CHAPITRE 3

# GRAMMAIRES GÉNÉRATIVES

Dans ce chapitre, nous allons étudier les grammaires génératives appelées également les grammaires formelles ou encore des systèmes générateurs de langages. Ce sont des systèmes (mathématiques) de substitution ou de remplacement qui décrivent la structure syntaxique d'un langage formel.

## 3.1 L'intérêt d'étudier les grammaires

Pour comprendre l'intérêt de la définition d'un langage par une grammaire, prenons l'exemple de quelqu'un souhaitant enseigner une langue (comme le français ou l'anglais) à une autre personne. Une approche simple mais peu pratique serait d'essayer d'apprendre toutes les phrases possibles dans cette langue. Si l'on réussissait cet exploit, on dirait alors que l'on maîtrise cette langue, bien que cela puisse s'avérer naïf, voire impossible. Cette méthode n'est donc pas efficace, étant donné le grand nombre (voire infini) de phrases et de mots dans la langue.

Une approche différente permettant d'enseigner une langue consisterait à expliquer comment former ces phrases en utilisant ce que l'on appelle une grammaire syntagmatique ou encore "PhraseStructureGrammar", une terminologie plutôt utilisée par les linguistes. Or, chez les informaticiens on dit communément les règles grammaticales ou plus simplement "rules".

Au lieu de mémoriser toutes les phrases, l'idée consiste à expliquer, par exemple, qu'une phrase basique en français se compose d'un sujet, d'un verbe et d'un complément d'objet direct. La grammaire française présente l'avantage d'avoir un nombre limité de règles tout en permettant de construire un vaste éventail de phrases, d'où l'intérêt des grammaires pour définir les langages.

### Exemple

*Prenons l'exemple suivant, dans lequel nous essayons de construire un type de phrases simples en français selon les règles suivantes :*

- *PHRASE* $\rightarrow$ *ARTICLE SUJET VERBE COMPLEMENT*
- *SUJET* $\rightarrow$ *"laboratoire" ou "chercheur" ou "entreprise"*
- *VERBE* $\rightarrow$ *"innove" ou "expérimente" ou "développe"*
- *COMPLEMENT* $\rightarrow$ *ARTICLE NOM ADJECTIF*
- *ARTICLE* $\rightarrow$ *"un" ou "le" ou ce*
- *NOM* $\rightarrow$ *"projet" ou "logiciel" ou "prototype"*
- *ADJECTIF* $\rightarrow$ *"innovant" ou "scientifique" ou "révolutionnaire"*

*Les concepts : PHRASE, ARTICLE, SUJET, VERBE, COMPLEMENT, NOM, ADJECTIF sont appelés des non-terminaux, car ils ne figurent pas dans la phrase finale (langage final). Alors que le reste (laboratoire, innove, le, projet, etc.) sont appelés des terminaux.*

*En effectuant une séquence de substitutions (remplacer les parties gauches par les parties droites), nous volons former la phrase suivante : Le laboratoire développe un projet scientifique. Le processus de génération de cette phrase est appelé : Une séquence de dérivation :*

*PHRASE → ARTICLE SUJET VERBE COMPLEMENT → le SUJET VERBE COMPLEMENT → le laboratoire VERBE COMPLEMENT → le laboratoire développe COMPLEMENT → le laboratoire développe ARTICLE NOM ADJECTIF → le laboratoire développe un NOM ADJECTIF → le laboratoire développe un projet ADJECTIF → le laboratoire développe un projet scientifique*

**Donc l'un intérêt majeur des grammaires formelles est le suivant :**

- **Générer les mots d'un langages par réécritures successives, ce qui permet de tracer les étapes de la construction**
- **À partir de ces grammaires se déduisent (de manière plus ou moins directe) des algorithmes permettant de reconnaître les mots d'un langage, de décider si un mot appartient ou non à un langage (Pour les sous-classes les plus simples)**
- **Permettre une classification (la classification de Chomsky) des langages en fonction de leur niveau de complexité** (voir plus loin).

