Chapitre 9

Grammaires non contextuelles

Hugo Salou MPI*

Dernière mise à jour le 13 mai 2023

Table des matières

Ce chapitre se rattache à la branche de l'informatique des langages formels. On l'a étudié au chapitre 1 avec les automates, et au chapitre 4 avec les machines. Pour le moment, nous avons 5 classes de langages : (1) les langages finis, (2) les langages locaux, (3) les langages réguliers, (4) les langages décidables en temps polynomial, (5) les langages décidables. L'objectif de ce chapitre se situe entre les points (3) et (4).

Intéressons-nous à un langage particulier, le langage des programmes OCaml avec une syntaxe valide. Ce langage est il régulier ? Non. On peut considérer l'expression

$$e_n = \text{« } \underbrace{((\cdots(}_{n}0\underbrace{)\cdots))}_{n} \text{ »}$$

qui est une expression OCaml valide. Par application du lemme de l'étoile, il n'est pas reconnaissable par un automate à N états en considérant e_{N+1}.

L'ensemble $\mathcal{B}$ des mots bien parenthésés est le plus petit ensemble tel que

— $\varepsilon \in \mathcal{B}$
— si $u \in \mathcal{B}$ et $v \in \mathcal{B}$, alors $u \cdot v \in \mathcal{B}$
— si $u \in \mathcal{B}$, alors $(\cdot u \cdot) \in \mathcal{B}$.

Une telle définition de langage est appelée une grammaire. Un autre exemple de langage est la grammaire de la langue française :

— phrase : sujet + verbe + complément,
— complément : COD + complément,
— complément : CCL + complément,
— complément : ε.

Avec cette définition [1], on peut reconnaître des phrases simples comme

$$\text{« } \underbrace{\text{Matthieu}}_{\text{sujet}} \ \underbrace{\text{aime}}_{\text{verbe}} \ \underbrace{\text{les trains}}_{\text{complément}} \text{. »}$$

1 Définition, vocabulaire, propriétés

1.1 Grammaires non contextuelles

Définition : On se munit d'un alphabet Σ qu'on appelle *terminaux*. On se munit d'un ensemble de symboles $\mathcal{V}$ qu'on appelle *non-terminaux*. On suppose $\mathcal{V} \cap \Sigma$.

Exemple :
Dans la suite, on pose $\Sigma = \{(,)\}$ et $\mathcal{V} = \{\mathrm{B}\}$.

Définition : On appelle *règle de production* la donnée
— d'un symbole $V \in \mathcal{V}$,
— d'un mot $w_1 w_2 \dots w_n$ sur l'alphabet $\mathcal{V} \cup \Sigma$,
que l'on note $V \to w_1 w_2 \dots w_n$.

Exemple :
L'ensemble $\mathcal{B}$ est décrit par les règles
— $\mathrm{B} \to \varepsilon$,
— $\mathrm{B} \to \mathrm{BB}$,

1. On néglige les règles manquantes à cette définition de la grammaire française.

— $\mathrm{B} \to (\mathrm{B})$.

Définition (Grammaire non contextuelle) **:** Une *grammaire non contextuelle* est la donnée de

— un alphabet de non-terminaux $\mathscr{V}$,
— un alphabet de terminaux Σ,[2]
— un ensemble fini de règles de production P,
— un symbole initial $S \in \mathscr{V}$,

que l'on note $(\mathscr{V}, \Sigma, P, S)$.

Exemple :
On note dans la suite

$$\mathscr{G} = \big(\{\mathrm{B}\}, \{(,)\}, \{\mathrm{B} \to \varepsilon, \mathrm{B} \to \mathrm{BB}, \mathrm{B} \to (\mathrm{B})\}, \mathrm{B}\big).$$

Interlude. Avec cette définition, on espère pouvoir appliquer ces règles pour définir des mots valides. Par exemple,

$$\mathrm{B} \overset{?}{\rightsquigarrow} \mathrm{BB} \overset{?}{\rightsquigarrow} (\mathrm{B})\mathrm{B} \overset{?}{\rightsquigarrow} (\mathrm{B})(\mathrm{B}) \overset{?}{\rightsquigarrow} (\mathrm{BB})(\mathrm{B}) \overset{?}{\rightsquigarrow} (\mathrm{BB})() \overset{?}{\rightsquigarrow} (\mathrm{B})() \overset{?}{\rightsquigarrow} ((\mathrm{B}))() \overset{?}{\rightsquigarrow} (())().$$

C'est l'objectif de la sous-section suivante.

1.2 Dérivation

Définition (Dérivation immédiate) **:** Soit $\mathscr{G} = (\mathscr{V}, \Sigma, P, S)$ une grammaire. Soient u et v deux mots de $(\Sigma \cup \mathscr{V})^\star$. On dit que v *dérive immédiatement* de u dans la grammaire $\mathscr{G}$, que l'on note $u \Rightarrow v$, si

— il existe x et y deux mots de $(\Sigma \cup \mathscr{V})^\star$
— il existe $(V \to w_1 w_2 \dots w_n) \in P$

tels que $u = x \cdot V \cdot y$ et $v = x \cdot w_1 w_2 \dots w_n \cdot y$.

