIRE( " CHAINE INCORREC " , " TE " );
        FIN
        SINON
        DEBUT
            Z:=NUMERIQUE(TCAR[TCDR[X]]);
            Z:=ETASUIVANT(TCAR[X],Z);
            X:=CONS(ENTIERENLISTE(Z),X);
            ALLERA  RETOURNE,
        FIN ;
    FIN ;
FIN ANALYSE ;


ENTIER PROCEDURE ETASUIVANT(X,Z) ;
VALEUR X,Z;
ENTIER X,Z;
DEBUT
    ENTIER I,J;
    E:= VRAI ;
    I:=RTABLE[Z-1];
    POUR I:=I+1 TANTQUE  I < RTABLE[Z] ET E FAIRE
    SI X=ETA[I] ALORS
    DEBUT
        ETASUIVANT:=I;
        E:= FAUX ;
    FIN ;
    SI E ALORS
    POUR I:=REP[Z],TCDR[I] TANTQUE  I ≠ NIL ET E
    FAIRE
    DEBUT
        J:=NUMERIQUE(TCAR[I]);
        SI X=ETA[J] ALORS
        DEBUT
            E:= FAUX ;
```

```
                ETASUIVANT:=J;
            FIN ;
        FIN ;
    FIN ETASUIVANT ;


    PROCEDURE TRANSLATE(X,Y,Z) ;
    ENTIER X,Y,Z;
    DEBUT
        ENTIER T;
        T:=TCAR[Y];
        SI APPARTENIR(T,TEZ[Z]) ALORS
        DEBUT
            X:=CONS(T,X);
            Y:=TCDR[Y];
        FIN
        SINON X:=
        SI TEZP[Z]=NIL ALORS NIL
        SINON REDUIRE(X,T,TEZP[Z]);
    FIN TRANSLATE ;


    ENTIER PROCEDURE REDUIRE(X,Y,T) ;
    VALEUR X,Y,T;
    ENTIER X,Y,T;
    DEBUT
        ENTIER U,V,I;
        U:=TCAR[T];
        SI APPARTENIR(Y,TCDR[U]) ALORS
        DEBUT
            U:=TCAR[U];
            U:=NUMERIQUE(U);
            V:=X;
            POUR I:=1 PAS 1 JUSQUA 2*(NOMBRE(L[U])-1) FAIRE
            V:=TCDR[V];
            V:=CONS(TCAR[L[U]],V);
            REDUIRE:=V;
        FIN
        SINON REDUIRE:=
        SI TCDR[T]=NIL ALORS NIL
        SINON REDUIRE(X,Y,TCDR[T]);
    FIN REDUIRE ;


MAINPROGRAM:

    PREPARER;

        COMMENTAIRE :LECTURE DE LA GRAMMAIRE ;
        POUR I:=0 PAS 1 JUSQUA N FAIRE
        NOMMER(2, "  " ,L[I]);

        COMMENTAIRE :LECTURE DE LA TABLE ;
        POUR I:=0 PAS 1 JUSQUA 12 FAIRE
        DEBUT
```

```
            NOMMER(2, "  " ,TEZP[I]);
            NOMMER(2, "  " ,TEZ[I]);
            NOMMER(2, "  " ,ETA[I]);
            NOMMER(2, "  " ,REP[I]);
            NOMMER(2, "  " ,RTABLE[I]);
            RTABLE[I]:=NUMERIQUE(RTABLE[I]);
        FIN ;

        NOMMER(0, " ( 0 ) " ,PILE[2]);

        COMMENTAIRE :LECTURE DE LA CHAINE A ANALYSER ;
        NOMMER(2, "  " ,PILE[1]);

        ALALIGNE(5);
        ECRIRECHAINE(1, "                                    " );
        ECRIRECHAINE(1, "                                    " );
        ECRIRECHAINE(1, " CHAINE ANALYSEE " );
        ALALIGNE(2);
        ECRIRELISTE(1,PILE[1]);
        RTABLE[-1]:=0;
        I1:=0;
        SI:=0;

        COMMENTAIRE :I1 :NOMBRE D'UNITES SYNTAXIQUES ;
        POUR I:=PILE[1],TCDR[I] TANTQUE  I ≠ NIL FAIRE
        I1:=I1+1;

        COMMENTAIRE PILE[2] :PILE DE MANOEUVRE , PILE[1]
        :CHAINE D'ENTREE , SI :ETAT CORRESPONDANT ;

        TEMPS1:=TEMPS;
        ANALYSE(PILE[2],PILE[1],SI);

