TD10 - Automates et Langages

novembre 2018

Exercice 1 :
Dans un langage de formatage de texte *à la Latex*, on utilise une notion de *boîte* pour positionner des caractères les uns par rapport aux autres. On distingue les boîtes horizontales dont les éléments sont disposés en ligne les uns derrière les autres et les boîtes verticales dont les éléments sont disposés en colonne, alignés à gauche. Les éléments composant une boîte sont des caractères ou peuvent être eux-mêmes des boîtes. Pour définir le langage de description de boîte, on propose la grammaire $G_1 = (X, V, B, R)$ avec $X = \{h\ ,\ v\ ,\ (\ ,\ )\ ,\ x\}$, $V = \{B, L, L'\}$ et $R = \{$

$$B \longrightarrow h\ (\ L\ )\ |\ v\ (\ L\ )\ \ |\ x$$
$$L \longrightarrow LB\ |\ B$$

$\}$ où $h(\dots)$ correspond à la description d'une boîte horizontale, $v(\dots)$ à la description d'une boîte verticale, et x à un caractère quelconque.
La description de boîte `v(x h(x v(x x) x) x)` correspond à :

```
x
x x x
  x
x
```

Q 1 . Trouver une description de boîte pour :

```
xx
 xx
```

Q 2 . La grammaire est récursive gauche. Construire une grammaire équivalente non récursive gauche
Q 3 . La grammaire obtenue est-elle LL(1) ?

Exercice 2 :
La grammaire ci-dessous est récursive gauche

$$S \to Sa \mid Bb$$
$$B \to Sc \mid Aa \mid b$$
$$A \to Sd \mid Be \mid f$$

Q 1 . Donnez un exemple illustrant la récursivité indirecte ($X \to \alpha.u \xrightarrow{i} Xw$ avec $\alpha \neq X$ et $i \geq 1$).
Q 2 . Nous allons, par étape, éliminer la récursivité gauche. Pour cela nous allons considérer chaque variable, successivement (nous prendrons par exemple l'ordre S,B,A qui est l'ordre dans lesquelles les règles ont été écrites, mais l'algorithme fonctionnerait de manière identique si on les prenait dans un ordre différent)

1. Éliminez la récursivité gauche **directe** concernant la variable S.
2. On s'intéresse maintenant aux règles dont B est partie gauche.
   (a) Supprimez la règle qui possède la forme $B \to Sw$. Vous la remplacerez par les règles $B \to uw$, pour chaque u tel que $S \to u$ est une règle existante.
   (b) Éliminez la récursivité gauche **directe** concernant B
3. On procède de la même manière avec les règles dont A est partie gauche.
   (a) Supprimez toutes les règles qui possèdent la forme $A \to Sw$ ou $A \to Bw$.
   (b) Éliminez la récursivité gauche **directe** concernant A

Exercice 3 :
On considère des arbres dont les nœuds sont étiquetés par des symboles. À chaque symbole est associé un entier positif ou nul : son arité qui définit le nombre de successeurs directs du nœud. Dans l'exercice, les symboles considérés seront f d'arité 2, g d'arité 1 et c de arité 0.

Un arbre t sera représenté par son écriture linéaire, c'est à dire sous forme d'une expression fonctionnelle : un symbole d'arité 0 est représenté seul. Un symbole d'arité > 0 doit être suivi entre parenthèses de la liste de ses successeurs.
Par exemple : c, $g(c)$, $f(c,c)$, $f(g(c), f(c, g(g(c))))$ et $f(f(g(c), c), g(g(c)))$ représentent bien des arbres sur cet alphabet tandis que ni $f(c)$ ni $f(g(c), g)$ ni $g(c,c)$ ne sont corrects.

Q 1 . Définissez une grammaire algébrique qui génère les écritures linéaires des arbres non vides corrects.
Q 2 . Définissez une grammaire algébrique qui génère les écritures des arbres non vides possédant un nombre pair de c
Q 3 . Définissez une grammaire algébrique qui génère les écritures des arbres non vides possédant un nombre pair de nœuds
Q 4 . On considère la grammaire obtenue en question 1. Montrez qu'elle est LL(1) en construisant sa table d'analyse.
Q 5 . On considère la grammaire obtenue en question 2. Expliquez de façon simple pourquoi elle n'est pas LL(1).
Q 6 . Transformez la grammaire pour faire en sorte qu'il n'y ait jamais, pour une même variable, deux règles commençant par le même symbole. Cela permet-il d'obtenir une grammaire LL(1) ?