Exemple :
Ainsi, $u \Rightarrow v$.

Lemme (de composition) **:** Soit $\mathscr{G} = (\mathscr{V}, \Sigma, P, S)$ une grammaire. Soient $u, v \in (\Sigma \cup \mathscr{V})^\star$ tels que $u \Rightarrow v$. Soit $w \in (\Sigma \cup \mathscr{V})^\star$. On a $u \cdot w \Rightarrow v \cdot w$ et $w \cdot u \Rightarrow w \cdot v$.

Preuve :
À faire à la maison. □

On propose trois définitions différentes mais équivalentes pour la dérivation $\overset{\star}{\Rightarrow}$.

2. avec "terminaux/non-terminaux" vient l'implication que $\mathscr{V} \cap \Sigma = \varnothing$

Définition : Étant donné une grammaire $\mathcal{G} = (\mathcal{V}, \Sigma, P, S)$, on étend $\Rightarrow$ en sa cloture réflexive et transitive, que l'on note $\overset{\star}{\Rightarrow}$, appelé *dérivation*.

Définition : On définit $\overset{\star}{\Rightarrow}$ comme $u \overset{\star}{\Rightarrow} v \overset{\text{def.}}{\iff} \exists n \in \mathbb{N},\ \exists (u_0, u_1, \dots, u_n) \in ((\mathcal{V} \cup \Sigma)^\star)^{n+1}$ tels que $u_0 = u$, $u_n = v$ et $\forall i \in [\![0, n-1]\!],\ u_i \Rightarrow u_{i+1}$.

Définition : Soit la suite $(\Rightarrow^n)_{n\in\mathbb{N}}$ définie inductivement par

— $u \Rightarrow^0 v \overset{\text{def.}}{\iff} u = v$,

— $u \Rightarrow^{n+1} v \overset{\text{def.}}{\iff} u \Rightarrow^n v$ ou $\exists w \in (\Sigma \cup \mathcal{V})^\star,\ u \Rightarrow w$ et $w \Rightarrow^n v$.

On pose alors $\overset{\star}{\Rightarrow} = \bigcup_{n\in\mathbb{N}} \Rightarrow^n$.

Lemme (de composition*) : Soit $\mathcal{G} = (\mathcal{V}, \Sigma, P, S)$ une grammaire, et soient $u, v \in (\Sigma \cup \mathcal{V})^\star$ tels que $u \Rightarrow^p v$, pour $p \in \mathbb{N}$. Et, soient $(w, t) \in (\Sigma \cup \mathcal{V})^2$ tels que $w \Rightarrow^q t$, pour $q \in \mathbb{N}$. Alors, $uw \Rightarrow^{p+q} vt$.

Preuve :
Soit $u_0 u_1 \dots u_p$ tels que $u_0 = u$, $u_p = v$ et $\forall i \in [\![0, p-1]\!],\ u_i \Rightarrow u_{i+1}$. Soit $w_0 w_1 \dots w_n$ tels que $w_0 = w$, $w_q = t$ et $\forall i \in [\![0, q-1]\!],\ w_i \Rightarrow w_{i+1}$. Soit alors la dérivation

$$uw = u_0 w \Rightarrow u_1 w \Rightarrow u_2 w \Rightarrow \cdots \Rightarrow u_p w \Rightarrow u_p w_1 \Rightarrow u_p w_2 \Rightarrow \cdots \Rightarrow u_p w_q = vt.$$

On a donc $uw \Rightarrow^{p+q} vt$. □

Définition : Soit $\mathcal{G} = (\mathcal{V}, \Sigma, P, S)$ une grammaire. Son langage est défini par

$$\mathcal{L}(\mathcal{G}) = \{u \in \Sigma^\star \mid S \overset{\star}{\Rightarrow} u\}.$$

Exemple :
Dans l'exemple précédent, on a

$$\mathcal{L}(\mathcal{G}) = \mathcal{B}.$$

Interlude : grammaires contextuelles. Dans une grammaire contextuelle, une règle de production peut-être de la forme $b\mathrm{B} \to a\mathrm{B}a\mathrm{B}$, où $\Sigma = \{a, b\}$. Ces règles dépendent du contexte, et non juste des symboles.

