Le problème de l'appartenance

- On va s'intéresser à un problème de décision de théorie des langages utile en compilation

- Le problème de l'appartenance :
  - **Données** :
    - $G=(N,T,S,R)$ une grammaire algébrique
    - $m \in T^*$ un mot
  - **Question** :
    - Est-ce que $m \in L(G)$?

Résolution

Plusieurs manières pour résoudre le problème:

- au moyen des formes normales
- A l'aide de la transformation grammaire vers AP.

Avec les grammaires

- On met G sous FNG
- Toutes les règles sont de la forme

$$X \rightarrow a\gamma \text{ pour } \gamma \in (N \cup T)^*$$

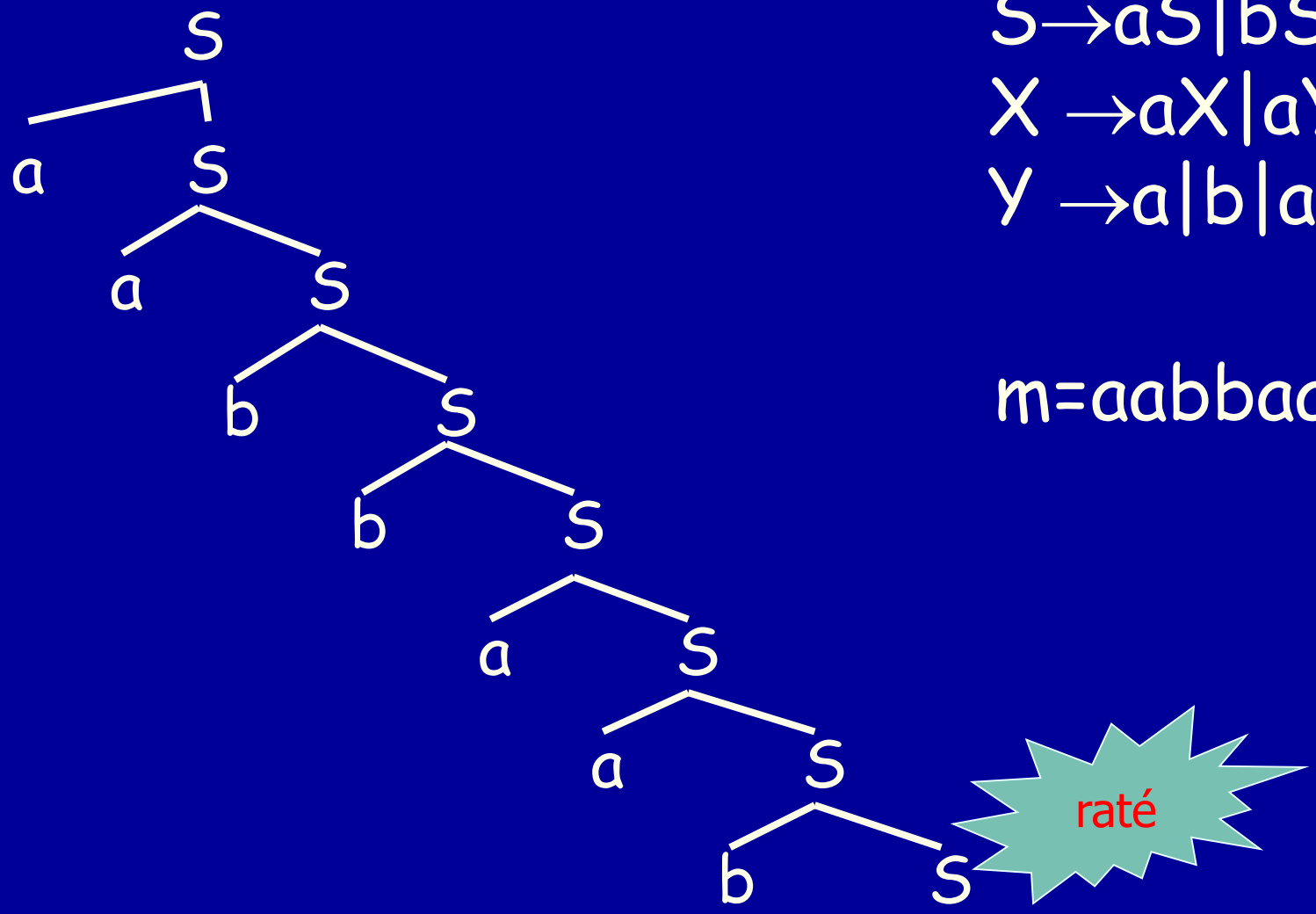
- Complexité de l'algorithme de décision:
  - Soit **k** le nombre maximal de règles associées aux variables
  - Chaque règle permet d'ajouter **un** terminal
  - Le mot est de longueur **|m|**
  - La complexité temporelle de cet algorithme est donc au plus $k^{|m|}$

Exemple

- m=aabbaab $\in$ L(G) pour G sous FNG de règles
  - S$\rightarrow$aS|bS|bX; X $\rightarrow$aX|aY|a; Y $\rightarrow$a|b|aY|bY

- On cherche les dérivations gauches qui permettent d 'engendrer m.

