

1.1 Langages formels

1.1.1 Monoïde

Déf. 1 (Alphabet)

Un **alphabet** X est un ensemble fini de symboles (lettres). La *taille* de l'alphabet est le nombre de symboles. On parle aussi de **vocabulaire**.

Déf. 2 (Mot)

Un *mot* sur l'alphabet X est une suite finie de lettres de X .

Les symboles qui composent le mot u peuvent donc être caractérisés par leur *position* dans la suite : on parle d'*occurrence*.

Le **mot vide**, que l'on notera ε (ou 1_X) est la suite qui ne comporte aucun élément. La longueur du mot u est notée $|u|$, c'est le nombre d'occurrences dans u . $|\varepsilon| = 0$.

Formellement, on définit $[p] = (1, 2, 3, 4, \dots, p)$ (suite entière ordonnée).

Alors le mot u sur l'alphabet X est une fonction

$$u : [p] \longrightarrow X$$

qui à chaque entier entre 1 et p associe un symbole.

Déf. 3 (Sous-mot, facteur)

w est un *sous-mot* de u si w est une sous-suite de lettres de u .

w est un *facteur* de u si w est un sous-mot de u , dont les lettres sont adjacentes dans u .

NB : L'ordre est conservé.

Déf. 4 (Concaténation)

Soient $[p] \xrightarrow{u} X$, $[q] \xrightarrow{w} X$. On définit la concaténation de u et w , notée uw (quelquefois $u.w$ ou $u \hat{ } w$) :

$$uw : [p+q] \longrightarrow X$$

$$uw_i = \begin{cases} u_i & \text{pour } i \in [1, p] \\ w_{i-p} & \text{pour } i \in [p+1, p+q] \end{cases}$$

Définition informelle : uw est le mot formé de tous les symboles de u suivis immédiatement de tous les symboles de w .

On démontre facilement que l'opération de concaténation ainsi définie a les propriétés suivantes :

- La concaténation est non commutative (en général)
- La concaténation est associative : $u(vw) = (uv)w = uvw$
- La concaténation admet un élément neutre, le mot vide, noté $\emptyset$ ou ε ou 1_X ou 1_X .

Cette opération munit donc l'ensemble des mots sur un alphabet X $((X, .), \text{noté } X^*)$ d'une

structure de *monoïde*. De plus, ce monoïde est *libre* car les éléments de l'alphabet, qu'on appelle la *base*, sont indépendants les uns des autres.

1.1.2 Langage formel

Déf. 5 (Langage)

Un langage sur un alphabet X est une partie de X^* (c'est-à-dire un sous-ensemble).

$L \subseteq X^*$. $L \in \mathcal{P}(X^*)$ ($= 2^{X^*}$)

Problèmes sur un langage

- Caractérisation
 - Reconnaissance
 - Classification
-

Opération ensemblistes courantes sur les langages :

$$\begin{aligned} L_1 \cup L_2 &= \{u / u \in L_1 \text{ ou } u \in L_2\} \\ L_1 \cap L_2 &= \{u / u \in L_1 \text{ et } u \in L_2\} \\ L_1 \setminus L_2 &= \{u / u \in L_1 \text{ et } u \notin L_2\} \end{aligned}$$

Opération induite par l'opération de concaténation, appelée produit :

$$L_1.L_2 = \{uv / u \in L_1 \text{ et } v \in L_2\}$$

On démontre facilement que l'opération de produit ainsi définie a les propriétés suivantes :

- Associative
- Admet un élément neutre, l'ensemble $\{\epsilon\}$.
- Admet un élément absorbant, l'ensemble vide $\emptyset$.
- Distributive par rapport à $\cup$.
- Non distributive par rapport à $\cap$.

Déf. 6 (Étoile)

Par analogie, on notera A^2 le produit $A.A$ pour un langage A .

En généralisant, on peut proposer la notation

$$\begin{aligned} A^0 &= \{\mathbb{1}_X\} \\ A^1 &= A \\ A^{i+1} &= A.A^i \end{aligned}$$

d'où

$$A^n = \{a_1 \dots a_n / a_i \in A\}$$

et définir ainsi l'*étoile* de A :

$$A^* = \bigcup_{n \geq 0} A^n$$

Remarque : La notation est cohérente : $X^* = \bigcup_{n \geq 0} X^n$ où $X^n = \{\text{mots de longueur } n\}$.

1.2 Grammaires formelles

1.2.1 Définition

Une **grammaire formelle** est un quadruplet $\langle X, V, S, P \rangle$ où

- X est l'alphabet (du langage engendré)
- V est un alphabet disjoint de X dit « *non terminal* »
- $S \in V$ est un élément distingué de V , appelé *axiome*
- P est un ensemble de « *règles (de production)* », c'est-à-dire une partie finie du produit cartésien $(X \cup V)^* V (X \cup V)^* \times (X \cup V)^*$.

