Précédence, analyse syntaxique et langages de programmation

Alain Colmerauer

n° d'ordre

THESES

présentées à la Faculté des Sciences
de Grenoble pour obtenir le
grade de Docteur ès Sciences Appliquées
par

Alain COLMERAUER

Ingénieur IMAG, Licencié ès Sciences

première thèse

PRECEDENCE, ANALYSE SYNTAXIQUE ET LANGAGES DE PROGRAMMATION

deuxième thèse

PROPOSITION DONNEE PAR LA FACULTE

Thèses soutenues le 26 Septembre 1967,
devant la commission d'examen

MM. VAUQUOIS	Président
GASTINEL	Examinateurs
Mme. BERTRANDIAS	
MM. BOLLIET	
GENUYS	

FACULTE DES SCIENCES

LISTE DES PROFESSEURS

DOYENS HONORAIRES :

M. MORET
M. WEIL

DOYEN :

M. BONNIER E.

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 intégral
	BENOIT Jean	Chaire de Radioélectricité
	CHENE Marcel	Chaire de Chimie Papetière
	WEIL Louis	Chaire de Thermodynamique
	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 M.P.C.
	GALVANI O.	Mathématiques
	MOUSSA André	Chaire de Chimie Nucléaire
	TRAYNARD 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
Melle	LUTZ Elisabeth	Mathématiques
MM.	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 Electronique
	BARJON R.	Physique Nucléaire
	BARBIER Jean-Claude	Chaire de Physique
	SILBER R.	Mécanique des Fluides
	BUYLE-BODIN Maurice	Chaire d'Electronique
	DREYFUS B.	Thermodynamique
	KLEIN J.	Mathématiques
	VAILLANT F.	Zoologie et Hydrobiologie
	ARNOUD Paul	Chaire de Chimie M.P.C.
	SENGEL P.	Chaire de Zoologie
	BARNOUD F.	Chaire de Biosynthèse de la Cellulose
	BRISSONNEAU P.	Physique
	GAGNAIRE Didier	Chaire de Chimie Physique

Mme	KOFLER L.	Botanique
MM.	DEGRANGE Charles	Zoologie
	PEBAY-PEROULA J.C.	Physique
	RASSAT A.	Chaire de Chimie Systématique

PROFESSEURS SANS CHAIRE :

MM.	GIDON P.	Géologie et Minéralogie
	GIRAUD P.	Géologie
	PERRET R.	Servomécanismes
Mme	BARBIER M.J.	Electrochimie
Mme	SOUTIF J.	Physique
MM.	COHEN J.	Electrotechnique
	DEPASSEL R.	Mécanique des Fluides
	GASTINEL N.	Mathématiques Appliquées
	ANGLES-d'AURIAC P.	Mécanique des Fluides
	DUCROS P.	Minéralogie et Cristallographie
	GLENAT R.	Chimie
	LACAZE A.	Thermodynamique
	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 Génie Atomique
	HACQUES Gérard	Calcul Numérique
	POLOUJADOFF M.	Electrotechnique

MAITRES DE CONFERENCES :

MM.	DODU J.	Mécanique des Fluides
	LANCIA Roland	Physique Automatique
Mme	KAHANE J.	Physique
MM.	DEPORTES C.	Chimie
Mme	BOUCHE L.	Mathématiques
MM.	SARROT-RAYNAUD J.	Géologie Propédeutique
Mme	BONNIER M.J.	Chimie
MM.	KAHANE A.	Physique Générale
	DOLIQUE J.M.	Electronique
	BRIERE G.	Physique M.P.C.
	DESPRE P.	Chimie S.P.C.N.
	LAJZEROWICZ J.	Physique M.P.C.
	VALENTIN P.	Physique M.P.C.
	BERTRANDIAS J.P.	Mathématiques Appliquées T.M.P.
	LAURENT P.	Mathématiques Appliquées T.M.P.
	CAUBET J.P.	Mathématiques Pures
	PAYAN J.J.	Mathématiques
Mme	BERTRANDIAS F.	Mathématiques Pures M.P.C.
MM.	LONGEQUEUE J.P.	Physique
	NIVAT M.	Mathématiques Appliquées
	SOHM J.C.	Electrochimie
	ZADWORNY F.	Electronique
	DURAND F.	Chimie Physique
	CARLIER 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
	BLOCH D.	Electrotechnique
	BRUGEL L.	I.U.T.
	SIBILLE R.	I.U.T.

Que Monsieur le Professeur J. Kuntzmann, Directeur de l'Institut de Mathématiques Appliquées, reçoive mes remerciements les plus sincères pour m'avoir donné la possiblité de préparer et de présenter cette thèse.

Je remercie Monsieur le Professeur B. Vauquois, Directeur du Centre d'Etudes pour la Traduction automatique, de m'avoir fait l'honneur de présider le jury, et je remercie Monsieur Le Professeur N. Gastinel, Directeur de l'Institut de Programmation de Grenoble, d'avoir eu la bienveillance de s'intéresser à ce travail. Enfin je ne manquerais pas de remercier Madame Bertrandiaz, Maître de Conférence à l'Université de Grenoble, qui dans le cadre de la deuxième thèse, m'a donné l'occasion d'apprécier un domaine des mathématiques nouveau pour moi.

Je remercie tout spécialement Monsieur L. Bolliet, Maître de Conférence à l'Institut Universitaire de Technologie de Grenoble, de m'avoir accordé sa confiance pour réaliser ce travail au sein de son équipe. Je remercie aussi très vivement Monsieur F. Genuys, Conseiller Scientifique à la Compagnie IBM, d'avoir bien voulu m'aider et me guider dans mes recherches.

Enfin je remercie toutes les personnes du laboratoire, qui par leurs entretiens ou leurs travaux, ont contribué à la réalisation de cette thèse, en particulier Jean-Claude Boussard, dont les critiques et les conseils me furent très utiles durant la rédaction, Philippe Jorrand qui s'est toujours montré un ami attentif et perspicace, Laurent Trilling qui fut un chaleureux compagnon de travail et François Martin dont certains travaux me furent nécessaires.

Je ne voudrais pas non plus oublier dans mes remerciements tous ceux qui ont participé à la réalisation matérielle de cette thèse.

A ma Femme

TABLE DES MATIERES

INTRODUCTION

Les grammaires "context-free" de N. Chomsky [Ch] semblent s'être imposées comme un modèle particulièrement commode pour décrire des langages, aussi bien en linguistique pour les langues naturelles qu'en compilation pour les langages de programmation. Dans ce dernier domaine, le langage Algol 60 [Na] en est l'exemple le plus typique.

Cependant l'utilisation de ces grammaires pour l'analyse automatique de textes se heurte à des difficultés d'ordre pratique : les algorithmes généraux d'analyse existants coûtent beaucoup de temps ou de place à l'ordinateur. Puisque la compilation nous intéresse surtout ici, nous avons étudié des méthodes d'analyse adaptées aux grammaires particulières que l'on y rencontre.

Les méthodes que nous préconisons utilisent des relations dites de "précédence" représentées par des matrices booléennes et calculées une fois pour toutes à partir de la grammaire considérée. Leur rôle est de permettre des regroupements facilitant l'analyse du texte. On peut considérer que la notion de précédence est une généralisation de celle de priorité entre les opérateurs d'une même expression algébrique.

Voici la démarche suivie au cours de ce travail. Nous avons tout d'abord défini un formalisme pratique pour manipuler nos relations binaires. Nous l'utilisons constamment, surtout pour le calcul de matrices booléennes représentant des relations de précédence. Nous nous intéressons aussi à quelques caractéristiques importantes des grammaires "context-free" et nous introduisons le problème de l'analyse syntaxique. Tout ceci fait l'objet du premier chapitre.

Dans le deuxième chapitre nous abordons l'étude d'une première méthode

d'analyse très agréable par son efficacité et sa simplicité. Elle nous a permis entre autres de découvrir une sous-classe intéressante des langages "context-free" les langages "de précédence totale".

Les notions d'opérateurs et d'opérandes dans une grammaire apparaissent au troisième chapitre et permettent de compléter les travaux de R. Floyd |F1.1| qui le premier, a introduit la notion de précédence.

Les quatrième et cinquième chapitres sont consacrés aux applications pratiques : tout d'abord la façon de programmer les différents algorithmes décrits, puis leur utilisation pour des langages de programmation déterminés.

CHAPITRE 1 : Eléments de base

1.1 Quelques opérations sur les relations binaires

Nous appellerons relation binaire sur un ensemble E, tout sous-ensemble ρ du produit cartésien $E\times E$. Par commodité nous noterons souvent $a\rho b$ au lieu de $(a,b)\in\rho$ et nous représenterons toujours une relation binaire par une lettre grecque minuscule ou un signe spécial. Le complément de ρ par rapport à $E\times E$ sera noté $\bar{\rho}$; on a donc

$$\bar{\rho} = E\times E-\rho$$

DEFINITION Etant donné deux relations binaires ρ et σ sur le même ensemble E, le produit $\rho\sigma$ sera une nouvelle relation binaire sur E définie ainsi : pour tout $a,b\in E$

$$a\rho\sigma b \equiv [\text{il existe } c\in E \text{ tel que } a\rho c \text{ et } c\sigma b]$$

Il est facile de vérifier que ce produit est associatif et qu'il est distributif, à droite et à gauche, par rapport à l'union. Du fait de l'associativité, nous pouvons donc définir sans ambiguïté l'élévation à une puissance entière non négative d'une relation binaire ρ sur E.

$$\rho^i = \rho\rho^{i-1} \text{ avec } a\rho^o b \equiv [a=b] \qquad \text{pour tout } a,b\in E$$

DEFINITION Etant donné une relation binaire ρ sur E, sa fermeture transitive ρ^+ et sa fermeture transitive reflexive ρ^* seront deux nouvelles relations binaires sur E définies ainsi :

$$\rho^+ = U_{i=1}^{\infty}\rho^i \qquad \rho^* = \rho^o U \rho^+$$

Il découle immédiatement que $\rho^+ = \rho\rho^* = \rho^*\rho$. Dans le cas où E est un ensemble fini de n éléments, pour tout couple $a,b\in E$ tel que $a\rho^k b$ avec $k\geq n$ il existe forcément $\ell < n$ tel que $a\rho^\ell b$. De ce fait $U_{i=1}^{\infty}\rho^i = U_{i=1}^{n-1}\rho^i$ et donc $\rho^+ = U_{i=1}^{n-1}\rho^i$.

De plus, comme nous le verrons au paragraphe 4.1, il est alors possible de transposer toutes les opérations sur des relations binaires de E en des opérations sur des matrices booléennes, de dimension n×n, les représentant.

1.2 Grammaire "context-free"

Considérons un ensemble fini W appelé vocabulaire dont les éléments sont appelés symboles. Une chaîne x sur W sera une suite finie de symboles $A_1, A_2, \ldots, A_n \in W$. Par commodité nous supprimerons les virgules séparant les éléments de la suite et nous noterons $x = A_1A_2\ldots A_n$. La longueur de la chaîne x sera l'entier n et sera notée $|x|$. Par W^* nous désignerons l'ensemble de toutes les chaînes que l'on peut construire sur W y compris la chaîne vide de longueur nulle que nous noterons Λ. Etant donné deux chaînes $x, y \in W^*$ avec $x = A_1A_2\ldots A_i$ et $y = B_1B_2\ldots B_j$ nous poserons $xy = A_1A_2\ldots A_iB_1B_2\ldots B_j$. Bien entendu $x\Lambda = \Lambda x = x$ quelque soit $x \in W^*$.

DEFINITION Tout sous-ensemble de W^* sera appelé langage.

DEFINITION Une grammaire "context-free" G sera définie par

- un vocabulaire V_T dit terminal,
- un vocabulaire V_N dit non terminal tel que $V_T \cap V_N = \emptyset$,
- un symbole distingué de V_N appelé axiome et noté S,
- une relation binaire finie sur $(V_T \cup V_N)^*$ notée $\rightarrow$ dont les éléments seront appelés règles et telle que pour tout $x, y \in (V_T \cup V_N)^*$, $x \rightarrow y$ entraîne $x \in V_N$.

Cette définition est très voisine de celle donnée par Chomsky en [Ch]. Sur $(V_T \cup V_N)^*$ nous définirons une deuxième relation binaire, notée $\Rightarrow$, qui sera une extension de la relation $\rightarrow$:

DEFINITION Pour tout couple de chaînes $x, y \in (V_T \cup V_N)^*$:
$x \Rightarrow y \equiv$ [il existe $Z \in V_N$, $u, v, z \in (V_T \cup V_N)^*$ tels que $x = uZv$, $y = uzv$, $Z \rightarrow z$]

Il faut remarquer que les éléments Z, z, u, v ne sont pas toujours déterminés d'une façon unique ; voici un exemple : $AB \Rightarrow AUB$ avec $A \rightarrow AU$ et $B \rightarrow UB$.

DEFINITION Le langage défini par la grammaire G est le sous-ensemble L(G) de V_T^* ainsi défini :

$$L(G) = \{t \in V_T^* \mid S \Rightarrow^* t\}$$

Un tel langage est dit "context-free".

En convenant d'appeler phrase toute chaîne $s \in (V_T \cup V_N)^*$ telle que $S \Rightarrow^* s$ et de qualifier de terminale toute chaîne $t \in V_T^*$, ce langage L(G) est donc l'ensemble des phrases terminales de G.

Signalons une propriété bien connue de la relation =>, propriété que nous utiliserons souvent par la suite :

LEMME 1.2.1 Si $x_1x_2 \Rightarrow^n y$ avec $x_1, x_2, y \in (V_T \cup V_N)^*$ et $n \geq o$ alors il existe $y_1, y_2 \in (V_T \cup V_N)^*$ et des entiers, $i,j \geq o$ tels que $y=y_1y_2$, $x_1 \Rightarrow^i y_1$, $x_2 \Rightarrow^j y_2$, $i+j=n$.

DEMONSTRATION Le théorème est vrai pour n=o. Supposons le vrai pour n et démontrons qu'il est vrai pour n+1. Si $x_1x_2 \Rightarrow^{n+1} y$ alors il existe $z \in (V_T \cup V_N)^*$ tel que $x_1x_2 \Rightarrow^n z \Rightarrow y$. Le théorème étant supposé vrai pour n il existe donc z_1, z_2 et k, ℓ tels que $z=z_1z_2$, $x_1 \Rightarrow^k z_1$, $x_2 \Rightarrow^\ell z_2$, $k+\ell=n$. Soit $U \to u$ la règle qui permet de transformer z en y. L'élément U est soit contenu dans z_1 soit dans z_2. Supposons par exemple qu'il soit contenu dans z_1. Il existe donc v,w tels que $z_1=vUw \Rightarrow vuw$. En posant $y_1=vuw$, $y_2=z_2$, $i=k+1$, $j=\ell$ on a donc $y=y_1y_2$, $x_1 \Rightarrow^i y_1$, $x_2 \Rightarrow^j y_2$, $i+j=n+1$.

1.3 Grammaire "context-free" parenthèsée

DEFINITION A toute grammaire "context-free" G, on peut associer une grammaire "context-free" G' dite grammaire parenthésée : soit V_T' et V_N' respectivement les vocabulaires terminal et non terminal de G'. A chaque symbole $U \in V_N$ nous ferons correspondre deux nouveaux symboles notés par exemple $[_U$ et $]_U$ et nous désignerons par $V_{[}$ l'ensemble des symboles de la forme $[_U$ et par $V_{]}$ l'ensemble des symboles de la forme $]_U$. Moyennant ces conventions G' sera définie ainsi :

- $V'_T = V_T \cup V \cup \overline{V}$,
- $V'_N = V_N$
- l'axiome de G' est le même que celui de G,
- à chaque règle $U \to u$ de G correspond biunivoquement la règle $U \to \underset{U}{[}u\underset{U}{]}$ de G'.

Les grammaires parenthésées possèdent de nombreuses propriétés.

Tout d'abord il faut remarquer que pour tout couple de chaînes x,y telles que $x \Rightarrow y$ dans G', les chaînes u,v et la règle $Z \to \underset{Z}{[}z\underset{Z}{]}$ telles que x=uZv et $y=u\underset{Z}{[}z\underset{Z}{]}v$ sont déterminées d'une façon unique.

Une deuxième propriété fondamentale est la suivante :

LEMME 1.3.1 Si dans une grammaire parenthésée G' on a $s \Rightarrow^n x\underset{X}{[}u\underset{Y}{]}y$ avec $X,Y \in V_N$, $s,u \in (V_T \cup V_N)^*$ et $x,y \in (V_T \cup V_N \cup \overline{V})^*$ alors il existe $U \in V_N$ tel que U=X=Y et tel que $s \Rightarrow^{n-1} xUy$.

DEMONSTRATION Le théorème est vrai pour n=1. Supposons le vrai pour n et démontrons qu'il est vrai pour n+1. Si $s \Rightarrow^{n+1} x\underset{X}{[}u\underset{Y}{]}y$ alors il existe $z \in (V_T \cup V_N \cup \overline{V})^*$ tel que $s \Rightarrow^n z \Rightarrow x\underset{X}{[}u\underset{Y}{]}y$. Trois cas se présentent alors :

(1) $z = xUy \Rightarrow x\underset{X}{[}u\underset{Y}{]}y$. On a donc U=X=Y et $s \Rightarrow^n xUy$.

(2) $z = r\underset{X}{[}u\underset{Y}{]}y \Rightarrow x\underset{X}{[}u\underset{Y}{]}y$ avec $r \in (V_T \cup V_N \cup \overline{V})^*$ et $r \Rightarrow x$.

Le théorème étant vrai pour n il existe donc $U \in V_N$ tel que U=X=Y et $s \Rightarrow^{n-1} rUy$. Du fait que $r \Rightarrow x$ on a $s \Rightarrow^n xUy$.

(3) $z = x\underset{X}{[}u\underset{Y}{]}t \Rightarrow x\underset{X}{[}u\underset{Y}{]}y$ avec $t \in (V_T \cup V_N \cup \overline{V})^*$ et $t \Rightarrow y$.

Même démonstration que pour le cas (2).

En appliquant j fois ce lemme on déduit :

COROLLAIRE 1.3.2 Dans une grammaire parenthésée, si $x \Rightarrow^{i} z$ et $y \Rightarrow^{j} z$ avec $x \in (V_T \cup V_N)^*$ et $y, z \in (V_T \cup V_N \cup V_{\lceil} \cup V_{\rceil})^*$ alors $i \geq j$ et $x \Rightarrow^{i-j} y$.

En utilisant ce corollaire nous pouvons maintenant démontrer cette dernière propriété.

LEMME 1.3.3. Si dans une grammaire parenthésée on considère deux suites $x_o, x_1, \ldots, x_i \in (V_T \cup V_N \cup V_{\lceil} \cup V_{\rceil})^*$ et $y_o, y_1, \ldots, y_j \in (V_T \cup V_N \cup V_{\lceil} \cup V_{\rceil})^*$ telles que

(a) $x_o \Rightarrow x_1 \Rightarrow \ldots \Rightarrow x_i$ (b) $y_o \Rightarrow y_1 \Rightarrow \ldots \Rightarrow y_j$

(c) $x_o, y_o \in (V_T \cup V_N)^*$ (d) $x_i = y_j$

alors

(1) $i=j$ et $x_o = y_o$,

(2) si l'on désigne par ρ_1 et ρ_2 les sous-ensembles de règles intervenant respectivement dans les relations (a) et (b) alors $\rho_1 = \rho_2$.

DEMONSTRATION Du corollaire 1.3.2 on déduit que $i \geq j$, $x_o \Rightarrow^{i-j} y_o$, $j \geq i$, $y_o \Rightarrow^{j-i} x_o$ d'où $i=j$ et $x_o = y_o$. Démontrons le deuxième point par récurrence sur i. Si $i=o$ le théorème est vrai car $\rho_1 = \rho_2 = \emptyset$. Supposons le théorème vrai pour i et démontrons qu'il est vrai pour $i+1$: on a

$$x_o \Rightarrow x_1 \Rightarrow \ldots \Rightarrow x_{i+1},\ y_o \Rightarrow y_1 \Rightarrow \ldots \Rightarrow y_{i+1} \text{ avec } x_o = y_o \text{ et } x_{i+1} = y_{i+1}$$

Si $x_i = y_i$ le théorème est démontré. Si $x_i \neq y_i$ il existe donc deux règles $U \to \underset{U}{\lceil} u \underset{U}{\rceil}$, $V \to \underset{V}{\lceil} v \underset{V}{\rceil}$ et des chaînes r_1, r_2, r_3 telles que

$$x_{i+1} = y_{i+1} = r_1 \underset{U}{\lceil} u \underset{U}{\rceil} r_2 \underset{V}{\lceil} v \underset{V}{\rceil} r_3$$

$$x_i = r_1 U r_2 \underset{V}{\lceil} v \underset{V}{\rceil} r_3$$

$$y_i = r_1 \underset{U}{\lceil} u \underset{U}{\rceil} r_2 V r_3$$

(le fait d'intervertir x_i avec y_i ne changerait rien à la suite). Posons

$z_{i-1}=r_1 U r_2 V r_3$. D'après le corollaire 1.3.2 il existe donc une suite de chaînes $z_o,z_1,\ldots,z_{i-1}$ telle que

$$x_o=y_o=z_o=>z_1=>\ldots=>z_{i-1}$$

Soit ρ le sous-ensemble de règles intervenant dans ces relations.
Soit d'autre part, ρ'_1 et ρ'_2 les sous-ensembles de règles intervenant respectivement dans les relations

$$x_o=>x_1=>\ldots=>x_i \text{ et } y_o=>y_1=>\ldots=>y_i$$

On a $\rho_1=\rho'_1 U\{U\to u\}$ et $\rho_2=\rho'_2 U\{V\to v\}$. Le théorème étant supposé vrai pour i, on a $\rho'_1=\rho\cup\{V\to v\}$ et $\rho'_2=\rho U\{U\to u\}$.
De ce fait $\rho_1=\rho_2=\rho\cup\{U\to u\}U\{V\to v\}$.

1.4 Dérivation

Etant donné une grammaire G de vocabulaire $V_T \cup V_N$ nous allons définir la notion de dérivation. Pour ceci nous aurons d'abord besoin de la notion de position dans une chaîne.

DEFINITION Considérons une chaîne $x=A_1A_2\ldots A_n$ avec $A_1,A_2,\ldots,A_n \in V_T \cup V_N$. De la suite $A_1,A_2,\ldots,A_n$ extrayons la sous-suite x_N des éléments non terminaux. Nous dirons que l'élément non terminal A_p de x est en <u>position</u> k dans x si son rang est k dans la sous-suite x_N.

DEFINITION Une dérivation de longueur n, n étant un entier positif ou nul, sera définie par :

- une suite $x_o,x_1,\ldots,x_n \in (V_T \cup V_N)^*$ telle que $x_o=>x_1=>\ldots=>x_n$,
- dans le cas où $n\neq o$, une suite d'entiers $i_1,i_2,\ldots,i_n$ telle que pour tout $j\in(1,2,\ldots,n)$ la transformation de x_{j-1} en x_j se passe en appliquant la règle au symbole en position i_j de x_{j-1}.

Elle sera notée $(x_o ; i_1,x_1 ; i_2,x_2 ;\ldots; i_n,x_n)$ et nous dirons que x_o est son origine et x_n son extrémité.

En faisant intervenir la grammaire parenthésée G' associée à G, définissons la notion de structure.

DEFINITION Etant donnée une dérivation $\Delta=(x_o ; i_1,x_1;\ldots;i_n,x_n)$ dans G, on peut lui faire correspondre une et une seule dérivation associée dans la grammaire parenthésée G', définie de la façon suivante :

(1) elle est de la forme $(y_o;i_1,y_1;\ldots;i_n,y_n)$,
(2) $x_o=y_o$,
(3) pour tout $k\in(1,2,\ldots,)$ la chaîne x_k est égale à la chaîne y_k dans laquelle on aurait remplacé tous les symboles appartenant à $V_{[}V_{]}$ par la chaîne vide Λ .

La chaîne y_n est dite <u>structure engendrée par la dérivation</u> Δ. Nous appellerons <u>structure d'une phrase</u> de G, toute structure engendrée par une dérivation d'origine S et ayant cette phrase pour extrémité.
En d'autres termes on peut dire que toute phrase s de G' qui ne diffère d'une phrase t de G que par des symboles appartenant à $V_{[}V_{]}$ est une structure de t.

Introduisons une relation d'équivalence entre les dérivations de G.

DEFINITION Deux dérivations Δ_1 et Δ_2 de G seront dites équivalentes si et seulement si elles engendrent la même structure. On notera $\Delta_1\equiv\Delta_2$.

Du lemme 1.3.3 il découle immédiatement :

LEMME 1.4.1 Deux dérivations équivalentes
(1) ont même longueur, même origine, même extrémité,
(2) font intervenir le même sous-ensemble de règles.

Pour démontrer une deuxième propriété essentielle des dérivations nous aurons besoin du lemme suivant qui ne concerne que les grammaires parenthésées.

LEMME 1.4.2 Pour toute application f qui à chaque chaîne $u\in(V_T\cup V_N\cup V_{[}\cup V_{]})^*$, contenant au moins un symbole non terminal, fait correspondre la position f(u) d'un symbole non terminal de u et pour tout couple de chaînes $x\in(V_T\cup V_N\cup V_{[}\cup V_{]})^*$ et $t\in(V_T\cup V_{[}\cup V_{]})^*$, tels que $x=>^n t$, il existe une et une seule dérivation de la forme

$$(x=x_o;f(x_o),x_1;f(x_1),x_2;\ldots;f(x_{n-1}),x_n=t)$$

DEMONSTRATION Remarquons tout d'abord que si une telle dérivation existe elle ne peut être qu'unique et ceci du fait que nous raisonnons dans une grammaire parenthésée. Démontrons son existence par récurrence sur n. Si n=o cette dérivation existe. Supposons qu'elle existe pour n et démontrons qu'il en existe une pour n+1. Si $x=>^{n+1}t$, soit $U\in V_N$ le symbole en position f(x) dans x.
Posons x=yUz. En appliquant deux fois le lemme 1.2.1, on déduit qu'il existe des chaînes terminales y',u',z' telles que t=y'u'z' et telles que $y=>^i y'$, $U=>^j u'$, $z=>^k z'$, i+j+k=n+1. Soit u la chaîne telle que $U=>u=>^{j-1}u'$. Nous avons donc la dérivation (x;f(x),yuz) et la relation $yuz=>^n t$. Par supposition, il existe une dérivation de la forme $(yuz=x_1;f(x_1),x_2;\ldots;f(x_n),x_{n+1}=t)$, d'où la dérivation $(x;f(x),yuz=x_1;f(x_1),x_2;\ldots;f(x_n),x_{n+1}=t)$.

De ce lemme et de la définition de deux dérivations équivalentes on déduit une propriété concernant toutes les grammaires :

COROLLAIRE 1.4.3 Pour toute application f qui à chaque chaîne $u\in(V_T\cup V_N)^*$ contenant au moins un symbole non terminal fait correspondre la position f(u) d'un symbole non terminal de u et pour toute dérivation Δ_1 dont l'extrémité est terminale, il existe une et une seule dérivation Δ_2 de la forme

$$\Delta_2=(y_o;f(x_o),x_1;\ldots;f(x_{n-1}),x_n)$$

telle que $\Delta_1\equiv\Delta_2$.

Nous pouvons maintenant donner deux définitions équivalentes de l'ambiguïté dans une grammaire "context-free".

DEFINITION Une grammaire "context-free" G sera dite <u>ambiguë</u> si et seulement si elle contient une phrase terminale ayant plusieurs structures.

L'ambiguïté réside donc dans l'existence de deux dérivations non équivalentes de même origine S et de même extrémité terminale. En utilisant le corollaire 1.4.1 dans lequel on aurait choisi l'application f de façon à ce que f(u)=1 pour tout u, on en déduit cette définition équivalente de l'ambiguïté :

DEFINITION EQUIVALENTE Une grammaire "context-free" G est ambiguë si et seulement si il existe deux dérivations distinctes Δ_1 et Δ_2 de la forme :

$$\Delta_1=(x_o;1,x_1;1,x_2;\ldots;1,x_i)$$

$$\Delta_2=(y_o;1,y_1;1,y_2;\ldots;1,y_j)$$

avec $x_o=y_o=S$, $x_i=y_j$ et $x_i,y_j \in V_T^*$.
Cette deuxième définition est adoptée par plusieurs auteurs, entre autres par Ginsburg [Gi]. Nous utiliserons la première qui a l'avantage d'être symétrique.

Considérons par exemple la grammaire G_1 et sa grammaire parenthésée G_1' :

G_1 :	G_1' :
$S \to XX$	$S \to \underset{S}{[}XX\underset{S}{]}$
$X \to a$	$X \to \underset{X}{[}a\underset{X}{]}$
$X \to Xa$	$X \to \underset{X}{[}Xa\underset{X}{]}$

Aux dérivations

(1) (S;1,XX;1,XaX;1,aaX;1,aaa)

(2) (S;1,XX;1,XaX;2,Xaa;1,aaa)

(3) (S;1,XX;1,aX;1,aXa;1,aaa)

sont associées les dérivations

$$(S;1,\underset{S}{[}XX\underset{S}{]};1,[[Xa]X];1,[[[a]a]X];1[[[a]a][a]])$$
$$\quad\quad S\ S\quad SX\ X\ S\quad SXXaX\ X\ S\quad SXX\ X\ XX\ XS$$

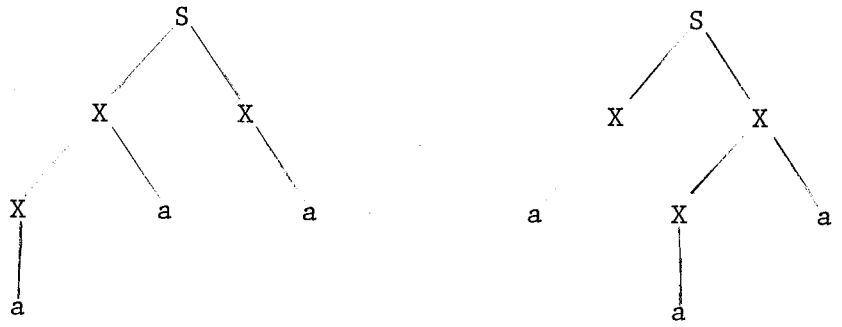
$$(S;1,\underset{S}{[}XX\underset{S}{]};1,[[Xa]X];2,[[Xa][a]];1[[[a]a][a]])$$
$$\quad\quad S\ S\quad SX\ X\ S\quad SX\ XX\ XS\quad SXX\ X\ XX\ XS$$

$$(S;1,\underset{S}{[}XX\underset{S}{]};1,[[a]X];1,[[a][Xa]];1,[[a]][a[a]])$$
$$\quad\quad S\ S\quad SX\ X\ S\quad SX\ XX\ XS\quad SX\ XXX\ X\ XS$$

Les dérivations (1) et (2) sont donc équivalentes. Les dérivations (1) et (3) engendrent les structures

$$\underset{SXX\ X\ XX\ XS}{[[[a]a][a]]} \qquad \text{et} \qquad \underset{SX\ XXX\ X\ XS}{[[a][[a]a]]}$$

de la phrase terminale "aaa". La grammaire G_1 est donc ambiguë. Les deux structures de "aaa" peuvent être représentées graphiquement ainsi :



1.5 Restrictions habituelles sur les grammaires "context-free"

L'étude que nous allons faire s'applique à toute grammaire "context-free" satisfaisant aux quatre restrictions habituelles :

(1) - elle ne comporte pas de cycle, c'est-à-dire de symbole non terminal U tel que $U \Rightarrow^{+} U$

(2) - elle ne comporte pas de règles dont le membre droit est la chaîne vide Λ ,

(3) - pour tout symbole $U \in V_T \cup V_N$ il existe toujours des chaînes $x,y \in (V_T \cup V_N)^*$ telles que $S \Rightarrow^{*} xUy$,

(4) - pour tout symbole $U \in V_T \cup V_N$ il existe toujours une chaîne $t \in V_T^*$ telle que $U \Rightarrow^{*} t$.

Bien entendu on peut toujours se ramener à ce cas $|Gi|$. Il faut toutefois que le langage défini par la grammaire ne contienne pas la chaîne vide. Les restrictions (1) et (2) confèrent deux propriétés intéressantes à la relation => :

- la relation $\Rightarrow^*$ devient une relation d'ordre partiel. En effet, par définition elle était déjà réflexive et transitive et des restrictions (1), (2) on déduit que $x\Rightarrow^* y$ et $y\Rightarrow^* x$ entraîne $x=y$ pour tout $x,y\in(V_T\cup V_N)^*$, c'est-à-dire que la relation $\Rightarrow^*$ est aussi antisymétrique.

- la deuxième propriété est exprimée dans le lemme qui suit :

LEMME 1.5.1 Si l'on désigne par g le nombre de règles de la grammaire considérée G alors, pour tout couple de chaînes $x,y\in(V_T\cup V_N)^*$ et tout entier $n>o$ tels que $x\Rightarrow^n y$, on a la relation :

$$n\leqslant 2g|y|-g$$

DEMONSTRATION Si $|y|=1$ alors, du fait de la restriction (2), $|x|=1$ et, du fait de la restriction (1), $n\leqslant g$. Le théorème est donc vrai pour $|y|=1$; supposons le vrai pour $|y|<p$ et démontrons qu'il est vrai pour $|y|=p$. Si $x\Rightarrow^n y$, avec $|y|=p$, soit i le plus grand entier tel que $o\leqslant i\leqslant n$ et tel qu'il existe une chaîne u avec $|u|>1$ et

$$x\Rightarrow^{n-i} u\Rightarrow^{i} y$$

Du fait des restrictions (1), (2) on a toujours

$$n-i\leqslant g$$

Du fait que $|u|>1$ il existe $u_1\neq\Lambda$ et $u_2\neq\Lambda$ tel que $u=u_1u_2$. D'après le lemme 1.2.1 il existe donc des entiers k et ℓ et des chaînes y_1,y_2 tels que $y=y_1y_2$, $k+\ell=i$, $u_1\Rightarrow^k y_1$, $u_2\Rightarrow^\ell y_2$. On a $|y_1|<p$, $|y_2|<p$ et donc par supposition

$$k\leqslant 2g|y_1|-g,\quad \ell\leqslant 2g|y_2|-g \text{ d'où } i\leqslant 2g|y|-2g$$

On en déduit que

$$n\leqslant 2g|y|-g$$

A partir de maintenant nous utiliserons le terme de grammaire pour désigner une grammaire "context-free" satisfaisant aux restrictions (1), (2), (3) et (4).

1.6 Relations α,γ,δ

Pour toute grammaire G définissons trois relations binaires sur son vocabulaire $V_T \cup V_N$. Nous les noterons α pour "adjacent", γ pour "gauche" et δ pour "droite". Il est à noter que ces relations sont définies sur un ensemble fini.

DEFINITION Pour tout $A,B \in V_T \cup V_N$:

$A\alpha B \equiv$ [il existe $U \in V_N$ et $x,y \in (V_T \cup V_N)^*$ tels que $U \to xABy$]

$A\gamma B \equiv$ [il existe $y \in (V_T \cup V_N)^*$ tel que $A \to By$]

$A\delta B \equiv$ [il existe $x \in (V_T \cup V_N)^*$ tel que $B \to xA$]

Il faut noter la dissymétrie entre γ et δ . Ces relations α,γ,δ ont pour principale propriété :

LEMME 1.6.1 Une condition nécessaire et suffisante pour que dans une grammaire G il existe une phrase dont la structure soit de la forme xAuvBy avec $x,y \in (V_T \cup V_N \cup V_{\lceil} \cup V_{\rceil})^*$, $A,B \in V_T \cup V_N, u \in V_{\rceil}^*$, $|u|=i$, $v \in V_{\lceil}^*$ $|v|=j$ est que $A\delta^i\alpha\gamma^j B$.

DEMONSTRATION La condition est suffisante. Si $A\delta^i\alpha\gamma^j B$ alors il existe $U,V \in V_T \cup V_N$ tels que $A\delta^i U$, $U\alpha V$, $V\gamma^j B$. Il existe donc une règle de G de la forme $Z \to \ldots UV \ldots [*]$. D'après la restriction (3) sur les grammaires au paragraphe 1.5 on a $S \Rightarrow^* \ldots Z \ldots$ dans G et donc aussi $S \Rightarrow^* \ldots Z \ldots$ dans la grammaire parenthésée G'. On en conclut que $S \Rightarrow^* \ldots UV \ldots$ dans G'. Du fait que $A\delta^i U$ et $V\gamma^j B$ on a $U \Rightarrow^i \ldots Au$, $V \Rightarrow^j vB \ldots$ dans G' avec $u \in V_{\rceil}^*$, $|u|=i$ et $v \in V_{\lceil}^*$, $|v|=j$. De ce fait $S \Rightarrow^* \ldots AuvB \ldots$ dans G'.

La condition est nécessaire. Soit n l'entier tel que $S \Rightarrow^n xAuvBy$ dans la grammaire parenthésée G'. Si n=1 le théorème est vrai car $|u|=|v|=0$ et $A\delta^0\alpha\gamma^0 B$. Supposons le théorème vrai pour n et démontrons qu'il est vrai n+1. Si $S \Rightarrow^{n+1} xAuvBy$ dans G' il existe donc une chaîne s telle que $S \Rightarrow^n s \Rightarrow xAuvBy$ dans G'. Soit $Z \to [z]^Z_Z$ la règle utilisée pour transformer s en xAuvBy. En remarquant que $z \neq \Lambda$ du fait de la

---

|*| Pour représenter une chaîne commençant par un symbole X (respectivement se terminant par un symbole X) on note X... (respectivement ...X).

restriction (2) sur les grammaires au paragraphe 1.5, la transformation de s en xAuvBy ne peut se faire que de quatre façons :

(1) $s=x_1Zx_2AuvBy=>x_1\underset{Z}{\lceil}z\underset{Z}{\rceil}x_2\ AuvBy=xAuvBy$ avec $x_1,x_2\in(V_T\cup V_N\cup V_{\lceil}\cup V_{\rfloor})^*$.

Le théorème étant supposé vrai pour n il est vrai pour s et donc $A\delta^i\alpha\gamma^jB$.

(2) $s=xAuvBy_1Zy_2=>xAuvBy_1\underset{Z}{\lceil}z\underset{Z}{\rceil}y_2=xAuvBy$ avec $y_1,y_2\in(V_T\cup V_N\cup V_{\lceil}\cup V_{\rfloor})^*$.

Même démonstration que dans le cas (1).

(3) $s=x_1Zx_2vBy=>x_1\underset{Z}{\lfloor}z\underset{Z}{\rfloor}x_2vBy=xAuvBy$ avec $x_1\in(V_T\cup V_N\cup V_{\lceil}\cup V_{\rfloor})^*$ et $x_2\in V_{\rfloor}^*$.

Le théorème étant supposé vrai pour n il est vrai pour s et $Z\delta^k\alpha\gamma^jB$ avec $k=|x_2|$. Du fait que $u=\underset{Z}{\rfloor}x_2$ on a i=k+1 et du fait que $A\delta Z$ on a $A\delta^{k+1}\alpha\gamma^jB$ donc $A\delta^i\alpha\gamma^jB$.

(4) $s=xAuy_1Zy_2=>xAuy_1\underset{Z}{\lceil}z\underset{Z}{\rceil}y_2=xAuvBy$ avec $y_1\in V_{\lceil}^*$ et $y_2\in(V_T\cup V_N\cup V_{\lceil}\cup V_{\rfloor})^*$.

La démonstration est analogue à celle du cas (3).

Signalons que ce lemme permet d'énumérer tous les couples de symboles adjacents A,B pouvant apparaître dans une phrase d'une grammaire G : en effet, déterminer s'il existe des chaînes x,y telles que $s=>^*xABy$ revient à savoir si $A\delta^*\alpha\gamma^*B$.

1.7 Problème de l'analyse syntaxique.

Etant donné une grammaire G de vocabulaire $V_T\cup V_N$, analyser une chaîne $s\in(V_T\cup V_N)^*$ consistera à reconnaître si cette chaîne est une phrase de G et, dans l'affirmative, d'en donner chacune de ses structures. Par algorithme d'analyse syntaxique ou analyseur nous entendrons un processus permettant d'analyser toute chaîne terminale de la grammaire G.

En fait l'analyseur ne fournit pas chaque structure sous forme explicite mais sous forme d'une dérivation $\Delta=(S=x_o;i_1,x_1;\ldots;i_n,x_n=s)$ l'engendrant (revoir au paragraphe 1.4 la définition de la structure engendrée par une dérivation). Suivant

que la suite $x_o, x_1, \ldots, x_n$ est fournie dans le sens croissant ou décroissant des indices, on parle d'analyse descendante ou ascendante, l'axiome $S=x_o$ étant considéré en quelque sorte comme un sommet. Trois remarques essentielles s'imposent au sujet des algorithmes d'analyse syntaxique les plus connus :

- Tout d'abord l'ordre dans lequel sont appliquées les règles pour construire la dérivation Δ est imposé d'avance ; plus précisément, si l'algorithme est descendant, à chaque étape, la transformation de x_i en x_{i+1} est faite en appliquant la règle au symbole non terminal se trouvant placé le plus à gauche dans x_i, tandis que si l'algorithme est ascendant la transformation de x_i en x_{i-1} est faite de telle façon que pour réobtenir x_i à partir de x_{i-1} il faudrait appliquer la règle au symbole non terminal placé le plus à droite dans x_{i-1}.

- Si l'algorithme est général c'est-à-dire s'il s'applique à n'importe quel langage "context-free", alors il procède toujours par essais successifs. Ces essais sont soit faits en séquence (algorithme d'Irons [Ir.1] [Ir.2]) soit en parallèle (algorithme de Cock [*]). Les algorithmes qui ne sont pas généraux (algorithme de Floyd décrit en [F1.2] ou de Knuth [Kn] par exemple) peuvent facilement être rendus généraux en leur adjoignant un dispositif permettant de procéder par essais successifs. A ce sujet signalons les travaux de Floyd [F1.3] sur une façon systématique de programmer les algorithmes nécessitant des essais successifs.

- A chaque grammaire et à chacun de ces algorithmes peut être associé un automate "pushdown" classique c'est-à-dire un automate à une pile qui, d'une part, permet de reconnaître les chaînes du langage défini par cette grammaire et d'autre part, est ou n'est pas déterministe suivant que l'algorithme considéré peut ou ne peut pas se passer d'essais successifs pour analyser toutes les chaînes du langage. De ceci il découle que les classes de langages pour lesquelles ces algorithmes n'ont

---

[*] Cet algorithme ainsi que d'autres est décrit en [BC].

pas besoin d'essais successifs sont incluses dans la classe des langages "déterministes", c'est-à-dire, l'ensemble des langages qui peuvent être acceptés par des automates "pushdown" déterministes. Cette inclusion n'est pas nécessairement stricte, c'est le cas de l'analyseur proposé par D.E. Knuth [Kn]. Pour le lecteur qui n'est pas familiarisé avec les automates "pushdown" et les langages déterministes nous renvoyons aux références [GL], [Gi] et [GG].

Signalons qu'une très bonne étude comparative des algorithmes d'analyse connus a été faite par T.V. Griffitiths et S.R. Petrick [GP]. Chaque algorithme y est décrit au moyen d'un automate à deux piles plus simple que l'automate à une pile. Nous utiliserons un automate à deux piles très analogue, pour décrire nos deux principaux analyseurs qui seront de type ascendant. Deux particularités les concernant doivent être mentionnées.

- L'ordre d'application des règles ne correspond pas à celui mentionné plus haut et même, dans le cas du premier algorithme, cet ordre n'est pas imposé d'avance ; ce degré de liberté permet d'éviter certains essais.

- Pour décrire le premier algorithme il faut nécessairement disposer de deux piles.

Décrivons maintenant l'automate à deux piles que nous utiliserons.

1.8 Automate à deux piles

DEFINITION Un automate à deux piles sera défini par :

- un vocabulaire W
- un premier sous-ensemble de $W^* \times W^*$ dont chaque élément sera appelé configuration initiale,
- un deuxième sous-ensemble de $W^* \times W^*$ dont chaque élément sera appelé configuration finale,
- une relation binaire finie sur $W^* \times W^*$ notée $\rightarrow$ dont les éléments seront appelés instructions.

Tout élément (x,y) de $W^* \times W^*$ sera appelé configuration. Physiquement l'automate comprend deux piles et nous dirons qu'elles ont la configuration (x,y) si les chaînes x et y sont disposées ainsi dans ces deux piles :

x	y

Le fonctionnement de l'automate est le suivant :

- les deux piles ont au départ une configuration initiale.
- si à un moment donné la configuration des deux piles est :

u_1x_1	x_2u_2

u_1, u_2, x_1, x_2 étant des chaînes quelconques de W^*, l'instruction

$$(x_1, x_2) \rightarrow (y_1, y_2)$$

si elle existe, s'exécutera et transformera cette configuration en :

u_1y_1	y_2u_2

- Le caractère non déterministe de l'automate se traduit par le fait que pour une configuration donnée il peut y avoir plusieurs instructions susceptibles d'être exécutées. Dans ce cas on peut considérer que l'automate se scinde en plusieurs "sous-automates" qui fonctionnent alors en parallèle.
- L'automate ou le sous-automate s'arrête de fonctionner quand il n'existe plus d'instruction correspondant à la configuration de ses deux piles ou que cette configuration est une configuration finale.

Précisons le caractère déterministe d'un automate à deux piles :

DEFINITION Un automate à deux piles de vocabulaire W sera considéré comme non déterministe si et seulement si il comporte deux instructions distinctes

$$(x_1,y_1)\rightarrow(x_1',y_1') \text{ et } (x_2,y_2)\rightarrow(x_2',y_2')$$

pour lesquelles il existe des chaînes $u,v\in V^*$ telles que

$$ux_1=ux_2 \text{ et } y_1v=y_2v$$

Définissons la notion de cheminement :

DEFINITION Nous appellerons cheminement d'un automate toute suite $C_o,C_1,\ldots,C_n$ de configurations telle que :

(1) C_o soit une configuration initiale
(2) C_n soit une configuration finale ou une configuration à laquelle ne s'applique aucune instruction de l'automate,
(3) si $n\neq o$ alors pour tout $i\in(1,2,\ldots,n)$ le passage de la configuration C_{i-1} à la configuration C_i se fasse en exécutant une instruction de l'automate.

Considérons par exemple, l'automate à deux piles de vocabulaire {a,b,#} défini ainsi :

#	x#	Configurations initiales x étant une chaîne quelconque sur {a,b}
#	#	Configuration finale

Instructions

(#,a)→(#a,Λ)
(#,b)→(#b,Λ)
(a,a)→(aa,Λ)
(b,b)→(bb,Λ)
(a,b)→(Λ,Λ)
(b,a)→(Λ,Λ)

Cet automate est déterministe et permet de reconnaître si la chaîne x contient ou ne contient pas autant de "a" que de "b". En effet, le cheminement ne se termine par la configuration finale que si l'on est dans la première éventualité. Voici deux exemples de cheminement :

(0)	#	aababb#	(0)	#	bababb#
(1)	#a	ababb#	(1)	#b	ababb#
(2)	#aa	babb#	(2)	#	babb#
(3)	#a	abb#	(3)	#b	abb#
(4)	#aa	bb#	(4)	#	bb#
(5)	#a	b#	(5)	#b	b#
(6)	#	#	(6)	#bb	#

CHAPITRE 2 : Précédence totale

2.1 Analyseur et relations de précédence totale : définition

Considérons une grammaire G, trois relations binaires quelconques $\lessdot$, $\doteq$, $\gtrdot$ définies sur son vocabulaire $V_T \cup V_N$, et l'automate à deux piles de vocabulaire $V_T \cup V_N \cup \{\#, [,]\}$, avec $(V_T \cup V_N) \cap \{\#, [,]\} = \emptyset$, défini ainsi :

		Configurations initiales
#	x #	x étant une chaîne quelconque de $(V_T \cup V_N)^*$
		Configuration finale
#	S #	S étant l'axiome de G
		Instructions

Pour tout $A,B \in V_T \cup V_N \cup \{\#\}$ et toute règle $U \to u$ de G on a :

(1) $(A,B) \to (A[B, \Lambda)$ si et seulement si $A \lessdot B$ ou $A = \#$ et $B \neq \#$

(2) $(A,B) \to (AB, \Lambda)$ si et seulement si $A \doteq B$

(3) $(A,B) \to (A], B)$ si et seulement si $A \gtrdot B$ ou $A \neq \#$ et $B = \#$

(4) $([u], \Lambda) \to (\Lambda, U)$

De la façon dont est défini cet automate il découle que s'il existe un cheminement se terminant par la configuration finale alors x est une phrase de la grammaire G. De plus, de ce cheminement on déduit directement une dérivation engendrant une structure de x. Nous considérons donc que cet automate est un analyseur pour la grammaire G, si, à chaque structure d'une phrase terminale x quelconque correspond au moins un cheminement se terminant par la configuration finale.

DEFINITION Si l'automate précédemment décrit est un analyseur pour la grammaire G, alors cet analyseur ainsi que les relations associées $\lessdot, \doteq, \gtrdot$ seront dits de précédence totale.

Bien entendu, pour une grammaire donnée G il existe toujours un analyseur de précédence totale ; il suffit, par exemple de prendre $A \lessdot B$, $A \doteq B$, $A \gtrdot B$ quelque soit $A,B \in V_T \cup V_N$. Donnons deux exemples de relations de précédence totale :

Soit le langage constitué uniquement de la chaîne "aba" défini par la grammaire :

G_2 : S→AB
A→ab
B→a

Considérons les relations $\lessdot, \doteq, \gtrdot$ définies par le tableau

	S	A	B	a	b
S	·	·	·	·	·
A	·	·	$\doteq$	$\lessdot$	·
B	·	·	·	·	·
a	·	·	·	·	$\doteq$
b	·	·	·	$\gtrdot$	·

sur lequel figurent, à l'intersection de la $i^{ème}$ ligne et de la $j^{ème}$ colonne, la ou les relations éventuellement vérifiées entre le $i^{ème}$ et le $j^{ème}$ symbole. Si nous considérons l'automate associé à cette grammaire et à ces relations, alors le cheminement suivant correspond à la structure $\underset{SA}{[}\underset{AB}{[}ab\underset{AB}{]}\underset{BS}{[}a\underset{BS}{]}\underset{SA}{]}$ de la phrase terminale "aba" :

(0) # ⋖ aba#
(1) # ⋖ a ⋖ ba#
(2) # ⋖ ab ⋖ a#
(3) # ⋖ ab ⋗ a#
(4) # ⋖ Aa#
(5) # ⋖ A ⋖ a#
(6) # ⋖ A ⋖ a ⋖
(7) # ⋖ A ⋖ a ⋗ #
(8) # ⋖ A ⋖ B#
(9) # ⋖ AB ⋖ #
(10) # ⋖ AB ⋗ #
(11) # ⋖ S#

La chaîne "aba" n'ayant qu'une seule structure, les relations $\lessdot, \doteq, \gtrdot$ du tableau précédent sont donc de précédence totale pour la grammaire G_2. Considérons maintenant le même langage défini par la grammaire ambiguë :

G_3 : S→aba
S→AB
A→ab
B→a

Les relations $<,=,>$ du tableau qui suit sont de précédence totale pour G_2.

	S	A	B	a	b
S	.	.	.	.	.
A	.	.	=	<	.
B	.	.	.	.	.
a	.	.	.	.	=
b	.	.	.	$\overset{\cdot}{=}$	.

En effet, aux deux structures $\underset{S}{[}\underset{S}{aba]}$ et $\underset{SA}{[}\underset{AB}{[}\underset{BS}{ab\ a]]}$ de la chaîne "aba" correspondent les deux cheminements :

(0) # aba#
(1) # [a ba#
(2) # [ab a#
(3a) # aba #
(4a) # [aba] #
(5a) # S#

(3b) # ab a#
(4b) # Aa#
(5b) # [A a#
(6b) # [A] a #
(7b) # [A] a] #
(8b) # A B#
(9b) # AB #
(10b) # [AB] #
(11b) # S#

Il faut remarquer que dans ces deux exemples il n'existe pas de cheminement "parasite" c'est-à-dire de cheminement ne se terminant pas par la configuration finale. On ne pourra pas toujours se ramener à ce cas. De plus, alors que l'automate correspondant au premier exemple est déterministe, celui du deuxième exemple ne l'est pas du fait des deux instructions $(b,a)\rightarrow(ba,\Lambda)$ et $(b,a)\rightarrow(b],a)$ provenant des relations $b=a$ et $b\gtrdot a$.

2.2 Une classe simple de relations de précédence totale

Pour une grammaire donnée, il existe en général plusieurs analyseurs de précédence totale et à chacun sont associées d'une façon biunivoque des relations de précédence totale.

En utilisant le théorème qui suit il est facile d'énumérer tous les analyseurs de précédence totale capables d'analyser non seulement les chaînes terminales d'une grammaire donnée, mais aussi toutes les autres chaînes construites sur son vocabulaire $V_T \cup V_N$.

THEOREME 2.2.1 Une condition nécessaire et suffisante pour que trois relations binaires $\lessdot, \doteq, \gtrdot$ sur le vocabulaire $V_T \cup V_N$ d'une grammaire G soient de précédence totale et que l'analyseur associé à ces relations puisse analyser toutes les chaînes construites sur $V_T \cup V_N$, est que chacune des inclusions suivantes soit vérifiée :

$$\alpha \subset \doteq, \quad \alpha\gamma^+ \subset \lessdot, \quad \delta^+\alpha \subset \gtrdot, \quad \delta^+\alpha\gamma^+ \subset \ (\lessdot \cup \doteq \cup \gtrdot)$$

(Les relations α, γ, δ sont celles qui sont définies au paragraphe 1.6.)

DEMONSTRATION Démontrons que la condition est nécessaire. Supposons que les relations $\lessdot, \doteq, \gtrdot$ sont de précédence totale et considérons l'analyseur qui leur est associé ainsi que deux symboles quelconques $A, B \in V_T \cup V_N$. D'après le lemme 1.6.1, si $A\delta^i\alpha\gamma^i B$ alors il existe une phrase s de structure xAuvBy avec $x, y \in (V_T \cup V_N \cup V_{[} \cup V_{]})^*$, $u \in V_{\lceil}^*$, $|u|=i$, $v \in V_{\rfloor}^*$, $|v|=j$.

- Si $A\alpha B$ alors $A\delta^i\alpha\gamma^j B$ avec $i=o$ et $j=o$ et de ce fait $u=v=\Lambda$. L'analyseur devra donc réduire A et B en même temps dans la phrase s et donc $A \doteq B$. On en conclut que $\alpha \subset \doteq$.
- Si $A\alpha\gamma^+ B$ alors $A\delta^i\alpha\gamma^j B$ avec $i=o$ et $j>o$ et de ce fait $u=\Lambda$ et $v \neq \Lambda$. L'analyseur devra donc réduire B avant A dans la phrase s et donc $A \lessdot B$.
  On en conclut que $\alpha\gamma^+ \subset \lessdot$.
- Si $A\delta^+\alpha B$ alors $A\delta^i\alpha\gamma^j B$ avec $i>o$ et $j=o$ et de ce fait $u \neq \Lambda$ et $v=\Lambda$.
  L'analyseur devra donc réduire A avant B dans la phrase s et donc $A \gtrdot B$.
  On en conclut que $\delta^+\alpha \subset \gtrdot$.
- Si $A\delta^+\alpha\gamma^+ B$ alors $A\delta^i\alpha\gamma^j B$ avec $i>o$ et $j>o$ et de ce fait $u \neq \Lambda$ et $v \neq \Lambda$.
  L'analyseur devra donc réduire soit B avant A, soit A avant B et donc $A \lessdot B$ ou $A \gtrdot B$.
  On en conclut que $\delta^+\alpha\gamma^+ \subset (\lessdot \cup \gtrdot)$.

Démontrons que la condition est suffisante. Considérons une phrase quelconque $x \neq S$ de G et une de ses structures s. On peut toujours poser :

$$x = X_1 X_2 \ldots X_n \text{ et } s = v_1 X_1 u_1 v_2 X_2 u_2 \ldots v_n X_n u_n$$

avec $n>o$ et pour $i=1,2,\ldots,n$

$$X_i \in V_T \cup V_N,\ u_i \in V^*\ ,\ v_i \in V^*$$

et de plus :

$$v_1 \neq \Lambda \text{ et } u_n \neq \Lambda$$

Pour démontrer que la condition est suffisante, compte tenu de la définition d'un analyseur de précédence totale (paragraphe 2.1) et du lemme 1.3.1., il suffit de montrer qu'il existe deux entiers p et q qui satisfont aux quatre conditions :

(a) $1 \leqslant p \leqslant q \leqslant n$

(b) $u_q \neq \Lambda$ et si $q \neq n$ alors $X_q \gtrdot X_{q+1}$

(c) si $p \neq q$ alors pour $i=p+1,p+2,\ldots,q$, $u_{i-1} v_i = \Lambda$ et $X_{i-1} \doteq X_i$

(d) $v_p \neq \Lambda$ et si $p \neq 1$ alors $X_{p-1} \lessdot X_p$ et pour $i=2,3,\ldots,p$, $X_{i-1} \lessdot X_i$ ou $X_{i-1} \doteq X_i$

- Soit q le plus petit entier tel que $1 \leqslant q \leqslant n$ et tel que la condition (b) soit satisfaite. Cet entier existe car pour $q=n$ la condition (b) est vérifiée.
- Soit p le plus petit entier tel que $1 \leqslant p \leqslant q$ et tel que la condition (c) soit satisfaite. Cet entier existe car pour $p=q$ la condition (c) est vérifiée.

