

在递归的预测分析过程中进行翻译 { D: Fval, T, 'inh, T, 'sı

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

〉例

SDT

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

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

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

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

```
{ 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
   LT'syn=T'inh;
    return T'syn;
   else Error;
```

在递归的预测分析过程中进行翻译

```
Tval T(token)
〉例
                                                               D: Fval, T'inh, T'syn;
SDT
                                                               Fval = F(token);
1) T \rightarrow F \{ T'.inh = F.val \} T'
                                                                T'inh = Fval;
   \{ T.val = T'.syn \}
                                                               Getnext(token);
2) T' \rightarrow *F \{ T_1'.inh = T'.inh \times F.val \} T_1'
                                                               T'syn = T_1' (token, T'inh);
   \{ T'.syn = T_1'.syn \}
                                                               Tval = T'syn;
3) T' \rightarrow \varepsilon \{ T'.syn = T'.inh \}
                                                               return Tval;
4) F \rightarrow \text{digit} \{ F.val = \text{digit.} lexval \}
```

在递归的预测分析过程中进行翻译

```
Fval F(token)

SDT

{
1) T \rightarrow F \{ T'.inh = F.val \} T' if token \neq digit then Error;
\{ T.val = T'.syn \} Fval=token.lexval;
\{ T'.syn = T_1'.inh = T'.inh \times F.val \} T_1' return Fval;
\{ T'.syn = T_1'.syn \} }

3) T' \rightarrow \varepsilon \{ T'.syn = T'.inh \}
4) F \rightarrow digit \{ F.val = digit.lexval \}
```

在递归的预测分析过程中进行翻译

```
Desent()

> ⑤ 

SDT

D: Tval;

Tval = T'.syn \}

T' \rightarrow *F \{ T'.inh = T'.inh \times F.val \} T'

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

T' \rightarrow *E \{ T'.syn = T'.syn \}

T' \rightarrow *E \{ T'.syn = T'.inh \}
```

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

算法(续)

- ▶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
 - \triangleright 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 $c:=B(b_1,b_2,...,b_k)$,其中, $b_1,b_2,...,b_k$ 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
 - ▶ 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





L-属性定义的自底向上翻译

▶给定一个以LL文法为基础的L-SDD,可以 修改这个文法,并在LR语法分析过程中计 算这个新文法之上的SDD

修改

例

```
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 \operatorname{digit} \{ F. val = \operatorname{digit} .lexval \}
                                                                         修改后的SDT,
           1) T \rightarrow F M T' \{ T.val = T'.syn \}
```

标记非终结符 (Marker Nonterminal)

继承属性

 $M \rightarrow \varepsilon \{M.i = F.val; M.s = M.i\}$ 2) $T' \rightarrow F N T_1' \{ T'.syn = T_1'.syn \}$ $N \rightarrow \varepsilon \{N.i1 = T'.inh\}$ N.i2 = F.val; $N.s = N.i1 \times N.i2$

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

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

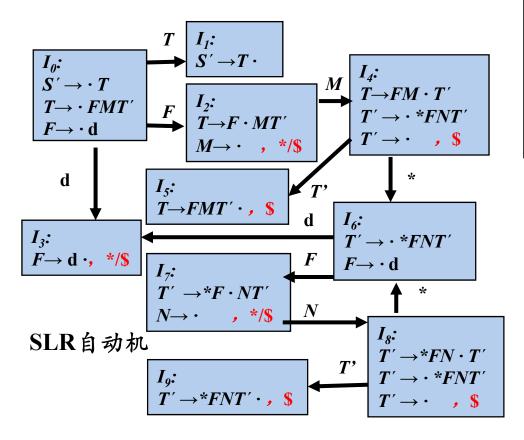
所有语义动作都 位于产生式末尾

访问未出现在 该产生式中的 符号的属性?

