

华中科技大学

编译原理

第7章 语法制导的语义计算

编译原理课程组

内容摘要

本章研究语义分析基本原理和方法，主要介绍属性文法中各类属性的计算方法。在语法分析过程的制导下，如何进行静态语义分析的各种相应的技术。

重点讲解

7.1 基于属性文法的语义计算

7.2 基于翻译模式的语义计算

7.3 语法分析自动生成工具YACC

7.1 基于属性文法的语义计算

在语言形式化研究领域，基于形式语言和自动机理论，可以较好地解决计算机高级语言语法的形式化描述。关于语言语义形式化问题，已经推出了诸如操作语义学、公理语义学和指称语义学等理论。但是，目前这些理论与实际应用尚有较大距离。

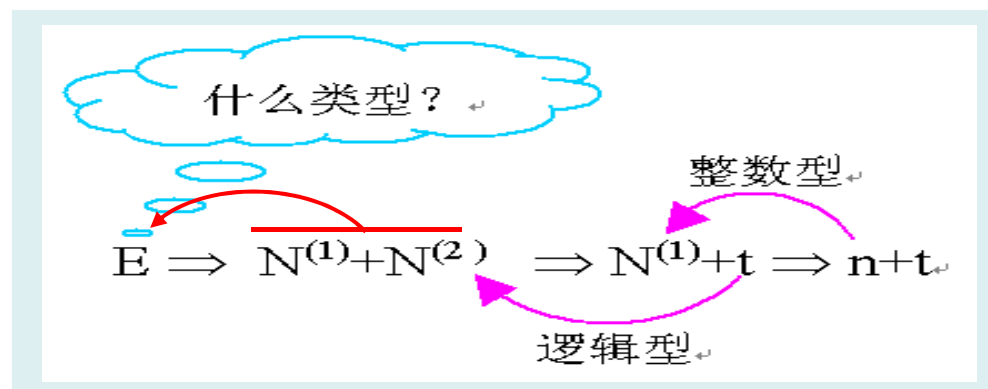
现实的编译程序通常采用一种在语法制导下的语义分析和代码生成方法。所谓“**语法制导**”是指伴随着语法分析过程，在语法分析每步推导或归约时刻，增加相应的语义分析和代码生成处理之含义。这种方法，可以采用一种非形式化、但接近形式化的‘**属性文法**’，作为语言语义的描述工具。

7.1.1 属性文法

例7.1 设文法 $G[E]$ 定义如下，试解释其语义。

- $$G[E]: \begin{array}{l} (1) E \rightarrow N^{(1)} + N^{(2)} \\ (2) E \rightarrow N^{(1)} \text{ or } N^{(2)} \\ (3) N \rightarrow n \\ (4) N \rightarrow t \\ (5) N \rightarrow f \end{array}$$

考察输入串 $n+t$ ，因为存在如下推导过程，所以输入串 $n+t$ 是文法 $G[E]$ 的一个句子。



最后一步归约时，可以发现其语义错误！

如果将数据值和数据类型两类语义称为文法符的属性，并分别命名为 **value**和**type**属性，任意文法符**X**的数据值和数据类型两类语义分别记为 **X.value**和**X.type**，则每个规则的语义要求就可以描述成形式化的断言或谓词形式如下，并称为语义规则。其中，**int**和**bool**分别表示整数型和逻辑型，**@(a)**表示词法分析提供的**a**之数据值。

G[E]:

- (1) **E** → **N**¹ + **N**² { **N**¹.type = int and **N**².type = int,
E.value ← **N**¹.value + **N**².value,
E.type ← **N**¹.type }
- (2) **E** → **N** ⁽¹⁾ or **N** ⁽²⁾ { **N**¹.type = bool and **N**².type =
bool, E.value ← **N**¹.value or
N².value, E.type ← **N**¹.type }
- (3) **N** → **n** { **N**.type ← int, **N**.value ← **@(n)** }
- (4) **N** → **t** { **N**.type ← bool, **N**.value ← **@(t)** }
- (5) **N** → **f** { **N**.type ← bool, **N**.value ← **@(f)** }

