Chapter 2:Grammars

Plan

Definition

- A grammar **G** is a quadruple **(V_N, V_T, S, R)** where:
- **V_N** represents a finite set called the non-terminal vocabulary.
- **V_T** represents a finite set called the terminal vocabulary (**$V = V_N \cup V_T$**).
- **S**: The initial symbol or axiom, which is an element of **V_N**.
- **R** is the finite set of rules, **$R \subseteq (V \backslash V_T) \times V^*$**, where a rule **$r \in R$** is noted as:
  - **Left side → Right side**.

$$\alpha(r) \rightarrow \beta(r)$$

Définition

- Let **G = ({S}, {0,1}, S, { S→0S1 , S→01 })**.
- **G** is a Chomsky grammar.
- There are multiple ways to represent the rules of a Chomsky grammar.

Couples	Dérivation	BNF
(α, β) (α, γ)	$\alpha \rightarrow \beta$ $\alpha \rightarrow \gamma$	$\alpha ::= \beta \mid \gamma$

Notation

- When multiple production rules of a grammar share the same left-hand side, they can be factorized by separating the right-hand parts with vertical bars.
- **Example:** Instead of writing:
    - $\mathbf{A} \rightarrow \mathbf{a}\ \mathbf{A}$ and $\mathbf{A} \rightarrow \boldsymbol{\varepsilon}$, we write: $\mathbf{A} \rightarrow \mathbf{a}\ \mathbf{A} \mid \boldsymbol{\varepsilon}$.

Derivation

- A word **y** is immediately derived from a word **x** if and only if there exists a rule **r** and two words **g** and **d** from **V**** such that:

$$x \xrightarrow{r} y \qquad x = g\alpha(r)d \text{ et } y = g\beta(r)d.$$

- Notation: ⇒ represents the reflexive and transitive closure of the relation →.

- The relation ⇒ is called **derivation**.

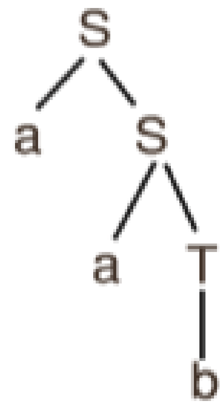
- We denote **r** as the sequence of rules allowing **y** to be derived from **x**.

$$x \overset{\hat{r}}{\Rightarrow} y$$

Exemple

- Consider the grammar **G = ({ a,b },{S,T},S,{S → aS | aT, T → bT |b})**.

- It generates the words **abb** and **aab** because:
  - **S → aT → abT → abb**
  - **S → aS → aaT → aab**



- The generated language consists of all words over **{a, b}** of the form $\mathbf{a^m\, b^n}$ with $\mathbf{m, n > 0}$.

Language generated by a grammar

- The language generated by a grammar **G**, denoted **L(G)**, is the set of terminal words derivable from **S**.
- **Formally:**
  - **Example:** The word **000111** (noted as $\mathbf{0^3 1^3}$) is a word in the language generated by the grammar **G**.
  - Given two rules $\mathbf{r_1}$ and $\mathbf{r_2}$:
    - $\mathbf{0^3 1^3}$ is derived from **S** by applying $\mathbf{r_1}$ twice and $\mathbf{r_2}$ once.

$$L(G) = \left\{ x \in V_T^* / \exists \hat{r} \in R^+, S \overset{\hat{r}}{\Rightarrow} x \right\}$$

Example

- 000111 (also $0^3 1^3$) is a word from the language generated by the grammar G of the previous example
- Let the two rules r1 and r2 be such that

.

$$r_1 : S \rightarrow 0S1 \text{ et } r_2 : S \rightarrow 01.$$

- Therefore : $0^3 1^3$ is derived from S by applying the rule r1 twice, then applying the rule r2 once..
- We will have: ***S***

$$S \xrightarrow{r_1} 0S1 \xrightarrow{r_1} 00S11 \xrightarrow{r_2} 000111$$

Derivation tree of a word

Let the grammar G, G = (V_N, V_T, S, R).

A labeled tree is a 'derivation tree' in G if and only if:

- The alphabet of labels is included in V_N and V_T.
- The nodes are labeled by elements of V_N.
- The leaves are labeled by elements of V_T.
- L'étiquette de tout noeud est un élément d The label of any node is an element of V_N.
- For each node $< \mathbf{A}, f_1, f_2, \ldots, f_n >$ a rule R is associated, which has the form: $\mathbf{A} \rightarrow f_1 f_2 \ldots f_n$ (derivation rule in G)

Derivation tree of a word

- A derivation tree is traditionally drawn with the root at the top. On such a drawn tree, the derived word is obtained by concatenating the labels of the leaves from left to right.
- Graphical representation of a tree

Examples

- The grammar G is as follows

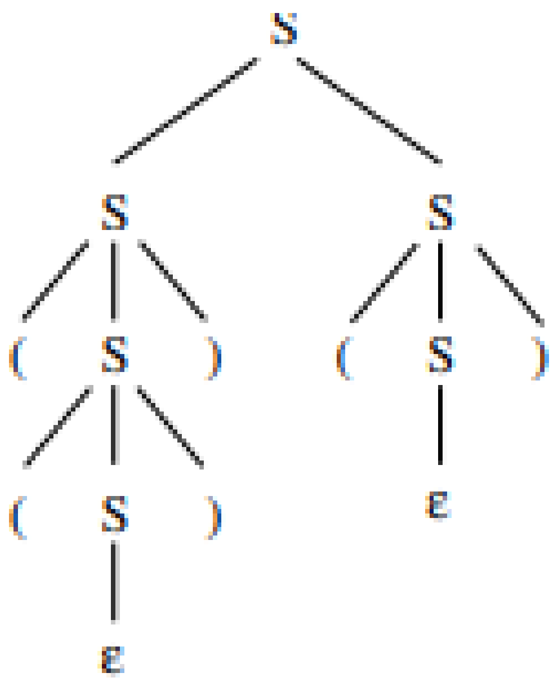
  $S \rightarrow \varepsilon,\ S \rightarrow SS\ ,\ S \rightarrow (S)$

