Les grammaires

suite

Dérivation directe

- Un mot m de $(N \cup T)^*$ se dérive directement en un mot m' de $(N \cup T)^*$
  $(m \rightarrow m')$ si :
  - $m = uXv$ pour $X \in N$ et $u,v \in (N \cup T)^*$
  - s'il existe une production $X \rightarrow w$ dans R
  - $m' = uwv$ pour $u,v \in (N \cup T)^*$
- formalise le fait d'appliquer une fois une production en réécrivant un non terminal en accord avec une production ayant pour membre gauche ce non terminal
- **Exemple**: 3 dérivations directes pour la grammaire de règles $S \rightarrow \varepsilon | aSb$
  - $S \rightarrow aSb$
  - $aSb \rightarrow aaSbb$
  - $aaaSbbb \rightarrow aaabbb$

Dérivation en k étapes

- m, mot de $(N \cup T)^*$ se dérive en m', mot de $(N \cup T)^*$ ($m \rightarrow^* m'$) si
  - Il existe k un entier
  - $m_0, m_1, \ldots, m_k$ des mots de $(N \cup T)^*$ tels que
  - m_{i+1} se dérive directement de m_i , $0 \leq i < k$
  - $m_0 = m$ et $m_k = m'$
- formalise le fait d'appliquer successivement k productions

- **Exemple** : pour la grammaire dont la règle est $S \rightarrow \varepsilon | aSb$
  - $S \rightarrow^*$ aaaSbbb est une dérivation en 3 étapes
  - $S \rightarrow^*$ aaabbb est une dérivation en 4 étapes

Remarque

- Dans la définition rien ne dit que les mots qui se dérivent directement les uns des autres soient uniques

- Exemple : $S \rightarrow \varepsilon | aSb | ab$, ab a 2 dérivations différentes :
  - $S \rightarrow ab$ (directe)
  - $S \rightarrow aSb \rightarrow ab$ (indirecte)

Mots engendrés

- Les mots engendrés par une grammaire G=(N,T,R,S) sont les mots $m \in T^*$ (uniquement composés de symboles terminaux) qui peuvent être dérivés depuis l'axiome :

$$S \rightarrow^* m$$

- Exemple : pour la grammaire de règle $S \rightarrow \varepsilon | aSb$

  aaabbb est un mot engendré par la grammaire

Langage engendré

- La grammaire G engendre un langage

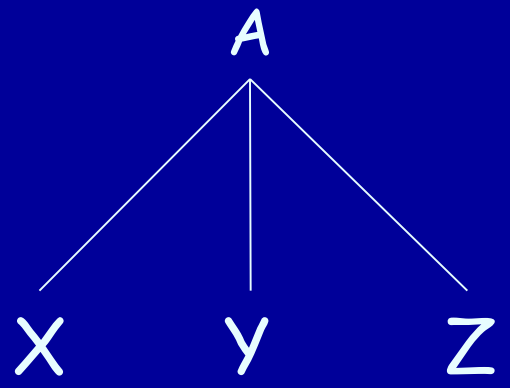
$$L(G)=\{m\in T^*: S \rightarrow^* m\}$$

- ensemble des mots engendrés par G en dérivant l'axiome
- Pour les grammaires de la forme $X\rightarrow\alpha$, $X \in N$ et $\alpha\in(N\cup T)^*$ (grammaire algébrique), on engendre des langages algébriques.
- Exemple : $G=(N,T,R,S)$
  - $N=\{S\}$
  - $T=\{a,b\}$
  - $R=\{S\rightarrow \varepsilon, S\rightarrow aSb\}$
- Définit une grammaire du langage $\{a^nb^n:n\geq 0\}$

Arbre syntaxique

- Un arbre syntaxique illustre graphiquement la manière dont l'axiome se dérive en une chaîne du langage

- Si le non-terminal A définit la production $A \rightarrow XYZ$, un arbre syntaxique possède un nœud interne et trois fils étiquetés X, Y et Z de gauche à droite



Arbre syntaxique (définition)