Exemples $\mathcal{G}_2 = \langle \{ (,) \}, \{ S \}, S, \{ S \rightarrow \epsilon \mid (S)S \} \rangle$

$\mathcal{G}_3 = E \rightarrow E + T \mid T, T \rightarrow T \times F \mid F, F \rightarrow (E) \mid a$ (Grammaire des expressions arithmétiques avec priorité).

1.2.2 Dérivation/ré-écriture

Soient $\mathcal{G} = \langle X, V, S, P \rangle$ une grammaire, $(f, g) \in (X \cup V)^*$, r une règle de production de P , de la forme $r : A \rightarrow u$ ($A \in (X \cup V)^* V (X \cup V)^* \times (X \cup V)^*$).

- f se **réécrit** (ou **dérive immédiatement**) en g avec la règle r (notation $f \xrightarrow{r} g$) ssi $\exists v, w$ t.q. $f = vAw$ et $g = vuw$
- f se **réécrit** (ou **dérive**) en g dans la grammaire $\mathcal{G}$ (notation $f \xrightarrow{\mathcal{G}} g$) ssi $\exists r \in P$ t.q. $f \xrightarrow{r} g$.
- $f \xrightarrow{\mathcal{G}^*} g$ si $f = g$, ou $\exists f_1 = f, f_2, \dots, f_n = g$ t.q. $f_{i-1} \xrightarrow{\mathcal{G}} f_i$

On note $L_{\mathcal{G}}(f)$ l'ensemble des terminaux engendrés par f dans la grammaire $\mathcal{G}$.

$$L_{\mathcal{G}}(f) = \{g \in X^* / f \xrightarrow{\mathcal{G}^*} g\}$$

Par convention, on notera $L_{\mathcal{G}}$ le langage $L_{\mathcal{G}}(S)$.

Proto-phrase, dérivations équivalentes, dérivation gauche Les combinaisons de symboles produits par une dérivation comprennent typiquement des éléments des deux alphabets (jusqu'au mot final qui ne comprend plus de non-terminal, par définition). On parlera de **proto-phrase** (ou plus proprement de **proto-mot**) pour ces mots.

Voici un exemple de dérivation du mot $a + a * a$ par la grammaire $\mathcal{G}_2$ précédente (8 étapes) :

$$E \Rightarrow E + T \Rightarrow E + T * F \Rightarrow T + T * F \Rightarrow T + F * F \Rightarrow T + a * F \Rightarrow F + a * F \Rightarrow a + a * F \Rightarrow a + a * a$$

On obtient des dérivations différentes selon l'ordre dans lequel on décide de dériver les non-terminaux. Toutes les dérivations qui ne diffèrent que par l'ordre sont dites **équivalentes**. Parmi les dérivations équivalentes d'un même mot par une grammaire, on en distingue deux, que l'on appelle **dérivation gauche** (respectivement droite). Dans une dérivation gauche, le non-terminal dérivé à chaque étape est celui qui se trouve le plus à gauche dans la proto-phrase. Par exemple, voici la dérivation gauche équivalente à la dérivation donnée plus haut :

$$E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow a + T \Rightarrow a + T * F \Rightarrow a + F * F \Rightarrow a + a * F \Rightarrow a + a * a$$

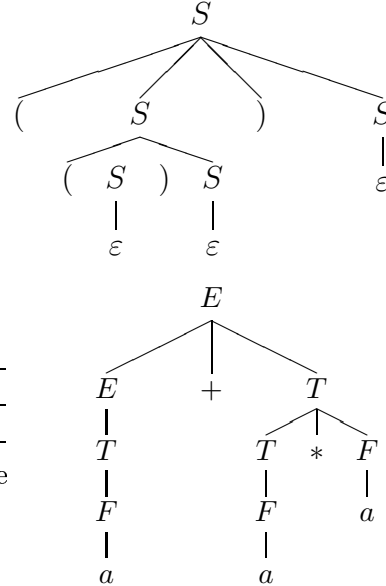
1.2.3 Arbre de dérivation

Il existe un moyen de représenter l'ensemble des dérivations équivalentes, au moyen d'un **arbre de dérivation** qui représente les dérivations indépendamment de l'ordre.