    FIN ;
FIN BLOC UN ;
FIN PROGRAMME LISP ;
```

CHAPITRE IV

TRANSFORMATION DE LA TABLE

IV - 1 Ordre particulier pour la consultation de la table.

Nous avons remarqué qu'après une translation, l'état suivant que nous allons chercher est toujours caractérisé par un élément du vocabulaire terminal, en effet, on translate toujours un élément de V_T et c'est en fonction de l'état dans lequel on se trouve et de cet élément que l'on détermine l'état suivant. Par contre après une réduction, l'état suivant est toujours caractérisé par un élément de V_N.

Il s'en suit que pour un état donné S_n, on a avantage à faire une partition de $\mathcal{R}(S_n)$ en deux sous-ensembles $\mathcal{R}_1(S_n)$ et $\mathcal{R}_2(S_n)$ tels que :

1°) $S_i \in \mathcal{R}_1(S_n)$ si et seulement si $S_i^* \in \mathcal{R}(S_n)$ avec $X_{p_i} \in V_N$

2°) $S_j \in \mathcal{R}_2(S_n)$ si et seulement si $S_j \in \mathcal{R}(S_n)$ avec $X_{p_j} \in V_T$

Ainsi, suivant que l'on vient d'effectuer une translation ou une réduction on cherchera l'état suivant soit dans $\mathcal{R}_1(S_n)$ soit dans $\mathcal{R}_2(S_n)$.

IV - 2 Réalisation pratique.

Avec un état S_n, on sait qu'on a créé RTABLE [n] - RTABLE [n-1] états numérotés de RTABLE [n-1] + 1 à RTABLE [n].

---

* $S_i = (X_{p_i} (p_1\ i_1\ \alpha_2)\ (p_2\ i_2\ \alpha_2) \ \ldots\ (p_n\ i_n\ \alpha_n))$ est maintenant égal à X_{p_i} (II - 4)

Pour chaque n, on examine tous les états ayant un numéro compris entre ces deux bornes. On construit deux listes Y et P. Dans la première, on rangera les numéros de tous les états caractérisés par un élément de V_T, dans la deuxième les autres. On agit de même pour la liste des états déjà créés : REP [n] .
Ensuite on fait l'union des deux listes Y et P de telle sorte qu'en tête, on ait les numéros des états caractérisés par un élément de V_N, en queue les autres. Le résultat de cette union est une liste dont le pointeur sera égal à REP[n] .

IV - 3 Transformation de la table.

Pour l'analyse, on essayera en premier de réduire. Si cela est impossible on translatera l'unité syntaxique suivante sans en vérifier la validité, cette vérification étant faite de toute façon lors de la recherche de l'état suivant.

On applique l'algorithme suivant, qui rend la connaissance complète de la liste TEZ inutile : Supposons que nous soyons dans l'état numéro I.

α) TEZP [I] est vide : on translate sans vérifier le contexte.

β) TEZP [I] n'est pas vide, elle contient n sous-listes, deux cas peuvent se présenter :

i) TEZ [I] est vide (si la chaîne analysée est correcte, ceci implique qu'il y a nécessairement une réduction à faire).
On ne consultera au plus que les contextes des n-1 premières sous-listes de TEZP [I]
Si l'un de ces contextes est applicable, on effectue la réduction correspondante
sinon on effectue la réduction correspondant à la $n^{ième}$ sous-liste sans vérifier le contexte

ii) TEZ [I] n'est pas vide : on consultera le contexte des n sous-liste de TEZP [I] . Si l'un de ces contextes est applicable on effectue la réduction correspondante sinon on effectue une translation sans vérifier le contexte.

Nous avons donc effectué la transformation des informations contenues dans REP et TEZP de la manière suivante :

IV - 3 - 1 Transformation de REP.

Nous savons maintenant que dans REP [I] il y a tous les numéros des états suivants. Nous avons remplacé le tableau REP par deux tableaux RE et RET tels que :

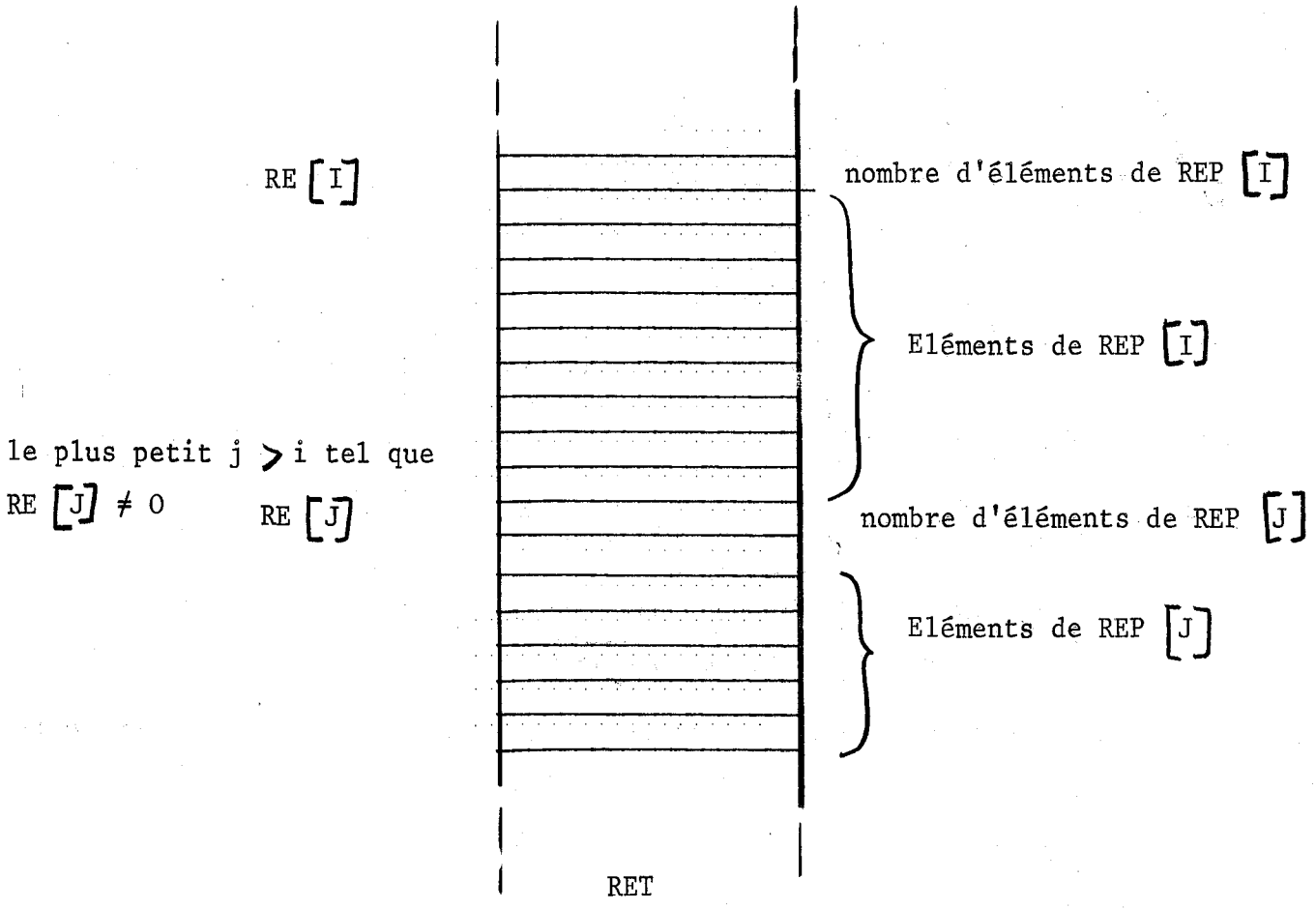
RE [I] = 0 ⇔ REP [I] pointe sur une liste vide

RE [I] = k ⇔ REP [I] pointe sur une liste non vide

ce k étant tel qu'en RET [K] on ait le nombre d'éléments de la liste REP [I] et que de RET [K+1] à RET [K+RET [K]] on trouve les éléments de cette liste (cette liste ne contient que des entiers : numéro d'états).

I varie de 0 à I2 (nombre d'états).

k part de 1. La prochaine valeur de k étant K+RET [K] +1 et ainsi de suite.

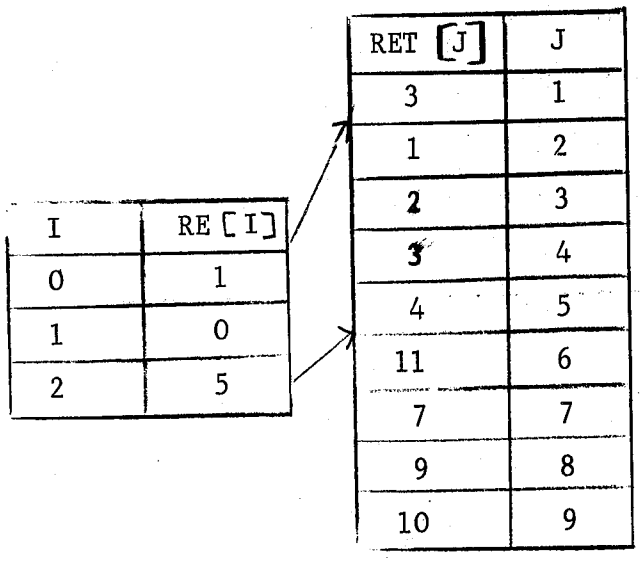


Exemple.

Supposons que

REP [0] = (1 2 3) REP [1] = ( )

REP [2] = (11 7 9 10)



I	RE [I]
0	1
1	0
2	5

RET [J]	J
3	1
1	2
2	3
3	4
4	5
11	6
7	7
9	8
10	9

IV - 3 - 2 Transformation de TEZP.

Pour remplacer TEZP, nous avons construit deux tableaux ZP et ZZP tels que :

ZP [I] = 0 $\Leftrightarrow$ TEZP [I] pointe sur une liste vide

ZP [I] = k $\Leftrightarrow$ TEZP [I] pointe sur une liste non vide

ce k pointe dans le tableau ZZP sur les informations équivalentes (compte tenu de la méthode d'analyse) aux informations de la liste TEZP [I].

Ces informations sont les suivantes :

α) ZZP [K] = n : nombre de sous-listes de TEZP [I]

β) A partir de ZZP [K+1] on a successivement pour chacune des n-1 premières sous-listes de TEZP [I] les informations suivantes, où j et m_j représentent respectivement le rang de la sous-liste, son nombre d'éléments :

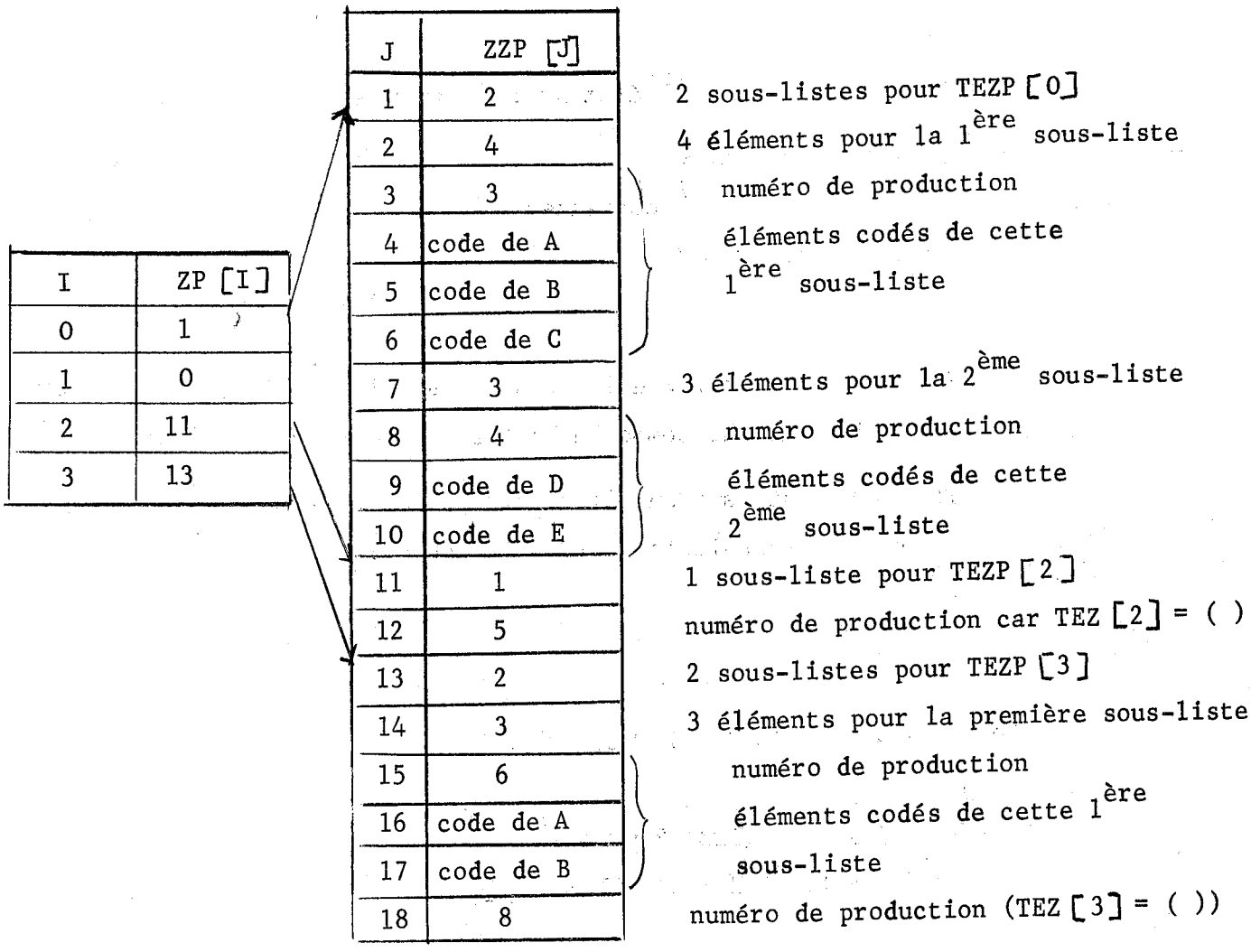
$$\text{sous-liste } j : \left\{ \begin{array}{ll} m_j & \\ P_j & 1^{er} \text{ élément : n° de production} \\ \text{code } 2^{ème} \text{ élément} & \\ \vdots & \text{contextes} \\ \text{code } m_j \text{ élément} & \end{array} \right\} m_j \text{ éléments de la sous-liste}$$

γ) - si TEZ [I] pointe sur une liste non vide, la $n^{ième}$ et dernière sous-liste de TEZP [I] est rangée dans ZZP comme les autres sous-listes.

- si TEZ [I] pointe sur une liste vide, seul le numéro de production correspondant à la dernière sous-liste de TEZ [I] est rangée dans ZZP.

<u>Exemple</u>.

TEZ [0] = ((3 A B C) (4 D E)), TEZ [0] ≠ ( )
TEZP [1] = ( )
TEZP [2] = ((5 A F)), (TEZ [2] = ( )
TEZP [3] = ((6 A B) (8 H I J)), TEZ [3] = ( )



IV - 3 - 3 Définitions de LPRIME [I, 1] et de LPRIME [I, 2]

Nous avons défini un tableau LPRIME à deux dimensions tel que LPRIME [I, 1] soit égal au code de la partie gauche de la production numéro I, LPRIME [I, 2] soit égal à $2 \times r_I - 1$, où r_I est le nombre d'éléments de la partie droite de la production numéro I. Ce tableau est utilisé lors d'une réduction.

Pour TEZ, il suffit de savoir si la liste est vide ou non. Il suffit de remarquer que TEZ [I] pointe sur une liste vide $\Leftrightarrow$ REP [I] point sur une liste vide $\Leftrightarrow$ RE [I] = 0. Le tableau TEZ ne nous sert donc plus à rien.

Comme codage, nous avons pris l'adresse des atomes du Lisp-Algol. De ce fait, le tableau ETA est inchangé.

On trouvera sur les pages suivantes un listage de ce programme.

IV - 4 Programme.