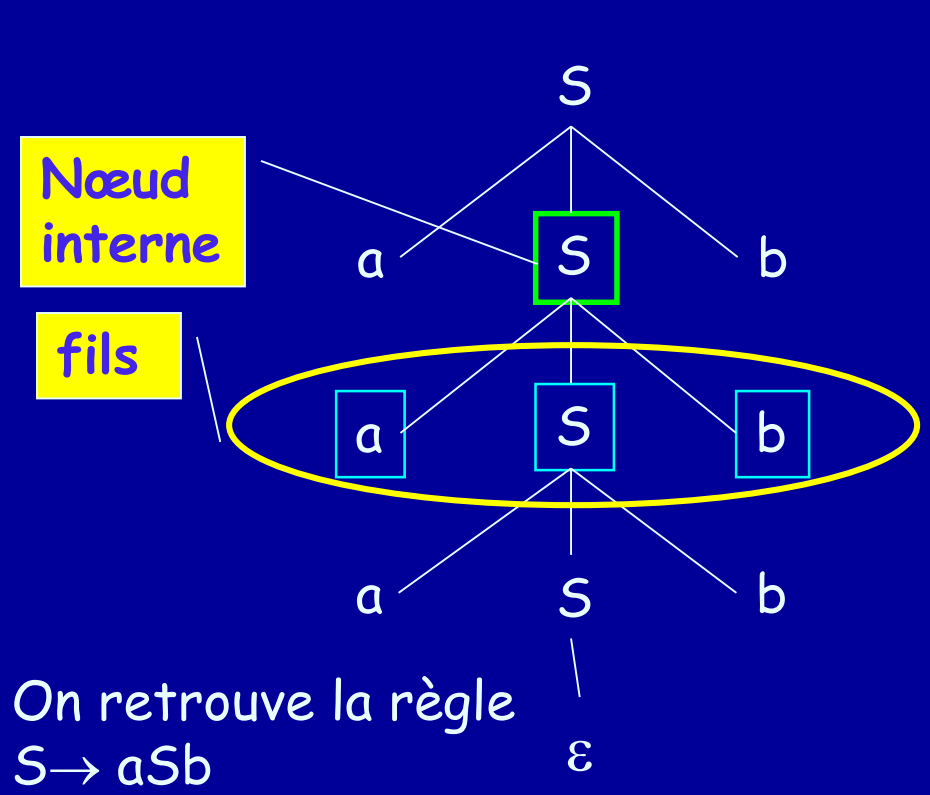
- La racine est l 'axiome
- Chaque feuille est soit ε soit un terminal
- Chaque nœud interne est un non-terminal
- Si A est l'étiquette d 'un nœud interne de fils (de gauche à droite) $X_1, X_2, \ldots, X_n$ alors

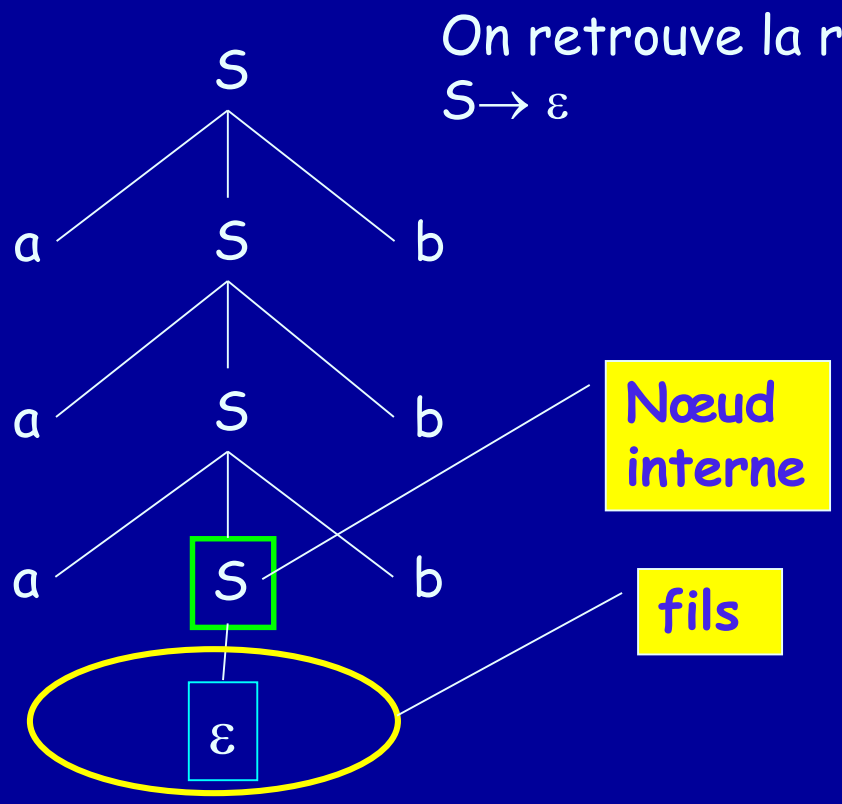
$$A \rightarrow X_1 X_2 \ldots X_n$$

est une production de la grammaire

Arbre syntaxique: $\{a^n b^n : n \geq 0\}$

- $S \rightarrow \varepsilon | aSb$, arbre syntaxique pour aaabbb



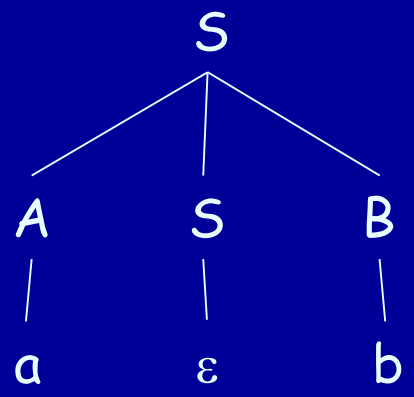


Avec l'arbre syntaxique d'un mot dont la dérivation contient toutes les règles, on peut retrouver l'ensemble des règles de la grammaire

