910 Langages Algébriques

Wolper
Carton

Hors Cat (...) Alg (...)

Introduction : Les langages algébriques sont l'une des classes de la hiérarchie de Chomsky et sont utiles dans l'analyse syntaxique des langages de programmation, notamment dans la reconnaissance des blocs parenthésés

I Grammaires algébriques. Définitions et propriétés

[C] p76

1) Premières définitions

Déf : * Une grammaire algébrique est un quadruplet $G=(\Sigma, V, P, S)$ où :
- Σ est un alphabet de terminaux fini
- V est un ensemble fini de variables
- $P \subseteq V \times (\Sigma+V)^*$ un ensemble fini de règles de production
- $S \in V$ un axiome

* On dit que $u \in (\Sigma+V)^*$ se dérive en $v \in (\Sigma+V)^*$ si $u = xXy$, $v = x\alpha y$ et $(X,\alpha) \in P$, noté $u \to v$
* On dit que $w \in \Sigma^*$ est engendré par G si on peut l'obtenir par une suite finie de dérivations à partir de S
* le langage engendré par G est l'ensemble des mots engendrés par G, noté $L(G)$

* Un langage est dit algébrique s'il est engendré par une grammaire algébrique

Exemple : Soit $G=(\Sigma, V, P, S)$ la grammaire telle que :
$V=\{S,A,B\}$, $\Sigma=\{a,b\}$ $P=\{S\to A, S\to B, A\to aA, B\to bB, A\to\varepsilon, B\to\varepsilon\}$

$L(G)=\{a^n, n\geq 0\} \cup \{b^n, n\geq 0\}$

par exemple $aaa \in L(G)$:
$S\to A\to aA\to aaA\to aaaA\to aaa$

Lemme fondamental : Soit $G=(\Sigma,V,P,S)$ une grammaire algébrique, $u \in (\Sigma+V)^*$, $v\in(\Sigma+V)^*$.
On suppose que u se factorise en $u_1u_2 = u$

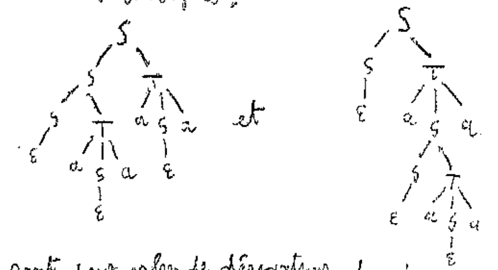
Alors : $u \xrightarrow{k} v \iff u_1 \xrightarrow{k_1} v_1$, $u_2 \xrightarrow{k_2} v_2$ $v=v_1v_2$ et $k_1+k_2=k$

2) Arbres de dérivation, ambiguïté

[C] p90

Déf : Soit $G=(\Sigma, V, P, S)$ une grammaire algébrique
* Un arbre de dérivation est un arbre fini étiqueté par $\Sigma \cup V \cup \{\varepsilon\}$ tel que si A est l'étiquette d'un noeud et $a_1, \dots a_n$ celles de ses fils, alors $(A, a_1, \dots a_n) \in P$
* la frontière d'un arbre de dérivation est le mot obtenu en concaténant les étiquettes des feuilles de gauche à droite
* une grammaire est dite ambiguë s'il existe un mot ayant au moins deux arbres de dérivation de racine étiquetés par S

Exemple : La grammaire définie par $\{S\to ST+\varepsilon, T\to aSa\}$ est ambiguë :



sont deux arbres de dérivations de a^4

3) Diverses formes de Grammaire

[C] p79

- Forme réduite : Une grammaire (Σ, V, P, S) est dite réduite si $\forall A\in V$, $L_G(A)\neq\emptyset$ et $\forall A \in V$, $\exists u,v \in (\Sigma+V)^*$ $S \xrightarrow{*} uAv$

Propriété : Pour toute grammaire, il existe une grammaire réduite engendrant le même langage.