属性文法的定义

文法符(对应于自然语言的语法成分)的语义, 可以通过引入属性来进行描述, 称为**文法符的属性**。如: “数据类型”、“数据值”和“存储地址”等等属性来描述。

每个规则的语义可以使用有关属性的断言或谓词形式加以描述。即为文法的每个规则配备计算属性的计算规则, 称为语义规则。

在文法基础上扩充属性和属性的断言, 就是所谓的“属性文法”近似描述语言语义的基本思路。

定义8.1 一个**属性文法**A定义为一个三元组 (G, V, F) , 记为 $A = (G, V, F)$ 。其中, G为文法, V为**文法符号属性集**, F为规则的**有关属性的断言或谓词集**也称为**语义规则集**。

综合属性与继承属性

属性可以分为两类：继承属性和综合属性。**继承属性**是指其属性值是自顶向下传递所得的属性。**综合属性**是指其属性值是自底向上传递所得的属性。

对每个产生式 $A \rightarrow \alpha$ ，都有一套与之相关联的语义规则，其形式为：

$b = f(c_1, c_2, \dots, c_k)$ ，其中 b, c_1, c_2, \dots, c_k 是该产生式文法符号的属性。

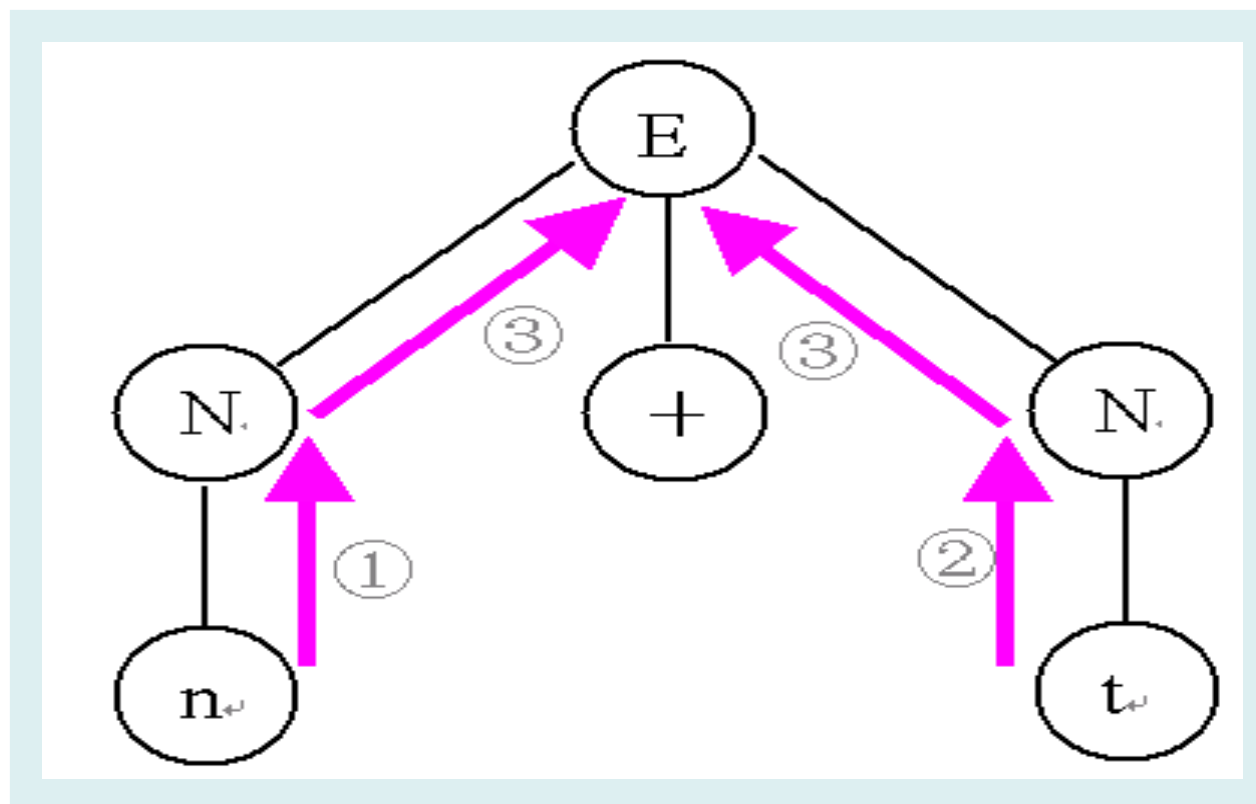
(1) 如果 b 是 A 的一个属性， c_1, c_2, \dots, c_k 是该产生式右部文法符号的属性。则称 b 是 A 的**综合属性**。