Arbre & dérivations

- Dérivations → arbre = représentation graphique de la dérivation
- Dans la dérivation ainsi obtenue, on fait disparaître les choix de l'ordre d'application des règles

- Pour la grammaire
  1. S→ASB
  2. S → ε
  3. A →a
  4. B →b
- On peut appliquer les règles selon différents ordres



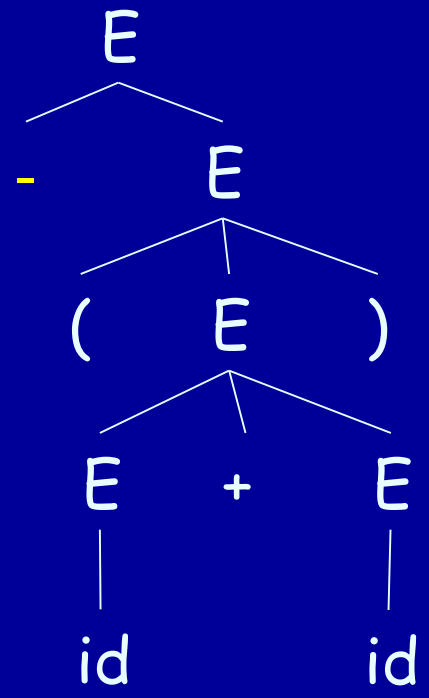
- S→ASB →aSB →aB →ab
  - 1;3;2;4

Ou

- S→ASB →AB →Ab →ab
  - 1;2;4;3

Dérivation arbre

- A partir d'une dérivation, on peut construire l'arbre syntaxique :
- Grammaire
  - $E \rightarrow E+E|E*E|(E)|-E|id$
- Mot engendré
  - -(id+id)



$E \rightarrow -E \rightarrow -(E) \rightarrow -(E+E) \rightarrow -(id+E) \rightarrow -(id+id)$

Analyse

Données : G une grammaire et m un mot

Calcul : trouver (s'il en existe) les dérivations de G qui engendrent m

Grammaire
- E → E+E|E*E|(E)|-E|id

- Mot engendré
  - -(id+id)

- -(id+id)
- -(id+id)
- -(id+id)
- -(id+id)

E→-E

E→(E)

E→E+E

E→id

E→-E→-(E) →-(E+E) →-(id+E) →-(id+id)

Attention

- Pour une grammaire donnée, on peut avoir plus d'une dérivation qui engendre un même mot
  - Soit avec des arbres syntaxiques différents
  - Soit au sein du même arbre syntaxique

- On passe ainsi de l'arbre syntaxique aux dérivations

Exemple

- Grammaire

  $E \rightarrow E+E|E*E|(E)|-E|id$

- Mot engendré

  -(id+id)

$E \rightarrow -E \rightarrow -(E) \rightarrow -(E+E) \rightarrow -(id+E) \rightarrow -(id+id)$

$E \rightarrow -E \rightarrow -(E) \rightarrow -(E+E) \rightarrow -(E+id) \rightarrow -(id+id)$

