Langages, Compilation, Automates. Partie 5: Grammaires, types de grammaires

Florian Bridoux

Polytech Nice Sophia

2022-2023

Table des matières

Table des matières

Grammaires (de réécriture)

Définition (Grammaires de réécriture)

Une grammaire est un 4-uplet (N, Σ, R, S) où :

- N est un ensemble fini de **symboles non terminaux**.
- Σ est un ensemble fini de **symboles terminaux** avec $N \cap \Sigma = \emptyset$.
- R est un sous ensemble fini de **règles**.
  Une règle de R est de la forme $\alpha \rightarrow \beta$ avec
  - $\alpha, \beta \in (N \cup \Sigma)^*$ et ,
  - α contient **au moins un non terminal**

  α et β sont appelé respectivement **partie gauche et droite de la règle**.
- S est un élément de N appelé l'**axiome** de la grammaire.

Notation

Pour alléger les notations, au lieu d'écrire plusieurs règles:

$$\begin{array}{lcl} \alpha_1 & \rightarrow & \beta_1 \\ \alpha_1 & \rightarrow & \beta_2 \\ \alpha_1 & \rightarrow & \beta_3 \end{array}$$

on écrit simplement:

$$\alpha_1 \rightarrow \beta_1 \mid \beta_2 \mid \beta_3$$

Proto-mots d'une grammaire

Les **proto-mots** d'une grammaire $G = (N, \Sigma, R, S)$ sont des mots construits sur l'alphabet $\Sigma \cup N$, on les définit récursivement de la façon suivante :

- S est un proto-mot de G
- si $\alpha\beta\gamma$ est un proto-mot de G et $\beta \to \delta \in R$ alors $\alpha\delta\gamma$ est un proto-mot de G.

Un proto-mot de G ne contenant aucun symbole non-terminal est appelé un mot **engendré** par G. Le **langage engendré par** G, noté $L(G)$ est l'ensemble des mots engendrés par G.

Dérivation

- L'opération qui consiste à générer un proto-mot $\alpha\delta\gamma$ à partir d'un proto-mot $\alpha\beta\gamma$ et d'une règle de production r de la forme $\beta \rightarrow \delta$ est appelée l'opération de **dérivation**. Elle se note à l'aide d'une double flèche :

$$\alpha\beta\gamma \Rightarrow \alpha\delta\gamma$$

- On note $\alpha \overset{k}{\Rightarrow} \beta$ pour indiquer que β se dérive de α en k étapes.
- On définit aussi les deux notations $\overset{+}{\Rightarrow}$ et $\overset{*}{\Rightarrow}$ de la façon suivante :
  - $\alpha \overset{+}{\Rightarrow} \beta \equiv \alpha \overset{k}{\Rightarrow} \beta$ avec $k > 0$
  - $\alpha \overset{*}{\Rightarrow} \beta \equiv \alpha \overset{k}{\Rightarrow} \beta$ avec $k \geq 0$

Attention

Les symboles $\Rightarrow$ et $\rightarrow$ ne représentent pas la même chose.

Conventions

- Les symboles non terminaux appartenant à N sont représentés par des lettres latines majuscules : $A, B, C, S, E, T \ldots$
- Les symboles terminaux appartenant à Σ sont représentés par des lettres latines minuscules : $a, b, c, d \ldots$
- Les proto-mots appartenant à $(N \cup \Sigma)^*$ sont représentés par des lettres grecques minuscules : $\alpha, \beta, \gamma, \varepsilon \ldots$
- L'axiome est très souvent représenté par le non-terminal S et constitue la partie gauche de la première règle de production

Remarque:

En Lex et yacc, tout est inversé: les terminaux sont en majuscule et les non terminaux sont en minuscule.

Langage engendré par une grammaire

- $L(G)$ est défini de la façon suivante :

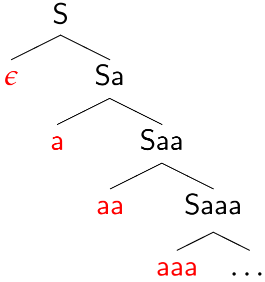
$$L(G) = \{m \in \Sigma^* \mid S \overset{+}{\Rightarrow} m\}$$

- Deux grammaires G et G' sont équivalentes si $L(G) = L(G')$.

