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

在递归的预测分析过程中进行翻译

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) $T \rightarrow F \{ T'.inh = F.val \} T'$
   $\{ T.val = T'.syn \}$
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, T1'inh, T1'syn;
  if token="*" then
  { Getnext(token);
    Fval=F(token);
    T1'inh= T'inh ×Fval;
    Getnext(token);
    T1'syn=T1'(token, T1'inh);
    T'syn=T1'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) $F \rightarrow$ digit $\{ F.val =$ digit.*lexval* $\}$

```
Tval T(token)
{
    D:  Fval, T'inh, T'syn;
    Fval = F(token);
     T'inh = Fval;
    Getnext(token);
    T'syn = T1'(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) $F \rightarrow$ digit $\{ F.val =$ digit$.lexval \}$

```
Fval F(token)
{
   if token ≠ 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) $F \rightarrow$ digit $\{ F.val =$ digit.*lexval* $\}$

```
Desent()
{
    D：  Tval;
    Getnext(token);
    Tval = T(token);

    if token ≠ "$" then Error;
    return ;
}
```

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
  - 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$c := B(b_1, b_2, \ldots, b_k)$，其中，$b_1, b_2, \ldots, b_k$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

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

第五章 语法制导翻译

L-属性定义的自底向上翻译

哈尔滨工业大学 陈鄞

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*

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

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

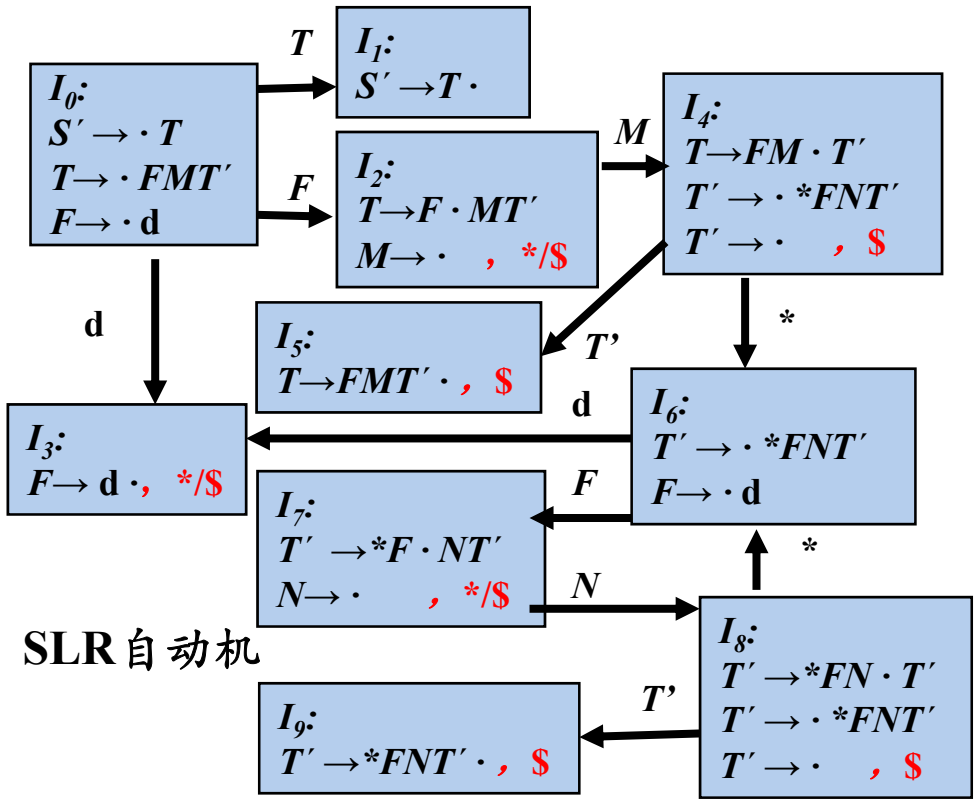
4) $F \rightarrow$ digit { $F.val$ = digit.$lexval$ }

标记非终结符
(*Marker Nonterminal*)

