RAIRO. INFORMATIQUE THÉORIQUE

M. P. FRANCHI-ZANNETTACCI

# Un algorithme de calcul formel des séries énumératrices de langage linéaire

*RAIRO. Informatique théorique*, tome 17, nº 4 (1983), p. 343-364
<http://www.numdam.org/item?id=ITA_1983__17_4_343_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 17, n° 4, 1983, p. 343 à 364)

# UN ALGORITHME DE CALCUL FORMEL DES SÉRIES ÉNUMÉRATRICES DE LANGAGE LINÉAIRE (*)

par M. P. FRANCHI-ZANNETTACCI [1]

Communiqué par J.-F. PERROT

Résumé. — *Le travail présenté ici, se situe dans le cadre d'une contribution à l'étude statistique de la configuration des protéines, mollécules biologiques codées par des mots.*

*Nous proposons un algorithme général, permettant d'obtenir ce nombre de mots chaque fois que la configuration recherchée est descriptible par un langage linéaire.*

*A partir d'une grammaire non ambiguë G, cet algorithme génère la série énumératrice du langage engendré par G et fournit donc une formule calculant le nombre de mots recherchés.*

Abstract. — *This work is a contribution to the statistical study of proteins which are biological mollecules coded in their primary structure by a word.*

*We propose a general algorithm giving the number of such structures in the case of constraints coded by a linear language.*

*Starting with an unambiguous linear grammar G, this algorithm produce the generating function of the language generated by G and thus a formula for computing the number of words according to the given structure.*

## 1. INTRODUCTION

Les problèmes qui aboutissent à la mise à jour de formules élégantes, sont ceux qui intéressent le plus les combinatoristes.

Toutefois dans de nombreux problèmes pratiques, on trouve des formules qui satisfont peu ce critère esthétique, tant dans leur expression, que dans la manière de les obtenir.

(*) Reçu en janvier 1981, révisé en mai 1983.

[1] Université de Bordeaux-II, Département d'Informatique appliquée, 146, rue Léo-Saignat, 33076 Bordeaux Cedex.

Pour éviter la répétition de calculs fastidieux, il est parfois souhaitable d'effectuer le calcul des formules énumératrices par programme. Il en est ainsi de l'énumération de langages rationnels, directement applicables à l'étude statistique de certaines cellules biologiques, telles les protéines et les $t$-RNA objets de nombreuses recherches en biologie [3, 5, 12].

Dans cet article, nous nous intéressons à la génération d'une série énumératrice d'un langage rationnel à partir d'une de ses grammaires non ambiguë.

L'algorithme que nous décrivons fournit une *formule*, qui donne les coefficients de la série en faisant intervenir des sommes et produits de coefficients multinomiaux. Pour exemple, choisissons le langage $L$ défini par sa grammaire non ambiguë :

$$G=\langle \{L\}, \{x, y\}, R, \langle L \rangle \rangle,$$

avec :

$$\mathscr{R} \begin{cases} \langle L \rangle ::= xy \langle L \rangle, \\ \langle L \rangle ::= yx \langle L \rangle, \\ \langle L \rangle ::= \varepsilon. \end{cases}$$

On a ainsi l'équation :

$$L=(xy+yx)L+\varepsilon, \tag{1.1}$$

d'où :

$$L=(xy+yx)^*. \tag{1.2}$$

On a donc en passant à l'image commutative :

$$l=\sum_{n \geqq 0} 2^n x^n y^n. \tag{1.3}$$

L'algorithme produit à partir de la grammaire $G$ la formule $2^n$.

Pour automatiser le processus de calcul, nous utilisons la méthode des attributs sémantiques [8] à partir d'une syntaxe abstraite [11] permettant de représenter les expressions régulières. Il s'agit d'un processus de compilation facilité par l'utilisation d'un attribut *hérité*, qui permet de différencier les expressions régulières construites à partir de l'opération étoile des autres.

De manière classique, tous les objets manipulés reçoivent une représentation d'arbre.

Le calcul se déroule de la manière suivante :

— interprétation des règles de grammaire afin d'obtenir une représentation des équations [dans l'exemple précédent on obtient une représentation de l'équation (1.1) à partir de $G$] ;

— obtention de l'arbre représentant l'expression régulière solution [passage de l'équation (1.1) à (1.2)];

— calcul de la formule d'énumération en deux passes de traduction (dans l'exemple par la première phase on obtient $n$, par la deuxième phase on obtient $2^n$);

— édition sous forme intelligible par le système réalisé par J. Hardouin-Duparc [7].

L'article comporte cinq parties, nous commençons par des rappels concernant les objets sur lesquels nous travaillons.

Dans la deuxième partie, nous définissons une fonction de traduction, permettant l'obtention d'une première forme pour la série à partir d'une expression régulière.

Dans la troisième partie, nous exposons un algorithme réalisant une forme agréable de cette série énumératrice.

Dans la quatrième partie, nous décrivons la mise en œuvre, en utilisant la méthode des attributs.

Dans la cinquième partie, nous montrons comment s'effectue le calcul de l'expression régulière à partir d'une grammaire étendue.

On trouvera :

— en annexe I : le système complet des attributs,

— en annexe II : quelques exemples d'exécution du programme Pascal réalisant les algorithmes.

## 2. DÉFINITIONS ET NOTATIONS

### 2.1. Notations

On note $\varepsilon$ le mot vide.

Soit $w \in X^*$ où $X$ est un alphabet, soit $x \in X$, nous notons $|w|_x$ le nombre d'occurrences de la lettre $x$ dans $w$.

Nous noterons $\begin{pmatrix} n \\ k_1, k_2, \ldots, k_r \end{pmatrix}$ le multinomial $\dfrac{n!}{k_1!, \ldots, k_r!}$.

