Partiel MIM2 – Compilation

Yves Robert

Mardi 30 Octobre 2001, 13h30–15h30

1 Grammaires (moyen)

Les questions sont indépendantes.

1. Donner une grammaire qui génère les langages suivants :
   (a) Toute chaîne de x suivie du même nombre de y
   (b) Toute chaîne de x de taille n suivie par une chaîne de y de taille $2n+1$, pour tout $n \geq 0$
   (c) Toute chaîne de x et de y avec le même nombre de x et de y. Les x et les y peuvent apparaître dans n'importe quel ordre.
2. On considère la grammaire
$$\begin{array}{l} E \rightarrow E+T|T \\ T \rightarrow id|id()|id(L) \\ L \rightarrow E;L|E \end{array}$$
   E est le symbole de départ, les non-terminaux sont E, T et L. On demande de donner une grammaire LL(1) qui génère le même langage (bien montrer que votre grammaire transformée est LL(1)).
3. On considère la grammaire $S \rightarrow xSx|x$
   (a) Quel est le langage généré ?
   (b) La grammaire est-elle ambiguë ?
   (c) La grammaire est-elle LR(1) ?
4. Comme pour les grammaires LR(1), on peut dans l'analyse LL(1) tenir compte du contexte en calculant les ensembles FOLLOW non plus globalement pour chaque non-terminal, mais pour chaque production. Donner un exemple de grammaire où cette technique permet de résoudre un conflit FIRST-FOLLOW de l'analyseur LL(1) classique.
5. Donner une grammaire qui est non-ambiguë, factorisée à gauche, non récursive à gauche, et qui (pourtant !) n'est pas LL(1).

2 Attributs (facile)

On parenthèse les expressions avec des parenthèses (...) et des crochets [...]. L'idée est qu'on lit plus facilement $[(3+1)*5]$ que $((3+1)*5)$. Un fragment de la grammaire est

$$E \rightarrow int|(E)|[E]|E+E$$

On veut alterner strictement les deux types de parenthèse. Pour cela, on caractérise les expressions *strictes* comme suit :

(i) un entier est strict
(ii) $E_1 + E_2$ est stricte si E_1 et E_2 sont strictes
(iii) $[E]$ (resp : (E)) est stricte si E est stricte et si toutes les (éventuelles) sous-expressions parenthésées de E ont des $(\dots)$ (resp : des $[\dots]$) comme parenthèses les plus externes.

Augmenter les règles de la grammaire avec un calcul d'attributs qui assigne une valeur booléenne $E.stricte$ à toute expression E selon que E est stricte ou non.

3 Eléments productifs et accessibles (difficile)

Le graphe d'une grammaire est un graphe orienté biparti construit comme suit : on a un sommet v_N par non-terminal N, et un noeud v_p par règle de production p. Pour toute production p, il y a un arc sortant qui va de v_p au sommet correspondant au non-terminal à gauche de la production p, et des arcs entrants issus de chaque sommet correspondant à un non-terminal qui apparaît à droite de la production p.

On dit qu'un non-terminal N est productif s'il existe un mot terminal w tel que $N \stackrel{*}{\Longrightarrow} w$ (c'est le contraire d'un non-terminal inutile). On dit qu'un non-terminal N est accessible s'il existe une dérivation du type $S \stackrel{*}{\Longrightarrow} \alpha N \beta$, où S est le symbole de départ.

On *réduit* une grammaire en éliminant les non-terminaux non productifs et ceux non accessibles.

1. Soit la grammaire
$$\begin{array}{rl} 1: & S \to aX \\ 2: & X \to bS \\ 3: & X \to aYbY \\ 4: & Y \to ba \\ 5: & Y \to aZ \\ 6: & Z \to aZX \end{array}$$
   Donner son graphe et les non-terminaux productifs.
2. Proposer un algorithme pour déterminer les non-terminaux productifs d'une grammaire.
3. Soit la grammaire
$$\begin{array}{rl} 1: & S \to Y \\ 2: & Y \to YZ \\ 3: & Y \to Ya \\ 4: & Y \to b \\ 5: & U \to V \\ 6: & X \to c \\ 7: & V \to Vd \\ 8: & V \to d \\ 9: & Z \to ZX \end{array}$$
   Donner son graphe et les non-terminaux accessibles.
4. Proposer un algorithme pour déterminer les non-terminaux accessibles d'une grammaire.