IR

INSA de Toulouse — 4ème année

TD4 : Parsing ascendant et grammaires LR

On prend la grammaire G_1 (d'axiome E) : $\begin{cases} E \to (\ E\ +\ E\ ) \\ E \to x \end{cases}$

Définition. *On va définir un automate dont les états sont des ensembles d'items.*

- *Un item est un couple dérivation/position : $A \to u \bullet v$.*
- *La fermeture d'un item est l'ensemble F des items tels que :*
  si $A \to u \bullet Xv \in F$, alors pour chaque règle $X \to w$, on a $X \to \bullet w \in F$.
- *On construit l'automate LR d'une grammaire de la manière suivante :*
  - *l'état initial est la fermeture des règles dérivant l'axiome,*
  - *les transitions δ étiquetées par le caractère a vont d'un état E à un état E', i.e. $\delta(E, a) = E'$ si*
  $$E' = FERMETURE\{A \to ua \bullet v | A \to u \bullet av \in E\}.$$

1. Déterminer l'automate correspondant à G_1.
2. Un algorithme de parsing utilisant cet automate fonctionne de la manière suivante :

   Algorithme 1 : Algorithme de parsing $LR(0)$

   Entrées : $(a_i)_{1 \leq i \leq n}$ un mot de T,
   $G = (\Sigma, S, V, T)$ une grammaire (Σ alphabet, S axiome, V non-terminaux et T terminaux).
   A automate LR de G d'états Q, d'état initial q_I et de fonction de transition δ

   $P \leftarrow$ pile vide d'états
   `Empiler`(P, q_I)
   $i \leftarrow 1$
   tant que $i \leq n$ *ou non* `Est_Vide`(P) **faire**
       si $i \leq n$ *et* $\exists e \in Q, \delta($`Sommet`$(P), a_i) = e$ **alors**
           `Empiler`(P, e)
           $i \leftarrow i + 1$
       sinon
           Soit X et k tels que $\exists I \in$ `Sommet`$(P), I : X \to u_1 \dots u_k \bullet$ avec $\forall i, u_i \in V \cup T$
           pour j *de* 1 *à* k **faire**
               `Depiler`(P)
           si $\exists e \in Q, \delta($`Sommet`$(P), X) = e$ **alors**
               `Empiler`(P, e)

   Pour une itération de la boucle principale, si i est incrémenté, on dit que cette étape est un *décalage* (shift). Sinon, on parle de *réduction* (reduce).
   Quels problèmes peut rencontrer l'algorithme 1 ?
3. Appliquer l'algorithme 1 pour parser $(x + (x + x))$.
4. Générer l'automate LR de la grammaire G_2 (d'axiome S) : $\begin{cases} S \to A\ x \\ A \to a\ A\ |\ a \end{cases}$
5. Expliquer pourquoi l'algorithme 1 n'est pas utilisable en l'état.
6. Proposer une adaptation simple pour G_2.
7. Mêmes questions pour la grammaire G_3 (d'axiome S) : $\begin{cases} S \to Is \\ Is \to I\ Is \\ Is \to \varepsilon \end{cases} \begin{cases} I \to affectation; \\ I \to If \\ If \to if\ \{\ Is\ \} \end{cases}$