Chapitre 9 : Grammaires non contextuelles

February 4, 2025

1 Grammaire non contextuelle (ou hors contexte)

1.1 Vocabulaire

Définition 9.1 - *grammaire au sens général, grammaire de type 0*

Une *grammaire* est défini par un quadruplet (Σ, V, P, S) où :

- Σ est un alphabet fini de *symboles terminaux*, dit aussi *alphabet terminal*
- V est un alphabet fini de *symboles non terminaux* (ou *variables*), dit aussi *alphabet non terminal*
- $P \subset (\Sigma \cup V)^* \times (\Sigma \cup V)^*$ est un ensemble de *règles de production*.
  Une règle de production $(w_1, w_2) \in P$, notée $w_1 \to w_2$ est un couple de mots écrits avec des symboles terminaux et non terminaux.
- $S \in V$ est un symbole non terminal avec un statut particulier de *symbole initial* (ou *axiome*, *variable initiale*)

Une grammaire sans propriété particulière est dite *type 0*.

Remarque 9.2 - *grammaires*

On note usuellement par des majuscules les symboles non terminaux, et en minuscule les terminaux.

Exemple 9.3 - *de grammaire de type 0*

Pour $\Sigma = \{\mathtt{a}\}$, $S = \mathtt{S}$, $V = \{S, \mathtt{D}, \mathtt{F}, \mathtt{X}, \mathtt{Y}, \mathtt{Z}\}$ et $P = \{\mathtt{S} \to \mathtt{DXaD}$, $\mathtt{Xa} \to \mathtt{aaX}$, $\mathtt{XF} \to \mathtt{YF}$, $\mathtt{aY} \to \mathtt{Ya}$, $\mathtt{DY} \to \mathtt{DX}$, $\mathtt{XZ} \to \mathtt{Z}$, $\mathtt{aZ} \to \mathtt{Za}$, $\mathtt{DZ} \to \epsilon\}$, $G = (\Sigma, V, P, S)$ est une grammaire de type 0.

Définition 9.4 - *dérivabilité immédiate*

Soit $G = (\Sigma, V, P, S)$ une grammaire. Soit α et β deux mots de $(\Sigma \cup V)^*$. On dit que *α se dérive immédiatement en β* lorsqu'il existe $(\alpha_1, \beta_1) \in P$ tel que :

$$\exists (u, v) \in \Big((\Sigma \cup V)^*\Big)^2, \begin{cases} \alpha = u\alpha_1 v \\ \beta = u\beta_1 v \end{cases}$$

Le cas échéant, on note $\alpha \Rightarrow \beta$. On parle de *dérivation immédiate*. Moralement, la règle de production (α_1, β_1) remplace le facteur α_1 par le facteur β_1.

Définition 9.5 - *clôture reflexive et transitive*

On note $\Rightarrow^*$ la *clôture reflexive et transitive* de la relation $\Rightarrow$ de dérivabilité immédiate.

$\Rightarrow^*$ est définie comme la plus petite relation au sens de l'inclusion tel que :

- $\forall \alpha \in (\Sigma \cup V)^*$, $\alpha \Rightarrow^* \alpha$
- $\forall (\alpha, \beta) \in \Big((\Sigma \cup V)^*\Big)^2, (\alpha \Rightarrow \beta) \implies (\alpha \Rightarrow^* \beta)$
- $\forall (\alpha, \beta, \gamma) \in \Big((\Sigma \cup V)^*\Big)^3$, $(\alpha \Rightarrow^* \beta \text{ et } \beta \Rightarrow^* \gamma) \implies (\alpha \Rightarrow^* \gamma)$

Autrement dit, $\alpha \Rightarrow^* \beta$ lorsqu'il existe $(\alpha = \alpha_0, \ldots, \alpha_k = \beta)$ une suite de mots dans $(\Sigma \cup V)^*$ telle que :

$$\forall i \in [\![0, k-1]\!], \alpha_i \Rightarrow \alpha_{i+1}$$

Définition 9.6 - *clôture reflexive et transitive de $\Rightarrow$*

Soit $G = (\Sigma, V, P, S)$ une grammaire. Soit α et β deux mots de $(\Sigma \cup V)^*$. On note *$\Rightarrow^*$ la clôture reflexive et transitive de la relation $\Rightarrow$*. Ainsi $\alpha \Rightarrow^* \beta$ lorsqu'il existe $(\alpha = \alpha_0, \ldots, \alpha_k = \beta)$ une suite de mots dans $(\Sigma \cup V)^*$ telle que :

