

## TD 7

**Exercice 1.**

If Then Else

On considère la grammaire suivante :

$$Stmt \rightarrow \text{if } b \text{ then } Stmt \mid \text{if } b \text{ then } Stmt \text{ else } Stmt \mid a$$

1. Montrer que cette grammaire est ambiguë.
2. Proposer une grammaire non ambiguë pour le même langage.

**Exercice 2.**

Chomsky

**Définition.** Une grammaire hors-contexte  $G = (V, \Sigma, P, S)$ <sup>1</sup> est sous forme normale de Chomsky ssi toutes ses règles sont de la forme

- (i)  $A \rightarrow BC$  avec  $B, C \in V$
- (ii)  $A \rightarrow a$  avec  $a \in \Sigma$
- (iii)  $S \rightarrow \varepsilon$

De plus, si  $S \rightarrow \varepsilon$  est une règle de  $P$ , alors  $B, C \in V - \{S\}$  dans (i).

Le but de cet exercice est de prouver le théorème suivant :

**Théorème.** Pour toute grammaire hors-contexte  $G = (V, \Sigma, P, S)$ , il existe une grammaire  $G' = (V', \Sigma, P', S)$  telle que  $\mathcal{L}(G') = \mathcal{L}(G)$  et  $G'$  est sous forme normale de Chomsky.

Considérons les cinq transformations suivants :

- † **START** : On introduit  $S_0$  nouvelle variable de départ et la règle  $S_0 \rightarrow S$ .
- † **TERM** : Pour chaque lettre  $a \in \Sigma$  terminale, on ajoute une variable  $V_a$  avec la règle  $V_a \rightarrow a$ . Puis, si dans une règle avec au moins 2 membres droits, un membre droit  $a$  est un terminal, on le remplace par  $V_a$ .
- † **BIN** : On remplace toute règle  $A \rightarrow B_1 \dots B_k$  avec  $k \geq 3$  par  $A \rightarrow B_1 V_1, V_1 \rightarrow B_2 V_2, \dots, V_{k-2} \rightarrow B_{k-1} B_k$  avec  $V_1, \dots, V_{k-2}$  qui sont  $k - 2$  nouvelles variables.
- † **DEL** : On élimine les  $\varepsilon$ -règles : Il faut d’abord trouver quelles sont les variables *annulables*, c’est à dire les variables qui peuvent engendrer le mot vide. On peut construire cet ensemble par induction : les variables qui ont une règle  $A \rightarrow \varepsilon$  son annulables, puis les variables qui ont une règle  $A \rightarrow A_1 A_2, \dots, A_n$  avec  $A_1, A_2, \dots, A_n$  annulables sont elles aussi annulables. Une fois que l’on a cet ensemble de variables annulables, on regarde chaque règle de notre grammaire, et s’il y a une variable annulable à droite, on ajoute la même règle mais sans cette variable. Par exemple, si on a la règle  $A \rightarrow A_1 A_2$  et que  $A_2$  est annulable, alors on ajoute la règle  $A \rightarrow A_1$ . Notez que cette construction peut introduire de nouvelles  $\varepsilon$ -règles. Une fois que l’on a fait ça, on supprime toutes les règles qui mènent à  $\varepsilon$  sauf pour l’axiome  $S_0$ .
- † **UNIT** : Enfin, on élimine les règles unitaires c’est-à-dire du type  $A \rightarrow B$  avec  $B \in V$ . Première étape, on considère le graphe de ces règles unitaires (donc on a une arrête de  $A$  vers  $B$  ssi la règle  $A \rightarrow B$  existe) et on regarde les composantes fortement connexes. Pour chacune d’entre elles, on choisit un représentant et on remplace toutes les autres variables de cette composante par ce représentant dans notre grammaire (aussi bien dans les membres gauches que dans les membres droits). Puis, on élimine les règles de la forme  $A \rightarrow A$ .  
Ensuite, ptant que l’on a une règle de la forme  $A \rightarrow B$ , on supprime cette règle et on la remplace par  $A \rightarrow \alpha$  pour chaque  $B \rightarrow \alpha$  que l’on a encore dans la grammaire. On fait ceci jusqu’à ce qu’il n’y ait plus de règles unitaires.

1. Rappel :  $V \cap \Sigma = \emptyset$

1. Montrez que cet algorithme est correct : Il finit, ne change pas le langage engendré et renvoie une grammaire en FNC
2. Quel est le langage reconnu par la grammaire :

$$\begin{aligned} S &\longrightarrow XC \mid \varepsilon \\ X &\longrightarrow aXbb \mid \varepsilon \\ C &\longrightarrow cC \mid \varepsilon \end{aligned}$$

3. Faites tourner l'algorithme sur cette grammaire (on pourra se passer de la première étape ici)

### Exercice 3.


Donner des automates à pile reconnaissant les langages suivants par état final, et justifier leur correction.

1.  $L = \{0^n 1^n \mid n \in \mathbb{N}\}$
2.  $L = \{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$
3.  $L = \{w \in \{a, b\}^* \mid |w|_a = 2|w|_b\}$
4.  $L = \{a^i b^j c^k \mid i, j, k \in \mathbb{N} \wedge i = j \vee j = k\}$

### Exercice 4.

Pour un automate à pile, on appellera configuration de l'automate tout triplet  $(q, p, m)$  où  $q$  est l'état courant,  $p \in \Gamma^*$  est le mot sur la pile et  $m \in \Sigma^*$  est le mot qu'il reste à lire en entrée.

On définit les automates à pile reconnaissant par pile vide de la manière suivante. Ce sont des automates à pile classiques qui commencent avec une lettre «\$» dans leur pile (*i.e.* dans la configuration :  $(q_0, \$, m)$ ) et qui acceptent le mot  $m$  si et seulement s'ils atteignent la configuration  $(q, \varepsilon, \varepsilon)$  pour un état  $Q$  quelconque. Dans ces automates, il n'y a donc pas d'état final.

 Montrer que ce modèle de calcul est équivalent à celui des automates à pile classiques.

### Exercice 5.

*Intersection*

1. Montrer que l'intersection d'un langage algébrique avec un langage rationnel sur le même alphabet est algébrique.