Chapitre 6
Grammaire Algébrique (à contexte libre)

Rappel sur les grammaires et langages algébrique

- **Définition d'une grammaire de type 2 :** Une grammaire G=(VT, VN, S, R) est dite de à contexte libre (Algébrique ou de type 2) si et seulement si toutes ses règles de production sont sous la forme : $A \longrightarrow B$ avec $A \in VN$ et $B \in (VT \cup VN)^*$.
- **Définition des langages de type2 (hors contexte ou algébrique)** : Ce sont les langages qui peuvent être définis par des grammaires de type 2.
- **Remarque**: L'ensemble des langages réguliers est inclus dans l'ensemble des langages algébriques.

L’ arbre syntaxique

- Vue l’utilisation d’un seul symbole non terminal à gauche des règles de production dans les grammaires à contexte libre, il est toujours possible de construire un arbre de dérivation pour un mot généré.
- **Définition**

Soit la grammaire G=(VT, VN, S, R) et soit $\omega \in L(G)$. Un arbre syntaxique associé à ω est construit tel-que :

- La racine de l’arbre est étiquetée par l’axiome;
- Les noeuds intermédiaires contiennent des non terminaux;
- Les feuilles sont des terminaux.

La lecture de gauche à droite des feuilles de l’arbre reconstitue le mot auquel l’arbre est associé

Dérivation gauche (droite)

Dérivation gauche: (resp. Droite)

Soit G<VT, VN, S, R> une grammaire algébrique, on dit qu'un mot w s'obtient par dérivation gauche (resp. droite) s'il est obtenu à partir de l'axiome en remplaçant au cours des dérivations toujours le non-terminal le plus à gauche (resp. à droite).

Exemple :

S → aAS | a

A →ba

Soit w=abaabaa

Dérivation gauche : $S \vdash aAS \vdash abaS \vdash abaaAS \vdash abaabaS \vdash abaabaa$

Dérivation droite : $S \vdash aAS \vdash aAaAS \vdash aAaAa \vdash aAabaa \vdash abaabaa$

Notion d'ambiguïté

- **Définition d'un mot ambigu** : Un mot ω est dit ambigu si et seulement s'il existe deux arbres de dérivation différents qui lui sont associés.

- **Définition d'une grammaire ambiguë** : Une grammaire G est dite ambiguë si et seulement s'il existe au moins un mot ambigu appartenant à L(G).

Ambiguïté

Définition :

Soit G<X,V,P,S> une grammaire algébrique, on dit qu'un mot w est ambiguë ssi il existe pour ce mot plus qu'une dérivation gauche (resp. plus d'une dérivation droite).

Exemple :

$$E \rightarrow E+E \mid E-E \mid E\times E \mid E\div E \mid (E) \mid ID$$

$$ID \rightarrow x \mid y$$

Appliquez pour le mot $\mathbf{w = x \times x + y}$

Exemple 1

- Soit G la grammaire qui possède les règles de production suivantes :

$$S \rightarrow S \wedge S | S \vee S | S \Rightarrow S | S \Leftrightarrow S | \neg S \ | q | p$$

Question : Montrer que G est ambiguë.

Réponse : Le mot $p \wedge q \Leftrightarrow p$ est ambigu, car il existe deux dérivations différentes qui nous permettent de l'atteindre, si nous numérotons les règles de 1 à 7 alors on aura :

$$S \vdash^{(4)} S \Leftrightarrow S \vdash^{(1)} S \wedge S \Leftrightarrow S \vdash^{(7)} p \wedge S \Leftrightarrow S \vdash^{(6)} p \wedge q \Leftrightarrow S \vdash^{(7)} p \wedge q \Leftrightarrow p.$$

$$S \vdash^{(1)} S \wedge S \vdash^{(6)} p \wedge S \vdash^{(4)} p \wedge S \Leftrightarrow S \vdash^{(6)} p \wedge q \Leftrightarrow S \vdash^{(7)} p \wedge q \Leftrightarrow p.$$

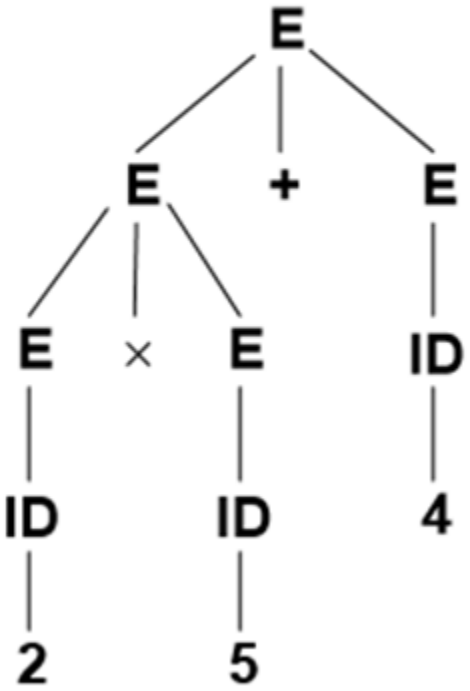
Par conséquent la grammaire G est ambiguë.

