TD6 : Parsing

Timothée Bernard

21 octobre 2015

1 Parsing LL

1. Éliminer la récursion (gauche directe) de la grammaire ETF :

$$\begin{array}{lcl} \text{E} & \to & \text{T} \mid \text{E} + \text{T} \\ \text{T} & \to & \text{F} \mid \text{T} * \text{F} \\ \text{F} & \to & \text{a} \mid (\ \text{E}\ ) \end{array}$$

Solution:

1	$E \to TE'$	2	$E' \to +\,TE'$	3	$E' \to \varepsilon$
4	$T \to FT'$	5	$T' \to *\,FT'$	6	$T' \to \varepsilon$
7	$F \to (\,E\,)$	8	$F \to a$		

2. Donner les ensembles PREMIER et SUIVANT pour chaque non terminal et construire la table de prédiction LL associée à la grammaire obtenue à la question précédente [1].

Solution: premier(E) = premier(T) = premier(F) = $\{(, a\}$
premier(E') = $\{+, \varepsilon\}$
premier(T') = $\{*, \varepsilon\}$
suivant(E) = suivant(E') = $\{), \$\}$
suivant(T) = suivant(T') = $\{+,), \$\}$
suivant(F) = $\{+, *,), \$\}$

	a	$+$	$*$	$($	$)$	$\$$
E	TE'			TE'		
E'		$+TE'$			ε	ε
T	FT'			FT'		
T'		ε	$*FT'$		ε	ε
F	a			(E)		

Construction de la table LL(1) quand on possède PREMIER et SUIVANT

Fonction CONSTRUCTIONLL(M) ▷ M est la table LL(1) et $G = \langle \Sigma, V, S, P \rangle$
  Pour $A \to \alpha \in P$ **Faire** ▷ α s'entend comme le premier élément de la règle.
    Pour $t \in$ Premier(α) **Faire**
      Ajouter $A \to \alpha$ à $M[A, t]$
    Fin Pour
    Si $\varepsilon \in$ Premier(α) **Alors**
      Pour $t \in$ Suivant(A) **Faire**
        Ajouter $A \to \alpha$ dans $M[A, t]$
      Fin Pour
      Si \$ $\in$ Suivant(A) **Alors**

1. Ne pas oublier le marqueur de fin \$.

```
11:                Ajouter A → α dans M[A, $]
12:            Fin Si
13:        Fin Si
14:    Fin Pour                          ▷ Toutes les cases indéfinies sont des erreurs.
15: Fin Fonction
```

3. Appliquer une version modifiée de l'algorithme de parsing descendant vu en TD il y a deux semaines, en utilisant la table LL, pour analyser le mot $a + a * (a + a)$.
4. Soit la grammaire suivante : $\left\{ \begin{array}{l} S \to a \mid b \mid (\ T\ ) \\ T \to T\ ,\ S \mid S \end{array} \right\}$
   (a) Donnez un arbre de dérivation pour les mots (a, b) et $(b, (a, a))$
   (b) La grammaire est-elle LL(1) ?
   (c) Eliminer la récursivité à gauche.
   (d) Montrer que la nouvelle grammaire est LL(1). Donner la table d'analyse.

Solution: La grammaire n'est pas LL(1) car la table contient deux règles dans la case $(T,\ a)$: la règle $T \to T,\ S$ et la règle $T \to S$.
Elimination de la récursivité gauche : il suffit de s'occuper de la récursivité directe de la règle $T \to T,\ S$. La version non récursive gauche est : $T \to SA'$, $A' \to ,\ SA'$ | ε

Table LL(1) :

	a	b	,	(	)
S	$S \to a$	$S \to b$		$S \to (T)$	
T	$T \to SA'$	$T \to SA'$		$T \to SA'$	
A'			$A' \to,\ SA'$		$A' \to \varepsilon$

2 Parsing LR

5. Soit la grammaire suivante :

$$\begin{array}{lcl} \text{S} & \to & \text{A B} \\ \text{A} & \to & \text{a A} \mid \text{a} \\ \text{B} & \to & \text{a B} \mid \text{b} \end{array}$$

   (a) Calculer les contextes LR(0) de chaque règle.
   (b) Appliquer l'algorithme naïf (shift-reduce + contextes LR(0)) sur l'entrée *aab* (dessiner l'arbre des configurations explorées, en précisant là où les contextes LR(0) ont permis d'éviter de fausses routes).