第7章语法制导的语义计算

- 主讲：王慧娇
- 办公室：金鸡岭3301-1
- 电话：13978321977
- QQ：248886622
- Email：whj7667@qq.com

- 答疑地点：5507
- 辅导时间：周三1、2节

主要内容

- 语法制导翻译概述
- 属性文法
  - 综合属性与继承属性
  - S_属性文法
  - L_属性文法
- 语义计算方法
- 翻译模式

语法制导翻译概述

- 语法制导翻译
  - 在进行语法分析的同时，完成相应的语义处理
  - $E \rightarrow E_1 + E_2$ {E.val:=E_1.val+E_2.val}
- 在产生式中嵌入语义动作的程序片段
- 语义动作的程序片段：根据被识别出的语法成分进行语义处理，有哪些工作要做
- 何时执行语义处理：归约或推导的时候

语法制导翻译概述

- 典型处理方法
- 对应每一个产生式编制一个语义子程序，当一个产生式获得匹配时，调用相应的语义子程序实现语义检查与翻译
  - $E \rightarrow E_1 + T$ {E.val:=E_1.val+T.val}
  - $T \rightarrow T_1 * F$ {T.val:=T_1.val*F.val}
  - $F \rightarrow id$ {F.val:=id.val}

语法制导翻译概述

语法制导举例

在一个移进-归约分析过程中采用以下的语法制导翻译模式

S→aB {print “0”；}

S→c {print “1”；}

B→Sb {print “2”；}

当分析器的输入为acb时，打印的字符串是什么？

7.1 基于属性文法的语义计算

- 属性文法是Knuth在1968年提出的
- 属性文法的特点
  - 是一种接近形式化的语义描述方法
  - 长于描述静态语义、短于描述动态语义
  - 每个语法符号有相应的属性符号
  - 每个产生式有相应的计算属性的规则
    - 属性变量:=属性表达式

1. 属性文法的定义

- 三元组：$A=(G, V, F)$
  - G 是上下文无关文法
  - V 属性的有穷集
  - F 关于属性的计算规则
- 属性及其计算规则
  - 语义信息作为终结符和非终结符的属性
  - 语义分析为产生式相关的属性计算：每个产生式设置语义规则，描述各属性的关系——计算规则

2. 属性文法的相关概念

- 属性（Attribute）可用来刻画一个文法符号的任何我们所关心的特性，如：符号的值，符号的名字串，符号的类型，符号的偏移地址，等等…
- 记号

  文法符号 X 关联属性 a 的属性值可通过 X.a 访问

2. 属性文法的相关概念

- 语义规则（Semantic Rule）

在属性文法中，每个产生式 $A \rightarrow \alpha$ 都关联一个语义规则的集合，用于描述如何计算当前产生式中文法符号的属性值或附加的语义动作

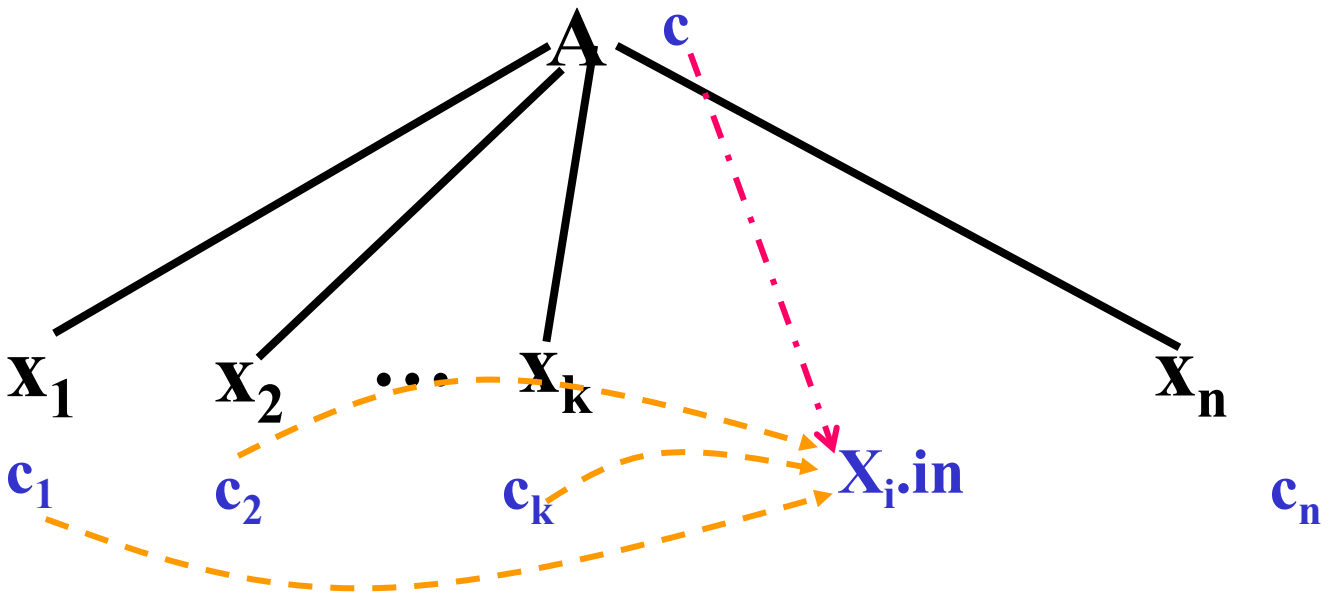
- 属性文法中允许如下语义规则
  - 复写（copy）规则，形如 X.a := Y.b
  - 基于语义函数（semantic function）的规则，形如
    $b:=f(c_1, c_2, \cdots, c_k)$ 或 $f(c_1, c_2, \cdots, c_k)$
    其中，$b, c_1, c_2, \cdots, c_k$ 是该产生式中文法符号的属性
- 实践中，语义函数的形式可以更灵活

