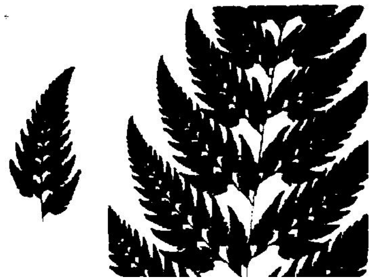


7 – Grammaires hors-contexte (suite)

Forme Normale de Greibach

- on peut faire en sorte que chaque production commence par « produire » un terminal, excepté pour l'axiome qui peut éventuellement engendrer ε
- une grammaire hors-contexte G = (N,T,P,S) est sous Forme Normale de Greibach (F.N.G) si toute production est de la forme :

$$A \rightarrow a\ \alpha$$

avec

$$A \in N,\ a \in T \text{ et } \alpha \in V^* = (N \cup T)^*$$

- toute grammaire hors-contexte propre et sans production vide est transformable en grammaire sous F.N.G.

Mise sous F.N.G : préliminaires

On part de G = (N,T,P,S) propre sans production vide et on construit G'= (N',T,P',S) équivalente et sous F.N.G.

- on suppose que toutes les productions sont de la forme :

$$A \rightarrow A_{i_1}\ A_{i_2}\ A_{i_3} \ldots A_{i_p}$$
$$A \rightarrow a$$

où $A_{i_k} \in N$ et $a \in T$

- N' reçoit les non-terminaux de N numérotés :

$$N' \leftarrow \{A_1, \ldots, A_n\}$$

- une production est « montante » si elle est de la forme :

$$A_i \rightarrow a\ \alpha \quad \text{avec } a \in T \text{ et } \alpha \in V^*$$

ou

$$A_i \rightarrow A_j\ \alpha \quad \text{avec } i < j \text{ et } \alpha \in V^*$$

Algorithme

1. on pose i=1
2. tant que i <= n, considérons les productions p de la forme $A_i \rightarrow \alpha$:
   - soit p est « montante » : ok
   - soit p est récursive à gauche
     - on retire cette récursivité en ajoutant un nouveau non-terminal numéroté à la suite de ceux de N' et on incrémente n de 1
   - sinon, on remplace A_k par ses parties droites dans la production p de la forme :

     $A_i \rightarrow A_k\ \alpha'$ avec $i > k$ et $\alpha' \in V^*$

     - en supprimant au passage les récursivités gauches et en incrémentant n à chacune

   $i \leftarrow i+1$
3. pour i de n-1 à 1 on remplace A_j par ses parties droites dans les productions de la forme :

   $A_i \rightarrow A_j\ \alpha$ avec $i<j$ et $\alpha \in V^*$

Exemple

- $G = (N,T,P,A_1)$ propre où :
  - $N = \{ A_1, A_2 \}$ et $T = \{ 0, 1 \}$
  - $P = \{ A_1 \to 0 \mid A_2\ A_2$
    $A_2 \to 1 \mid A_1\ A_2 \}$

1 $i \leftarrow 1$
2
- $A_1 \to 0$ ✏ ok, production « montante »
- $A_1 \to A_2\ A_2$ ✏ ok, production « montante »

$i \leftarrow 2$
- $A_2 \to 1$ ✏ ok, production « montante »
- $A_2 \to A_1\ A_2$ ✏ on remplace A_1 par ses parties droites
  - ➥ $A_2 \to 0\ A_2$ ✏ ok, production « montante »
  - ➥ $A_2 \to A_2\ A_2\ A_2$ ✏ pb : production récursive à gauche
- $A_2 \to A_2\ A_2\ A_2$ ✏ on supprime la récursivité
  - ➥ $A_2 \to 1\ A_3 \mid 0\ A_2\ A_3$ ✏ ok, productions « montantes »
  - ➥ $A_3 \to A_2\ A_2 \mid A_2\ A_2\ A_3$ ✏ pb : productions « descendantes »

Exemple (suite)