Les entiers p et q satisfont aux conditions (a), (b), (c) ; il ne reste plus qu'à démontrer qu'ils satisfont aussi à la condition (d). Si $p=1$ la condition (d) est satisfaite sinon considérons un entier quelconque i tel que $1 \leqslant i \leqslant p$. Quatre cas se présentent et dans chaque cas nous nous servirons de la condition nécessaire du lemme 1.6.1 :

(1) $u_{i-1}=\Lambda$, $v_i=\Lambda$. Donc $X_{i-1}\alpha X_i$ et du fait que $\alpha\doteq$, $X_{i-1}\doteq X_i$. Le cas où i=p est exclu car il contredit la définition de p.

(2) $u_{i-1}=\Lambda$, $v_i\neq\Lambda$. Donc $X_{i-1}\alpha\gamma^+X_i$ et du fait que $\alpha\gamma^+\subset\leqslant$, $X_{i-1}\leqslant X_i$.

(3) $u_{i-1}\neq\Lambda$, $v_i=\Lambda$. Donc $X_{i-1}\delta^+\alpha X_i$ et du fait que $\delta^+\alpha\subset\geqslant$, $X_{i-1}\geqslant X_i$. Ce cas est exclus car il contredit la définition de q.

(4) $u_{i-1}\neq\Lambda$, $v_i\neq\Lambda$. Donc $X_{i-1}\delta^+\alpha\gamma^+X_i$ et du fait que $\delta^+\alpha\gamma^+\subset(\lessgtr)$, $X_{i-1}\lessgtr X_i$ où $X_{i-1}\gtrless X_i$. Le cas où $X_{i-1}\gtrless X_i$ est exclus car il contredit la définition de q.

La condition d est donc satisfaite.

2.3 Analyseur de précédence totale déterministe

Le cas où l'analyseur de précédence totale est déterministe est particulièrement intéressant du fait de la simplicité de l'algorithme d'analyse qui lui correspond. Un autre point important est la "rapidité" d'un tel analyseur. Attachons un temps d'exécution à chaque instruction de l'automate que constitue l'analyseur. On a alors la propriété suivante :

THEOREME 2.3.1 L'analyse d'une chaîne terminale t par un analyseur de précédence totale déterministe se fait en un temps au plus proportionnel à la longueur $|t|$ de cette chaîne. Plus exactement, en désignant par N le nombre d'instructions exécutées par l'automate et par g le nombre de règles de la grammaire considérée, on a $N\leqslant(6g+1)|t|$.

DEMONSTRATION Reportons nous à la description d'un analyseur de précédence totale au paragraphe 2.1. Soit n le nombre d'instructions du type (4) exécutées par l'automate pour analyser t et soit p le nombre de symboles différents de # contenus dans la pile de droite à sa dernière configuration. Le nombre d'instructions du type (3) exécutées est au plus égal à n+1 et du fait que le nombre de symboles qui sont passés de la pile droite dans la pile gauche est $|t|+n-p$, le nombre d'instructions du type (1) ou (2) exécutées est $|t|+n-p$. Au total on a donc $N\leqslant 3n+|t|+1-p$, d'où $N\leqslant 3n+|t|+1$. D'après le lemme 1.5.1, on a $n\leqslant 2g|t|-g$ et donc $N\leqslant(6g+1)|t|$. En majorant le temps d'exécution de chaque instruction par celui de l'instruction la plus longue à s'exécuter, le premier point du thèorème se trouve démontré.

Mentionnons un sous-produit intéressant de cette démonstration : si t est une phrase, alors, pour l'analyser, l'automate exécutera exactement $3n+|t|-1$ instructions, n étant l'entier tel que $S \Rightarrow^n t$.

Comment reconnaître si un analyseur de précédence totale est déterministe ?
En se reportant à la définition d'un automate à deux piles déterministe (paragraphe 1.8) et à la description d'un analyseur de précédence totale (paragraphe 2.1) on conclut immédiatement :

THEOREME 2.3.2 Une condition nécessaire et suffisante pour qu'un analyseur de précédence totale défini dans une grammaire G soit déterministe est que :

(1) toutes les règles de G aient des membres droits distincts, c'est-à-dire que $X \to u$ et $Y \to u$ entraîne $X=Y$,

(2) les relations de précédence totale $\lessdot, \doteq, \gtrdot$ associées à cet analyseur soient incompatibles deux à deux, c'est-à-dire que $(\lessdot \cap \doteq) \cup (\gtrdot \cap \doteq) \cup (\lessdot \cap \gtrdot) = \phi$.

Nous allons étudier sous quelles conditions il est possible de définir des relations de précédence totale incompatibles deux à deux dans une grammaire donnée.

2.4 Existence de relations de précédence totale incompatibles deux à deux

Le théorème qui suit permet de décider si dans une grammaire donnée il existe des relations de précédence totale incompatibles deux à deux.

THEOREME 2.4.1 Une condition nécessaire et suffisante pour qu'il existe des relations de précédence incompatibles deux à deux pour une grammaire G, dans laquelle ont été définies les relations α, γ, δ, est que

$(\delta^+\alpha\cap\alpha)\cup(\alpha\gamma^+\cap\alpha)\cup(\delta^+\alpha\gamma^+\cap\alpha)\cup(\delta^+\alpha\cap\alpha\gamma^+) = \phi$.

DEMONSTRATION Démontrons par l'absurde que la condition est nécessaire. Supposons que dans la grammaire G il existe les relations $\lessdot,\doteq,\gtrdot$ incompatibles deux à deux et que $(\delta^+\alpha\cap\alpha)\cup(\alpha\gamma^+\cap\alpha)\cup(\delta^+\alpha\gamma^+\cap\alpha)\cup(\delta^+\alpha\cap\alpha\gamma^+) \neq \phi$.

De ce fait il existe deux entiers i,j et deux symboles $A,B\in V_T\cup V_N$ tels que l'une des deux éventualités suivantes se présente :

(1) $A\alpha B$ et $A\delta^i\alpha\gamma^jB$ avec $i>o$, $i+j>o$, $j>o$

(2) $A\delta^i\alpha B$ et $A\alpha\gamma^jB$ avec $i>o$, $j>o$

Eventualité (1)

D'après le lemme 1.6.1, il existe donc deux phrases s_1 et s_2 de structures respectives

$\ldots AB\ldots$ et $\ldots Au_1v_1B\ldots$ avec $u_1\in V_{\rfloor}^*$, $|u_1|=i$, $v_1\in V_{\lfloor}^*$, $|v_1|=j$.

Du fait de la restriction (4) du paragraphe 1.5 il existe donc deux phrases terminales t_1 et t_2 de structures respectives

$\ldots au_2v_2b\ldots$ et $\ldots au_2u_1v_1v_2b\ldots$ avec $a,b\notin V_T$, $u_2\in V_{\rfloor}^*$, $v_2\in V_{\lfloor}^*$,

$u_2=\ldots_A$ ou $u_2=\Lambda$ et $A=a$, $v_2=_B\ldots$ ou $v_2=\Lambda$ et $B=b$.

Au cours de l'analyse de t_1 l'analyseur traitera une phrase de structure $\ldots AB\ldots$. Les relations $\lessdot,\doteq,\gtrdot$ étant incompatibles deux à deux, l'analyseur sera donc amené, au cours de l'analyse de t_2, à traiter une phrase de structure $..Au_1v_1B\ldots$. Il devra donc, d'une part réduire A et B en même temps et d'autre part réduire A avant B ou B avant A, car $u_1v_1\neq\Lambda$ du fait que $i+j>o$. Par conséquent $A\doteq B$ et $A\lessdot B$ ou $A\doteq B$ et $A\gtrdot B$ ce qui contredit le fait que les relations $\lessdot$, $\doteq$, $\gtrdot$ sont incompatibles deux à deux.

Eventualité (2)

D'après le lemme 1.6.1 il existe donc deux phrases s_1 et s_2 de structures respectives

$$\ldots Au_1B\ldots \text{ et } \ldots Av_1B\ldots \text{ avec } u_1 \in V_{\rfloor}^*, |u_1|=i,\ v_1 \in V_{\lfloor}^*, |v_1|=j.$$

Du fait de la restriction (4) du paragraphe 1.5 il existe donc deux phrases terminales t_1 et t_2 de structures respectives

$$\ldots au_2u_1v_2b\ldots \text{ et } \ldots au_2v_1v_2b\ldots \text{ avec } a,b \in V_T,\ u_2 \in V^*,\ v_2 \in V^*,$$

$u_2=\ldots\rfloor_A$ ou $u_2=\Lambda$ et $A=a$, $v_2=\lfloor_B\ldots$ ou $v_2=\Lambda$ et $B=b$.

Les relations $\lessdot,\doteq,\gtrdot$ étant incompatibles deux à deux, au cours des analyses respectives de t_1 et t_2, l'analyseur sera amené à traiter des phrases ayant respectivement pour structures $\ldots Au_1B\ldots$ et $\ldots Av_1B\ldots$. Du fait que $u_1 \neq \Lambda$ et $v_1 \neq \Lambda$ car $i>o$ et $j>o$, il devra donc d'une part réduire A avant B et d'autre part réduire B avant A. Par conséquent $A \gtrdot B$ et $B \lessdot A$ ce qui contredit le fait que les relations $\lessdot,\doteq,\gtrdot$ sont incompatibles deux à deux.

Démontrons que la condition est suffisante. Sur le vocabulaire $V_T \cup V_N$ de G considérons les relations binaires $\lessdot,\doteq,\gtrdot$ ainsi définies :

$$\doteq = \alpha$$
$$\lessdot = \alpha\gamma^+$$
$$\gtrdot = \delta^+\alpha \cup (\delta^+\alpha\gamma^+ \cap \overline{\alpha\gamma^+})$$

Du fait que $\alpha \subset \doteq$, $\alpha\gamma^+ \subset \lessdot$, $\delta^+\alpha \subset \gtrdot$, $\delta^+\alpha\gamma^+ \subset (\lessdot \cup \gtrdot)$, ces relations sont de précédence totale (théorème 2.2.1). On a

$$(\delta^+\alpha \cap \alpha) \cup (\alpha\gamma^+ \cap \alpha) \cup (\delta^+\alpha\gamma^+ \cap \alpha) \cup (\delta^+\alpha \cap \alpha\gamma^+) = (\lessdot \cap \doteq) \cup (\gtrdot \cap \doteq) \cup (\lessdot \cap \gtrdot)$$

Si la condition du théorème est vérifiée on en conclut que ces relations de précédence totale sont incompatibles deux à deux.

2.5 Famille optimale de triplets de relations de précédence totale

DEFINITION Considérons une application qui à toute grammaire G fait correspondre un triplet de relations de précédence totale $(<,\doteq,>)_G$. La famille de triplets ainsi définie sera dite optimale si et seulement si, quelle que soit la grammaire G, on est toujours dans l'un des deux cas :

(1) les relations $<,\doteq,>$ sont incompatibles deux à deux,

(2) les relations $<,\doteq,>$ ne sont pas incompatibles deux à deux, mais on ne peut trouver dans G d'autres relations de précédence totale qui le soient.

THEOREME 2.5.1 Les familles de triplets $(<,\doteq,>)_G$ de relations définies comme suit sont des familles optimales de relations de précédence totale :

$$\doteq = \alpha$$
$$< = \alpha\gamma^+ \cup \rho_1$$
$$> = \delta^+\alpha \cup \rho_2$$

ρ_1 et ρ_2 étant des relations binaires sur le vocabulaire $V_T \cup V_N$ de G satisfaisant aux deux conditions :

$$\rho_1 \cap \rho_2 = \delta^+\alpha\gamma^+ \cap \overline{\delta^+\alpha} \cap \overline{\alpha\gamma^+}$$

$$\rho_1 / \rho_2 = \emptyset$$

De plus l'analyseur associé aux relations $<,\doteq,>$ peut analyser toutes les chaînes construites sur le vocabulaire $V_T \cup V_N$ de la grammaire G.

DEMONSTRATION Du fait que $\alpha \doteq \alpha$, $\alpha\gamma^+ < \alpha$, $\delta^+\alpha > \alpha$, $\delta^+\alpha\gamma^+ > \alpha$ $(< >)$, les relations $<,\doteq,>$ sont de précédence totale et l'analyseur associé est capable d'analyser toutes les chaînes sur $V_T \cup V_N$ (théorème 2.2.1). Du fait que

$$(\doteq \cup <) \cap (\doteq \cup >) \cap (< \cap >) = (\alpha\gamma^+ \cap \alpha) \cup (\delta^+\alpha \cap \alpha) \cup (\delta^+\alpha\gamma^+ \cap \alpha) \cup (\delta^+\alpha \cap \alpha\gamma^+)$$

la famille de triplets $(<,\doteq,>)_G$ est optimale (théorème 2.4.1).

Il est intéressant de remarquer que les relations $\lessdot, \gtrdot$ sont définies ici d'une façon symétrique. Cependant pour éviter certains reculs à l'analyseur associé au triplet $(\lessdot, \doteq, \gtrdot)_G$ on a intérêt à prendre $\rho_1 = \emptyset$. On obtient alors :

$$\doteq = \alpha$$
$$\lessdot = \alpha\gamma^+$$
$$\gtrdot = \delta^+\alpha \cup (\delta^+\alpha\gamma^+ \cap \overline{\alpha\gamma^+})$$

Considérons par exemple le langage dont les chaînes sont de la forme :

$a^i b^j a^j \cup a^i b^i a^j c$ avec $i>o$ et $j>o$,

on peut le définir par la grammaire :

G_4 :	S→X	S→VY
	X→aU	Y→ac
	X→aX	Y→aY
	U→bA	V→aB
	U→bUA	V→aVB
	A→a	B→b

et en utilisant les formules ci-dessus on obtient le tableau [*] :

	S	X	Y	U	V	A	B	a	b	c
S	·	·	·	·	·	·	·	·	·	·
X	·	·	·	·	·	·	·	·	·	·
Y	·	·	·	·	·	·	·	·	·	·
U	·	·	·	·	·	$\doteq$	·	$\lessdot$	·	·
V	·	·	$\doteq$	·	·	·	$\doteq$	$\lessdot$	$\lessdot$	·
A	·	·	·	·	·	$\gtrdot$	·	(>)	·	·
B	·	·	$\gtrdot$	·	·	·	$\gtrdot$	(>)	(>)	·
a	·	$\doteq$	$\doteq$	$\doteq$	$\doteq$	$\gtrdot$	$\doteq$	$\lessdot$	$\lessdot$	$\doteq$
b	·	·	$\gtrdot$	$\doteq$	·	$\doteq$	$\gtrdot$	$\lessdot$	$\lessdot$	·
c	·	·	·	·	·	·	·	·	·	·

[*] En remplaçant par le signe $\lessdot$, un ou plusieurs signes $\gtrdot$ encerclés de ce tableau, on obtient toutes les relations de précédence que l'on peut obtenir en faisant varier ρ_1 et ρ_2 dans les formules du théorème précédent.

Remarquons que toutes les règles de la grammaire G_4 ont des membres droits distincts et que les relations du tableau sont incompatibles deux à deux. De ce fait l'analyseur associé à ces relations est donc déterministe. Voici les principales étapes de l'analyse des chaînes "aaabbaa" et "aabbaaac".

# aaabbaa#	# aabbaaac#
#|a|a|a|b|b|a|a| #	#|a|a|b|b|a|a|ac| #
#|a|a|a|b|b|a| A#	#|a|a|b|b|a|aY| #
#|a|a|a|b|bA| A#	#|a|a|b|b|aX| #
#|a|a|a|b|A| #	#|a|a|b|b| Y#
#|a|a|aX| #	#|a|a|b| BY#
#|a|aX| #	#|a|aB| BY#
#|aX| #	#|aYB| Y#
#|X| #	#|Y| #
# S#	# S#

Signalons, comme nous le verrons dans la démonstration du théorème 2.6.3 que le langage défini par la grammaire G_4 n'est pas reconnaissable par un automate à une pile classique déterministe.

2.6 Grammaires et langages de précédence totale

DEFINITION Nous appellerons grammaire de précédence totale toute grammaire pour laquelle il existe un analyseur de précédence totale déterministe.

Des théorèmes 2.3.2 et 2.4.1 on déduit immédiatement une définition équivalente.

DEFINITION EQUIVALENTE : Une grammaire de précédence totale G est une grammaire qui satisfait aux deux conditions :

(1) Toutes ses règles ont des membres droits distincts,

(2) $(\delta^+\alpha\cap\alpha)\cup(\alpha\gamma^+\cap\alpha)\cup(\delta^+\alpha\gamma^+\cap\alpha)\cup(\delta^+\alpha\cap\alpha\gamma^+) = \emptyset$.

Donnons une propriété essentielle des grammaires de précédence totale.

THEOREME 2.6.1 Une grammaire de précédence totale ne peut être ambiguë.

DEMONSTRATION Considérons une grammaire de précédence totale G. Il existe donc un analyseur de précédence totale déterministe pour G. Si G est ambiguë il existe, par définition, une phrase terminale t de G ayant plusieurs structures et à chacune de celles-ci correspond au moins un cheminement de l'analyseur pour analyser t. Cet analyseur ne peut donc être déterministe et il y a contradiction.

Si nous reconsidérons par exemple la grammaire G_1 de la fin du paragraphe 1.3

G_1 : $S \to XX$
$X \to a$
$X \to Xa$

qui était ambiguë, nous pouvons facilement vérifier qu'effectivement elle n'est pas de précédence totale car, du fait que $X\delta X$ et $X\alpha a$, on a $X\delta\alpha a$ et, du fait que $X\alpha a$, on a $\delta\alpha\cap\alpha\neq\emptyset$.

Nous savons que le problème de déterminer si une grammaire est ou n'est pas ambiguë, ne peut être résolu en général, cependant le théorème 2.6.1 peut contribuer à sa solution dans certains cas particuliers.

Intéressons nous maintenant aux langages que l'on peut définir au moyen d'une grammaire de précédence totale.

DEFINITION Tout langage "context-free" qui peut être défini au moyen d'une grammaire de précédence totale sera dit de précédence totale.

En remarquant la symétrie de la définition équivalente d'une grammaire de précédence totale on déduit cette propriété :

THEOREME 2.6.2 Si L est un langage de précédence totale défini sur le vocabulaire V_T, son réfléchi, qui est le langage $\tilde{L}$ défini par

$$\tilde{L} = \{A_1A_2\ldots A_n\, V_T^* \mid A_1, A_2, \ldots, A_n \in V_T \text{ et } A_nA_{n-1}\ldots A_1 \in L\}$$

est aussi de précédence totale.

Il est intéressant de comparer l'ensemble des langages de précédence totale à un autre sous-ensemble bien connu de langages "context-free", celui des langages déterministes. Rappelons que ces langages sont ceux que l'on sait reconnaître au moyen d'un automate à une pile (pushdown) déterministe. Nous avons obtenu le résultat suivant :

THEOREME 2.6.3 L'ensemble des langages de précédence totale n'est pas inclus dans l'ensemble des langages déterministes et de leurs réfléchis.

DEMONSTRATION Pour cette démonstration nous utiliserons les résultats suivants démontrés par Ginsburg et Greibach en [GG] :

(1) Si D désigne un langage déterministe et R un langage régulier, les trois langages

D/R,

$DR = \{xy \mid x \in D \text{ et } y \in R\}$,

$D/R = \{x \mid \text{il existe } y \in R \text{ avec } xy \in D\}$

sont déterministes.

(2) Le langage $L_5 = \{a^i b^j a^j \cup a^i b^i a^j \mid i>o,\ j>o\}$ n'est pas déterministe. (C'est d'ailleurs un langage intrinsèquement ambigu c'est-à-dire qui ne peut être défini par une grammaire non ambiguë).

Considérons maintenant le langage

$$L_4 = \{a^i b^j a^j \cup a^i b^i a^j c \mid i>o,\ j>o\}$$

défini par la grammaire de précédence totale G_4 du paragraphe 2.5 et le langage

$$L_6 = \{fd^i e^j d^j \cup d^i e^i d^j \mid i>o,\ j>o\}$$

que l'on peut définir par une grammaire de précédence totale G_6 symétrique de la grammaire G_4. Le langage $L_4 \cup L_6$ est de précédence totale. En effet on peut toujours s'arranger pour que les vocabulaires de G_4 et de G_6 soient disjoints et de ce fait il n'y aura pas d'interférence entre les relations de précédence totale de G_4 et celles de G_6 lorsque l'on construira la grammaire définissant le langage $L_4 \cup L_6$. Du fait que

$$L_5 = ((L_4 \cup L_6)c^* \cap a^* b^* a^* c)/c$$

le langage $L_4 \cup L_6$ ne peut être déterministe et du fait de la symétrie de sa définition son réfléchi ne peut l'être non plus.

Nous n'avons malheureusement pas pu trouver une démonstration rigoureuse permettant d'affirmer que l'ensemble des langages de précédence totale n'incluait pas le sous-ensemble de langages déterministes dont les réfléchis sont aussi déterministes. Cependant il semble bien que le langage déterministe :

$$\{a^i b^i \cup c a^i b^{2i} c \mid i>o\}$$

dont le réfléchi est aussi déterministe n'est pas de précédence totale.

En terminant ce chapitre, signalons un langage de précédence totale très simple qui ne semble pas être déterministe. Il s'agit du langage dont chaque chaîne est de longueur impaire et est formée d'une suite de "a" et de "b" telle que le symbole "a" placé le plus à droite se trouve au milieu de celle-ci. Ce langage peut être défini par la grammaire de précédence totale :

S→a

S→aSB

S→bSB

B→b

de laquelle on déduit le tableau suivant en utilisant les formules données au paragraphe 2.5.

	S	B	a	b
S	.	$\doteq$	.	$\lessdot$
B	.	$\gtrdot$	.	$\gtrdot$
a	$\doteq$	$\gtrdot$	$\lessdot$	$\lessdot$
b	$\doteq$	$\gtrdot$	$\lessdot$	$\lessdot$

Pour voir de quelle façon l'analyseur procède dans ce cas fournissons lui par exemple la chaîne "ababb" en configuration initiale. On obtient :

(0) # ababb#

(1) #⌈a babb#

(2) #⌈a⌋b abb#

(3) #⌈a⌋b⌈a bb#

(4) #⌈a⌋b⌈a⌋b b#

(5) #⌈a⌋b⌈a⌋b⌋b #

(6) #⌈a⌋b⌈a⌋b⌋b⌋ #

(7) #⌈a⌋b⌈a⌋b B#

(8) #⌈a⌋b⌈a⌋b⌋ B#

(9) #⌈a⌋b⌈a BB#

(10) #⌈a⌋b⌈a⌋ BB#

(11) #⌈a⌋b SBB#

(12) #⌈a⌋bS BB#

(13) #⌈a⌋bSB B#

(14) #⌈a⌋bSB⌋ B#

(15) #⌈a SB#

(16) #⌈aS B#

(17) #⌈aSB #

(18) #⌈aSB⌋ #

(19) # S#

CHAPITRE 3 : Précédence de gauche à droite

3.1 Analyseurs et relations de précédence de gauche à droite : définition

DEFINITION Etant donné une grammaire G de vocabulaire $V_T \cup V_N$, nous appellerons opérateurs et opérandes les éléments respectifs de deux vocabulaires disjoints V_R et V_E tels que $V_R \cup V_E = V_T \cup V_N$ et choisis de telle façon que :

(1) tout symbole terminal soit un opérateur,
(2) il n'existe pas de règle de G de la forme $U \to xXYy$ avec $x,y \in (V_T \cup V_N)^*$ et X,Y opérandes,
(3) il n'existe pas de règles de G de la forme $U \to V$ avec V opérande.

La condition (3) pourrait aussi s'écrire : tout membre droit de règle contient au moins un opérateur. Bien entendu il est toujours possible de choisir V_R et V_E de façon à ce qu'ils satisfassent aux trois restrictions ; il suffit de prendre par exemple $V_R = V_T \cup V_N$ et $V_E = \emptyset$.

Supposons qu'un tel choix ait été fait et considérons maintenant trois relations binaires $\lessdot, \doteq, \gtrdot$, sur $V_T \cup V_N$ telles que $\lessdot \cup \doteq \cup \gtrdot \subset V_R \times V_R$, et l'automate à deux piles de vocabulaire $V_T \cup V_N \cup \{\#, [,]\}$, avec $(V_T \cup V_N) \cap \{\#, [,]\} = \emptyset$, défini ainsi :

Configuration initiales des deux piles

#	x#

avec $x \in V_T^*$

Configuration finale des deux piles

#	S#

Instructions

Pour tout $A,B \in V_R \cup \{\#\}$, tout $E \in V_E \cup \{\Lambda\}$ et toute règle $U \to u$ de G on a :

(1) $(A,EB)\rightarrow(A\lfloor EB,\Lambda)$ si et seulement si $A\lessdot B$ ou $A=\#$ et $B\neq\#$

(2) $(A,EB)\rightarrow(AEB,\Lambda)$ si et seulement si $A\doteq B$

(3) $(A,EB)\rightarrow(AE\rfloor,B)$ si et seulement si $A\gtrdot B$ ou $A\neq\#$ et $B=\#$

(4) $(\lfloor u\rfloor,B)\rightarrow(\Lambda,UB)$ si et seulement si $U\in V_E$ ou $U\in V_R$ et $B\in V_T\cup\{\#\}$

DEFINITION Si l'automate précédemment décrit est un analyseur pour la grammaire G, alors cet analyseur ainsi que les relations associées $\lessdot,\doteq,\gtrdot$ seront dits de précédence de gauche à droite.

Comme au paragraphe 2.1 nous considérons que cet automate est un analyseur pour la grammaire G si à chaque structure d'une phrase terminale x qui lui est fournie en configuration initiale correspond au moins un cheminement se terminant sur la configuration finale. Il faut remarquer que la forme particulière des instructions du type (4) impose d'appliquer les règles dans un ordre bien précis : entre autre la pile droite ne peut jamais contenir plus de deux symboles non terminaux .

Donnons un exemple d'analyseur de précédence de gauche à droite : soit la grammaire

G_7 : $S\rightarrow a$ $\quad V_T = \{a,b,+,-\}$

$S\rightarrow b$

$S\rightarrow(SRS)$

$R\rightarrow +$ $\quad V_N = \{S,R\}$

$R\rightarrow -$

Choisissons les ensembles d'opérateurs et d'opérandes ainsi

$$V_R = \{a,b,+,-,R\} \qquad V_E = \{S\}$$

Comme on pourrait le vérifier par la suite les relations $\lessdot,\doteq,\gtrdot$ définies par le tableau qui suit sont de précédence de gauche à droite.

	a				
	b	(	)	R	±
ab	·	·	⋗	⋗	⋖
(	⋖	⋖	·	≐	·
)	·	·	⋗	⋗	·
R	⋖	⋖	≐	·	·
+-	⋗	⋗	·	·	⋖

Pour analyser la phrase terminale "(a-(b+a))" l'analyseur de précédence de gauche à droite associé à ces relations passe successivement par les configurations :

(0) # (a-(b+a)) #
(1) # ⌊(a-(b+a)) #
(2) ≠ ⌊(⌊a -(b+a)) #
(3) # ⌈(⌊a⌈- (b+a)) #
(4) # ⌈(⌊a⌊-⌋ (b+a)) #
(5) ≠ ⌈(⌊a R(b+a)) #
(6) ≠ ⌈(⌊a⌋ R(b+a)) ≠
(7) ≠ ⌈(SR(b+a)) #
(8) # ⌊(SR (b+a)) #
(9) ≠ ⌈(SR⌊(b+a)) #
(10) # ⌈(SR⌊(⌊b +a)) #
(11) # ⌈(SR⌈(⌊b⌈+ a)) #
(12) # ⌈(SR⌈(⌊b⌈+⌋ a)) #
(13) # ⌊(SR⌈(⌊b Ra)) #
(14) # ⌈(SR⌈(⌊b⌋ Ra)) #
(15) # ⌈(SR⌈(SRa)) #
(16) # ⌊(SR⌈(SR a)) #
(17) ≠ ⌈(SR⌈(SR⌈a)) ≠
(18) ≠ ⌊(SR⌈(SR⌈a⌋)) ≠
(19) ≠ ⌊(SR⌈(SR S)) ≠
(20) ≠ ⌈(SR⌈(SRS)) ≠
(21) ≠ ⌈(SR⌈(SRS)⌋) ≠
(22) ≠ ⌈(SR S) ≠
(23) ≠ ⌈(SRS) #
(24) ≠ ⌈(SRS)⌋ #
(25) ≠ S #

3.2 Analyseur de précédence de gauche à droite déterministe

Sur l'exemple précédent on peut remarquer que l'analyseur est déterministe. D'une façon générale en revoyant la définition d'un automate à deux piles déterministe (paragraphe 1.8) et en considérant l'automate décrit au début du paragraphe 3.1 on déduit :

THEOREME 3.2.1 Une condition nécessaire et suffisante pour qu'un analyseur de précédence de gauche à droite défini dans une grammaire G soit déterministe est que :

(1) toutes les règles de G aient des membres droits distincts,
(2) les relations de précédence de gauche à droite $\lessdot, \doteq, \gtrdot$ associées à cet analyseur soient incompatibles deux à deux.

Ce résultat est le même que celui concernant les analyseurs de précédence totale déterministes (théorème 2.3.2). En utilisant une démonstration très proche de celle du théorème 2.3.1 qui concerne la rapidité d'un analyseur de précédence totale déterministe, on obtient aussi :

THEOREME 3.2.2 L'analyse d'une chaîne terminale t par un analyseur de précédence de gauche à droite se fait en un temps au plus proportionnel à la longueur $|t|$ de cette chaîne. Plus exactement, en désignant par N le nombre d'instructions exécutées par l'automate et par g le nombre de règles de la grammaire considérée, on a $N \leq (6g+1)|t|$.

Voici maintenant une propriété qui ne concerne que ce type d'analyseur :

THEOREME 3.2.3 Tout langage reconnaissable par un analyseur de précédence de gauche à droite déterministe est aussi reconnaissable par un automate classique à une pile déterministe.

DEMONSTRATION En se reportant à la définition d'une analyseur de précédence de gauche à droite au début du paragraphe 3.1, on peut constater que, du fait des instructions de type (4), la pile droite de l'automate ne contient jamais plus de deux symboles non terminaux. De ce fait il est possible de définir un automate équivalent qui n'empilerait jamais de symboles dans la pile droite ; celle-ci se comporterait donc comme un ruban de lecture. En adjoignant maintenant un nombre fini d'états à ce nouvel automate on peut alors le transformer en un automate qui ne puisse lire qu'un seul symbole à la fois sur le sommet de la pile gauche. Ce dernier automate est donc un automate classique à une pile.

3.3 Une définition équivalente des relations de précédence de gauche à droite

Considérons une grammaire G de vocabulaire $V_T \cup V_N$ dans laquelle on a choisi des ensembles d'opérateurs et d'opérandes V_R et V_E conformément à la définition du début du paragraphe 3.1. Il est possible de donner une définition équivalente de relations de précédence de gauche à droite qui ne fait plus intervenir d'automate. Pour ceci nous aurons besoin de définir une position privilégiée h(x) de chaque chaîne x.

DEFINITION Pour toute chaîne $x \in (V_T \cup V_N)^*$ qui contient au moins un symbole non terminal, h(x) désignera la position du symbole qui, parmi les symboles non terminaux de x non immédiatement précédés d'un opérande, se trouve placé le plus à droite.

Il faut remarquer que la position h(x) est toujours définie à partir du moment ou x contient au moins un symbole non terminal. A partir de h il nous est alors possible de définir la relation $\Rightarrow\!\!\!\!\!\cdot$ qui sera une restriction de la relation $\Rightarrow$:

DEFINITION Pour tout couple de chaîne $x,y \ (V_T \cup V_N)^*$:
$x \Rightarrow\!\!\!\!\!\cdot \ y \equiv [(x;h(x),y)$ est une dérivation dans la grammaire $G]$

Au prochain paragraphe nous verrons quelques propriétés de la relation $\Rightarrow\!\!\!\!\!\cdot$, cependant dès maintenant nous disposons de suffisamment d'éléments pour donner une définition équivalente des relations de précédence de gauche à droite.

DEFINITION EQUIVALENTE Trois relations binaires $\lessdot, \doteq, \gtrdot$ sur $V_T \cup V_N$ telles que $\lessdot \cup \doteq \cup \gtrdot \subset V_R \times V_R$ seront dites de précédence de gauche à droite si et seulement si pour tout $A,B \in V_R$, $U \in V_N$, $E \in V_E \cup \{\Lambda\}$, $x,y,u \in (V_T \cup V_N)^*$ on a :

(1) $U \rightarrow xAEBy$ entraîne $A \doteq B$,

(2) $S \Rightarrow^* xAUy \Rightarrow xAEBuy$ entraîne $A \lessdot B$

(la transformation de xAUy en xAEBuy est supposée faite en appliquant la règle $U \rightarrow EBu$ à l'élément U).

(3) $S \Rightarrow^* xUBy \Rightarrow xuAEBy$ entraîne $A \gtrdot B$

(la transformation de xUBy en xAEBuy est supposée faite en appliquant la règle $U \rightarrow uAE$ à l'élément U).

DEMONSTRATION En faisant intervenir les restrictions habituelles sur les grammaires du paragraphe 1.5 et les propriétés des dérivations énoncées au lemme 1.4.1 et au corollaire 1.4.3, on déduit immédiatement que la condition énoncée dans cette définition équivalente peut aussi s'exprimer ainsi :

Quelle que soit la phrase terminale t et quelle que soit sa structure s, il existe une dérivation engendrant s qui est de la forme :

$$(S=r_0; h(r_1), r_1; \ldots; h(r_{n-1}), r_n=t)$$

avec pour tout $i \in (1,2,\ldots,n)$, tout $U \in V_N$ en position $h(r_{i-1})$ dans r_{i-1}, tout $A,B \in V_R$, tout $E \in V_E \cup \{\Lambda\}$ et tout $x,y,u,v \in (V_T \cup V_N)^*$:

(1) $r_{i-1}=xUy$, $U \rightarrow uAEBv$, $r_i=xuAEBvy$ entraîne $A \doteq B$,

(2) $r_{i-1}=xAUy$, $U \rightarrow EBv$, $r_i=xAEBvy$ entraîne $A \lessdot B$,

(3) $r_{i-1}=xUBy$, $U \rightarrow uAE$, $r_i=xuAEBy$ entraîne $A \gtrdot B$.

Considérons maintenant trois relations binaires quelconques $\lessdot, \doteq, \gtrdot$ sur $V_T \cup V_N$ et associons leur l'automate du paragraphe 3.1 dans lequel on aurait permuté les configurations initiales avec les configurations finales et le membre droit de chaque instruction avec son membre gauche. On a donc l'automate :

Configuration initiale : | # | S# |

Configurations finales : | # | t# | avec $t \in V_t^*$

Instructions : pour tout $A,B \in V_R \cup \{\#\}$, $E \in V_R \cup \{\Lambda\}$ et toute règle $U \rightarrow u$:

(4) $(\Lambda,UB)\rightarrow(\lfloor u\rfloor,B)$ si et seulement si $U\in V_E$ ou $V\in V_R$ et $B\in V_T\cup\{\#\}$

(3) $(AE\rceil,B)\rightarrow(A,EB)$ si et seulement si $A\dot{>}B$ ou $A\neq\#$ et $B=\#$

(2) $(AEB,\Lambda)\rightarrow(A,EB)$ si et seulement si $A\doteq B$

(1) $(A\lfloor EB,\Lambda)\rightarrow(A,EB)$ si et seulement si $A\dot{<}B$ ou $A=\#$ et $B\neq\#$

De cet automate on déduit que la condition énoncée précédemment est une condition nécessaire et suffisante pour que les relations $\dot{<},\doteq,\dot{>}$ soient de précédence de gauche à droite.

3.4 Relation σ

Sur $V_T\cup V_N$ définissons la relation binaire σ ainsi :

DEFINITION Pour tout couple $A,B\in V_T\cup V_N$:

$A\sigma B \equiv \lceil$il existe $x,y\in(V_T\cup V_N)^*$ tels que $S \Rightarrow^* xABy\rceil$

Nous allons essayer d'évaluer σ en fonction des relations α,γ,δ du paragraphe 1.6. Pour ceci nous aurons besoin de deux lemmes.

LEMME 3.4.1 Si $A,B\in V_T\cup V_N$ et $A\sigma B$ alors $A\in V_R$ ou $B\in V_R$

DEMONSTRATION Si $A\sigma B$ alors il existe un entier n et deux chaînes $x,y\in(V_T\cup V_N)^*$ tels que $S \Rightarrow^n xABy$. Raisonnons par récurrence sur n. Si n=1 le théorème est vrai car il ne peut y avoir deux opérandes adjacents dans une règle de la grammaire (début du paragraphe 3.1). Supposons le théorème vrai pour n et démontrons qu'il est vrai pour n+1. Si $S \Rightarrow^{n+1} xABy$ alors il existe $s\in(V_T\cup V_N)^*$ tel que $S \Rightarrow^n s \Rightarrow xABy$. Soit $U\rightarrow u$ la règle utilisée pour transformer s en xABy. Dans la phrase s il ne peut y avoir d'opérande adjacent au symbole U et ceci du fait que s ne contient pas deux opérandes adjacents par supposition et du fait de la définition de la relation $\Rightarrow$. La chaîne u ne contient pas deux opérandes adjacents. Lors de la transformation de s en xABy il ne peut donc se former un couple d'opérandes adjacents.

LEMME 3.4.2 Si $S \Rightarrow^* xABy$ et $AB \Rightarrow u$ avec $A,B \in V_T \cup V_N$ et $u,x,y \in (V_T \cup V_N)^*$ alors il existe $x',y' \in (V_T \cup V_N)^*$ tels que $S \Rightarrow^* x'ABy' \Rightarrow x'uy'$, l'élément auquel est appliquée la règle pour transformer x'ABy' en x'uy' étant soit A soit B.

DEMONSTRATION D'après la restriction (4) du paragraphe 1.5 il existe $t \in V_T^*$ tel que $xABy \not\Rightarrow^* t$ et d'après le corollaire 1.4.3 $xABy \Rightarrow^* t$. Il existe donc $s_o, s_1, \ldots, s_n$ $(V_T \cup V_N)^*$ telsque :

$$xABy = s_o \Rightarrow s_1 \Rightarrow \ldots \Rightarrow s_n = t$$

Soit i le plus grand entier tel que $i \leq n$ et tel que dans la chaîne s_i on n'ait appliqué aucune règle à l'élément A et à l'élément B. On a donc

$$S \Rightarrow^* s_i = x'ABy' \Rightarrow x'vy' = s_{i+1} \text{ avec } x',y',v \in (V_T \cup V_N)^*$$

L'élément auquel est appliqué la règle pour transformer x'ABy' en x'vy' étant soit A soit B. En se reportant à la définition de la relation $\Rightarrow$ on conclut que si $AB \Rightarrow u$ alors

$$S \Rightarrow^* x'ABy' \Rightarrow xuy$$

et que l'élément auquel est appliqué la règle pour transformer x'ABy' en xuy est le même que celui pour transformer x'ABy' en xvy.

En plus des deux lemmes précédents nous aurons besoin de quatre relations binaires sur $V_T \cup V_N$ notées τ pour "terminal", γ pour "non terminal", ρ pour "opérateur", ξ pour "opérande".

DEFINITION Pour tout couple $A,B \in V_T \cup V_N$:

$$[A \tau B \equiv A=B \text{ et } A,B \in V_T]$$

$$[A \gamma B \equiv A=B \text{ et } A,B \in V_N]$$

$$[A \rho B \equiv A=B \text{ et } A,B \in V_R]$$

$$[A \xi B \equiv A=B \text{ et } A,B \in V_E]$$

Moyennant ces éléments on a :

THEOREME 3.4.3 $\sigma=\delta^{*}\rho(\delta\varepsilon)^{*}\alpha\gamma^{*}\tau\cup\rho(\delta\varepsilon)^{*}\alpha\gamma^{*}\gamma\cup\varepsilon(\delta\varepsilon)^{*}\alpha$

DEMONSTRATION Démontrons tout d'abord les cinq points suivants :

(1) $\alpha \subset \sigma$

(2) $\delta\varepsilon\sigma\gamma \subset \sigma$

(3) $\delta\sigma\tau \subset \sigma$

(4) $\rho\sigma\gamma \subset \sigma$

(5) σ ne contient pas d'autres éléments que ceux énumérés en (1), (2), (3) et (4).

Point (1) : Considérons deux symboles A,B $V_T \cup V_N$ tels que $A\alpha B$. Il existe donc une règle de la forme $Y \to uABv$ avec $Y \in V_N$ et $u,v \in (V_T \cup V_N)^{*}$. D'après les restrictions (3) et (4) du paragraphe 1.5 il existe $t \in V_T^{*}$ tel que $S \Rightarrow^{*} t$ et tel que la règle $Y \to uABv$ soit utilisée pour construire t à partir de S. D'après le corollaire 1.4.3 et le lemme 1.4.1 on en conclut qu'il existe $x,y \in (V_T \cup V_N)^{*}$ avec $S \Rightarrow^{*} xYy \Rightarrow xuABvy \Rightarrow^{*} t$.

Points (2), (3) et (4) : Ils découlent immédiatement des lemmes 3.4.1 et 3.4.2.

Points (5) : Il découle immédiatement de la définition de la relation $\vdash$ et du lemme 3.4.1.

Des points (1), (2), (3), (4), (5) et du fait que $\gamma\tau=\tau\gamma=\varepsilon\rho=\emptyset$ on en déduit que :

$$\sigma = \bigcup_{i=o}^{n-1} \bigcup_{j=o}^{n-1} \bigcup_{k=o}^{n-1} \delta^{k}\rho^{j}(\delta\varepsilon)^{i}\alpha\gamma^{i}\gamma^{j}\tau^{k}$$

En sommant par rapport à k :

$$\sigma = \bigcup_{i=o}^{n-1} \bigcup_{j=o}^{n-1} (\delta^{+}\rho^{j}(\delta\varepsilon)^{i}\alpha\gamma^{i}\gamma^{j}\tau \cup \rho^{j}(\delta\varepsilon)^{i}\alpha\gamma^{i}\gamma^{j})$$

En sommant par rapport à j :

$$\sigma = \bigcup_{i=0}^{n-i} (\delta^+ \rho(\delta\ )^i \alpha\gamma^+ \tau \cup \delta^+ \alpha\tau \cup \rho(\delta\ )^i \alpha\gamma^+ \cup (\delta\ )^i \alpha\ ^i)$$

En sommant par rapport à i :

$$\sigma = \delta^+ \rho(\delta\varepsilon)^* \alpha\gamma^+ \tau \cup \delta^+ \alpha\tau \cup \rho(\delta\varepsilon)^* \alpha\gamma^+ \cup (\delta\varepsilon)^+ \alpha\nu \cup \alpha$$

Remarquons que :

$$\delta^+ = \delta^+ \rho(\delta\varepsilon)^* \cup (\delta\varepsilon)^+ \quad \text{d'où} \quad \delta^+ \alpha\tau = \delta^+ \rho(\delta\varepsilon)^+ \alpha\tau \cup (\delta\ )^+ \alpha\tau$$

On a donc :

$$\sigma = \delta^+ \rho(\delta\ )^* \alpha\gamma^* \tau \cup \rho(\delta\ )^* \alpha\gamma^+ \cup (\delta\ )^* \alpha.$$

Du fait que $(\delta\varepsilon)^* \alpha = \rho(\delta\varepsilon)^* \alpha \cup \varepsilon(\delta\varepsilon)^* \alpha$:

$$\sigma = \delta^+ \rho(\delta\varepsilon)^* \alpha\gamma^* \tau \cup \rho(\delta\varepsilon)^* \alpha\gamma^* \cup \varepsilon(\delta\varepsilon)^* \alpha$$

Et du fait que $\rho(\delta\varepsilon)^* \alpha\gamma^* = \rho(\delta\varepsilon)^* \alpha\gamma^* \tau \cup \rho(\delta\varepsilon)^* \alpha\gamma^* \nu$

$$\sigma = \delta^* \rho(\delta\varepsilon)^* \alpha\gamma^* \tau \cup \rho(\delta\varepsilon)^* \alpha\gamma^* \nu \cup \varepsilon(\delta\varepsilon)^* \alpha$$

3.5 Plus petites relations de précédence de gauche à droite

En nous servant des résultats précédents il nous est maintenant possible d'exprimer des relations de précédence de gauche à droite en fonction de relations plus simples. En plus des relations α, γ, δ et τ, ν, ρ, ξ nous aurons besoin de trois autres relations binaires sur $V_T \cup V_N$ notées $\underline{\alpha}, \underline{\gamma}, \underline{\delta}$ par analogie avec α, γ, δ.

DEFINITION Pour tout couple $A,B \in V_T \cup V_N$:

$A \underline{\alpha} B \equiv$ [$A,B \in V_R$ et il existe $U \in V_N$, $E \in V_R \cup \{\Lambda\}$, $x,y \in (V_T \cup V_N)^*$ tels que $U \rightarrow xAEBy$]

$A \underline{\gamma} B \equiv$ [$B \in V_R$ et il existe $E \in V_E \cup \{\Lambda\}$, $y \in (V_T \cup V_N)^*$ tels que $A \rightarrow EBy$]

$A \underline{\delta} B \equiv$ [$A \in V_R$ et il existe $E \in V_E \cup \{\Lambda\}$, $x \in (V_T \cup V_N)^*$ tels que $B \rightarrow xAE$]

Nous avons alors la propriété suivante :

LEMME 3.5.1 Une condition nécessaire et suffisante pour que trois relations binaires $\lessdot, \doteq, \gtrdot$ sur $V_T \cup V_N$ soient de précédence de gauche à droite est que

(1) $\underline{\alpha} \subset \doteq$

(2) $\rho(\delta \cup \varepsilon)^* \alpha \gamma^* \underline{\gamma} \subset \lessdot$

(3) $(\delta \cup (\delta \varepsilon)^* \alpha \cup \underline{\delta} \delta^* \rho(\delta \varepsilon)^* \alpha \gamma^* \tau) \subset \gtrdot$

DEMONSTRATION Remarquons que les points (2) et (3) de ce lemme sont équivalents aux points (2 bis) et (3 bis) qui suivent et ceci du fait du théorème 3.4.3

(2bis) $\rho \sigma \underline{\gamma} \subset \lessdot$

(3bis) $(\delta \varepsilon \sigma \cup \underline{\delta} \sigma \tau) \subset \gtrdot$

Pour démontrer le théorème il suffit donc de se reporter à la définition équivalente du paragraphe 3.3 et d'utiliser le lemme 3.4.2.

Introduisons maintenant le concept de plus petites relations de précédence de gauche à droite.

DEFINITION Etant donné une grammaire dans laquelle on a choisi des opérateurs et des opérandes, nous dirons que les relations de précédence de gauche à droite $\lessdot, \doteq, \gtrdot$ sont les plus petites si et seulement pour toutes autres relations de précédence de gauche à droite $\lessdot', \doteq', \gtrdot'$ dans la même grammaire avec les mêmes

opérandes on a toujours $\lessdot \subset \lessdot'$, $\doteq \subset \doteq'$, $\gtrdot \subset \gtrdot'$

Du lemme précédent il découle

COROLLAIRE 3.5.2 Quel que soit la grammaire et quel que soit le choix des opérateurs et des opérandes, il existe toujours de plus petites relations de précédence de gauche à droite $\lessdot, \doteq, \gtrdot$ qui sont :

$$\doteq = \underline{\alpha}$$
$$\lessdot = \rho(\delta\varepsilon)^*\alpha\gamma^*\underline{\gamma}$$
$$\gtrdot = \underline{\delta}\,\varepsilon(\delta\varepsilon)^*\alpha \cup \underline{\delta}\delta^*\rho(\delta\varepsilon)^*\alpha\gamma^*\tau$$

Rappelons que les relations $\underline{\alpha}, \underline{\gamma}, \underline{\delta}$, sont définies au début de ce paragraphe, les relations ε, ρ, τ au paragraphe 3.3 et les relations α, γ, δ, au paragraphe 4.6.

3.6 Une variante plus générale de l'analyseur de précédence de gauche à droite

Nous avons vu, au début du paragraphe 3.1, que le choix d'opérateurs et d'opérandes dans une grammaire G devait entre autres satisfaire à la condition (3) qui interdit d'avoir dans G des règles de la forme $U \to V$ avec V opérande. Cette condition restreint beaucoup les choix possibles. Nous nous proposons donc de la remplacer par une condition (3 bis) moins forte :

DEFINITION ETENDUE Nous appellerons opérateurs et opérandes les éléments respectifs de deux vocabulaires disjoints V_R et V_E tels que $V_R \cup V_E = V_T \cup V_N$ et choisis de telle façon que :

(1) tout symbole terminal soit un opérateur,

(2) il n'existe pas de règle de la forme $U \to xXYy$ avec $x, y \in (V_T \cup V_N)^*$ et X,Y opérandes,

(3bis) il n'existe pas de règle de la forme $U \to V$ avec U opérateur et V opérande.

Considérons une grammaire G de vocabulaire $V_T \cup V_N$ dans laquelle on a choisi des ensembles V_R et V_E conformément à la définition étendue. A cette grammaire faisons correspondre la grammaire G' de vocabulaire opérande et opérateur V'_R, V'_E ainsi définie :

- $V'_R = V_R$
- $V'_E = \{E \in V_E \mid$ il existe $R \in V_R$ et $x, y \in (V_T \cup V_N)^*$ tel que $E \to xRy$ dans $G\}$
- toute règle de G' est de la forme $X \to E'_o R_1 E'_1 \ldots R_n E'_n$ avec

  $n \geq o$, $E'_i \in V'_E \cup \{\Lambda\}$ et $R_i \in V'_R$ et on a

$$\left[X \to E'_o R_1 E'_1 \ldots R_n E'_n \text{ dans } G\right] \equiv \left[\begin{array}{l} \text{il existe } E_o, E_1, \ldots, E_n \in V_E \cup \{\Lambda\} \\ \text{avec } n \geq o,\ X \to E_o R_1 E_1 \ldots R_n E_n \\ \text{dans G et pour tout } i \in (0,1,\ldots,n) \\ E_i \Rightarrow^* E'_i. \end{array}\right]$$

- si l'axiome S de G est un opérateur alors G' aura le même axiome, sinon la grammaire G' aura pour axiomes l'ensemble des $S_1, S_2, \ldots, S_t \in V'_E$ tels que $S \Rightarrow^* S_i$ pour tout $i \in (1,0,\ldots\ t)$. Il va de soi que tous les résultats antérieurs se généralisent à des grammaires à plusieurs axiomes.

On peut alors remarquer trois choses :

- La grammaire G et la grammaire G' définissent le même langage.
- Les vocabulaires V'_R et V'_E de G' sont choisis d'une façon conforme à la définition non étendue d'opérandes et d'opérateurs (paragraphe 3.1).
- Si l'on désigne respectivement par $\lessdot, \doteq, \gtrdot$ et $\lessdot', \doteq', \gtrdot'$ les relations du corollaire 3.5.2 associées aux grammaires G et G', pour tout couple d'opérateurs A,B on a $A \lessdot B \equiv A \lessdot' B$, $A \doteq B \equiv A \doteq' B$, $A \gtrdot B \equiv A \gtrdot' B$.

Aux relations $\lessdot, \doteq, \gtrdot$ définies dans la grammaire G au corollaire 3.5.2 on peut donc associer l'analyseur de précédence de gauche à droite dans G' qui est :

Configurations initiales

=	x#	avec $x \in V_T^*$

Configurations finales

#	U#	avec si $S \in V_R$, $U=S$ et, si $S \in V_E$, $U \in V_E$ et $S \Rightarrow^* U$

Instructions [*]

Pour tout $A,B \in V_R \cup \{\#\}$, tout $E \in V_E \cup \{\Lambda\}$, toute règle de G de la forme $U \rightarrow E_o R_1 E_1 \ldots R_n E_n$ avec $n>o$, $E_o, E_1, \ldots, E_n \in V_E \cup \{\Lambda\}$, $R_1, R_2, \ldots, R_n \in V_R$ et tout $E_o, E_1, \ldots, E_n \in V_E \cup \{\Lambda\}$ tels que $E_i \Rightarrow^* E_i'$ $(o \leq i \leq n)$ on a :

(1) $(A \lessdot B) \rightarrow (A \lessdot B, \Lambda)$ si et seulement si $A \lessdot B$ ou $A=\#$ et $B \neq \#$

(2) $(A \doteq B) \rightarrow (AB, \Lambda)$ si et seulement si $A \doteq B$

(3) $(A \gtrdot B) \rightarrow (A \gtrdot, B)$ si et seulement si $A \gtrdot B$ ou $A=$= et B==

(4) $(E_o' R_1 E_1' \ldots R_n E_n' \gtrdot, B) \rightarrow (\Lambda, UB)$ si et seulement si $U \in V_E$ ou $U \in V_R$ et $B \in V_T \cup \{\#\}$

Cet analyseur constitue donc une variante plus générale de l'analyseur de précédence de gauche à droite du fait que le choix des opérateurs et des opérandes n'a besoin que d'être conforme à la définition étendue d'opérateurs et d'opérandes. Comme nous l'avons déjà mentionné les formules des relations $\lessdot, \doteq, \gtrdot$ sont les mêmes que celles concernant l'analyseur de précédence de gauche à droite classique, c'est-à-dire que :

$$\doteq = \underline{\alpha}$$

$$\lessdot = \rho(\delta\varepsilon)^*_{\gamma\underline{\gamma}}$$

$$\gtrdot = \underline{\delta}\varepsilon(\delta\varepsilon)^*\alpha \cup \underline{\delta}\delta^*\rho(\delta\varepsilon)^*\alpha\gamma^*\tau$$

---

[*] En fait il faudrait raisonner sur V_E' et non sur V_E mais ceci ne fait qu'introduire des instructions parasites qui ne gênent en rien le fonctionnement de l'automate.