```
      DEBUT
COMMENTAIRE :BLOC ZERO :DECLARATION DE L'ENTIER DIMENSION
UTILISEE COMME BORNE SUPERIEURE DES TABLEAUX TCAR ET TCDR ;
ENTIER DIMENSION;
LIRE(DIMENSION);

DEBUT
    COMMENTAIRE :BLOC UN :DECLARATION DES VARIABLES GLOBALES ET
    DES PROCEDURES ALGOL-LISP ;
    ENTIER TABLEAU TCAR,TCDR[1:DIMENSION],TE[1:11,1:4];
    ENTIER NIL,IPD,IPDMAX,LISTEDATOMES,BOITE;
    ENTIER I,N,I2,T,K,P,Y,J,I3,M;
    LIRE(N);
    LIRE(I2);
    DEBUT
        ENTIER TABLEAU L[0:N],ETA,Z,ZP,RE,TEZ,TEZP,REP[0:I2],
        RTABLE[-1:I2],LPRIME[0:N,1:2],ZZP,RET[1:800];

MAINPROGRAM:

        PREPARER;

        COMMENTAIRE :LECTURE DE LA GRAMMAIRE ;
        POUR I:=0 PAS 1 JUSQUA N FAIRE
        NOMMER(2, "  " ,L[I]);

        COMMENTAIRE :LECTURE DE LA TABLE LISP-ALGOL ;
        POUR I:=0 PAS 1 JUSQUA I2 FAIRE
        DEBUT
            NOMMER(2, "  " ,TEZP[I]);
            NOMMER(2, "  " ,TEZ[I]);
            NOMMER(2, "  " ,ETA[I]);
            NOMMER(2, " " ,REP[I]);
            NOMMER(2, "  " ,RTABLE[I]);
        FIN ;

        COMMENTAIRE :ORDONNANCEMENT DE LA TABLE ;
        POUR I:=0 PAS 1 JUSQUA I2 FAIRE
        DEBUT
            Z[I]:=I;
            RTABLE[I]:=NUMERIQUE(RTABLE[I]);
        FIN ;
        POUR I:=0 PAS 1 JUSQUA I2 FAIRE
        DEBUT
            T:=RTABLE[I];
            K:=RTABLE[I-1]+1;
            P:=NIL;
            Y:=NIL;
            POUR J:=K PAS 1 JUSQUA T FAIRE
            SI TERMINAL(ETA[J]) ALORS Y:=CONS(ENTIERENLISTE(J),
            Y)
            SINON P:=CONS(ENTIERENLISTE(J),P);
            SI REP[I] ≠ NIL ALORS
            POUR T:=REP[I],TCDR[T] TANTQUE  T ≠ NIL FAIRE
            SI TERMINAL(ETA[NUMERIQUE(TCAR[T])]) ALORS Y:=
```

```
                CONS(TCAR[T],Y)
        SINON P:=CONS(TCAR[T],P);
        SI Y=NIL ALORS REP[I]:=P
        SINON
        SI P=NIL ALORS REP[I]:=Y
        SINON
        DEBUT
            POUR T:=P,TCDR[T] TANTQUE  T ≠ NIL FAIRE
            Y:=CONS(TCAR[T],Y);
            REP[I]:=Y;
        FIN ;
    FIN ;