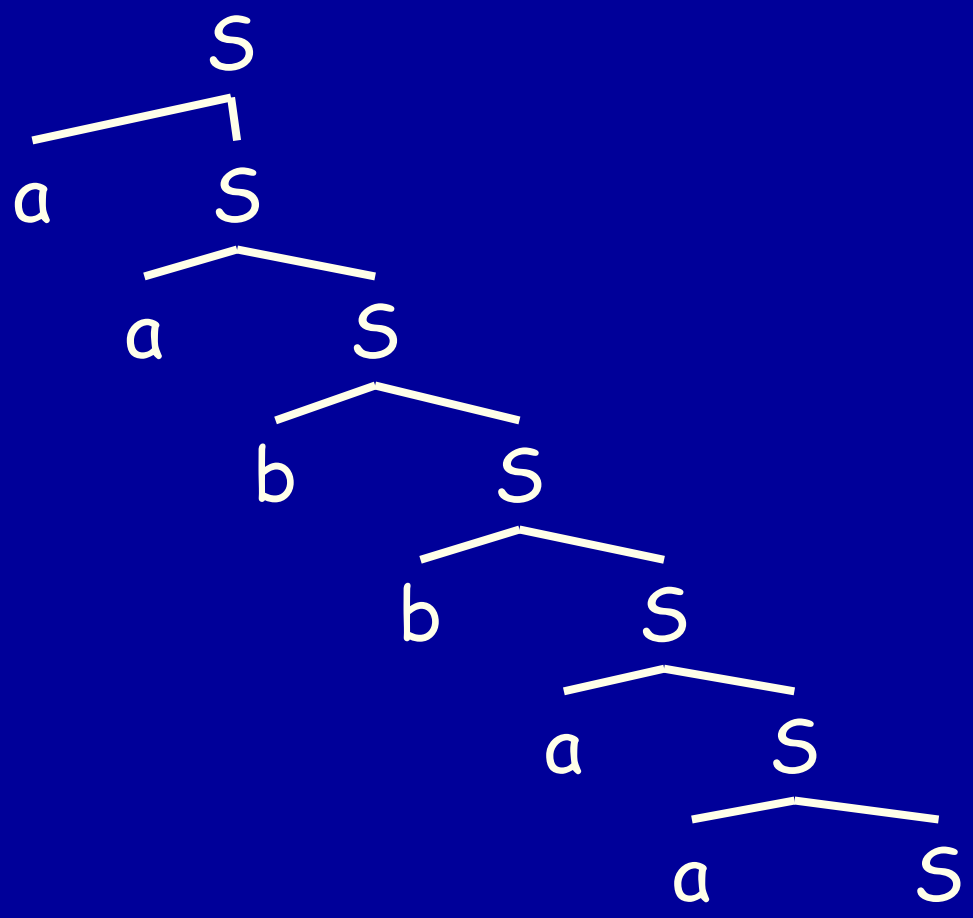
Exemple



S→aS|bS|bX;
X →aX|aY|a;
Y →a|b|aY|bY

m=aabbaab

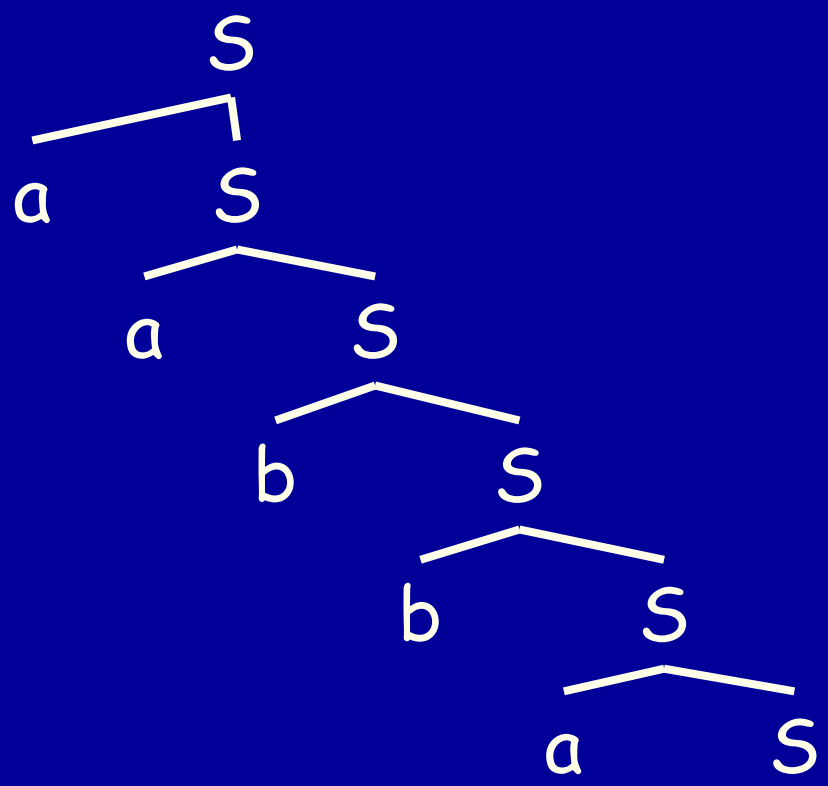
Exemple



S→aS|bS|bX;
X →aX|aY|a;
Y →a|b|aY|bY

m=aabbaaab

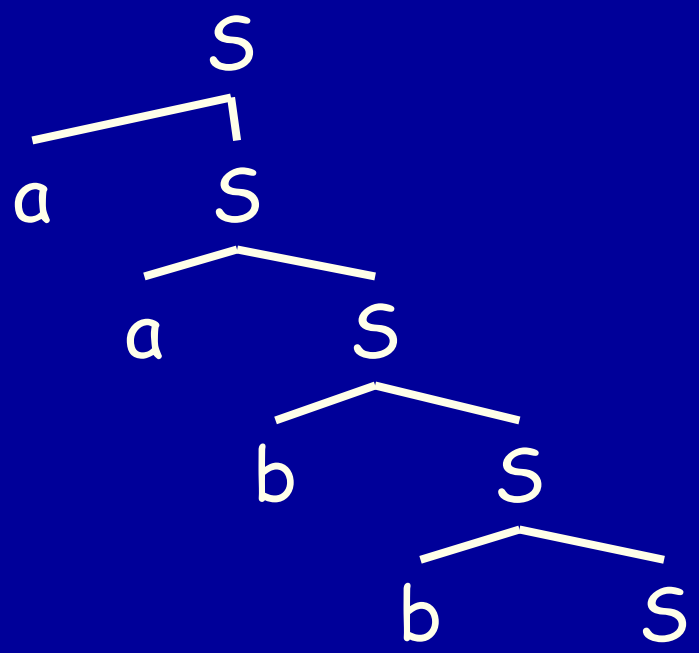
Exemple



S→aS|bS|bX;
X →aX|aY|a;
Y →a|b|aY|bY

m=aabbaab

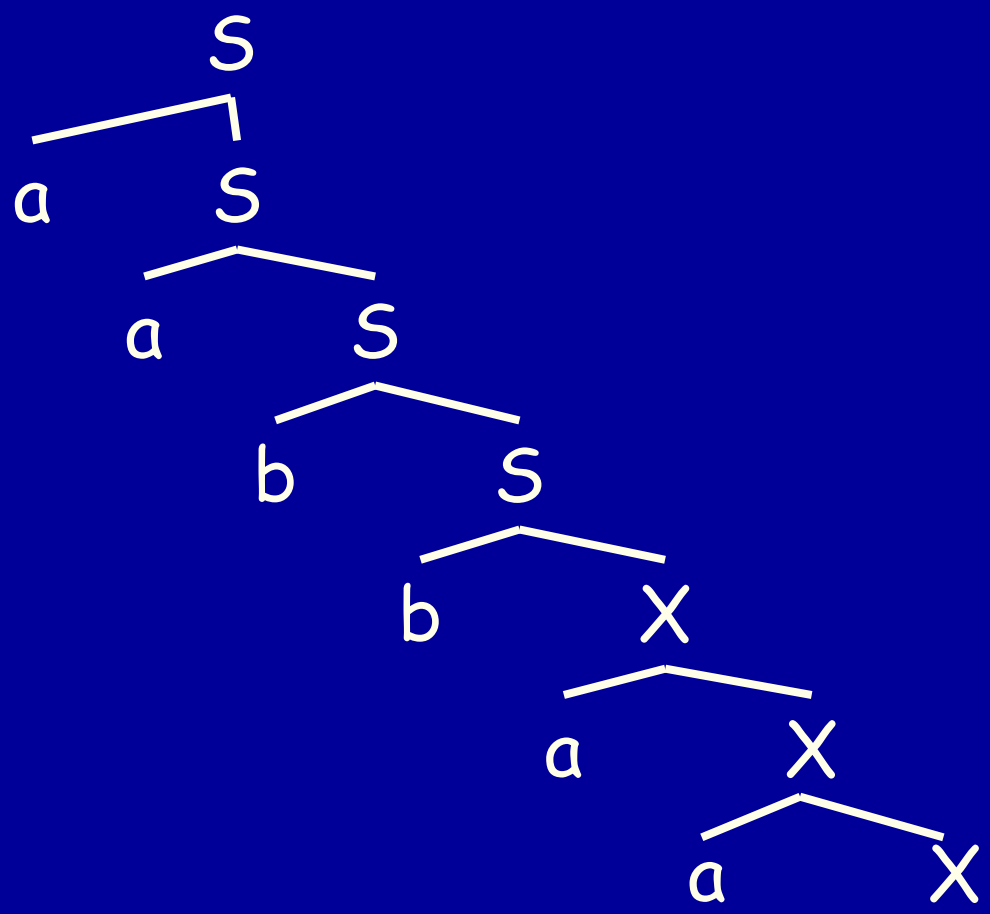
Exemple



S→aS|bS|bX;
X →aX|aY|a;
Y →a|b|aY|bY

m=aabbaab

Exemple

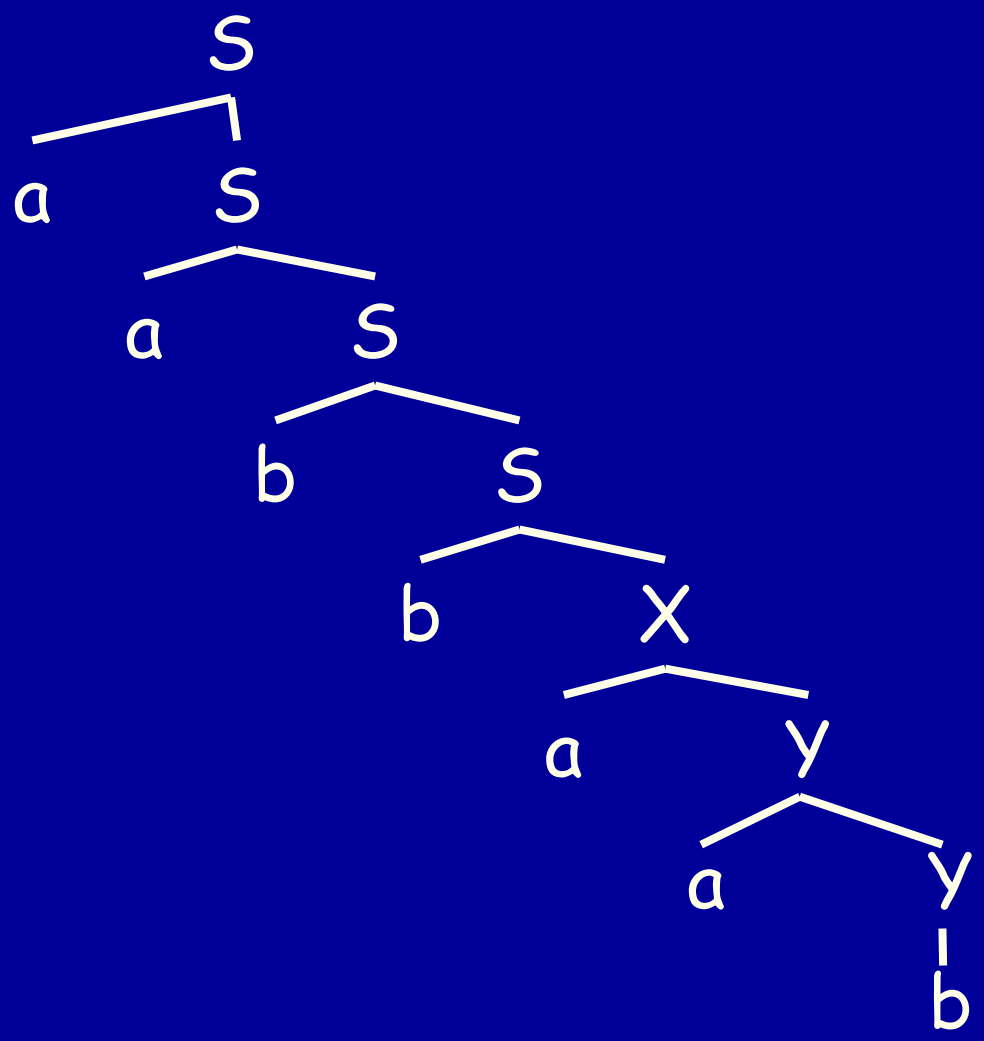


S→aS|bS|bX;
X →aX|aY|a;
Y →a|b|aY|bY

m=aabbaab

raté

Exemple



S→aS|bS|bX;
X →aX|aY|a;
Y →a|b|aY|bY

m=aabbaab

gagné

Conclusion

Il vaut mieux chercher un autre algorithme

Celui-ci est beaucoup trop lent!!!!

Avec les grammaires

- On met G sous FNC
- Toutes les règles sont de la forme

  $X \rightarrow AB$ ou $X \rightarrow a$ pour $X, A, B \in N$ et $a \in T$
- Complexité de l'algorithme de décision?
  - Soit k le nombre maximal de règles associées aux variables; Le mot est de longueur $|m|$
  - Dans le pire des cas, on a un arbre binaire à $|m|$ feuilles et $|m|-1$ nœuds internes et k choix possibles par nœud
  - Le temps de cet algorithme est donc au plus $k^{|m|}$
- analogue au cas précédent; envisager une autre solution

Dernière tentative : automates à pile