$$\forall i \in [\![0, k-1]\!], \alpha_i \Rightarrow \alpha_{i+1}$$

Exemple 9.7 - *de dérivation*

Dans la grammaire précédemment introduite :
Pour $\Sigma = \{a\}$ et $V = \{S, D, F, X, Y, Z\}$ et $P = \{\mathtt{S} \rightarrow \mathtt{DXaD}, \mathtt{Xa} \rightarrow \mathtt{aaX}, \mathtt{XF} \rightarrow \mathtt{YF}, \mathtt{aY} \rightarrow \mathtt{Ya}, \mathtt{DY} \rightarrow \mathtt{DX}, \mathtt{XZ} \rightarrow \mathtt{Z}, \mathtt{aZ} \rightarrow \mathtt{Za}, \mathtt{DZ} \rightarrow \epsilon\}$, $G = (\Sigma, V, P, S)$ est une grammaire de type 0.

$$\begin{aligned} S &\Rightarrow \mathtt{DXaF} \\ &\Rightarrow \mathtt{DaaXF} \\ &\Rightarrow \mathtt{DaaYF} \\ &\Rightarrow \mathtt{DaYaF} \\ &\Rightarrow \mathtt{DYaaF} \\ &\Rightarrow \mathtt{DXaaF} \\ &\Rightarrow \mathtt{DaaXaF} \\ &\Rightarrow \mathtt{DaaaaXF} \\ &\Rightarrow \mathtt{DaaaaZ} \\ &\Rightarrow \mathtt{DaaaZa} \\ &\Rightarrow \mathtt{DaaZaa} \\ &\Rightarrow \mathtt{DaZaaa} \\ &\Rightarrow \mathtt{DZaaaa} \\ &\Rightarrow \mathtt{aaaa} \end{aligned}$$

D'où $S \Rightarrow^* \mathtt{aaaa}$

Définition 9.8 - *langage engendré, langage élargi engendré par une grammaire depuis un mot*

Soit $G = (\Sigma, V, P, S)$ une grammaire et $\alpha \in (\Sigma \cup V)^*$.

- le *langage engendré par G depuis α* est l'ensemble des mots de Σ^* que l'on peut obtenir par dérivation de α en utilisant les règles de production de G :

$$\mathcal{L}_G(\alpha) = \{\beta \in \Sigma^*, \, \alpha \Rightarrow^* \beta\}$$

- le *langage élargi engendré par G depuis α* est l'ensemble des mots de $(\Sigma \cup V)^*$ que l'on peut obtenir par dérivation de α en utilisant les règles de production de G :

$$\widehat{\mathcal{L}_G(\alpha)} = \{\beta \in (\Sigma \cup V)^*, \, \alpha \Rightarrow^* \beta\}$$

Définition 9.9 - *langage engendré par une grammaire*

Soit $G = (\Sigma, V, P, S)$ une grammaire et $\alpha \in (\Sigma \cup V)^*$. Le *langage engendré par G* désigne $\mathcal{L}_G(S)$ le langage engendré par G depuis son symbole initial S.

Exemple 9.10

Pour la grammaire de l'exemple, on pourrait montrer que $\mathcal{L}_G(S) = \{a^{2^n},\, n \in \mathbb{N}^*\}$ On montre que ce langage n'est pas régulier (absurde + lemme de l'étoile tmtc)

Définition 9.11 - *langage de type 0*

On dit qu'un *langage est de type 0* s'il peut être engendré par une grammaire de type 0.

Théorème 9.12 - *de Chomsky (HP)*

Les langages de type 0 sont exactement les langages récursivement énumérables, c'est-à-dire les langages reconnaissables par une machine de Turing.

Définition 9.13 - *grammaire contextuelle (HP)*

Une grammaire $G = (\Sigma, V, P, S)$ de type 0 est appelée *grammaire contextuelle (ou de type 1 ou monotone)* lorsque :

$$\forall (\alpha, \beta) \in P \setminus \{(S, \epsilon)\},\ |\alpha| \leq |\beta|$$

Moralement, tous les facteurs "produits" sont plus long que les facteurs remplacés.