    COMMENTAIRE :TRANSFORMATION DE LA TABLE ;
    POUR I:=0 PAS 1 JUSQUA N FAIRE
    DEBUT
        COMMENTAIRE :LPRIME[I,1] EST EGAL AU CODE DE LA
        PARTIE GAUCHE DE LA REGLE NUMERO I , LPRIME[I,2] EST
        EGAL A 2 * ( NOMBRE D'ELEMENTS DE LA PARTIE DROITE ) -
        1 ;
        LPRIME[I,1]:=TCAR[L[I]];
        LPRIME[I,2]:=2*(NOMBRE(L[I])-1)-1;
    FIN ;
    T:=1;
    Y:=K:=1;
    POUR I:=0 PAS 1 JUSQUA I2 FAIRE
    DEBUT

        COMMENTAIRE :TRANSFORMATION DE LA LISTE CONTENANT LE
        NUMERO DES ETATS ;
        SI REP[I]=NIL ALORS
        DEBUT
            RE[I]:=0;
            ALLERA  FINIREP;
        FIN ;
        RE[I]:=T;
        RET[T]:=NOMBRE(REP[I]);
        J:=T;
        POUR J:=J+1 TANTQUE  REP[I] ≠ NIL FAIRE
        DEBUT
            RET[J]:=NUMERIQUE(TCAR[REP[I]]);
            REP[I]:=TCDR[REP[I]];
            T:=J+1;
        FIN ;
FINIREP:

        COMMENTAIRE :TRANSFORMATION DE L'ENSEMBLE ZP ;
        SI TEZP[I]=NIL ALORS
        DEBUT
            ZP[I]:=0;
            ALLERA  TERMINE;
        FIN ;
        ZP[I]:=Y;
        I3:=ZZP[Y]:=NOMBRE(TEZP[I]);
        Y:=Y+1;
```

```
                    POUR J:=1 PAS 1 JUSQUA 13 FAIRE
                DEBUT
                    SI J=13 ET RE[I]=0 ALORS
                    DEBUT
                        COMMENTAIRE :DERNIERE SOUS-LISTE DE TEZP[I]
                        ET TEZ[I] POINTE SUR UNE LISTE VIDE ;
                        ZZP[Y]:=
                        NUMERIQUE(TCAR[TCAR[TEZP[I]]]);
                        Y:=Y+1;
                        ALLERA  TERMINE
                    FIN ;
                    P:=TCAR[TEZP[I]];
                    TEZP[I]:=TCDR[TEZP[I]];
                    ZZP[Y]:=NOMBRE(P);
                    M:=Y+1;
                    ZZP[M]:=NUMERIQUE(TCAR[P]);
                    P:=TCDR[P];
                    POUR M:=M+1 TANTQUE  P ≠ NIL FAIRE
                    DEBUT
                        ZZP[M]:=TCAR[P];
                        P:=TCDR[P];
                        Y:=M+1;
                    FIN
                FIN ;
TERMINE:    FIN ;

            I2:=I2+1;
            N:=2*(N+1);
            Y:=Y-1;
            T:=T-1;

            COMMENTAIRE :PERFORATION DE LA TABLE-ALGOL ;
            SORTIE(7,M2,1,I2,1,N,1,Y,1,T);
            SORTIE(7,M1,I2,ZP[0],I2,ETA[0],I2,RE[0],N,LPRIME[0,
            1],Y,ZZP[1],T,RET[1]);
M1:         MODELE( " (1X,14(I4,1X)) " );
M2:         MODELE( " ( 1X, 7HCOURTIN,4X,4(I3,2X)) " );




        FIN ;
    FIN BLOC UN ;
FIN PROGRAMME LISP ;
```

CHAPITRE V

ANALYSEUR ALGOL

A partir de la nouvelle table, nous avons écrit un analyseur en Algol. Nous avons simulé les piles d'entrée et d'analyse à l'aide de deux tableaux PILE1 et PILE2.
- Si ZP [I] est égal à zéro, on translate l'unité syntaxique suivante sans tester sa validité.
- Si ZP [I] est différent de zéro on va essayer de réduire. Le programme ira à SPECIALREDUI lorsque, RE [I] étant nul, on arrive à la consultation de la dernère sous-liste de ZP [I].

Après une translation ou après une réduction on ira à ETASUIVANT. Nous avons introduit un booléen "VT" qui a pris la valeur vrai si on vient d'effectuer une translation, faux si on vient d'effectuer une réduction. Ainsi on effectue la recherche de l'état suivant dans les meilleures conditions.

On peut aussi si on le désire obtenir la suite des règles de réécriture utilisées.
En effet, le programme admet deux sortes de données.
En premier un booléen "arbre" qui prend la valeur vrai si l'on désire obtenir cette suite, faux autrement. En second une chaîne à analyser.

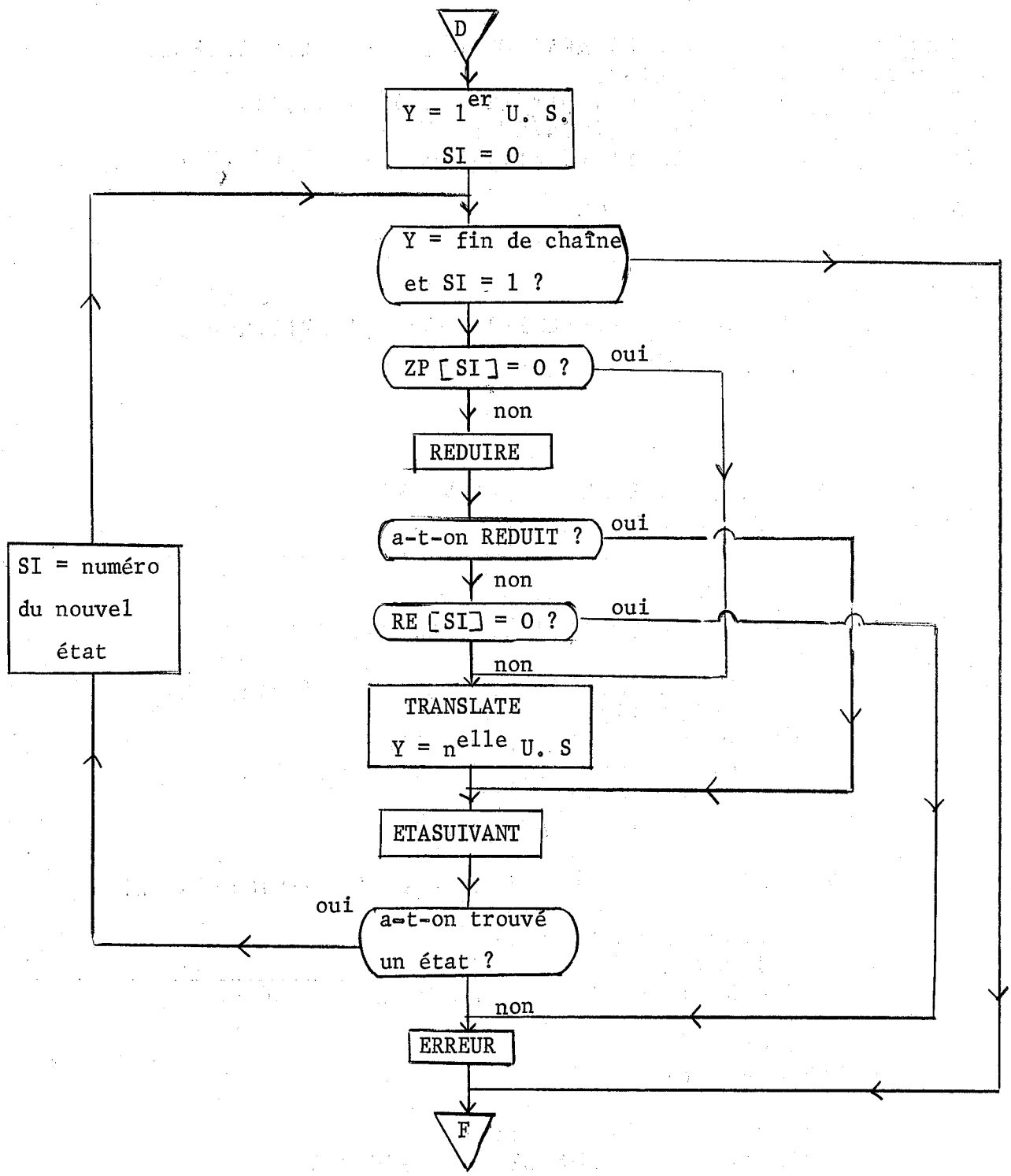
Résultats.

Dans un premier temps, nous avions effectué la transformation de la table sans tenir compte des cas particuliers de TEZ, de TEZP et des états (IV - 3).
On analysait en testant toujours les unités syntaxiques et la recherche des états n'était pas optimum (c'est-à-dire que l'on ne tenait pas compte si l'on venait de réduire ou de progresser). Nous avions quand même obtenu une rapidité sept fois plus grande que pour l'analyseur Lisp-Algol.

Actuellement, pour la même chaîne que précedemment (III) nous analysons 1230 unités syntaxiques par seconde en employant le compilateur optimisé de J.C. BOUSSARD.

On trouvera sur les pages suivantes un listage du programme ainsi que l'analyse de plusieurs chaînes.

Organigramme de la méthode d'analyse.