- Forme propre : Une grammaire $G=(\Sigma,V,P,S)$ est dite propre si P ne contient pas de règles $A\to\varepsilon$ ou $A\to B$, $A,B\in V$

⚠ gram ≠ gram alg

preuves constructives (algos)

Forme normale de Greibach

FNG dérivat $S \to w$ et forcément de $|w|$ en ($n = |w|$)

Propriété : Pour toute grammaire G, il existe une grammaire propre G' telle que $L(G') = L(G) / \{\varepsilon\}$

• Forme normale de Chomsky : Une grammaire est sous forme normale de Chomsky si toutes les règles sont de la forme : $A \to A_1 A_2$, $A_1, A_2 \in V$ ou $A \to a$, $a \in \Sigma$

Propriété : Pour toute grammaire G, il existe une grammaire sous forme de Chomsky G' telle que $L(G') = L(G) \setminus \{\varepsilon\}$

Exemple : La grammaire G définie par $S \to aSb + \varepsilon$ est réduite,
La forme propre associée est $S \to aSb + ab$
et sa forme de Chomsky $\left\{ \begin{array}{ll} S \to AS_1 + AB & A \to a \\ S_1 \to SB & B \to b \end{array} \right\}$

[C] p92

II Propriétés des langages algébriques

1) Lemme d'Ogden

Lemme : Pour tout langage algébrique L, il existe $K \in \mathbb{N}$ tel que pour tout $f \in L$ ayant au moins K lettres distinguées, f se factorise en $f = \alpha u \beta v \gamma$ avec :

- $\forall n \in \mathbb{N}$ $\alpha u^n \beta v^n \gamma \in L$
- au moins α, u, β ou β, v, γ contiennent des lettres distinguées
- $u \beta v$ contient moins de K lettres distinguées

Application : $\{a^n b^n c^n, n \geq 0\}$ n'est pas algébrique

[C] p97

2) Propriétés de clôture

• **Proposition** : la classe des langages algébriques est close par union, concaténation et étoile

• **Proposition** : l'intersection de deux langages algébriques et le complémentaire d'un langage algébrique ne sont pas algébriques [C] p94

Contre exemple pour l'intersection :
$\{a^n b^n c^n, n \geq 0\} = \{a^n b^n c^m, n, m \geq 0\} \cap \{a^m b^n c^n, n, m \geq 0\}$

3) Problèmes décidables et indécidables

• **Proposition** : • Il existe un algorithme permettant de vérifier si un langage algébrique est vide
• Il existe un algorithme permettant de vérifier si un mot appartient à un langage algébrique
c'est l'Algorithme de Cocke Younger, Kasami $\boxed{\text{DVPT}}$

• **Proposition** : Les problèmes suivants sont indécidables : [C] p.153
- deux langages algébriques sont-ils égaux ?
- l'intersection de deux langages algébriques est-elle vide ?
- un langage algébrique est-il universel ?

Preuve par réduction du Problème de Correspondance de Post

4) Théorème de Parikh

[C] p86

Théorème : Tout langage algébrique est commutativement équivalent à un langage rationnel $\boxed{\text{DVPT}}$

III Automates à pile

[C] p 104

1) **Définition** : * Un automate à pile à reconnaissance par état final est un objet de la forme $(Q, \Sigma, \Gamma, A, q_0, z_0, F)$ où :
- Q ensemble fini d'états
- Σ un alphabet d'entrée
- Γ un alphabet de pile
- $F \subset Q$ des états finaux
- $q_0 \in Q$ un état initial
- $z_0 \in \Gamma$ un symbole de pile initial
- A un ensemble de transitions de la forme $(q, z) \xrightarrow{y} (q', z')$, inclus dans $Q \times \Gamma \times \Sigma \times Q \times \Gamma^*$

* Un mot w est accepté s'il existe une suite de transitions lisant w issue de q_0 et arrivant dans un état de F.

* Le langage reconnu par l'automate est l'ensemble des mots acceptés

[C] p112

Théorème : Un langage est algébrique si et seulement si il existe un automate à pile reconnaissant L.

Application : l'intersection d'un langage algébrique et d'un langage rationnel est un langage algébrique.

2) Langages déterministes

Définition : * Un automate à pile est déterministe si : pour tout $(q, \gamma) \in Q \times \Gamma$, ou bien il existe une unique transition de la forme $(q, \gamma) \xrightarrow{\varepsilon} (q', \gamma')$ et aucune permettant de lire une lettre de Σ, ou bien il n'existe pas d'ε-transition sur (q, γ) et pour tout $a \in \Sigma$, il existe au plus une transition de la forme $(q, \gamma) \xrightarrow{a} (q', \gamma')$

Proposition : le complémentaire d'un langage déterministe est déterministe

Proposition : • le problème de l'universalité est décidable pour les langages algébriques déterministes.
• le problème de l'égalité est décidable pour les langages algébriques déterministes

IV Analyse syntaxique

Objectif : étant donné un mot w et une grammaire G, on veut trouver si possible une dérivation de w dans G.