On part de la grammaire G

- On construit l'AP correspondant
- On donne en entrée à l'AP le mot m
  - Si AP accepte m, $m \in L(G)$
  - Sinon, $m \notin L(G)$

- Complexité :
  - Comme on ne sait pas déterminiser les AP, la simulation déterministe d'un AP ND est a priori exponentielle.
  - Il faut envisager tous les arbres de calcul et en trouver un pour lequel la lecture a réussi.
- Il faut donc une autre méthode

Méthode Cocke Younger et Kasami (1965)

- Utilise :
  - Une grammaire G sous FNC
  - La **programmation dynamique**
- Combine les avantages des
  - Algorithmes **gloutons** qui effectuent le meilleur choix localement
  - Algorithmes de **recherche exhaustive** qui essayent toutes les possibilités et choisissent la meilleure
- Clairement, les solutions précédentes sont des algorithmes de recherche exhaustifs

Algorithme CYK

- **Notation** : $x_{i,j}$ facteur de x contenant les lettres x(i)x(i+1)…x(i+j-1) i.e. le facteur de longueur j qui commence en position i
- **Exemple :** Pour x=abracadabra, on a $x_{3,3}$=abracadabra=rac
- **Principe :** On calcule l'ensemble des variables $V_{i,j}$

$$V_{i,j} = \{A : A \in N : A \rightarrow^* x_{i,j}\}$$

et ceci pour tout i et pour tout j

- Le problème de l'appartenance se formule :