```
DEBUT
    COMMENTAIRE :BLOC ZERO :DECLARATION DE L'ENTIER DIMENSION
    UTILISEE COMME BORNE SUPERIEURE DES TABLEAUX TCAR ET TCDR ;
    ENTIER DIMENSION;
    LIRE(DIMENSION);

    DEBUT
        COMMENTAIRE :BLOC UN :DECLARATION DES VARIABLES GLOBALES ET
        DES PROCEDURES ALGOL-LISP ;
        ENTIER TABLEAU TCAR,TCDR[1:DIMENSION],TE[1:11,1:4];
        ENTIER NIL,IPD,IPDMAX,LISTEDATOMES,BOITE;
        ENTIER I,N,I2,Y,T,PILE,Y1,Y2,SI,TEMPS1,Z1,ZP1,K,P,M,R1,R2,PAS,
        I1;
        BOOLEEN ARBRE,E,VT;
        LIRE(N);

        DEBUT
            ENTIER TABLEAU L[0:N],PILE1[1:1000],PILE2[1:100];

MAINPROGRAM:

            PREPARER;

            COMMENTAIRE :LECTURE DE LA GRAMMAIRE ;
            POUR I:=0 PAS 1 JUSQUA N FAIRE
            NOMMER(2, "  " ,L[I]);

            ENTREE(5,M2,1,I2,1,N,1,Y,1,T);
M2:         MODELE( " ( 1X, 7HCOURTIN,4X,4(I3,2X)) " );
            DEBUT
                ENTIER TABLEAU ZP,ETA,RE[0:I2],LPRIME[0:N,1:2],
                ZZP[1:Y],RET[1:T];
                COMMENTAIRE :LECTURE DE LA TABLE ;
                ENTREE(5,M1,I2,ZP[0],I2,ETA[0],I2,RE[0],N,
                LPRIME[0,1],Y,ZZP[1],T,RET[1]);
M1:             MODELE( " (1X,14(I4,1X)) " );

                COMMENTAIRE :T EST EGALE AU CODE DU DELIMITEUR DE
                CHAINE ;
                T:=TCAR[TCDR[TCDR[L[0]]]];
                ARBRE:=BDONNEE;
                COMMENTAIRE :LECTURE DE LA CHAINE A ANALYSER ;
                NOMMER(2, "  " ,PILE);
                ALALIGNE(5);                                          "
                ECRIRECHAINE(1,"
                );
                ECRIRECHAINE(1, "    " );
                ECRIRECHAINE(1, " CHAINE ANALYSEE " );
                ALALIGNE(1);
                ECRIRELISTE(1,PILE);
                ALALIGNE(2);
                I:=0;
                COMMENTAIRE :CODAGE DE LA CHAINE ;
                POUR I:=I+1 TANTQUE  PILE ≠ NIL FAIRE
                DEBUT
```

```
        PILE1[I]:=TCAR[PILE];
        PILE:=TCDR[PILE];
    FIN ;
    SI:=PILE2[1]:=0;
    I1:=I2:=1;

    COMMENTAIRE :LE TABLEAU PILE1 CONTIENT LA CHAINE
    CODEE ,PILE1[I] CONTIENT LE CODE DE LA IEME UNITE
    SYNTAXIQUE. ON UTILISE PILE2 COMME PILE DE MANOEUVRE.
    LPRIME[P,1] CONTIENT LE CODE DE LA PARTIE GAUCHE DE
    LA REGLE NUMERO P , LPRIME[P,2] ETANT EGAL A ( 2
    FOIS LA LONGUEUR DE LA PARTIE DROITE DE LA REGLE P ) -
    1 ;

    TEMPS1:=TEMPS;
    Y1:=PILE1[I1];

ANALYSE:

    SI Y1=T ET SI=1 ALORS
    DEBUT
        TEMPS1:=TEMPS-TEMPS1;
        COMMENTAIRE :ON A TROUVE LE DELIMITEUR DE FIN DE
        CHAINE ET L'ETAT DANS LEQUEL ON SE TROUVE EST EGAL
        A 1 , L'ANALYSE EST DONC TERMINEE ET LA CHAINE EST
        CORRECTE ;
        ECRIRECHAINE(1, "                                " );
        ECRIRECHAINE(1, " LA CHAINE PRECEDENTE " );
        ECRIRECHAINE(1,
        " EST CORRECTE , LE TEMPS D ANALYSE EST DE " );
        ECRIREATOME(1,CDR(ENTIERENLISTE(TEMPS1)));
        ECRIRECHAINE(1, " DIXIEMES DE SECONDE " );
        ALALIGNE(1);
        ECRIRECHAINE(1, "                                " );
        COMMENTAIRE :I1 :NOMBRE D'UNITES SYNTAXIQUES ;
        ECRIRECHAINE(1, " NOMBRE D UNITES SYNTAXIQUES " );
        ECRIREATOME(1,CDR(ENTIERENLISTE(I1)));
        ECRIRECHAINE(1, " DONC " );
        ECRIREATOME(1,CDR(ENTIERENLISTE(CONVRE(I1*10/
        TEMPS1))));
        ECRIRECHAINE(1, " UNITES SYNTAXIQUES PAR SECONDE "
        );
    FIN
    SINON
    DEBUT

        COMMENTAIRE :SI ZP N'EST PAS VIDE , ON VA ESSAYER
        DE REDUIRE . SI ON NE PEUT PAS REDUIRE , ON
        TRANSLATE SANS TESTER L'UNITE SYNTAXIQUE ;

        Z1:=RE[SI];
        ZP1:=ZP[SI];
        SI ZP1=0 ALORS ALLERA  TRANSLATE;

REDUIRE:
```

```
                    COMMENTAIRE :SI LA LISTE ZP NE CONTIENT QU'UNE
                    SOUS-LISTE ON REMPLACE LA PARTIE DROITE DE LA
                    REGLE P PAR LA PARTIE GAUCHE SINON ON EST OBLIGE
                    DE TESTER L'UNITE SYNTAXIQUE SUIVANTE POUR
                    DETERMINER LA VALEUR DE P ;

                    K:=ZZP[ZP1];
                    POUR I:=1 PAS 1 JUSQUA K FAIRE
                    DEBUT
                        SI I=K ET Z1=0 ALORS
                        DEBUT
                            P:=ZZP[ZP1+1];
                            ALLERA  SPECIALREDUI
                        FIN ;
                        Y:=ZZP[ZP1+1]+ZP1+1;
                        POUR M:=ZP1+3 PAS 1 JUSQUA Y FAIRE
                        SI Y1=ZZP[M] ALORS
                        DEBUT
                            P:=ZZP[ZP1+2];
SPECIALREDUI:               I2:=I2-LPRIME[P,2];
                            SI ARBRE ALORS
                            DEBUT
                                ECRIRECHAINE(1, "                    "
                                );
                                ECRIRECHAINE(1,
                                "                              " );
                                POUR I:=I2 PAS 2 JUSQUA I2+LPRIME[P,2]
                                FAIRE
                                DEBUT
                                    ECRIRECHAINE(1, " " );
                                    ECRIREATOME(1,TCDR[PILE2[I]]);
                                FIN ;
                                ECRIRECHAINE(1, " SE REECRIT " );
                                ECRIREATOME(1,TCDR[LPRIME[P,1]]);
                                ALALIGNE(1);
                            FIN ;
                            Y2:=PILE2[I2]:=LPRIME[P,1];
                            SI:=PILE2[I2-1];
                            Z1:=RE[SI];
                            VT:= FAUX ;
                            ALLERA  ETASUIVANT;
                        FIN ;
                        ZP1:=Y;
                    FIN ;