Exemple 9.14 - *de grammaire qui n'est pas "contextuelle"*

la grammaire exemple définie par $\Sigma = \{a\}$ et $V = \{S, D, F, X, Y, Z\}$ et $P = \{\mathtt{S} \to \mathtt{DXaD}, \mathtt{Xa} \to \mathtt{aaX}, \mathtt{XF} \to \mathtt{YF}, \mathtt{aY} \to \mathtt{Ya}, \mathtt{DY} \to \mathtt{DX}, \mathtt{XZ} \to \mathtt{Z}, \mathtt{aZ} \to \mathtt{Za}, \mathtt{DZ} \to \epsilon\}$ puis $G = (\Sigma, V, P, S)$ n'est pas contextuelle :

$$(\mathtt{DZ} \to \epsilon) \in P$$

mais :

$$|DZ| = 2 > |\epsilon| = 0$$

Remarque 9.15 - *indépendance des caractères "contextuel" et "non contextuel"*

Nous le reverrons, mais une grammaire est "contextuelle" indépendamment de son caractère "non contextuel".

Remarque 9.16

Sans l'autorisation d'avoir $(S, \epsilon) \in P$, les langages engendrés par des grammaires contextuelles ne peuvent pas contenir ϵ.
En effet, on a initialement S de longueur 1 et les règles de productions ne peuvent qu'augmenter la longueur du mot. 0 serait donc une longueur inexistente.

Définition 9.17 - *grammaire non contextuelle*

Une grammaire $G = (\Sigma, V, P, S)$ est dite *non contextuelle* (ou *hors contexte* ou encore *algébrique* ou *de type 2*) lorsque :

$$P \subset V \times (\Sigma \cup V)^*$$

Moralement, les règles de production ne permettent de remplacer que des symboles non terminaux (et pas des mots) : des majuscules.

Exemple 9.18 - *de règle de production non valide pour une grammaire "non contextuelle"*

`DZ` $\rightarrow \epsilon$ n'est pas une règle possible pour une grammaire hors contexte.

Exemple 9.19 - *de grammaire "non contextuelle"*

$G_1 = (\Sigma_1, V_1, P_1, S_1)$ définie par :

- $\Sigma_1 = \{a, b\}$
- $V_1 = \{S_1\}$
- $P_1 = \{S_1 \rightarrow aS_1b,\ S_1 \rightarrow \epsilon\}$

est une grammaire hors contexte.

Remarque 9.20

On dit "hors contexte" car les symboles non terminaux sont considérés seuls par les règles de production : on ne regarde pas les lettres autour.

Exemple 9.21 - *langage engendré par une grammaire hors contexte*

$G_1 = (\Sigma_1, V_1, P_1, S_1)$ définie par :

- $\Sigma_1 = \{a, b\}$
- $V_1 = \{S_1\}$
- $P_1 = \{S_1 \to aS_1b,\, S_1 \to \epsilon\}$

Alors $\mathcal{L}_{G_1}(S_1) = \{a^nb^n,\, n \in \mathbb{N}\}$. On le démontre (Démo 1). Par ailleurs $\widehat{\mathcal{L}_{G_1}(S_1)} = \{a^nS_1b^n,\, n \in \mathbb{N}\} \cup \{a^nb^n,\, n \in \mathbb{N}\}$.

Exemple 9.22 - *Langage de Dyck*

Le langage de Dyck (bon parenthésage) est engendré par :

- $\Sigma = \{\texttt{(},\texttt{)}\}$
- $S \to (S)S$
- $S \to \epsilon$

Remarque 9.23 - *grammaires "non contextuelles" uniquement définies par les règles de production*

En général, on donne une grammaire hors contexte uniquement avec les règles de production.
On ne précise le symbole initial que si ce n'est pas S.
Les symboles non terminaux sont exactement ceux que l'on rencontre à gauche des règles et les terminaux sont les autres qu'on rencontre.

Remarque 9.24 - *abréviation d'une famille de règles de production*

Pour noter rapidement un ensemble de règles de production utilisant le même symbole non terminal comme départ :

$$\{\texttt{X,}\alpha_1 \to \texttt{,} \dots, \texttt{X} \to \alpha_k\} \subset P$$

avec $X \in V$, on note :

$$X \to \alpha_1 \,|\, \cdots \,|\, \alpha_k$$

Ce n'est qu'une notation, on a bien k règles de production derrière ça.

Exemple 9.25 - *exploitant ce raccourci*

G_1 s'écrit :

$$S_1 \to aS_1b \,|\, \epsilon$$

1.2 Langages réguliers et langages hors contexte

Un résultat à connaître et savoir redémontrer.

