Grammaires et langages

Intro

- L'ensemble des mots est contenu dans un dictionnaire.
- L'ensemble des règles de combinaisons de mots constitue la grammaire.
- 1 phrase : correcte par rapport à une syntaxe.
- Sémantique : ensemble des connaissances sur l'ensemble des phrases.
- Pour les langages de programmation, on utilise des grammaires formelles (ou algébriques).

Grammaires formelles

Définitions :

- Soit l'ensemble non vide V de symboles appelé **vocabulaire** ou **alphabet**.
- On note **V*** l'ensemble V muni de la concaténation et de l'élément neutre ε.

ex : (a,b) ∈ V^2, ab ∈ V* ou aabb ∈ V*
- On appelle **mot** ou **chaîne** toute suite d'éléments de V.
- V* c'est l'ensemble de toutes les chaînes construites sur V y compris ε.
- $V_+ = V^* - \{ \varepsilon \}$
- On appelle **langage formel**, n'importe quel ensemble de chaînes. (= Toutes parties de V*)

Grammaires formelles

2 types de problèmes :

- la génération d'un langage
- déterminer si une chaîne quelconque de V* appartient ou non au langage (correction ou vérification syntaxique)→ algo qui vérifie ceci : **accepteur** ou **automate de reconnaissance**.

Notations : En général, on note
les éléments terminaux en minuscules,
les non terminaux en majuscules et
les chaînes en lettres grec.

Grammaires formelles

Définition d'une grammaire formelle ou algébrique :
quadruplet : $G = (V_N, V_T, P, S)$
V_N : vocabulaire non terminal ou auxiliaire ou de variables ou de notions.
V_T : vocabulaire terminal, sur lequel sont construites les chaînes.
P : ensemble des règles de grammaire de la forme : $\alpha \rightarrow \beta$ avec $\alpha \in V^+$ et $\beta \in V^*$ (pas de ε à gauche !).
S : élément de l'ensemble V_N, symbole de départ ou axiome.
$V_N \cap V_T = \emptyset$
$V_N \cup V_T = V$
Tous les éléments de V_N sont notés en majuscules,
V_T en minuscules
et les chaînes sont notées par des lettres grecques.

Grammaires formelles

<u>Exemple :</u> **$G = (V_N, V_T, P, S)$**
avec
$V_N = \{S,E,T,F\}$
$V_T = \{ a,(,),*,+\}$
P=

{S→E	1
E→E+T	2
E→T	3
T→T*F	4
T→F	5
F→(E)	6
F→a	7
}	

Les règles 2 et 3 peuvent s'écrire : E→ E+T | T

dérivation

$\alpha \rightarrow \beta$, α dérive β
Une dérivation c'est une séquence de chaînes dérivées.

D : $\alpha_0, \alpha_1, \ldots, \alpha_n$ avec $\alpha_0 = S$
$\alpha_{i-1} \rightarrow \alpha_i$
α_n : dérivation terminale ($\alpha_n \in V_T^*$)

dérivation

- Toute chaîne qu'on peut dériver de S est une **forme syntaxique.**
- Une **phrase** est une forme syntaxique qui n'est constituée que de symboles terminaux.
- Un langage L **engendré** par une grammaire G, L(G) , c'est l'ensemble des mots dont les symboles appartiennent à l'ensemble des symboles terminaux de la grammaire G.

$L(G) = \{ w \in V_T^* \ / \ S \rightarrow^* w \}$ (* : 0 ou plus , + : 1 ou plus)

L(G) : c'est l'ensemble de toutes les phrases .

forme syntaxique

Dans la pratique ou cherche **l'adéquation d'un langage à une grammaire pour 2 raisons** :

- parce qu'on a construit une grammaire pour reconnaître un langage
- parce qu'on veut expliciter le langage engendré par la grammaire

Grammaires

Classification de Chomsky (classification des grammaires) : soit la règle $\alpha \rightarrow \beta$

4 types de grammaires donc 4 types de langages :

- **type 0 :** pas de restriction sur la partie gauche et la partie droite de P
- **type 1 : grammaires contextuelles :** pour toutes les règles on vérifie que la longueur de β et supérieure ou égale à la longueur du mot α : $|\beta| \geq |\alpha|$
- **type 2 : grammaires hors contexte :** la partie gauche est un symbole non terminal, $\alpha \in V_N$ et $\beta \in V^*$
- **type 3 : grammaires régulières :** $A \rightarrow a$; $A \rightarrow aB$; $A \rightarrow Ba$

Type d'un Langage : Type de la grammaire de plus haut numéro générant le langage.

