

# 1 Grammaires non contextuelles

**Exercice 1.** Considérons la grammaire suivante sur l'alphabet  $\{x, y, +, -, *\}$  :

$$E \rightarrow +EE \mid *EE \mid -EE \mid x \mid y$$

1. Trouver les dérivations gauches (*leftmost*) et droites (*rightmost*) ainsi qu'un arbre de dérivation pour la chaîne  $+*-xyxy$ .
2. Prouver que cette grammaire est non-ambiguë.
3. Trouver un automate à pile pour cette grammaire.

## Solution 1.

Voici les dérivations gauche et droite avec l'arbre induit noté par des parenthèses :

$$\begin{aligned} E &\rightarrow +EE && \rightarrow +(*EE)E && \rightarrow +*(-EE)E \\ &\rightarrow +*(-xE)E && \rightarrow +*(-xy)E && \rightarrow +*(-xy)x \\ &\rightarrow +*(-xy)x && && \rightarrow +*(-xy)x \end{aligned}$$

$$\begin{aligned} E &\rightarrow +EE && \rightarrow +Ey && \rightarrow +(*EE)y \\ &\rightarrow +(*Ex)y && \rightarrow +*(-EE)x && \rightarrow +*(-Ey)x \\ &\rightarrow +*(-xy)x && && \rightarrow +*(-xy)x \end{aligned}$$

Si la grammaire est ambiguë alors il existe deux dérivations gauches. Mais toutes les règles transformant  $E$  insèrent un symbole terminal à gauche et toutes les règles ont des symboles différents donc les dérivations gauches sont forcées par la chaîne à atteindre (et donc la  $i$ -ème dérivation est  $E \rightarrow \lambda EE$  si  $\lambda \in \{+, -, *\}$  est le  $i$ -ème caractère ou  $E \rightarrow z$  si  $z \in \{x, y\}$  est le  $i$ -ème caractère).

Notre automate est  $(\{E, OK\}, \{+, -, *, x, y\}, \{\epsilon, E\}, \delta, E, \epsilon, \{OK\})$  avec pour  $\lambda \in \{+, -, *\}$  et  $z \in \{x, y\}$  on a  $\delta(E, \lambda, S) = (E, SE)$ ,  $\delta(E, z, E) = (E, \epsilon)$ ,  $\delta(E, z, \epsilon) = (OK, \epsilon)$ .

**Exercice 2.** En utilisant le lemme d'Ogden ou le lemme de pompage, montrer que les langages suivants ne sont pas algébriques :

1.  $\mathcal{L}_0 = \{a^i b^j c^k \mid i < j < k\}$
2.  $\mathcal{L}_1 = \{a^n b^n c^m \mid n \leq m \leq 2n\}$
3.  $\mathcal{L}_2 = \{a^{2^n} \mid n \in \mathbb{N}\}$
4.  $\mathcal{L}_3 = \{a^{n^2} \mid n \in \mathbb{N}\}$

**Solution 2.** Soit  $n$  de pompage, on a :

1.  $u = a^n b^{n+1} c^{n+2} \in \mathcal{L}_0$  donc on peut décomposer en  $u = xvywz$  tel que  $xw^i y w^i z$  est dans le langage pour tout  $i$  avec  $|vw| \geq 1$ .  
Comme c'est vrai pour  $i = 0$  on a  $vw$  qui contient plus de  $a$  que de  $b$  que de  $c$  mais comme c'est vrai pour  $i = 2$  on a que  $vw$  contient plus de  $c$  que de  $b$  que de  $a$  donc  $vw$  en contient autant de chaque et au moins un mais  $v$  et  $w$  ne peuvent chacun contenir qu'une lettre (sinon le motif  $a^* b^* c^*$  est brisé).
2.  $a^n b^n c^{2n} \in \mathcal{L}_1$  peut être décomposé en  $u = xvywz$  avec  $xv^i y w^i z$  dans le langage avec  $|vw| \geq 1$  mais alors les  $v$  et  $w$  ne peuvent contenir qu'une lettre (sinon le motif  $a^* b^* c^*$ ). selon  $v$  et  $w$  on voit que le ratio entre  $a$ ,  $b$  et  $c$  va être brisé pour  $i = 2n$ .
3.  $a^{2^n} \in \mathcal{L}_2$  donc on a comme pour le cas rationnel  $u = xvywz$  et  $xv^i y w^i z$  dans le langage donc  $a^{2^n + i \times k}$  dans le langage pour tout  $i$  et  $k > 0$  ce qui est contradictoire car les puissances de deux ne sont pas séparés de  $k$ .
4.  $a^{n^2} \in \mathcal{L}_2$  donc on a comme pour le cas rationnel  $u = xvywz$  et  $xv^i y w^i z$  dans le langage donc  $a^{n^2 + i \times k}$  dans le langage pour tout  $i$  et  $k > 0$  ce qui est contradictoire car les carrés sont séparés d'un nombre qui tend vers l'infini.