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

Algorithme CYK

Pour i:=1 **à** n **faire**

$V_{i,1} := \{A \mid A \in N,\ A \rightarrow x(i) \in R\}$

Pour j:=2 **à** n **faire**

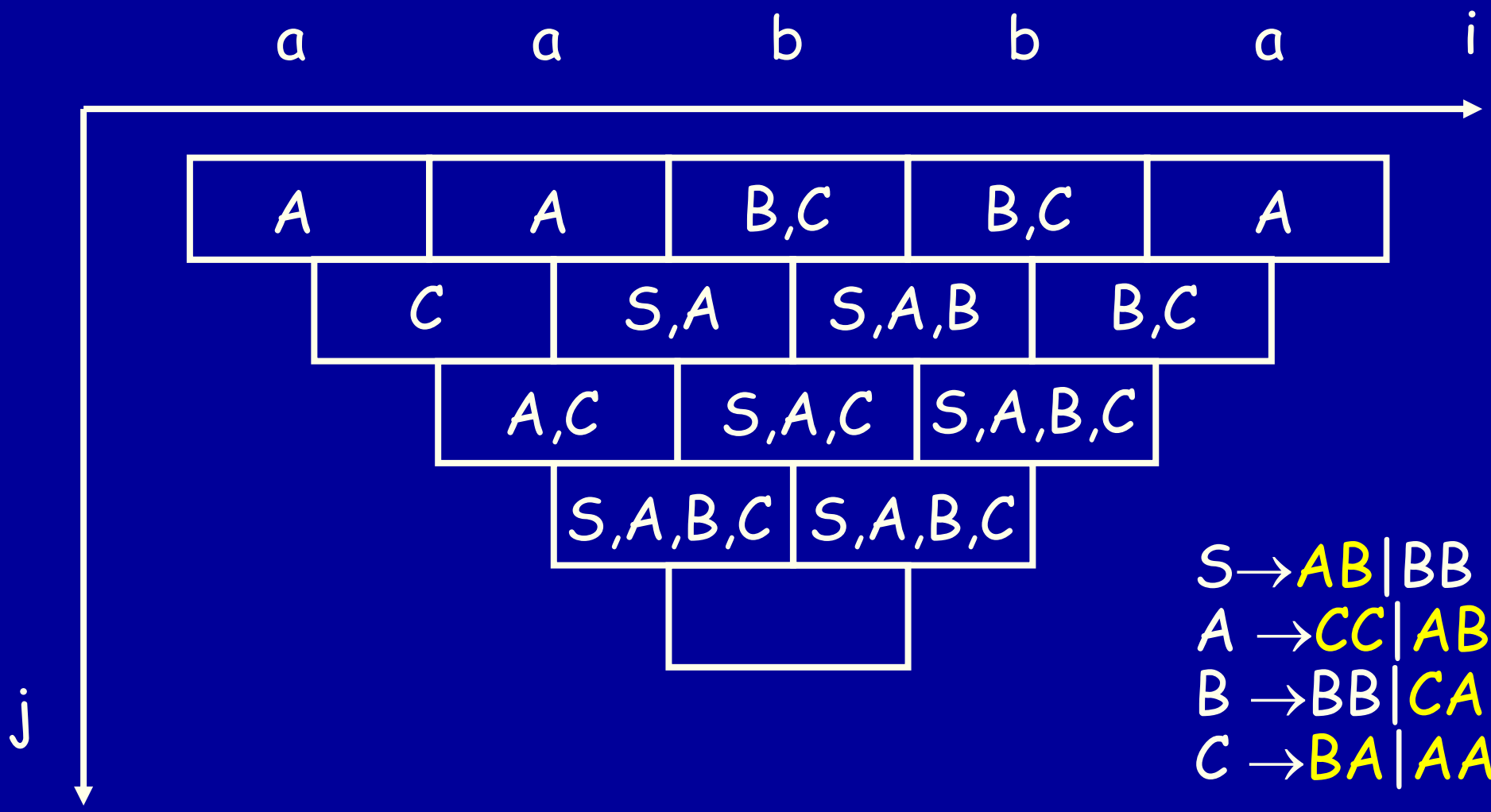
Pour i:=1 **à** n-j+1 **faire**

$V_{i,j} := \varnothing$

Pour k:=1 **à** j-1 **faire**

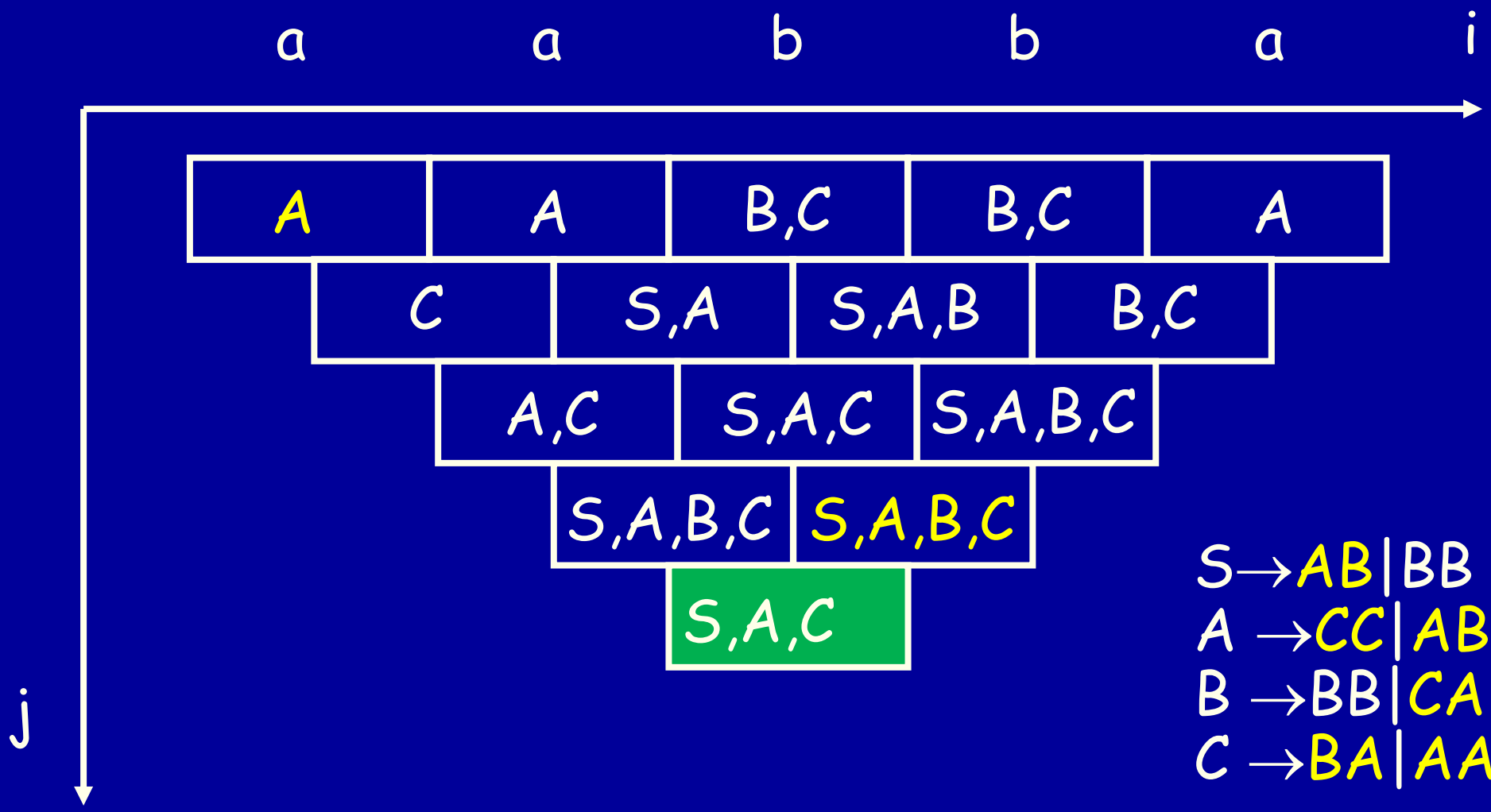
$V_{i,j} := V_{i,j} \cup \{A \mid A \rightarrow BC \in R,\ B \in V_{i,k},\ C \in V_{i+k,j-k}\}$

Exemple



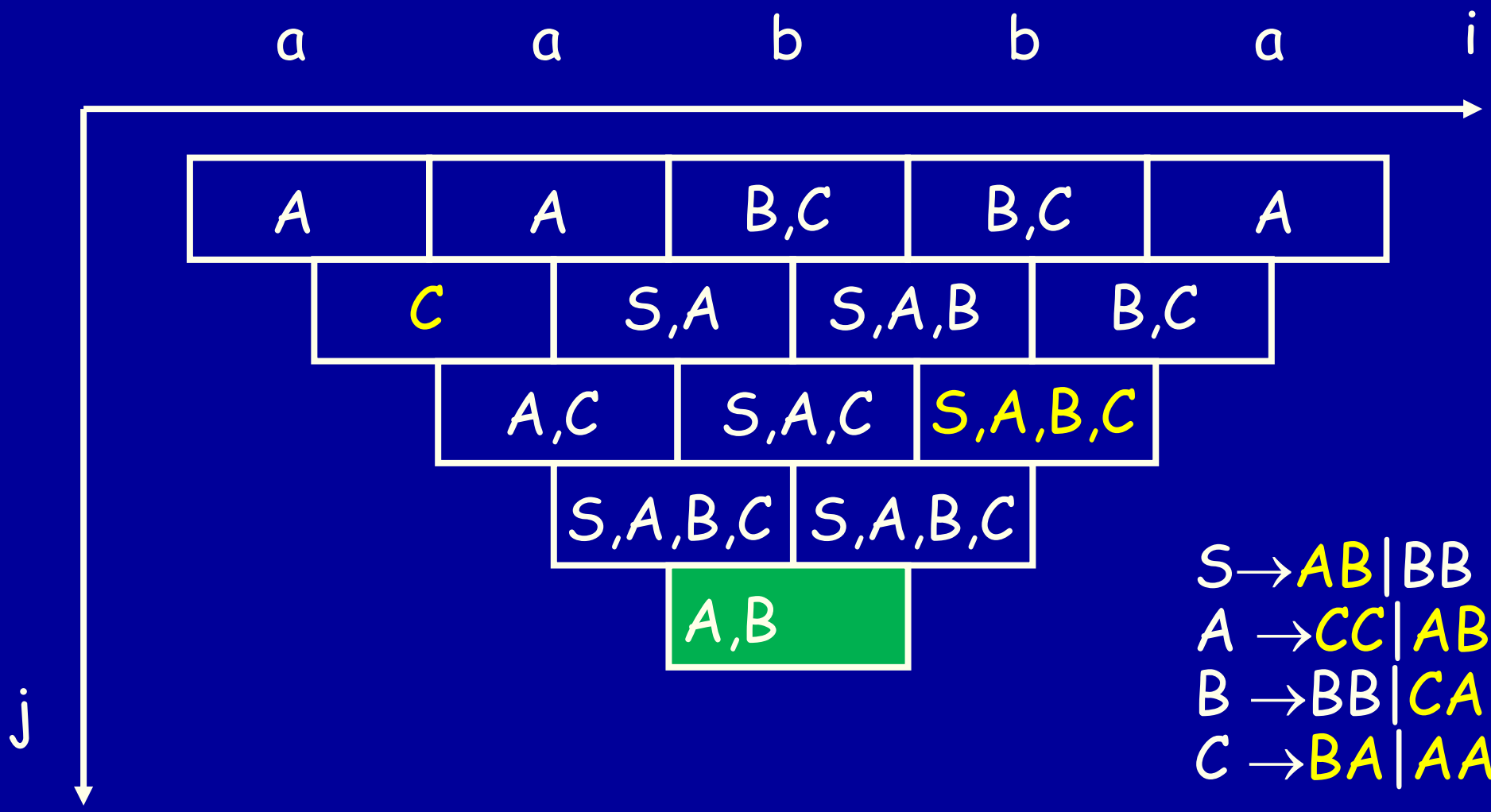
$S \rightarrow AB \mid BB$
$A \rightarrow CC \mid AB \mid a$
$B \rightarrow BB \mid CA \mid b$
$C \rightarrow BA \mid AA \mid b$