### 2.2. Définitions

DÉFINITION 1 [2] : Soit $L$, un langage sur un alphabet $X = \{x_1, x_2, \ldots, x_r\}$ la série énumératrice $l$ du langage $L$ est la série de $\mathbb{N}[\![X]\!]$ donnée par :

$$l = \sum_{(i_1, i_2, \ldots, i_r)} N(i_1, i_2, \ldots, i_r) x_1^{i_1} x_2^{i_2} \ldots x_r^{i_r},$$

où $N(i_1, i_2, \ldots, i_r)$ est le nombre de mots $w$ de $L$, tels que :

$$j \in [1, r], \qquad |w|_{x_j} = i_j.$$

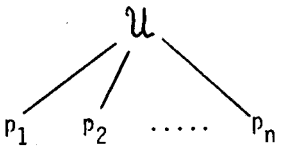
DÉFINITION 2 [1] : Une expression régulière sur un alphabet $X$ est définit à partir des expressions suivantes :

(i) $\emptyset$, $\varepsilon$, $a\,(a \in X)$ sont des expressions régulières.

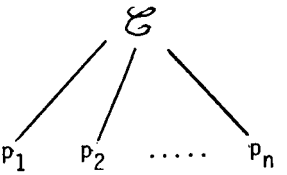
(ii) si $p$ et $q$ sont des expressions régulières (représentant des langages rationnels $P$ et $Q$), alors $(p+q)$, $(p.q)$, $p^*$ sont des expressions régulières représentant respectivement $P \cup Q$, $P.Q$ et $P^*$.

Nous associerons à ces expressions régulières une représentation arborescente.

$(p_1 + p_2 + \ldots + p_n)$ sera représenté :




$(p_1 . p_2, \ldots, p_n)$ sera représenté :




$(p)^*$ sera représenté :


$\mathcal{U}$ représente l'union, $\mathcal{C}$ la concaténation, $\mathcal{E}$ l'étoile.

## 2.3. Principe de l'algorithme

Notre cadre de travail étant les langages rationnels, le calcul d'une série énumératrice peut s'effectuer à partir d'une expression régulière solution d'une grammaire non ambiguë (voir exemple fourni en introduction).

Notre algorithme utilise le principe que nous avons énoncé.

Par une opération de traduction, nous associons à une expression régulière représentant un langage rationnel $L$, une série énumératrice de $L$.

Une expression régulière sur un alphabet $X$ est un mot d'un langage $R$ écrit sur l'alphabet.

$$Z = X \cup \{(,), *, +, .\}.$$

Nous lui faisons correspondre, par traduction, un mot écrit sur :

$$X \cup \{K_i\}_{i \geqq 0} \cup \{\Sigma, +, *, (,), \mathfrak{C}, =, 1, -\}$$

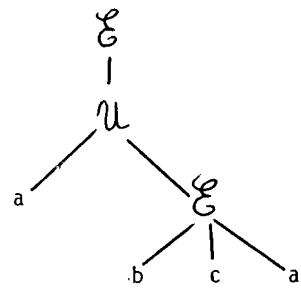
Un binomial sera représenté par $\mathfrak{C}^{e_1}_{e_2}$, où $e_1$ et $e_2$ sont deux expressions dépendantes des $\{K_i\}_{i \geqq 0}$.

Un multinomial sera représenté par $\begin{pmatrix} e_0 \\ e_1, e_2, \ldots, e_k \end{pmatrix}$ où $\forall j \in [0, k]$, les $e_j$ sont des expressions dépendantes des $\{K_i\}_{i \geqq 0}$.

Ce type de traitement relève de manière évidente des techniques de compilations.

A tout mot de $R$, nous associons une représentation intermédiaire construite à partir de la représentation arborescente donnée précédemment (§2.2) (Abstract Syntax [10]).

*Exemple* 1.2: Soit $(a+b.c.a)^*$ une expression régulière, on lui associe :




## 2.4. Choix de la méthode de traduction

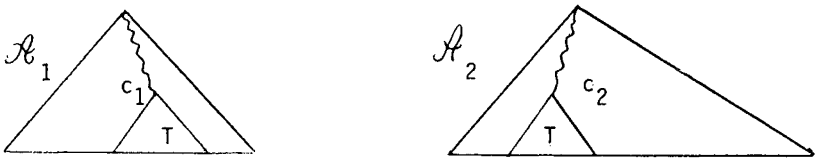
Plusieurs techniques sont envisageables, en particulier, la traduction dirigée par la syntaxe [1], la méthode des attributs [8]. La première méthode n'est pas utilisable; en effet, la traduction d'une expression régulière ne peut être effectuée indépendamment de son contexte, comme le montre l'exemple suivant.

*Exemple* 2.1 : $(b+c)$ se traduit $b+c$;

$(b+c)^*$ se traduit $\sum_{n \geqq 0} \sum_{m \geqq 0} \binom{n}{n+m} b^n c^m$.

La traduction de $b+c$ dépend de la mise à l'étoile de l'expression.

En d'autres termes, si $\mathscr{A}_1$ et $\mathscr{A}_2$ sont des représentations de deux expressions régulières, et si $T$ est un sous-arbre commun à $\mathscr{A}_1$ et $\mathscr{A}_2$ :




la traduction de $T$ dans $\mathscr{A}_i$ dépend du chemin $c_i$ (elle dépend en particulier, de l'existence d'un opérateur $*$ dans $c_i$).

Pour ces raisons, il nous a paru préférable d'effectuer la traduction en utilisant la méthode des attributs. Ceci permet de prendre en compte facilement, tout le contexte d'un sous arbre quelconque, en utilisant un attribut hérité. Nous reviendrons sur l'utilisation de cette méthode au paragraphe 5.