$i \leftarrow 3$
- $A_3 \to A_2\ A_2 \mid A_2\ A_2\ A_3$ ✏ on remplace A_2 par ses parties droites
  - ➥ $A_3 \to 1\ A_2 \mid 0\ A_2\ A_2$
  - ➥ $A_3 \to 1\ A_3\ A_2 \mid 0\ A_2\ A_3\ A_2$
  - ➥ $A_3 \to 1\ A_2\ A_3 \mid 0\ A_2\ A_2\ A_3$
  - ➥ $A_3 \to 1\ A_3\ A_2\ A_3 \mid 0\ A_2\ A_3\ A_2\ A_3$

3 $i \leftarrow 2$

$i \leftarrow 1$
- $A_1 \to A_2\ A_2$ ✏ on remplace A_2 par ses parties droites
  - ➥ $A_1 \to 1\ A_2 \mid 0\ A_2\ A_2 \mid 1\ A_3\ A_2 \mid 0\ A_2\ A_3\ A_2$
- la grammaire G' dont les productions sont écrites en orange est sous F.N.G. et équivalente à G.

Problème de l'appartenance

- il s'agit d'un problème de décision sur les langages hors-contexte (essentiel en compilation, traitement des langues naturelles ...)

  Données :
  - $G = (N,T,P,S)$ une grammaire hors-contexte
  - $\mu \in T^*$ un mot

  Question :
  - est-ce que $\mu \in L(G)$?
- une idée consiste à faire une analyse descendante du mot

  soit G=(N,T,P,S) telle que P ait k productions : on mime depuis l'axiome les dérivations de G. Avec G sous F.N.C, les algorithmes obtenus sont en $k^{|m|}$, |m| étant la longueur du mot à traiter
- une autre idée consiste à faire une analyse ascendante du mot

  Algorithme de Cocke, Younger et Kasami (1965) :
  - il part d'une grammaire sous F.N.C.
  - il utilise l'idée de programmation dynamique vue en Algorithmique.

Idée

- Soit $G = (N,T,P,S)$ une grammaire hors-contexte sous F.N.C. pour

$$L = L(G) = L(G) \backslash \{\varepsilon\}$$

- Facteur de μ débutant à la lettre $\mu(i)$ et de longueur j :

$$\mu_{i,j} = \mu(i)\ \mu(i+1) \ldots\ \mu(i+j-1)$$

- Pour tout couple (i,j), on calcule l'ensemble des non-terminaux :

$$V_{i,j} = \{ A,\ A \in N \text{ et tel que } A \Rightarrow^* \mu_{i,j} \}$$

- Le problème de l'appartenance se formule ainsi :

$$\mu \in L(G)\ ? \Leftrightarrow S \in V_{1,|\mu|}\ ?$$

Algorithme CYK

début

pour i **de** 1 **à** n **faire** *// initialisation 1re ligne*

$V_{i,1} \leftarrow \{A,\ A \in N,\ A \to \mu(i) \in P\}$ *// avec j fixé à 1*

pour j **de** 2 **à** n **faire**

pour i **de** 1 **à** n-j+1 **faire** *// pour chaque case de la ligne j*

$V_{i,j} \leftarrow \varnothing$

pour k **de** 1 **à** j-1 **faire**

$V_{i,j} \leftarrow V_{i,j} \cup \{$ A tels que $A \to BC \in P$
avec $B \in V_{i,k}$
et $C \in V_{i+k,j-k}$ $\}$

si $S \in V_{1,|\mu|}$ **alors** *μ est engendré* par G **sinon** *non*

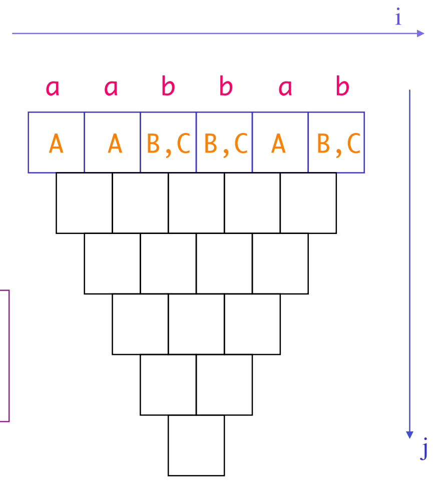
fin

Exemple

G = ({S,A,B,C}, T={a,b}, S, P)
P = { S → AB | BB
A → CC | AB | a
B → BB | CA | b
C → BA | AA | b }

A-t-on μ = aabbab $\in$ L(G) ?