## 3.2 Formalisation

### Définition 3.2.1 Une grammaire $G$

Une grammaire formelle est définie par le quadruplet $(V, N, S, R)$ tel que :

— $V$ est un ensemble fini de symboles terminaux, appelé aussi vocabulaire (Alphabet).

— $N$ est un ensemble fini (disjoint de $V$) de symboles non-terminaux ou concepts.

— $S \in N$ est un axiome (point de départ d'une dérivation).

— $R$ est un ensemble de règles de production (de génération) définies sur :
$(V+N)^+ \to (V+N)^*$.
La forme générale de chaque règle : $g \to d$ tel que $g \in (V+N)^+$ et $d \in (V+N)^*$.

### Remarque 3.2.1

— Les règles de la forme $\varepsilon \to d$ sont interdites.

— Par convention, les non-terminaux sont représentés par des lettres majuscules, et les terminaux sont représentés par des lettres minuscules.

### Dérivation et arbre de dérivation

### Définition 3.2.2 Dérivation

- $w_1 \to w_2 \to \ldots \to w_n$, est une séquence de dérivation notée aussi par $w_1 \overset{*}{\to} w_n$ qui part de $w_1$ vers $w_n$.
- On dit que $w_{i+1} = x\beta y$ $(x, y \in (V+N)^*)$ est une dérivation immédiate de $w_i = x\alpha y$, s'il existe une production $\alpha \to \beta$

### Définition 3.2.3 Un langage généré par une grammaire $L(G)$

Un langage $L$ généré par une grammaire $G = (V, N, S, R)$, noté $L(G)$ est l'ensemble de tous les mots $w \in V^*$ qui sont dérivés (générés) à partir de $G$.
$G$ est aussi considérée comme une définition récursive de $L$

**Déduction 3.2.1** $w \in L(G)$ **d'un point de vue grammatical**

$w \in L(G)$ si et seulement s'il existe une suite (une chaîne) de dérivations qui, partant de l'axiome $S$, permettent de générer $w$ : $S \xrightarrow{*} w$. Notons que $w$ ne contient aucun non-terminal

**Remarque 3.2.2** Une séquence de dérivation qui n'aboutit pas

En fonction de la classe de grammaire du langage, la recherche de cette chaîne de dérivation peut être efficace ou non ou carrément impossible. Dans le cas des langages de programmation usuels, cette approche est couramment employée pour résoudre ce problème. C'est pourquoi les grammaires des langages sont soigneusement conçues de manière à optimiser la recherche.

**Exemple**

*Soit la grammaire* $G = (\{a\}, \{S\}, S, \{S \to aS|\varepsilon\})$*.Essayons de générer quelques mots :*

- $S \to \varepsilon$.
- $S \to aS \to a$.
- $S \to aS \to a\underbrace{aS}_{S} \to aa$.
- $S \to aS \to aaS \to aaS \to aaa$.
- ...

*Donc* $L(G) = \{\varepsilon, a, aa, aaa, \ldots\} = \{a\}^*$ *ce sont tous les mots générés par cette grammaire. La dérivation de chaque mot se termine par une suite de symboles qui ne comporte aucun non-terminal.*

Les grammaires permettent également d'associer des structures (Arbres) aux mots d'un langage :

**Définition 3.2.4** Arbre syntaxique

Pour prouver que $w$ est généré par une grammaire $G = (V, N, S, R)$, on peut chercher également son arbre syntaxique (ou arbre de dérivation). S'il y a une règle $A \to x_1 \ldots x_n$, (Dans ce cours, on considère que $A \in N$, $x_i$ est un terminal ou non-terminal). Alors nous aurons un nœud $A$ qui possède $x_1 \ldots x_n$ comme fils.