修改后的*SDT*，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？

例



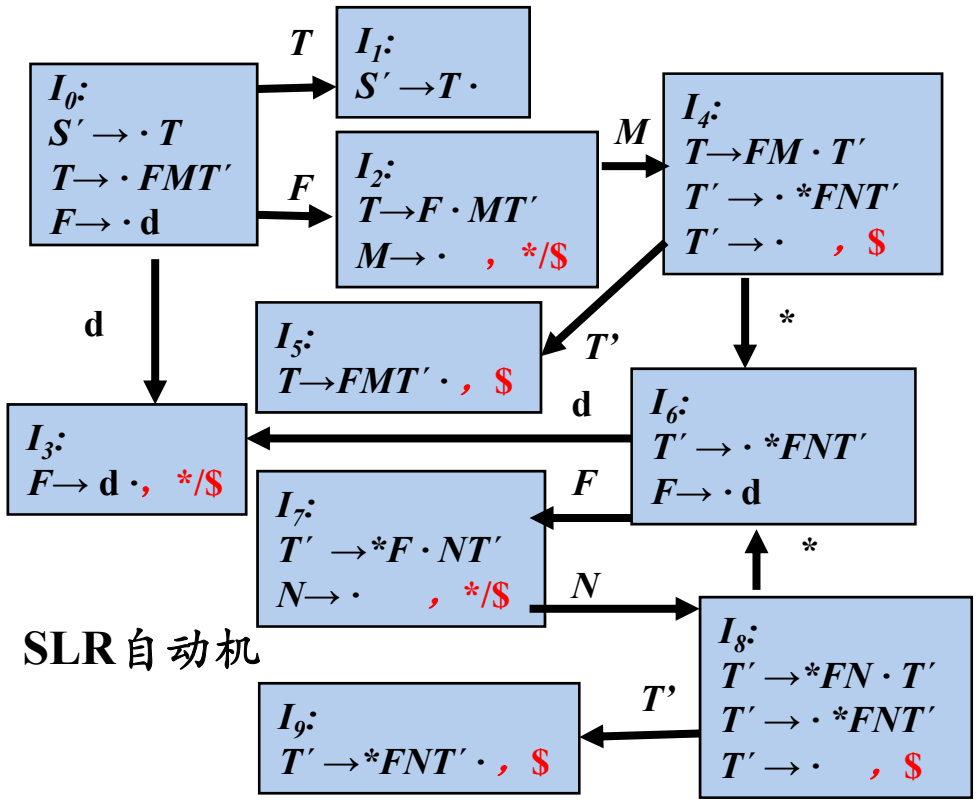
SLR自动机

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

输入： 3 * 5

0	3
$	d
	3

例



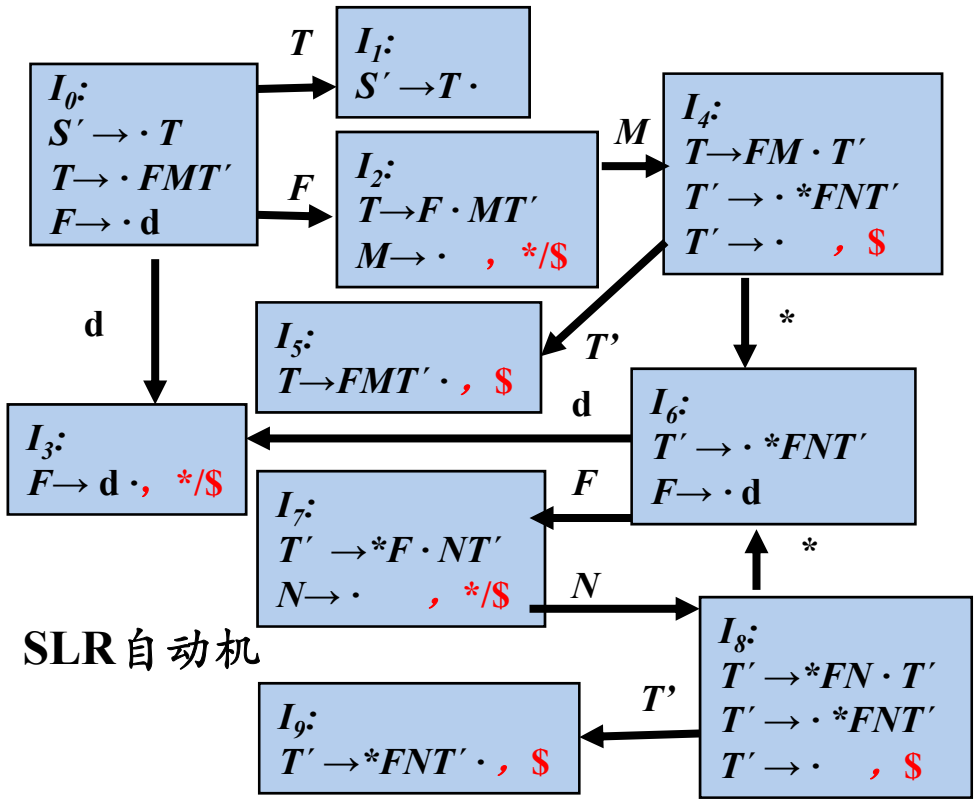
SLR自动机

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

输入： 3 * 5

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

例



