Corrigé TD 2
Langages Formels

Luc Lapointe
luc.lapointe@ens-paris-saclay.fr
home.lmf.cnrs.fr/LucLapointe/

Exercice 1 – Forme normale de Chomsky

Mettre les grammaires suivantes sous forme normale de Chomsky :

1. G_1 définie par les règles:

   $S \to aAa$
   $A \to Sb$
   $A \to bBB$
   $B \to abb$
   $B \to aC$
   $C \to aCA$

2. G_2 définie par les règles:

   $S \to AB|aS|a$
   $A \to Ab|\varepsilon$
   $B \to AS$

3. G_3 définie par les règles:

   $S \to TbT$
   $T \to TaT\ |ca$

Solution

1. $V_a \to a, V_b \to b$
   $S \to V_aS_1, S_1 \to AV_a$
   $A \to SV_b$
   $A \to V_bA_1, A_1 \to BB$
   $B \to V_aB_1, B_1 \to V_bV_b$
   $B \to V_aC$
   $C \to V_aC_1, C_1 \to CA$

2. $V_a \to a, V_b \to b$
   $S_0 \to \varepsilon \mid AB \mid V_aS \mid a$
   $S \to AB \mid V_aS \mid a$
   $A \to AV_b$
   $B \to AS$

3. $V_a \to a, V_b \to b, V_c \to c$
   $S \to TS_1, S_1 \to V_bT$
   $T \to TT_1, T_1 \to V_aT$
   $T \to V_cV_a$

Exercice 2 – Langages non algébriques

Montrer que les langages suivants ne sont pas algébriques :

$$L_1 = \left\{a^{n^2} \mid n \in \mathbb{N}\right\} \qquad L_2 = \{a^n b^m c^n d^m \mid n, m \in \mathbb{N}\}$$

Solution

1. Supposons que L_1 soit algébrique. Soit N la constante issue du lemme d'itération. On considère alors le mot a^{N^2} qui est dans L_1. Peu importe où sa factorisation est faite (vu que le mot ne contient qu'une seule lettre), il existe une constante $k \in [\![1, N]\!]$ telle que

$$a^{N^2} = xuyvz$$

vérifiant $|u| + |v| = k$ et $xu^2yv^2z \in L_1$. Or ce mot est a^{N^2+k} qui n'est pas dans L_1 : contradiction.

2. Supposons que L_2 soit algébrique. Soit N la constante issue du lemme d'itération. On considère le mot $a^N b^N c^N d^N$ qui est dans L_2. Quelle que soit sa factorisation par le lemme d'itération, un facteur de taille N va forcément se trouver dans un de ces cas :
   - Comprendre des a mais pas de c dans la partie de pomper, ou vice-versa
   - Comprendre des b dans la partie à pomper mais pas de d, ou vice-versa

   En appliquant le lemme de pompage, on se retrouve donc avec un mot qui n'a pas autant de a que de c, ou pas autant de b que de d, dans L_2. Contradiction.

Exercice 3 – Ambiguïté

1. Montrer que la grammaire suivante est ambiguë :
   $S \to \text{if } c \text{ then } S \text{ else } S$
   $S \to \text{if } c \text{ then } S$
   $S \to a$
2. Montrer que le langage engendré n'est pas ambigu.

Idée de solution

- « if c then if c then a else a » peut être généré par deux dérivations différentes : une qui associe le else au premier if, une au deuxième. C'est un problème dans la conception de compilateurs ou de syntaxes de langages connu sous le nom de *dangling else*.
- De la même manière que l'expression $a + b * c$ est ambiguë sans la règle de priorité du $*$ sur le $+$, on peut fixer une règle de priorité sur les else. Par exemple, les rattacher au if le plus proche.

$$S \to \text{if } c \text{ then } S_{\text{ie}} \text{ else } S$$
$$S \to \text{if } c \text{ then } S$$
$$S \to a$$
$$S_{\text{ie}} \to \text{if } c \text{ then } S_{\text{ie}} \text{ else } S_{\text{ie}}$$
$$S_{\text{ie}} \to a$$

Exercice 4 – Automate à pile à double sens

Un automate à pile à double sens peut, à chaque transition, déplacer sa tête de lecture vers la gauche ou vers la droite ou encore la laisser sur place. De plus, on supposera que la donnée w est encadrée par 2 symboles spéciaux (marqueurs) $\triangleleft$ et $\triangleright$. Le mot w est donc accepté par l'automate s'il y a un calcul réussi sur $\triangleleft\, w \,\triangleright$. De façon équivalente, un automate à pile à double sens est une machine de Turing qui ne peut pas modifier sa bande d'entrée et qui a une seule bande de travail qui est utilisée comme une pile.

1. Montrer que le langage $\{a^n b^n c^n \mid n \geq 1\}$ peut être accepté par un automate à pile déterministe à double sens.
2. Montrer que le langage $\{ww | w \in \Sigma^\star\}$ peut être accepté par un automate à pile à double sens.
3. Montrer que le langage $\{vcuvw | u, v, w \in \{a, b\}^\star\}$ peut être accepté par un automate à pile à double sens.

Idée de solution

- Commencer par lire des a et empiler. Au premier b lu, dépiler jusqu'aux c pour vérifier qu'il y a autant de a que de b. Puis, si c'est le cas, retourner en arrière pour recompter les b, et vérifier qu'il y en a autant que de c.
- Commencer par empiler les lettres du mot lu. Choisir de manière non déterministe le moment où on atteint le milieu, et depuis le milieu aller à la fin du mot et dépiler en vérifiant que le mot lu depuis la fin correspond à la pile.
- Empiler les lettres jusqu'au c. Puis aller jusqu'au bout du mot pour vérifier qu'il n'y a pas de c ailleurs. Depuis la fin du mot, retourner au début et décider de manière non déterministe quand est-ce qu'il faut commencer à vérifier que le mot lu correspond à la pile.

Contrôle continu – Langages non algébrique

Montrer que chacun des trois langages de l'exercice précédent n'est pas algébrique.