$L_1 = \{\varepsilon, a, aa, aaa, \dots\}$

$$G = (\{S\}, \{a\}, \{S \to Sa \mid \varepsilon\}, S)$$

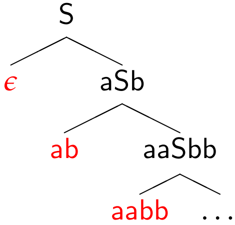
Proto-mots de G:



$L_2 = \{\varepsilon, ab, aabb, aaabbb, aaaabbbb, \dots\}$

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

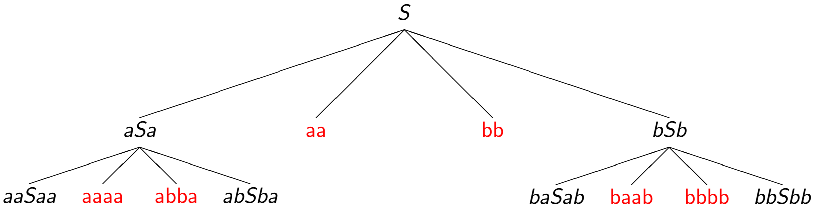
Proto-mots de G:



$L_3 = \{aa, bb, aaaa, abba, baab, bbbb, \dots\}$

$$G = (\{S\}, \{a, b\}, \{S \to aSa \mid bSb \mid aa \mid bb\}, S)$$

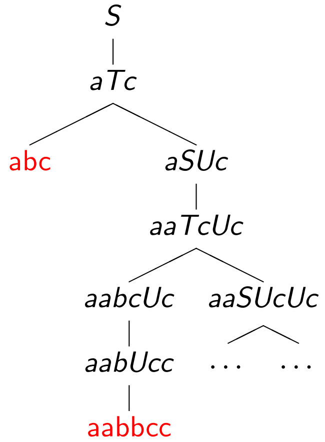
Proto-mots de G:



$L_4 = \{\varepsilon, abc, aabbcc, aaabbbccc, \ldots\}$

$$G = (\{S, T, U\}, \{a, b, c\}, \left\{ \begin{array}{rclrcl} S & \rightarrow & aTc, & T & \rightarrow & b \mid SU, \\ cU & \rightarrow & Uc, & bU & \rightarrow & bb \end{array} \right\}, S)$$

Proto-mots de G:



Sens de dérivation

$G = (\{E, T, F\}, \{+, *, a\}, \{E \to T + E \mid T, T \to F * T \mid F, F \to a\}, E)$
Les proto-mots engendrés lors d'une dérivation peuvent comporter plus d'un symbole non-terminal :

$$
\begin{array}{rcl}
E & \Rightarrow & T + E \\
  & \Rightarrow & T + T \\
  & \Rightarrow & F + T \\
  & \Rightarrow & F + F * T \\
  & \Rightarrow & F + a * T \\
  & \Rightarrow & F + a * F \\
  & \Rightarrow & a + a * F \\
  & \Rightarrow & a + a * a
\end{array}
$$

Sens de dérivation

Dérivation gauche : on réécrit le non-terminal le plus à gauche :

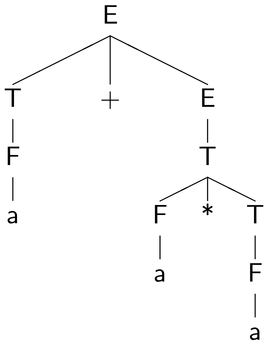
$$
\begin{aligned}
E &\Rightarrow T + E \\
&\Rightarrow F + E \\
&\Rightarrow a + E \\
&\Rightarrow a + T \\
&\Rightarrow a + F * T \\
&\Rightarrow a + a * T \\
&\Rightarrow a + a * F \\
&\Rightarrow a + a * a
\end{aligned}
$$