3. 属性分类——继承属性

- 设$A \rightarrow X_1X_2\ldots X_n$为一个产生式，$X_i$的属性

  $X_i.in=f(c,c_1,c_2,\ldots,c_{i-1})$ $c,c_1,c_2,\ldots,c_{i-1}$是A, $X_1\ldots$， X_{i-1}的属性

- 这种属性叫做继承(Inherited)属性



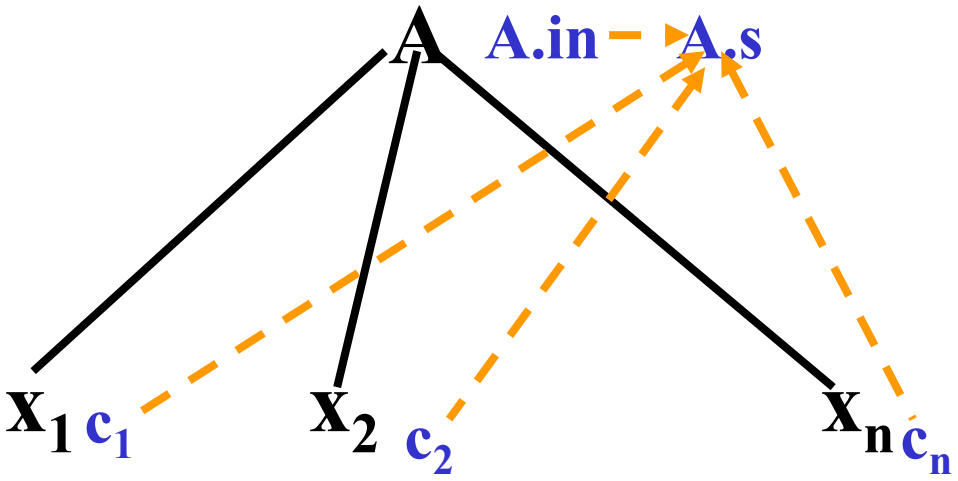
3. 属性分类——综合属性

- 设 $A \rightarrow X_1X_2 \ldots X_n$ 为一个产生式
  - $A.s=f(c_1,c_2,\ldots,c_k)$
  - $c_1,c_2,\ldots,c_k$是X_1，X_2，…，X_n的属性和A的继承属性
  - A.s是根据其子结点的属性值计算出来的
- 这种属性叫做综合(Synthesized)属性

3. 属性分类——综合属性

$A \rightarrow X_1 X_2 \ldots X_n$

$A.s = f(c_1, c_2, \ldots, c_k)$



3. 属性分类——固有属性

- 语言中的标识符、常数（数值的、符号的）、常量，它们的属性是用户给定的、不变的

T → int　　T.type := ‘integer’

固有(Inherent)属性（单词属性）

归类于综合属性

✧ 属性文法举例

– 仅含综合属性的例子（开始符号S）

产生式	语义动作
$S \to E$	{ print(E.val) }
$E \to E_1 + T$	{ E.val := E_1.val + T.val }
$E \to T$	{ E.val := T.val }
$T \to T_1 * F$	{ T.val := T_1.val × F.val }
$T \to F$	{ T.val := F.val }
$F \to (E)$	{ F.val := E.val }
$F \to d$	{ F.val := d.lexval }

注：d.lexval 是词法分析程序确定的固有属性值

✧ 属性文法举例

– 含继承属性的例子（开始符号S）

产生式	语义动作
$S \rightarrow ABC$	{B.in_num := A .num; C.in_num := A .num; if (B.num=0 and (C.num=0) then print(“Accepted!”) else print(“Refused!”) }
$A \rightarrow A_1a$	{ A.num := A_1.num + 1 }
$A \rightarrow \varepsilon$	{ A.num := 0 }
$B \rightarrow B_1b$	{ B_1.in_num := B.in_num; B.num := B_1.num-1 }
$B \rightarrow \varepsilon$	{ B.num := B.in_num }
$C \rightarrow C_1c$	{ C_1.in_num := C.in_num; C.num := C_1.num-1 }
$C \rightarrow \varepsilon$	{ C.num := C.in_num }