SLR自动机

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

   $M \rightarrow \varepsilon$ {$M.i = F.val$; $M.s = M.i$ }

2) $T' \rightarrow *F\ N\ T_1'$ { $T'.syn = T_1'.syn$ }

   $N \rightarrow \varepsilon$ {$N.i1 = T'.inh$;

   $N.i2 = F.val$;

   $N.s = N.i1 \times N.i2$ }

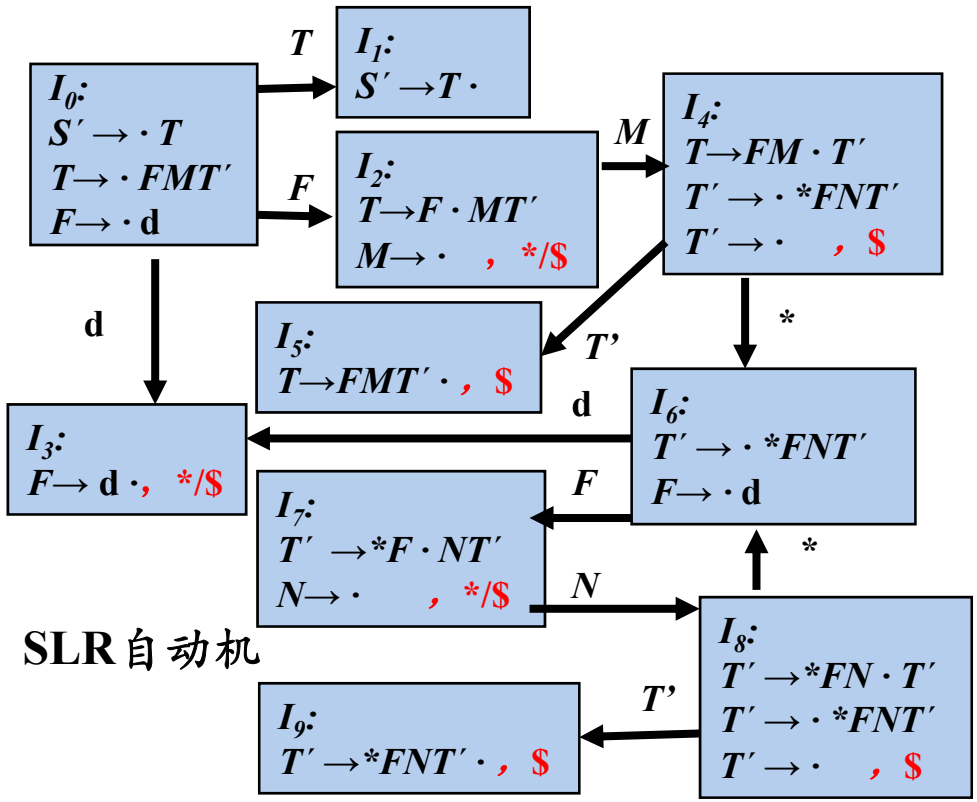
3) $T' \rightarrow \varepsilon$ { $T'.syn = T'.inh$ }