p113 [AUT]

1) Analyse descendante LL(1)

Principe : On va construire une dérivation gauche en regardant le premier non terminal de la dérivation en cours et la prochaine lettre à produire pour choisir la règle de production à appliquer

Exemple : $w = acbbcb$, $G = \left\{ \begin{array}{l} S \xrightarrow{r_1} aSbS,\ S \xrightarrow{r_2} cT \\ T \xrightarrow{r_3} aTbc\ \ T \xrightarrow{r_4} b \end{array} \right\}$

$S,\ acbbcb \xrightarrow[r_1]{} aSbS,\ cbbcb \xrightarrow[r_2]{} acTbS,\ bbcb \xrightarrow[r_4]{} acbbS,\ cb \xrightarrow[r_2]{} acbbcT,\ b \xrightarrow[r_4]{} acbbcb,\ \emptyset$

Définition : Une grammaire est dite $LL(1)$ si il est possible de déterminer quelle règle doit être appliquée au non-terminal le plus à gauche d'une dérivation gauche par la connaissance de ce non terminal et de la première lettre restant à lire

2) Analyse ascendante $LR(0)$

Principe : On va construire une dérivation droite en lisant w de gauche à droite et en réduisant le mot si l'on reconnaît un membre droit d'une règle de production

Exemple : $w = acdb$ $G = \{ S \xrightarrow[r_1]{} aSb,\ S \xrightarrow[r_2]{} cT,\ T \xrightarrow[r_3]{} d \}$

$\emptyset \xrightarrow[\text{lecture}]{} a \xrightarrow[\text{lec.}]{} ac \xrightarrow[\text{lec}]{} ac\underline{d} \xrightarrow[r_3]{} a\underline{cT} \xrightarrow[r_2]{} aS \xrightarrow[\text{lec.}]{} aSb \xrightarrow[r_1]{} S$

Définition : Une grammaire est dite $LR(0)$ si on sait, à chaque instant, s'il faut effectuer une lecture ou une réduction, et quelle réduction

Références :

[C] : Olivier Carton, Langages formels, Calculabilité et Complexité

[Aut] : J-M Autebert Théorie des langages et des Automates

Développement : Théorème de Parikh.

Déf: Deux mots w et w' sont anagrammes si $w = a_1 \dots a_n$ et il existe σ une permutation de $\{1, \dots, n\}$ tellque $w' = a_{\sigma(1)} \dots a_{\sigma(n)}$

L'ensemble des anagramme de w est noté $\overline{w}$.

On appel image commutative d'un langage L, notée $\overline{L}$, l'ensemble $\{\overline{w}, w \in L\}$.

Pour deux langages L et M, on dit qu'ils sont commutativement équivalent si $\overline{L} = \overline{M}$.

Th: (Parikh) Tout langage algébrique L est commutativement équivalent à un langage rationnel R.

Pour prouver cela on va s'appuyer sur beaucoup d'outils et de lemmes intermédiaires.

Tout d'abord on définit <u>les opérations de substitution</u>

pour une grammaire $G = (A, V, P)$, $V = \{X_1, \dots X_n\}$ et un n-uplet de langages $L = (L_1, \dots L_n)$ on définit par induction :

- $\varepsilon(L) = \{\varepsilon\}$; $a(L) = \{a\}$ pour $a \in A$; $X_i(L) = L_i$

  $xy(L) = x(L)y(L)$ pour $x, y \in (A+V)^*$; $K(L) = \bigcup_{w \in K} w(L)$ pour $K \subseteq (A+V)^*$

On introduit ensuite <u>le système d'équations associé à une grammaire.</u>

déf: $G = (A, V, P)$, $V = (X_1, \dots X_n)$ on a alors le système $S(G)$ formé des n équations

$$L_i = \sum_{X_i \to w} w(L) \qquad \text{pour } i \leq n$$

On note alors L_G le n-uplet $(L_G(X_1), \dots L_G(X_n))$

et on a une proposition "évidente"

prop : $\forall w \in (A+V)^*$ on a $L_G(w) = w(L_G)$.

preuve par induction sur les substitutions.

Ensuite on se donne le lemme :

lemme : $G = (A, V, P)$ et L solution de $S(G)$.