其中，A .num，B.num 和 C.num 是综合属性值，而 B.in_num 和 C.in_num 是继承属性值

4.基于属性文法的语义计算

- 基于属性文法的语义计算
  - 树遍历方法

    通过遍历分析树进行属性计算

  - 单遍的方法

    语法分析遍的同时进行属性计算

4.基于属性文法的语义计算

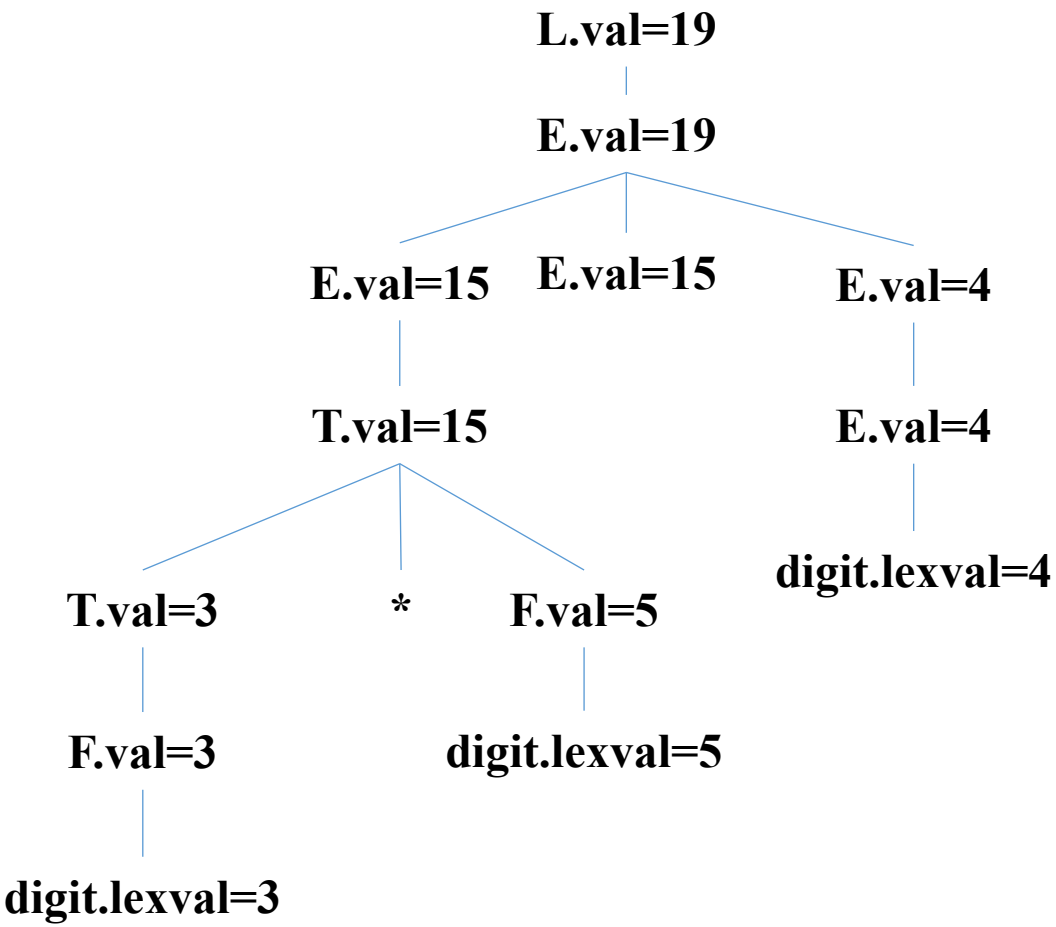
✧ 单遍的方法

– 语法分析遍的同时进行属性计算
  - 自下而上方法
  - 自上而下方法
– 只适用于特定文法

  本课程只讨论如下两类属性文法：
  - S-属性文法
  - L-属性文法

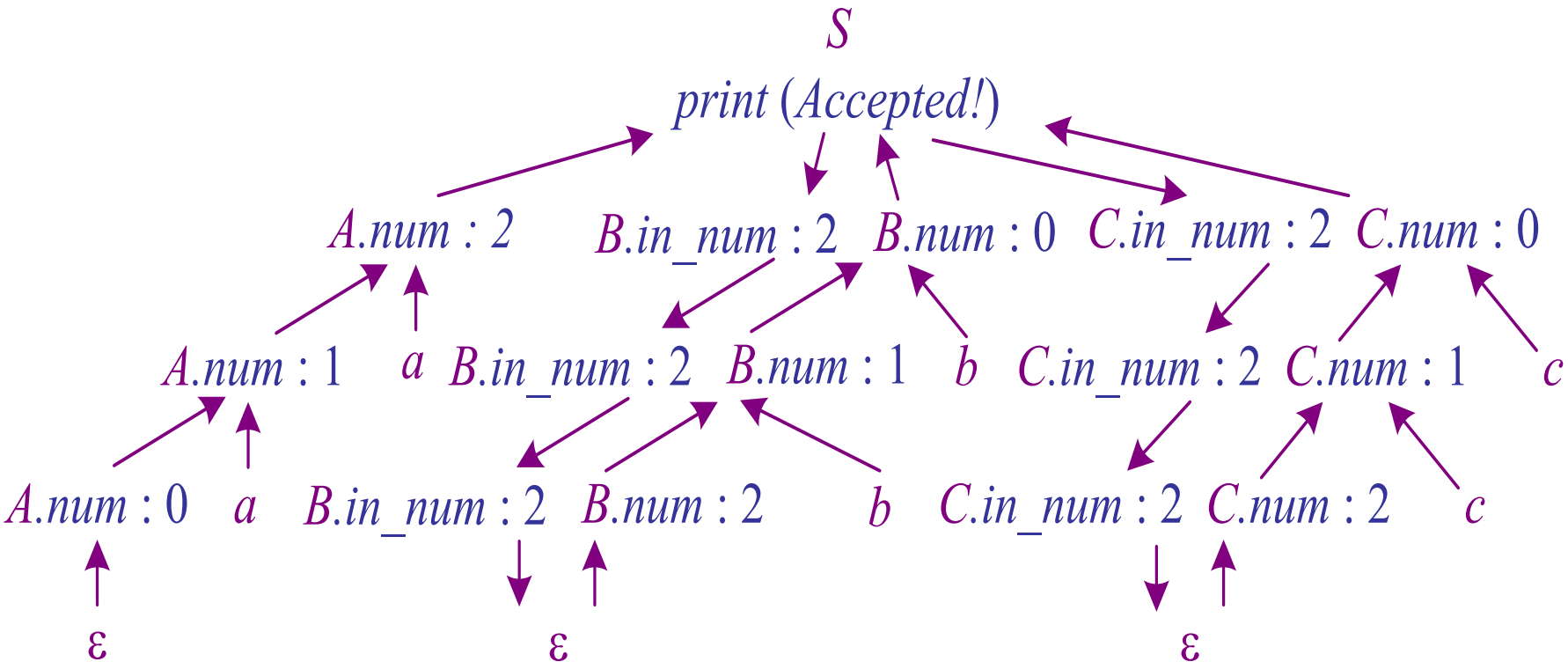
注释语法分析树：显示各个属性值的语法树

例：i*i+i的注释语法树(3*5+4)



4.基于属性文法的语义计算

- 带标注语法树



5. S-属性定义和L-属性定义

- S-属性定义：仅包含综合属性的属性文法（一种语法制导定义）
- S-attributed Definition
- S-attributed Grammar
- 如：算术表达式求值的属性文法

例 算术表达式的属性文法

$L \rightarrow E$	{L.val= E.val }
$E \rightarrow E_1 + T$	{E.val := E_1.val + T.val}
$E \rightarrow T$	{E.val := T.val}
$T \rightarrow T_1 * F$	{T.val := T_1.val * F.val}
$T \rightarrow F$	{T.val := F.val}
$F \rightarrow (E)$	{F.val := E.val}
$F \rightarrow digit$	{F.val := digit.lexval}

lexval 是单词 digit 的属性

5. S-属性定义和L-属性定义

- L属性定义和L属性文法：包含综合属性和继承属性的属性文法（语法制导定义）
- L-attributed Definition
- L-attributed Grammar
- 如：算术表达式求值的属性文法、说明语句的属性文法

例：建立说明语句的翻译方案

语法制导定义：

$D \rightarrow T\ L$	L.in := T.type
$T \rightarrow int$	T.type := integer
$T \rightarrow real$	T.type := real
$L \rightarrow L_1$,id	L_1.in := L.in
	addtype(id.entry, L.in)
$L \rightarrow id$	addtype(id.entry, L.in)