Exemple Grammaire $\mathcal{G}_2$, soit le mot $(())$.

$$\langle \{ (,) \}, \{ S \}, S, \{ S \rightarrow \varepsilon \mid (S)S \} \rangle$$

$$S \rightarrow (S)S \rightarrow ((S)S)S \rightarrow ((S)S) \rightarrow ((S)) \rightarrow (())$$



En plus de cet aspect « factorisation », l'arbre présente un autre avantage crucial : il donne une décomposition structurelle du mot reconnu. Ainsi, pour reprendre l'exemple $a + a * a$, la grammaire précédente nous donne l'arbre de dérivation :

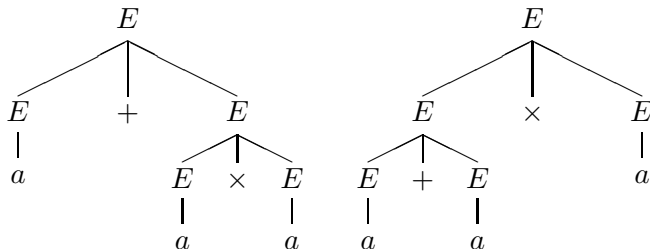
Cette décomposition structurelle ne sert à rien pour la reconnaissance du langage, mais en revanche elle est très utile pour l'interprétation des mots du langage : en effet, le plus souvent, la signification que vont prendre les mots du langage (par exemple un programme dans un langage de programmation, ou une formule logique) est directement liée à la structure interne des mots. Dans l'exemple, l'arbre syntaxique suggère une interprétation prioritaire de la multiplication par rapport à l'addition.

1.2.4 Ambiguïté

Si une grammaire G permet d'attribuer plus d'un arbre de dérivation à un mot $m \in L(G)$, elle est dite *ambiguë*. La grammaire

$$G_2 = \langle \{ E \}, \{ a, +, \times \}, E, \{ E \rightarrow E + E \mid E \times E \mid a \} \rangle$$

par exemple, est ambiguë, elle permet d'attribuer au mot $a + a \times a$ les deux arbres ci-dessous :



On pourra remarquer que le langage engendré par les deux grammaires est identique. Elles ont le même **pouvoir génératif faible**. En revanche, elles n'assignent pas la même structure syntaxique aux mots reconnus, et dans ce sens, elles n'ont pas le même **pouvoir génératif fort**.

L'ambiguïté est une propriété gênante pour les grammaires algébriques, puisqu'elles conduisent à donner plusieurs interprétations différentes à certains mots, ce qui est rarement souhaité (langages de programmation, par exemple).

Mais il faut noter que parmi les langages algébriques il en existe qui ne peuvent être reconnus que par une grammaire ambiguë. Ces langages sont dits intrinsèquement ambigus.

1.2.5 Classes de grammaire

La fameuse « hiérarchie de Chomsky » :

type 0 Aucune restriction sur $P \subset (X \cup V)^* V (X \cup V)^* \times (X \cup V)^*$.

type 1 (*grammaires contextuelles, context-sensitive*) Tout élément de P est de la forme $(u_1 S u_2, u_1 m u_2)$, où u_1 et $u_2 \in (X \cup V)^*$, $S \in V$ et $m \in (X \cup V)^+$.¹

type 2 (*grammaires algébriques, context-free*)

Tout élément de P est de la forme (S, m) , où $S \in V$ et $m \in (X \cup V)^*$.

type 3 (*grammaires régulières*)

Tout élément de P est de la forme (S, m) , où $S \in V$ et $m \in X.V \cup X \cup \{\varepsilon\}$.

Exemples

type 3

$$\begin{array}{lcl} S & \rightarrow & +E \\ & | & -E \\ & | & E \\ E & \rightarrow & 0E \\ & | & ,D \\ & | & 0 \\ D & \rightarrow & 0D \\ & | & 0 \end{array}$$

type 2

$$\begin{array}{lcl} S & \rightarrow & SN \, SV \\ SN & \rightarrow & Det \, N \\ & & \text{etc...} \end{array}$$

type 1

$$\begin{array}{lcl} S & \rightarrow & TZ \\ T & \rightarrow & 0U1 \\ & | & 01 \\ U & \rightarrow & 0U1C \\ & | & 01C \\ C1 & \rightarrow & 1C \\ CZ & \rightarrow & Z2 \\ 1Z & \rightarrow & 12 \end{array}$$

type 0

$$\begin{array}{lcl} S & \rightarrow & TZ \\ T & \rightarrow & 0T1C \\ & | & \varepsilon \\ C1 & \rightarrow & 1C \\ CZ & \rightarrow & Z2 \\ 1Z & \rightarrow & 1 \end{array}$$

¹Autre formulation équivalente : le membre droit est au moins aussi long que le membre gauche. ε n'appartient à aucun langage de type 1.