

第五章 语法制导翻译 在递归的预测分析 过程中进行翻译

哈尔滨工业大学 陈鄞



算法

- 为每个非终结符 A 构造一个函数， A 的每个继承属性对应该函数的一个形参，函数的返回值是 A 的综合属性值。对出现在 A 产生式中的每个文法符号的每个属性都设置一个局部变量
- 非终结符 A 的代码根据当前的输入决定使用哪个产生式
 - 若当前输入在 A 的某个候选式的 Select 集中：使用该候选式
 - 若当前输入不在 A 的所有候选式的 Select 集中：出错

算法（续）

终结符

➤ 与每个产生式有关的代码执行如下动作：从左到右考虑产生式右部的词法单元、非终结符及语义动作

➤ 对于带有综合属性 x 的词法单元 X ，把 x 的值保存在局部变量 $X.x$ 中；然后产生一个匹配 X 的调用，并继续输入

➤ 对于非终结符 B ，产生一个右部带有函数调用的赋值语句 $c := B(b_1, b_2, \dots, b_k)$ ，其中， b_1, b_2, \dots, b_k 是代表 B 的继承属性的变量， c 是代表 B 的综合属性的变量

返回值

➤ 对于每个动作，将其代码复制到语法分析器，并把对属性的引用改为对相应变量的引用

形参

在递归的预测分析过程中进行翻译

➤ 例

SDT

1) $T \rightarrow F \{ T'.inh = F.val \} T'$
 $\{ T.val = T'.syn \}$

2) $T' \rightarrow *F \{ T_1'.inh = T'.inh \times F.val \} T_1'$
 $\{ T'.syn = T_1'.syn \}$

3) $T' \rightarrow \varepsilon \{ T'.syn = T'.inh \}$

4) $F \rightarrow \text{digit} \{ F.val = \text{digit.lexval} \}$

为每个非终结符 A 构造一个函数， A 的每个继承属性对应该函数的一个形参，函数的返回值是 A 的综合属性值

对出现在 A 产生式右部中的每个文法符号的每个属性都设置一个局部变量

对于每个动作，将其代码复制到语法分析器，并把对属性的引用改为对相应变量的引用

```
 $T'.syn$   $T'(token, T'.inh)$   
{  $D: Fval, T_1'.inh, T_1'.syn$ ;  
  if token="*" then  
  {  $Getnext(token)$ ;  
     $Fval = F(token)$ ;  
     $T_1'.inh = T'.inh \times Fval$ ;  
     $Getnext(token)$ ;  
     $T_1'.syn = T_1'(token, T_1'.inh)$ ;  
     $T'.syn = T_1'.syn$ ;  
    return  $T'.syn$ ;  
  }  
  else if token = "$" then  
  {  $T'.syn = T'.inh$ ;  
    return  $T'.syn$ ;  
  }  
  else Error;  
}
```

在递归的预测分析过程中进行翻译

➤ 例

SDT

1) $T \rightarrow F \{ T'.inh = F.val \} T'$

$\{ T.val = T'.syn \}$

2) $T' \rightarrow *F \{ T_1'.inh = T'.inh \times F.val \} T_1'$

$\{ T'.syn = T_1'.syn \}$

3) $T' \rightarrow \varepsilon \{ T'.syn = T'.inh \}$

4) $F \rightarrow \text{digit} \{ F.val = \text{digit.lexval} \}$

Tval T(token)

{

D: Fval, T'inh, T'syn;

Fval = F(token);

T'inh = Fval;

Getnext(token);

T'syn = T_1' (token, T'inh);

Tval = T'syn;

return Tval;

}

在递归的预测分析过程中进行翻译

➤ 例

SDT

1) $T \rightarrow F \{ T'.inh = F.val \} T'$

$\{ T.val = T'.syn \}$

2) $T' \rightarrow *F \{ T_1'.inh = T'.inh \times F.val \} T_1'$

$\{ T'.syn = T_1'.syn \}$

3) $T' \rightarrow \varepsilon \{ T'.syn = T'.inh \}$

4) $F \rightarrow \text{digit} \{ F.val = \text{digit.lexval} \}$

Fval $F(token)$

{

if token \neq digit then Error;

Fval=token.lexval;

return Fval;

}

在递归的预测分析过程中进行翻译

➤ 例

SDT

- 1) $T \rightarrow F \{ T'.inh = F.val \} T'$
 $\{ T.val = T'.syn \}$
- 2) $T' \rightarrow *F \{ T_1'.inh = T'.inh \times F.val \} T_1'$
 $\{ T'.syn = T_1'.syn \}$
- 3) $T' \rightarrow \varepsilon \{ T'.syn = T'.inh \}$
- 4) $F \rightarrow \text{digit} \{ F.val = \text{digit.lexval} \}$

Desent()

{

D: Tval;


Getnext(token);

Tval = T(token);

if token ≠ “\$” then Error;

return ;

}



第五章 语法制导翻译 在递归的预测分析 过程中进行翻译

哈尔滨工业大学 陈鄞