Dérivation droite :

...

$$
\begin{aligned}
E &\Rightarrow T + E \\
&\Rightarrow T + T \\
&\Rightarrow T + F * T \\
&\Rightarrow T + F * F \\
&\Rightarrow T + F * a \\
&\Rightarrow T + a * a \\
&\Rightarrow F + a * a \\
&\Rightarrow a + a * a
\end{aligned}
$$

Arbre de dérivation



Un arbre de dérivation pour $G = (N, \Sigma, R, S)$ est un arbre ordonné et étiqueté dont les étiquettes appartiennent à l'ensemble $N \cup \Sigma \cup \{\varepsilon\}$. Si un nœud de l'arbre est étiqueté par le non terminal A et ses fils sont étiquetés $X_1, X_2, ..., X_n$ alors la règle $A \rightarrow X_1 X_2 ... X_n$ appartient à P.

Arbre de dérivation

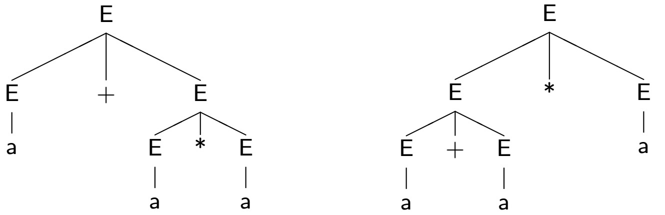
- Un arbre de dérivation indique les règles qui ont été utilisées dans une dérivation, mais pas l'ordre dans lequel elles ont été utilisées.

Ambiguïté

Une grammaire G est **ambiguë** s'il existe au moins un mot m dans $L(G)$ auquel correspond plus d'un arbre de dérivation.

Exemple :

$$E \rightarrow E + E \mid E * E \mid a$$



Ambiguïté

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

Ambiguïté

Exemple :

$$E \to A + E \mid A, A \to a * A \mid a$$

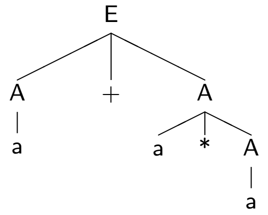


Table des matières

Types de règles

Les grammaires peuvent être classées en fonction de la forme de leurs règles de production. On définit cinq types de règles :

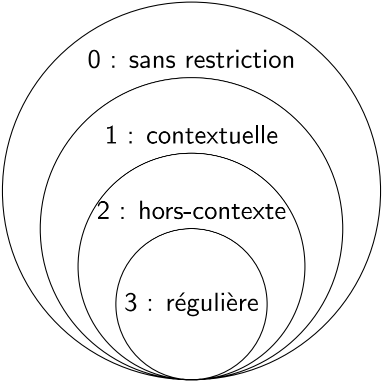
- Une règle est **régulière à gauche** si et seulement si elle est de la forme $A \to xB$, $A \to x$ ou $A \to \epsilon$ avec $A, B \in N$ et $x \in \Sigma^*$.
- Une règle est **régulière à droite** si et seulement si elle est de la forme $A \to Bx$, $A \to x$ ou $A \to \epsilon$ avec $A, B \in N$ et $x \in \Sigma^*$.
- Une règle $A \to \alpha$ est un règle **hors-contexte** si et seulement si : $A \in N$ et $\alpha \in (N \cup \Sigma)^*$.
- Une règle $\alpha \to \beta$ est une règle **contextuelle** si et seulement si : $\alpha = gAd$ et $\beta = g\delta d$ avec $g, d, \delta \in (N \cup \Sigma)^*$ et $A \in N$. *Le nom "contextuelle" provient du fait que A se réecrit B uniquement dans le contexte g_d.*
- Une règle $\alpha \to \beta$ **sans restriction** sont les plus générale. Il faut juste que α ait au moins un non terminal.

