UNIVERSITE DE NANCY I

FACULTE DES SCIENCES

SYSTEME GENERAL

D'ANALYSE SYNTAXIQUE

THESE

pour l'obtention du

DOCTORAT de SPECIALITE MATHEMATIQUES (3ème CYCLE)

Soutenue devant le Jury le 6 Novembre 1971

par

CLAUDE LEMAIRE

Jury : Mr LEGRAS J. , Président
Mr DEPAIX M. , Examinateur
Mr PAIR C. , Examinateur

UNIVERSITE DE NANCY I FACULTE DES SCIENCES

SYSTEME GENERAL

D'ANALYSE SYNTAXIQUE

par CLAUDE LEMAIRE

Je remercie Monsieur le Professeur J. LEGRAS, Directeur de l'Institut Universitaire de Calcul Automatique, qui m'honore de sa présence dans le Jury.

Ce travail a été effectué sous la direction de Monsieur le Professeur C. PAIR à qui j'adresse l'expression de ma haute considération pour l'enseignement qu'il m'a prodigué, ainsi que pour l'aide précieuse qu'il m'a apportée dans ce travail.

J'exprime ma gratitude à Monsieur le Professeur M. DEPAIX qui a accepté de faire partie du Jury.

Je tiens également à remercier toute l'équipe de l'Institut de Calcul Automatique et particulièrement Mademoiselle D. COLIN qui a réalisé matériellement ce travail.

SOMMAIRE

INTRODUCTION

Jusqu'à présent pour réaliser l'analyse syntaxique d'un mot appartenant au langage engendré par une grammaire de Chomsky (ou C-grammaire) nous avions à notre disposition un certain nombre de méthodes d'analyse qui pouvaient être utilisées séparément (PAIR [1], [4]). Il s'agit dans cette étude de jeter les prémices de la construction d'un "système général d'analyse", qui pour toute C-grammaire, $G=(T \cup N, \rightarrow, X)$, nous fournirait un "programme d'analyse" constitué essentiellement par un quadruplet (V', g, K, h), avec

- g une application de V' dans V $(V = T \cup N)$
- K un langage régulier sur $V' \cup \overline{V}'$
- h un homomorphisme alphabétique de V^* dans T^*, qui à tout mot α de T^* associe l'ensemble, $h^{-1}(\alpha) \cap g^*(K \cap P(V'))$, (1), des analyses de α selon la grammaire G.

Ce système d'analyse permettrait en particulier l'étude du choix automatique de la méthode d'analyse la mieux adaptée à une grammaire G. Nous nous contenterons en fait, de donner une formalisation de l'analyse syntaxique, suggérée par Monsieur PAIR [4], qui conduise aux ensembles définis ci-dessus, (1), ainsi qu'aux techniques qui conviennent à leur construction.

ERRATA

p. 8 - ligne 14 :

- $\nu(r+s) > \nu(r)$ si $s \neq \Lambda$, $\nu(r+s) > \nu(s)$ si $r \neq \Lambda$

p. 12 - ligne 12 :

" Z est une initiale de Y "

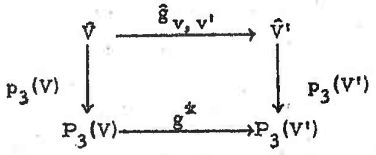
p. 18 - ligne 24 :

1) Soit L un bilangage régulier de $P_1(V)$, non réduit au mot vide

p. 20 - ligne 14 :

, la structure S est dite ≪compatible≫ si

p. 25 -



p. 32 - avant dernière ligne :

une analyse de gauche à droite qui

p. 44 - ligne 3 :

$K_3^- = K_3 \cap L(V)$

p. 74 - ligne 13 :

, par l'isomorphisme de binoïdes, p, de $\hat{W}$ dans le binoïde ...

p. 95 -

Dans le titre, il faut lire : Chapitre V et non : Chapitre IV.

p. 117 - ligne 16 :

Soit une fausse piste pour le couple (v, γ)

- ligne 22 :

$(g_q(\tilde{v}), g_k(\gamma))$

- ligne 21 :

Si nous supposons, $q' \geq q$ et $k' \geq k$, et que nous connaissions les ensembles $\mathcal{C}(q, k, x)$, une condition suffisante ....

p. 121

Dans la proposition 5, il faut lire $(p_2(t)v, a\gamma)$ au lieu de $(p_1(t)v, q\gamma)$

CHAPITRE I

Généralités

RAPPEL SUR LES GRAPHES

Définition : Un graphe est un couple, (E, Γ), formé :
- d'un ensemble E,
- d'une relation binaire, Γ, dans E.

Chemin d'un graphe (E, Γ)

Une suite $(x_0, \ldots, x_n)$ de points du graphe (E, Γ) telle que :

$x_{i-1} \Gamma x_i$, pour $i=1, 2, \ldots, n$, est un chemin du graphe (E, Γ) qui joint x_0 à x_n ; x_0 est son origine et x_n, son extrémité.

On emploiera dans la suite la notation, $C(A, B)$, pour désigner l'ensemble des chemin d'un graphe (E, Γ), d'origine dans le sous-ensemble A de E et d'extrémité dans le sous-ensemble B de E (ceci, lorsqu'il n'y aura pas d'ambiguïté sur le graphe utilisé)

Successeurs d'un point, x, de (E, Γ)

On notera, $\Gamma(x)$, l'ensemble des successeurs de x :

$$y \in \Gamma(x) \iff x \Gamma y.$$

Prédécesseurs d'un point x de (E, Γ)

On notera, $\Gamma^{-1}(x)$, l'ensemble des prédécesseurs de x :

$$y \in \Gamma^{-1}(x) \iff y \Gamma x$$

Circuit d'un graphe (E, Γ)

Un circuit sur x est un chemin de (E, Γ) dont l'origine et l'extrémité sont confondues avec le point x.

Chemin élémentaire et circuit élémentaire :

Un chemin (ou un circuit), $(x_0, x_1, \ldots, x_n)$ est élémentaire si :

$$(\forall i, j \in \{0, 1, \ldots, n\})\ (i \neq j)\ (x_i \neq x_j).$$

Sous-graphe d'un graphe (E, Γ)

Un couple (E', Γ') est un sous-graphe de (E, Γ) si E' est un sous-ensemble de E et Γ' la restriction de Γ à E.

1.2 - QUELQUES DEFINITIONS A PROPOS DU MONOIDE LIBRE, V^*, SUR UN VOCABULAIRE V

Longueur d'un mot α de V^* :

Soit, $\alpha = a_1 a_2 \cdots a_k$ un mot sur V : k est la longueur de α.

On notera : $|\alpha| = k$.

Facteurs d'un mot sur V :

Soient α, β, γ trois mots sur V tels que : $\alpha = \beta\gamma$.

On dit que β est facteur gauche de α, et que γ est facteur droit de α.

Facteurs gauches de longueur donnée :

Soit, g_k, l'application qui associe à tout mot α de V^*, son facteur gauche de longueur k, lorsque $|\alpha| \geq k$ et α lorsque $|\alpha| < k$.

On démontre aisément les propriétés suivantes :

- $\forall \alpha, \beta \in V^* \qquad g_k(\alpha\beta) = g_k(g_k(\alpha)\, g_k(\beta))$.
- et d'une manière générale en désignant par $\prod_{i=1}^{p} \alpha_i$, la concaténation de p mots, α_i, de V^* on a :

$$g_k\left(\prod_{i=1}^{p} \alpha_i\right) = g_k\left(\prod_{i=1}^{p} g_k(\alpha_i)\right)$$

- $(\forall k, k' \in \mathbb{N})\ (k \leq k') : \quad g_k \circ g_{k'} \equiv g_k$

Réfléchi ou miroir d'un mot de V^*

Si un mot α de V^* s'écrit $a_1 a_2 \cdots a_k$, le mot réfléchi de α, noté $\tilde{\alpha}$ est : $a_k a_{k-1} \cdots a_2 a_1$.

1.3 - LANGAGE LOCAL ET LANGAGE REGULIER

Définition d'un langage local sur V : Soient D et F deux sous-ensembles de et t une partie de $V \times V$. Le langage local sur V est le langage formé des mots $a_1 a_2 \cdots a_n$ non vides qui possèdent les propriétés suivantes :

- $a_1 \in D$
- $a_n \in F$
- les couples (a_{i-1}, a_i) appartiennent à t pour : $2 \leq i \leq n$.

Graphe des transitions d'un langage local L sur V :

Soit le graphe, $g\ell = (V, \Gamma)$, défini par,

$$\forall x, y \in V \qquad x \Gamma y \iff (x, y) \in t$$

Le langage local L, d'ensemble initial D, d'ensemble final F, et d'ensemble de transitions t, est l'ensemble des chemins du graphe $g\ell$ dont l'origine appartient à D, et l'extrémité à F. On représentera généralement un tel langage local par le quadruplet : $(V, D, F, g\ell)$. Le graphe $g\ell$ est appelé graphe des transitions de L.

Langage régulier (ou langage de Kleene)

Nous ne rappellerons pas les diverses définitions des langages réguliers, car nous nous intéresserons plus particulièrement aux langages réguliers considérés comme transcrits de langages locaux; on sait en effet (voir [3]) :

Les langages réguliers ne contenant pas le mot vide sont les langages transcrits des langages locaux.

On peut ainsi caractériser les langages réguliers, comme étant les transcrits des langages locaux, éventuellement réunis au mot vide.

- LANGAGE DE PARENTHESES

Soit un ensemble V ; ses éléments sont considérés comme des parenthèses ouvrantes. A chacune d'elles, a, on associe une parenthèse fermante, $\bar{a}$, n'appartenant pas à V.

On appelle, $\bar{V}$, l'ensemble de ces parenthèses fermantes ($W = V \cup \bar{V}$).

Soit, 0, un élément n'appartenant pas à V^*.

Définition : Soit ι l'application du monoïde libre V^* dans $V^* \cup \{0\}$ définie par récurrence : $\forall a \in V$,

- $\iota(\Lambda) = \Lambda$ (Λ est le mot vide de V^*)
- si $\iota(\alpha) \neq 0$, alors $\iota(\alpha a) = \iota(\alpha)\, a$, sinon $\iota(\alpha a) = 0$
- si $\iota(\alpha) = \beta a$, alors $\iota(\alpha\bar{a}) = \beta$
- si $\iota(\alpha) = 0$ ou si $\iota(\alpha)$ est un mot non terminé par a, alors $\iota(\alpha\bar{a}) = 0$.

L'ensemble $P(V)$ des mots α sur W tels que : $\iota(\alpha) = \wedge$, s'appelle langage de parenthèses déduit de V. On énoncera sans les démontrer quelques propriétés de l'application ι, et du langage de parenthèses ([4]).

Propriétés :

Soient α, α', β des mots sur W.

1) $\iota(\alpha) = \iota(\alpha') \Rightarrow \iota(\alpha\beta) = \iota(\alpha'\beta)$
2) $\iota(\alpha) = 0 \Rightarrow \iota(\alpha\beta) = 0$
3) $\iota(\alpha) \neq 0$ et $\iota(\beta) \neq 0 \Rightarrow \iota(\alpha\beta) = \iota(\alpha)\,\iota(\beta)$
4) Soit α un mot non vide appartenant à $P(V)$.
Il existe $a \in V$, $\beta \in P(V)$, $\gamma \in P(V)$ uniques et tels que :
$\alpha = a\,\beta\,\bar{a}\,\gamma$.
5) $\iota(\iota(\alpha)) = \iota(\alpha)$

Lemme 1 : Soient α et β des mots sur W, alors $\iota(\alpha\beta) = \iota[\iota(\alpha)\beta]$

Démontrons-le par récurrence sur la longueur de β.

Pour β appartenant à W, ($|\beta| = 0$), le lemme traduit exactement la définition de l'application ι. Supposons le démontré pour des mots de longueur inférieure à n.

Soit β un mot de W^*, de longueur égale à n : $\beta = \beta' b$, $b \in W$;

$\iota(\alpha\beta) = \iota(\alpha\beta' b) = \iota(\iota(\alpha\beta')b)$, mais β' étant de longueur égale à $(n-1)$:

$\iota(\alpha\beta') = \iota[\iota(\alpha)\beta']$.

$\iota(\alpha\beta) = \iota[\iota(\iota(\alpha)\beta')b] = \iota(\iota(\alpha)\beta' b) = \iota(\iota(\alpha)\beta)$.

Lemme 2 : $\forall \alpha, \beta \in W^*$, $\forall a, b \in V$: $\alpha a \bar{b} \beta \in P(V) \Rightarrow a = b$.

Démonstration : Soit $m = \alpha a \bar{b} \beta$, un mot de $P(V)$.

En particulier : $\iota(m) = \wedge \neq 0$.

$\iota(\alpha a \bar{b} \beta) = \iota[\iota(\alpha a \bar{b})\beta]$, et $\iota(\alpha a \bar{b}) = \iota[\iota(\alpha a)\bar{b}]$, mais $\iota(\alpha a) \neq 0$, car sinon on $\iota(m) = 0$; on peut donc écrire, $\iota(\alpha a) = \alpha' a$, α' appartenant à V^*.

La condition nécessaire pour que m puisse appartenir à $P(V)$ est l'égalité, car sinon : $\iota(\alpha a \bar{b}) = \iota(\alpha' a \bar{b}) = 0 \Rightarrow \iota(m) = 0$.

La proposition suivante nous sera utile au chapitre 2.

Proposition 1 : Soit K un langage régulier sur $W = V \cup \bar{V}$, et L^- le langage local sur W, admettant comme ensemble de transitions :
$\{ab, \bar{a}\bar{b}, a\bar{a} ; \forall a, b \in V\}$.
Alors le langage régulier sur W, $K^- = K \cap L^-$ vérifie :
$P(V) \cap K = P(V) \cap K^-$.

Démonstration : La proposition est une conséquence du lemme 2 dont on déduit l'égalité : $P(V) = P(V) \cap L^-$.

Transcriptions entre monoïdes libres et transcriptions de parenthèses :

Toute application g de V dans V' se prolonge en un homomorphisme unique, g^* du monoïde libre V^* dans le monoïde libre V'^* :

$$\forall a_i \in V, \quad \text{pour } i = 1, 2, \ldots, n, \quad g^*\left[\prod_{i=1}^{n} a_i\right] = \prod_{i=1}^{n} g(a_i)$$

Nous dirons qu'un tel homomorphisme est une transcription entre monoïdes libres.

Toute transcription, g^*, d'un monoïde libre V^* dans un monoïde libre V'^*, peut être prolongée en un homomorphisme du langage de parenthèses $P(V)$ dans le langage de parenthèses $P(V')$ en convenant : $\forall a \in V$, $g(\bar{a}) = \overline{g(a)}$.

Une telle transcription sera dénommée transcription de parenthèses.

PILE

Définition [1] : Soit un ensemble V. On appelle pile sur V toute suite finie $U = (u_0, u_1, \ldots, u_n)$ d'éléments de V^* telle que :

1) $u_0 = u_n = \wedge$
2) pour $i = 1, \ldots, n$ il existe $a_i \in V$ tel que $u_i = u_{i-1} a_i$ ou $u_{i-1} = u_i a_i$; dans le premier cas, i est une entrée de a_i (a_i est le sommet de u_i), dans le second i est une sortie de a_i (a_i est le sommet de u_{i-1}).

Les mots $u_0, u_1, \ldots, u_n$ sont les états de la pile.

Remarque :

L'application, ι, associe à tout mot $\alpha = \alpha_1 \cdots \alpha_n$, appartenant à P(V) la pile, $(\iota(\alpha_0), \iota(\alpha_1), \ldots, \iota(\alpha_n))$, de façon biunivoque, si l'on pose : $\alpha_0 = \wedge$ et $\alpha_i = x_1 \cdots$

Nous utiliserons le théorème suivant, [1], dans le chapitre 3.

Théorème 1 : Soit u_i un état non vide d'une pile U, et j le plus petit entier supérieur à i tel que : $|u_j| < |u_i|$
a) pour $i \leq k < j$, u_i est facteur gauche de u_k.
b) j est une sortie du sommet de u_i.

Pile simple : Une pile U sur un ensemble fini V est simple lorsque tout élément de V possède une entrée et une seule dans U.

Les piles simples jouent un rôle important dans l'étude des "piles attachées aux ramifications" ([1]).

Mot lié à une pile :

Soit $U = (u_0, \ldots, u_n)$ une pile sur V ; associons à U une suite de mots sur V de la manière suivante :

$$m_0 = \wedge$$

pour $i = 1, 2, \ldots, n$ $\begin{cases} m_i = m_{i-1}\, e_i & \text{si i est une entrée de } e_i, \\ m_i = m_{i-1} & \text{si i est une sortie de } e_i, \end{cases}$

Le mot $m = m_n$ est appelé mot d'entrée de la pile U.

1.6 - RAMIFICATIONS SUR V

Rappelons brièvement que les ramifications sur V ([2]) peuvent être définies comme des suites finies de pseudo-arborescences sur V. L'ensemble des ramifications sur V c'est à dire le monoïde libre déduit de l'ensemble, $\mathcal{A}(V)$, des pseudo-arborescences sur V sera noté $\hat{V}$. La loi de composition de ce monoïde sera notée + ; l'élément neutre de cette loi sera nommée ramification vide et noté $\wedge$.

La figure 1 schématise une ramification sur $V = \{a, b, c, f, e\}$

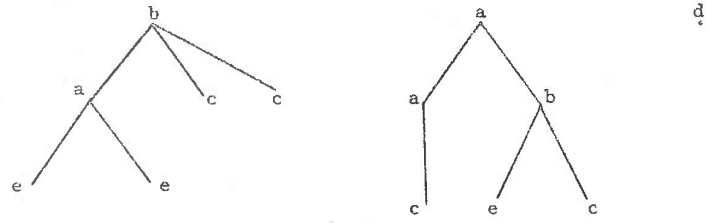


Figure 1

Une approche plus intéressante de la notion de ramification peut être faite par celle de binoïde.

Binoïde sur V : C'est un ensemble muni,
1) d'une loi de composition interne associative, + , avec élément neutre, noté $\wedge$.
2) d'une loi de composition externe, X , à opérateurs dans V. On peut donner une définition axiomatique de $\hat{V}$ ([4]) qui est d'un emploi facile dans l'étude des ramifications.

Axiomes :
1) $\hat{V}$ est un binoïde sur V
2) Soit $r \neq \wedge$ une ramification de $\hat{V}$; il existe $a \in V$, $s \in \hat{V}$, $t \in \hat{V}$ uniques tels que : $r = a \chi s + t$
3) Il existe une application ν de $\hat{V}$ dans $\mathbb{N}$(*) telle que :
   - $\nu(\wedge) = 0$
   - $\nu(r+s) > \nu(r)$ si $r \neq \wedge$, $\nu(r+s) \geq \nu(s)$ si $s \neq \wedge$
   - $\nu(a \chi r) > \nu(r)$.

En utilisant les lois + et X , la ramification de la figure 1 peut se décomposer comme suit :

$$\{b \chi [a \chi (e+e) + c + c]\} + \{a \chi [(a \chi c) + b \chi (e+c)]\} + d$$

Principe de récurrence : On définit dans $\hat{V}$ un principe de récurrence qui permet de démontrer de nombreuses propriétés sur les ramifications :

(*) Nota : On notera toujours $\mathbb{N}$ pour désigner l'ensemble des entiers naturels.

- Soit P un prédicat tel que :
  1) $P(\wedge)$ est vrai
  2) $(\forall r, s \in \hat{V})$ $P(r)$ et $P(s) \Longrightarrow (\forall a \in V)$ $P(r+axs)$, alors $P(r)$ est vrai pour tout $r \in \hat{V}$.

En particulier il nous permet de définir par récurrence un certain nombre d'applications :

. Appelons mot des feuilles d'une ramification r de $\hat{V}$ le mot $\varphi(r)$ sur V tel que l'application φ de $\hat{V}$ dans V^* vérifie : $\forall r, s \in \hat{V}$, $\forall a \in V$

- $\varphi(\wedge) = \wedge$; $\varphi(r+s) = \varphi(r) . \varphi(s)$
- $\varphi(axr) =$ SI $\varphi(r) \neq \wedge$ ALORS $\varphi(r)$ SINON a

. Appelons mot des racines de r le mot $\rho(r)$ de V^* défini par :
$\forall r, s \in \hat{V}$; $\forall a \in V$; $\rho(\wedge) = \wedge$; $\rho(r+s) = \rho(r)\rho(s)$; $\rho(axr) = a$

. Soit a un élément de V, et F_a l'application de $\hat{V}$ dans l'ensemble $P(V^*)$ des parties de V^*, définie par :

- $F_a(\wedge) = \emptyset$, $F_a(r+s) = F_a(r) \cup F_a(s)$
- $F_a(bxr) =$ SI $b=a$ ALORS $F_a(r) \cup \{\rho(r)\}$ SINON $F_a(r)$

. Par définition $F_a(r)$ est l'ensemble des familles de prédécesseur a dans r.

- Chaîne (ou rangée) d'une ramification :

Soit ch l'application de $\hat{V}$ dans $P(V^*)$ définie par :

$\forall a \in V$, $\forall r, s \in \hat{V}$, $ch(\wedge) = \wedge$
$ch(axr) = a . ch(r) = \{am \mid m \in ch(r)\}$
$ch(r+s) = ch(r) \cup ch(s)$

Pour toute ramification r de $\hat{V}$, $ch(r)$ représente l'ensemble des chaînes (ou rangées) de r.

Dans l'exemple de la figure 1 :

$\rho(r) =$ bad ; $\varphi(r) =$ eecccec ; $F_a(r) = \{ec, ab, e\}$

$ch(r) = \{bae ; bc ; aac ; abe ; abc ; d\}$

Transcription d'une ramification : Soit V' un ensemble fini, et g une application de V dans V'. L'application $\hat{g}$ de $\hat{V}$ dans $\hat{V}'$ définie ci-dessous remplace toute "étiquette", a appartenant à V, d'une ramification r de $\hat{V}$ par $g(a)$ appartenant à V' :

$\hat{g}(\wedge) = \wedge$; $\hat{g}(axs+t) = g(a) x \hat{g}(s) + \hat{g}(t)$.

L'application $\hat{g}$ s'appelle transcription de $\hat{V}$ dans $\hat{V}'$, associée à l'application g.

Ensemble de ramifications, $R(T, N)$: Appelons noeud d'une ramification sur V tout élément de V qui est prédécesseur d'une famille non vide. L'étude des grammaires nous conduira à étudier des ramifications dont les feuilles et les noeuds appartiennent à des ensembles disjoints. Soient T et N deux ensembles disjoints ; nous désignerons par $R(T, N)$, l'ensemble des ramifications sur la réunion V, de T et N, dont les feuilles appartiennent à T et les noeuds à N.

Binoïde universel : Un binoïde universel sur un ensemble V (ou engendré par V) est un ensemble $\hat{V}$, qui vérifie les axiomes 1, 2, 3.
Si B et B' sont deux binoïdes sur un même ensemble V, l'homomorphisme de B dans B' est l'application ψ de B dans B', telle que pour r et s dans B, et A dans V :

$\psi(\wedge) = \wedge$; $\psi(r+s) = \psi(r) + \psi(s)$; $\psi(Axr) = A x \psi(r)$.

On démontre que les binoïdes universels sur V sont tous isomorphes (isomorphisme de binoïdes).
Un binoïde universel sur un alphabet V est ainsi toujours défini à un isomorphisme près. C'est une conséquence directe de la proposition suivante :

- Soit B un binoïde sur V. Il existe un, et un seul, homomorphisme de binoïdes sur V, de $\hat{V}$ dans B.

1.7 - GRAMMAIRES DE CHOMSKY

Nous appellerons grammaire de Chomsky (ou simplement grammaire) un tri
$(V, \rightarrow, X)$, où :

- V est un ensemble fini, appelé alphabet de la grammaire, qui est la réunio de deux sous-ensembles disjoints, $(V=N \cup T)$, l'alphabet terminal T et l'al bet non terminal (ou auxiliaire) N.
- $\rightarrow$ est une relation binaire entre N et V^* (relation de production) telle que pour tout a de N, les mots α vérifiant $a \rightarrow \alpha$ forment un langage régulie contenant pas le mot vide ; ce langage régulier est nommé langage des prod tions de a, et on le note K_a.
- X est un langage régulier ne contenant pas le mot vide (ensemble des axiom

Grammaire locale :
Une grammaire, $G = (N \cup T, \rightarrow, X)$, est dite locale si les langages de production, pour tout a de N, et le langage X sont des langages locaux possédant le mêm ensemble de transitions.

Règles d'une grammaire :

Etant donnés un symbole auxiliaire A (élément de l'alphabet auxiliaire N) d'u grammaire, $G = (N \cup T, \rightarrow, X)$, et un mot φ de V^* qui vérifient : $A \rightarrow \varphi$, on app règle de la grammaire G la donnée d'un tel couple en relation (A, φ) ; on préf quelque fois la notation, $A ::= \varphi$, pour désigner une règle de grammaire.

Exemple : Soient les ensembles $N = \{X, A, C\}$ et $T = \{a, b, c\}$ la grammaire d'axiome X, admettant les règles suivantes :

$$X ::= AC \mid C$$
$$A ::= aAb \mid ab$$
$$C ::= cC \mid c$$

engendre le langage $L = \{a^n b^n c^p \mid n \geq 0, p > 0\}$.

Dérivation :

Etant donnée la grammaire, $G = (N \cup T, ::=, X)$, on lui associe la relation binair $\underset{G}{\succ\!\!\!-\!\!\!-}$, dans V^* définie par :

$$\alpha \underset{G}{\succ\!\!\!-\!\!\!-} \beta \iff (\exists A \in N, \lambda \in V^*, X \in V^*, \varphi \in V^*)(\alpha = \lambda A \lambda' \text{ et } \beta = \lambda \varphi \lambda' \text{ et } A::$$

Lorsqu'il existe une suite de mots de V^* $\delta_0, \delta_1, \ldots, \delta_n$ $(n \geq 0)$ telle que $\delta_0 = \alpha$, $\delta_n = \beta$ et

$$\delta_0 \underset{G}{\succ\!\!\!-\!\!\!-} \delta_1 \underset{G}{\succ\!\!\!-\!\!\!-} \cdots\cdots \underset{G}{\succ\!\!\!-\!\!\!-} \delta_n$$

on dit que α dérive de β.

Une telle suite est appelée une dérivation.
Lorsque $\delta_i \in N$ $(i=0, \ldots, n)$, et $\delta_0 = \delta_n = A$, avec $A \in N$ on dit que le symbole auxiliaire A dérive strictement de lui-même.

Initiales et finales
Soit une grammaire $(V, ::=, X)$, et un élément Y de N $(V = N \cup T)$. On dit que $Z \in V$ est une initiale de Y s'il existe un mot φ de V^* tel que $Z\varphi$ dérive de Y. On dit que $Z' \in V$ est une finale de Y s'il existe un mot φ' de V^* tel que $\varphi' Z'$ dérive de Y.
La relation "Z est une intiale de V" est la fermeture transitive ([8]) de la relation in définie dans V par :

$$Z \text{ in } Y \iff (\exists \psi \in V^*)\ (Y ::= Z\psi).$$

"Z est une finale de Y" est la fermeture transitive de fn :

$$Z \text{ fn } Y \iff (\exists \psi \in V^*)\ (Y ::= \psi Z)$$

Le graphe (V, in) (respectivement (V, fn)) sera nommé graphe des initiales (respectivement graphe des finales) de la grammaire.

BILANGAGES

Une partie de $\hat{V}$ est un "langage à deux dimensions", ce qu'on peut appeler un bilangage.

1.8.1 - Bilangages grammaticaux

Une ramification r sur V est engendrée par une grammaire $G = (V, \mapsto, X)$ si, dans r: avec $V = N \cup T$,

a) pour toute famille α de prédécesseur a appartenant à N, $a \rightarrow \alpha$;
b) toute famille de prédécesseur a appartenant à T est vide ;
c) le mot des racines, $\rho(r)$, appartient à X.

L'ensemble des ramifications engendrées par une grammaire G, noté S(G) est appelé bilangage engendré au sens strict, (ou simplement, engendré) par G.
Les bilangages engendrés par une grammaire sont dits bilangages grammaticaux
L'ensemble S(G) appartient à R(T, N).
L'ensemble des mots des feuilles des ramifications de S(G) s'appelle langage engendré par la grammaire G.
Les langages engendrés par une grammaire sont les langages de Chomsky ou C-langages.

1.8.2 - Bilangages réguliers

La notion d'homomorphisme de binoïdes permet de généraliser la notion de langa régulier en celle de bilangages réguliers (référence [2]).
Un bilangage L sur V est dit régulier s'il existe un binoïde fini B, une partie de B tels que, si ψ est l'homomorphisme de $\hat{V}$ dans B, $L = \psi^{-1}(B')$.
Nous dirons que L est associé au triplet (B, B', ψ).

Bilangage local : Etant donné trois sous-ensembles de V^2 : τ, d, f et trois sous-ensembles de V : D, F et ϕ, on appelle bilangage local défini par τ, d, f, D, F, ϕ le bilangage grammatical S(G) engendré par la grammaire $G = (V, \rightarrow, X)$ où :

- pour tout A de V, l'ensemble des productions de A est le langage local, $[V, \{B \mid (A, B) \in d\}, \{B \mid (A, B) \in f\}, \tau]$, réuni à $\{\wedge\}$ si A appartient à
- X est le langage local $[V, D, F, \tau]$.

Il est démontré dans [2], les propositions suivantes :

1) Les bilangages réguliers ne contenant pas la ramification vide sont les transformées par transcription des bilangages locaux et eux seuls.
2) Tout bilangage régulier est déduit par transcription d'un bilangage grammatical
3) Tout bilangage grammatical ne contenant pas la ramification vide est déduit par transcription d'un bilangage local.
4) L'image par une transcription d'un bilangage régulier est un bilangage régulier

Bilangage régulier d'un binoïde universel sur un alphabet V

Les bilangages ont été définis comme les parties d'un binoïde universel sur un alphabet fini V; celui-ci étant défini à un isormorphisme près les bilangages réguliers d'un binoïde universel sur V, B, seront considérés, au chapitre suivant, comme les images des bilangages réguliers de $\hat{V}$, par l'isomorphisme de $\hat{V}$ dans B. En particulier on emploiera une définition similaire pour les bilangages locaux.

CHAPITRE II

Etude de l'analyse syntaxique par ≪les structures compatibles≫ de binoïde universel sur les langages de parenthèses

INTRODUCTION

Soit une grammaire, $G = (V, \rightarrow, X)$, d'alphabet $V = N \cup T$.
Effectuer l'analyse syntaxique, d'un mot α sur l'alphabet terminal T, pour la grammaire G, c'est déterminer les ramifications engendrées par G, dont α est le mot des feuilles, autrement dit l'ensemble :

$$E = \varphi^{-1}(\alpha) \cap S(G)$$

Cependant, le binoïde universel $\hat{V}$, auquel appartiennent les ramifications engendrées par G, S(G), est déterminé à un isomorphisme de binoïde près ; c'est pourquoi la résolution pratique de l'analyse syntaxique dépendra du choix de ce binoïde.
Nous allons étudier dans ce qui suit des représentations de $\hat{V}$, par le langage de parenthèses P(V) muni de certaines lois, pour lesquelles la recherche de l'ensemble E se ramène à la construction d'algorithmes, relativement simples, qui construisent des mots de P(V).
Les lois de compositions définies sur l'ensemble des ramifications, $\hat{V}$, seront toujours notées : $\rightarrow$, pour la loi interne et $\uparrow$, pour la loi externe afin d'éviter toute confusion avec celles du langage P(V).

ETUDE DU BINOIDE UNIVERSEL PARENTHETIQUE SUR UN ALPHABET V

2.1.1 - Définition 1 :

Munissons le langage de parenthèses P(V) d'un couple, $S_1 = (+, \times)$, de lois de composition définies ci-dessous,

- une loi de composition interne notée, + :

$$\forall \beta, \gamma \in \Gamma(V) \qquad \beta + \gamma = \beta\gamma$$

- une loi de composition externe à opérateurs dans V, notée $\times$:

$$\forall a \in V, \ \forall \beta \in P(V) \qquad a \times \beta = a\beta\bar{a}$$

Il est démontré dans PAIR [4], que S_1 définit sur $P(V)$ une structure de binoïde universel, que nous appellerons le binoïde universel parenthétique et que nous noterons : $P_1(V) = (P(V), S_1)$.

Nous désignerons par, p_1, l'isomorphisme de binoïdes, unique, de $\hat{V}$ dans P

- $\forall r, r' \in \hat{V}$: $\quad p_1(r \rightarrow r') = p_1(r)\ p_1(r')$
- $\forall A \in V$, $\forall r \in \hat{V}$: $\quad p_1(A \uparrow r) = A p_1(r)\ \bar{A}$

2.1.2 - Nature des transcriptions de binoïdes parenthétiques

Propriété 1 : Les transcriptions du binoïde universel $P_1(V)$ dans le binoïde universel $P_1(V')$, définies à partir d'une application, g, de V dans V' (que nous avions notés $\hat{g}$ au chapitre 1) sont les transcriptions de parenthèses g^*.

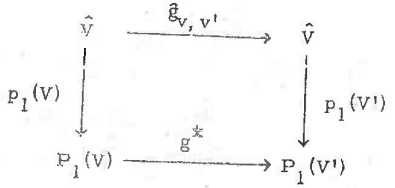
Démonstration : L'homomorphisme, g^*, de $P(V)$ dans $P(V')$ verifie,

- $g^*(\wedge) = \wedge$
- $\forall \alpha, \beta \in P_1(V) : g^*(\alpha + \beta) = g^*(\alpha\,\beta) = g^*(\alpha)\, g^*(\beta) = g^*(\alpha) + g^*(\beta)$
- $\forall A \in V$, $\forall \alpha \in P_1(V) : g^*(A \times \alpha) = g^*(A\,\alpha\,\bar{A}) = g(A)\, g^*(\alpha)\, \overline{g(A)} = g(A) \times g^*(\alpha)$

D'après le "principe de récurrence", l'homomorphisme g^* est identique à l'unique transcription de binoïdes, de $P_1(V)$ dans $P_1(V')$, $\hat{g}$, que nous avions définie au chapitre 1 à partir d'une application, g, de V dans V'.

Remarque 1 :

Cette propriété traduit en fait la commutativité du diagramme,



$\hat{g}_{V, V'}$ étant la transcription de $\hat{V}$ dans $\hat{V}'$, définie à partir de l'application, g, de V dans V'

$$P_1(V') \circ \hat{g}_{V, V'} = g^* \circ p_1(V)$$

2.1.3 - Mot des feuilles des ramifications de $R(T, N)$

Propriété 2 : Soit $V = T \cup N$, T et N étant deux ensembles disjoints ; alors le mot des feuilles, $\varphi(r)$, pour toute ramification, r, de $R(T, N)$ est égal à $h 0 p_1(r)$, où h est l'homomorphisme alphabétique de $(V \cup \bar{V})^*$ dans T^*, défini par : $\forall a \in V$

$h(\bar{a})$ = SI $a \in T$ ALORS a SINON $\wedge$

$h(a) = \wedge$.

Démonstration :

- $h 0 p_1(\wedge) = \wedge$
- $\forall r', r'' \in R(T, N) : h 0 p_1(r' \rightarrow r'') = h[p_1(r')\ p_1(r'')] = h 0 p_1(r')\, h 0 p_1(r'')$
- $\forall A \in V$, $\forall r \in R(T, N) : h 0 p_1(A \uparrow r) = h[A p_1(r) \bar{A}] = h 0 p_1(r) h(\bar{A})$

Si $h 0 p_1(r) \neq \wedge$, en particulier r est différent de $\wedge$ et A appartient à N, puisque $A \uparrow r$ appartient à $R(T, N)$: $h 0 p_1(A \uparrow r) = h 0 p_1(r)$.

Si $h 0 p_1(r) = \wedge$, alors $r = \wedge$ et par conséquent A appartient à T :

$$h 0 p_1(A \uparrow r) = h(\bar{A}) = A$$

D'après le "principe de récurrence" (chapitre 1), on a l'identité :

$$\forall r \in R(T, N) \qquad h 0 p_1(r) = \varphi(r)$$

2.1.4 - Recherche des bilangages réguliers de $P_1(V)$

Théorème 1 : Les bilangages réguliers de $P_1(V)$ sont les transcrits par une transcription de parenthèses, g^*, (définie à partir d'une application g de V' dans V) de l'intersection d'un langage régulier sur $V' \cup \bar{V}'$ avec le langage de parenthèses $P(V')$.

Démonstration

1) Soit L un bilangage régulier de $P_1(V)$, non vide ; il existe un bilangage local L' de $P_1(V')$, tel que L est le transcrit de L' par une transcription g^*, g étant une application de V' dans V (paragraphe 1.8.2 et propriété 1, 2.1.2) : $L = g^*(L')$.

Soit L'_1, le bilangage local de $\hat{V}'$ tel que : $L' = p_1(L'_1)$.

Par définition, ce bilangage local est en fait un bilangage grammatical, S(G), engendré par une grammaire locale, $G = (V', ::=, X)$.

Il est, en outre, démontré dans PAIR [4], que l'ensemble des représentation des ramifications engendrées par une grammaire locale, G, dans le binoïde parenthétique (sur son alphabet) est l'intersection d'un langage régulier local a le langage de parenthèses qui constitue ce binoïde, soit :

$$p_1[S(G)] = L_1 \cap P(V') \quad , \quad L_1 \text{ langage régulier local sur } V' \cup \bar{V}'.$$

On en déduit :

$$L = g^*(L') = g^* \circ p_1(L'_1) = g^* \circ p_1[S(G)] = g^*(L_1 \cap P(V')).$$

2) Soit K un langage régulier sur $V' \cup \bar{V}'$ et, g^*, une transcription de parenth de $P_1(V')$ dans $P_1(V)$. Il suffit de démontrer que $K \cap P(V')$ est un bilangage régulier de $P_1(V')$. (propositions 1.8.2).

K étant un langage régulier, il existe un monoïde fini M, une partie M' de M et un homomorphisme de monoïdes, ψ de $(V' \cup \bar{V}')^*$ dans M tels que : $K = \psi^{-1}(M$

Munissons le monoïde M, des lois $\dot{+}$ et $\dot{\times}$, définies par :

- $\forall u \in M$ et $\forall a \in V'$ $\quad a \dot{\times} u = \psi(a)\, u\, \psi(\bar{a})$
- $\forall u, v \in M$ $\quad u \dot{+} v = u \,.\, v$

Le triplet $(M, \dot{+}, \dot{\times})$ définit une structure de binoïde, B, sur M :

- la loi , . ,(ou $\dot{+}$) sur M est une loi interne associative à élément neutr puisqu'elle définit sur M une structure de monoïde.
- la loi , $\dot{\times}$, est une loi externe dans M à opérateurs dans V'.

La restriction à P(V') de ψ, $\psi_{P(V')}$, est un homomorphisme de binoïdes de $P_1(V')$ dans B :

- $\forall \alpha \in P_1(V')$, $\forall a \in V'$, $\quad a \times \alpha = a\, \alpha\, \bar{a}$:

$$\psi(a \times \alpha) = \psi(a)\, \psi(\alpha)\, \psi(\bar{a}) = a \dot{\times} \psi(\alpha)$$

- $\forall \alpha, \beta \in P_1(V')$, $\alpha + \beta = \alpha\beta$:

$$\psi(\alpha + \beta) = \psi(\alpha\beta) = \psi(\alpha)\, \psi(\beta) = \psi(\alpha) \dot{+} \psi(\beta)$$

De plus :

$$\begin{aligned} K \cap P(V') &= \{ x \in P(V') \cap \psi^{-1}(M') \} \\ &= \{ x \in P(V') \mid \psi(x) \in M' \} \\ &= \{ x \in P(V') \mid \psi_{P(V')}(x) \in M' \} = \psi^{-1}_{P(V')}(M') \end{aligned}$$

Le triplet $(B, M', \psi_{P(V')})$ détermine une structure de bilangage régulier sur $P_1(V')$: le bilangage $K(V') \cap P(V')$.

DEFINITION DES STRUCTURES COMPATIBLES D'UN BINOIDE UNIVERSEL SUR LES LANGAGES DE PARENTHESES

Les trois propriétés que nous avons élaborées dans l'exemple du binoïde parenthétique, va nous permettre de définir ce que nous appellerons ≪les structures compatibles≫.

2.2.1 - Définition 2 :

> Soit V un ensemble fini. Un couple, S, d'une loi interne et d'une loi externe à opérateurs dans V définissant une structure de binoïde universel sur un langage de parenthèses P(V) est dite ≪compatible≫ si l'on a les trois propriétés :
>
> a) Pour toute application g de V dans V' la transcription de parenthèses, g^*, de $(V \cup \bar{V})^*$ dans $(V' \cup \bar{V}')^*$ est la transcription du binoïde universel (P(V), S) dans le binoïde universel (P(V'), S), définie à partir de g.
>
> b) La restriction à R(T, N) du mot des feuilles des ramifications de $\hat{V}, \varphi$, où T et N sont deux ensembles disjoints tels que $V = T \cup N$ est la restriction à P(V) d'un homomorphisme alphabétique de $(V \cup \bar{V})^*$ dans T^*.
>
> c) Les bilangages réguliers de (P(V), S) sont les transformés par les transcriptions de parenthèses de (P(V'), S) dans (P(V), S) de l'intersection d'un langage régulier sur $V' \cup \bar{V}'$ et du langage de parenthèses P(V').

De telles structures existent, puisque nous en avons exhibé une au paragraphe 2.1, la structure parenthétique S_1.

2.2.2 - Analyse syntaxique et ≪structure compatible≫

Soit, $B = (P(V), S)$, un binoïde universel de parenthèses, S étant une structur compatible de binoïde universel sur P(V). Proposons nous d'étudier plus préc sément l'ensemble $E = \varphi^{-1}(\alpha) \cap S(G)$, dont nous parlions dans l'introduction en choisissant le binoïde universel B comme représentation de $\hat{V}$. Un tel choix n conduit non pas à déterminer E, mais son transformé par l'isomorphisme de binoïdes, p, de $\hat{V}$ dans B ; c'est à dire l'ensemble

$$p[S(G) \cap \varphi^{-1}(\alpha)] = p[S(G)] \cap p[\varphi^{-1}(\alpha)].$$

Le bilangage grammatical, S(G), de $\hat{V}$ est un bilangage régulier de $\hat{V}$ (1.8.2, chapitre 1), et par conséquent $p[S(G)]$ est un bilangage régulier de B.
S, étant ≪une structure compatible≫ sur P(V),une propriété de la définition 2 (§ 2.2.1) nous indique la nature des bilangages réguliers de B :

> il existe un ensemble V', une application g de V' dans V et un langage régulier K sur $V' \cup \bar{V}'$ tels que : $p(S[G]) = g^*(K \cap P(V'))$, où g^* est l'homo morphisme alphabétique de $(V' \cup \bar{V}')^*$ dans $(V \cup \bar{V})^*$ déduit de g.

En outre, on peut déterminer un homomorphisme alphabétique, h, de $(V \cup \bar{V})^*$ dans T^* tel que :

$$\varphi[S(G)] = h \circ p[S(G)]$$
$$\varphi^{-1}(\alpha) = (h \circ p)^{-1}(\alpha) = p^{-1} \circ h^{-1}(\alpha)$$
$$p \circ \varphi^{-1}(\alpha) = p \circ p^{-1} \circ h^{-1}(\alpha) = h^{-1}(\alpha) \quad (p \text{ est un isomorphism}$$

Une résolution possible du problème de l'analyse syntaxique par ≪les structure compatibles sur P(V)≫ peut, ainsi, se formaliser par :

> Etant données une grammaire $G = (V, \rightarrow, X)$ (d'alphabet $V = T \cup N$, T et N étan disjoints, et T l'alphabet terminal), et ≪une structure compatible≫ S sur P(V), il faut associer à tout mot, α, sur l'alphabet terminal T de G, l'ensemble :

$$(I) \qquad \mathcal{L}(\alpha) = h^{-1}(\alpha) \cap g^*(K \cap P(V')),$$

les ensembles V', K, g^* étant définis à partir de S comme précédemment.

La recherche de différentes ≪structures compatibles≫ sur les langages de parenthèses P(V), nous fournissant autant de méthodes d'analyse, distinctes, constitue ainsi la première étape de la mise en oeuvre d'un système d'analyse. Elle est en fait la finalité de ce chapitre. La seconde étape consistera pour chaque structure compatible, en l'association à toute grammaire $G = (V, \rightarrow, X)$ d'un quadruplet, (V', g, K, h), tel que

$$p[S(G)] = g^*(K \cap P(V')) \quad \text{et} \quad \forall r \in S(G) \ \ \varphi(r) = h \circ p(r)$$

Cette association sera développée au chapitre 3 à partir des résultats obtenus dans ce chapitre.
Il faut noter que pour une structure donnée, on peut associer à toute grammaire, G, plusieurs de ces quadruplets. Ainsi, connaissant un tel quadruplet, on pourra lui préférer le quadruplet (V', g, K^-, h), dans lequel le langage régulier K^- est l'intersection de K et du langage local sur $V' \cup \bar{V}'$, qui admet pour ensemble de transitions, $\{ ab, \bar{a}\bar{b}, a\bar{a} \ ; \ \forall a, b \in V' \}$ (proposition 1, paragraphe 1.4, chapitre 1), et l'on a l'égalité :

$$K^- \cap P(V') = K \cap P(V')$$

ETUDE DE QUELQUES ≪STRUCTURES COMPATIBLES≫

2.3.1 - Analyse descendante parenthétique

Nous avons étudié au paragraphe 2.1 une première ≪structure compatible≫ de binoïde universel sur les langages de parenthèses. Il s'agit dans ce paragraphe de mettre en évidence certains des quadruplets associés à cette structure, que nous avions nommée, S_1.
Nous utiliserons essentiellement, les deux résultats suivants :

1) - PAIR [4] - L'ensemble des représentations des ramifications engendrées par une grammaire locale, G, dans le binoïde universel parenthétique (sur un alphabet) est l'intersection d'un langage régulier local, avec le langage de parenthèses qui constitue ce binoïde, et l'on a : Soient D_0 et F_0 l'ensemble initial et final du langage X (axiome de G), D_A et F_A ceux des langages des productions de G, K_A $(A \in N)$, et I, l'ensemble des transitions communes à ces langages.

- $P_1(V)[S(G)] = L_1 \cap P(V)$, avec pour, L_1, le langage local d'ensemble initial D_0, d'ensemble final F_0 et d'ensemble de transitions,

$$I' = \Big(\bigcup_{A \in N} AD_A \Big) \cup \{ A\bar{A} \mid A \in T \} \cup \Big(\bigcup_{B \in N} F_B \bar{B} \Big) \cup \{ \bar{A}B \mid AB \in I \}$$

- $\varphi(r) = h \circ p_1(r) = h' \circ p_1(r) \quad \forall r \in S(G)$, où h est l'homomorphisme défini au paragraphe 2.1.3 et h', l'homomorphisme défini par, $h'(\bar{A}) = \Lambda$, $h'(A) =$ SI $A \in T$ ALORS A SINON Λ, pour tout A de $V = T \cup N$.

2) - A toute grammaire G, d'alphabet V, on peut associer une grammaire loca G', d'alphabet V', pour laquelle il existe une transcription, $\hat{g}$, du, binoïde universel $\hat{V}'$ dans le binoïde $\hat{V}$, qui transforme biunivoquement S(G') en S(G). G' et $\hat{g}$ peuvent être définies par :

. Soit $G = (V, \rightarrow, X)$ avec $V = T \cup N$, les langages réguliers, X et K_A (respectivement axiome et langages des productions), pour A appartenant à N, sont déduits biunivoquement de langages locaux L_0 et L_A par des transcriptions de monoïdes libres, t_0 et t_A, respectivement. On peut toujours supposer deux à deux disjoints les alphabets de L_0 et des L_A, en modifiant au besoin t_0 et les t_A. Soient V' la réunion de ces alphabets, D_0 et F_0 l'ensemble initial et final de L_0, D_A et F_A ceux des langages L_A, I la réunion des ensembles de transitions des langages L_A et L_0, $D = D_0 \cup \Big(\bigcup_{A \in N} D_A \Big)$, $F = F_0 \cup \Big(\bigcup_{A \in N} F_A \Big)$.

Définissons une application g de V' dans V :

. $g(x) = t_0(x)$, pour x appartenant à l'alphabet de L_0

(pour tout A de N) . $g(x) = t_A(x)$, pour x appartenant à celui de L_A.