## 2 Forme Normale de Chomsky (CNF)

**Definition 1.** On rappelle qu'une grammaire est sous forme normale de Chomsky quand toutes ses productions sont de la forme :

$$A \rightarrow BC \quad \text{ou} \quad A \rightarrow a \quad \text{ou} \quad S \rightarrow \epsilon$$

avec  $B \neq S$  et  $C \neq S$  où  $S$  est le symbole initial.

**Exercice 3.** Soit  $G = (\Sigma, N, P, S)$  un grammaire algébrique. On suppose que  $S$  n'apparaît jamais à droite d'une règle de production, comment éliminer toutes les transitions  $A \rightarrow \epsilon$  (sauf éventuellement  $S \rightarrow \epsilon$ ) d'une grammaire ?

**Solution 3.** On calcule jusqu'à atteindre un point fixe l'ensemble  $\mathcal{E}_n = \mathcal{E}_n \cup \{A \mid A \rightarrow \epsilon \in P_n\}$  où  $\mathcal{E}_0 = \emptyset$ ,  $P_0 = P$  où l'on a gardé que les règles sans symboles non-terminaux et  $P_{n+1} = P_n$  où on a remplacé dans les règles de production de  $P_n$  les symboles de  $\mathcal{E}_n$  par  $\epsilon$ .

On ajoute alors pour chaque règle  $A \rightarrow \alpha_1 \dots \alpha_n$  toutes les règles obtenables en supprimant des  $\alpha_i \in \mathcal{E}$  puis enfin on supprime toutes les règles de production d' $\epsilon$  sauf éventuellement  $S \rightarrow \epsilon$ .

**Exercice 4.** Proposer des règles pour transformer les productions suivantes en CNF (on suppose que  $S$  n'apparaît pas) :

1.  $A \rightarrow bC$
2.  $A \rightarrow Bc$
3.  $A \rightarrow bc$
4.  $A \rightarrow BCD$
5.  $A \rightarrow \alpha_1 \alpha_2 \alpha_3$  avec  $\alpha_i \in \Sigma \cup N$
6.  $A \rightarrow \alpha_1 \dots \alpha_p$  avec  $p \geq 3$
7.  $A \rightarrow B$