Grammaires - langages

- Les grammaires de type 3 et les langages réguliers qu'elles engendrent sont utilisés pour la génération des mots.
- Les grammaires de type 2 permettent de définir la syntaxe des langages de programmation

Arbre de dérivation

- Soit la grammaire G suivante :
- G = (Vn,Vt, P, S) avec
- Vn = {S, E, T, F}
- Vt = {a, +, *}
- S : l'axiome de G
- P = { S → E, E → E + T | T, T → T *F | F, F → (E) | a }

Arbre de dérivation

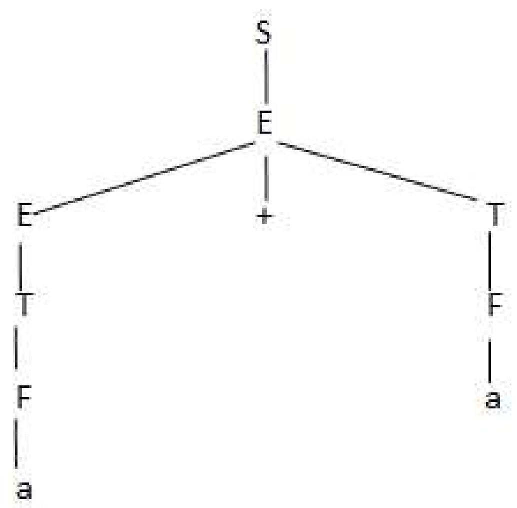
- Soit la dérivation suivante :

$S \rightarrow E \rightarrow E+T \rightarrow T+T \rightarrow F+T \rightarrow a+T \rightarrow a+F \rightarrow a+a$

1 2 3 5 7 5 7

Cette dérivation peut être représentée par l'arbre suivant :

Arbre de dérivation



Arbre de dérivation

Il existe 10 dérivations possibles, les plus importantes sont les dérivations **gauche** et **droite.**

gauche : on remplace toujours le symbole le plus à gauche (cf : exemple précédent)

droite : on remplace toujours le symbole le plus à droite

ambiguïté

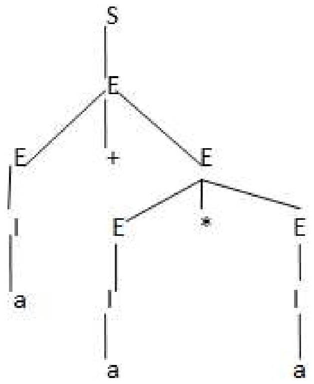
Une grammaire G est dite ambigue s'il existe dans le langage L une phrase qui a plusieurs arbres de dérivation distincts.

Exemple : **G = (V_N, V_T, P, S)**
avec
V_N = {S,E,I}
V_T = { a ,*,+}
P=
{ S→E 1
E→E+E |E*E|I 2
I→ a 3
}

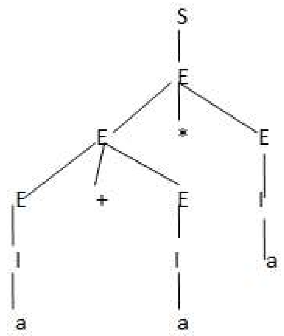
ambiguïté

a+a *a   donne 2 arbres de dérivations distincts

Arbre 1 :



Arbre 2 :



ambiguïté

On lève l'ambiguïté en introduisant un symbole T.

G = (V_N, V_T, P, S)
avec
V_N = {S,E,I,**T**}
V_T = { a ,*,+}
P=
{   S→E 1
E→E+T |T   2
T→T*I | I   3
I→a   4

}

Propriétés des grammaires

- Un symbole $x \in V$ est **improductif** ou **inutile** s'il n'existe pas de dérivation de la forme suivante :

$S \rightarrow^* \alpha\, x\, \beta \rightarrow^* \alpha\beta\sigma \in V_T^*$

- Un symbole $x \in V$ est **inaccessible** dans une grammaire G si x n'apparaît dans aucune chaîne de L(G).

- Une grammaire est dite réduite si elle ne comporte ni de symboles improductifs ni de symboles inaccessibles.

Propriétés (suite)

- Une grammaire G est dite ε libre si elle ne comporte pas de règle ε ou elle comporte une seule règle ε de la forme $S \rightarrow \varepsilon$ à condition que l'axiome S n'apparaisse dans aucune partie droite des règles de la grammaire.
- Une grammaire sans cycle est une grammaire hors contexte qui n'a pas de dérivation de la forme suivante :

$$A \overset{*}{\Rightarrow} A$$

- Une grammaire G hors contexte est propre si elle est sans cycle, ε libre et ne contient pas de symboles inutiles.

Propriétés (suite)

- Une grammaire G hors contexte est propre si elle est sans cycle, ε libre et ne contient pas de symboles inutiles.
- Si il existe des étapes de dérivation intermédiaires pour obtenir la récursivité, on dit de la grammaire qu'elle est récursive indirectement.

$$A \overset{+}{\Rightarrow} A$$

- L'auto-imbrication : $A \overset{+}{\Rightarrow} \alpha A \beta$
- Les grammaires régulières ne permettent pas l'auto-imbrication (pas de {}, pas de begin…end).

- Des algorithmes permettent d'éliminer la récursivité (certains analyseurs ne tolèrent pas la récursivité).

Les formes normales

Une grammaire est dite sous forme normale de Chomsky : $A,B,C \in V_N$ et $a \in V_T$

$A \rightarrow B\ C$

$A \rightarrow a$

Eventuellement : $S \rightarrow \varepsilon$

- La forme normale de Greibach :

$A \rightarrow a\alpha$

La partie droite de la règle doit toujours commencer par un terminal.