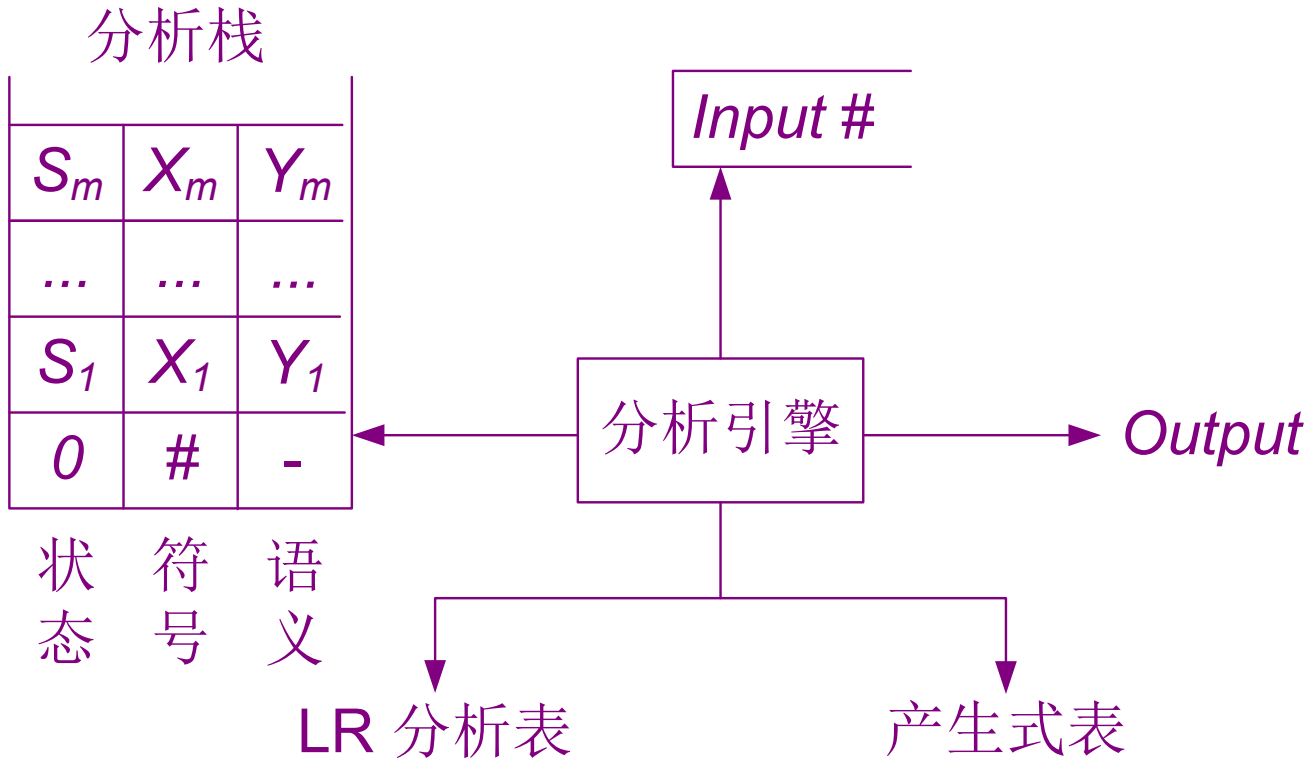
上述动作的执行时机分析(依照自顶向下分析)

6.5-属性文法的语义计算

- 通常采用自下而上的方式进行
- 以语法分析采用LR分析技术为例，可以通过扩充分析栈中的域，形成语义栈来存放综合属性的值，计算相应产生式左部文法符号的综合属性值刚好发生在每一步归约之前的时刻

6. S-属性文法的语义计算

- 采用LR分析技术进行S-属性文法的语义计算
  - 扩充分析栈中的域形成语义栈存放综合属性的值



6. S-属性文法的语义计算

- 采用LR分析技术进行S-属性文法的语义计算
  - 语义动作中的综合属性可以通过存在于当前语义栈栈顶部分的属性进行计算
  - 例如，假设有相应于产生式 A→XYZ 的语义规则

    A.a := f (X.x, Y.y, Z.z)

    在 XYZ 归约为 A 之前，Z.z, Y.y, 和 X.x 分别存放于语义栈的 top，top-1 和 top-2 的相应域中，因此 A.a 可以顺利求出。归约后，X.x, Y.y, Z.z 被弹出，而在栈顶 top 的位置上存放 A.a。

6. S-属性文法的语义计算

- 用LR分析技术进行S-属性文法的语义计算举例

产生式	语义动作
$S \to E$	{ print(E.val) }
$E \to E_1 + T$	{ E.val := E_1.val + T.val }
$E \to T$	{ E.val := T.val }
$T \to T_1 * F$	{ T.val := T_1.val × F.val }
$T \to F$	{ T.val := F.val }
$F \to (E)$	{ F.val := E.val }
$F \to d$	{ F.val := d.lexval }

6.5-属性文法的语义计算

✧ 文法G' [S] 的LR分析表

（0）S→E （1）E→E+T （2） E→T
（3）T→T*F （4） T→F
（5）F→(E) （6）F→d

状态	ACTION						GOTO		
	d	*	+	(	)	#	E	T	F
0	s5			s4			1	2	3
1			s6			acc			
2		s7	r2		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8			s6		s11				
9		s7	r1		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