                    SI Z1=0 ALORS ALLERA  ERREUR;

TRANSLATE:

                    COMMENTAIRE :ON MET L'UNITE SYNTAXIQUE SUIVANTE
                    AU SOMMET DE LA PILE PILE2 ;

                    I2:=I2+1;
                    Y2:=PILE2[I2]:=Y1;
```

```
                    I1:=I1+1;
                    Y1:=PILE1[I1];
                    VT:= VRAI ;

ETASUIVANT:

                    COMMENTAIRE :RECHERCHE DE L'ETAT SUIVANT ;

                    Y:=Z1;
                    SI VT ALORS
                    DEBUT
                        R1:=RET[Y]+Y;
                        PAS:=-1;
                        R2:=Y+1;
                    FIN
                    SINON
                    DEBUT
                        R1:=Y+1;
                        PAS:=1;
                        R2:=RET[Y]+Y;
                    FIN ;
                    POUR I:=R1 PAS PAS JUSQUA R2 FAIRE
                    DEBUT
                        SI:=RET[I];
                        SI Y2=ETA[SI] ALORS
                        DEBUT
                            I2:=I2+1;
                            PILE2[I2]:=SI;
                            ALLERA  ANALYSE;
                        FIN ;
                    FIN ;

ERREUR:             ECRIRECHAINE(1,
                    " LA CHAINE PRECEDENTE EST INCORRECTE " );
                    ECRIRECHAINE(1, " , ERREUR A L UNITE SYNTAXIQUE "
                    );
                    ECRIREATOME(1,CDR(ENTIERENLISTE(I1)));
                FIN ;
            FIN ;
        FIN ;
    FIN BLOC UN ;
FIN PROGRAMME LISP ;
```

CHAINE ANALYSEE

```
'DEBUT' 'DEBUT' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A'
S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON'
S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B
'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S
, 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S
'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE'
'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S ,
'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS'
'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR'
I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI'
B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI'
B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I
:= 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B
'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B
'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I :=
'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS'
S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS'
S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI'
B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S
'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S
'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'DEBUT' L := B ::
H := S 'FIN' 'FIN' :: 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS'
S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI'
B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S
'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S
'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B
'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON'
S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON'
S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS'
S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON'
'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S
'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S
'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE'
B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A'
S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON'
S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B
'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S
, 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S
'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE'
'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S ,
'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'DEBUT' L := B :: H := S 'FIN' 'FIN'
//
```

LA CHAINE PRECEDENTE EST CORRECTE , LE TEMPS D ANALYSE EST DE 10 DIXIEMES DE SECONDE

CHAINE ANALYSEE

'DEBUT' 'AIGUILLAGE' A := E , E :: 'TABLEAU' M .'L' 'SI' B 'ALORS' S 'SINON' S : S
'J'. :: P 'L' A .'L' S 'J'. , M 'J' :: 'POUR' R := S 'PAS' S 'A'
S 'FAIRE' 'POUR' F := 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := S 'PAS'
S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'FAIRE' 'DEBUT' L := B :: H := S
:: 'ALLERA' 'SI' B 'ALORS' A .'L' S 'J'. 'SINON' A .'L' S 'J'. 'FIN' 'FIN' //

LA CHAINE PRECEDENTE EST CORRECTE , LE TEMPS D ANALYSE EST DE 1 DIXIEMES DE SECONDE
NOMBRE D UNITES SYNTAXIQUES 93 DONC 930 UNITES SYNTAXIQUES PAR SECONDE

CHAINE ANALYSEE