Exemple



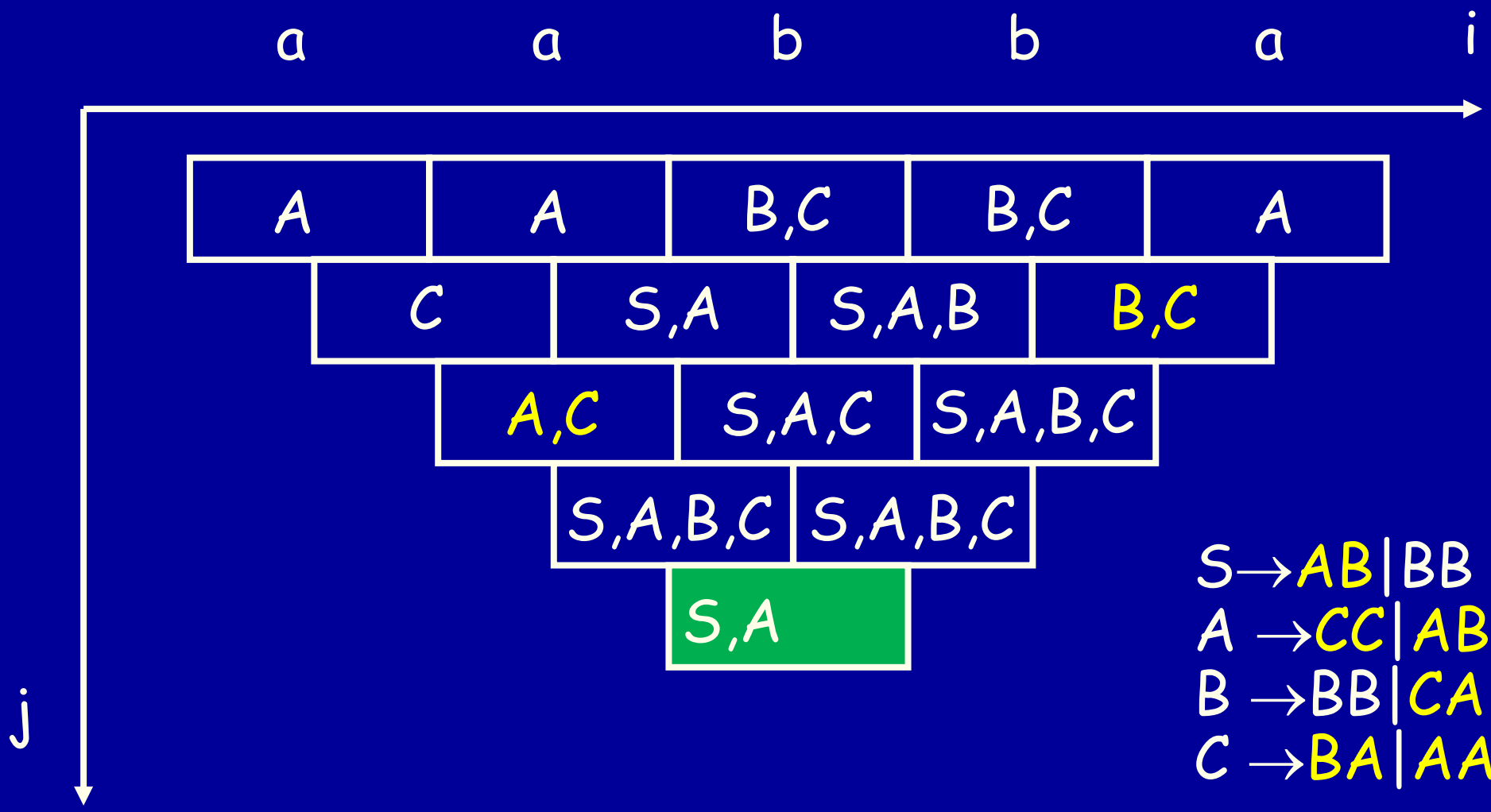
$S \rightarrow AB \mid BB$
$A \rightarrow CC \mid AB \mid a$
$B \rightarrow BB \mid CA \mid b$
$C \rightarrow BA \mid AA \mid b$

Exemple



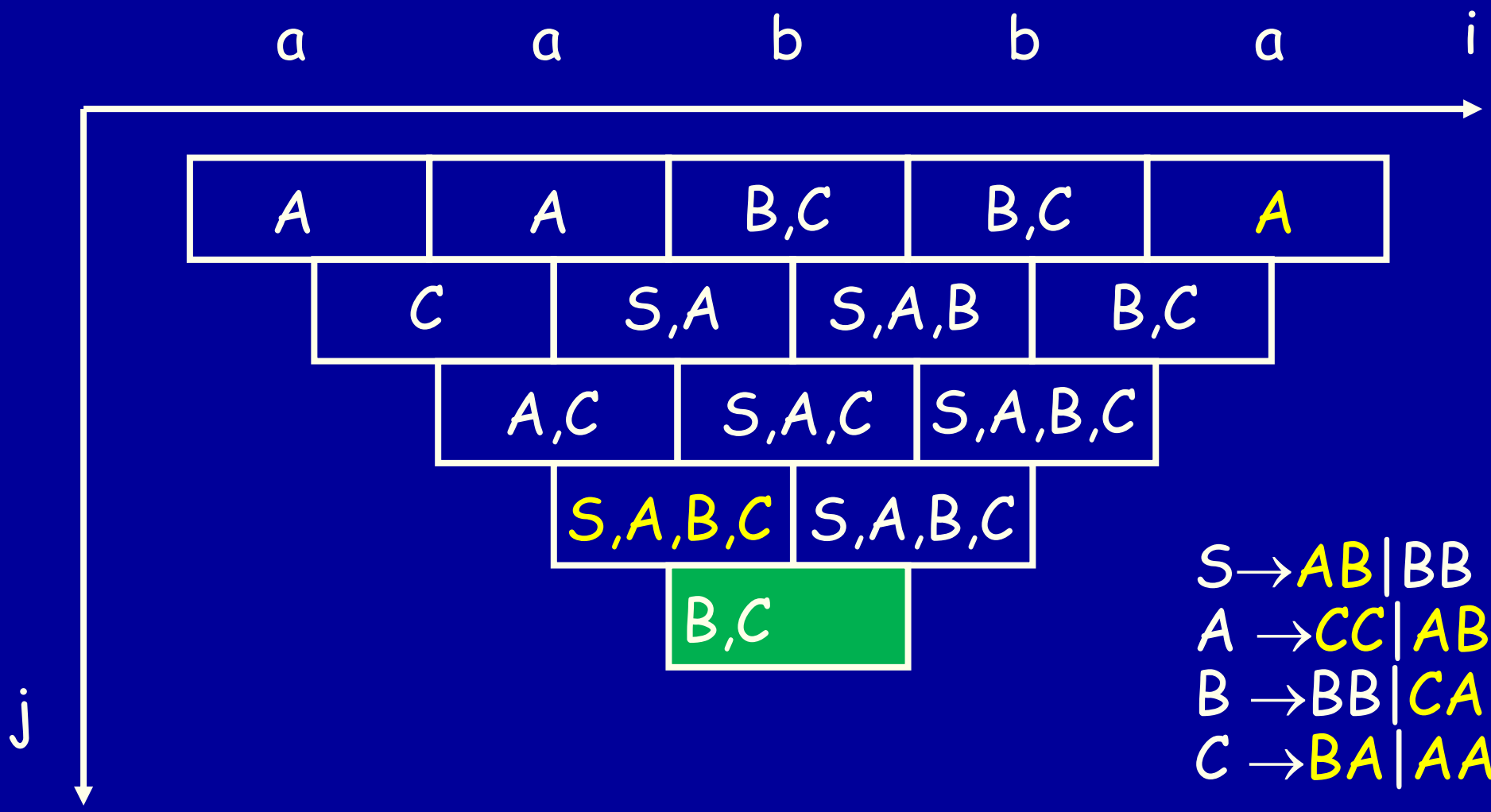
$S \rightarrow AB \mid BB$
$A \rightarrow CC \mid AB \mid a$
$B \rightarrow BB \mid CA \mid b$
$C \rightarrow BA \mid AA \mid b$