Lemme (de décomposition) : Étant donnée une grammaire non contextuelle $(\mathcal{V}, \Sigma, P, S)$, soit $w = w_1 \dots w_n$ un mot de n lettres tel qu'il existe $p \in \mathbb{N}$ et $v \in (\Sigma \cup \mathcal{V})^\star$ tel que $w \Rightarrow^p v$ alors il existe $\tilde{v}_1, \tilde{v}_2, \dots, \tilde{v}_n \in (\Sigma \cup \mathcal{V})^\star$ et $p_1, \dots, p_n \in \mathbb{N}$ tels que $w_1 \Rightarrow^{p_1} \tilde{v}_1$, $w_2 \Rightarrow^{p_2} \tilde{v}_2$, …, $w_n \Rightarrow^{p_n} \tilde{v}_n$, et $v = \tilde{v}_1 \cdots \tilde{v}_n$, et $\sum_{i=1}^n p_i = p$.

Preuve :
On procède par récurrence sur $p \in \mathbb{N}$.

— **cas de base ($p = 0$).** On a $w_1 w_2 \dots w_n \Rightarrow^0 v$, d'où $v = w_1 w_2 \dots w_n$. On choisit donc $\tilde{v}_i = w_i$ et $p_i = 0$ pour tout $i \in [\![1, n]\!]$. On a alors, pour tout $i \in [\![1, n]\!]$, $w_i \Rightarrow^0 \tilde{v}_i$, et $v = \tilde{v}_1 \dots \tilde{v}_n$ et $\sum_{i=1}^{n} p_i = 0 = p$.

— **hérédité.** Soit $w \Rightarrow w_1 \dots w_{q-1} u_1 u_2 \dots u_r w_{q+1} \dots w_n \Rightarrow^{p-1} v$, où $(w_q \to u_1 \dots u_r) \in P$. Soit donc, par hypothèse de récurrence, $\hat{v}_1, \hat{v}_2, \dots, \hat{v}_{n-1+r}$ tels que

  — $\forall i \in [\![1, q_1]\!]$, $w_i \Rightarrow^{s_i} \hat{v}_i$,
  — $\forall i \in [\![1, r]\!]$, $u_i \Rightarrow^{s_{i+q-1}} \hat{v}_{i+q-1}$,
  — $\forall i \in [\![q+1, n]\!]$, $w_i \Rightarrow^{s_{i+r-1}} \hat{v}_{i+r-1}$,
  — et $\sum_{i=1}^{n-1+r} s_i = p - 1$.

  On pose alors, pour tout $i \in [\![1, q-1]\!]$, $\tilde{v}_i = \hat{v}_i$, $\tilde{v}_q = \hat{v}_q \hat{v}_{q+1} \cdots \hat{v}_{q+r-1}$, et pour tout $i \in [\![q+1, n]\!]$, $\tilde{v}_i = \hat{v}_{i+r-1}$. On pose aussi, pour $i \in [\![1, q-1]\!]$, $p_i = s_i$, puis $p_q = 1 + \sum_{i=1}^{r} s_{i+q-1}$, et, pour $i \in [\![q+1, n]\!]$, $p_i = s_{i+r-1}$.

  On a clairement $v = \tilde{v}_1 \tilde{v}_2 \cdots \tilde{v}_n$. De plus, $\sum_{i=1}^{n} p_i = \left(\sum_{i=1}^{n+r-1} s_i\right) + 1 = (p-1) + 1 = p$. Et surtout, pour $i \in [\![1, q-1]\!]$, $w_i \Rightarrow^{s_i} \hat{v}_i$ donc $w_i \Rightarrow^{p_i} \tilde{v}_i$. Ainsi,

$$w_q \Rightarrow u_1 \dots u_r \Rightarrow^{\sum_{i=1}^{r} s_{i+q-1}} \underbrace{\hat{v}_q \dots \hat{v}_{q+r-1}}_{\tilde{v}_q},$$

  donc $w_q \Rightarrow^{p_q} \tilde{v}_q$ et, pour $i \in [\![q+1, n]\!]$, $w_i \Rightarrow^{s_{i+r-1}} \hat{v}_{r+r-1}$ et $w_i \Rightarrow^{p_i} \tilde{v}_i$.

□

1.3 Preuves par induction

Propriété (principe d'induction) **:** Étant donnée une grammaire $\mathcal{G} = (\mathcal{V}, \Sigma, P, S)$, et un non-terminal $V \in \mathcal{V}$, on note localement $V_\downarrow = \{w \in \Sigma^\star \mid V \overset{\star}{\Rightarrow} w\}$.[3] Soit alors un ensemble de propriétés $\mathcal{P}_V$ pour $V \in \mathcal{V}$, tel que, $\forall (V \to w_1 w_2 \dots w_n) \in P$, $\forall (\hat{w}_1, \hat{w}_2, \dots, \hat{w}_m) \in (\Sigma^\star)^m$

$$\left(\forall i \in [\![1, m]\!], \begin{cases} w_i \in \Sigma \implies \hat{w}_i = w_i \\ w_i \in \mathcal{V} \implies \hat{w}_i \text{ vérifie } \mathcal{P}_{w_i} \end{cases}\right) \implies \hat{w}_i \dots \hat{w}_n \text{ vérifie } \mathcal{P}_V,$$

alors pour tout $V \in \mathcal{V}$, $V_\downarrow$ vérifie $\mathcal{P}_V$.