deux dérivations pour un même arbre syntaxique

Exemple

- Grammaire

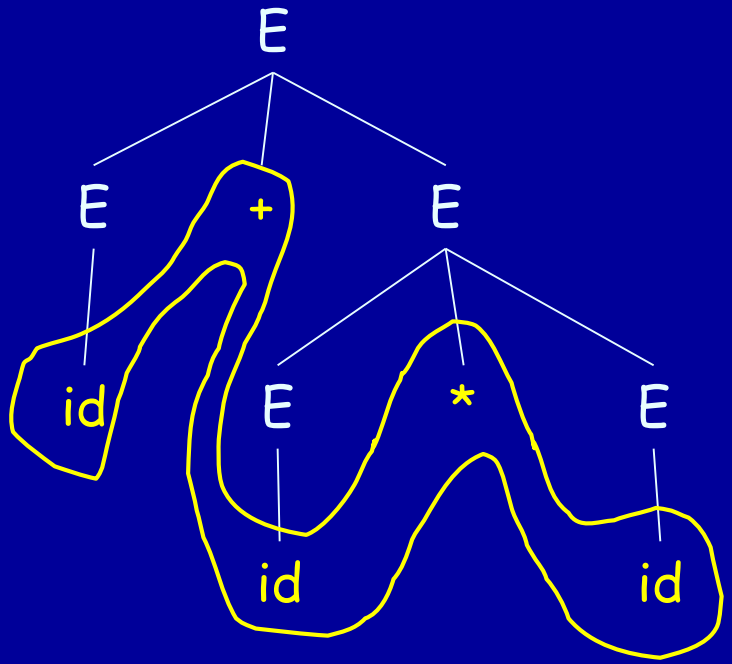
$E \rightarrow E+E|E*E|(E)|-E|id$

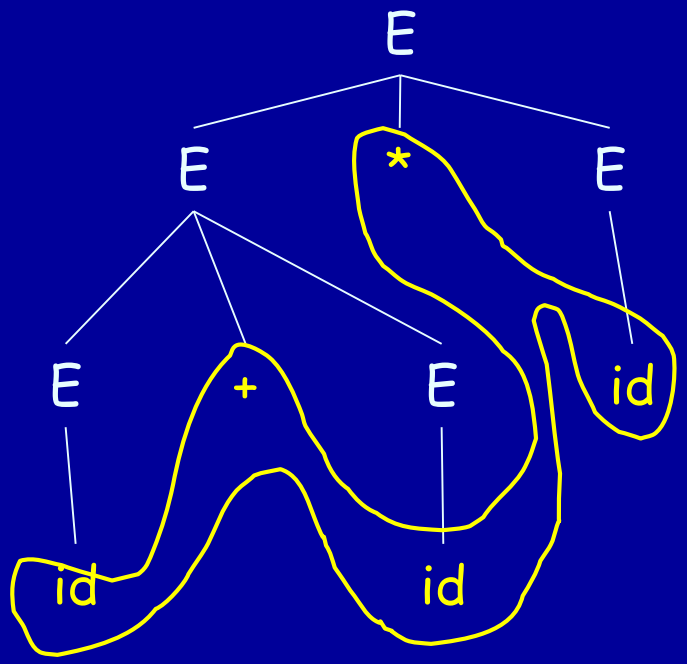
- Mot engendré id+id*id

Deux arbres syntaxiques différents engendrant le même mot

$E \rightarrow E+E \rightarrow id+E \rightarrow id+E*E \rightarrow id+id*E \rightarrow id+id*id$

$E \rightarrow E*E \rightarrow E*id \rightarrow E+E*id \rightarrow id+E*id \rightarrow id+id*id$





Dérivations gauches et droites

- Comme on peut trouver plusieurs dérivations à partir d 'un même arbre syntaxique, on parle alors

- De <u>dérivation gauche</u>
- Chaque étape d'une dérivation gauche s 'écrit

$$wA\gamma \rightarrow w\delta\gamma$$

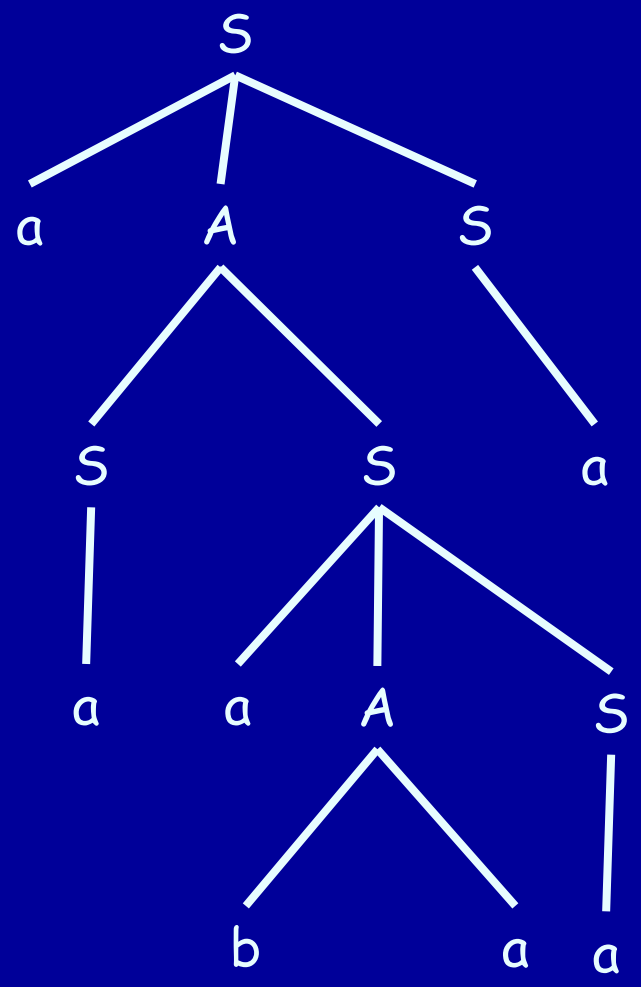
- De <u>dérivation droite</u>
- Chaque étape d'une dérivation droite s'écrit

$$\gamma Aw \rightarrow \gamma\delta w$$

- Pour w un mot formé de terminaux
- γ une chaîne de symboles grammaticaux (de $(N \cup T)^*$)
- et $A \rightarrow \delta$ est la production utilisée