Proposition 9.26 - *régulier $\implies$ hors contexte*

Soit Σ un alphabet, Tout langage sur Σ régulier est hors contexte.

Voir Démo 2 (elle est incomplète)

Remarque 9.27 - *hors contexte $\not\Rightarrow$ régulier*

il existe des langages hors contexte qui ne sont pas réguliers. En effet on a montré que pour $G_1 = (\Sigma_1, V_1, P_1, S_1)$ définie par :

- $\Sigma_1 = \{a, b\}$
- $V_1 = \{S_1\}$
- $P_1 = \{S_1 \to aS_1b,\ S_1 \to \epsilon\}$

Alors $\mathcal{L}_{G_1}(S_1) = \{a^n b^n,\ n \in \mathbb{N}\}$.

Ce langage est donc hors contexte, mais n'est pas régulier.

Définition 9.28 - *grammaire hors contexte linéaire (HP)*

Une grammaire $G = (\Sigma, V, P, S)$ hors contexte est dite *linéaire* si :

$$P \subset V \times \left(\Sigma^* \cup \Sigma^* V \Sigma^*\right)$$

Moralement, on a toujours au plus un symbole non terminal (voir Fig.1)

Définition 9.29 - *grammaire hors contexte linéaire gauche (HP)*

Une grammaire $G = (\Sigma, V, P, S)$ hors contexte est dite *linéaire* si :

$$P \subset V \times \left(\Sigma^* \cup V \Sigma^*\right)$$

Moralement, Le facteur "produit" commence par un unique symbole non terminal ou n'en a aucun.

Définition 9.30 - *grammaire hors contexte linéaire droite (HP)*

Une grammaire $G = (\Sigma, V, P, S)$ hors contexte est dite *linéaire* si :

$$P \subset V \times \left(\Sigma^* \cup \Sigma^* V\right)$$

Moralement, Le facteur "produit" termine par un unique symbole non terminal ou n'en a aucun.

Remarque 9.31 - *chaine d'implications*

linéaire droit implqiue linéaire implique hors contexte linéaire gauche implique linéaire implique hors contexte.

Définition 9.32 - *langages linéaire, linéaire droit, linéaire gauche*

Un langage est dit linéaire (resp. simple, droit, gauche) lorsqu'il peut être engendré par une grammaire linéaire (resp. simple, linéaire droite).

Remarque 9.33 - *existence de langages linéaires non réguliers*

$$\mathtt{S} \rightarrow \mathtt{aSb} \,|\, \epsilon$$

est linéaire et engendre $\{a^n b^n,\, n \in \mathbb{N}\}$

Proposition 9.34 - *pour un langage, régulier $\iff$ linéaire droit $\iff$ linéaire gauche*

Soit L un langage. Les propriétés suivantes sont équivalentes.

1. L est régulier
2. L est linéaire droit
3. L est linéaire gauche

Remarque 9.35 - *sur la démo*

En montrant régulier implqiue linéaire droit, on montre à nouveau que régulier implqiue hors contexte, ceci constitue une autre démo que la propriété précédente.

Démo 3.

1.3 Système d'équations d'une grammaire

Définition 9.36

Soit $G = (\Sigma, V, P, S)$ une grammaire hors contexte avec $V = \{S, X_1, \dots, X_n\}$. Soit $L = (L_0, \dots, L_n)$ un $(n+1)$-uplet de langages sur Σ. On définit par induction :

- $\epsilon(L) = \{\epsilon\}$ (base)
- $\forall a \in \Sigma$, $a(L) = \{a\}$ (base)
- $S(L) = L_0$ (base)
- $\forall i \in [\![1, n]\!]$, $X_i(L) = L_i$ (base)
- $\forall (\alpha, \beta) \in \left((\Sigma \cup V)^*\right)^2$, $\alpha\beta(L) = \alpha(L) \cdot \beta(L)$ (compatibilité avec la concaténation)
- $\forall K \subset (\Sigma \cup V)^*$, $K(L) = \bigcup_{w \in K} w(L)$ (compatibilité avec l'union)

Le *système d'équations $\mathcal{S}(G)$ de la grammaire G* est le suivant :

$$\begin{cases} \forall i \in [\![1, n]\!],\ L_i = \bigcup_{(X_i, \alpha) \in P} \alpha(L) \\ L_0 = \bigcup_{(S, \alpha) \in P} \alpha(L) \end{cases}$$