Considérons par exemple la grammaire suivante qui définit des expressions arithmétiques :

G_8 :

A→PB
A→APB
A→B
B→BMC
B→C
C→(A)
C→a
P→+
P→-
M→*
M→/

$V_T = \{a,+,-,*,/,(,)\}$

$V_N = \{A,B,C,P,M\}$

axiome : A (exceptionnellement nous ne l'avons pas noté S)

Conformément à la définition étendue on peut choisir les opérateurs et les opérandes ainsi :

$V_R = \{a,+,-,*,/,(,),P,M\}$ $\quad V_E = \{A,B,C\}$

En utilisant les formules précédentes on obtient

	a	$\overset{+}{-}$	*/	(	)	P	M
a	.	$\lessdot$	$\lessdot$	.	$\gtrdot'$	$\gtrdot'$	$\gtrdot'$
+-	$\gtrdot'$	.	.	$\gtrdot'$	.	.	.
*/	$\gtrdot'$	.	.	$\gtrdot'$	.	.	.
(	$\lessdot$	.	.	$\lessdot$	$=$	$\lessdot$	$\lessdot$
)	.	$\lessdot$	$\lessdot$	.	$\gtrdot'$	$\gtrdot'$	$\gtrdot'$
P	$\lessdot$	.	.	$\lessdot$	$\gtrdot'$	$\gtrdot'$	$\lessdot$
M	$\lessdot$	.	.	$\lessdot$	$\gtrdot'$	$\gtrdot'$	$\gtrdot'$

Le tableau suivant représente la relation $\not\Rightarrow^*$ entre deux opérandes :

	A	B	C
A	1	1	1
B	.	1	1
C	.	.	1

L'analyse de la chaîne "a+a*(-a)" se fera donc ainsi :

```
#          a+a*(-a)#
# [a        +a*(-a)#
# [a[+       a*(-a)#
# [a[+]      a*(-a)#
# [a         Pa*(-a)#
# [a]        Pa*(-a)#
#          CPa*(-a)#
# [CP         a*(-a)#
# [CP[a        *(-a)#
# [CP[a[*       (-a)#
# [CP[a[*]      (-a)#
# [CP[a         M(-a)#
# [CP[a]        M(-a)#
# [CP         CM(-a)#
# [CP[CM        (-a)#
# [CP[CM[(       -a)#
# [CP[CM[([-     a)#
# [CP[CM[([-]    a)#
# [CP[CM[(       Pa)#
# [CP[CM[([P     a)#
# [CP[CM[([P[a    )#
# [CP[CM[([P[a]   )#
# [CP[CM[([P     C)#
# [CP[CM[([PC]    )#
# [CP[CM[(       A)#
# [CP[CM[(A)      #
# [CP[CM[(A)]     #
# [CP[CM          C#
# [CP[CMC]        #
# [CP             B#
# [CPB]           #
#                 A#
```

3.7 Cas particulier : Relations de précédence terminale

Soit G une grammaire de vocabulaire $V_T \cup V_N$ qui ne comporte aucune règle de la forme $U \rightarrow xXYy$ avec $x,y \in (V_T \cup V_N)^*$ et $X,Y \in V_N$. Conformément à la définition étendue d'opérateurs et d'opérandes du paragraphe précédent on peut prendre $V_R = V_T$ et $V_E = V_N$ c'est-à-dire que l'on peut considérer que les symboles terminaux sont des opérateurs et les symboles non terminaux des opérandes. En se reportant aux formules du corollaire 3.5.2 et en tenant compte du fait que $\alpha = \tau\alpha \cup \alpha\tau$, $\rho = \tau$, $\mathcal{E} = \mathcal{V}$ on obtient alors :

$$\doteq = \underline{\alpha}$$

$$\lessdot = \alpha\gamma^*\underline{\gamma}$$

$$\gtrdot = \underline{\delta}\delta^*\alpha$$

Ces relations correspondent à celles définies par R. Floyd en [F1.1]. Du fait de la symétrie entre les relations $\lessdot$ et $\gtrdot$ et du fait qu'elles ne concernent que les symboles terminaux nous les appellerons relations de précédence terminale. On peut leur associer l'analyseur du paragraphe précédent qui se présente alors sous cette forme très simple :

Configurations initiales

#	x#	avec $x \in V_T$

Configurations finales

#	U#	avec $U \in V_N$ et $S \Rightarrow^* U$

Instructions

Pour tout $A,B \in V_T \cup \{\#\}$, tout $E \in V_N \cup \{\Lambda\}$, toute règle de G de la forme $U \rightarrow E_o R_1 E_1 \ldots E_n R_n$ avec $n>o$, $E_o, E_1, \ldots, E_n \in V_N \cup \{\Lambda\}$, $R_1, R_2, \ldots, R_n \in V_T$ et tout $E'_o, E'_1, \ldots, E'_n \in V_N \cup \{\Lambda\}$ tels que $E_i \Rightarrow^* E'_i (o \leqslant i \leqslant n)$ on a :

(1) $(A,EB)\rightarrow(A\lfloor EB,\Lambda)$ si et seulement si $A\lessdot B$ ou $A=\#$ et $B\neq\#$

(2) $(A,EB)\rightarrow(AEB,\Lambda)$ si et seulement si $A\doteq B$

(3) $(A,EB)\rightarrow(AE\rceil,B)$ si et seulement si $A\gtrdot B$ ou $A\neq\#$ et $B=\#$

(4) $(E_o'R_1E_1'\ldots R_nE_n',\Lambda)\rightarrow(\Lambda,U)$

Considérons la grammaire G_8 de l'exemple précédent, elle peut s'écrire sous une forme équivalente :

A→+B
A→-B
A→A+B
A→A-B
A→B
B→B*C
B→B/C
B→C
C→(A)
C→a

$V_T = \{a,+,-,*,/,(,)\}$

$V_N = \{A,B,C\}$

axiome : A

Du fait que dans aucune règle il n'apparaît deux symboles non terminaux adjacents on peut définir des relations de précédence terminale. En utilisant les formules précédentes on obtient :

	a	$\overset{+}{-}$	*/	(	)
a	.	$\gtrdot$	$\gtrdot$	.	$\gtrdot$
+-	$\lessdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\gtrdot$
*/	$\lessdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$
)	.	$\gtrdot$	$\gtrdot$	.	$\gtrdot$

La relation $\Rightarrow^*$ entre deux symboles non terminaux se résume au tableau

	A	B	C
A	1	1	1
B	.	1	1
C	.	.	1

L'analyse de la chaîne "a+a*(-a)" se fera ainsi :

```
#        a+a*(-a)#
#[a       +a*(-a)#
#[a]      +a*(-a)#
#        C+a*(-a)#
#[C+       a*(-a)#
#[C+[a      *(-a)#
#[C+[a]     *(-a)#
#[C+       C*(-a)#
#[C+[C*      (-a)#
#[C+[C*[(     -a)#
#[C+[C*[([-    a)#
#[C+[C*[([-[a   )#
#[C+[C*[([-[a]  )#
#[C+[C*[([-    C)#
#[C+[C*[([-C]   )#
#[C+[C*[(      A)#
#[C+[C*[(A)      #
#[C+[C*[(A)]     #
#[C+[C*C       C #
#[C+[C*C]       #
#[C+           B #
#[C+B]          #
#              A #
```

3.8 Cas particulier, relations de précédence totale de gauche à droite

Dans une grammaire G de vocabulaire $V_T \cup V_N$ on peut toujours considérer que tous les symboles sont des opérateurs c'est-à-dire que $V_R = V_T \cup V_N$ et $V_E = \emptyset$. L'automate se présente alors sous la forme :

Configurations initiales

$\vdash$	$x\dashv$	$x \in V_T^*$

Configuration finale

$\dashv$	$S\dashv$

Instructions

Pour tout $A,B \in V_T \cup V_N \cup \{\#\}$ et toute règle $U \to u$ de G on a :

(1) $(A,B) \to (A\,|\,B,\Lambda)$ si et seulement si $A \lessdot B$ ou $A = \vdash$ et $B = \dashv$

(2) $(A,B) \to (AB,\Lambda)$ si et seulement si $A \doteq B$

(3) $(A,B) \to (A\,|\,,B)$ si et seulement si $A \gtrdot B$ ou $A \neq \#$ et $B = \dashv$

(4) $(|u\,|,B) \to (\Lambda,UB)$ si et seulement si $B \in V_T \cup \{\dashv\}$

DEFINITION Si l'automate précédemment décrit est un analyseur pour la grammaire G, alors cet analyseur ainsi que les relations $\lessdot, \doteq, \gtrdot$ qui lui sont associées seront dits de précédence totale de gauche à droite.

En se reportant au corollaire 3.5.2 et en remarquant que dans ce cas $\underline{\alpha} = \alpha$, $\underline{\gamma} = \gamma$, $\underline{\delta} = \delta$, $\rho = \tau \cup \gamma$, $\mathcal{E} = \emptyset$ on déduit le théorème :

THEOREME 3.8.1 Dans toute grammaire les relations $\lessdot, \doteq, \gtrdot$ définies comme suit sont les plus petites relations de précédence totale de gauche à droite.

$$\doteq = \alpha$$

$$\lessdot = \alpha\gamma^+$$

$$\gtrdot = \delta^+\alpha\gamma^*\tau$$

Ces relations correspondent à celles données par C. Pair en [Pa.1]. N. Wirth et H. Weber [WW] ont défini des relations $\overset{\circ}{<}, \overset{\circ}{=}, \overset{\circ}{>}$ très voisines de celles de C. Pair. Seule la relation $\overset{\circ}{>}$ diffère et peut être exprimée ainsi :

$$\overset{\circ}{>} = \delta^{+}\alpha\gamma^{*}$$

On a la propriété

LEMME 3.8.2 $\alpha\gamma^{*} \cap \delta^{+}\alpha\gamma^{*} \neq \emptyset$ entraîne $\alpha\gamma^{*} \cap \delta^{*}\tau \neq \emptyset$

DEMONSTRATION Si $\alpha\gamma^{*} \cap \delta^{+}\alpha\gamma^{*} \neq \emptyset$ alors il existe $A,B \in V_T \cup V_N$ tels que $A\alpha\gamma^{*}B$ et $A\delta^{+}\alpha\gamma^{*}B$. Du fait de la restriction (4) (paragraphe 1.4) dans une grammaire il existe donc b V_T tel que $B\gamma^{*}b$. Donc $A\alpha\gamma^{*}b$ et $A\delta^{+}\alpha\gamma^{*}b$ d'où $\alpha\gamma^{*} \cap \delta^{+}\alpha\gamma^{*}\tau \neq \emptyset$.

Du théorème 3.8.1 et du lemme 3.8.2 on déduit que les relations de N. Wirth et H. Weber sont des relations de précédence totale de gauche à droite telles que leur non incompatibilité deux à deux entraîne l'inexistence de relations de précédence totale de gauche à droite incompatibles deux à deux dans la grammaire considérée. De ce fait on a :

COROLLAIRE 3.8.3 Une condition nécessaire et suffisante pour que dans une grammaire il existe des relations de précédence totale de gauche à droite incompatibles deux à deux est que :

$$(\delta^{+}\alpha \cap \alpha) \cup (\alpha\gamma^{+} \cap \alpha) \cup (\delta^{+}\alpha\gamma^{+} \cap \alpha) \cup (\delta^{+}\alpha \cap \alpha\gamma^{+}) \cup (\delta^{+}\alpha\gamma^{+} \cap \alpha\gamma^{+}) = \emptyset$$

Cette condition peut s'écrire aussi :

$$(\alpha\gamma^{+} \cap \alpha) \cup (\delta^{+}\alpha\gamma^{*} \cap \alpha\gamma^{*}) = \emptyset$$

La condition de ce corollaire est plus forte que la condition d'existence de relations de précédence totale incompatibles deux à deux donnée au théorème 2.4.1. Il faut donc s'attendre à trouver des grammaires pour lesquelles il existe des relations de précédence totale incompatibles deux à deux mais pour lesquelles il n'existe pas de relations de précédence totale de gauche à droite incompatible

deux à deux. La grammaire G_6 du paragraphe 2.6 en est un exemple. Reportons nous aux relations $\lessdot,\doteq,\gtrdot$ définies à la fin du paragraphe 2.5.

COROLLAIRE 3.8.4 Pour toute grammaire dans laquelle il existe des relations de précédence totale de gauche à droite incompatibles deux à deux, les relations de précédence totale $\lessdot,\doteq,\gtrdot$ ainsi définies :

$$\doteq = \alpha$$

$$\lessdot = \alpha\gamma^+$$

$$\gtrdot = \delta^+\alpha \cup (\delta^+\alpha \cap \alpha\gamma^+$$

sont de précédence totale de gauche à droite.

DEMONSTRATION Si dans G il existe des relations de précédence totale incompatibles deux à deux alors $\delta^+\alpha\gamma^+\cap\alpha\gamma^+=\emptyset$ d'après le corollaire 3.8.3. De ce fait $\delta^+\alpha\gamma^+\cap\alpha\gamma^+ = \delta^+\alpha\gamma^+$ et donc $\gtrdot = \delta^+\alpha\gamma^*$. D'après le théorème 3.8.1 on en conclut que les relations $\lessdot,\doteq,\gtrdot$ sont de précédence totale de gauche à droite.

3.9 Langages de précédence totale, de précédence totale de gauche à droite et autres.

Au paragraphe 2.6 nous avons comparé l'ensemble des langages de précédence totale à celui des langages déterministes. Nous allons compléter les résultats obtenus en faisant intervenir d'autres classes de langages.

DEFINITION Nous appellerons grammaire de précédence totale de gauche à droite toute grammaire pour laquelle il existe un analyseur de précédence totale de gauche à droite déterministe.

Du théorème 3.2.1 et du corollaire 3.8.3 on déduit immédiatement cette définition équivalente :

DEFINITION EQUIVALENTE Une grammaire de précédence totale de gauche à droite est une grammaire qui satisfait aux deux conditions :

(1) Toutes les règles ont des membres droits distincts

(2) $(\alpha\gamma^+\cap\alpha)\cup(\delta^+\alpha\gamma^*\cap\alpha\gamma^*) = \emptyset$

Rappelons que la condition (2) peut aussi s'écrire :

$$(\delta^+\alpha\cap\alpha)\cup(\alpha\gamma^+\cap\alpha)\cup(\delta^+\alpha\gamma^+\cap\alpha)\cup(\delta^+\alpha\cap\alpha\gamma^+)\cup(\delta^+\alpha\gamma^+\cap\alpha\gamma^+) = \emptyset$$

et que de ce fait les grammaires de précédence totale de gauche à droite sont bien un cas particulier des grammaires de précédence totale. Passons maintenant aux langages définis par de telles grammaires :

DEFINITION Tout langage "context free" qui peut être défini au moyen d'une grammaire de précédence totale de gauche à droite sera dit de précédence totale de gauche à droite.

Comparons la classe de ces langages avec celles des langages réguliers (états finis).

THEOREME 3.9.1 L'ensemble des langages réguliers est strictement inclus dans le sous-ensemble des langages de précédence totale de gauche à droite dont les réfléchis sont aussi de précédence totale de gauche à droite.

DEMONSTRATION A tout langage régulier R défini sur un vocabulaire V_T on sait associer un automate d'état fini déterministe capable de l'analyser. Soit Q l'ensemble de ses états. Considérons la grammaire G de vocabulaire $V_T \cup V_N$ ainsi définie :

- $V_N = Q \cup \{S\}$ (on s'arrange de façon à ce que $S \notin Q$)
- A chaque instruction de l'automate de la forme

  $aE \rightarrow F$ avec $a \in V_T$ et $E, F \in Q$

  on fait correspondre la règle de G

  $F \rightarrow a$ si E est l'état initial

  $F \rightarrow Ea$ si E n'est pas l'état initial
- A chaque état final E de l'automate, on fait correspondre la règle de G

  $S \rightarrow E$

Le langage défini par la grammaire G est R. De plus du fait que l'automate était déterministe, toutes les règles de G ont des membres droits distincts. Si l'on regarde la forme des règles de G on s'aperçoit que $\alpha\gamma^+ = \emptyset$, $\delta^+\alpha\gamma^* = \tau\delta^+\alpha\gamma^*$, $\alpha\gamma^* = \backslash\alpha\gamma^*$. De ce fait $(\alpha\gamma^+ \cap \alpha) \cup (\delta^+\alpha\gamma^* \cap \alpha\gamma^*) = \emptyset$ et R est un langage de précédence totale de gauche à droite. Tout langage régulier R est donc de précédence totale de gauche à droite et du fait que le réfléchi d'un langage régulier est toujours régulier, R est aussi le réfléchi d'un langage de précédence totale de gauche à droite. Il ne reste plus qu'à démontrer que l'inclusion est stricte. Pour ceci considérons la grammaire

$S \rightarrow ab$

$S \rightarrow aSb$

Le langage ainsi défini ainsi que son réfléchi n'est pas régulier mais est de précédence totale de gauche à droite.

Comparons maintenant l'ensemble des langages de précédence totale de gauche à droite avec l'ensemble des langages déterministes.

THEOREME 3.9.2 L'ensemble des langages de précédence totale de gauche à droite est inclus dans l'ensemble des langages déterministes mais n'est pas inclus dans l'ensemble de leurs réfléchis.

DEMONSTRATION La première partie du théorème se démontre en particularisant le théorème 3.2.3 aux analyseurs de précédence totale de gauche à droite. Pour démontrer la deuxième partie il suffit de trouver un exemple de réfléchi de langage de précédence totale de gauche à droite qui ne soit pas déterministe. Considérons le langage

$$L_4 = \{a^i b^j a^j \cup a^i b^i a^j c \mid i>o,\ j>o\}$$

défini par la grammaire G_4 du paragraphe 2.5. Il est facile de vérifier que $\widetilde{L}_4$ est de précédence totale de gauche à droite. Pour ceci il suffit de définir $\widetilde{L}_4$ par une grammaire symétrique de G_4. En se reportant à la démonstration de théorème 2.6.3 et en remarquant que

$$\{a^i b^j a^j \cup a^i b^i a^j \mid i>o,\ j>o\} = (L_4\ c^* \cap a^* b^* a^* c)/c$$

on conclut que L_4 n'est pas déterministe.

Pour terminer ce chapitre récapitulons les imbrications des différentes classes de langages. Chaque classe sera désignée par une lettre.

R : ensemble des langages réguliers (états finis)

D : ensemble des langages déterministes

$\widetilde{D}$: ensemble des réfléchis des langages déterministes

P_t : ensemble des langages de précédence totale

P_{tgd} : ensemble des langages de précédence totale de gauche à droite

$\tilde{P}_{tgd}$: ensemble des réfléchis des langages de précédence totale de gauche à droite.

D'après le paragraphe 2.6 :

$$P_t \not\subset D \cup \tilde{D} \qquad \text{et} \qquad D \cap \tilde{D} \not\subset P_t$$

Les langages de précédence totale de gauche à droite sont un cas particulier des langages de précédences totale et du fait que tout réfléchi d'un langage de précédence totale est encore un langage de précédence totale (théorème 2.6.2) :

$$P_{tgd} \cup \tilde{P}_{tgd} \subset P_t$$

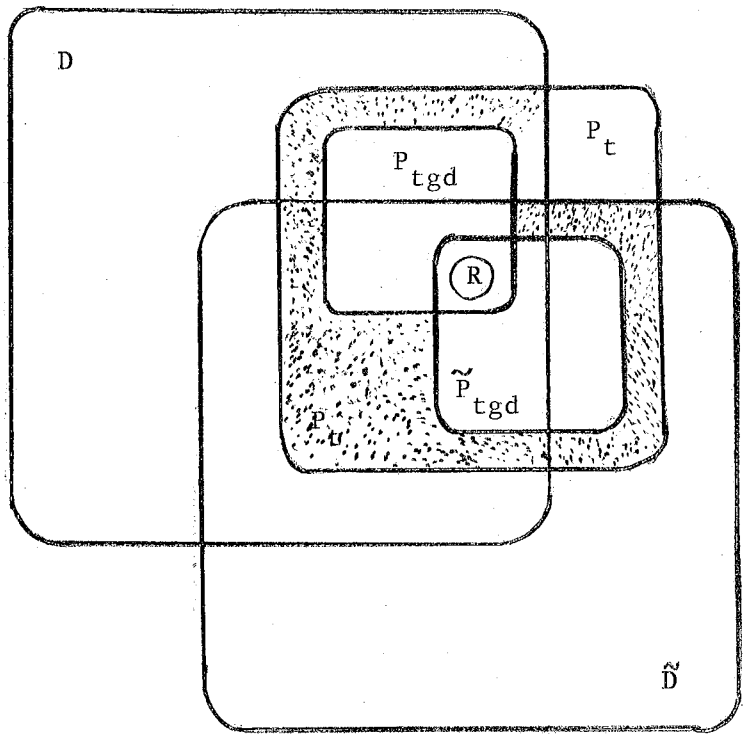
Nous venons de voir (théorème 3.9.2) que :

$$P_{tgd} \subset D \qquad \text{et} \qquad P_{tgd} \not\subset \tilde{D} \qquad \text{d'où} \qquad \tilde{P}_{tgd} \subset \tilde{D} \qquad \text{et} \qquad \tilde{P}_{tgd} \not\subset D$$

Du fait du théorème 3.9.1 :

$$R \subset P_{tgd} \cap \tilde{P}_{tgd} \text{ et } R \neq P_{tgd} \cap \tilde{P}_{tgd}$$

Nous pouvons donc tracer le schéma :



Nous ignorons si la zone marquée de points existe ou n'existe pas. A remarquer que si cette zone était vide, l'intersection d'un langage de précédence totale avec un langage déterministe serait toujours un langage de précédence totale de gauche à droite.

CHAPITRE 4 : Programmation des différents algorithmes

4.1 Calcul de matrices booléennes représentant des relations de précédence

Au cours des deux chapitres précédents nous avons vu que les relations de précédences pouvaient s'exprimer en fonction de relations simples. Nous avions entre autres obtenu les résultats suivants :

Relations de précédence totale (paragraphe 2.5)

$$\overset{\bullet}{=} = \alpha$$

$$\overset{\bullet}{<} = \alpha\gamma^{+}$$

$$\overset{\bullet}{>} = \delta^{+}\alpha \cup (\delta^{+}\alpha\gamma^{+} \cap \overline{\alpha\gamma^{+}})$$

Relations de précédence de gauche à droite (paragraphe 3.5)

$$\overset{\bullet}{=} = \underline{\alpha}$$

$$\overset{\bullet}{<} = \rho(\delta\varepsilon)^{*}\alpha\gamma^{*}\underline{\gamma}$$

$$\overset{\bullet}{>} = \underline{\delta}(\delta)^{*} \cup \underline{\delta}\delta^{*}\rho(\delta\varepsilon)^{*}\alpha\gamma^{*}\tau$$

Relations de précédence terminale (paragraphe 3.7)

$$\overset{\bullet}{=} = \underline{\alpha}$$

$$\overset{\bullet}{<} = \alpha\gamma^{*}\underline{\gamma}$$

$$\overset{\bullet}{>} = \underline{\delta}\delta^{*}\alpha$$

Relations de précédence totale de gauche à droite (paragraphe 3.8)

$$\overset{\bullet}{=} = \alpha$$

$$\overset{\bullet}{<} = \alpha\gamma^{+}$$

$$\overset{\bullet}{>} = \delta^{+}\alpha\gamma^{*}\tau$$

Rappelons que les relations α,γ,δ ont été définies au paragraphe 1.6, les relations ε,ρ,τ au paragraphe 3.4 et les relations $\underline{\alpha},\underline{\gamma},\underline{\delta}$ au paragraphe 3.5.

Soit n le nombre d'éléments du vocabulaire $V_T \cup V_N$ de la grammaire considérée. Posons

$$V_T \cup V_N = \{e_1, e_2, \ldots, e_n\}$$

et représentons chaque relation binaire ρ sur $V_T \cup V_N$ par une matrice booléenne $R=\{r_{ij}\}$ de dimension nxn ainsi définie :

r_{ij}=vrai si et seulement si $e_i \rho e_j$

Les opérations définies sur les relations binaires induisent alors des opérations sur des matrices booléennes, opération que nous noterons de la même façon.

Soient $A=\{a_{ij}\}, B=\{b_{ij}\}, C=\{c_{ij}\}$ trois matrices booléennes de dimensions nxn représentant des relations binaires.

- Si $C = A \cup B$ alors $c_{ij} = a_{ij} \vee b_{ij}$.
- Si $C = A \cap B$ alors $c_{ij} = a_{ij} \wedge b_{ij}$.
- Si $C = \bar{A}$ alors $c_{ij} = \neg a_{ij}$.
- Si $C = AB$ alors $c_{ij} = \bigvee_{k=1}^{n} a_{ik} \wedge b_{kj}$ et de ce fait le produit AB peut être calculé par l'algorithme :

```
pour i:=1 pas 1 jusqua n faire
pour j:=1 pas 1 jusqua n faire
pour k:=1 pas 1 jusqua n faire
    début s:=faux ;
    pour k:=1 pas 1 jusqua n faire
    s:=s∨A[i,k]∧B[k,j] ;
    C[i,j] :=s fin
```

- Du fait que $A^+ = \bigcup_{i=1}^{n-1} A^i$ la fermeture transitive A^+ de A peut-être calculée en utilisant l'algorithme simple et efficace de Warshall [w_a] : l'exécution de

```
pour k:=1 pas 1 jusqua n faire
pour i:=1 pas 1 jusqua n faire
pour j:=1 pas 1 jusqua n faire
A[i,j] :=A[i,j] ∨A[i,k]∧A[k,j]
```

transforme A en sa fermeture transitive. Pour obtenir la fermeture transitive réflexive A^* il suffit de complèter A^+ en mettant des vrai sur toute sa première diagonale.

En fait nous avons utilisé une variante de ces deux algorithmes. La procédure PROFER qui suit permet de calculer un produit ou une fermeture transitive :

```
procédure PROFER(A,B,C) ; booleen tableau A,B,C ;
pour k:=1 pas 1 jusqua n faire
pour i:=1 pas 1 jusqua n faire
si A[i,k] alors
pour j:=1 pas 1 jusqua n faire
C[i,j] :=C[i,j] ∨B[k,j]
```

si après avoir remis C à zéro (on met des faux partout dans C) on appelle PROFER(A,B,C) on obtient C=AB, à condition que C soit différent de A et de B. Si on appelle PROFER(A,A,A) alors on transforme A en sa fermeture transitive.

Si chaque ligne de matrice booléenne est représentée par une suite de "bits" découpée en plusieurs mémoires on peut alors utiliser le fait que dans la plupart des calculateurs digitaux il existe une instruction permettant de réaliser l'union logique de deux mémoires d'un seul coup. L'exécution de la dernière

boucle pour de la procédure PROFER peut alors être considérablement accélérée. D'autre part le fait de représenter chaque élément booléen par un seul "bit" permet de n'utiliser qu'un nombre réduit de mémoires pour représenter une matrice booléenne.

Moyennant tous ces éléments, la façon de programmer un calcul quelconque de matrices booléennes représentant des relations de précédence se déduit immédiatement des formules rappelées en tête de ce paragraphe.

4.2 Généralités sur la programmation des différents analyseurs

La programmation de chaque analyseur de précédence est inspirée directement de sa description correspondante par un automate à deux piles. Signalons cependant deux différences :

- On n'intercale pas de symboles $\lfloor , \rfloor$ entre les éléments de la pile gauche. Les éléments passent de la pile droite à la gauche jusqu'à rencontre d'une précédence du type $\gtrdot$. Après avoir reculé dans la pile gauche jusqu'à rencontre d'une précédence du type $\lessdot$, la sous-chaîne ainsi isolée est alors réduite, c'est-à-dire remplacée par un seul symbole non terminal. Ce processus est alors répété autant de fois qu'il le faut.

- Au départ la pile droite ne contient pas la totalité de la chaîne à analyser. Cette dernière est introduite dans la pile en cours d'analyse, élément par élément, au fur et à mesure des besoins.

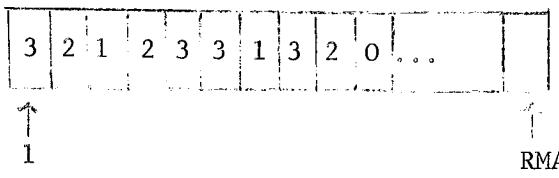
Voici les conventions générales adoptées:

- Les symboles du vocabulaire de la grammaire considérée sont numérotés de 1 à NBSYMBOLES. Dans le cas où il est nécessaire de faire la distinction

entre opérateurs et opérandes, les opérateurs sont numérotés de 1 à NBOPERATEURS et les opérandes de NBOPERATEURS+1 à NBSYMBOLES. AXIOME a pour valeur le numéro de l'axiome.

- La chaîne à analyser est supposée être encadrée de deux symboles spéciaux # dont le numéro, différent de celui de tous les autres symboles, est égal à STOP. Cette chaîne est lue au moyen d'une entier procédure PROCHAINSYMBOLE dont la valeur est égale au numéro du prochain symbole lu.

- Les règles sont placées à la suite les unes des autres dans l'entier tableau R[1:RMAX], en remplaçant chaque symbole par son numéro, et chaque signe → par un entier égal à la longueur de la chaîne qui suit. Le nombre o marque la fin de l'ensemble de toutes les règles. Par exemple la grammaire S→ab,S→aSb serait représentée par le tableau R qui suit



en numérotant a par 1, b par 2 et S par 3.

- Les relations $\overset{\circ}{<},\overset{\circ}{=},\overset{\circ}{>}$ et éventuellement la relation $\overset{*}{\Rightarrow}$ entre deux opérandes, sont respectivement représentées par les booléen tableau MI,ME,MS et SU.

- Les deux piles sont groupées en un seul entier tableau P[1:PMAX], chaque pile se trouvant à une extrémité de ce tableau.

4.3 Analyseur de précédence totale déterministe

L'analyseur de précédence totale déterministe ne s'applique qu'aux grammaires de précédence totale (paragraphe 2.6) ou de précédence totale de

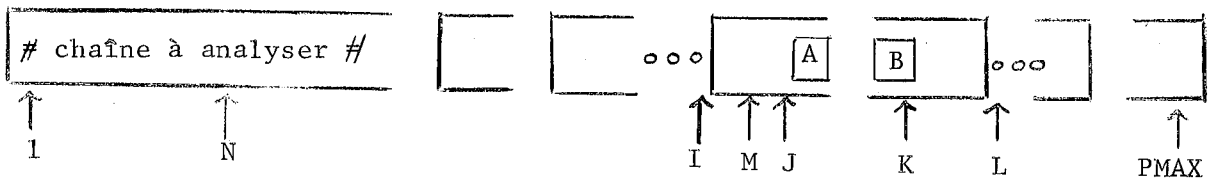
gauche à droite (paragraphe 3.9). Du fait que dans ces grammaires les relations $\overset{\circ}{<}, \overset{\circ}{=}, \overset{\circ}{>}$ sont incompatibles deux à deux il n'est pas nécessaire d'utiliser les trois relations ; on se contentera par exemple des relations $\overset{\circ}{=}$ et $\overset{\circ}{>}$. Cependant afin de détecter au plus tôt si la chaîne à analyser n'est pas une phrase, il peut être intéressant de compléter la relation $\overset{\circ}{>}$ en considérant qu'elle est vérifiée partout où aucune des relations $\overset{\circ}{<}, \overset{\circ}{=}$ ne l'est. L'algorithme d'analyse étant très simple, le programme correspondant qui suit se passe de commentaires.

```
   DEBUT   ENTIER   TABLEAU  P[1:PMAX],R[1:RMAX] ;
   BOOLEEN TABLEAU ME,MS[1:NBSYMBOLES,1:NBSYMBOLES] ;
   ENTIER A,AXIOME,B,J,K,M,STOP,U,V ;

INITIALISATION:
   A:=STOP:=PROCHAINSYMBOLE ; B:=PROCHAINSYMBOLE ; J:=0 ; K:=PMAX+1;
AVANCEMENT:
   ECRIRE( " AVANCEMENT " ,B) ; J:=J+1 ; P[J]:=A ; A:=B ;
   SI K > PMAX ALORS B:=PROCHAINSYMBOLE SINON
      DEBUT B:=P[K] ; K:=K+1 FIN ;
TEST D AVANCEMENT:
   SI A=STOP ALORS ALLERA  AVANCEMENT ;
   SI B=STOP ALORS ALLERA  DEBUT DE RECUL ;
   SI NON MS[A,B] ALORS ALLERA  AVANCEMENT ;
DEBUT DE RECUL:
   J:=J+1 ; P[J]:=A ; K:=K-1 ; P[K]:=B ; M:=J ;
RECUL:
   ECRIRE( " RECUL " ,A) ; B:=A ; M:=M-1 ; A:=P[M] ;
TEST DE RECUL:
   SI A=STOP ALORS ALLERA  RECHERCHE DE REGLES ;
   SI ME[A,B] ALORS ALLERA  RECUL ;
RECHERCHE DE REGLES:
   POUR U:=2,U+R[U]+2 TANTQUE  R[U-1] ≠ 0 FAIRE
   SI R[U]=J-M ALORS
      DEBUT POUR V:=1 PAS 1 JUSQUA J-M FAIRE
      SI P[M+V] ≠ R[U+V] ALORS ALLERA  RIEN ;
      B:=R[U-1] ; ALLERA  REDUCTION ; RIEN: FIN ;
   ECRIRE( " ERREUR " ) ; ALLERA  FIN D ANALYSE ;
REDUCTION:
   ECRIRE( " REDUCTION " ,B) ; J:=M-1 ;
   SI B ≠ AXIOME OU A ≠ STOP OU P[K] ≠ STOP
   ALORS ALLERA  TEST D AVANCEMENT ;
   ECRIRE( " CHAINE CORRECTE " ) ;
FIN D ANALYSE:
   FIN
```

4.4 Analyseur de précédence totale non déterministe

L'analyseur de précédence totale non déterministe s'applique à n'importe quelle grammaire. Il se présente sous forme d'une procédure récursive ANALYSE qui se rappelle chaque fois que plusieurs éventualités se présentent : plusieurs relations de précédence vérifiées entre deux symboles ou plusieurs règles ayant le même membre droit. A chaque appel de cette procédure qui correspond donc à une bifurcation, il est nécessaire de retenir l'état dans lequel on se trouve car, si l'on arrive à une impasse, il faut revenir à la dernière bifurcation pour essayer une autre voie. De ce fait on est amené d'une part à conserver toute la chaîne à analyser en mémoire et d'autre part à recopier le contenu des deux piles à chaque appel de la procédure ANALYSE. Cette recopie se fait d'ailleurs à l'intérieur même du tableau P dont voici la configuration :



- A et B ont respectivement pour valeurs les sommets des deux piles.

- Les indices I,J et K,L permettent de délimiter la pile gauche et la pile droite.

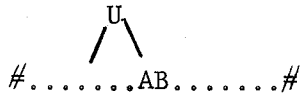
- L'indice M repère jusqu'où on a reculé dans la pile gauche.

- L'indice N repère à quelle endroit on est dans la chaîne.

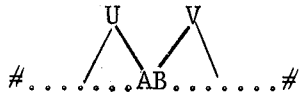
Dans le cas où il existe deux symboles A,B, pour lesquelles on a à la fois $A \lessdot B$ et $A \gtrdot B$ il se peut que la chaîne en cours d'analyse

#.....AB.....#

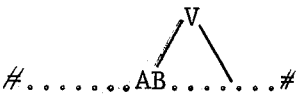
soit d'une part réduite ainsi



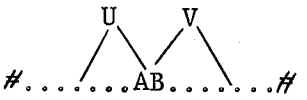
puis ainsi



et d'autre part réduite ainsi



puis ainsi



Dans ce cas l'analyseur cheminerait par deux voies différentes correspondant à une même structure. Nous avons donc prévu certains tests et certaines actions pour éviter ceci. Cependant cette légère modification de l'analyseur implique que les relations de précédence qui lui sont fournies soient de précédence totale de gauche à droite ou soit telles qu'elles permettent d'analyser des chaînes comportant aussi des symboles non terminaux ce qui est notamment le cas pour les relations $\overset{\cdot}{<} = \alpha\delta^+$, $\overset{\cdot}{=} = \alpha$, $\overset{\cdot}{>} = \gamma^+\alpha \cup (\gamma^+\alpha\ \delta^+ \cap \overline{\alpha\delta^+})$.

Une autre conséquence de cette modification est la suivante : si on considère que pour tout couple de symboles A,B on a toujours $A\overset{\cdot}{<}B$, $A\overset{\cdot}{=}B$, $B\overset{\cdot}{>}A$, alors cet analyseur se comporte comme l'analyseur par "substitution directe" le plus classique.

```
DEBUT ENTIER TABLEAU P[1:PMAX],R[1:RMAX] ;
BOOLEEN TABLEAU MI,ME,MS[1:NBSYMBOLES,1:NBSYMBOLES] ;
ENTIER AXIOME,STOP,V ;

PROCEDURE ANALYSE(NO,A,B,I,J,K,L,M,N) ;
VALEUR NO,A,B,I,J,K,L,M,N ; ENTIER NO,A,B,I,J,K,L,M,N ;
   DEBUT ENTIER U ; BOOLEEN BON ;
   AIGUILLAGE CAS:=DEBUT DE RECUL CAS 1,
   RECHERCHE DE REGLES CAS 2,REDUCTION CAS 3,CAS 4 ;
   SI NO=0 ALORS ALLERA  INITIALISATION CAS 0 ;
RECOPIE:
   ECRIRE( " BIFURCATION " ) ; U:=J-I ;
   POUR V:=1 PAS 1 JUSQUA U FAIRE P[J+V]:=P[I+V] ;
   I:=J ; J:=J+U ; M:=M+U ; U:=L-K ;
   POUR V:=1 PAS 1 JUSQUA U FAIRE P[K-V]:=P[L-V] ;
   L:=K ; K:=K-U ; ALLERA  CAS[NO] ;
INITIALISATION CAS 0:
   STOP:=P[1]:=PROCHAINSYMBOLE ;
   POUR U:=2,U+1 TANTQUE  P[U-1] ≠ STOP FAIRE
      DEBUT P[U]:=PROCHAINSYMBOLE ; I:=U FIN ;
   A:=P[1] ; B:=P[2] ; J:=I ; K:=L:=PMAX ; N:=3 ;
AVANCEMENT:
   ECRIRE( " AVANCEMENT " ,B) ; J:=J+1 ; P[J]:=A ; A:=B ;
   SI K=L ALORS DEBUT B:=P[N] ; N:=N+1 FIN
   SINON            DEBUT B:=P[K] ; K:=K+1 FIN ;
TEST D AVANCEMENT:
   SI A=STOP ALORS ALLERA  AVANCEMENT ;
   SI B=STOP ALORS ALLERA  DEBUT DE RECUL CAS 1 ;
   SI MI[A,B] OU ME[A,B] ALORS
      DEBUT SI MS[A,B] ALORS ANALYSE(1,A,B,I,J,K,L,M,N) ;
      ALLERA  AVANCEMENT FIN ;
   SI NON MS[A,B] ALORS ALLERA  IMPASSE ;
DEBUT DE RECUL CAS 1:
   J:=J+1 ; P[J]:=A ; K:=K-1 ; P[K]:=B ; M:=J ;
RECUL:
   ECRIRE( " RECUL " ,A) ; B:=A ; M:=M-1 ; A:=P[M] ;
TEST DE RECUL:
   SI A=STOP ALORS ALLERA  RECHERCHE DE REGLES CAS 2 ;
   SI ME[A,B] ALORS
      DEBUT SI MI[A,B] ALORS ANALYSE(2,A,B,I,J,K,L,M,N) ;
      ALLERA  RECUL FIN ;
RECHERCHE DE REGLES CAS 2:
   BON:= FAUX ;
   NO:= SI A=STOP ALORS 3 SINON SI MS[A,B] ALORS 4 SINON 3 ;
   POUR U:=2,U+R[U]+2 TANTQUE  R[U-1] ≠ 0 FAIRE
   SI R[U]=J-M ALORS
      DEBUT POUR V:=1 PAS 1 JUSQUA J-M FAIRE
      SI P[M+V] ≠ R[U+V] ALORS ALLERA  RIEN ;
      SI NO=4 ALORS DEBUT SI NON MI[A,R[U-1]] ET NON
                          ME[A,R[U-1]] ALORS ALLERA  RIEN FIN ;
      SI BON ALORS ANALYSE(NO,A,B,I,J,K,L,M,N) SINON BON:= VRAI ;
      B:=R[U-1] ; RIEN: FIN ;
```

```
   SI NON BON ALORS ALLERA  IMPASSE ;
REDUCTION CAS 3: CAS 4:
   ECRIRE( " REDUCTION " ,B) ; J:=M-1 ;
   SI B ≠ AXIOME OU A ≠ STOP OU P[K] ≠ STOP ALORS
   ALLERA  SI NO=4 ALORS AVANCEMENT SINON TEST D AVANCEMENT ;
   ECRIRE( " CHAINE CORRECTE " ) ;
IMPASSE:
   ECRIRE( " IMPASSE " ) FIN DE LA PROCEDURE ;

ANALYSE(0,0,0,0,0,0,0,0,0) FIN
```

4.5 Analyseur de précédence de gauche à droite

L'analyseur de précédence de gauche à droite que nous présentons ici correspond à la variante plus générale décrite au paragraphe 3.6. Du point de vue de déterminisme il est un peu plus général qu'un analyseur déterministe, tout en ne procédant pas par essais successifs. Nous imposons que les relations $\overset{\cdot}{<},\overset{\cdot}{=},\overset{\cdot}{>}$ soient incompatibles deux à deux, mais nous n'imposons pas que toutes les règles aient des membres droits distincts. Cette dernière restriction est remplacée par une condition plus pratique à satisfaire.

(1) Les règles ayant un opérande comme membre gauche n'ont pas besoin d'avoir des membres droits distincts.

(2) Si deux règles ont le même opérateur R comme membre gauche et sont de la forme

$$R \to E_o R_1 E_1 \ldots R_n E_n$$
$$R \to E'_o R_1 E'_1 \ldots R_n E'_n$$

avec $R_1, R_2, \ldots, R_n \in V_R$ et $E_o, E_1, \ldots, E_n \in V_E \cup \{\Lambda\}$, alors il n'existe pas

$E''_o, E''_1, \ldots E''_n \in V_E \cup \{\Lambda\}$ avec $E_i \not\Rightarrow {}^* E''_i$ et $E'_i \not\Rightarrow {}^* E''_i$ pour tout $i \in \{o,1,\ldots,n\}$.

Il faut remarquer que cet analyseur accepte certaines ambiguités mineures dans la grammaire ce qu'un analyseur rigoureusement déterministe ne pourrait faire.

```
    DEBUT   ENTIER   TABLEAU  P[1:PMAX],R[1:RMAX] ;
   BOOLEEN TABLEAU ME,MS[1:NBOPERATEURS,1:NBOPERATEURS],
   SU[NBOPERATEURS+1:NBSYMBOLES,NBOPERATEURS+1:NBSYMBOLES] ;
   ENTIER A,AXIOME,B,C,I,J,K,LR,N,STOP,U,V,W ; BOOLEEN FINI ;

INITIALISATION:
   I:=J:=1 ; K:=PMAX+1 ; FINI:= FAUX ;
   A:=P[J]:=STOP:=PROCHAINSYMBOLE ; B:=PROCHAINSYMBOLE ;
AVANCEMENT:
   ECRIRE( " AVANCEMENT " ,B) ; I:=J:=J+1 ; A:=P[J]:=B ;
   SI K > PMAX ALORS B:=PROCHAINSYMBOLE SINON
      DEBUT B:=P[K] ; K:=K+1 FIN ;
TEST D AVANCEMENT:
   SI A=STOP ALORS ALLERA  AVANCEMENT ;
   SI B=STOP ALORS ALLERA  DEBUT DE RECUL ;
   SI NON MS[A,B] ALORS ALLERA  AVANCEMENT ;
DEBUT DE RECUL:
   LR:= SI I=J ALORS 0 SINON 1 ; K:=K-1 ; P[K]:=B ;
RECUL:
   ECRIRE( " RECUL " ) ; C:=A ; I:=I-1 ; A:=P[I] ; LR:=LR+1 ;
   SI A > NBOPERATEURS ALORS
      DEBUT LR:=LR+1 ; E1: I:=I-1 ; A:=P[I] ;
      SI A > NBOPERATEURS ALORS ALLERA  E1 FIN ;
TEST DE RECUL:
   SI A=STOP ALORS ALLERA  RECHERCHE DE REGLES ;
   SI ME[A,C] ALORS ALLERA  RECUL ;
RECHERCHE DE REGLES:
   N:=0 ; POUR U:=2,U+R[U]+2 TANTQUE  R[U-1] ≠ 0 FAIRE
   SI LR=R[U] ALORS
      DEBUT V:=J ; POUR W:=U+LR PAS -1 JUSQUA U+1 FAIRE
         DEBUT SI R[W] ≤ NBOPERATEURS ALORS
         ALLERA  SI R[W]=P[V] ALORS E4 SINON E6 ;
         SI P[V] ≤ NBOPERATEURS ALORS ALLERA  E6 ;
         E2: SI SU[R[W],P[V]] ALORS ALLERA  E3 ;
         V:=V-1 ;
         ALLERA  SI P[V] > NBOPERATEURS ALORS E2 SINON E6;
         E3: V:=V-1 ;
         ALLERA  SI P[V] > NBOPERATEURS ALORS E3 SINON E5;
         E4: V:=V-1 ; E5: FIN ;
      N:=N+1 ; P[K-N]:=R[U-1] ; E6: FIN ;
REDUCTION:
   SI N=0 ALORS ALLERA  ERREUR ;
   SI P[K-1] ≤ NBOPERATEURS ALORS
      DEBUT ECRIRE( " R-REDUCTION " ,P[K-1]) ;
      SI A=STOP ET B=STOP ET P[K-1]=AXIOME ALORS ALLERA  BON ;
      J:=I ; B:=P[K-1] ; ALLERA  TEST D AVANCEMENT FIN ;
   SI A=STOP ET A=B ET AXIOME > NBOPERATEURS ALORS FINI:= VRAI ;
   POUR U:=1 PAS 1 JUSQUA N FAIRE
      DEBUT ECRIRE( " E-REDUCTION " ,P[K-U]) ; P[I+U]:=P[K-U] ;
      SI FINI ALORS
         DEBUT SI SU[AXIOME,P[K-U]] ALORS ALLERA  BON FIN
      FIN ;
```

```
   J:=I+N ; K:=K+1 ;
   SI NON FINI ALORS ALLERA  TEST D AVANCEMENT ;
ERREUR:
   ECRIRE( " ERREUR " ) ; ALLERA  FIN D ANALYSE ;
BON:
   ECRIRE( " CHAINE CORRECTE " ) ;
FIN D ANALYSE:
   FIN
```

4.6 Analyseur de précédence terminale

L'analyseur de précédence terminale que nous décrivons est un cas particulier de l'analyseur décrit au paragraphe précédent. Il s'applique aux grammaires dont les règles ne contiennent pas deux symboles non terminaux adjacents. Le vocabulaire terminal est choisi comme ensemble d'opérateurs et de ce fait la seule autre restriction qui subsiste est l'existence de relations de précédence terminales incompatibles deux à deux. NBOPERATEURS est donc aussi égal au nombre de symboles terminaux de la grammaire. Comme dans le cas précédent cet analyseur accepte aussi des ambiguités mineures dans la grammaire.

Il faut remarquer que du fait que les symboles non terminaux n'interviennent jamais dans les relations de précédence ils ne jouent qu'un rôle secondaire dans l'analyse. Il est donc possible de décrire un analyseur qui n'aurait pas besoin des règles de la grammaire. Cet analyseur trouverait la structure de toute phrase mais malheureusement ne reconnaîtrait pas toujours si celle-ci est correcte.

De plus si l'on représentait la structure obtenue par un arbre syntaxique, on ne connaîtrait pas le symbole non terminal qui se trouve à chaque noeud. C'est sous cette forme que Floyd [F1.1] avait présenté l'analyseur de précédence terminale.

```
    DEBUT  ENTIER  TABLEAU  P[1:PMAX],R[1:RMAX] ;
    BOOLEEN TABLEAU ME,MS[1:NBOPERATEURS,1:NBOPERATEURS],
    SU[NBOPERATEURS+1:NBSYMBOLES,NBOPERATEURS+1:NBSYMBOLES] ;
    ENTIER A,AXIOME,B,C,I,J,LR,N,STOP,U,V,W ; BOOLEEN FINI ;

INITIALISATION:
   I:=J:=1 ; FINI:= FAUX ;
   A:=P[J]:=STOP:=PROCHAINSYMBOLE ; B:=PROCHAINSYMBOLE ;
AVANCEMENT:
   ECRIRE(" AVANCEMENT ",B) ;
   I:=J:=J+1 ; A:=P[J]:=B ; B:=PROCHAINSYMBOLE ;
TEST D AVANCEMENT:
   SI A=STOP ALORS ALLERA AVANCEMENT ;
   SI B=STOP ALORS ALLERA DEBUT DE RECUL ;
   SI NON MS[A,B] ALORS ALLERA AVANCEMENT ;
DEBUT DE RECUL:
   LR:= SI I=J ALORS 0 SINON 1 ;
RECUL:
   ECRIRE(" RECUL ") ; C:=A ; I:=I-1 ; A:=P[I] ; LR:=LR+1 ;
   SI A > NBOPERATEURS ALORS
      DEBUT LR:=LR+1 ; E1: I:=I-1 ; A:=P[I] ;
      SI A > NBOPERATEURS ALORS ALLERA E1 FIN ;
TEST DE RECUL:
   SI A=STOP ALORS ALLERA RECHERCHE DE REGLES ;
   SI ME[A,C] ALORS ALLERA RECUL ;
RECHERCHE DE REGLES:
   N:=0 ; POUR U:=2,U+R[U]+2 TANTQUE R[U-1] ≠ 0 FAIRE
   SI LR=R[U] ALORS
      DEBUT V:=J ; POUR W:=U+LR PAS -1 JUSQUA U+1 FAIRE
         DEBUT SI R[W] ≤ NBOPERATEURS ALORS
         ALLERA SI R[W]=P[V] ALORS E4 SINON E6 ;
         SI P[V] ≤ NBOPERATEURS ALORS ALLERA E6 ;
         E2: SI SU[R[W],P[V]] ALORS ALLERA E3 ;
         V:=V-1 ;
         ALLERA SI P[V] > NBOPERATEURS ALORS E2 SINON E6;
         E3: V:=V-1 ;
         ALLERA SI P[V] > NBOPERATEURS ALORS E3 SINON E5;
         E4: V:=V-1 ; E5: FIN ;
      N:=N+1 ; P[J+N]:=R[U-1] ; E6: FIN ;
REDUCTION:
   SI N=0 ALORS ALLERA ERREUR ;
   SI A=STOP ET A=B ET AXIOME > NBOPERATEURS ALORS FINI:= VRAI ;
   POUR U:=1 PAS 1 JUSQUA N FAIRE
      DEBUT ECRIRE(" REDUCTION ",P[J+U]) ; P[I+U]:=P[J+U] ;
      SI FINI ALORS
         DEBUT SI SU[AXIOME,P[J+U]] ALORS ALLERA BON FIN
      FIN ;
   J:=I+N ;
   SI NON FINI ALORS ALLERA TEST D AVANCEMENT ;
ERREUR:
   ECRIRE(" ERREUR ") ; ALLERA FIN D ANALYSE ;
BON:
   ECRIRE(" CHAINE CORRECTE ") ;
FIN D ANALYSE:
   FIN
```

CHAPITRE 5 : Applications aux langages de programmation

5.1 Précédence totale et Algol 60

Le langage de programmation Algol a été défini au moyen d'une grammaire "context-free" (ou sous forme normale de Backus). Cette grammaire est malheureusement ambiguë du fait qu'elle ne permet pas d'exprimer le lien entre l'apparition d'un identificateur et sa déclaration, ce qui entraîne parfois plusieurs interprétation possibles. Afin de se ramener à une grammaire non ambiguë, nous avons été amené à introduire quatre nouveaux symboles terminaux qui sont :

IDENTIFICATEUR-ARITHMETIQUE
IDENTIFICATEUR-BOOLEEN
IDENTIFICATEUR-DE-DESIGNATION
IDENTIFICATEUR-DE-PROCEDURE-INSTRUCTION

Après cette modification et quelques autres n'altérant en rien le langage Algol, nous avons obtenu une grammaire de précédence totale [*] dont nous donnons la liste des règles et les relations de précédence totale aux pages suivantes. La seule restriction introduite est l'interdiction des étiquettes purement numériques, ce que d'ailleurs la plupart des compilateurs Algol interdisent aussi.

Pour l'écriture des règles de grammaire nous avons adopté les conventions suivantes :

- Chaque symbole terminal (symbole de base) ou non terminal (métavariable) est représenté par une suite quelconque de caractères ne comportant aucun espace libre ; les espaces servent de séparateurs.

- Lorsqu'une règle a un membre gauche identique à celui de la règle précédente, celui-ci est supprimé et le signe → est remplacé par le mot OU (ou OR).

---

[*] Du fait que dans cette grammaire les symboles terminaux 'CROCHET-GAUCHE' et 1 sont à la fois dans la relation $\alpha \gamma^+$ et $\delta^+ \alpha \gamma^+$ celle-ci n'est pas de précédence totale de gauche à droite.

- Le signe → est remplacé par ::=.

Dans le tableau représentant des relations de précédence les signes $\lessdot, \doteq, \gtrdot$ sont respectivement remplacés par 1,2,4. Si le tableau est trop grand pour tenir sur une seule page, il est découpé en plusieurs bandes verticales.