Type d'une grammaire

Une grammaire est :

- **régulière** ou de type 3 si elle est régulière à droite ou régulière à gauche. Une grammaire est régulière à gauche si **toutes** ses règles sont régulières à gauche et une grammaire est régulière à droite si **toutes** ses règles sont régulières à droite.
- **hors contexte** ou de type 2 si toutes ses règles de production sont hors contexte.
- **dépendante du contexte** ou de type 1 si toutes ses règles de production sont dépendantes du contexte.
- **sans restrictions** ou de type 0 si toutes ses règles de production sont sans restrictions.

Hiérarchie de Chomsky-Schützenberger



Type d'un langage

Un langage pouvant être engendré par une grammaire de type x et pas par une grammaire d'un type supérieur dans la hiérarchie, est appelé un **langage de type** x.

Type du langage	Nom
3	régulier
2	hors contexte
1	dépendant du contexte
0	récursivement énumérable

Les grammaires hors contextes sont aussi appelé algébriques.

Exemples de langages réguliers

$$L_1 = \{m \in \{a, b\}^*\}$$

$$L_2 = \{m \in \{a, b\}^* \mid |m|_a \mod 2 = 0\}$$

$$L_3 = \{m \in \{a, b\}^* \mid m = xaaa \text{ avec } x \in \{a, b\}^*\}$$

$$L_4 = \{m \in \{a, b\}^* \mid |m|_a \mod 2 = 0 \text{ et} |m|_b \mod 2 = 0\}$$

Exemples de langages réguliers

$$\begin{aligned} L_1 &= \{m \in \{a,b\}^*\} \\ G_1 &= (\{S\}, \{a,b\}, \{S \to aS \mid bS \mid \varepsilon\}, S) \end{aligned}$$

$$L_2 = \{m \in \{a,b\}^* \mid |m|_a \mod 2 = 0\}$$

$$L_3 = \{m \in \{a,b\}^* \mid m = xaaa \text{ avec } x \in \{a,b\}^*\}$$

$$L_4 = \{m \in \{a,b\}^* \mid |m|_a \mod 2 = 0 \text{ et} |m|_b \mod 2 = 0\}$$

Exemples de langages réguliers

$$
\begin{array}{rcl}
L_1 & = & \{m \in \{a,b\}^*\} \\
G_1 & = & (\{S\},\{a,b\},\{S \to aS \mid bS \mid \varepsilon\},S)
\end{array}
$$

$$
\begin{array}{rcl}
L_2 & = & \{m \in \{a,b\}^* \mid |m|_a \mod 2 = 0\} \\
G_2 & = & \left(\{S,T\},\{a,b\},\left\{\begin{array}{lcl} S & \to & aT \mid bS \mid \varepsilon, \\ T & \to & aS \mid bT \end{array}\right\},S\right) \\
L_3 & = & \{m \in \{a,b\}^* \mid m = xaaa \text{ avec } x \in \{a,b\}^*\}
\end{array}
$$

$$
L_4 = \{m \in \{a,b\}^* \mid |m|_a \mod 2 = 0 \text{ et} |m|_b \mod 2 = 0\}
$$

Exemples de langages réguliers

$$
\begin{array}{rcl}
L_1 & = & \{m \in \{a,b\}^*\} \\
G_1 & = & (\{S\}, \{a,b\}, \{S \rightarrow aS \mid bS \mid \varepsilon\}, S) \\
\\
L_2 & = & \{m \in \{a,b\}^* \mid |m|_a \mod 2 = 0\} \\
G_2 & = & \left(\{S,T\}, \{a,b\}, \left\{ \begin{array}{rcl} S & \rightarrow & aT \mid bS \mid \varepsilon, \\ T & \rightarrow & aS \mid bT \end{array} \right\}, S\right) \\
L_3 & = & \{m \in \{a,b\}^* \mid m = xaaa \text{ avec } x \in \{a,b\}^*\} \\
G_3 & = & \left(\{S,T,U\}, \{a,b\}, \left\{ \begin{array}{rcl} S & \rightarrow & aS \mid bS \mid aT, \\ T & \rightarrow & aU, \\ U & \rightarrow & a \end{array} \right\}, S\right) \\
L_4 & = & \{m \in \{a,b\}^* \mid |m|_a \mod 2 = 0 \text{ et} |m|_b \mod 2 = 0\}
\end{array}
$$