j = 1
pour i **de** 1 **à** n **faire**
$V_{i,1} \leftarrow \{A,\ A \in N$ et $A \to \mu(i) \in P\}$

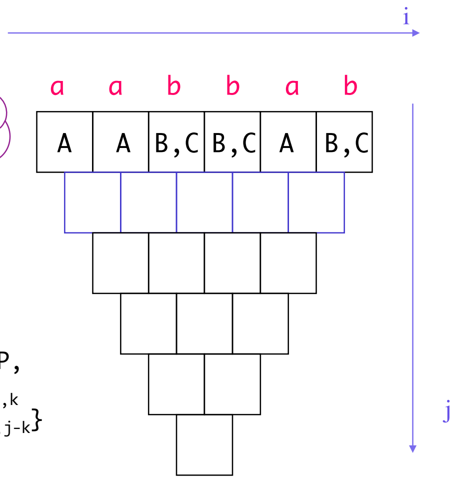


Exemple

S → AB | BB
A → CC | AB | a
B → BB | CA | b
C → BA | AA | b

nombre de cases par lignes

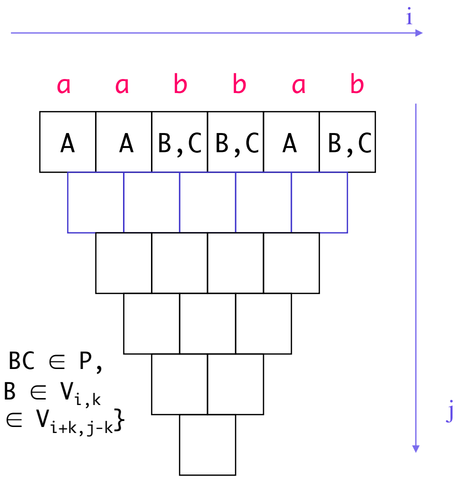
pour j = 2
pour i **de** 1 **à** n-j+1 **faire**
$V_{i,j} \leftarrow \varnothing$
pour k **de** 1 **à** j-1 **faire**
$V_{i,j} \leftarrow V_{i,j} \cup \{A,\ A \to BC \in P,$
avec $B \in V_{i,k}$
et $C \in V_{i+k,j-k}\}$



Exemple

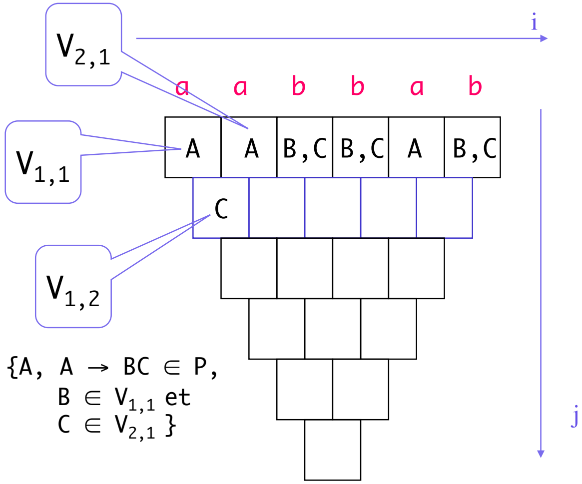
S → AB | BB
A → CC | AB | a
B → BB | CA | b
C → BA | AA | b

pour j = 2
pour i = 1
$V_{i,j} \leftarrow \varnothing$
pour k **de** 1 **à** j-1 **faire**
$V_{i,j} \leftarrow V_{i,j} \cup \{A,\ A \to BC \in P,$
avec $B \in V_{i,k}$
et $C \in V_{i+k,j-k}\}$