Si $w, w' \in (A+V)^*$ vérifient $w \xrightarrow{*} w'$ alors $w(L) \supset w'(L)$

preuve : Il suffit de démontrer le résultat pour une dérivation et le lemme s'en déduit.

si $w = u X_i v \longrightarrow u \gamma v = w'$, $X_i \rightarrow \gamma$ règle de la grammaire

L est solution de $S(G)$ donc $L_i = \sum_{X_i \rightarrow \delta} \delta(L)$ et donc $L_i \supset \gamma(L)$

d'où $w(L) \supset w'(L)$

de ce lemme on déduit :

prop : $G = (A, V, P)$, $V = \{X_1, \dots X_n\}$ alors L_G est la solution minimale de $S(G)$ pour l'inclusion.

preuve : – on vérifie que c'est bien une solution :

$$L_G(X_i) = \sum_{X_i \rightarrow \alpha} L_G(\alpha) = \sum_{X_i \rightarrow \alpha} \alpha(L_G) \quad \text{d'après la proposition précédente}$$

– la minimalité découle elle du lemme précédent.

On se penche ensuite sur le cas des grammaires propres !

i.e. une grammaire dont tous les L_i ne contiennent pas ε.

prop : G propre alors L_G est l'unique solution propre de $S(G)$

preuve : G propre donc L_G propre et d'après la prop. précédente c'est une solution de $S(G)$.

On introduit la relation sur les langages $\overset{l}{\sim}$ avec $l \in \mathbb{N}$

telle que $K \overset{l}{\sim} K'$ si $\{w \in K, |w| \leq l\} = \{w \in K', |w| \leq l\}$.

On étend cette relation aux n-uplets, composante par composante.

Soient alors L et L' deux solutions propres de G.

On va montrer que $L \overset{l}{\simeq} L'$ pour tout l et donc $L = L'$.

par récurrence, pour $l=0$, L et L' sont propres donc on a le résultat.

maintenant supposons $L \overset{l}{\simeq} L'$ pour $l \geq 0$ et montrons $L \overset{l+1}{\simeq} L'$ c'est-à-dire

$$L_i \overset{l+1}{\simeq} L'_i \quad \text{pour } 1 \leq i \leq n.$$

On introduit $S_i = \{ w \mid X_i \to w \}$, alors par définition $L_i = S_i(L)$

On veut donc montrer $S_i(L) \overset{l+1}{\simeq} S_i(L')$.

Soit donc $1 \leq i \leq n$ fixé et $w \in S_i$, $w = w_1 \dots w_p$. $w_j \in (A+V)$

$p \neq 0$ car la grammaire est propre.

Soit $u \in w(L)$ de longueur inférieure à $l+1$. alors $u = u_1 \dots u_p$

avec chaque u_j étant soit une lettre de A soit, si $w_j = X_k$ est un mot dans L_k.

de plus $|u_j| \leq l$ sinon $p=1$ ce qui voudrait dire que $X_i \to X_k \in P$

impossible car la grammaire est propre. Donc par hypothèse $u_j \in L'_k$

et donc $u \in w(L')$ d'où $L_i \subset L'_i$, l'inclusion inverse se montre

par symétrie des rôles. et on déduit le résultat. ■

On définit alors le <u>système d'équation en commutatif</u>.

cela part que si $\bar{u} = \bar{v}$ on a aisément $\overline{u(L)} = \overline{v(L)}$ et par

extension aux langages si $\bar{T} = \bar{K}$, $\overline{T(L)} = \overline{K(L)}$ on peut donc

définir de manière cohérente $\bar{S}(G)$ comme

$$\overline{L_i} = \sum_{X_i \to w} \overline{w(L)} \quad \text{pour } 1 \leq i \leq n$$

prop: $\bar{L}_G$ est l'unique solution propre de $\bar{S}(G)$ si la grammaire est propre.

la preuve est la même que la précédente.

Et on a enfin :

Th : pour $G = (A, V, P)$ grammaire telle que $\{w, (X, w) \in P\}$ est rationnel, alors il existe $R_1, \ldots R_n$ langages rationnels tels que $(\overline{R_1}, \ldots \overline{R_n})$ soient solution de $\overline{S(G)}$.

lemme : K langage sur $A + \{X\}$ et L et M deux langages sur A alors $\overline{K(L^*M)L^*} = \overline{K(M)L^*}$

preuve : pour chaque $u \in K$ on regroupe les occurences de X à la fin, $\overline{u} = \overline{w X^n}$ avec w mot sur A. Alors on vérifie facilement $\overline{u(L^*M)L^*} = \overline{u(M)L^*}$ et on somme ces égalités pour avoir le résultat.