Exemple 2

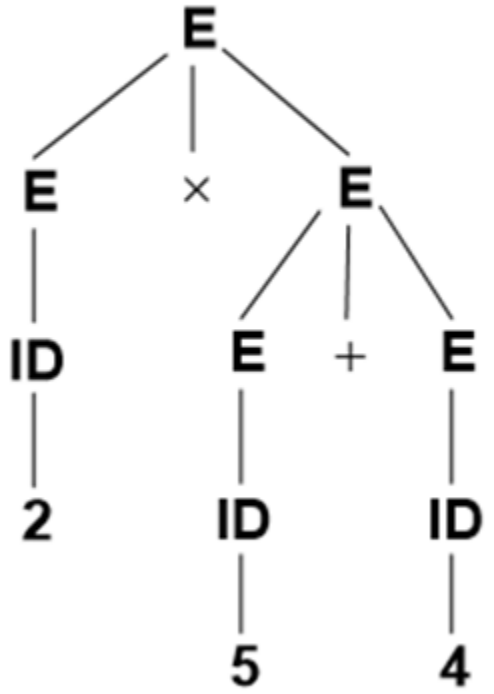
w=2×5+4

$E \rightarrow E+E \mid E-E \mid E\times E \mid E\div E \mid (E) \mid ID$

$ID \rightarrow 1 \mid 2 \mid \ldots \mid 9$



10 + 4 = 14



2 × 9 = 18

Définitions

Non terminal productif et improductif

- Un non terminal A∈VN est dit productif si et seulement si ∃ω ∈ VT* tel-que $A \vdash^* \omega$.
- Un non terminal A∈VN est dit improductif si et seulement si

∀ω ∈VT*, il n'existe aucune dérivation indirecte tel-que $A \vdash^* \omega$.

Non terminal accessible et inaccessible

- Un non terminal A∈VN est dit accessible si et seulement si

∃α ∈(VT ∪VN)* tel-que $S \vdash^* \alpha$ et A apparaît dans α;

- Un non terminal A∈VN est dit inaccessible si et seulement si ∀α ∈ (VT ∪VN)*, si $S \vdash^* \alpha$ alors A n'apparaît pas dans α.

Définition

- **Grammaire réduite**

Une grammaire est dite réduite si et seulement si tous les non terminaux de ses règles de production sont atteignables (accessibles) et productifs

Remarque

Les règles de production qui contiennent des non terminaux improductifs ou inaccessibles sont inutiles et peuvent être supprimées sans aucune influence sur le langage généré par la grammaire

Production unitaire

- On appelle production unitaire toute règle de production sous la forme $A \longrightarrow B$ où $A, B \in VN$.
- **Remarque** : Pour supprimer la règle unitaire $A \longrightarrow B$, il suffit de rajouter aux règles de production de A toutes les productions de B.Cette suppression peut faire apparaître d'autres productions unitaires, c'est pour cela qu'il faut appliquer un algorithme récursif

Définition

- **Grammaire propre**

Une grammaire est dite propre si et seulement si :

- Elle est réduite;
- Elle ne contient pas de productions unitaires;
- Il n ya que l'axiome qui peut générer ε, avec la condition qu'il n'apparaisse dans aucun membre droit des règles.

- Pour éliminer les ε-productions, il faut déterminer d'abord l'ensemble des non terminaux dérivables en ε (directement ou indirectement); ensuite,on modifie les productions contenant ces non terminaux, de sorte à remplacer dans toutes les parties gauches des productions les symboles annulables par le mot vide, de toutes les manières possibles.

Exemples

Exemple 1 :

1) $S \rightarrow aSb \mid ab$
2) $S \rightarrow aAb \mid \varepsilon$; $A \rightarrow aAb \mid ab$

G est ε-libre.

Exemple 2 :

$S \rightarrow aSb \mid \varepsilon$; G est non ε-libre.