```
'DEBUT' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S
, 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S
'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE'
'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S ,
'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS'
'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR'
I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI'
B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI'
B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I
:= 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B
'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B
'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I :=
'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS'
S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS'
S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI'
B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S
'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S
'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B
'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON' S 'A' S , 'SI' B 'ALORS' S 'SINON'
S 'TANTQUE' B 'FAIRE' 'POUR' I := 'SI' B 'ALORS' S 'SINON' S 'PAS' 'SI' B 'ALORS' S 'SINON'
S 'A' S , 'SI' B 'ALORS' S 'SINON' S 'TANTQUE' B 'FAIRE' 'DEBUT' L := B :: H
:= S 'FIN' 'FIN' //
```

LA CHAINE PRECEDENTE EST CORRECTE , LE TEMPS D ANALYSE EST DE 5 DIXIEMES DE SECONDE
NOMBRE D UNITES SYNTAXIQUES 404 SOIT 808 UNITES SYNTAXIQUES PAR SECONDE

CHAINE ANALYSEE

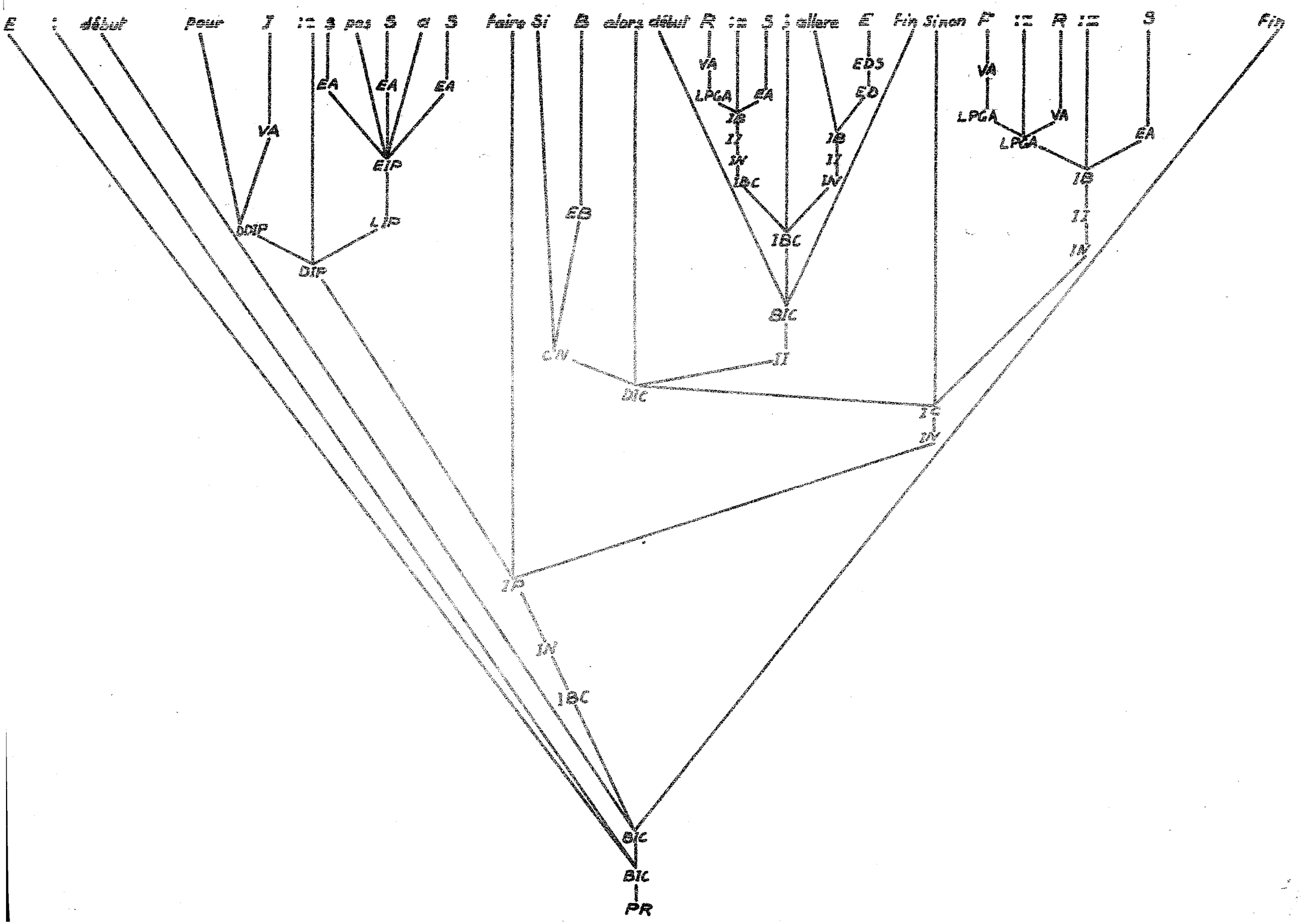
A A B B A A B B B B A A B B A A B B B B B B A A B B A A B B B B B B B
B A A B B A A B B B B A A B B A A B B B B B B A A B B A A B B B B B B B
B B B B A A B B A A B B B B A A B B A A B B B B B B A A B B A A B B B B B B
B B B B B A A B B A A B B B B A A B B A A B B B B B B A A B B A A B B B B B B
B B B B B B B B B B A A B B A A B B B B A A B B A A B B B B B B A A B B A A B B B B A A B
B A A B B B B B B B B A A B B A A B B B B A A B B A A B B B B B B A A B B A A B B B B A A
B B A A B B B B B B B B B B A A B B A A B B B B A A B B A A B B B B B B A A B B A A B B
B A A B B A A B B B B B B B A A B B A A B B B B A A B B A A B B B B B B A A B B A A B B
B B A A B B A A //

LA CHAINE PRECEDENTE EST CORRECTE , LE TEMPS D ANALYSE EST DE 3 DIXIEMES DE SECONDE
NOMBRE D UNITES SYNTAXIQUES 369 DONC 1230 UNITES SYNTAXIQUES PAR SECONDE

CHAINE ANALYSEE

E : 'DEBUT' 'POUR' I := S 'PAS' S 'A' S 'FAIRE' 'SI' B 'ALORS' 'DEBUT' R := S
:: 'ALLERA' E 'FIN' 'SINON' F := R := S 'FIN' //

I SE REECRIT VA
'POUR' VA SE REECRIT DDIP
S SE REECRIT EA
S SE REECRIT EA
S SE REECRIT EA
EA 'PAS' EA 'A' EA SE REECRIT ELP
ELP SE REECRIT LIP
DDIP := LIP SE REECRIT DIP
B SE REECRIT EB
'SI' EB SE REECRIT CN
R SE REECRIT VA
VA SE REECRIT LPGA
S SE REECRIT EA
LPGA := EA SE REECRIT IB
IB SE REECRIT II
II SE REECRIT IN
IN SE REECRIT IBC
E SE REECRIT EDS
EDS SE REECRIT ED
'ALLERA' ED SE REECRIT IB
IB SE REECRIT II
II SE REECRIT IN
IBC :: IN SE REECRIT IBC
'DEBUT' IBC 'FIN' SE REECRIT BIC
BIC SE REECRIT II
CN 'ALORS' II SE REECRIT DIC
F SE REECRIT VA
VA SE REECRIT LPGA
R SE REECRIT VA
LPGA := VA SE REECRIT LPGA
S SE REECRIT EA
LPGA := EA SE REECRIT IB
IB SE REECRIT II
II SE REECRIT IN
DIC 'SINON' IN SE REECRIT IC
IC SE REECRIT IN
DIP 'FAIRE' IN SE REECRIT IP
IP SE REECRIT IN
IN SE REECRIT IBC
'DEBUT' IBC 'FIN' SE REECRIT BIC
E : BIC SE REECRIT BIC


E
début
pour
I
pas
S
a
S
faire
Si
B
alors
début
R
S
allere
E
Fin
Sinon
F
R
S
Fin
VA
EA
EA
EA
EIP
LIP
DDIP
DIP
EB
CN
DIC
II
IC
IN
IP
IN
IBC
BIC
BIC
PR
VA
LPGA
EA
IB
II
IN
IBC
EDS
ED
IB
II
IN
IBC
BIC
II
VA
LPGA
VA
LPGA
EA
IB
II
IN

CHAPITRE VI

CONCLUSION

VI - 1 Rapidité de l'analyseur.

Les temps que nous avons obtenus (800 à 900 unités syntaxiques par seconde) engagent à passer en langage machine pour l'écriture de l'analyseur.

En effet, comparons ces temps à ceux d'un compilateur écrit "à la main" en langage machine. On évalue le temps de compilation total à 150 unités syntaxiques par seconde sur IBM 7044 à disques [BOU]. D'autre part, on estime généralement que le temps d'analyse est environ le 1/10 du temps total de compilation. Ce qui revient à dire que le temps d'analyse syntaxique est de l'ordre de 1500 unités syntaxiques par seconde.

Les temps que nous avons obtenus sont donc tout à fait comparables étant donné qu'il existe un facteur temps ≥ 3 entre l'exécution d'un programme écrit en Algol et celle d'un programme écrit en langage machine.

Il est à remarquer que la transformation de la table reste valable. Il serait peut être intéressant d'ordonner les états lors de la construction de la table : Avec un état S_n, on sait qu'on crée RTABLE [n]. RTABLE [n-1] états numérotés de RTABLE [n-1] +1 à RTABLE [n]. On peut donc définir un nouveau tableau FRONTIERE tel que de RTABLE n-1 +1 à FRONTIERE n on trouve les états caractérisés par un élément de V_N, et de FRONTIERE [n]+1 à RTABLE [n] les autres. Ce processus permettrait lors de la recherche des états suivants d'éviter au maximum la recherche à l'aide de deux tableaux. En effet, supposons que l'on vienne d'effectuer une réduction, on peut d'abord chercher l'état suivant au moyen de l'instruction suivante :