Exemple

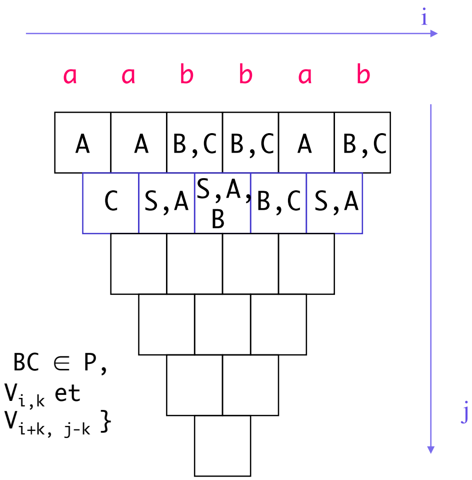
S → AB | BB
A → CC | AB | a
B → BB | CA | b
C → BA | AA | b



pour j = 2
  pour i = 1
    $V_{1,2} \leftarrow \varnothing$
    pour k = 1
      $V_{1,2} \leftarrow V_{1,2} \cup \{A,\ A \to BC \in P,$
        $B \in V_{1,1}$ et
        $C \in V_{2,1}\ \}$

Exemple

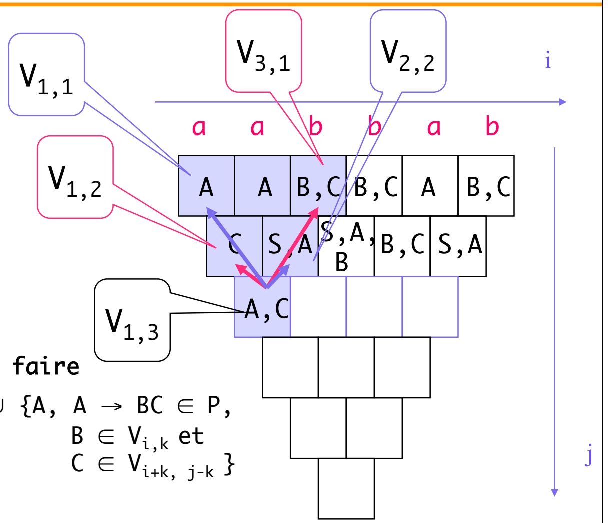
S → AB | BB
A → CC | AB | a
B → BB | CA | b
C → BA | AA | b



pour j = 2
  pour i = 2, 3, 4, 5
    $V_{i,j} \leftarrow \varnothing$
    pour k = 1
      $V_{i,j} \leftarrow V_{i,j} \cup \{A,\ A \to BC \in P,$
        $B \in V_{i,k}$ et
        $C \in V_{i+k,\ j-k}\ \}$

Exemple

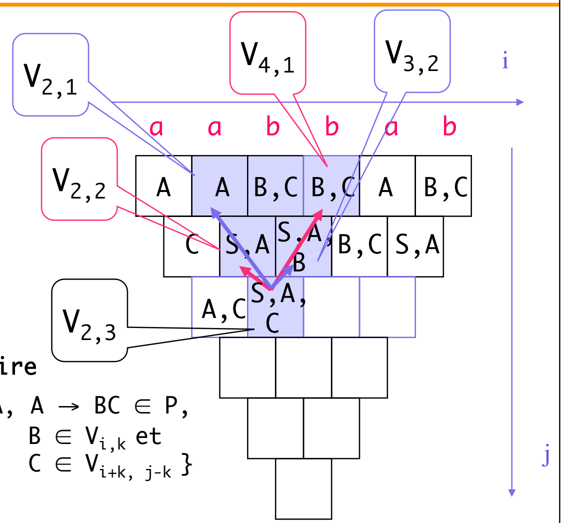
S → AB | BB
A → CC | AB | a
B → BB | CA | b
C → BA | AA | b



pour j = 3
  pour i = 1
    $V_{i,j} \leftarrow \varnothing$
    pour k de 1 à 2 faire
      $V_{i,j} \leftarrow V_{i,j} \cup \{A,\ A \to BC \in P,$
        $B \in V_{i,k}$ et
        $C \in V_{i+k,\ j-k}\ \}$