. Il est démontré dans (PAIR [4]), que S(G) se déduit biunivoquement de S(G') par la transcription, $\hat{g}$, de $\hat{V}'$ dans $\hat{V}$ définie sur g, si G' est la grammaire définie par : $(g^{-1}(T) \cup g^{-1}(N), \rightarrow', L_0)$, où $g^{-1}(T)$ est l'alphabet terminal de G', $\rightarrow'$, la relation de production telle que : $B \rightarrow' \beta \iff \beta \in L_{g(B)}$, $\forall B \in g^{-1}(N)$.

. Nous pouvons alors nous intéresser aux quadruplets associés à S_1.

. Soit une grammaire G ; associons lui la grammaire locale G' comme en (2) :

$$p_1(V')[S(G')] = L_1 \cap P(V') \quad \text{et} \quad S[G] = \hat{g}(S[G']).$$

En utilisant la remarque 1 du paragraphe 2.1.2, nous avons :

$$p_1(V)[S(G)] = p_1(V) \circ \hat{g}[S(G')] = g^* \circ p_1(V')[S(G')] = g^*(L_1 \cap P(V'))$$

Nous obtenons ainsi deux quadruplets, (V', g, L_1, h) et (V', g, L_1, h') qui peuvent être associés à S_1.

La méthode d'analyse, ainsi dégagée à l'aide de S_1, correspond à une analyse effectuée en "dents de scie", qui procède de la racine vers les feuilles ; nous la dénommerons ≪analyse descendante parenthétique≫.

2.3.2 - Analyse descendante prédictive

2.3.2.1 - Le binoïde universel descendant gauche

Munissons le langage de parenthèses P(V), d'un couple, $S_3 = (+, \times)$, de lois de composition définies ci-dessous,

- une loi de composition interne notée, + :

$\forall \beta, \gamma \in P(V)$ $\beta + \gamma = \gamma_v(\gamma) \beta \Delta_v(\gamma)$, où $\gamma_v(\gamma)$ est le plus grand facteur gauche de γ qui appartient à V^*, et $\Delta_v(\gamma)$ le quotient à gauche de γ par $\gamma_v(\gamma)$: $\gamma = \gamma_v(\gamma) \Delta_v(\gamma)$

- une loi de composition externe à opérateurs dans V, notée $\times$:

$\forall a \in V$, $\forall \beta \in P(V)$ $\quad a \times \beta = a \bar{a} \beta$.

Il est démontré dans PAIR [4] que S_3 définit sur P(V) une structure de binoïde universel, que nous appellerons le binoïde universel descendant gauche et que nous noterons : $P_3(V) = (P(V), S_3)$. Nous désignerons par, p_3, l'isomorphisme de binoïdes, unique, de $\hat{V}$ dans $P_3(V)$:

- $\forall r, r' \in \hat{V}$ $\quad : p_3(r \rightarrow r') = \gamma_v[p_3(r')]\, p_3(r)\, \Delta_v[p_3(r')]$
- $\forall r \in \hat{V}$, $\forall A \in V$: $p_3(A\,;\, r) = A \bar{A} p_3(r)$.

Lorsqu'il n'y aura pas de confusion possible sur l'alphabet V employé, γ_v et Δ_v seront notés γ et Δ.

2.3.2.2 - Nature des transcriptions de binoïdes universels descendants gauches

Propriété 3 : Les transcriptions du binoïde universel $P_3(V)$ dans le binoï universel $P_3(V')$, définies à partir d'une application, g, de V dans V' (que nous avions notées $\hat{g}$ au chapitre 1) sont l transcriptions de parenthèses g^*.

Démonstration : L'homomorphisme, g^*, de $P(V)$ dans $P(V')$ vérifie,

- $g^*(\wedge) = \wedge$
- $\forall \alpha, \beta \in P_3(V)\quad g^*(\alpha + \beta) = g^*[\neg_v(\beta)\alpha\Delta_v(\beta)] = g^*(\neg_v(\beta))g^*(\alpha)g^*(\Delta_v(\beta))$

$$= g^*(\alpha) + g^*(\beta).$$

car les implications : $\neg_v(\beta) \in V^* \Longrightarrow g^*(\neg_v(\beta)) \in V'^*$,

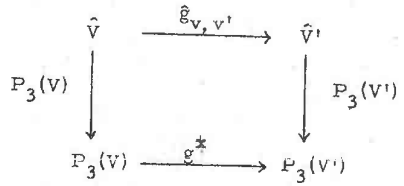
et $\{\Delta_v(\beta) = \bar{A}m,\ A \in V \text{ et } m \in (V \cup \bar{V})^*\} \Longrightarrow g^*(\Delta_v\beta) = \overline{g(A)}\, g^*(m)$,

entrainent les égalités, $g^*(\neg_v(\beta)) = \neg_{v'}(g^*(\beta))$ et $g^*[\Delta_v(\beta)] = \Delta_{v'}[g^*(\beta)]$

- $\forall A \in V,\ \forall \alpha \in P_3(V) : g^*(A \times \alpha) = g^*(A\bar{A}\alpha) = g(A)\,\overline{g(A)}\, g^*(\alpha) = g(A) \times g^*(\alpha)$,

D'après le "principe de récurrence" (chapitre 1), l'homomorphisme, g^*, est identique à l'unique transcription de binoïdes, $\hat{g}$, de $P_3(V)$ dans $P_3(V')$.

Remarque 2 : Cette propriété traduit la commutativité du diagramme :



soit $\quad p_3(V') \circ \hat{g}_{V,V'} = g^* \circ p_3(V)$.

2.3.2.3 - Mot des feuilles des ramifications de R(T, N)

Propriété 4 : Soit $V = T \cup N$, T et N étant deux ensembles disjoints ; alors le mot des feuilles, $\varphi(r)$, pour toute ramification, r, de $R(T, N)$ est égal à $h \circ p_3(r)$, où h est l'homomorphisme alphabétique de $(V \cup \bar{V})^*$ dans T^* défini par :

$\forall A \in V \quad h(\bar{A}) = SI\ A \in T\ ALORS\ A\ SINON\ \wedge$

$\forall A \in V \quad h(A) = \wedge$

Démonstration :

- $h0p_3(\wedge) = \wedge$
- $\forall r, r' \in R(T, N) : h0p_3(r \to r') = h(\neg[p_3(r')]p_3(r)\Delta[p_3(r')])$

$$= h(\neg(p_3(r')))h0p_3(r)h(\Delta(p_3(r'))) = h0p_3(r)h(\Delta(p_3(r'))),$$

puisque $\neg(p_3(r'))$ appartient à V^*, soit $h(\neg(p_3(r'))) = \wedge$.

On peut alors écrire : $h0p_3(r \to r') = h0p_3(r)\, h(\neg(p_3(r')))\, h(\Delta(p_3(r')))$

$$= h0p_3(r)\, h(\neg(p_3(r'))\, \Delta\, (p_3(r')))$$

$$= h0p_3(r)\, h0p_3(r')$$

- $\forall A \in V,\ \forall r \in R(T, N) : h0p_3(A \uparrow r) = h(A)\, h(\bar{A})\, h0p_3(r) = h(\bar{A})\, h0p_3(r)$.

La même discussion, qu'à la propriété 2 (§ 2.1.3), nous donne :

Si $h0p_3(r)$ est différent de, $\wedge$, alors $h0p_3(A \uparrow r)$ est égal à $h0p_3(r)$,

sinon il est égal à A.

D'après le "principe de récurrence" on a l'identité :

$$\forall r \in R(T, N) \quad h0p_3(r) = \varphi(r).$$

2.3.2.4 - Recherche des bilangages réguliers de $P_3(V)$

Théorème 2 : Les bilangages réguliers de $P_3(V)$ sont les transcrits par une transcription de parenthèses, g^*, (définie à partir d'une application, g, de V' dans V) de l'intersection d'un langage régulier sur $V' \cup \bar{V}'$ avec le langage de parenthèses $P(V')$.

Démonstration : 1) Soit L un bilangage régulier de $P_3(V)$, non réduit au mot vide, il existe un bilangage régulier, L', de $\hat{V}$, tel que : $L = p_3(L')$. On peut trouver un bilangage grammatical, S(G), sur un alphabet V' et une application, g, de V' dans V tels que : $L' = \hat{g}(S[G])$.
En vertu de la remarque 2, § 2.3.2.2, on a :

$$L = p_3(V)[L'] = p_3(V) \circ \hat{g}(S[G]) = g^{*} \circ p_3(V')[S(G)].$$

Il est démontré dans PAIR [4] que l'ensemble des représentations des ramifications engendrées par une grammaire G, dans le binoïde universel descendant gauche (sur son alphabet) est l'intersection d'un langage régulier avec le langage de parenthèses qui constitue ce binoïde, soit :

$p_3(V')[S(G)] = K(V') \cap P(V')$, K est un langage régulier sur $V' \cup \bar{V}'$.

On a ainsi : $L = g^{*}(K \cap P(V'))$.

2) Soient K un langage régulier sur $V' \cup \bar{V}'$, et g^{*} une transcription de parenthèses de $P_3(V')$ dans $P_3(V)$, il nous suffit de démontrer que $K \cap P(V')$ est un bilangage régulier de $P_3(V')$. (1.8.2 et propriété 3, 2.3.2.3).
Soient M le monoïde fini, M' la partie de M, et ψ, l'homomorphisme de monoïdes de $(V' \cup \bar{V}')^{*}$ dans M, pour lesquels : $K = \psi^{-1}(M')$. Définissons deux parties de M, par : $M_1 = \psi(V'^{*})$ et $M_2 = \psi(\bar{V}'^{*})$.
Munissons le monoïde M, des lois $\dot{+}$ et $\dot{\times}$, définies par :

- $\forall u \in M,\ \forall a \in V' \qquad a \dot{\times} u = \psi(a)\,\psi(\bar{a})\,u$
- $\forall \alpha, \beta \in M \qquad \alpha \dot{+} \beta = \dot{\neg}(\beta)\,\alpha\,\dot{\Delta}(\beta)$, où $\dot{\neg}(\beta)$

est le plus grand facteur gauche de β appartenant à M_1 et $\dot{\Delta}(\beta)$ le quotient à gauche de β par $\dot{\neg}(\beta)$. On a : $\beta = \dot{\neg}(\beta)\,\dot{\Delta}(\beta)$.

Le triplet $(M, \dot{+}, \dot{\times})$ définit une structure de binoïde, B, sur M :
- la loi, $\dot{+}$, est associative et admet, $\psi(\wedge)$, comme élément neutre :

· $\forall \alpha, \beta, \gamma \in M : (\alpha \dot{+} \beta) \dot{+} \gamma = \dot{\neg}(\gamma)\,\dot{\neg}(\beta)\,\alpha\,\dot{\Delta}(\beta)\,\dot{\Delta}(\gamma) = \alpha \dot{+} (\beta \dot{+} \gamma)$

· $\forall \alpha \in M$: soit $m \in (V' \cup \bar{V}')^{*}$, tel que, $\alpha = \psi(m)$, alors

$\alpha \dot{+} \psi(\wedge) = \dot{\neg}[\psi(\wedge)]\,\alpha\,\dot{\Delta}[\psi(\wedge)] = \psi(\wedge)\,\alpha = \psi(\wedge)\,\psi(m) = \psi(m) = \alpha$

$\psi(\wedge) \dot{+} \alpha = \dot{\neg}(\alpha)\,\psi(\wedge)\,\dot{\Delta}(\alpha) = \dot{\neg}(\psi(m))\,\psi(\wedge)\,\dot{\Delta}(\alpha) = \psi(\neg m)\,\dot{\Delta}(\alpha) = \alpha$,

puisque l'on a : $\forall m \in (V' \cup \bar{V}')^{*}$ $\quad \dot{\neg}[\psi(m)] = \psi(\neg m)$

$\dot{\Delta}[\psi(m)] = \psi(\Delta m)$

- la loi, $\dot{\times}$, est une loi externe dans M, à opérateurs dans V'. La restriction à P(V' de ψ, $\psi_{P(V')}$, est un homomorphisme de binoïdes de $P_3(V')$ dans B :

- $\forall \alpha \in P_3(V'),\ \forall a \in V' : a \times \alpha = a\bar{a}\alpha$

$\psi(a \times \alpha) = \psi(a)\,\psi(\bar{a})\,\psi(\alpha) = a \dot{\times} \psi(\alpha)$

- $\forall \alpha, \beta \in P_3(V') : \alpha + \beta = \neg(\beta)\,\alpha\,\Delta(\beta)$

$\psi(\alpha + \beta) = \psi[\neg(\beta)]\,\psi(\alpha)\,\psi[\Delta(\beta)] = \dot{\neg}[\psi(\beta)]\,\psi(\alpha)\,\dot{\Delta}[\psi(\beta)]$

$= \psi(\alpha) \dot{+} \psi(\beta)$

De plus : $\quad K \cap P(V') = \psi^{-1}_{P(V')}[M']$.

Le triplet $(B, M', \psi_{P(V')})$ détermine sur $P_3(V')$, une structure de bilangage régulier : le bilangage $K \cap P(V')$.

Le théorème 2, et les propriétés 3 et 4 nous permettent d'affirmer que S_3, définit une structure de binoïde universel compatible sur les langages de parenthèses.

2.3.2.5 - Etude des quadruplets associés à S_3

Il est démontré dans PAIR [4], que l'ensemble des représentations des ramifications engendrées par une grammaire $G = (V, \rightarrow, X)$ dans le binoïde universel descendant gauche, $P_3(V)$, vérifie :

- $p_3(V)[S(G)] = K_3 \cap P(V)$, où K_3 est le langage régulier défini par : $K_3 = \overleftarrow{K}^{*}$, avec pour K, $\bigcup_{A \in V} \bar{A} \overleftarrow{K}_A$; les langages, K_A, étant les langages réguliers des productions de G.

- $\forall r \in S(G) : \varphi(r) = h \circ p_3(V)(r)$.

Nous pouvons prendre comme quadruplet associé à S_3, et pour la grammaire G : (V, I_v, K_3, h), I_v étant l'identité sur $V \cup \bar{V}$.

Les mots du langage de parenthèses, P(V), se terminant par une parenthèse fermante, $\bar{A}$, $(A \in V)$ et puisque $K_A = \{\wedge\}$, pour A appartenant à l'alphabet terminal T, on a l'égalité : $K_3 \cap P(V) = [K_3 \cap (V \cup \bar{V})^{*} \bar{T}] \cap P(V)$.

Un autre quadruplet possible est donc : $(V, I_v, K_3 \cap (V \cup \bar{V})^{*} \bar{T}, h)$. L'emploi de la structure S_3, dans le problème de l'analyse syntaxique conduit à une analyse de droite à gauche qui procède des racines vers les feuilles ; nous la dénommerons analyse descendante prédictive.

2.3.3 - Analyse ascendante

2.3.3.1 - Binoïde universel ascendant gauche

Munissons le langage de parenthèses $P(V)$, d'un couple, $S_2 = (+, \times)$, de lois de composition définies ci-dessous,

- Une loi de composition interne notée, + :

$\forall \beta, \gamma \in P(V)$ $\beta + \gamma = \gamma'_V(\beta)\, \gamma\, \Delta'_V(\beta)$, où $\Delta'_V(\beta)$ est le plus grand facteur droit de β qui appartient à $\overline{V}^*$, et $\gamma'_V(\beta)$, le quotient de β par $\Delta'_V(\beta)$, à droite : $\beta = \gamma'_V(\beta)\, \Delta'_V(\beta)$

- Une loi de composition externe à opérateurs dans V, notée $\times$:

$$\forall a \in V, \quad \forall \beta \in P(V) \qquad a \times \beta = \beta\, a\, \bar{a}$$

Il est démontré dans PAIR [4], que S_2 définit sur $P(V)$ une structure de binoïde universel, que nous appellerons le binoïde universel ascendant gauche et que nous noterons : $P_2 = (P(V), S_2)$. Nous désignerons par, p_2, l'isomorphisme de binoïdes, unique, de $\hat{V}$ dans $P_2(V)$:

- $\forall r, r' \in \hat{V} \qquad p_2(r \to r') = \gamma'_V[p_2(r)]\; p_2(r')\, \Delta'_V[p_2(r)]$
- $\forall r \in \hat{V}, \quad \forall A \in V : p_2(A \uparrow r) = p_2(r)\, A\, \bar{A}$.

Lorsqu'il n'y aura pas de confusion possible sur l'alphabet V, employé, γ'_V et Δ'_V seront respectivement représentés par γ' et Δ'.

2.3.3.2 - Nature des transcriptions de binoïdes universels ascendants gauches

Propriété 5 | Les transcriptions du binoïde universel $P_2(V)$ dans le binoïde universel $P_2(V')$, définies à partir des applications, g, de V dans V' sont les transcriptions de parenthèses g^*.

Démonstration L'homomorphisme, g^*, de $P(V)$ dans $P(V')$ vérifie :

- $g^*(\Lambda) = \Lambda$
- $\forall \alpha, \beta \in P_2(V)\; g^*(\alpha + \beta) = g^*(\gamma'_V(\alpha)\, \beta\, \Delta'_V(\alpha)) = g^*[\gamma'_V(\alpha)]\, g^*(\beta)\, g^*[\Delta'_V(\alpha)]$
$= g^*(\alpha) + g^*(\beta)$,

car les implications : $\Delta'_V(\alpha) \in \overline{V}^* \Longrightarrow g^*[\Delta'_V(\alpha)] \in \overline{V'}^*$.

et, $\gamma'_V(\alpha) = mA$, $A \in V$ et $m \in (V \cup \overline{V})^* \Longrightarrow g^*[\gamma'_V(\alpha)] = g^*(m)\, g(A)$, entrainent les égalités : $g^*[\gamma'_V(\alpha)] = \gamma'_{V'}[g^*(\alpha)]$ et $g^*[\Delta_V(\alpha)] = \Delta_{V'}[g^*(\alpha)]$

- $\forall A \in V, \ \forall \alpha \in P_2(V) : g^*(A \times \alpha) = g^*(\alpha)\, g(A)\, \overline{g(A)} = g(A) \times g^*(\alpha)$

D'après le principe de récurrence, l'homomorphisme, g^*, est identique à l'unique transcription de binoïdes, $\hat{g}$, de $P_2(V)$ dans $P_2(V')$, définie sur l'application g.

Remarque 3 : Cette propriété traduit la commutativité du diagramme,

$$\begin{array}{ccc} \hat{V} & \xrightarrow{\hat{g}_{V,V'}} & \hat{V}' \\ \Big\downarrow p_2(V) & & \Big\downarrow p_2(V') \\ P_2(V) & \xrightarrow{g^*} & P_2(V') \end{array}$$

soit $p_2(V') \circ \hat{g}_{V,V'} = g^* \circ p_2(V)$.

2.3.3.3 - Mot des feuilles des ramifications de R(T, N)

Propriétés | Soit $V = T \cup N$, T et N étant deux ensembles disjoints, alors le mot des feuilles $\varphi(r)$, pour toute ramification r de $R(T, N)$ est égal à $h' \circ p_2(r)$, où h' est l'homomorphisme alphabétique de $(V \cup \overline{V})^*$ dans T^*, défini par :
$A \in V \cup \overline{V} \quad h'(A) =$ SI $A \in T$ ALORS A SINON Λ.

Démonstration :

- $h' \circ p_2(\Lambda) = \Lambda$
- $\forall r, r' \in R(T, N) : h' \circ p_2(r \to r') = h'(\gamma'[p_2(r)]\, h' \circ p_2(r')\, h'(\Delta'[p_2(r)]))$
$= h'(\gamma'[p_2(r)])\, h' \circ p_2(r')$, puisque

$\Delta'[p_2(r)]$ appartient à $\overline{V}^*$, soit $h'(\Delta'[p_2(r)]) = \Lambda$.

On peut alors écrire :

$$h' \circ p_2(r \to r') = h'(\gamma'[p_2(r)])\, h'(\Delta'[p_2(r)])\; h' \circ p_2(r') = h' \circ p_2(r)\, h' \circ p_2(r').$$

- $\forall A \in V, \ \forall r \in R(T, N) \quad h' \circ p_2(A \uparrow r) = h' \circ p_2(r)\; h'(A)$.

Si $h' \circ p_2(r)$ est différent du mot vide, alors $h' \circ p_2(A \uparrow r)$ est égal à $h' \circ p_2(r)$, sinon il est égal à A.

D'après le "principe de récurrence", on a l'identité

$$\forall r \in R(T,N) \quad h' \circ p_2(r) = \varphi(r)$$

2.3.3.4 - Recherche des bilangages réguliers de $P_2(V)$

Théorème 3 | Les bilangages réguliers de $P_2(V)$ sont les transcrits par une transcription de parenthèses, g^*, (définie à partir d'une application, g, de V' dans V), de l'intersection d'un langage régulier sur $V' \cup \overline{V}'$ avec le langage de parenthèses $P(V')$.

Démonstration

1) Compte-tenu du fait que l'ensemble des représentations des ramifications engendrées par une grammaire, G, dans le binoïde universel ascendant gauche (sur son alphabet) est l'intersection d'un langage régulier avec le langage de parenthèses qui constitue ce binoïde, et de la remarque 3, § 2.3.3.2 on démontre aisément la partie directe du théorème comme celle du théorème 2.

2) Etant donnés un ensemble V' et un langage régulier K sur $V' \cup \overline{V}'$, il suffit de démontrer que $K \cap P(V')$ est un bilangage régulier de $P_2(V')$. Nous utilisons les mêmes notations qu'au théorème 2 : $K(V') = \psi^{-1}(M')$ et $M' \subset M$, $M_2 = \psi(\overline{V}'^*)$.

Munissons le monoïde M, des lois $\dot{+}$ et $\dot{x}$:

- $\forall u \in M, \ \forall a \in V' \qquad a \dot{x} u = u \, \psi(a) \, \psi(\tilde{a})$
- $\forall \alpha, \beta \in M \qquad \alpha \dot{+} \beta = \dot{\gamma}'(\alpha) \, \beta \, \dot{\Delta}'(\alpha)$, où $\dot{\Delta}'(\alpha)$

est le plus grand facteur droit de α, qui appartient à M_2 et $\dot{\gamma}'(\alpha)$ le quotient à droite de α par $\dot{\Delta}'(\alpha)$: $\alpha = \dot{\gamma}'(\alpha) \dot{\Delta}'(\alpha)$. Le triplet $(M, \dot{+}, \dot{x})$ définit une structure de binoïde, B, sur M :

- la loi, $\dot{+}$, est associative et admet, $\psi(\wedge)$, comme élément neutre.
  - $\forall \alpha, \beta, \gamma \in M : (\alpha \dot{+} \beta) \dot{+} \gamma = \dot{\gamma}'(\alpha) \, \dot{\gamma}'(\beta) \gamma \, \dot{\Delta}'(\beta) \, \dot{\Delta}'(\alpha) = \alpha \dot{+} (\beta \dot{+} \gamma)$
  - $\forall \alpha \in M$, si $\alpha = \psi(m)$ avec m appartenant à $(V \cup \overline{V})^*$, alors on a,

$$\alpha \dot{+} \psi(\wedge) = \dot{\gamma}'(\alpha) \, \psi(\wedge) \, \dot{\Delta}'(\alpha) = \dot{\gamma}'[\psi(m)] \, \psi(\wedge) \, \dot{\Delta}'(\alpha) = \psi(\gamma m) \, \dot{\Delta}'(\alpha) = \alpha$$

$$\psi(\wedge) \dot{+} \alpha = \alpha \ \psi(\wedge) = \psi(m) \, \psi(\wedge) = \psi(m) = \alpha,$$

puisque pour tout m de $(V' \cup \overline{V}')^*$, $\dot{\gamma}'[\psi(m)] = \psi[\gamma'(m)]$ et $\dot{\Delta}'[\psi(m)] = \psi[\Delta'(m)]$.

La loi, $\dot{x}$, est une loi externe dans M, à opérateurs dans V'.

La restriction à $P(V')$ de ψ, $\psi_{P(V')}$, est un homomorphisme de binoïdes de $P_2(V')$ dans B :

- $\forall \alpha \in P_2(V'), \ \forall a \in V' : \psi(a \times \alpha) = \psi(\alpha) \, \psi(a) \, \psi(\tilde{a}) = a \dot{x} \psi(\alpha)$
- $\forall \alpha, \beta \in P_2(V') : \psi(\alpha + \beta) = \psi[\gamma'(\alpha)] \, \psi(\beta) \, \psi[\Delta'(\alpha)]$

$$= \dot{\gamma}'[\psi(\alpha)] \, \psi(\beta) \, \dot{\Delta}'[\psi(\alpha)] = \psi(\alpha) \dot{+} \psi(\beta)$$

De plus : $K \cap P(V') = \psi^{-1}_{P(V')}(M')$.

Le triplet $(B, M', \psi_{P(V')})$ détermine sur $P_2(V')$ une structure de bilangage régulier: le bilangage $K(V') \cap P(V')$.

Le théorème 3 et les propriétés 5 et 6 nous permettent d'affirmer que S_2, définit une structure de binoïde universel compatible sur les langages de parenthèses.

2.3.3.5 - Etude des quadruplets associés à S_2

Il est démontré dans PAIR [4], que l'ensemble des représentations des ramification engendrées par une grammaire, $G = (V, \rightarrow, X)$, dans le binoïde universel ascendant gauche, $P_2(V)$, est l'intersection d'un langage régulier K_2, avec le langage de parenthèse qui constitue ce binoïde :

- $p_2(V)[S(G)] = K_2 \cap P(V)$, avec pour K_2, le langage, $K^* \widetilde{\widetilde{X}}$, où $K = \bigcup_{A \in V} \widetilde{\widetilde{K}}_A A$.
- $\forall r \in S(G) : \varphi(r) = h' \circ p_2(V)[r]$.

Le quadruplet, (V, I_v, K_2, h'), peut être associé à la structure S_2. Tout mot de $P(V)$, commence par un élément de V ; on peut ainsi définir un autre quadruplet : $(V, I_v, T(V \cup \overline{V})^* \cap K_2, h')$, puisque $[T(V \cap \overline{V})^* \cap K_2] \cap P(V) = K_2 \cap P(V)$.

L'emploi de la structure, S_2, dans le problème de l'analyse syntaxique, conduit à une analyse de droite à gauche qui procède des feuilles vers les racines, nous la dénommerons analyse ascendante.

2.4 - ANALYSE AVEC MARQUEURS

Soit (V, g, K, h), un quadruplet associé à une structure compatible ; montrons q
l'on peut toujours se ramener au cas où tous les mots de K, commencent par l
même lettre, $\#$, et se terminent par la même lettre $\overline{\#}$, $\#$ et $\overline{\#}$ n'ayant pas d'
occurrence dans les mots de K.
Adjoignons à V, un marqueur $\#$, n'appartenant pas à ce dernier, et soit, $\overline{\#}$,
parenthèse fermante associée à $\#$, de telle sorte que $\overline{\#}$ n'appartient pas à
posons $V_1 = V \cup \#$ et $K_1 = \# K \overline{\#}$

Alors $K_1 \cap P(V_1) = \# (K \cap P(V)) \overline{\#}$.

Si nous prolongeons l'application g par $g(\#) = g(\overline{\#}) = \wedge$, le langage régulier
vérifie l'égalité :

$$h^{-1}(\alpha) \cap g^*(K_1 \cap P(V_1)) = h^{-1}(\alpha) \cap g^*(K \cap P(V)).$$

La résolution du problème de l'analyse syntaxique par le quadruplet, $(V_1, g, K_1$
nous conduit au même ensemble, $\mathcal{A}(\alpha^*)$, que pour (V, g, K, h).
Dans la suite, nous traiterons toujours des analyses avec adjonction de marque
ceci nous simplifiera, en particulier, l'étude du chapitre 5 et la réalisation des
programmes d'analyse.

CHAPITRE III

Etude du passage d'une grammaire,

aux quadruplets (V', g, K, h),

pour une structure compatible donnée.

INTRODUCTION

Il s'agit dans ce chapitre d'étudier des techniques simples, nous permettant pour chacune des structures compatibles étudiées au chapitre précédent, de construire les quadruplets (V', g, K, h).

Il nous faudra en particulier étudier les modes de représentations dans la mémoire de l'ordinateur, d'une grammaire, d'un langage régulier, et d'un graphe.

REPRESENTATION EN MEMOIRE D'UN GRAPHE, D'UN LANGAGE LOCAL, D'UN LANGAGE REGULIER ET D'UNE GRAMMAIRE

3.1.1 - Représentation d'un graphe

Les graphes que nous utilisons seront toujours mémorisés, dans l'ordinateur sous forme de deux tableaux unidimentionnels TA1, et TS :

- Le tableau, TS, contient la liste des familles de successeurs de chacun des points du graphe.

Chaque dernier élément d'une famille est affecté d'un signe moins.

On peut diminuer la place en mémoire pour une telle représentation d'un graphe, $G = (E, \Gamma)$, en procédant comme suit :

· Soit, F_i, pour $i=1, \ldots, n$, les différentes familles de G, et supposons que l'intersection de p familles, $I = \bigcap_{i=1}^{p} F_i$, soit non vide ($p \leq n$).

Chaque famille, F_i, de successeurs d'un point x de G pour laquelle, $F_i \cap I = \emptyset$, sera rangée dans le tableau TS comme précédemment, par contre pour les autres points on ne le fera que pour les sous-familles : $F_i' = F_i - I$. Pour différencier ces sous-familles on n'affectera pas leur dernier élément d'un signe moins, mais on leur imposera comme dernier élément, un point fictif n'appartenant pas à E.

L'ensemble, E, étant codifié en mémoire par des entiers on pourra prendre zéro pour celui-ci ; on considère ainsi que l'ensemble de définition d'un graphe sera toujours codifié à partir de un.

La sous-famille commune, I, sera implantée à la fin du tableau TS, à partir d'un rang fixé, NP1.

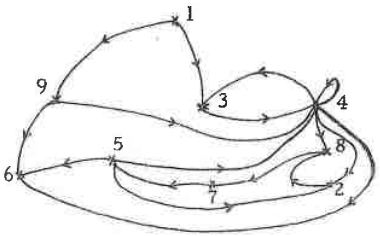
Un tel procédé sera avantageux, si il existe un entier, p, (p> 1) pour lequel une sous-famille I possède un nombre d'éléments au moins supérieur ou égal à 2 :

$$\text{card } I + p < p \text{ card } I$$

$$\text{card } I \geq \left[\frac{p}{p-1}\right] + 1 = 2 + \left[\frac{p}{p-1}\right] = 2$$

- Le tableau, TA1, donne pour chaque point du graphe l'indice dans, TS, du premier élément de la famille de ses successeurs.
Lorsqu'un point ne possède pas de successeur, cet indice sera nul.

Exemple :



Matrice booléenne associée

	1	2	3	4	5	6	7	8	9
1	0	0	1	0	0	0	0	0	1
2	0	0	0	0	0	0	0	1	0
3	0	0	0	1	0	0	0	0	0
4	0	1	1	1	0	1	0	1	0
5	0	1	0	1	0	1	0	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	1	0	0	0	0
8	0	0	0	0	0	0	1	0	0
9	0	0	0	1	0	0	0	0	0

(Voir [8])

	1	2	3	4	5	6	7	8	9
TA1	1	3	4	5	12	0	8	9	10

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

Dans ce cas : NP1 = 12.

Remarque : Lorsque pour un point, x, du graphe, $I \cap F_i = I$, la valeur de l'élém de rang x dans le tableau, TA1, sera l'entier NP1.

3.1.2 - Représentation d'un langage local sur un alphabet V

Un langage local, L, sur un alphabet V, sera toujours donné sous la forme d'un quadruplet, $(V, D, F, g\ell)$ où :

- D est l'ensemble initial $(D \subset V)$
- F est l'ensemble final $(F \subset V)$
- $g\ell = (V, \Gamma)$, est le graphe des transitions associé au langage L.

L'alphabet V, étant codifié dans l'ordinateur sous forme d'entiers, les ensembles D et F pourront être rangés dans des tableaux entiers, cependant comme les langages locaux nous sont utiles lors de la détermination des grammaires nous les représenterons d'une façon particulière que nous exposerons dans un paragraphe suivant.
Le graphe, $g\ell$, sera représenté comme nous l'avons convenu précédemment.

3.1.3 - Représentation d'un langage régulier

Un langage régulier étant considéré comme transcrit d'un langage local, tout langage régulier, K, sur un alphabet V sera donné sous la forme d'un triplet (V', L, t), tel que : $K(V) = t(L)$, avec pour L, le langage local $(V', D, F, g\ell)$ et t, une transcription de V'^* dans V^*.
La représentation en mémoire de L, est définie par deux tableaux : TA1 et TS, ainsi que par les ensembles D et F.
Un troisième tableau, TA2, nous donnera pour tout point x de V', la lettre associée à t(x) dans l'alphabet V.
Les deux ensembles D et F, et les trois tableaux TA1, TA2, et TS détermineront entièrement le langage régulier K.
Etant donné un langage régulier, K, nous nous intéresserons au langage régulier $K_1 = \# K \overline{\#}$, que nous avons défini au chapitre 2. L'introduction des marqueurs # et $\overline{\#}$ nous conduit à adjoindre au graphe, $g\ell$, qui intervient dans la définition de K, deux points $\mathcal{E}$ et $\mathcal{E}'$ vérifiant, $t(\mathcal{E}) = \#$ et $t(\mathcal{E}') = \overline{\#}$, de telle sorte que K_1 sera le transcrit par t, du langage local, $L_1 = (V'_1, D_1, F_1, g\ell_1)$, avec :

- $V'_1 = V' \cup \{\mathcal{E}, \mathcal{E}'\}$
- $D_1 = \{\mathcal{E}\}$, est l'ensemble initial
- $F_1 = \{\mathcal{E}'\}$, est l'ensemble final
- $g\ell_1 = (V'_1, \Gamma_1)$, est le graphe des transitions défini par :

$$(\forall x, y \in V')\ (x \Gamma_1 y \iff x \Gamma y)$$
$$(\forall x \in D)\ (\mathcal{E}\ \Gamma_1\ x)$$
$$(\forall x \in F)\ (x\ \Gamma_1\ \mathcal{E}')$$

Tous les alphabets utilisés étant codifiés en mémoire par des entiers, on prendra toujours, un, pour codifier le point $\mathcal{E}$: $c_1(\mathcal{E}) = 1$, c_1 étant l'application qui réalise cette codification.

3.1.4 - Représentation d'une grammaire

Toute grammaire $G = (V, \rightarrow, X)$ pourra être mémorisée sous la forme du langage régulier : $K^0 = X \cup (\bigcup_{A \in N} K_A)$ où,

- N, est l'alphabet auxiliaire représenté par les entiers de 2 à N.
- T, est l'alphabet terminal représenté par les entiers de N+1 à VOC.
- Pour tout A de N, $K_A = t_A(L_A)$ est le langage de production associé à L_A est un langage local défini par le quadruplet, $(V_A, D_A, F_A, g\ell_A)$ et t_A, une transcription.
- X, son ensemble d'axiomes est donné par $X = t_0(L_0)$, où L_0 est le lang local $(V_0, D_0, F_0, g\ell_0)$ et t_0 est une transcription.

Il nous est loisible de choisir les langages locaux L_A (pour A appartenant à N) L_0 avec des alphabets disjoints de telle sorte que la représentation en mémoire d toute grammaire G, sera celle de la grammaire locale, G', associée à G comm nous l'avons réalisé au chapitre 2.

Soient V', l'alphabet $V_0 \cup (\bigcup_{A \in N} V_A)$, et c_1 (respectivement c_2) l'application qu réalise la codification numérique de V' (respectivement de l'alphabet V).

Si NB désigne le nombre d'éléments de V', on conviendra désormais :

$\forall x \in V'$, $c_1(x) \in \{2, 3 \ldots . NB+1\}$.

Soit le langage local $L^0 = L_0 \cup (\bigcup_{A \in N} L_A) = (V', D, F, g\ell^0)$:

- le graphe $g\ell^0 = (V', \Gamma^0)$ est défini par,

$$(\forall A \in N)\ (\forall x, y \in V_A)\ (x \Gamma^0 y \iff x \Gamma_A y)$$

$$(\forall x, y \in V_0)\ (x \Gamma^0 y \iff x \Gamma_0 y)$$

Il est rangé en mémoire sous la forme de deux tableaux TA1 et TS.

- l'ensemble initial, D est défini par le graphe $(\mathbb{N}, D)$:

$\forall x \in \{1, \ldots, N\}$, $\forall y \in \{2, \ldots, NB+1\}$ $x D y \iff (x \neq 1$ et $c_1^{-1}(y) \in D_{c_2^{-1}(x)})$

ou $(x = 1$ et $c_1^{-1}(y) \in D_0)$.

Il est rangé en mémoire à l'aide des deux tableaux TA1D et TSD.

- l'ensemble final F, est donné par le graphe $(\mathbb{N}, F)$:

$\forall x \in \{1, \ldots, N\}$, $\forall y \in \{2, \ldots, NB+1\}$ $x F y \iff (x \neq 1$ et $c_1(y) \in F_{c_2^{-1}(x)})$

où $(x = 1$ et $c_1^{-1}(y) \in F_0)$

Les tableaux TA1F et TSF le représente en mémoire.

- Le langage régulier K^0, est le transcrit du langage local L^0, par la transcription, t^0, de V'^* dans V^* :

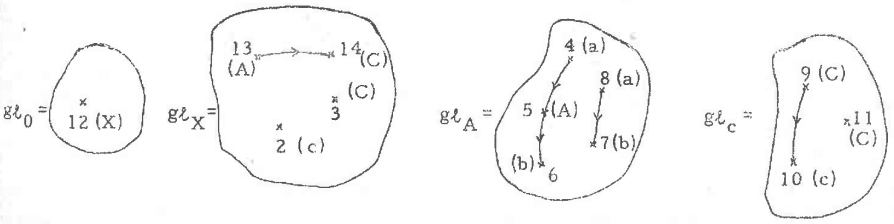
$$- \forall x \in V_A \qquad t^0(x) = t_A(x)$$

$$- \forall x \in V_0 \qquad t^0(x) = t_0(x)$$

Une grammaire sera donc introduite dans l'ordinateur sous la forme de septs tableaux unidimensionnels :

- TA1 [2:NB+1], TA2[2:NB+1], TS
- TA1D[2:N] , TSD
- TA1F[2:N] , TSF

Exemple : Soit $N = \{A, C, X\}$, et $T = \{a, b, c\}$; X, l'axiome.



$K_0 = \{X\}$ $K_X = \{AC, C, c\}$ $K_A = \{aAb, ab\}$ $K_C = \{Cc, c\}$

	X	A	C	a	b	c
C2	2	3	4	5	6	7

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
TA1		0	0	1	2	0	0	3	4	0	0	0	5	0
TA2		7	4	5	3	6	6	5	4	7	7	2	3	4
TS	-5	-6	-7	-10	-14									

	1	2	3	4
TA1D	1	2	5	7
TA1F	1	2	5	7

	1	2	3	4	5	6	7	8
TSD	-12	2	3	-13	4	-8	11	-9
TSF	-12	2	3	-14	6	-7	11	-10

3.1.5 - Cas d'une grammaire donnée sous forme de règles

Le passage d'une grammaire, donnée par ses règles, aux langages réguliers des quadruplets associés aux structures compatibles a été étudié par Melle Studenman [5]. Nous nous contenterons d'indiquer ici la façon de passer des règles de grammaire à la représentation d'une grammaire, comme nous l'avons réalisée au paragraphe précédent. Il nous faut définir les transformations à effectuer pour construire le langage L^0, à partir des règles de grammaire.

Une méthode simple consiste à associer à chaque occurrence d'une lettre dans une règle, un entier qui caractérisera l'ordre de cette occurrence, et que l'on désignera par ORD. Ceci suppose que l'alphabet auxiliaire (c'est à dire l'ensemble des membres de gauche des règles) sera ordonné, ainsi que chacune des alternatives d'une règle ; on prendra par exemple l'ordre de rangement dans le tableau qui nous servira à introduire ces règles en mémoire.

Nous définissons alors un graphe, $g\ell = (O, \Gamma)$, sur l'ensemble O, des entiers associés à chacune des occurrences d'une lettre, dans une règle :

$$x \Gamma y \iff y = x+1.$$

Ce graphe sera en fait le graphe $g\ell^0$, puisque le sous-graphe partiel associé à L_0 est formé de points sans successeur, que nous pourrons représenter par les entiers de 2 à NAXIOM, si l'on désigne par NAXIOM le nombre, plus un, des axiomes. Si l'application, c_3, est la codification entière des lettres de l'alphabet de la grammaire donnée, la transcription t^0, sera :

- $\forall x \in O$, $t^0(x) = c_3[ORD^{-1}(x)]$
- $\forall x$, $2 \leq x \leq$ NAXIOM $\quad t^0(x) = c_3(x)$.

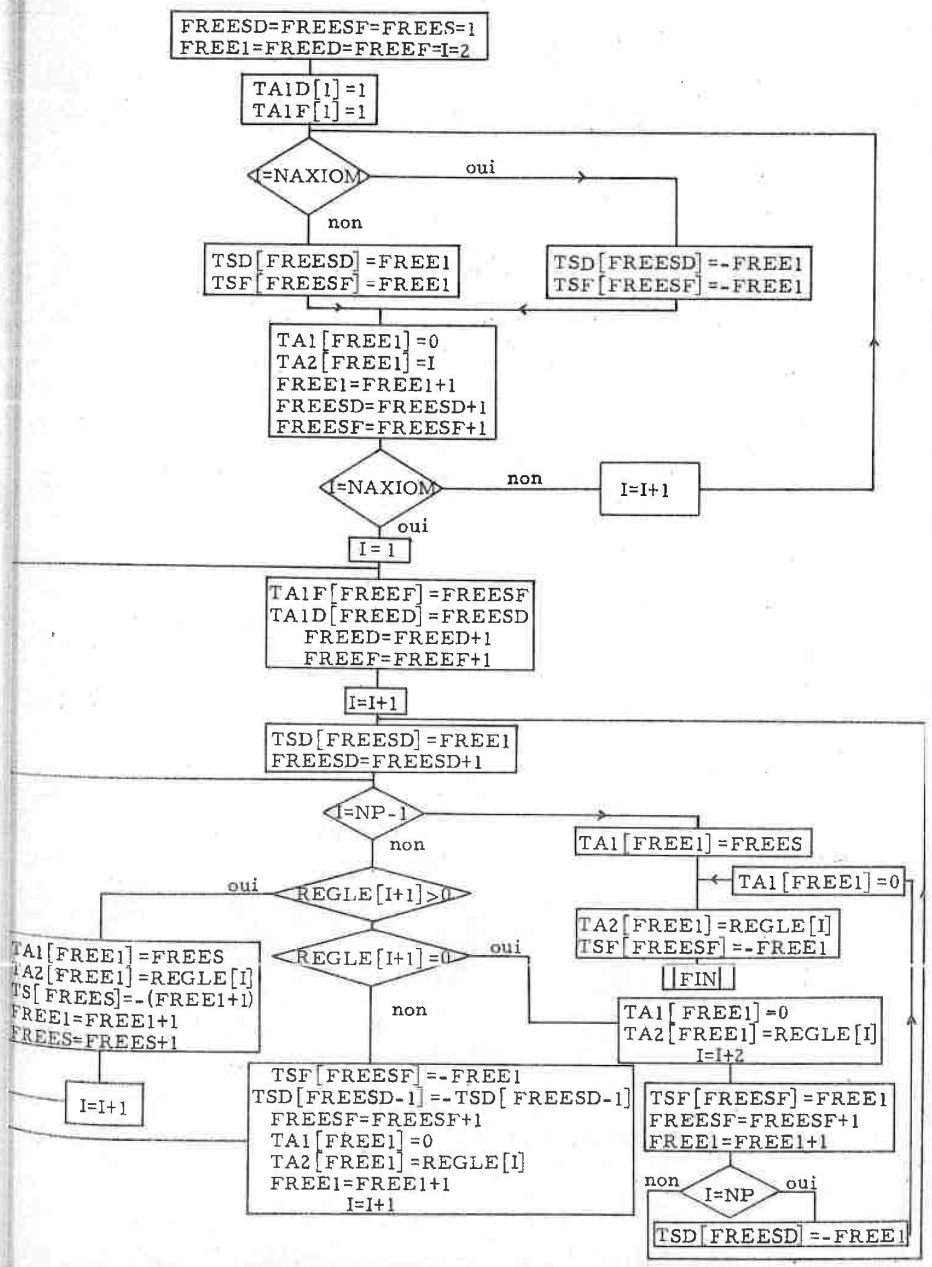
Les règles de grammaire sont rangées dans un tableau unidimensionnel, REGLE[1:NP]. Pour chaque règle on affectera au tableau, REGLE, le membre de gauche que l'on précédera du signe moins, puis les éléments du membre de droite avec le signe plus et un zéro pour chaque alternative d'une règle.

Cette méthode a le désavantage de ne pas optimiser la place en mémoire ; en effet une lettre apparaissant deux fois dans les membres de droite d'une même règle de grammaire est dédoublée. Au point de vue de l'étude contextuelle cela présentera au contraire l'avantage dans certains cas, de permettre un choix entre les deux possibilités d'une même règle.

Algorithme

Les tableaux TA1, TA2, TS, TSD, TSF, TA1D, TA1F sont remplis séquentiellement ; une variable nous permet de connaître pour chacun d'eux le premier élément du tableau non encore occupé, soit respectivement

FREE1, FREE1, FREES, FREESD, FREESF, FREED, FREEF.


FREESD=FREESF=FREES=1
FREE1=FREED=FREEF=I=2
TA1D[1]=1
TA1F[1]=1
I=NAXIOM
oui
non
TSD[FREESD]=FREE1
TSF[FREESF]=FREE1
TSD[FREESD]=-FREE1
TSF[FREESF]=-FREE1
TA1[FREE1]=0
TA2[FREE1]=I
FREE1=FREE1+1
FREESD=FREESD+1
FREESF=FREESF+1
I=NAXIOM
non
I=I+1
oui
I=1
TA1F[FREEF]=FREESF
TA1D[FREED]=FREESD
FREED=FREED+1
FREEF=FREEF+1
I=I+1
TSD[FREESD]=FREE1
FREESD=FREESD+1
I=NP-1
non
TA1[FREE1]=FREES
TA1[FREE1]=0
oui
REGLE[I+1]>0
TA2[FREE1]=REGLE[I]
TSF[FREESF]=-FREE1
TA1[FREE1]=FREES
TA2[FREE1]=REGLE[I]
TS[FREES]=-(FREE1+1)
FREE1=FREE1+1
FREES=FREES+1
REGLE[I+1]=0
oui
FIN
non
TA1[FREE1]=0
TA2[FREE1]=REGLE[I]
I=I+2
I=I+1
TSF[FREESF]=-FREE1
TSD[FREESD-1]=-TSD[FREESD-1]
FREESF=FREESF+1
TA1[FREE1]=0
TA2[FREE1]=REGLE[I]
FREE1=FREE1+1
I=I+1
TSF[FREESF]=FREE1
FREESF=FREESF+1
FREE1=FREE1+1
non
I=NP
oui
TSD[FREESD]=-FREE1

ETUDE DES PROGRAMMES DE PASSAGE D'UNE GRAMMAIRE AUX QUADRUPLETS (V, g, K, h), ASSOCIES AUX STRUCTURES COMPATIBLES

3.3.1 - Mémorisation du langage régulier K et de l'homomorphisme, g^*

3.3.1.1 - Données et résultats des programmes de passage

- Les données sont les sept tableaux unidimensionnels nécessaires à la définition d'une grammaire, en mémoire et dont les identificateurs sont : TAB1, TABS, TA1D, TSD, TA1F, TSF, TAB2.

Nous avons étudié en 3.2 la représentation en mémoire d'une grammaire, G, par le langage régulier, $K^0=t^0(L^0)$, où $L^0=(V', D, F, g\ell^0)$:

- TAB1 et TABS définissent les familles du graphe des transitions de L^0.
- TA1D et TSD définissent les familles du graphe représentant l'ensemble initial de L^0 ; TA1F et TSF celles du graphe représentant l'ensemble final de L^0.
- TAB2 définit la transcription, t^0.