## 3. TRADUCTION D'UNE EXPRESSION RÉGULIÈRE

DÉFINITION 3 : Soit $\mathrm{Trad}^n$, l'application définie récursivement sur les expressions régulières, sur un alphabet $X$, à image dans le monoïde d'alphabet :

$$X \cup \{X_i\}_{i \geqq 0} \cup \{\Sigma, +, (, ), >, =, 1, -\},$$

telle que $n \in \{k_i\}_{i \geqq 0} \cup \{1\}$;

$\mathrm{Trad}^n(A)$ est la traduction de $A^n$ et vérifie :

(1) $\mathrm{Trad}^n(\varepsilon) = 1$;

(2) $\mathrm{Trad}^n(a) =$ si $n=1$, alors $a$, sinon $a^n$;

(3) $\mathrm{Trad}^n(e^*) =$ si $n=1$, alors $\sum_{k_1 \geqq 0} \mathrm{Trad}^{k_1}(e)$;

sinon $\sum_{k_1 \geqq 0} C_{k_1+n-1}^{k_1} \mathrm{Trad}^{k_1}(e)$;

(4) $\mathrm{Trad}^n(e_1+e_2+\ldots+e_r) =$ si $n=1$,
alors :

$\mathrm{Trad}^n(e_1)+\mathrm{Trad}^n(e_2)+\ldots+\mathrm{Trad}^n(e_r)$;

sinon :

$$\sum_{k_1+k_2+\ldots+k_r=n} \binom{n}{k_1, k_2, \ldots, k_r}$$

$$\mathrm{Trad}^{k_1}(e_1) \ldots \mathrm{Trad}^{k_r}(e_r);$$

(5) $\mathrm{Trad}^n(e_1.e_2 \ldots e_r) = \mathrm{Trad}^n(e_1) \ldots \mathrm{Trad}^n(e_r)$,

avec la convention suivante : tous les $k_i$ intervenant dans $\mathrm{Trad}^n(A)$ sont distincts.

La traduction d'une expression régulière $e$, sera le résultat de $\mathrm{Trad}^1(e)$.

*Exemple* 3.1 : Soit l'expression régulière sur $a$, $b$, $c$ : $e = (a+b.c.a)^*$.

$$\mathrm{Trad}^1(e) = \sum_{K_1 \geqq 0} \mathrm{Trad}^{K_1}(a+b.c.a)$$

$$= \sum_{K_1 \geqq 0} \sum_{K_2+K_3=K_1} \binom{K_1}{K_2, K_3} \mathrm{Trad}^{K_2}(a)\, \mathrm{Trad}^{K_3}(b.c.a)$$

$$= \sum_{K_1 \geqq 0} \sum_{K_2+K_3=K_1} \binom{K_1}{K_2, K_3} a^{K_2}\, \mathrm{Trad}^{K_3}(b)\, \mathrm{Trad}^{K_3}(c)\, \mathrm{Trad}^{K_3}(a)$$

$$= \sum_{K_1 \geqq 0} \sum_{K_2+K_3=K_1} \binom{K_1}{K_2, K_3} a^{K_2} b^{K_3} c^{K_3} a^{K_3},$$

REMARQUE 1 : Pour la mise en œuvre de l'algorithme par la méthode des attributs, « l'exposant » de Trad sera communiqué par un attribut hérité.

REMARQUE 2 : Nous pouvons remarquer sur cet exemple, que la forme proposée n'est pas tout à fait satisfaisante, il serait « plus agréable » d'obtenir $\mathrm{Trad}^1(e)$ sous la forme :

$$\sum_{\substack{i_1 \geqq 0 \\ i_2 \geqq 0 \\ i_3 \geqq 0}} N(i_1, i_2, i_3)\, a^{i_1} b^{i_2} c^{i_3}.$$

Pour obtenir ce résultat, il faut regrouper toutes les occurences d'une même lettre dans les différents monomes de la traduction.

*Exemple* 3.2 : Dans l'exemple précédent, on doit :

— regrouper $a^{K_2}$ et $a^{K_3}$ en : $a^{K_2+K_3}$ ;

— substituer $N_a - K_3$ à $K_2$ dans la traduction ;

— supprimer le signe $\Sigma$ portant sur $K_2$;

— ajouter un signe $\Sigma$ portant sur $N_a$.

Les mêmes opérations doivent être effectuées sur $b$ et $c$, pour obtenir :

$$\sum_{N_a \geqq 0} \sum_{N_b \geqq 0} \binom{N_a}{N_a - N_b,\ N_b} a^{N_a} b^{N_b} c^{N_b}.$$

Dans le paragraphe suivant, nous donnons un algorithme permettant l'obtention de cette forme.

## 4. OBTENTION D'UNE FORME AGRÉABLE DE LA SÉRIE ÉNUMÉRATRICE

### 4.1. Propriété des exposants d'une variable

Soient :

$e$ une expression régulière;

$a$ une variable intervenant dans un monome de Trad$^1$ $(e)$;

$A(a)$ l'expression arithmétique de l'exposant de cette variable dans le monôme.

On peut montrer par induction que $A(a)$ est une fonction linéaire des $K_i$.

$$A(a) = \alpha_0 + \alpha_1 K_{N_1} + \alpha_2 K_{N_2} + \ldots + \alpha_r K_{N_r}.$$

Ainsi cette fonction permet de remplacer $K_{N_1}$ par :

$$(1/\alpha_1)(N_a - (\alpha_0 + \alpha_2 K_{N_2} + \ldots + \alpha_1 K_{N_r}))$$

L'algorithme que nous utilisons pour effectuer cette opération manipule des expressions arithmétiques représentées sous forme arborescente. Nous notons $\mathscr{A}(a)$, la représentation de $A(a)$. Cette représentation peut être construite par la fonction Trad$^n$ $(a)$, en remplaçant la règle (2) par :

Trad$^n$ $(a)$ = si $\mathscr{A}(a)$ est vide,

alors :

$$\mathscr{A}(a) := .n$$

sinon :

$$\mathscr{A}(a) := \begin{array}{ccc} & + & \\ \mathscr{A}(a) & & n \end{array}$$

*Exemple* 4.1 : Dans l'exemple du paragraphe 3, on a ainsi :

$\mathrm{Trad}^{K_2}(a)$ produit $\mathscr{A}(a) := . K_2$

$\mathrm{Trad}^{K_3}(a)$ produit $\mathscr{A}\, a) := \begin{array}{ccc} & + & \\ K_2 \diagup & & \diagdown K_3 \end{array}$

$\mathrm{Trad}^1(e)$ est alors un texte ne contenant pas de variables. De plus, son calcul produit la représentation de l'exposant des variables. On peut alors donner un algorithme réalisant une forme habituelle de la série.