- Un arbre syntaxique associé à $w$ est construit tel-que :
  - La racine est étiquetée par l'axiome $S$.
  - Les nœuds intermédiaires (nœuds parents ayant des fils) sont étiquetés par des symboles de $N$.
  - Les feuilles sont étiquetées par des symboles de $V$
  - Les nœuds fils d'un nœud parent sont ajoutés à l'arbre en allant de la gauche vers la droite

  Les feuilles de l'arbre forment le mot $w$ dérivé et dont la lecture se fait de la gauche vers la droite, du bas vers le haut.

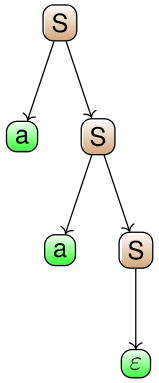
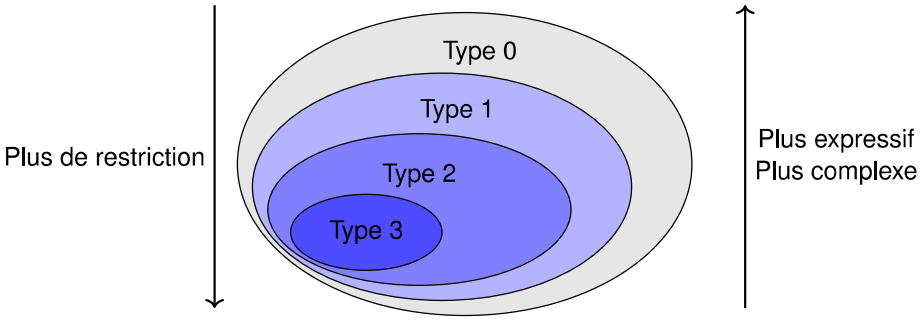



FIGURE 3.1 – Arbre syntaxique d'un mot ($aa$) généré par la grammaire (de type 3) de l'exemple précédent

## 3.3 Classification de Chomsky

Selon la complexité des grammaires, Noam Chomsky a proposé, en 1956, une classification hiérarchique des grammaires en se basant sur la forme des règles $g \to d$. Il existe quatre classes (familles) de grammaires (et donc de langages), de telle sorte qu'une grammaire $G = (V, N, S, R)$ de type $i$ génère un langage de type $j$, avec $j \geq i$.

FIGURE 3.2 – La classification de Chomsky : Une classification hiérarchique des grammaires




Chaque classe ou type de grammaire correspond à une contrainte particulière sur la forme de ses règles. Le niveau de complexité est inversement proportionnel au type de la grammaire (du langage généré) car le nombre d'algorithmes existants tend à diminuer en laissant la place à plus d'intuition.

### 3.3.1 Grammaire de type 3 (régulière)

**Définition 3.3.1 Grammaire régulière à droite**

Une grammaire $G$ est dite régulière à droite si toutes les règles de production sont de la forme $g \to d$ où $g \in N$ et $d = \alpha B$ tel que $\alpha \in V^*$ et $B \in N + \{\varepsilon\}$.
▶ Le type du langage généré $L(G)$ est régulier

Donc le trait spécifique de ce type de grammaires est que toutes les règles produisent $n$ symboles terminaux ($n \geq 0$), suivis d'au plus un non-terminal.

**Définition 3.3.2 Grammaire régulière à gauche**

Une grammaire $G$ est dite régulière à gauche si toutes les règles de production sont de la forme $g \to d$ où $g \in N$ et $d = B\alpha$ tel que $\alpha \in V^*$ et $B \in N + \{\varepsilon\}$.
▶ Le type du langage généré $L(G)$ est régulier.

### Grammaire à choix finis

Les grammaires à choix finis sont encore plus simples que les grammaires régulières. Elles génèrent que des langages finis et constituent ainsi un sous-type ou encore le type 4.