Dérivation gauche resp droite

- $S \rightarrow aAS|a$
- $A \rightarrow SbA|SS|ba$
- $S \rightarrow aAS$
  $\rightarrow aSSS$
  $\rightarrow aaSS$
  $\rightarrow aaaASS$
  $\rightarrow aaabaSS$
  $\rightarrow aaabaaS$
  $\rightarrow aaabaaa$



- $S \rightarrow aAS$
  $\rightarrow aAa$
  $\rightarrow aSSa$
  $\rightarrow aSaASa$
  $\rightarrow aSaAaa$
  $\rightarrow aSabaaa$
  $\rightarrow aaabaaa$

Ambiguïté

- Problème
  - G une grammaire
  - G est-elle ambiguë ?
- Pour le résoudre, il suffit de trouver un mot qui admet au moins deux dérivations gauches (resp. droites) différentes
- Dans le contexte de la compilation,
  - Soit on essaye d'éviter les grammaires ambiguës,
  - Soit on ajoute des règles pour résoudre les problèmes de conflit liés à l'ambiguïté de la grammaire.
- Pour qu'il y ait unicité de l'analyse

Langages algébriques & grammaires

Rationnels et grammaires linéaires

Questions :

- Peut on trouver des grammaires qui engendrent des langages rationnels ?

- Est-ce que tout langage rationnel peut être engendré par une grammaire ?

Rationnels et grammaires linéaires

- Une grammaire algébrique est dite linéaire (droite) si toutes ses règles de dérivation sont de la forme

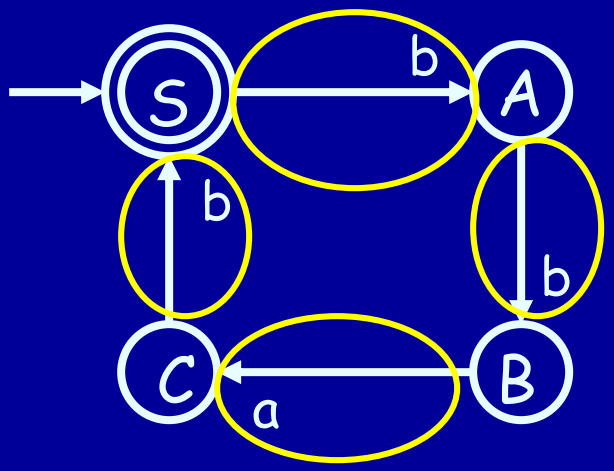
$$X \rightarrow aY$$
$$X \rightarrow a$$
$$X \rightarrow \varepsilon$$

Avec $X, Y \in N$ et $a \in T$

Si L est rationnel, L est engendré par une grammaire linéaire.

Exemple

L=(bbab)* rationnel $\Rightarrow$il existe AFD qui le reconnaît



T=Σ ={a,b}

N=Q ={S ,A,B,C}

S (axiome définie par l'état initial)