Exemple :
On considère $\mathcal{G}$ la grammaire

$$\mathcal{G} = (\{\mathrm{P}, \mathrm{I}, \mathrm{X}\}, \{a\}, \{\mathrm{P} \to \varepsilon, \mathrm{I} \to a, \mathrm{P} \to a\mathrm{P}a, \mathrm{I} \to a\mathrm{I}a, \mathrm{X} \to \mathrm{IPI}\}, \mathrm{X}).$$

On pose les propriétés $\mathcal{P}_\mathrm{P}(w)$: « $|w|$ est pair » ; $\mathcal{P}_\mathrm{I}(w)$: « $|w|$ est impair » ; et, $\mathcal{P}_\mathrm{X}(w)$: « $|w|$ est pair. »

— **cas $\mathrm{P} \to \varepsilon$.** Le mot ε est de taille pair donc $\mathcal{P}_\mathrm{P}(\varepsilon)$ est vrai.
— **cas $\mathrm{I} \to a$.** $|a|$ est impair donc $\mathcal{P}_\mathrm{I}(a)$ est vrai.
— **cas $\mathrm{P} \to a\mathrm{P}a$.** Soit donc $w = aw'a$ avec w' vérifiant $\mathcal{P}_\mathrm{P}$. Alors, $|w| = 2 + |w'|$ qui est pair.
— **cas $\mathrm{I} \to a\mathrm{I}a$.** Soit donc $w = aw'a$ avec w' vérifiant $\mathcal{P}_\mathrm{I}$. Alors, $|w| = 2 + |w'|$ qui est impair.
— **cas $\mathrm{X} \to \mathrm{IPI}$.** Soit donc $w = xyz$, avec x et z vérifiant $\mathcal{P}_\mathrm{I}$, et y vérifiant $\mathcal{P}_\mathrm{P}$. Alors, $|w| = |x| + |y| + |z|$, qui est pair.

3. Avec cette définition, $\mathscr{L}(\mathcal{G}) = S_\downarrow$.

1.4 Définitions équivalentes

Comment représenter une dérivation en machine ? On considère les règles de production $\mathrm{X} \to \mathrm{XX}$ et $\mathrm{X} \to a$. On peut, par exemple, représenter une dérivation par un ensemble de possibilités :

$$\mathrm{X} \Rightarrow \mathrm{XX} \Rightarrow \{a\mathrm{X}, \mathrm{X}a, \mathrm{XXX}\} \Rightarrow \cdots .$$

Mais, avec une telle définition, il y a explosions du nombre de possibilités.

Définition (Dérivation immédiatement gauche (resp. droite)) **:** Étant donnée une grammaire $\mathcal{G} = (\mathcal{V}, \Sigma, P, I)$, et étant donnés deux mots u et v de $(\Sigma \cup \mathcal{V})^\star$, on dit que u *dérive immédiatement à gauche* (resp. droite) de u dès lors qu'il existe $x \in \Sigma^\star$, $y \in (\Sigma \cup \mathcal{V})^\star$ (resp. $x \in (\Sigma \cup \mathcal{V})^\star$ et $y \in \Sigma^\star$), et $(V \to w_1 \dots w_n) \in P$ tels que $u = xVy$ et $v = x \cdot w_1 w_2 \dots w_n \cdot y$. On note alors $u \Rightarrow_{\mathrm{g}} v$ (resp. $u \Rightarrow_{\mathrm{d}} v$). On définit, de la même manière que pour la dérivation simple, $\overset{\star}{\Rightarrow}_{\mathrm{g}}$ et $\overset{\star}{\Rightarrow}_{\mathrm{d}}$.

Exemple :
On considère une grammaire ayant pour règles de production $\mathrm{X} \to \mathrm{XX}$ et $\mathrm{X} \to a$. A-t-on

— $\mathrm{X} \overset{\star}{\Rightarrow}_{\mathrm{g}} a\mathrm{X}$? ✓ ($\mathrm{X} \Rightarrow_{\mathrm{g}} \mathrm{XX} \Rightarrow_{\mathrm{g}} a\mathrm{X}$)

— $\mathrm{X} \overset{\star}{\Rightarrow}_{\mathrm{g}} \mathrm{X}a$? ✗

Définition (Arbre de dérivation) **:** Étant donnée une grammaire $\mathcal{G} = (\mathcal{V}, \Sigma, P, S)$, on appelle *arbre de dérivation* un arbre dont les nœuds sont étiquetés par des éléments de $\{\varepsilon\} \cup \mathcal{V} \cup \Sigma$ et tel que

— tout nœud interne (ayant des fils) est étiquetés par un élément de $\mathcal{V}$,

— la racine est étiquetée par S,

— tout nœud interne ayant une étiquette V et ayant des fils $T_1, \dots, T_n$ où $n \neq 0$ dont les racines sont étiquetées par $w_1, \dots, w_n$ avec $(V \to w_1 \dots w_n) \in P$.