preuve : Par récurrence sur le nombre de variables de la grammaire n.

si $G = (A, \{X\}, P)$ on note les règles $S \to S(X)$

ou $S(X)$ est une expression rationnelle sur $(A + \{X\})^*$

Une solution de $\overline{S(G)}$ est un langage L tel que $\overline{L} = \overline{S(L)}$

On dans $S(X)$ les mots n'ayant occurence de X des autres :

$P(X) = S(X) \cap (A + \{X\})^* X (A + \{X\})^*$

$T = S(X) \cap A^*$

les langages rationnels sont clos par intersection donc T et $P(X)$ son rationnels. ensuite modulo la commutativité on peut écrire : $\overline{P(X)} = \overline{Q(X) X}$ ou $Q(X)$ peut être choisi rationnel en prenant les mots de $P(X)$ ou on a supprimé la première occurence de X.

Alors $\overline{S(G)}$ s'écrit $\overline{L} = \overline{Q(L) L + T}$

on vérifie que le langage rationnel $R = Q(T)^* T$ est solution de $\overline{S(G)}$

$$\begin{aligned}
\overline{Q(R) R + T} &= \overline{Q(Q(T)^* T) \, Q(T)^* T + T} \\
&= \overline{Q(T) Q(T)^* T + T} \\
&= \overline{Q(T)^* T} \qquad \text{grâce au lemme.} \\
&= \overline{R}
\end{aligned}$$

on suppose maintenant si notre grammaire a $n+1$ variables $X_0, X_1, \dots X_n$ et que l'on a le résultat pour les nombres inférieurs.

Les règles de G s'écrivent

$$X_i \Rightarrow S_i(X_0, \dots X_n) \quad \text{pour } 0 \leq i \leq n.$$

où chaque S_i est rationnel sur $A+X$ avec $X = \{X_0, \dots X_n\}$.

soit $X' = X \setminus \{X_0\}$, on introduit la grammaire G_0 en considérant $X_1, \dots X_n$ comme des lettres terminales et on ne prenant que les règles associées à X_0 dans G. i.e

$$G_0 = (A+X', \{X_0\}, \{X_0 \Rightarrow S_0(X_0, X')\})$$

qui d'après le résultat précédent on a $R_0(X')$ solution rationnel du système $S(G_0)$

donc $\quad \overline{R_0(X')} = S_0(R_0(X'), X')$

soit alors la grammaire G' obtenue en remplaçant X_0 dans les autres règles par $R_0(X')$ donc

$$G' = (A, X', \{X_i \Rightarrow S_i(R_0(X'), X') \mid 1 \leq i \leq n\})$$

la grammaire G' vérifie les hypothèse de récurrence donc il existe n langages rationnels $R_1, \dots R_n$ sur A tel que $R' = (R_1, \dots R_n)$ soit solution du système $S(G')$ donc $\overline{R_i} = S_i(R_0(R'), R') \quad 1 \leq i \leq n$

Il reste à vérifié $R = (R_0(R'), R_1, \dots R_n)$ est solution de $S(G)$.

En substituant X' par R' dans l'équation $\overline{R_0(X')} = S_0(R_0(X'), X')$ on obtient une équation complétant les n équations $\overline{R_i} = S_i(R_0(R'), R')$ et ceci termine la preuve. ▨

On prouve enfin le théorème de Parikh :

soit L algébrique, $G = (A, V, P)$, $L = L_G(X_1)$

L_G est l'unique solution de $S(G)$ quitte à se ramener à une grammaire propre.

On a aussi d'après le th précédent une solution rationnelle. d'où le résultat par unicité.

Si au départ L n'est pas propre on applique le résultat à $L \setminus \{\varepsilon\}$.

CQFD

ref : Carton p. 83 à 90

Carton / Hopcroft.

Dev : algorithme CYK.

Cocke, Kasami et Younger.

Prend en entrée un mot et une grammaire en forme normale quadratique G et décide en temps cubique si w est engendré par G.

Il utilise la programmation dynamique.

$G = (A, V, P)$ $\qquad V = \{S_0, \ldots S_{m-1}\}$

Soit $w = a_1 \ldots a_n$ alors pour tous entiers $1 \leq i \leq j \leq n$ on note $w[i,j]$ le facteur $a_i \ldots a_j$ de w. On note alors $E_{i,j}$ l'ensemble des variables $S \in V$ telles que $w[i,j] \in L_G(S)$. L'algorithme calcul tous les ensembles $E_{i,j}$.