- Les résultats seront trois tableaux unidimensionnels, et, dans le cas de certaines structures compatibles, un entier d'identificateur NP1 :

- les tableaux, TA1 et TS, de dimension respective NPG1 et NPF, définissent entièrement le langage local L tel que : K = t(L), puisque l'on convient qu'en mémoire l'ensemble initial de L (respectivement final) est représenté par {1} (respectivement NPG1), et que nous n'effectuerons que des analyses avec adjonction de marqueurs.
- le tableau TA2 à NPG1 éléments nous donne soit la transcription, t, soit l'application g^*
- l'entier, NP1, définit dans certains cas l'adresse dans TS, d'une sous-famille commune aux familles du graphe $g\ell^0$.

Remarque 1 : Soit un alphabet V possédant n éléments ; on conviendra toujours que les parenthèses fermantes associées à V, $\bar{V}$ se déduisent de V de la façon suivante : si x appartient à V et c réalise la codification entière de V en mémoire, alors $c(\bar{x}) = c(x) + n$.

3.3.1.2 - Etude des programmes dans chacune des structures compatibles étudiées au chapitre 2

I. Structure S_3

Le quadruplet associé S_3, que nous choisissons est (V_1, I_{V_1}, K, h), avec :

$$K = \# [K_3 \cap (V \cup \bar{V})^* \bar{T}]^- \# \quad \text{et} \quad K_3 = \tilde{x} \left(\bigcup_{A \in V} \bar{A} \tilde{K}_A \right)^*$$

L'alphabet V_1 est la réunion de # avec l'alphabet V de G. Rappelons que $K_3^- = K_3 \cap L(V)$ où L(V) est le langage local admettant pour transitions :

$$\{ \forall\, a, b \in V \quad | \quad ab, \bar{a}b, \bar{a}\bar{b}, a\bar{a} \}$$

Il nous reste à préciser le langage local L_3 et à définir la transcription t_3, tel que : $K = t_3(L_3)$ - Le tableau, TA2, associera à tout x de L_3, l'élément $t_3(x)$ de K.

1. Etude de la génération de K

Le langage K, peut être défini à partir d'une suite de langages réguliers : K^1, K^2, K^3, K^4 chacun d'eux étant mémorisé sous la forme d'un langage local $L^i (i=1,2,3,4)$ et d'une transcription $t^i (i=1,2,3,4)$ tels que : $K^i = t^i(L^i)$ $(i=1,2,3,4)$.
Nous considérerons que tous les alphabets utilisés dans ce chapitre seront identifi avec leur codification entière dans l'ordinateur.

1.1) $K^1 = \tilde{K}^0 = \tilde{x} \cup \left(\bigcup_{A \in N} \tilde{K}_A \right) = t^0(L^1)$, avec pour L^1 le langage local, (V', F, Γ^1) :

- l'ensemble, V', est formé des entiers de 2 à NB+1.
- le graphe $g\ell^1 = (V', \Gamma^1)$ est donné par :

$$\forall\, x, y \in V' \qquad x \Gamma^1 y \iff y \Gamma^0 x.$$

On définit ainsi "les transitions du type 1".

1.2) $K^2 = \# \tilde{x} \cup \left(\bigcup_{A \in V} \bar{A} \tilde{K}_A \right) = t^2(L^2)$.

Définissons par, V_T, (respectivement par V_N) les entiers de N+NB+1 à NB+V (respectivement de NB+1 à NB+N). On rappelle que l'alphabet de G possède, VOC-1, éléments.
Le langage local, L^2, est le quadruplet $(V^2, V^2 - V', D \cup V_T, g\ell^2)$.
(Les ensembles D et F seront souvent identifiés avec les graphes de même nom définis en 3.1.4). L'ensemble de définition de L^2, V^2, est égal à $V_1' \cup V_T \cup V_N$, c'est à dire les entiers de 1 à NB+VOC.
La transcription, t^2, est définie par :

- $\forall\, x \in V' \qquad t^2(x) = t^0(x)$
- $t^2(1) = 1 \qquad (C_2(\#) = 1)$
- $\forall\, x \in V^2 - V_1' \qquad t^2(x) = x + VOC - NB$ (V_1' est l'ensemble des entiers de 1 à NB+1).

Quant au graphe $g\ell^2 = (V^2, \Gamma^2)$:

- $(\forall\, x \in V')\ (x \Gamma^2 y \iff x \Gamma^1 y)$ (transitions du type 1)
- $(\forall\, x \in V_N)\ (\forall\, y \in F(x-NB))\ (x \Gamma^2 y)$; on crée ainsi "les transitions du type 2", plus explicitement définies par :

$$(\forall\, A \in N)\ (\forall\, B \in t_A(F_A))\ (\bar{A}B)$$

- $(\forall\, y \in F(1))\ (1 \Gamma^2 y)$, ce qui nous donne les "transitions du type 3" :

$$(\forall\, A \in t_0(F_0))\ (\# A).$$

1.3) $K^3 = \# [\tilde{x} \left(\bigcup_{A \in V} \bar{A} \tilde{K}_A \right)^* \cap L^-(V)] = t^2(L^3)$, avec pour L^3, le langage local $(V^2, \{1\}, D \cup V_T, g\ell^3)$.

Le graphe, $g\ell^3 = (V^2, \Gamma^3)$, est donné par :

- $(\forall\, x, x=1,2,\ldots N)\ (\forall\, y \in D(x))\ (y \Gamma^3 z, z = NB + t^2(y))$, qui nous donne les "transitions du type 4" :

$$(\forall\, A \in V)\ (\forall\, A \in t_0(D_0) \cup \left(\bigcup_{C \in N} t_C(D_C) \right))\ (A\bar{A})$$

- $(\forall\, x \in V_T)\ (\forall\, y \in V^2 - V_1')\ (x \Gamma^3 y)$, soit les "transitions du type 5" :

$$(\forall\, A \in T)\ (\forall\, B \in V)\ (\bar{A}\bar{B})$$

1.4) $K^4 = t^4(L^4) = [\# \tilde{x} \left(\bigcup_{A \in V} \bar{A} \tilde{K}_A \right)^* \cap L^-(V) \cap (V \cup \bar{V}) \bar{T}] \#$; ou ce qui est équivalent : $K^4 = K$.
L^4 est le langage local, L_3, que nous cherchions ; il est donné par le quadruplet $(V^4, \{1\}, NPG1, g\ell^4)$

- L'ensemble, V^4, est formé des entiers de 1 à NB+VOC+1
- NPG1 = NB+VOC+1
- La transcription, t^4, est obtenue en prolongeant t^2 :
  - $\forall\, x \in V^2 \qquad t^4(x) = t^2(x)$
  - $t^4(NPG1) = VOC+1$
- Quant au graphe $g\ell^4 = (V^4, \Gamma^4)$:
  - $(\forall\, x, y \in V^2)\ (x \Gamma^4 y \iff x \Gamma^3 y)$
  - $(\forall\, x \in V_T)\ (x \Gamma^4 NPG1)$, ce qui nous donne "les transitions du type 6" :

$$(\forall\, A \in T)\ (\bar{A}\#).$$

2. Programmation de la mémorisation des langages K^1, K^2, K^3, K^4

Il s'agit de construire la liste des familles des points du graphe des transitions d langage local, L^4. Notre analyse précédente nous a conduit à définir progressive les transitions de L^4. Nous créerons ainsi une liste, LIS, qui sera celle des familles de $g\ell^4$. Nous générerons séquentiellement cette liste au cours de plusie étapes de façon à définir successivement les transitions de L^1, puis celles de L^2, L^3 et L^4.

Nous associons à cette liste un tableau unidimensionnel, TA1 , possédant NPG1 éléments ; il donne à chaque étape, i , (i=1, 2, 3, 4) l'adresse dans LIS, du prem élément de la famille des successeurs des points du graphe des transitions de L^i. Un autre tableau unidimensionnel, TA2, possédant le même nombre d'éléments, donne à chaque étape, i , le transcrit par t^i, de chacun des points du graphe des transitions de L^i.

2.1) Définition de la liste LIS

Cette liste sera divisée en enregistrements de deux mots. Une adresse, FREE, donne à chaque instant l'adresse dans LIS, du premier enregistrement non défini. Supposons que nous ayons trouvé un point, y , tel que $x\Gamma^i y$ à la ième étape ; nou créons un enregistrement à l'adresse FREE.

Dans le premier mot de l'enregistrement, nous plaçons zéro, si y est le premi successeur de x que nous connaissions ; dans le cas contraire nous y placerons l'adresse du premier successeur de x ; mais nous conviendrons que, y , sera désormais le premier successeur de x :

- LIS[FREE] = 0 et TA1[x] = FREE ,

ou - LIS[FREE] = TA1[x] et TA1[x] = FREE.

Dans le deuxième mot, nous plaçons y.

Exemple :

	1		x		NPG1
TA1			n_1		

		n_1		n_2		n_3		...
LIS		n_2	y_1	n_3	y_2	0	y_3	...

Si nous sommes à l'étape, i : $y_1, y_2, y_3 \in \Gamma^4(x)$.

2.2) Organigramme général

L'analyse précédente nous invite à définir quatre modules que nous exécuterons successivement, et qui sont :

LECTURE , MIROIR1 , TRANSITION1 , et PASSAGE1.

Module LECTURE

Ce module réalise la lecture des tableaux :

TAB1 , TABS ; TA1D , TA1F , TSD , TSF ; TA2.

Il initialise à zéro chacun des éléments de TA1.

Module MIROIR1

Il crée les transitions du type 1.

Module TRANSITIONS1

Ce module crée les transitions des types 2, 3, et 4 ; à chacune d'elles est associé un sous-module :

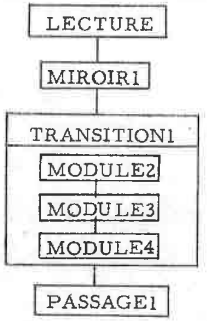
MODULE2 , MODULE3 , MODULE4.

Module PASSAGE1

Il s'agit dans ce module de construire à partir de LIS le tableau des familles, TS , tel que nous l'avons déjà défini. La seule famille de successeurs que nous n'avons pas rangée dans LIS est la famille de successeurs, commune aux éléments de V_T (soient les transitions des types 5 et 6).

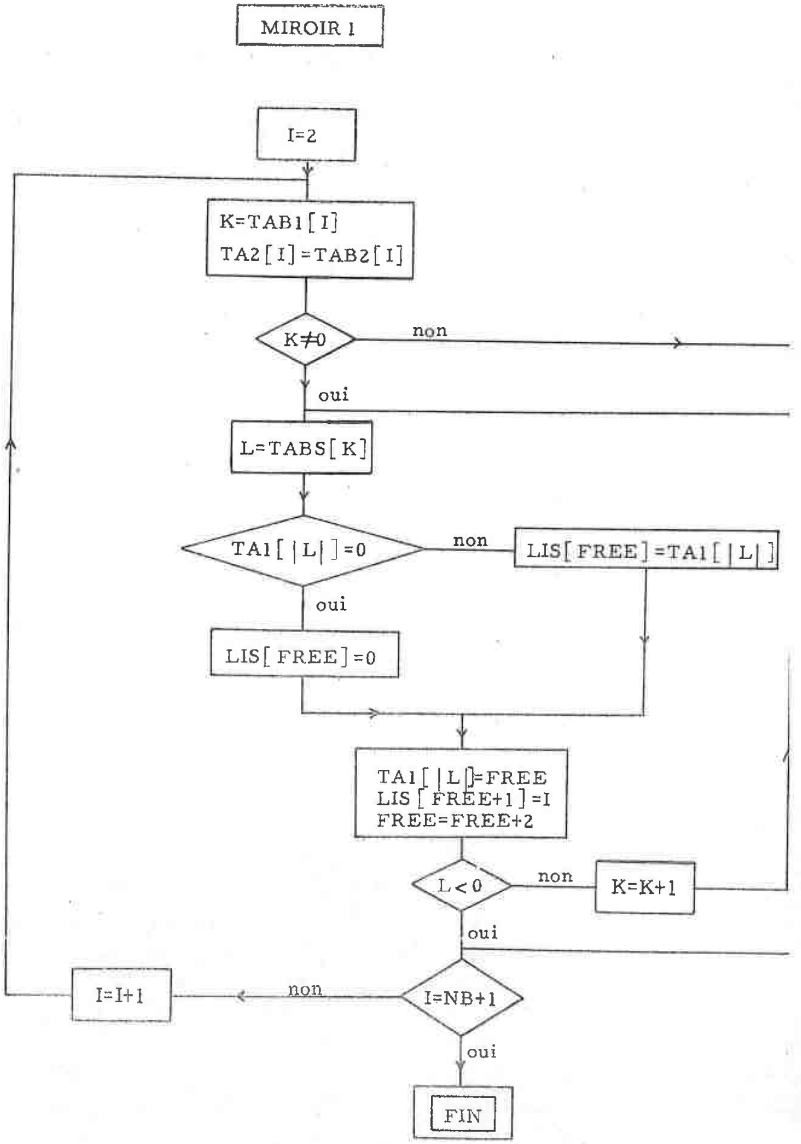
Il nous suffit ainsi de ranger séquentiellement les familles de LIS, puis la famille commune pour définir totalement le tableau TS.

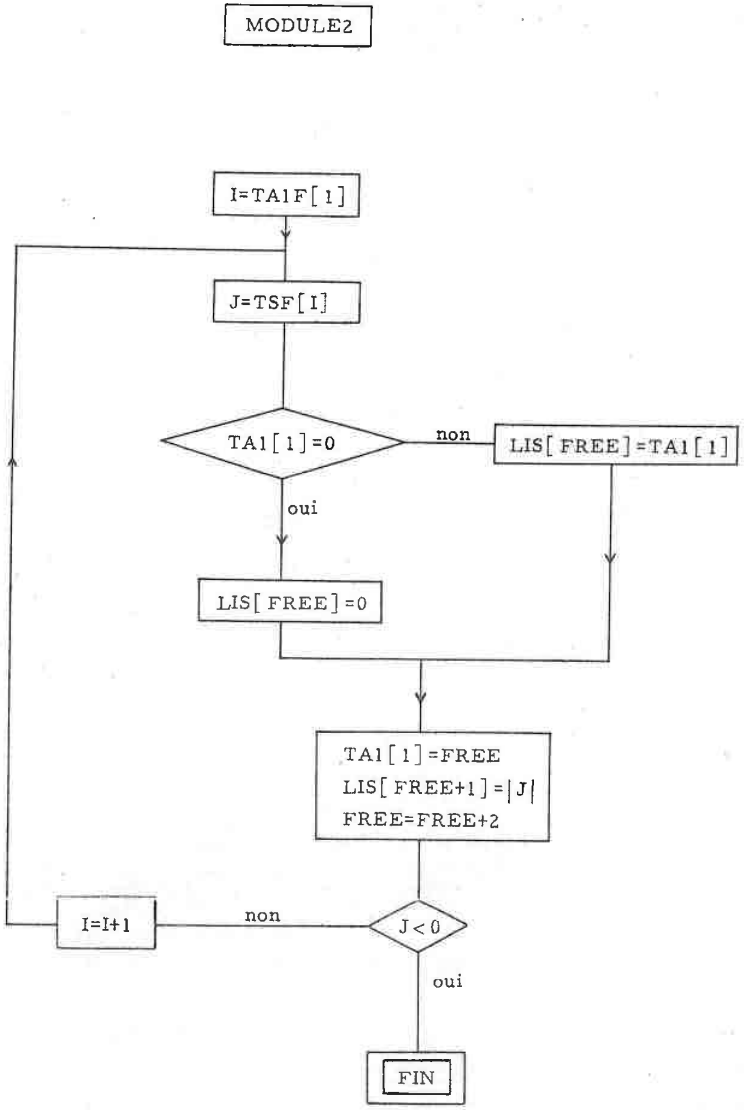
Algorithme général

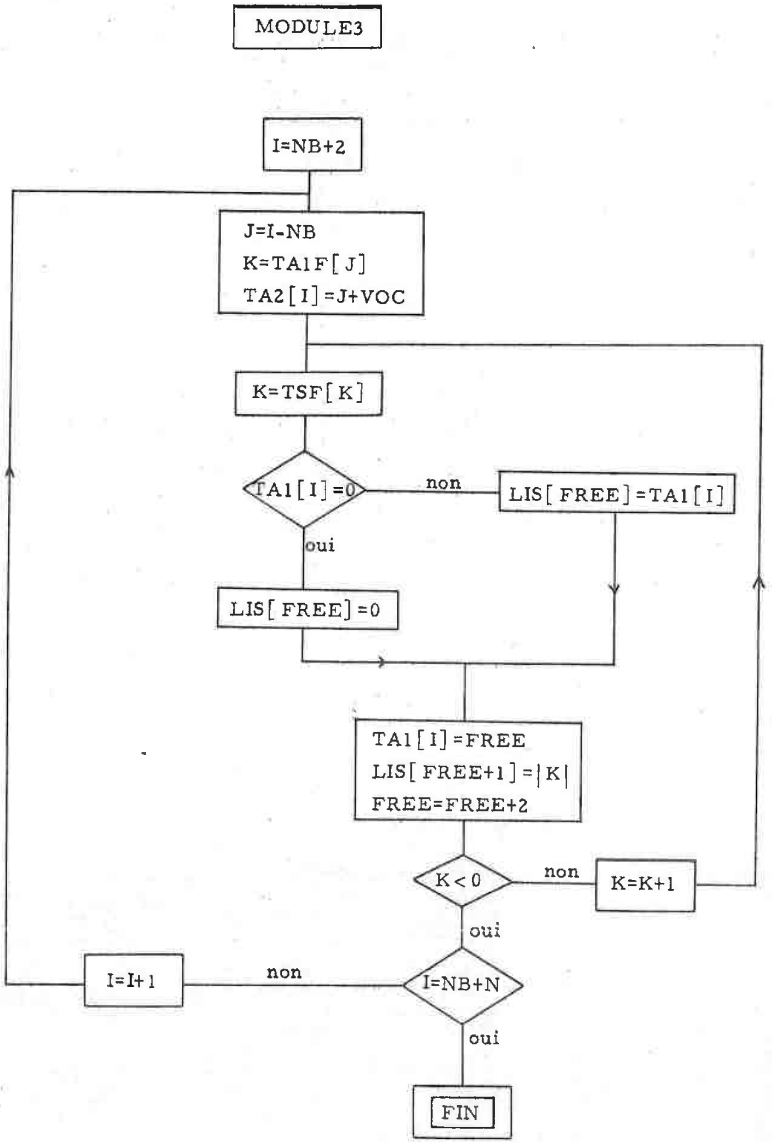


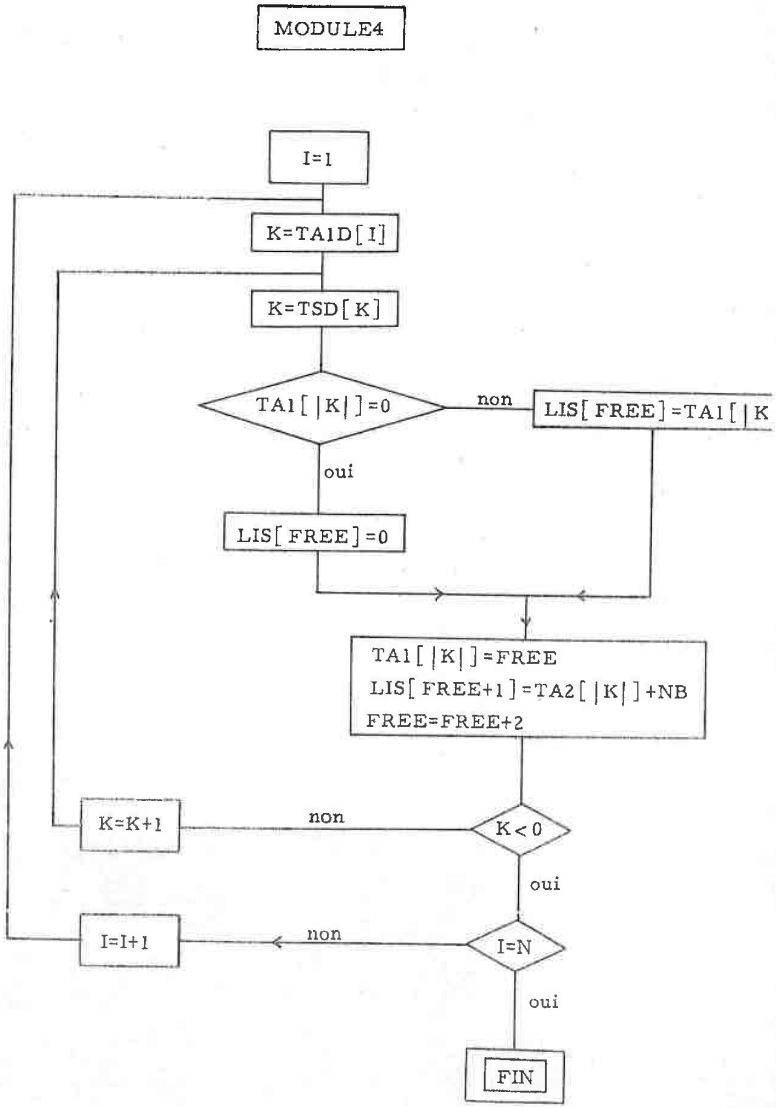
3. Définition en mémoire de g^*

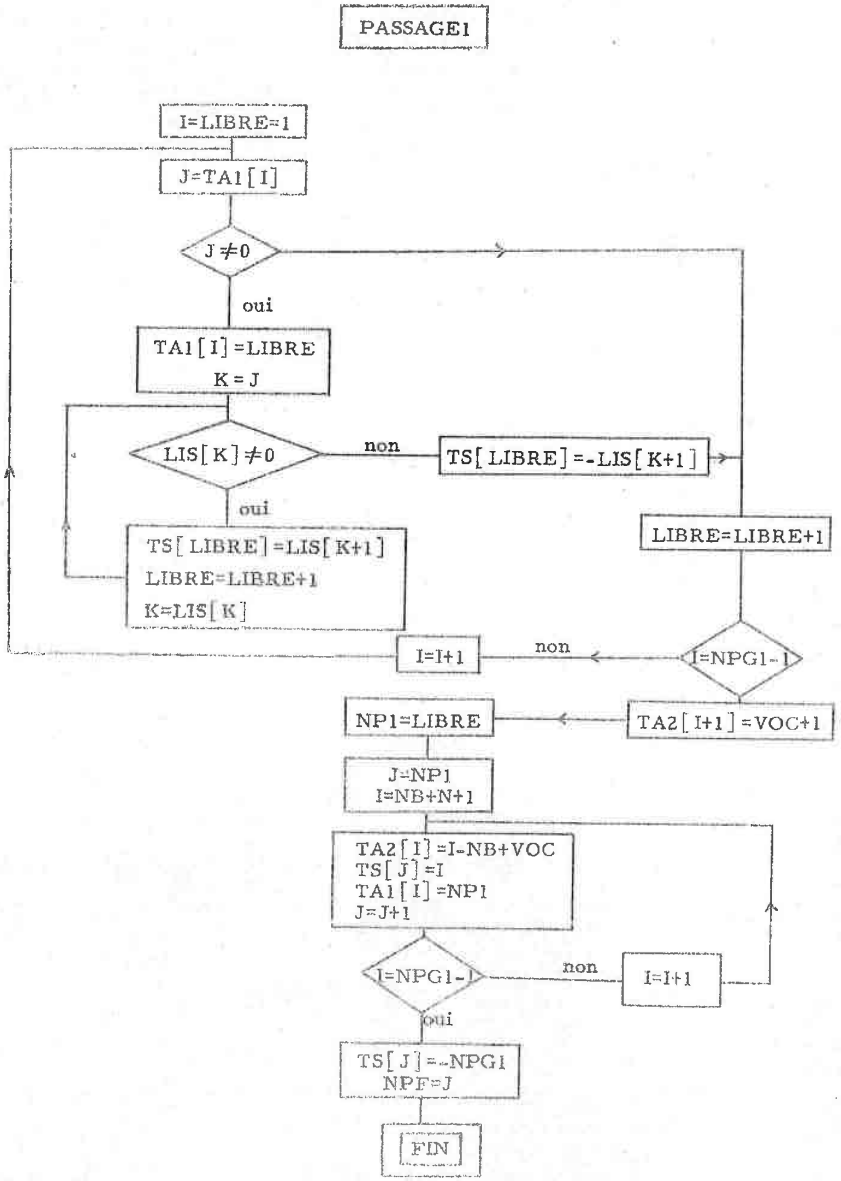
La restriction à V de g étant l'identité sur V, une procédure simple pourra être utilisée pour définir, g^*, en mémoire (Voir annexe).


MIROIR 1
I=2
K=TAB1[I]
TA2[I]=TAB2[I]
K≠0
non
oui
L=TABS[K]
TA1[|L|]=0
non
LIS[FREE]=TA1[|L|]
oui
LIS[FREE]=0
TA1[|L|]=FREE
LIS[FREE+1]=I
FREE=FREE+2
L<0
non
K=K+1
oui
I=I+1
non
I=NB+1
oui
FIN


MODULE2
I=TA1F[1]
J=TSF[I]
TA1[1]=0
non
LIS[FREE]=TA1[1]
oui
LIS[FREE]=0
TA1[1]=FREE
LIS[FREE+1]=|J|
FREE=FREE+2
I=I+1
non
J<0
oui
FIN


MODULE3
I=NB+2
J=I-NB
K=TA1F[J]
TA2[I]=J+VOC
K=TSF[K]
TA1[I]=0
non
LIS[FREE]=TA1[I]
oui
LIS[FREE]=0
TA1[I]=FREE
LIS[FREE+1]=|K|
FREE=FREE+2
K<0
non
K=K+1
oui
I=I+1
non
I=NB+N
oui
FIN


MODULE4
I=1
K=TA1D[I]
K=TSD[K]
TA1[|K|]=0
non
LIS[FREE]=TA1[|K
oui
LIS[FREE]=0
TA1[|K|]=FREE
LIS[FREE+1]=TA2[|K|]+NB
FREE=FREE+2
K=K+1
non
K<0
oui
I=I+1
non
I=N
oui
FIN


PASSAGE1
I=LIBRE=1
J=TA1[I]
J≠0
oui
TA1[I]=LIBRE
K=J
LIS[K]≠0
non
TS[LIBRE]=-LIS[K+1]
oui
TS[LIBRE]=LIS[K+1]
LIBRE=LIBRE+1
K=LIS[K]
LIBRE=LIBRE+1
I=I+1
non
I=NPG1-1
NP1=LIBRE
TA2[I+1]=VOC+1
J=NP1
I=NB+N+1
TA2[I]=I-NB+VOC
TS[J]=I
TA1[I]=NP1
J=J+1
I=NPG1-1
non
I=I+1
oui
TS[J]=-NPG1
NPF=J
FIN

II. Structure S_2

Le quadruplet associé que nous choisissons est (V_1, I_{V_1}, K, h'), avec

$$K = \# [K_2 \cap T(V \cup \overline{V})^*]^- \overline{\#} \quad \text{et} \quad K_2 = \Big(\bigcup_{A \in V} \overline{\widetilde{K}}_A A \Big)^* \overline{\widetilde{x}}.$$

Il nous faut déterminer le langage local L_2 et définir la transcription t_2, tels que $K = t_2(L_2)$ - Comme au § 1, le tableau TA2, associera à tout x de L_2, l'élément $t_2(x)$ de K.

1. Etude de la mémorisation de K

Nous allons construire, comme au paragraphe I, une suite de langages réguliers : $K^0, K^1, K^5, K^6, K^7, K^8$.

1.1) Les langages réguliers K^0 et K^1 sont ceux que nous avons définis aux paragraphes I 1.1 et I 1.2 :

$$K^1 = t^0(L^1) = \widetilde{K}^0 = \widetilde{x} \cup \Big(\bigcup_{A \in N} \widetilde{K}_A \Big).$$

1.2) $K^5 = t^5(L^5) = \overline{K^1} = \overline{\widetilde{x}} \cup \Big(\bigcup_{A \in N} \overline{\widetilde{K}}_A \Big)$, avec pour L^5, le langage local $(V', F, D, g\ell^1)$. Il nous est loisible de choisir, L^5, identique à L^1 : ils possèdent en effet les mêmes transitions, puisque le langage régulier K^5 se déduit de K^1, en remplaçant dans chacun de ses mots une parenthèses ouvrante par la parenthèse fermante, qui lui est associée biunivoquement.

La définition de la transcription t^5 est alors immédiate :

$$\forall x \in V' \qquad t^5(x) = t^0(x) + VOC \qquad \text{(remarque 1 ; § 3.3.1.1)}$$

1.3) $K^6 = t^6(L^6) = \overline{\widetilde{x}}\ \overline{\#} \cup \Big(\bigcup_{A \in V} \overline{\widetilde{K}}_A A \Big)$, avec $L^6 = (V^4 - \{1\}, F \cup V_T, W, g\ell^6)$, et $W = V_T \cup V_N$. Les ensembles V^4 et V_T ont été définis précédemment, ainsi que l'entier NPG1 : NPG1=NB+VOC+1.

La transcription t^6 est donnée par :

- $(\forall x \in V')\ (t^6(x) = t^5(x))$ (V' est défini au § I)
- $t^0(NPG1) = VOC+1$
- $(\forall x \in V_N \cup V_T)\ (t^6(x) = x - NB)$

Quant au graphe $g\ell^6=(V^4-\{1\}, \Gamma^6)$:

$-\ (\forall x, y \in V')\ \ (x\Gamma^6 y \iff x\Gamma^1 y)$

$-\ (\forall x \in V_N)\ \ (\forall y \in D(x\text{-}NB))\ (y\Gamma^6 x)$, ce qui nous donne "les transitions du type 7" :

$$(\forall A \in N)\ \ (\forall B \in t_A(D_A))\ (\overline{B}A)$$

$-\ (\forall x \in D(1))\ \ (x\Gamma^6 NPG1)$; on crée ainsi les "transitions du type 8" : $(\forall B \in t_0(D_0))\ (\overline{B}\,\overline{\#})$.

1.4) $K^7=t^6(L^7)=\Big(\bigcup_{A\in V} \overline{\overline{K}}_A A\Big)^* \overline{\overline{X}}\ \overline{\#} \cap L^-(V_1)$, avec pour L^7, le langage local $(V^4-\{1\}, F'\cup V_T, NPG1, g\ell^7)$. L'ensemble F' est égal à $F-F_0$, mais comme nous identifions F et le graphe, $(\mathbb{N}, F)$, qui le représente, F' sera le sous-graphe de obtenu en supprimant dans F, le point $\{1\}$.

Définissons le graphe $g\ell^7=(V^4-\{1\}, \Gamma^7)$:

$$-\ (\forall x, y \in V^4-\{1\})\ \ (x\Gamma^6 y \Longrightarrow x\Gamma^7 y)$$

$$-\ (\forall z \in V_T \cup V_N)\ \ (\forall y \in F[t^6(z)])\ \ (t^6(y)=t^6(z)+VOC,\ z\Gamma^7 y)$$

On crée les "transitions du type 9" :

$$(\forall A \in t_0(F_0) \cup \Big(\bigcup_{C\in N} t_c(F_c)\Big))\ \ (A\overline{A})$$

$-\ (\forall x \in V_T)\ \ (\forall y \in V_T)\ \ (x\Gamma^7 y)$, soit les "transitions du type 10" :

$$(\forall A \in T)\ \ (\forall B \in T)\ \ (AB).$$

1.5) $K^8=t^8(L^8)=\#\left[\Big(\bigcup_{A\in V} \overline{\overline{K}}_A A\Big)^* \overline{\overline{X}}\ \overline{\#} \cap L^-(V_1) \cap T(V\cup\overline{V})^* \overline{\#}\right]$

On a en particulier : $K^8=\#\left[T(V\cup\overline{V})^* \cap \Big(\bigcup_{A\in V} \overline{\overline{K}}_A A\Big)^* \overline{\overline{X}}\right]^- \overline{\#}=K.$

Le langage local L^8 est donc le langage local, L_2, que nous recherchions, et il est donné par : $(V^4, \{1\}, NPG1, g\ell^8)$.

Nous avons pour la transcription t^8 :

$$-\ (\forall x \in V^4-\{1\})\ \ (t^8(x)=t^6(x))$$

$$-\ t^8(1)=1.$$

Quant au graphe $g\ell^8=(V^4, \Gamma^8)$:

$$-\ (\forall x, y \neq 1)\ \ (x\Gamma^8 y \iff x\Gamma^7 y)$$

$-\ (\forall x \in V_T)\ \ (1\Gamma^8 x)$, soit "les transitions du type 11" :

$(\forall A \in T)\ \ (\#A)$.

2. Programmation de la génération de K

Il s'agit de construire la liste des familles des points du graphe des transitions du langage local L^8. Nous procéderons de la même façon qu'au paragraphe I. 2, et nou créerons une liste, LIS, qui sera celle des familles de $g\ell^8$, dans laquelle nous rangerons successivement les transitions de L^1, L^5, L^6, L^7 et L^8. Nous allons définir cinq modules, que nous exécuterons successivement, et qui engendreront ainsi L^8 et t^8 :

LECTURE , MIROIR1 , MIROIR2 , TRANSITION2 , et PASSAGE2.

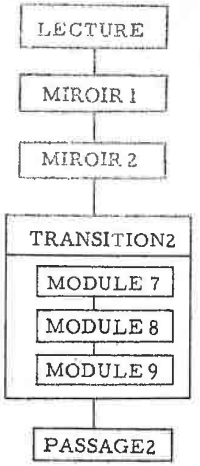
- Les modules LECTURE et MIROIR1, ont déjà été étudiés.
- Module MIROIR2 : Il crée le langage L^5 ; mais en fait il se résumera à une instruction définissant, t^5.
- Module TRANSITION2 : Ce module crée les transitions des types 7, 8, 9 ; un sous-module est associé à chacune d'elles, soit respectivement : MODULE7, MODULE8, MODULE9.
- Module PASSAGE2 : Il nous faut dans ce module définir le tableau des familles, TS à partir de la liste LIS.

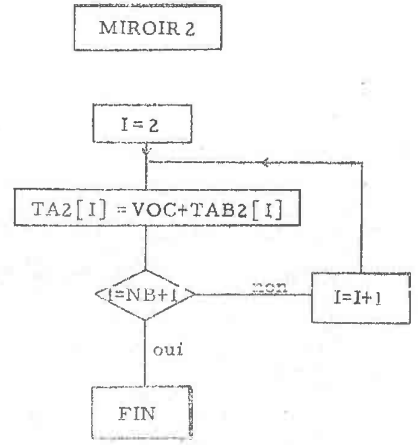
Les seuls successeurs des points de $g\ell^8$ que nous n'avons pas rangés dans LIS, sont ceux de $V_T\cup\{1\}$ qui répondent aux transitions des types 10 et 11 ; alors que précédemment l'ensemble des successeurs des éléments de V_T était précisément V_T, il n'en constitue plus maintenant, qu'une sous-famille commune.

Après avoir rangé séquentiellement dans TS, les familles de LIS , nous devrons non seulement ranger à la fin de ce dernier la famille V_T, mais placer pour chacun des points de V_T qui possède une sous-famille de successeurs, I , différente de V_T , le marqueur zéro à la suite de I, comme nous l'avons convenu en 3. 1. 1.

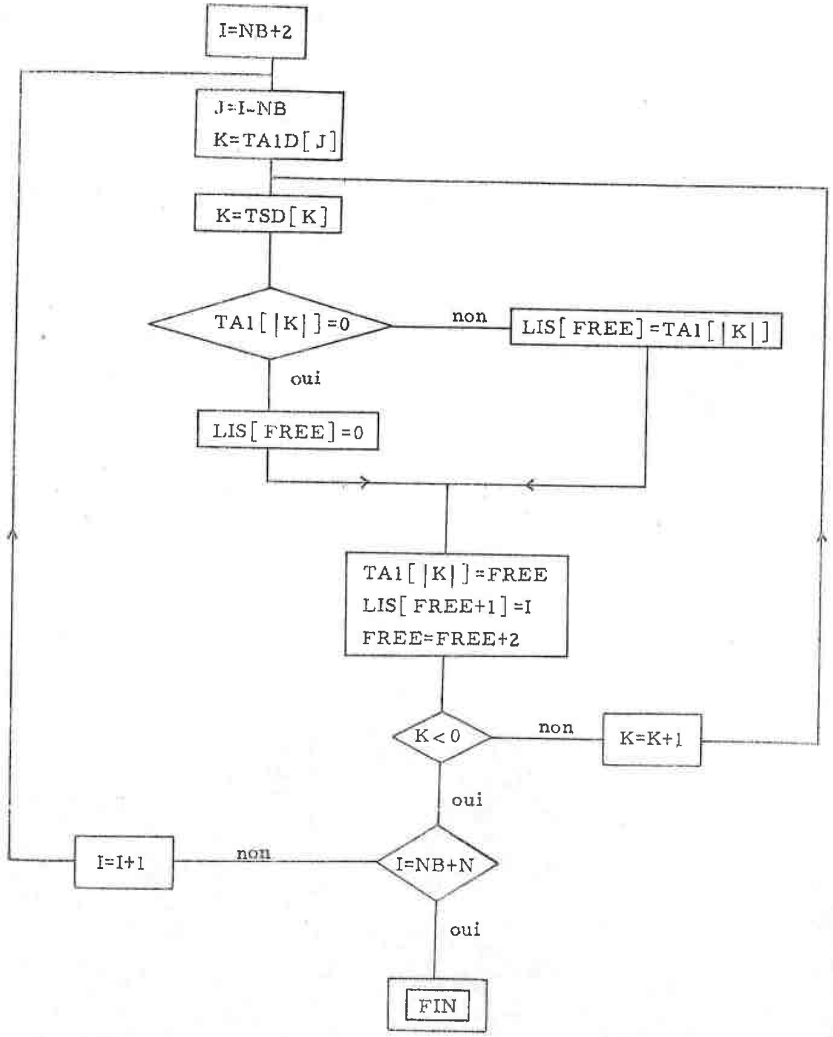
3. Définition de l'homomorphisme g^*, en mémoire

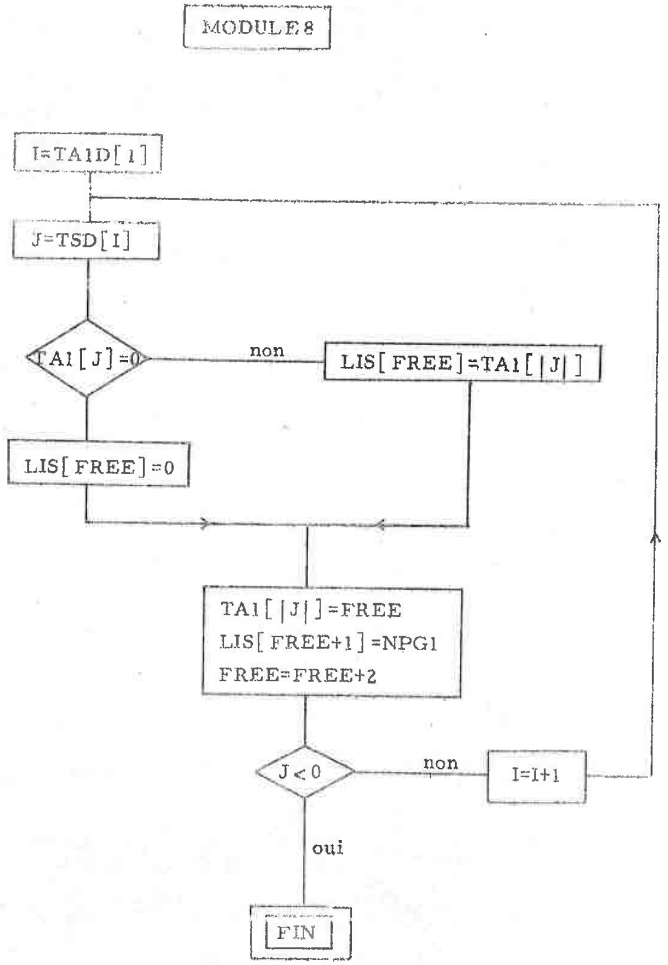
On se trouve dans le même cas que celui de la structure S_2 (§ I. 3).

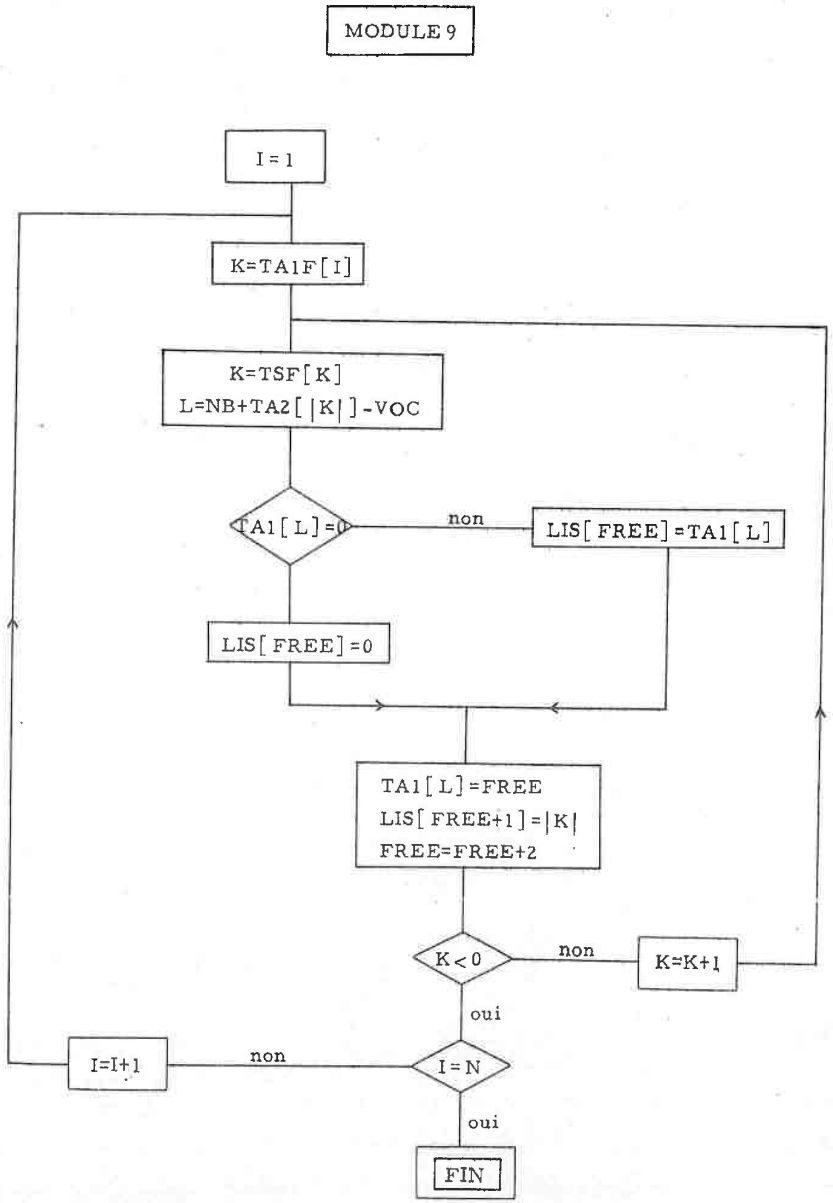

LECTURE
MIROIR 1
MIROIR 2
TRANSITION2
MODULE 7
MODULE 8
MODULE 9
PASSAGE2

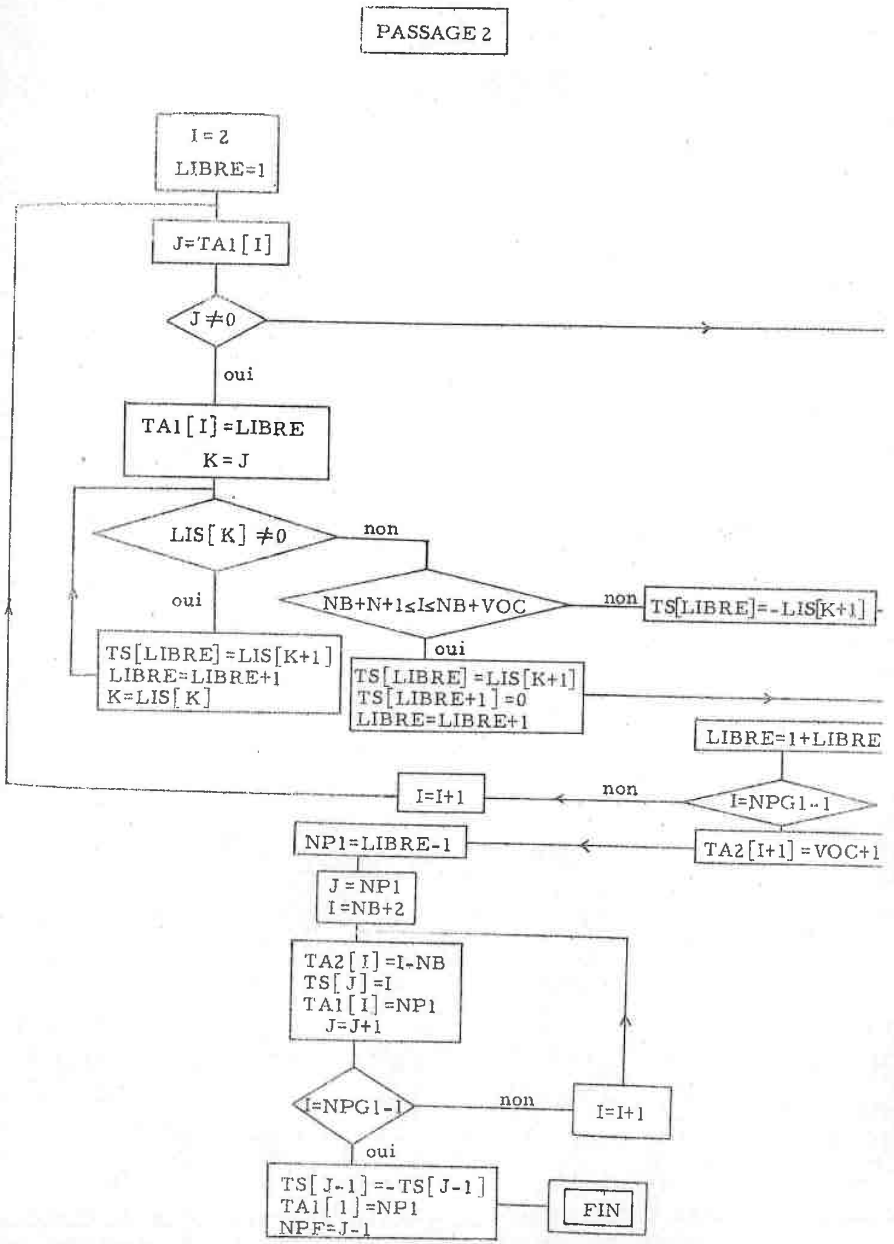

MIROIR 2
I = 2
TA2[I] = VOC+TAB2[I]
I=NB+1
non
I=I+1
oui
FIN

MODULE 7


I=NB+2
J=I-NB
K=TA1D[J]
K=TSD[K]
TA1[|K|]=0
non
LIS[FREE]=TA1[|K|]
oui
LIS[FREE]=0
TA1[|K|]=FREE
LIS[FREE+1]=I
FREE=FREE+2
K<0
non
K=K+1
oui
I=I+1
non
I=NB+N
oui
FIN


MODULE 8
I=TA1D[I]
J=TSD[I]
TA1[J]=0
non
LIS[FREE]=TA1[|J|]
LIS[FREE]=0
TA1[|J|]=FREE
LIS[FREE+1]=NPG1
FREE=FREE+2
J<0
non
I=I+1
oui
FIN


MODULE 9
I = 1
K=TA1F[I]
K=TSF[K]
L=NB+TA2[|K|]-VOC
TA1[L]=0
non
LIS[FREE]=TA1[L]
LIS[FREE]=0
TA1[L]=FREE
LIS[FREE+1]=|K|
FREE=FREE+2
K<0
non
K=K+1
oui
I=I+1
non
I=N
oui
FIN


PASSAGE 2
I = 2
LIBRE=1
J=TA1[I]
J≠0
oui
TA1[I]=LIBRE
K=J
LIS[K]≠0
non
oui
NB+N+1≤I≤NB+VOC
non
TS[LIBRE]=-LIS[K+1]
oui
TS[LIBRE]=LIS[K+1]
LIBRE=LIBRE+1
K=LIS[K]
TS[LIBRE]=LIS[K+1]
TS[LIBRE+1]=0
LIBRE=LIBRE+1
LIBRE=1+LIBRE
I=I+1
non
I=NPG1-1
NP1=LIBRE-1
TA2[I+1]=VOC+1
J=NP1
I=NB+2
TA2[I]=I-NB
TS[J]=I
TA1[I]=NP1
J=J+1
I=NPG1-1
non
I=I+1
oui
TS[J-1]=-TS[J-1]
TA1[1]=NP1
NPF=J-1
FIN

III. Structure S_1

1. Définition du quadruplet associé à S_1

Le quadruplet associé que nous choisissons est (V'_1, g, L_1, h') ; ce quadruplet est celui que nous avions défini en 2.3.1, en tenant compte de l'introduction des marqueurs $\#$ et $\overline{\#}$:

- L_1 est le langage local, $(V'_1 \cup \overline{V}'_1, \mathcal{E}, \mathcal{E}', g\ell_1)$, avec $V'_1 = V' \cup \mathcal{E}$ et $\overline{V}'_1 = \overline{V}' \cup \mathcal{E}'$; $g\ell_1 = (V'_1 \cup \overline{V}'_1, \Gamma_1)$ est le graphe des transitions I', (2.3.1), complété des transitions supplémentaires : $\forall x \in \overline{F}_0 \quad x \Gamma_1 \mathcal{E}'$

$$\forall x \in D_0 \quad \mathcal{E} \Gamma_1 x$$

- g est une application de V'_1 dans V pour laquelle :

$$g(\mathcal{E}) = g(\#) = \wedge \quad \text{et} \quad g(\mathcal{E}') = g(\overline{\#}) = \wedge$$

En effet, d'après la définition de $\mathcal{E}$ et $\mathcal{E}'$ (paragraphe 3.1.3), et puisque L_1 est un langage local, on a les identités : $\mathcal{E} \equiv \#$ et $\mathcal{E}' \equiv \overline{\#}$

- h' est l'homomorphisme alphabétique de V^* dans T^* défini en 2.3.3.3.

