Théorie des langages : THL

CM 5

Uli Fahrenberg

EPITA Rennes

S5 2024

Aperçu

Programme du cours

1. Langages rationnels, automates finis
3. Langages algébriques, grammaires hors-contexte, automates à pile
5. Parsage LL, partie 1
6. TP 1 : `flex`
7. Parsage LL, partie 2
8. TP 2 : parsage LL
9. Parsage LR
10. TP 3, 4 : `flex` & `bison`

La dernière fois : parsage

Problème de parsage

Pour une grammaire hors contexte G, construire un algorithme de parsage qui

- pour un mot w, decide si $w \in L(G)$,
- et dans le cas $w \in L(G)$, retourne l'arbre de dérivation.

- arbre de dérivation de w $\hat{=}$ sémantique de w

Nos algorithmes de parsage devrait

- pouvoir traiter des grammaires non-ambiguës
- avoir une complexité linéaire en taille d'entrée
- lire w de gauche à droite sans retour arrière

La dernière fois : parsage LL(1)

- approche descendante
- lire le mot w de gauche à droite / Left-to-right
  - sans passer à l'arrière
- construire une dérivation gauche / Leftmost
- en accordant, à chaque pas, le premier symbole de w avec le côté droit d'une production
  - donc avec lookahead 1

- parsage LL(k) : lookahead k / « fenêtre de k lexèmes »
- peu utilisé

La dernière fois : algorithme LL(1)

0. entrée : une grammaire hors contexte $G = (N, \Sigma, P, S)$
   - si-dessous, $V = N \cup \Sigma$
   - éliminer récursion à gauche dans G ; factoriser G à gauche
1. calculer NULL
   - $\text{NULL} = \{A \in N \mid A \Rightarrow^* \varepsilon\}$
2. construire la table FIRST
   - $\text{FIRST}(A) = \{a \in \Sigma \mid \exists w \in V^* : A \Rightarrow^* aw\}$
3. construire la table FOLLOW
   - $\text{FOLLOW}(A) = \{a \in \Sigma \mid \exists B \in N, \alpha, \beta \in V^* : B \Rightarrow^* \alpha A a \beta\}$
4. construire la TABLE de parsage :
   1. pour chaque production $X \to w \quad (n)$:
      1. pour chaque $a \in \text{FIRST}(w)$: $\text{TABLE}(X, a)$ += $\{n\}$
      2. si $w \in \text{NULL}$ ou $w = \varepsilon$:
         – pour chaque $a \in \text{FOLLOW}(X)$: $\text{TABLE}(X, a)$ += $\{n\}$

La dernière fois : LL(1)

Définition (8.5)

G est LL(1) si chaque TABLE(A, a) contient au maximum une production.

La dernière fois : exemple

$$
\begin{aligned}
Z &\to XYZ && (1)\\
&\mid c && (2)\\
X &\to a && (3)\\
&\mid Y && (4)\\
Y &\to b && (5)\\
&\mid \varepsilon && (6)
\end{aligned}
$$

La dernière fois : exemple

$$
\begin{aligned}
Z &\to XYZ && (1)\\
&\mid c && (2)\\
X &\to a && (3)\\
&\mid Y && (4)\\
Y &\to b && (5)\\
&\mid \varepsilon && (6)
\end{aligned}
$$

$$\text{NULL} = \{X, Y\}$$

La dernière fois : exemple

$$
\begin{aligned}
Z &\to XYZ && (1)\\
&\mid c && (2)\\
X &\to a && (3)\\
&\mid Y && (4)\\
Y &\to b && (5)\\
&\mid \varepsilon && (6)
\end{aligned}
$$

$\text{NULL} = \{X, Y\}$

A	FIRST(A)
X	a, b
Y	b
Z	c, a, b

La dernière fois : exemple

$$
\begin{aligned}
Z &\to XYZ && (1)\\
&\mid c && (2)\\
X &\to a && (3)\\
&\mid Y && (4)\\
Y &\to b && (5)\\
&\mid \varepsilon && (6)
\end{aligned}
$$

$\text{NULL} = \{X, Y\}$

A	FIRST(A)
X	a, b
Y	b
Z	c, a, b

A	FOLLOW(A)
X	a, b, c
Y	a, b, c
Z	

La dernière fois : exemple

$$
\begin{aligned}
Z &\to XYZ && (1)\\
&\mid c && (2)\\
X &\to a && (3)\\
&\mid Y && (4)\\
Y &\to b && (5)\\
&\mid \varepsilon && (6)
\end{aligned}
$$

$\mathrm{NULL} = \{X, Y\}$

A	FIRST(A)
X	a, b
Y	b
Z	c, a, b

A	FOLLOW(A)
X	a, b, c
Y	a, b, c
Z	

	a	b	c
X	3, 4	4	4
Y	6	5, 6	6
Z	1	1	1, 2

Exemples

Exemple (tableau)

$$\begin{aligned} S &\to FS && (1)\\ &\mid Q && (2)\\ &\mid \text{'('}\, S\, \text{')'}\, S && (3)\\ F &\to \text{'!'} && (4)\\ Q &\to \text{'?'} && (5) \end{aligned}$$

« Une session est une séquence de faits suivi par une question ; sous-sessions sont permis »

Exemple (tableau)

$$
\begin{aligned}
Z &\to S\$ && (1)\\
S &\to LQ && (2)\\
&\quad \mid \texttt{'('}\ S\ \texttt{')'}\ S && (3)\\
L &\to FL && (4)\\
&\quad \mid \varepsilon && (5)\\
F &\to \texttt{'!'} && (6)\\
Q &\to \texttt{'?'} && (7)
\end{aligned}
$$

Plus ça change, ...

Implémentation

Grammaire :

$$
\begin{aligned}
S &\to F && (1)\\
&\quad \mid \text{'('}\ S\ \text{'+'}\ F\ \text{')'} && (2)\\
F &\to \text{'}\mathtt{a}\text{'} && (3)
\end{aligned}
$$

Simple parseur en Python :

`parse.py`


That's all Folks!