PL nº 0 GRAMMAIRE ALGOL DE PRECEDENCE TOTALE

REGLES

```
                                PROGRAMME ::= IDENTIFICATEUR : PROGRAMME
                                          OU  PROGRAMME-INCOMPLET INSTRUCTION-FINALE
                           INSTRUCTION-FINALE ::= 'FIN'
                                          OU  INSTRUCTION 'FIN'
                          PROGRAMME-INCOMPLET ::= DECLARATIONS
                                          OU  PROGRAMME-INCOMPLET INSTRUCTION-NON-FINALE
                       INSTRUCTION-NON-FINALE ::= .,
                                          OU  INSTRUCTION .,
                                DECLARATIONS ::= 'DEBUT'
                                          OU  DECLARATIONS DECLARATION-AUTRE-QUE-DE-PROCEDURE .,
                                          OU  DECLARATIONS DECLARATION-OU-SPECIFICATION-DE-TYPE .,
                                          OU  DECLARATIONS-INCOMPLETES .,
                                          OU  DECLARATIONS-INCOMPLETES INSTRUCTION .,
                     DECLARATIONS-INCOMPLETES ::= DECLARATIONS-BIEN-INCOMPLETES .,
                                          OU  DECLARATIONS-INCOMPLETES SPECIFICATION .,
                                          OU  DECLARATIONS-INCOMPLETES DECLARATION-OU-SPECIFICATION-DE-TYPE .,
                                          OU  DECLARATIONS-INCOMPLETES ANNONCE-DE-PROCEDURE .,
                  DECLARATIONS-BIEN-INCOMPLETES ::= DECLARATIONS-TRES-INCOMPLETES .,
                                          OU  DECLARATIONS-TRES-INCOMPLETES PARTIE-VALEUR .,
                                PARTIE-VALEUR ::= 'VALEUR' LISTE-D'IDENTIFICATEURS
                  DECLARATIONS-TRES-INCOMPLETES ::= DECLARATIONS ANNONCE-DE-PROCEDURE .,
                                          OU  DECLARATIONS ANNONCE-DE-PROCEDURE PARTIE-FORMELLE .,
                             ANNONCE-DE-PROCEDURE ::= ANNONCE-DE-PROCEDURE-BIS
                         ANNONCE-DE-PROCEDURE-BIS ::= 'PROCEDURE' IDENTIFICATEUR-BIS
                                          OU  TYPE 'PROCEDURE' IDENTIFICATEUR-BIS
                               PARTIE-FORMELLE ::= ( PARAMETRES-FORMELS )
                               PARAMETRES-FORMELS ::= LISTE-D'IDENTIFICATEURS
                 DECLARATION-AUTRE-QUE-DE-PROCEDURE ::= DECLARATION-DE-TYPE-REMANENT
                                          OU  DECLARATION-DE-TABLEAU
                                          OU  DECLARATION-D'AIGUILLAGE
                 DECLARATION-OU-SPECIFICATION-DE-TYPE ::= DECLARATEUR-DE-TYPE LISTE-D'IDENTIFICATEURS
                      DECLARATION-DE-TYPE-REMANENT ::= 'REMANENT' DECLARATEUR-DE-TYPE LISTE-D'IDENTIFICATEURS
                               DECLARATEUR-DE-TYPE ::= TYPE
                                             TYPE ::= 'ENTIER'
                                          OU  'REEL'
                                          OU  'BOOLEEN'
                           DECLARATION-DE-TABLEAU ::= DECLARATEUR-DE-TABLEAU LISTE-DE-TABLEAUX
                                          OU  'REMANENT' DECLARATEUR-DE-TABLEAU LISTE-DE-TABLEAUX
                           DECLARATEUR-DE-TABLEAU ::= 'TABLEAU'
                                          OU  TYPE 'TABLEAU'
                              LISTE-DE-TABLEAUX ::= IDENTIFICATEUR-AVEC-BORNES
                                          OU  IDENTIFICATEUR , LISTE-DE-TABLEAUX
                                          OU  IDENTIFICATEUR-AVEC-BORNES , LISTE-DE-TABLEAUX
                      IDENTIFICATEUR-AVEC-BORNES ::= IDENTIFICATEUR 'CROCHET-GAUCHE' PAIRES-DE-BORNES
                                PAIRES-DE-BORNES ::= EXPRESSION-ARITHMETIQUE : EXPRESSION-ARITHMETIQUE 'CROCHET-DROIT'
                                          OU  EXPRESSION-ARITHMETIQUE : EXPRESSION-ARITHMETIQUE , PAIRES-DE-BORNES
                        DECLARATION-D'AIGUILLAGE ::= 'AIGUILLAGE' FIN-DE-DECLARATION-D'AIGUILLAGE
                    FIN-DE-DECLARATION-D'AIGUILLAGE ::= IDENTIFICATEUR := LISTE-D'EXPRESSIONS-DE-DESIGNATION
                 LISTE-D'EXPRESSIONS-DE-DESIGNATION ::= EXPRESSION-DE-DESIGNATION
                                          OU  EXPRESSION-DE-DESIGNATION , LISTE-D'EXPRESSIONS-DE-DESIGNATION
                                SPECIFICATION ::= DECLARATEUR-DE-TABLEAU LISTE-D'IDENTIFICATEURS
                                          OU  'AIGUILLAGE' LISTE-D'IDENTIFICATEURS
                                          OU  ANNONCE-DE-PROCEDURE-BIS , LISTE-D'IDENTIFICATEURS
                                          OU  'ETIQUETTE' LISTE-D'IDENTIFICATEURS
                                          OU  'CHAINE' LISTE-D'IDENTIFICATEURS
                                  INSTRUCTION ::= INSTRUCTION-QUELCONQUE
                          INSTRUCTION-QUELCONQUE ::= INSTRUCTION-INCONDITIONNELLE
                                          OU  INSTRUCTION-CONDITIONNELLE
                                          OU  INSTRUCTION-POUR
                      INSTRUCTION-INCONDITIONNELLE ::= INSTRUCTION-DE-BASE
                                          OU  PROGRAMME
                                INSTRUCTION-DE-BASE ::= INSTRUCTION-D'AFFECTATION-ARITHMETIQUE
                                          OU  INSTRUCTION-D'AFFECTATION-BOOLEENNE
                                          OU  INSTRUCTION-ALLER-A
                                          OU  INSTRUCTION-PROCEDURE
                                          OU  IDENTIFICATEUR :
                                          OU  IDENTIFICATEUR : INSTRUCTION-DE-BASE
INSTRUCTION-D'AFFECTATION-ARITHMETIQUE ::= PARTIE-GAUCHE-ARITHMETIQUE := PARTIE-DROITE-ARITHMETIQUE
                                          OU  PARTIE-GAUCHE-ARITHMETIQUE := INSTRUCTION-D'AFFECTATION-ARITHMETIQUE
          PARTIE-DROITE-ARITHMETIQUE ::= EXPRESSION-ARITHMETIQUE
    INSTRUCTION-D'AFFECTATION-BOOLEENNE ::= PARTIE-GAUCHE-BOOLEENNE := PARTIE-DROITE-BOOLEENNE
                                          OU  PARTIE-GAUCHE-BOOLEENNE := INSTRUCTION-D'AFFECTATION-BOOLEENNE
          PARTIE-DROITE-BOOLEENNE ::= EXPRESSION-BOOLEENNE
                      INSTRUCTION-ALLER-A ::= 'ALLER A' EXPRESSION-DE-DESIGNATION
                    INSTRUCTION-PROCEDURE ::= IDENTIFICATEUR-DE-PROCEDURE-INSTRUCTION
                                          OU  DEBUT-D'INSTRUCTION-PROCEDURE PARAMETRES-EFFECTIFS
        DEBUT-D'INSTRUCTION-PROCEDURE ::= IDENTIFICATEUR-DE-PROCEDURE-INSTRUCTION (
                      PARAMETRES-EFFECTIFS ::= FIN-DE-PRIMAIRE-ARITHMETIQUE
                                          OU  FIN-DE-PRIMAIRE-BOOLEEN
                                          OU  FIN-D'EXPRESSION-DE-DESIGNATION-SIMPLE
                                          OU  IDENTIFICATEUR-DE-PROCEDURE-INSTRUCTION )
                                          OU  CHAINE )
                                          OU  EXPRESSION-ARITHMETIQUE , PARAMETRES-EFFECTIFS
                                          OU  EXPRESSION-BOOLEENNE , PARAMETRES-EFFECTIFS
                                          OU  EXPRESSION-DE-DESIGNATION , PARAMETRES-EFFECTIFS
                                          OU  IDENTIFICATEUR-DE-PROCEDURE-INSTRUCTION , PARAMETRES-EFFECTIFS
                                          OU  CHAINE , PARAMETRES-EFFECTIFS
```

PL nº 1 GRAMMAIRE ALGOL DE PRECEDENCE TOTALE

REGLES (SUITE)

```
INSTRUCTION-CONDITIONNELLE ::= INSTRUCTION-SI
                           OU  INSTRUCTION-SI 'SINON' INSTRUCTION-QUELCONQUE
                           OU  INSTRUCTION-SI 'SINON'
                           OU  PROPOSITION-SI INSTRUCTION-POUR
                           OU  IDENTIFICATEUR : INSTRUCTION-CONDITIONNELLE
          INSTRUCTION-SI   ::= PROPOSITION-SI INSTRUCTION-INCONDITIONNELLE
                           OU  PROPOSITION-SI
          PROPOSITION-SI   ::= 'SI' EXPRESSION-BOOLEENNE 'ALORS'
          INSTRUCTION-POUR ::= IDENTIFICATEUR : INSTRUCTION-POUR
                           OU  PROPOSITION-POUR
                           OU  PROPOSITION-POUR INSTRUCTION-QUELCONQUE
          PROPOSITION-POUR ::= 'POUR' PARTIE-GAUCHE-ARITHMETIQUE := FIN-DE-PROPOSITION-POUR
      FIN-DE-PROPOSITION-POUR ::= EXPRESSION-ARITHMETIQUE 'FAIRE'
                           OU  EXPRESSION-ARITHMETIQUE 'PAS' EXPRESSION-ARITHMETIQUE 'JUSQUA'
                               EXPRESSION-ARITHMETIQUE 'FAIRE'
                           OU  'TANTQUE' EXPRESSION-BOOLEENNE 'FAIRE'
                           OU  EXPRESSION-ARITHMETIQUE , FIN-DE-PROPOSITION-POUR
                           OU  EXPRESSION-ARITHMETIQUE 'PAS' EXPRESSION-ARITHMETIQUE 'JUSQUA'
                               EXPRESSION-ARITHMETIQUE , FIN-DE-PROPOSITION-POUR
                           OU  'TANTQUE' EXPRESSION-BOOLEENNE , FIN-DE-PROPOSITION-POUR
     EXPRESSION-ARITHMETIQUE ::= EXPRESSION-ARITHMETIQUE-BIS
   EXPRESSION-ARITHMETIQUE-BIS ::= EXPRESSION-ARITHMETIQUE-SIMPLE
                           OU  DEBUT-D'EXPRESSION-ARITHMETIQUE 'SINON' EXPRESSION-ARITHMETIQUE-BIS
DEBUT-D'EXPRESSION-ARITHMETIQUE ::= PROPOSITION-SI EXPRESSION-ARITHMETIQUE-SIMPLE
  EXPRESSION-ARITHMETIQUE-SIMPLE ::= EXPRESSION-ARITHMETIQUE-SIMPLE-BIS
EXPRESSION-ARITHMETIQUE-SIMPLE-BIS ::= EXPRESSION-ARITHMETIQUE-SIMPLE-TER
EXPRESSION-ARITHMETIQUE-SIMPLE-TER ::= TERME
                           OU  OPERATEUR-ADDITIF TERME
                           OU  EXPRESSION-ARITHMETIQUE-SIMPLE-TER OPERATEUR-ADDITIF TERME
        OPERATEUR-ADDITIF  ::= +
                           OU  -
                TERME      ::= TERME-BIS
              TERME-BIS    ::= FACTEUR
                           OU  TERME-BIS OPERATEUR-MULTIPLICATIF FACTEUR
   OPERATEUR-MULTIPLICATIF ::= *
                           OU  /
                           OU  'OPERATEUR-DE-DIVISION-ENTIERE'
              FACTEUR      ::= FACTEUR-BIS
            FACTEUR-BIS    ::= PRIMAIRE-ARITHMETIQUE
                           OU  FACTEUR-BIS 'OPERATEUR-D'EXPONENTIATION' PRIMAIRE-ARITHMETIQUE
     PRIMAIRE-ARITHMETIQUE ::= NOMBRE
                           OU  PARTIE-GAUCHE-ARITHMETIQUE
                           OU  FONCTION-ARITHMETIQUE
                           OU  PARENTHESE-GAUCHE FIN-DE-PRIMAIRE-ARITHMETIQUE
        PARENTHESE-GAUCHE  ::= (
   FIN-DE-PRIMAIRE-ARITHMETIQUE ::= EXPRESSION-ARITHMETIQUE )
                NOMBRE     ::= NOMBRE-DECIMAL
                           OU  OPERATEUR-D'EXPONENTIATION-DECIMALE ENTIER
                           OU  NOMBRE-DECIMAL OPERATEUR-D'EXPONENTIATION-DECIMALE ENTIER
OPERATEUR-D'EXPONENTIATION-DECIMALE ::= '10'
                           OU  '10' +
                           OU  '10' -
         NOMBRE-DECIMAL    ::= ENTIER
                           OU  . ENTIER
                           OU  ENTIER . ENTIER
                ENTIER     ::= CHIFFRES
              CHIFFRES     ::= CHIFFRE
                           OU  CHIFFRES CHIFFRE
     PARTIE-GAUCHE-ARITHMETIQUE ::= IDENTIFICATEUR-ARITHMETIQUE
                           OU  DEBUT-DE-PARTIE-GAUCHE-ARITHMETIQUE INDICES
DEBUT-DE-PARTIE-GAUCHE-ARITHMETIQUE ::= IDENTIFICATEUR-ARITHMETIQUE 'CROCHET-GAUCHE'
        FONCTION-ARITHMETIQUE ::= DEBUT-DE-FONCTION-ARITHMETIQUE PARAMETRES-EFFECTIFS
DEBUT-DE-FONCTION-ARITHMETIQUE ::= IDENTIFICATEUR-ARITHMETIQUE (
          EXPRESSION-BOOLEENNE ::= EXPRESSION-BOOLEENNE-BIS
   EXPRESSION-BOOLEENNE-BIS ::= EXPRESSION-BOOLEENNE-SIMPLE
                           OU  DEBUT-D'EXPRESSION-BOOLEENNE 'SINON' EXPRESSION-BOOLEENNE-BIS
DEBUT-D'EXPRESSION-BOOLEENNE ::= PROPOSITION-SI EXPRESSION-BOOLEENNE-SIMPLE
  EXPRESSION-BOOLEENNE-SIMPLE ::= EXPRESSION-BOOLEENNE-SIMPLE-BIS
EXPRESSION-BOOLEENNE-SIMPLE-BIS ::= IMPLICATION
                           OU  EXPRESSION-BOOLEENNE-SIMPLE-BIS 'EQUIVALENT' IMPLICATION
          IMPLICATION      ::= IMPLICATION-BIS
        IMPLICATION-BIS    ::= TERME-BOOLEEN
                           OU  IMPLICATION-BIS 'IMPLIQUE' TERME-BOOLEEN
           TERME-BOOLEEN   ::= TERME-BOOLEEN-BIS
         TERME-BOOLEEN-BIS ::= FACTEUR-BOOLEEN
                           OU  TERME-BOOLEEN-BIS 'OU' FACTEUR-BOOLEEN
         FACTEUR-BOOLEEN   ::= FACTEUR-BOOLEEN-BIS
       FACTEUR-BOOLEEN-BIS ::= SECONDAIRE-BOOLEEN
                           OU  FACTEUR-BOOLEEN-BIS 'ET' SECONDAIRE-BOOLEEN
       SECONDAIRE-BOOLEEN  ::= PRIMAIRE-BOOLEEN
                           OU  'NON' PRIMAIRE-BOOLEEN
        PRIMAIRE-BOOLEEN   ::= 'VRAI'
                           OU  'FAUX'
                           OU  PARTIE-GAUCHE-BOOLEENNE
                           OU  RELATION
                           OU  FONCTION-BOOLEENNE
                           OU  PARENTHESE-GAUCHE FIN-DE-PRIMAIRE-BOOLEEN
```

PL nº 2 GRAMMAIRE ALGOL DE PRECEDENCE TOTALE

REGLES (SUITE)

```
FIN-DE-PRIMAIRE-BOOLEEN ::= EXPRESSION-BOOLEENNE )
RELATION ::= EXPRESSION-ARITHMETIQUE-SIMPLE-BIS OPERATEUR-DE-RELATION
EXPRESSION-ARITHMETIQUE-SIMPLE-BIS
OPERATEUR-DE-RELATION ::= 'INFERIEUR'
OU 'INFERIEUR-OU-EGAL'
OU =
OU 'NON-EGAL'
OU 'SUPERIEUR-OU-EGAL'
OU 'SUPERIEUR'
FONCTION-BOOLEENNE ::= DEBUT-DE-FONCTION-BOOLEENNE PARAMETRES-EFFECTIFS
DEBUT-DE-FONCTION-BOOLEENNE ::= IDENTIFICATEUR-BOOLEEN (
PARTIE-GAUCHE-BOOLEENNE ::= IDENTIFICATEUR-BOOLEEN
OU DEBUT-DE-PARTIE-GAUCHE-BOOLEENNE INDICES
DEBUT-DE-PARTIE-GAUCHE-BOOLEENNE ::= IDENTIFICATEUR-BOOLEEN 'CROCHET-GAUCHE'
EXPRESSION-DE-DESIGNATION ::= EXPRESSION-DE-DESIGNATION-BIS
EXPRESSION-DE-DESIGNATION-BIS ::= EXPRESSION-DE-DESIGNATION-SIMPLE
OU DEBUT-D'EXPRESSION-DE-DESIGNATION 'SINON' EXPRESSION-DE-DESIGNATION-BIS
DEBUT-D'EXPRESSION-DE-DESIGNATION ::= PROPOSITION-SI EXPRESSION-DE-DESIGNATION-SIMPLE
EXPRESSION-DE-DESIGNATION-SIMPLE ::= IDENTIFICATEUR-DE-DESIGNATION
OU INDICATEUR-D'AIGUILLAGE INDICE-FINAL
OU PARENTHESE-GAUCHE FIN-D'EXPRESSION-DE-DESIGNATION-SIMPLE
FIN-D'EXPRESSION-DE-DESIGNATION-SIMPLE ::= EXPRESSION-DE-DESIGNATION )
INDICATEUR-D'AIGUILLAGE ::= IDENTIFICATEUR-DE-DESIGNATION 'CROCHET-GAUCHE'
INDICES ::= INDICE-FINAL
OU EXPRESSION-ARITHMETIQUE , INDICES
INDICE-FINAL ::= EXPRESSION-ARITHMETIQUE 'CROCHET-DROIT'
LISTE-D'IDENTIFICATEURS ::= IDENTIFICATEUR
OU IDENTIFICATEUR , LISTE-D'IDENTIFICATEURS
IDENTIFICATEUR ::= IDENTIFICATEUR-BIS
IDENTIFICATEUR-BIS ::= IDENTIFICATEUR-TER
IDENTIFICATEUR-TER ::= LETTRE
OU IDENTIFICATEUR-TER LETTRE
OU IDENTIFICATEUR-TER CHIFFRE
CHIFFRE ::= 0
OU 1
OU 2
OU 3
OU 4
OU 5
OU 6
OU 7
OU 8
OU 9
LETTRE ::= A
OU B
OU C
OU D
OU E
OU F
OU G
OU H
OU I
OU J
OU K
OU L
OU M
OU N
OU O
OU P
OU Q
OU R
OU S
OU T
OU U
OU V
OU W
OU X
OU Y
OU Z
```

PL n° 3

GRAMMAIRE ALGOL DE PRECEDENCE TOTALE
RELATIONS DE PRECEDENCE TOTALE

Symbole	Code	Relations
		CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
		CCCCCCCCC1111111111222222222233333333334444444444555555555566666666667
		1234567890123456789012345678901234567890123456789012345678901234567890
:	C01	.1..11..1.111111111111111111111111.111111111114.11..4....1..14.....1..
(	C02	.4..44..4.444444444411111111111111.11111111111..44....................
)	C03	4.444444.4....................................4...4444.......4.4....4.
,	C04	.1..11..1.111111111111111111111111.11111111111..11....................
+	C05	.4......4.4444444444.............................4....................
-	C06	.4......4.4444444444.............................4....................
*	C07	.4......4.4444444444.............................4....................
/	C08	.4......4.4444444444.............................4....................
.	C09	..........1111111111..
=	C10	.4..44..4.4444444444.............................4....................
0	C11	4444444444444444444444444444444444.4444444444444.44444.......4.4....4.
1	C12	4444444444444444444444444444444444.4444444444444.44444.......4.4....4.
2	C13	4444444444444444444444444444444444.4444444444444.44444.......4.4....4.
3	C14	4444444444444444444444444444444444.4444444444444.44444.......4.4....4.
4	C15	4444444444444444444444444444444444.4444444444444.44444.......4.4....4.
5	C16	4444444444444444444444444444444444.4444444444444.44444.......4.4....4.
6	C17	4444444444444444444444444444444444.4444444444444.44444.......4.4....4.
7	C18	4444444444444444444444444444444444.4444444444444.44444.......4.4....4.
8	C19	4444444444444444444444444444444444.4444444444444.44444.......4.4....4.
9	C20	4444444444444444444444444444444444.4444444444444.44444.......4.4....4.
A	C21	4444......444444444444444444444444.4444444444444......................
B	C22	4444......444444444444444444444444.4444444444444......................
C	C23	4444......444444444444444444444444.4444444444444......................
D	C24	4444......444444444444444444444444.4444444444444......................
E	C25	4444......444444444444444444444444.4444444444444......................
F	C26	4444......444444444444444444444444.4444444444444......................
G	C27	4444......444444444444444444444444.4444444444444......................
H	C28	4444......444444444444444444444444.4444444444444......................
I	C29	4444......444444444444444444444444.4444444444444......................
J	C30	4444......444444444444444444444444.4444444444444......................
K	C31	4444......444444444444444444444444.4444444444444......................
L	C32	4444......444444444444444444444444.4444444444444......................
M	C33	4444......444444444444444444444444.4444444444444......................
N	C34	4444......444444444444444444444444.4444444444444......................
O	C35	4444......444444444444444444444444.4444444444444......................
P	C36	4444......444444444444444444444444.4444444444444......................
Q	C37	4444......444444444444444444444444.4444444444444......................
R	C38	4444......444444444444444444444444.4444444444444......................
S	C39	4444......444444444444444444444444.4444444444444......................
T	C40	4444......444444444444444444444444.4444444444444......................
U	C41	4444......444444444444444444444444.4444444444444......................
V	C42	4444......444444444444444444444444.4444444444444......................
W	C43	4444......444444444444444444444444.4444444444444......................
X	C44	4444......444444444444444444444444.4444444444444......................
Y	C45	4444......444444444444444444444444.4444444444444......................
Z	C46	4444......444444444444444444444444.4444444444444......................
.,	C47	....................44444444444444.444444444444.4...4..4.4..4....444..
:=	C48	.1..11..1.1111111111............................11....................
'SI'	C49	....1.....................
'10'	C50	....22....4444444444..
'OU'	C51	.1..11..1.1111111111.............................1....1...11..........
'ET'	C52	.1..11..1.1111111111.............................1....1...11..........
'FIN'	C53	....4.....4........4........
'PAS'	C54	.1..11..1.1111111111............................11....................
'NON'	C55	.1..11..1.1111111111.............................1........11..........
'REEL'	C56	....................44444444444444.44444444444........................
CHAINE	C57	..22..
'POUR'	C58	....
'VRAI'	C59	....44..................
'FAUX'	C60	....44..................
'DEBUT'	C61	....................44444444444444.444444444444.4...4..4.4..4....4.4..
'SINON'	C62	.1..11..1.111111111111111111111111.111111111114.11..4....1..1......1..
'ALORS'	C63	.4..44..4.444444444444444444444444.444444444444..4..4....4..44.....4..
'FAIRE'	C64	....................44444444444444.444444444444.4...4....4..4......4..
'VALEUR'	C65	....................11111111111111.11111111111........................
'ENTIER'	C66	....................44444444444444.44444444444........................
'CHAINE'	C67	....................11111111111111.11111111111........................
'ALLERA'	C68	.1..1.....................
'JUSQUA'	C69	.1..11..1.1111111111............................11....................
'BOOLEEN'	C70	....................44444444444444.44444444444........................
'TABLEAU'	C71	....................44444444444444.44444444444........................
'TANTQUE'	C72	....1.....................
'REMANENT'	C73	....1.........1....
'IMPLIQUE'	C74	.1..11..1.1111111111.............................1....1...11..........
'NON-EGAL'	C75	.4..44..4.4444444444.............................4....................
'PROCEDURE'	C76	....................11111111111111.11111111111........................
'ETIQUETTE'	C77	....................11111111111111.11111111111........................
'INFERIEUR'	C78	.4..44..4.4444444444.............................4....................
'SUPERIEUR'	C79	.4..44..4.4444444444.............................4....................
'AIGUILLAGE'	C80	....................11111111111111.11111111111........................
'EQUIVALENT'	C81	....
'CROCHET-DROIT'	C82	4.444444.4....................................44..4444.......4.4....4.
'CROCHET-GAUCHE'	C83	.1..11..1.1111111111............................11....................
'INFERIEUR-OU-EGAL'	C84	.4..44..4.4444444444.............................4....................
'SUPERIEUR-OU-EGAL'	C85	.4..44..4.4444444444.............................4....................
IDENTIFICATEUR-BOOLEEN	C86	.2...4..44..................
IDENTIFICATEUR-ARITHMETIQUE	C87	42444444.4....................................44..4444.......4.4....4.
'OPERATEUR-D'EXPONENTIATION'	C88	.1......1.1111111111.............................1....................
IDENTIFICATEUR-DE-DESIGNATION	C89	..44..4.....4........4........
'OPERATEUR-DE-DIVISION-ENTIERE'	C90	.4......4.4444444444.............................4....................
IDENTIFICATEUR-DE-PROCEDURE-INSTRUCTION	C91	.222..4.....4........4........
PROGRAMME	C92	....4.....4........4........
IDENTIFICATEUR	C93	2.42..42......................
PROGRAMME-INCOMPLET	C94	....................11111111111111.111111111111.1...1....1..1......1..
INSTRUCTION-FINALE	C95	....4.....4........4........
INSTRUCTION	C96	....2.....2.................
DECLARATIONS	C97	....................44444444444444.444444444444.4...4..1.4..4....1.4..

PL n°4 GRAMMAIRE ALGOL DE PRECEDENCE TOTALE

RELATIONS DE PRECEDENCE TOTALE (SUITE)

```
                    INSTRUCTION-NON-FINALE  098  ....................4444444444444.444444444444.4...4....4..4.....4..
  DECLARATION-AUTRE-QUE-DE-PROCEDURE        099  ....................................................2.....................
DECLARATION-OU-SPECIFICATION-DE-TYPE        100  ....................................................2.....................
                    DECLARATIONS-INCOMPLETES 101  ..................1111111111111.1111111111112.1.....1.1..1....111..
                    DECLARATIONS-BIEN-INCOMPLETES 102  ....................................................2.....................
                                SPECIFICATION 103  ....................................................2.....................
                                ANNONCE-DE-PROCEDURE 104  .1..................................................2.....................
                    DECLARATIONS-TRES-INCOMPLETES 105  ....................................................2.....1...............
                                PARTIE-VALEUR 106  ....................................................2.....................
                            LISTE-D'IDENTIFICATEURS 107  ..4...................................................4.....................
                                PARTIE-FORMELLE 108  ....................................................2.....................
                            ANNONCE-DE-PROCEDURE-BIS 109  .4.2.................................................4.....................
                                IDENTIFICATEUR-BIS 110  4444..................................................44....................
                                        TYPE 111  ..................44444444444444.44444444444.....................
                                PARAMETRES-FORMELS 112  ..2.....................................................4.....................
                    DECLARATION-DE-TYPE-REMANENT 113  ....................................................4.....................
                            DECLARATION-DE-TABLEAU 114  ....................................................4.....................
                            DECLARATION-D'AIGUILLAGE 115  ....................................................4.....................
                                DECLARATEUR-DE-TYPE 116  ..................1111111111111.1111111111.....................
                            DECLARATEUR-DE-TABLEAU 117  ..................1111111111111.1111111111.....................
                                LISTE-DE-TABLEAUX 118  ....................................................4.....................
                        IDENTIFICATEUR-AVEC-BORNES 119  ...2................................................4.....................
                                PAIRES-DE-BORNES 120  ...4..................................................4.....................
                                EXPRESSION-ARITHMETIQUE 121  2.22.....................................4.....42.......4.2....2.
                    FIN-DE-DECLARATION-D'AIGUILLAGE 122  ....................................................4.....................
                LISTE-D'EXPRESSIONS-DE-DESIGNATION 123  ....................................................4.....................
                            EXPRESSION-DE-DESIGNATION 124  ..22..................................4.....4.....4.....4.........
                        INSTRUCTION-QUELCONQUE 125  ....................................4.....4.....4.....4.....4.........
                    INSTRUCTION-INCONDITIONNELLE 126  ....................................4.....4.....4.....4.....4.........
                        INSTRUCTION-CONDITIONNELLE 127  ....................................4.....4.....4.....4.....4.........
                                INSTRUCTION-POUR 128  ....................................4.....4.....4.....4.....4.........
                            INSTRUCTION-DE-BASE 129  ....................................4.....4.....4.....4.....4.........
INSTRUCTION-D'AFFECTATION-ARITHMETIQUE      130  ....................................4.....4.....4.....4.....4.........
    INSTRUCTION-D'AFFECTATION-BOOLEENNE      131  ....................................4.....4.....4.....4.....4.........
                        INSTRUCTION-ALLER-A 132  ....................................4.....4.....4.....4.....4.........
                        INSTRUCTION-PROCEDURE 133  ....................................4.....4.....4.....4.....4.........
                    PARTIE-GAUCHE-ARITHMETIQUE 134  4.444444.4.....................................42..4444.....4.4....4.
                    PARTIE-DROITE-ARITHMETIQUE 135  ....................................4.....4.....4.....4.....4.........
                        PARTIE-GAUCHE-BOOLEENNE 136  ....................................2..44..................4.........
                        PARTIE-DROITE-BOOLEENNE 137  ....................................4.....4.....4.....4.....4.........
                                EXPRESSION-BOOLEENNE 138  ..22.....................................4.....4.....422.....4.........
                    DEBUT-D'INSTRUCTION-PROCEDURE 139  ....................................................4.....4.....4.....4.....
                            PARAMETRES-EFFECTIFS 140  4.444444.4.....................................4...4444.....4.4....4.
                    FIN-DE-PRIMAIRE-ARITHMETIQUE 141  4.444444.4.....................................4...4444.....4.4....4.
                        FIN-DE-PRIMAIRE-BOOLEEN 142  ....................................44....................4.....
FIN-D'EXPRESSION-DE-DESIGNATION-SIMPLE      143  ..44................................4.....4.....4.....4.....4.....
                                INSTRUCTION-SI 144  ....................................4.....4.....4.....4.....4.....
                                PROPOSITION-SI 145  .1..11..1.11111111111111111111111111.1111111114..1..4....1..14.....1..
                                PROPOSITION-POUR 146  ..................1111111111111.1111111111114.1...4....1..1.....1..
                    FIN-DE-PROPOSITION-POUR 147  ..................4444444444444.444444444444.4...4....4..4.....4..
                    EXPRESSION-ARITHMETIQUE-BIS 148  4.44.....................................4.....44.....4.4....4.
                    EXPRESSION-ARITHMETIQUE-SIMPLE 149  4.44.....................................4.....44.....4.4....4.
                    DEBUT-D'EXPRESSION-ARITHMETIQUE 150  ....................................................2.....4.....
                    EXPRESSION-ARITHMETIQUE-SIMPLE-BIS 151  4.44.....1.....................................4...4444.....4.4....4.
                    EXPRESSION-ARITHMETIQUE-SIMPLE-TER 152  4.4411...4.....................................4...4444.....4.4....4.
                                TERME 153  4.4444...4.....................................4...4444.....4.4....4.
                            OPERATEUR-ADDITIF 154  .1......1.1111111111.....................................1.....4.....
                                TERME-BIS 155  4.444411.4.....................................4...4444.....4.4....4.
                                FACTEUR 156  4.444444.4.....................................4...4444.....4.4....4.
                            OPERATEUR-MULTIPLICATIF 157  .1......1.1111111111.....................................1.....4.....
                                FACTEUR-BIS 158  4.444444.4.....................................4...4444.....4.4....4.
                            PRIMAIRE-ARITHMETIQUE 159  4.444444.4.....................................4...4444.....4.4....4.
                                NOMBRE 160  4.444444.4.....................................4...4444.....4.4....4.
                            FONCTION-ARITHMETIQUE 161  4.444444.4.....................................4...4444.....4.4....4.
                                PARENTHESE-GAUCHE 162  .1..11..1.1111111111.....................................11....4.....
                                NOMBRE-DECIMAL 163  4.444444.4.....................................4..14444.....4.4....4.
                    OPERATEUR-D'EXPONENTIATION-DECIMALE 164  ..........1111111111.....................................4.....
                                ENTIER 165  4.44444424.....................................4..44444.....4.4....4.
                                CHIFFRES 166  4.4444444111111111.....................................4..44444.....4.4....4.
                                CHIFFRE 167  4444444444444444444444444444444444.4444444444444.44444.....4.4....4.
                    DEBUT-DE-PARTIE-GAUCHE-ARITHMETIQUE 168  .1..11..1.1111111111.....................................11....4.....
                                INDICES 169  4.444444.4.....................................44..4444.....4.4....4.
                    DEBUT-DE-FONCTION-ARITHMETIQUE 170  ....................................................4.....4.....
                            EXPRESSION-BOOLEENNE-BIS 171  ..44..................................4.....4.....444.....
                        EXPRESSION-BOOLEENNE-SIMPLE 172  ..44..................................4.....4.....444.....
                        DEBUT-D'EXPRESSION-BOOLEENNE 173  ....................................4.....4.....444.....
                    EXPRESSION-BOOLEENNE-SIMPLE-BIS 174  ..44.....................................................2.....
                                IMPLICATION 175  ..44..................................4.....4.....444.....
                            IMPLICATION-BIS 176  ....................................4.....4.....444.....
                                TERME-BOOLEEN 177  ....................................................4.....
                            TERME-BOOLEEN-BIS 178  ....................................................4.....
                            FACTEUR-BOOLEEN 179  ....................................2.....4.....
                        FACTEUR-BOOLEEN-BIS 180  ....................................4.....4.....
                        SECONDAIRE-BOOLEEN 181  ....................................42....4.....
                            PRIMAIRE-BOOLEEN 182  ....................................44....4.....
                                RELATION 183  ....................................44....4.....
                            FONCTION-BOOLEENNE 184  ....................................44....4.....
                        OPERATEUR-DE-RELATION 185  .1..11..1.1111111111.....................44....4.....
                    DEBUT-DE-FONCTION-BOOLEENNE 186  ....................................1.....4.....
                DEBUT-DE-PARTIE-GAUCHE-BOOLEENNE 187  .1..11..1.1111111111.....................11....4.....
                            EXPRESSION-DE-DESIGNATION-BIS 188  ..44..................................4.....4.....4.....
                        EXPRESSION-DE-DESIGNATION-SIMPLE 189  ..44..................................4.....4.....4.....
                    DEBUT-D'EXPRESSION-DE-DESIGNATION 190  ....................................................2.....
                            INDICATEUR-D'AIGUILLAGE 191  .1..11..1.1111111111.....................11....4.....
                                INDICE-FINAL 192  4.444444.4.....................................44..4444.....4.4....4.
                            IDENTIFICATEUR-TER 193  4444.....1111111111111111111111.1111111111144.....
                                LETTRE 194  4444.....44444444444444444444444.444444444444.....
```

PL n° 5

GRAMMAIRE ALGOL DE PRECEDENCE TOTALE
RELATIONS DE PRECEDENCE TOTALE (SUITE)

```
                                                  000000000000000000000000000011111111111111111111111111111111111111
                                                  777777777888888888899999999990000000000111111111122222222233333333334
                                                  123456789012345678901234567890123456789012345678901234567890123456789
                                 :  C01 ................11...1211..1...11.......1..........2.....22211111.1..1.
                                 (  C02 ................4.4...1.............1..1.2........4..4..........4...4..
                                 )  C03 ...44..44..4.44..4.4.....................................................
                                 ,  C04 .1..............1.1...1............2..1.......2121.21.........1.......
                                 +  C05 ................4.....................................................4.....
                                 -  C06 ................4.....................................................4.....
                                 *  C07 ................4.....................................................4.....
                                 /  C08 ................4.....................................................4.....
                                 .  C09 .....................................................................
                                 =  C10 ................4.....................................................4.....
                                 0  C11 ...44..44..4444..4.4.....................4.............................
                                 1  C12 ...44..44..4444..4.4.....................4.............................
                                 2  C13 ...44..44..4444..4.4.....................4.............................
                                 3  C14 ...44..44..4444..4.4.....................4.............................
                                 4  C15 ...44..44..4444..4.4.....................4.............................
                                 5  C16 ...44..44..4444..4.4.....................4.............................
                                 6  C17 ...44..44..4444..4.4.....................4.............................
                                 7  C18 ...44..44..4444..4.4.....................4.............................
                                 8  C19 ...44..44..4444..4.4.....................4.............................
                                 9  C20 ...44..44..4444..4.4.....................4.............................
                                 A  C21 ............4........................4.....................................
                                 B  C22 ............4........................4.....................................
                                 C  C23 ............4........................4.....................................
                                 D  C24 ............4........................4.....................................
                                 E  C25 ............4........................4.....................................
                                 F  C26 ............4........................4.....................................
                                 G  C27 ............4........................4.....................................
                                 H  C28 ............4........................4.....................................
                                 I  C29 ............4........................4.....................................
                                 J  C30 ............4........................4.....................................
                                 K  C31 ............4........................4.....................................
                                 L  C32 ............4........................4.....................................
                                 M  C33 ............4........................4.....................................
                                 N  C34 ............4........................4.....................................
                                 O  C35 ............4........................4.....................................
                                 P  C36 ............4........................4.....................................
                                 Q  C37 ............4........................4.....................................
                                 R  C38 ............4........................4.....................................
                                 S  C39 ............4........................4.....................................
                                 T  C40 ............4........................4.....................................
                                 U  C41 ............4........................4.....................................
                                 V  C42 ............4........................4.....................................
                                 W  C43 ............4........................4.....................................
                                 X  C44 ............4........................4.....................................
                                 Y  C45 ............4........................4.....................................
                                 Z  C46 ............4........................4.....................................
                                 .,  C47 4.4..44..4.....44...44444444444444....444.44444.......4444444444.4..4.
                                 :=  C48 .1..............11.1.....................................1.21.....22..12121..
                              'SI'  C49 .....................................................................
                              'IC'  C50 .....................................................................2..
                              'OU'  C51 ................11...................................................1.....
                              'ET'  C52 ................11...................................................1.1...
                             'FIN'  C53 .....................................................................1.1...
                             'PAS'  C54 ..................1.....................................2.............1.....
                             'NON'  C55 ................11.....................................2.............1.....
                            'REEL'  C56 4....4.................4.............4..4.....................................1.1...
                           CHAINE   C57 .....................................................................
                             'POUR'  C58 ..................1.....................................................
                             'VRAI'  C59 ...4.....................................................................2.....
                             'FAUX'  C60 ...4.....................................................................
                            'DEBUT'  C61 4.4..4...4.....44...444444444444.4....444.44444.......4444444444.4..4.
                            'SINON'  C62 ................11.1.1111..1...11.......1.............2111111111.1..1.
                            'ALORS'  C63 ................44.4.4444..4...44.......4.............4.4444444.4..4.
                            'FAIRE'  C64 ................44...4444..4...44.......4.............4444444444.4..4.
                           'VALEUR'  C65 ......................1.............2..1.....................................
                           'ENTIER'  C66 4....4.................4.............4..4.....................................
                           'CHAINE'  C67 ......................1.............2..1.....................................
                           'ALLERA'  C68 ......................1.............2..1.....................................
                           'JUSQUA'  C69 ..................1.....................................2.....................
                          'BOOLEEN'  C70 4....4.................4.............4..4.....................2.....1.....
                           'TABLEAU'  C71 ......................4.............4..4.....44.....................
                           'TANTQUE'  C72 .....................................................................
                          'REMANENT'  C73 1.....................................................................2..
                           'IMPLIQUE'  C74 ................11.....................1....22.....................
                           'NON-EGAL'  C75 ................4.....................................................1.1...
                          'PROCEDURE'  C76 .....................................................................4.....
                          'ETIQUETTE'  C77 ..........................2..........................................
                          'INFERIEUR'  C78 ......................1.............2..1.....................................
                          'SUPERIEUR'  C79 ................4.....................................................4.....
                          'RACCOURCI'  C80 ................4.....................................................4.....
                          'EQUIVALENT'  C81 ......................1.............2..1.....2.....................
                        'CROCHET-DROIT'  C82 ...44..44..4.44..4.4.....................................................
                        'CROCHET-GAUCHE'  C83 ..................1.....................................21.....1.....
                     'INFERIEUR-OU-EGAL'  C84 ................4.....................................................4.....
                     'SUPERIEUR-OU-EGAL'  C85 ................4.....................................................4.....
                  IDENTIFICATEUR-BOOLEEN  C86 ...4.....2.....................................................4.....
              IDENTIFICATEUR-ARITHMETIQUE  C87 ...44..44..4244..4.4.....................4.....................
                  'OPERATEUR-RELATIONNEL'  C88 ..................1.....................................................
                IDENTIFICATEUR-DE-DESIGNATION  C89 ..................2.....................................1.....
              'OPERATEUR-DE-DIVISION-ENTIERE'  C90 ................4.....................................................
IDENTIFICATEUR-DE-PROCEDURE-INSTRUCTION  C91 .....................................................4.....
                          PROGRAMME  C92 .....................................................................
                       IDENTIFICATEUR  C93 ............2.....................................................
                    PROGRAMME-INCOMPLET  C94 ..................11...111112112..11.....................1111111111.1..1.
                  INSTRUCTION-FINALE  C95 .....................................................................
                         INSTRUCTION  C96 .....................................................................
                        DECLARATIONS  C97 1.1..1...1.....44...444444442244.2...141.3411.....444444444.4..4.
```

PL nº 6 GRAMMAIRE ALGOL DE PRECEDENCE TOTALE

RELATIONS DE PRECEDENCE TOTALE (SUITE)

INSTRUCTION-NON-FINALE	098	..............44...44444444..44.....4.....4444444444.4..4.
DECLARATION-AUTRE-QUE-DE-PROCEDURE	099	..4.....
DECLARATION-DE-SPECIFICATION-DE-TYPE	100	.....
DECLARATIONS-INCOMPLETES	101	1....11..1.....11...1111.21..21122....111....11.....1111111111.1..1.
DECLARATIONS-BIEN-INCOMPLETES	102	.....
SPECIFICATION	103	.....
ANNONCE-DE-PROCEDURE	104	.....2.....
DECLARATIONS-TRES-INCOMPLETES	105	.....2.....
PARTIE-VALEUR	106	.....
LISTE-D'IDENTIFICATEURS	107	.....
PARTIE-FORMELLE	108	.....
ANNONCE-DE-PROCEDURE-BIS	109	.....4.....
IDENTIFICATEUR-BIS	110	.....4.....4.....
TYPE	111	2....2.....4.....4..4.....
PARAMETRES-FORMELS	112	.....
DECLARATION-DE-TYPE-PERMANENT	113	.....
DECLARATION-DE-TABLEAU	114	.....
DECLARATION-D'AIGUILLAGE	115	.....
DECLARATEUR-DE-TYPE	116	.....1.....2..1.....
DECLARATEUR-DE-TABLEAU	117	.....1.....2..1.....21.....
LISTE-DE-TABLEAUX	118	.....
IDENTIFICATEUR-AVEC-BORNES	119	.....
PAIRES-DE-BORNES	120	.....
EXPRESSION-ARITHMETIQUE	121	.....2.....
FIN-DE-DECLARATION-D'AIGUILLAGE	122	.....
LISTE-D'EXPRESSIONS-DE-DESIGNATION	123	.....
EXPRESSION-DE-DESIGNATION	124	.....
INSTRUCTION-QUELCONQUE	125	.....
INSTRUCTION-INCONDITIONNELLE	126	.....
INSTRUCTION-CONDITIONNELLE	127	.....
INSTRUCTION-POUR	128	.....
INSTRUCTION-DE-BASE	129	.....
INSTRUCTION-D'AFFECTATION-ARITHMETIQUE	130	.....
INSTRUCTION-D'AFFECTATION-BOOLEENNE	131	.....
INSTRUCTION-ALLER-A	132	.....
INSTRUCTION-PROCEDURE	133	.....
PARTIE-GAUCHE-ARITHMETIQUE	134	...44..44..4.44..4.4.....
PARTIE-DROITE-ARITHMETIQUE	135	.....
PARTIE-GAUCHE-BOOLEENNE	136	...4.....
PARTIE-DROITE-BOOLEENNE	137	.....
EXPRESSION-BOOLEENNE	138	.....
DEBUT-D'INSTRUCTION-PROCEDURE	139	.....
PARAMETRES-EFFECTIFS	140	...44..44..4.44..4.4.....
FIN-DE-PRIMAIRE-ARITHMETIQUE	141	...44..44..4.44..4.4.....
FIN-DE-PRIMAIRE-BOOLEEN	142	...4.....
FIN-D'EXPRESSION-DE-DESIGNATION-SIMPLE	143	.....
INSTRUCTION-SI	144	.....
PROPOSITION-SI	145	.............11.1.1111..1...11.....1.....2.2111111.1..1.
PROPOSITION-POUR	146	.............11...1111..1...11.....1.....2111111111.1..1.
FIN-DE-PROPOSITION-POUR	147	.............44...4444..4...44.....4.....444444444.4..4.
EXPRESSION-ARITHMETIQUE-BIS	148	.....4.....
EXPRESSION-ARITHMETIQUE-SIMPLE	149	.....4.....
DEBUT-D'EXPRESSION-ARITHMETIQUE	150	.....
EXPRESSION-ARITHMETIQUE-SIMPLE-BIS	151	...41..11..4.11.....
EXPRESSION-ARITHMETIQUE-SIMPLE-TER	152	...44..44..4.44.....
TERME	153	...44..44..4.44.....
OPERATEUR-ADDITIF	154	.....1.....1.....
TERME-BIS	155	...44..44..4.44...1.....
FACTEUR	156	...44..44..4.44....4.....
OPERATEUR-MULTIPLICATIF	157	.....1.....1.....
FACTEUR-BIS	158	...44..44..4.44..2.4.....
PRIMAIRE-ARITHMETIQUE	159	...44..44..4.44..4.4.....
NOMBRE	160	...44..44..4.44..4.4.....
FONCTION-ARITHMETIQUE	161	...44..44..4.44..4.4.....
PARENTHESE-GAUCHE	162	.....1.1.....1..1.....1..1..
NOMBRE-DECIMAL	163	...44..44..4.44..4.4.....
OPERATEUR-D'EXPONENTIATION-DECIMALE	164	.....
ENTIER	165	...44..44..4.44..4.4.....
CHIFFRES	166	...44..44..4.44..4.4.....
CHIFFRE	167	...44..44..4444..4.4.....4.....
DEBUT-DE-PARTIE-GAUCHE-ARITHMETIQUE	168	.....1.....1.....1.....
INDICES	169	...44..44..4.44..4.4.....
DEBUT-DE-FONCTION-ARITHMETIQUE	170	.....
EXPRESSION-BOOLEENNE-BIS	171	.....
EXPRESSION-BOOLEENNE-SIMPLE	172	.....
DEBUT-D'EXPRESSION-BOOLEENNE	173	.....
EXPRESSION-BOOLEENNE-SIMPLE-BIS	174	.....2.....
IMPLICATION	175	.....4.....
IMPLICATION-BIS	176	...2.....
TERME-BOOLEEN	177	...4.....
TERME-BOOLEEN-BIS	178	...4.....
FACTEUR-BOOLEEN	179	...4.....
FACTEUR-BOOLEEN-BIS	180	...4.....
SECONDAIRE-BOOLEEN	181	...4.....
PRIMAIRE-BOOLEEN	182	...4.....
RELATION	183	...4.....
FONCTION-BOOLEENNE	184	...4.....
OPERATEUR-DE-RELATION	185	.....1.....1.....
DEBUT-DE-FONCTION-BOOLEENNE	186	.....
DEBUT-DE-PARTIE-GAUCHE-BOOLEENNE	187	.....1.....1.....1.....
EXPRESSION-DE-DESIGNATION-BIS	188	.....
EXPRESSION-DE-DESIGNATION-SIMPLE	189	.....
DEBUT-D'EXPRESSION-DE-DESIGNATION	190	.....
INDICATEUR-D'AIGUILLAGE	191	.....1.....1.....1.....
INDICE-FINAL	192	...44..44..4.44..4.4.....
IDENTIFICATEUR-TER	193	.....4.....4.....
LETTRE	194	.....4.....4.....

PL n° 7 GRAMMAIRE ALGOL DE PRECEDENCE TOTALE

RELATIONS DE PRECEDENCE TOTALE (SUITE)

		111111111111111111111111111111111111111111111111111111
		444444444555555555566666666667777777777888888888899999
		123456789012345678901234567890123456789012345678901234
:	001	...111.111111111.11111111111.1.................1....11
(	002	444.4..444444444.44444444444.44444.............4444.11
)	003	.............4..4...........................4.........
,	004	....1.2111111111.1111111111121.................1111111
+	005	............4.44.44444444444.4........................
−	006	............4.44.44444444444.4........................
*	007	...............4.44444444444.4........................
/	008	...............4.44444444444.4........................
.	009	........................211...........................
=	010	..........444444.44444444444.4........................
0	011	.............4..4......4..4.................4........4
1	012	.............4..4......4..4.................4........4
2	013	.............4..4......4..4.................4........4
3	014	.............4..4......4..4.................4........4
4	015	.............4..4......4..4.................4........4
5	016	.............4..4......4..4.................4........4
6	017	.............4..4......4..4.................4........4
7	018	.............4..4......4..4.................4........4
8	019	.............4..4......4..4.................4........4
9	020	.............4..4......4..4.................4........4
A	021	..........................4..........................4
B	022	..........................4..........................4
C	023	..........................4..........................4
D	024	..........................4..........................4
E	025	..........................4..........................4
F	026	..........................4..........................4
G	027	..........................4..........................4
H	028	..........................4..........................4
I	029	..........................4..........................4
J	030	..........................4..........................4
K	031	..........................4..........................4
L	032	..........................4..........................4
M	033	..........................4..........................4
N	034	..........................4..........................4
O	035	..........................4..........................4
P	036	..........................4..........................4
Q	037	..........................4..........................4
R	038	..........................4..........................4
S	039	..........................4..........................4
T	040	..........................4..........................4
U	041	..........................4..........................4
V	042	..........................4..........................4
W	043	..........................4..........................4
X	044	..........................4..........................4
Y	045	..........................4..........................4
Z	046	..........................4..........................4
.,	047	...444.....................4..................4.....44
:=	048	....1.2111111111.11111111111.11111............11111...
'SI'	049	....1........................1111.....................
'10'	050	........................444...........................
'OU'	051	..........111111.11111111111.1........211111.11.......
'ET'	052	..........111111.11111111111.1..........2111.11.......
'FIN'	053	....
'PAS'	054	....1..111111111.11111111111.1........................
'NON'	055	..........111111.11111111111.1...........211.11.......
'REEL'	056	44
CHAINE	057	....
'POUR'	058	...........................1..........................
'VRAI'	059	....
'FAUX'	060	....
'DEBUT'	061	...444.....................4..................4.....44
'SINON'	062	...111.211111111.11111111111.12111............12111.11
'ALORS'	063	.....4..4.444444.44444444444.4.4.4............4.4.4.44
'FAIRE'	064	...444.....................4..................4.....44
'VALEUR'	065	11
'ENTIER'	066	44
'CHAINE'	067	11
'ALLERA'	068	....1................1.........................1111...
'JUSQUA'	069	....1..111111111.11111111111.1........................
'BOOLEEN'	070	44
'TABLEAU'	071	44
'TANTQUE'	072	....1........................1111.....................
'REMANENT'	073	....
'IMPLIQUE'	074	..........111111.11111111111.1......21111111.11.......
'NON-EGAL'	075	..........444444.44444444444.4........................
'PROCEDURE'	076	11
'ETIQUETTE'	077	11
'INFERIEUR'	078	..........444444.44444444444.4........................
'SUPERIEUR'	079	..........444444.44444444444.4........................
'AIGUILLAGE'	080	11
'EQUIVALENT'	081	....
'CROCHET-DROIT'	082	.............4..4...........................4.........
'CROCHET-GAUCHE'	083	....1..111111111.1111111111141.....................4..
'INFERIEUR-OU-EGAL'	084	..........444444.44444444444.4........................
'SUPERIEUR-OU-EGAL'	085	..........444444.44444444444.4........................
IDENTIFICATEUR-BOOLEEN	086	....
IDENTIFICATEUR-ARITHMETIQUE	087	.............4..4...........................4.........
'OPERATEUR-D'EXPONENTIATION'	088	..................2111111111.1........................
IDENTIFICATEUR-DE-DESIGNATION	089	....
'OPERATEUR-DE-DIVISION-ENTIERE'	090	...............4.44444444444.4........................
IDENTIFICATEUR-DE-PROCEDURE-INSTRUCTION	091	....
PROGRAMME	092	....
IDENTIFICATEUR	093	....
PROGRAMME-INCOMPLET	094	...111.....................1..................1.....11
INSTRUCTION-FINALE	095	....
INSTRUCTION	096	....
DECLARATIONS	097	...444.....................4..................4.....44

PL nº 8 GRAMMAIRE ALGOL DE PRECEDENCE TOTALE

RELATIONS DE PRECEDENCE TOTALE (SUITE)

INSTRUCTION-NON-FINALE	098	...444.....4.....4.....44
DECLARATION-AUTRE-QUE-DE-PROCEDURE	099	
DECLARATION-OU-SPECIFICATION-DE-TYPE	100	
DECLARATIONS-INCOMPLETES	101	...111.....1.....1.....11
DECLARATIONS-BIEN-INCOMPLETES	102	
SPECIFICATION	103	
ANNONCE-DE-PROCEDURE	104	
DECLARATIONS-TRES-INCOMPLETES	105	
PARTIE-VALEUR	106	
LISTE-D'IDENTIFICATEURS	107	
PARTIE-FORMELLE	108	
ANNONCE-DE-PROCEDURE-BIS	109	
IDENTIFICATEUR-BIS	110	
TYPE	111	44
PARAMETRES-FORMELS	112	
DECLARATION-DE-TYPE-REMANENT	113	
DECLARATION-DE-TABLEAU	114	
DECLARATION-D'AIGUILLAGE	115	
DECLARATEUR-DE-TYPE	116	11
DECLARATEUR-DE-TABLEAU	117	11
LISTE-DE-TABLEAUX	118	
IDENTIFICATEUR-AVEC-BORNES	119	
PAIRES-DE-BORNES	120	
EXPRESSION-ARITHMETIQUE	121	
FIN-DE-DECLARATION-D'AIGUILLAGE	122	
LISTE-D'EXPRESSIONS-DE-DESIGNATION	123	
EXPRESSION-DE-DESIGNATION	124	
INSTRUCTION-QUELCONQUE	125	
INSTRUCTION-INCONDITIONNELLE	126	
INSTRUCTION-CONDITIONNELLE	127	
INSTRUCTION-POUR	128	
INSTRUCTION-DE-BASE	129	
INSTRUCTION-D'AFFECTATION-ARITHMETIQUE	130	
INSTRUCTION-D'AFFECTATION-BOOLEENNE	131	
INSTRUCTION-ALLER-A	132	
INSTRUCTION-PROCEDURE	133	
PARTIE-GAUCHE-ARITHMETIQUE	134	.....4..4.....4.....
PARTIE-DROITE-ARITHMETIQUE	135	
PARTIE-GAUCHE-BOOLEENNE	136	
PARTIE-DROITE-BOOLEENNE	137	
EXPRESSION-BOOLEENNE	138	
DEBUT-D'INSTRUCTION-PROCEDURE	139	
PARAMETRES-EFFECTIFS	140	.....4..4.....4.....
FIN-DE-PRIMAIRE-ARITHMETIQUE	141	.....4..4.....4.....
FIN-DE-PRIMAIRE-BOOLEEN	142	
FIN-D'EXPRESSION-DE-DESIGNATION-SIMPLE	143	
INSTRUCTION-SI	144	
PROPOSITION-SI	145	.....1..2.111111.11111111111.1.2.1.....1.2.1.11
PROPOSITION-POUR	146	...111.....1.....1.....11
FIN-DE-PROPOSITION-POUR	147	...444.....4.....4.....44
EXPRESSION-ARITHMETIQUE-BIS	148	
EXPRESSION-ARITHMETIQUE-SIMPLE	149	
DEBUT-D'EXPRESSION-ARITHMETIQUE	150	
EXPRESSION-ARITHMETIQUE-SIMPLE-BIS	151	.....2.....
EXPRESSION-ARITHMETIQUE-SIMPLE-TER	152	.....2.....4.....
TERME	153	.....4.....4.....
OPERATEUR-ADDITIF	154	.....2.11.11111111111.1.....
TERME-BIS	155	.....4..2.....4.....
FACTEUR	156	.....4..4.....4.....
OPERATEUR-MULTIPLICATIF	157	.....2.11111111111.1.....
FACTEUR-BIS	158	.....4..4.....4.....
PRIMAIRE-ARITHMETIQUE	159	.....4..4.....4.....
NOMBRE	160	.....4..4.....4.....
FONCTION-ARITHMETIQUE	161	.....4..4.....4.....
PARENTHESE-GAUCHE	162	222.1..111111111.11111111111.11111.....1111...
NOMBRE-DECIMAL	163	.....4..4.....2.....4.....
OPERATEUR-D'EXPONENTIATION-DECIMALE	164	.....211.....
ENTIER	165	.....4..4.....4.....4.....
CHIFFRES	166	.....4..4.....4..2.....4.....
CHIFFRE	167	.....4..4.....4..4.....4.....4
DEBUT-DE-PARTIE-GAUCHE-ARITHMETIQUE	168	....1..111111111.1111111111121.....1..
INDICES	169	.....4..4.....4.....
DEBUT-DE-FONCTION-ARITHMETIQUE	170	
EXPRESSION-BOOLEENNE-BIS	171	
EXPRESSION-BOOLEENNE-SIMPLE	172	
DEBUT-D'EXPRESSION-BOOLEENNE	173	
EXPRESSION-BOOLEENNE-SIMPLE-BIS	174	
IMPLICATION	175	
IMPLICATION-BIS	176	
TERME-BOOLEEN	177	
TERME-BOOLEEN-BIS	178	
FACTEUR-BOOLEEN	179	
FACTEUR-BOOLEEN-BIS	180	
SECONDAIRE-BOOLEEN	181	
PRIMAIRE-BOOLEEN	182	
RELATION	183	
FONCTION-BOOLEENNE	184	
OPERATEUR-DE-RELATION	185	.....211111.11111111111.1.....
DEBUT-DE-FONCTION-BOOLEENNE	186	
DEBUT-DE-PARTIE-GAUCHE-BOOLEENNE	187	....1..111111111.1111111111121.....1..
EXPRESSION-DE-DESIGNATION-BIS	188	
EXPRESSION-DE-DESIGNATION-SIMPLE	189	
DEBUT-D'EXPRESSION-DE-DESIGNATION	190	
INDICATEUR-D'AIGUILLAGE	191	....1..111111111.11111111111.1.....2..
INDICE-FINAL	192	.....4..4.....4.....
IDENTIFICATEUR-TER	193	.....2.....2
LETTRE	194	.....4.....4

5.2 Précédence de gauche à droite et recherche d'erreurs dans un programme Algol [*]

Nous avons été amené à résoudre le problème suivant : étant donné un programme Algol, trouver ses erreurs syntaxiques en un seul passage afin d'accélérer sa mise au point. Pour ceci nous avons utilisé l'analyseur à relations de précédence de gauche à droite décrit au paragraphe 4.5. Du fait que cet analyseur s'applique aussi à des grammaires comportant certaines ambiguïtés nous n'avons pas eu besoin d'introduire de nouveaux symboles terminaux dans la grammaire Algol. Par contre afin d'éviter l'écriture d'un programme spécial (éditeur) dont le rôle aurait été de reconnaître chaque symbole de base Algol et de le coder, nous avons introduit des règles de grammaire supplémentaires spécifiant la façon d'écrire chaque symbole de base Algol. Nous avons adopté les mêmes conventions que celles utilisées pour le compilateur Algol (réalisé par J.C. Boussard [Bou]) qui équipe notre calculateur (IBM 7044). De ce fait le vocabulaire terminal de notre grammaire se réduit à l'ensemble des caractères disponibles sur le matériel d'entrée-sortie.

Cependant afin d'obtenir des relations de précédence incompatibles deux à deux il a fallu dédoubler certains caractères. Ce fut le cas des lettres, suivant qu'elles sont utilisées pour écrire un symbole de base ou pour composer un identificateur, du caractère " et du caractère '. Ces distinctions ainsi que la suppression de certaines séquences inutiles telles que les commentaires, sont faites par quelques instructions intercalées dans le programme général d'analyse.