Nous avons convenu en 3.1.4 de représenter, dans la mémoire de l'ordinateur, la grammaire G sur un alphabet V, sous la forme d'une grammaire locale G' d'alphabet V', telle que : $S[G] = \hat{g}(S[G'])$; mais g et V' sont précisément les éléments qui apparaissent dans le quadruplet : (V'_1, g, L_1, h'), avec $V' = V'_1 - \{\mathcal{E}\}$. G étant ainsi identifié en mémoire au langage régulier $K^0 = t^0(L^0)$, avec pour L^0 le langage local, $(V', D, F, g\ell^0)$, le graphe $g\ell_1$ peut être aisément défini à partir des graphes : $(\mathbb{N}, D)$, $(\mathbb{N}, F)$ et $g\ell^0$.

2. Définition en mémoire de g^*

Le tableau TA2, qui apparaît dans les résultats des programmes de passage nous permettra d'associer à tout élément x de $V'_1 \cup \overline{V}'_1$, son transformé par g^*.

3. Définition du graphe $g\ell_1$

L'ensemble V'_1 étant identifié avec les entiers de 1 à NB+1, on associe à chaque x de V'_1, sa parenthèse fermante $\bar{x} = x+NB+1$, de telle sorte que, $g(\bar{x}) = t^0(x)+VOC$ puisque $g(x) = t^0(x)$ (on rappelle que, t^0, est définie par le tableau TAB2).

Le graphe $g\ell_1$ possède 2xNB+2 éléments, et on a :

1) $(\forall x = 1, 2, \ldots, NB+1)(1 \leq g(x) \leq N)(\forall y \in D[g(x)])(x \Gamma_1 y)$, ou ce qui est équivalent :

$$(\forall a \in g^{-1}(N))(\forall b \in D_{g(a)})\ (a\ b)$$

$$(\forall b \in D_0)\ (\#\ b)$$

2) $(\forall x = 1, 2, \ldots, NB+1)(N+1 \leq g(x) \leq VOC)(z = x+NB+1, x\Gamma_1 z)$, ou ce qui est équivalent : $(\forall a \in g^{-1}(T))\ (a\ \bar{a})$

3) $(\forall x = 1, 2, \ldots, NB+1)(1 \leq g(x) \leq N)(\forall y \geq NB+2)((y-NB-1) \in F[g(x)])$ $(z = x+NB+1 ; y\Gamma_1 z)$

Soit encore : $(\forall a \in g^{-1}(N))\ (\forall \bar{b} \in \bar{F}_{g(a)})\ (\bar{b}\ \bar{a})$

$(\forall b \in \bar{F}_0)\ (b \not\equiv)$

4) $(\forall x, z = 1, 2, \ldots, NB+1)\ (x\Gamma^0 z)\ (y = x+NB+1\ ;\ y\Gamma_1 z)$

Soit : $(\forall a \in V')\ (\forall b\ ;\ ab \in I)\ (\bar{a}\ b)$

Nous avons ainsi quatre types de successeurs.

4. Programmation

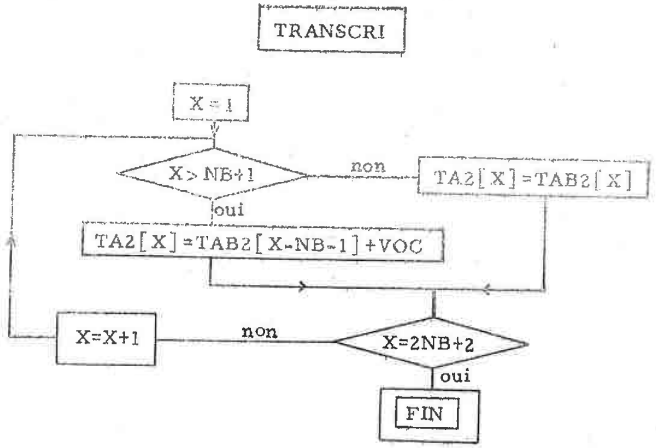
Il n'est plus nécessaire, comme dans le cas des structures S_2 et S_3, de construi des langages intermédiaires pour définir L_1 ; et de ce fait non plus de créer une liste auxiliaire des familles.

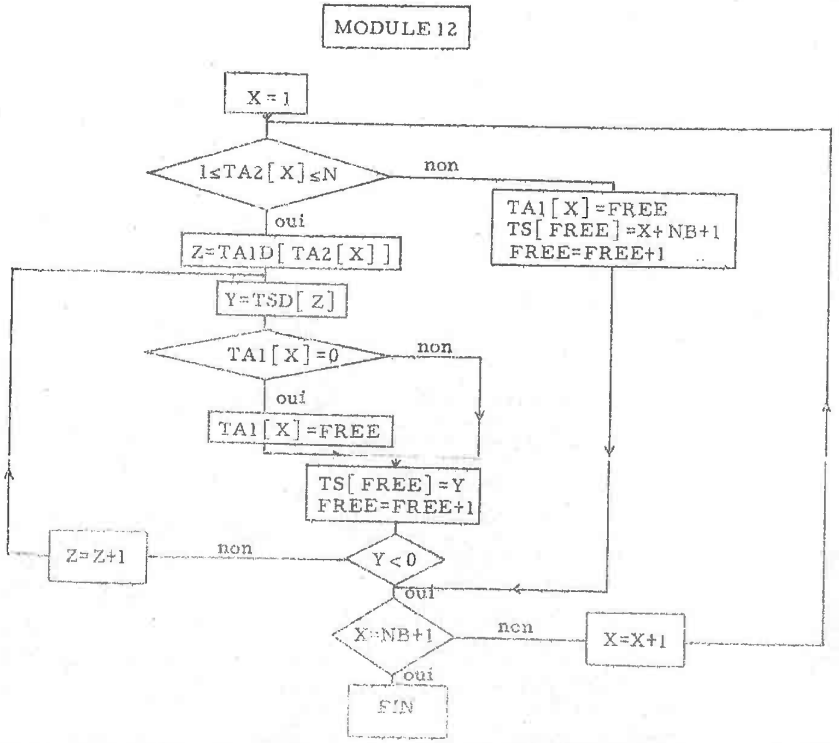
Nous définirons directement le tableau, TS, des familles de successeurs, en créa successivement les successeurs des types 1, 2, puis 3 et 4 ; ceci est rendu possible p le fait qu'un point ne peut avoir plus de deux types de successeurs :

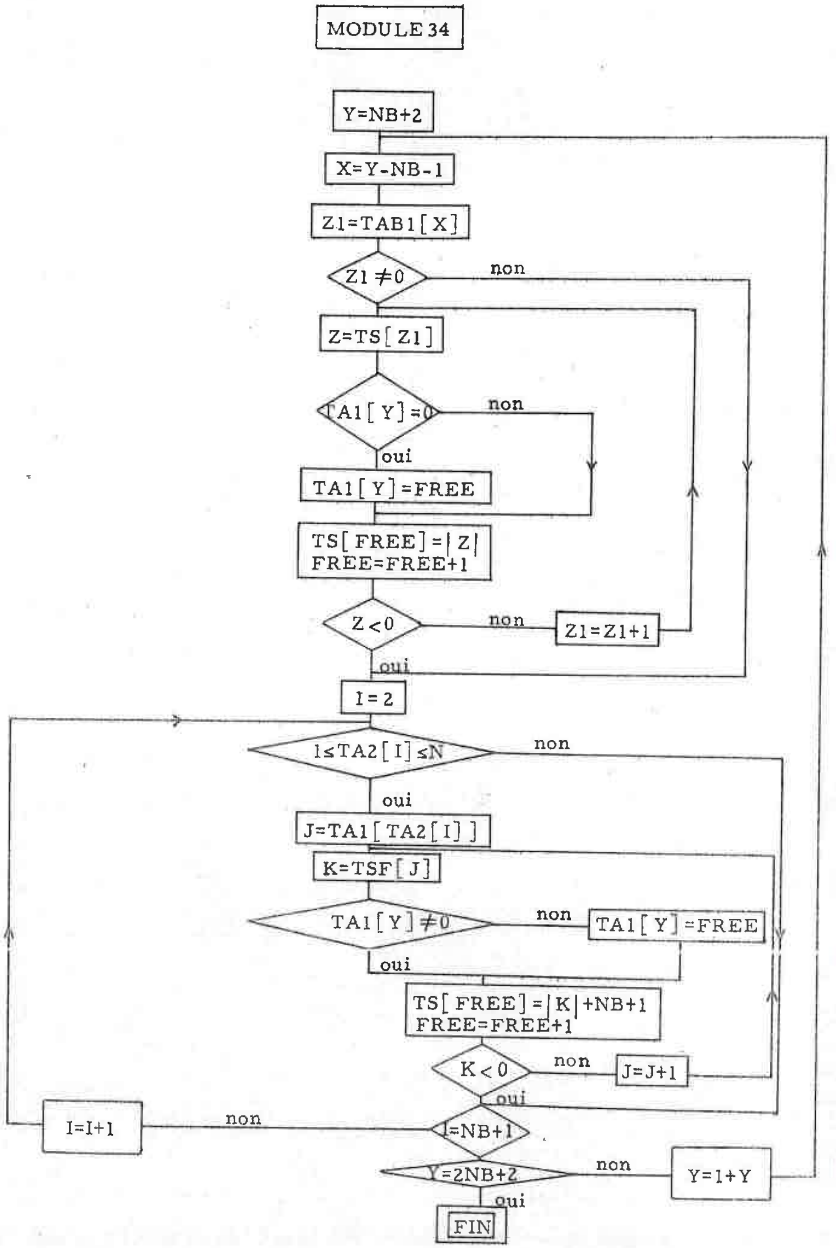
- soit du type 1
- soit du type 2
- soit des types 3 et 4.

Le programme qui définira L_1 s'articule en trois modules : LECTURE, TRANSC TRANSITION3.

Le module TRANSITION3 possède deux sous-modules MOD12 et MOD34, le premier définissant les successeurs des types 1 et 2 et le second ceux des types 3 et 4. Le module TRANSCRI définit l'homomorphisme g^*.






MODULE 34
Y=NB+2
X=Y-NB-1
Z1=TAB1[X]
Z1≠0
non
Z=TS[Z1]
TA1[Y]=0
non
oui
TA1[Y]=FREE
TS[FREE]=|Z|
FREE=FREE+1
Z<0
non
Z1=Z1+1
oui
I=2
1≤TA2[I]≤N
non
oui
J=TA1[TA2[I]]
K=TSF[J]
TA1[Y]≠0
non
TA1[Y]=FREE
oui
TS[FREE]=|K|+NB+1
FREE=FREE+1
K<0
non
J=J+1
oui
I=I+1
non
I=NB+1
Y=2NB+2
non
Y=1+Y
oui
FIN

3.4 - ETUDE DES CIRCUITS DES GRAPHES DES TRANSITIONS DES LANGAGES LOCAUX L_1 , L_2 , L_3

Nous ne nous intéresserons qu'aux circuits élémentaires, c , pour lesquels on a :

- (I) , dans le cas de L_1 : $h'0\,g^{*}(c) = \wedge$
- (II) , dans le cas de L_2 : $h'0\,g^{*}0\,t_2(c) = \wedge$
- (III) , dans le cas de L_3 : $h\,0\,g^{*}0\,t_3(c) = \wedge$

Les transcriptions t_2 et t_3 sont celles qui vérifient :

$$K_2 = t_2(L_2) \quad \text{et} \quad K_3 = t_3(L_3).$$

3.4.1 - Circuits du graphe des transitions de L_1

1) Etude des circuits sur x, appartenant à $V'_1(x \leq NB+1 \text{ et } 1 \leq g(x) \leq N)$:

Les seuls circuits sur x, $c = x_1 x_2 \ldots x_n$, $(x_1 = x_n = x)$ pour lesquels $h'0\,g^{*}(c) = \wedge$ sont ceux dans lesquels chacun des points, x_i , ne possède comme successeurs, que ceux du type 1 :

$$\forall\, i=1, 2, \ldots, n-1 \qquad x_{i+1} \in D[g(x_i)] \qquad (g(x_i) \in N)$$

Nous les appellerons, circuits de forme 1.

Mais on a l'équivalence :

$$x_{i+1} \in D[g(x_i)] \iff g(x_{i+1}) \text{ in } g(x_i)$$

Une condition nécessaire et suffisante pour qu'il n'y ait pas de circuit de ce type est la condition C1 :

«Le graphe des initiales des règles de la grammaire G ne possède pas de circuit»

2) Etude des circuits sur x, appartenant à $\bar{V}'_1(x \geq NB+2)$:

Les seuls circuits sur x , c , pour lesquels $h'0\,g^{*}(c) = \wedge$ doivent vérifier : $g^{*}(c) \in \bar{N}^{*}$; chacun des points de ce circuit ne possédant que des successeurs du type 3.

Ce sont les circuits de forme 2.

3.4.2 - Circuits élémentaires du graphe des transitions de L_2, vérifiant (II)

Soit c_0, un circuit élémentaire sur x :

$$c_0 = x_1 x_2 \cdots x_p \quad , \qquad x = x_1 = x_p .$$

3.4.2.1 - Supposons que pout tout entier i, inférieur ou égal à p, x_i appartienne à V'

Il est aisé de voir que $x_p x_{p-1} \cdots x_2 x_1$ est un circuit sur le point x de $g\ell^0$ (transitions du type 1, au paragraphe I, 1.1), et que l'on a : $t_2(x_i) \in \overline{V}$, puisque appartient à V'.

3.4.2.2 - Il existe un entier, i : $i \leq p$ et $x_i \notin V'$

Soit $i(0)$, le plus petit de ces entiers, i, supérieur à un. Cet entier existe, puisque si x_1 n'appartient pas à V', il en est de même pour x_p.
Deux cas se présentent :

- ou $c_0 = x_1 m_{i(0)} x_{i(0)} = x_1 m_{i(0)} x_p$
- ou $c_0 = x_1 m_{i(0)} x_{i(0)} c_1$, avec c_1, un chemin de longueur supérieure ou égale à un.

Dans les deux cas, nous pouvons mettre en évidence un certain nombre de propri

- $m_{i(0)} = x_2 x_3 \cdots x_{i(0)-1}$ et , x_k, appartient à V' pour tout entier k : $2 \leq k \leq i(0)-1$.

En outre d'après la condition (II), $x_{i(0)}$ appartient à V_N, et de ce fait la transiti $c_2^{-2} 0\, t_2(x_{i(0)-1})\ c_2^{-1} 0\, t_2(x_{i(0)})$ est du type 7 : $x_{i(0)-1} \in D[x_{i(0)}\text{-NB}]$.

On en déduit ainsi que , $x_{i(0)-1} x_{i(0)-2} \cdots x_2$, est un chemin du graphe $g\ell_{c_2^{-1} 0\, t_2(x_{i(0)})}$, avec $t_2(x_{i(0)}) = x_{i(0)}\text{-NB}$.

Cas 1) : $x = x_1 = x_p = x_{i(0)}$

En particulier, x appartient à V_N et , $c_2^{-1} 0\, t_2(x_1)\ c_2^{-1} 0\, t_2(x_2)$, doit être une transition du type 9 : $x_2 \in F(x_{i(0)}\text{-NB})$ et $t_2(x_2) = t_2(x) + \text{VOC}$. Il en résulte que, $x_{i(0)-1} x_{i(0)-2} \cdots x_3 x_2$, appartient à $L_{c_2^{-1} 0\, t_2[x_{i(0)}]}$, et ainsi on a :

Rappel : c_2 , est la codification entière du vocabulaire V de G.

$A ::= \varphi A$, si $\varphi = c_2^{-1}[\{t_2(x_{i(0)-1})\text{-VOC}\} \ldots \{t_2(x_4)\text{-VOC}\}\{t_2(x_3)\text{-VOC}\}]$ et $A = c_2^{-1} 0\, t_2(x)$.

Cas 2) $i(0) \neq p$ et $c_0 = x_1 m_{i(0)} x_{i(0)} c_1$

On a : $c_1 = x_{i(0)+1} \cdots x_p$.

Deux possibilités s'offrent de nouveau :

a) Pour tout entier k supérieur ou égal à $i(0)+1$, x_k appartient à V'

On en déduit que x appartient à V' et que $x_p \cdots x_{i(0)+1}$ est un chemin de $g\ell^0$; mais d'après l'étude précédente $x_{i(0)+1} x_{i(0)-2} \cdots x_2 x_1$ est un chemin de $g\ell_{c_2^{-1} 0\, t_2(x_{i(0)}}$, par conséquent $x_p \cdots x_{i(0)+1}$ appartient à l'ensemble des chemins de $g\ell_{c_2^{-1} 0\, t_2[x_{i(0)}]}$ les sous-graphes partiels de $g\ell^0$, $g\ell_A$ et $g\ell_0$, ayant des numérotations disjointes.
La transition, $c_2^{-1} 0\, t_2(x_{i(0)})\ c_2^{-1} 0\, t_2(x_{i(0)+1})$, devant être du type 9, $x_{i(0)+1}$ appartient à $F(t_2[x_{i(0)}])$, avec $t_2[x_{i(0)}] = x_{i(0)}\text{-NB} = t_2(x_{i(0)+1})\text{-VOC}$.
En conséquence, $x_{i(0)-1} x_{i(0)-2} \cdots x_2 x x_{p-1} x_{p-2} \cdots x_{i(0)+1}$, appartient au langage $L_{c_2^{-1} 0\, t_2[x_{i(0)}]}$: si nous posons $A = c_2^{-1} 0\, t_2[x_{i(0)}]$ et

$$\varphi = c_2^{-1}\{[t_2(x_{i(0)-1})\text{-VOC}]\,[t_2(x_{i(0)-2})\text{-VOC}] \ldots [t_2(x_1)\text{-VOC}] \ldots [t_2(x_{i(0)+1})\text{-VOC}]\}$$

on a : $A ::= \varphi A$

b) Il existe un entier k tel que : $i(0)+1 \leq k \leq p$ et x_k n'appartient pas à V' :

On peut ainsi définir $i(1)$ comme étant le plus petit de ces entiers k.
Nous pouvons reprendre la discussion que nous avons entreprise en 3.4.2.2, de telle sorte que nous définirons une suite de couples, $(m_{i(k)}, x_{i(k)})$, pour $k = 0, 1, \ldots, p-1$, pour laquelle on a :

- ou $c_0 = x_1 m_{i(0)} x_{i(0)} x_{i(0)+1} m_{i(1)} x_{i(1)} \cdots x_{i(k-1)+1} m_{i(k)} x_{i(k)} \cdots x_{i(p-2)+1} m_{i(p-1)} x_{i(p-1)} m_{i(p)}$ et x_p appartient à V' pour tout entier i supérieur ou égal à $i(p-1)$
- ou $c_0 = x_1 m_{i(0)} x_{i(0)} x_{i(0)+1} m_{i(1)} x_{i(1)} \cdots x_{i(k-1)+1} m_{i(k)} x_{i(k)} \cdots x_{i(p-2)+1} m_{i(p-1)} x_{i(p-1)}$ et $i(p-1) = p$.

Pour $k = 1, 2, \ldots, p-1$:

- $i(k)$ est le plus petit entier strictement supérieur à $i(k-1)$, pour lequel $x_{i(k)}$ n'appartient pas à V' et tel que $x_i \neq x$, pour tout entier i vérifiant : $i = i(k-1)+1, \ldots, i(k)$ (sauf éventuellement pour $i(p-1)$)

$\tilde{m}_{i(k)} = x_{i(k-1)+2} \cdots x_{i(k)-1}$, et si on pose : $A_k = c_2^{-1} 0 t_2[x_{i(k)}]$ et

$$\varphi_k = c_2^{-1}\left\{[t_2(x_{i(k)-1})\text{-VOC}] \ldots [t_2(x_{i(k-1)+2})\text{-VOC}]\right\} \quad \text{alors :}$$

$$A_k ::= \varphi_k A_{k-1}$$

- Examinons le cas, $i(p-1) = p$; x appartient à V_N, et d'après l'étude faite en 1) nous savons que, x_2 appartient à $F(x_{i(0)}\text{-NB})$, avec $t_2(x_2) = t_2(x_{i(0)})+\text{VOC}$; $x_{i(0)-1} \cdots x_3 x_2$ appartient à $L_{c_2^{-1} 0 t_2[x_{i(0)}]}$, donc en particulier :

en posant $A_0 = c_2^{-1} 0 t_2[x_{i(0)}]$, $A = c_2^{-1} 0 t_2(x)$ et,

$\varphi = c_2^{-1} 0 \left[\{t_2(x_{i(0)-1})\text{-VOC}\} \ldots \{t_2(x_2)\text{-VOC}\}\right]$, on a : $A_0 ::= \varphi A$.

Mais, $A_{p-1} = c_2^{-1} 0 t_2(x_{i(p-1)}) = c_2^{-1} 0 t_2(x_p) = A$, donc : $A ::= \varphi_{p-1} A_{p-2}$

D'où les règles de grammaire :

$$A_0 ::= \varphi A \; ; \; A_1 ::= \varphi_1 A_0 \; ; \ldots ; \; A_{p-2} ::= \varphi_{p-2} A_{p-3} \; ; \; A ::= \varphi_{p-1} A_{p-2},$$ ce qui est équivalent d'écrire : $A\, f_n A_0 f_n A_1 \cdots f_n A_{p-2} f_n A$.

Le graphe des finales de la grammaire possède un circuit sur A.

- Examinons le cas : $i(p-1) \neq p$, et x_k appartient à V' pour tout entier k supérieur ou égal à $i(p-1)+1$

En conséquence $\tilde{m}_{i(p)} = x_p x_{p-1} \cdots x_{i(p-1)+1}$ est un chemin de $g\ell^0$; mais $x_{i(0)-1} x_{i(0)-2} \cdots x_{i(p-1)+1}$ est un chemin de $g\ell_{c_2^{-1} 0 t_2[x_{i(0)}]}$, il en est donc de même pour $\tilde{m}_{ip}$ ($x_i = x_p = x$ et définition de $g\ell^0$). De plus la transition,

$c_2^{-1} 0 t_2[x_{i(p-1)}]\; c_2^{-1} 0 t_2[x_{i(p-1)+1}]$, est du type 9 : $x_{i(p-1)+1} \in F(x_{i(0)}\text{-NB})$ et $t_2[x_{i(p-1)+1}] = t_2[x_{i(p-1)}]+\text{VOC}$; ainsi le chemin,

$x_{i(0)-1} x_{i(0)-2} \cdots x_2 x x_{p-1} \cdots x_{i(p-1)+1}$, appartient à $L_{c_2^{-1} 0 t_2[x_{i(0)}]}$.

Si on pose : $\varphi = c_2^{-1}\left[\{t_2(x_{i(0)-1})\text{-VOC}\}\{\ldots\}\ldots\{t_2(x_{i(p-1)+1})\text{-VOC}\}\right]$,

on a : $A_0 ::= \varphi A_{p-1}$.

D'où les règles de grammaire :

$$A_0 ::= \varphi A_{p-1} \; ; \; A_1 ::= \varphi_1 A_0 \; ; \ldots ; \; A_{p-2} ::= \varphi_{p-2} A_{p-3} \; ; \; A_{p-1} ::= \varphi_{p-1} A_{p-2},$$

ce qui est équivalent d'écrire : $A_{p-1} f_n A_0 f_n A_1 f_n \cdots f_n A_{p-2} f_n A_{p-1}$

Le graphe des finales des règles de la grammaire G possède un circuit sur A_{p-1}.

En conclusion, le graphe des transitions de L_2 ne peut posséder que deux types de circuits élémentaires vérifiant (II) :

Forme 3 - ceux dus à l'existence de circuits élémentaires de $g\ell^0$, et qui vérifient : $t_2(c) \in \bar{V}^*$

- ceux dus à l'existence de circuits élémentaires pour le graphe des finales des règles de la grammaire G, et dont les transcrits par, t_2, se présentent sous les trois formes :

Forme 4 - $\bar{\psi}_1 A_2 \bar{A}_2 \bar{\psi}_2 A_3 \bar{A}_3 \cdots A_{p-1} \bar{A}_{p-1} \bar{\psi}_{p-1}$, avec $\bar{\psi}_1, \bar{\psi}_2, \cdots \bar{\psi}_{p-1}$, appartenant à $\bar{V}^*$ ($\bar{\psi}_1 \neq \wedge$, et $\bar{\psi}_{p-1} \neq \wedge$), et $A_2, \ldots, A_{p-1}$ appartenant à N.

Forme 5 - $\bar{A}_1 \bar{\psi}_1 A_2 \bar{A}_2 \bar{\psi}_2 \cdots A_{p-1} \bar{A}_{p-1} \bar{\psi}_{p-1} \bar{A}_1$, avec $\bar{\psi}_i$ (respectivement A_i) appartenant à $\bar{V}^*$ (respectivement N), pour $i = 1, 2, \ldots, p-1$.

Forme 6 - $A_1 \bar{A}_1 \bar{\psi}_1 A_2 \bar{A}_2 \bar{\psi}_2 \cdots A_{p-1} \bar{A}_{p-1} \bar{\psi}_{p-1} A_1$, avec $\bar{\psi}_i$ (respectivement A_i) appartenant à $\bar{V}^*$ (respectivement N).

3.4.3 - Circuits élémentaires du graphe des transitions de L_3, vérifiant (III)

Soit, $c_0 = x_1 x_2 \cdots x_p$, un circuit élémentaire sur x ($x_1 = x_p = x$).

3.4.3.1 - Supposons que pout tout entier, i, inférieur ou égal à p, x_i appartient à V' :

Il est aisé de démontrer que $x_p x_{p-1} \cdots x_2 x_1$ est un circuit sur x de $g\ell^0$ (transitions du type 1, au paragraphe I.1.1), et que l'on a : $t_3(x_i) \in V$ pour $i = 1, 2, \ldots, p$

3.4.3.2 - Il existe un entier i : $i \leq p$ et $x_i \notin V'$

Soit $i(0)$, le plus petit de ces entiers, i, supérieur à un. Deux cas se présentent :

- ou $c_0 = x_1 m_{i(0)} x_{i(0)} = x_1 m_{i(0)} x_p$
- ou $c_0 = x_1 m_{i(0)} x_{i(0)} c_1$.

Dans les deux cas, nous avons :

$m_{i(0)} = x_2 x_3 \cdots x_{i(0)-1}$ et x_k appartient à V' pour tout entier k : $2 \leq k \leq i(0)$.
En outre d'après la condition (III), $x_{i(0)}$ appartient à V_N et de ce fait la transition $c_2^{-1} 0 t_3[x_{i(0)-1}] c_2^{-1}[t_3(x_{i(0)})]$ est du type 4 : il existe un entier, y, avec $1 \leq y \leq N$, pour lequel, $x_{i(0)-1}$ appartient à $D(y)$, et $x_{i(0)} = NB + t_3(x_{i(0)-1})$.
On en déduit que, $x_{i(0)-1} \cdots x_2$, est un chemin de $g\ell_{c_2^{-1}(y)}$.

Cas 1) : $p = i(0)$

En particulier, x appartient à V_N et, $c_2^{-1}[t_3(x_1)]\ c_2^{-1} 0 t_3(x_2)$, est une transition du type 2 : $x_2 \in F(x\text{-}NB)$ (avec $t_3(x) = x\text{-}NB+VOC$).
Il en résulte que $y = x\text{-}NB + x$ et que, $x_{i(0)-1} x_{i(0)-2} \cdots x_3 x_2$, appartient à $L_{c_2^{-1}[t_3(x)\text{-}VOC]}$ et de plus, $t_3(x_{i(0)-1}) = t_3(x)\text{-}VOC$.

Si on pose, $A = c_2^{-1} 0 t_3(x_{i(0)-1})$ et $\varphi = c_2^{-1} 0 t_3(x_{i(0)-2} \cdots x_3 x_2)$, alors :

$$A ::= A\varphi$$

Cas 2) : $p \neq i(0)$ et $c_0 = x_1 m_{i(0)} x_{i(0)} c_1$

On peut envisager deux possibilités :

a) Pour tout entier k supérieur ou égal à $i(0)+1$, x_k appartient à V'

On en déduit que x appartient à V' et que $x_p \cdots x_{i(0)+1}$ est un chemin de $g\ell^0$, mais puisque $x_{i(0)-1} \cdots x_2 x_1$ est un chemin de $g\ell_{c_2^{-1}(y)}$, il en est de même pour $x_p \cdots x_{i(0)+1}$.

La transition, $c_2^{-1} 0 t_3[x_{i(0)}]\ c_2^{-1}[t_3(x_{i(0)+1})]$, doit être du type 2 :

$x_{i(0)+1} \in F(x_{i(0)}\text{-}NB)$; par conséquent $y = x_{i(0)}\text{-}NB$ et, $x_{i(0)-1} \cdots x_2 x x_{p-1} \cdots x_{i(0)+1}$ appartient à $L_{c_2^{-1}[t_3(x_{i(0)})\text{-}VOC]}$;

si nous posons $A = c_2^{-1}[t_3(x_{i(0)})\text{-}VOC]$ et $\varphi = c_2^{-1} 0 t_3[x_{i(0)-1} \cdots x_2 x_1 x_{p-1} \cdots x_{i(0)}]$ alors : $A ::= A\varphi$.

b) Il existe un entier k, tel que : $i(0)+1 \leq k \leq p$, et x_k n'appartient pas à

on définit, $i(1)$, comme étant le plus petit de ces entiers k. Nous pouvons alors, reprendre la même discussion qu'en 3.4.3.2, de telle sorte que nous définirons une suite de couples, $(m_{i(k)}, x_{i(k)})$, pour $k = 0, 1, \ldots p-1$, pour laquelle on a :

- ou, $c_0 = x_1 m_{i(0)} x_{i(0)} m_{i(1)} x_{i(1)-1} x_{i(1)} \cdots x_{i(k-1)} m_{i(k)} x_{i(k)-1} x_{i(k)}$

$$\cdots x_{i(p-2)} m_{i(p-1)} x_{i(p-1)-1} x_{i(p-1)} ,$$

et $i(p-1) = p$.

- ou, $c_0 = x_1 m_{i(0)} x_{i(0)} m_{i(1)} x_{i(1)-1} \cdots x_{i(k-1)} m_{i(k)} x_{i(k)-1} x_{i(k)}$

$$\cdots x_{i(p-2)} m_{i(p-1)} x_{i(p-1)-1} x_{i(p-1)} m_{i(p)} ,$$

et x_i appartient à V' pour tout entier, i, supérieur ou égal à $i(p-1)+1$.
Pour $k = 1, 2, \ldots, p-1$:

- $i(k)$ est le plus petit entier strictement supérieur à $i(k-1)$ pour lequel $x_{i(k)}$ n'appartient pas à V', et tel que x_i est différent de x pour $i = i(k-1)+1, \ldots, i(k)$ (sauf éventuellement pour $i(p-1)$).
- $m_{i(k)} = x_{i(k-1)+1} \cdots x_{i(k)-2}$, et si on pose : $A_k = c_2^{-1} 0 t_3[x_{i(k)-1}]$ (Rappelons que $t_3[x_{i(k)-1}] = t_3[x_{i(k)}]\text{-}VOC$) et $\varphi_k = c_2^{-1} 0 t_3[m_{i(k)}]$, alors : $A_{k-1} ::= A_k \varphi_k$;

Examinons le cas, $i(p-1) = p$: x appartient à V_N et d'après l'étude faite en 1) nous savons que x_2 appartient à $F(x\text{-}NB)$ (avec $y = x\text{-}NB$) ; $x_{i(0)-1} \cdots x_3 x_2$ appartient à $L_{c_2^{-1}[y]}$ et si nous posons, $A = c_2^{-1}(y)$, $A_0 = c_2^{-1} 0 t_3(x_{i(0)-1})$ et $\varphi = c_2^{-1} 0 t_3[x_{i(p-2)} \cdots x_3 x_2]$, alors : $A ::= A_0 \varphi$.

Mais $A_{p-1} = c_2^{-1} 0 t_3[x_{i(p-1)-1}] = c_2^{-1}\{t_3[x_{i(p-1)}]\text{-}VOC\} = c_2^{-1}[t_3(x)\text{-}VOC]$

$$= c_2^{-1}(x\text{-}NB) = A.$$

D'où les règles de grammaire :

$$A_0 ::= A_0 \varphi_1 ; A_0 ::= A_1 \varphi_1 ; \ldots ; A_{p-3} ::= A_{p-2} \varphi_{p-2} ; A_{p-2} ::= A \varphi_{p-1} ,$$

ce qui est équivalent à dire : A_0 in A_{p-2} in A_{p-3} in $\ldots$ in A_1 in A_0 in A.

Le graphe des initiales de la grammaire possède un circuit sur A.

Examinons le cas : $i(p-1)\neq e$ et x_k appartient à V' pour tout entier k supérieur ou égal à $i(p-1)+1$

En conséquence, $\tilde{m}_{i(p)}$ est un chemin de $g\ell^0$; mais $x_{i(0)-1}\cdots x_2 x_1$, étant un chemin de $g\ell_{c_2^{-1}(y)}$, il en est de même pour $\tilde{m}_{i(p)}$. La transition, $c_2^{-1}\circ t_3(x_{i(p-1)})\ c_2^{-1}\circ t_3(x_{i(p-1)+1})$, est du type 2 : $x_{i(p-1)+1}\in F(x_{i(p-1)}\text{-NB})$; y est donc égal à $[x_{i(p-1)}\text{-NB}]$ et $x_{i(0)-1}\cdots x_2 x x_{p-1}\cdots x_{i(p-1)+1}$, appartient à $L_{c_2^{-1}[t_3(x_{i(p-1)})\text{-VOC}]}$: en posant $A_0 = c_2^{-1}[t_3(x_{i(p-1)})\text{-VOC}]$ et $\varphi = c_2^{-1}\circ t_3(x_{i(0)-1}\cdots x_2 x x_{p-1}\cdots x_{i(p-1)+1})$ on a : $A_{p-1} ::= A_0\varphi$.

D'où les règles de grammaire :

$$A_{p-1} ::= A_0\varphi\ ;\ A_{p-2} ::= A_{p-1}\varphi_{p-1}\ ;\ \ldots\ ;\ A_1 ::= A_2\varphi_2\ ;\ A_0 ::= A_1\varphi_1\ .$$

Le graphe des initiales des règles de la grammaire G possède un circuit sur A_0 A_0 in A_{p-1} in A_{p-2} in $\ldots$ A_2 in A_1 in A_0 .

En conclusion, le graphe des transitions de L_3 ne peut posséder que deux types de circuits élémentaires vérifiant (III) :

- ceux dus à l'existence de circuits élémentaires de $g\ell^0$, et qui vérifient $t_3(c)\in V^*$ (Forme 7).

- ceux dus à l'existence de circuits élémentaires pour le graphe des initiales des règles de G et dont les transcrits par, t_3, se présentent sous les trois formes : $A_i\in N$ et $\psi_i\in V^*$, pour $i=1,\ldots,p-1$.

- Forme 8 - $\quad \psi_1 A_2 \bar{A}_2 \psi_2 A_3 \bar{A}_3 \ldots\ldots A_{p-1}\overline{A_{p-1}}\psi_{p-1}$ (ψ_1 et $\psi_{p-1}\neq$

- Forme 9 - $\quad \bar{A}_1 \psi_1 A_2 \bar{A}_2 \psi_2 \ldots\ldots\ldots A_{p-1}\overline{A_{p-1}}\psi_{p-1}\bar{A}_1$

- Forme 10 - $\quad A_1 \bar{A}_1 \psi_1 A_2 \bar{A}_2 \psi_2 \ldots\ldots A_{p-1}\overline{A_{p-1}}\psi_{p-1} A_1$.

CHAPITRE IV

Programmes d'Analyse

I - INTRODUCTION

Les programmes d'analyse construisent les ensembles, $\mathcal{A}(\alpha)$, (chapitre 2), associé à tout mot, α, de T^* que l'on désire analyser suivant la grammaire $G = (T \cup N, \rightarrow,)$ Ils admettent pour données, d'une part un quadruplet, (V_1, g, K_1, h), déterminé préalablement par les programmes du chapitre 3, à partir de G et de la structure compatible choisie S, et d'autre part un mot α de T^*.

Rappelons que :

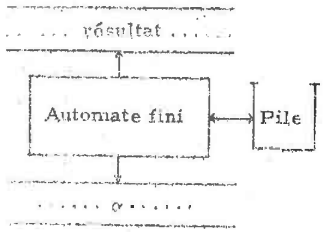
- K_1, est un langage régulier sur $V_1 \cup \bar{V}_1$, et que tous les mots de K_1 commencer par la lettre $\#$, n'appartenant pas à V $(V_1 = V \cup \#)$
- $W = T \cup N$ étant l'alphabet de G ; g est une application de V_1 dans W,
- h est un homomorphisme alphabétique de W^* dans T^*.

Les résultats de ces programmes sont les mots, m, transformés des analyses de c (c'est à dire des ramifications engendrées par G, et admettant α comme mot des feuilles), par l'isomorphisme de binoïdes universels, p, de $\tilde{W}$ dans le binoïde universel défini par la structure compatible S.

Si $K_1 = t(L_1)$ et $L_1 = (\mathcal{V}_1, \{\mathcal{E}\}, \{\mathcal{E}'\}, g\mathcal{L}_1)$ avec $g\mathcal{L}_1 = (\mathcal{V}_1, \Gamma_1)$, la recherche d'un ensemble, $\mathcal{A}(\alpha) = h^{-1}(\alpha) \cap g^*(K_1 \cap P(V_1))$, équivaut à celle des chemins, c, du graphe $g\mathcal{L}_1$, d'origine $\mathcal{E}$, et d'extrémité $\mathcal{E}'$, qui vérifient :
$\iota \circ t(c) = \Lambda$ et $h \circ g^* \circ t(c) = \alpha$.

Les mots de K_1 sont engendrés par un automate fini représenté par le graphe, $g\mathcal{L}_1$, et la transcription t : cet automate doit vérifier qu'ils appartiennent à $P(V_1)$ et pour cela il entretient une pile de reconnaissance de $P(V_1)$; mais il est également pourvu d'une donnée, le mot α ; au fur et à mesure qu'apparaissent les lettres d'un mot engendré, l'automate doit vérifier que leurs images par l'homomorphisme alphabétique, $h \circ g^*$, sont : soit le mot vide, soit les lettres successives de α. (on doit lire à chaque étape, une ou zéro lettre de α).

La machine ainsi obtenue comme modèle d'un algorithme d'analyse syntaxique est un transducteur à pile.



Les états de l'automate seront les différentes étapes de l'analyse ; nous les nommerons, «états analytiques» ; ils précisent à chaque in
- l'état de la pile de reconnaissance
- le facteur droit de α, non lu par l'automat
- le point du graphe gt_1, choisi par l'autom

L'automate ainsi construit est non déterministe ; on pourra le programmer de de façons :

- Une première méthode consiste à élaborer séquentiellement les différentes analyses, possibles, de α. Nous obtenons un premier type de programme : le programme de traitement par empilement et retour-arrière (Procédure TRAV
- La deuxième méthode consiste à mener simultanément les différentes analyses. Cette dernière est utilisée dans le programme de traitement en parallèle (Procédure TRAVPA).

4. 2 - ETUDE DU FONCTIONNEMENT DE L'AUTOMATE

L'automate doit engendrer les mots de $\mathcal{A}(\alpha)$, ou «analyses» :

$\gamma \in \mathcal{A}(\alpha) \iff (\exists\, \gamma' \in L_1)(\varepsilon\, 0\, t(\gamma') = \wedge$ et $h\, 0\, g^* 0\, t(\gamma') = \alpha$ et $\gamma = g^* 0\, t(\gamma'))$.

Si on appelle, «parcours», les mots de l'ensemble $\mathcal{A}'(\alpha)$ défini par :
$\mathcal{A}'(\alpha) = [\gamma' \in L_1 \mid \varepsilon\, 0\, t(\gamma') = \wedge$, $h\, 0\, g^* 0\, t(\gamma') = \alpha\}$, on a , $\mathcal{A}(\alpha) = g^* 0\, t[\mathcal{A}'(\alpha)]$.

Définition 1 : Un mot, $\gamma' = x_1 x_2 \dots x_n$, de $\mathcal{V}_1^*$ est un parcours si et seulement si :

- $x_1 = \mathcal{E}$, $x_n = \mathcal{E}'$
- $x_i \Gamma_1 x_{i+1}$, $\forall\, i = 1, 2, \dots n-1$
- $\varepsilon\, 0\, t(\gamma') = \wedge$, $h\, 0\, g^* 0\, t(\gamma') = \alpha$

Pour élaborer des analyses de α, l'automate devra rechercher les parcours et pour ce faire il effectuera deux types de calculs :

- calcul de $\varepsilon\, 0\, t(\gamma')$, avec γ' appartenant à $\mathcal{V}_1^*$
- calcul de $h\, 0\, g^* 0\, t(\gamma')$, avec γ' appartenant à $\mathcal{V}_1^*$.

4. 2. 1 - Calcul de $\varepsilon\, 0\, t(\gamma')$

Soit, $x = x_1 x_2 \dots x_i$, un mot de $\mathcal{V}_1^*$, posons :

$u_i = \varepsilon\, [t(x_1 \dots x_i)]$ et $u_0 = \wedge$.

On rappelle que $V_1 = V_U \#$ est l'alphabet du langage régulier K_1.

. Soit un parcours $\gamma' = x_1 x_2 \dots x_n$, alors les mots de $\mathcal{V}_1^*$, $u_0, u_1, \dots, u_n$, sont les états d'une pile sur V_1, que nous désignerons par $U = (u_0, u_1, \dots, u_n)$ (Voir remarque de la définition [1] du paragraphe 1. 5 au chapitre 1).

. Soit f l'application qui à l'entier i de $[0, p]$ (1) associe l'élément de V_1 dont i est l'occurrence dans le mot d'entrée de $U = (u_0, u_1, \dots, u_n)$; celle-ci définit une transcription f^{**} qui transforme une pile simple sur les entiers de 0 à p, $P = (P_0, P_1, \dots, P_n)$ en la pile U ; p étant l'entier tel que $n+1 = 2p$ (PAIR [1]).

Définition 2 : Si U est une pile définie par un parcours, et $P(U) = (P_0, P_1, \dots, P_n)$, la pile simple dont la transcription par f^{**} est U, nous appellerons e(i) l'entrée du sommet de l'état P_i de P.

Théorème 1 : $e(i) = \text{Max}\{j \leq i \mid u_j = u_i$ et $t(x_j) \in V_1 = V_U \#\}$

Démonstration :

1) Supposons que $t(x_i)$ appartienne à V_1 ; il est évident que $e(i) = i = \text{Max}\{j \leq i \mid u_j = u_i$ et $t(x_j) \in V_1\}$.

2) Supposons que i soit une sortie de la pile U ; en particulier i est une sortie de la pile P(U).

Posons $E(i) = \{j \leq i \mid u_j = u_i$ et $t(x_j) \in V_1\}$.

Par définition de e(i), on a :

$P_i = P_{e(i)}$ et $U_i = f^{**}(P_i) = f^{**}[P_{e(i)}] = u_{e(i)}$, par conséquent e(i)

(1) On notera $[a, b]$ l'ensemble des entiers de a à b.

Soit j_0 la sortie du sommet de $P_{e(i)}$; j_0 est le plus petit indice supérieur à i tel que : $|P_{j_0}| < |P_{e(i)}|$ (I.6, théorème 1 ; PAIR [1]). De plus, on a : $e(i) \leq i < j_0$ et pour tout entier k, tel que : $e(i) \leq k < j_0$, $P_{e(i)}$ est facteur gauche de P_k. On a en particulier :

$$(\forall\, k)\ (e(i) \leq k \leq i)\ (P_k = P_{e(i)}\alpha, \text{ avec } \alpha \text{ appartenant à l'itéré de } [1,p], \text{ soit } [1,p]^*)$$

a) si $k=i$, $k \notin E(i)$; en effet $t(x_i) \notin V_1$

b) Supposons, $e(i) < k < i$:

- si $\alpha \neq \wedge$, alors $k \notin E(i)$
- si $\alpha = \wedge$, alors ou $P_k = P_{e(i)} = P_{k-1}$ a, ou $P_{k-1} = P_k a = P_{e(i)}$

le premier cas est incompatible avec le fait que k est strictement inférieur à j_0 car $|P_{k-1}| < |P_k| = |P_{e(i)}|$ et $k-1 \geq e(i)$; il reste le second cas pour lequel i n'appartient pas à $E(i)$; k étant une sortie de la pile P, $t(x_k)$ n'appartient pas à V_1.

Il n'existe pas d'entier k appartenant à $E(i)$ et qui soit strictement supérieur à $e(i)$.

Théorème 2 : Soit i une sortie de la pile $P(U)$, alors : $e(i) = e(e(i-1)-1)$

Démonstration : i est une sortie de la pile U :

$u_{i-1} = u_i t(x)$ et $u_{i-1} = u_{e(i-1)} = u_{e(i-1)-1} t(y)$, par conséquent, $t(x) = t(y)$ et $u_i = u_{e(i-1)-1}$.

D'autre part, i étant la sortie du sommet de P_{i-1}, on a :

$|P_{e(i)}| = |P_i| < |P_{i-1}| = |P_{e(i)}|$; ce qui nous donne en appliquant le théorème 1 (I.6 ; PAIR [1]), $e(i) < e(i-1)$, ou encore $e(i) \leq e(i-1)-1$.

On en déduit : $e(i) = \mathrm{Max}\{j \leq e(i-1)-1 \mid u_j = u_i \text{ et } t(x_j) \in V_1\}$, et puisque $u_i = u_{e(i-1)}$, $e(i) = \mathrm{Max}\{j \leq e(i-1)-1 \mid u_i = u_{e(i)-1} \text{ et } t(x_i) \in V_1\} = e(e(i-1)-1)$

Définition 3 Soit $U = (u_0, u_1, \ldots, u_n)$ la pile associée à un parcours. On appelle couplé de i, pour tout entier i de $[1,n]$ et on le notera $c(i)$, l'entier qui vaut $e(i-1)$ si i est une entrée et $e(i)$ dans le cas contraire.

$c(i) =$ SI $e(i) = i$ ALORS $e(i-1)$ SINON $e(i)$

Propriété 1 L'entier $c(i)$ couplé de i est déterminé par les couplés des entiers inférieurs à i ; et $c(i)$ est l'un des entiers, $i-1$, $e(i-1)$ ou $c[e(i-1)]$.

Démonstration :

i) $t(x_i) \in V_1$: $c(i) = e(i-1)$; si $(i-1)$ est une entrée $e(i-1) = i-1$, sinon $e(i-1) = c(i-1)$

ii) $t(x_i) \notin V_1$: $c(i) = e(i) = e(e(i-1)-1) = c(e(i-1))$

Si $(i-1)$ est une entrée $c(i) = c(i-1)$, sinon $c(i) = c[c(i-1)]$.