6. S-属性文法的语义计算

- LR分析过程伴随常量表达式2 + 3 * 5的求值

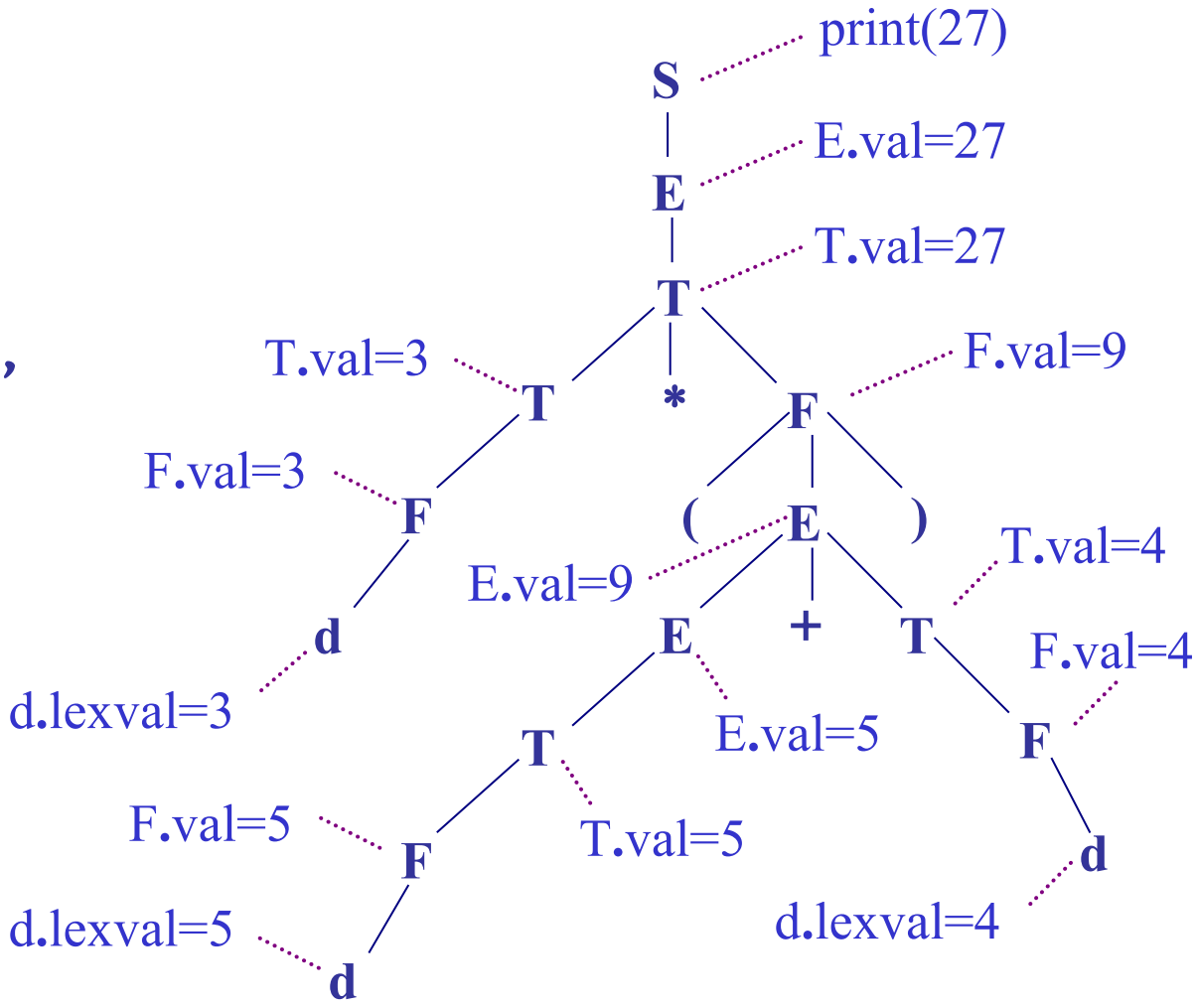
（0）S→E （1）E→ E+T （2） E → T
（3）T →T*F （4） T → F
（5）F → (E) （6）F → d

分析栈（状态，符号，语义值）	余留输入串	动作	语义动作
0 # –	2 + 3 * 5 #	s5	
0 # – 5 2 2	+ 3 * 5 #	r6	F.val := d.lexval
0 # – 3 F 2	+ 3 * 5 #	r4	T.val := F.val
0 # – 2 T 2	+ 3 * 5 #	r2	E.val := T.val
0 # – 1 E 2	+ 3 * 5 #	s6	
0 # – 1 E 2 6 + –	3 * 5 #	s5	
0 # – 1 E 2 6 + – 5 3 3	* 5 #	r6	F.val := d.lexval
0 # – 1 E 2 6 + – 3 F 3	* 5 #	r4	T.val := F.val
0 # – 1 E 2 6 + – 9 T 3	* 5 #	s7	
0 # – 1 E 2 6 + – 9 T 3 7 * –	5 #	s5	
0 # – 1 E 2 6 + – 9 T 3 7 * – 5 5 5	#	r6	F.val := d.lexval
0 # – 1 E 2 6 + – 9 T 3 7 * – 10 F 5	#	r3	$T.val:=T_1.val \times F.val$
0 # – 1 E 2 6 + – 9 T 15	#	r1	$E.val:=E_1.val+T.val$
0 # – 1 E 17	#	acc	print(E.val)

6.5-属性文法的语义计算

✧ 综合属性代表自下而上传递的信息

- 接上页的例子，对表达式 3 * (5+4) 的分析树进行自下而上（后序）遍历，并执行相应的语义规则，得到该表达式的一种求值过程



7. L-属性文法的语义计算

- 采用自上而下基于深度优先后序遍历的算法

```
procedure dfvisit(n: node);
  begin
    for n 的每一孩子m, 从左到右 do
        begin
            计算 m 的继承属性值;
            dfvisit(m)
        end;
        计算n的综合属性值
  end
```