Que se passe-t-il lorsque l'analyseur "cale" sur une erreur ? Du fait que nous avons complété d'une façon empirique les relations de précédence de façon à ce que pour tout couple d'opérateurs il y ait une relation de vérifiée, ceci ne peut se produire que lorsque l'analyseur ne trouve pas de règle à appliquer à une sous-chaîne. Après avoir signalé l'erreur on supprime cette sous-chaîne ou on la remplace par un symbole spécial qui se comporte comme la liste de tous

---

[*] Pour plus de détails à ce sujet consulter l'article donné en référence [Co.1].

les opérandes de la grammaire et on poursuit l'analyse pour détecter d'autres erreurs.

Voici successivement la grammaire Algol utilisée, ses relations de précédence de gauche à droite, la relation $\Rightarrow^{*}$ entre deux opérandes et un exemple de recherche d'erreurs dans un programme Algol. Il faut noter que l'analyseur a été programmé en Algol et que de ce fait il fonctionnerait au moins 6 fois plus vite s'il avait été écrit en langage machine.

PL n° 9 GRAMMAIRE ALGOL POUR RECHERCHE D'ERREURS

REGLES

```
                PROGRAMME-DELIMITE ::= =//= PROGRAMME =//=
                          PROGRAMME ::= BLOC
                                    OU  INSTRUCTION-COMPOSEE
                                    OU  BLOC 'POINT-VIRGULE'
                                    OU  INSTRUCTION-COMPOSEE 'POINT-VIRGULE'
                              BLOC ::= IDENTIFICATEUR 'DEUX-POINTS' BLOC
                                    OU  'DEBUT' INTERIEUR-DE-BLOC 'FIN'
                 INTERIEUR-DE-BLOC ::= SUITE-DE-DECLARATIONS 'POINT-VIRGULE'
                                    OU  SUITE-DE-DECLARATIONS 'POINT-VIRGULE' INSTRUCTION
                                    OU  INTERIEUR-DE-BLOC 'POINT-VIRGULE'
                                    OU  INTERIEUR-DE-BLOC 'POINT-VIRGULE' INSTRUCTION
                                    OU  SUITE-DE-DECLARATIONS 'POINT-VIRGULE' 'COMMENTAIRE' 'POINT-VIRGULE'
                                    OU  SUITE-DE-DECLARATIONS 'POINT-VIRGULE' 'COMMENTAIRE' 'POINT-VIRGULE'
                                        INSTRUCTION
                                    OU  INTERIEUR-DE-BLOC 'POINT-VIRGULE' 'COMMENTAIRE' 'POINT-VIRGULE'
                                    OU  INTERIEUR-DE-BLOC 'POINT-VIRGULE' 'COMMENTAIRE' 'POINT-VIRGULE'
                                        INSTRUCTION
          INSTRUCTION-COMPOSEE ::= IDENTIFICATEUR 'DEUX-POINTS' INSTRUCTION-COMPOSEE
                                   OU 'DEBUT' 'FIN'
                                    OU  'DEBUT' SUITE-D'INSTRUCTIONS 'FIN'
          SUITE-D'INSTRUCTIONS ::= INSTRUCTION
                                    OU  'POINT-VIRGULE'
                                    OU  'POINT-VIRGULE' INSTRUCTION
                                    OU  SUITE-D'INSTRUCTIONS 'POINT-VIRGULE'
                                    OU  SUITE-D'INSTRUCTIONS 'POINT-VIRGULE' INSTRUCTION
                                    OU  'COMMENTAIRE' 'POINT-VIRGULE'
                                    OU  'POINT-VIRGULE' 'COMMENTAIRE' 'POINT-VIRGULE'
                                    OU  'COMMENTAIRE' 'POINT-VIRGULE' INSTRUCTION
                                    OU  'POINT-VIRGULE' 'COMMENTAIRE' 'POINT-VIRGULE' INSTRUCTION
                                    OU  SUITE-D'INSTRUCTIONS 'POINT-VIRGULE' 'COMMENTAIRE' 'POINT-VIRGULE'
                                    OU  SUITE-D'INSTRUCTIONS 'POINT-VIRGULE' 'COMMENTAIRE' 'POINT-VIRGULE'
                                        INSTRUCTION
                   INSTRUCTION ::= INSTRUCTION-POUR
                                    OU  INSTRUCTION-INCONDITIONNELLE
                                    OU  INSTRUCTION-CONDITIONNELLE
             INSTRUCTION-POUR ::= IDENTIFICATEUR 'DEUX-POINTS' INSTRUCTION-POUR
                                    OU  'POUR' VARIABLE := LISTE-DE-POUR 'FAIRE'
                                    OU  'POUR' VARIABLE := LISTE-DE-POUR 'FAIRE' INSTRUCTION
                  LISTE-DE-POUR ::= ELEMENT-D'UNE-LISTE-DE-POUR
                                    OU  ELEMENT-D'UNE-LISTE-DE-POUR , LISTE-DE-POUR
      ELEMENT-D'UNE-LISTE-DE-POUR ::= EXPRESSION-ARITHMETIQUE
                                    OU  EXPRESSION-ARITHMETIQUE 'TANTQUE' EXPRESSION-BOOLEENNE
                                    OU  EXPRESSION-ARITHMETIQUE 'PAS' EXPRESSION-ARITHMETIQUE 'JUSQUA'
                                        EXPRESSION-ARITHMETIQUE
INSTRUCTION-INCONDITIONNELLE ::= BLOC
                                    OU  INSTRUCTION-COMPOSEE
                                    OU  INSTRUCTION-DE-BASE
          INSTRUCTION-DE-BASE ::= IDENTIFICATEUR 'DEUX-POINTS'
                                    OU  IDENTIFICATEUR 'DEUX-POINTS' INSTRUCTION-DE-BASE
                                    OU  INSTRUCTION-D'AFFECTATION
                                    OU  INSTRUCTION-ALLER-A
                                    OU  INSTRUCTION-PROCEDURE
    INSTRUCTION-D'AFFECTATION ::= LISTE-DE-PARTIES-GAUCHES := EXPRESSION-ARITHMETIQUE
                                    OU  LISTE-DE-PARTIES-GAUCHES := EXPRESSION-BOOLEENNE
      LISTE-DE-PARTIES-GAUCHES ::= VARIABLE
                                    OU  LISTE-DE-PARTIES-GAUCHES := VARIABLE
          INSTRUCTION-ALLER-A ::= 'ALLER A' EXPRESSION-DE-DESIGNATION
       INSTRUCTION-PROCEDURE ::= IDENTIFICATEUR
                                    OU  IDENTIFICATEUR(PAR.EFFECTIFS)
IDENTIFICATEUR(PAR.EFFECTIFS) ::= IDENTIFICATEUR 'PARENTHESE-GAUCHE' LISTE-DE-PARAMETRES-EFFECTIFS
                                    'PARENTHESE-DROITE'
                                    OU  IDENTIFICATEUR(PAR.EFFECTIFS)LETTRES 'DEUX-POINTS'
                                        (LISTE-DE-PARAMETRES-EFFECTIFS)
IDENTIFICATEUR(PAR.EFFECTIFS)LETTRES ::= IDENTIFICATEUR 'PARENTHESE-GAUCHE' LISTE-DE-PARAMETRES-EFFECTIFS
                                        'PARENTHESE-DROITE' CHAINE-DE-LETTRES
     (LISTE-DE-PARAMETRES-EFFECTIFS) ::= 'PARENTHESE-GAUCHE' LISTE-DE-PARAMETRES-EFFECTIFS 'PARENTHESE-DROITE'
                                    OU  (LISTE-DE-PAR.EFFECTIFS)LETTRES 'DEUX-POINTS'
                                        (LISTE-DE-PARAMETRES-EFFECTIFS)
(LISTE-DE-PAR.EFFECTIFS)LETTRES ::= 'PARENTHESE-GAUCHE' LISTE-DE-PARAMETRES-EFFECTIFS 'PARENTHESE-DROITE'
                                        CHAINE-DE-LETTRES
              CHAINE-DE-LETTRES ::= LETTRE
                                    OU  CHAINE-DE-LETTRES LETTRE
   INSTRUCTION-CONDITIONNELLE ::= IDENTIFICATEUR 'DEUX-POINTS' INSTRUCTION-CONDITIONNELLE
                                    OU  PROPOSITION-SI
                                    OU  PROPOSITION-SI INSTRUCTION-INCONDITIONNELLE
                                    OU  PROPOSITION-SI INSTRUCTION-POUR
                                    OU  PROPOSITION-SI 'SINON'
                                    OU  PROPOSITION-SI 'SINON' INSTRUCTION
                                    OU  PROPOSITION-SI INSTRUCTION-INCONDITIONNELLE 'SINON'
                                    OU  PROPOSITION-SI INSTRUCTION-INCONDITIONNELLE 'SINON' INSTRUCTION
             PROPOSITION-SI ::= 'SI' EXPRESSION-BOOLEENNE 'ALORS'
      SUITE-DE-DECLARATIONS ::= DECLARATION-AUTRE-QUE-DE-PROCEDURE
                                    OU  SUITE-DE-DECLARATIONS 'POINT-VIRGULE' DECLARATION-AUTRE-QUE-DE-PROCEDURE
                                    OU  SUITE-DE-DECLARATIONS-INCOMPLETE 'POINT-VIRGULE'
                                    OU  SUITE-DE-DECLARATIONS-INCOMPLETE 'POINT-VIRGULE' CORPS-DE-PROCEDURE
                                    OU  'COMMENTAIRE' 'POINT-VIRGULE' DECLARATION-AUTRE-QUE-DE-PROCEDURE
                                    OU  SUITE-DE-DECLARATIONS 'POINT-VIRGULE' 'COMMENTAIRE' 'POINT-VIRGULE'
                                        DECLARATION-AUTRE-QUE-DE-PROCEDURE
                                    OU  SUITE-DE-DECLARATIONS-INCOMPLETE 'POINT-VIRGULE' 'COMMENTAIRE'
                                        'POINT-VIRGULE'
                                    OU  SUITE-DE-DECLARATIONS-INCOMPLETE 'POINT-VIRGULE' 'COMMENTAIRE'
                                        'POINT-VIRGULE' CORPS-DE-PROCEDURE
          CORPS-DE-PROCEDURE ::= INSTRUCTION
                                    OU  'CODE' 'FCODE'
DECLARATION-AUTRE-QUE-DE-PROCEDURE ::= DECLARATION-DE-TYPE
                                    OU  DECLARATION-DE-TABLEAU
                                    OU  DECLARATION-D'AIGUILLAGE
```

PL nº 10 GRAMMAIRE ALGOL POUR RECHERCHE D'ERREURS

REGLES (SUITE)

```
                DECLARATION-DE-TYPE  ::=  TYPE  LISTE-D'IDENTIFICATEURS
                                     OU  'REMANENT'  TYPE  LISTE-D'IDENTIFICATEURS
                             TYPE    ::=  'REEL'
                                     OU  'ENTIER'
                                     OU  'BOOLEEN'
           DECLARATION-DE-TABLEAU    ::=  'TABLEAU'  LISTE-DE-TABLEAU
                                     OU  'REMANENT'  'TABLEAU'  LISTE-DE-TABLEAU
                                     OU  TYPE  'TABLEAU'  LISTE-DE-TABLEAU
                                     OU  'REMANENT'  TYPE  'TABLEAU'  LISTE-DE-TABLEAU
                 LISTE-DE-TABLEAU    ::=  IDENTIFICATEUR-AVEC-BORNES
                                     OU  IDENTIFICATEUR-AVEC-BORNES  ,  LISTE-DE-TABLEAU
                                     OU  IDENTIFICATEUR  ,  LISTE-DE-TABLEAU
       IDENTIFICATEUR-AVEC-BORNES    ::=  IDENTIFICATEUR  'CROCHET-GAUCHE'  LISTE-DE-PAIRES-DE-BORNES
                                          'CROCHET-DROIT'
          LISTE-DE-PAIRES-DE-BORNES  ::=  PAIRE-DE-BORNES
                                     OU  PAIRE-DE-BORNES  ,  LISTE-DE-PAIRES-DE-BORNES
                    PAIRE-DE-BORNES  ::=  EXPRESSION-ARITHMETIQUE  'DEUX-POINTS-SEPARATEUR-DE-BORNES'
                                          EXPRESSION-ARITHMETIQUE
             DECLARATION-D'AIGUILLAGE  ::=  'AIGUILLAGE'  IDENTIFICATEUR  :=  LISTE-D'AIGUILLAGES
                LISTE-D'AIGUILLAGES    ::=  EXPRESSION-DE-DESIGNATION
                                     OU  EXPRESSION-DE-DESIGNATION  ,  LISTE-D'AIGUILLAGES
     SUITE-DE-DECLARATIONS-INCOMPLETE  ::=  SUITE-DE-DECLARATIONS-BIEN-INCOMPLETE
                                     OU  SUITE-DE-DECLARATIONS-INCOMPLETE  'POINT-VIRGULE'  SPECIFICATION
                                     OU  SUITE-DE-DECLARATIONS-INCOMPLETE  'POINT-VIRGULE'  'COMMENTAIRE'
                                          'POINT-VIRGULE'  SPECIFICATION
 SUITE-DE-DECLARATIONS-BIEN-INCOMPLETE  ::=  SUITE-DE-DECLARATIONS-TRES-INCOMPLETE
                                     OU  SUITE-DE-DECLARATIONS-TRES-INCOMPLETE  'POINT-VIRGULE'  PARTIE-VALEUR
                                     OU  SUITE-DE-DECLARATIONS-TRES-INCOMPLETE  'POINT-VIRGULE'  'COMMENTAIRE'
                                          'POINT-VIRGULE'  PARTIE-VALEUR
                      PARTIE-VALEUR  ::=  'VALEUR'  LISTE-D'IDENTIFICATEURS
 SUITE-DE-DECLARATIONS-TRES-INCOMPLETE  ::=  ENTETE-DE-PROCEDURE
                                     OU  SUITE-DE-DECLARATIONS  'POINT-VIRGULE'  ENTETE-DE-PROCEDURE
                                     OU  'COMMENTAIRE'  'POINT-VIRGULE'  ENTETE-DE-PROCEDURE
                                     OU  SUITE-DE-DECLARATIONS  'POINT-VIRGULE'  'COMMENTAIRE'  'POINT-VIRGULE'
                                          ENTETE-DE-PROCEDURE
                 ENTETE-DE-PROCEDURE  ::=  'PROCEDURE'  IDENTIFICATEUR
                                     OU  'PROCEDURE'  IDENTIFICATEUR(PAR.FORMELS)
                                     OU  TYPE  'PROCEDURE'  IDENTIFICATEUR
                                     OU  TYPE  'PROCEDURE'  IDENTIFICATEUR(PAR.FORMELS)
       IDENTIFICATEUR(PAR.FORMELS)  ::=  IDENTIFICATEUR  'PARENTHESE-GAUCHE'  LISTE-D'IDENTIFICATEURS
                                          'PARENTHESE-DROITE'
                                     OU  IDENTIFICATEUR(PAR.FORMELS)LETTRES  'DEUX-POINTS'
                                          (LISTE-DE-PARAMETRES-FORMELS)
  IDENTIFICATEUR(PAR.FORMELS)LETTRES  ::=  IDENTIFICATEUR  'PARENTHESE-GAUCHE'  LISTE-D'IDENTIFICATEURS
                                          'PARENTHESE-DROITE'  CHAINE-DE-LETTRES
       (LISTE-DE-PARAMETRES-FORMELS)  ::=  'PARENTHESE-GAUCHE'  LISTE-D'IDENTIFICATEURS  'PARENTHESE-DROITE'
                                     OU  (LISTE-DE-PARARAMETRES-FORMELS)LETTRES  'DEUX-POINTS'
                                          (LISTE-DE-PARAMETRES-FORMELS)
(LISTE-DE-PARARAMETRES-FORMELS)LETTRES  ::=  'PARENTHESE-GAUCHE'  LISTE-D'IDENTIFICATEURS  'PARENTHESE-DROITE'
                                          CHAINE-DE-LETTRES
                     SPECIFICATION   ::=  TYPE  LISTE-D'IDENTIFICATEURS
                                     OU  'CHAINE'  LISTE-D'IDENTIFICATEURS
                                     OU  'ETIQUETTE'  LISTE-D'IDENTIFICATEURS
                                     OU  'AIGUILLAGE'  LISTE-D'IDENTIFICATEURS
                                     OU  'TABLEAU'  LISTE-D'IDENTIFICATEURS
                                     OU  'PROCEDURE'  LISTE-D'IDENTIFICATEURS
                                     OU  TYPE  'TABLEAU'  LISTE-D'IDENTIFICATEURS
                                     OU  TYPE  'PROCEDURE'  LISTE-D'IDENTIFICATEURS
                 E-D'IDENTIFICATEURS ::=  IDENTIFICATEUR
                                     OU  IDENTIFICATEUR  ,  LISTE-D'IDENTIFICATEURS
                      IDENTIFICATEUR ::=  LETTRE
                                     OU  IDENTIFICATEUR  LETTRE
                                     OU  IDENTIFICATEUR  CHIFFRE
                          EXPRESSION ::=  EXPRESSION-ARITHMETIQUE
                                     OU  EXPRESSION-BOOLEENNE
                                     OU  EXPRESSION-DE-DESIGNATION
            EXPRESSION-ARITHMETIQUE  ::=  EXPRESSION-ARITHMETIQUE-SIMPLE
                                     OU  PROPOSITION-SI  EXPRESSION-ARITHMETIQUE-SIMPLE  'SINON'
                                          EXPRESSION-ARITHMETIQUE
               EXPRESSION-BOOLEENNE  ::=  EXPRESSION-BOOLEENNE-SIMPLE
                                     OU  PROPOSITION-SI  EXPRESSION-BOOLEENNE-SIMPLE  'SINON'  EXPRESSION-BOOLEENNE
          EXPRESSION-DE-DESIGNATION  ::=  EXPRESSION-DE-DESIGNATION-SIMPLE
                                     OU  PROPOSITION-SI  EXPRESSION-DE-DESIGNATION-SIMPLE  'SINON'
                                          EXPRESSION-DE-DESIGNATION
     EXPRESSION-ARITHMETIQUE-SIMPLE  ::=  TERME
                                     OU  OPERATEUR-ADDITIF  TERME
                                     OU  EXPRESSION-ARITHMETIQUE-SIMPLE  OPERATEUR-ADDITIF  TERME
                OPERATEUR-ADDITIF    ::=  +
                                     OU  -
                            TERME    ::=  FACTEUR
                                     OU  TERME  OPERATEUR-MULTIPLICATIF  FACTEUR
          OPERATEUR-MULTIPLICATIF    ::=  *
                                     OU  /
                                     OU  'OPERATEUR-DE-DIVISION-ENTIERE'
                          FACTEUR    ::=  PRIMAIRE-ARITHMETIQUE
                                     OU  FACTEUR  'PUISSANCE'  PRIMAIRE-ARITHMETIQUE
            PRIMAIRE-ARITHMETIQUE    ::=  VARIABLE
                                     OU  INDICATEUR-DE-FONCTION
                                     OU  NOMBRE-SANS-SIGNE
                                     OU  'PARENTHESE-GAUCHE'  EXPRESSION-ARITHMETIQUE  'PARENTHESE-DROITE'
                NOMBRE-SANS-SIGNE    ::=  NOMBRE-DECIMAL
                                     OU  OPERATEUR-DECIMAL-D'EXPONENTIATION  ENTIER-SANS-SIGNE
                                     OU  NOMBRE-DECIMAL  OPERATEUR-DECIMAL-D'EXPONENTIATION  ENTIER-SANS-SIGNE
OPERATEUR-DECIMAL-D'EXPONENTIATION   ::=  '10'
                                     OU  '10'  +
                                     OU  '10'  -
```

PL n° 11 GRAMMAIRE ALGOL POUR RECHERCHE D'ERREURS

REGLES (SUITE)

```
                NOMBRE-DECIMAL  ::=  ENTIER-SANS-SIGNE
                                OU   'POINT-DECIMAL'  ENTIER-SANS-SIGNE
                                OU   ENTIER-SANS-SIGNE  'POINT-DECIMAL'  ENTIER-SANS-SIGNE
                ENTIER-SANS-SIGNE  ::=  CHIFFRE
                                OU   ENTIER-SANS-SIGNE  CHIFFRE
      EXPRESSION-BOOLEENNE-SIMPLE  ::=  IMPLICATION
                                OU   EXPRESSION-BOOLEENNE-SIMPLE  'EQUIVALENT'  IMPLICATION
                      IMPLICATION  ::=  TERME-BOOLEEN
                                OU   IMPLICATION  'IMPLIQUE'  TERME-BOOLEEN
                     TERME-BOOLEEN  ::=  FACTEUR-BOOLEEN
                                OU   TERME-BOOLEEN  'OU'  FACTEUR-BOOLEEN
                   FACTEUR-BOOLEEN  ::=  SECONDAIRE-BOOLEEN
                                OU   FACTEUR-BOOLEEN  'ET'  SECONDAIRE-BOOLEEN
                SECONDAIRE-BOOLEEN  ::=  PRIMAIRE-BOOLEEN
                                OU   'NON'  PRIMAIRE-BOOLEEN
                  PRIMAIRE-BOOLEEN  ::=  VARIABLE
                                OU   INDICATEUR-DE-FONCTION
                                OU   VALEUR-LOGIQUE
                                OU   'PARENTHESE-GAUCHE'  EXPRESSION-BOOLEENNE  'PARENTHESE-DROITE'
                                OU   EXPRESSION-ARITHMETIQUE-SIMPLE  OPERATEUR-DE-RELATION
                                     EXPRESSION-ARITHMETIQUE-SIMPLE
                     VALEUR-LOGIQUE  ::=  'VRAI'
                                OU   'FAUX'
           OPERATEUR-DE-RELATION  ::=  'INFERIEUR'
                                OU   'INFERIEUR-OU-EGAL'
                                OU   'EGAL'
                                OU   'DIFFERENT'
                                OU   'SUPERIEUR-OU-EGAL'
                                OU   'SUPERIEUR'
                       VARIABLE  ::=  IDENTIFICATEUR
                                OU   IDENTIFICATEUR  'CROCHET-GAUCHE'  LISTE-D'EXPRESSIONS-ARITHMETIQUES
                                     'CROCHET-DROIT'
LISTE-D'EXPRESSIONS-ARITHMETIQUES  ::=  EXPRESSION-ARITHMETIQUE
                                OU   EXPRESSION-ARITHMETIQUE  ,  LISTE-D'EXPRESSIONS-ARITHMETIQUES
          INDICATEUR-DE-FONCTION  ::=  IDENTIFICATEUR
                                OU   IDENTIFICATEUR(PAR.EFFECTIFS)
   LISTE-DE-PARAMETRES-EFFECTIFS  ::=  PARAMETRE-EFFECTIF
                                OU   ,
                                OU   ,  LISTE-DE-PARAMETRES-EFFECTIFS
                                OU   PARAMETRE-EFFECTIF  ,
                                OU   PARAMETRE-EFFECTIF  ,  LISTE-DE-PARAMETRES-EFFECTIFS
             PARAMETRE-EFFECTIF  ::=  IDENTIFICATEUR
                                OU   EXPRESSION
                                OU   'CROCHET-DE-CHAINE-GAUCHE'  'CROCHET-DE-CHAINE-DROIT'
                                OU   DEUX-APOSTROPHES  DEUX-APOSTROPHES
EXPRESSION-DE-DESIGNATION-SIMPLE  ::=  'PARENTHESE-GAUCHE'  EXPRESSION-DE-DESIGNATION  'PARENTHESE-DROITE'
                                OU   IDENTIFICATEUR
                                OU   IDENTIFICATEUR  'CROCHET-GAUCHE'  EXPRESSION-ARITHMETIQUE  'CROCHET-DROIT'
                      LETTRE  ::=  A
                                OU   B
                                OU   C
                                OU   D
                                OU   E
                                OU   F
                                OU   G
                                OU   H
                                OU   I
                                OU   J
                                OU   K
                                OU   L
                                OU   M
                                OU   N
                                OU   O
                                OU   P
                                OU   Q
                                OU   R
                                OU   S
                                OU   T
                                OU   U
                                OU   V
                                OU   W
                                OU   X
                                OU   Y
                                OU   Z
                     CHIFFRE  ::=  0
                                OU   1
                                OU   2
                                OU   3
                                OU   4
                                OU   5
                                OU   6
                                OU   7
                                OU   8
                                OU   9
                  'ALLERA'  ::=  '  A  L  L  E  R  A  '
                                OU   '  G  O  T  O  '
                     'SI'  ::=  '  S  I  '
                                OU   '  I  F  '
                  'ALORS'  ::=  '  A  L  O  R  S  '
                                OU   '  T  H  E  N  '
                  'SINON'  ::=  '  S  I  N  O  N  '
                                OU   '  E  L  S  E  '
                   'POUR'  ::=  '  P  O  U  R  '
                                OU   '  F  O  R  '
                    'PAS'  ::=  '  P  A  S  '
                                OU   '  S  T  E  P  '
```

PL n° 12 GRAMMAIRE ALGOL POUR RECHERCHE D'ERREURS

REGLES (SUITE)

```
                    'JUSQUA'  ::=  ' J U S Q U A '
                              OU   ' U N T I L '
                              OU   ' A '
                   'TANTQUE'  ::=  ' T A N T Q U E '
                              OU   ' W H I L E '
                     'FAIRE'  ::=  ' F A I R E '
                              OU   ' D O '
                  'REMANENT'  ::=  ' P E R S I S T A N T '
                              OU   ' O W N '
                              OU   ' R E M A N E N T '
                  'BCCLEEN'   ::=  ' B O O C L E E N '
                              OU   ' B O O C L E A N '
                   'ENTIER'   ::=  ' E N T I E R '
                              OU   ' I N T E G E R '
                     'REEL'   ::=  ' R E E L '
                              OU   ' R E A L '
                  'TABLEAU'   ::=  ' T A B L E A U '
                              OU   ' A R R A Y '
                'AIGUILLAGE'  ::=  ' A I G U I L L A G E '
                              OU   ' S W I T C H '
                'PROCEDURE'   ::=  ' P R O C E D U R E '
                   'CHAINE'   ::=  ' C H A I N E '
                              OU   ' S T R I N G '
                 'ETIQUETTE'  ::=  ' E T I Q U E T T E '
                              OU   ' L A B E L '
                    'VALEUR'  ::=  ' V A L E U R '
                              OU   ' V A L U E '
               'COMMENTAIRE'  ::=  ' C O M M E N T A I R E '
                              OU   ' C O M M E N T '
                    'DEBUT'   ::=  ' D E B U T '
                              OU   ' B E G I N '
                     'FIN'    ::=  ' F I N '
                              OU   ' E N D '
                    'CODE'    ::=  ' C O D E '
                   'FCODE'    ::=  ' F C O D E '
          'PARENTHESE-GAUCHE' ::=  (
          'PARENTHESE-DROITE' ::=  )
             'CROCHET-GAUCHE' ::=  . (
                              OU   CAR-CROCHET-GAUCHE
              'CROCHET-DROIT' ::=  ) .
                              OU   CAR-CROCHET-DROIT
'CROCHET-DE-CHAINE-GAUCHE'    ::=  ' ( '
 'CROCHET-DE-CHAINE-DROIT'    ::=  ' ) '
           DEUX-APOSTROPHES   ::=  ' '
                              OU   CAR-GUILLEMET
                'INFERIEUR'   ::=  ' I N F E R '
                              OU   ' L E S S '
                              OU   ' I N F '
                              OU   CAR-INFERIEUR
       'INFERIEUR-OU-EGAL'    ::=  ' I N F E G '
                              OU   ' I = '
                              OU   ' N O T G R E A T E R '
                              OU   CAR-INFERIEUR =
                    'EGAL'    ::=  =
                'DIFFERENT'   ::=  ' N O N E G '
                              OU   ' N O T E Q U A L '
                              OU   ' N = '
       'SUPERIEUR-OU-EGAL'    ::=  ' S U P E G '
                              OU   ' S = '
                              OU   ' N O T L E S S '
                              OU   CAR-SUPERIEUR =
                'SUPERIEUR'   ::=  ' S U P E R '
                              OU   ' G R E A T E R '
                              OU   ' S U P '
                              OU   CAR-SUPERIEUR
'OPERATEUR-DE-DIVISION-ENTIERE' ::=  ' / '
                 'PUISSANCE'  ::=  * *
                              OU   CAR-POINT-D'EXCLAMATION
                        '10'  ::=  : *
                              OU   CAR-DELTA
                              OU   '1 0 '
                         '1   ::=  ' 1
                'EQUIVALENT'  ::=  ' I D E N T '
                              OU   ' E Q U I V '
                  'IMPLIQUE'  ::=  ' I M P L '
                        'OU'  ::=  ' O R '
                              OU   ' O U '
                        'ET'  ::=  ' A N D '
                              OU   ' E T '
                       'NON'  ::=  ' N O N '
                              OU   ' N O T '
                      'VRAI'  ::=  ' V R A I '
                              OU   ' T R U E '
                              OU   ' 1 '
                      'FAUX'  ::=  ' F A L X '
                              OU   ' F A L S E '
                              OU   ' 0 '
           'POINT-VIRGULE'    ::=  : :
                              OU   CAR-POINT-VIRGULE
              'DEUX-POINTS'   ::=  :
                              OU   CAR-DEUX-POINTS
'DEUX-POINTS-SEPARATEUR-DE-BORNES' ::=  :
                         :=   ::=  : =
                              OU   CAR-DEUX-POINTS =
        'POINT-DECIMAL'       ::=  .
```

PL nº 13 GRAMMAIRE ALGOL POUR RECHERCH D'ERREURS

RELATIONS DE PRECEDENCE DE GAUCHE A DROITES ENTRE LES OPERATEURS

```
                 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
                 000000000111111111122222222223333333333444444444455555555556666666666777777777788
                 123456789012345678901234567890123456789012345678901234567890123456789012345678901
=//=  001  2.........1111111111111111111111111111.........................1.................
0     002  .44444444444444444444444444444444444444.....................42444444444444.44
1     003  .44444444444444444444444444444444444444.....................42444444444444.44
2     004  .44444444444444444444444444444444444444.....................4.444444444444.44
3     005  .44444444444444444444444444444444444444.....................4.444444444444.44
4     006  .44444444444444444444444444444444444444.....................4.444444444444.44
5     007  .44444444444444444444444444444444444444.....................4.444444444444.44
6     008  .44444444444444444444444444444444444444.....................4.444444444444.44
7     009  .44444444444444444444444444444444444444.....................4.444444444444.44
8     010  .44444444444444444444444444444444444444.....................4.444444444444.44
9     011  .44444444444444444444444444444444444444.....................4.444444444444.44
A     012  .44444444444444444444444444444444444444.....................4.444444444444.44
B     013  .44444444444444444444444444444444444444.....................4.444444444444.44
C     014  .44444444444444444444444444444444444444.....................4.444444444444.44
D     015  .44444444444444444444444444444444444444.....................4.444444444444.44
E     016  .44444444444444444444444444444444444444.....................4.444444444444.44
F     017  .44444444444444444444444444444444444444.....................4.444444444444.44
G     018  .44444444444444444444444444444444444444.....................4.444444444444.44
H     019  .44444444444444444444444444444444444444.....................4.444444444444.44
I     020  .44444444444444444444444444444444444444.....................4.444444444444.44
J     021  .44444444444444444444444444444444444444.....................4.444444444444.44
K     022  .44444444444444444444444444444444444444.....................4.444444444444.44
L     023  .44444444444444444444444444444444444444.....................4.444444444444.44
M     024  .44444444444444444444444444444444444444.....................4.444444444444.44
N     025  .44444444444444444444444444444444444444.....................4.444444444444.44
O     026  .44444444444444444444444444444444444444.....................4.444444444444.44
P     027  .44444444444444444444444444444444444444.....................4.444444444444.44
Q     028  .44444444444444444444444444444444444444.....................4.444444444444.44
R     029  .44444444444444444444444444444444444444.....................4.444444444444.44
S     030  .44444444444444444444444444444444444444.....................4.444444444444.44
T     031  .44444444444444444444444444444444444444.....................4.444444444444.44
U     032  .44444444444444444444444444444444444444.....................4.444444444444.44
V     033  .44444444444444444444444444444444444444.....................4.444444444444.44
W     034  .44444444444444444444444444444444444444.....................4.444444444444.44
X     035  .44444444444444444444444444444444444444.....................4.444444444444.44
Y     036  .44444444444444444444444444444444444444.....................4.444444444444.44
Z     037  .44444444444444444444444444444444444444.....................4.444444444444.44
A     038  ..................................2....2.2..2.2...2222...2..2.................
B     039  ..................................2......2.2.....2.........................
C     040  ..................................2..2.....2.....................................
D     041  ..................................2.........2.....2......2.....................
E     042  ..............................22.22.2....222.222222.....2.....................
F     043  ..............................2.2.2...2.....2..........2.....................
G     044  ..............................2...2.....2..2..2.....2.....2.....................
H     045  ..............................2...2...2.....2.....2.....2.....................
I     046  ..............................2222....222..2222.2.....2.....2.....2.....2
J     047  ..................................................2.....2.....................
K     048  ..............................................................................
L     049  ..............................2...2......2..2...2.2......2.....2.....................
M     050  ..............................2...2.......2..2.....2.....2.....2.....2.....2
N     051  ..............................2222.......2....2.....2.....2.....2.....2.....2
O     052  ..............................22.......2222..2.22.2....2.....2.....2.....2.....2
P     053  ..............................2...2......2..2..2.....2.....2.....2.....2.....2
Q     054  ..............................2...2......2..2..2.....2.....2.....2.....2.....2
R     055  ..............................2...2...2.....2..22.2......2.....2.....2.....2.....2
S     056  ..............................2...2.......2.222.2....2.....2.....2.....2.....2
T     057  ..............................2.2.2.222..2..2.22.2.....2.....2.....2.....2.....2
U     058  ..............................2...2...2....2.2.222...2...2.....2.....2.....2.....2
V     059  ..............................2.................2.....2.....2.....2.....2.....2
W     060  ..............................22....2.....2.....2.....2.....2.....2.....2.....2
X     061  ..............................2.....2.....2.....2.....2.....2.....2.....2.....2
Y     062  ..............................2.....2.....2.....2.....2.....2.....2.....2.....2
Z     063  ..............................2.....2.....2.....2.....2.....2.....2.....2.....2
'     064  .22.....................2222222.22.2.222.222222....2......222.....2.....2.....2
'     065  4444444444444444444444444444444444444444.....4.4.444..444..4..4.....4.....4.....4
:     066  ..........4444444444444444444444444444.....4.42...2.4.....2.....2.....2.....2.....2
:     067  4444444444444444444444444444444444444444.....4.4..44..4.4.....4.....4.....4.....4
,     068  .1111111111111111111111111111111111111.....1.1.111..111..1..1.....1.....1.....1
+     069  .44444444444444444444444444444444444444.....4.4......4.4.....4.....4.....4.....4
-     070  .44444444444444444444444444444444444444.....4.4......4.4.....4.....4.....4.....4
*     071  .44444444444444444444444444444444444444.....4.4..442.4.4.....4.....4.....4.....4
/     072  .44444444444444444444444444444444444444.....424.....4.4.....4.....4.....4.....4
(     073  .44444444444444444444444444444444444444.....424.444..4.4..4..4.....4.....4.....4
```

PL nº 14 GRAMMAIRE ALGOL POUR RECHERCHE D'ERREURS

RELATIONS DE PRECEDENCE DE GAUCHE A DROITE ENTRE LES OPERATEURS
(SUITE)

)	074	..........4444444444444444444444444.....42444444.42.4.44
.	075	.444444444.....4.444444424..4.44
CAR-CROCHET-GAUCHE	076	.4444444444444444444444444444444444.....4.4..44..4.4.....
CAR-CROCHET-DROIT	077	.....4.4444444.4..4.44
CAR-GUILLEMET	078	.....4...4.....4...4..
CAR-INFERIEUR	079	.4444444444444444444444444444444444.....4.4..44..4.4....2
=	080	.4444444444444444444444444444444444.....424..44..4.4.....
CAR-SUPERIEUR	081	.4444444444444444444444444444444444.....4.4..44..4.4....2
CAR-POINT-D'EXCLAMATION	082	.4444444444444444444444444444444444.....4.4.....4.4.....
CAR-DELTA	083	.444444444.....44.....
CAR-POINT-VIRGULE	084	4.....4444444444444444444444.....4.4.....
CAR-DEUX-POINTS	085	.....4444444444444444444444.....4.4.....4.....2
PROPOSITION-SI	086	.1111111111111111111111111111111111.....1.1..11..1.1.....
TYPE	087	.....1111111111111111111111111.....1...1.....
OPERATEUR-DE-RELATION	088	.1111111111111111111111111111111111.....1.1.411..1.1.....
OPERATEUR-ADDITIF	089	.1111111111111111111111111111111111.....1.1.4....1.1.....
OPERATEUR-MULTIPLICATIF	090	.1111111111111111111111111111111111.....1.1.4....1.1.....
OPERATEUR-DECIMAL-D'EXPONENTIATION	091	.1111111111.....4.....
LETTRE	092	.1111111111111111111111111111111111.....1.1141111111111.11
CHIFFRE	093	.1111111111111111111111111111111111.....1.1141111111111.11
'ALLERA'	094	.....1111111111111111111111111.....1.....1.....
'SI'	095	.1111111111111111111111111111111111.....1.1..11..1.1.....
'ALORS'	096	.4444444444444444444444444444444444.....4.4..44..4.4.....
'SINON'	097	.1111111111111111111111111111111111.....1.1.411..1.1.....
'POUR'	098	.....1111111111111111111111111.....
'PAS'	099	.1111111111111111111111111111111111.....1.1..11..1.1.....
'JUSQUA'	100	.1111111111111111111111111111111111.....1.1.411..1.1.....
'TANTQUE'	101	.1111111111111111111111111111111111.....1.1.411..1.1.....
'FAIRE'	102	.....1111111111111111111111111.....1.1.....
'REMANENT'	103	.....1.....
'BOOLEEN'	104	.....4444444444444444444444.....4.....
'ENTIER'	105	.....4444444444444444444444.....4.....
'REEL'	106	.....4444444444444444444444.....4.....
'TABLEAU'	107	.....1111111111111111111111111.....1.....
'AIGUILLAGE'	108	.....1111111111111111111111111.....1.....
'PROCEDURE'	109	.....1111111111111111111111111.....1.....
'CHAINE'	110	.....1111111111111111111111111.....1.....
'ETIQUETTE'	111	.....1111111111111111111111111.....1.....
'VALEUR'	112	.....1111111111111111111111111.....1.....
'COMMENTAIRE'	113	.....1.....
'DEBUT'	114	.....1111111111111111111111111.....1.1.....
'FIN'	115	4.....1.1.....
'CODE'	116	.....1.....
'FCODE'	117	.....1.....
'PARENTHESE-GAUCHE'	118	.1111111111111111111111111111111111.....1.1.111..1.1..1..
'PARENTHESE-DROITE'	119	.....1111111111111111111111111.....1.1141111.1..1.11
'CROCHET-GAUCHE'	120	.1111111111111111111111111111111111.....1.1.111..1.1.....
'CROCHET-DROIT'	121	.....1.1141111.1..1.11
'CROCHET-DE-CHAINE-GAUCHE'	122	.....1.....
'CROCHET-DE-CHAINE-DROIT'	123	.....4.....1.....
DEUX-APOSTROPHES	124	.....1...4.....1...1..
'INFERIEUR'	125	.4444444444444444444444444444444444.....4.4..44..4.4.....
'INFERIEUR-OU-EGAL'	126	.4444444444444444444444444444444444.....4.4..44..4.4.....
'EGAL'	127	.4444444444444444444444444444444444.....4.4..44..4.4.....
'DIFFERENT'	128	.4444444444444444444444444444444444.....4.4..44..4.4.....
'SUPERIEUR-OU-EGAL'	129	.4444444444444444444444444444444444.....4.4..44..4.4.....
'SUPERIEUR'	130	.4444444444444444444444444444444444.....4.4..44..4.4.....
'OPERATEUR-DE-DIVISION-ENTIERE'	131	.4444444444444444444444444444444444.....4.4.....4.4.....
'PUISSANCE'	132	.1111111111111111111111111111111111.....1.1.4....1.1.....
'10'	133	.444444444.....22.....
'1	134	.2.....
'EQUIVALENT'	135	.1111111111111111111111111111111111.....1.1.411..1.1.....
'IMPLIQUE'	136	.1111111111111111111111111111111111.....1.1.411..1.1.....
'OU'	137	.1111111111111111111111111111111111.....1.1.411..1.1.....
'ET'	138	.1111111111111111111111111111111111.....1.1.411..1.1.....
'NON'	139	.1111111111111111111111111111111111.....1.1.411..1.1.....
'VRAI'	140	.....1.1.4.....1.....
'FAUX'	141	.....1.1.4.....1.....
'POINT-VIRGULE'	142	4.....1111111111111111111111111.....1.1.....
'DEUX-POINTS'	143	4.....1111111111111111111111111.....1.1.4....1.....
'DEUX-POINTS-SEPARATEUR-DE-BORNES'	144	.1111111111111111111111111111111111.....1.1.411..1.1.....
:=	145	.1111111111111111111111111111111111.....1.1.111..1.1.....
'POINT-DECIMAL'	146	.1111111111.....4.....

PL nº 15 GRAMMAIRE ALGOL POUR RECHERCHE D'ERREURS

RELATIONS DE PRECEDENCE DE GAUCHE A DROITE ENTRE LES OPERATEURS
(SUITE)

```
                  000000000000000000011111111111111111111111111111111111111111111111111
                  888888888999999999900000000001111111111222222222233333333334444444
                  123456789012345678901234567890123456789012345678901234567890123456

=//=   001  ...........11.....................1.....,.....................11...
   0   002  44444.....
   1   003  44444.....
   2   004  44444.....
   3   005  44444.....
   4   006  44444.....
   5   007  44444.....
   6   008  44444.....
   7   009  44444.....
   8   010  44444.....
   9   011  44444.....
   A   012  44.44.....
   B   013  44.44.....
   C   014  44.44.....
   D   015  44.44.....
   E   016  44.44.....
   F   017  44.44.....
   G   018  44.44.....
   H   019  44.44.....
   I   020  44.44.....
   J   021  44.44.....
   K   022  44.44.....
   L   023  44.44.....
   M   024  44.44.....
   N   025  44.44.....
   O   026  44.44.....
   P   027  44.44.....
   Q   028  44.44.....
   R   029  44.44.....
   S   030  44.44.....
   T   031  44.44.....
   U   032  44.44.....
   V   033  44.44.....
   W   034  44.44.....
   X   035  44.44.....
   Y   036  44.44.....
   Z   037  44.44.....
   A   038  .....
   B   039  .....
   C   040  .....
   D   041  .....
   E   042  .....
   F   043  .....
   G   044  .....
   H   045  .....
   I   046  .....
   J   047  .....
   K   048  .....
   L   049  .....
   M   050  .....
   N   051  .....
   O   052  .....
   P   053  .....
   Q   054  .....
   R   055  .....
   S   056  .....
   T   057  .....
   U   058  .....
   V   059  .....
   W   060  .....
   X   061  .....
   Y   062  .....
   Z   063  .....
   '   064  .....
   '   065  ..44.....
   :   066  ...4.....
   :   067  ..44.....
   ,   068  ..1..1.111111.1...1.14.............14141.1.......1111111111411.1
   +   069  ..4.....
   -   070  ..4.....
   *   071  ..4.....
   /   072  ..4.....
   (   073  ..4.....
```

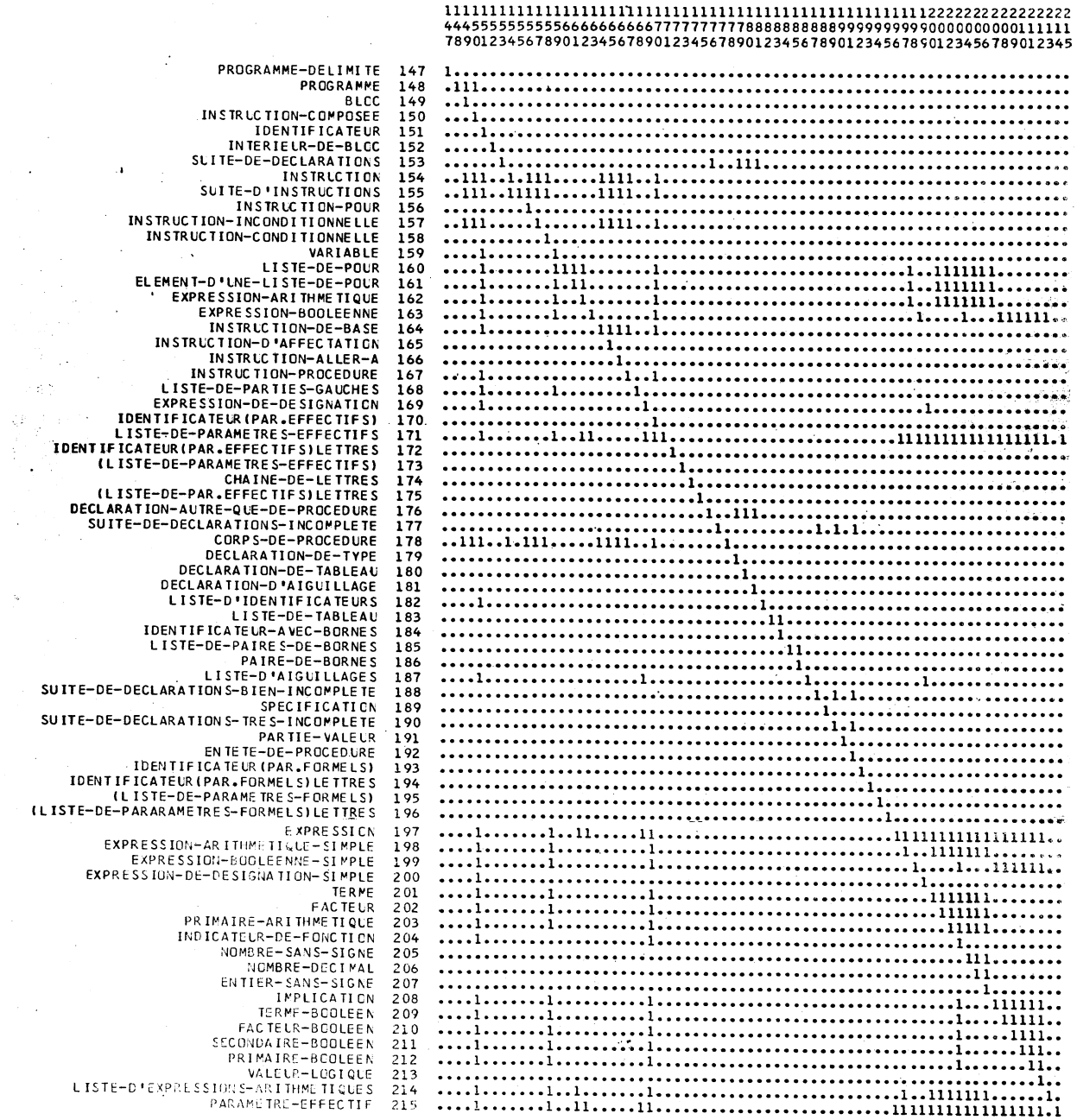
PL nº 16 GRAMMAIRE ALGOL POUR RECHERCHE D'ERREURS

RELATIONS DE PRECEDENCE DE GAUCHE A DROITE ENTRE LES OPERATEURS (SUITE)

)	074	44.4.
.	075	44.44.
CAR-CROCHET-GAUCHE	076	..4.
CAR-CROCHET-DROIT	077	44.44.
CAR-GUILLEMET	078	
CAR-INFERIEUR	079	..4.
=	080	..4.
CAR-SUPERIEUR	081	..4.
CAR-POINT-D'EXCLAMATION	082	..4.
CAR-DELTA	083	
CAR-POINT-VIRGULE	084	...4.
CAR-DEUX-POINTS	085	...4.
' PROPOSITION-SI	086	..11...1111111..21.....14..1.1.....111111111141.11
TYPE	087	.....11.....2.2.....4....
OPERATEUR-DE-RELATION	088	..1.....11111..44....4.....4..141.....1114444...41..1
OPERATEUR-ADDITIF	089	..1....441111..44.4444.....4..1414.....1114444...414.1
OPERATEUR-MULTIPLICATIF	090	..1....444111..44.4444.....4..1414.....1114444...414.1
OPERATEUR-DECIMAL-D'EXPONENTIATION	091	.....444..1..44.4444.....4...4.4.....4..4444...4.4..
LETTRE	092	11.11..444.44..44.4444.....4..4444...11111114..4444...4444.
CHIFFRE	093	11111..444444..44.4444.....4..4444...111111141144444...44444
'ALLERA'	094	.....1.....11.1.4.....4..1.1.....4....
'SI'	095	..1..1.111111.12.....1.1.....1111111111.1..1
'ALORS'	096	..44.
'SINON'	097	..11.1.1111111144144444.....14..1414.....1111111111414 11
'POUR'	098	.....11.....1.....2.
'PAS'	099	..1..1..11111.1....2.....1.1.....111.....1..1
'JUSQUA'	100	..1..1..11111.1.....4.....1.1.....111.....1..1
'TANTQUE'	101	..1..1.111111.1.....4.....1.1.....1111111111.1..1
'FAIRE'	102	...1.1.....1111..1.....14..1.1.....41.1.
'REMANENT'	103	.....2.....1112.
'BOOLEEN'	104	
'ENTIER'	105	
'REEL'	106	
'TABLEAU'	107	.....11.....1.....4....
'AIGUILLAGE'	108	.....11.....4..2.
'PROCEDURE'	109	.....11.....1.....41...
'CHAINE'	110	.....11.....4....
'ETIQUETTE'	111	.....11.....4....
'VALEUR'	112	.....11.....4....
'COMMENTAIRE'	113	...1.....2....
'DEBUT'	114	...1.11....1111..1....1111111...112..1.1.....11.1.
'FIN'	115	...1.....4.....4.....4....
'CODE'	116	.....2.....4....
'FCODE'	117	...1.....4....
'PARENTHESE-GAUCHE'	118	..1..1.111111.1.....121.1.1.....1111111111.1..1
'PARENTHESE-DROITE'	119	11.1...444.1...44.4444.....4...4.4...11111114..4444...444..
'CROCHET-GAUCHE'	120	..1..1..11111.1.....1.12.....111.....11.1
'CROCHET-DROIT'	121	11.11..444.....44.4444.....4...4.4...11111114..4444...4.44.
'CROCHET-DE-CHAINE-GAUCHE'	122	.....2.....
'CROCHET-DE-CHAINE-DROIT'	123	.....4.....
DEUX-APOSTROPHES	124	.....4....2.....
'INFERIEUR'	125	..4.
'INFERIEUR-OU-EGAL'	126	..4.
'EGAL'	127	..4.
'DIFFERENT'	128	..4.
'SUPERIEUR-OU-EGAL'	129	..4.
'SUPERIEUR'	130	..4.
'OPERATEUR-DE-DIVISION-ENTIERE'	131	..4.
'PUISSANCE'	132	..1....444111..44.4444.....4..1414.....4114444...414.1
'10'	133	
'1	134	
'EQUIVALENT'	135	..1....111111..44....4.....4..141.....1114111111141..1
'IMPLIQUE'	136	..1....111111..44....4.....4..141.....1114411111141..1
'OU'	137	..1....111111..44....4.....4..141.....1114441111141..1
'ET'	138	..1....111111..44....4.....4..141.....1114444111141..1
'NON'	139	..1....111111..44....4.....4..141.....1114444.1141..1
'VRAI'	140	...1.....44....4.....4...4.....4444...4....
'FAUX'	141	...1.....44....4.....4...4.....4444...4....
'POINT-VIRGULE'	142	...1.11....1111..1....1111111111214 1.1.1.....41.1.
'DEUX-POINTS'	143	...1.1.444.111144144444.....14..1414.....4..4444...4141.
'DEUX-POINTS-SEPARATEUR-DE-BORNES'	144	..1..1..11111.1.....1.14.....111.....1..1
:=	145	..1..1.111111.1.4.1.12.....4..1.1.....1111111111141.41
'POINT-DECIMAL'	146	.....4444.1..44.4444.....4...4.4.....4..4444...4.4..

PL n° 17 GRAMMAIRE ALGOL POUR RECHERCHE D'ERREURS

RELATION $\Rightarrow^*$ ENTRE LES OPERANDES



PL n° 18 EXEMPLE DE RECHERCHE D'ERREURS DANS UN PROGRAMME ALGOL

```
 2      'DEBUT' 'REEL' I ::
 3        'DEBUT' 'TABLEAU' M.(1:I). ::
 4        'ENTIER' I,J,K :: 'REEL' X,Y,Z :: 'BOOLEEN' P,Q,R ::
 5        'AIGUILLAGE' S=L1 ::
 6        'REEL' 'PROCEDURE' A(B,C) :: 'ENTIER' B :: 'ETIQUETTE' C ::
                                                  -----ERREUR. LA SEQUENCE SUIVANTE DELIMITEE PAR LES CARACTERES 1 DE
                                                       LA LIGNE 5 INCLUS ET 17 DE LA LIGNE 5 EXCLUS N A PAS DE SENS:
                                                          'AIGUILLAGE'
                                                          PRIMAIRE-BOOLEEN
                                                       ON ATTRIBUE A CETTE SEQUENCE UN SENS COMPATIBLE AVEC LE CONTEXTE.
 7        'VALEUR' B ::
 8          'DEBUT' 'ENTIER' R ::
                                                  -----ERREUR. LA SEQUENCE SUIVANTE DELIMITEE PAR LES CARACTERES 8 DE
                                                       LA LIGNE 3 INCLUS ET 10 DE LA LIGNE 7 EXCLUS N A PAS DE SENS:
                                                          SUITE-DE-DECLARATIONS-INCOMPLETE
                                                          'POINT-VIRGULE'
                                                          PARTIE-VALEUR
                                                       ON ATTRIBUE A CETTE SEQUENCE UN SENS COMPATIBLE AVEC LE CONTEXTE.
 9            X:=(B**-2)**(-2) ::
                                                  -----ERREUR. LA SEQUENCE SUIVANTE DELIMITEE PAR LES CARACTERES 5 DE
                                                       LA LIGNE 9 INCLUS ET 8 DE LA LIGNE 9 EXCLUS N A PAS DE SENS:
                                                          CHAINE-DE-LETTRES OU IDENTIFICATEUR
                                                          'PUISSANCE'
                                                       ON ATTRIBUE A CETTE SEQUENCE UN SENS COMPATIBLE AVEC LE CONTEXTE.
10            'ALLERA' 'SI' P 'ALORS' S.(B). 'SINON' L2 ::
11            L1: Y:=P ::
12            'SI' M(5)=X'/'Y 'ALORS'
13            'POUR' B=1 'PAS' 0 'JUSQUA' A(B,C) 'FAIRE'
14            L2: 'COMMENTAIRE' MAINTENANT ON VA S'AMUSER ::
                                                  -----ERREUR. LA SEQUENCE SUIVANTE DELIMITEE PAR LES CARACTERES 7 DE
                                                       LA LIGNE 13 INCLUS ET 30 DE LA LIGNE 13 EXCLUS N A PAS DE SENS:
                                                          PRIMAIRE-BOOLEEN
                                                          'PAS'
                                                          ENTIER-SANS-SIGNE
                                                          'JUSQUA'
                                                          IDENTIFICATEUR(PAR.FORMELS) OU IDENTIFICATEUR(PAR.EFFECTIFS)
                                                       ON ATTRIBUE A CETTE SEQUENCE UN SENS COMPATIBLE AVEC LE CONTEXTE.
15              'DEBUT' 'REMANENT' 'TABLEAU'
16              N.('SI' R 'IDENT' 'VRAI' 'ALORS' 2:10 'SINON' 1:10). ::
                                                  -----ERREUR. LA SEQUENCE SUIVANTE DELIMITEE PAR LES CARACTERES 4 DE
                                                       LA LIGNE 16 INCLUS ET 41 DE LA LIGNE 16 EXCLUS N A PAS DE SENS:
                                                          PROPOSITION-SI
                                                          PAIRE-DE-BORNES
                                                          'SINON'
                                                          ENTIER-SANS-SIGNE
                                                       ON ATTRIBUE A CETTE SEQUENCE UN SENS COMPATIBLE AVEC LE CONTEXTE.
17              'SI' X 'NONEG' 0 'ALORS' 'SI' Y=1 'OU' 2 'ALORS' 'ALLERA' C
                                                  -----ERREUR. LA SEQUENCE SUIVANTE DELIMITEE PAR LES CARACTERES 25 DE
                                                       LA LIGNE 17 INCLUS ET 33 DE LA LIGNE 17 EXCLUS N A PAS DE SENS:
                                                          PRIMAIRE-BOOLEEN
                                                          'OU'
                                                          ENTIER-SANS-SIGNE
                                                       ON ATTRIBUE A CETTE SEQUENCE UN SENS COMPATIBLE AVEC LE CONTEXTE.
18              'FIN' ::
                                                  -----ERREUR. LA SEQUENCE SUIVANTE DELIMITEE PAR LES CARACTERES 1 DE
                                                       LA LIGNE 17 INCLUS ET 1 DE LA LIGNE 18 EXCLUS N A PAS DE SENS:
                                                          PROPOSITION-SI
                                                          PROPOSITION-SI
                                                          INSTRUCTION-ALLER-A
                                                       ON ATTRIBUE A CETTE SEQUENCE UN SENS COMPATIBLE AVEC LE CONTEXTE.
19            'RETOUR' 'FIN' DE LA PROCEDURE A ::
                                                  -----ERREUR. LA SEQUENCE SUIVANTE DELIMITEE PAR LES CARACTERES 1 DE
                                                       LA LIGNE 13 INCLUS ET 6 DE LA LIGNE 18 EXCLUS N A PAS DE SENS:
                                                          'POUR'
                                                          SEQUENCE CITEE DANS UNE ERREUR PRECEDENTE
                                                          'FAIRE'
                                                          INSTRUCTION-DE-BASE
                                                          'COMMENTAIRE'
                                                          'POINT-VIRGULE'
                                                          BLOC
                                                       ON ATTRIBUE A CETTE SEQUENCE UN SENS COMPATIBLE AVEC LE CONTEXTE.
                                                  -----ERREUR. LA SEQUENCE SUIVANTE DELIMITEE PAR LES CARACTERES 1 DE
                                                       LA LIGNE 19 INCLUS ET 9 DE LA LIGNE 19 EXCLUS N A PAS DE SENS:
                                                          '
                                                          R
                                                          E
                                                          T
                                                          O
                                                          U
                                                          R
                                                          '
                                                       CETTE SEQUENCE EST SUPPRIMEE DU PROGRAMME.
20        X:=I:=0 ::
21        'SI' X=0 'ALORS'
22        'POUR' C:='VRAI','FAUX' 'FAIRE'
23        L3: Z:=A(B)LE 2EME PARAMETRE EST:(L3) 'SINON' 'FIN'
                                                  -----ERREUR. LA SEQUENCE SUIVANTE DELIMITEE PAR LES CARACTERES 7 DE
                                                       LA LIGNE 23 INCLUS ET 29 DE LA LIGNE 23 EXCLUS N A PAS DE SENS:
                                                          CHAINE-DE-LETTRES OU IDENTIFICATEUR
                                                          'PARENTHESE-GAUCHE'
                                                          CHAINE-DE-LETTRES OU IDENTIFICATEUR
                                                          'PARENTHESE-DROITE'
                                                          IDENTIFICATEUR
                                                       ON ATTRIBUE A CETTE SEQUENCE UN SENS COMPATIBLE AVEC LE CONTEXTE.
24      'FIN' DU PROGRAMME ::
                                                  -----ERREUR. LA SEQUENCE SUIVANTE DELIMITEE PAR LES CARACTERES 1 DE
                                                       LA LIGNE 22 INCLUS ET 41 DE LA LIGNE 23 EXCLUS N A PAS DE SENS:
                                                          'POUR'
                                                          CHAINE-DE-LETTRES OU IDENTIFICATEUR
                                                          :=
                                                          LISTE-DE-PARAMETRES-EFFECTIFS
                                                          'FAIRE'
                                                          INSTRUCTION-DE-BASE
                                                          'SINON'
                                                       ON ATTRIBUE A CETTE SEQUENCE UN SENS COMPATIBLE AVEC LE CONTEXTE.
L ANALYSE DE CE PROGRAMME A DURE 0 MINUTES ET 7 SECONDES.
```

5.3 Précédence de gauche à droite et Algol de N. Wirth et C.A.R. Hoare

N. Wirth et C.A.R. Hoare [WH] ont proposé un nouvel Algol plus complet en vue de remplacer Algol 60. Cet Algol est décrit à l'aide d'une grammaire "context-free" et d'un certain nombre de petits tableaux spécifiant le type d'un résultat obtenu par une opération portant sur deux opérandes. Par exemple la somme d'un "entier" et d'un "réel-long" et un "réel-long". Nous avons supprimé ces tableaux en rajoutant à la grammaire un certain nombre de règles permettant d'exprimer toutes les contraintes exprimées dans ceux-ci. Nous avons transformé cette nouvelle grammaire en une grammaire équivalente analysable par l'algorithme décrit au paragraphe 4.5.