On a ainsi :

- i et $(i-1)$ sont des entrées : $c(i) = i-1$
- i est une entrée, $(i-1)$ est une entrée : $c(i) = c(i-1)$
- i est une sortie, et $(i-1)$ est une entrée : $c(i) = c(i-1)$
- i et $(i-1)$ sont des sorties : $c(i) = c(c(i-1))$.

Théorème 3 : Soit $\gamma' = x_1 \ldots x_n$, un mot de L_1 et U la suite admettant comme éléments : $u_i = \iota[t(x_1 x_2 \ldots x_i)]$, pour $i = 1, 2, \ldots, n$, alors avec $u_0 = \wedge$:

$U = (u_0, u_1, \ldots, u_n)$ est une pile $\iff$ $c(n-1) = 1$

Démonstration

1) Soit γ' appartenant à L_1 et $u_n = \wedge$, autrement dit

U est une pile sur V_1 : $u_n = \iota[u_{n-1} \#] = \wedge$, donc $u_{n-1} = \#$, mais la seule entrée possible de $\#$ est un, puisque $\Gamma_1^{-1}(\mathcal{E}) = \emptyset$, d'où $c(n-1) = e(n-1) = 1$; $(n-1)$ est obligatoirement une sortie, car $\mathcal{E}'$ n'appartient pas à $\Gamma_1(\mathcal{E})$ et par conséquent $(n-1)$ ne peut être égal à un, seule entrée de $\#$ dans la pile.

2) Soit $c(n-1) = 1$; $(n-1)$ est différent de 1 puisque $\mathcal{E}'$ n'appartient pas à $\Gamma_1(\mathcal{E})$; on a ainsi, $c(n-1) = e(n-1) = 1$, et $u_{e(n-1)} = u_1 = u_{n-1} = \#$

$u_n = \iota[u_{n-1} t(\mathcal{E}')] = \iota[\# \#] = \wedge$.

La démonstration est établie compte-tenu des rappels du début de ce même paragraphe.

Le couplage nous permet de connaître chacun des états de la pile $U(u_0, u_1, \ldots, u_n)$ construite à partir d'un parcours $\gamma' = x_1 x_2 \ldots x_n$:

En effet, supposons que nous soyons à l'état, $p \leq n$ et que par conséquent nous connaissions les couplés, $c(i)$, pour les entiers, $i = 1, 2, \ldots p$.

- Soit $k(1)$, le plus grand entier de $[1, p-1]$, tel que : $c[k(1)] < c(p)$; nous pouvons ainsi construire une suite d'entiers, $k(i)$, pour $i = 1, 2, \ldots, i_p$ qui vérifie : si $c[k(i)]$ est différent de un, alors $k(i+1)$ est le plus grand entier de $[1, k(i)-1]$ pour lequel : $c[k(i+1)] < c[k(i)]$.

Nous avons l'égalité $c[k(i_p)] = 1$ et :

- si p est une entrée, $u_p = t\left(x_{c[k(i_p)]} \cdots x_{c[k(1)]}\, x_{c(p)} x_p\right)$

- si p est une sortie, $u_i = t\left(x_{c[k(i_p)]} x_{c[k(i_{p-1})]} \cdots x_{c[k(1)]} x_{c(p)}\right)$

Démontrons ce résultat par récurrence sur p :

Le résultat est évident pour p égal à un. Supposons le démontré jusqu'au rang (p-1) :

- Si (p-1) est une entrée, $u_{p-1} = t[x_{c[k(i_{p-1})]} \cdots x_{c[k(1)]} x_{c(p-1)} x_{p-1}]$.

Dans le cas où p est une entrée : $u_p = u_{p-1}\, t(x_p)$ et $c(p) = p-1$; (p-1) est le plus grand entier k de $[1, p-1]$, pour lequel $c(k) < c(p)$ $(c(p-1) = e(p-2) \leq p-2 < p-1)$.
Si p est une sortie, on a $c(p) = c(p-1)$ et $u_{p-1} = u_p \cdot y$ d'où
$u_p = t[x_{c[k(i_{p-1})]} \cdots x_{c[k(1)]} x_{c(p)}]$ et k(1) est le plus grand entier k de $[1, p-1]$ tel que : $c(k) < c(p)$.

- Envisageons maintenant le cas où (p-1) est une sortie :
$u_{p-1} = t[x_{c[k(i_{p-1})]} \cdots x_{c[k(1)]} x_{c(p-1)}]$.

Lorsque p est une entrée, le résultat est évident puisque $c(p) = c(p-1)$ et $u_p = u_{p-1}\, t(x_p)$. Quant au cas où p est une sortie,
$u_p = t[x_{c[k(i_{p-1})]} \cdots x_{c[k(2)]} x_{c[k(1)]}]$; mais $c[k(1)]$ est l'entrée du sommet de u_p : $c(p) = e(p) = c[k(1)]$.

Cette propriété nous permettra lors du "traitement en parallèle", de reconstituer les piles associées à chacune des rangées de la ramification des débuts de parcours en ne connaissant que les couplés de chacun des points.
L'étude précédente nous fournit pour tout mot $\gamma' = x_1 x_2 \cdots x_n$ de L_1 deux méthodes de calcul des mots $z\,0\,t(\gamma')$:

L'empilement, soit l'étude des variations des mots $u_i = z\,0\,t(x_1 \cdots x_i)$, pour les entiers i de $[1, n]$; et l'on a l'équivalence :

$$z\,0\,t(\gamma') = \Lambda \iff u_n = \Lambda .$$

Le couplage, soit le calcul des couplés, c(i), de chacun des entiers i de $[1, n]$

$$z\,0\,t(\gamma') = \Lambda \iff c(n-1) = 1.$$

4.2.2 - Calcul de $h\,0\,g^{*}\,0\,t(\gamma')$ pour tout γ' de L_1

Soit $\gamma' = x_1 x_2 \cdots x_n$ de L_1 : il suffit de calculer successivement les quotients à gauche β_i de α par $h\,0\,g^{*}\,0\,t(x_1 \cdots x_i)$: $\alpha = h\,0\,g^{*}\,0\,t(x_1 \cdots x_i)\,\beta_i$

- $\alpha = \beta_0$
- $\beta_i = h\,0\,g^{*}\,0\,t(x_i)\,\beta_{i+1}$

Si γ' est un parcours, on doit obtenir : $\beta_n = \Lambda$.

4.3.3 - Etude de la ramification des débuts de parcours

Nous allons dans ce paragraphe, définir le fonctionnement de l'automate en décrivant ses changements d'état ; et nous définirons une analyse de α à partir d'une suite d'états analytiques.

4.2.3.1 - Introduction de la notion de ramification des débuts de parcours

Définition 4

Un calcul de l'automate engendrant $\mathcal{A}(\alpha)$ est une suite de triplets, appartenant à $\mathcal{V}_1 \times V_1^{*} \times T^{*}$, $(x_i, u_i, \beta_i)_{i=1}^{i=n}$, qui vérifient :

- (I) $u_{i+1} = z[u_i\, t(x_i)] \neq 0$, $x_{i+1} \in \Gamma_1(x_i)$, $\beta_i = h\,0\,g^{*}\,0\,t(x_i)\,\beta_{i+1}$
- (II) $x_1 =$, $u_1 = \#$, $\beta_1 = \alpha$

On appellera pseudo-calcul toute suite de triplets de $\mathcal{V}_1 \times V_1^{*} \times T^{*}$, qui vérifient la condition (I).

Les états de l'automate, que nous avions appelés les états analytiques, pourront ainsi être identifiés aux triplets (x_i, u_i, β_i) d'un calcul $c = (x_i, u_i, \beta_i)_{i=1}^{i=n}$.
Le résultat d'un tel calcul, c, sera le mot de $\mathcal{V}_1$: $\gamma' = x_1 x_2 \cdots x_n$.
Les parcours sont les résultats des «calculs finaux», si on entend par ce terme les calculs dont le dernier état analytique est le triplet $(\mathcal{E}', \Lambda, \Lambda)$.
En fait, les parcours seront obtenus en déterminant leurs facteurs gauches de longueur croissante : nous appellerons «débuts de parcours», les facteurs gauches des résultats des calculs de l'automate :

Un mot $\gamma' = x_1 \cdots x_p$ de $\mathcal{V}_1^{*}$ est un début de parcours si et seulement si il existe un calcul $c = (x^i, u_i, \beta^i)_{i=1}^{i=n}$ pour lequel : $x^i = x_i$, pour $i = 1, 2, \ldots p$.

Puisque tous les facteurs gauches de parcours sont des débuts de parcours (la réciproque étant fausse en général), l'automate devra construire les débuts de parcours et par conséquent il devra engendrer tous les calculs possibles, en effectuant tous les changements d'états qui lui sont permis.

Changement d'état de l'automate

Soit $t_i = (x_i, u_i, \beta_i)$, le dernier état analytique d'un calcul, $c = (t_j)_{j=1}^{j=i}$, les nouveaux états analytiques que l'automate peut prendre pour élaborer un nouveau calcul, c', à partir de c, sont donnés par l'ensemble :

$$F[t_i] = \{t_k = (x_k, u_k, \beta_k) \in \mathcal{V}_1 \times V_1^* \times T^* \mid x_k \in \Gamma_1(x_i),\ \beta_i = h\,0\,g^*\,0\,t(x_k)\beta_k,\ u_k = \iota[u_i\, t(x_k)] \neq 0\}$$

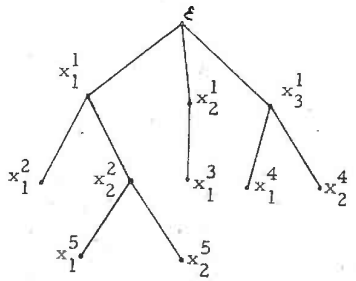
L'automate fonctionnera, en général, comme un automate indéterministe. On convient que les éléments, t_k, des ensembles $F(t_i)$, définis pour des états $t_i = (x_i, u_i, \beta_i)$ sont ordonnés suivant l'ordre de rangement dans le tableau des familles de successeurs des éléments x_k.

Ramification des débuts de parcours

L'ensemble des débuts de parcours peut être représenté par une ramification $\mathcal{R}(\alpha)$

Soit $t_0 = (\mathcal{E}, \#, \wedge)$ et $t_i^j = (x_i^j, u_i^j, \beta_i^j)$

Exemple :



Chaque rangée de $\mathcal{R}(\alpha)$ est le résultat d'un calcul :

- $F(t_0) = \{t_1^1, t_2^1, t_3^1\}$, $F(t_1^1) = \{t_1^2, t_2^2\}$, $F(t_2^2) = \{t_1^5, t_2^5\}$
- $\mathcal{E}\, x_1^1 x_1^2$ est le résultat du calcul ; $c_1 = (t_0, t_1^1, t_1^2)$
- $\mathcal{E}\, x_1^1 x_2^2 x_1^5$ est le résultat du calcul ; $c_2 = (t_0, t_1^1, t_2^2, t_1^5)$

Dans les paragraphes suivants nous allons étudier deux façons de construire la ramification $\mathcal{R}(\alpha)$; cependant il nous faut avant, poser le problème de

«l'arrêt des calculs de l'automate» : - l'automate peut-il prendre une infinité d'états analytiques ?

ou encore - existe-t-il une infinité de calculs ?

4.2.3.2 - «Arrêt» de l'automate

Le graphe, $g\mathcal{L}_1 = (\mathcal{V}_1, \Gamma_1)$, ne possédant qu'un nombre fini de points, il n'y aura qu'un nombre fini de calculs ; par contre cela n'implique pas que les calculs soient de longueur finie. Il nous faut étudier les conditions pour lesquelles les calculs de l'automate seront «finis».

Théorème 5 - Une condition nécessaire et suffisante pour que les calculs soient «finis» est que le graphe, $g\mathcal{L}_1$, ne possède pas de circuit élémentaire, $c = x_1 x_2 \ldots x_n$, avec $x_1 = x_n = x$ pour lequel, $h\,0\,g^*\,0\,t(c) = \wedge$, et tel qu'il existe un chemin, ℓ, de $G(\mathcal{E}, x)$ qui vérifie :

$$\forall\, n \geq 0 \quad \iota\,0\,t(\ell' c^n) \neq 0, \text{ avec } \ell = \ell' x.$$

Démonstration : 1) Soit c un tel circuit ; alors la suite de triplets, $s = (x^i, u^i, \beta^i)_{i \geq 1}$, définie par : si $\ell = \ell_1 \ldots \ell_p$,

($x^1 = \ell_1 = \mathcal{E}$ et $\ell_p = x = x^p$)

- $x^i = \ell_i$, pour $i = 1, 2, \ldots p$
- $x^i = x_{i-p+1}$, pour $i = p+1, \ldots, p+n-1$
- pour tout k supérieur à un : $x^{i+p+k(n-1)} = x^{p+i}$, pour $i = 1, \ldots, n-1$

. Quant aux mots u^i et β^i :

$$u^i = \iota\,0\,t[x^1 x^2 \ldots x^i], \text{ et } \beta^i = h\,0\,g^*\,0\,t(x^i)\ \beta^{i+1}, \text{ avec } \beta^1 = \alpha.$$

Compte-tenu des hypothèses du théorème, s est un calcul de l'automate qui est de longueur infinie.

2) Soit $s = (x^i, u^i, \beta^i)_{i \geq 1}$ un calcul de longueur infinie. Cela entraîne un certain nombre de conséquences :

- $g\mathcal{L}_1$ possède un chemin de longueur infinie
- il existe une infinité de x^i pour lesquels $h\,0\,g^*\,0\,t(x^i) = \wedge$, car $|\beta^{i+1}| \leq |\beta^i| - |\alpha|$, pour tout entier i supérieur à un.

Mais, $g\mathcal{L}_1$ ayant un nombre fini de points, la seule possibilité pour $g\mathcal{L}_1$ de posséder un chemin de longueur infinie est d'avoir un circuit élémentaire qui, répété une infinité de fois dans le déroulement du calcul s : soit r le résultat en x_i [illegible] pour lequel

- $c = x_1 \dots x_n$, avec $x_1 = x_n$, et x_{i+1} appartient à $\Gamma_1(x_i)$ pour tout entier i de $[1, n-1]$.

- $r = r' \pi_{i \geq 1} c_i$, où $\ell = r' x_1$ est un chemin de $\mathcal{C}(\mathcal{E}, x_1)$ et $c_i = c$, pour $i \geq 1$.

De plus, $h\,0\,g^*\,0\,t(c)$, est égal au mot vide, car sinon il existerait un entier k, pour lequel $|h\,0\,g^*\,0\,t(c^k)| > |\alpha|$, et par conséquent le calcul s ne pourrait pas être de longueur infinie.

On peut déduire de ce théorème une condition suffisante pour que les calculs soient finis ; ce qui nous permettra, dans le cadre de chacune des structures compatibles étudiées, de mettre en évidence des conditions simples qui lui sont équivalentes.

Une condition suffisante pour que les calculs de l'automate engendrant $\mathcal{A}(\alpha)$ soient ≪finis≫, est :

> Il n'existe pas de circuit élémentaire, $c = x_1 x_2 \dots x_p$, $(x_1 = x_p)$ du graphe $g\ell_1$ qui vérifie la condition (A) :
>
> $$(A) = \begin{cases} - \; h\,0\,g^*\,0\,t(c) = \Lambda \\ - \; (\exists \ell \in \mathcal{V}_1'^*)\,(\forall n \in \mathbb{N})\,(\ell\,0\,t(\ell c^n) \neq 0) \end{cases}$$

Structure S_1 :

Les circuits du graphe des transitions de L_1, de forme 1 vérifient la condition (A). On a déjà déterminé au paragraphe 3.4.1. 1) une condition nécessaire et suffisante pour qu'il n'y ait pas de tels circuits : la condition C1.

Les circuits de forme 2 ne vérifient pas la condition (A) : en effet si c est un de ces circuits, on a démontré en 3.4.1. 2) que c appartenait à $\overline{V}_1'$ (dans le cas de la structure S_1, on a $\mathcal{V}_1' = V_1' \cup \overline{V}_1'$ et t est l'identité sur V_1'). On en déduit immédiatement que pour tout mot ℓ de $\mathcal{V}_1'^*$, $\ell\,0\,t[\ell c^n]$ est nul, lorsque l'entier n, est strictement supérieur à la longueur de ℓ.

On peut ainsi énoncer :

> Une condition suffisante pour que les calculs soient ≪finis≫ est :
> ≪le graphe des initiales des règles de la grammaire G ne possède pas de circuit≫ (condition C1).

Structure S_2 :

Pour la même raison que celle évoquée pour les circuits de forme 2, soit la non satisfaction de la deuxième partie de (A), les circuits de forme 3, 4 ou 5 ne vérifient pas la condition (A).

Seuls les circuits de forme 6, c, pour lesquels :

$i_2(c) = A_1 \overline{A}_1 \overline{V}_1 A_2 \overline{A}_2 \overline{V}_2 \dots A_{p-1} \overline{A}_{p-1} \overline{V}_{p-1} A_1$, et où tous les mots $\overline{V}_i$ de $\overline{V}^*$ se réduisent au mot vide, satisfont à (A). Ce cas se présente lorsque la grammaire G possède de règles du type : $A_i ::= A_{i+1}$, pour $i = 1, 2, \dots, p-1$ et $A_p = A_1$.

On peut ainsi énoncer la condition (C2), suffisante, sous les deux formes équivalentes :

- un symbole auxiliaire ne peut pas dériver strictement de lui-même dans G.

- Il n'existe pas d'entier p pour lequel :

$A_i ::= A_{i+1}$, pour $i = 1, 2, p-1$, avec $A_p = A_1$; les lettres A_i étant des symboles auxiliaires (éléments de N).

Structure S_3 :

Les circuits de L_3 que nous avions étudiés en 3.4.3 vérifient tous la condition (A).

Deux conditions suffisantes peuvent être énoncées :

- la condition (C1), déjà étudiée.

et - la condition (C3) : ≪les graphes des transitions des langages locaux dont les transmits sont les langages de production ou l'ensemble des axiomes de G ne possèdent pas de circuits≫ ; ce qui est équivalent à la grammaire ne possède qu'un nombre fini de règles

ANALYSE PAR EMPILEMENT ET RETOUR ARRIERE

Une méthode pour parcourir la ramification $\mathcal{R}(\alpha)$ est l'utilisation de sa pile attachée (1.6 et PAIR [1]) qui nous permet d'engendrer successivement les rangées de $\mathcal{R}(\alpha)$. L'automate fonctionnera de façon déterministe en n'élaborant qu'un seul calcul au cours de sa recherche. Lors de l'analyse la ramification en cours de génération sera du type suivant :

Les pointillés indiquent les choix ultérieurs possibles pour l'automate.

Le calcul associé est donné par la suite : $(\mathcal{E}, \#, \alpha), (x_1^1, u_1^1, \beta_1^1)(x_1^2, u_1^2, \beta_1^2)$.

Le résultat de ce calcul est le début de parcours : $\mathcal{E} x_1^1 x_1^2$. Cependant si nous définissons une pile sur les éléments de $\mathcal{V}_1$, il nous faudrait à chaque entrée de l'un d'eux dans cette dernière, stocker en mémoire tous ceux qui appartiennent à une même famille de successeurs. Soit dans l'exemple précédent, lorsque x_1^1 entre dans la pile, il serait nécessaire de ranger en mémoire x_2^1 et x_3^1, de façon à pouvoir construire ultérieurement les rangées de $\mathcal{R}(\alpha)$, issues de ces derniers. En fait ces stockages sont inutiles si nous définissons une "pile attachée" P sur les adresses dans le tableau, TS, des familles de successeurs de chacun des points de $g\ell_1$; nous noterons $e\ell(x)$, le point de $g\ell_1$, rangé à l'adresse x, dans le tableau TS. Pour calculer les mots $u_1^1, u_1^2, \ldots$, nous définirons une pile, U, sur V_1 comme nous l'avons fait au paragraphe 4.2.1.

4.3.1 - Etude des changements d'état des piles P et U

S(i) (respectivement s(i)) désignera le sommet de l'état P_i (respectivement u_i) de la pile P (respectivement U). Nous indiquerons à chaque changement d'état des piles P et U, le dernier état analytique, $t_f^i = (x_i, u_i, \beta_i)$, du calcul associé à la rangée $\mathcal{R}(\alpha)$, en cours d'élaboration.

1) $P_0 = \Lambda$ et $u_0 = \Lambda$

2) $\mathcal{E}$, ne possédant pas de prédécesseur, on pose $P_1 = 0$; d'où $t_f^1 = (\mathcal{E}, \#, \alpha)$.

3) Soient P_i et u_i les états de même indice dans P et U :

- a) Si i est une entrée de la pile P et si $F(t_f^i) = \emptyset$, alors i+1 est la sortie du sommet S(i) : $t_f^{i+1} = t_f^{i-1}$ et $P_{i+1} = P_{i-1}$

- b) Si i est une entrée de P et si $F(t_f^i) \neq \emptyset$, alors i+1 est l'entrée de l'adresse, dans TS, du premier successeur de x_f^i ; soit $t_j = (x_j, u_j, \beta_j)$ le premier élément de $F(t_f^i)$, on a : $t_f^{i+1} = t_j$.

- c) i est une sortie de P :

  - c.1) si S(i-1) est l'adresse du dernier élément d'une famille (c'est à dire $TS[S(i-1)] < 0$), alors :

    - ou, $P_i = 0$ et (i+1) est le dernier état de P
    - ou, $P_i \neq 0$ et i+1 est une sortie

  - c.2) sinon, $TS[S(i-1)]$ n'est pas strictement négatif :

    + si $TS[S(i-1)+1]$ est différent de zéro, et si l'on a :

. $\tau[u_i t(e\ell\{S(i-1)+1\})] \neq 0$,

et , il existe un mot β de T^* tel que $\beta_i = h\,0\,g^*\,0\,t(e\ell\{S(i-1)+1\})\beta$.

alors i+1 est l'entrée de S(i-1)+1, avec : $x_{i+1} = e\ell\{S(i-1)+1\}$, $\beta_{i+1} = \beta$ et $u_{i+1} = h[u_i t(x_{i+1})]$.

+ si $TS[S(i+1)+1]$ est nul, et si l'entier NP1 vérifie les propriétés :

. $\tau[u_i t(e\ell\{NP1\})] \neq 0$, et il existe un mot β de T^* pour lequel

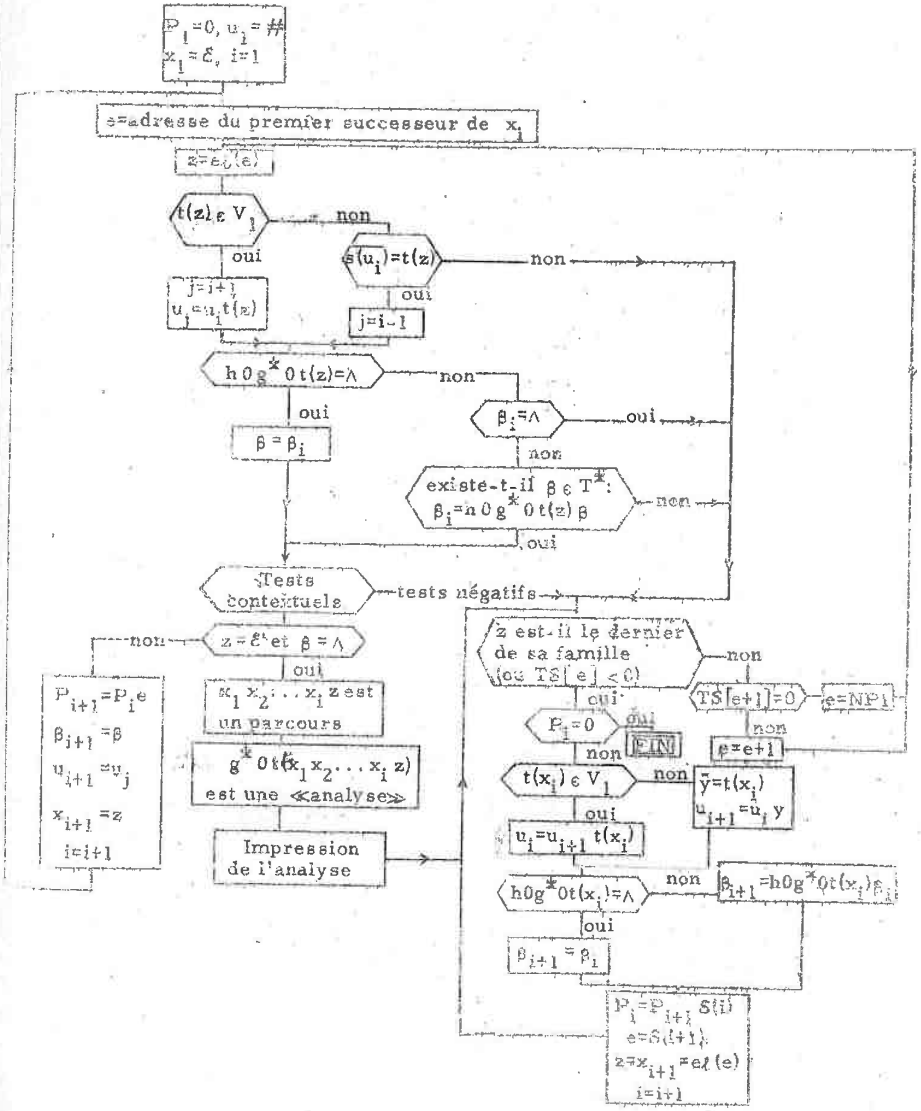
$\beta_i = h\,0\,g^*\,0\,t(e\ell\{NP1\})$, alors i+1 est l'entrée de NP1, avec :

$x_{i+1} = e\ell\{NP1\}$, $\beta_{i+1} = \beta$ et $u_{i+1} = h[u_i t(x_{i+1})]$.

On rappelle que NP1 est l'adresse d'une sous-famille de successeurs commune à plusieurs points du graphe $g\ell_1$.

+ dans les autres cas, i+1 est une sortie.

4.3.2 - Algorithme de fonctionnement de l'automate



4 - ANALYSE PAR TRAITEMENT EN PARALLELE

Le fonctionnement de l'automate de "traitement en parallèle" diffère du précédent par le fait qu'il correspond à celui d'un automate indéterministe qui génère simultanément un certain nombre de calculs $c_1, c_2, \ldots, c_p$ dont les états analytiques finaux possèdent tous la même troisième composante β :

$$- c_1 = (\mathcal{E}, \#, \alpha)(x_1^1, u_1^1, \alpha_1^1) \ldots\ldots (x_{n_1}^1, u_{n_1}^1, \beta)$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$- c_p = (\mathcal{E}, \#, \alpha)(x_1^p, u_1^p, \alpha_1^p) \ldots\ldots (x_{n_p}^p, u_{n_p}^p, \beta)$$

L'automate traite parallèlement les calculs qui ont lu le même nombre de lettres dans le mot α à analyser, en l'occurrence celles du mot γ dans l'exemple précédent avec $\alpha = \gamma\beta$. C'est la prochaine lettre à lire dans α qui dicte à l'automate l'ensemble des calculs stockés en mémoire qu'il doit traiter dans une étape ultérieure de l'analyse ; les calculs qui auront lu celle-ci seront interrompus.
La construction de la ramification, $\mathcal{R}(\alpha)$, peut ainsi être partagée en phases de calcul :

Définition d'un train de calculs : Etant donnée une liste, LIST, d'états analytiques (finaux) $t_f = (x, u, \beta)$, appartenant à des calculs de l'automate, «le train de calculs» associé à LIST est l'ensemble des pseudo-calculs $c = (c^i)_{i=1}^{i=n}$ dont l'état analytique initial, c^1, appartient à l'un des ensembles $F(t_f)$.
Nous convenons que : $c^i = (x_i, u_i, \beta_i)$ et que, α_q, désigne la première lettre de α non lue par l'automate.
. Un train de calculs sera appelé «phase de calcul» pour α_q, si les deux conditions suivantes sont vérifiées :
1) Pour tous les pseudo-calculs, c, on a :

$$- h0g^*0t(x_n) = \alpha_q$$

$$- h0g^*0t(x_1 \ldots x_{n-1}) = \wedge$$

2) Pour tous les états, t_f, de la liste associée :

- ou $h0g^*0t(x) = \alpha_{q-1}$ (α_{q-1} étant la dernière lettre de α lue par l'automate).

- ou x est égal à $\mathcal{E}$, et par conséquent la liste associée se réduit au triplet $(\mathcal{E}, \#, \alpha)$.

Etape de calcul : «une étape de calcul» est un train de calculs dont les pseudo-calculs sont de longueur un. Une phase de calcul associée à une liste, LIST, est ainsi obtenue en déterminant une suite d'étapes de calcul, définie de la façon suivante :

- la première étape est celle qui admet LIST comme liste associée.
- Toute étape différente de la première est constituée par le train de calculs admettant pour liste associée la sous-liste des états analytiques, $t=(x, u, \beta)$, de l'étape précédente, qui vérifient : $h\,0\,g^*\,0\,t(x) \neq \wedge$.
- La dernière étape est celle dont les pseudo-calculs, $c=(x, u, \beta)$, satisfont à : $h\,0\,g^*\,0\,t(x) = \alpha_q$.

. Si le mot α de T^*, que l'automate doit lire , a pour longueur l'entier ℓ, la génération de $\mathcal{R}(\alpha)$ se partagera en un nombre ℓ de phases de calcul, et un train de calculs dont les pseudo-calculs admettent tous le triplet, $(\mathcal{E}', \wedge, \wedge)$, comme état analytique final.

Le contrôle de l'empilement ne peut plus être réalisé à l'aide d'une seule pile, puisqu'à chacun des calculs, $c_1, \ldots, c_p$, définis à partir des pseudo-calculs des différentes phases de calcul est associée la pile $u^i=(u^i_1, u^i_2, \ldots, u^i_{n_i})$, pour $i=1, 2, \ldots p$; on utilisera ainsi la notion de couplage en associant à chaque état u^i_j, pour $j=1, \ldots, n_i$, d'une pile u^i son couplé, $c(j)$, qui nous permet de rechercher aisément l'élément qui peut sortir de la pile u^i, sans que nous connaissions les divers états u^i_j : $c(j)$ désigne en fait l'indice de l'état $u^i_{c(j)}$ de la pile u^i dont le sommet est, soit le dernier, soit l'avant-dernier élément qui est entré dans la pile u^i et qui ne possède pas de sortie.

4.4.1 - Mémorisation des calculs et des pseudo-calculs :

Les calculs sont stockés en mémoire sous forme d'une liste, LIS, mémoire unidimensionnelle possédant NIL éléments. Cette liste est divisée en enregistrements de quatre mots. En outre, les enregistrements rendus disponibles dans cette liste sont organisés en sous-liste par l'intermédiaire d'un pointeur placé dans le deuxième mot. L'adresse du premier enregistrement de cette sous-liste est définie par la variable FREE. Lorsque cette liste est vide, la création des enregistrements de LIS se fait séquentiellement ; cela se produit lorsque la variable FREE est égale à NIL et lorsque la sous-liste des enregistrements disponibles est vide.

Soit, t^p, un état analytique appartenant à la liste associée d'une étape de calcul ; c'est en particulier le dernier état d'un des calculs élaborés par l'automate : $c=(t^j)_{j=1}^{j=p}$, avec $t^k=(x_k, u_k, \beta_k)$, pour tout entier k de $[1, p]$.

Si l'ensemble, $F(t^p)$, n'est pas vide l'automate engendrera le train de calculs formé par les éléments de $F(t^p)$. L'ensemble $F(t^p)$ est la réunion de deux sous-ensembles disjoints $F'(t^p)$ et $F''(t^p)$, définis par :

$$F'(t^p) = \{ t=(x, u, \beta) \in F(t^p) \mid h\,0\,g^*\,0\,t(x) = \wedge \}$$

$$F''(t^p) = \{ t=(x, u, \beta) \in F(t^p) \mid h\,0\,g^*\,0\,t(x) \neq \wedge \}$$

. Pour chaque triplet, $t=(x, u, \beta)$, de $F'(t^p)$ (respectivement de $F''(t^p)$) l'automate crée un enregistrement de LIS :

- le deuxième mot contient l'adresse de l'enregistrement du triplet t' de $F'(t^p)$ (respectivement t'' de $F''(t^p)$) majorant de t suivant l'ordre que nous avions défini au paragraphe 4.2.3.1.

. Lorsque $F(t^p)$ se réduit à un seul triplet, t, le deuxième mot de l'enregistrement attaché à t contiendra zéro. Pour un ensemble $F(t^p)$ donné, le deuxième mot du dernier enregistrement créé contiendra l'adresse du premier enregistrement créé, affecté d'un signe moins.

En outre, lorsque $F'(t^p)$ et $F''(t^p)$ ne se réduisent pas à l'ensemble vide, le deuxième mot de l'enregistrement du dernier élément de $F'(t^p)$ contient l'adresse de celui qui est attaché au premier élément de $F''(t^p)$.

- x est rangé dans le premier mot.
- dans le troisième mot se trouve l'adresse d'implantation dans LIS, du triplet t^p ce qui nous permettra de reconstituer les rangées de la ramification $\mathcal{R}(\alpha)$ et ainsi d'imprimer les résultats ; les analyses de α.

Cette adresse sera affectée d'un signe moins, lorsque t est le dernier élément de $F'(t^p)$; ce qui nous donne un test de fin de liste pour la sous-liste de LIS, constituée par les enregistrements attachés aux sous-ensembles $F'(t^p)$.

- Quant au quatrième mot : le triplet, t, est le dernier état analytique du calcul, $c'=(x^\ell, u^\ell, \beta^\ell)_{\ell=1}^{\ell=p+1} = (t^\ell)_{\ell=1}^{\ell=p+1}$ défini à partir du calcul, $c=(x_k, u_k, \beta_k)_{k=1}^{k=p}$, introduit au début de ce paragraphe : $x_k=x^k$, $u_k=u^k$, $\beta_k=\beta^k$, pour $k=1, 2, \ldots p$, et $x^{p+1}=x$, $u^{p+1}=u$, $\beta^{p+1}=\beta$.

Les mots, u^ℓ (pour $\ell=1, 2, \ldots, p+1$), peuvent être considérés comme les états de rang inférieur ou égal à $p+1$ d'une pile $W=(w_0, w_1, \ldots, w_{p+1}, \ldots, w_n)$ avec $n \geq p+1$, et $w_i=u^i$, pour i appartenant à $[1, p+1]$, et $w_0=\wedge$. (Voir paragraphe 4.2).

Nous placerons dans le quatrième mot l'adresse, ADD, d'implantation dans la liste LIS de l'enregistrement attaché au triplet, $t^{c(p+1)}$; c(p+1) étant le couplé de l'entier, p+1, pour la pile W.

On peut aisément déterminer l'adresse ADD, compte-tenu de la propriété 1 du paragraphe 4.2.1.

Soit, AD, l'adresse d'implantation du triplet t^p ; trois cas se présentent :

- 1) c(p+1)=p : on a ADD = AD
- 2) c(p+1)=c(p) : ADD est le contenu du quatrième mot de l'enregistrement de t^p, c'est à dire : ADD = LIS[AD+3]
- 3) c(p+1)=c(c(p)) : par définition ADD est égal au contenu du quatrième mot de l'enregistrement attaché à $t^{c(p)}$, soit LIS[LR+3] si LR est l'implantation de $t^{c(p)}$; mais LR est aussi le contenu du quatrième mot de l'enregistrement de t^p, soit LR = LIS[AD+3]. On a en définitive : ADD = LIS[LIS[AD+3]+3].

4.4.2 - Passage d'une étape de calcul à une autre étape ; changement de phase de calcul :

Nous désignerons par LIS (ADR, k, ℓ) avec k un entier inférieur ou égal à ℓ toute liste définie par des enregistrements dont le nombre de mots est ℓ, et pour laquelle ADR est l'adresse du premier enregistrement ; k représente le rang du mot qui est utilisé comme pointeur de liaison des différents enregistrements de la liste.

Au cours d'une phase de calcul, l'automate doit ranger les états analytiques, t = (x, u, β), tels que : $h\,0\,g^{*}\,0\,t(x) = \alpha_{q_i}$ (α_q est la lettre du mot α que l'automate doit lire au cours de cette phase). Les calculs créés dans cette phase, et possédant comme état final l'un de ces états, sont interrompus. C'est pourquoi l'automate doit définir une sous-liste de LIS pour stocker les derniers états de ces calculs : en fait ces états appartiennent à un ensemble $F''(t^p)$ dont les éléments sont déjà ranger dans LIS ; il suffit donc de créer une liste, LIS[ADFEU, 2, 2] dont les enregistrements précisent ceux des premiers triplets, t'', de chacun des ensembles $F''(t)$ qui ont été définis au cours de cette phase de calcul.

Le problème est le même pour chacun des ensembles $F'(t^p)$ obtenus au cours d'une même étape de calcul. A chaque étape on crée une liste, LIS(ADNFE, 2, 2), dont les enregistrements nous donnent ceux de LIS, attachés aux premiers triplets, t', de chacun des ensembles $F'(t^p)$, définis au cours de celle-ci.

Définition de LIS(ADNFE, 2, 2) (respectivement de LIS(ADFEU, 2, 2))

- Le premier mot contient l'adresse d'implantation dans LIS d'un triplet t' (respectivement t'')
- Le deuxième mot est le pointeur reliant les enregistrements de la liste On lui affecte zéro pour chaque dernier enregistrement.

Plaçons-nous à la fin d'une étape de calcul :

Les pseudo-calculs de cette étape ont été rangés dans LIS, et les deux listes LIS (ADNFE, 2, 2) et LIS (ADFEU, 2, 2) ont été complétées. Pour définir l'étape suivante il suffit d'engendrer le train de calculs admettant comme liste associée celle formée pour les sous-listes de LIS dont les débuts de liste sont donnés par l'une des deux listes LIS (ADNFE, 2, 2) et LIS (ADFEU, 2, 2).

Un certain nombre de cas peuvent se présenter :

1) la liste LIS (ADNFE, 2, 2) est vide :

a - si LIS (ADFEU, 2, 2) est vide, l'automate s'arrête puisqu'il ne peut plus engendrer de nouveaux calculs,

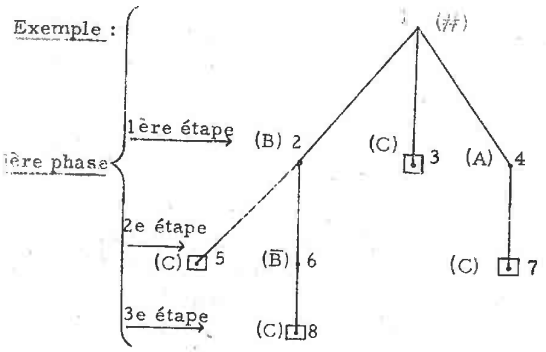
b - sinon une nouvelle phase de calcul débute : l'automate détermine le train de calculs associé à la liste, LIS (ADFEU, 2, 2).

2) LIS (ADNFE, 2, 2) n'est pas la liste vide : la phase de calcul en cours n'est pas terminée. L'automate débute l'étape de calcul associée à LIS (ADNFE, 2, 2).

. Le passage d'une étape à une autre nécessite, en réalité, la définition de deux nouvelles listes du même type que les précédentes : LIS (AADNFE, 2, 2) et LIS (AADFEU, 2, 2). A la fin d'une phase de calcul la seconde est identifiée à LIS (ADFEU, 2, 2) ; l'adresse, ADFEU, étant alors initialisée à zéro pour la phase suivante.

A la fin d'une étape la première est identifiée à LIS (ADNFE, 2, 2) ; l'adresse, ADNFE, étant initialisée à zéro pour l'étape suivante.

Exemple :



- 1, 2, 3, 4, 5, 6, 7, 8 sont les codifications des points du graphe $g\ell_1$.

- $h\,0\,g^*\,0\,t(5) = h\,0\,g^*\,0\,t(8) = h\,0\,g^*\,0\,t(3) = h\,0\,g^*\,0\,t(7) = C \in T$.

Etat de LIS avant la première "étape de calcul" :

	1	2	3	4	5	6	7	8	9	.....
LIST	1	0	0	0	1	0		5		.....

ADNFE=5 ; FREE=NIL=9

Etat de LIS à la fin de la première "étape de calcul" :

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

NIL=33 ; FREE=5 ; AADNFE=29 ; ADFEU=21 ; ADNFE=0

Etat de LIS à la fin de la deuxième "étape de calcul" :

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	...
LIST	1	0	0	0	5	45	9	9	2	25	1	1	9	0		13	3	-9	1	1	17	0		21	4	17	-1	1	25	13		13		...

libérée libérée

	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	...
	5	21		21	6	5	-9	1	37	0		41	7	-37	25	25	45	33		21		...

NIL=53 ; FREE=29 ; AADNFE=41 ; ADFEU=49 ; ADNFE=0

Etat de LIS à la fin de la première "phase de calcul" :

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
LIST	1	0	0	0	5	45	9	9	2	25	1	1	29	49		21	3	-9	1	1	17	0		21	4	17	-1	1	8	0	37	1

	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53
	5	21		21	6	5	-9	1	37	0		41	7	-37	25	25	45	33		21	

libérée

NIL=53 ; FREE=41 ; AADFEU=13 ; ADFEU=0 ; AADNFE=0 ;

Afin de simplifier la gestion des enregistrements disponibles et du même coup la programmation, les enregistrements des sous-listes LIS(ADNFE, 2, 2), LIS(ADFEU, 2, 2), ... ont été rangés dans des enregistrements de quatre mots. Le quatrième mot est utilisé pour transmettre l'adresse dans LIS du dernier enregistrement de ces sous-listes.
Ce procédé occupe plus de place en mémoire ; cependant cela n'est pas gênant puisque ces sous-listes sont temporaires.

CHAPITRE IV

Etude des tests

§.1 - INTRODUCTION

On se propose dans ce chapitre d'élaborer des tests permettant d'orienter la recherche quelque peu aveugle des programmes d'analyse qui se contentent d'explorer systématiquement un graphe en ne validant leurs choix successifs que par le dernier élément de la pile de reconnaissance d'un langage de parenthèses et quelque fois par la prochaine lettre du mot des feuilles, α, que l'automate doit lire.

On peut améliorer l'analyse en effectuant des tests de contrôle qui porteraient sur un certain nombre de dernières lettres de la pile de reconnaissance ainsi que sur certaines des premières lettres du facteur droit de α, non encore lu. Les programmes d'analyse pourraient ainsi s'interroger à chaque étape de calcul, d'état analytique (x, u, β), par l'étude des facteurs gauches de β et des facteurs droits de u, sur l'opportunité de la poursuite de l'analyse en cours à partir du point x.

§.2 - DEFINITION DES ≪TRIPLETS ACCEPTABLES≫

. Soit (V_1, g, K_1, h) le quadruplet associé à la structure compatible S, que nous avons choisie pour réaliser des analyses d'une grammaire G, d'alphabet $\mathcal{V}=T\cup N$. Le langage régulier K_1 est défini par la transcription t et le langage local $L_1=(\mathcal{L}_1, \{\mathcal{E}\}, \{\mathcal{E}'\}, g\mathcal{L}_1)$ tels que : $K_1=t(L_1)$ et $g\mathcal{L}_1=(\mathcal{L}_1, \Gamma_1)$.

On rappelle que : $V_1=V\cup\{\#\}$, puisque nous réalisons des analyses avec adjonction de marqueurs et que g^* est la transcription de V_1^* dans $\mathcal{V}^*$ définie à partir de l'application g de V_1 dans $\mathcal{V}$.

Quant à h, c'est un homomorphisme alphabétique de $\mathcal{V}^*$ dans T^*.

Définition 1 : Un triplet (x, u, γ) de $\mathcal{L}_1 \times \{\# V^*\} \times T^*$ est ≪acceptable≫ si et seulement si :

$$(\exists \lambda \in C(x, \{\mathcal{E}'\}))(\exists \gamma' \in T^*)(\mathcal{L}[ut(\lambda)]=\wedge, \text{ et } h\circ g^*\circ t(\lambda)=\gamma\gamma')$$

. L'ensemble $C(x, \{\mathcal{E}'\})$ désigne les chemins du graphe $g\mathcal{L}_1$, d'origine x et d'extrémité $\mathcal{E}'$.

Comment pouvons nous interpréter la notion de triplet acceptable, dans le contexte d'une analyse ? :

Soit α un mot de T^* ; nous avons vu au chapitre 4 que la recherche des analyses de α nous conduisait à celle des calculs de l'automate qui générait l'ensemble $\mathcal{A}(\alpha) = h^{-1}(\alpha) \cap g^*(K_1 \cap P(V_1))$.

Soit (x, u, β) le dernier état analytique d'un calcul et supposons que nous connaissions tous les triplets acceptables (x, u, γ) avec $|\gamma| = k$. Si β' est le facteur droit de β ayant pour longueur k et que (x, u, β') ne soit pas acceptable, nous ne pouvons pas espérer engendrer un élément de $\mathcal{A}(\alpha)$ à partir de l'état (x, u, β). Tout calcul d'état final (x, u, β) doit être abandonné. La connaissance des triplets acceptables nous permet d'éviter les fausses pistes.

Lemme : Soit $u \in V^*$ et $\lambda \in (V \cup \bar{V})^*$ tels que $\varepsilon[u\lambda] = \wedge$.
Alors $\lambda = \lambda_1 \bar{a}_1 \lambda_2 \bar{a}_2 \cdots \lambda_p \bar{a}_p \lambda_{p+1}$ avec $\lambda_i \in P(V)$, pour $i=1, 2, \ldots, p$, si $u = a_p a_{p-1} \cdots a_2 a_1$.

Démonstration :

1) Soit $u = \wedge$; le lemme est vérifié : $p = 0$ et $\lambda = \lambda_1$

2) Supposons le lemme démontré pour des mots de V^* de longueur strictement inférieure à p.

Soient $u = a_p a_{p-1} \cdots a_2 a_1$ un mot de V^* et λ un mot de $(V \cup \bar{V})^*$ qui vérifient : $\varepsilon[u\lambda] = \wedge$.

Si on pose $u = u' a_1$ et $\lambda' = a_1 \lambda$, on a : $\varepsilon[u'\lambda'] = \wedge$.

Mais u' est de longueur p-1 et vérifie l'hypothèse de récurrence ; par conséquent il existe λ'_i appartenant à $P(V)$, pour $i=1, 2, \ldots, p$, pour lesquels : $\lambda' = \lambda'_1 \bar{a}_2 \lambda'_2 \cdots \lambda'_{p-1} \bar{a}_p \lambda'_p$. De plus λ' commençant par a_1, il en est de même pour λ'_1 ; et comme ce dernier appartient à $P(V)$, il existe λ_1 et λ_2 de $P(V)$ tels que : $\lambda'_1 = a_1 \lambda_1 \bar{a}_1 \lambda_2$. Le mot λ se décompose alors comme suit :

$\lambda = \lambda_1 \bar{a}_1 \lambda_2 \bar{a}_2 \lambda'_2 \cdots \lambda'_{p-1} \bar{a}_p \lambda'_p$, les λ_i, pour $i=1, 2$ et les λ'_i, pour $i=1, 2, \ldots, p$ appartenant à $P(V)$.

Remarque : Désormais les notations seront celles introduites lors de la définition 1.

Proposition 1: Soit $u = a_p a_{p-1} \cdots a_1$ un mot de $\# V^*$ et γ un mot de T^* de longueur égale à k. Le triplet (x, u, γ) de $\mathcal{L}_1 \times \{\# V^*\} \times T^*$ est acceptable si et seulement si :
$(\exists \gamma_1, \gamma_2, \ldots, \gamma_p \in T^*)\ (\exists \lambda_1, \lambda_2, \ldots, \lambda_p \in P(V))\ (\gamma = g_k(\gamma_1 \cdots \gamma_p), \ldots$
$\ldots \lambda_1 \bar{a}_1 \lambda_2 \cdots \lambda_p \bar{a}_p \in t[C(x, \{\mathcal{C}'\})]$ et $\gamma_i = g_k 0 h 0 g^*(\lambda_i \bar{a}_i)$, pour $i=1, \ldots, p)$

Démonstration :

1) La condition suffisante est évidente si l'on prend, pour le chemin λ de la définition 1, un chemin m de $C[x, \{\mathcal{C}'\}]$ de la proposition 1, vérifiant : $t(m) = \lambda_1 \bar{a}_1 \lambda_2 \cdots \lambda_p \bar{a}_p$.