— tout nœud interne d'étiquette V ayant pour unique fils l'arbre feuille réduit à ε et tel que $(V \to \varepsilon) \in P$.

Dans la suite du chapitre, on fixe $\mathcal{G} = (\mathcal{V}, \Sigma, I, P)$ une grammaire.

Exemple :

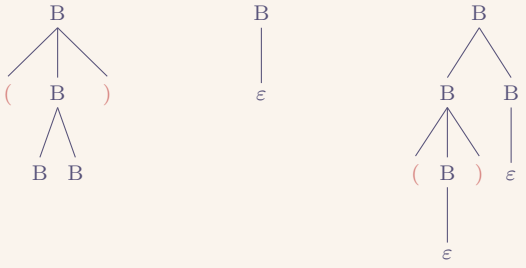


Figure 1 – Exemple d'arbres de dérivation

Définition (« production » d'un arbre) **:** On définit inductivement la fonction prod de l'ensemble des arbres de la grammaire $\mathcal{G}$ vers $(\Sigma \cup \mathcal{V})^\star$, comme

— $\mathrm{prod}\big(\mathrm{Leaf}(x)\big) = x$,

— $\mathrm{prod}\big(\mathrm{Node}(\_, [T_1, \dots, T_n])\big) = \mathrm{prod}(T_1) \cdot \mathrm{prod}(T_2) \cdot \dots \cdot \mathrm{prod}(T_n)$.

Exemple :
Dans les arbres de dérivation exemples précédents, la fonction prod retourne (BB), ε et $(\,)$.

Remarque (Notation) :
On dit qu'un arbre de dérivation T est « *clos* » lorsque $\mathrm{prod}(T) \in \Sigma^\star$.

Exemple :
Dans les exemples précédents, le premier arbre n'est pas « clos » mais les deux suivants le sont.

Propriété : Étant donné $w \in \Sigma^\star$, il est équivalent de dire que

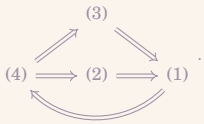
(1) $w \in \mathscr{L}(\mathcal{G})$,

(2) $S \overset{\star}{\Rightarrow}_{\mathrm{g}} w$

(3) $S \overset{\star}{\Rightarrow}_{\mathrm{d}} w$

(4) il existe un arbre de dérivation T dont w est le produit.

Preuve :
On procède la démonstration dans l'ordre suivant.



— **(4) $\Longrightarrow$ (2).** Soit la propriété $\mathcal{P}(T)$: « si $w \in \Sigma^\star$ admet T comme arbre de dérivation (w est le produit de T) avec T un arbre de dérivation généralisé, enraciné en $V \in \mathcal{V}$, alors $V \overset{\star}{\Rightarrow}_{\mathrm{g}} w$. » Montrons cette propriété par induction.

— Si $T = \mathrm{Leaf}(x)$ avec $x \in \Sigma \cup \{\varepsilon\}$, alors ok par un arbre de dérivation.

— Si $\mathrm{T} = \mathrm{Node}(V, [T_1, \dots, T_n])$, alors, pout $i \in [\![1, n]\!]$, raisonnons par disjonction.

— Si T_i a une racine d'étiquette $w_i \in \Sigma$, alors $\mathrm{prod}(T_i) = w_i$, donc $w_i \overset{\star}{\Rightarrow}_{\mathrm{g}} \mathrm{prod}(T_i)$

— Si T_i a une racine d'étiquette $w_i \in \mathcal{V}$, alors, par hypothèse d'induction sur T_i, on a $w_i \overset{\star}{\Rightarrow}_{\mathrm{g}} \mathrm{prod}(T_i)$.

Ainsi, par concaténation, $\mathrm{prod}(T) = \mathrm{prod}(T_1) \cdot \dots \cdot \mathrm{prod}(T_n)$. Or, $(V \to w_1 \dots w_n) \in P$. Alors, considérons la dérivation

$$\begin{aligned} V &\Rightarrow_{\mathrm{g}} w_1 w_2 \dots w_n \\ &\overset{\star}{\Rightarrow}_{\mathrm{g}} \mathrm{prod}(T_1) \cdot w_2 \dots w_n \\ &\overset{\star}{\Rightarrow}_{\mathrm{g}} \mathrm{prod}(T_1) \cdot \mathrm{prod}(T_2) \\ &\overset{\star}{\Rightarrow}_{\mathrm{g}} \dots \overset{\star}{\Rightarrow}_{\mathrm{g}} \mathrm{prod}(T_1) \cdot \mathrm{prod}(T_2) \cdot \dots \cdot \mathrm{prod}(T_n) \end{aligned}$$

— **(2) $\Longrightarrow$ (1).** Vrai car $\Rightarrow_{\mathrm{g}}$ est « inclus dans » $\Rightarrow$ (c'est un cas particulier).