(2) 如果 b 是产生式右部某个文法符号 X 的属性， c_1, c_2, \dots, c_k 是 A 或产生式右部任何文法符号的属性。则称 b 是 X 的**继承属性**。

注：(1) 非终结符既可以有综合属性，也可以有继承属性。但文法开始符号没有继承属性。

(2) 终结符只有综合属性，它们是由词法程序提供的。

例7.1文法G[E]语义规则，实际上，是在假定归约分析前提下设计的。涉及到的value，和type属性都是综合属性。输入串n+t的属性值传递过程如下图所示。



例7.2 设说明语句文法G[D]定义如下，其中int、real 和id是3个终结符。试设计描述数据类型的语义规则。

G[D]: (1) $D \rightarrow TL$
(2) $T \rightarrow \text{int}$
(3) $T \rightarrow \text{real}$
(4) $L \rightarrow L^1, \text{id}$
(5) $L \rightarrow \text{id}$

在文法G[D]中，int、real和id为3个终结符号，int和real分别是整型和实型的数据类型名称，id表示运算对象。实际上，文法G[D]描述的说明语句一般形式为

$\text{real } id_1, id_2, \dots, id_n$ 或 $\text{int } id_1, id_2, \dots, id_n$ 。

设计两个属性type和in，均表示数据类型的语义，
enter(id, ...)是将符号id及其相关信息登记到符号表中的语义处理
子程序。文法G[D] 数据类型的语义规则设计如下。

```
G[D]: (1) D → TL      { L.in ← T.type }  
      (2) T → int     { T.type ← integer }  
      (3) T → real    { T.type ← real }  
      (4) L → L1, id { L1.in ← L.in ; enter(id, L.in ) }  
      (5) L → id      { enter(id, L.in ) }
```

规则(2)和(3)的语义规则中integer 和real表示数据类型整型和实型的内部名称。属性type是综合属性，属性in是继承属性。输入串int a, b的属性值传递过程如下图所示，其中天蓝色和粉红色分别表示属性type和in的传递线路。

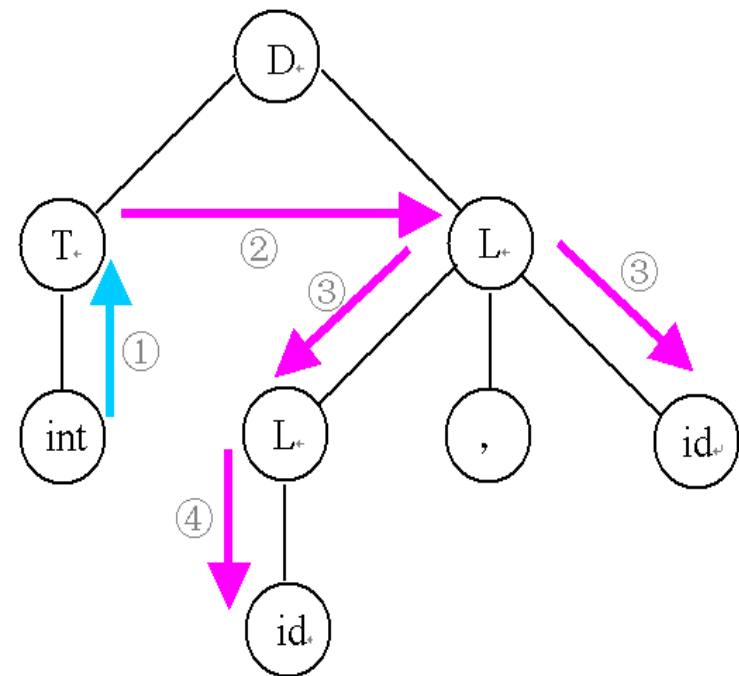
$G[D]: (1) D \rightarrow TL \quad \{ L.in \leftarrow T.type \}$