Exemples de langages réguliers

$$
\begin{array}{rcl}
L_1 & = & \{m \in \{a,b\}^*\} \\
G_1 & = & (\{S\}, \{a,b\}, \{S \to aS \mid bS \mid \varepsilon\}, S) \\
\\
L_2 & = & \{m \in \{a,b\}^* \mid |m|_a \mod 2 = 0\} \\
G_2 & = & (\{S,T\}, \{a,b\}, \left\{ \begin{array}{lcl} S & \to & aT \mid bS \mid \varepsilon, \\ T & \to & aS \mid bT \end{array} \right\}, S) \\
L_3 & = & \{m \in \{a,b\}^* \mid m = xaaa \text{ avec } x \in \{a,b\}^*\} \\
G_3 & = & (\{S,T,U\}, \{a,b\}, \left\{ \begin{array}{lcl} S & \to & aS \mid bS \mid aT, \\ T & \to & aU, \\ U & \to & a \end{array} \right\}, S) \\
L_4 & = & \{m \in \{a,b\}^* \mid |m|_a \mod 2 = 0 \text{ et} |m|_b \mod 2 = 0\} \\
G_4 & = & (\{S,T,U,V\}, \{a,b\}, \\
 & & \left\{ \begin{array}{lclclcl} S & \to & aT \mid bU \mid \varepsilon, & T & \to & aS \mid bV, \\ V & \to & aU \mid bT, & U & \to & aV \mid bS \end{array} \right\}, S)
\end{array}
$$

Exemples de langages hors-contexte

$L_1 = \{a^n b^n \mid n \geq 0\}$

$L_2 = \{mm^{-1} \mid m \in \{a, b\}^*\}$ (langage miroir - palindromes paires)

Exemples de langages hors-contexte

$L_1 = \{a^n b^n \mid n \geq 0\}$
$G_1 = (\{S\}, \{a, b\}, \{S \to aSb \mid \varepsilon\}, S)$

$L_2 = \{mm^{-1} \mid m \in \{a, b\}^*\}$ (langage miroir - palindromes paires)

Exemples de langages hors-contexte

$L_1 = \{a^n b^n \mid n \geq 0\}$
$G_1 = (\{S\}, \{a, b\}, \{S \rightarrow aSb \mid \varepsilon\}, S)$

$L_2 = \{mm^{-1} \mid m \in \{a, b\}^*\}$ (langage miroir - palindromes paires)
$G_2 = (\{S\}, \{a, b\}, \{S \rightarrow aSa \mid bSb \mid \varepsilon\}, S)$

Exemples de langages contextuels

$$L_1 = \{a^n b^n c^n \mid n \geq 0\}$$

Exemples de langages contextuels

$$
\begin{array}{rcl}
L_1 & = & \{a^n b^n c^n \mid n \geq 0\} \\
G_1 & = & (\{S, B, C\}, \{a, b, c\}, \\
 & & \left\{ \begin{array}{rclrcl} S & \rightarrow & aSBC \mid \varepsilon, & CB & \rightarrow & BC, \\ aB & \rightarrow & ab, & bB & \rightarrow & bb, \\ bC & \rightarrow & bc, & cC & \rightarrow & cc \end{array} \right\}, S\rangle
\end{array}
$$

Grammaire v/s Reconnaisseur

- Une **grammaire** d'un langage L permet de générer tous les mots appartenant à L.
- Un **reconnaisseur** pour un langage L est un programme qui prend en entrée un mot m et répond **oui** si m appartient à L et **non** sinon.
- Pour chaque classe de grammaire, il existe une classe de reconnaisseurs qui définit la même classe de langages.

Type de grammaire	**Type de reconnaisseur**
régulière	Automate fini
hors contexte	Automate à pile
contextuelle	Automate linéairement borné
sans restriction	Machine de Turing