Exemple

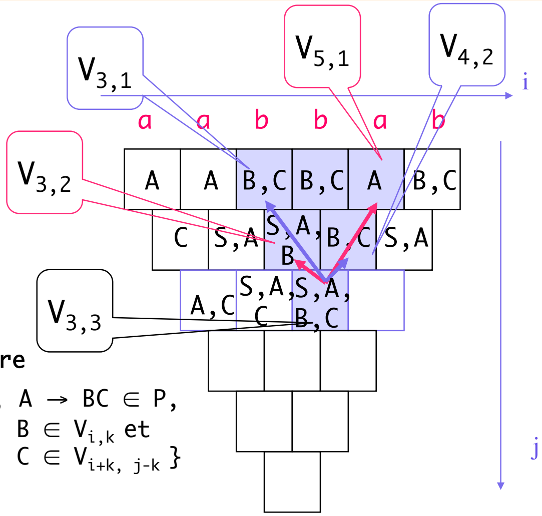
S → AB | BB
A → CC | AB | a
B → BB | CA | b
C → BA | AA | b



pour j = 3
  pour i = 2
    $V_{i,j} \leftarrow \varnothing$
    pour k de 1 à 2 faire
      $V_{i,j} \leftarrow V_{i,j} \cup \{A,\ A \to BC \in P,$
        $B \in V_{i,k}$ et
        $C \in V_{i+k,\ j-k}\ \}$

Exemple

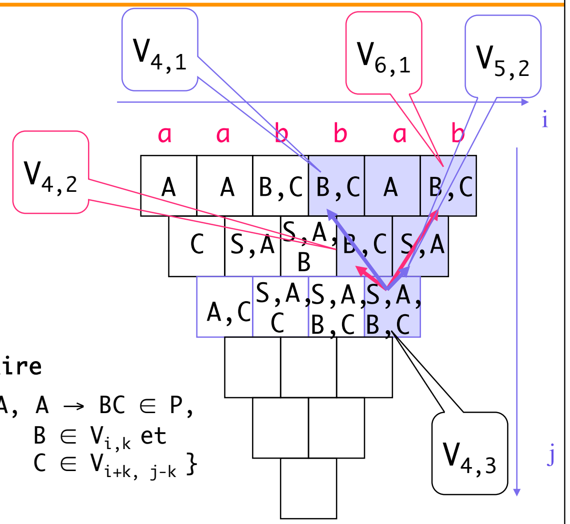
S → AB | BB
A → CC | AB | a
B → BB | CA | b
C → BA | AA | b



pour j = 3
pour i = 3
$V_{i,j} \leftarrow \varnothing$
pour k de 1 à 2 faire
$V_{i,j} \leftarrow V_{i,j} \cup \{A,\ A \rightarrow BC \in P,$
$B \in V_{i,k}$ et
$C \in V_{i+k,\ j-k}\ \}$

Exemple

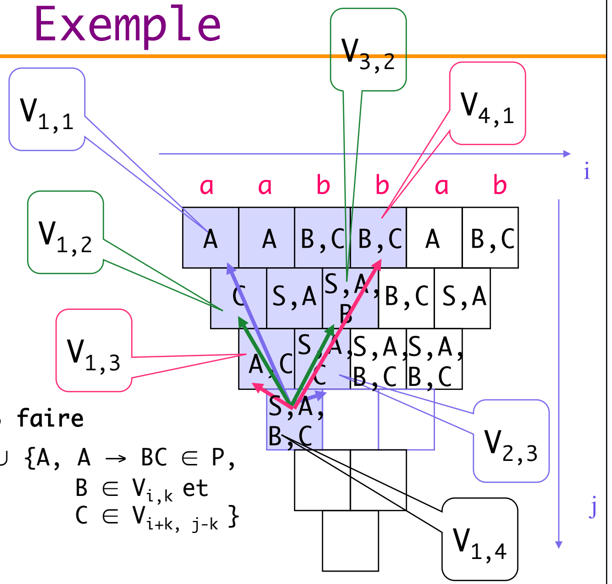
S → AB | BB
A → CC | AB | a
B → BB | CA | b
C → BA | AA | b



pour j = 3
pour i = 4
$V_{i,j} \leftarrow \varnothing$
pour k de 1 à 2 faire
$V_{i,j} \leftarrow V_{i,j} \cup \{A,\ A \rightarrow BC \in P,$
$B \in V_{i,k}$ et
$C \in V_{i+k,\ j-k}\ \}$