Remarque 9.37

À chaque symbole on associe un langage L_i. On veut montrer que $\mathcal{L}_G = (\mathcal{L}_G(S), \mathcal{L}_G(X_1), \dots, \mathcal{L}_G(X_n))$ est solution du système $\mathcal{S}(G)$

Exemple 9.38

$$\begin{aligned} G_1 :& \\ & S \to aX_1b \\ & X_1 \to cX_2d \\ & X_2 \to \epsilon \mid S \end{aligned}$$

Avec cette grammaire, on obtient le système suivant :

$$\begin{cases} L_0 = \{a\}L_1\{b\} \\ L_1 = \{c\}L_2\{d\} \\ L_2 = \{\epsilon\} \cup L_0 \end{cases}$$

Remarque 9.39 - *notation liées à un sytème d'équations d'une grammaire*

On note $\mathcal{L}_G = (\mathcal{L}_G(S), \mathcal{L}_G(X_1), \dots, \mathcal{L}_G(X_n))$ le $(n+1)$-uplet de langages engendrés par les symboles non terminaux.

Proposition 9.40 - *lemme 1 concernant ce $(n+1)$-uplet*

$\forall \alpha \in (V \cup \Sigma)^*$, $\mathcal{L}_G(\alpha) = \alpha(\mathcal{L}_G)$

Démo 4

Proposition 9.41 - *Lemme 2 concernant ce ...*

Soit L un $(n+1)$-uplet de langages sur Σ solution de $\mathcal{S}(G)$.

$$\forall (\alpha, \beta) \in (\Sigma \cup V)^*, \, (\alpha \Rightarrow^* \beta) \implies (\beta(L) \subset \alpha(L))$$

demo 5

Théorème 9.42 - *utilisant les deux lemmes*

$\mathcal{L}_G = (\mathcal{L}_G(S), \mathcal{L}_G(X_1), \dots, \mathcal{L}_G(X_n))$ est l'unique solution minimale pour l'inclusion composante par composannte de $\mathcal{S}(G)$.

Démo 6

Exemple 9.43 - *pratique*

En pratique on peut résoudre le système en utilisant le lemme d'Arden pour déterminer le langage engendré par une grammaire hors contexte pas trop complexe. D'après la théorie précédente, la solution sera l'unique solution minimale pour le système de grammaire.

$$\begin{aligned} G : & S \to SX_1 \mid \epsilon \\ & X_1 \to aX_1 \mid b \end{aligned}$$

$$\begin{cases} L_0 = L_0 L_1 \cup \{\epsilon\} \\ L_1 = \{a\} L_1 \{b\} \end{cases}$$

Voir Démo 7

Exemple 9.44 - *pénible*

Pour $S \to aSb \,|\, \epsilon$, On ne peut pas appliquer le lemme d'Arden, il faut faire une induction :

$$L = (A_1 L A_2) \cup B$$

Une solution minimale est $\{m_1 m_0 m_2,\, k \in \mathbb{N},\, m_1 \in A_1^k,\, m_2 \in A_2^k,\, m_0 \in B\}$. La démo est celle du lemme d'Arden mais il ne couvre pas ce cas c'est tout.

Remarque 9.45

Les systèmes d'équation de grammaire et le lemme d'Arden sont des outils HORS PROGRAMME. À l'écrit, il faut le redémontrer une fois avant d'utiliser le lemme d'Arden ou faire une induction structurelle dans les cas particuliers.
Pour $\mathcal{S}(G)$, on justifie par une phrase en s'appuyant sur la première règle utilisée et on donne $\mathcal{S}(G)$.

1.4 Liens avec les définitions inductives

Les définitions des grammaires hors contexte sont très liées aux définitions inductives.

1.5 Hiérarchie de Chomsky - HP, culture générale

Théorème 9.46 - *hiérarchie de Chomsky*

Il s'agit d'un théorème qui définit une "hiérarchie" sur les langages :

1. Tout langage régulier (rationnel, de type 3) est un langage linéaire (réciproque fausse)
2. Tout langage linéaire est un langage hors contexte (de type 2). La réciproque est fausse.
3. Tout langage hors contexte (de type 2) est un langage contextuel (de type 1). La réciproque est fausse
4. Tout langage contextuel (de type 1) est un langage récursif (réciproque fausse). C'est à dire les langages récurssifs (*i.e.* reconnus par une machine de Turing déterministe, sans calcul infini).
5. Tout langage récursif est récursivement énumérable (= de type 0 d'après le théorème de Chomsky) (reconnaissable par une machine de Turing).