- The terminal letters are (and) and S is a non-terminal letter, which is also the axiom of the grammar. For example, the word (())() is obtained by:

$$\begin{aligned} S \rightarrow SS &\rightarrow (S)S \\ &\rightarrow ((S))S \\ &\rightarrow ((\varepsilon))S = (())S \\ &\rightarrow (())(S) \\ &\rightarrow (())(\varepsilon) = (())(). \end{aligned}$$

Examples

- Derivation tree of the word (())():



Different types of grammars

- A grammar is said to be of type 3 (linear, regular) if and only if every production $\alpha \to \beta$ is either of the form A $\to$ aB, with a in X and A, B in N, or of the form A $\to$ a.
- Formally:
  - $\forall \boldsymbol{r} \in \boldsymbol{R}\ \boldsymbol{\alpha}(\boldsymbol{r}) \in \boldsymbol{V_N} \boldsymbol{et}\ \boldsymbol{\beta}(\boldsymbol{r}) \in \boldsymbol{V_T V_N} \cup \boldsymbol{V_T}$
- A grammar is said to be of type 2 (context-free, algebraic) if and only if:
  - $\forall \boldsymbol{r} \in \boldsymbol{R}\ \boldsymbol{\alpha}(\boldsymbol{r}) \in \boldsymbol{V_N}$

Different types of grammars

- A grammar is said to be of type 1 (context-sensitive, or monotonic) when it introduces a first restriction on the form of the rules, by requiring that the right-hand side of each production be necessarily longer than the left-hand side.
- Formally: $-\forall \boldsymbol{r} \in \boldsymbol{R}\ |\boldsymbol{\alpha}(\boldsymbol{r})| \leq |\boldsymbol{\beta}(\boldsymbol{r})|$
- A grammar with no restrictions on the rules is said to be of type 0.
- Note: – Usual programming languages generally have a type 2 grammar.

Type ?

$$
\begin{aligned}
S &\rightarrow AB \mid CA \\
A &\rightarrow a \mid b \mid \varepsilon \\
B &\rightarrow BC \mid DB \\
C &\rightarrow E \mid \varepsilon \\
D &\rightarrow a \mid d \\
E &\rightarrow aB \mid c \mid d \mid \varepsilon
\end{aligned}
$$

1. G1<{a,b},{S,A,B},P,S>

où P : { S → AS / bB

A → a / ε

B → aB / a / ε }

2.G2<{a,b,c},{S,A,B,C, },P,S>

où P :

S → aSSB | ASC | a,

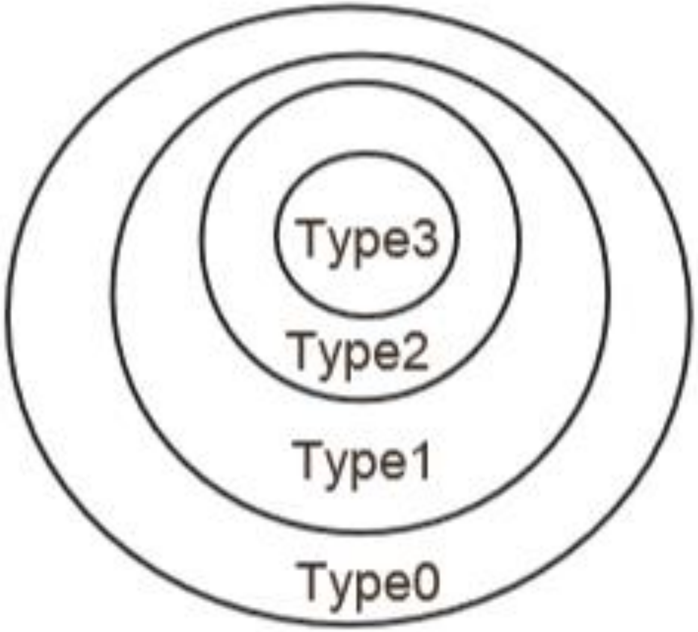
A → AAB | B | C,

B → a | ε,

C → AC | CB }

Different types of grammars

- There is an inclusion relation between the types of grammars according to the following figure:



Type d'un langage

- The type selected for a grammar is the smallest one that satisfies the conditions.
- To determine the class of a language, the following procedure is followed:

– Look for a type 3 grammar that generates it; if it exists, the language is of type 3 (or regular).

– If not, look for a type 2 grammar that generates it; if it exists, the language is of type 2 (or algebraic).

– If not, look for a type 1 grammar that generates it; if it exists, the language is of type 1 (or context-sensitive).

– Otherwise, the language is of type 0.

Exercices

- Give, without proof, the languages generated by the following grammars. State, each time, which type the grammar is:

– $G = (\{a\}, \{S\}, S, \{S \to abS \mid b\});$

– $G = (\{a\}, \{S\}, S, \{S \to aSa \mid \varepsilon\});$

– $G = (\{a\}, \{S\}, S, \{S \to aSb \mid \varepsilon\});$

– $G = (\{a,b\}, \{S\}, S, \{S \to aSa \mid bSb \mid \varepsilon\});$

- Give the grammars that generate the following languages:

  Binary numbers;

  Words over $\{a, b\}$ that contain the factor 'aa'.