## 4.2. Algorithme de traduction

*début*

*faire* pour tout $a \in X$ de représentation $\mathscr{A}(a)$

*début*

— rechercher dans $\mathscr{A}(a)$ le premier indice $K_{j_0}$;

— *si* $K_{j_0}$ existe *faire :*

*début*

— effectuer formellement l'expression arithmétique de représentation $A(a)$, en prenant les valeurs pour $K_j$ suivantes :

(1) pour tout $j$, $K_j = 0$ on trouve $r_1$,

(2) pour tout $j \neq j_0$, $K_j = 0$ et $K_{j_0} = 1$ on trouve $r_2$;

—$\alpha_{j_0} := r_2 - r_1$;

— supprimer dans $\mathscr{A}(a)$ les monômes dépendants de $K_{j_0}$ en remplaçant $K_{j_0}$ par 0,

résultat $\mathscr{A}'(a)$;

— la représentation de $K_{j_0}$ est :

$$\mathscr{A}(K_{j_0}) := \text{si } \alpha_{j_0} \neq 1 \qquad \text{et} \qquad \mathscr{A}'(a) \neq 0.$$

alors $\begin{array}{ccc} & / & \\ & - & \alpha_{j_0} \\ N_a \diagup & & \diagdown \mathscr{A}'(a) \end{array}$

sinon si :

$\alpha_{j_0} \neq 1$ et $\mathscr{A}'(a) = .\ 0$ alors $/$ ($N_a$, $\alpha_{j_0}$),

sinon si :

$\alpha_{j_0} = 1$ et $\mathscr{A}'(a) \neq .\ 0$ alors $-$ ($N_a$, $\mathscr{A}'(a)$),

sinon :

— $\mathscr{A}(a) := .N_a$;

— $\forall x \in X$, $X \neq a$ substituer dans $\mathscr{A}(x)$, $\mathscr{A}(K_{j_0})$ aux feuilles étiquetées $K_{j_0}$;

— $\forall r$, substituer dans $\mathscr{A}(K_r)$, $\mathscr{A}(K_{j_0})$ aux feuilles étiquettées $K_{j_0}$

*fin*

*fin*

— supprimer les signes $\Sigma$ portant sur des indices ayant une représentation;

— dans $\mathrm{Trad}^1(e)$ remplacer tous les indices par leur représentation (si elle existe);

— pour tout $a \in X$ *faire*

*début*

si $\mathscr{A}(a) = .N_a$, alors écrire $\sum_{N_a \geqq 0}$ devant $\mathrm{Trad}^1(e)$;

si $\mathscr{A}(a) \neq \emptyset$, alors écrire $a^{\mathscr{A}(a)}$ à la suite de $\mathrm{Trad}^1(e)$.

*fin*

*fin*

*Exemple* 4.2 : L'exemple du paragraphe 3 fournit le déroulement suivant, par $\mathrm{Trad}^1(e)$, on produit :

$\mathscr{A}(a) = +$ ($K_2$, $K_3$) $\quad \mathscr{A}(b) = .K_3 \quad \mathscr{A}(c) = .K_3$;

*pour a :*

$j_0 = 2, \quad r_1 = 0, \quad r_2 = 1, \quad \alpha_{j_0} = 1,$

$\mathscr{A}'(a) = +$ ($0$, $K_3$)

d'où :

$$\mathscr{A}(K_2) = \begin{array}{c} - \\ N_a \quad K_3 \end{array} \qquad \text{et} \qquad \mathscr{A}(a) = .\,N_a;$$

*pour b :*

$$j_0 = 3, \qquad r_1 = 0, \qquad r_2 = 1, \qquad \alpha_{j_0} = 1,$$

$\mathscr{A}'(b) = .\,0$

d'où :

$$\mathscr{A}(K_3) = .\,N_b \qquad \text{donc} \qquad \mathscr{A}(K_2) = \begin{array}{c} - \\ N_a \quad N_b \end{array},$$

$$\mathscr{A}(b) = .\,N_b, \qquad \mathscr{A}(c) = .\,N_b,$$

*pour c*, $j_0$ n'existe pas.

Le remplacement de $K_2$ et de $K_3$ dans $\mathrm{Trad}^1(e)$ donne successivement :

$$\sum_{K_1 \geqq 0} \sum_{N_a - K_3 + K_3 = K_1} \binom{K_1}{N_a - K_3,\ K_3},$$

*remplacer* $K_3$ *dans* $\mathrm{Trad}^1(e)$ :

$$\sum_{K_1 \geqq 0} \binom{K_1}{N_a - N_b,\ N_b}.$$

$K_1$ est construit et après ajout des signes somme, on obtient :

$$\sum_{N_a \geqq 0} \sum_{N_b \geqq 0} \binom{N_a}{N_a - N_b,\ N_b} a^{N_a} b^{N_b} c^{N_b}.$$

La réalisation de la traduction directement à partir de ces algorithmes est laborieuse. Aussi, nous avons utilisé la méthode des attributs. Nous effectuons le travail en deux étapes :

— 1re étape : calcul des indices et des exposants;

— 2e étape : écriture de la traduction.

## 5. RÉALISATION PAR LA MÉTHODE DES ATTRIBUTS SÉMANTIQUES

La méthode des attributs introduite par D. E. Knuth [8] est une technique fréquemment utilisée dans les problèmes de compilation.

Nous ne la redéfinissons pas ici, nous renvoyons les lecteurs aux exposés de Livercy [10], et aux travaux de Lewis et Stearns [9].

### 5.1. Principe

Les attributs sont des fonctions définies pour chaque règle d'une grammaire permettant d'associer une traduction à un arbre de dérivation engendré par cette grammaire. On distingue deux types d'attributs :

— les attributs *synthétisés* permettant de transférer les informations des feuilles de l'arbre vers la racine;

— les attributs *hérités* permettant de définir les informations circulant de la racine vers les feuilles.

Dans ce travail, les attributs doivent être définis sur une grammaire des expressions régulières.

Or dans le cas d'arbre de syntaxe abstraite, on effectue une composition des attributs décrits sur la grammaire.

Nous les définissons donc directement sur les arbres de syntaxe abstraite donnés au paragraphe 2.

Nous ne donnons, dans ce paragraphe, qu'une partie du système d'attributs, il se trouve dans son intégralité en annexe I.