Voici successivement, la grammaire obtenue, ses relations de précédence de gauche à droite, et la relation $\Rightarrow^*$ entre ses opérandes.

PL nº 19 GRAMMAIRE D'UN NOUVEL ALGOL

REGLES

```
                                  PROGRAM  ::=  BLOCK
                                    BLOCK  ::=  'BEGIN'  'END'
                                           OR   'BEGIN'  BLOCK-INTERIOR  'END'
                           BLOCK-INTERIOR  ::=  STATEMENT
                                           OR   .,
                                           OR   'ALSO'
                                           OR   .,  STATEMENT
                                           OR   'ALSO'  STATEMENT
                                           OR   BLOCK-INTERIOR-OR-DECLARATION-PART  .,
                                           OR   BLOCK-INTERIOR  'ALSO'
                                           OR   BLOCK-INTERIOR-OR-DECLARATION-PART  .,  STATEMENT
                                           OR   BLOCK-INTERIOR  'ALSO'  STATEMENT
                                           OR   INCOMPLETE-BLOCK-INTERIOR  :
                                           OR   INCOMPLETE-BLOCK-INTERIOR  :  STATEMENT
                 INCOMPLETE-BLOCK-INTERIOR  ::=  IDENTIFIER
                                           OR   .,  IDENTIFIER
                                           OR   'ALSO'  IDENTIFIER
                                           OR   BLOCK-INTERIOR-OR-DECLARATION-PART  .,  IDENTIFIER
                                           OR   BLOCK-INTERIOR  'ALSO'  IDENTIFIER
                                           OR   INCOMPLETE-BLOCK-INTERIOR  :  IDENTIFIER
BLOCK-INTERIOR-OR-DECLARATION-PART  ::=  BLOCK-INTERIOR
                                           OR   DECLARATION-PART
                       DECLARATION-PART  ::=  NON-PROCEDURE-DECLARATION
                                           OR   DECLARATION-PART  .,  NON-PROCEDURE-DECLARATION
                                           OR   PROPER-INCOMPLETE-DECLARATION-PART  .,
                                           OR   PROPER-INCOMPLETE-DECLARATION-PART  .,  STATEMENT
                                           OR   INTEGER-INCOMPLETE-DECLARATION-PART  .,  INTEGER-PROCEDURE-BODY
                                           OR   SHORT-REAL-INCOMPLETE-DECLARATION-PART  .,
                                                SHORT-REAL-PROCEDURE-BODY
                                           OR   LONG-REAL-INCOMPLETE-DECLARATION-PART  .,  LONG-REAL-PROCEDURE-BODY
                                           OR   SHORT-COMPLEX-INCOMPLETE-DECLARATION-PART  .,
                                                SHORT-COMPLEX-PROCEDURE-BODY
                                           OR   LONG-COMPLEX-INCOMPLETE-DECLARATION-PART  .,
                                                LONG-COMPLEX-PROCEDURE-BODY
                                           OR   LOGICAL-INCOMPLETE-DECLARATION-PART  .,  LOGICAL-PROCEDURE-BODY
                                           OR   BIT-INCOMPLETE-DECLARATION-PART  .,  BIT-PROCEDURE-BODY
                                           OR   STRING-INCOMPLETE-DECLARATION-PART  .,  STRING-PROCEDURE-BODY
                                           OR   REFERENCE-INCOMPLETE-DECLARATION-PART  .,  REFERENCE-PROCEDURE-BODY
                  NON-PROCEDURE-DECLARATION  ::=  SIMPLE-VARIABLE-DECLARATION
                                           OR   ARRAY-DECLARATION
                                           OR   RECORD-CLASS-DECLARATION
                SIMPLE-VARIABLE-DECLARATION  ::=  SIMPLE-TYPE-FOLLOWED-BY-IDENTIFIER
                                           OR   SIMPLE-VARIABLE-DECLARATION  ,  IDENTIFIER
          SIMPLE-TYPE-FOLLOWED-BY-IDENTIFIER  ::=  'INTEGER'  IDENTIFIER
                                           OR   'REAL'  IDENTIFIER
                                           OR   'LONG'  'REAL'  IDENTIFIER
                                           OR   'COMPLEX'  IDENTIFIER
                                           OR   'LONG'  'COMPLEX'  IDENTIFIER
                                           OR   'LOGICAL'  IDENTIFIER
                                           OR   'BITS'  IDENTIFIER
                                           OR   'BITS'  (  INTEGER-NUMBER  )  IDENTIFIER
                                           OR   'STRING'  IDENTIFIER
                                           OR   'REFERENCE'  IDENTIFIER
                                           OR   'REFERENCE'  (  RECORD-CLASS-IDENTIFIER  )  IDENTIFIER
                       ARRAY-DECLARATION  ::=  INCOMPLETE-ARRAY-DECLARATION  .(  BOUND-PAIR  ).  IDENTIFIER
                                           OR   ARRAY-DECLARATION  ,  IDENTIFIER
              INCOMPLETE-ARRAY-DECLARATION  ::=  SIMPLE-TYPE  'ARRAY'
                                           OR   INCOMPLETE-ARRAY-DECLARATION  .(  BOUND-PAIR  ),
                              BOUND-PAIR  ::=  INTEGER-EXPRESSION  :  INTEGER-EXPRESSION
                  RECORD-CLASS-DECLARATION  ::=  'RECORD'  IDENTIFIER  (  FIELD-LIST  )
                              FIELD-LIST  ::=  SIMPLE-VARIABLE-DECLARATION
                                           OR   FIELD-LIST  .,  SIMPLE-VARIABLE-DECLARATION
        PROPER-INCOMPLETE-DECLARATION-PART  ::=  PROPER-PROCEDURE-HEADING
                                           OR   DECLARATION-PART  .,  PROPER-PROCEDURE-HEADING
      INTEGER-INCOMPLETE-DECLARATION-PART  ::=  INTEGER-PROCEDURE-HEADING
                                           OR   DECLARATION-PART  .,  INTEGER-PROCEDURE-HEADING
   SHORT-REAL-INCOMPLETE-DECLARATION-PART  ::=  SHORT-REAL-PROCEDURE-HEADING
                                           OR   DECLARATION-PART  .,  SHORT-REAL-PROCEDURE-HEADING
    LONG-REAL-INCOMPLETE-DECLARATION-PART  ::=  LONG-REAL-PROCEDURE-HEADING
                                           OR   DECLARATION-PART  .,  LONG-REAL-PROCEDURE-HEADING
SHORT-COMPLEX-INCOMPLETE-DECLARATION-PART  ::=  SHORT-COMPLEX-PROCEDURE-HEADING
                                           OR   DECLARATION-PART  .,  SHORT-COMPLEX-PROCEDURE-HEADING
 LONG-COMPLEX-INCOMPLETE-DECLARATION-PART  ::=  LONG-COMPLEX-PROCEDURE-HEADING
                                           OR   DECLARATION-PART  .,  LONG-COMPLEX-PROCEDURE-HEADING
     LOGICAL-INCOMPLETE-DECLARATION-PART  ::=  LOGICAL-PROCEDURE-HEADING
                                           OR   DECLARATION-PART  .,  LOGICAL-PROCEDURE-HEADING
       BIT-INCOMPLETE-DECLARATION-PART  ::=  BIT-PROCEDURE-HEADING
                                           OR   DECLARATION-PART  .,  BIT-PROCEDURE-HEADING
    STRING-INCOMPLETE-DECLARATION-PART  ::=  STRING-PROCEDURE-HEADING
                                           OR   DECLARATION-PART  .,  STRING-PROCEDURE-HEADING
 REFERENCE-INCOMPLETE-DECLARATION-PART  ::=  REFERENCE-PROCEDURE-HEADING
                                           OR   DECLARATION-PART  .,  REFERENCE-PROCEDURE-HEADING
              PROPER-PROCEDURE-HEADING  ::=  'PROCEDURE'  PROCEDURE-HEADING
                                           OR   'NONREENTRANT'  'PROCEDURE'  PROCEDURE-HEADING
           INTEGER-PROCEDURE-HEADING  ::=  SIMPLE-INTEGER-TYPE  'PROCEDURE'  PROCEDURE-HEADING
        SHORT-REAL-PROCEDURE-HEADING  ::=  SIMPLE-SHORT-REAL-TYPE  'PROCEDURE'  PROCEDURE-HEADING
         LONG-REAL-PROCEDURE-HEADING  ::=  SIMPLE-LONG-REAL-TYPE  'PROCEDURE'  PROCEDURE-HEADING
      SHORT-COMPLEX-PROCEDURE-HEADING  ::=  SIMPLE-SHORT-COMPLEX-TYPE  'PROCEDURE'  PROCEDURE-HEADING
       LONG-COMPLEX-PROCEDURE-HEADING  ::=  SIMPLE-LONG-COMPLEX-TYPE  'PROCEDURE'  PROCEDURE-HEADING
          LOGICAL-PROCEDURE-HEADING  ::=  SIMPLE-LOGICAL-TYPE  'PROCEDURE'  PROCEDURE-HEADING
            BIT-PROCEDURE-HEADING  ::=  SIMPLE-BIT-TYPE  'PROCEDURE'  PROCEDURE-HEADING
         STRING-PROCEDURE-HEADING  ::=  SIMPLE-STRING-TYPE  'PROCEDURE'  PROCEDURE-HEADING
      REFERENCE-PROCEDURE-HEADING  ::=  SIMPLE-REFERENCE-TYPE  'PROCEDURE'  PROCEDURE-HEADING
                  PROCEDURE-HEADING  ::=  IDENTIFIER
                                           OR   IDENTIFIER  (  FORMAL-PARAMETER-LIST  )
               INTEGER-PROCEDURE-BODY  ::=  INTEGER-EXPRESSION
                                           OR   'BEGIN'  INTEGER-PROCEDURE-BODY-INTERIOR  'END'
            SHORT-REAL-PROCEDURE-BODY  ::=  SHORT-REAL-EXPRESSION
                                           OR   'BEGIN'  SHORT-REAL-PROCEDURE-BODY-INTERIOR  'END'
```

PL nº 20 GRAMMAIRE D'UN NOUVEL ALGOL

REGLES (SUITE)

```
                LONG-REAL-PROCEDURE-BODY ::= LONG-REAL-EXPRESSION
                                         OR 'BEGIN' LONG-REAL-PROCEDURE-BODY-INTERIOR 'END'
             SHORT-COMPLEX-PROCEDURE-BODY ::= SHORT-COMPLEX-EXPRESSION
                                         OR 'BEGIN' SHORT-COMPLEX-PROCEDURE-BODY-INTERIOR 'END'
              LONG-COMPLEX-PROCEDURE-BODY ::= LONG-COMPLEX-EXPRESSION
                                         OR 'BEGIN' LONG-COMPLEX-PROCEDURE-BODY-INTERIOR 'END'
                   LOGICAL-PROCEDURE-BODY ::= LOGICAL-EXPRESSION
                                         OR 'BEGIN' LOGICAL-PROCEDURE-BODY-INTERIOR 'END'
                       BIT-PROCEDURE-BODY ::= BIT-EXPRESSION
                                         OR 'BEGIN' BIT-PROCEDURE-BODY-INTERIOR 'END'
                    STRING-PROCEDURE-BODY ::= STRING-EXPRESSION
                                         OR 'BEGIN' STRING-PROCEDURE-BODY-INTERIOR 'END'
                 REFERENCE-PROCEDURE-BODY ::= REFERENCE-EXPRESSION
                                         OR 'BEGIN' REFERENCE-PROCEDURE-BODY-INTERIOR 'END'
          INTEGER-PROCEDURE-BODY-INTERIOR ::= INTEGER-EXPRESSION
                                         OR ., INTEGER-EXPRESSION
                                         OR 'ALSO' INTEGER-EXPRESSION
                                         OR BLOCK-INTERIOR-OR-DECLARATION-PART ., INTEGER-EXPRESSION
                                         OR BLOCK-INTERIOR 'ALSO' INTEGER-EXPRESSION
                                         OR INCOMPLETE-BLOCK-INTERIOR : INTEGER-EXPRESSION
       SHORT-REAL-PROCEDURE-BODY-INTERIOR ::= SHORT-REAL-EXPRESSION
                                         OR ., SHORT-REAL-EXPRESSION
                                         OR 'ALSO' SHORT-REAL-EXPRESSION
                                         OR BLOCK-INTERIOR-OR-DECLARATION-PART ., SHORT-REAL-EXPRESSION
                                         OR BLOCK-INTERIOR 'ALSO' SHORT-REAL-EXPRESSION
                                         OR INCOMPLETE-BLOCK-INTERIOR : SHORT-REAL-EXPRESSION
        LONG-REAL-PROCEDURE-BODY-INTERIOR ::= LONG-REAL-EXPRESSION
                                         OR ., LONG-REAL-EXPRESSION
                                         OR 'ALSO' LONG-REAL-EXPRESSION
                                         OR BLOCK-INTERIOR-OR-DECLARATION-PART ., LONG-REAL-EXPRESSION
                                         OR BLOCK-INTERIOR 'ALSO' LONG-REAL-EXPRESSION
                                         OR INCOMPLETE-BLOCK-INTERIOR : LONG-REAL-EXPRESSION
    SHORT-COMPLEX-PROCEDURE-BODY-INTERIOR ::= SHORT-COMPLEX-EXPRESSION
                                         OR ., SHORT-COMPLEX-EXPRESSION
                                         OR 'ALSO' SHORT-COMPLEX-EXPRESSION
                                         OR BLOCK-INTERIOR-OR-DECLARATION-PART ., SHORT-COMPLEX-EXPRESSION
                                         OR BLOCK-INTERIOR 'ALSO' SHORT-COMPLEX-EXPRESSION
                                         OR INCOMPLETE-BLOCK-INTERIOR : SHORT-COMPLEX-EXPRESSION
     LONG-COMPLEX-PROCEDURE-BODY-INTERIOR ::= LONG-COMPLEX-EXPRESSION
                                         OR ., LONG-COMPLEX-EXPRESSION
                                         OR 'ALSO' LONG-COMPLEX-EXPRESSION
                                         OR BLOCK-INTERIOR-OR-DECLARATION-PART ., LONG-COMPLEX-EXPRESSION
                                         OR BLOCK-INTERIOR 'ALSO' LONG-COMPLEX-EXPRESSION
                                         OR INCOMPLETE-BLOCK-INTERIOR : LONG-COMPLEX-EXPRESSION
          LOGICAL-PROCEDURE-BODY-INTERIOR ::= LOGICAL-EXPRESSION
                                         OR ., LOGICAL-EXPRESSION
                                         OR 'ALSO' LOGICAL-EXPRESSION
                                         OR BLOCK-INTERIOR-OR-DECLARATION-PART ., LOGICAL-EXPRESSION
                                         OR BLOCK-INTERIOR 'ALSO' LOGICAL-EXPRESSION
                                         OR INCOMPLETE-BLOCK-INTERIOR : LOGICAL-EXPRESSION
              BIT-PROCEDURE-BODY-INTERIOR ::= BIT-EXPRESSION
                                         OR ., BIT-EXPRESSION
                                         OR 'ALSO' BIT-EXPRESSION
                                         OR BLOCK-INTERIOR-OR-DECLARATION-PART ., BIT-EXPRESSION
                                         OR BLOCK-INTERIOR 'ALSO' BIT-EXPRESSION
                                         OR INCOMPLETE-BLOCK-INTERIOR : BIT-EXPRESSION
           STRING-PROCEDURE-BODY-INTERIOR ::= STRING-EXPRESSION
                                         OR ., STRING-EXPRESSION
                                         OR 'ALSO' STRING-EXPRESSION
                                         OR BLOCK-INTERIOR-OR-DECLARATION-PART ., STRING-EXPRESSION
                                         OR BLOCK-INTERIOR 'ALSO' STRING-EXPRESSION
                                         OR INCOMPLETE-BLOCK-INTERIOR : SRING-EXPRESSION
        REFERENCE-PROCEDURE-BODY-INTERIOR ::= REFERENCE-EXPRESSION
                                         OR ., REFERENCE-EXPRESSION
                                         OR 'ALSO' REFERENCE-EXPRESSION
                                         OR BLOCK-INTERIOR-OR-DECLARATION-PART ., REFERENCE-EXPRESSION
                                         OR BLOCK-INTERIOR 'ALSO' REFERENCE-EXPRESSION
                                         OR INCOMPLETE-BLOCK-INTERIOR : REFERENCE-EXPRESSION
                      FORMAL-PARAMETER-LIST ::= FORMAL-PARAMETER-SEGMENT
                                         OR FORMAL-PARAMETER-LIST ., FORMAL-PARAMETER-SEGMENT
                   FORMAL-PARAMETER-SEGMENT ::= SIMPLE-TYPE-FOLLOWED-BY-IDENTIFIER
                                         OR TYPE 'ARRAY' IDENTIFIER
                                         OR 'PROCEDURE' IDENTIFIER
                                         OR SIMPLE-TYPE 'PROCEDURE' IDENTIFIER
                                         OR SIMPLE-TYPE 'VALUE' IDENTIFIER
                                         OR SIMPLE-TYPE 'RESULT' IDENTIFIER
                                         OR SIMPLE-TYPE 'VALUE' 'RESULT' IDENTIFIER
                                         OR FORMAL-PARAMETER-SEGMENT , IDENTIFIER
                                 TYPE ::= SIMPLE-TYPE
                                         OR TYPE 'ARRAY'
                          SIMPLE-TYPE ::= SIMPLE-INTEGER-TYPE
                                         OR SIMPLE-SHORT-REAL-TYPE
                                         OR SIMPLE-LONG-REAL-TYPE
                                         OR SIMPLE-SHORT-COMPLEX-TYPE
                                         OR SIMPLE-LONG-COMPLEX-TYPE
                                         OR SIMPLE-LOGICAL-TYPE
                                         OR SIMPLE-BIT-TYPE
                                         OR SIMPLE-STRING-TYPE
                                         OR SIMPLE-REFERENCE-TYPE
                   SIMPLE-INTEGER-TYPE ::= 'INTEGER'
                SIMPLE-SHORT-REAL-TYPE ::= 'REAL'
                 SIMPLE-LONG-REAL-TYPE ::= 'LONG' 'REAL'
             SIMPLE-SHORT-COMPLEX-TYPE ::= 'COMPLEX'
              SIMPLE-LONG-COMPLEX-TYPE ::= 'LONG' 'COMPLEX'
                   SIMPLE-LOGICAL-TYPE ::= 'LOGICAL'
                      SIMPLE-BIT-TYPE ::= 'BITS'
                                         OR 'BITS' ( INTEGER-NUMBER )
                   SIMPLE-STRING-TYPE ::= 'STRING'
```

PL Nº 21 GRAMMAIRE D'UN NOUVEL ALGOL

REGLES (SUITE)

```
                    SIMPLE-REFERENCE-TYPE ::= 'REFERENCE'
                                          OR 'REFERENCE' ( RECORD-CLASS-IDENTIFIER )
                                STATEMENT ::= SIMPLE-STATEMENT
                                          OR IF-STATEMENT
                                          OR CASE-STATEMENT
                                          OR ITERATIVE-STATEMENT
                         SIMPLE-STATEMENT ::= BLOCK
                                          OR NUMERICAL-ASSIGNMENT-STATEMENT
                                          OR LOGICAL-ASSIGNMENT-STATEMENT
                                          OR BIT-ASSIGNMENT-STATEMENT
                                          OR STRING-ASSIGNMENT-STATEMENT
                                          OR REFERENCE-ASSIGNMENT-STATEMENT
                                          OR PROCEDURE-STATEMENT
                                          OR GO-TO-STATEMENT
          NUMERICAL-ASSIGNMENT-STATEMENT ::= COMPLEX-ASSIGNMENT-STATEMENT
                                          OR NUMERICAL-NON-COMPLEX-ASSIGNMENT-STATEMENT
NUMERICAL-NON-COMPLEX-ASSIGNMENT-STATEMENT ::= INTEGER-ASSIGNMENT-STATEMENT
                                          OR REAL-ASSIGNMENT-STATEMENT
          INTEGER-ASSIGNMENT-STATEMENT ::= INTEGER-VARIABLE := INTEGER-EXPRESSION
                                          OR INTEGER-VARIABLE := INTEGER-ASSIGNMENT-STATEMENT
             REAL-ASSIGNMENT-STATEMENT ::= REAL-VARIABLE := NUMERICAL-NON-COMPLEX-EXPRESSION
                                          OR REAL-VARIABLE := NUMERICAL-NON-COMPLEX-ASSIGNMENT-STATEMENT
          COMPLEX-ASSIGNMENT-STATEMENT ::= COMPLEX-VARIABLE := NUMERICAL-EXPRESSION
                                          OR COMPLEX-VARIABLE := NUMERICAL-ASSIGNMENT-STATEMENT
          LOGICAL-ASSIGNMENT-STATEMENT ::= LOGICAL-VARIABLE := LOGICAL-EXPRESSION
                                          OR LOGICAL-VARIABLE := LOGICAL-ASSIGNMENT-STATEMENT
             BIT-ASSIGNMENT-STATEMENT ::= BIT-VARIABLE := BIT-EXPRESSION
                                          OR BIT-VARIABLE := BIT-ASSIGNMENT-STATEMENT
          STRING-ASSIGNMENT-STATEMENT ::= STRING-VARIABLE := STRING-EXPRESSION
                                          OR STRING-VARIABLE := STRING-ASSIGNMENT-STATEMENT
       REFERENCE-ASSIGNMENT-STATEMENT ::= REFERENCE-VARIABLE := REFERENCE-EXPRESSION
                                          OR REFERENCE-VARIABLE := REFERENCE-ASSIGNMENT-STATEMENT
                    PROCEDURE-STATEMENT ::= PROCEDURE-IDENTIFIER
                                          OR PROCEDURE-IDENTIFIER ( )
                                          OR PROCEDURE-IDENTIFIER ( ACTUAL-PARAMETER-LIST )
                  ACTUAL-PARAMETER-LIST ::= ACTUAL-PARAMETER
                                          OR ,
                                          OR , ACTUAL-PARAMETER
                                          OR ACTUAL-PARAMETER-LIST ,
                                          OR ACTUAL-PARAMETER-LIST , ACTUAL-PARAMETER
                      ACTUAL-PARAMETER ::= EXPRESSION
                                          OR STATEMENT
                                          OR INTEGER-ARRAY-DESIGNATOR
                                          OR SHORT-REAL-ARRAY-DESIGNATOR
                                          OR LONG-REAL-ARRAY-DESIGNATOR
                                          OR SHORT-COMPLEX-ARRAY-DESIGNATOR
                                          OR LONG-COMPLEX-ARRAY-DESIGNATOR
                                          OR LOGICAL-ARRAY-DESIGNATOR
                                          OR BIT-ARRAY-DESIGNATOR
                                          OR STRING-ARRAY-DESIGNATOR
                                          OR REFERENCE-ARRAY-DESIGNATOR
                                          OR PROCEDURE-IDENTIFIER
                                          OR INTEGER-FUNCTION-IDENTIFIER
                                          OR SHORT-REAL-FUNCTION-IDENTIFIER
                                          OR LONG-REAL-FUNCTION-IDENTIFIER
                                          OR SHORT-COMPLEX-FUNCTION-IDENTIFIER
                                          OR LONG-COMPLEX-FUNCTION-IDENTIFIER
                                          OR LOGICAL-FUNCTION-IDENTIFIER
                                          OR BIT-FUNCTION-IDENTIFIER
                                          OR STRING-FUNCTION-IDENTIFIER
                                          OR REFERENCE-FUNCTION-IDENTIFIER
                      GO-TO-STATEMENT ::= 'GOTO' LABEL-IDENTIFIER
                        IF-STATEMENT ::= IF-CLAUSE
                                          OR IF-CLAUSE STATEMENT
                                          OR IF-CLAUSE 'ELSE'
                                          OR IF-CLAUSE 'ELSE' STATEMENT
                                          OR IF-CLAUSE SIMPLE-STATEMENT 'ELSE'
                                          OR IF-CLAUSE SIMPLE-STATEMENT 'ELSE' STATEMENT
                           IF-CLAUSE ::= 'IF' LOGICAL-EXPRESSION 'THEN'
                                          OR 'IF' DISCRIMINATOR 'THEN'
                       DISCRIMINATOR ::= REFERENCE-VARIABLE-IDENTIFIER 'IS' RECORD-CLASS-IDENTIFIER
                      CASE-STATEMENT ::= CASE-CLAUSE 'BEGIN' 'END'
                                          OR CASE-CLAUSE 'BEGIN' STATEMENT-LIST 'END'
                         CASE-CLAUSE ::= 'CASE' INTEGER-EXPRESSION 'OF'
                    STATEMENT-LIST ::= STATEMENT
                                          OR ;;
                                          OR STATEMENT-LIST ;;
                                          OR ;; STATEMENT
                                          OR STATEMENT-LIST ;; STATEMENT
                  ITERATIVE-STATEMENT ::= FOR-CLAUSE
                                          OR FOR-CLAUSE STATEMENT
                          FOR-CLAUSE ::= 'WHILE' LOGICAL-EXPRESSION 'DO'
                                          OR 'FOR' CONTROL-IDENTIFIER := INTEGER-EXPRESSION 'STEP'
                                          INTEGER-EXPRESSION 'UNTIL' INTEGER-EXPRESSION 'DO'
                         EXPRESSION ::= LOGICAL-EXPRESSION
                                          OR BIT-EXPRESSION
                                          OR STRING-EXPRESSION
                                          OR REFERENCE-EXPRESSION
                                          OR NUMERICAL-EXPRESSION
                 NUMERICAL-EXPRESSION ::= SHORT-COMPLEX-EXPRESSION
                                          OR LONG-COMPLEX-EXPRESSION
                                          OR NUMERICAL-NON-COMPLEX-EXPRESSION
     NUMERICAL-NON-COMPLEX-EXPRESSION ::= INTEGER-EXPRESSION
                                          OR SHORT-REAL-EXPRESSION
                                          OR LONG-REAL-EXPRESSION
INTEGER-OR-LONG-NUMERICAL-EXPRESSION ::= INTEGER-EXPRESSION
                                          OR LONG-REAL-EXPRESSION
                                          OR LONG-COMPLEX-EXPRESSION
```

PL nº 22 GRAMMAIRE D'UN NOUVEL ALGOL

REGLES (SUITE)

```
                    INTEGER-EXPRESSION ::= SIMPLE-INTEGER-EXPRESSION
                                       OR  CASE-CLAUSE ( INTEGER-EXPRESSION-LIST )
                                       OR  IF-CLAUSE SIMPLE-INTEGER-EXPRESSION 'ELSE' INTEGER-EXPRESSION
                   SHORT-REAL-EXPRESSION ::= SIMPLE-SHORT-REAL-EXPRESSION
                                       OR  CASE-CLAUSE ( SHORT-REAL-EXPRESSION-LIST )
                                       OR  IF-CLAUSE SIMPLE-SHORT-REAL-EXPRESSION 'ELSE'
                                           NUMERICAL-NON-COMPLEX-EXPRESSION
                                       OR  IF-CLAUSE SIMPLE-INTEGER-OR-LONG-REAL-EXPRESSION 'ELSE'
                                           SHORT-REAL-EXPRESSION
                    LONG-REAL-EXPRESSION ::= SIMPLE-LONG-REAL-EXPRESSION
                                       OR  CASE-CLAUSE ( LONG-REAL-EXPRESSION-LIST )
                                       OR  IF-CLAUSE SIMPLE-LONG-REAL-EXPRESSION 'ELSE' INTEGER-EXPRESSION
                                       OR  IF-CLAUSE SIMPLE-INTEGER-OR-LONG-REAL-EXPRESSION 'ELSE'
                                           LONG-REAL-EXPRESSION
                 SHORT-COMPLEX-EXPRESSION ::= SIMPLE-SHORT-COMPLEX-EXPRESSION
                                       OR  CASE-CLAUSE ( SHORT-COMPLEX-EXPRESSION-LIST )
                                       OR  IF-CLAUSE SIMPLE-SHORT-COMPLEX-EXPRESSION 'ELSE'
                                           NUMERICAL-EXPRESSION
                                       OR  IF-CLAUSE SIMPLE-NUMERICAL-NON-SHORT-COMPLEX-EXPRESSION 'ELSE'
                                           SHORT-COMPLEX-EXPRESSION
                                       OR  IF-CLAUSE SIMPLE-LONG-COMPLEX-EXPRESSION 'ELSE'
                                           SHORT-REAL-EXPRESSION
                                       OR  IF-CLAUSE SIMPLE-SHORT-REAL-EXPRESSION 'ELSE'
                                           LONG-COMPLEX-EXPRESSION
                  LONG-COMPLEX-EXPRESSION ::= SIMPLE-LONG-COMPLEX-EXPRESSION
                                       OR  CASE-CLAUSE ( LONG-COMPLEX-EXPRESSION-LIST )
                                       OR  IF-CLAUSE SIMPLE-LONG-COMPLEX-EXPRESSION 'ELSE'
                                           INTEGER-OR-LONG-NUMERICAL-EXPRESSION
                                       OR  IF-CLAUSE SIMPLE-INTEGER-OR-LONG-REAL-EXPRESSION 'ELSE'
                                           LONG-COMPLEX-EXPRESSION
                     LOGICAL-EXPRESSION ::= SIMPLE-LOGICAL-EXPRESSION
                                       OR  CASE-CLAUSE ( LOGICAL-EXPRESSION-LIST )
                                       OR  IF-CLAUSE SIMPLE-LOGICAL-EXPRESSION 'ELSE' LOGICAL-EXPRESSION
                        BIT-EXPRESSION ::= SIMPLE-BIT-EXPRESSION
                                       OR  CASE-CLAUSE ( BIT-EXPRESSION-LIST )
                                       OR  IF-CLAUSE SIMPLE-BIT-EXPRESSION 'ELSE' BIT-EXPRESSION
                     STRING-EXPRESSION ::= SIMPLE-STRING-EXPRESSION
                                       OR  CASE-CLAUSE ( STRING-EXPRESSION-LIST )
                                       OR  IF-CLAUSE SIMPLE-STRING-EXPRESSION 'ELSE' STRING-EXPRESSION
                  REFERENCE-EXPRESSION ::= SIMPLE-REFERENCE-EXPRESSION
                                       OR  CASE-CLAUSE ( REFERENCE-EXPRESSION-LIST )
                                       OR  IF-CLAUSE SIMPLE-REFERENCE-EXPRESSION 'ELSE' REFERENCE-EXPRESSION
NUMERICAL-NON-SHORT-COMPLEX-EXPRESSION-LIST ::= SHORT-REAL-EXPRESSION-LIST
                                       OR  LONG-COMPLEX-EXPRESSION-LIST
                                       OR  INTEGER-OR-LONG-REAL-EXPRESSION-LIST
      INTEGER-OR-LONG-REAL-EXPRESSION-LIST ::= INTEGER-EXPRESSION-LIST
                                       OR  LONG-REAL-EXPRESSION-LIST
                 INTEGER-EXPRESSION-LIST ::= INTEGER-EXPRESSION
                                       OR  INTEGER-EXPRESSION-LIST , INTEGER-EXPRESSION
               SHORT-REAL-EXPRESSION-LIST ::= SHORT-REAL-EXPRESSION
                                       OR  SHORT-REAL-EXPRESSION-LIST , NUMERICAL-NON-COMPLEX-EXPRESSION
                                       OR  INTEGER-OR-LONG-REAL-EXPRESSION-LIST , SHORT-REAL-EXPRESSION
                LONG-REAL-EXPRESSION-LIST ::= LONG-REAL-EXPRESSION
                                       OR  LONG-REAL-EXPRESSION-LIST , INTEGER-EXPRESSION
                                       OR  INTEGER-OR-LONG-REAL-EXPRESSION-LIST , LONG-REAL-EXPRESSION
            SHORT-COMPLEX-EXPRESSION-LIST ::= SHORT-COMPLEX-EXPRESSION
                                       OR  SHORT-COMPLEX-EXPRESSION-LIST , NUMERICAL-EXPRESSION
                                       OR  NUMERICAL-NON-SHORT-COMPLEX-EXPRESSION-LIST ,
                                           SHORT-COMPLEX-EXPRESSION
                                       OR  SHORT-REAL-EXPRESSION-LIST , LONG-COMPLEX-EXPRESSION
                                       OR  LONG-COMPLEX-EXPRESSION-LIST , SHORT-REAL-EXPRESSION
             LONG-COMPLEX-EXPRESSION-LIST ::= LONG-COMPLEX-EXPRESSION
                                       OR  LONG-COMPLEX-EXPRESSION-LIST ,
                                           INTEGER-OR-LONG-NUMERICAL-EXPRESSION
                                       OR  INTEGER-OR-LONG-REAL-EXPRESSION-LIST , LONG-COMPLEX-EXPRESSION
               LOGICAL-EXPRESSION-LIST ::= LOGICAL-EXPRESSION
                                       OR  LOGICAL-EXPRESSION-LIST , LOGICAL-EXPRESSION
                  BIT-EXPRESSION-LIST ::= BIT-EXPRESSION
                                       OR  BIT-EXPRESSION-LIST , BIT-EXPRESSION
                STRING-EXPRESSION-LIST ::= STRING-EXPRESSION
                                       OR  STRING-EXPRESSION-LIST , STRING-EXPRESSION
               REFERENCE-EXPRESSION-LIST ::= REFERENCE-EXPRESSION
                                       OR  REFERENCE-EXPRESSION-LIST , REFERENCE-EXPRESSION
             SIMPLE-NUMERICAL-EXPRESSION ::= SIMPLE-SHORT-COMPLEX-EXPRESSION
                                       OR  SIMPLE-NUMERICAL-NON-SHORT-COMPLEX-EXPRESSION
SIMPLE-NUMERICAL-NON-SHORT-COMPLEX-EXPRESSION ::= SIMPLE-LONG-COMPLEX-EXPRESSION
                                       OR  SIMPLE-NUMERICAL-NON-COMPLEX-EXPRESSION
      SIMPLE-NUMERICAL-NON-COMPLEX-EXPRESSION ::= SIMPLE-SHORT-REAL-EXPRESSION
                                       OR  SIMPLE-INTEGER-OR-LONG-REAL-EXPRESSION
       SIMPLE-INTEGER-OR-LONG-REAL-EXPRESSION ::= SIMPLE-INTEGER-EXPRESSION
                                       OR  SIMPLE-LONG-REAL-EXPRESSION
                SIMPLE-INTEGER-EXPRESSION ::= INTEGER-TERM
                                       OR  SIGN INTEGER-TERM
                                       OR  SIMPLE-INTEGER-EXPRESSION SIGN INTEGER-TERM
              SIMPLE-SHORT-REAL-EXPRESSION ::= SHORT-REAL-TERM
                                       OR  SIGN SHORT-REAL-TERM
                                       OR  SIMPLE-SHORT-REAL-EXPRESSION SIGN NUMERICAL-NON-COMPLEX-TERM
                                       OR  SIMPLE-INTEGER-OR-LONG-REAL-EXPRESSION SIGN SHORT-REAL-TERM
               SIMPLE-LONG-REAL-EXPRESSION ::= LONG-REAL-TERM
                                       OR  SIGN LONG-REAL-TERM
                                       OR  SIMPLE-LONG-REAL-EXPRESSION SIGN INTEGER-TERM
                                       OR  SIMPLE-INTEGER-OR-LONG-REAL-EXPRESSION SIGN LONG-REAL-TERM
            SIMPLE-SHORT-COMPLEX-EXPRESSION ::= SHORT-COMPLEX-TERM
                                       OR  SIGN SHORT-COMPLEX-TERM
                                       OR  SIMPLE-SHORT-COMPLEX-EXPRESSION SIGN
                                           NUMERICAL-NON-SHORT-COMPLEX-TERM
                                       OR  SIMPLE-NUMERICAL-EXPRESSION SIGN SHORT-COMPLEX-TERM
                                       OR  SIMPLE-LONG-COMPLEX-EXPRESSION SIGN SHORT-REAL-TERM
                                       OR  SIMPLE-SHORT-REAL-EXPRESSION SIGN LONG-COMPLEX-TERM
```

PL nº 23 GRAMMAIRE D'UN NOUVEL ALGOL

REGLES (SUITE)

```
SIMPLE-LONG-COMPLEX-EXPRESSION ::= LONG-COMPLEX-TERM
                                OR SIGN LONG-COMPLEX-TERM
                                OR SIMPLE-LONG-COMPLEX-EXPRESSION SIGN INTEGER-OR-LONG-NUMERICAL-TERM
                                OR SIMPLE-INTEGER-OR-LONG-REAL-EXPRESSION SIGN LONG-COMPLEX-TERM
                     SIGN ::= +
                                OR -
NUMERICAL-NON-SHORT-COMPLEX-TERM ::= LONG-COMPLEX-TERM
                                OR NUMERICAL-NON-COMPLEX-TERM
      NUMERICAL-NON-COMPLEX-TERM ::= INTEGER-TERM
                                OR REAL-TERM
  INTEGER-OR-LONG-NUMERICAL-TERM ::= INTEGER-TERM
                                OR LONG-REAL-TERM
                                OR LONG-COMPLEX-TERM
                   REAL-TERM ::= SHORT-REAL-TERM
                                OR LONG-REAL-TERM
                 COMPLEX-TERM ::= SHORT-COMPLEX-TERM
                                OR LONG-COMPLEX-TERM
                INTEGER-TERM ::= INTEGER-FACTOR
                                OR INTEGER-TERM '*' INTEGER-FACTOR
                                OR INTEGER-TERM 'DIV' INTEGER-FACTOR
                                OR INTEGER-TERM 'REM' INTEGER-FACTOR
           SHORT-REAL-TERM ::= SHORT-REAL-FACTOR
                                OR NUMERICAL-NON-COMPLEX-TERM / SHORT-REAL-FACTOR
                                OR SHORT-REAL-TERM / INTEGER-OR-LONG-REAL-FACTOR
                                OR INTEGER-TERM / INTEGER-FACTOR
            LONG-REAL-TERM ::= LONG-REAL-FACTOR
                                OR NUMERICAL-NON-COMPLEX-TERM * REAL-FACTOR
                                OR REAL-TERM * INTEGER-FACTOR
                                OR LONG-REAL-TERM / INTEGER-OR-LONG-REAL-FACTOR
                                OR INTEGER-TERM / LONG-REAL-FACTOR
         SHORT-COMPLEX-TERM ::= SHORT-COMPLEX-FACTOR
                                OR SHORT-COMPLEX-TERM / NUMERICAL-FACTOR
                                OR NUMERICAL-NON-SHORT-COMPLEX-TERM / SHORT-COMPLEX-FACTOR
                                OR LONG-COMPLEX-TERM / SHORT-REAL-FACTOR
                                OR SHORT-REAL-TERM / LONG-COMPLEX-FACTOR
          LONG-COMPLEX-TERM ::= LONG-COMPLEX-FACTOR
                                OR COMPLEX-TERM * NUMERICAL-FACTOR
                                OR NUMERICAL-NON-COMPLEX-TERM * COMPLEX-FACTOR
                                OR LONG-COMPLEX-TERM / INTEGER-OR-LONG-REAL-FACTOR
                                OR INTEGER-OR-LONG-NUMERICAL-TERM / LONG-COMPLEX-FACTOR
        NUMERICAL-FACTOR ::= INTEGER-FACTOR
                                OR REAL-FACTOR
                                OR COMPLEX-FACTOR
  INTEGER-OR-LONG-REAL-FACTOR ::= INTEGER-FACTOR
                                OR LONG-REAL-FACTOR
               REAL-FACTOR ::= SHORT-REAL-FACTOR
                                OR LONG-REAL-FACTOR
            COMPLEX-FACTOR ::= SHORT-COMPLEX-FACTOR
                                OR LONG-COMPLEX-FACTOR
           INTEGER-FACTOR ::= INTEGER-SECONDARY
       SHORT-REAL-FACTOR ::= SHORT-REAL-SECONDARY
                                OR INTEGER-FACTOR ** INTEGER-SECONDARY
                                OR SHORT-REAL-FACTOR ** INTEGER-SECONDARY
        LONG-REAL-FACTOR ::= LONG-REAL-SECONDARY
                                OR LONG-REAL-FACTOR ** INTEGER-SECONDARY
     SHORT-COMPLEX-FACTOR ::= SHORT-COMPLEX-SECONDARY
                                OR SHORT-COMPLEX-FACTOR ** INTEGER-SECONDARY
      LONG-COMPLEX-FACTOR ::= LONG-COMPLEX-SECONDARY
                                OR LONG-COMPLEX-FACTOR ** INTEGER-SECONDARY
       INTEGER-SECONDARY ::= INTEGER-PRIMARY
                                OR INTEGER-NUMBER
                                OR 'ABS' INTEGER-PRIMARY
     SHORT-REAL-SECONDARY ::= SHORT-REAL-PRIMARY
                                OR SHORT-REAL-NUMBER
                                OR 'ABS' SHORT-REAL-PRIMARY
                                OR 'ABS' SHORT-COMPLEX-PRIMARY
      LONG-REAL-SECONDARY ::= LONG-REAL-PRIMARY
                                OR LONG-REAL-NUMBER
                                OR 'ABS' LONG-REAL-PRIMARY
                                OR 'ABS' LONG-COMPLEX-PRIMARY
  SHORT-COMPLEX-SECONDARY ::= SHORT-COMPLEX-PRIMARY
                                OR SHORT-COMPLEX-NUMBER
   LONG-COMPLEX-SECONDARY ::= LONG-COMPLEX-PRIMARY
                                OR LONG-COMPLEX-NUMBER
         INTEGER-PRIMARY ::= INTEGER-VARIABLE
                                OR INTEGER-FUNCTION-IDENTIFIER
                                OR INTEGER-FUNCTION-IDENTIFIER ( )
                                OR INTEGER-FUNCTION-IDENTIFIER ( ACTUAL-PARAMETER-LIST )
                                OR ( INTEGER-EXPRESSION )
      SHORT-REAL-PRIMARY ::= SHORT-REAL-VARIABLE
                                OR SHORT-REAL-FUNCTION-IDENTIFIER
                                OR SHORT-REAL-FUNCTION-IDENTIFIER ( )
                                OR SHORT-REAL-FUNCTION-IDENTIFIER ( ACTUAL-PARAMETER-LIST )
                                OR ( SHORT-REAL-EXPRESSION )
       LONG-REAL-PRIMARY ::= LONG-REAL-VARIABLE
                                OR LONG-REAL-FUNCTION-IDENTIFIER
                                OR LONG-REAL-FUNCTION-IDENTIFIER ( )
                                OR LONG-REAL-FUNCTION-IDENTIFIER ( ACTUAL-PARAMETER-LIST )
                                OR 'LONG' SHORT-REAL-PRIMARY
                                OR 'LONG' LONG-REAL-PRIMARY
                                OR ( LONG-REAL-EXPRESSION )
    SHORT-COMPLEX-PRIMARY ::= SHORT-COMPLEX-VARIABLE
                                OR SHORT-COMPLEX-FUNCTION-IDENTIFIER
                                OR SHORT-COMPLEX-FUNCTION-IDENTIFIER ( )
                                OR SHORT-COMPLEX-FUNCTION-IDENTIFIER ( ACTUAL-PARAMETER-LIST )
                                OR ( SHORT-COMPLEX-EXPRESSION )
```

PL nº 24 GRAMMAIRE D'UN NOUVEL ALGOL

REGLES (SUITE)

```
                LONG-COMPLEX-PRIMARY ::= LONG-COMPLEX-VARIABLE
                                     OR  LONG-COMPLEX-FUNCTION-IDENTIFIER
                                     OR  LONG-COMPLEX-FUNCTION-IDENTIFIER  (  )
                                     OR  LONG-COMPLEX-FUNCTION-IDENTIFIER  (  ACTUAL-PARAMETER-LIST  )
                                     OR  'LONG'  SHORT-COMPLEX-PRIMARY
                                     OR  'LONG'  LONG-COMPLEX-PRIMARY
                                     OR  (  LONG-COMPLEX-EXPRESSION  )
           SIMPLE-LOGICAL-EXPRESSION ::= LOGICAL-TERM
                                     OR  RELATION
                            RELATION ::= LOGICAL-TERM  EQUALITY-OPERATOR  LOGICAL-TERM
                                     OR  SIMPLE-BIT-EXPRESSION  EQUALITY-OPERATOR  SIMPLE-BIT-EXPRESSION
                                     OR  SIMPLE-STRING-EXPRESSION  EQUALITY-OPERATOR
                                         SIMPLE-STRING-EXPRESSION
                                     OR  SIMPLE-REFERENCE-EXPRESSION  EQUALITY-OPERATOR
                                         SIMPLE-REFERENCE-EXPRESSION
                                     OR  SIMPLE-NUMERICAL-EXPRESSION  EQUALITY-OPERATOR
                                         SIMPLE-NUMERICAL-EXPRESSION
                                     OR  SIMPLE-NUMERICAL-NON-COMPLEX-EXPRESSION  ORDER-RELATION-OPERATOR
                                         SIMPLE-NUMERICAL-NON-COMPLEX-EXPRESSION
                   EQUALITY-OPERATOR ::= =
                                     OR  'NOT='
                 ORDER-RELATION-OPERATOR ::= 'GREATER'
                                     OR  'NOTGREATER'
                                     OR  'LESS'
                                     OR  'NOTLESS'
                        LOGICAL-TERM ::= LOGICAL-FACTOR
                                     OR  LOGICAL-TERM  'OR'  LOGICAL-FACTOR
                      LOGICAL-FACTOR ::= LOGICAL-SECONDARY
                                     OR  LOGICAL-FACTOR  'AND'  LOGICAL-SECONDARY
                   LOGICAL-SECONDARY ::= LOGICAL-PRIMARY
                                     OR  'NOT'  LOGICAL-PRIMARY
                     LOGICAL-PRIMARY ::= 'TRUE'
                                     OR  'FALSE'
                                     OR  LOGICAL-VARIABLE
                                     OR  LOGICAL-FUNCTION-IDENTIFIER
                                     OR  LOGICAL-FUNCTION-IDENTIFIER  (  )
                                     OR  LOGICAL-FUNCTION-IDENTIFIER  (  ACTUAL-PARAMETER-LIST  )
                                     OR  (  LOGICAL-EXPRESSION  )
             SIMPLE-BIT-EXPRESSION ::= BIT-TERM
                                     OR  SIMPLE-BIT-EXPRESSION  'OR'  BIT-TERM
                          BIT-TERM ::= BIT-FACTOR
                                     OR  BIT-TERM  'AND'  BIT-FACTOR
                        BIT-FACTOR ::= BIT-SECONDARY
                                     OR  'NOT'  BIT-SECONDARY
                      BIT-SECONDARY ::= BIT-PRIMARY
                                     OR  BIT-SECONDARY  **  INTEGER-SECONDARY
                                     OR  BIT-SECONDARY  //  INTEGER-SECONDARY
                        BIT-PRIMARY ::= BIT-SEQUENCE
                                     OR  BIT-VARIABLE
                                     OR  BIT-FUNCTION-IDENTIFIER
                                     OR  BIT-FUNCTION-IDENTIFIER  (  )
                                     OR  BIT-FUNCTION-IDENTIFIER  (  ACTUAL-PARAMETER-LIST  )
                                     OR  (  BIT-EXPRESSION  )
                       BIT-SEQUENCE ::= 'B'  BIT
                                     OR  BIT-SEQUENCE  BIT
           SIMPLE-STRING-EXPRESSION ::= STRING-PRIMARY
                                     OR  SIMPLE-STRING-EXPRESSION  'CAT'  STRING-PRIMARY
                     STRING-PRIMARY ::= ''  ''
                                     OR  STRING-VARIABLE
                                     OR  STRING-FUNCTION-IDENTIFIER
                                     OR  STRING-FUNCTION-IDENTIFIER  (  )
                                     OR  STRING-FUNCTION-IDENTIFIER  (  ACTUAL-PARAMETER-LIST  )
                                     OR  (  STRING-EXPRESSION  )
        SIMPLE-REFERENCE-EXPRESSION ::= 'NULL'
                                     OR  REFERENCE-VARIABLE
                                     OR  REFERENCE-FUNCTION-IDENTIFIER
                                     OR  REFERENCE-FUNCTION-IDENTIFIER  (  )
                                     OR  REFERENCE-FUNCTION-IDENTIFIER  (  ACTUAL-PARAMETER-LIST  )
                                     OR  RECORD-CLASS-IDENTIFIER
                                     OR  RECORD-CLASS-IDENTIFIER  (  EXPRESSION-LIST  )
                                     OR  (  REFERENCE-EXPRESSION  )
                   EXPRESSION-LIST ::= EXPRESSION
                                     OR  EXPRESSION-LIST  ,  EXPRESSION
                  INTEGER-VARIABLE ::= INTEGER-VARIABLE-IDENTIFIER
                                     OR  INTEGER-FIELD-IDENTIFIER  (  REFERENCE-EXPRESSION  )
                                     OR  INTEGER-ARRAY-DESIGNATOR  .(  INTEGER-EXPRESSION  ).
                     REAL-VARIABLE ::= SHORT-REAL-VARIABLE
                                     OR  LONG-REAL-VARIABLE
                  COMPLEX-VARIABLE ::= SHORT-COMPLEX-VARIABLE
                                     OR  LONG-COMPLEX-VARIABLE
            SHORT-REAL-VARIABLE ::= SHORT-REAL-VARIABLE-IDENTIFIER
                                     OR  SHORT-REAL-FIELD-IDENTIFIER  (  REFERENCE-EXPRESSION  )
                                     OR  SHORT-REAL-ARRAY-DESIGNATOR  .(  INTEGER-EXPRESSION  ).
             LONG-REAL-VARIABLE ::= LONG-REAL-VARIABLE-IDENTIFIER
                                     OR  LONG-REAL-FIELD-IDENTIFIER  (  REFERENCE-EXPRESSION  )
                                     OR  LONG-REAL-ARRAY-DESIGNATOR  .(  INTEGER-EXPRESSION  ).
          SHORT-COMPLEX-VARIABLE ::= SHORT-COMPLEX-VARIABLE-IDENTIFIER
                                     OR  SHORT-COMPLEX-FIELD-IDENTIFIER  (  REFERENCE-EXPRESSION  )
                                     OR  SHORT-COMPLEX-ARRAY-DESIGNATOR  .(  INTEGER-EXPRESSION  ).
           LONG-COMPLEX-VARIABLE ::= LONG-COMPLEX-VARIABLE-IDENTIFIER
                                     OR  LONG-COMPLEX-FIELD-IDENTIFIER  (  REFERENCE-EXPRESSION  )
                                     OR  LONG-COMPLEX-ARRAY-DESIGNATOR  .(  INTEGER-EXPRESSION  ).
            LOGICAL-VARIABLE ::= LOGICAL-VARIABLE-IDENTIFIER
                                     OR  LOGICAL-FIELD-IDENTIFIER  (  REFERENCE-EXPRESSION  )
                                     OR  LOGICAL-ARRAY-DESIGNATOR  .(  INTEGER-EXPRESSION  ).
                 BIT-VARIABLE ::= BIT-VARIABLE-IDENTIFIER
                                     OR  BIT-FIELD-IDENTIFIER  (  REFERENCE-EXPRESSION  )
                                     OR  BIT-ARRAY-DESIGNATOR  .(  INTEGER-EXPRESSION  ).
               STRING-VARIABLE ::= STRING-VARIABLE-IDENTIFIER
                                     OR  STRING-FIELD-IDENTIFIER  (  REFERENCE-EXPRESSION  )
                                     OR  STRING-ARRAY-DESIGNATOR  .(  INTEGER-EXPRESSION  ).
```