**Définition 3.3.3 Grammaire à choix finis (de type 4)**

Une grammaire $G$ est une grammaire à choix finis si tout non-terminal ne génère que des terminaux mais pas $\varepsilon$. Ses règles de production sont de la forme $A \to w, A \in N, w \in V^+$

**Exemple**

*$G = (\{a, b, c\}, \{S, T\}, S, \{S \to abS|acT, T \to S|\varepsilon\})$ est une grammaire de type 3, ou encore régulière à droite. Pour justifier, on analyse les règles une par une. On remarque que la partie gauche de toutes les règles $g \in N$ et la partie à droite $d = \alpha B$ :*

- *$S \to abS$ est de type 3 ($\alpha = ab, B = S$).*
- *$S \to acT$ est type 3 ($\alpha = ac, B = T$)*
- *$T \to S$ est de type 3 ($\alpha = \varepsilon, B = S$)*
- *$T \to \varepsilon$ est de type 3 ($\alpha = \varepsilon, B = \varepsilon$)*

*Quant au type du langage généré, il est régulier.*
*Un exemple d'un mot généré : $S \to abS \to abacT \to abac$. On peut définir le langage généré : $\{((ab)^p ac)^q \mid p \geqslant 0, q > 0\}$.*

**Remarque 3.3.1**

- Tout langage régulier possède une grammaire régulière à droite et une autre grammaire régulière à gauche. (On peut passer aisément d'une forme à l'autre).
- Si on applique le miroir sur toutes les parties droites $\alpha B$ des règles d'une grammaire régulière à droite $G$ générant $L$, alors on obtient une grammaire régulière à gauche $G'$ générant le langage miroir $L^R$. (On peut illustrer ça avec la grammaire qui génère $\{a^n b^m, n, m \geq 0\}$)
- Une grammaire régulière est soit à droite soit à gauche. Elle ne peut pas être régulière à droite et à gauche en même. Si une grammaire présente à la fois les formes régulières à gauche et les formes régulières à droite on passe au type supérieur dans la hiérarchie (type 2).

## 3.3.2 Grammaire de type 2 (à contexte libre)

**Définition 3.3.4 Grammaire à contexte libre ou hors-contexte**

Une grammaire dite hors-contexte ou de type 2 lorsque toutes les règles de production sont de la forme $g \to d$ tel que $g \in N$ et $d \in (V + N)^*$.
Le type du langage généré par ce type de grammaire est dit algébrique.

**Exemple**

*$G = (\{a, b\}, \{S, T\}, S, \{S \to aSbc|\varepsilon\})$ est une grammaire de type 2 ou encore une grammaire hors-contexte. Car, la partie gauche de toutes les règles : $g \in N$ Alors qu'à droite on a :*

- *La règle $S \to aSbc$, $d = aSbc \in (V + N)^*$ prend la forme du type 2.*
- *La règle $S \to \varepsilon$, $d = \varepsilon$ prend la forme du type 3*

*Pour conclure, le type de cette grammaire revient au type supérieur selon la hiérarchie. Donc c'est le type 2*
*Un exemple de mot généré est : $S \to aSbc \to aaSbcbc \to aabcbc$. $L(G) = \{a^n (bc)^n \mid n \geqslant 0\}$.*

*Le type du langage généré est algébrique, à condition qu'il n'y ait pas de grammaire régulière qui génère le même langage. Si on arrive à trouver une grammaire de type 3 qui génère $L(G)$ alors ce langage est en réalité régulier. Mais, ce n'est pas le cas ici (on l'accepte sans passer par la démonstration qui prouve que $L(G)$ n'est pas régulier).*

### 3.3.3 Grammaire de type 1 (Contextuelle)

**Définition 3.3.5 Grammaire contextuelle**

Une grammaire dite contextuelle ou de type 1, si toutes les règles de production sont de la forme $g \to d$ tel que $g \in (N+V)^+, d \in (V+N)^*$ et $|g| \leqslant |d|$. Si $d = \varepsilon$ alors la partie gauche $g = S$ (l'axiome). On peut aussi trouver une autre définition donnée aux grammaires contextuelles.
Le type du langage généré par ce type de grammaire est dit contextuel.