Le calcul se déroule en deux étapes :

(1) calcul des exposants;

(2) production de la traduction.

### 5.2. Attributs utilisés

La fonction $\mathrm{Trad}^n$ dépend de l'exposant $n$, nous avons donc deux attributs :

* TRAD : fournit la traduction (synthétisé).

* EXP : fournit l'exposant $n$ (hérité).

En fait, EXP est lui-même un arbre comme nous l'avons vu au paragraphe 4.

Deux attributs permettent d'assurer que les $K_j$ sont différents, et donnent donc la valeur $j$ maximale des $K_i$ utilisés (ainsi $K_{j+1}$ sera disponible) :

* KD : attribut hérité donne cette information en descendant dans l'arbre représentant l'expression régulière.

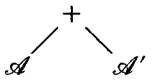
* KM : attribut synthétisé donne cette même information en montant.

### 5.3. 1re étape : Calcul des exposants

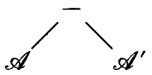
L'attribut TRAD n'est pas utilisé; en effet, l'écriture d'une série énumératrice s'effectuera lors de la deuxième étape.

Nous utilisons trois fonctions agissant sur la représentation d'un exposant :

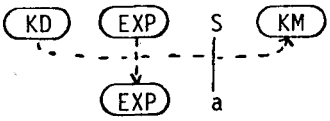
* CREE($K$) : crée un sommet d'arbre étiqueté $K$.
* PLUS ($\mathscr{A}'$, $\mathscr{A}$) : ajoute $\mathscr{A}'$ à l'arbre $\mathscr{A}$ pour obtenir :




* MOINS ($\mathscr{A}'$, $\mathscr{A}$) : produit :



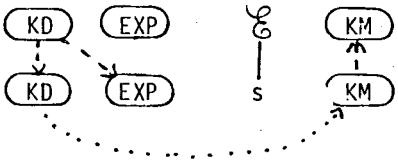

Le calcul pour une feuille de l'arbre est le suivant :




$$\mathrm{KM}(s) := \mathrm{KD}(s).$$
$$\mathrm{EXP}(a) := \mathrm{PLUS}\,(\mathscr{A}(a),\ \mathrm{EXP}(s)).$$

Ceci signifie que l'on ajoute l'exposant provenant de $S$ à l'ancien exposant de $a$.

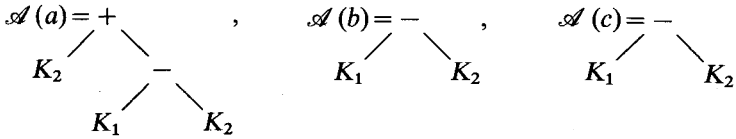
Pour un sous-arbre de sommet $\mathscr{E}$ (représentant l'opération étoile) on doit utiliser un nouvel indice qui devient l'exposant du fils de $\mathscr{E}$ et incrémenter KD, ce qui donne :




$$\mathrm{KD}(s) := \mathrm{KD}(\mathscr{E}) + 1;$$
$$\mathrm{EXP}(s) := \mathrm{CREE}(\mathrm{KD}(\mathscr{E}));$$
$$\mathrm{KM}(\mathscr{E}) := \mathrm{KM}(s).$$

Cette première étape produit ainsi une représentation de l'exposant des variables.

*Exemple* 5.1 : Reprenons l'exemple du paragraphe 4, $(a+b.c.a)^*$, après cette première étape, on a :



Après avoir effectué l'algorithme du paragraphe 4 :

$$\mathscr{A}(a) = .\,N_a, \qquad \mathscr{A}(b) = .\,N_b, \qquad \mathscr{A}(c) = .\,N_b,$$

$$\mathscr{A}(K_1) = .\,N_a, \qquad \mathscr{A}(K_2) = \underset{N_a \quad N_b}{-}, \qquad \mathscr{A}(K_3) = .\,N_b.$$

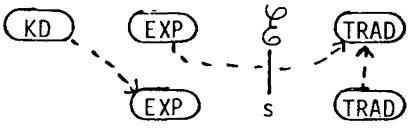
### 5.4. 2e étape : Production de la traduction

Les attributs KD et KM sont inchangés, il suffit donc de définir EXP et TRAD.

Soit $A$, un arbre représentant une expression arithmétique, la fonction REP($A$) calcule la chaîne alphanumérique correspondant à $A$.

La fonction $\mathscr{A}$(i) renvoie l'arbre correspondant à l'indice $K_i$. Ainsi, l'attribut EXP est un arbre provenant de la première étape :

– pour un sous-arbre de racine $\mathscr{E}$ (opération étoile), on doit écrire un signe somme, un binomial et modifier l'exposant.




* EXP($s$) = si, $\mathscr{A}$(KD($\mathscr{E}$)) = $\emptyset$ alors CREE(KD($\mathscr{E}$))
  sinon $\mathscr{A}$(KD($\mathscr{E}$))

* TRAD($\mathscr{E}$) = si $\mathscr{A}$(KD($\mathscr{E}$)) = $\emptyset$ alors début.
  $R$ := CREE(KD($\mathscr{E}$))
  écrire :

$$\sum_{\mathrm{REP}(R) \geqq 0}$$

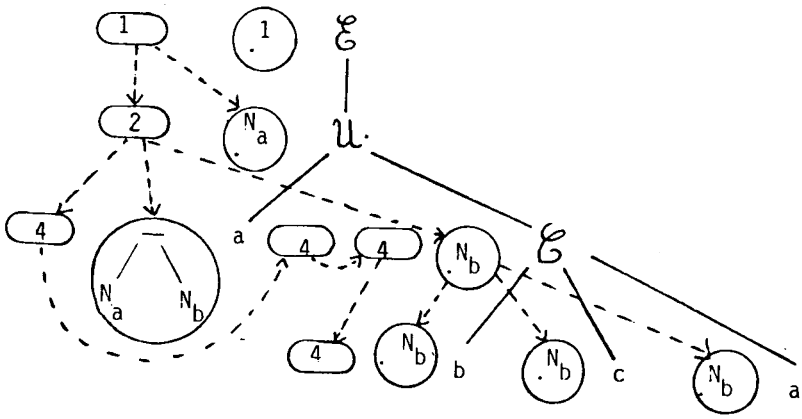
fin
  sinon $R$ := $\mathscr{A}$(KD($\mathscr{E}$));
si EXP ($\mathscr{E}$) $\neq 1$
alors écrire :

$$\begin{pmatrix} \mathrm{REP}(R) \\ \mathrm{REP}(R) + \mathrm{REP}(\mathrm{EXP}(\mathscr{E})) - 1 \end{pmatrix};$$

TRAD($s$).