$S' \rightarrow S \mid \varepsilon$

$S \rightarrow aSb \mid ab$

Exercice :

$S \rightarrow AbB$
$A \rightarrow aAb \mid \varepsilon$
$B \rightarrow Ba \mid \varepsilon$

$\Rightarrow$

$S \rightarrow b \mid bB \mid Ab \mid AbB$
$A \rightarrow aAb \mid ab$
$B \rightarrow Ba \mid a$

Exemple

- Soit G la grammaire définie par les règles de production suivantes :

$$S \rightarrow AB|EaE \qquad E \rightarrow D$$
$$A \rightarrow Aa|aB \qquad D \rightarrow dD|\varepsilon$$
$$B \rightarrow bB|aA$$
$$C \rightarrow AB|aS$$

1. Trouver le langage généré par G.
2. Transformer G en une grammaire réduite.
3. Transformer G en une grammaire propre.
4. Vérifier le langage trouvé dans la question 1

Solution

1. $L(G)=\{d^i a d^j / i,j \geq 0\}$.
2. Le non terminal C n'est pas atteignable, A et B ne sont pas productifs, donc on enlève les règles contenant A, B ou C et on aura la grammaire suivante :

   $S \rightarrow EaE$

   $E \rightarrow D$

   $D \rightarrow dD \mid \varepsilon$

3. La grammaire possède une production unitaire $E \rightarrow D$, on l'enlève et on remplace tous les E par des D et on aura la grammaire suivante :

   $S \rightarrow DaD$

   $D \rightarrow dD \mid \varepsilon$

La grammaire n'est toujours pas propre à cause de la règle $D \rightarrow \varepsilon$, donc on l'élimine et pour chaque D qui apparaît à droite d'une règle on crée une autre règle, on obtient la grammaire propre suivante :

$S \rightarrow DaD \mid aD \mid Da \mid a$

$D \rightarrow dD \mid d$

4. Le langage trouvé est le bon,mais plus facile à trouver avec la grammaire propre.

Forme Normale de Chomsky

• Une grammaire G=(VT, VN, S, R) est dite sous forme normale de Chomsky (FNC) si et seulement si toutes ses règles de production sont sous la forme

$A \rightarrow BC$ où $A \rightarrow a$ avec $A,B,C \in VN$ et $a \in VT$.

•**Proposition** :
Pour toute grammaire algébrique, il existe une grammaire équivalente sous forme normale de Chomsky.

•L'intérêt pratique de la FNC est que les arbres de dérivations sont des arbres binaires, ce qui facilite l'application des algorithmes d'exploration des arbres

Forme Normale de Chomsky

Pour obtenir une grammaire sous forme normale de Chomsky équivalente à une grammaire algébrique G, il faut :

1. Transformer la grammaire en une grammaire propre ;
2. Pour chaque terminal a, introduire le non terminal C_a, puis rajouter la règle $C_a \to a$;
3. Pour chaque règle $A \to \alpha$, avec $|\alpha| \geq 2$, on remplace chaque terminal par le non terminal qui lui est associé ;
4. Pour chaque règle $A \to \beta$, avec $|\beta| \geq 3$, $(\beta = \beta_1\beta_2...\beta_n)$, créer les non terminaux D_i, puis remplacer la règle concernée par les règles suivantes : $A \to \beta_1 D_1$, $D_1 \to \beta_2 D_2, ...D_{n-2} \to B_{n-1}B_n$. où $D_i = B_{i+1}B_{i+2}...B_n$, avec i variant de 1 jusqu'à $n-2$.

Exemple

$$
\begin{aligned}
&S \to Ba \mid Ab \\
&A \to Sa \mid AAb \mid a \\
&B \to Sb \mid BBa \mid b
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
&S \to BA_1 \mid AB_1 \\
&A \to SA_1 \mid AAB_1 \mid A_1 \\
&B \to SB_1 \mid BBA_1 \mid B_1 \\
&A_1 \to a \\
&B_1 \to b
\end{aligned}
$$

$$
\Rightarrow \quad
\begin{aligned}
&S \to BA_1 \mid AB_1 \\
&A \to SA_1 \mid AX_1 \mid a \\
&B \to SB_1 \mid BX_2 \mid b \\
&X_1 \to AB_1 \\
&X_2 \to BA_1 \\
&A_1 \to a \\
&B_1 \to b
\end{aligned}
$$

Forme Normale de Greibach :

•Une grammaire algébrique est sous forme normale de Greibach (FNG) si et seulement si toutes ses règles de production sont sous la forme

$$\mathbf{A \rightarrow x\alpha \text{ ou } S \rightarrow \varepsilon, \text{ avec } x \in VT, \alpha \in VN*}$$

et S est l'axiome.

•**Proposition** :

Pour toute grammaire algébrique G1, il existe une grammaire G2 sous forme normale de Greibach tel-que L(G2)= L(G1).

•L'intérêt pratique de la FNG est qu'à chaque dérivation, on détermine un préfixe de plus en plus long formé uniquement de symboles terminaux. Cela permet de construire des automates à piles à partir des grammaires plus aisément, et par conséquent des analyseurs syntaxiques sont facilement implémentables.

Construction de la FNG d'une grammaire

- **Entrée:** G réduite et propre
- **Sortie:** G' sous FNG

Soit G une grammaire algébrique, réduite et propre.