**Exemple**

$G = (\{a,"\}, \{S,B\}, S, \{S \to "a", "a \to "B, Ba \to aaB, B" \to aa"\})$

— $"a \to "B, Ba \to aaB, B" \to aa"$ *: Type 1*

— $S \to "a"$ *:Type 3*

*$G$ est une grammaire de type 1 ou une grammaire contextuelle. Car il existe au moins une règle qui ne vérifie ni la forme du type 3, ni celle du type 2 mais plutôt la forme du type 1.*
*Un exemple d'un mot généré : $S \to "a" \to "B" \to "aa" \to "Ba" \to "aaB" \to "aaaa"$. $L(G) = \{"a^{2^p}" \mid p \geqslant 0\}$ . Pour arriver à générer un mot (avec $p \geq 1$), on remarque que le $B$ est déplacé de la gauche vers la droite en remplaçant chaque $a$ se trouvant devant $B$ par 2 $a$ derrière ce $B$.*
*Le langage généré $L(G)$ est contextuel, évidement car $G$ est contextuelle et en plus il ne peut pas être généré par une grammaire de type inférieur dans la hiérarchie (type 2 ou 3).*

### 3.3.4 Grammaire de type 0

**Définition 3.3.6**

Une grammaire de type 0 est une grammaire sans quasiment aucune restriction. Toutes les règles sont de la forme : $d \to g, g \in (V+N)^+, d \in (V+N)^*$. Les grammaires de type 0 ont donc une expressivité maximale.
Le type du langage généré est tout simplement le type 0.
Mais il existe d'autres sous-types plus "raffinés" (langages récursivement énumérable ou encore récursifs), qui seront abordés dans la chapitre suivant.

**Exemple**

$G = (\{a\}, \{S,T,U\}, S, \{S \to UaT|a, T \to TT, aT \to Taa, UT \to U, U \to \varepsilon\})$.
*On remarque qu'il existe au moins une règle qui ne respecte pas la forme des trois types précédents : $UT \to U$.(Sachant qu'il faut à chaque fois analyser toutes règles).*
*La grammaire $G$ est dans ce cas de type 0.*
*Un exemple de mot généré : $S \to UaT \to UaTT \to UaTTT \to UTaaTT \to UTaTaaT \to UTTaaaaT \to UTTaaaTaa \to UTTaaTaaaa \to UTTaTaaaaa \to UTTTaaaaaaa \to UTTaaaaaaa \to UTaaaaaaa \to Uaaaaaaa \to aaaaaaa$ ou encore $a^8$. $L(G) = \{a^{2^p} \mid p \geqslant 0\}$.*
*Le type de ce langage généré $L(G)$ est le type 0.*

**Remarque 3.3.2 sur la classfication de Chomsky d'une manière générale**

- Pour reconnaître le type d'une grammaire $G$, on doit analyser la forme de ses règles, une par

une afin de déterminer le type de chacune, tout en sachant qu'il existe une relation d'inclusion entre les types de grammaires. Le type (classe) retenu $i$ est celui le plus petit qui satisfait les conditions (restrictions).

- C'est à cause de cette inclusion (hiérarchie) que le type du langage $L(G)$ n'est pas systématiquement le type $i$ mais peut être déterminé à travers une autre grammaire $G'$ qui génère le même langage et dont le type est $j$, tel que $j \geq i$.
  Par conséquent, un langage simple, par exemple hors-contexte, règulier, voire fini peut être défini par une grammaire plus complexe.

> **Définition 3.3.7 Équivalence entre grammaires**
>
> Deux grammaires $G_1$ et $G_2$ sont équivalentes si elles génèrent le même langage $L$. $\forall w \in L$ si ses arbres syntaxiques fournis par $G_1$ et $G_2$ sont identiques, alors $G_1$ et $G_2$ sont fortement équivalentes sinon elles sont faiblement équivalentes.