On peut ainsi pour chaque sous-arbre donner une traduction associée (*voir* annexe I).

*Exemple* 5.2 : Dans l'exemple précédent, on a la figure suivante, où l'on montre la circulation des attributs EXP et KD.




et on produit la traduction comme suit :

$$\sum_{N_a \geqq 0} \sum_{N_b \geqq 0} \mathrm{TRAD}(\mathscr{E})\, a^{N_a} b^{N_b} c^{N_b};$$

$$\sum_{N_a \geqq 0} \sum_{N_b \leqq 0} \mathrm{TRAD}(\mathscr{U})\, a^{N_a} b^{N_b} c^{N_b};$$

$$\sum_{N_a \geqq 0} \sum_{N_b \geqq 0} \binom{N_a}{N_a - N_b,\ N_b} \mathrm{TRAD}(a)\, \mathrm{TRAD}(\mathscr{E})\, a^{N_a} b^{N_b} c^{N_b};$$

$$\sum_{N_a \geqq 0} \sum_{N_b \geqq 0} \binom{N_a}{N_a - N_b,\ N_b} a^{N_a} b^{N_b} c^{N_b}.$$

## 6. EXPRESSION RÉGULIÈRE ASSOCIÉE A UNE GRAMMAIRE

A toute grammaire rationnelle $G$, on peut associer le langage rationnel $\mathscr{L}(G)$ dont une représentation possible est une expression régulière le caractérisant.

Or, à toute grammaire linéaire correspond par passage à l'image commutative, une grammaire rationnelle (peut-être ambiguë), donc une expression régulière en image commutative.

Cette notion peut être étendue à une classe plus vaste que nous définissons comme suit :

DÉFINITION 6.1 : Une grammaire $G = \langle V_T, V_N, \mathcal{R}, A \rangle$ où $A \in V_N$, est dite linéaire étendue si toutes ses règles de la forme :

$$\langle B \rangle ::= e_1 \langle C \rangle e_2 \qquad \text{ou} \qquad \langle B \rangle ::= e_1,$$

avec $e_1$ et $e_2$ expressions régulières sur $V_T$ et $B, C \in V_N$.

REMARQUE : Une grammaire rationnelle ou linéaire peut être transformée en une linéaire étendue. Cette transformation permet de simplifier la traduction de l'expression régulière associée.

Nous avons donc développé des algorithmes permettant le passage de grammaires linéaires étendues à la représentation arborescente d'une expression régulière associée en image commutative [6].

Ainsi en utilisant les algorithmes de traduction, nous pouvons produire une série énumératrice pour les langages associés.

Pour une grammaire linéaire étendue $G = \langle V_T, V_N, \mathcal{R}, A \rangle$ on effectue l'analyse syntaxique des règles de $G$.

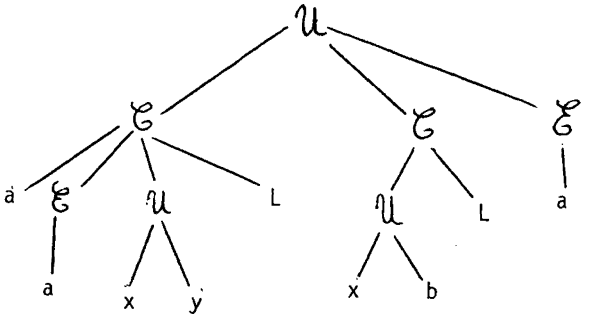
Par contraction de l'arbre de dérivation, on produit un arbre de syntaxe abstraite pour chaque non terminal. Cet arbre représente l'ensemble des règles qui lui sont associés.

C'est en fait la représentation arborescente d'une expression régulière écrite sur l'alphabet $V_T \cup V_N$.

*Exemple* 6.1 : Soit $G = \langle \{a, b, x, y\}, \{L\}, \mathcal{R}, \langle L \rangle$ avec :

$$\mathcal{R} \begin{cases} \langle L \rangle ::= aa^*(x+y)\langle L \rangle, \\ \langle L \rangle ::= (x+b)\langle L \rangle, \\ \langle L \rangle ::= a^*. \end{cases}$$

$\langle L \rangle$ est alors associé à la représentation.



qui est la représentation arborescente de l'expression régulière :

$$aa^*(x+y)L+(x+b)L+a^*.$$

Le calcul de l'expression régulière associée à $\mathscr{L}(G)$ s'effectue sur cette représentation par un algorithme très voisin de celui que l'on utilise quand $G$ est rationnelle. Il procède par substitution de sous-arbre. Les modifications permettent dans de nombreux cas de simplifier la formule produite ultérieurement. Tous ces algorithmes sont programmés en Pascal 80, nous donnons des exemples d'exécution en annexe II.

## ANNEXE I

### COMPLÉMENTS SUR LE SYSTÈME D'ATTRIBUTS

**1re étape**

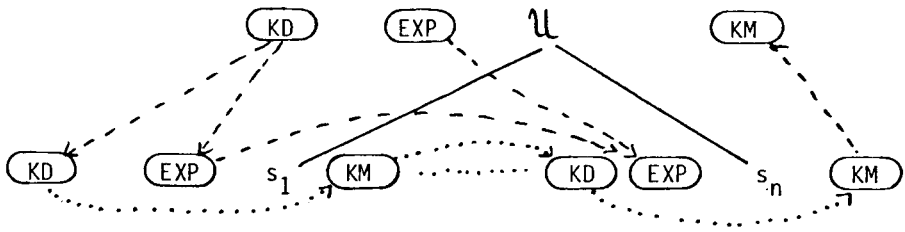
Les attributs associés aux sous-arbres de racine $\mathscr{E}$ ou de feuille $a$, ont été donnés au paragraphe 5.3.