$(2) T \rightarrow int \quad \{ T.type \leftarrow integer \}$

$(3) T \rightarrow real \quad \{ T.type \leftarrow real \}$

$(4) L \rightarrow L^1, id \quad \{ L^1.in \leftarrow L.in ; enter(id, L.in) \}$

$(5) L \rightarrow id \quad \{ enter(id, L.in) \}$



7.1.2 S-属性文法与L-属性文法

一个一般性的属性文法的翻译器可能是很难建立，但对于L-属性文法的翻译器却很好建立。

S-属性文法：只含有综合属性的属性文法。

L-属性文法：如果对每个产生式 $A \rightarrow X_1 X_2 \cdots X_n$, 其每个语义规则中的每个属性或者为综合属性，或者是 $X_j (1 \leq j \leq n)$ 的一个继承属性且这个继承属性仅依赖于：

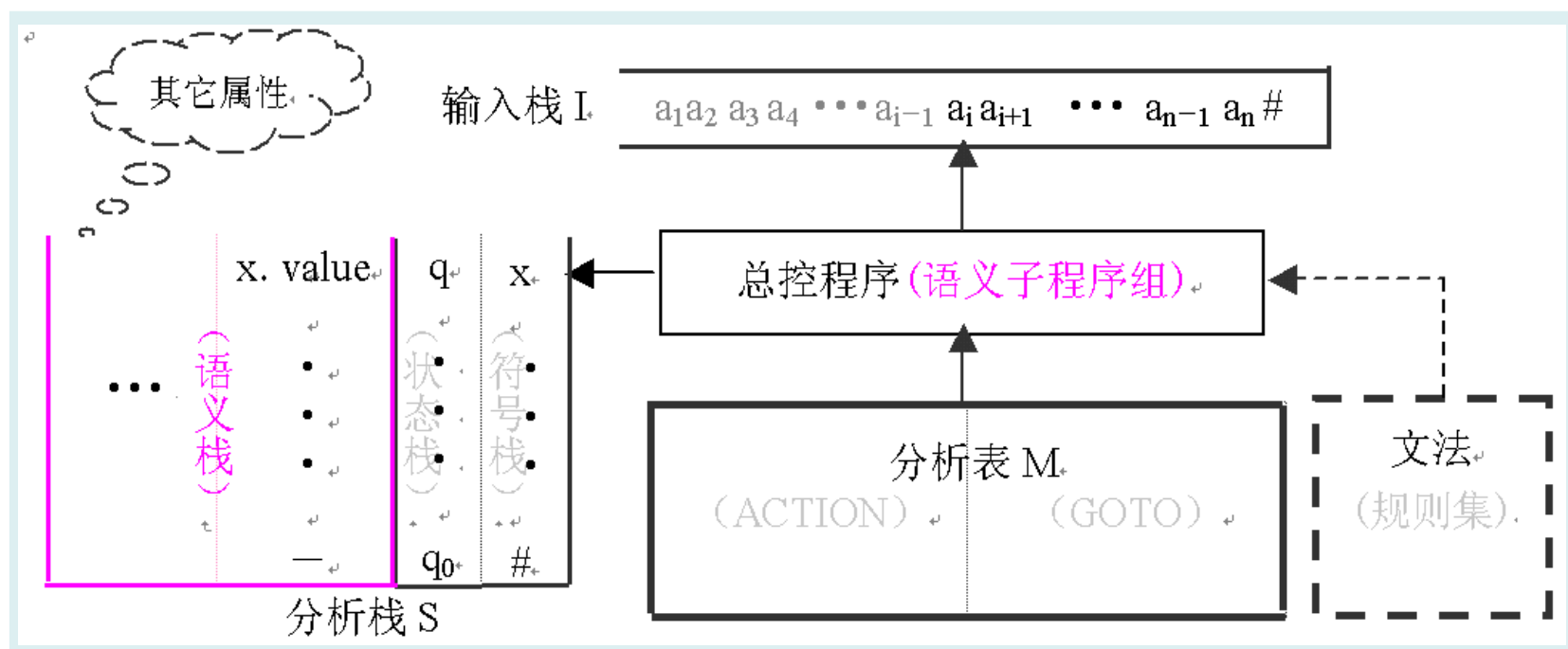
(1) 产生式 X_j 在左边的 $X_1 X_2 \cdots X_{j-1}$