— **(1) $\Longrightarrow$ (4).** Soit la propriété $\mathcal{P}_n$ « $\forall V \in \mathcal{V}, \forall w \in \Sigma^\star$, si $V \Rightarrow^n w$, alors w admet un arbre de dérivation généralisé enraciné en V. » Montrons le par induction.

— Si $n = 0$, absurde.
— Si $V \Rightarrow^n w$ avec $n \geqslant 1$, donc $V \Rightarrow w_1 w_2 \dots w \Rightarrow^{n-1} w$ donc $(V \to w_1 w_2 \dots w_n) \in P$, alors, par lemme de décompositions, il existe $\tilde{w}_1, \dots, \tilde{w}_p$ tels que $w = \tilde{w}_1 \cdot \dots \cdot \tilde{w}_p$ et $\forall i \in [\![1, p]\!]$, $w_i \Rightarrow^{p_i} \tilde{w}_i$ avec $\sum_{i=1}^{p} p_i = n - 1$. D'où, $\forall i \in [\![1, p]\!]$, $p_i < n$. Par hypothèse d'induction, il existe, pour tout $i \in [\![1, p]\!]$, un arbre T_i produisant $\tilde{w}_i$ enraciné en w_i. Soit alors $T = \mathrm{Node}(w, [T_1, \dots, T_p])$. Ainsi, $\mathrm{prod}(T) = \tilde{w}_1 \cdot \dots \cdot \tilde{w}_p = w$. Dans l'éventualité où l'un des p_i aurait le mauvais goût d'être nul, on fabrique, à la place, l'arbre $T_i = \mathrm{Leaf}(w_i)$.

□

Définition : Une grammaire est dite *ambigüe* s'il existe un mot w de son langage admettant au moins deux arbres de dérivations.

Exemple :
On considère la grammaire de non-terminal initial B ayant pour règles de production $\mathrm{F} \to 0 \mid 1 \mid \cdots \mid 9$ et $\mathrm{B} \to \mathrm{B} + \mathrm{B} \mid \mathrm{B} - \mathrm{B} \mid \mathrm{F}$. Le mot « $1 - 1 + 9$ » admet les deux arbres de dérivations ci-dessous.

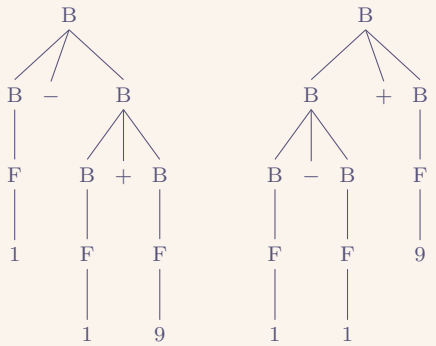


Figure 2 – Arbres de dérivations de « $1 - 1 + 9$ »

Interlude : langages algébriques. Les langages reconnus par des grammaires non-contextuelles sont des solutions d'équations polynômiales, où la multiplication correspond à la concaténation et l'addition correspond à l'union. D'où le terme langage *algébrique*.

Définition : Deux grammaires $\mathcal{G}_1$ et $\mathcal{G}_2$ sont dites *faiblement équivalentes* dès lors que $\mathcal{L}(\mathcal{G}_1) = \mathcal{L}(\mathcal{G}_2)$.

Exemple :
On considère la grammaire $\mathcal{G}_1$ de règle de production $\mathrm{S} \to a\mathrm{S} \mid \varepsilon$, et la grammaire $\mathcal{G}_2$ de règle de production $\mathrm{S} \to \mathrm{SS} \mid a \mid \varepsilon$. Les deux grammaires $\mathcal{G}_1$ et $\mathcal{G}_2$ sont faiblement équivalentes. En effet, $\mathcal{L}(\mathcal{G}_1) = \mathcal{L}(a^\star) = \mathcal{L}(\mathcal{G}_2)$. Mais, elles ne sont pas fortement équivalentes. [4]

4. Deux grammaires sont fortement équivalentes si elles produisent les mêmes arbres de dérivation.

2 La hiérarchie de Chomsky

Le terme « *hiérarchie de Chomsky* » n'est pas au programme. Dans cette partie, on traite des inclusions avec les autres familles des langages au programme.