Définitions:

- Un symbole non terminal A est dit *récursif à gauche* si il existe au moins une règle de production dans P telle que

$$A \rightarrow A\alpha \ (A \in VN \ , \alpha \in (VN \cup VT)^*).$$

- Une grammaire est dite *récursive à gauche* si il existe au moins un non terminal récursif à gauche.

Algorithme d'élimination de la récursivité à gauche

- Appliquer la transformation suivante pour toutes les règles de production jusqu'à éliminer toutes les récursivités à gauche directes et cachée.

Si $(A \rightarrow A\alpha_i / \beta_i) \in P$ avec $\alpha_i , \beta_i , \phi \in (VN \cup VT)^*$ et $\beta_i \neq A\phi$

Alors $(A \rightarrow \beta_i / \beta_i A' \text{ et } A' \rightarrow \alpha_i / \alpha_i A')$

Exemple : récursivité à gauche simple

La règle $\mathbf{A \rightarrow AAS / aS / b}$ est transformée en

$A \rightarrow aS / b / aSA' / bA'$

$A' \rightarrow AS / ASA'$

Exemple : récursivité à gauche cachée

- Soit la grammaire définit par l'ensemble P suivant :
- $S \rightarrow AA \ / \ a$
- $A \rightarrow SS \ / \ b$

Transformation:

Remplaçons S dans $A \rightarrow SS \ / \ b$, on a donc $A \rightarrow AAS \ / \ aS \ / \ b$ (transformée déjà)

$A \rightarrow aS \ / \ b \ / \ aSA' \ / \ bA'$

$A' \rightarrow AS \ / \ ASA'$

Donc la grammaire résultante (non récursive à gauche) est défini par l'ensemble P' suivant :

$S \rightarrow AA \ / \ a$
$A \rightarrow aS \ / \ b \ / \ aSA' \ / \ bA'$
$A' \rightarrow AS \ / \ ASA'$

Définition

- On dit que A < B si et seulement si

$$A \rightarrow B\alpha \; \forall \; A \text{ et } B \in V \; ; \; \alpha \in (X \cup V)^{*}.$$

- Obtention de la FNG:

a) Etablir l'ordre linéaire.

b) Substitutions des non terminaux.

Exemple

S → AA / a
A → aS / b / aSA' / bA'

A' → AS / ASA'

- On a S < A et A' < A donc cela signifie que la règle sur A est sous FNG.
- Substitutions :
  - De A dans S:

    S → aSA / bA / aSA'A / bA'A / a [F1]
  - De A dans A':

    A' → aSS / bS / aSA'S / bA'S / bSA' / aSA'SA' / bA'SAA' [F2]

Dans la grammaire FNG de G est G' tel que: **[F1] , A → aS / b / aSA' / bA' , [F2]**

Langages Algébriques

- **Proposition** : La classe des langages algébrique est fermée par rapport à l'union, la concaténation, le miroir et l'itération.

- **Remarque** : La classe des langages algébriques n'est pas fermée par rapport à l'intersection et le complément.

- **Théorème** : La classe des langages reconnus par les automates à pile est égale à la classe des langages engendrés par les grammaires algébriques.

Lemme de la double étoile (lemme d'Ogden)

- Pour tout langage algébrique L, il existe un entier k tel que pour tout mot $w \in L$ de longueur $|w| \geq k$, admet une factorisation en $w = \alpha\mu\beta\nu\delta$ avec $|\mu\nu| > 0$ et $|\mu\beta\nu| < k$ telle que:

$$\forall n \in N \; \alpha\mu^n\beta\nu^n\delta \in L$$

Exemple

Montrons que L=$\{a^n b^n c^n, n\geq 0\}$ n'est pas un langage algébrique.

Supposons que le langage L soit algébrique et w un mot quelconque de L.

$\exists$ n≥0 tel que w= $a^n b^n c^n$ et w peut se mettre sous la forme w=$\alpha u \beta v \gamma$ et la propriété $\alpha u^k \beta v^k \gamma \in L$,

$\forall$k≥0 doit être vérifiée.

Il suffit donc de trouver un k qui ne vérifie pas la propriété, suivant le découpage choisi.

Prenons $\alpha=a^n$; $u=b^n$; $\beta=c^n$; $v=\gamma=\varepsilon$

Formons $\alpha u^n \beta v^n \gamma$ pour k=2 par exemple: $\alpha u^2 \beta v^2 \gamma = a^n (b^n)^2 c^n \varepsilon\varepsilon = a^n b^{2n} c^n \notin L$.

Il y a une contradiction et ainsi l'hypothèse est fausse et donc le langage $\{a^n b^n c^n, n\geq 0\}$ n'est pas algébrique.