(2) A 的继承属性

S-属性文法是L-属性文法的一个特例。

7.1.3 基于S-属性文法的语义计算

自底向上语法制导翻译的实现总体框架是基于自底向上语法分析程序总体框架基础上，(1)增加一个与语法分析栈同步的语义栈，存放与分析栈中文法符对应属性值，(2)依据每个语法规则的语义规则、语义信息处理和代码生成处理，编写成独立的语义处理子程序，待语法规则用于归约时刻调用。自底向上语法制导翻译程序总体框架如下图所示。



例7.3 设文法 $G[E]$ 定义如下，仅仅文法考虑数据值value属性，试设计属性文法，并基于特殊二义性文法的LR分析法，给出输入串 $7+8*5$ 的语法制导翻译过程。

$G[E]$: (1) $E \rightarrow E^1 + E^2$ (2) $E \rightarrow E^1 * E^2$
(3) $E \rightarrow (E^1)$ (4) $E \rightarrow d$

i) 设计属性文法如下，其中 $@(a)$ 表示词法分析提供的a之数据值。

$G[E]$:
(1) $E \rightarrow E^1 + E^2 \{ E.value \leftarrow E^1.value + E^2.value \}$
(2) $E \rightarrow E^1 * E^2 \{ E.value \leftarrow E^1.value * E^2.value \}$
(3) $E \rightarrow (E^1) \{ E.value \leftarrow E^1.value \}$
(4) $E \rightarrow d \{ E.value \leftarrow @(d) \}$

ii) 文法G[E]的LR分析表M如下。

$\begin{matrix} V_T \cup V_N \\ A/G \\ \text{状态} \end{matrix}$	ACTION						GOTO
	+	d	*	()	#	E
0		S ₃		S ₂			1
1	S ₄		S ₅			acc	
2		S ₃		S ₂			6
3	r ₄		r ₄		r ₄	r ₄	
4	r ₂	S ₃		S ₂	r ₂	r ₂	7
5		S ₃		S ₂			8
6	S ₄		S ₅		S ₉		
7	r ₁		S ₅		r ₁	r ₁	
8	r ₂		r ₂		r ₂	r ₂	
9	r ₃		r ₃		r ₃	r ₃	

iii) 输入串7+8*5的语法制导翻译过程演示，其中符号“—”表示空属性值。

7.1.4 基于L-属性文法的语义计算

假定在语法分析中建立了抽象语法树，可以采用下列基于深度优先的后序遍历的算法，遍历语法树，完成属性计算。

```
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
```