2.1 Avec les langages réguliers

Propriété : Soient $\mathcal{G}_1$ et $\mathcal{G}_2$ des grammaires non contextuelles. Alors,

1. $\mathcal{L}(\mathcal{G}_1) \cup \mathcal{L}(\mathcal{G}_2)$ est reconnu par une grammaire non-contextuelle ;
2. $\mathcal{L}(\mathcal{G}_1) \cdot \mathcal{L}(\mathcal{G}_2)$ est reconnu par une grammaire non-contextuelle ;
3. $(\mathcal{L}(\mathcal{G}_1))^\star$ est reconnu par une grammaire non-contextuelle.

Preuve :
Soient $\mathcal{G}_1 = (\mathcal{V}_1, \Sigma, P_1, S_1)$ et $\mathcal{G}_2 = (\mathcal{V}_2, \Sigma, P_2, S_2)$. On peut supposer, quitte à renommer les non-terminaux, que $\mathcal{V}_1 \cap \mathcal{V}_2 = \varnothing$.

1. Soit alors $S \notin \mathcal{V}_1 \cup \mathcal{V}_2$. On pose $\mathcal{G} = (\mathcal{V}_1 \cup \mathcal{V}_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\}, S)$. Soit $w \in \Sigma^\star$.

$$\begin{aligned} w \in \mathcal{L}(\mathcal{G}) &\iff S \overset{\star}{\Rightarrow}_{\mathcal{G}} w \\ &\iff (S \Rightarrow_{\mathcal{G}} S_1 \overset{\star}{\Rightarrow}_{\mathcal{G}} w) \text{ ou } (S \Rightarrow_{\mathcal{G}} S_2 \overset{\star}{\Rightarrow}_{\mathcal{G}} w) \\ &\iff (S_1 \overset{\star}{\Rightarrow}_{\mathcal{G}} w) \text{ ou } (S_2 \overset{\star}{\Rightarrow}_{\mathcal{G}} w) \\ &\iff (S_1 \overset{\star}{\Rightarrow}_{\mathcal{G}_1} w) \text{ ou } (S_2 \overset{\star}{\Rightarrow}_{\mathcal{G}_2} w) \\ &\iff w \in \mathcal{L}(\mathcal{G}_1) \cup \mathcal{L}(\mathcal{G}_2) \end{aligned}$$

2. Soit alors $S \notin \mathcal{V}_1 \cup \mathcal{V}_2$. On pose $\mathcal{G} = (\mathcal{V}_1 \cup \mathcal{V}_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \to S_1 \cdot S_2\}, S)$. Soit $w \in \Sigma^\star$.

$$\begin{aligned} w \in \mathcal{L}(\mathcal{G}) &\iff S \overset{\star}{\Rightarrow}_{\mathcal{G}} w \\ &\iff S \Rightarrow S_1 \cdot S_2 \overset{\star}{\Rightarrow}_{\mathcal{G}} w \\ &\iff \exists (u,v) \in (\Sigma^\star)^2,\ S_1 \overset{\star}{\Rightarrow}_{\mathcal{G}} u \text{ et } S_2 \overset{\star}{\Rightarrow}_{\mathcal{G}} v \text{ et } w = u \cdot v \\ &\iff \exists (u,v) \in (\Sigma^\star)^2,\ S_1 \overset{\star}{\Rightarrow}_{\mathcal{G}_1} u \text{ et } S_2 \overset{\star}{\Rightarrow}_{\mathcal{G}_2} v \text{ et } w = u \cdot v \\ &\iff \exists (u,v) \in (\Sigma^\star)^2,\ u \in \mathcal{L}(\mathcal{G}_1) \text{ et } v \in \mathcal{L}(\mathcal{G}_2) \text{ et } w = u \cdot v \\ &\iff w \in \mathcal{L}(\mathcal{G}_1) \cdot \mathcal{L}(\mathcal{G}_2) \end{aligned}$$

3. Soit alors $S \notin \mathcal{V}_1 \cup \mathcal{V}_2$. On pose $\mathcal{G} = (\mathcal{V}_1 \cup \{S\}, \Sigma, P_1 \cup \{S \to S \cdot S_1 \mid \varepsilon\}, S)$. Soit $w \in \Sigma^\star$.

$$\begin{aligned} w \in \mathcal{L}(\mathcal{G}) &\iff S \overset{\star}{\Rightarrow}_{\mathcal{G}} w \\ &\iff S \overset{\star}{\Rightarrow}_{\mathcal{G}, gw} \\ &\iff S \Rightarrow \overbrace{S \cdot S_1 \Rightarrow SS_1S_1 \Rightarrow \cdots \Rightarrow \underbrace{S_1S_1\cdots S_1}_{n}}^{n} \overset{\star}{\Rightarrow}_{\mathcal{G}} w^\star \\ &\iff \exists n \in \mathbb{N},\ \exists (\tilde{w}_1, \ldots, \tilde{w}_n),\ (\forall i \in [\![1,n]\!],\ S_1 \overset{\star}{\Rightarrow}_{\mathcal{G}} \tilde{w}_i) \text{ et } w = \tilde{w}_1 \cdots \tilde{w}_n \\ &\iff \exists n \in \mathbb{N},\ \exists (\tilde{w}_1, \ldots, \tilde{w}_n),\ (\forall i \in [\![1,n]\!],\ S_1 \overset{\star}{\Rightarrow}_{\mathcal{G}_1} \tilde{w}_i) \text{ et } w = \tilde{w}_1 \cdots \tilde{w}_n \\ &\iff \exists n \in \mathbb{N},\ \exists (\tilde{w}_1, \ldots, \tilde{w}_n),\ (\forall i \in [\![1,n]\!],\ \tilde{w}_i \in \mathcal{L}(\mathcal{G}_1)) \text{ et } w = \tilde{w}_1 \cdots \tilde{w}_n \\ &\iff w \in (\mathcal{L}(\mathcal{G}_1))^\star \end{aligned}$$

