Grammaires et analyse - Semaine du 26/01/2003

Principes des Analyseurs Lexicaux et Syntaxiques

Dominique Poulalhon, Aurelian Lavric , Yan Jurski

**Analyse syntaxique descendente**

Soit $G = (\Sigma, N, S, P)$ une grammaire non ambigüe, non récursive à gauche et factorisée à gauche. Si $S$ est le symbole initial de $G$, rajouter à $G$ un nouveau symbole initiale $S'$, un nouveau terminal $eof$ et la production $S' \to S\ eof$. Construire une table dont les indices de ligne sont les variables et les indices de colonnes les terminaux de $G$. La case $[X, x]$ sera remplie par l'ensemble de productions de la forme $X \longrightarrow \gamma$, tels que une de deux conditions suivantes est vérifiée :

- A partir de $\gamma$ on peut dériver une chaîne dont le premier élément est $x$ (on dit dans ce cas que $x \in premier(\gamma)$).
- La chaîne $\gamma$ est effaçable (c.à.d. $\gamma \longrightarrow^* \epsilon$), et $x$ peut suivre $X$ dans une phrase d'une dérivation gauche de $G'$ (on dit dans ce cas que $x \in suivant(X)$).

La table $T$ ainsi obtenue est la *table d'analyse LL(1)* de $G$, et, si chaque case de $T$ est remplie avec au plus une production, on peut obtenir à partir de $T$ un analyseur déterministe pour $G$, comme suit :

$$M_G = (\Sigma, \{q\}, \Sigma \bigcup N, \delta, q, S, \emptyset)$$

où $\delta$ est défini par :

$$\delta(q, a, a) = (q, \epsilon) \quad a \in \Sigma$$

$$\delta(q, A, a) = (q, \gamma) \;\; A \in N,\; a \in \Sigma,\; si\ T[A, a] = A \to \gamma$$

**Exercice 1** Soit $G$ la grammaire définie par :

S $\to$ a S b | a b

Construire la table d'analyse LL(1) de $G$ et simuler le fonctionnement de l'analyseur descendant associé sur le mot : aabb.

**Exercice 2** On considère la grammaire suivante qui reconnait un langage $\mathcal{L}$ sans epsilon :

$$\begin{array}{lcl} S & \longrightarrow & MS \mid c \\ M & \longrightarrow & Ac \mid AbV \\ V & \longrightarrow & S \\ A & \longrightarrow & a \end{array}$$

- Proposez le plus directement possible un automate à pile pour cette grammaire
- Mettez ensuite cet automate sous forme normale de Greibach et proposez à nouveau un automate à pile pour ce langage.
- Comparez l'interêt des deux automates pour obtenir un outil efficace qui reconnait $\mathcal{L}$
- Dans le cas de l'exemple précédent, le non-déterminisme est t'il réellement présent dans la grammaire ?
- On appelle k-forme normale de greibach une grammaire dont toutes les règles sont, soit de la forme $X \longrightarrow w$ avec $w \in \Sigma^*$ et $|w| \leq k$, soit de la forme $X \longrightarrow w.\gamma$ avec $w \in \Sigma^*, |w| = k$ et $\gamma \in (N + \Sigma)^*$. Montrer que pour tout entier k, un langage algébrique admet une k-forme normale de greibach.
- On considère à présent la grammaire

  $$\begin{array}{lcl} S & \longrightarrow & MNP \mid NP \\ M & \longrightarrow & NM \mid P \\ P & \longrightarrow & b \mid bP \\ N & \longrightarrow & a \end{array}$$

  Comment reconnaitre le mot aabab

  Mettre cette grammaire sous 2-forme normale de greibach. Avez vous un analyseur deterministe ?

  Les mot suivants sont ils dans le langage precedent : aaab, abaab, abba,baba

**Exercice 3** En lex les unités lexicales sont décrites à l'aide d'expressions régulières. Etant donné une expression régulière quelconque, peut-on se passer de l'opérateur de complémentation ?

Que peut on dire de :

- $\overline{a.(b|c)^*.d}$
- $\overline{(a.b|a.a.b)^*.a.a}$

**Exercice 4** Soit $\Sigma = \{a, b\}$. Donner un automate non-détérministe qui réconnait le langage $\Sigma^* ababaa \Sigma^*$. Détérminiser cette automate. Minimiser l'automate. En déduire un algorithme pour tester si $ababaa$ est un facteur d'un mot $w$.

**Exercice 5** *Lex construit les automates deterministes minimaux assocés aux expressions régulières. Que cela donne t'il sur les expressions :*

- $(a|b)^*.a.(a|b)$
- $(a|b)^*.a.(a|b).(a|b)$
- $(a|b)^*.a.(a|b).(a|b).(a|b)$
- *Montrer que tout automate fini pour* $(a|b)^*.a.(a|b).(a|b).\ldots.(a|b)$*, où il y a* $n-1$ *occurrences de* $(a|b)$ *à la fin, doit avoir au moins* $2^n$ *etats.*

*Peut t'on s'attendre à un tel phénomène dans l'utilisation de Lex pour un compilateur ?*

**Exercice 6** Donner des expressions rationnelles pour les langages suivants :

- L'ensemble des identificateurs dans PASCAL (ou C)
- L'ensemble des chaînes de caractères pour
  - Les entiers (par exemple 100 ou $-42$ mais pas 007),
  - Les nombres réelles avec exponent (ou non), par exemple $1.02E-3$.
- toutes les suites de 0 et de 1 qui ne contiennent pas la sous suite 011
- toutes les suites de 0 et de 1 qui ne contiennent pas la sous chaine 011
- toutes les suites de chiffres sans chiffre répété