Exemple



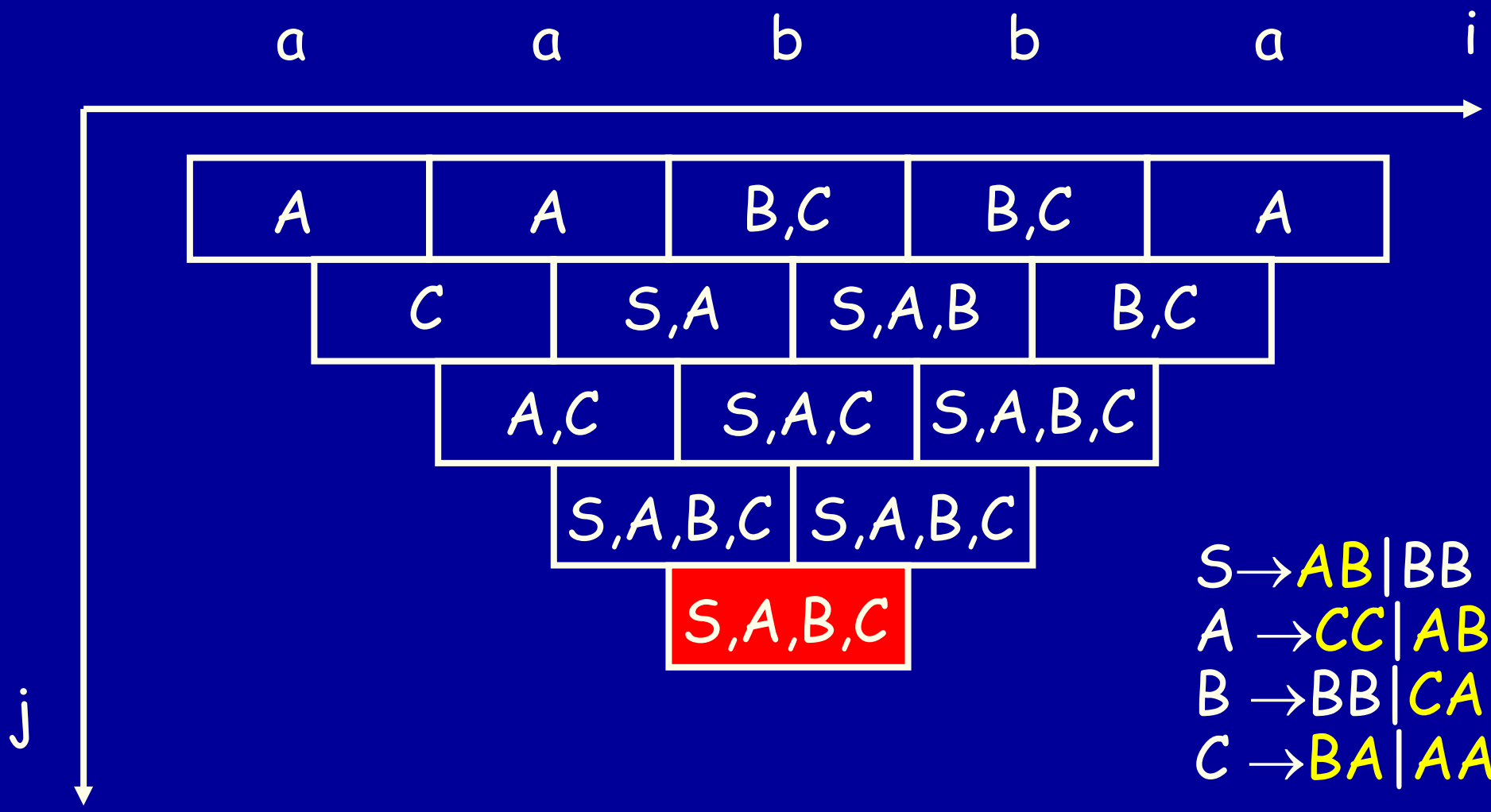
$S \rightarrow AB \mid BB$
$A \rightarrow CC \mid AB \mid a$
$B \rightarrow BB \mid CA \mid b$
$C \rightarrow BA \mid AA \mid b$

Exemple



$S \rightarrow AB \mid BB$
$A \rightarrow CC \mid AB \mid a$
$B \rightarrow BB \mid CA \mid b$
$C \rightarrow BA \mid AA \mid b$

Exemple



$S \rightarrow AB \mid BB$
$A \rightarrow CC \mid AB \mid a$
$B \rightarrow BB \mid CA \mid b$
$C \rightarrow BA \mid AA \mid b$

Complexité

n { Pour i:=1 à n faire
$V_{i,1} := \{A : A \in N,\ A \rightarrow x(i) \in R\}$ — Temps constant

Pour j:=2 à n faire

$\sum_{i=1}^{n-j+1}$ { Pour i:=1 à n-j+1 faire
$V_{i,j} := \emptyset$

$\sum_{k=1}^{j-1}$ { Pour k:=1 à j-1 faire
$V_{i,j} := V_{i,j} \cup \{A : A \rightarrow BC \in R,\ B \in V_{i,k},\ C \in V_{i+k,j-k}\}$ — Temps constant c

$$\sum_{j=2}^{n}\left(\sum_{i=1}^{n-j+1}\left(\sum_{k=1}^{j-1} c\right)\right)$$

$$n + \sum_{j=2}^{n}\left(\sum_{i=1}^{n-j+1}\left(\sum_{k=1}^{j-1} c\right)\right)$$