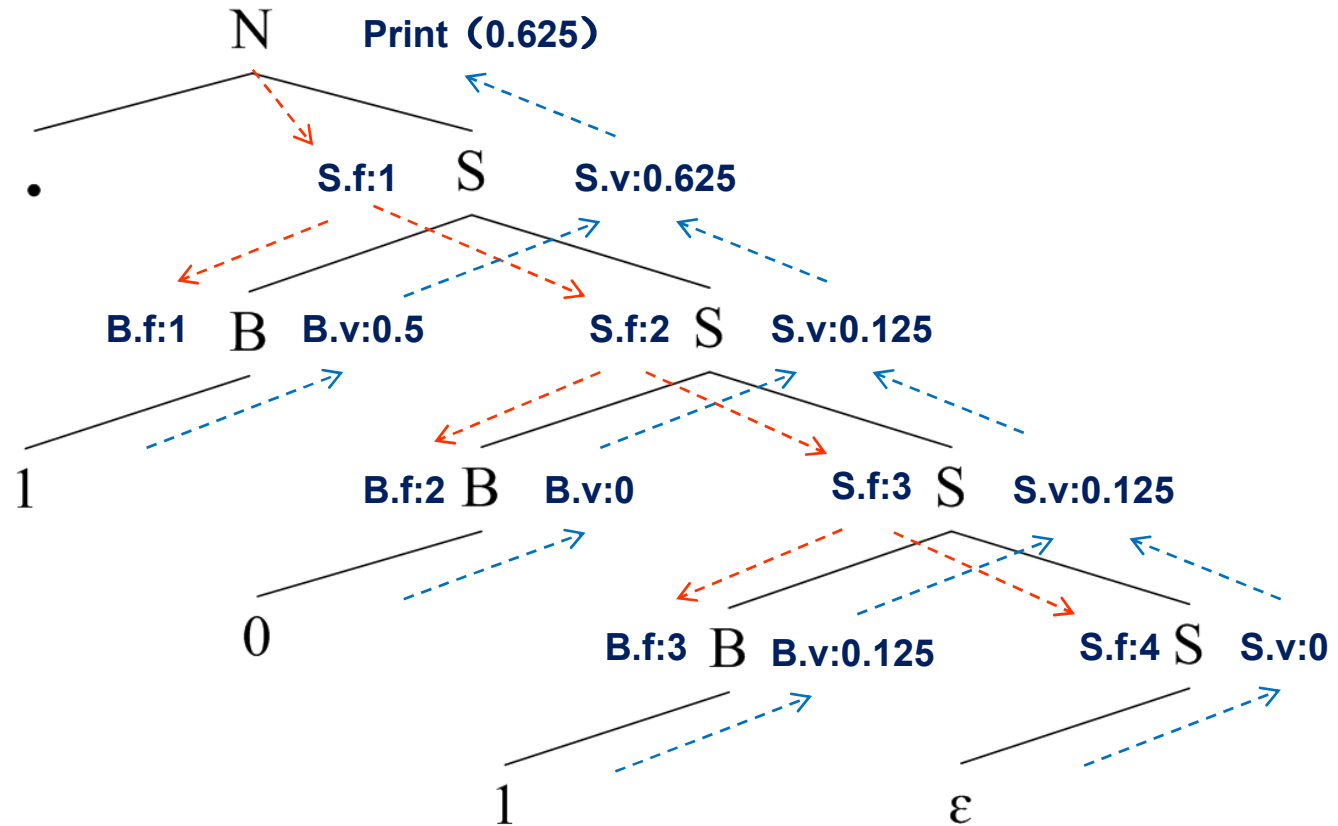
遍历语法树完成语义计算

例7.4 将二进制小数转变成十进制小数的属性文法。分析符号串 .101

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

$S \rightarrow BS_1 \quad \{ S_1.f := S.f + 1; B.f := S.f; S.v := S.v + B.v \}$

$S \rightarrow \epsilon \quad \{ S.v := 0 \} \quad B \rightarrow 0 \quad \{ B.v := 0 \} \quad B \rightarrow 1 \quad \{ B.v := 2^{-B.f} \}$



7.2 基于翻译模式的语义计算

7.2.1 翻译 (Translation Scheme) 模式

适合语法制导语义计算的另一种描述形式，面向实现的一种计算模式。可以体现一种合理调用语义动作的翻译算法

形式上类似于属性文法，但允许由 {} 括起来的语义规则集合出现在产生式右端的任何位置。这样做的好处是可以显式地表达动作和属性计算的次序，而在前述的属性文法中不体现这种次序

