TD4 : Parsing, Transformation & Automates à pile

Timothée Bernard

7 octobre 2015

1 Parsing descendant

1. Ébaucher l'arbre d'exploration des solutions pour une analyse descendante pour la grammaire $S \to S + S \mid a \mid b$ et le mot reconnu $a + b$.

   Solution: Le problème qui se pose ici est le problème dit de « récursivité gauche »

2. Est-ce qu'une grammaire régulière apporte un avantage par rapport à une grammaire algébrique quelconque du point de vue des algorithmes d'analyse vus en cours (descendant) ?

   Solution: Dans le cas d'une analyse descendante, un grammaire régulière nous permet de faire du « regard en avant » : dans une situation $(S\gamma, a\delta)$, on peut se contenter de tenter les règles de la forme $S \to aA$, ce qui réduit donc l'espace de recherche. De plus, une grammaire régulière n'est jamais récursive gauche.
   Pas d'avantage particulier pour l'algorithme ascendant, le nombre de conflits « shift/reduce » n'étant pas réduit par rappport à une grammaire algébrique.

2 Transformations de Grammaires

3. Soit le langage L_1 sur le vocabulaire $V = \{$*l'*, *homme*, *ours*, *qui*, *a*, *vu*$\}$ formé de l'ensemble des phrases finies de la forme *l'homme qui a vu l'ours*, *l'homme qui a vu l'homme qui a vu l'ours*, *l'homme qui a vu l'homme qui a vu ... qui a vu l'ours*.
   (a) Donner une grammaire algébrique (*context-free*) engendrant L_1.

   Solution:

S	$\to$	NP Rel
NP	$\to$	l' homme
Rel	$\to$	qui a vu NP Rel
	$\mid$	qui a vu l' ours

   (b) Mettre en forme normale de Chomsky la grammaire obtenue.

   Solution: La grammaire est déja propre (ϵ-libre, pas de symboles inutiles, pas de productions singulières).
   On garde la règle $S \to NP\ Rel$ et on ajoute :

$$
\begin{array}{lcl}
NP & \to & X_{l'}\ X_{homme} \\
Rel & \to & X_{qui}\ A_1 \\
A_1 & \to & X_a\ A_2 \\
A_2 & \to & X_{vu}\ A_3 \\
A_3 & \to & NP\ Rel \\
Rel & \to & X_{qui}\ B_1 \\
B_1 & \to & X_a\ B_2 \\
B_2 & \to & X_{vu}\ B_3 \\
B_3 & \to & X_{l'}\ X_{ours} \\
X_{l'} & \to & \text{l'} \\
X_{homme} & \to & \text{homme} \\
X_{ours} & \to & \text{ours} \\
X_{qui} & \to & \text{qui} \\
X_a & \to & \text{a} \\
X_{vu} & \to & \text{vu}
\end{array}
$$

4. Dérécursiver à gauche la grammaire suivante :
$$
\begin{array}{lcl}
X_1 & \to & X_2\ X_3 \\
X_2 & \to & X_3\ X_1 \mid b \\
X_3 & \to & X_2\ X_2 \mid a
\end{array}
$$

Solution: Rappel du principe d'élimination de la récursivité :

On commence par empêcher de se retrouver dans la situation :

- $A_i \to A_j\alpha$
- $A_j \to A_i\beta$

Pour cela, pour toute paire A_i,A_j, on laisse (arbitrairement) $A_j \to A_i\beta$, mais on remplace intelligemment $A_i \to A_j\alpha$, en y remplaçant A_j par toutes ses parties droites.

Algorithme :

1: Nettoyer la grammaire.
2: **for** i de 1 à n **do**
3: **for** j de 1 à $i-1$ **do**
4: **if** $A_i \to A_j\alpha$ **then**
5: la remplacer pour $A_i \to \beta_1\alpha \mid \beta_2\alpha \mid \ldots \mid \beta_h\alpha$ ▷ où $A_j \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_h$
6: **end if**
7: **end for**
8: Éliminer récursivité immédiate des A_i-productions.
9: **end for**

En suivant l'algorithme, on arrive à :

$$
\begin{array}{lcl}
X_1 & \to & X_2\ X_3 \\
X_2 & \to & X_3\ X_1 \mid b \\
X_3 & \to & b\ X_2\ X_3' \mid a\ X_3' \\
X_3' & \to & X_1\ X_2\ X_3' \mid \varepsilon
\end{array}
$$

5. Proposer une grammaire en forme de Greibach équivalente à la grammaire suivante :

$S \longrightarrow a\ S\ B \mid B\ A$
$A \longrightarrow S\ b\ c \mid A\ c$
$B \longrightarrow b\ S\ B \mid b$

Solution: Il faut nettoyer la grammaire. Mais ici la grammaire est déjà propre. Pas de récursivité indirecte, il faut juste traiter la récursivité directe de A. Cela donne :

$S \longrightarrow aSB \mid BA$
$A \longrightarrow SbcA'$
$A' \longrightarrow cA' \mid \epsilon$
$B \longrightarrow bSB \mid b$

Résultat final :

$X \longrightarrow b$
$Y \longrightarrow c$
$S \longrightarrow aSB \mid bSBA \mid bA$
$A \longrightarrow aSBXYA' \mid bSBAXYA' \mid bXYA'$
$A' \longrightarrow cA' \mid \epsilon$
$B \longrightarrow bSB \mid b$

3 Automates à pile

6. Trouver un automate à pile qui accepte le langage $\{w\overline{w} \,/\, w \in \{a,b\}^*\}$, où $\overline{w}$ désigne le mot miroir de w.

Solution: Principe : on empile tous les symboles, et on dépile ensuite. Automate non déterministe.

Automate à pile : $(1,1)$ $x, \varepsilon \to x$
$(1,2)$ $\varepsilon, \varepsilon \to \varepsilon$
$(2,2)$ $x, x \to \varepsilon$

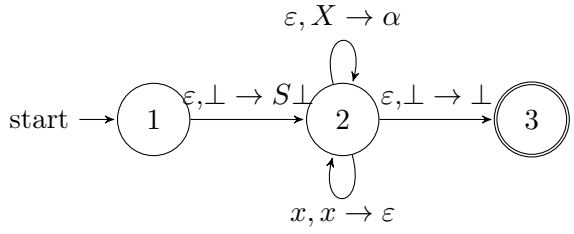
(1 initial , 2 terminal)

7. Proposer un automate à pile pour la grammaire algébrique suivante : $S \rightarrow a \mid aa \mid b \mid bb \mid aSa \mid bSb$

Solution:

$\forall$ règle $X \to \alpha \in P$

$\forall x \in \Sigma$



8. Proposer un automate à pile pour $L_1 = \{w \in \{a,b\}^* | w_a = w_b\}$ (tout mot contenant autant de a que de b).

Solution:

$(1,1)$ $a, \epsilon \to +$
$(1,1)$ $b, + \to \epsilon$

1 initial et final

9. Proposer un automate à pile pour $L_2 = \{a^n b^n | n \geq 0\}$.

Solution:

$(1,1)$ $a, \epsilon \to +$

$(1,2)$ $\epsilon \to \epsilon$

$(1,1)$ $b, + \to \epsilon$

1 initial et 2 final