*Sous arbre de racine* $\mathscr{U}$ (opération Union)

On utilise l'égalité suivante :

$$(e_1+e_2+\ldots+e_l)^m=\sum_{k_1+\ldots+k_l=m}\binom{m}{k_1,\ldots,k_l}e_1^{k_1}\ldots e_l^{k_l},$$

On figure la circulation des attributs comme suit :



— $\mathrm{KD}(s_1)=\mathrm{KD}(\mathscr{U})+n$;

— $\forall i \in [1, n-1]\ \mathrm{EXP}(s_i)=\mathrm{CREE}(\mathrm{KD}(\mathscr{U})+i-1)$;

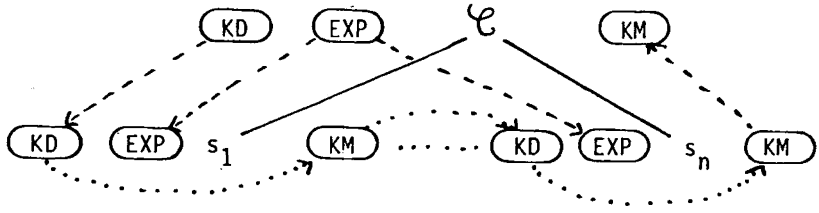
— $\mathrm{EXP}(s_n)=\mathrm{MOINS}(\mathrm{PLUS}(\mathrm{EXP}(s_1),$
$\mathrm{PLUS}\ \mathrm{EXP}(s_2, \ldots, \mathrm{EXP}(s_{n-1}))), \mathrm{EXP}(\mathscr{U}))$;
$\{\mathscr{A}(\mathrm{KD}(\mathscr{U})+n-1) := \mathrm{EXP}(s_n)\}$;

— $\mathrm{KM}(\mathscr{U})=\mathrm{KM}(s_n)$;

— $\forall i \in [2, n]\ \mathrm{KD}(s_i)=\mathrm{KM}(s_{i-1})$

*Sous-arbre de racine* $\mathscr{C}$ (opération Concaténation)

Dans ce cas, les informations sont transférées sans modification.




— $\mathrm{KD}(s_1)=\mathrm{KD}(\mathscr{C})$;

— $\forall i \in [2, n],\quad \mathrm{KD}(s_i)=\mathrm{KD}(s_{i-1})$;

— $\mathrm{KM}(\mathscr{C})=\mathrm{KM}(s_n)$;

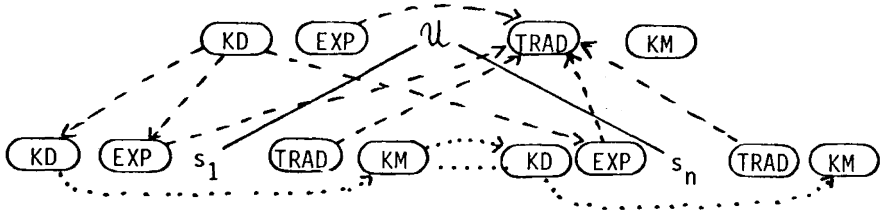
— $\forall i \in [1, n],\quad \mathrm{EXP}(s_i)=\mathrm{EXP}(\mathscr{C})$.

**2e étape**

Au paragraphe 5.4, nous avons donné les attributs pour un sous-arbre de racine $\mathscr{E}$.

*Sous-arbre de racine* $\mathscr{U}$

On a la représentation :



— KD et KM sont définis dans la première étape.

— $\forall i \in [1, n]$, $\mathrm{EXP}(s_i) = \text{si } \mathscr{A}(\mathrm{KD}(\mathscr{U}) + i - 1) = \emptyset$,

alors :

$$\mathrm{CREE}(\mathrm{KD}(\mathscr{U}) + i - 1) \quad \text{sinon } \mathscr{A}(\mathrm{KD}(\mathscr{U}) + i - 1).$$

— $\mathrm{TRAD}(\mathscr{U}) = \text{si } \exists\, s_i / \mathscr{A}(\mathrm{KD}(\mathscr{U}) + i - 1) = \emptyset$,

alors écrire :

$$\sum_{\mathrm{REP}(\mathrm{EXP}(s_1)) + \ldots + \mathrm{REP}(\mathrm{EXP}(s_n)) = \mathrm{REP}(\mathrm{EXP}(\mathscr{U}))} ;$$

écrire :

$$\begin{pmatrix} \mathrm{REP}(\mathrm{EXP}(\mathscr{U})) \\ \mathrm{REP}(\mathrm{EXP}(s_1)), \ldots, \mathrm{REP}(\mathrm{EXP}(s_n)) \end{pmatrix};$$

$$\mathrm{TRAD}(s_1);\ \mathrm{TRAD}(s_2);\ \ldots;\ \mathrm{TRAD}(s_n);$$

*Sous-arbre de racine $\mathscr{C}$*

De même que dans la première étape, il n'y a pas de modification, on a donc :

— KD et KM, même définition ;

— $\forall i \in [1, n]$, $\mathrm{EXP}(s_i) = \mathrm{EXP}(\mathscr{C})$ ;

— $\mathrm{TRAD}(\mathscr{C}) = \mathrm{TRAD}(s_1)$ ; ... ; $\mathrm{TRAD}(s_n)$.

*Sous-arbre de feuille terminale*

Aucune production n'est faite.

## ANNEXE II

### EXEMPLES D'EXÉCUTION

Les exemples 1 et 2, montrent des exécutions sur des grammaires formelles quelconques.

Les exemples 3 et 4 sont appliqués aux protéines. Il s'agit d'étudier la distribution théorique du nombre de paires adjacentes d'acides aminés.

Dans l'exemple 3, la grammaire est entrée sous une forme non réduite, la série apparaît comme somme de deux séries.

Dans l'exemple 4, le même langage est décrit par une grammaire étendue. Ceci permet de produire une série de forme plus « agréable ».