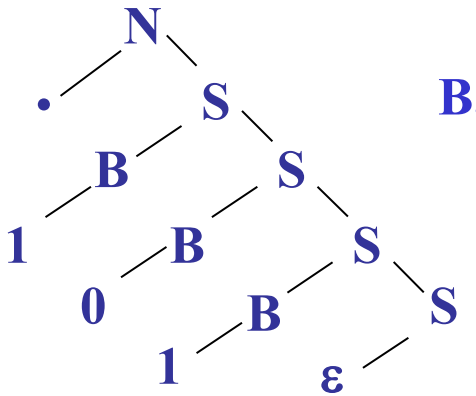
- 该算法与自上而下预测分析过程对应. 因此,基于 LL(1) 文法的 L-属性文法可以采用这种方法进行语义计算.

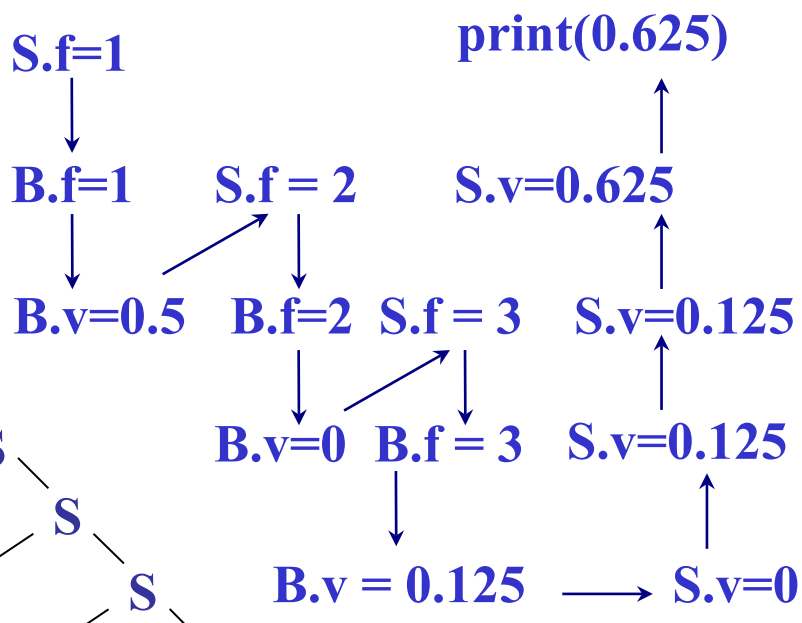
7. L-属性文法的语义计算

- 采用基于深度优先后序遍历算法进行L-属性文法的语义计算举例

– 考虑对于下列L-属性文法，输入串为 .101 时的计算过程

产生式	语义动作
$N \to .S$	{ S.f : =1； print(S.v) }
$S \to BS_1$	{ S_1.f := S.f+1; B.f : =S.f; S.v := S_1.v+B.v }
$S \to \varepsilon$	{ S.v := 0 }
$B \to 0$	{ B.v := 0 }
$B \to 1$	{ B.v := $2^{-B.f}$ }