S$\rightarrow$bA

A$\rightarrow$bB

B$\rightarrow$aC

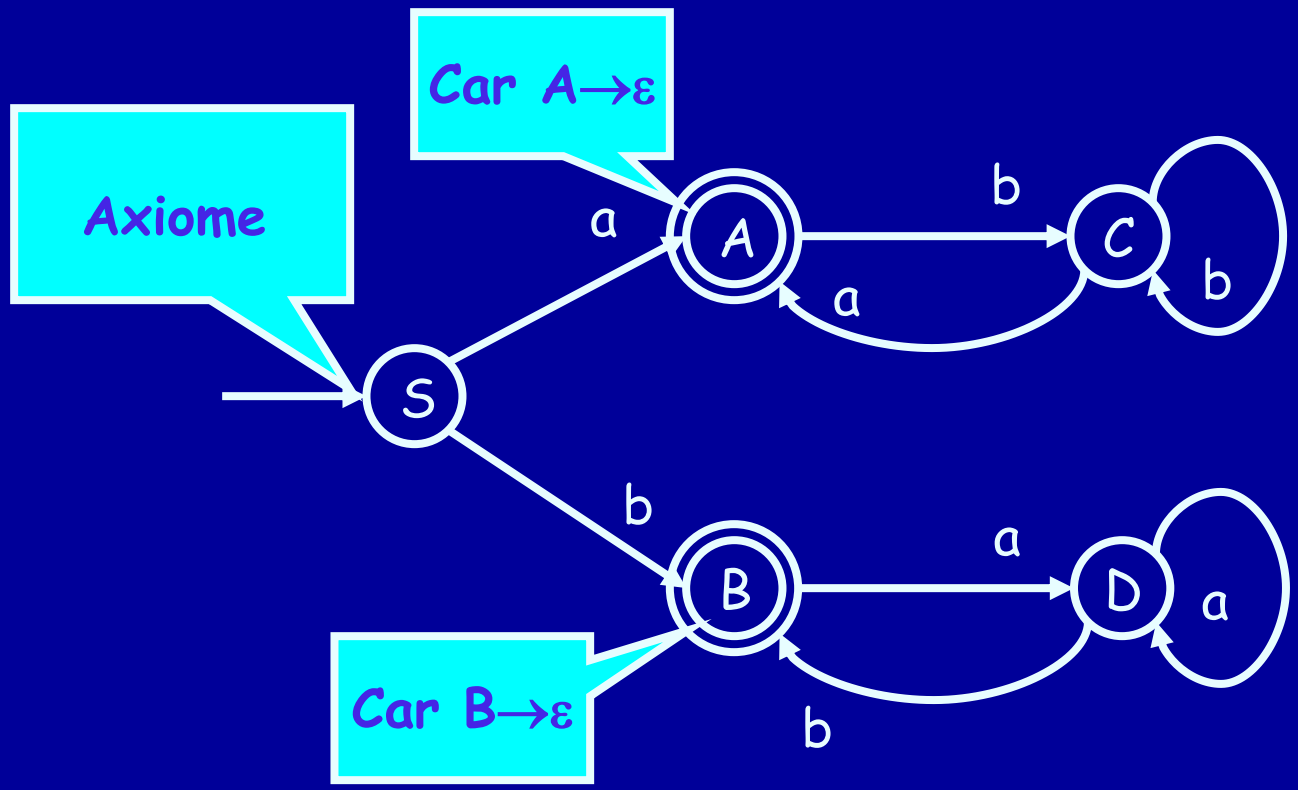
C$\rightarrow$bS

S$\rightarrow\varepsilon$

Car état de reconnaissance

Réciproque, exemple

- S→aA|bB; A→bC|ε; B→aD|ε; C→bC|aA; D→aD|bB



- ici on a retrouvé un AFD correspondant à la grammaire linéaire.
- Est-ce vrai pour toute grammaire linéaire ?

Réciproque

Si un langage est engendré par une grammaire linéaire alors il est rationnel

- L est engendré par G=(N,T,S,R); on construit un AFND $A=(\Sigma = T, Q = N \cup \{f\}, \delta, i = S, T=\{f\})$ tel que L(A)=L(G). La fonction non déterministe δ est
  - à chaque règle de la forme $X \rightarrow wY$ on introduit une transition de la forme $\delta(X,w)=Y$
  - à chaque règle de la forme $X \rightarrow w$ une transition $\delta(X,w)=f$
- il faudrait montrer par récurrence sur la longueur des dérivations que L(G)=L(A) (raisonnement analogue au précédent).

Remarques & conclusion

- Observons que dans la démonstration on n'a pas de règle qui donne le mot vide.

- On verra qu'il est toujours possible de trouver une grammaire sans ε-production qui engendre le même langage (sauf si ε appartient au langage).

Un langage est rationnel ssi il est engendré par une grammaire linéaire droite

Les grammaires linéaires gauches

- linéaire droite: productions de la forme

  $X \rightarrow aY|a|\varepsilon$ avec $X,Y \in N$ et $a \in T$

- linéaire gauche: productions de la forme

  $X \rightarrow Ya|a|\varepsilon$ avec $X,Y \in N$ et $a \in T$

Intérêt des grammaires linéaires

- Soit la grammaire (qui n'est pas linéaire)
  - Exp $\rightarrow$var|cst
  - Exp $\rightarrow$Exp * Exp| Exp + Exp

- On peut lui associer une expression rationnelle

  (var|cst) [(+|*) (var|cst)]*

  Et trouver une grammaire linéaire équivalente

décimaux JAVA BNF

- DecimalNumeral: 0|NonZeroDigit [Digits] N
- Digits:Digit|Digits Digit D
- Digit: 0| NonZeroDigit A
- NonZeroDigit: one of 1 2 3 4 5 6 7 8 9 C

- $C=\{1,2,3,4,5,6,7,8,9\}$ $A=C\cup\{0\}$
- $D=A^+$
- $N=0+CD+C=0+CA^*$

DecimalNumeral=0+(1+2+3+4+5+6+7+8+9)(0+1+2+3+4+5+6+7+8+9)*

Rationnels et compilation

- Si une partie des langages informatiques est rationnelle (et cela est fort utile pour la compilation), ce n'est hélas pas toujours le cas
- L'exemple le plus simple d'une partie de langage informatique non rationnelle est celui des mots de Dyck :

$$G=\langle N=\{S\},T=\{(,)\},R=\{S\rightarrow(S)|SS|\varepsilon\},S\rangle;\ L(G)=D$$