En effet, $\varepsilon[u\, t(m)] = \wedge$, et $g_k 0 h 0 g^* 0 t(m) = g_k \left[\underset{i=1}{\overset{p}{\pi}} \gamma_i \right]$, ce qui équivaut à l'existence d'un mot γ' de T^* tel que $h 0 g^* 0 t(m) = \gamma\gamma'$.

2) Démontrons la condition nécessaire : soit (x, u, γ) un triplet acceptable. Il existe un chemin λ d'origine x et d'extrémité $\mathcal{C}'$, pour lequel : $\varepsilon[u\, t(\lambda)] = \wedge$, et il existe un mot γ' de T^* tel que $h 0 g^* 0 t(\lambda) = \gamma\gamma'$.

Le mot $t(\lambda)$ de $(V_1 \cup \bar{V}_1)^*$ vérifie les hypothèses du lemme en substituant l'ensemble V_1 à l'ensemble V dans son énoncé. Il existe des mots λ_i de $P(V_1)$, pour $i=1, 2, \ldots, p+1$, vérifiant : $t(\lambda) = \lambda_1 \bar{a}_1 \lambda_2 \bar{a}_2 \cdots \lambda_p \bar{a}_p \lambda_{p+1}$. Le point $\mathcal{C}'$ de $g\mathcal{L}_1$ est le seul point tel que $t(\mathcal{C}') = \bar{\#} (= \bar{a}_p)$ comme $\mathcal{C}'$ n'a pas de successeur dans le graphe $g\mathcal{L}_1$, cela implique que λ_{p+1} est le mot vide. Si nous posons $\gamma_i = g_k 0 h 0 g^*(\lambda_i \bar{a}_i)$, pour $i=1, \ldots, p$, on en déduit :

$$h 0 g^* 0 t(\lambda) = \underset{i=1}{\overset{p}{\pi}} h 0 g^*(\lambda_i \bar{a}_i) \quad \text{et} \quad g_k 0 h 0 g^* 0 t(\lambda) = g_k \left[\underset{i=1}{\overset{p}{\pi}} h 0 g^*(\lambda_i \bar{a}_i) \right]$$

$$= g_k \left[\underset{i=1}{\overset{p}{\pi}} g_k 0 h 0 g^*(\lambda_i \bar{a}_i) \right] = g_k \left[\underset{i=1}{\overset{p}{\pi}} \gamma_i \right] = \gamma$$

puisque γ est facteur gauche de $h 0 g^* 0 t(\lambda)$ d'après la définition 1 et a pour longueur k.

3 - ETUDE DES «TRIPLETS INTERMEDIAIRES»

La proposition 1 nous a conduit à décomposer les mots de $(V \cup \bar{V})^* \bar{\#}$, qui sont les transcrits par t de chemins du graphe $g\mathcal{L}_1$ en des mots de la forme $\lambda \bar{a}$ où le mot λ appartient à $P(V)$ et a à $\bar{V}_1$. Cette décomposition justifie l'introduction des triplets intermédiaires.

3.3.1 - Définition des triplets intermédiaires d'ordre k :

Soient $\mathcal{L}_1'$ l'ensemble des éléments de $\mathcal{L}_1$ dont les transcrits par t appartiennent à $\bar{V}_1$ et T_k l'ensemble des mots de T^* de longueur inférieure ou égale à k.

L'ensemble des triplets intermédiaires d'ordre k (k étant un entier positif ou nul), $\mathcal{E}_k$ est l'ensemble des triplets appartenant à $\mathcal{L}_1 \times \mathcal{L}_1' \times T_k$ qui vérifient l'une des deux conditions suivantes :

- un triplet (x, y, δ) de $(\mathcal{L}_1 - \mathcal{L}_1') \times \mathcal{L}_1' \times T_k$ est un triplet intermédiaire d'ordre k si et seulement si : $(\exists \lambda \in C(x, \Gamma_1^{-1}(y)))\ (t(\lambda) \in P(V)$ et $g_k 0 h 0 g^* 0 t(\lambda y) = \delta)$
- un triplet (y', y, δ) de $\mathcal{L}_1' \times \mathcal{L}_1' \times T_1$ est un triplet intermédiaire d'ordre k, si et seulement si : $(y = y')\ (g_k 0 h 0 g^* 0 t(y) = \delta)$.

Cette définition implique, en particulier, que les mots δ soient de longueur inférieure ou égale à k.

Nous pouvons définir sur $\mathcal{E}_k$ le langage local $\mathcal{L}_k = (\mathcal{E}_k, \mathcal{E}_k, \mathcal{E}'_k, \mathcal{G})$ avec :

- l'ensemble initial est l'ensemble de définition $\mathcal{E}_k$.
- l'ensemble final est l'ensemble $\mathcal{E}'_k$, sous-ensemble de $\mathcal{E}_k$, dont les triplets (x, y, δ) possèdent une deuxième composante y égale à $\mathcal{E}'$.
- le graphe des transitions $\mathcal{G} = \{\mathcal{E}_k, g^k\}$ est défini par :

$(\forall (x, y, \delta), (x', y', \delta') \in \mathcal{E}_k)\ ((x, y, \delta)\, g^k (x', y', \delta') \iff y \Gamma_1 x')$.

. La proposition 2 qui suit va nous donner le moyen de déterminer pratiquement «les triplets acceptables» par l'intermédiaire du langage local $\mathcal{L}_k$. Avant de l'énoncer, introduisons un certain nombre de transcriptions et d'homomorphismes alphabétiques que nous allons utiliser . Soit p_1, l'application de $\mathcal{E}_k$ dans $\mathcal{L}_1$, définie par :

$\forall (a, b, c) \in \mathcal{E}_k \qquad p_1[(a, b, c)] = a$, ce que nous pouvons appeler la première projection, de la même façon soit p_3, la troisième projection.

Enfin, prenons pour p_2 l'application de $\mathcal{E}_k$ dans V_1 définie par :

$\forall (a, b, c) \in \mathcal{E}_k \qquad p_2[(a, b, c)] = f 0 t(b)$, avec pour f la bijection de $\bar{V}_1$ dans V_1 qui à tout élément $\bar{a}$ de $\bar{V}_1$ associe l'élément correspondant a de V_1.

A partir de ces trois applications nous pouvons définir les transcriptions entre monoïdes libres p_1^*, p_2^*, p_3^* comme nous l'avons convenu au chapitre 1 et qui sont des homomorphismes alphabétiques de $\mathcal{E}_k^*$ dans $\mathcal{L}_1^*$ pour p_1^*, de $\mathcal{E}_k^*$ dans V_1^* pour p_2^*, et de $\mathcal{E}_k^*$ dans T^* pour p_3^*.

Proposition 2 : Soit γ un mot de T^* de longueur égale à k. Un triplet (x, u, γ) de $\mathcal{L}_1 \times \{\# V^*\} \times T_k$ est acceptable, si et seulement si, il existe un mot m de $\mathcal{L}_k$ tel que :

$$p_1 0 g_1(m) = x \ , \quad p_2^*(m) = \tilde{u} \ , \quad g_k 0 p_3^*(m) = \gamma$$

Démonstration : Soit (x, u, γ) un triplet acceptable ; d'après la proposition 1, cela équivaut à : $u = a_p a_{p-1} \cdots a_1$,

$$(\exists \gamma_i \in V^*, i = 1, 2, \ldots, p)(\exists \lambda_i \in P(V), i = 1, 2, \ldots p)\ \Big(\prod_{i=1}^{p} \lambda_i \bar{a}_i \in t[C(x, \{\mathcal{E}'\})],\ \gamma = g_k\Big(\prod_{i=1}^{p} \gamma_i\Big),$$

$$\gamma_i = g_k 0 h 0 g^*(\lambda_i \bar{a}_i))$$

. Il est équivalent d'écrire qu'il existe un chemin λ de $C(x, \{\mathcal{E}'\})$ tel que : $t(\lambda) = \prod_{i=1}^{p} \lambda_i \bar{a}_i$ et de telle façon que, pour $i = 1, 2, \ldots p$, l'on puisse trouver deux éléments x_i et y_i de $\mathcal{L}_1$, vérifiant : $\lambda_i \bar{a}_i \in t[C(x_i, y_i)]$, $t(y_i) = \bar{a}_i$, $t(x_i) = g_1(\lambda_i)$. De plus, x_{i+1} appartient à $\Gamma_1(y_i)$, pour $i = 1, 2, \ldots, p-1$ (en particulier $x_1 = x$ et $y_p = \mathcal{E}'$).

. Si nous posons $m_i = (x_i, y_i, \gamma_i)$, par définition m_i appartient à $\mathcal{E}_k$. Il est ainsi équivalent d'écrire que le mot $m = \prod_{i=1}^{p} m_i$ de $\mathcal{E}_k^*$ appartient au langage local $\mathcal{L}_k$ et qu'il vérifie :

$$- \ g_k 0 p_3^*(m) = g_k\Big[\prod_{i=1}^{p} p_3(m_i)\Big] = g_k\Big[\prod_{i=1}^{p} \gamma_i\Big] = \gamma$$

$$- \ p_2^*(m) = \prod_{i=1}^{p} f 0 t(y_i) = \prod_{i=1}^{p} a_i = \tilde{u}.$$

Corollaire : Soit x de $\mathcal{L}_1$ et γ un mot de T^* de longueur égale à k ; alors l'ensemble $L(x, \gamma)$ des mots u de $\# V^*$ tels que les triplets (x, u, γ) soient acceptables est un langage régulier.

Démonstration : En effet, d'après la proposition 2, on a $L(x, \gamma) = \Sigma'$, avec pour L' le langage :

$$p_2^*\{\mathcal{E}_k \cap D(x)\, \mathcal{E}_k'^* \cap p_3^{-1*}(\gamma T^*)\}\ ,$$

D(x) étant l'ensemble des triplets de $\mathcal{E}_k$, admettant x pour première composante :

$$D(x) = \{t \in \mathcal{E}_k \mid p_1(t) = x\}.$$

L' est un langage régulier puisqu'il est formé d'intersections et d'image réciproque ou de transformés par homomorphismes entre monoïdes libres, de langages réguliers.

La recherche des triplets acceptables dépend ainsi de la génération des mots de $\mathscr{C}_{k'}$ c'est pourquoi nous allons étudier au paragraphe suivant la construction de l'ensemble $\mathscr{C}_k$ des triplets intermédiaires.

5.3.2 - Recherche de l'ensemble $\mathscr{C}_k$

. Si nous notons, pout tout a de V_1, $P_{\bar{a}}$ le langage $P(V_1)\,\bar{a}$, la détermination de l'ensemble $\mathscr{C}_k$ se ramène à la recherche de l'un des deux types de langages suivants :

- pour tout couple (x, y) de $(\ell_1 - \ell_1')\times\ell_1'$, le langage
  $\delta_k(x, y) = g_k \circ h \circ g^* \{P_{t(y)} \cap t[C(x, y)]\}$.
- pour tout couple (y, y) de $\ell_1'\times\ell_1'$, le langage
  $\delta_k(y, y) = \{g_k \circ h \circ g^* \circ t(y)\}$.

5.3.2.1 - Génération des langages $P_{\bar{a}}$ $(a \in V_1)$

Elle est intuitive compte-tenu de la forme de tout élément d'un langage de parenthèses.

Proposition 3 : Pour tout a de V_1, le langage $P_{\bar{a}}$ obéit au système (1) :

$$P_{\bar{a}} = \bar{a} \cup \Big(\bigcup_{b \in V_1} b\, P_{\bar{b}}\, P_{\bar{a}} \Big)$$

Démonstration : L'inclusion directe est évidente. Soit x appartenant à $P_{\bar{a}}$:

1) Si x est de longueur égale à un, on a : $x = \bar{a}$.

2) Le mot x est de longueur différente de 1 :

Alors $x = x'\bar{a}$ avec x' appartenant à $P(V_1)$.

Pour tout élément x' de $P(V_1)$, il existe b appartenant à V_1 et des mots x_0 et x_1 de $P(V_1)$ tels que : $x' = b\,x_0\,\bar{b}\,x_1$.

Par conséquent, $x = b(x_0\bar{b})(x_1\bar{a}) = b\,y\,.\,y'$ avec $y = x_0\bar{b}$ appartenant à $P_{\bar{b}}$ et $y' = x_1\bar{a}$ à $P_{\bar{a}}$; c'est à dire x appartient à $b\,P_{\bar{b}}\,P_{\bar{a}}$.

5.3.2.2 - Etude de l'algorithme de génération des langages $\delta_k(x, y)$

Nous noterons :

- $\forall\,(x, y) \in (\ell_1 - \ell_1')\times\ell_1'$: $L(x, y) = P_{t(y)} \cap t[C(x, y)]$
- $\forall\,(y, y) \in \ell_1'\times\ell_1'$: $L(y, y) = \{t(y)\}$

On déduit aisément du système (1) de la proposition 3, la proposition suivante.

Proposition 4 : Pour tout couple (x, y) de $(\ell_1 - \ell_1')\times\ell_1'$, on a :

(2) $$L(x, y) = \bigcup_{(z, x', x'') \in V_0(x, y)} t(x)\; L(z, x')\, L(x'', y), \text{ avec}$$

$$V_0(x, y) = \{(x_1, x_2, x_3) \in \ell_1\times\ell_1'\times\ell_1 \mid x_1 \in \Gamma_1(x),\ x_3 \in \Gamma_1(x_2),\ t(x_2) = \overline{t(x)},\ L(x_1, x_2) \neq \emptyset \text{ et } L(x_3, y) \neq$$

Démonstration : Soit δ appartenant à $L(x, y)$. D'après le système (1), il est équivalent de dire qu'il existe b appartenant à V_1 tel que δ appartienne à $b\,P_{\bar{b}}\,P_{t(y)}$. En effet, t étant une transcription et $t(x)$ appartenant à V_1, δ doit être de longueur supérieure à 1 pour qu'il puisse appartenir à $t[C(x, y)]$. On peut alors trouver deux chemins c_1 et c_2 de $g\ell_1$, $(t(c_1) \in P_{\bar{b}}$ et $t(c_2) \in P_{t(y)})$, tels que : $\delta = t(x' c_1 c_2)$ ou ce qui est équivalent des points z, x' et x'' de ℓ_1 vérifiant :

$$c_1 \in C(z, x'),\ c_2 \in C(x'', y),\ z \in \Gamma_1(x),\ x'' \in \Gamma_1(x') \text{ et } t(x') = \bar{b} = \overline{t(x)}.$$

Il est à remarquer que la détermination des langages $L(x, y)$ par l'algorithme de la proposition 4, ne se prête pas facilement au calcul sur ordinateur. C'est pourquoi nous le modifierons pour permettre une programmation plus aisée et plus efficace (en évitant en particulier les répétitions).

La forme du système (1) nous invite à construire $L(x, y)$, par «approximations successives», à partir de $L(y, y) = t(y)$, (pour y appartenant à ℓ_1'), qui appartient à $\bar{V}_1$. Nous engendrerons ainsi de proche en proche les mots de $L(x, y)$, de longueur croissante, en emboîtant ceux de longueur inférieure déjà obtenus.

Les éléments de $L(x, y)$ étant de longueur impaire, nous appellerons $L(x, y, i)$ l'ensemble de ses éléments de longueur $2i+1$. Nous obtenons ainsi l'algorithme (2 bis) :

- Pour tout y de ℓ_1' : $L(y, y, 0) = \{t(y)\}$
- Pour tout x de $\ell_1 - \ell_1'$ et tout y de ℓ_1' :

$$L(x, y, i) = \bigcup_{(z, x', x'') \in V_0(x, y)} \Big(\bigcup_{\substack{0 \le m \le i-1 \\ m+1 = i-1}} t(x)\; L(z, x', m)\, L(x'', y, n) \Big)$$

Mais les langages qui nous intéressent sont les transformés des langages $L(x, y)$ par $g_k 0 h 0 g^*$: $\delta_k(x, y)$. Leur algorithme de génération est obtenu en transformant (2 bis) par l'application $g_k 0 h 0 g^*$.

Soient $\Delta_k(x, y, i) = \{g_k 0 h 0 g^*(\delta) \mid \delta \in L(x, y, i)\}$ et

$V_0(x, y, i) = \{t \in V_0(x, y) \mid$ si $t = (x_1, x_2, x_3)$ il existe des entiers ℓ et p, ... tels que : $\Delta_k(x_1, x_2, \ell) \neq \emptyset$, $\Delta_k(x_3, y, p) \neq \emptyset$, $\ell + p = i-1$, $0 \leq \ell$, $p \leq i-1\}$

Pour tout couple (x, y) de $(\ell_1 - \ell_1') \times \ell_1'$ et tout entier i supérieur à 0 :

$$(3)\quad \Delta_k(x, y, i) = \bigcup_{(z, x', x'') \in V_0(x, y, i)} \left(\bigcup_{\substack{0 \leq m,\ n \leq i-1 \\ m+n=i-1}} g_k[h 0 g^* 0 t(x) \Delta_k(z, x', m) \Delta_k(x'', y, n)] \right)$$

Pour tout y de ℓ_1' :

$$\Delta_k(y, y, 0) = \{g_k 0 h 0 g^* 0 t(y)\}^*$$

On a : $\delta_k(x, y) = \bigcup_{i \geq 1} \Delta_k(x, y, i)$, pour tout couple (x, y) de $(\ell_1 - \ell_1') \times \ell_1'$.

Il nous faut cependant démontrer que l'algorithme s'arrêtera après un nombre fini d'itérations, c'est à dire qu'il existe un entier i_0 pour lequel :

$$\delta_k(x, y) = \bigcup_{i=1}^{i=i_0} \Delta_k(x, y, i).$$

En outre, pour programmer l'algorithme (3), il sera nécessaire de pouvoir déterminer pratiquement l'entier i_0 .

Etude d'un test d'arrêt de l'algorithme (3) :

Pour tout entier i supérieur ou égal à 1 , nous noterons :

$$\delta_k(x, y, i) = \bigcup_{j=1}^{j=i} \Delta_k(x, y, j).$$

On a évidemment l'inclusion : (4)

$$\forall\, i \geq 1 \qquad \delta_k(x, y, i) \subset \delta_k(x, y, i+1).$$

On pourrait penser se contenter de l'inclusion :

$\Delta_k(x, y, i+1) \subset \delta_k(x, y, i)$ comme test d'arrêt, mais on ne peut pas démontrer que ce test est suffisant pour arrêter les itérations, car on ne réalise jamais deux fois le même emboîtement d'une étape i à une autre. Le test précédent est seulement une condition nécessaire pour que l'étape i soit la dernière itération.

Propriété 1 : <<existence d'un nombre fini d'itérations>>
Soit (x, y) appartenant à $(\ell_1 - \ell_1') \times \ell_1'$, alors :
il existe un entier i_0 , tel que :
$(\forall\, j \geq i_0) \quad \delta_k(x, y, j) = \delta_k(x, y, i_0)$

Démonstration : Si la proposition était fausse, on aurait :
$(\forall\, i \geq 1)\ (\exists\, j \geq i)\ \ (\delta_k(x, y, j) \neq \delta_k(x, y, j+1))$,
l'inclusion (4) impliquerait que l'ensemble $\delta_k(x, y)$ possède une infinité d'éléments ; ce qui est faux, puisque si n , est le nombre de lettres de l'alphabet terminal T de la grammaire G , $\delta_k(x, y)$ possède un nombre de mots majoré par $n^k + 1$, c'est à dire le nombre de mots de k lettres prises parmi n lettres plus un (pour le mot vide).

Corollaire : Il existe un entier j_0 tel que :
$(\forall\, (x, y) \in (\ell_1 - \ell_1') \times \ell_1')\ (\forall\, i \geq j_0)\ \ (\delta_k(x, y, j_0) = \delta_k(x, y, i))$

Il suffit de prendre le plus grand des entiers i_0, de la propriété 1, associés à chacun des couples de $(\ell_1 - \ell_1') \times \ell_1'$.

Propriété 2 : Soit $E(k)$ l'ensemble des entiers pairs, $2i$, pour lesquels, on a :
$(\forall\, j)\ (i \leq j \leq 2i-1)(\forall\, (x, y) \in (\ell_1 - \ell_1') \times \ell_1')(\Delta_k(x, y, j) \subset \delta_k(x, y, j-1))$
Ce qui est équivalent à :
$(\forall\, j)\ (i \leq j \leq 2i-1)(\forall\, (x, y) \in (\ell_1 - \ell_1') \times \ell_1')\ (\delta_k(x, y, j) = \delta_k(x, y, i-1))$
i) L'ensemble $E(k)$ n'est pas l'ensemble vide
ii) Soit $2i$ un élément de $E(k)$, alors :
$(\forall\, \ell \geq 2i)\ (\forall\, (x, y) \in (\ell_1 - \ell_1') \times \ell_1')\ (\Delta_k(x, y, \ell) \subset \delta_k(x, y, i-1))$.

Démonstration :
i) D'après le corollaire précédent il est trivial que l'ensemble $E(k)$ n'est pas l'ensemble vide.
ii) Soit ℓ un élément de $E(k)$ et M un élément de $\Delta_k(x, y, \ell)$ avec (x, y) appartenant à $(\ell_1 - \ell_1') \times \ell_1'$. L'algorithme (3) nous conduit à décomposer M de la façon suivante : $M = h 0 g^* 0 t(x) M' M''$, avec M' (respectivement M'') appartenant à $\Delta_k(z', x', m)$ (respectivement $\Delta_k(x'', y, n)$) et
$m + n = \ell - 1$, $0 \leq m \leq \ell - 1$, $0 \leq n \leq \ell - 1$.

L'entier $(\ell-1)$ étant égal à $2i-1$, l'un des deux entiers m et n est supérieur ou égal à i, pendant que l'autre est inférieur ou égal à i-1.

Soit par exemple : $n \geq i$ et $m \leq i-1$.

D'après la définition de l'ensemble E(k), on a : $\delta_k(x'', y, n) = \delta_k(x'', y, i-1)$, il existe donc un entier n', $n' \leq i-1$ tel que M'' appartienne à $\Delta_k(x'', y, n')$. Mais $m+n'=p$, avec $p \leq 2i-2$ et $p+1 \leq 2i-1$, par conséquent M appartient à $\delta_k(x, y, i-1)$.

On obtient ainsi : $\delta_k(x, y, 2i) = \delta_k(x, y, i-1)$, pour tout couple (x, y) de $(\ell_1-\ell_1')\times\ell_1'$. Compte-tenu de ce résultat, et par une démonstration similaire à la précédente, on en déduit : (I) $\delta_k(x, y, 2i+1) = \delta_k(x, y, i-1)$; et ainsi l'entier $2(i+1)$ appartient à E(k).

- On a démontré l'implication : $2i \in E(k) \Longrightarrow 2(i+1) \in E(k)$.

On a immédiatement : $\forall\, p \geq i$, $2p \in E(k)$ (II). Nous pouvons alors démontrer, par récurrence sur la différence $(p-i)$, p étant un entier strictement supérieur à i, les égalités suivantes :

$$(\forall\, p > i)(\forall\, \ell)\,(2i-1 < \ell \leq 2p-1)\,(\forall (x, y) \in (\ell_1-\ell_1')\times\ell_1')\,(\delta_k(x, y, \ell) = \delta_k(x, y, i-1)).$$

Si $p = i+1$, le résultat se déduit de l'égalité (I).

Supposons la propriété vraie pour les entiers p tels que $p-i \leq k$ et soit $p = i+k+1$. $(k > 1)$

D'après la relation (II) :

$$(\forall\, \ell)(p \leq \ell \leq 2p-1)(\forall (x, y) \in (\ell_1-\ell_1')\times\ell_1')(\delta_k(x, y, \ell) = \delta_k(x, y, p-1)).$$

Mais, $(p-1)-i = k$ et $(p-1)$ vérifie l'hypothèse de récurrence, d'où : $\delta_k(x, y, p-1) = \delta_k(x, y, i-1)$.

La proposition 2 nous donne le moyen de décider du choix de la dernière itération :

≪Pour un entier k, donné, la dernière itération sera celle correspondant à l'entier $2i(k)-1$, tel que $2i(k)$ soit égal au plus petit élément de E(k) ≫.

$$\delta_k(x, y) = \bigcup_{i=1}^{i=i(k)-1} \Delta_k(x, y, i) = \delta_k(x, y, i(k)-1).$$

Il est à remarquer que les tests :

(III) $\Delta_k(x, y, j) \subset \delta_k(x, y, j-1)$ seront coûteux lors de la programmation de l'algorithme (3), c'est pourquoi nous allons étudier quelques propriétés qui nous permettrons de réduire le nombre de ces tests.

Soit $V_0^i(k)$ l'ensemble défini par :

$$V_0^i(k) = \{(x, y) \in (\ell_1-\ell_1')\times\ell_1' \mid (\exists\, j \leq i)(V_0(x, y, j) \neq \emptyset)\}$$

On a évidemment : $\forall\, i \geq 1$, $V_0^i \subset V_0^{i+1}$ (IV).

Propriété 3 : Il existe un entier j_0 tel que :
$(\forall\, i \geq j_0)\,(V_0^i(k) \subset V_0^{j_0}(k))$

Démonstration : Si la propriété n'était pas vérifiée, on aurait :
$(\forall\, j)\,(\exists\, i \geq j)\,(V_0^i(k) \not\subset V_0^j(k))$, compte-tenu de (IV), cela est équivalent à :

$(\forall\, j)\,(\exists\, i \geq j)\,(V_0^i(k) \neq V_0^j(k))$; ce qui impliquerait que $(\ell_1-\ell_1')\times\ell_1'$ possède une infinité d'éléments.

Propriété 4 : Soit j(k) le plus petit des entiers j_0 vérifiant la propriété 3, alors : $i(k) > j(k)$.

Démonstration : Soit un entier i strictement supérieur à i(k).

- Pour tout couple (x, y) de $V_0^i(k)$, on a : si j est le plus petit des entiers ℓ tels que $\ell \leq i$ et $V_0(x, y, \ell) \neq \emptyset$ deux possibilités se présentent :

i) $j < i(k)$, et ainsi : $(x, y) \in V_0^{i(k)-1}$

ii) $i(k) \leq j \leq i$; d'après la définition de i(k), on en déduit que $\delta_k(x, y, i(k)-1) \neq \emptyset$ et qu'il existe un entier $\ell \leq i(k)-1$ tel que : $V_0(x, y, \ell) \neq \emptyset$, ce qui est équivalent : $(x, y) \in V_0^{i(k)-1}$. Il en résulte : $\forall\, i \geq i(k)-1$ $V_0^i(k) \subset V_0^{i(k)-1}(k)$, et d'après la définition de j(k) cela implique : $i(k)-1 \geq j(k)$.

L'entier j(k) peut être caractérisé par une propriété intuitive :

$$(\forall (x, y) \in (\ell_1-\ell_1')\times\ell_1')\,((x, y) \notin V_0^{j(k)}(k) \Longrightarrow \delta_k(x, y) = \emptyset).$$

Nous pourrons déterminer précisément cet entier dans le paragraphe suivant, lorsque nous introduirons la notion de ≪couple créé≫.

Les tests du type (III) ne seront exécutés que pour les entiers j strictement supérieurs à j(k).

La propriété qui suit nous sera utile lors de l'étude contextuelle.

Propriété 5 : Pour tout entier j tel que $1 \leq j < k$, on a :

i) $\delta_{k-j}(x, y) = g_{k-j}[\delta_k(x, y)]$, pour tout couple (x, y) de $(\ell_1-\ell_1')\times\ell_1'$.

ii) $i(k) \geq i(k-j)$

Démonstration

i) Par définition, $\delta_{k-j}(x, y, i) = g_{k-j}[\Delta_k(x, y, i)]$ et par conséquent :

$$g_{k-j}[\delta_k(x, y)] = g_{k-j}\Big[\bigcup_{i \geq 1} \Delta_k(x, y, i)\Big] = \delta_{k-j}(x, y).$$

ii) $\delta_{k-j}(x, y) = \bigcup_{i=1}^{i=i(k-j)-1} \Delta_{k-j}(x, y, i) = g_{k-j}[\delta_k(x, y)] = \bigcup_{\ell=1}^{\ell=i(k)-1} \Delta_{k-j}(x, y, \ell)$.

. Pour tout entier i strictement supérieur à $i(k)-1$, on en déduit :

$\Delta_{k-j}(x, y, i) \subset \delta_{k-j}(x, y, i(k-j)-1)$, ce qui implique $i > i(k-j)$, et ainsi $i(k) \geq i(k-j)$.

5.4 - PROGRAMMATION DE L'ALGORITHME (3)

Ce paragraphe a pour sujet la définition d'une procédure, TRIPLE(K), possédant un paramètre entier K, qui pour chaque entier K doit déterminer l'ensemble des triplets intermédiaires d'ordre K, soit $\mathcal{C}_K$.

Définition 1 : ≪couples créés≫

On dira qu'un couple (x, y) de $\ell_1 \times \ell'_1$ a été créé à l'étape i, si i est le plus petit entier pour lequel l'ensemble $\Delta_k(x, y, i)$ est différent de l'ensemble vide.

Les couples créés sont rangés dans un tableau unidimensionnel, LIS, qui sera notre support d'information principal et qui sera organisé en structure de liste.
On lui adjoint cependant un tableau d'identificateur, LONG, qui permet de connaître, pour toute information rangée dans LIS, l'étape au cours de laquelle celle-ci a été enregistrée. Ceci est rendu possible par le fait que les informations enregistrées sont crées séquentiellement.
Si une adresse, AD, vérifie l'inéquation : $LONG[p-1] < AD \leq LONG[p]$ cela signifie que l'information a été rangée en AD, à l'étape P.

Définition 2 : ≪couples productifs≫

Un couple créé à l'étape i, (x_1, x_2), de $\ell_1 \times \ell'_1$ est dit productif, s'il existe un point x de $(\ell_1 - \ell'_1)$, qui vérifie :

$$x \Gamma_1 x_1 \quad \text{et} \quad \overline{t(x)} = t(x_2).$$

Ces couples sont intéressants, car c'est parmi eux que nous rechercherons les premières et secondes composantes des triplets appartenant aux ensembles $V_0(x, y, j)$ $(j > i)$, et c'est à partir d'eux seuls que l'on réalisera les emboîtements qui apparaissent dans l'algorithme (3).

Définition 3 : Un élément, x_2, de ℓ'_1 est dit productif s'il existe un couple, (x_1, x_2), productif.

5.4.1 - Organisation du tableau, LIS, en structure de liste

Chaque liste que nous créons est définie par un ensemble d'enregistrements ; un enregistrement est formé par un ensemble de mots contigus. Chaque mot est un élément du tableau, LIS, contenant un entier qui peut représenter un élément de ℓ_1 ou de $\mathcal{U}_k$, un pointeur indiquant l'adresse du premier mot de l'enregistrement suivant celui auquel il appartient ; ou encore un pointeur repérant le premier mot du premier enregistrement d'une sous-liste associée à l'enregistrement auquel il appartient.

- Une liste dont les enregistrements sont de longueur fixe sera définie par un triplet : (AD, N1, N2).
  - AD, est l'adresse dans le tableau, LIS, du premier mot du premier enregistrement
  - N1, est la longueur d'un enregistrement (nombre de mots)
  - N2, est le rang dans un enregistrement du mot contenant le pointeur liant les enregistrements entre eux ; le contenu du mot de rang, N2, sera nul pour le dernier enregistrement de la liste.

Le tableau, LIS, est utilisé comme mémoire de travail à partir d'une adresse, évidemment variable, et dont l'identificateur est LIBRE.
L'identificateur, FREE, indique l'adresse à partir de laquelle on pourra, ultérieurement, ranger des informations concernant les couples créés. On doit nécessairement avoir l'inégalité, FREE < LIBRE ; cependant on ne teste jamais celle-ci pour ne pas alourdir la programmation.

- Le tableau, LIS, comporte deux listes principales, (I) et (II), permettant la recherche des couples créés, ainsi que toutes les informations les concernant ; soit par leurs premières composantes, pour la liste (II), soit par leurs secondes composantes, pour la liste (I).

Liste (I) : Elle est définie par le triplet (PI, 3, 2).
- Le premier mot d'un enregistrement contient la seconde composante, y, d'un couple créé.
- Le troisième mot contient un pointeur repérant une sous-liste de type (III), associée à y.

Lorsque $x \Gamma_1 y$ est productif, le contenu du premier mot est affecté d'un signe moins.

Liste (II) : Elle est définie par le triplet (PI, 3, 2).
- Le premier mot d'un enregistrement contient la première composante, x, d'un couple créé.
- Le troisième mot contient un pointeur repérant une sous-liste de type

Listes de type (III) : A chaque enregistrement de la liste (I), dont l'adresse du premier mot est ADR, est associée une sous-liste définie par : (LIS[ADR+2], 3, 3), qui représente l'ensemble des points x de ℓ_1, tels que les couples (x, y), y étant le contenu du mot d'adresse ADR, soient des couples créés.

- Le premier mot d'un enregistrement contient une des premières composantes x ; lorsque x appartient à un couple productif, le contenu du premier mot est affecté d'un signe moins, en outre contigument à l'enregistrement on place les points z, de $\ell_1 - \ell_1'$, qui vérifient :

$$\overline{t(z)} = t(y) \qquad \text{et} \qquad z \in \Gamma_1^{-1}(x)$$

- Le second mot contient un pointeur P_1, repérant la sous-liste de type (IV), qui définit les ensembles $\delta_k(x, y, i)$.

Listes de type (IV) : A chaque enregistrement de la liste (II), dont l'adresse du premier mot est RDA, est associée une sous-liste définie par :
(LIS[RDA+2], 3, 2) qui représente l'ensemble des points y, de ℓ_1', tels que les couples (x, y), x étant le contenu du mot d'adresse RDA, soient des couples créés.

- Le premier mot d'un enregistrement contient une des secondes composantes y.

- Le troisième mot contient un pointeur, P_2, repérant la sous-liste de type (V), qui définit les ensembles $\delta_k(x, y, i)$.

Pour un même couple, (x, y), les pointeurs P_1 et P_2 sont identiques ; on les identifiera à P(x, y).

Listes de type (V) : A chaque couple créé, (x, y), est associé une liste de type (V), définie par le triplet (P(x, y), 2, 2). Le premier mot de chaque enregistrement contient l'adresse de rangement dans LIS, d'un mot de T^* appartenant à un ensemble $\delta_k(x, y, i)$.

Un mot de T^*, δ, est rangé dans LIS sous la forme d'un enregistrement de longueur variable.

Le premier mot de cet enregistrement contient la longueur du mot δ. Les mots suivants représentent la codification entière des lettres de δ, et dans l'ordre de leurs occurrences dans δ.

La création de la liste (V) permet d'éviter celle de deux enregistrements définissant un même mot de T^*. Pour ce faire on utilise une liste auxiliaire, la liste VI, qui apparaît comme la liste des adresses de rangement dans LIS, de tous les mots de T^*, pour lesquels on a créé un enregistrement.

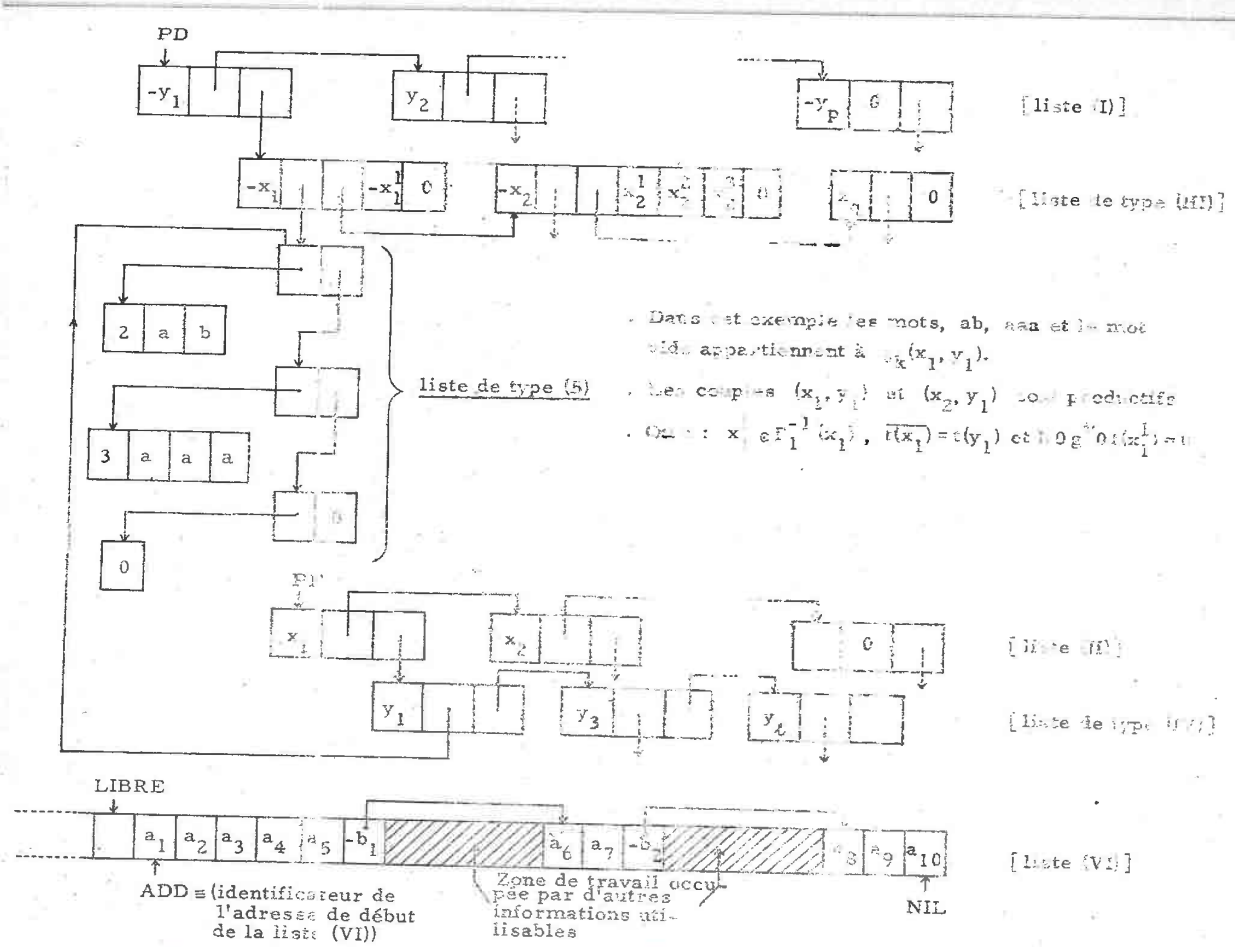
En outre, on s'astreint à ne pas créer deux enregistrements [illegible] type (V) qui fassent référence à un même mot de T^* [illegible]

Liste (VI) : Cette liste est définie dans la zone de travail du tableau LIS ; c'est une liste à enregistrements de longueur variable. Le dernier mot de chaque enregistrement contient l'adresse de l'enregistrement suivant, affectée d'un signe moins.

Tout autre mot nous donne l'adresse d'un mot de T^* qui a été enregistré dans LIS.

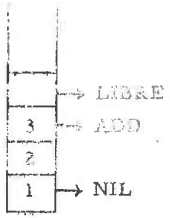
Chaque fois que cela est possible, les enregistrements de cette liste sont «tassés» à la fin du tableau LIS afin de permettre une utilisation de la zone de travail qui soit économique en place. Il en résulte en particulier que cette liste ne pourra jamais avoir plus de trois enregistrements.

Cette liste permet, en outre, un passage aisé des listes (I) à (V) [illegible] de la procédure TRIPLE à celles utilisées lors de l'étude contextuelle, [illegible] paragraphe 5.5.4.


PD
[liste (I)]
[liste de type (III)]
liste de type (5)
. Dans cet exemple les mots, ab, aaa et le mot vide appartiennent à
. Les couples et sont productifs
[liste (II)]
[liste de type (IV)]
LIBRE
[liste (VI)]
ADD ≡ (identificateur de l'adresse de début de la liste (VI))
Zone de travail occupée par d'autres informations utilisables
NIL

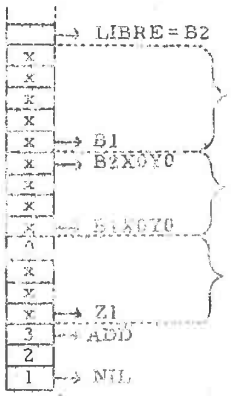
Liste (V1)

a)



Zone de la liste [V1] après l'appel de la procédure INITIA

b)

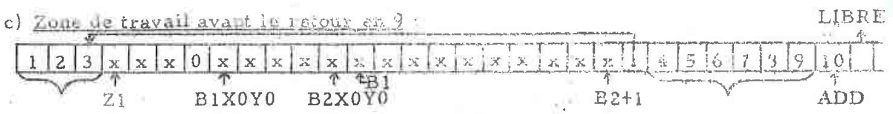


Zone de travail utilisée par COMPAT

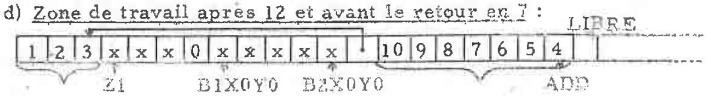
Zone de travail utilisée par X0Y0

Zone de travail utilisée par FAMI

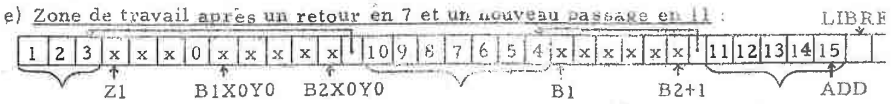
c) Zone de travail avant le retour en 9 :



d) Zone de travail après 12 et avant le retour en 7 :

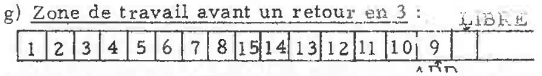


e) Zone de travail après un retour en 7 et un nouveau passage en 11 :



f) Zone de travail avant un retour en 5 :

g) Zone de travail avant un retour en 3 :



5.4.2 - Analyse de la procédure triple

(1) Initialisation : recherche des triplets $(y, y, h \circ g^* \circ \theta_2(y))$ avec $t'(y) \in \nabla_i$ → Procédure INITIA

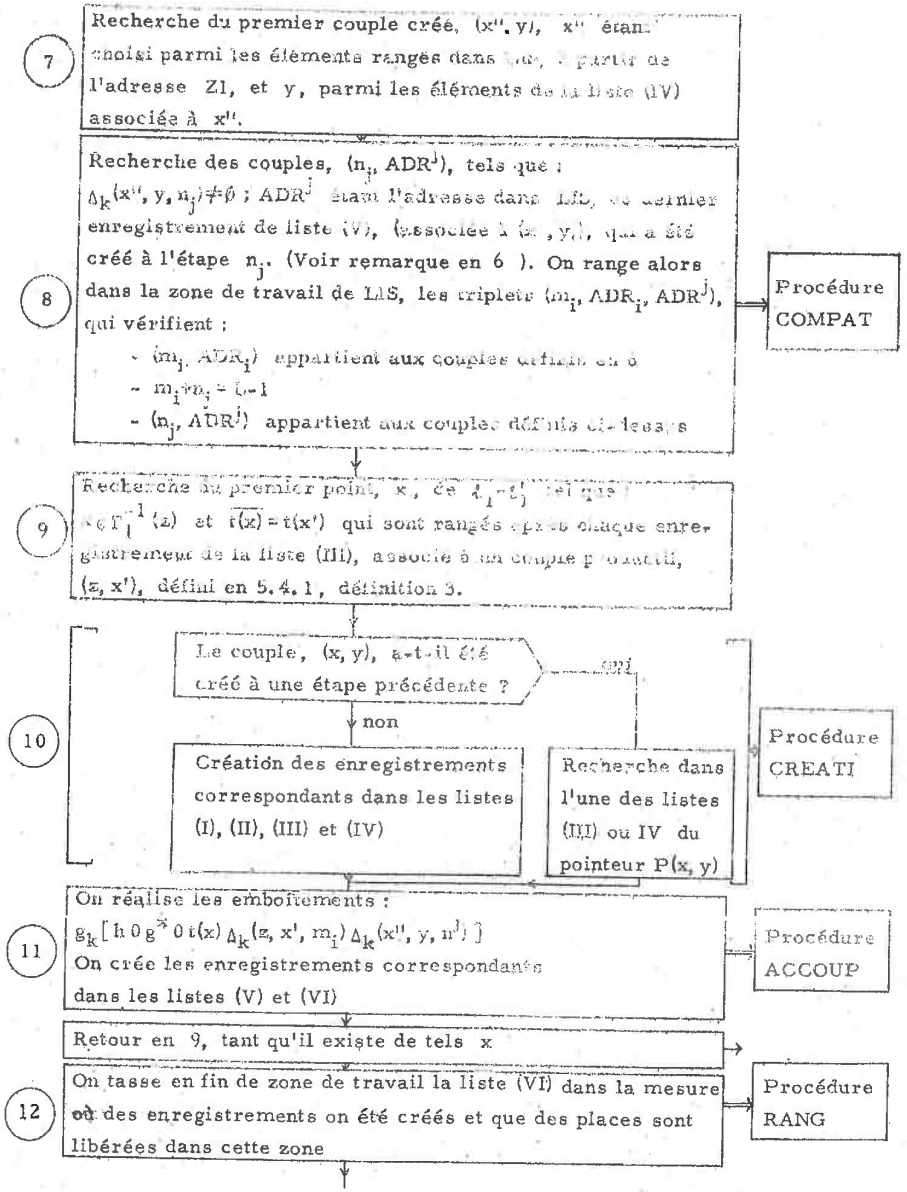
(2) (Initialisation) Itération sur la longueur k des éléments des langages $L(x, y)$, autrement dit, recherche pour tout couple (x, y) de $(\ell_1 \cup \ell'_1) \times \ell'_1$ des ensembles $\Delta_k(x, y, L)$

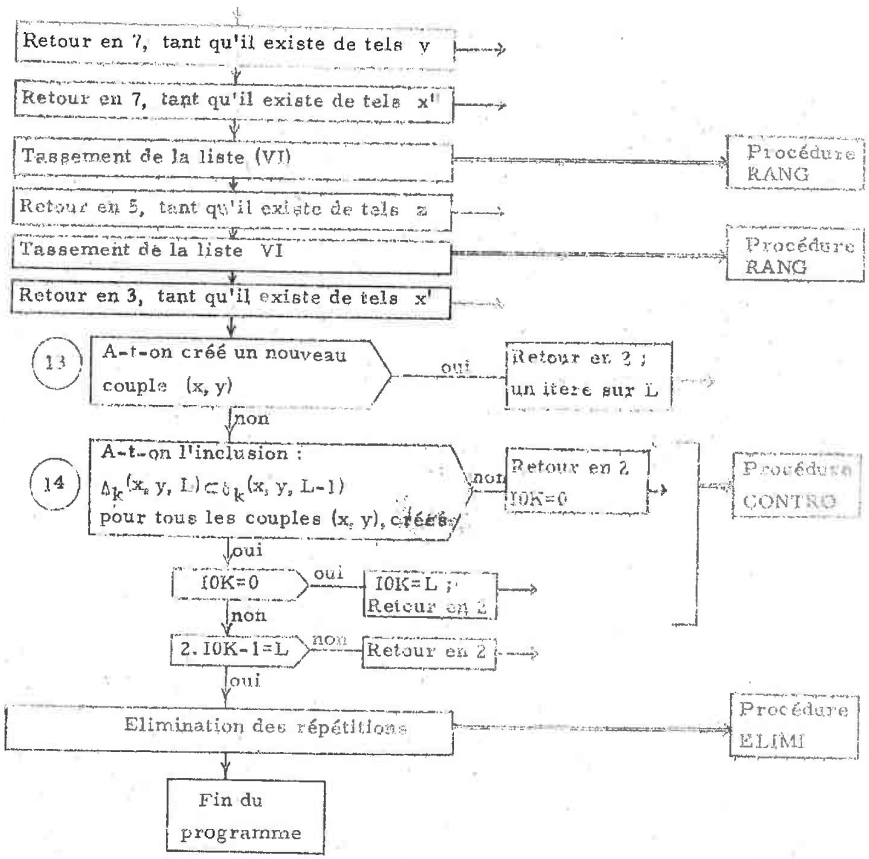
(3) Recherche du premier point, x', de ℓ'_1, productif dans la liste (I).