Voir Fig.8

2 Arbre d'analyse

Définition 9.47 - *arbre d'analyse*

Un *arbre d'analyse* (ou *arbre d'analyse syntaxique*, ou *arbre de dérivation*) pour une grammaire hors contexte $G = (\Sigma, V, P, S)$ et un mot $w \in \mathcal{L}_G(S)$ est un arbre :

- dont la racine est étiquetée par S
- dont tout noeud interne d'étiquette $X \in V$, de fils d'étiquettes $x_1, \ldots, x_k$ (numérotation de gauche à droite), est tel que $(X, x_1 \ldots x_k) \in P$
- dont les feuilles sont dans $\Sigma \cup \{\epsilon\}$

On appelle alors *frontière* de l'arbre d'analyse le mot obtenu en concaténant de gauche à droite les symboles des feuilles de l'arbre d'analyse : c'est w.

Exemple 9.48 - *d'arbre d'analyse*

Pour $G : S \to aSb \,|\, \epsilon$, Voir Fig.9

Remarque 9.49 - *propriété de l'arbre d'analyse*

L'arbre de d'analyse est la preuve qu'un mot est engendré par une grammaire. Il rend compte de toutes les règles de production utilisées mais pas entièrement de l'ordre. Voir Fig 10.

Exemple 9.50 - *d'arbre d'analyse*

On se donne une grammmaire hors contexte qui engendre un langage de bon parenthésage.

$$G : S \to \epsilon \,|\, SS \,|\, (S) \,|\, [S]$$

$w = (([])())[]() \in \mathcal{L}_G(S)$ On peut construire plusieurs arbres d'analyse pour ce mot, qui sont représentés en Fig.11

Remarque 9.51 - *applications de l'arbre d'analyse*

À partir d'un arbre d'analyse on peut reconstruire, on peut reconstruire une ou plusieurs dérivations correspondant exactement à l'arbre (mêmes règles utilisées) en suivant un parcours en profondeur (préfixe) :
On applique les règles sur les symboles non terminaux dans l'ordre dans lequel on les rencontre dans le parcours en commençant arbitrairement par la gauche (préfixe gauche) ou la droite (préfixe droite).

Définition 9.52 - *dérivation à gauche*

Soit $G = (\Sigma, V, P, S)$ une grammaire hors contexte. On appelle *dérivation à gauche* de S en $w \in \Sigma^*$ une suites de dérivations immédiates $(u_i X_i v_i \Rightarrow u_i \beta_i v_i)_{i \in [\![0,\, n]\!]}$ avec pour $i \in [\![1,\, n]\!]$:

- $u_i \in \Sigma^*$
- $(X_i, \beta_i) \in P$
- $v_i \in (\Sigma \cup V)^*$

et $u_0 X_0 v_0 = S$, $u_n X_n v_n = w$.
Moralement, pour arriver à w on ne remplace que le premier symbole non terminal le plus à gauche.

Définition 9.53 - *dérivation à droite*

Soit $G = (\Sigma, V, P, S)$ une grammaire hors contexte. On appelle *dérivation à droite* de S en $w \in \Sigma^*$ une suites de dérivations immédiates $(u_i X_i v_i \Rightarrow u_i \beta_i v_i)_{i \in [\![0,\, n]\!]}$ avec pour $i \in [\![1,\, n]\!]$:

- $u_i \in (\Sigma \cup V)^*$
- $(X_i, \beta_i) \in P$
- $v_i \in \Sigma^*$

et $u_0 X_0 v_0 = S$, $u_n X_n v_n = w$.
Moralement, pour arriver à w on ne remplace que le premier symbole non terminal le plus à droite.

pareil pour la droite.

Proposition 9.54 - *lien avec l'arbre d'analyse*

Le parcours en profondeur à gauche (préfixe gauche) d'un arbre d'analyse permet de produire une dérivation à gauche.

résultat analogue pour la droite.

Exemple 9.55 - *d'exploitation de cette proposition*

$G : S \to AB$, $S \to aA \mid \epsilon$, $B \to bB \mid \epsilon$.
Soit $aab \in \mathcal{L}_G(S)$. Voir Fig 12.

Définition 9.56 - *grammaire ambiguë*

Soit $G = (\Sigma, V, P, S)$ une grammaire hors contexte. G est *ambiguë* s'il existe un mot de $\mathcal{L}_G(S)$ admettant au moins deux arbre d'analyse (de même frontière) distincts.