### Solution 4.

1. Pour  $A \rightarrow bC$  on introduit un nouvel symbole  $B$  et notre règle devient  $A \rightarrow BC$  et  $B \rightarrow b$ ;
2. pour  $A \rightarrow Bc$  de même, on introduit  $C$  et notre règle devient  $A \rightarrow BC$  et  $C \rightarrow c$ ;
3. pour  $A \rightarrow bc$  on introduit  $B$  et  $C$  et notre règle devient  $A \rightarrow BC$ ,  $B \rightarrow b$  et  $C \rightarrow c$ ;
4. pour  $A \rightarrow BCD$  on introduit  $E$  et notre règle devient  $A \rightarrow BE$  et  $E \rightarrow CD$ ;
5. pour  $A \rightarrow \alpha_1 \alpha_2 \alpha_3$  avec  $\alpha_i \in \Sigma \cup N$  on introduit  $B$  puis  $A \rightarrow \alpha_1 B$  et  $B \rightarrow \alpha_2 \alpha_3$  et on selon les cas on ré-applique les règles plus-haut ;
6.  $A \rightarrow \alpha_1 \dots \alpha_p$  avec  $p \geq 3$  on introduit  $B$  et on remplace la règle par  $A \rightarrow \alpha_1 B$  et  $B \rightarrow \alpha_2 \dots \alpha_p$  puis on procède par récurrence sur  $B$ .
7. Pour  $A \rightarrow B$  on ajoute des règles, pour chaque règle  $X \rightarrow AA$  on ajoute  $X \rightarrow AB, X \rightarrow BA$  et  $X \rightarrow BB$ , pour chaque règle  $X \rightarrow AZ$  on ajoute  $X \rightarrow BZ$ , pour chaque règle  $X \rightarrow AZ$  on ajoute  $X \rightarrow ZB$  puis on supprime  $A \rightarrow B$ .

**Exercice 5.** Proposer une grammaire CNF équivalente à la grammaire suivante :

$$\begin{array}{lcl} S & \rightarrow & CSC \mid aB \\ C & \rightarrow & B \mid S \\ B & \rightarrow & b \mid \epsilon \end{array}$$

**Solution 5.** D'abord on ajoute un symbole initial  $S_0 \rightarrow S$  et on élimine les règles à trois membres à droite :

$$\begin{array}{ll} S_0 \rightarrow CD \mid aB & D \rightarrow SC \\ S \rightarrow CD \mid AB & C \rightarrow B \mid S \\ B \rightarrow b \mid \epsilon & A \rightarrow a \end{array}$$

Ensuite on note que le non terminal  $C$  se transforme en  $\epsilon$ , alors  $S \rightarrow D$  et  $D \rightarrow S$  et donc on remplace  $D$  en  $S$  :

$$\begin{array}{ll} S_0 \rightarrow CS \mid AB \mid SB & S \rightarrow SC \\ S \rightarrow CS \mid AB & C \rightarrow B \mid S \\ B \rightarrow b \mid \epsilon & A \rightarrow a \end{array}$$

On élimine le non-terminal  $C$  en le remplaçant par ses deux possibles productions :

$$\begin{array}{ll} S_0 \rightarrow SS \mid BS \mid AB \mid SB & S \rightarrow SS \mid BS \mid AB \mid SB \\ B \rightarrow b \mid \epsilon & A \rightarrow a \end{array}$$

On élimine  $B \rightarrow \epsilon$  :

$$\begin{array}{ll} S_0 \rightarrow S & S \rightarrow SS \mid BS \mid AB \mid SB \mid A \\ B \rightarrow b & A \rightarrow a \end{array}$$

On élimine  $S \rightarrow A$  (en utilisant  $A \rightarrow a$ ) :

$$\begin{array}{ll} S_0 \rightarrow S & S \rightarrow SS \mid BS \mid AB \mid SB \mid a \\ B \rightarrow b & A \rightarrow a \end{array}$$

On remarque que la grammaire est celle des mots contenant au moins un  $a$ .

**Exercice 6.** Proposer un algorithme de temps polynomial (en la taille cumulée du mot et de la grammaire) qui reconnaît si un mot appartient à une grammaire CNF.

**Solution 6.** Soit  $u = u_1 \dots u_n$ ,  $A_1, \dots, A_k$  pour chaque paire  $1 \leq i \leq j \leq n$  on va calculer  $T(i, j, k)$  qui vaut vrai si  $u_i \dots u_j$  peut être transformé en le symbole  $A_k$ .

$$T(i, j, k) = \bigvee_{\substack{i \leq l < j \\ A_k \rightarrow A_x A_y \in G}} T(i, l, x) \wedge T(l+1, j, y)$$

when  $i \neq j$  and  $T(i, i, k) = (A_k \rightarrow u_i \in G)$ .

On est bien polynomial car le vecteur  $T(i, j)$  peut être calculé par une boucle sur  $l$  et une itération sur les règles de  $G$ .