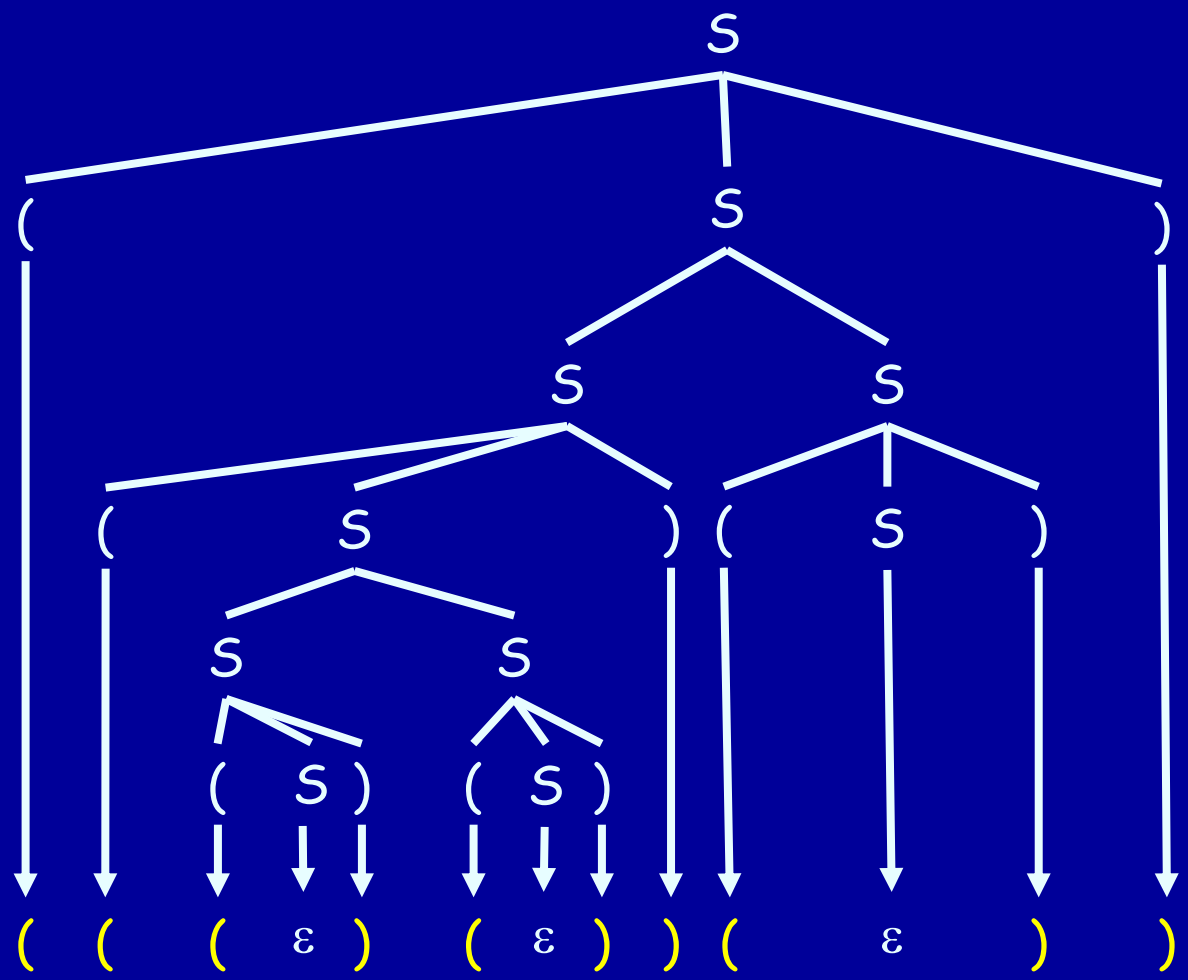
- D n'est pas rationnel :
  - Il suffit de considérer $D\cap(^*)^* = (^n)^n$
  - $(^*)^*$ est un langage rationnel
  - $D\cap(^*)^*$ devrait être rationnel (propriétés de clôture)
  - Or $(^n)^n$ n'est pas rationnel (c'est a^nb^n)
  - Donc D n'est pas rationnel

Exemple d'EBP*

*Expression Bien Parenthésé

R={S→(S)|SS|ε}

((()())())