□

Théorème : Tout langage régulier est reconnu par une grammaire non contextuelle.

Preuve :
On a déjà montré que les grammaires non-contextuelles sont stables par union, concaténation et passage à l'étoile. Il ne reste qu'à montrer le résultat sur les cas de base. On a

— $\{\varepsilon\} = \mathscr{L}((\{S\}, \Sigma, \{S \to \varepsilon\}, S))$,
— $\varnothing = \mathscr{L}((\{S\}, \Sigma, \varnothing, S))$, (▷ voir TD 14.)
— $\{\ell\} = \mathscr{L}((\{S\}, \Sigma, \{S \to \ell\}, S))$.

Il suffit alors de conclure par induction. □

REMARQUE :
L'inclusion précédente est stricte. En effet le langage $\{a^n \cdot b^n \mid n \in \mathbb{N}\}$ est reconnu par une grammaire non contextuelle mais n'est pas un langage régulier.

REMARQUE (Digression) :

« *Tout langage est-il le langage d'une grammaire non contextuelle ?* »

On procède par un argument de taille d'ensembles. Soit $\mathcal{G} = (\mathcal{V}, \Sigma, P, I)$ une grammaire. On considère $\Sigma = \{0, 1\}$. On pose $|\mathcal{G}| = |\Sigma| + |\mathcal{V}| + \sum_{(V \to w_1 \dots w_n) \in P} (n + 1) + 1$. On considère $\mathbb{G}_n = \{\mathcal{G} \mid |\mathcal{G}| = n\}$. L'ensemble $\mathbb{G} = \bigcup_{n \in \mathbb{N}} \mathbb{G}_n$ est dénombrable comme union dénombrable d'ensembles finis. Montrons qu'il existe une bijection entre $\wp(\{0, 1\}^\star)$ et $[0, 1[$. À tout $x \in [0, 1[$, on peut poser $x = 0, x_1 x_2 \dots x_n \dots$. D'où,

$$[0, 1[\xleftrightarrow[\text{encodage binaire}]{\text{bijection}} (\mathbb{N} \to \{0, 1\}) \xleftrightarrow[\mathbb{1}]{\text{bijection}} \wp(\mathbb{N}) \xleftrightarrow[\text{encodage binaire}]{\text{bijection}} \wp(\{0, 1\}^\star).$$

2.2 Lien avec les langages décidables

Propriété : Les langages des grammaires non contextuelles sont des langages décidables. □

La preuve de cette propriété est dans le TD 15. (On peut montrer, par programmation dynamique, que l'appartenance d'un mot à une grammaire non contextuelle est calculable en temps polynômial, en $\mathcal{O}(n^3)$.)

EXEMPLE :
▷ TD 15, exercice 1.

EXEMPLE :
On considère l'alphabet $\Sigma = \{C, LP, RP\}$ et les règles de production S $\to$ TS | C et T $\to$ $LP \cdot$ S $\cdot$ RP. Codons un programme reconnaissant la grammaire définie par ces règles. On représente Σ par le type `token` où C correspond à `C`, LP à `LP` et RP à `RP`.

```
type token = C | LP | RP
type word = token list

type non_term = S | T

(* closed derivation tree *)
type cdt =
  | Node of non_term * cdt list
  | Leaf of token option
```

```
10
11
12  exception Non
13
14  let rec parse_s (w: word): cdt * word =
15    match w with
16    | C ::  w' -> (Node(S, [Leaf(Some C)]), w')
17    | _        ->
18        let (g, w')  = parse_t w  in
19        let (d, w'') = parse_s w' in
20        (Node (S, [g; d]), w'')
21  and parse_t (w: word): cdt * word =
22    match w with
23    | LP :: w' -> begin
24        let (u, w'') = parse_s w' in
25        match w'' with
26        | RP :: w''' -> (Node (T,
27            [Leaf (Some LP); u; Leaf (Some RP)]), w''')
28        | _          -> raise Non
29      end
30    | _ -> raise Non
```

Code 1 – Programme reconnaissant une grammaire $\mathcal{G}$