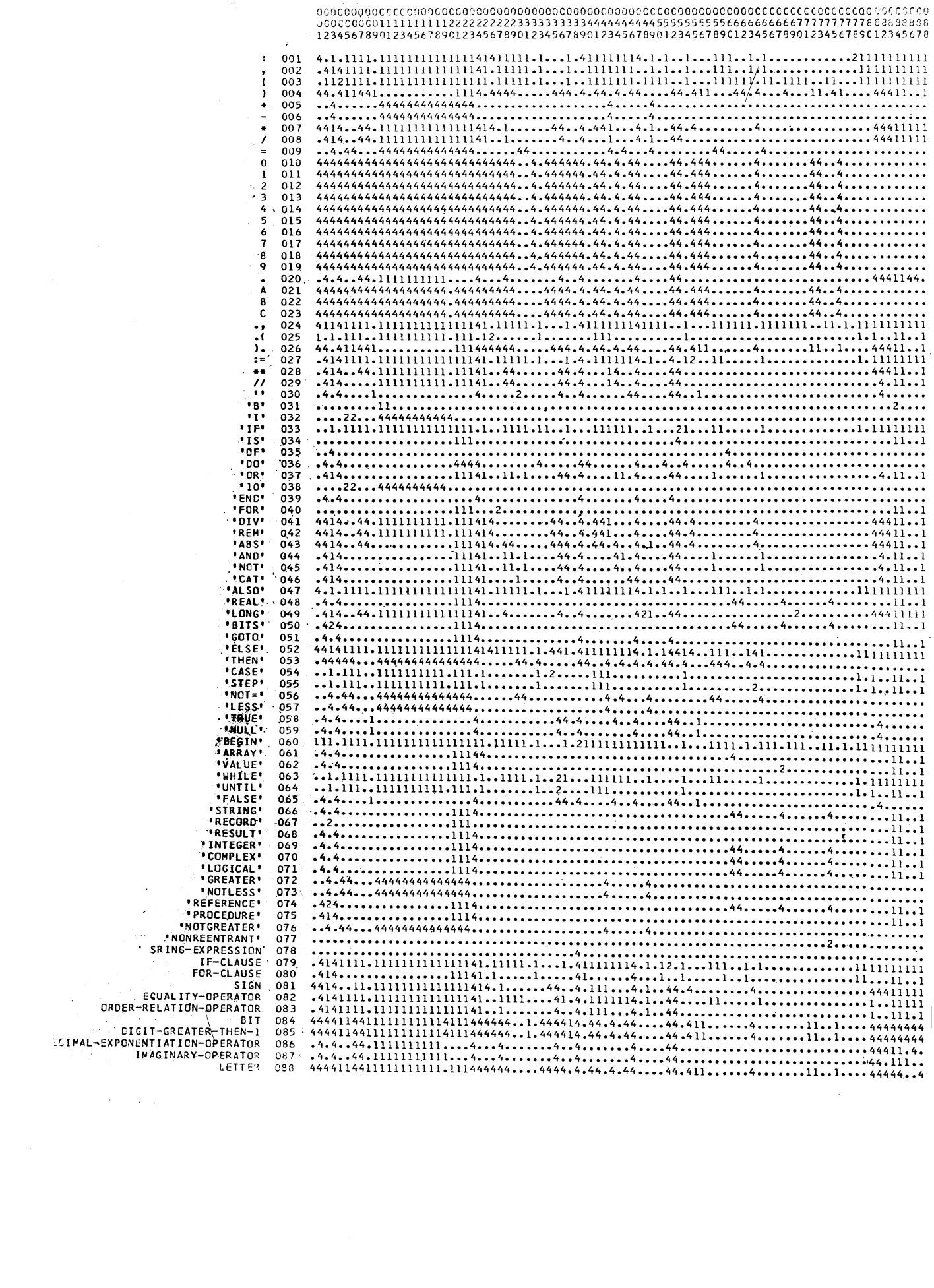
PL nº 25 GRAMMAIRE D'UN NOUVEL ALGOL

REGLES (SUITE)

```
                REFERENCE-VARIABLE ::= REFERENCE-VARIABLE-IDENTIFIER
                                    OR REFERENCE-FIELD-IDENTIFIER ( REFERENCE-EXPRESSION )
                                    OR REFERENCE-ARRAY-DESIGNATOR .( INTEGER-EXPRESSION ).
             INTEGER-ARRAY-DESIGNATOR ::= INTEGER-ARRAY-IDENTIFIER
                                    OR INTEGER-ARRAY-DESIGNATOR .( INTEGER-EXPRESSION ).
           SHORT-REAL-ARRAY-DESIGNATOR ::= SHORT-REAL-ARRAY-IDENTIFIER
                                    OR SHORT-REAL-ARRAY-DESIGNATOR .( INTEGER-EXPRESSION ).
            LONG-REAL-ARRAY-DESIGNATOR ::= LONG-REAL-ARRAY-IDENTIFIER
                                    OR LONG-REAL-ARRAY-DESIGNATOR .( INTEGER-EXPRESSION ).
        SHORT-COMPLEX-ARRAY-DESIGNATOR ::= SHORT-COMPLEX-ARRAY-IDENTIFIER
                                    OR SHORT-COMPLEX-ARRAY-DESIGNATOR .( INTEGER-EXPRESSION ).
         LONG-COMPLEX-ARRAY-DESIGNATOR ::= LONG-COMPLEX-ARRAY-IDENTIFIER
                                    OR LONG-COMPLEX-ARRAY-DESIGNATOR .( INTEGER-EXPRESSION ).
             LOGICAL-ARRAY-DESIGNATOR ::= LOGICAL-ARRAY-IDENTIFIER
                                    OR LOGICAL-ARRAY-DESIGNATOR .( INTEGER-EXPRESSION ).
                BIT-ARRAY-DESIGNATOR ::= BIT-ARRAY-IDENTIFIER
                                    OR BIT-ARRAY-DESIGNATOR .( INTEGER-EXPRESSION ).
             STRING-ARRAY-DESIGNATOR ::= STRING-ARRAY-IDENTIFIER
                                    OR STRING-ARRAY-DESIGNATOR .( INTEGER-EXPRESSION ).
          REFERENCE-ARRAY-DESIGNATOR ::= REFERENCE-ARRAY-IDENTIFIER
                                    OR REFERENCE-ARRAY-DESIGNATOR .( INTEGER-EXPRESSION ).
                   INTEGER-NUMBER ::= BIT
                                    OR DIGIT-GREATER-THEN-1
                                    OR INTEGER-NUMBER BIT
                                    OR INTEGER-NUMBER DIGIT-GREATER-THEN-1
                              BIT ::= 0
                                    OR 1
                DIGIT-GREATER-THEN-1 ::= 2
                                    OR 3
                                    OR 4
                                    OR 5
                                    OR 6
                                    OR 7
                                    OR 8
                                    OR 9
                     SHORT-REAL-NUMBER ::= UNSCALED-REAL
                                    OR UNSCALED-REAL DECIMAL-EXPONENTIATION-OPERATOR INTEGER-NUMBER
                                    OR INTEGER-NUMBER DECIMAL-EXPONENTIATION-OPERATOR INTEGER-NUMBER
   DECIMAL-EXPONENTIATION-OPERATOR ::= '10'
                                    OR '10' +
                                    OR '10' -
                       UNSCALED-REAL ::= . INTEGER-NUMBER
                                    OR INTEGER-NUMBER . INTEGER-NUMBER
                     LONG-REAL-NUMBER ::= 'LONG' SHORT-REAL-NUMBER
                                    OR 'LONG' INTEGER-NUMBER
                  SHORT-COMPLEX-NUMBER ::= INTEGER-OR-SHORT-REAL-NUMBER IMAGINARY-OPERATOR
                                           INTEGER-OR-SHORT-REAL-NUMBER
                    IMAGINARY-OPERATOR ::= 'I'
                                    OR 'I' +
                                    OR 'I' -
          INTEGER-OR-SHORT-REAL-NUMBER ::= INTEGER-NUMBER
                                    OR SHORT-REAL-NUMBER
                   LONG-COMPLEX-NUMBER ::= 'LONG' SHORT-COMPLEX-NUMBER
        INTEGER-VARIABLE-IDENTIFIER ::= IDENTIFIER
      SHORT-REAL-VARIABLE-IDENTIFIER ::= IDENTIFIER
       LONG-REAL-VARIABLE-IDENTIFIER ::= IDENTIFIER
    SHORT-COMPLEX-VARIABLE-IDENTIFIER ::= IDENTIFIER
     LONG-COMPLEX-VARIABLE-IDENTIFIER ::= IDENTIFIER
        LOGICAL-VARIABLE-IDENTIFIER ::= IDENTIFIER
           BIT-VARIABLE-IDENTIFIER ::= IDENTIFIER
        STRING-VARIABLE-IDENTIFIER ::= IDENTIFIER
     REFERENCE-VARIABLE-IDENTIFIER ::= IDENTIFIER
         INTEGER-ARRAY-IDENTIFIER ::= IDENTIFIER
       SHORT-REAL-ARRAY-IDENTIFIER ::= IDENTIFIER
        LONG-REAL-ARRAY-IDENTIFIER ::= IDENTIFIER
    SHORT-COMPLEX-ARRAY-IDENTIFIER ::= IDENTIFIER
     LONG-COMPLEX-ARRAY-IDENTIFIER ::= IDENTIFIER
        LOGICAL-ARRAY-IDENTIFIER ::= IDENTIFIER
           BIT-ARRAY-IDENTIFIER ::= IDENTIFIER
        STRING-ARRAY-IDENTIFIER ::= IDENTIFIER
     REFERENCE-ARRAY-IDENTIFIER ::= IDENTIFIER
           PROCEDURE-IDENTIFIER ::= IDENTIFIER
      INTEGER-FUNCTION-IDENTIFIER ::= IDENTIFIER
    SHORT-REAL-FUNCTION-IDENTIFIER ::= IDENTIFIER
     LONG-REAL-FUNCTION-IDENTIFIER ::= IDENTIFIER
  SHORT-COMPLEX-FUNCTION-IDENTIFIER ::= IDENTIFIER
   LONG-COMPLEX-FUNCTION-IDENTIFIER ::= IDENTIFIER
      LOGICAL-FUNCTION-IDENTIFIER ::= IDENTIFIER
         BIT-FUNCTION-IDENTIFIER ::= IDENTIFIER
      STRING-FUNCTION-IDENTIFIER ::= IDENTIFIER
   REFERENCE-FUNCTION-IDENTIFIER ::= IDENTIFIER
         RECORD-CLASS-IDENTIFIER ::= IDENTIFIER
       INTEGER-FIELD-IDENTIFIER ::= IDENTIFIER
     SHORT-REAL-FIELD-IDENTIFIER ::= IDENTIFIER
      LONG-REAL-FIELD-IDENTIFIER ::= IDENTIFIER
  SHORT-COMPLEX-FIELD-IDENTIFIER ::= IDENTIFIER
   LONG-COMPLEX-FIELD-IDENTIFIER ::= IDENTIFIER
      LOGICAL-FIELD-IDENTIFIER ::= IDENTIFIER
         BIT-FIELD-IDENTIFIER ::= IDENTIFIER
      STRING-FIELD-IDENTIFIER ::= IDENTIFIER
   REFERENCE-FIELD-IDENTIFIER ::= IDENTIFIER
               LABEL-IDENTIFIER ::= IDENTIFIER
             CONTROL-IDENTIFIER ::= IDENTIFIER
                 IDENTIFIER ::= LETTER
                                    OR IDENTIFIER LETTER
                                    OR IDENTIFIER BIT
                                    OR IDENTIFIER DIGIT-GREATER-THEN-1
                     LETTER ::= A
                                    OR B
                                    OR C
```

PL nº 26 GRAMMAIRE D'UN NOUVEL ALGOL

RELATIONS DE PRECEDENCE DE GAUCHE A DROITE ENTRE LES OPERATEURS



PL n° 27 GRAMMAIRE D'UN NOUVEL ALGOL

RELATION $\Rightarrow^*$ ENTRE LES OPERANDES

		CCCCOCOOOCO11111111111111111111111111111111111111111111111111111111111
		8999999999900000000001111111111222222222233333333334444444444555555555
		9012345678901234567890123456789012345678901234567890123456789012345678
PROGRAM	089	11..
BLOCK	090	.1..
BLOCK-INTERIOR	091	.111..1...
STATEMENT	092	.1.1..1...
BLOCK-INTERIOR-OR-DECLARATION-PART	093	.1111.111...................1111......................................
INCOMPLETE-BLOCK-INTERIOR	094	.....11...
IDENTIFIER	095	......1...
DECLARATION-PART	096	.......11...................1111......................................
NON-PROCEDURE-DECLARATION	097	........1...................1111......................................
PROPER-INCOMPLETE-DECLARATION-PART	098	.........1.............................1..............................
INTEGER-INCOMPLETE-DECLARATION-PART	099	..........1.............................1.............................
INTEGER-PROCEDURE-BODY	100	......1....1....................1....1................................
SHORT-REAL-INCOMPLETE-DECLARATION-PART	101	............1............................1............................
SHORT-REAL-PROCEDURE-BODY	102	......1......1...1........
LONG-REAL-INCOMPLETE-DECLARATION-PART	103	..............1...........................1...........................
LONG-REAL-PROCEDURE-BODY	104	......1........1...1......
SHORT-COMPLEX-INCOMPLETE-DECLARATION-PART	105	................1..........................1..........................
SHORT-COMPLEX-PROCEDURE-BODY	106	......1..........1...1....
LONG-COMPLEX-INCOMPLETE-DECLARATION-PART	107	..................1.........................1.........................
LONG-COMPLEX-PROCEDURE-BODY	108	......1............1...1..
LOGICAL-INCOMPLETE-DECLARATION-PART	109	....................1........................1........................
LOGICAL-PROCEDURE-BODY	110	......1..............1...1
BIT-INCOMPLETE-DECLARATION-PART	111	......................1.......................1.......................
BIT-PROCEDURE-BODY	112	......1................1..
STRING-INCOMPLETE-DECLARATION-PART	113	........................1......................1......................
STRING-PROCEDURE-BODY	114	......1..................1..
REFERENCE-INCOMPLETE-DECLARATION-PART	115	..........................1.....................1.....................
REFERENCE-PROCEDURE-BODY	116	......1....................1.....1....................................
SIMPLE-VARIABLE-DECLARATION	117	............................1..1......................................
ARRAY-DECLARATION	118	.............................1..
RECORD-CLASS-DECLARATION	119	..............................1.......................................
SIMPLE-TYPE-FOLLOWED-BY-IDENTIFIER	120	...............................1......................................
INTEGER-NUMBER	121	................................1.....................................
RECORD-CLASS-IDENTIFIER	122	......1..........................1....................................
INCOMPLETE-ARRAY-DECLARATION	123	..................................1...................................
BOUND-PAIR	124	...................................1..................................
SIMPLE-TYPE	125	....................................1.............111111111...........
INTEGER-EXPRESSION	126	......1.........................1....1................................
FIELD-LIST	127	............................1..1......1...............................
PROPER-PROCEDURE-HEADING	128	....1..............................
INTEGER-PROCEDURE-HEADING	129	....1.............................
SHORT-REAL-PROCEDURE-HEADING	130	....1............................
LONG-REAL-PROCEDURE-HEADING	131	....1...........................
SHORT-COMPLEX-PROCEDURE-HEADING	132	....1..........................
LONG-COMPLEX-PROCEDURE-HEADING	133	....1.........................
LOGICAL-PROCEDURE-HEADING	134	....1........................
BIT-PROCEDURE-HEADING	135	....1.......................
STRING-PROCEDURE-HEADING	136	....1......................
REFERENCE-PROCEDURE-HEADING	137	....1.....................
PROCEDURE-HEADING	138	......1..1....................
SIMPLE-INTEGER-TYPE	139	....1...................
SIMPLE-SHORT-REAL-TYPE	140	....1..................
SIMPLE-LONG-REAL-TYPE	141	....1.................
SIMPLE-SHORT-COMPLEX-TYPE	142	....1................
SIMPLE-LONG-COMPLEX-TYPE	143	....1...............
SIMPLE-LOGICAL-TYPE	144	....1..............
SIMPLE-BIT-TYPE	145	....1.............
SIMPLE-STRING-TYPE	146	....1............
SIMPLE-REFERENCE-TYPE	147	....1...........
FORMAL-PARAMETER-LIST	148	...............................1...........................1..........
INTEGER-PROCEDURE-BODY-INTERIOR	149	......1.........................1....1......................1.........
SHORT-REAL-EXPRESSION	150	......1..1........
SHORT-REAL-PROCEDURE-BODY-INTERIOR	151	......1..11.......
LONG-REAL-EXPRESSION	152	......1..1......
LONG-REAL-PROCEDURE-BODY-INTERIOR	153	......1..11.....
SHORT-COMPLEX-EXPRESSION	154	......1..1....
SHORT-COMPLEX-PROCEDURE-BODY-INTERIOR	155	......1..11...
LONG-COMPLEX-EXPRESSION	156	......1..1..
LONG-COMPLEX-PROCEDURE-BODY-INTERIOR	157	......1..11.
LOGICAL-EXPRESSION	158	......1..1
LOGICAL-PROCEDURE-BODY-INTERIOR	159	......1..1
BIT-EXPRESSION	160	......1...
BIT-PROCEDURE-BODY-INTERIOR	161	......1...
STRING-EXPRESSION	162	......1...
STRING-PROCEDURE-BODY-INTERIOR	163	......1...

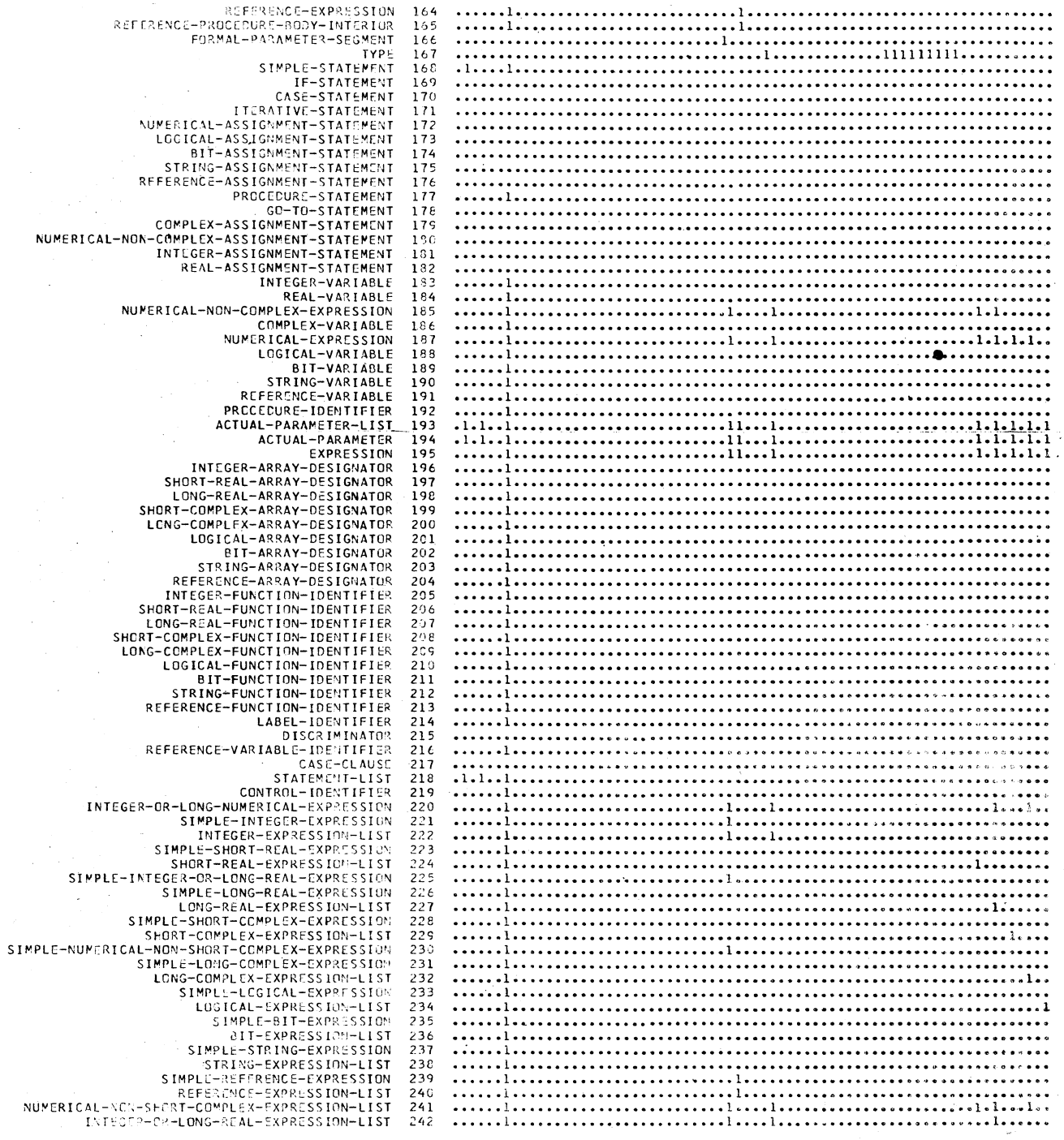
PL nº 28 GRAMMAIRE D'UN NOUVEL ALGOL

RELATION ⇒ ENTRE LES OPERANDES (SUITE)

```
                                  SIMPLE-NUMERICAL-EXPRESSION 243 .....1.....1.....
                SIMPLE-NUMERICAL-NON-COMPLEX-EXPRESSION 244 .....1.....1.....
                                        INTEGER-TERM 245 .....1.....1.....
                                     SHORT-REAL-TERM 246 .....1.....
                          NUMERICAL-NON-COMPLEX-TERM 247 .....1.....1.....
                                     LONG-REAL-TERM 248 .....1.....
                                  SHORT-COMPLEX-TERM 249 .....1.....
                    NUMERICAL-NON-SHORT-COMPLEX-TERM 250 .....1.....1.....
                                  LONG-COMPLEX-TERM 251 .....1.....
                       INTEGER-OR-LONG-NUMERICAL-TERM 252 .....1.....1.....
                                        REAL-TERM 253 .....1.....
                                     COMPLEX-TERM 254 .....1.....
                                   INTEGER-FACTOR 255 .....1.....1.....
                                SHORT-REAL-FACTOR 256 .....1.....
                       INTEGER-OR-LONG-REAL-FACTOR 257 .....1.....1.....
                                   LONG-REAL-FACTOR 258 .....1.....
                                        REAL-FACTOR 259 .....1.....
                                  SHORT-COMPLEX-FACTOR 260 .....1.....
                                    NUMERICAL-FACTOR 261 .....1.....1.....
                                  LONG-COMPLEX-FACTOR 262 .....1.....
                                     COMPLEX-FACTOR 263 .....1.....
                                 INTEGER-SECONDARY 264 .....1.....1.....
                              SHORT-REAL-SECONDARY 265 .....1.....
                                 LONG-REAL-SECONDARY 266 .....1.....
                            SHORT-COMPLEX-SECONDARY 267 .....1.....
                              LONG-COMPLEX-SECONDARY 268 .....1.....
                                   INTEGER-PRIMARY 269 .....1.....
                                SHORT-REAL-PRIMARY 270 .....1.....
                                  SHORT-REAL-NUMBER 271 .....
                              SHORT-COMPLEX-PRIMARY 272 .....1.....
                                 LONG-REAL-PRIMARY 273 .....1.....
                                   LONG-REAL-NUMBER 274 .....
                               LONG-COMPLEX-PRIMARY 275 .....1.....
                                 SHORT-COMPLEX-NUMBER 276 .....
                                  LONG-COMPLEX-NUMBER 277 .....
                                 SHORT-REAL-VARIABLE 278 .....1.....
                                   LONG-REAL-VARIABLE 279 .....1.....
                              SHORT-COMPLEX-VARIABLE 280 .....1.....
                                LONG-COMPLEX-VARIABLE 281 .....1.....
                                      LOGICAL-TERM 282 .....1.....
                                          RELATION 283 .....
                                    LOGICAL-FACTOR 284 .....1.....
                                LOGICAL-SECONDARY 285 .....1.....
                                  LOGICAL-PRIMARY 286 .....1.....
                                         BIT-TERM 287 .....1.....
                                       BIT-FACTOR 288 .....1.....
                                    BIT-SECONDARY 289 .....1.....
                                      BIT-PRIMARY 290 .....1.....
                                     BIT-SEQUENCE 291 .....
                                   STRING-PRIMARY 292 .....1.....
                                    EXPRESSION-LIST 293 .....1.....11...1.....1.1.1.1.1
                      INTEGER-VARIABLE-IDENTIFIER 294 .....1.....
                        INTEGER-FIELD-IDENTIFIER 295 .....1.....
                   SHORT-REAL-VARIABLE-IDENTIFIER 296 .....1.....
                     SHORT-REAL-FIELD-IDENTIFIER 297 .....1.....
                     LONG-REAL-VARIABLE-IDENTIFIER 298 .....1.....
                       LONG-REAL-FIELD-IDENTIFIER 299 .....1.....
                SHORT-COMPLEX-VARIABLE-IDENTIFIER 300 .....1.....
                  SHORT-COMPLEX-FIELD-IDENTIFIER 301 .....1.....
                LONG-COMPLEX-VARIABLE-IDENTIFIER 302 .....1.....
                  LONG-COMPLEX-FIELD-IDENTIFIER 303 .....1.....
                    LOGICAL-VARIABLE-IDENTIFIER 304 .....1.....
                      LOGICAL-FIELD-IDENTIFIER 305 .....1.....
                       BIT-VARIABLE-IDENTIFIER 306 .....1.....
                         BIT-FIELD-IDENTIFIER 307 .....1.....
                     STRING-VARIABLE-IDENTIFIER 308 .....1.....
                       STRING-FIELD-IDENTIFIER 309 .....1.....
                     REFERENCE-FIELD-IDENTIFIER 310 .....1.....
                      INTEGER-ARRAY-IDENTIFIER 311 .....1.....
                   SHORT-REAL-ARRAY-IDENTIFIER 312 .....1.....
                     LONG-REAL-ARRAY-IDENTIFIER 313 .....1.....
                SHORT-COMPLEX-ARRAY-IDENTIFIER 314 .....1.....
                  LONG-COMPLEX-ARRAY-IDENTIFIER 315 .....1.....
                      LOGICAL-ARRAY-IDENTIFIER 316 .....1.....
                        BIT-ARRAY-IDENTIFIER 317 .....1.....
                     STRING-ARRAY-IDENTIFIER 318 .....1.....
                   REFERENCE-ARRAY-IDENTIFIER 319 .....1.....
                                  UNSCALED-REAL 320 .....
                          ...-SHORT-REAL-NUMBER 321 .....1.....
```

PL nº 29 GRAMMAIRE D'UN NOUVEL ALGOL

RELATION $\Rightarrow^*$ ENTRE LES OPERANDES (SUITE)



PL nº 30 GRAMMAIRE D'UN NOUVEL ALGOL

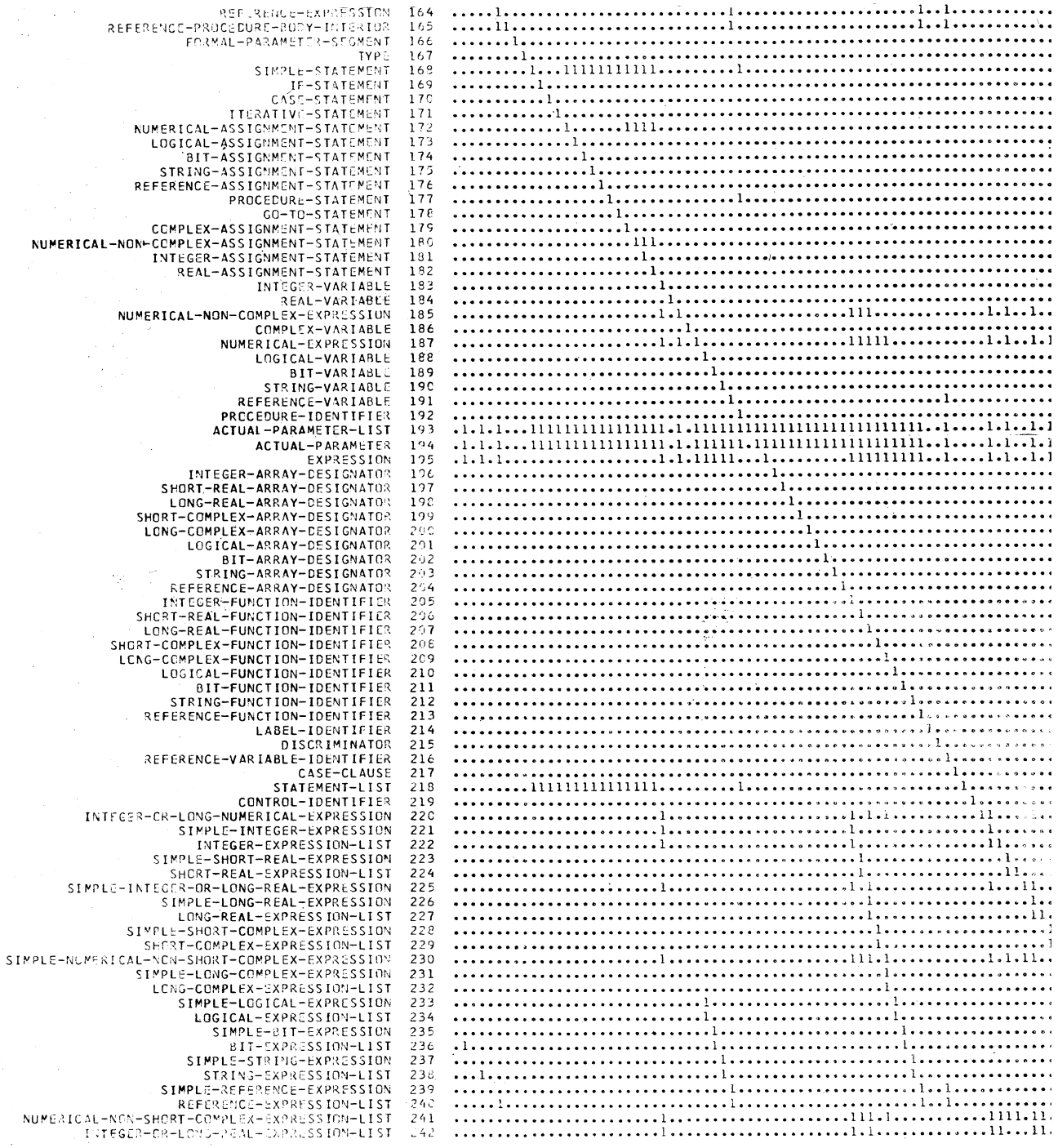
RELATION $\Rightarrow^*$ ENTRE LES OPERANDES (SUITE)

```
                                                  1111111111111111111111111111111111111111111122222222222222222222222222222222
                                                  5666666666677777777778888888889999999999000000000011111111111222222222
                                                  9012345678901234567890123456789012345678901234567890123456789012345678

                                  PROGRAM  089 ................................................................................
                                    BLOCK  090 ................................................................................
                            BLOCK-INTERIOR  091 .........111111111111111.........1...........................................
                                 STATEMENT  092 .........111111111111111.........1...........................................
                  BLOCK-INTERIOR-OR-DECLARATION-PART  093 .........111111111111111.........1...........................................
                       INCOMPLETE-BLOCK-INTERIOR  094 ................................................................................
                                      IDENTIFIER  095 ................................................................................
                             DECLARATION-PART  096 ................................................................................
                        NON-PROCEDURE-DECLARATION  097 ................................................................................
                  PROPER-INCOMPLETE-DECLARATION-PART  098 ................................................................................
                 INTEGER-INCOMPLETE-DECLARATION-PART  099 ................................................................................
                          INTEGER-PROCEDURE-BODY  100 ..........................1.....................1.................1.......
               SHORT-REAL-INCOMPLETE-DECLARATION-PART  101 ................................................................................
                        SHORT-REAL-PROCEDURE-BODY  102 ..................................................1.................1.....
                LONG-REAL-INCOMPLETE-DECLARATION-PART  103 ................................................................................
                           LONG-REAL-PROCEDURE-BODY  104 ..................................................1.................1..
            SHORT-COMPLEX-INCOMPLETE-DECLARATION-PART  105 ................................................................................
                      SHORT-COMPLEX-PROCEDURE-BODY  106 ..................................................1.................1
             LONG-COMPLEX-INCOMPLETE-DECLARATION-PART  107 ................................................................................
                        LONG-COMPLEX-PROCEDURE-BODY  108 ..................................................1.............................
                LOGICAL-INCOMPLETE-DECLARATION-PART  109 ................................................................................
                           LOGICAL-PROCEDURE-BODY  110 ..........................1.....................1.................1.......
                   BIT-INCOMPLETE-DECLARATION-PART  111 ................................................................................
                              BIT-PROCEDURE-BODY  112 .1..........................1.....................1.................1.......
                STRING-INCOMPLETE-DECLARATION-PART  113 ................................................................................
                           STRING-PROCEDURE-BODY  114 ...1.....................1.....................1.................1.......
            REFERENCE-INCOMPLETE-DECLARATION-PART  115 ................................................................................
                         REFERENCE-PROCEDURE-BODY  116 .....1.....................1.....................1..1.............
                       SIMPLE-VARIABLE-DECLARATION  117 ................................................................................
                                 ARRAY-DECLARATION  118 ................................................................................
                          RECORD-CLASS-DECLARATION  119 ................................................................................
              SIMPLE-TYPE-FOLLOWED-BY-IDENTIFIER  120 ................................................................................
                                  INTEGER-NUMBER  121 ................................................................................
                       RECORD-CLASS-IDENTIFIER  122 ................................................................................
                   INCOMPLETE-ARRAY-DECLARATION  123 ................................................................................
                                      BOUND-PAIR  124 ................................................................................
                                      SIMPLE-TYPE  125 ................................................................................
                               INTEGER-EXPRESSION  126 ..........................1.....................1.................1.......
                                      FIELD-LIST  127 ................................................................................
                         PROPER-PROCEDURE-HEADING  128 ................................................................................
                        INTEGER-PROCEDURE-HEADING  129 ................................................................................
                      SHORT-REAL-PROCEDURE-HEADING  130 ................................................................................
                       LONG-REAL-PROCEDURE-HEADING  131 ................................................................................
                    SHORT-COMPLEX-PROCEDURE-HEADING  132 ................................................................................
                     LONG-COMPLEX-PROCEDURE-HEADING  133 ................................................................................
                        LOGICAL-PROCEDURE-HEADING  134 ................................................................................
                          BIT-PROCEDURE-HEADING  135 ................................................................................
                       STRING-PROCEDURE-HEADING  136 ................................................................................
                    REFERENCE-PROCEDURE-HEADING  137 ................................................................................
                              PROCEDURE-HEADING  138 ................................................................................
                           SIMPLE-INTEGER-TYPE  139 ................................................................................
                      SIMPLE-SHORT-REAL-TYPE  140 ................................................................................
                       SIMPLE-LONG-REAL-TYPE  141 ................................................................................
                    SIMPLE-SHORT-COMPLEX-TYPE  142 ................................................................................
                     SIMPLE-LONG-COMPLEX-TYPE  143 ................................................................................
                        SIMPLE-LOGICAL-TYPE  144 ................................................................................
                             SIMPLE-BIT-TYPE  145 ................................................................................
                          SIMPLE-STRING-TYPE  146 ................................................................................
                       SIMPLE-REFERENCE-TYPE  147 ................................................................................
                          FORMAL-PARAMETER-LIST  148 .......1........................................................................
               INTEGER-PROCEDURE-BODY-INTERIOR  149 ..........................1.....................1.................1.......
                      SHORT-REAL-EXPRESSION  150 ..................................................1.................1.....
            SHORT-REAL-PROCEDURE-BODY-INTERIOR  151 ..................................................1.................1.....
                       LONG-REAL-EXPRESSION  152 ..................................................1.................1..
             LONG-REAL-PROCEDURE-BODY-INTERIOR  153 ..................................................1.................1..
                    SHORT-COMPLEX-EXPRESSION  154 ..................................................1.................1
          SHORT-COMPLEX-PROCEDURE-BODY-INTERIOR  155 ..................................................1.................1
                     LONG-COMPLEX-EXPRESSION  156 ..................................................1.................
           LONG-COMPLEX-PROCEDURE-BODY-INTERIOR  157 ..................................................1.................
                        LOGICAL-EXPRESSION  158 ..........................1.....................1.................
             LOGICAL-PROCEDURE-BODY-INTERIOR  159 1.........................1.....................1.................
                          BIT-EXPRESSION  160 .1..........................1.....................1.................
               BIT-PROCEDURE-BODY-INTERIOR  161 .11.........................1.....................1.................
                       STRING-EXPRESSION  162 ...1.....................1.....................1.................
            STRING-PROCEDURE-BODY-INTERIOR  163 ...11....................1.....................1.................
```

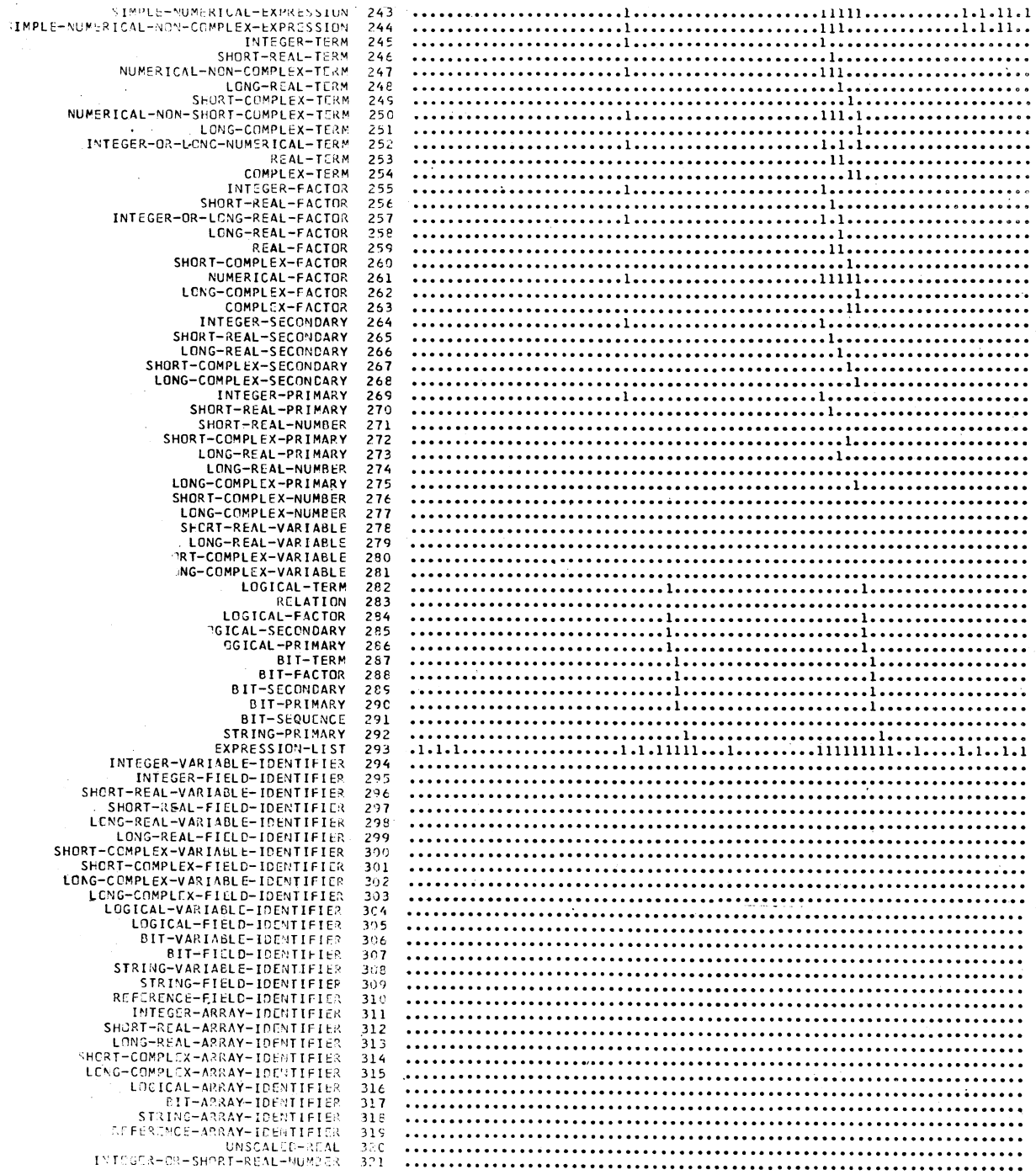
PL nº 31 GRAMMAIRE D'UN NOUVEL ALGOL

RELATION $\Rightarrow^*$ ENTRE LES OPERANDES (SUITE)



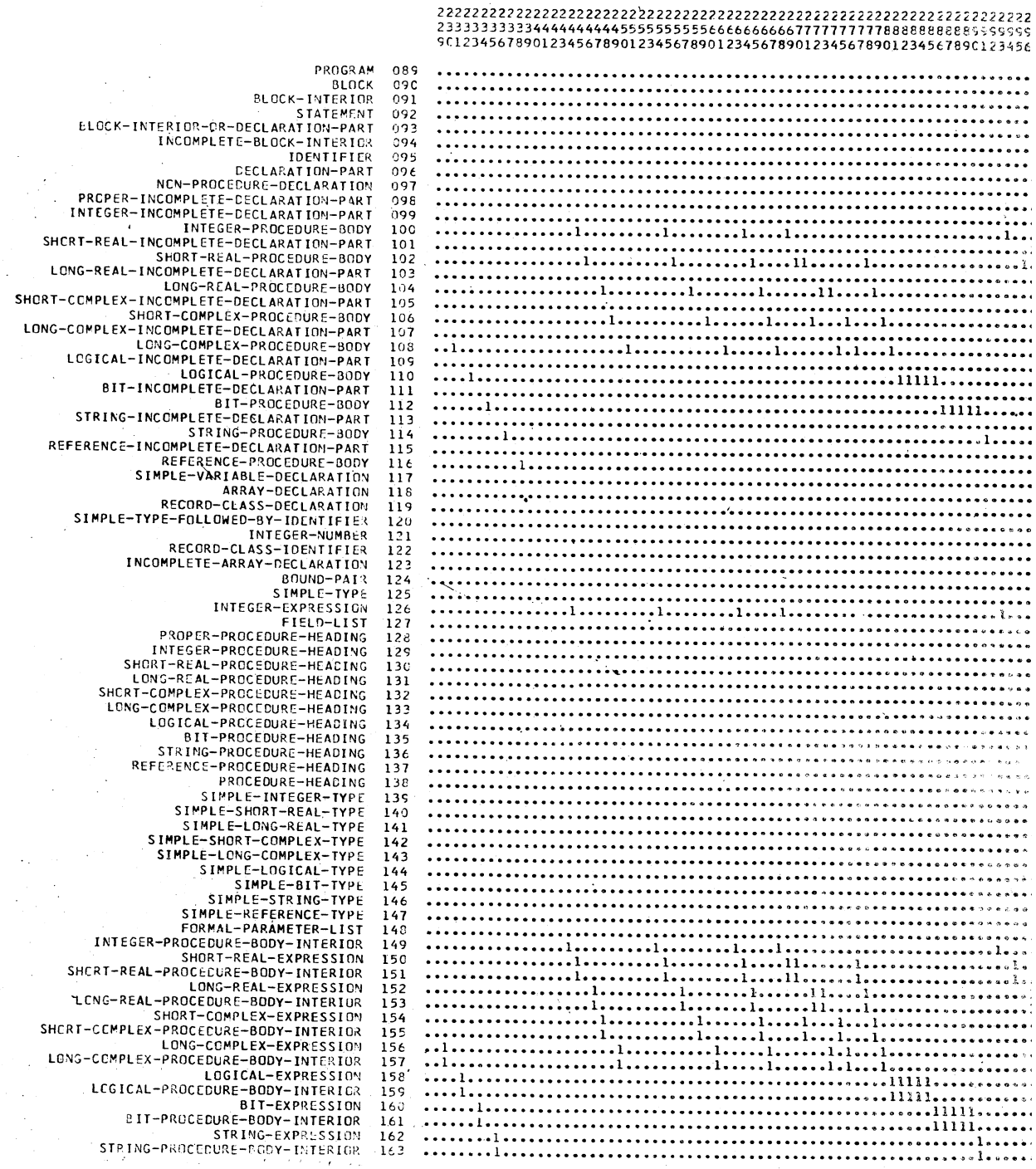
PL nº 32 GRAMMAIRE D'UN NOUVEL ALGOL

RELATION $\Rightarrow^*$ ENTRE LES OPERANDES (SUITE)



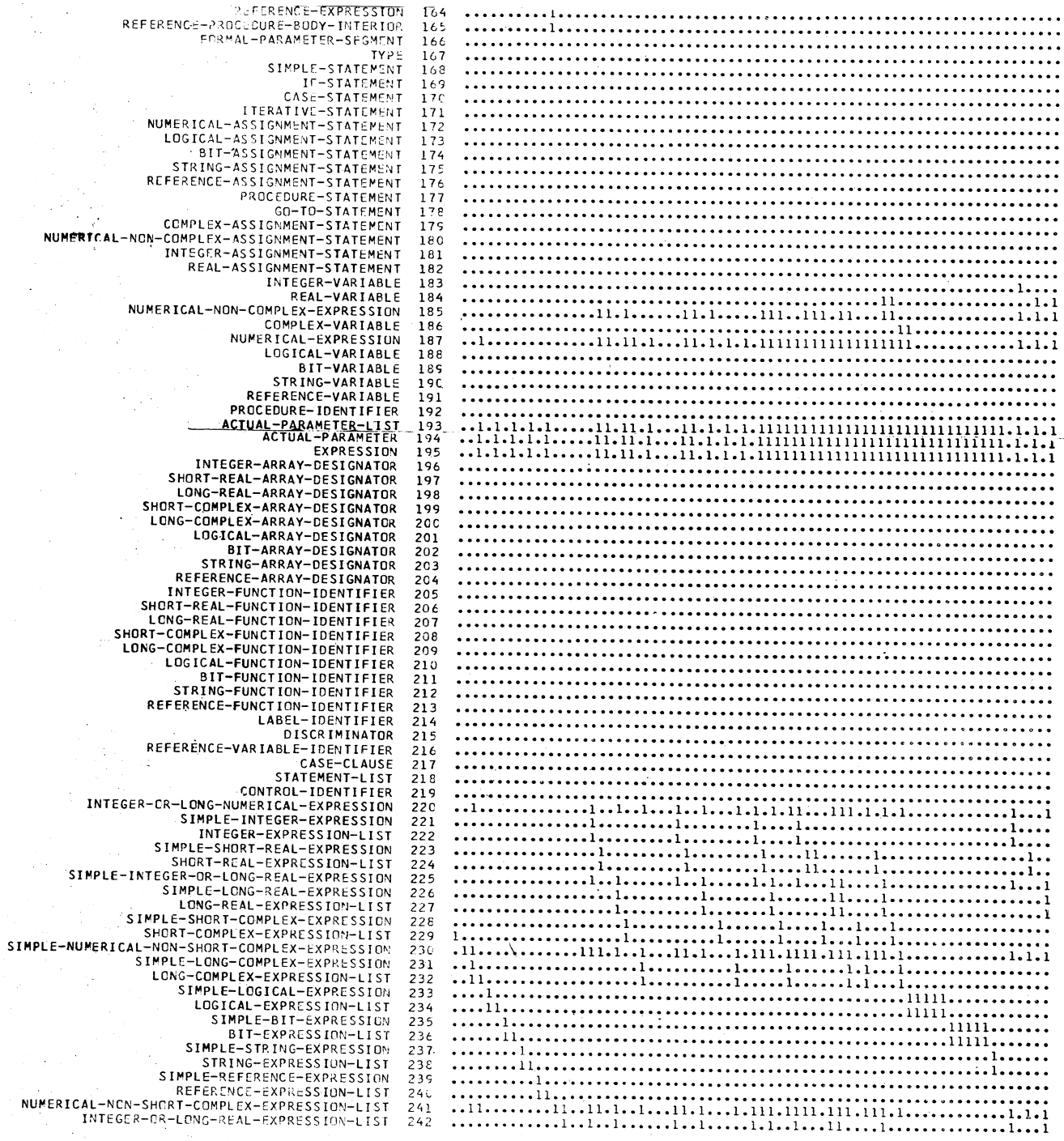
PL nº 33 GRAMMAIRE D'UN NOUVEL ALGOL

RELATION $\Rightarrow^*$ ENTRE LES OPERANDES (SUITE)



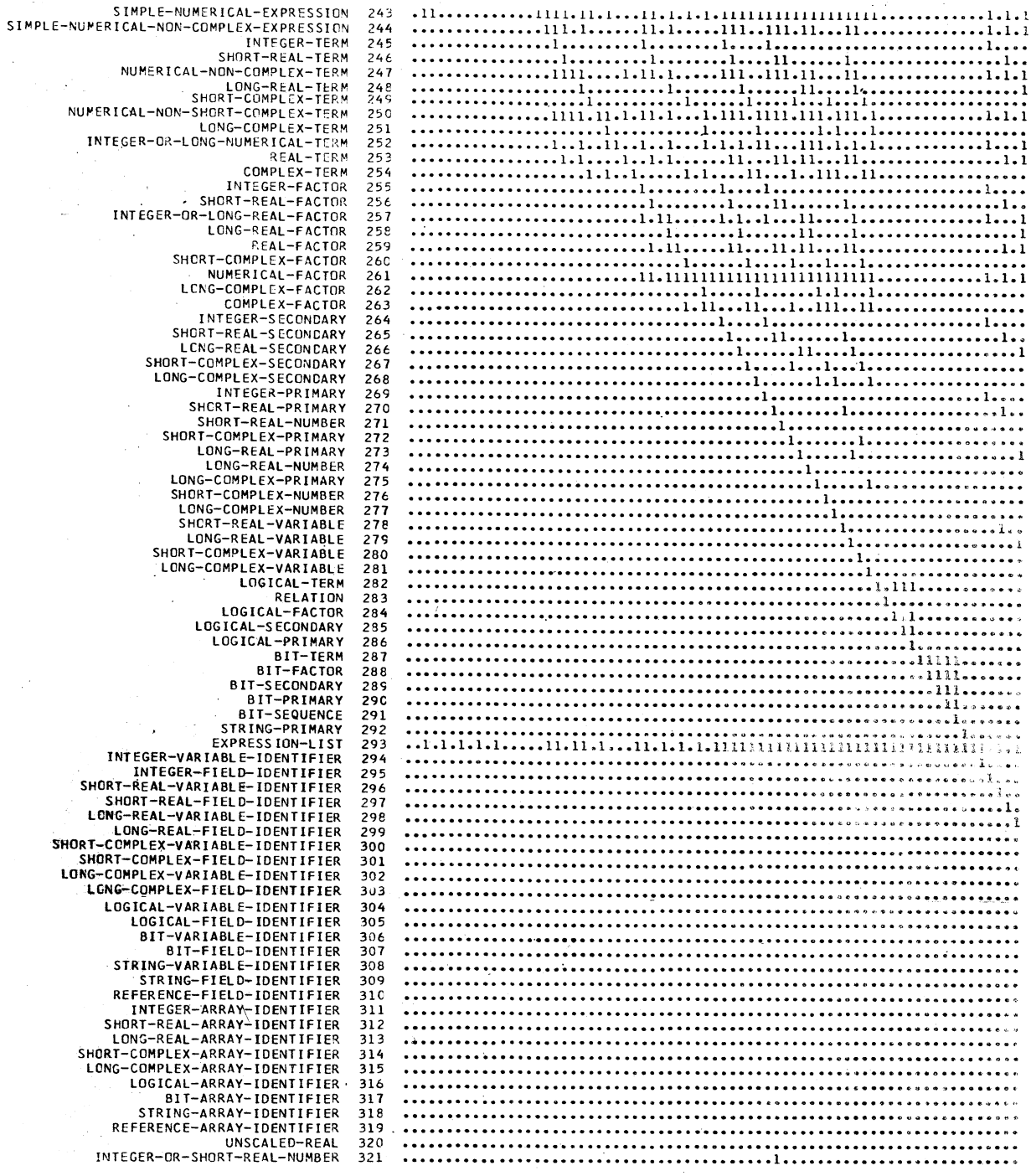
PL nº 34 GRAMMAIRE D'UN NOUVEL ALGOL

RELATION $\Rightarrow^*$ ENTRE LES OPERANDES (SUITE)



PL nº 35 GRAMMAIRE D'UN NOUVEL ALGOL

RELATION $\Rightarrow^*$ ENTRE LES OPERANDES (SUITE)



PL nº 36 GRAMMAIRE D'UN NOUVEL ALGOL

RELATION $\Rightarrow^*$ ENTRE LES OPERANDES (SI

```
                                                  2333333333333333333333
                                                  9C0CCCCCCCC0111111111122
                                                  9C1234567890123456789O1
                                  PROGRAM  089 ......................
                                    BLOCK  090 ......................
                             BLOCK-INTERIOR  091 ......................
                                  STATEMENT  092 ......................
              BLOCK-INTERIOR-OR-DECLARATION-PART  093 ......................
                 INCOMPLETE-BLOCK-INTERIOR  094 ......................
                                 IDENTIFIER  095 ......................
                           DECLARATION-PART  096 ......................
                      NON-PROCEDURE-DECLARATION  097 ......................
                  PROPER-INCOMPLETE-DECLARATION-PART  098 ......................
               INTEGER-INCOMPLETE-DECLARATION-PART  099 ......................
                        INTEGER-PROCEDURE-BODY  100 ......................
          SHORT-REAL-INCOMPLETE-DECLARATION-PART  101 ......................
                    SHORT-REAL-PROCEDURE-BODY  102 ......................1.
            LONG-REAL-INCOMPLETE-DECLARATION-PART  103 ......................
                     LONG-REAL-PROCEDURE-BODY  104 ......................
         SHORT-COMPLEX-INCOMPLETE-DECLARATION-PART  105 ......................
                  SHORT-COMPLEX-PROCEDURE-BODY  106 .1....................
        LONG-COMPLEX-INCOMPLETE-DECLARATION-PART  107 ......................
                  LONG-COMPLEX-PROCEDURE-BODY  108 ...1..................
           LOGICAL-INCOMPLETE-DECLARATION-PART  109 ......................
                     LOGICAL-PROCEDURE-BODY  110 .....1.................
              BIT-INCOMPLETE-DECLARATION-PART  111 ......................
                        BIT-PROCEDURE-BODY  112 ........1................
            STRING-INCOMPLETE-DECLARATION-PART  113 ......................
                     STRING-PROCEDURE-BODY  114 .........1.............
         REFERENCE-INCOMPLETE-DECLARATION-PART  115 ......................
                  REFERENCE-PROCEDURE-BODY  116 ......................
                SIMPLE-VARIABLE-DECLARATION  117 ......................
                           ARRAY-DECLARATION  118 ......................
                    RECORD-CLASS-DECLARATION  119 ......................
          SIMPLE-TYPE-FOLLOWED-BY-IDENTIFIER  120 ......................
                          INTEGER-NUMBER  121 ......................
                    RECORD-CLASS-IDENTIFIER  122 ......................
               INCOMPLETE-ARRAY-DECLARATION  123 ......................
                              BOUND-PAIR  124 ......................
                             SIMPLE-TYPE  125 ......................
                       INTEGER-EXPRESSION  126 ......................
                              FIELD-LIST  127 ......................
                   PROPER-PROCEDURE-HEADING  128 ......................
                  INTEGER-PROCEDURE-HEADING  129 ......................
                SHORT-REAL-PROCEDURE-HEADING  130 ......................
                 LONG-REAL-PROCEDURE-HEADING  131 ......................
              SHORT-COMPLEX-PROCEDURE-HEADING  132 ......................
               LONG-COMPLEX-PROCEDURE-HEADING  133 ......................
                 LOGICAL-PROCEDURE-HEADING  134 ......................
                    BIT-PROCEDURE-HEADING  135 ......................
                 STRING-PROCEDURE-HEADING  136 ......................
              REFERENCE-PROCEDURE-HEADING  137 ......................
                       PROCEDURE-HEADING  138 ......................
                     SIMPLE-INTEGER-TYPE  139 ......................
                  SIMPLE-SHORT-REAL-TYPE  140 ......................
                   SIMPLE-LONG-REAL-TYPE  141 ......................
                 SIMPLE-SHORT-COMPLEX-TYPE  142 ......................
                  SIMPLE-LONG-COMPLEX-TYPE  143 ......................
                    SIMPLE-LOGICAL-TYPE  144 ......................
                       SIMPLE-BIT-TYPE  145 ......................
                     SIMPLE-STRING-TYPE  146 ......................
                  SIMPLE-REFERENCE-TYPE  147 ......................
                       FORMAL-PARAMETER-LIST  148 ......................
            INTEGER-PROCEDURE-BODY-INTERIOR  149 ......................
                     SHORT-REAL-EXPRESSION  150 ......................1.
          SHORT-REAL-PROCEDURE-BODY-INTERIOR  151 ......................1.
                   LONG-REAL-EXPRESSION  152 ......................
        LONG-REAL-PROCEDURE-BODY-INTERIOR  153 ......................
               SHORT-COMPLEX-EXPRESSION  154 .1....................
      SHORT-COMPLEX-PROCEDURE-BODY-INTERIOR  155 .1....................
                LONG-COMPLEX-EXPRESSION  156 ...1..................
        LONG-COMPLEX-PROCEDURE-BODY-INTERIOR  157 ...1..................
                     LOGICAL-EXPRESSION  158 .....1.................
           LOGICAL-PROCEDURE-BODY-INTERIOR  159 .....1.................
                       BIT-EXPRESSION  160 .......1...............
                BIT-PROCEDURE-BODY-INTERIOR  161 .......1...............
                     STRING-EXPRESSION  162 .........1.............
                STRING-PROCEDURE-BODY-INTERIOR  163 .........1.............
```

PL nº 37 GRAMMAIRE D'UN NOUVEL ALGOL

RELATION $\Rightarrow^*$ ENTRE LES OPERANDES (SUITE)

REFERENCE-EXPRESSION	164	.....
REFERENCE-PROCEDURE-BODY-INTERIOR	165	.....
FORMAL-PARAMETER-SEGMENT	166	.....
TYPE	167	.....
SIMPLE-STATEMENT	168	.....
IF-STATEMENT	169	.....
CASE-STATEMENT	170	.....
ITERATIVE-STATEMENT	171	.....
NUMERICAL-ASSIGNMENT-STATEMENT	172	.....
LOGICAL-ASSIGNMENT-STATEMENT	173	.....
BIT-ASSIGNMENT-STATEMENT	174	.....
STRING-ASSIGNMENT-STATEMENT	175	.....
REFERENCE-ASSIGNMENT-STATEMENT	176	.....
PROCEDURE-STATEMENT	177	.....
GO-TO-STATEMENT	178	.....
COMPLEX-ASSIGNMENT-STATEMENT	179	.....
NUMERICAL-NON-COMPLEX-ASSIGNMENT-STATEMENT	180	.....
INTEGER-ASSIGNMENT-STATEMENT	181	.....
REAL-ASSIGNMENT-STATEMENT	182	.....
INTEGER-VARIABLE	183	.....
REAL-VARIABLE	184	.....
NUMERICAL-NON-COMPLEX-EXPRESSION	185	.....1.
COMPLEX-VARIABLE	186	.1.1.....
NUMERICAL-EXPRESSION	187	.1.1.....1.
LOGICAL-VARIABLE	188	.....1.....
BIT-VARIABLE	189	.....1.....
STRING-VARIABLE	190	.....1.....
REFERENCE-VARIABLE	191	.....
PROCEDURE-IDENTIFIER	192	.....
ACTUAL-PARAMETER-LIST	193	.1.1.1.1.1..1111111111.
ACTUAL-PARAMETER	194	.1.1.1.1.1..1111111111.
EXPRESSION	195	.1.1.1.1.1.....1.
INTEGER-ARRAY-DESIGNATOR	196	.....1.....
SHORT-REAL-ARRAY-DESIGNATOR	197	.....1.....
LONG-REAL-ARRAY-DESIGNATOR	198	.....1.....
SHORT-COMPLEX-ARRAY-DESIGNATOR	199	.....1.....
LONG-COMPLEX-ARRAY-DESIGNATOR	200	.....1.....
LOGICAL-ARRAY-DESIGNATOR	201	.....1.....
BIT-ARRAY-DESIGNATOR	202	.....1.....
STRING-ARRAY-DESIGNATOR	203	.....1...
REFERENCE-ARRAY-DESIGNATOR	204	.....1..
INTEGER-FUNCTION-IDENTIFIER	205	.....
SHORT-REAL-FUNCTION-IDENTIFIER	206	.....
LONG-REAL-FUNCTION-IDENTIFIER	207	.....
SHORT-COMPLEX-FUNCTION-IDENTIFIER	208	.....
LONG-COMPLEX-FUNCTION-IDENTIFIER	209	.....
LOGICAL-FUNCTION-IDENTIFIER	210	.....
BIT-FUNCTION-IDENTIFIER	211	.....
STRING-FUNCTION-IDENTIFIER	212	.....
REFERENCE-FUNCTION-IDENTIFIER	213	.....
LABEL-IDENTIFIER	214	.....
DISCRIMINATOR	215	.....
REFERENCE-VARIABLE-IDENTIFIER	216	.....
CASE-CLAUSE	217	.....
STATEMENT-LIST	218	.....
CONTROL-IDENTIFIER	219	.....
INTEGER-OR-LONG-NUMERICAL-EXPRESSION	220	...1.....
SIMPLE-INTEGER-EXPRESSION	221	.....
INTEGER-EXPRESSION-LIST	222	.....
SIMPLE-SHORT-REAL-EXPRESSION	223	.....1.
SHORT-REAL-EXPRESSION-LIST	224	.....1.
SIMPLE-INTEGER-OR-LONG-REAL-EXPRESSION	225	.....
SIMPLE-LONG-REAL-EXPRESSION	226	.....
LONG-REAL-EXPRESSION-LIST	227	.....
SIMPLE-SHORT-COMPLEX-EXPRESSION	228	.1.....
SHORT-COMPLEX-EXPRESSION-LIST	229	.1.....
SIMPLE-NUMERICAL-NON-SHORT-COMPLEX-EXPRESSION	230	...1.....1.
SIMPLE-LONG-COMPLEX-EXPRESSION	231	...1.....
LONG-COMPLEX-EXPRESSION-LIST	232	...1.....
SIMPLE-LOGICAL-EXPRESSION	233	.....1.....
LOGICAL-EXPRESSION-LIST	234	.....1.....
SIMPLE-BIT-EXPRESSION	235	.....1.....
BIT-EXPRESSION-LIST	236	.....1.....
SIMPLE-STRING-EXPRESSION	237	.....1.....
STRING-EXPRESSION-LIST	238	.....1.....
SIMPLE-REFERENCE-EXPRESSION	239	.....
REFERENCE-EXPRESSION-LIST	240	.....
NUMERICAL-NON-SHORT-COMPLEX-EXPRESSION-LIST	241	...1.....1.
INTEGER-OR-LONG-REAL-EXPRESSION-LIST	242	.....