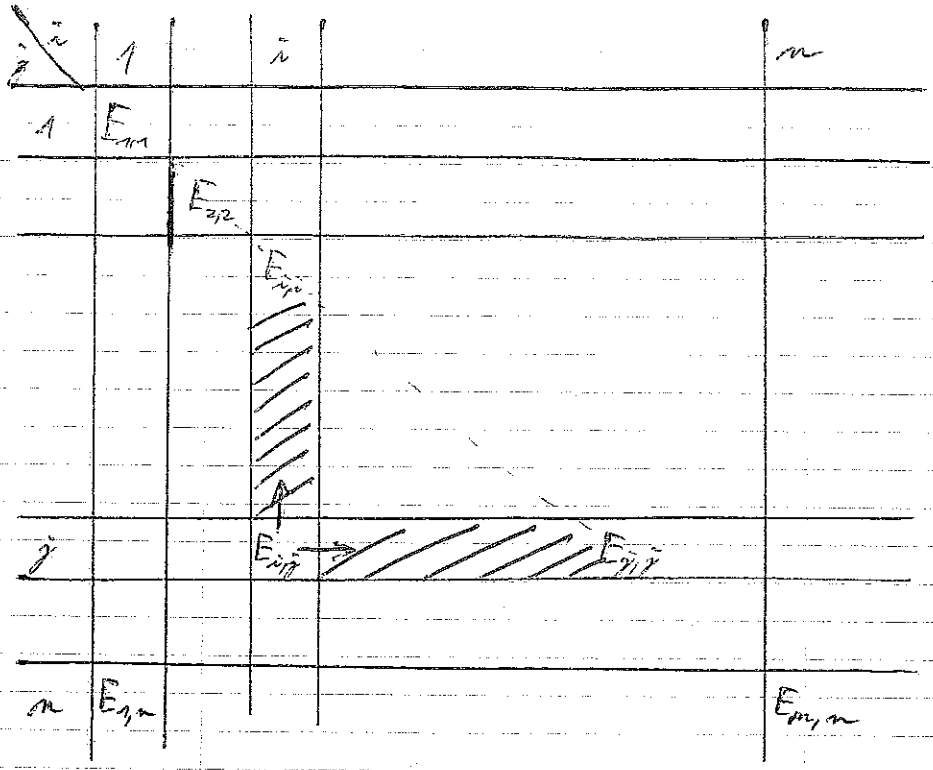
Le mot w appartient alors à $L_G(S_0)$ si $S_0 \in E_{1,n}$.

L'initialisation se fait sur $i = j$. alors $w[i,j]$ est réduit à la lettre a_i. Comme la grammaire est sous forme normale quadratique, $S \in E_{ii}$ si et seulement si $S \to a_i$ est une règle de G.

Ensuite, pour $i < j$, $S \in E_{ij}$ si et seulement si il existe une règle $S \to S_1 S_2$ de G et un entier $i \leq k < j$ tels que $w[i,k] \in L_G(S_1)$ et $w[k+1, j] \in L_G(S_2)$. (déduit du lemme fondamental).

Donc E_{ij} peut être calculé à partir des ensemble $E_{i,k}$ et $E_{k+1,j}$.

On calcul donc les E_{ij} par récurrence sur la différence $j-i$.



⚠ les lignes s'app j et les col i ...

$$E_{i,j} = \bigcup_{\substack{S \to S_1 S_2 \\ \text{tel que} \\ S_1 \in E_{i,k} \text{ et} \\ S_2 \in E_{k+1,j}}} \{S\}$$

On a donc l'algorithme:

```
C-Y-K pour G  (w = a1 ... an) =
  pour 1 ≤ i ≤ j ≤ n faire
      E(i,j) ← ∅                                  (* initialisation *)
  fin
  pour i de 1 à n faire
      pour toute règle S → a faire
                                                   (* cas i=j *)
          si ai = a alors E(i,i) ← E(i,i) ∪ {S}
      fin
  fin
  pour 1 ≤ d ≤ n-1 faire                           (* cas i<j ; d=j-i *)
      pour 1 ≤ i ≤ n-d faire
          pour i ≤ k < i+d faire
              pour toute règle S → S1 S2 faire
                  si S1 ∈ E(i,k) et S2 ∈ E(k+1,i+d) alors
                      E(i,i+d) ← E(i,i+d) ∪ {S}
              fin
          fin
      fin
  fin
```

Court → ajouter ex