- Les langages hors contexte (générés par les grammaires de type 2 ou d'autres types supérieurs) constituent d'une certaine manière la base des langages de programmation, d'ou l'importance des grammaires qui les génèrent dans la conception des analyseurs syntaxiques. Cependant ces langages ne peuvent pas être entièrement décrits par une grammaire à contexte libre

**Théorème 3.3.1 La fermeture d'une classes de langages**

Les classes $i$ ($i = 0 \ldots 3$) des langages reste fermées par rapport aux opérations régulières.
Autrement dit, Si $L_1$ et $L_2$ sont deux langages de type $i$ alors :

- $L_1 + L_2$ est de type $i$.
- $L_1^*$ est de type $i$.
- $L_1.L_2$ est de type $i$.
- $L_1^R$ est de type $i$.

En revanche,ce n'est pas le cas pour les opérations telles que la complémentation et l'intersection.

## 3.4 Série d'exercices de TD N°2

### Exercice 1 : Défintion grammaticale d'un langage et dérivation

*Pour chacune des grammaire suivantes :*

1. $G_1 = (\{a\}, \{S\}, S, \{S \to aS|\varepsilon\})$
2. $G_2 = (\{a\}, S, \{S\}, S, \{S \to aSa|\varepsilon\})$
3. $G_3 = (\{a, b\}, S, \{S\}, S, \{S \to aSa|bSb|\varepsilon\})$

— *Donnez le type de la grammaire (Justifiez)*

— *Donnez les dérivations de quelques mots générés par la grammaire*

— *Donnez la définition formelle du langage généré.*

### Exercice 2 : Classification des grammaires/ langages générés

*Donnez le type de chacune des grammaires suivantes ainsi que le types du langage généré. Justifez ( Il est parfois nécessaire de chercher une grammaire équivalente pour reconnaître le vrai type du langage).*

1. $G_1 = (\{a, b\}, \{S, T\}, S, \{S \to aabS|aT, T \to bS|\varepsilon\})$.
2. $G_2 = (\{a, b, c\}, \{S, T, U\}, S, \{S \to bSTa|aTb, T \to abS|cU, U \to S|\varepsilon\})$.
3. $G_3 = (\{a, b, c\}, \{S, T\}, S, \{S \to Ta|Sa, T \to Tb|Sb|\varepsilon\})$.
4. $G_4 = (\{x, +, *\}, \{S\}, S, \{S \to S + S|S * S|x\})$.
5. $G_5 = (\{0, 1, 2\}, \{S, T, C, Z, U\}, S, \{S \to TZ, T \to 0U1, T \to 01, U \to 0U1C|01C, C1 \to 1C, CZ \to Z2, 1Z \to 12\})$.
6. $G_6 = (\{0, 1, 2\}, \{S, T, C, Z, U\}, S, \{S \to TZ, T \to 0T1C|\varepsilon, C1 \to 1C, CZ \to Z2, 1Z \to 1\})$.

### Exercice 3 : Construire des grammaires

*Donnez les grammaires qui génèrent les langages suivants :*

1. *Les nombres binaires.*
2. *Les mots définis sur $\{a, b\}$ qui contiennent le facteur $a$.*
3. *Les langage des mots : $L = \{w \in \{0, 1\}^* || w|_1 = 2k, k \geq 0\}$.*

### Exercice 4 : Construire des grammaires suivant des opérations sur les langages

*Soient $G$ et $G'$ deux grammaires qui génèrent respectivement les langages $L$ et $L'$.*

1. *Construisez les grammaires qui génèrent :*
   — $L.L'$.
   — $L + L'$.
   — $L^*$.
2. *Appliquez les constructions proposées pour déduire la grammaire de ce langage $(\{a^m b^n | n, m \geq 0\} + \{c^m | m \geq 0\})^*$*
3. *Déduisez le type de ce langage.*