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

输入： 3 * 5

0	2	4	6	7	8	9
$	F	M	*	F	N	T'
	3	T'inh=3		5	T_1'inh=15	syn=15

例



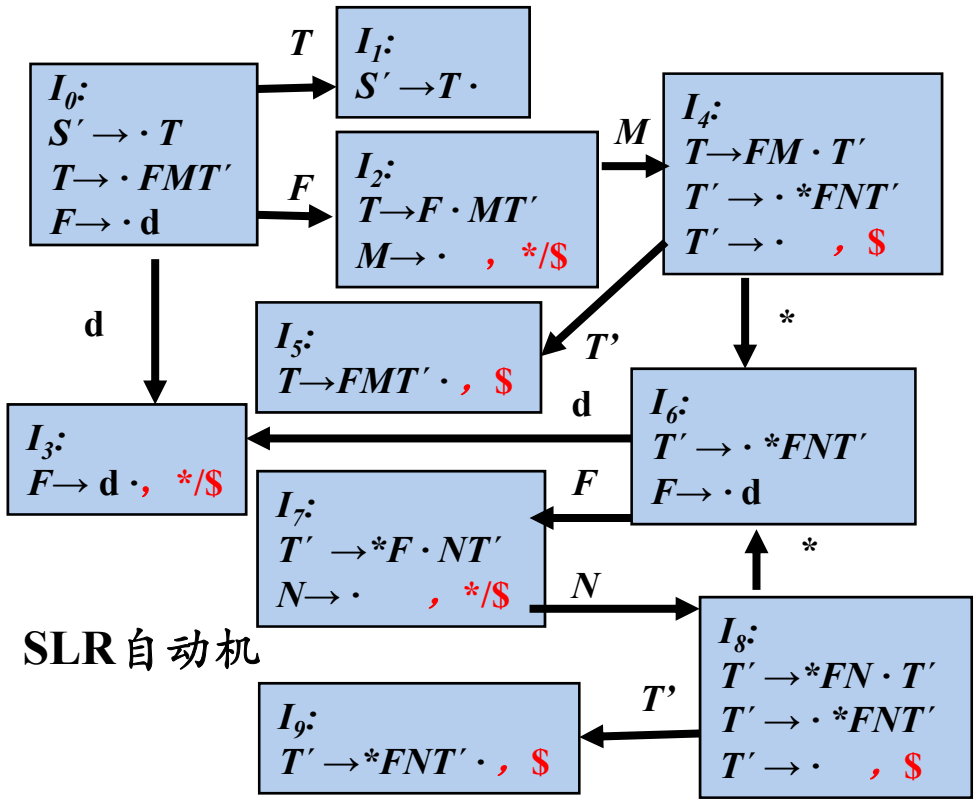
SLR自动机

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

输入： 3 * 5

0	2	4	5
$	F	M	T'
	3	T'inh=3	syn=15

例



SLR自动机

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

输入： 3 * 5

0	1
$	T
	val=15

将语义动作改写为可执行的栈操作

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

1) $T \rightarrow F\ M\ T'$ {$stack[top-2].\ val = stack[top].syn$; $top = top-2$;}

   $M \rightarrow \varepsilon$ {$stack[top+1].\ T'inh = stack[top].val$; $top = top+1$;}

2) $T' \rightarrow *\ F\ N\ T_1'$ {$stack[top-3].\ syn = stack[top].syn$; $top = top-3$;}

   $N \rightarrow \varepsilon$ {$stack[top+1].\ T'inh = stack[top-2].\ T'inh \times stack[top].val$; $top = top+1$;}

   3) $T' \rightarrow \varepsilon$ {$stack[top+1].syn = stack[top].\ T'inh$; $top = top+1$;}

4) $F \rightarrow$ 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
- 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$M \rightarrow \varepsilon$
- 如果标记非终结符*M*在某个产生式$A \rightarrow \alpha\{a\}\beta$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$M \rightarrow \varepsilon$ 上。动作a'
  - (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

第五章 语法制导翻译

L-属性定义的自底向上翻译

哈尔滨工业大学 陈鄞