编译原理与技术 H5

PB18111697 王章瀚

4.12

为文法

$$S \to (L)|a$$
$$L \to L, S|S$$

(b)

分别写出相应的语法制导定义, 翻译方案以及预测翻译器, 它打印出每个 a在句子中是第几个字符. 例如, 当句子是 $(a,(a,(a,a),(a)))$ 时, 打印结果是 2 5 8 10 14

消除了左递归的1,2问答案在第3问都有描述

1. 语法制导定义

分析:

- 字符a的位置能够通过其左边部分判断, 因此使用L属性即可解决
- 字符a的位置不可能由字符a本身就判定, 因此需要继承属性

综上: 用$first$(继承属性)和$last$(综合属性)表示一个文法符号的第一个字符和最后一个字符的位置， 并加入一个$S' \to S$以作为初始情况.

产生式	语义规则
$S' \to S$	$L.first = 1$
$S \to (L)$	$L.first = S.first + 1$ $S.last = L.last + 1$
$S \to a$	$S.last = S.first$ $print(S.first)$
$L \to L_1, S$	$L_1.first = L.first$ $S.first = L_1.last + 2$ $L.last = S.last$
$L \to S$	$S.first = L.first$ $L.last = S.last$

2. 翻译方案

根据$first$和$last$的综合和继承性, 很容易写出以下翻译方案:

$$
\begin{aligned}
&S' \to \{S'.first = 1;\}S\\
&S \to \{L.first = S.first;\}(L)\{S.last = L.last + 1;\}\\
&S \to a\{S.last = S.first;\ print(S.first);\}\\
&L \to \{L_1.first = L.first\}L_1, \{S.first = L_1.last + 2;\}S\{L.last = S.last;\}\\
&L \to \{S.first = L.first;\}S\{L.last = S.last;\}
\end{aligned}
$$

3. 预测翻译器

为了设计预测翻译器, 首先需要消除左递归, 使之成为LL(1)文法. 改进后文法如下:

$$
\begin{aligned}
&Q \to S\\
&S \to (L)\\
&S \to a\\
&L \to SR\\
&R \to , SR_1\\
&R \to \epsilon
\end{aligned}
$$

相应语法制导定义为:

产生式	语义规则
$Q \to S$	S.first=1
$S \to (L)$	L.first=S.first S.last=L.last+1
$S \to a$	S.last=S.first print(S.first)
$L \to SR$	S.first=L.first R.first=S.last+1 L.last=R.last
$R \to , SR_1$	$S.first = R.first + 1$ $R_1.first = S.last + 1$ $R.last = R_1.last$
$R \to \epsilon$	$R.last = R.fisrt$

从而得到翻译方案:

$$
\begin{aligned}
&Q \to \{S.first = 1;\}S\\
&S \to \{L.first = S.first;\}(L)\{S.last = L.last + 1;\}\\
&S \to a\{S.last = S.first;\ print(S.first);\}\\
&L \to \{S.first = L.first\}S\{R.first = S.last + 1\}R\{L.last = R.last\}\\
&R \to , \{S.first = R.first + 1;\}S\{R_1.first = S.last + 1;\}R_1\{R.last = R_1.last\}\\
&R \to \{R.last = R.fisrt\}
\end{aligned}
$$

进而能够写出其预测翻译器代码如下(见下页), 为了代码言简意赅, 这里没有加入错误处理:

```
// 分析 Q 的函数
void Q() {
    int s_first = 1;
    S(s_first);
}

// 分析 S 的函数
int S(int s_first) {
    int s_last;

    if(lookahread == 'a') { // 产生式 S->a
        match('a');
        s_last = s_first;
    }
    else { // 产生式 S->(L)
        int l_first, l_last;
        l_first = s_first;
        match('(');
        l_last = L(l_first);
        match(')');
        s_last = l_last + 1;
    }
    return s_last
}

// 分析的函数L
int L(int l_first) { //产生式 L->SR
    int l_last, s_first, s_last, r_first, r_last;

    s_first = l_first;
    s_last = S(s_first);
    r_first = s_last + 1;
    r_last = R(r_first);
    l_last = r_last;
    return l_last;
}

// 分析的函数R
int R(int r_first) {
    int r_last;

    if(lookahead == ',') { // 产生式 R->,SR1
        int s_first, s_last, r1_first, r1_last;
        s_first = r_first + 1;
        s_last = S(s_first);
        r1_first = s_last + 1;
        r1_last = R(r1_first);
        r_last = r1_last;
    }
    else {
        r_last = r_first;
    }
    return r_last;
}
```