Exemple

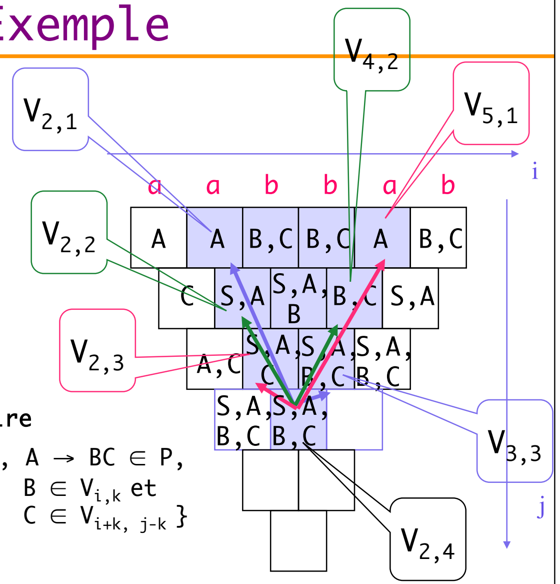
S → AB | BB
A → CC | AB | a
B → BB | CA | b
C → BA | AA | b



pour j = 4
pour i = 1
$V_{i,j} \leftarrow \varnothing$
pour k de 1 à 3 faire
$V_{i,j} \leftarrow V_{i,j} \cup \{A,\ A \rightarrow BC \in P,$
$B \in V_{i,k}$ et
$C \in V_{i+k,\ j-k}\ \}$

Exemple

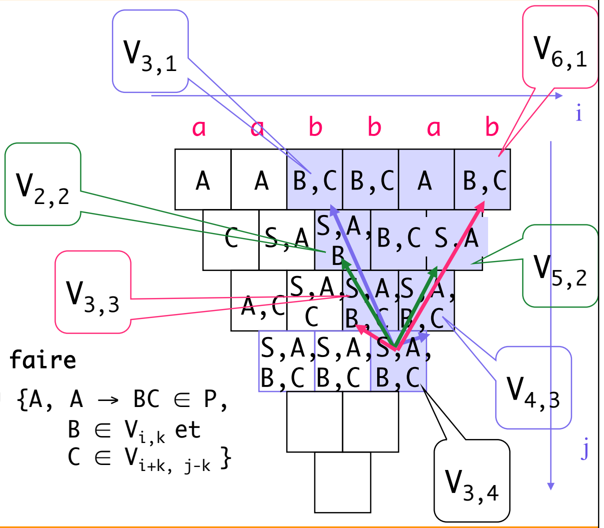
S → AB | BB
A → CC | AB | a
B → BB | CA | b
C → BA | AA | b



pour j = 4
pour i = 2
$V_{i,j} \leftarrow \varnothing$
pour k de 1 à 3 faire
$V_{i,j} \leftarrow V_{i,j} \cup \{A,\ A \rightarrow BC \in P,$
$B \in V_{i,k}$ et
$C \in V_{i+k,\ j-k}\ \}$

Exemple

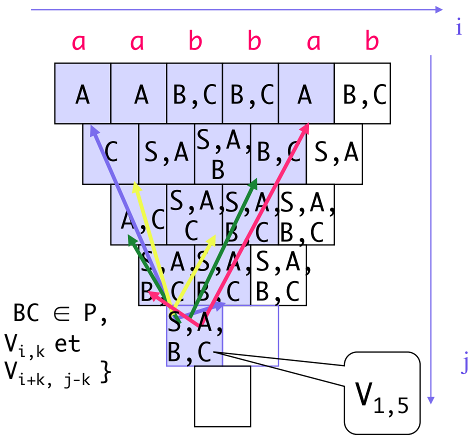
S → AB | BB
A → CC | AB | a
B → BB | CA | b
C → BA | AA | b



pour j = 4
  pour i = 3
    $V_{i,j} \leftarrow \varnothing$
    pour k de 1 à 3 faire
      $V_{i,j} \leftarrow V_{i,j} \cup \{A,\ A \rightarrow BC \in P,$
        $B \in V_{i,k}$ et
        $C \in V_{i+k,\ j-k}\ \}$