Simplifications de grammaires

Nouvelle notion

- On dit que deux grammaires sont équivalentes si elles engendrent le même langage

$$G \approx G' \Leftrightarrow L(G)=L(G')$$

Exemple : $\{a^n b^n : n \geq 0\}$

- engendré

soit par

$$S \rightarrow aSb|\varepsilon$$

soit par

$$S \rightarrow aSb|ab\ |\varepsilon$$

Motivation de la simplification

- Il s'agit de « nettoyer » les grammaires pour en retirer tout ce qui n'est pas strictement indispensable à la génération des mots du langage :
  - Retirer les variables et règles inutiles
- on peut ensuite mettre les grammaires sous des formes standard simples :

Les formes normales

Idées

- Comment modifier les grammaires sans pour autant en restreindre l'« expressivité »?
- Un langage algébrique non vide L peut être engendré par une grammaire G vérifiant :
  - Chaque variable doit mener à la génération d'un mot de L **(variable productif)**
  - Chaque variable doit servir à quelque chose, donc on doit pouvoir le retrouver lors d'une dérivation **(variable accessible)**

Supprimer les improductifs

- $X \in N$ est productif s'il existe $w \in T^*$ tq $X \rightarrow^* w$
- Si X n 'est pas productif, cela signifie qu'on peut supprimer X sans retirer de mots au langage engendré par la grammaire.
- On construit inductivement P, l'ensemble des variables productives :
  - Base : $P_0 = \varnothing$
  - Règle : $P_{i+1} = \{X \in N : X \rightarrow \alpha, \alpha \in (T \cup P_i)^*\}$

  On s'arrête lorsque $P_{i+1} = P_i = P$

Étant donnée $G=(N,T,S,R)$ t.q. $L(G) \neq \varnothing$, on peut trouver une grammaire équivalente $G'=(N',T,S,R')$ sans symboles improductifs.

Exemple

- Soit la grammaire
  - $S \rightarrow AB|a$, $A \rightarrow a$, $B \rightarrow BA$

$P_0 = \varnothing$

$P1=\{X \in N : X \rightarrow \alpha, \alpha \in T^*\}=\{S,A\}$

$P2=\{X \in N : X \rightarrow \alpha, \alpha \in (T \cup \{S,A\})^*\}=\{S,A\}$

P1 =P2 . On en déduit que B est improductif

On supprime les règles contenant B : i.e. $S \rightarrow AB$ et $B \rightarrow BA$ pour obtenir la grammaire équivalente $S \rightarrow a$, $A \rightarrow a$

Retour à l'exemple

- Pour la grammaire

$$S \rightarrow AB|a,\ A \rightarrow a,\ B \rightarrow BA$$

- En supprimant les symboles improductifs on a obtenu la grammaire équivalente

$$S \rightarrow a,\ A \rightarrow a$$

- Mais à quoi sert la production $A \rightarrow a$?

Supprimer les inaccessibles

- $X \in N$ est accessible s'il existe α et $\beta \in (N \cup T)^*$ tq $S \rightarrow^* \alpha X \beta$
- Si X n'est pas accessible, cela signifie qu'on peut supprimer X sans retirer de mots au langage engendré par la grammaire.
- Pour trouver les variables accessibles, il suffit de parcourir les règles en partant de l'axiome (cailloutage)

Étant donnée G=(N,T,S,R), on peut trouver une grammaire équivalente G'=(N',T,S,R') sans symbole inaccessible.

Exemple

- S →aAb|a
- A →aAC|b

  B →d
- C →aSbS|aba

Variables accessibles : {S,A,C}

Nettoyage de grammaires

- En appliquant les deux algorithmes précédents, on peut transformer une grammaire algébrique en une grammaire équivalente qui ne contient pas de symbole inutile (improductif ou inaccessible).
- Observons que si on retire tout d'abord les inaccessibles puis les improductifs, on ne retire pas forcément l'ensemble des symboles inutiles.
- **Ordre d'application** :
  1. Retirer les improductifs
  2. Retirer les inaccessibles

Exemple du mauvais ordre

- S →aAb|bAB|a
- A →aAC
- B →d
- C →aSbS|aba

On retire tout d'abord les inaccessibles

Tous les symboles non terminaux sont accessibles

Exemple du mauvais ordre

$S \rightarrow aAb|bAB|a$

$A \rightarrow aAC$

$B \rightarrow d$

$C \rightarrow aSbS|aba$

inaccessibles ! $\begin{cases} B \rightarrow d \\ C \rightarrow aSbS|aba \end{cases}$

$S \rightarrow a$

équivalente

On retire tous les symboles improductifs :

- $P_0 = \varnothing$
- $P_1 = \{X \in N : N \rightarrow a,\ a \in T\} = \{S, B, C\}$
- $P_2 = \{X \in N : N \rightarrow \alpha,\ \alpha \in (\{S, B, C\} \cup T)^*\} = \{S, B, C\} = P_1$
- A est donc improductif