Définition 9.57 - *inhérente ambiguité d'un langage*

Un langage hors contexte est dit *inhéremment ambigu* lorsque toutes les grammaires l'engendrant sont ambiguës. Sinon il est dit *non ambigu*.

Remarque 9.58

Il existe des langages hors contextes inhéremment ambigu :

$$L = \{a^i b^j c^k,\ i = j \text{ ou } j = k,\ (i, j, k) \in \mathbb{N}^3\}$$

Voir Fig.13.

Remarque 9.59 - *grammaires ambiguë*

Les grammaires ambiguë posent problèment pour les compilateurs ! Ils ne savent pas quel arbre choisir.

Remarque 9.60 - *condition suffisante d'ambiguité*

Avec une grammaire hors contexte avec une règle du type $X \to XX$ utilisée par au moins une dérivation d'un mot engendré, on peut toujours montrer l'ambiguité. en particulier $S \to SS$ rend le langage ambiguë : on l'a vu. Voir Fig.14.

Passe 3
Comment lever une ambiguité ? Aucune méthode précise n'est attendue. On doit savoir "bidouiller" pour des cas simples.

2.0.1 Première méthode

Ajouter de nouveaux symboles Pour imposer des règles de priorité ou forcer un ordre... C'est la méthode au programme (On doit se débrouiller quoi).

2.0.2 Deuxième méthode (HP)

Ajout de *méta-règles*, extérieures à la grammaire qui définissent des priorités sur les règles à appliquer.

Définition 9.61 - *faible équivalence de deux grammaires hors contexte*

On dit de deux grammaires G et G' hors contexte qu'elles sont *faiblement équivalentes* si elles engendrent un même langage :

$$\mathcal{L}_G(S) = \mathcal{L}_{G'}(S')$$

Remarque 9.62 - *faible équivalence*

Pour certaines grammaires ambiguë, il existe une grammaire non ambiguë faiblement équivalente. (pas toujours).

Définition 9.63 - *forte équivalence de deux grammaires (HP)*

Deux grammaires sont *fortement équivalentes* lorsqu'elles engendrent le même langage et utilisent les mêmes dérivations pour un même mot à l'ordre près.

2.1 Exemple des expression arithmétiques

On peut engendrer les langages des expressions arithmétiques avec une grammaire hors contexte naïve :

$$S \to S + S \,|\, S - S \,|\, S/S \,|\, S \times S \,|\, N$$

$$N \to 0 \,|\, \dots \,|\, 9 \,|\, NN$$

Le mot $1 + 2 + 3 \times 4$ est dans $\mathcal{L}_G(S)$ et admet deux arbres d'analyse distincts. Voir Fig.15.

Première solution : On rajoute des parenthèses :

$$S \to (S + S) \,|\, (S - S) \,|\, (S/S) \,|\, (S \times S) \,|\, N$$

$$C \to 0 \,|\, \dots \,|\, 9$$

$$N \to CN \,|\, C$$

Ce n'est plus ambiguë, mais le langage engendré n'est plus le même :/

Deuxième solution : (méthode à connaître)

$$S \to S + D \,|\, D \qquad \text{(somme)}$$

$$D \to D - P \,|\, P \qquad \text{(différence)}$$

$$P \to P \times Q \,|\, Q \qquad \text{(produit)}$$

$$Q \to Q/N \,|\, N \,|\, (S) \qquad \text{(quotient)}$$

$$N \to CN \,|\, C \qquad \text{(nombre)}$$

$$C \to 0 \,|\, \ldots \,|\, 9 \qquad \text{(chiffre)}$$

On a rajouté des parenthèses, mais on peut les enlever, on limitera alors les opérations imbriquées. Par contre, on a imposé un ordre de priorité qui rend le langage engendré non ambigu.