(4) Recherche des éléments, x'', rangés dans la liste (II) qui appartiennent à $\Gamma_1(x')$. S'il n'en existe pas, on retourne en 3 ; dans le cas contraire, on les range dans la zone de travail de LIS à partir de l'adresse ZI, le dernier élément est adjacent à un mot contenant la valeur zéro. → Procédure FAMI

(5) Recherche du premier, z, de ℓ_i appartenant à la liste (III) associée à x' tel que le couple (z, x') soit productif

(6) On range dans la zone de travail de LIS les couples, (m_i, ADR_i), tels que : $\Delta_k(z, x', m_i) \neq \emptyset$; ADR_i étant l'adresse dans LIS, du dernier enregistrement de la liste (V) associée à (z, x') qui a été créé à l'étape m_i. Cette adresse nous permettra d'atteindre tous les enregistrements créés à l'étape m_i, car la création d'un enregistrement d'une quelconque liste rangée dans LIS respecte toujours la règle suivante :
≪tout enregistrement créé devient le premier enregistrement de la liste à laquelle il appartient ; cela implique en particulier une modification constante des pointeurs de début de liste, tels que PD, PF, → Procédure X0Y0


7
Recherche du premier couple créé, (x'', y), x'' étant choisi parmi les éléments rangés dans LIS, à partir de l'adresse Z1, et y, parmi les éléments de la liste (IV) associée à x''.
8
Recherche des couples, (n_j, ADR^j), tels que :
Δ_k(x'', y, n_j) ≠ ∅ ; ADR^j étant l'adresse dans LIS, du dernier enregistrement de liste (V), associée à (x , y), qui a été créé à l'étape n_j. (Voir remarque en 6). On range alors dans la zone de travail de LIS, les triplets (m_i, ADR_i, ADR^j), qui vérifient :
- (m_j, ADR_i) appartient aux couples définis en 6
- m_i m_j = ζ-1
- (n_j, ADR^j) appartient aux couples définis ci-dessus
Procédure COMPAT
9
Recherche du premier point, x , de Δ_1-Σ_j tel que
x ∈ Γ_1^{-1}(z) et t(x) = t(x') qui sont rangés après chaque enregistrement de la liste (III), associé à un couple précédent, (z, x'), défini en 5.4.1, définition 3.
10
Le couple, (x, y), a-t-il été créé à une étape précédente ?
oui
non
Création des enregistrements correspondants dans les listes (I), (II), (III) et (IV)
Recherche dans l'une des listes (III) ou IV du pointeur P(x, y)
Procédure CREATI
11
On réalise les emboîtements :
g_k[h ∘ g^{*} ∘ t(x) Δ_k(z, x', m_i) Δ_k(x'', y, n^j)]
On crée les enregistrements correspondants dans les listes (V) et (VI)
Procédure ACCOUP
Retour en 9, tant qu'il existe de tels x
12
On tasse en fin de zone de travail la liste (VI) dans la mesure où des enregistrements on été créés et que des places sont libérées dans cette zone
Procédure RANG



Remarque : L'entier, IOK, représente le numéro de l'étape à partir de laquelle le test effectué en 14 est positif.
Lorsque l'on a l'égalité, 2.IOK-1=L, à l'étape L, cela signifie que : IOK=i(k) (voir paragraphe 5.3).
Lorsque le test effectué en 13 est positif, cela implique l'inégalité : $L \leq j(k)$ (voir paragraphe 5.3).

5.5 - ETUDE CONTEXTUELLE

Jusqu'à présent notre seul moyen d'éviter des recherches superflues dans le graphe $g\ell_1$, était de détecter ce que l'on avait appelé "les fausses pistes" (paragraphe 5.2
Dans le cadre d'une analyse du mot α et en une étape donnée d'un programme d'analyse, d'état analytique (x, u, β), un point x de ℓ_1 [illegible] une fausse piste si on ne peut pas élaborer d'analyse de α, à partir de x, compte-tenu des mots u et β
D'après le corollaire de la proposition 2, (paragraphe 5.3.1), il est équivalent d'affirmer que u n'appartient pas à $L(x, \beta)$.
Mais $L(x, \beta)$ est un langage régulier, on pourrait donc envisager de définir un algorithme de génération de ces langages. Déterminer une fausse piste, nous conduirait ainsi à générer pour tout point x de ℓ_1 et tout facteur droit, β, de α les langages $L(x, \beta)$; ce qui en fait nous amène à réaliser des mini-analyses.
Cependant, une étude moins exigeante est celle des facteurs droits, de longueur bornée, des mots de $L(x, \beta)$.
Pour que cette étude contextuelle soit efficace, il ne faut pas qu'elle se fasse dans le cadre de l'analyse d'un mot donné : cette étude doit être valable pour toute analyse de mots du langage engendré par une grammaire donnée. Les seules données d'une grammaire G et d'une structure compatible nous donneront une étude contextuelle qui précédera toute analyse.

5.5.1 - Définition des contextes d'ordre donné d'un point du graphe $g\ell_1$

Définition 4 : Un couple, (v, γ), appartenant à $V_1^* \times T^*$ sera dit un contexte d'ordre (q, k), d'un point x du graphe $g\ell_1$, si : $|v| = q$, $|\gamma| = k$; il existe un mot u, de $\# V^*$, tel que le triplet (x, u, γ) soit acceptable et que l'on ait : $g_q(\tilde{u}) = v$.

On appellera : $\mathcal{C}(q, k, x)$ l'ensemble des contextes d'ordre (q, k), d'un point de $g\ell_1$.

La proposition 2 du paragraphe 5.3.1 nous permet de donner une définition équivalente à la précédente, qui présente l'avantage d'indiquer les grandes lignes d'une recherche pratique des contextes.

Définition 5 : Un couple (v, γ), appartenant à $V_1^* \times T^*$, sera dit un contexte d'ordre (q, k) d'un point x de $g\ell_1$, si : $|v|=q$, $|\gamma|=k$; et il existe un mot m de $\mathcal{L}_k$ tel que :

$p_1 \circ g_1(m)=x$; $g_q[p_2^*(m)]=v$; $g_k[p_3^*(m)]=\gamma$.

Comme pour "les triplets acceptables" la recherche des contextes d'ordre (q, k) dépend essentiellement de la génération des mots du langage $\mathcal{L}_k$. Avant d'étudier plus précisément les ensembles $\mathcal{C}(q, k, x)$ ne retenant que la possibilité de leur recherche par un algorithme, nous nous intéressons à leur rôle dans la détection de certaines "fausses pistes".

5.5.2 - Recherche des "fausses pistes"

Définition 6 : Un point x de ℓ_1 est une fausse piste pour le couple (u, γ), appartenant à $\# V^* \times T^*$, si le triplet (x, u, γ) n'est pas acceptable ou ce qui est équivalent, u n'appartient pas à $L(x, \gamma)$.

Soit x appartenant à ℓ_1 et (x, v, γ) un état analytique avec : $|v|=q'$ et $|\gamma|=k'$.

En raison des définitions 4 et 6 une condition nécessaire et suffisante pour que x soit une fausse piste pour le couple (x, γ) est :

« $(\tilde{v}, \gamma)$ n'est pas un contexte d'ordre (q', k') ».

Comme nous l'avons dit dans l'introduction, cette condition ne présente pas d'intérêt; son utilisation nécessiterait la recherche de contextes d'ordre variable pour chacun des points de ℓ_1. Nous pouvons cependant en déduire une condition moins exigeante :

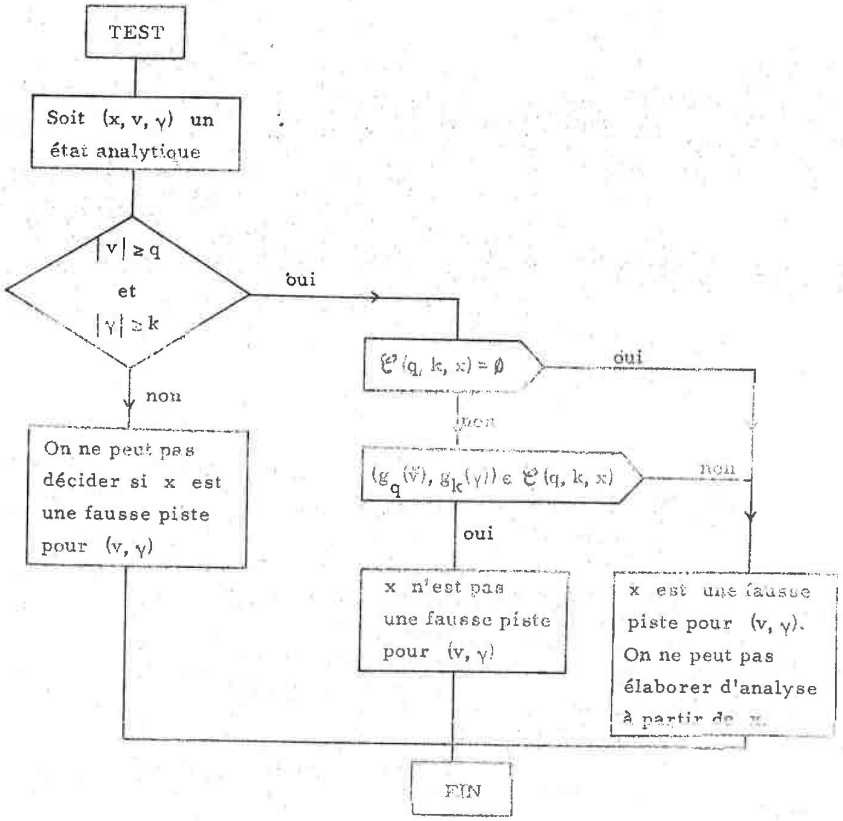
Si nous supposons, $q' \geq q$ et $k' \geq k$(1), une condition suffisante pour que x soit une fausse piste pour le couple (v, γ) est que le couple $(g_q(\tilde{v}), g_k(\gamma))$ ne soit pas un contexte d'ordre (q, k). En effet :

- Si (x, v, γ) est acceptable, a fortiori $(x, v, g_k(\gamma))$ l'est et par suite $(g_q(\tilde{v}), g_k(\gamma))$ est un contexte d'ordre (q, k).

- Si l'une des inégalités (1) n'est pas satisfaite, on ne pourra pas décider si x est fausse piste en ne connaissant que les contextes d'ordre (q, k) ; à l'exception du cas suivant : si, $q' \geq q$ et $k' < k$, et si il existe un mot δ de T_k pour lequel on a,

$g_{k'}(\delta) = \gamma$ et $(g_q(\tilde{v}), \delta) \in \mathcal{C}(q, k, x)$, alors x n'est pas une fausse piste pour (v, γ) puisque le triplet (x, v, δ) étant acceptable, le triplet (x, v, γ) l'est a fortiori.

A partir de ces remarques, on peut définir une procédure qui dans un cas précis nous déterminera les fausses pistes ; cette procédure est celle que l'on avait dénommée, TEST , dans l'analyse par empilement et retour-arrière.



5.5.3 - Etude des ensembles $\mathcal{C}(q, k, x)$

Proposition 3 : Soit $\mathcal{C}(q, k, x)$, x étant un point du graphe $g\mathcal{L}_1$, alors :

$$\mathcal{C}(q, k, x) = \bigcup_{f \in \mathcal{F}} g_q[p_2^*(f)] \times g_k[p_3^*(f)] \qquad (I),$$

où $\mathcal{F}$ est l'ensemble des plus petits facteurs gauches de mots de $\mathcal{L}_k$ qui vérifient :

$$p_1 \circ g_1(f) = x \; ; \; |f| \geq q \; ; \; |p_3^*(f)| \geq k.$$

Démonstration : Soit (v, γ) un contexte d'ordre (q, k) de x.
Par définition il existe un mot m de $\mathcal{L}_k$ tel que :

$g_k \circ p_2^*(m) = v$ et $g_k \circ p_3^*(m) = \gamma$; soit f, le plus petit facteur fauche de m de longueur supérieure ou égale à q qui vérifie : $|p_3^*(f)| \geq k$; on a :
$(v, \gamma) = (g_q[p_2^*(f)], g_k[p_3^*(f)])$.

La réciproque est toute aussi évidente. Si (v, γ) appartient au second membre de l'égalité (I), il existe un mot m de $\mathcal{L}_k$ pour lequel on a : $m = f m'$, $v = g_q[p_2^*(f)]$, et $\gamma = g_k[p_3^*(f)]$.

Par conséquent : $g_q[p_2^*(m)] = g_q[v\, g_q\{p_2^*(m')\}] = v$, $(|v| = q)$,

et, $g_k[p_3^*(m)] = g_k[\gamma\, g_k\{p_3^*(m')\}] = \gamma$, $(|\gamma| = k)$.

La formule (I) nous fournit un algorithme relativement simple pour déterminer les ensembles $\mathcal{C}(q, k, x)$; en particulier la conduite de cet algorithme est analogue à celle de l'analyse par empilement et retour arrière, c'est à dire une recherche séquentielle de chemins, dans un graphe donné, qui doivent vérifier un certain nombre de conditions. Une première difficulté surgit avec la nature du graphe donné qui, en l'occurrence, est celui des transitions du langage $\mathcal{L}_k$. Ce graphe, $\mathcal{G}$, est défini par la donnée des listes où sont rangés les triplets intermédiaires (ensemble $\mathcal{C}_k$), et celle du graphe $g\mathcal{L}_1$; une recherche dans ce graphe est donc mal aisée.
En outre, un tel algorithme conduit à de nombreuses répétitions et ne permet pas l'utilisation des contextes déjà calculés. C'est pourquoi nous ne l'emploierons que pour certains des points du graphe $g\mathcal{L}_1$ (Algorithme 3, 5.5.5.3).

Proposition 4 :

$$\mathcal{C}(q, k, x) = \left\{ \bigcup_{(v,\gamma) \in C_4(x)} (v, \gamma) \right\} \cup \left\{ \bigcup_{(y,\delta) \in C_2(x)} \; \bigcup_{(v,\gamma) \in C_3(y,\delta)} (v, \delta\gamma) \right\} \cup \left\{ \bigcup_{(y,\delta) \in C_0(x)} \; \bigcup_{(v,\gamma) \in C_1(y,\delta)} (v, \delta \cdot \right.$$

avec :

$C_0(x) = \{(y, \delta) \in \mathcal{L}_1' \times T^* \mid h \circ g^* \circ t(y) = \wedge \; ; (x, y, \delta) \in \mathcal{C}_k \; ; \; |\delta| < k\}$

$C_2(x) = \{(y, \delta) \in \mathcal{L}_1' \times T^* \mid h \circ g^* \circ t(y) \neq \wedge \; ; (x, y, \delta) \in \mathcal{C}_k \; ; \; |\delta| < k\}$

$C_1(y, \delta) = \{(v, \gamma) \in V_1^* \times T^* \mid |\gamma| = k - |\delta| \; ; \; (v, \gamma) \in \mathcal{C}(q, k - |\delta|, y)\}$

$C_3(y, \delta) = \{(v, \gamma) \in V_1^* \times T^* \mid |\gamma| = k - |\delta| \; ; \; (v, a\gamma) \in \mathcal{C}(q, k - |\delta| + 1, y) \; ;$
$a = h \circ g^* \circ t(y)\}$

$C_4(x) = \{(v, \gamma) \in V_1^* \times T^* \mid |\delta| = k$; il existe un chemin m de $\mathcal{L}_k$ d'origine (x, y, γ), de longueur supérieure ou égale à q, pour lequel : $g_q[p_2^*(m)] = v\}$.

Démonstration : Soit (v, β) appartenant à $\mathcal{C}(q, k, x)$ et m un mot de $\mathcal{L}_k$ tel que :

$$m = \prod_{i=1}^{p} m_i = \prod_{i=1}^{p} (x_i, y_i, \delta_i) \; ; \; x_1 = x \; ;$$

$$g_k \left[\prod_{i=1}^{p} \delta_i \right] = \delta_1 \gamma = \beta \; ; \; g_q[p_2^*(m)] = v.$$

1) Si, $|\delta_1| = k$, il est équivalent d'affirmer que (v, γ) appartient à $C_4(x)$.

2) Soit, $|\delta_1| < k$: posons $\ell = 1$, si $h \circ g^* \circ t(y) \neq \wedge$, et $\ell = 0$ dans le cas contraire.
En vertu de la propriété 5 du paragraphe 5.4, le mot m' de $(\mathcal{L}_1 \times \mathcal{L}_1' \times T^*)^*$ défini par

$$m' = (y, y, h \circ g^* \circ t(y)) \prod_{i=2}^{p} m_i' \; , \text{ avec } m_i' = (x_i, y_i, \delta_i') \text{ et } \delta_i' = g_{k-|\delta_1|+\ell}(\delta_i) \; , \text{ pour}$$

$i = 2, 3, \ldots, p$, appartient au langage $\mathcal{L}_{k-|\delta_1|+\ell}$, et l'on a :

$$p_2^*(m') = p_2^*(m) \quad ; \quad p_3^*(m') = h \circ g^* \circ t(y) \prod_{i=2}^{p} \delta_i'.$$

En outre, on peut écrire : $g_{k-|\delta_1|+\ell}[p_2^*(m')] = g_\ell[h \circ g^* \circ t(y)]\, g_{k-|\delta_1|} \left[\prod_{i=2}^{p} \delta_i' \right]$,

et $g_{k-|\delta_1|} \left[\prod_{i=2}^{p} \delta_i' \right] = g_{k-|\delta_1|} \left[\prod_{i=2}^{p} g_{k-|\delta_1|+\ell}(\delta_i) \right] = g_{k-|\delta_1|+\ell} \left[\prod_{i=2}^{p} g_{k-|\delta_1|}(\delta_i) \right] =$

Il est donc équivalent d'affirmer :

$|\gamma| = k - |\delta_1|$, et $(v, a\gamma) \in \mathcal{C}(q, k-|\delta_1|+\ell, x)$, avec $q = h\,0\,g^*\,0\,t(y)$.

Proposition 5 : Soit q un entier différent de un, y un élément de $\mathcal{L}'_1$, et ℓ un entier égal à 0 si $h\,0\,g^*\,0\,t(y)$ est le mot vide, et à 1 dans le cas contraire : alors :

$$\mathcal{C}(q, k, y) = \bigcup_{x \in \Gamma_1(y)} \; \bigcup_{(v,\gamma) \in \mathcal{C}(q-1,\, k-\ell,\, x)} (p_2(t)v,\, q\gamma),$$

avec $t = (y, y, q)$ et $a = h\,0\,g^*\,0\,t(y)$.

Démonstration : Soit (u, δ), appartenant à $\mathcal{C}(q, k, y)$, et m le mot de $\mathcal{L}_k$ intervenant dans la définition 5 :

$a = h\,0\,g^*\,0\,t(y)$; $m = \overset{p}{\underset{i=1}{\pi}} (x_i, y_i, \delta_i) = (y, y, a) \; \overset{p}{\underset{i=2}{\pi}} (x_i, y_i, \delta_i)$, avec $g_k\,0\,p_3^*(m)\delta = a\gamma$

et $g_q\,0\,p_2^*(m) = p_2(t)v = u$.

Le mot m' égal à $\overset{p}{\underset{i=2}{\pi}} (x_i, y_i, \delta'_i)$, avec $\delta'_i = g_{k-\ell}(\delta_i)$ pour $i = 2, \ldots, p$, appartient à $\mathcal{L}_{k-\ell}$ et possède les propriétés suivantes : $g_{k-\ell}\,0\,p_3^*(m') = \gamma$ et $g_{q-1}\,0\,p_2^*(m') = v$.

Il est donc équivalent de dire que (u, δ) appartient à $\mathcal{C}(q, k, y)$, et qu'il existe x_2 appartenant à $\Gamma_1(y)$ tel que :

$n = p_2(t)\,v$, $\delta = a\gamma$ et $(v, \gamma) \in \mathcal{C}(q-1, k-\ell, x_2)$.

Remarques :

Les propriétés précédentes mettent en évidence des relations entre les contextes d'ordre (q, k) des éléments de $(\mathcal{L}_1 - \mathcal{L}'_1)$ et ceux, d'un certain ordre, des éléments de $\mathcal{L}'_1$; et en particulier la possibilité de construire les premiers à partir des seconds. Cependant cette solution ne sera intéressante que dans la mesure où l'on peut facilement obtenir les ensembles $\mathcal{C}(q-i, k-j, x)$ $(1 \leq i, j < q, k)$ à partir des ensembles $\mathcal{C}(q, k, x)$ et réciproquement.
Or, s'il est trivial que l'on ait l'implication :

$$(v, \gamma) \in \mathcal{C}(q, k, x) \Longrightarrow (g_{q-i}(v), g_{k-j}(\gamma)) \in \mathcal{C}(q-i, k-j, x) \;,$$

il semble moins évident d'obtenir des contextes d'ordre (q, k) à partir de contextes d'ordre inférieur.

De ce fait, on ne peut espérer obtenir un algorithme de construction directe des contextes des points de $g\mathcal{L}_1$, qui lors de sa recherche, utilise des contextes déjà déterminés. C'est pourquoi nous nous intéresserons essentiellement à un algorithme de génération des contextes qui évitera (lorsque cela sera possible), non pas les répétitions contextuelles mais celles des facteurs gauches de mots de $\mathcal{L}_k$ que l'on engendre simultanément.

5.5.4 - Etude de la liste de rangement des ensembles $\mathcal{C}_k$

La procédure, TRIPLE, nous fournira l'ensemble $\mathcal{C}_k$ sous la forme de la liste (VII définie par le triplet, (TR, 4, 2), et dont les enregistrements sont constitués comme suit :

- le premier mot d'un enregistrement nous donne les premières composantes, x, des couples créés.
- le troisième mot est le pointeur de la sous-liste de type (VIII) associée à une première composante, x, et qui définit les points, y, de $\mathcal{L}'_1$ tels que : $\delta_k(x, y) \neq \emptyset$.
- le quatrième mot contiendra la valeur zéro, il nous servira ultérieurement, lors de la recherche des ensembles $\mathcal{C}(q, k, x)$, comme pointeur des listes de définition des contextes.

Sous-liste de type (VIII) : Une sous-liste de type VIII est associée à chaque première composante, x, d'un couple créé. C'est une liste à enregistrements variables, rangés séquentiellement, et formés :

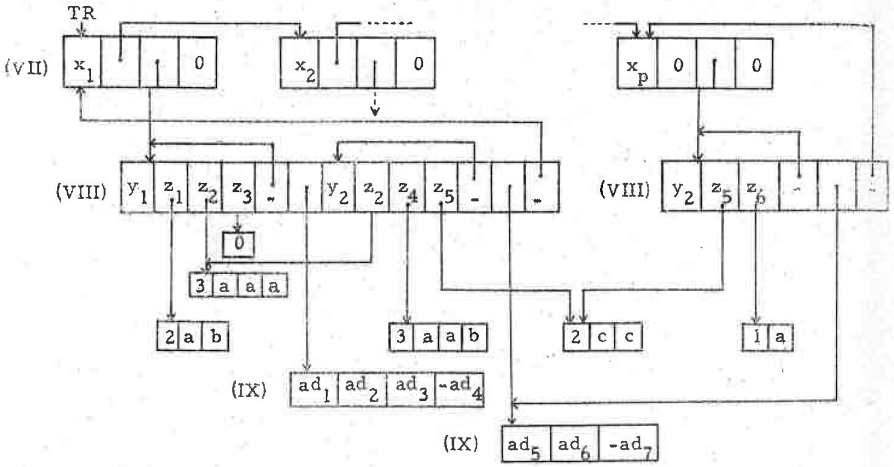
- d'un premier mot qui définit les secondes composantes, y, des triplets (x, y, δ) de $\mathcal{C}_k$.
- puis d'un ensemble de mots adjacents contenant l'adresse de rangement des enregistrements concernant les mots de $\delta_k(x, y)$; ces enregistrements sont constitués de la même façon que dans le tableau LIS, que nous avons introduit au paragraphe 5.4.
- enfin, de deux mots définissant des pointeurs :
  - le premier est l'adresse du premier mot de l'enregistrement, affecté d'un signe moins.
  - le second est l'adresse d'un enregistrement de type (IX).

Le dernier enregistrement d'une sous-liste de type (VIII) est suivi par un mot contenant l'adresse du mot où est rangé le point x de $\mathcal{L}_1$ affectée d'un signe moins. Ceci nous permettra de retrouver le couple créé (x, y) correspondant à chacune des informations de cette sous-liste.

Enregistrement de type (IX) : Il est de longueur variable.

Un tel enregistrement est défini pour chaque point y de ℓ_1' apparaissant comme seconde composante d'un couple créé. Chacun de ses mots contient l'adresse dans la liste (VII) des points x' de ℓ_1 appartenant à $\Gamma_1(y)$. Le contenu du dernier mot est affecté d'un signe moins.

L'ensemble de ces enregistrements définit en quelque sorte la liste des successeurs des points du graphe des transitions, $\mathcal{G}$, du langage $\mathcal{L}_k$.



$\delta_k(x_1, y_1) = \{ab, aaa, \Lambda\}$

$\delta_k(x_1, y_2) = \{aab, aaa, cc\}$

$\delta_k(x_p, y_2) = \{cc, a\}$

Les successeurs de y_1, appartenant à ℓ_1', sont rangés aux adresses : ad_1, ad_2, ad_3 et ad_4 (successeurs pour le graphe g_{ℓ_1}, et appartenant à l'ensemble des premières composantes des triplets de $\mathcal{C}_k$).

On désignera toujours par LIS, le tableau dans lequel sont définies les listes introduites précédemment.

5. 5. 5 - Pseudo-contextes

Nous avons constaté en 5. 5. 3 que la recherche directe des contextes était gênée par une perte d'information : on ne peut pas aisément, à partir d'un contexte calculé, générer d'autres contextes, car dès l'instant où on le range en mémoire, toutes les informations concernant sa construction, et en particulier ses rapports éventuels avec ceux-ci, sont définitivement perdus.

Par contre, une recherche se situant au niveau des mots des langages $\mathcal{L}_k$ peut nous permettre de connaître à tout moment de précieux renseignements sur leur construction.

C'est la notion de pseudo-contextes, basée sur la recherche d'une suite d'adresses (adresse dans la liste de rangement, introduite en 5. 5. 4) qui va nous permettre d'éviter les difficultés précédentes, et nous pourrons en particulier appréhender aussi bien des mots du langage $\mathcal{L}_k$, que ceux des langages $\mathcal{L}_{k'}$, si $k' < k$ et si les listes définies en mémoire déterminent "l'ensemble des triplets intermédiaires $\mathcal{C}_k$".

5. 5. 5. 1 - Définition

Soient deux entiers, k et k', donnés :

Pour tout x de ℓ_1, on désignera par $F_g(k, k', x)$, l'ensemble des facteurs gauches des mots m de $\mathcal{L}_k$ qui vérifient :

$$|p_3^*(m)| \geq k' \quad \text{et} \quad p_1 0 g_1(m) = x.$$

Pour tout M de $F_g(k, k', x)$, on définit :

$$q_2(k', M) = \text{Min}\{j \in \mathbb{N} \mid |p_3^*[g_j(M)]| \geq k'\}.$$

Lorsque la longueur de M est supérieure ou égale à q, on pose :

$q_1(q, k', M) = \text{Max}\{q, q_2(k', M)\} = |M|$: on a ainsi les inégalités : $q_1(q;k';M) > q$ et $q_2(k', M) \leq q_1(q, k', M)$.

Donnons quelques définitions :

- l'application, ADD, associe à tout δ de $\delta_k(x, y)$ l'adresse dans la liste de rangement de l'enregistrement concernant δ.

- l'application, ADY, associe à tout triplet, (x, y, δ), l'adresse dans les listes de type (VIII) de y.

- l'application, DY, associé à l'adresse de rangement d'une deuxième ou d'une troisième composante d'un triplet, (x, y, δ), l'adresse de rangement de la troisième composante.

- soit un entier k' inférieur à k, on désigne par $\underline{ext_k}$ l'application de $\mathcal{L}_{k'}$ dans $\mathcal{L}_k$, qui à tout mot de $\mathcal{L}_{k'}$, $m' = \prod_{i=1}^{p} (x_i, y_i, \delta_i')$, fait correspondre le mot m de $\mathcal{L}_k$ défini par : $m = \prod_{i=1}^{n} (x_i, y_i, \delta_i)$, avec $\delta_i' = g_k(\delta_i)$ et $ADD(\delta_i) = ADD(\delta_i')$.

L'implication, ext_k, étant une bijection, on appellera, $\underline{contra_{k'}}$, l'application réciproque ext_k^{-1}, pour un couple donné $(k;k')$.

Propriété 7 : Soit M' appartenant à $F_g(k, x)$ et k' un entier inférieur à k, alors $M = contra_{k'}(M')$ appartient à $F_g(k', x)$ et $q_2(k, M') \geq q_2(k', M)$.

Démonstration :

$$\left| \prod_{i=1}^{q_2(k, M')} \delta_i \right| \geq k \quad \text{et} \quad \prod_{i=1}^{q_2(k, M')} \delta_i' = \prod_{i=1}^{q_2(k, M')} g_{k'}(\delta_i) ;$$

on en déduit $g_{k'}\left[\prod_{i=1}^{q_2(k, M')} \delta_i \right] = g_{k'}\left[\prod_{i=1}^{q_2(k, M')} g_{k'}(\delta_i) \right] = k'$, et par conséquent

M appartient à $F_g(k', x)$ et $q_2(k, M') \geq q_2(k', M)$.

Définition 7 : Soient les entiers q, k et k', avec $k' \leq k$.
Un pseudo-contexte d'ordre (q, k, k') d'un point x de $g\ell_1$ est un mot ℓ de $\mathbb{N}^*$ pour lequel il existe un mot m de $F_g(k, k', x)$, de longueur supérieure ou égale à q, de telle façon qu'en posant :

$M = g_{\ell_1(q, k', m)}[m] = \prod_{i=1}^{q_1(q, k', m)} M_i$, $(M_i \in \mathcal{L}_k^*)$, on a :

$\ell = \prod_{i=1}^{q_1(q, k', m)} \ell_i$ avec

ℓ_i = SI $i \leq q_2(k', m)$ ALORS $ADD(M_i)$ SINON $ADY(M_i)$.

Cette définition sera surtout utilisée avec $k = k'$, auquel cas on obtient l'ensemble des pseudo-contextes d'ordre (q, k) des points de $g\ell_1$, que l'on note : $P(q, k, x)$.
Lorsque k est différent de k', on a les ensembles $P_{k'}(q, k, x)$. Tout mot p de $P(q, k, x)$ est déterminé par les trois éléments :

$$M(p) \in F_g(k, x) \quad q_1(q, k, M) = |M(p)| \quad \text{et} \quad q_2(k, M).$$

On utilisera toujours ces notations pour désigner un mot de $P(q, k, x)$.

Contextes et pseudo-contextes d'ordre (q, k) des points de $g\ell_1$

On obtient aisément les contextes d'ordre (q, k) à partir des pseudo-contextes d'ordre (q, k) ; en effet, si p appartient à $P(q, k, x)$:

- pour obtenir la première composante, v, de contexte associé à p, il suffit d'une part de rechercher, pour $i \leq q_2(k, M(p))$, et à partir de l'adresse p_i, celle contenue dans le dernier mot de l'enregistrement contenant le mot adressé par p_i et appartenant à une sous-liste (VIII) ; et d'autre part de concaténer les contenus des premières à ceux des adresses p_i, pour $i > q_2(k, M(p))$. Ces opérations doivent être exécutées tant que i est inférieur à q.

- la seconde composante, quant à elle, s'obtient directement en concaténant les contenus des adresses, $(p_i'+1)$ à $(p_i')+((p_i'))$, ((ADD) désignant le contenu du mot adressé par ADD), pour $i \leq q_2(k, M(p))$.

5. 5. 5. 2 - Quelques propriétés sur les pseudo-contextes

Proposition 6 : I) Supposons que nous connaissions les pseudo-contextes d'ordre (q, k') de x, avec $k' < k$, alors :

$P(q, k, x) = U_1 \cup U_2 \cup U_3$, avec

$$U_1 = \bigcup_{p' \in A_1} \bigcup_{x' \in B_1(p')} g_{q_2(k', M')}(p') \; P_{j(p')}[q - q_2(k', M'), k, x']$$

$$U_3 = \{p' \in P(q, k', x) \mid M' = M'(p') \; ; \; M = ext_k(M') \; ; \; q_2(k', M') = q_2(k, M)\}$$

$$A_1 = \{p' \in P(q, k', x) \mid M' = M'(p') \; ; \; M = ext_k(M') \; ; \; q_2(k', M') < q_2(k, M) \; ; \; q_2(k', M') < q\}$$

$$j(p') = k - |p_3^*[g_{q_2(k', M')}(M)]|$$

$$B_1(p') = \{x' \in \Gamma_1[v_{q_2(k', M')}] \mid P_{j(p')}(q - q_2(k', M'), k, x') \neq \emptyset\}$$

$$U_2 = \bigcup_{p' \in A_2} \bigcup_{x' \in B_2(p')} p' \, P_{j'(p')}[1, k, x']$$

$$A_2 = \{p' \in P(q, k', x) \mid M' = M'(p'); M = ext_k(M'); q_2(k', M') < q_2(k, M); q_2(k', M') \geq q\}$$

$$j'(p') = k - |p_3^*[M]|$$

$$B_2(p) = \left\{x' \in \Gamma_1 \left[y_{q_2(k', M')}\right] \mid P_{j'(p')}[1, k, x'] \neq \emptyset\right\}$$

II) Supposons que nous connaissions les pseuso-contextes d'ordre (q, k) de x, alors : $\forall k' < k$,

$$P(q, k', x) = U_1' \cup U_2' \cup U_3', \quad \text{avec}$$

$$U_3' = \{p \in P(q, k, x) \mid M = M(p) ; M' = contra_{k'}(M) ; q_2(k', M') = q_2(k, M)\}$$

$$U_1' = \bigcup_{p \in B_1'} g_{q_2(k', M')}[p] \; DY^*\left[\pi_{i=q_2(k', M')+1}^{p} p_i\right]$$

$$B_1' = \{p \in P(q, k, x) \mid M = M(p) ; M' = contra_{k'}(M) ; q_2(k', M') < q_2(k, M) ; q_2(k', M') < q\}$$

$$U_2' = \bigcup_{p \in B_2'} g_{q_2(k', M')}[p]$$

$$B_2' = \{p \in P(q, k, x) \mid M = M(p) ; M' = contra_{k'}(M) ; q_2(k', M') < q_2(k, M) ; q_2(k', M') \geq q\}$$

Démonstration :

Passage de $P(q, k', x)$ à $P(q, k, x)$: Soit p' appartenant à $P(q, k', x)$ et $M' = M'(p')$, le facteur gauche de $F_g(k', x)$ associé à p'.

. Si $M = ext_k(M')$, deux cas se présentent : $p' = \pi_{i=1}^{q_1(k', M')} p'_i$, avec $p'_i = (x_i, y_i, \delta'_i)$

1) $g_{q_2(k', M')}[M]$ n'appartient pas à $F_g(k, x)$, ce qui est équivalent à $q_2(k', M') < q_2(k, M)$, d'après la propriété 7 :

1.1 - Supposons $q_2(k', M') < q$: soit x' appartenant à $\Gamma_1\left[y_{q_2(k', M')}\right]$; si $P_{j(p')}[q - q_2(k', M'), k, x]$ est non vide et p'', l'un de ses éléments alors le mot p égal à $g_{q_2(k', M')}[p'] . p''$ appartient à $P(q, k, x)$; en outre il est équivalent

d'affirmer que $g_{q_2(k', M')}[p] \; DY^*\left[\pi_{i=q_2(k', M')+1}^{q_1(q, k, M)} p_i\right]$ appartient à $P(q, k', x)$, si $M' = contra_{k'}[M(p)]$.

1.2. Si $q_2(k', M') = q$: soit x' appartenant à $\Gamma_1\left[y_{q_2(q, k', M')}\right]$ et p' à $P_{j'(p')}(1, k, x')$, alors $p = g_{q_2(k', M')}(p')$ p' appartient à $P(q, k, x)$: il est équivalent d'affirmer que $g_{q_2(k', M')}[p]$ appartient à $P(q, k', x)$, avec $M' = contra_{k'}[M(p)]$.

2) $c_2(k', M') = q_2(k, M)$: on en déduit $q_1(q, k, M) = q_1(q, k', M')$ et ainsi le mot p associé à M appartient à $P(q, k, x)$

On obtient ainsi les inclusions :

$$U_1 \cup U_2 \cup U_3 \subset P(q, k, x) \quad \text{et} \quad P(q, k', x) \subset U_1' \cup U_2' \cup U_3' .$$

Le passage de $P(q, k, x)$ à $P(q, k', x)$, nous donnerait les inclusions symétriques par des considérations toutes aussi évidentes.

Proposition 7 :

$$P(q, k, x) = \left\{ \bigcup_{(y, \delta) \in C_0(x, k)} \bigcup_{p \in D_1(y)} ADD[(x, y, \delta)] p \right\} \cup \left\{ \bigcup_{(y, \delta) \in C_1(x, k)} \bigcup_{p \in D_2(y)} ADD[(x, y, \delta)] p \right\}$$

$$C_0(x, k) = \{(y, \delta) \in \mathcal{L}_1' \times \Gamma^* \mid |\delta| < k \; , \; \delta \in \delta_k(x, y)\}$$

$$D_1(y) = \{p \in \mathbb{N}^* \mid ADD[(y, y, h \circ g^* \circ t(y))] \, p \in P(q, k - i(y, \delta), y)\}$$

$$i(y, \delta) = \text{SI } h \circ g^* \circ t(y) = \wedge \text{ ALORS } |\delta| \text{ SINON } |\delta| - 1$$

$$C_1(x, k) = \{(y, \delta) \in \mathcal{L}_1' \times T^* \mid |\delta| = k, \; \delta \in \delta_k(x, y)\}$$

$$D_2(y) = \{p \in P_0(q-1, k, x') \; , \; x' \in \Gamma_1(y)\}$$

Démonstration : Soit p appartenant à P(q, k, x) et M(p), le facteur gauche associé à p : $p = p_1 p'$, $M_1 = (x, y, \delta)$ et $M = M_1 M' = \pi_{i=1}^{q_1(q,k,M)} (x_i, y_i, \delta_i')$

1) $q_2(k, M) = 1$: il est évident que p' appartient à $P_0(q-1, k, x')$ avec x' appartenant à $p_i \circ \mathcal{Q}(M_1)$.

2) $q_2(k, M) \neq 1$: soit $\delta = ADD^{-1}(M_1)$ et si $\ell = i(y, \delta)$, posons $M''' = (y, y, h \circ g^{*} \circ t(y)) M''$ avec $M'' = contra_{k-\ell}(M')$.

On a : $|M'''| = |M''| + 1 = q_1(q, k, M) \geq q$ et $p_3^{*}(M''') = a\, p_3^{*}(M'') = a \pi_{i=2}^{q_1(q,k,M)} g_{k-\ell}(\delta_i')$

M''' appartient à $F_g(k-\ell, y)$, et de plus :

$\forall j < q_2(k, M) : |p_3^{*}[g_j(M)]| < k \Longrightarrow |p_3^{*}[g_{j-1}(M')]| < k - |\delta|$

Mais, $k-\ell \geq k-|\delta|$, par conséquent :

$$p_3^{*}[g_{j-1}(M')] = g_{k-\ell}\left\{p_3^{*}[g_{j-1}(M')]\right\} = g_{k-\ell}\left[\pi_{i=2}^{j} \delta_i'\right] = g_{k-\ell}\left[\pi_{i=2}^{j} g_{k-\ell}(\delta_i')\right]$$

$$= g_{k-|\delta|}\left[\pi_{i=2}^{j} g_{k-\ell}(\delta_i')\right] = g_{k-\ell}\left[p_3^{*} \circ g_{j-1}(M'')\right]$$

il en résulte : $|p_3^{*}[g_j(M''')]| = |a\, p_3^{*} \circ g_{j-1}(M'')| < k-\ell$ et $q_2(k-\ell, M'') = q_2(k, M)$

. Le mot p'' égal à $ADD[(y, y, h \circ g^{*} \circ t(y))]p'$ appartient à $P(q, k-\ell, y)$; son facteur gauche associé est M''', et $q_1(q, k-\ell, M''') = q_1(q, k, M)$.

Le raisonnement inverse, partant de M''', s'établirait de la même façon en prenant $M' = ext_k(M'')$.

Proposition 8 : Soit q un entier différent de 1 et x appartenant à $g\ell_1$, alors :

1) $P(q-1, k, x) = \left(\bigcup_{p \in C_1} p\right) \cup \left(\bigcup_{p \in C_2} g_{q-1}(p)\right) \cup \left(\bigcup_{p \in C_3} p\right)$

$C_1 = \{p \in P(q, k, x) \mid q_2(k, M) \geq q, \text{ avec } M = M(p)\}$

$C_2 = \{p \in P(q, k, x) \mid q_2(k, M) < q, \text{ avec } M = M(p)\}$

$C_3 = \{p \in P(q-1, k, x) \mid M = M(p) ; q_2(k, M) < q ; C_4(p) = \{\emptyset\}\}$

$C_4(p) = \{t \in \mathcal{C}_k \mid M = M(p) = \pi_{i=1}^{|M|} M_i ; t \in \mathcal{Q}[M_{q-1}]\}$

2) $P(q, k, x) = \left(\bigcup_{p' \in C_1'} p'\right) \cup \left(\bigcup_{p' \in C_2'} \bigcup_{t \in C_4(p')} p' ADY[t]\right)$

$C_1' = \{p' \in P(q-1, k, x) \mid q_2(k, M') \geq q, \text{ avec } M' = M'(p')\}$

$C_2' = \{p' \in P(q-1, k, x) \mid q_2(k, M') < q ; M' = M'(p') ; C_4(p') \neq \{\emptyset\}\}$

Démonstration :

1) Passage de P(q, k, x) à P(q-1, k, x) : Soit p appartenant à P(q, k, x) et M(p), le facteur gauche associé à p.

1.1. Si $q_2(k, M) \geq q$, alors $q_2(k, M) = q_1(q, k, M) = q_1(q-1, k, M)$, et p appartient à P(q-1, k, x).

1.2. Si $q_2(k, M) < q$, $q_1(q, k, M) = q$ et par conséquent $g_{q-1}(p)$ appartient à P(q-1, k, x) ; on en déduit aussi $p = g_{q-1}(p) ADY[t]$, avec t appartenant à $\mathcal{Q}[M'_{q-1}]$ si on prend pour M' le facteur $g_{q-1}(M)$ (on a en effet $q_1(q-1, k, M') = q-1$).

2) Passage de P(q-1, k, x) à P(q, k, x) : Soit p' appartenant à P(q-1, k, x) et M'(p') le facteur gauche associé à p'.

2.1. Si $q_2(k, M') \geq q$, alors $q_2(k, M') = q_1(q-1, k, M') = q_1(q, k, M')$ et p' appartient à P(q, k, x).

2.2. Si $q_2(k, M') < q$, alors $q_1(q-1, k, M') = q-1$. Deux cas se présentent :

soit $M' = \pi_{i=1}^{q-1} M_i' = \pi_{i=1}^{q-1} (x_i, y_i, \delta_i)$,

- dans le cas où M'_{q-1} ne possède pas de successeur dans le graphe $\mathcal{G}$, p' appartient $\left\{\bigcup_{p \in C_3} p\right\}$.

- dans le cas contraire, soit t appartenant à $\mathcal{G}(M'_{q-1})$, alors p=p' ADY(t) appartient à P(q, k, x) ; son facteur gauche associé est M=M't.

Proposition 9 : Pour tout y de ℓ'_1 et tout entier q supérieur à un, on a :

(I) $P(q, k, y) = \bigcup_{x \in \Gamma_1(y)} ADD[(y, y, h\,0\,g^{*}\,0\,t(y))]\,P(q-1, k-\ell, x)$,

avec ℓ = SI $h\,0\,g^{*}\,0\,t(y) = \Lambda$ ALORS 0 SINON 1

Démonstration : Soit p appartenant à P(q, k, x) : $p = \pi_{i=1}^{q_1(q,k,M)} p_i$, si le facteur gauche associé à p est $M = M(p) = \pi_{i=1}^{q_1(q,k,M)} M_i$, avec $M_i = (x_i, y_i, \delta_i)$.

Il est équivalent d'écrire : $p = p_1 p'$ et $M = M_1 M'$, avec $M_1 = (y, y, h\,0\,g^{*}\,0\,t(y))$ et $p_1 = ADD(M_1)$.

1) $h\,0\,g^{*}\,0\,t(y) = \Lambda$: M' appartient à $F_g(k, x_2)$ et x_2 à $\Gamma_1(y)$; en outre, la longueur de M' est supérieure ou égale à q-1.

$\forall j < q_2(k, M)-1 : g_{j+1}(M) \in F_g(k, y)$; mais $g_{j+1}(M) = M_1 g_j(M')$.

Il est donc équivalent d'écrire :

$$\forall j < q_2(k, M)-1, \; | p_3^{*}\,0\,g_j(M') | < k.$$

On en déduit : $q_1(q-1, k, M') = q_1(q, k, M)-1$ et $q_2(k, M') = q_2(k, M)-1$.

Le mot p' associé au facteur gauche M' appartient à P(q-1, k, x). On obtient ainsi l'égalité (I), avec $\ell = 0$.

2) $h\,0\,g^{*}\,0\,t(y) \neq \Lambda$: Soit $M'' = contra_{k-1}(M')$, d'après la propriété 7, on a :

M'' appartient à $F_g(k-1, x_2)$ et $q_2(k-1, M'') \leq q_2(k, M')$.

En outre : $\forall j < q_2(k, M), | p_3^{*}\,0\,g_j(M) | < k$; il est équivalent d'affirmer :

$\forall j < q_2(k, M)-1, | p_3^{*}\,0\,g_j(M'') | < k-1$, puisque

$$g_{k-1}\,0\,p_3^{*}\,0\,g_j(M') = \pi_{i=2}^{j} g_{k-1}(\delta_i) = g_{k-1}\,0\,p_3^{*}\,0\,g_j(M'').$$

On en déduit : $q_2(k-1, M'') = q_2(k, M)-1$ et $q_1(q-\ell, k-1, M'') = q_1(q, k, M)-1$.

Le mot p' associé au facteur gauche M'' appartient à P(q-1, k-1, x). On obtient ainsi l'égalité (I) avec $\ell = 1$, compte-tenu de l'équivalence :

$$M'' = contra_{k-1}(M') \Longleftrightarrow M' = ext_x(M'').$$

Les propositions 6, 7, 8 et 9 sont susceptibles de nous donner des algorithmes de génération de pseudo-contextes à partir d'autres pseudo-contextes:

Soit un entier k' inférieur à k, la proposition 6 détermine les pseudo-contextes d'ordre (q', k') de tout point x de ℓ_1 à partir des pseudo-contextes d'ordre (q, k) et réciproquement.

$\forall x \in \ell_1$ $\quad P(q, k, x) \longrightarrow P(q, k', x)$ (Algorithme 6.1)

$\quad P(q', k', x) \longrightarrow P(q, k, x)$ (Algorithme 6.2)

La proposition 8 définit les pseudo-contextes d'ordre (q, k) de tout point de ℓ_1, en fonction des pseudo-contextes d'ordre (q-1, k), et réciproquement,

$\forall x \in \ell_1$ $\quad P(q, k, x) \longrightarrow P(q-1, k, x)$ (Algorithme 8.1)

$\quad P(q-1, k, x) \longrightarrow P(q, k, x)$ (Algorithme 8.2)

En fait, seul l'algorithme 8.2 est intéressant.

Soit x appartenant à $(\ell_1 - \ell'_1)$, et si l'on connaît les pseudo-contextes d'ordre (q, k) pour tous les points y de ℓ'_1, tels que (x, y) soit un couple créé (§ 5.4), la proposition 7, par l'intermédiaire de l'algorithme 6.1 détermine les contextes d'ordre (q, k) de x.

$(\forall x \in (\ell_1 - \ell'_1))\,(\forall y \in \ell'_1)\,(\delta_k(x, y) \neq \emptyset)$.

. $P(q, k, y) \longrightarrow (\forall \delta \in \delta_k(x, y))\,(P(q, k-|\delta|+\ell, y)) \longrightarrow P(q, k, x)$ (Algorithme 7)

. ℓ = SI $h\,0\,g^{*}\,0\,t(y) = \Lambda$ ALORS 0 SINON 1 $\quad (k-|\delta|+\ell < k)$

Lorsque l'on connaît les pseudo-contextes d'ordre (q, k) d'un point de ℓ'_1, la proposition 9 nous donne le moyen de rechercher les ensembles $P(q-\ell, k-\ell, x)$, avec x appartenant à $\Gamma_1(y)$.