Complexité

$$n+\sum_{j=2}^{n}\left(\sum_{i=1}^{n-j+1}\left(\sum_{k=1}^{j-1}c\right)\right)=n+c\sum_{j=2}^{n}\left(\sum_{i=1}^{n-j+1}(j-1)\right)=$$

$$n+c\sum_{j=2}^{n}((n-j+1)(j-1))=n+c\sum_{j=2}^{n}(-j^2+j(n+2)-(n+1))=$$

$$n-c\left(\frac{n(n+1)(2n+1)}{6}-1\right)+c(n+2)\left(\frac{n(n+1)}{2}-1\right)-c(n+1)(n-1)=$$

$$O(n^3)$$

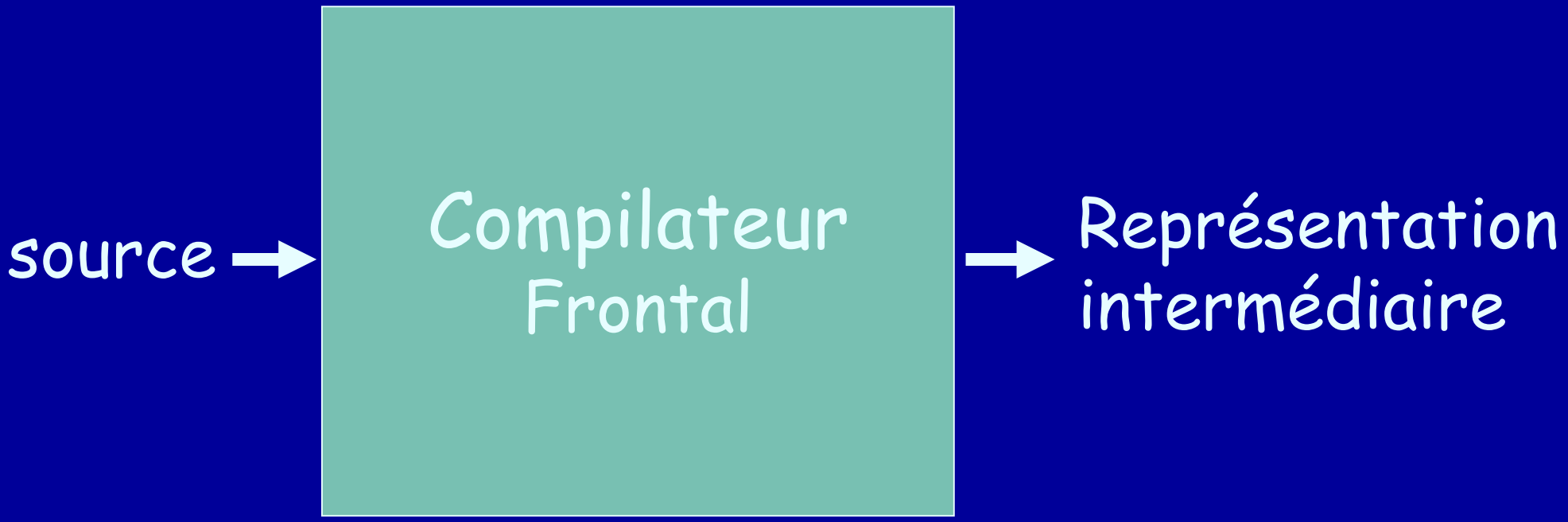
Explication

- Si on a la règle
  - $A \rightarrow BC$ avec
    - $B \in V_{i,k}$
    - $C \in V_{i+k,j-k}$
  - $B \rightarrow^* x_{i,k}$ et $C \rightarrow^* x_{i+k,j-k}$
  - Donc, $A \rightarrow^* x_{i,j}$ et $A \in V_{i,j}$
- Et vice-versa

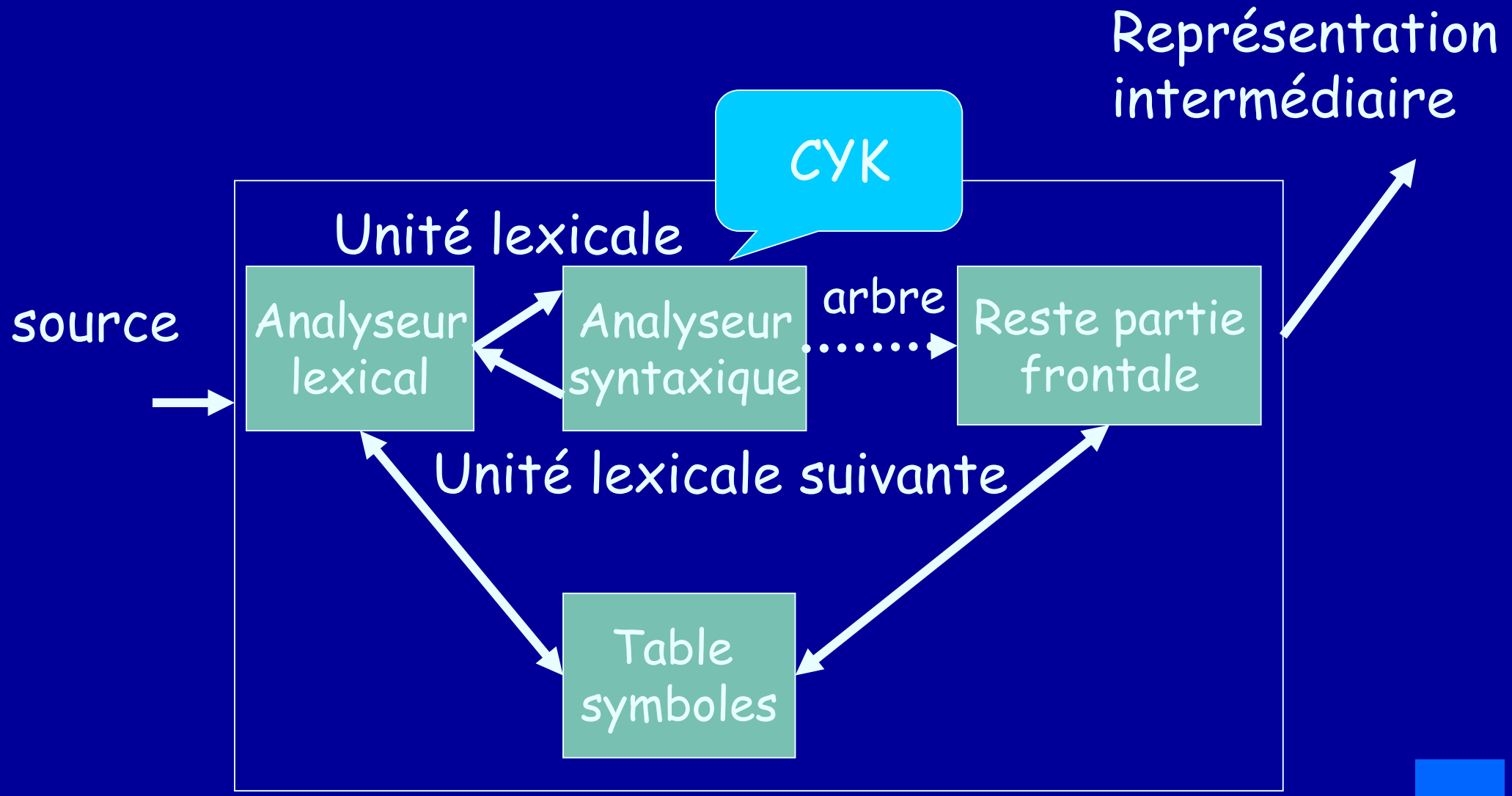
La programmation dynamique

- L'apport de la programmation dynamique est dans la construction de la « piramide »
- Celle-ci aurait tout aussi bien pu être remplacée par des appels récursifs
- Dans ce cas, on retombe sur les idées du début, car cela revient de faire un grand nombre de fois le même appel.
- Par contre, une implémentation avec les appels récursifs qu'on fait uniquement une fois, en gardant le résultat est équivalent à CYK.

Compilateur



Compilateur



Plus concrètement

- Un compilateur n'utilise pas CYK
- Il préfère utiliser un AP
- Problème du non-déterminisme
- Celui-ci est résolu en ne s'intéressant dans la mesure du possible qu'à des classes de grammaires déterministes

Limites des langages algébriques

Tout est algébrique?

- On connaît bien les langages algébriques
  - Plusieurs façons de les caractériser
  - Les opérations qui préservent l'algébricité
    - Union, concaténation et étoile

- Tout langage est-il algébrique ?
  - Comme pour les rationnels, il existe des langages non algébriques.
  - Pour le prouver, on utilise un argument de diagonalisation.