S-翻译模式：一种仅涉及到综合属性的情形，通常语义动作集置于相应产生式右端的末尾。常采用LR的自底向上的分析法，和S属性文法类似。

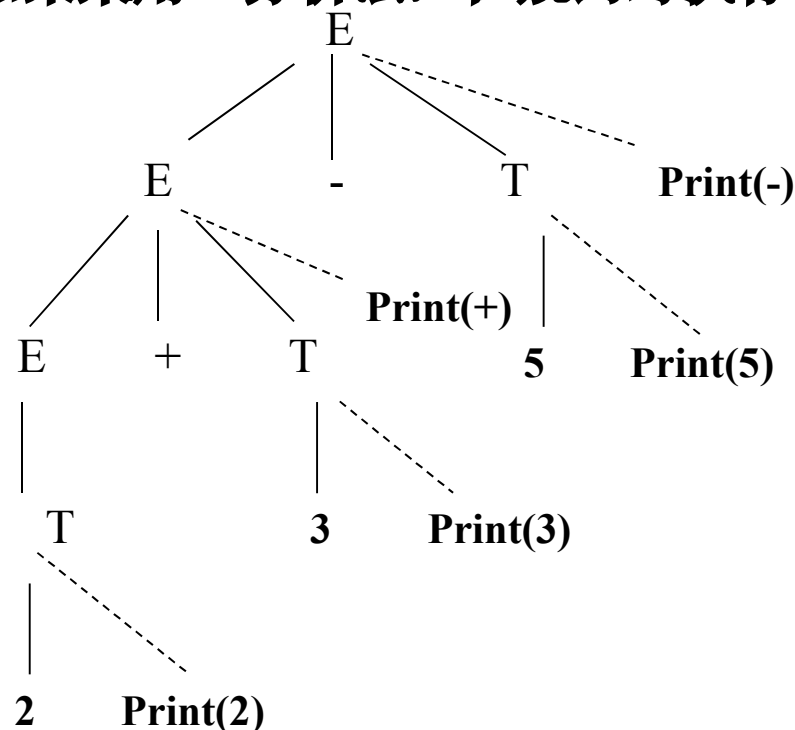
L-翻译模式：既包含综合属性，也可以包含继承属性，需要满足2个条件（1）产生式右部的某个符号的继承属性的计算必须位于该符号前，语义动作不访问右边符号的属性；（2）综合属性位于产生式的尾部。

7.2.2 基于S-翻译模式的语义计算

例7.5将中缀表达式翻译成后缀表达式的属性文法

$$\begin{aligned} E &\rightarrow E \text{ op } T && \{ \text{print(addopLexeme)} \} \mid T \\ T &\rightarrow \text{num} && \{ \text{print(num.val)} \} \end{aligned}$$

对表达式2+3-5，如果采用LR分析法，在规约时执行语义动作即可完成。



7.2.3 基于L-翻译模式的自顶而下的语义分析

将7.5的改写成LL(1)文法后，中缀表达式翻译成后缀表达式的属性文法

$$E \rightarrow TR \quad R \rightarrow op \ T \quad \{ \text{print}(op.\text{Lexeme}) \} \ R \mid \epsilon$$

$$T \rightarrow \text{num} \quad \{ \text{print}(\text{num.val}) \}$$

处理成了将语义动作嵌入规则右部文法符号之间的形式(翻译模式)
对表达式2+3-5,


步骤	符号栈	当前符号	剩余输入串	动作
1	#E	2	+3-5#	$E \rightarrow TR$
2	#RT	2	+3-5#	$T \rightarrow \text{num}$
3	#Rprint(2)2	2	+3-5#	匹配、显示2
4	#R	+	3-5#	$T \rightarrow opTR$
5	#Rprint(+)T+	+	3-5#	匹配
6	#Rprint(+)T	3	-5#	$T \rightarrow \text{num}$

步骤	符号栈	当前符号	剩余输入串	动作
6	#Rprint(+) print(3)3	3	-5#	匹配显示3+
6	#R	-	-5#	$T \rightarrow opTR$
6	#Rprint(-)T-	-	5#	匹配-
6	#Rprint(-)T	5	#	$T \rightarrow num$
6	#Rprint(-)5	5	#	匹配显示5-
6	#R	#		$R \rightarrow \varepsilon$
6	#	#		结束