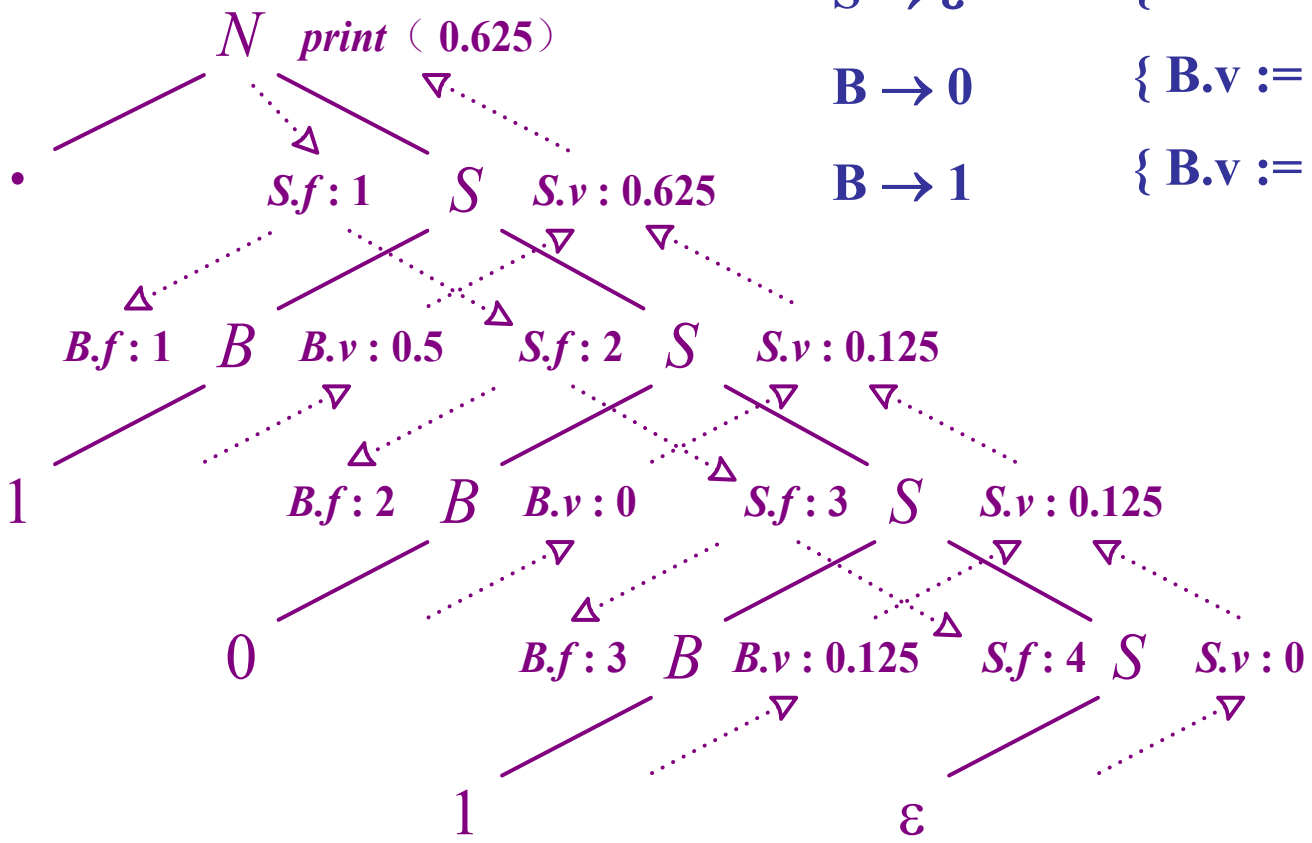




7.L-属性文法的语义计算

✧ 接上页例子

$N \rightarrow .S$	{ S.f : =1； print(S.v) }
$S \rightarrow BS_1$	{ S_1.f := S.f+1; B.f : =S.f; S.v := S_1.v+B.v }
$S \rightarrow \varepsilon$	{ S.v := 0 }
$B \rightarrow 0$	{ B.v := 0 }
$B \rightarrow 1$	{ B.v := $2^{-B.f}$ }



7.2 翻译模式(Translation Scheme

- 将语义动作插入到产生式的某个位置
- 属性翻译文法（属性文法+语义动作+语义动作属性）
- S-属性文法+波兰翻译模式
  - 规定：将语义动作放在产生式的最右面
  - 规定：所有的属性计算都是综合属性
- 表示形式：
  - Expr→ Expr '+' Expr {$$.value=$1.value+$3.value;}
  - $$:表示产生式左侧符号
  - $1，$2…:表示产生式右侧第一个符号、第二个符号

7.2 翻译模式

- L-翻译模式
  - 对于既包含继承属性又包含综合属性的情形，但需要满足：
  - （1）产生式右端某个符号继承属性的计算必须位于该符号之前，其语义动作不访问位于它右边符号的属性，只依赖于该符号左边符号的属性（对于产生式左部的符号，只能是继承属性）；
  - （2）产生式左部非终结符的综合属性的计算只能在所用到的属性都已计算出来之后进行，通常将相应的语义动作置于产生式的尾部。

7.2 翻译模式

✧ 翻译模式举例

– 定点二进制小数转换为十进制小数

$N \rightarrow . \{S.f := 1\}\ S\ \{print(S.v)\}$

$S \rightarrow \{B.f := S.f\}\ B\ \{S_1.f := S.f + 1\}\ S_1\ \{S.v := S_1.v + B.v\}$

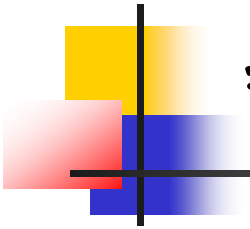
$S \rightarrow \varepsilon\ \{S.v := 0\}$

$B \rightarrow 0\ \{B.v := 0\}$

$B \rightarrow 1\ \{B.v := 2^{-B.f}\}$

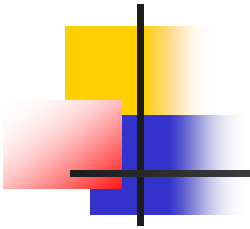
7. 翻译模式(Translation Scheme)

```
Program→Expr
  {printf(“  %d\n”,$1.value);}
Expr→ Expr ’+’ Expr
        {$$.value=$1.value+$3.value;}
       | Expr ‘*’ Expr
         {$$.value=$1.value*$3.value;}
       | ‘(’ Expr ‘)’ {$$.value=$2.value;}
       |Number
       { $$=ToIntValue($1.yytext);}
```



翻译模式的特点

- 语义动作被嵌入到产生式右部的适当位置，在推导过程中完成语义处理



小结

- 语法制导翻译
- 属性文法
- 综合属性与继承属性
- S-属性文法
- L-属性文法
- 语义计算方法
- 翻译模式
- 语法分析器自动生成工具yacc