Hypothèse: tout est algébrique

- algébriques = reconnaissables; en bijection avec AP et **N**.
- On énumère les AP sur un alphabet à une lettre et on les ordonne dans une liste; L_0 est reconnu par le 1er AP de la liste, L_1 par 2e…
- Si M était dans T, il existerait un indice j tel que $M=L_j$. Puisque $M=L_j$, si
  - $j \in L_j$ alors, par définition de M, $j \notin M$
  - $j \notin L_j$ alors, par définition de M, $j \in M$
- Une contradiction dans les deux cas

L'ensemble des algébriques est infini mais dénombrable

	L_0	L_1	L_2	L_3
mot_0	O	N	N	O
mot_1	N	N	O	N
mot_2	O	N	O	O

T[i,j]=Oui si $i \in L_j$
Non sinon

$i \in M \Leftrightarrow i \notin L_i$
M n'est pas dans T

Question

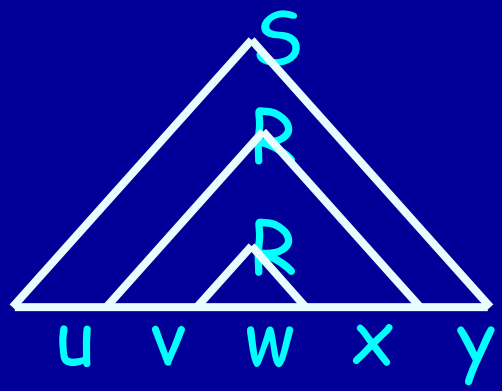
- Comment montrer qu'un langage L donné **n'est pas** algébrique.
- **Problème**

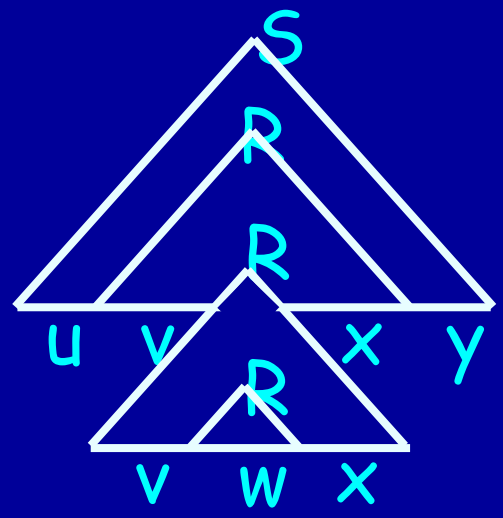
  Donnée : L un langage

  Question : L est-il non algébrique?
- Pour les rationnels, on a utilisé le principe des tiroirs sur les états parcourus pour montrer qu'on passe plusieurs fois par le même état

Question

- Pour les algébriques, on utilise le principe des tiroirs sur l'arbre syntaxique qui doit contenir plusieurs fois le même sous-arbre car les variables sont en nombre fini.





Lemme de la pompe

Lemme : Soit L un langage algébrique. Il existe une constante n (qui ne dépend que de L) telle que si $z \in L$, $|z| \geq n$, z se factorise en z=uvwxy tel que

i. $|vx| > 0$ et
ii. $|vwx| \leq n$ et
iii. $\forall i \geq 0$, $uv^iwx^iy \in L$

Utilisation du Lemme de la pompe

- Comme pour les rationnels, ce lemme ne sert qu'à montrer la **non algébricité** d'un langage.
- On utilise la contraposée, en supposant L algébrique et on cherche une contradiction
- Si, pour un $z \in L$ quelconque de longueur suffisante $\forall$ décomposition $z=uvwxy$ vérifiant
  1) $|vx|>0$ et
  2) $|vwx| \leq n$ alors
  3) $\exists i \geq 0, uv^iwx^iy \notin L$

  On conclut que L n'est pas algébrique

Exemple $L=\{a^ib^ic^i : i>0\}$

- On suppose L algébrique et on fixe n.
- Soit $z=a^nb^nc^n=uvwxy$
- $|vwx|\leq n \Rightarrow$ vx ne peut avoir à la fois des a et c
  - v et x ne contiennent que des a $\Rightarrow$ uwy manque de a
  - v et x ne contiennent que des b $\Rightarrow$ uwy manque de b
  - v et x ne contiennent que des c $\Rightarrow$ uwy manque de c
  - vx contient des a et b $\Rightarrow$ uwy manque de a et de b
  - vx contient des b et c $\Rightarrow$ uwy manque de b et de c
- Pour chaque factorisation, on aboutit à une contradiction (le mot uwy n'est pas dans le langage). On en déduit donc que L n'est pas algébrique.

Autre exemple : $L=\{a^i b^j c^i d^j : i, j \geq 1\}$

- On suppose L algébrique; soit n la constante du lemme
- On choisit $z=a^n b^n c^n d^n$

  1) $|vx|>0$ et
  2) $|vwx|\leq n$ et
  3) $\exists i \geq 0,\ uv^i wx^i y \notin L$

- Par la condition 2, vx contient soit
  - Qu'une seule lettre a ou b ou c ou d
    uwy n'est pas dans le langage, car cette lettre manque
  - Que des a et des b
  - Que des b et des c
  - Que des c et des d

  } cas analogues

  uwy n'est pas dans le langage, car deux lettres manquent

Propriétés de clôture

Propriété

Les langages algébriques sont clos

- Pour l'union
- Pour la concaténation
- Pour l'étoile de Kleene

- Cela nous donne une manière de construire une grammaire algébrique en « découpant » astucieusement le langage à engendrer

Clôture par intersection

- Si L et M sont deux langages algébriques, alors on ne peut en déduire que L∩M est algébrique

  $L=\{a^ib^ic^i : i>0\}$ n'est pas algébrique

  - $L_1=\{a^ib^ic^j : i>0,\ j>0\}$ est algébrique
    - Concaténation de $L=\{a^ib^i : i>0\}$ et $M=\{c^j : j>0\}$
  - $L_2=\{a^ib^jc^j : i>0,\ j>0\}$ est algébrique
    - Concaténation de $N=\{a^i : i>0\}$ et $P=\{b^jc^j : j>0\}$
  - $L_1 \cap L_2=L$ n 'est pas algébrique
- **Remarque** :On ne peut pas dire que l'intersection de deux langages algébriques n'est jamais algébrique
- Il suffit de prendre L=M avec L et M algébriques pour que L∩M=L soit algébrique

Corollaire

- On en déduit que les langages algébriques ne sont pas clos par complémentation
- Sinon, par les lois de Morgan, on aurait

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

et les algébriques seraient clos par intersection

Intersection avec rationnels

- **Théorème**: Si L est algébrique et M rationnel, alors L∩M est algébrique

- L'idée est de faire fonctionner en parallèle un AP pour L et un AF pour M

Intersection avec rationnels

- **Construction** :
  - On a $A = (Q_1, \Sigma, \Gamma, \delta_1, q_1, Z, F_1)$ un AP qui accepte L par EF
  - Et $B = (Q_2, \Sigma, \delta_2, q_2, F_2)$ un AF qui accepte M
  - On construit un automate à pile produit

$$(Q_1 \times Q_2, \Sigma, \Gamma, \delta, (q_1, q_2), Z, F_1 \times F_2)$$

  - Transition normale : Si $\delta_1(q,a,X)=(q',\gamma)$ et $\delta_2(p,a)=p'$, alors
    - $\delta((q,p),a,X)=((q',p'),\gamma)$
  - ε-transition : Si $\delta_1(q, \varepsilon, X)=(q',\gamma)$, on ne s'intéresse pas à M:
    - $\delta((q,p),\varepsilon,X)=((q',p),\gamma)$

Utilité de ces propriétés

- Comme pour les langages rationnels, les propriétés de clôture sont utiles
  - Pour construire des langages algébriques
  - Pour montrer qu'un langage donné n'est pas algébrique

Moralité

- permet de simplifier les preuves de non-algébricité de certains langages
- Par exemple pour un langage qui conduit à une « explosion » de cas à considérer :

$$D=\{ww : w \in \{a,b\}^*\}$$

  - On suppose D algébrique
  - Soit $L=a^+b^+a^+b^+$ (un langage rationnel)
  - Alors, $D \cap L$ est algébrique
  - Mais $D \cap L = \{a^i b^j a^i b^j : i,j>0\}$ réputé non algébrique (analogue à $\{a^i b^j c^i d^j : i,j>0\}$)
  - Donc D n'est pas algébrique