2.2 Problème du "sinon" pendant ("dangling else")

On considère la grammaire suivante correspondant à une partie du langage C pour les blocs conditionnels :

$$\texttt{I} \to \texttt{if(E)I}$$

$$\texttt{I} \to \texttt{if(E)I else I}$$

$$\texttt{I} \to \texttt{E;I} \,|\, \epsilon$$

$$\texttt{E} \to \texttt{x=2} \,|\, \texttt{E<E} \,|\, \texttt{E>E} \,|\, \texttt{E==E} \,|\, \ldots$$

On considère le morceau de code :

```
if (x>4) if (x<5) x=10; else x=42;
```

Deux interprétations possibles : Voir Fig.16. Deux solutions :

- Savoir coder dignement (avec des accolades)
- En C, on a une *méta-règle* utilisée par le compilateur : le `else` est associé au `if` le plus proche (antérieurement) non déjà associé à un autre bloc `else`.s

3 Automates à pile (HP)

Les automates à pile constituent une extension des automates finis : on rajoute une mémoire, sous la forme d'une pile, à un automate fini.
Les transitions dépilent un symbole et en empilent d'autres.
Ainsi les langages reconnus par les automates à pile sont exactement les langages hors contexte.

Définition 9.64 - *automate à pile*

On appelle *automate à pile* un sixtuplet $A = (\Sigma, Z, z_0, Q, q_0, \delta)$ tel que :

- Σ est un alphabet fini dit *alphabet d'entrée*. C'est l'alphabet des mots que l'automate reconnaît.
- Z est un alphabet fini dit *alphabet de pile*. Il contient symboles que l'on peut utiliser dans la pile
- $z_0 \in Z$ est le *symbole initial*. La pile contient initialement z_0.
- Q est un *ensemble fini d'états*.
- $q_0 \in Q$ est l'*état initial*.
- δ est la *fonction de transition* :

$$\delta : \Big(\Sigma \cup \{\epsilon\}\Big) \times Q \times Z \to \mathcal{P}(Q \times Z^*)$$

Définition 9.65 - *transition dans un automate à pile*

Soit $A = (\Sigma, Z, z_0, Q, q_0, \delta)$ un automate à pile. On note une *transition de A* de la façon suivante. Voir Fig.17

Définition 9.66 - *configuration dans un calcul sur un automate à pile*

Soit $A = (\Sigma, Z, z_0, Q, q_0, \delta)$ un automate à pile.
Une *configuration dans un calcul sur A* est un couple $(q, w) \in Q \times Z^*$ tel que :

- q est l'état courant
- w est le mot représentant l'état de la pile

Moralement, il s'agit de l'information de l'état actuel de l'automate.

Voir Fig.18. p19

Définition 9.67 - *étape de calcul d'un automate à pile*

Soit $A = (\Sigma, Z, z_0, Q, q_0, \delta)$ un automate à pile. On appelle *étape de calcul* dans A le passage d'une configuration $(q, uz) \in Q \times Z^*$ à une configuration $(q', vz) \in Q \times Z^*$ associé à $y \in \Sigma \cup \{\epsilon\}$, avec :

- $u \in Z$ est le symbole dépilé au cours de l'étape.
- $v \in Z^*$ est le mot empilé en échange
- $z \in Z^*$ est le contenu inchangé de la pile

On note $(q, uz) \xrightarrow{y} (q', vz)$ une telle étape de calcul : y est son *étiquette*.

Définition 9.68 - *calcul*

Un *calcul* dans l'automate à pile $A = (\Sigma, Z, z_0, Q, q_0, \delta)$ est une suite d'étapes de calculs :

$$C_0 \xrightarrow{y_1} C_1 \xrightarrow{y_2} \dots \xrightarrow{y_n} C_n$$

On appelle $y = y_1 \dots y_n$ *l'étiquette du calcul*.

Définition 9.69 - *calcul acceptant*

Un calcul $C_0 \xrightarrow{y_1} C_1 \xrightarrow{y_2} \dots \xrightarrow{y_n} C_n$ est *accpetant* lorsuqe :

- $C_0 = (q_0, z_0)$ est la configuration initiale de l'automate à pile : celle par laquelle il commence ses calculs.
- C_n est une configuration finale.

Il existe différents modes d'acceptation, correspondant à des ensembles de configurations finales distinctes :

- par *pile vide* : (q, ϵ) pour $q \in Q$ est une configuration finale
- par *états finaux* : $(q_f, w) \in F \times Z^*$ est une configuration finale.
- par *sommet de la pile* : on ajoute $Z_f \subset Z$ un ensemble de symboles finaux :

  $$(q, z_f w) \in Q \times (Z_f Z^*)$$

  est alors une configuration finale.

On peut utiliser aussi une combinaison de ces trois modes.

Exemple 9.70 - *d'automate à pile*

Voir Fig.19 (p19)

Proposition 9.71 - *peu importe le mode d'acceptation*

Les différents modes d'acceptation permettent de reconnaître le même ensemble de langages.