Exemple

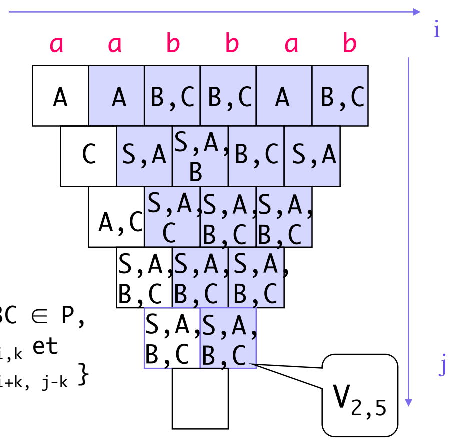
S → AB | BB
A → CC | AB | a
B → BB | CA | b
C → BA | AA | b



pour j = 5
  pour i = 1
    $V_{i,j} \leftarrow \varnothing$
    pour k de 1 à 4 faire
      $V_{i,j} \leftarrow V_{i,j} \cup \{A,\ A \rightarrow BC \in P,$
        $B \in V_{i,k}$ et
        $C \in V_{i+k,\ j-k}\ \}$

Exemple

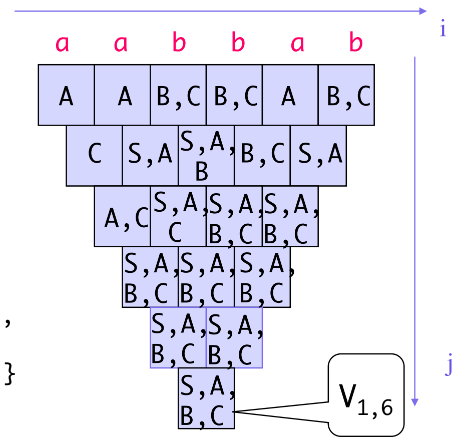
S → AB | BB
A → CC | AB | a
B → BB | CA | b
C → BA | AA | b



pour j = 5
  pour i = 1
    $V_{i,j} \leftarrow \varnothing$
    pour k de 1 à 4 faire
      $V_{i,j} \leftarrow V_{i,j} \cup \{A,\ A \rightarrow BC \in P,$
        $B \in V_{i,k}$ et
        $C \in V_{i+k,\ j-k}\ \}$

Exemple

S → AB | BB
A → CC | AB | a
B → BB | CA | b
C → BA | AA | b



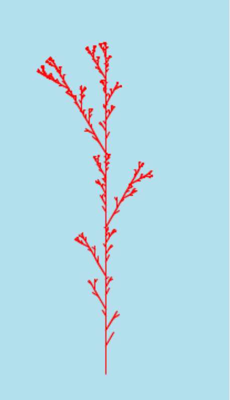
pour j = 6
  pour i = 1
    $V_{i,j} \leftarrow \varnothing$
    pour k de 1 à 5 faire
      $V_{i,j} \leftarrow V_{i,j} \cup \{A,\ A \rightarrow BC \in P,$
        $B \in V_{i,k}$ et
        $C \in V_{i+k,\ j-k}\ \}$

➥ $S \in V_{1,6}$ donc $\mu \in L(G)$

Systèmes de Lindenmayer

- Les systèmes de Lindenmayer* sont des systèmes de réécriture inventés en 1968 par le biologiste A. Lindenmayer
- ils modélisent entre autres la croissance des plantes
- ce sont des grammaires de Type I ou II
- parmi eux, les D0L-systems sont hors-contexte

```
def plante(n,T) :     # un D0L-system
    if n == 0 : forward(T)
    else :
        r = randint(0,5)
        if r<3:
            plante(n-1,2*T/3)
            right(30)
            plante(n-1,T/3)
            back(T/3)
            left(30)
            plante(n-1,T/3)
        else :
            plante(n-1,2*T/3)
            left(30)
            plante(n-1,T/3)
            back(T/3)
            right(30)
            plante(n-1,T/3)
```



* L-systems en anglais