PL nº 38 GRAMMAIRE D'UN NOUVEL ALGOL

RELATION $\Rightarrow^*$ ENTRE LES OPERANDES (SUITE)

SIMPLE-NUMERICAL-EXPRESSION	243	.1.1.....1.
SIMPLE-NUMERICAL-NON-COMPLEX-EXPRESSION	244	.....1.
INTEGER-TERM	245	.....
SHORT-REAL-TERM	246	.....1.
NUMERICAL-NON-COMPLEX-TERM	247	.....1.
LONG-REAL-TERM	248	.....
SHORT-COMPLEX-TERM	249	.1.....
NUMERICAL-NON-SHORT-COMPLEX-TERM	250	...1.....1.
LONG-COMPLEX-TERM	251	...1.....
INTEGER-OR-LONG-NUMERICAL-TERM	252	...1.....
REAL-TERM	253	.....1.
COMPLEX-TERM	254	.1.1.....
INTEGER-FACTOR	255	.....
SHORT-REAL-FACTOR	256	.....1.
INTEGER-OR-LONG-REAL-FACTOR	257	.....
LONG-REAL-FACTOR	258	.....
REAL-FACTOR	259	.....1.
SHORT-COMPLEX-FACTOR	260	.1.....
NUMERICAL-FACTOR	261	.1.1.....1.
LONG-COMPLEX-FACTOR	262	...1.....
COMPLEX-FACTOR	263	.1.1.....
INTEGER-SECONDARY	264	.....
SHORT-REAL-SECONDARY	265	.....1.
LONG-REAL-SECONDARY	266	.....
SHORT-COMPLEX-SECONDARY	267	.1.....
LONG-COMPLEX-SECONDARY	268	...1.....
INTEGER-PRIMARY	269	.....
SHORT-REAL-PRIMARY	270	.....
SHORT-REAL-NUMBER	271	.....1.
SHORT-COMPLEX-PRIMARY	272	.1.....
LONG-REAL-PRIMARY	273	.....
LONG-REAL-NUMBER	274	.....
LONG-COMPLEX-PRIMARY	275	...1.....
SHORT-COMPLEX-NUMBER	276	.....
LONG-COMPLEX-NUMBER	277	.....
SHORT-REAL-VARIABLE	278	.....
LONG-REAL-VARIABLE	279	.....
SHORT-COMPLEX-VARIABLE	280	.1.....
LONG-COMPLEX-VARIABLE	281	...1.....
LOGICAL-TERM	282	.....1.....
RELATION	283	.....
LOGICAL-FACTOR	284	.....1.....
LOGICAL-SECONDARY	285	.....1.....
LOGICAL-PRIMARY	286	.....1.....
BIT-TERM	287	.....1.....
BIT-FACTOR	288	.....1.....
BIT-SECONDARY	289	.....1.....
BIT-PRIMARY	290	.....1.....
BIT-SEQUENCE	291	.....
STRING-PRIMARY	292	.....1.....
EXPRESSION-LIST	293	.1.1.1.1.1.....1.
INTEGER-VARIABLE-IDENTIFIER	294	.....
INTEGER-FIELD-IDENTIFIER	295	.....
SHORT-REAL-VARIABLE-IDENTIFIER	296	.....
SHORT-REAL-FIELD-IDENTIFIER	297	.....
LONG-REAL-VARIABLE-IDENTIFIER	298	.....
LONG-REAL-FIELD-IDENTIFIER	299	1.....
SHORT-COMPLEX-VARIABLE-IDENTIFIER	300	.1.....
SHORT-COMPLEX-FIELD-IDENTIFIER	301	..1.....
LONG-COMPLEX-VARIABLE-IDENTIFIER	302	...1.....
LONG-COMPLEX-FIELD-IDENTIFIER	303	....1.....
LOGICAL-VARIABLE-IDENTIFIER	304	.....1.....
LOGICAL-FIELD-IDENTIFIER	305	.....1.....
BIT-VARIABLE-IDENTIFIER	306	.....1.....
BIT-FIELD-IDENTIFIER	307	.....1.....
STRING-VARIABLE-IDENTIFIER	308	.....1.....
STRING-FIELD-IDENTIFIER	309	.....1.....
REFERENCE-FIELD-IDENTIFIER	310	.....1.....
INTEGER-ARRAY-IDENTIFIER	311	.....1.....
SHORT-REAL-ARRAY-IDENTIFIER	312	.....1.....
LONG-REAL-ARRAY-IDENTIFIER	313	.....1.....
SHORT-COMPLEX-ARRAY-IDENTIFIER	314	.....1.....
LONG-COMPLEX-ARRAY-IDENTIFIER	315	.....1.....
LOGICAL-ARRAY-IDENTIFIER	316	.....1.....
BIT-ARRAY-IDENTIFIER	317	.....1.....
STRING-ARRAY-IDENTIFIER	318	.....1.....
REFERENCE-ARRAY-IDENTIFIER	319	.....1.....
UNSCALED-REAL	320	.....1.
INTEGER-OR-SHORT-REAL-NUMBER	321	11

5.4 Précédence terminale et compilateur de conversation Fortran [*]

Il s'agissait d'écrire un compilateur de conversation Fortran c'est-à-dire un compilateur qui au fur et à mesure que l'on tape son programme sur la machine à écrire d'une console, en signale les erreurs et permet de les corriger. Afin d'éviter la réécriture complète d'un compilateur Fortran qui existait déjà sur la machine envisagée (UNIVAC 1108) l'option suivante a été prise :

- Le programme à compiler est d'abord traité par un analyseur dont le rôle consiste uniquement à détecter les erreurs. En cas d'erreur, après correction de la ligne erronée, l'analyseur reprend le programme à son début.

- Quand la totalité du programme a été jugée correcte par l'analyseur il est alors transmis au compilateur Fortran non conversationnel.

L'analyseur lui-même est composé de deux parties :

- Une partie édition dont le rôle est de coder la chaîne à analyser et de détecter certaines erreurs particulières telles que les erreurs de déclaration.

- Une partie analyse proprement dite qui consiste en l'analyseur de précédence terminale décrit au paragraphe 4.6.

Voici successivement la grammaire Fortran utilisée, ses relations de précédence terminale et la relation $\overset{*}{\rightarrow}$ entre ses opérandes (c'est-à-dire ses symboles non terminaux).

---

[*] Les réalisations décrites ici ont été principalement l'oeuvre de H. Beauchataud J.Fama et J.P. Tamiatto du Centre de Calcul Scientifique de l'Armement

PL nº 39 GRAMMAIRE FORTRAN

REGLES

```
                    PROGRAMME-DELIMITE  ::=  =//=  PROGRAMME  =//=
                             PROGRAMME  ::=  LISTE-D'ORDRES  RC  END
                                        OU   SOUS-PROGRAMME  RC  END
                           SOUS-PROGRAMME  ::=  SOUS-PROGRAMME-INCOMPLET  RC  RETURN
                                        OU   SOUS-PROGRAMME  RC  ORDRE
                                        OU   SOUS-PROGRAMME  RC  RETURN
                  SOUS-PROGRAMME-INCOMPLET  ::=  ORDRE-DE-SOUS-PROGRAMME
                                        OU   SOUS-PROGRAMME-INCOMPLET  RC  ORDRE
                   ORDRE-DE-SOUS-PROGRAMME  ::=  ORDRE-DE-SOUS-PROGRAMME-SANS-ETIQUETTE
                                        OU   NOMBRE-DE-MOINS-DE-5-CHIFFRES  ORDRE-DE-SOUS-PROGRAMME-SANS-ETIQUETTE
ORDRE-DE-SOUS-PROGRAMME-SANS-ETIQUETTE  ::=  ORDRE-FUNCTION
                                        OU   ORDRE-SUBROUTINE
                         LISTE-D'ORDRES  ::=  ORDRE
                                        OU   LISTE-D'ORDRES  RC  ORDRE
                                  ORDRE  ::=  ORDRE-EXECUTABLE
                                        OU   ORDRE-NON-EXECUTABLE
                                        OU   COMMENTAIRE
                       ORDRE-EXECUTABLE  ::=  ORDRE-EXECUTABLE-SANS-ETIQUETTE
                                        OU   NOMBRE-DE-MOINS-DE-5-CHIFFRES  ORDRE-EXECUTABLE-SANS-ETIQUETTE
                   ORDRE-NON-EXECUTABLE  ::=  ORDRE-DIMENSION
                                        OU   ORDRE-EXTERNAL
                                        OU   ETIQUETTE  CONTINUE
                                        OU   NOMBRE-DE-MOINS-DE-5-CHIFFRES  ORDRE-FORMAT
        ORDRE-EXECUTABLE-SANS-ETIQUETTE  ::=  ORDRE-DE-CALCUL
                                        OU   ORDRE-DE-TRANSFERT
                                        OU   ORDRE-D'ENTREE-SORTIE
                        ORDRE-DE-CALCUL  ::=  ORDRE-ARITHMETIQUE
                                        OU   ORDRE-CALL
                                        OU   ORDRE-DO
                     ORDRE-DE-TRANSFERT  ::=  ORDRE-GOTO-SIMPLE
                                        OU   ORDRE-GOTO-CALCULE
                                        OU   ORDRE-IF
                                        OU   STOP
                  ORDRE-D'ENTREE-SORTIE  ::=  ORDRE-PRINT
                                        OU   ORDRE-READ
                          ORDRE-FUNCTION  ::=  FUNCTION  IDENTIFICATEUR-SIMPLE  (  LISTE-D'IDENTIFICATEURS-SIMPLES
        LISTE-D'IDENTIFICATEURS-SIMPLES  ::=  IDENTIFICATEUR-SIMPLE
                                        OU   LISTE-D'IDENTIFICATEURS-SIMPLES  ,  IDENTIFICATEUR-SIMPLE
                       ORDRE-SUBROUTINE  ::=  SUBROUTINE  IDENTIFICATEUR-SIMPLE
                                        OU   SUBROUTINE  IDENTIFICATEUR-SIMPLE  (  LISTE-D'IDENTIFICATEURS-SIMPLES
                    IDENTIFICATEUR-SIMPLE  ::=  IDENTIFICATEUR-SIMPLE-REEL
                                        OU   IDENTIFICATEUR-SIMPLE-ENTIER
                         ORDRE-DIMENSION  ::=  DIMENSION  LISTE-POUR-DIMENSION
                    LISTE-POUR-DIMENSION  ::=  VARIABLE-DIMENSION
                                        OU   LISTE-POUR-DIMENSION  ,  VARIABLE-DIMENSION
                      VARIABLE-DIMENSION  ::=  IDENTIFICATEUR-SIMPLE  (  DECLARATION-DE-DIMENSION  )
                 DECLARATION-DE-DIMENSION  ::=  CONSTANTE-ENTIERE-SANS-SIGNE
                                        OU   DEUX-DIMENSIONS
                                        OU   DEUX-DIMENSIONS  ,  CONSTANTE-ENTIERE-SANS-SIGNE
                        DEUX-DIMENSIONS  ::=  CONSTANTE-ENTIERE-SANS-SIGNE  ,  CONSTANTE-ENTIERE-SANS-SIGNE
                              ORDRE-DO  ::=  DO  NOMBRE-DE-MOINS-DE-5-CHIFFRES  LISTE-POUR-DO
                         LISTE-POUR-DO  ::=  IDENTIFICATEUR-SIMPLE-ENTIER-AUXILIAIRE  =  SUITE-D'INDICES-POUR-DO
                SUITE-D'INDICES-POUR-DO  ::=  DEUX-INDICES-POUR-DO
                                        OU   DEUX-INDICES-POUR-DO  ,  INDICE-POUR-DO
                   DEUX-INDICES-POUR-DO  ::=  INDICE-POUR-DO  ,  INDICE-POUR-DO
                         INDICE-POUR-DO  ::=  CONSTANTE-ENTIERE-SANS-SIGNE
                                        OU   IDENTIFICATEUR-SIMPLE-ENTIER
                           ORDRE-FORMAT  ::=  FORMAT  (  LISTE-SPECIFICATION-FORMAT  )
                           ORDRE-CALL  ::=  CALL  IDENTIFICATEUR-SIMPLE
                                        OU   CALL  IDENTIFICATEUR-SIMPLE  (  LISTE-D'ARGUMENTS  )
                             ORDRE-IF  ::=  IF  CORPS-DE-IF
                          CORPS-DE-IF  ::=  CORPS-DE-IF-INCOMPLET  ,  ETIQUETTE
                 CORPS-DE-IF-INCOMPLET  ::=  CORPS-DE-IF-TRES-INCOMPLET  ,  ETIQUETTE
              CORPS-DE-IF-TRES-INCOMPLET  ::=  EXPRESSION-ARITHMETIQUE-PARENTHESEE  NOMBRE-DE-MOINS-DE-5-CHIFFRES
EXPRESSION-ARITHMETIQUE-PARENTHESEE  ::=  (  EXPRESSION-ARITHMETIQUE  )
                   ORDRE-GOTO-SIMPLE  ::=  GOTO  ETIQUETTE
                  ORDRE-GOTO-CALCULE  ::=  GOTO  LISTE-D'AIGUILLAGES  ,  IDENTIFICATEUR-SIMPLE-ENTIER-AUXILIAIRE
                 LISTE-D'AIGUILLAGES  ::=  (  LISTE-D'ETIQUETTES  )
                 LISTE-D'ETIQUETTES  ::=  ETIQUETTE
                                        OU   LISTE-D'ETIQUETTES  ,  ETIQUETTE
                     ORDRE-EXTERNAL  ::=  EXTERNAL  LISTE-DE-VARIABLES
                        ORDRE-PRINT  ::=  PRINT  LISTE-D'ENTREE-SORTIE
                         ORDRE-READ  ::=  READ  LISTE-D'ENTREE-SORTIE
                 LISTE-D'ENTREE-SORTIE  ::=  ETIQUETTE  ,  ARGUMENT-D'ENTREE-SORTIE
                                        OU   LISTE-D'ENTREE-SORTIE  ,  ARGUMENT-D'ENTREE-SORTIE
               ARGUMENT-D'ENTREE-SORTIE  ::=  VARIABLE
                                        OU   GROUPE-D'ARGUMENTS-D'E/S-PARENTHESES
  GROUPE-D'ARGUMENTS-D'E/S-PARENTHESES  ::=  (  GROUPE-D'ARGUMENTS-D'E/S  )
            GROUPE-D'ARGUMENTS-D'E/S  ::=  LISTE-DE-VARIABLES  ,  LISTE-POUR-DO
                                        OU   GROUPE-D'ARGUMENTS-D'E/S-PARENTHESES  ,  LISTE-POUR-DO
                   LISTE-DE-VARIABLES  ::=  VARIABLE
                                        OU   LISTE-DE-VARIABLES  ,  VARIABLE
                    ORDRE-ARITHMETIQUE  ::=  VARIABLE  =  EXPRESSION-ARITHMETIQUE
                              VARIABLE  ::=  VARIABLE-REELLE
                                        OU   VARIABLE-ENTIERE
                       FONCTION-REELLE  ::=  IDENTIFICATEUR-SIMPLE-REEL-AUXILIAIRE  (  LISTE-D'ARGUMENTS  )
                      FONCTION-ENTIERE  ::=  IDENTIFICATEUR-SIMPLE-ENTIER-AUXILIAIRE  (  LISTE-D'ARGUMENTS  )
                   LISTE-D'ARGUMENTS  ::=  ARGUMENT
                                        OU   LISTE-D'ARGUMENTS  ,  ARGUMENT
                            ARGUMENT  ::=  EXPRESSION-ARITHMETIQUE
                 EXPRESSION-ARITHMETIQUE  ::=  EXPRESSION-ENTIERE
                                        OU   EXPRESSION-REELLE
                     EXPRESSION-REELLE  ::=  TERME-REEL
                                        OU   SIGNE  TERME-REEL
                                        OU   EXPRESSION-REELLE  SIGNE  TERME-REEL
                    EXPRESSION-ENTIERE  ::=  TERME-ENTIER
                                        OU   SIGNE  TERME-ENTIER
                                        OU   EXPRESSION-ENTIERE  SIGNE  TERME-ENTIER
                           TERME-REEL  ::=  FACTEUR-REEL
                                        OU   TERME-REEL  *  FACTEUR-REEL
                                        OU   TERME-REEL  /  FACTEUR-REEL
```

PL nº 40 GRAMMAIRE FORTRAN

REGLES (SUITE)

```
                TERME-ENTIER  ::=  FACTEUR-ENTIER
                              OU   TERME-ENTIER  *  FACTEUR-ENTIER
                              OU   TERME-ENTIER  /  FACTEUR-ENTIER
                FACTEUR-REEL  ::=  PRIMAIRE-REEL
                              OU   PRIMAIRE-REEL  **  PRIMAIRE
              FACTEUR-ENTIER  ::=  PRIMAIRE-ENTIER
                              OU   PRIMAIRE-ENTIER  **  PRIMAIRE-ENTIER
                    PRIMAIRE  ::=  PRIMAIRE-ENTIER
                              OU   PRIMAIRE-REEL
               PRIMAIRE-REEL  ::=  CONSTANTE-REELLE-SANS-SIGNE
                              OU   VARIABLE-REELLE
                              OU   FONCTION-REELLE
                              OU   (  EXPRESSION-REELLE  )
              PRIMAIRE-ENTIER  ::=  CONSTANTE-ENTIERE-SANS-SIGNE
                              OU   VARIABLE-ENTIERE
                              OU   FONCTION-ENTIERE
                              OU   (  EXPRESSION-ENTIERE  )
            VARIABLE-ENTIERE  ::=  IDENTIFICATEUR-SIMPLE-ENTIER
                              OU   IDENTIFICATEUR-ENTIER-UNE-DIMENSION  (  INDICE  )
                              OU   IDENTIFICATEUR-ENTIER-DEUX-DIMENSIONS  (  INDICE-DEUX  )
                              OU   IDENTIFICATEUR-ENTIER-TROIS-DIMENSIONS  (  INDICE-TROIS  )
             VARIABLE-REELLE  ::=  IDENTIFICATEUR-SIMPLE-REEL
                              OU   IDENTIFICATEUR-REEL-UNE-DIMENSION  (  INDICE  )
                              OU   IDENTIFICATEUR-REEL-DEUX-DIMENSIONS  (  INDICE-DEUX  )
                              OU   IDENTIFICATEUR-REEL-TROIS-DIMENSIONS  (  INDICE-TROIS  )
                 INDICE-DEUX  ::=  INDICE  ,  INDICE
                INDICE-TROIS  ::=  INDICE-DEUX  ,  INDICE
                      INDICE  ::=  IDENTIFICATEUR-SIMPLE-ENTIER
                              OU   CONSTANTE-ENTIERE-SANS-SIGNE
                              OU   IDENTIFICATEUR-SIMPLE-ENTIER-AUXILIAIRE  SIGNE  CONSTANTE-POUR-INDICE
                              OU   CONSTANTE*IDENTIFICATEUR  SIGNE  CONSTANTE-POUR-INDICE
  CONSTANTE*IDENTIFICATEUR  ::=  CONSTANTE-POUR-INDICE  *  IDENTIFICATEUR-SIMPLE-ENTIER-AUXILIAIRE
     CONSTANTE-POUR-INDICE  ::=  CONSTANTE-ENTIERE-SANS-SIGNE
                  ETIQUETTE  ::=  NOMBRE-DE-MOINS-DE-5-CHIFFRES
 CONSTANTE-ENTIERE-SANS-SIGNE  ::=  NOMBRE-DE-MOINS-DE-5-CHIFFRES
                              OU   NOMBRE-DE-PLUS-DE-5-CHIFFRES
IDENTIFICATEUR-SIMPLE-REEL-AUXILIAIRE  ::=  IDENTIFICATEUR-SIMPLE-REEL
IDENTIFICATEUR-SIMPLE-ENTIER-AUXILIAIRE  ::=  IDENTIFICATEUR-SIMPLE-ENTIER
```

RELATIONS DE PRECEDENCE TERMINALE ENTRE LES SYMBOLES TERM

```
                                                  0000000000000000000000000000000000000000
                                                  0000000001111111111222222222233333333
                                                  1234567890123456789012345678901234567 8

                                (   001  121.11...1.......1........1211111111111
                                )   002  .444444..4.......4.............4.......
                                ,   003  1441114..1.......1........1.1111111111
                                =   004  141.114..1.......1........1.1111111111
                                *   005  144.444..1.......4........1.1111111111
                                /   006  144.444..1.......4........1.1111111111
                               RC   007  ...1..411.2.11111.2.1.11.11..1.1111111
                               DO   008  ..............................2.......
                               IF   009  1.1...4..........................1.....
                               **   010  144.444...........4........1.1111111111
                              END   011  ............4.............................
                             =//=   012  ...1..111..211111...1111111..1.1111111
                             STOP   013  ......4.............................
                             CALL   014  2.....4.....................1..1.........
                             GOTO   015  1.2...4..........................1.....
                             READ   016  ..1...4..........................1.....
                            PRINT   017  ..1...4..........................1.....
                            SIGNE   018  144.114..1.......4........1.1111111111
                           RETURN   019  .......4.............................
                           FORMAT   020  2.....................................
                         CONTINUE   021  ......4.............................
                         FUNCTION   022  2.....................1..1.........
                         EXTERNAL   023  ..1...4.................1..1..111111
                        DIMENSION   024  1.1...4.................1..1.........
                       SUBROUTINE   025  2.....4.................1..1.........
                      COMMENTAIRE   026  ......4.............................
          IDENTIFICATEUR-SIMPLE-REEL  027  4444444..4.......4.....................
     LISTE-SPECIFICATION-FORMAT  028  .2....................................
     CONSTANTE-REELLE-SANS-SIGNE  029  .44.444..4.......4.....................
    IDENTIFICATEUR-SIMPLE-ENTIER  030  4444444..4.......4.....................
    NOMBRE-DE-PLUS-DE-5-CHIFFRES  031  .44.444..4.......4.....................
   NOMBRE-DE-MOINS-DE-5-CHIFFRES  032  .441444114..111114.141..1.1..1..111111
  IDENTIFICATEUR-REEL-UNE-DIMENSION  033  2.....................................
 IDENTIFICATEUR-ENTIER-UNE-DIMENSION  034  2.....................................
 IDENTIFICATEUR-REEL-DEUX-DIMENSIONS  035  2.....................................
 IDENTIFICATEUR-REEL-TROIS-DIMENSIONS  036  2.....................................
IDENTIFICATEUR-ENTIER-DEUX-DIMENSIONS  037  2.....................................
IDENTIFICATEUR-ENTIER-TROIS-DIMENSIONS  038  2.....................................
```

PL n° 41 GRAMMAIRE FORTRAN

RELATION $\Rightarrow^*$ ENTRE LES SYMBOLES NON TERMINAUX

```
                                                     00000000000000000000000000000000000000000000000000000000000001111111111111
                                                     34444444444555555555566666666667777777777888888888899999999990000000000111
                                                     90123456789012345678901234567890123456789012345678901234567890123456789012
                         PROGRAMME-DELIMITE     039  1.........................................................................
                                  PROGRAMME     040  .1........................................................................
                             LISTE-D'ORDRES     041  ..1..1....11111..11111111111..............................................
                              SOUS-PROGRAMME    042  ...1......................................................................
                    SOUS-PROGRAMME-INCOMPLET    043  ....1.1111................................................................
                                      ORDRE     044  .....1....11111..11111111111..............................................
                     ORDRE-DE-SOUS-PROGRAMME    045  ......1111................................................................
ORDRE-DE-SOUS-PROGRAMME-SANS-ETIQUETTE          046  .......111................................................................
                             ORDRE-FUNCTION     047  ........1.................................................................
                           ORDRE-SUBROUTINE     048  .........1................................................................
                           ORDRE-EXECUTABLE     049  ..........1.1....11111111111..............................................
                       ORDRE-NON-EXECUTABLE     050  ...........1.11...........................................................
            ORDRE-EXECUTABLE-SANS-ETIQUETTE     051  ............1....11111111111..............................................
                            ORDRE-DIMENSION     052  .............1............................................................
                             ORDRE-EXTERNAL     053  ..............1...........................................................
                                  ETIQUETTE     054  ...............1..........................................................
                               ORDRE-FORMAT     055  ................1.........................................................
                             ORDRE-DE-CALCUL    056  .................1..111...................................................
                          ORDRE-DE-TRANSFERT    057  ..................1....111................................................
                      ORDRE-D'ENTREE-SORTIE     058  ...................1......11..............................................
                         ORDRE-ARITHMETIQUE     059  ....................1.....................................................
                                 ORDRE-CALL     060  .....................1....................................................
                                   ORDRE-DO     061  ......................1...................................................
                          ORDRE-GOTO-SIMPLE     062  .......................1..................................................
                          ORDRE-GOTO-CALCULE    063  ........................1.................................................
                                   ORDRE-IF     064  .........................1................................................
                                ORDRE-PRINT     065  ..........................1...............................................
                                 ORDRE-READ     066  ...........................1..............................................
                      IDENTIFICATEUR-SIMPLE     067  ............................1.............................................
         LISTE-D'IDENTIFICATEURS-SIMPLES        068  ............................11............................................
                      LISTE-POUR-DIMENSION      069  ..............................11..........................................
                         VARIABLE-DIMENSION     070  ...............................1..........................................
                  DECLARATION-DE-DIMENSION      071  ................................111.......................................
              CONSTANTE-ENTIERE-SANS-SIGNE      072  .................................1........................................
                            DEUX-DIMENSIONS     073  ..................................1.......................................
                              LISTE-POUR-DO     074  ...................................1......................................
IDENTIFICATEUR-SIMPLE-ENTIER-AUXILIAIRE         075  ....................................1.....................................
                     SUITE-D'INDICES-POUR-DO    076  .....................................11...................................
                      DEUX-INDICES-POUR-DO      077  ......................................1...................................
                            INDICE-POUR-DO      078  .................................1.....1..................................
                           LISTE-D'ARGUMENTS    079  .................................1......1....1........111.111111111.1.....
                                 CORPS-DE-IF    080  .........................................1................................
                       CORPS-DE-IF-INCOMPLET    081  ..........................................1...............................
                  CORPS-DE-IF-TRES-INCOMPLET    082  ...........................................1..............................
          EXPRESSION-ARITHMETIQUE-PARENTHESEE   083  ............................................1.............................
                     EXPRESSION-ARITHMETIQUE    084  .................................1...........1........111.1.1111111.1.....
                         LISTE-D'AIGUILLAGES    085  ..............................................1...........................
                          LISTE-D'ETIQUETTES    086  ...............1...............................1..........................
                           LISTE-DE-VARIABLES   087  ................................................1..1..11..................
                       LISTE-D'ENTREE-SORTIE    088  .................................................1........................
                    ARGUMENT-D'ENTREE-SORTIE    089  ..................................................111.11..................
                                    VARIABLE    090  ...................................................1..11..................
      GROUPE-D'ARGUMENTS-D'E/S-PARENTHESES      091  ....................................................1.....................
                     GROUPE-D'ARGUMENTS-D'E/S   092  .....................................................1....................
                             VARIABLE-REELLE    093  ......................................................1...................
                            VARIABLE-ENTIERE    094  .......................................................1..................
                             FONCTION-REELLE    095  ........................................................1.................
    IDENTIFICATEUR-SIMPLE-REEL-AUXILIAIRE       096  .........................................................1................
                            FONCTION-ENTIERE    097  ..........................................................1...............
                                    ARGUMENT    098  .................................1...........1........111.111111111.1.....
                          EXPRESSION-ENTIERE    099  .................................1.....................1..1.1..1.1..1.....
                           EXPRESSION-REELLE    100  ......................................................1.1....11.1.1.......
                                 TERME-REEL     101  ......................................................1.1.....1.1.1.......
                               TERME-ENTIER     102  .................................1.....................1..1....1.1..1.....
                              FACTEUR-REEL      103  ......................................................1.1.......1.1.......
                            FACTEUR-ENTIER      104  .................................1.....................1..1......1..1.....
                             PRIMAIRE-REEL      105  ......................................................1.1.........1.......
                                  PRIMAIRE      106  .................................1....................111.1.......111.....
                            PRIMAIRE-ENTIER     107  .................................1.....................1..1.........1.....
                                    INDICE      108  .................................1...................................1....
                               INDICE-DEUX      109  ......................................................................1...
                              INDICE-TROIS      110  .......................................................................1..
                      CONSTANTE-POUR-INDICE     111  .................................1......................................1.
                 CONSTANTE*IDENTIFICATEUR       112  .........................................................................1
```

5.5 Précédence terminale et compilateur de conversation pour un sous-ensemble d'Algol

Nous décrivons ici un compilateur de conversation pour un sous-ensemble d'Algol. Ce compilateur est composé de trois parties :

- un éditeur dont le rôle est de reconnaître les symboles de base, de supprimer les déclarations et de constituter le dictionnaire des identificateurs utilisés ;

- un analyseur de précédence terminale (celui décrit au paragraphe 4.6) ;

- un générateur qui à chaque réduction d'une sous-chaîne par l'analyseur génère une partie du programme objet sous forme de macro-instructions.

Ce compilateur est principalement l'oeuvre de G. Tassart ; les macro-instructions ont été définies en langage d'assemblage MAP (IBM 7044) par A. Auroux. Nous donnons successivement la grammaire du sous-ensemble d'Algol, ses relations de précédence terminale, la relation $\Rightarrow^{*}$ entre ses symboles non terminaux et un exemple commenté de compilation d'un programme.

PL n°42 GRAMMAIRE D'UN SOUS-ENSEMBLE D'ALGOL

REGLES

```
                    PROGRAMME  ::=  'DEBUT'  SUITE-D'INSTRUCTIONS  'FIN'
                               OU   'DEBUT'  'FIN'
          SUITE-D'INSTRUCTIONS  ::=  INSTRUCTION
                               OU   SUITE-D'INSTRUCTIONS  'POINT-VIRGULE'  INSTRUCTION
                               OU   'POINT-VIRGULE'  INSTRUCTION
                               OU   SUITE-D'INSTRUCTIONS  'POINT-VIRGULE'
                               OU   'POINT-VIRGULE'
                   INSTRUCTION  ::=  INSTRUCTION-NON-ETIQUETEE
                               OU   ETIQUETTE  :  INSTRUCTION-NON-ETIQUETEE
                               OU   ETIQUETTE  :
       INSTRUCTION-NON-ETIQUETEE  ::=  VARIABLE-REELLE  :=  EXPRESSION-ARITHMETIQUE
                               OU   VARIABLE-ENTIERE  :=  EXPRESSION-ARITHMETIQUE
                               OU   VARIABLE-BOOLEENNE  :=  EXPRESSION-BOOLEENNE
                               OU   INSTRUCTION-PROCEDURE
                               OU   'ALLERA'  EXPRESSION-DE-DESIGNATION
         EXPRESSION-ARITHMETIQUE  ::=  EXPRESSION-ARITHMETIQUE-SIMPLE
                               OU   'SI'  EXPRESSION-BOOLEENNE  'ALORS'  EXPRESSION-ARITHMETIQUE-SIMPLE  'SINON'
                                    EXPRESSION-ARITHMETIQUE
   EXPRESSION-ARITHMETIQUE-SIMPLE  ::=  TERME
                               OU   +  TERME
                               OU   -  TERME
                               OU   EXPRESSION-ARITHMETIQUE-SIMPLE  +  TERME
                               OU   EXPRESSION-ARITHMETIQUE-SIMPLE  -  TERME
                         TERME  ::=  FACTEUR
                               OU   TERME  *  FACTEUR
                               OU   TERME  /  FACTEUR
                               OU   TERME  '/'  FACTEUR
                      FACTEUR  ::=  PRIMAIRE-ARITHMETIQUE
                               OU   FACTEUR  'PUISSANCE'  PRIMAIRE-ARITHMETIQUE
           PRIMAIRE-ARITHMETIQUE  ::=  VARIABLE-REELLE
                               OU   VARIABLE-ENTIERE
                               OU   NOMBRE-REEL
                               OU   NOMBRE-ENTIER
                               OU   (  EXPRESSION-ARITHMETIQUE  )
                               OU   FONCTION  (  EXPRESSION-ARITHMETIQUE  )
          EXPRESSION-BOOLEENNE  ::=  EXPRESSION-BOOLEENNE-SIMPLE
                               OU   'SI'  EXPRESSION-BOOLEENNE  'ALORS'  EXPRESSION-BOOLEENNE-SIMPLE  'SINON'
                                    EXPRESSION-BOOLEENNE
    EXPRESSION-BOOLEENNE-SIMPLE  ::=  FACTEUR-BOOLEEN
                               OU   EXPRESSION-BOOLEENNE-SIMPLE  'OU'  FACTEUR-BOOLEEN
              FACTEUR-BOOLEEN  ::=  SECONDAIRE-BOOLEEN
                               OU   FACTEUR-BOOLEEN  'ET'  SECONDAIRE-BOOLEEN
           SECONDAIRE-BOOLEEN  ::=  PRIMAIRE-BOOLEEN
                               OU   'NON'  PRIMAIRE-BOOLEEN
             PRIMAIRE-BOOLEEN  ::=  'VRAI'
                               OU   'FAUX'
                               OU   VARIABLE-BOOLEENNE
                               OU   (  EXPRESSION-BOOLEENNE  )
                               OU   EXPRESSION-ARITHMETIQUE-SIMPLE  =  EXPRESSION-ARITHMETIQUE-SIMPLE
                               OU   EXPRESSION-ARITHMETIQUE-SIMPLE  'INFERIEUR'  EXPRESSION-ARITHMETIQUE-SIMPLE
    EXPRESSION-DE-DESIGNATION  ::=  EXPRESSION-DE-DESIGNATION-SIMPLE
                               OU   'SI'  EXPRESSION-BOOLEENNE  'ALORS'  EXPRESSION-DE-DESIGNATION-SIMPLE  'SINON'
                                    EXPRESSION-DE-DESIGNATION
_EXPRESSION-DE-DESIGNATION-SIMPLE_  ::=  ETIQUETTE
                               OU   (  EXPRESSION-DE-DESIGNATION  )
```

RELATIONS DE PRECEDENCE TERMINALE

```
                                  000000000000000000000C000C0C0000
                                  000000000111111111122222222233
                                  123456789012345678901234567890 12
          'DEBUT'  001  .211.11.................1.11...
            'FIN'  002  ..............................
      'POINT-VIRGULE'  003  .441.11.................1.11...
     VARIABLE-REELLE  004  .44.2...44444444...444.........44
                :=  005  .441.111..111111111.11111....111
     VARIABLE-ENTIERE  006  .44.2...44444444...444.........44
   VARIABLE-BOOLEENNE  007  .44.2...44..........444..........
            'SI'  008  ...1.1112.111111111.11111....111
          'ALORS'  009  ...1.11..2111111111.111111...111
         'SINON'  010  .441.1114.1111111114111111...111
                +  011  .441.1..44441111111444.......144
                -  012  .441.1..44441111111444.......144
                *  013  .441.1..44444441111444.......144
                /  014  .441.1..44444441111444.......144
              '/'  015  .441.1..44444441111444.......144
      'PUISSANCE'  016  .441.1..444444441111444.......144
     NOMBRE-REEL  017  .44.....44444444...444.........44
   NOMBRE-ENTIER  018  .44.....44444444...444.........44
                (  019  ...1.111..1111111112111111...111
                )  020  .44.....44444444...444.........44
           'OU'  021  .441.11.44111111111441111....111
           'ET'  022  .441.11.44111111111444111....111
          'NON'  023  .441.11.44111111111444.11....111
         'VRAI'  024  .44.....44..........444..........
         'FAUX'  025  .44.....44..........444..........
     ETIQUETTE  026  .44......4..........4.......2.....
                :  027  .441.11.......................11...
INSTRUCTION-PROCEDURE  028  .44.............................
        'ALLERA'  029  .44....1..........1.......1.....
      FONCTION  030  ..................2..............
                =  031  .441.1..44111111111444.......1..
    'INFERIEUR'  032  .441.1..44111111111444.......1..
```

RELATION $\Rightarrow^+$ ENTRE LES SYMBOLES NON TERMINAUX

```
                                  0000000000000C0
                                  3333333444444444
                                  3456789012345678
                    PROGRAMME  033  1................
          SUITE-D'INSTRUCTIONS  034  .111............
                  INSTRUCTION  035  ..11............
       INSTRUCTION-NON-ETIQUETEE  036  ...1............
           EXPRESSION-ARITHMETIQUE  037  ....1..1111.....
              EXPRESSION-BOOLEENNE  038  .....1.....1111.
          EXPRESSION-DE-DESIGNATION  039  ......1.........1
   EXPRESSION-ARITHMETIQUE-SIMPLE  040  .......1111......
                     TERME  041  ........111.....
                   FACTEUR  042  .........11.....
               PRIMAIRE-ARITHMETIQUE  043  ..........1.....
      EXPRESSION-BOOLEENNE-SIMPLE  044  ...........1111.
          FACTEUR-BOOLEEN  045  ............111.
            SECONDAIRE-BOOLEEN  046  .............11.
               PRIMAIRE-BOOLEEN  047  ..............1.
   EXPRESSION-DE-DESIGNATION-SIMPLE  048  ..............1
```

PL n° 43 EXEMPLE D'ANALYSE ET DE COMPILATION D'UN PROGRAMME ALGOL

ANALYSE DU PROGRAMME

```
'DEBUT' 'REEL' X1,X2 :: X2:=3 ::

        DEBUT
        RESER   X1,1
        RESER   X2,1

C1: X1:=X2 :: X2:=(X1+3/X1)/2 ::

                                        *****LA SEQUENCE
                                             =3              NOMBRE-ENTIER
                                          EST REMPLACEE PAR
                                             PRIMAIRE-ARITHMETIQUE
                                        *****LA SEQUENCE
                                             X2      VARIABLE-REELLE
                                             :=
                                             =3              000006 PRIMAIRE-ARITHMETIQUE
                                          EST REMPLACEE PAR
                                             INSTRUCTION-NON-ETIQUETEE
        CONER1  =3
        AFFE    X2

                                        *****LA SEQUENCE
                                             X2      VARIABLE-REELLE
                                          EST REMPLACEE PAR
                                             PRIMAIRE-ARITHMETIQUE
                                        *****LA SEQUENCE
                                             X1      VARIABLE-REELLE
                                             :=
                                             X2      000004 PRIMAIRE-ARITHMETIQUE
                                          EST REMPLACEE PAR
                                             INSTRUCTION-NON-ETIQUETEE
E1      ANATO   X2
        AFFE    X1

                                        *****LA SEQUENCE
                                             E1      ETIQUETTE
                                             :
                                             INSTRUCTION-NON-ETIQUETEE
                                          EST REMPLACEE PAR
                                             INSTRUCTION
                                        *****LA SEQUENCE
                                             INSTRUCTION-NON-ETIQUETEE
                                             'POINT-VIRGULE'
                                             INSTRUCTION
                                          EST REMPLACEE PAR
                                             SUITE-D'INSTRUCTIONS
                                        *****LA SEQUENCE
                                             X1      VARIABLE-REELLE
                                          EST REMPLACEE PAR
                                             PRIMAIRE-ARITHMETIQUE
                                        *****LA SEQUENCE
                                             =3              NOMBRE-ENTIER
                                          EST REMPLACEE PAR
                                             PRIMAIRE-ARITHMETIQUE
                                        *****LA SEQUENCE
                                             X1      VARIABLE-REELLE
                                          EST REMPLACEE PAR
                                             PRIMAIRE-ARITHMETIQUE
                                        *****LA SEQUENCE
                                             =3              000006 PRIMAIRE-ARITHMETIQUE
                                             /
                                             X1      000004 PRIMAIRE-ARITHMETIQUE
                                          EST REMPLACEE PAR
                                             TERME
        CONER1  =3
        FDIV2   X1

                                        *****LA SEQUENCE
                                             X1      000004 PRIMAIRE-ARITHMETIQUE
                                             +
                                             000000 000004 TERME
                                          EST REMPLACEE PAR
                                             EXPRESSION-ARITHMETIQUE-SIMPLE
        FADD3   X1

                                        *****LA SEQUENCE
                                             (
                                             000000 000004 EXPRESSION-ARITHMETIQUE-SIMPLE
                                             )
                                          EST REMPLACEE PAR
                                             PRIMAIRE-ARITHMETIQUE
'ALLERA' 'SI' X1-X2 'INFER' 0,001 'ALORS' E2 'SINON' E1 ::

                                        *****LA SEQUENCE
                                             =2              NOMBRE-ENTIER
                                          EST REMPLACEE PAR
                                             PRIMAIRE-ARITHMETIQUE
                                        *****LA SEQUENCE
                                             000000 000004 PRIMAIRE-ARITHMETIQUE
                                             /
                                             =2              000006 PRIMAIRE-ARITHMETIQUE
                                          EST REMPLACEE PAR
                                             TERME
        CONER1  =2
        FDIV4
```

```
                                        *****LA SEQUENCE
                                              X2      VARIABLE-REELLE
                                              :=
                                              000000 000004 TERME
                                           EST REMPLACEE PAR
                                              INSTRUCTION-NON-ETIQUETEE

AFFE    X2                              *****LA SEQUENCE
                                              SUITE-D'INSTRUCTIONS
                                              'POINT-VIRGULE'
                                              INSTRUCTION-NON-ETIQUETEE
                                           EST REMPLACEE PAR
                                              SUITE-D'INSTRUCTIONS
                                        *****LA SEQUENCE
                                              X1      VARIABLE-REELLE
                                           EST REMPLACEE PAR
                                              PRIMAIRE-ARITHMETIQUE
                                        *****LA SEQUENCE
                                              X2      VARIABLE-REELLE
                                           EST REMPLACEE PAR
                                              PRIMAIRE-ARITHMETIQUE
                                        *****LA SEQUENCE
                                              X1      000004 PRIMAIRE-ARITHMETIQUE
                                              -
                                              X2      000004 PRIMAIRE-ARITHMETIQUE
                                           EST REMPLACEE PAR
                                              EXPRESSION-ARITHMETIQUE-SIMPLE
FSUB1   X1,X2
                                        *****LA SEQUENCE
                                              =0.001          NOMBRE-REEL
                                           EST REMPLACEE PAR
                                              PRIMAIRE-ARITHMETIQUE
                                        *****LA SEQUENCE
                                              000000 000004 EXPRESSION-ARITHMETIQUE-SIMPLE
                                              'INFERIEUR'
                                              =0.001          000004 PRIMAIRE-ARITHMETIQUE
                                           EST REMPLACEE PAR
                                              PRIMAIRE-BOOLEEN
INFE2   =0.001
SCD     0
TRA     E2

                                        *****LA SEQUENCE
                                              E2      ETIQUETTE
                                           EST REMPLACEE PAR
                                              EXPRESSION-DE-DESIGNATION-SIMPLE
SIC     0,1
TRA     E1

E2: 'FIN'

                                        *****LA SEQUENCE
                                              E1      ETIQUETTE
                                           EST REMPLACEE PAR
                                              EXPRESSION-DE-DESIGNATION-SIMPLE
                                        *****LA SEQUENCE
                                              'SI'
                                              000000 PRIMAIRE-BOOLEEN
                                              'ALORS'
                                              EXPRESSION-DE-DESIGNATION-SIMPLE
                                              'SINON'
                                              EXPRESSION-DE-DESIGNATION-SIMPLE
                                           EST REMPLACEE PAR
                                              EXPRESSION-DE-DESIGNATION

                                        *****LA SEQUENCE
                                              'ALLERA'
                                              EXPRESSION-DE-DESIGNATION
                                           EST REMPLACEE PAR
                                              INSTRUCTION-NON-ETIQUETEE
                                        *****LA SEQUENCE
                                              SUITE-D'INSTRUCTIONS
                                              'POINT-VIRGULE'
                                              INSTRUCTION-NON-ETIQUETEE
                                           EST REMPLACEE PAR
                                              SUITE-D'INSTRUCTIONS
                                        *****LA SEQUENCE
                                              E2      ETIQUETTE
                                              :
                                           EST REMPLACEE PAR
                                              INSTRUCTION
                                        *****LA SEQUENCE
                                              SUITE-D'INSTRUCTIONS
                                              'POINT-VIRGULE'
                                              INSTRUCTION
                                           EST REMPLACEE PAR
                                              SUITE-D'INSTRUCTIONS
                                        *****LA SEQUENCE
                                              'DEBUT'
                                              SUITE-D'INSTRUCTIONS
                                              'FIN'
                                          EST REMPLACEE PAR
                                              PROGRAMME
E2      FIN
```

L ANALYSE DE CE PROGRAMME A DURE 0 MINUTES ET 3 SECONDES.

Commentaire concernant les deux planches précédentes :

Le programme compilé calcule $\sqrt{3}$ par une méthode itérative ; le voici :

début réel X1,X2 ; X2 := 3 ;
E_1 : X1 := X2 ; X2 := (X1+3/X1)/2 ;
allera si X1-X2 < 0.001 alors E2 sinon E1 ;
E2 : fin

sur les planches il apparait souligné. A droite sont imprimés les résultats de l'analyse et à gauche les macro-instructions générées dont voici la liste avec quelques commentaires :

	DEBUT		début du programme ;
	RESER	X1,1	réserver une mémoire pour le réel X1 ;
	RESER	X2,1	réserver une mémoire pour le réel X2 ;
	CONER	1 =3	convertir l'entier 3 en réel et l'empiler ;
	AFFE	X2	affecter le sommet de la pile à X2 ;
E1	ANATO	X2	empiler X2 ;
	AFFE	X1	affecter le sommet de la pile à X1 ;
	CONER	1 =3	convertir l'entier 3 en réel et l'empiler ;
	FDIVZ	X1	diviser le sommet de la par X1 et empiler le résultats ;
	FADD	3 X1	ajouter X1 au sommet de la pile et empiler le résultats ;
	CONER	1 =2	convertir l'entier 2 en réel et l'empiler ;
	FDIV	4	diviser l'avant-sommet de la pile par son sommet et empiler le résultat ;
	AFFE	X2	affecter le sommet de la pile à X2 ;
	FSUB	1 X1,X2	soustraire X2 à X1 et empiler le résultat ;
	INFE	2 =0.001	si le sommet de la pile est inférieur à 0.001 alors le remplacer par 1 sinon par 0 ;
	SCD	0	si le sommet de la pile est égal à 0 alors sauter à l'instruction qui suit l'instruction SIC portant le numéro 0 sinon exécuter l'instruction qui suit ;
	TRA	E2	aller à E2 ;
	SIC	0,1	cette instruction porte le numéro 0, son effet est de sauter à l'instruction FEC portant le numéro 1 ;
	TRA	E1	allera à E1 ;
	FEC	1	cette instruction porte le numéro 1, son action est sans effet ;
E2	FIN		fin du programme.

CONCLUSION

Après cet exposé qui s'est voulu méthodique, nous aimerions terminer sur quelques remarques dictées par l'expérience.

Si nous avons décrit en premier lieu la méthode d'analyse utilisant des relations de précédence sur la totalité du vocabulaire, ceci est uniquement dû à sa simplicité. En fait cette méthode fut notre dernière découverte. Elle s'est montrée très efficace et le fait qu'elle permette d'analyser d'une façon déterministe des textes, non analysables, par un automate à une pile déterministe, en est sûrement la principale raison. On peut cependant lui faire un reproche, celui de devoir introduire des règles superflues dans la grammaire, afin d'obtenir des relations de précédence incompatibles deux à deux. Les règles IDENTIFICATEUR ::= IDENTIFICATEUR-BIS, TERME ::= TERME-BIS etc. de la grammaire Algol de précédence totale du paragraphe 5.1 en sont des exemples. Nous n'avons pas présenté d'application de cet analyseur sous sa forme non-déterministe c'est à dire sous sa forme la plus générale. Il semble cependant qu'il se prêterait très bien à l'analyse de langages naturels, les relations de précédence permettant d'éviter de nombreuses impasses dans le processus de reconnaissance. Le fait que les grammaires utilisées pour ces langages soient souvent ambiguës n'est pas gênant puisque cet analyseur permet de trouver toutes les structures d'un texte donné.

En ce qui concerne la deuxième méthode d'analyse qui utilise des relations de précédence définies sur une partie plus ou moins arbitraire du vocabulaire, on peut se demander pour quelle raison nous avons imposé un certain ordre dans l'application des règles. Sans cette contrainte le problème de trouver de "bonnes" relations de précédence serait devenu trop complexe et il est douteux que l'on eût pu les exprimer à partir de relations plus simples et des opérations définies sur les relations binaires. Il va de soi que cette restriction peut rendre plus difficile la construction d'une grammaire à relations de précédence incompatibles deux à deux, mais cet inconvénient est compensé par la

liberté du choix des opérateurs, liberté qui permet de mieux ajuster son modèle de description syntaxique. En général on est d'ailleurs amené à choisir comme opérateurs les symboles qui effectivement jouent un rôle d'opérateur au sens habituel du terme. Les métavariables OPERATEUR-DE-RELATION, OPERATEUR-ADDITIF de la grammaire du paragraphe 5.2 en sont des exemples typiques. Enfin rappelons une caractéristique bien agréable de cette méthode : celle de ne pas avoir à procéder par essais successifs pour traiter certaines ambiguïtés comme celles dûes à l'impossibilité d'exprimer le lien entre l'endroit où une variable est utilisée et l'endroit où elle est déclarée.

REFERENCES

BC M. Brasseur et J. Cohen, Algorithmes d'analyse syntaxique pour langages "context-free", Chiffres, vol 8, n° 2 et 3, 1965.

BGL L. Bolliet, N. Gastinel, et P.J. Laurent, Un nouveau langage scientifique ALGOL manuel pratique, Herrmann, 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.

Ch N. Chomsky, On certain formal properties of grammars, Inf. and Control, vol 2, Décembre 1959.

Co-1 A. Colmerauer, Notions d'opérateurs dans une grammaire "context-free", RIRO, n° 2, 1967.

Co-2 A. Colmerauer, Total precedence relations, a paraître dans J.ACM.

Fl-1 R.W. Floyd, Syntatic analysis and operator precedence, J.ACM, vol 10, Juillet 1963.

Fl-2 R.W. Floyd, Bounded context syntatic analysis, Comm. ACM, vol 7, Février 1964.

Fl-3 R.W. Floyd, Non-deterministic algorithms, Publication interne, Carnegie Institute of Technology, Pittsburgh, Novembre 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

GP T.H. Griffiths et S.R. Petrick, On the relative effecicencies of context-free grammar recogniezers, Comm. ACM, Mai 1965

Ir-1 F.T. Irons, A syntax directed compiler for Algol 60, Comm. ACM, vol 4, Janvier 1961.

Ir-2 F.T. Irons, An error-correcting parse algorithm, Comm. ACM, vol 6, Novembre 1963.

Kn D.E. Knuth, On the translation of languages from left to right, Inf. And Control, n° 8, 1965.

Na P. Naur, Report on the algorithmic language Algol 60, Comm. ACM, vol 3 Mai 1960 ;
Revised report on the algorithmic language Algol 60, Comm. ACM, vol 6, Janvier 1963.

Pa-1 C. Pair, Arbres piles et compilation, RFTI n° 3, 1964.

Pa-2 C. Pair, Etude de la notion de pile application à l'analyse syntaxique, Thèse, Université de Nancy, Décembre 1965.

Wa S. Warshall, A theorem on boolean matrices, J. ACM, vol 9, Janvier 1962.

WH N. Wirth et C.A.R. Hoare, A contribution to the developpement of Algol, Comm. ACM, vol 9, Juin 1966.

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