*Exemple* 1 : Grammaire :

$$G=\langle\{A, B\}, \{L\}, R, L\rangle$$

avec :

$$R\begin{cases}\langle L\rangle ::= AB\langle L\rangle,\\ \langle L\rangle ::= BA\langle L\rangle,\\ \langle L\rangle ::= 1,\end{cases}$$

$$\sum_{n_a\geq 0}\ \sum_{k_1\geq 0}\ \binom{n_a}{k_1, n_a-k_1}\ a^{n_a}b^{n_a}$$

$$\begin{array}{ccc} & * & \\ * & & * \end{array}$$

*Exemple* 2 : Grammaire :

$$G=\langle\{A, B\}, \{L\}, R, L\rangle$$

avec :

$$R\begin{cases}\langle L\rangle ::= 2AB\langle L\rangle;\\ \langle L\rangle ::= 1.\end{cases}$$

$$\sum_{n_a\geq 0}\ 2^{n_a}a^{n_a}b^{n_a}$$

$$\begin{array}{ccc} & * & \\ * & & * \end{array}$$

*Exemple* 3 : Grammaire :

$$G=\langle\{A, B, X, Y\}, \{L, M\}, R, L\rangle;$$

avec :

$$R \begin{cases} \langle L \rangle ::= A\langle M \rangle; \quad \langle M \rangle ::= A\langle M \rangle ; \\ \langle L \rangle ::= (X+B)\langle L \rangle ; \quad \langle M \rangle ::= (X+Y)\langle L \rangle ; \\ \langle L \rangle ::= 1, \quad \langle M \rangle ::= 1. \end{cases}$$

$$\sum_{n_x \geq 0} \sum_{n_y \geq 0} \sum_{n_a \geq 0} \sum_{n_b \geq 0} \sum_{k_4 \geq 0}$$

$$\binom{n_b+n_x+n_y}{k_4+n_y-n_a+n_x, n_b, n_a-k_4} \left[ \begin{matrix} k_4 \\ n_a-1 \end{matrix} \right. \binom{n_a-k_4}{n_a-k_4-n_y, n_y}$$

$$x^{n_x} y^{n_y} a^{n_a} b^{n_b}$$

$$* \qquad * \qquad *$$

$$\sum_{n_x \geq 0} \sum_{n_y \geq 0} \sum_{n_a \geq 0} \sum_{n_b \geq 0} \sum_{k_4 \geq 0}$$

$$\sum_{k_7 \geq 0} \binom{n_b+n_x+n_y}{1+k_7+k_4+n_y-n_a+n_x, n_b, n_a-1-k_7-k_4}$$

$$\left[ \begin{matrix} k_4 \\ n_a-2-k_7 \end{matrix} \right. \binom{n_a-1-k_7-k_4}{n_a-1-k_7-k_4-n_y, n_y}$$

$$x^{n_x} y^{n_y} a^{n_a} b^{n_b}$$

$$* \qquad * \qquad *$$

*Exemple* 4 : Grammaire :

$$G = \langle \{A, B, X, Y\}, \{L\}, R, L \rangle;$$

avec :

$$R \begin{cases} \langle L \rangle ::= AA^* \ (X+B)\langle L \rangle \ ; \\ \langle L \rangle ::= (X+B)\langle L \rangle \ ; \\ \langle L \rangle ::= A^* \ . \end{cases}$$

$$\sum_{n_x \geq 0} \sum_{n_y \geq 0} \sum_{n_a \geq 0} \sum_{n_b \geq 0} \sum_{k_4 \geq 0}$$

$$\sum_{k_7 \geq 0} \binom{n_x+n_y+n_b}{n_a-k_7-k_4, k_7-n_a+k_4+n_x+n_y, n_b} \binom{k_4}{n_a-1-k_7}$$

$$\binom{n_a-k_7-k_4}{n_a-k_7-k_4-n_y, n_y} \ {}_x{}^{n_x} {}_y{}^{n_y} {}_a{}^{n_a} {}_b{}^{n_b}$$

$$* \qquad *$$

$$*$$

## BIBLIOGRAPHIE


1. A. V. Aho et J. D. Ullman, The Theory of Parsing, Translation and Compiling, p. 103-124, Prentice Hall, 1972.
2. J. Brestel et coll., *Séries formelles en variables non commutatives*, Actes de la cinquième école de printemps d'informatique théorique, Vieux-Boucau-les-Bains, 1976, L.I.T.P., 1978.
3. R. J. Cedergreen, B. Larue, D. Sankoff, G. Lapalme et H. Grosjean, *Convergence and Minimal Mutation Criteria for Evaluating Early Events in t-RNA Evolution*, Proc. Nat., Acad., Sc., U.S.A., vol. 77, n° 5, 1980, p. 2791-2795.
4. N. Chomsky et M. P. Schutzenberger, *The Algebric Theory of Context free Languages*, North Holland, Amsterdam, 1963.
5. D. Debouverie et J. Schlusselberg, *Research of Specific Contraints in Pair Association of Amino-acids*, Biochimie, 56, 1974, p. 1045-1051.
6. M. P. Franchi-Zannettacci, *Étude statistique des enchaînements d'acides aminés par des méthodes combinatoires et l'utilisation du calcul formel*, Thèse de 3e cycle, Bordeaux, 1980.
7. J. Hardouin Duparc, *Une méthode pour la résolution de certains problèmes combinatoires*, Thèse d'état, Bordeaux, 1980.
8. D. E. Knuth, *Examples of Formal Semantics*, Lecture Notes in Mathematics, n° 188, Springer Verlag, 1971.
9. P. M. Lewis, D. J. Rosenckrantz et R. E. Stearns, *Attributed Translations*, A.C.M. Symposium on Theory of Computing, 1973, p. 160-171.
10. Livercy, *Theorie des programmes, schémas, preuves, sémantique*, Dunos, Paris, 1978.
11. P. Lucas et K. Walk, *On the Formal Definition of PL/1*, Annual review in automatic programming, 1969, p. 105-182.
12. S. Tanaka et H. A. Sheraga, *Statistical Treatment of Protein Conformation*, Macromolecules, vol. 9, n° 5, 1976, p. 812-833.