$\forall y \in \ell'_1$: $P(q, k, y) \longrightarrow (\forall x \in \Gamma_1(y))\,(P(q-\ell, k-\ell, x))$ (Algorithme 9)

Nous élaborerons à partir de ceux-ci un algorithme de recherche des pseudo-contextes d'ordre (q, k) et en particulier des contextes d'ordre (q, k), de tous les points de $g\ell_1$, qui toutefois ne traite pas systématiquement en séquence chacun d'eux mais explore globalement le graphe $g\ell_1$.

Avant d'étudier cet algorithme nous nous intéresserons à celui qui dérive de la proposition 3, que nous avions évoqué au paragraphe 5. 5. 3. Il nous donnera le moyen de déterminer les pseudo-contextes des points pour lesquels on ne peut pas utiliser des pseudo-contextes déjà calculés, en particulier pour les points suivants :

$$(\forall\, y \in \ell_1')\ (P(q, k, y) = \{\wedge\})\ (\exists\, z \in \ell_1')\ (y \in \Gamma_1(z) \text{ et } P(q, k, z) \neq \{\wedge\})$$

5. 5. 5. 3 - Détermination des pseudo-contextes par la recherche de facteurs gauches de longueur croissante

Cet algorithme recherche les ensembles $P(q, k, x)$ et $P_{k'}(q, k, x)$, avec pour k' un entier inférieur à k.

Il utilise trois piles : $p = (p_i)_i$; $v = (v_i)_i$; $u = (u_j)_j$.

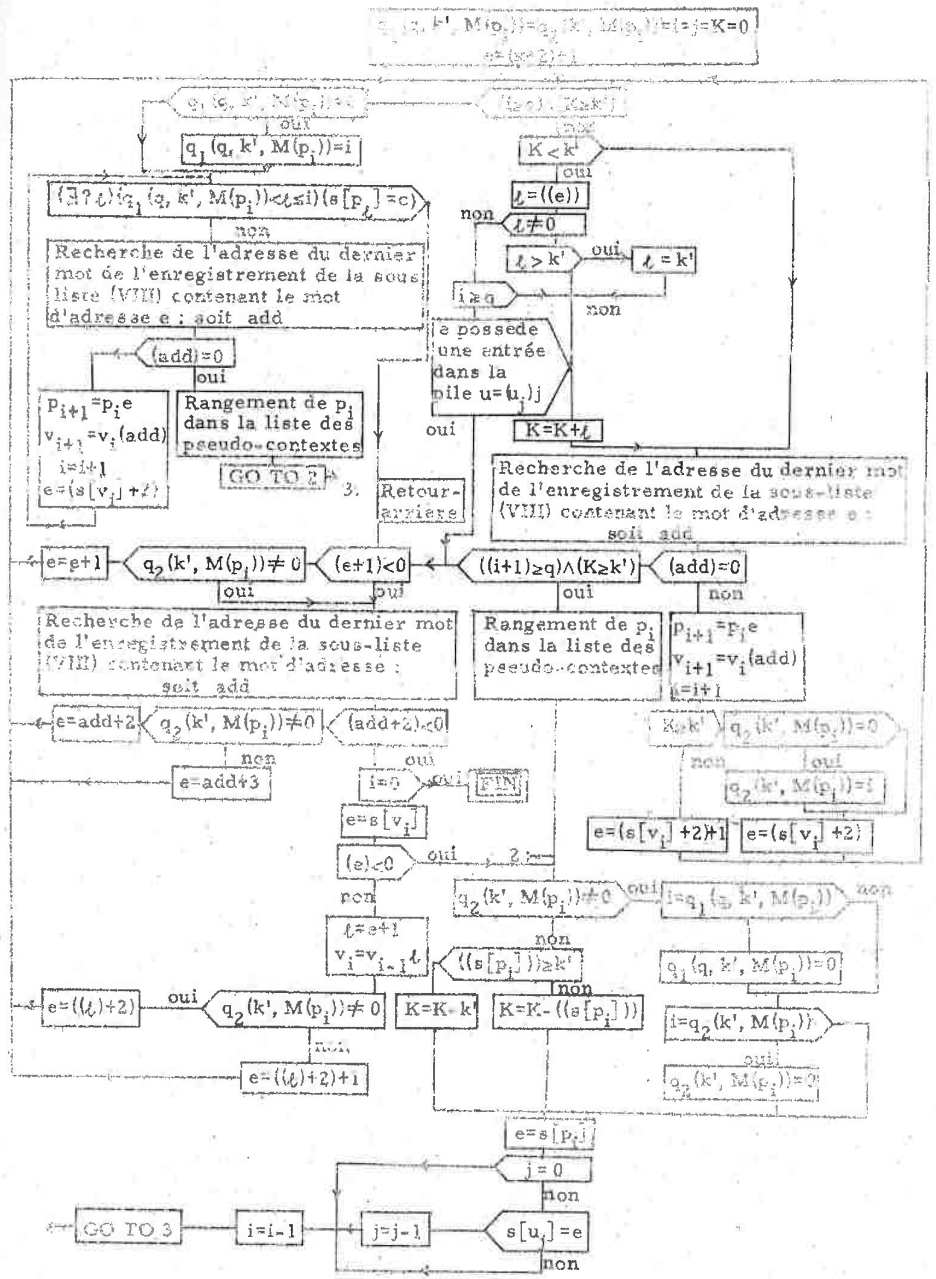
La pile, $p = (p_i)_i$, définit le facteur, $g_i(p)$, engendré par l'algorithme, susceptible d'être le facteur gauche d'un pseudo-contexte p. Comme il l'a été dit au paragraphe 5. 5. 5. 1, à chaque état, p_i, ou encore à chaque facteur $g_i(p)$, est associé un mot $M(p_i)$ de $\mathscr{C}_k^{\pm}$ susceptible d'être le facteur gauche d'un mot de $F_g(k, k', x)$. Tant que l'inégalité $|p_3^{\pm}[M(p_i)]| < k'$ est vérifiée, la pile p est une pile sur l'ensemble des adresses des enregistrements définissant les mots δ de $T^{\pm}$, appartenant à $p_3(\mathscr{C}_k)$. Lorsque pour un entier i, on a l'inégalité $|p_3^{\pm}[M(p_i)]| \geq k'$, le mot $M(p_i)$ appartient à $F_g(k, k', x)$, (modulo (1)), et par conséquent : $i = q_2(k', M(p_i))$; de plus, si i est inférieur à q, la pile p est employée comme pile sur l'ensemble des adresses des mots contenant les secondes composantes des couples créés (sous-listes VIII).

Lorsque les deux inégalités, $|p_3^{\pm}[M(p_i)]| \geq k'$ et $i \geq q$, sont satisfaites, l'entier i est égal à $q_1[q, k', M(p_i)]$ (modulo (1)); il suffit alors de trouver «un chemin élémentaire du graphe $\mathscr{G}$, d'origine $M_{q_1}(q, k', M(p_i))$ et d'extrémité appartenant à $\mathscr{C}_k'$», soit (1), pour affirmer que p_i appartient à $P_{k'}(q, k, x)$: la pile p est utilisée comme pile simple pour cette recherche.

La pile v est entretenue simulatnément à la pile p de telle façon que chaque entrée i dans la pile p est suivie de l'entrée dans v, de l'adresse dans l'enregistrement de type (IX), du mot contenant la première composante du premier successeur de $g_1[\widetilde{M(p_i)}]$ pour le graphe $\mathscr{G}$.
La pile simple u sert à accélérer la recherche dans certains cas : en effet lorsque, $i \geq q$ et $|p_3^{\pm}[M(p_i)]| < k'$, il est inutile d'engendrer des facteurs gauches de mots de $F_g(k, k', x)$, en répétant des triplets de la forme : $(x', y', \wedge)$, dont la contribution au mot des feuilles est nulle.

Algorithme 3 : Recherche de $P_{k'}(q, k, x)$, avec $k' \leq k$

- X représente l'adresse dans la liste (VII) du mot contenant x
- $K = |p_3^{\pm}[M(p_i)]|$
- Pour chaque adresse adr, repérant un mot des listes définies en 5. 5. 5. 3, (adr) désigne le contenu de ce mot.
- Pour chaque pile, $u = (u_i)_i$, utilisée, $s[u_i]$, désigne le sommet de l'état u_i.

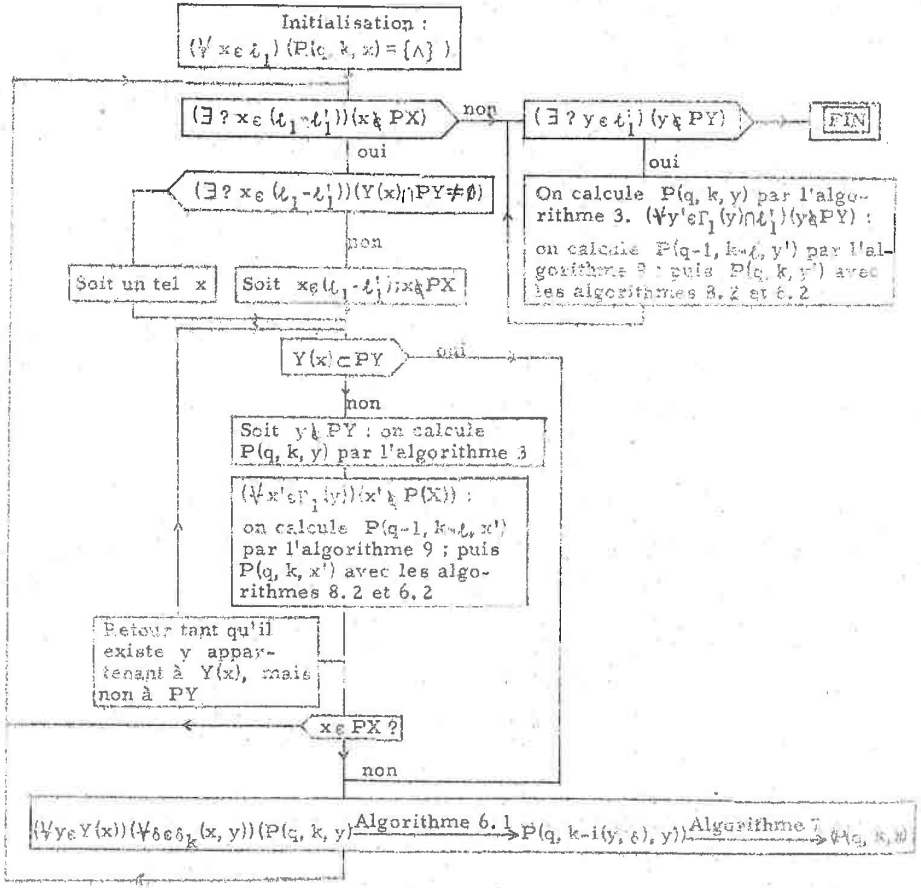

oui
non
Recherche de l'adresse du dernier mot de l'enregistrement de la sous-liste (VIII) contenant le mot d'adresse e : soit add
(add)=0
oui
Rangement de p_i dans la liste des pseudo-contextes
GO TO 2
3.
Retour-arrière
e possede une entrée dans la pile u=(u_j)j
K=K+ℓ
Recherche de l'adresse du dernier mot de l'enregistrement de la sous-liste (VIII) contenant le mot d'adresse e : soit add
e=e+1
e=add+2
e=add+3
i=0
FIN
e=s[v_i]
e=s[p_i]
j=0
j=j-1
s[u_j]=e
i=i-1
GO TO 3

5. 5. 5. 4 - Algorithme général de recherche des pseudo-contextes d'ordre (q, k), des points de $g\mathcal{L}_1$

$Y(x) = \{ y \in \mathcal{L}_1' \mid (x, y)$ est un couple créé $\}$, avec x appartenant à $\mathcal{L}_1$

$PY = \{ y \in \mathcal{L}_1' \mid P(q, k, y) \neq \{\wedge\} \}$

$PX = \{ x \in \mathcal{L}_1 - \mathcal{L}_1' \mid P(q, k, x) \neq \{\wedge\} \}$



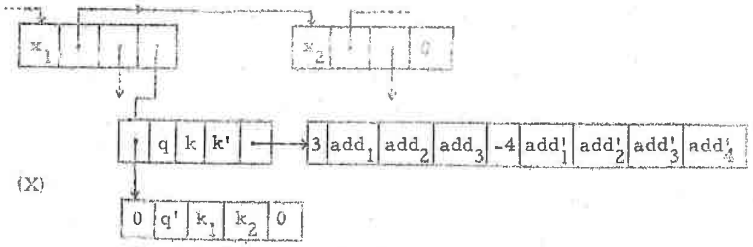
5. 5. 5. 5 - Liste de rangement des pseudo-contextes

Le quatrième mot de chaque enregistrement de liste (VII) repère une sous-liste de définition des pseudo-contextes : liste (X).

Liste (X), associée à un point x de t_1 : Si ADD est le contenu du quatrième mot de l'enregistrement définissant x dans (VII), cette liste est définie par le triplet : (ADD, 5, 1).

- Supposons qu'un algorithme ait déterminé P(q, k, x) , ($k' \leq k$) :
  - dans le deuxième mot, on place q ; k dans le troisième et k' dans le quatrième.
  - le cinquième mot repère la sous-liste de rangement des éléments de P(q, k, k', x) qui est une liste à enregistrements séquentiels du même type que ceux où l'on range les mots des langages $\delta_k(x, y)$.

(VII)



- $P_{k'}(q, k, x_1) = \{(add_1)(add_2)(add_3) ; (add'_1)(add'_2)(add'_3)(add'_4)\}$
- $P_{k_2}(q', k_1, x_1) = \{\emptyset\}$
- $(\forall q)(\forall k)(\forall k')(P_{k'}(q, k, x_2) = \{\wedge\}) \Longleftrightarrow$ { On n'fait aucune tentative de recherche de $P_{k'}(q, k, x_2)$ }

5. 5. 6 - Choix des "ordres contextuels"

Jusqu'à présent toute l'étude sur les contextes disposait a priori d'un ensemble de triplets intermédiaires $\mathcal{C}_k$, d'ordre donné k, défini par une structure de liste. Il nous faut cependant déterminer un critère de choix d'un entier k qui permettra, lors de la recherche des contextes, d'atteindre les ensembles $\mathcal{C}_{k'}$ (pour tout entier $k' < k$) dont on aura éventuellement besoin.

Un choix plus ambitieux serait, pour chaque point x du graphe $g\ell_1$, celui d'un couple d'entiers (q, k) pour lequel on recherchait les contextes de x.

Une première nécessité est de déterminer par l'algorithme 3 les contextes d'ordre (1,1) afin d'éliminer les points de $g\ell_1$ dont les transformés par g^*0t ne possèdent pas d'occurrence dans les mots du langage de parenthèses (choisi comme représentation de S(G)) associés aux ramifications de R(T, N) :

- En effet, d'après les remarques qui suivent la proposition 5 de ce même chapitre,

$$(\forall\, x \in g\ell_1)\ (\mathcal{C}(1,1,x) = \emptyset) \Longrightarrow (\forall k, q \in \mathbb{N})\ (\mathcal{C}(q,k,x) = \emptyset).$$

Cette recherche sera inutile lorsque la grammaire donnée, G = (V, →, X), est réduite, c'est à dire si pour tout A de V, il existe une ramification de S(G) où A possède au moins une occurrence.

Soient x un point de $g\ell_1$, et $\Gamma_1(x) = \{y_1, y_2\}$; supposons que nous connaissions un couple (q, k) : $\mathcal{C}(q,k,y_1) \cap \mathcal{C}(q,k,y_2) = \emptyset$. Les programmes d'analyse devront choisir à partir du point x, les calculs à élaborer pour engendrer une "analyse" : au moins l'un des points y_1 et y_2, sera une "fausse piste".

On dispose ainsi d'un moyen de connaître certaines fausses pistes d'une quelconque analyse d'une grammaire donnée : il suffit d'associer à chaque x de $g\ell_1$ tel que $\Gamma_1(x) = \{y_1, \ldots, y_n\}$ avec $n > 1$, un couple (q, k) pour lequel il existe des entiers i et j distincts, appartenant à $\{1, \ldots, n\}$ et vérifiant : $\mathcal{C}(q,k,y_i) \cap \mathcal{C}(q,k,y_j) = \emptyset$.

Il faudra cependant se contenter de couples (q, k) dont les composantes ne soient pas trop grandes de telle façon que la recherche des "fausses pistes" par une "procédure TEST" (paragraphe 5. 5. 2) ne soient pas compromise : en effet, plus l'ordre des contextes calculés sera grand et plus rares seront les occasions d'appliquer les tests de cette procédure.

CONCLUSION

L'étude précédente nous permet de tracer les grandes lignes qui conduisent à la conception d'un "système général d'analyse pour les grammaires de Chomsky". Tout d'abord, la diversité des analyses effectuées dépend des structures compatibles mises en évidence. Nous en avons étudiées trois, S_1, S_2 et S_3 au chapitre 2 ; il pourrait être intéressant d'utiliser d'autres structures et de les déterminer toutes.

Au chapitre 3, nous avons recherché à partir d'une grammaire G les quadruplets, Q = (V, g, K, h), associés à chacune des structures compatibles. Le passage de G à Q peut se faire en deux phases ;

- Si G est donnée par des "règles" il est nécessaire de réaliser une transformation qui nous donnera G sous forme de graphes (Module 1[G]),

- puis pour chaque structure S une seconde phase détermine les quadruplets Q (Module 2[S]).

Le chapitre 5 définit "l'étude contextuelle" qui accélérera les analyses d'un mot quelconque de l'alphabet terminal de G (Module 3[S, G]). Cette étude est la plus coûteuse en temps et en place. Cependant il semble nécessaire de l'approfondir car il ne faut pas oublier qu'elle se fait pour une grammaire donnée et non pas pour une analyse d'un mot donné.

En outre, si l'on peut préciser dans un sens à déterminer la qualité de cette étude en fonction d'une grammaire donnée G, on possèdera le moyen de choisir automatiquement la méthode d'analyse adéquate de l'ensemble des mots engendrés par G.

L'enchaînement des modules, (Module 1[G]→) Module 2[S]→Module 3[S, G] nous fournit ce qu'on peut appeler "un programme d'analyse" constitué par la donnée d'un quadruplet (V, g, K, h) et d'une liste de contextes.

Le chapitre 4 définit des procédures (TRAVEM, TRAVPA) de génération d'analyses, qui à partir d'un mot α sur l'alphabet terminal d'une G-grammaire G et d'un programme d'analyse associé à G engendre l'ensemble $h^{-1}(\alpha) \cap g^*(K \cap P)$ des analyses de α (Module 4[S, G, α]).

Les modules 1, 2, 3, 4, auxquels on adjoindrait un "module de choix" pourraient constituer les segments d'un système général d'analyse.

Une dernière remarque porte sur la programmation d'un tel système. Les programmes réalisés ont été écrits en Algol 60, la difficulté essentielle avec ce langage survient avec l'emploi de listes. Il serait nécessaire d'utiliser ou de définir un langage se prêtant aisément à la génération de listes ainsi qu'à leur mise à jour.

ANNEXE 1

Le problème de l'analyse syntaxique pour une grammaire, $G = (V, \rightarrow, X)$, et une structure compatible $S = (V'_1, g, K_1, h)$ est la recherche pour tout mot α de T^* $(V = T \cup N)$ des ensembles : $\mathcal{A}(\alpha) = h^{-1}(\alpha) \cap g^*(K_1 \cap P(V'_1))$.

Nous avons convenu d'effectuer des analyses avec adjonction de marqueurs :

$$V'_1 = V' \cup \# \quad , \quad \# \notin V'$$

Il s'agit dans cette annexe de préciser les procédures de définition des applications suivantes :

- l'homomorphisme alphabétique h de $(V \cup \overline{V})^*$ dans T^*
- la transcription de parenthèses g^* de $(V' \cup \overline{V}'_1)^*$ dans $(V \cup \overline{V})^*$ avec $g(\#) = g(\overline{\#}) = \wedge$.
- la transcription t, telle que $K_1 = t(L)$, L étant un langage local.

1) Structure compatible $S_3 = (V_1, I_{V_1}, K, h)$

L'application g (c'est à dire I_{V_1}) est donnée par :

- $\forall x \in V \cup \overline{V} : g(x) = x$
- $g(\#) = g(\overline{\#}) = \wedge$

D'où la procédure :

```
'INTEGER' 'PROCEDURE' G(X) ; 'INTEGER' X ;
G:='IF'(X=1) 'OR'(X=NPG1) 'THEN' 0 'ELSE' X ;
```

L'homomorphisme h de $(V \cup \overline{V})^*$ dans T^* est défini par :

$\forall A \in V$ $\quad h(\overline{A}) = SI\ A \in T\ ALORS\ A\ SINON\ \wedge$
$\quad h(A) = \wedge$

D'où la procédure :

```
'INTEGER''PROCEDURE' PETITH(X) ; 'VALUE' X ; 'INTEGER' X ;
PETITH:='IF'X > N+VOC 'THEN' X-VOC 'ELSE'0 ;
```

Le mot vide $\wedge$ est codifié par zéro.

Au chapitre 3, le langage régulier K a été défini comme le transcrit par t_3 du langage local L_3 ; le tableau TA2 associant à tout x de L_3 l'élément $t_3(x)$ de K.
La transcription t_3 est définie par la procédure :

'INTEGER''PROCEDURE' EMP(X) ; 'VALUE' X ; 'INTEGER' X ;
EMP:=TA2(/X/) ;

2) Structure compatible $S_2 = (V_1, I_{V_1}, K, h')$

L'ensemble V_1 et l'application g (c'est à dire I_{V_1}) sont identiques aux éléments de même nom qui apparaissent dans le quadruplet définissant S_3. La procédure, G(X), définie précédemment sera employée pour représenter l'application g.
L'homomorphisme h' de $(V \cup \bar{V})^*$ dans T^* est défini par :

$$\forall A \in V \cup \bar{V} \quad h(A) = \text{SI } A \in T \text{ ALORS } A \text{ SINON } \wedge$$

D'où la procédure :

'INTEGER''PROCEDURE' PETITH(X) ; 'VALUE'X ; 'INTEGER'X ;
PETITH:='IF' N+1 ≤ X < VOC 'THEN' X 'ELSE' 0 ;

Le langage régulier K est, dans ce cas, le transcrit par t_2 du langage local L_2 ; comme précédemment le tableau TA2 associe à tout x de L_2 l'élément $t_2(x)$ de K ; on utilisera donc la procédure EMP(X) définie ci-dessus pour représenter t_2.

3) Structure compatible $S_1 = (V_1', g, L_1, h')$

On a la même procédure, PETITH(X), qu'au paragraphe 2 pour définir l'homomorphisme h'.
De plus, la transcription t est l'identité sur V_1', puisque L_1 est un langage local.

D'où la procédure associée :

'INTEGER''PROCEDURE' EMP(X) ; 'VALUE'X ; 'INTEGER'X ;
EMP:=X ;

Quant à l'application g qui à tout x de $V_1' \cup \bar{V}_1'$

($V_1' = V' \cup \mathcal{C}' = V' \cup \#$ et $\bar{V}_1' = \bar{V}' \cup \mathcal{C}' = \bar{V}' \cup \bar{\#}$) associe l'élément g(x) de $V \cup \bar{V}$, elle est représentée par la procédure :

'INTEGER''PROCEDURE' G(X) ; 'VALUE'X ; 'INTEGER'X ;
G:='IF' (X-1) 'OR' (X=NPG1) 'THEN' 0 'ELSE' TA2(/X/) ;

En effet $g(\#) = g(\bar{\#}) = \wedge$ $(C_2(\#) = 1$, $C_2(\bar{\#}) = NPG1)$

ANNEXE II

Listings des programmes

```
JJOB MAP,IR07,A:LEMAIRE,1.
JALGOL SI,LS,GO
'BEGIN'
'INTEGER' VOC,VOC1,AL,NP1,NPF,NPG1,I,N;
'READ'(105,FORM)VOC1,VOC,N,AL,NPG1,NPF,NP1;
FORM:'FORMAT' (20I4);
'WRITE'(108,FMT)VOC1,VOC,N,AL,NPG1,NPF,NP1;
FMT:'FORMAT'(////' NOMBRE D ELEMENTS DE L ALPHABET DU LANGAGE REGULIER
K',10X,I5/' NOMBRE D ELEMENTS DE L ALPHABET DE LA GRAMMAIRE',16X,I5/' NO
MBRE D ELEMENTS DE L ALPHABET AUXILIAIRE DE G',64X,I5 /' NOMBRE DE LETTR
ES DE LA PHRASE A ANALYSER',21X,I5/' NOMBRE D ELEMENTS DU GRAPHE ASSOCIE
 A K',24X,I5/' NOMBRE D ELEMENTS DU TABLEAU DES FAMILLES DU GRAPHE ASSOC
IE',4X,I5/' ADRESSE DANS LE TABLEAU DES FAMILLES DE LA SOUS FAMILLE COMM
UNE',I5///);
'BEGIN'
'INTEGER''ARRAY' TA1(/1:NPG1/),TA2(/1:NPG1/),TS(/1:NPF/),ALPHA(/1:AL/);
'PROCEDURE' TEST ; 'BEGIN'     'END';
'INTEGER''PROCEDURE'PETITH(X);'VALUE'X;   'INTEGER' X ;
 PETITH:='IF'X > N*VOC 'THEN'X-VOC 'ELSE' 0 ;
'INTEGER''PROCEDURE'G(X);'VALUE' X ; 'INTEGER' X ;
G:='IF'(X=1)'OR'(X=NPG1)'THEN'0'ELSE'X;
```

PROCEDURE TRAVPA

```
'INTEGER''PROCEDURE'EMP(X);'INTEGER'X ; EMP:=TA2(/X/);
'READ'(105,FORM)'FOR'I:=1'STEP'1'UNTIL'NPG1'DO'(TA1(/I/)),
                'FOR'I:=1'STEP'1'UNTIL'NPG1'DO'(TA2(/I/)),
                'FOR'I:=1'STEP'1'UNTIL'NPF 'DO'(TS(/I/)),
                'FOR'I:=1'STEP'1'UNTIL'NPF 'DO'(TS(/I/));
'READ'(105,FORM)'FOR'I:=1'STEP'1'UNTIL'AL  'DO'(ALPHA(/I/));
'WRITE'(108,MROF)'FOR'I:=1'STEP'1'UNTIL'AL  'DO'(ALPHA(/I/));
MROF:'FORMAT'(//' PHRASE A ANALYSER :',(' ',2014));
TRAVPA
'END''END';
JLOAD (GO),(UNSAT,(ALGLIB),(PGLIB)),(MAP)
JRUN
JPMDI
JDATA
  14   7   4   8  20  26   0
 600
   1   2   3   5   6   7   8  10  12  13  14  15  16  17  18  19  20  20  20
   1   2   3   4   4   5  10  11   6   6   5   7   4   5   7   5  12  13  14
  -2  -3   4  -5  -6  -8  -7   9 -10  13 -12 -11 -14 -16 -19 -15 -17 -19 -17
   7  17   8  18  19 -20
   5   5   5   6   6   6   7   7
```

```
'PROCEDURE' TRAVPA ;
'BEGIN'
'INTEGER'Q,LV,LR,ADFEU,ADNFE,PARAM,X,X1,Y,Z,Z1,F,REE1,REE2,REE,NIL,L,
        FREE ;
'BOOLEAN'VIDE,TERM,FEU ;
'READ'(105,FMT)L ;
FMT:'FORMAT'(I4);
'BEGIN''INTEGER''ARRAY'LIS(/1:L/);
'PROCEDURE' RANGE(X,Y);'INTEGER'X,Y;
'BEGIN'
'INTEGER'I;
'PROCEDURE'PLACE ;
'BEGIN'
'IF'FREE=NIL'THEN'FREE:=NIL:=NIL+4
            'ELSE'FREE:='IF'LIS(/FREE+1/)=0'THEN'NIL'ELSE'LIS(/FREE+1/)
'END'  ;
REE:=FREE; PLACE;
'IF'X'NE'0'THEN'X:=LIS(/X+1/):=REE
          'ELSE''BEGIN'
                I:=FREE;PLACE;X:=REE;LIS(/I+1/):=Y;LIS(/I/):=REE;
                LIS(/I+3/):='IF'Y=0'THEN'I'ELSE'LIS(/Y+3/); Y:=I
                  'END'
'END';
'PROCEDURE'EXTRAC;
'BEGIN'
'INTEGER'I ;   I:=LR;
'WRITE'(108,FMT2)G(EMP(LIS(/LR/))),'FOR'I:=ABS(LIS(/I+2/))'WHILE'I>1'DO'
                  (G(EMP(LIS(/I/))));
FMT2:'FORMAT'('0','ANALYSE :',(' ',3OI3));
'END';
'PROCEDURE'LISPLA(X);'INTEGER'X;
'BEGIN'
'IF'FREE'NE'NIL'THEN'LIS(/LIS(/X+3/)+1/):=FREE;
FREE:=X
'END';
'PROCEDURE'ARRIER(X);'INTEGER' X ;
'BEGIN'
'INTEGER' H,K,M,L;
L:=X;
'FOR'H:=L'WHILE'LIS(/H+1/)=0'DO''BEGIN'
    LIS(/L+1/):='IF'FREE=NIL'THEN'0'ELSE'FREE;
    FREE:=L ;L:=ABS(LIS(/L+2/));
    'IF'L=1'THEN''GOTO'RETURN  'END' ;
'FOR'M:=LIS(/H+1/)'WHILE'M>0 'DO' H:=M ;
'IF' LIS(/-M+1/)=H'THEN'
'BEGIN''IF'H=L'THEN'LIS(/-M+1/):=0 'ELSE'LIS(/H+1/):=0;'GOTO'LA 'END';
'IF'L'NE'-H'THEN''BEGIN
                H:=-M
                'FOR'M:=LIS(/H+1/)'WHILE'M'NE'L'DO'H:=M;
                LIS(/H+1/):=LIS(/L+1/)
                  'END'
            'ELSE'LIS(/H+1/):=-LIS(/L+1/);
LA: LIS(/L+1/):='IF'FREE=NIL'THEN'0'ELSE'FREE ; FREE:=L;
RETURN:'END';
```

```
'WRITE'(108,FMT2);
FMT2 :'FORMAT'(/////30X,'********** ANALYSE AVC TRAITEMENT EN PARALLELE
**********'//41X,'************************************');
Q:=LIS(/1/);=LIS(/5/);=1;LV:=PARAM:=LIS(/8/);=5;NIL:=FALE:=9;
LIS(/2/);=LIS(/3/);=LIS(/4/);=LIS(/6/);=ADFEU:=ADNFE:=0;TERM:='FALSE' ;
LABA:  LR:=LIS(/LV/);
LA  :  Y :=TA1(/LIS(/LR/)//);REE1:=REE2:=0;VIDE:='TRUE';
ICI :  FEU:='FALSE'; X:=TS(/Y/); X1:=ABS(X);Z1:= EMP(VOC1);
Z:=LIS(/'IF'EMP(LIS(/LR/))>VOC1 'THEN'LIS(/LR+3/)'ELSE'LR/);
'IF'Z1>VOC1 'THEN''BEGIN''IF'EMP(Z)'NE'Z1-VOC1'THEN''GOTO'CONTIN'END';
F:=PETITH(G(Z1));
'IF'F'NE'0'THEN' 'BEGIN'
  'IF'TERM'OR'(F'NE'ALPHA(/Q/))'THEN''GOTO'CONTIN ;
  'IF'X1=NPG1'THEN''BEGIN''IF'TERM'THEN'EXTRAC;'GOTO'CONTIN'END';
   FEU:='TRUE'     'END'   ;
TEST ; VIDE:='FALSE';
'IF'FEU 'THEN'RANGE(REE2,ADFEU)'ELSE'RANGE(REE1,ADNFE);
LIS(/REE/);:=X1; LIS(/REE+2/);:=LR;
Z1:=EMP(LIS(/LR/));Z:=LIS(/LR+3/);
'IF'EMP(X1)<=VOC1'THEN' LIS(/REE+3/);='IF'Z1<=VOC1 'THEN'LR'ELSE'Z
                'ELSE'LIS(/REE+3/);:='IF'Z1>VOC1'THEN'LIS(/Z+3/)'ELSE'Z;

CONTIN:
'IF'X>0'THEN''ELSE'Y+1;'GOTO'ICI'END';
'IF'  VIDE 'THEN'       'BEGIN'
    'IF'LIS(/LR+2/)>0 'THEN' 'BEGIN'
                              'IF'LIS(/LR+1/)>0 'THEN''BEGIN'
                                                  F:=LR; LR:=LIS(/LR+1/);
                                                  ARRIER (F);'GOTO' LA
                                                        'END'
                                'END';
ARRIER(LR);'GOTO'ENCORE 'END';
'IF'REE1*REE2 'NE'0'THEN''BEGIN'
                  LIS(/REE1+1/);=LIS(/ADFEU/) ;
                  LIS(/REE1+2/);=-LIS(/REE1+2/);
                  LIS(/REE2+1/);=-LIS(/ADNFE/)
                        'END'
                  'ELSE''IF'REE2=0'THEN'
        LIS(/REE1+1/);='IF'LIS(/ADNFE/)=REE1'THEN'0'ELSE'-LIS(/ADNFE/)
                                'ELSE'
        LIS(/REE2+1/);='IF'LIS(/ADFEU/)=REE2'THEN'0'ELSE'-LIS(/ADFEU/);
'IF' LIS(/LR+2/)>0 'AND'LIS(/LR+1/)>0 'THEN'
              'BEGIN'LR:=LIS(/LR+1/);'GOTO'LA 'END';
ENCORE:
'IF' LIS(/LV+1/)'NE'0'THEN''BEGIN'LV:=LIS(/LV+1/);'GOTO'LABA 'END';
LISPLA(PARAM);
'IF' ADNFE'NE'0 'THEN''BEGIN'
              LV:=PARAM:=ADNFE ;
              ADNFE:=0;'GOTO'LABA
                    'END'
              'ELSE''IF'ADFEU'NE'0'THEN''BEGIN'
                          LV:=PARAM:=ADFEU; ADFEU:=0;
                         'IF'Q=AL'THEN'TERM:='TRUE''ELSE'Q:=Q+1;
                         'GOTO' LABA  'END'
'END''END';
```

```
//JOB MAPA;RO/,A:LEMAIRE,T.
//ALGOL ST,LS,GO
'BEGIN'
'INTEGER' VOC,VOC1,AL,NPT,NPF,NPG1,I,N;
'READ'(105,FORM)VOC1,VOC,N,AL,NPG1,NPF,NPT;
FORM:'FORMAT' (20I4);
'WRITE'(108,FMT)VOC1,VOC,N,AL,NPG1,NPF,NPT;
FMT:'FORMAT'(/////' NOMBRE D ELEMENTS DE L ALPHABET DU LANGAGE REGULIER
X',10X,I5/' NOMBRE D ELEMENTS DE L ALPHABET DE LA GRAMMAIRE',10X,I5/' NO
MBRE D ELEMENTS DE L ALPHABET AUXILIAIRE DE G',64X,I5 /' NOMBRE DE LETTR
ES DE LA PHRASE A ANALYSER',21X,I5/' NOMBRE D ELEMENTS DU GRAPHE ASSOCIE
 A K',24X,I5/' NOMBRE D ELEMENTS DU TABLEAU DES FAMILLES DU GRAPHE ASSOC
IE',4X,I5/' ADRESSE DANS LE TABLEAU DES FAMILLES DE LA SOUS FAMILLE COMM
UNE',I5////);
'BEGIN'
'INTEGER''ARRAY' TA1(/1:NPG1/),TA2(/1:NPG1/),TS(/1:NPF/),ALPHA(/1:AL/));
'PROCEDURE' TEST ; 'BEGIN'      'END';
'INTEGER''PROCEDURE' PETITH(X) ;'VALUE'X ; 'INTEGER' X ;
 PETITH:='IF'X > N+VOC 'THEN'X-VOC 'ELSE' 0 ;
'INTEGER''PROCEDURE'GI(X);'VALUE' X ; 'INTEGER' X ;
GI:='IF'(X=1)'OR'(X=NPG1)'THEN'0'ELSE'X;
'INTEGER''PROCEDURE'EMP(X);'INTEGER'X ; EMP:=TA2(/X/);
```

PROCEDURE TRAVEM

```
'READ'(105,FORM)'FOR'I:=1'STEP'1'UNTIL'NPG1'DO'(TA1(/I/)),
                'FOR'I:=1'STEP'1'UNTIL'NPG1'DO'(TA2(/I/)),
                'FOR'I:=1'STEP'1'UNTIL'NPF 'DO'(TS(/I/));
'READ'(105,FORM)'FOR'I:=1'STEP'1'UNTIL'AL  'DO'(ALPHA(/I/));
'WRITE'(108,PROF)'FOR'I:=1'STEP'1'UNTIL'AL  'DO'(ALPHA(/I/));
PROF:'FORMAT'(///' PHRASE A ANALYSER :',(' ',2014));
TRAVEM
'END''END';
//LOAD (GO),(UNSAT,(ALGLIB),(PALIB)),(MAP)
//GO.SYSIN DD *
//GO.FT05F001 DD *
  14   7   4  15  20  26   0
 300 300
   1   2   3   5   6   7   8  10  12  13  14  15  16  17  18  19  20  20  20
   1   2   5   4   4   5  10  11   0   6   3   7   4   5   7   5  12  13  14
  -2  -3   4  -5  -6  -6  -7   9 -10  13 -12 -11 -14 -10 -19 -15 -17 -19 -17
   7  17   8  18  19 -20
   5   5   5   5   5   5   0   6   6   6   6   6   7   7   7
//GO.FT06F001 DD *
```

```
'PROCEDURE' TRAVEM  ;
'BEGIN'
'INTEGER' P,IP,IP1,Q,Q1,LR,X,X1,Y,Z,Z1;
'BOOLEAN' TERM,TERM1;
'READ'(105,FORMA)IP,LR ;
FORMA:'FORMAT'(2I4);
'BEGIN'
'INTEGER''ARRAY' PILATT(/1:LR/),PILE(/1:IP/);
'PROCEDURE'EXTRAC;
'BEGIN'
'INTEGER' I ;
'WRITE'(108,F) 'FOR'I:=1 'STEP'1 'UNTIL' LR 'DO'
                (G(EMP(ABS(TS(/PILATT(/I/)/))))));
F : 'FORMAT'( 'O','ANALYSE  :', (' ',3013));
'END';
'WRITE'(108,FMT);
FMT: 'FORMAT'(/////30X,'**********ANALYSE PAR EMPILEMENT ET RETOUR ARRI
ERE**********'//40X,'****************************************');
IP:=0;=PILE(/1/):=1;LR:=0; TERM;=TERM1:='FALSE' ;
ICI:
Y:='IF' LR 'NE'0'THEN'TA1(/ABS(TS(/PILATT/LR/)/)/)/)'ELSE'TA1(/1/);
LA :  X:= TS(/Y/); X1:=ABS(X);
Z:=EMP(X1);
'IF' Z<=VOC1'THEN''BEGIN' IP1:=IP+1; PILE(/IP1/):=Z 'END'
          'ELSE''IF'Z=VOC1+PILE(/IP/)'THEN'IP1:=IP1-1'ELSE''GOTO'LABA;
F:=PETITH(G(Z));
'IF'F 'NE'0 'THEN' 'BEGIN'
      'IF'TERM'OR'F'NE'ALPHA(/Q/)'THEN' 'GOTO' LABA  ;
      'IF'Q=AL'THEN'TERM1:='TRUE''ELSE' Q1:=Q1+1
                  'END'
            'ELSE'Q1:=Q;
 TEST ;
'IF'TERM'AND'(X1=NPG1)'THEN''BEGIN'
                  EXTRAC ; TERM:='FALSE';
                  Q1:=AL+1 ; 'GOTO'LABA
                              'END';
 LR:=LR+1; PILATT(/LR/):=Y ; IP:=IP1; Q:=Q1 ;TERM:=TERM1;'GOTO' ICI ;
LABA :
'IF'TS(/Y/)>0'THEN''BEGIN'
                Y:='IF'TS(/Y+1/)=0'THEN'NP1'ELSE'Y+1;
                'GOTO' LA
                  'END';
'IF' LR=0 'THEN' 'GOTO'FIN ;
Z:=ABS(TS(/PILATT(/LR/)/)/);Z1:=EMP(Z);F:=PETITH(G(Z1));
'IF' Z1<=VOC1'THEN'IP:=IP+1
          'ELSE''BEGIN'IP:=IP+1;PILE(/IP/):=Z1-VOC1'END';
'IF' F 'NE' 0 'THEN''BEGIN'
        'IF'TERM.'THEN' TERM :=TERM1:='FALSE'
                  'ELSE''IF'Q1'NE'AL+1'THEN'Q:=Q-1'ELSE'Q1:=AL
                    'END';
Y:=PILATT(/LR/);LR:=LR-1;'GOTO' LABA ;
FIN: 'END''END';
```

ANNEXE III

Exemple d'une "analyse descendante prédictive"

- Soit la grammaire, $G = (N \cup T, ::=, X)$ donnée par ses règles :

$N = \{ A, C, X \}$ $\quad T = \{a, b, c\}$

$X ::= AC \mid C$

$A ::= aAb \mid ab$

$C ::= cC \mid c$

La structure compatible associée est : $S_3 = (V_1, I_{V_1}, K, h)$

Le graphe des transitions, $g\ell_1$, du langage local L_3 tel que $K = t_3(L_3)$ est donné par les tableaux TA1 et TS ; la transcription t_3 est définie par le tableau TA2 ; l'application C_2 codifie l'alphabet $N \cup T$.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TA1	1	2	3	5	6	7	8	10	12	13	14	15	16	17	18	19	20	20	20	0
TA2	1	2	9	4	4	3	10	11	6	6	3	7	4	5	7	5	12	13	14	8

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
TS	-2	-3	4	-5	-6	8	-7	9	-10	13	-12	-11	-14	-16	-19	-15	-17

18	19	20	21	22	23	24	25	26
-19	-17	3	7	17	8	18	19	-20

	#	X	A	C	a	b	c
C_2	1	2	3	4	5	6	7

N=4 VOC=7 NPG1=20 NPF=26

```
RUN
PMDI
ALGOL EXEC 17:56



NOMBRE D ELEMENTS DE L ALPHABET DU LANGAGE REGULIER K          14
NOMBRE D ELEMENTS DE L ALPHABET DE LA GRAMMAIRE                 7
NOMBRE D ELEMENTS DE L ALPHABET AUXILIAIRE DE G                 4
NOMBRE DE LETTRES DE LA PHRASE A ANALYSER                       8
NOMBRE D ELEMENTS DU GRAPHE ASSOCIE                            20
NOMBRE D ELEMENTS DU TABLEAU DES FAMILLES DU GRAPHE ASSOCIE    26
ADRESSE DANS LE TABLEAU DES FAMILLES DE LA SOUS FAMILLE COMMUNE 0



PHRASE A ANALYSER :
  5  5  5  6  6  6  7  7



                    ********** ANALYSE AVEC TRAITEMENT EN PARALLELE **********

                                ************************************

ANALYSE :
14  7 11 14  7  4 11 13 13 13 12  5  6 10 12  5  3  6 10 12  5  3  6 10  3  4  9  2
```

```
RUN
PMDI
ALGOL EXEC 17:55


NOMBRE D ELEMENTS DE L ALPHABET DU LANGAGE REGULIER K          14
NOMBRE D ELEMENTS DE L ALPHABET DE LA GRAMMAIRE                 7
NOMBRE D ELEMENTS DE L ALPHABET AUXILIAIRE DE G                 4
NOMBRE DE LETTRES DE LA PHRASE A ANALYSER                      15
NOMBRE D ELEMENTS DU GRAPHE ASSOCIE                            20
NOMBRE D ELEMENTS DU TABLEAU DES FAMILLES DU GRAPHE ASSOCIE    26
ADRESSE DANS LE TABLEAU DES FAMILLES DE LA SOUS FAMILLE COMMUNE 0


PHRASE A ANALYSER :
   5  5  5  5  5  5  6  6  6  6  6  6  7  7  7


                    ********** ANALYSE PAR EMPILEMENT ET RETOUR ARRIERE **********
                                ****************************************

ANALYSE :
 2  9  4  3 10  6  3  5 12 10  6  3  5 12 10  6  3  5 12 10  6  3  5 12 10  6  3  5 12 10

 6  5 12 13 13 13 13 13 13 11  4  7 14 11  4  7 14 11  7 14
```

Interprétation des résultats

I) Analyse avec traitement en parallèle

- La phrase à analyser est :

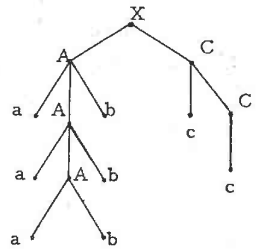
α = a a a b b b c c

- Résultat de l'analyse

X $\bar{X}$ C A $\bar{A}$ b A a $\bar{a}$ $\bar{A}$ b A a $\bar{a}$ $\bar{A}$ b A a $\bar{a}$ $\bar{A}$ b a $\bar{a}$ $\bar{b}$ $\bar{b}$ $\bar{b}$ $\bar{c}$ C c $\bar{c}$ $\bar{C}$ c $\bar{c}$

. Le résultat obtenu sur le listing est, en fait, le mot réfléchi de l'analyse du mot α.

- Ramification associée à l'analyse



II) Analyse par empilement et retour-arrière

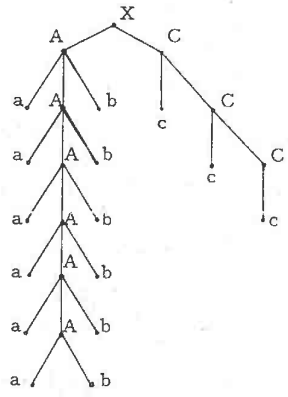
- La phrase à analyser est :

α = a a a a a a b b b b b b c c c

- Résultat de l'analyse

X $\bar{X}$ C A b A a $\bar{a}$ $\bar{A}$ b A a $\bar{a}$ $\bar{A}$ b A a $\bar{a}$ $\bar{A}$ b A a $\bar{a}$ $\bar{A}$ b A a $\bar{a}$ $\bar{A}$ b a $\bar{a}$ $\bar{b}$ $\bar{b}$ $\bar{b}$ $\bar{b}$ $\bar{b}$ $\bar{b}$ $\bar{C}$ C c $\bar{c}$ $\bar{C}$ C c $\bar{c}$ $\bar{C}$ c $\bar{c}$

- Ramification associée à l'analyse



REFERENCES

[1] C. PAIR ≪Etude de la notion de pile ; application à l'analyse syntaxique≫ (Collection des thèses de la faculté des Sciences . 1965).

[2] C. PAIR et A. QUERE ≪Définition et étude des bilangages réguliers≫. (Information and control 13 (1968) p. 565. 593)

[3] C. PAIR et A. QUERE Cours de D. E. A. : Théorie des langages (1967).

[4] C. PAIR ≪Sur des notions algébriques liées à l'analyse syntaxique≫. (Exposé fait le 10 mars 1969 au centre d'Automatique de l'Ecole des Mines, à Fontainebleau ; Revue Française d'Informatique, 1970 p. 329).

[5] B. STUDENMANN ≪Rapport de D. E. A. ≫ (Centre de Calcul Automatique de Nancy, 1969-1970).

[6] N. CHOMSKY et M. P. SCHUTZENBERGER ≪The algebric theory of context-free languages. Computer programming and formal systems≫ (Braffort. Hirschberg, editors North Holland Amsterdam . 1963, p. 188. 161 .

[7] S. GINSBURG ≪The mathematical theory of context-free languages≫ (Mac Graw. Hill, New-York, 1966).

[8] C. BERGE ≪Théorie des graphes et ses applications≫ (Dunod, Paris 1967).

[9] J. LOECKX ≪An algorithm for le construction of bonnded-context parsers≫ (Communication ACM 13 may 1970).

[10] R. W. FLOYD ≪Bonnded-context syntactic analysis≫ (Communication ACM 7 february 1964 p. 62. 67).

[11] D. E. KNUTH ≪On the translation of languages form left to night≫ (Information and Control, décembre 1965).

[12] D. E. KNUTH ≪Top down syntax analysis≫. International Summer School on Computer Programming, Copenhague (1967).

NOM DE L'ETUDIANT : LEMAIRE Claude

Nature de la thèse : Doctorat de Spécialité en Mathématiques Appliquées

Vu, Approuvé

et permis d'imprimer

NANCY, le 2 octobre 1971

Le Président du Conseil de l'Université de NANCY I

J.R. HELLUY