综合属性

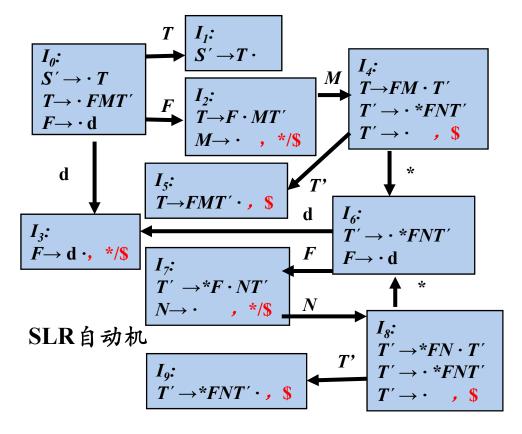
一系列分析过程





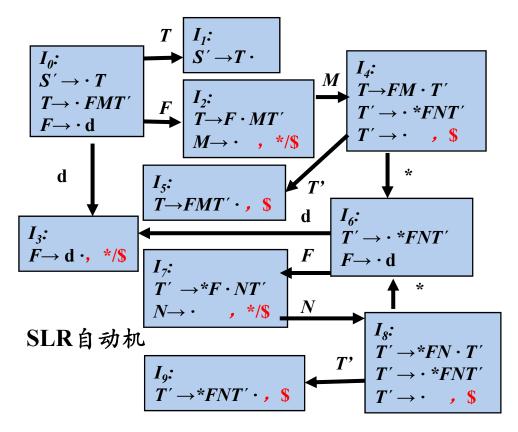
```
1) T \rightarrow F M T ' \{ T.val = T '.syn \}
M \rightarrow \varepsilon \{ M.i = F.val; M.s = M.i \}
2) T' \rightarrow *F N T_1' \{ T'.syn = T_1'.syn \}
N \rightarrow \varepsilon \{ N.i1 = T'.inh;
N.i2 = F.val;
N.s = N.i1 \times N.i2 \}
3) T' \rightarrow \varepsilon \{ T'.syn = T'.inh \}
4) F \rightarrow \text{digit } \{ F.val = \text{digit.} lexval \}
```

0	3
\$	d
	3



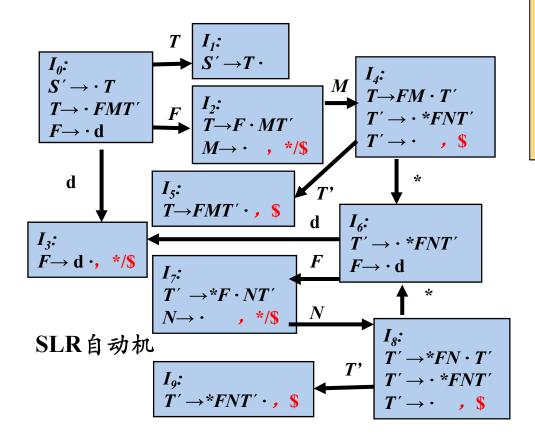
1) $T \rightarrow FMT' \{ T.val = T'.syn \}$ $M \rightarrow \varepsilon \{ M.i = F.val; M.s = M.i \}$ 2) $T' \rightarrow *FNT_1' \{ T'.syn = T_1'.syn \}$ $N \rightarrow \varepsilon \{ N.i1 = T'.inh;$ N.i2 = F.val; $N.s = N.i1 \times N.i2 \}$ 3) $T' \rightarrow \varepsilon \{ T'.syn = T'.inh \}$ 4) $F \rightarrow \text{digit } \{ F.val = \text{digit.} lexval \}$