```
pour I := RTABLE [n-1] +1 jusqua FRONTIERE [n] faire
    si Y2 = ETA [ I] alors
        .........................
```

si on ne l'a pas trouvé alors on effectuera la recherche dans le tableau RET (si Y2 = ETA⌜RET⌜I⌝⌝ alors .....).

VI - 2 Construction de la table dans le cas général.

Comme nous l'avons vu au chapître III, la construction de la table n'est possible actuellement (en Lisp-Algol) que pour des grammaires comptant une centaine de productions et dans le cas particulier des grammaires LR(1). Il est naturel de se poser les deux questions suivantes :

1) Est-il possible de construire la table (k=1) pour des grammaires importantes ?

2) Est-il possible de construire cette table dans le cas général (k > 1)

1 - k=1

Nous pensons qu'il sera possible de le faire avec les ordinateurs de la troisième génération, en particulier avec le nouvel ensemble IBM 360-67 de l'Université de Grenoble. Nous avons programmé cette construction de table en Lisp-Algol, ce langage nous ayant paru le mieux adapté à ce problème. On peut garder ce langage sur 360, le problème devenant possible étant donné que l'utilisation des disques serait réduite à néant ou au maximum à cause de la grande capacité de la mémoire centrale.

Néanmoins, il nous semble plus intéressant d'essayer de programmer l'algorithme en "Algol-Wirth"⌜WH⌝. Actuellement, les quelques programmes tests que nous avons vuspermettent tous les espoirs quant à la rapidité de compilation et d'exécution. Il semblerait que ce langage (sur 360) soit au moins trois fois plus rapide à l'exécution qu'Algol 60 (sur 7044). Il est intéressant de noter que ce langage comprend la notion de "record" et de "référence". Actuellement le compilateur avec les "records" n'est pas encore au point. En outre, le D.C.E. du C.E.N. de Saclay est en train de réaliser l'inclusion du Lisp en "Algol-Wirth".

Il nous semble que cette inclusion qui est beaucoup plus naturelle pour ce langage que pour Algol 60 à cause des "records" doit donner des performances intéressantes. Nous pensons qu'ainsi on pourra d'une part améliorer considérablement le temps d'exécution du programme, d'autre part construire la table pour des grammaires LR(1) importantes.

2 - Cas général ($k > 1$)

Lorsque le problème précédent aura été résolu, nous pensons qu'il sera intéressant de savoir si une grammaire est LR(k) (ainsi on pourra affirmer que la grammaire est "non ambiguë" et que le langage défini par cette grammaire est déterministe).

La construction de la table devra être modifiée :

α) La convention que nous avons prise pour les états n'étant plus valable : $[p, j ; a_1\ a_2 \ldots a_i]$ à la place de $[p, j ; a_1]$, $[p, j ; a_2]$, ..., $[p, j ; a_i]$.

β) La construction de $H_k(\varphi)$ risquant d'être très longue car il faudra énumérer toutes les chaînes terminales θ de longueur k telles que $\varphi \Rightarrow \theta\ \Psi$.

Pour déterminer la valeur de k telle que la grammaire soit LR(k), nous proposons une méthode légèrement différente de celle proposée par J. COHEN [CO] . Au lieu de faire des essais pour k=1, 2, 3, ..., nous pensons qu'il est préférable d'essayer par exemple pour k=1, 4, 7 ... .

Supposons que nous ayons trouvé une grammaire LR(7). Nous savons qu'elle n'est pas LR(4). Par contre elle peut être LR($5 \leq k < 7$). Pour déterminer ce k, il suffit de parcourir tous les ensembles Z et Z_p. Pour chaque état, on déterminera la longueur minimum des chaînes appartenant à Z, Z^0, Z^1, ... Z^s telles que ces ensembles soient disjoints deux à deux. Soit ℓ_i la longueur minimum correspondant à l'état numéro i. La grammaire sera LR(k) pour $k = \max_i \ell_i$.

Pour l'analyse, nous pensons qu'on aura intérêt, ayant une grammaire LR(k), à chercher une grammaire équivalente qui soit LR(1). Bien que le problème soit théoriquement possible, il se peut qu'on n'y arrive pas. Supposons qu'on ait trouvé un

$k_0 (1 < k_0 \leq k)$ tel qu'il existe une grammaire équivalente qui soit $LR(k_0)$ et qu'on n'arrive pas à trouver une grammaire équivalente qui soit LR(1). Il sera intéressant de déterminer tous les états pour lesquels la condition LR(0), LR(1) ... $LR(k_0-1)$ est vérifiée. On fera la transformation de la table et l'analyse en fonction de cette détermination. Ainsi, on ne testera que le contexte minimum. C'est ce que nous avons réalisé dans le cas des grammaires LR(1) (chapîtres V et VI).

Table d'états minimum.

En général, la table d'états que nous avons construite n'est pas minimum. En effet, nous ne groupons des états que dans des cas bien particuliers pour ne pas perdre trop de temps à l'exécution (chapître III). Le problème reste donc ouvert quant à la détermination d'une telle table et quant aux répercutions sur le temps d'exécution et sur la détection des erreurs lors de l'analyse.

VI - 3 Lien avec la génération.

Nous pensons, qu'avec cette méthode d'analyse, il est possible de faire en même temps de la génération. A chaque réduction on peut essayer de définir une action, celle-ci ayant pour but de générer du langage machine et d'effectuer éventuellement certaines vérifications sur les déclarations, les appels de procédure, les paramètres effectifs, les étiquettes, etc ... .

Nous pensons que l'évaluation et la génération des expressions arithmétiques, des expressions booléennes, des boucles pour, etc..., ne présenteraient pas trop de difficultés, étant donné que la priorité des opérateurs nous est donnée implicitement par la grammaire.

De plus, il serait utile de présenter les productions sous une forme facilitant la génération.

Par exemple, au lieu de définir <EA> par

(1) <EA> ::= si <EB> alors <EAS> sinon <EA>

il semble plus intéressant de définir <EA> par

(1') <EA> ::= <proposition si> <EAS-sinon> <EA>

(2') <proposition si> ::= si <EB> alors

(3') <EAS-sinon> ::= <EAS> sinon

dans l'ordre, on évalue <EB>, un certain nombre de réductions ayant été effectuées pour obtenir <EB>, puis on effectue la réduction (2') à laquelle on peut associer une action, ensuite évaluation de <EAS> puis réduction avec (3') à laquelle on associe une nouvelle action et enfin évaluation de <EA>, réduction avec (1') et action correspondante.

Par contre avec (1), il faut attendre d'avoir évalué <EA> pour générer une action correspondant à (1).

Ces considérations ont été étudiées par Wirth et Weber [WW] et il semble que le compilateur PL 360 [WI] soit réalisé ainsi à l'aide d'un analyseur de précédences.

La grande difficulté dans la réalisation d'un compilateur dirigé par la syntaxe et plus généralement d'un compilateur paramétrique vient du fait qu'en général un langage de programmation ne peut pas être défini entièrement par une grammaire context-free.

Nous manquons d'expérience dans ce domaine. Le problème de réalisation d'un tel compilateur en vue de son incorporation, au niveau opérationnel, dans un système de programmation, semble toutefois rester ouvert.

BIBLIOGRAPHIE

BGL L. BOLLIET, N. GASTINEL, et P.J. LAURENT, Un nouveau langage scientifique ALGOL, manuel pratique, Hermann, Paris, 1964.

BOL L. BOLLIET, Notation et processus de traduction des langages symboliques, Thèse, Université de Grenoble, Juin 1967.

BOU J.C. BOUSSARD, Etude et réalisation d'un compilateur algol 60 sur calculatrice du type IBM 7090/94 et 7040/44, Thèse, Université de Grenoble, Juin 1964.

CD J. COHEN et NGUYEN HUU DUNG, Définition de procédures Lisp en algol. Exemples d'utilisation, chiffres, vol. 8, n° 4, 1965.

CO J. COHEN, Langages pour l'écriture des compilateurs, Thèse, Université de Grenoble, Juin 1967.

Col A. COLMERAUER, Précédence, analyse syntaxique et langages de programmation, Thèse, Université de Grenoble, Septembre 1967.

FL R.W. FLOYD, Special classes of C.F. languages and their reconisers. N.A.T.O. summer school in programming languages, Villard de Lans. September, 1966.

GG S. GINSBURG et S. GREIBACH, Deterministic context-free languages, inf. and Control, n° 9, 1966.

GL M. GROSS et A. LENTIN, Notions sur les grammaires formelles, Gauthiers-Villars, 1967.

Kn-1 D.E. KNUTH, on the translation of languages from left to right, inf. and Control, n° 8, 1965.

Kn-2 D.E. KNUTH, Top-down syntax-analysis. International N.A.T.O. summer school on computer programming, Copenhagen, Denmark, August 7-25, 1967.

MA F. MARTIN. Détermination de certaines caractéristiques des grammaires et langages context-free, Thèse, Université de Grenoble, à paraître.

VA B. VAUQUOIS. Cours pratique de logique et programmation, polycopié, Université de Grenoble, 1964.

TR L. TRILLING. Contribution à l'étude des mécanismes de traduction des langages de programmation. Application au traitement des structures définies dynamiquement, Thèse, Université de Grenoble, Décembre 1967.

VVV G. VEILLON, J. VEYRUNES, B. VAUQUOIS. Un métalangue de grammaires transformationnelles, C.E.T.A.G2300-A. Janvier 1967.

VW A. VAN WIJNGAARDEN, Rapport préliminaire Algol X, Varsovie, Octobre 1966.

WH N. WIRTH et C.A.R. HOARE. A contribution to the development of Algol. Comm. ACM, vol. 9, Juin 1966.

WI N. WIRTH. A programming language for the 360 computers. Technical report NO.CS.53. December 20, 1966 revised June 20, 1967 Computer Science department Stanford University.

WW N. WIRTH et H. WEBER, Euler : a generalization of Algol and its formal definition, Comm. ACM, Janvier 1966.

VU

Grenoble, le

Le Président de la Thèse

VU

Grenoble, le

Le Doyen de la Faculté des Sciences

VU, et permis d'imprimer,

Le Recteur de l'Académie de GRENOBLE