2.0 Rappels sur les grammaires syntagmatiques

2.0.1 Définition

Une **grammaire formelle** est un quadruplet $\langle X, V, S, P\rangle$ où
- X est l'alphabet (du langage engendré) (noté aussi Σ)
- V est un alphabet disjoint de X dit « *non terminal* » (noté aussi N)
- $S \in V$ 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

$$\mathcal{G}_1 = \left\langle \{jean, dort\}, \{Np, SN, SV, V, S\}, S, \left\{ \begin{array}{l} S \to SN\ SV \\ SN \to Np \\ SV \to V \\ Np \to jean \\ V \to dort \end{array} \right\} \right\rangle\}$$

$\mathcal{G}_2 = \langle \{(,)\}, \{S\}, S, \{S \longrightarrow \varepsilon \mid (S)S\}\rangle$

$G_3 = E \to E + T \mid T, T \to T \times F \mid F, F \to (E) \mid a$
(Grammaire des expressions arithmétiques avec priorité).

2.0.2 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 $m \in (X \cup V)^+$.

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

type 3

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

type 1

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

type 2

$$\begin{array}{lcl} S & \to & SN\ SV \\ SN & \to & Det\ N \end{array}$$

etc...

type 0

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

2.0.3 Récriture/dérivation, arbre syntaxique

2.0.3.1 (Sens de) dérivation

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 $r : A \longrightarrow u$ ($u \in (X \cup V)^*$).

- 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 $f = g$
  $\exists f_1 = f, f_2, ..., f_n = g$ t.q. $f_{i-1} \longrightarrow f_i$

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 $L_{\mathcal{G}}(S)$.

Exemple de dérivation (avec $\mathcal{G}_2$ plus haut) : $S \to (S)S \to ()S \to () \in L_G(S)$, etc. (aussi ((())), ()()(), ((()()())), ...)

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 :

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

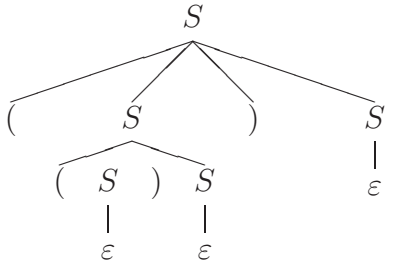
2.0.3.2 Arbre de dérivation

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.

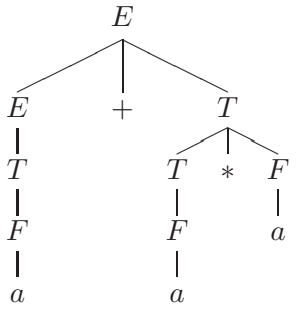
Exemple Grammaire $\mathcal{G}_2$, soit le mot (()).

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

$S \to (S)S \to ((S)S)S \to ((S)S) \to ((S)) \to (())$



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.

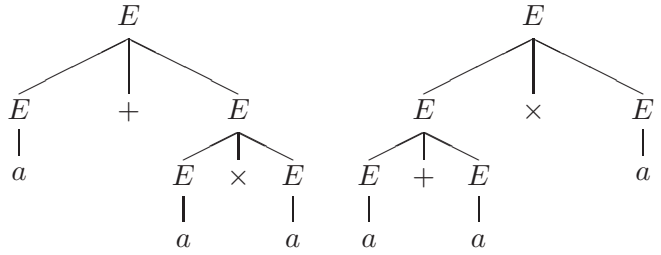
Bien noter que c'est une propriété des grammaires algébriques

2.0.4 Ambiguïté

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

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

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 ambigüe. Ces langages sont dits intrinsèquement ambigus.