0	2	4	6	3
\$	F	M	*	d
	3	T'inh=3		5

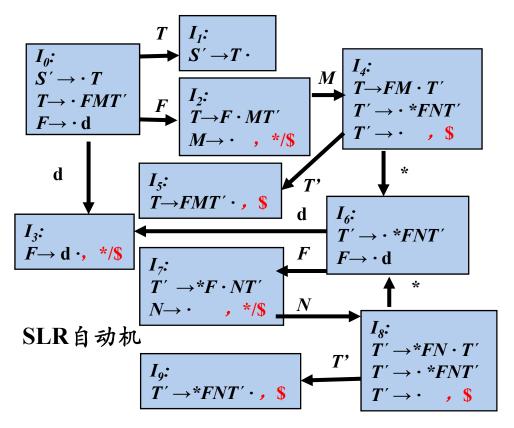


1) $T \rightarrow F M T ' \{ T.val = T '.syn \}$ $M \rightarrow \varepsilon \{ M.i = F.val; M.s = M.i \}$ 2) $T' \rightarrow *F N T_1' \{ T'.syn = T_1'.syn \}$ $N \rightarrow \varepsilon \{ N.i1 = T'.inh;$ N.i2 = F.val; $N.s = N.i1 \times N.i2 \}$ 3) $T' \rightarrow \varepsilon \{ T'.syn = T'.inh \}$ 4) $F \rightarrow \text{digit } \{ F.val = \text{digit.} lexval \}$

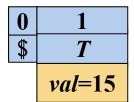
0	2	4	(6	7	8	9
\$	F	M	*		F	N	T'
	3	T'inh=3			5	T_1 'inh=15	<i>syn</i> =15
推断继承属性的位置							
`T'的预期位置							



0	2	4	5		
\$	F	M	T'		
	3	T'inh=3	<i>syn</i> =15		



1) $T \rightarrow FMT' \{ T.val = T'.syn \}$ $M \rightarrow \varepsilon \{ M.i = F.val; M.s = M.i \}$ 2) $T' \rightarrow *FNT_1' \{ T'.syn = T_1'.syn \}$ $N \rightarrow \varepsilon \{ N.i1 = T'.inh;$ N.i2 = F.val; $N.s = N.i1 \times N.i2 \}$ 3) $T' \rightarrow \varepsilon \{ T'.syn = T'.inh \}$ 4) $F \rightarrow \text{digit } \{ F.val = \text{digit.} lexval \}$



将语义动作改写为 可执行的栈操作

规约后T会处于F所在的位置,所以是top-2

- T→F M T' {stack[top-2]. val = stack[top].syn; top = top-2;}
 M→ε {stack[top+1]. T'inh = stack[top].val; top = top+1;}
 T'→*F N T₁' {stack[top-3]. syn = stack[top].syn; top = top-3;}
 - $N \rightarrow \varepsilon$ {stack[top+1]. T'inh = stack[top-2]. T'inh \times stack[top].val; top = top+1;}
 - 3) $T' \rightarrow \varepsilon \{ stack[top+1]. syn = stack[top]. T'inh; top = top+1; \}$
- 4) $F \rightarrow \text{digit } \{stack[top].val = stack[top]. lexval;\}$

给定一个以LL文法为基础的L-属性定义,可以修改这个文法,并在LR 语法分析过程中计算这个新文法之上的SDD

- ▶ 首先构造SDT,在各个非终结符之前放置语义动作来计算它的继承属性, 并在产生式后端放置语义动作计算综合属性
- \nearrow 对每个内嵌的语义动作,向文法中引入一个标记非终结符来替换它。每个这样的位置都有一个不同的标记,并且对于任意一个标记M都有一个产生式 $M \rightarrow \varepsilon$
- ightharpoons 如果标记非终结符M在某个产生式 $A
 ightharpoons lpha \{a\} eta$ 中替换了语义动作a,对a进行修改得到a',并且将a'关联到 $M
 ightharpoons \epsilon$ 上。动作a'
 - \triangleright (a) 将动作a需要的A或 α 中符号的任何属性作为M的继承属性进行复制
 - ► (b) 按照a中的方法计算各个属性,但是将计算得到的这些属性作为M的综合属性