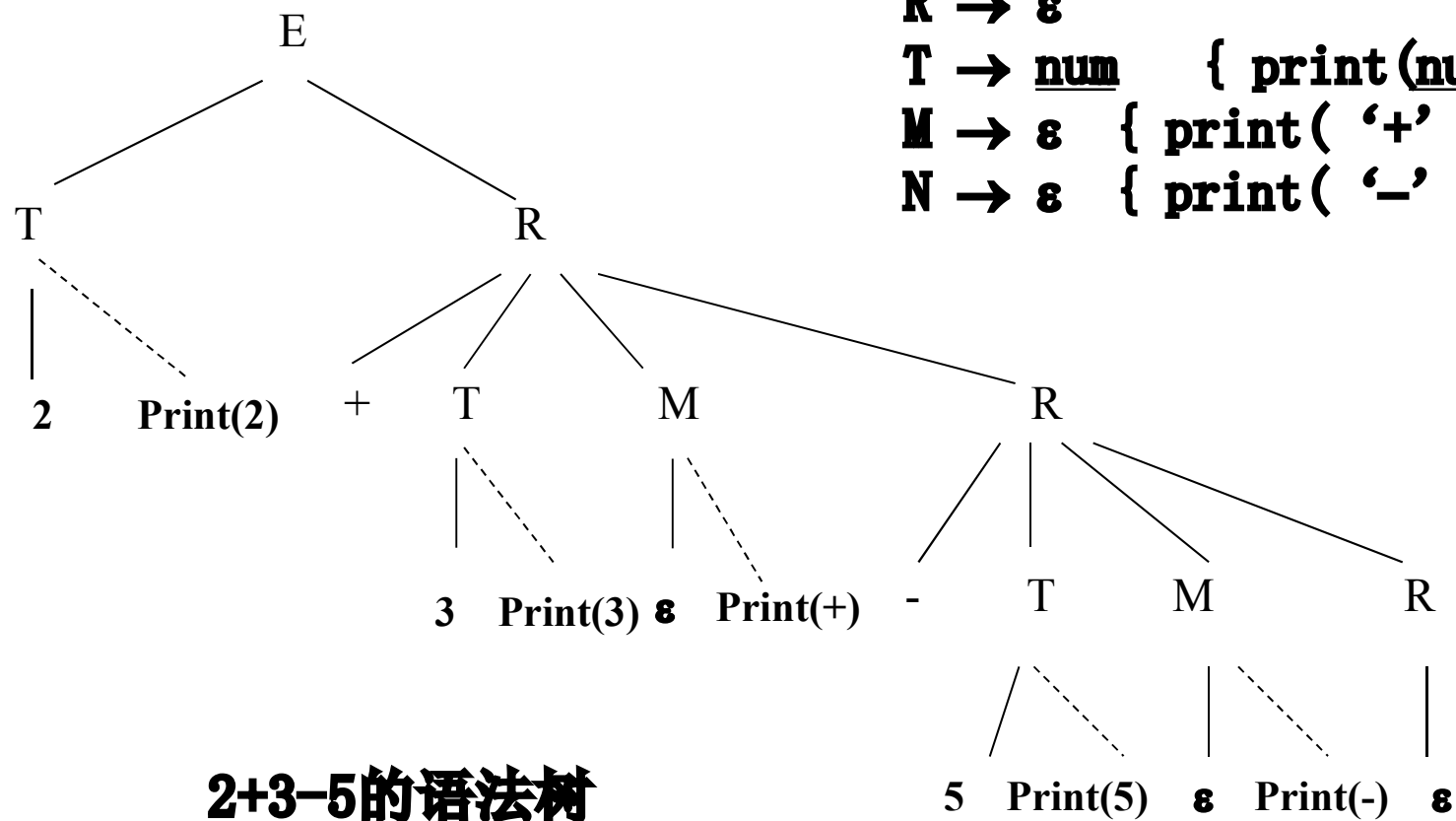
以上只是一个简化了的分析过程，实际上还需要一个语义栈维护有关属性的值，进行属性计算。

7.2.4 基于L-翻译模式在自底向上语义计算

引入新的非终结符，从翻译模式中去掉嵌在产生式中间的语义规则集。对于语义动作集中未关联任何继承属性的，翻译模式可以变换后按S-属性文法的处理方式

$\begin{aligned} E &\rightarrow T \ R \\ R &\rightarrow + \ T \ \{ \text{print}('+') \} \ R_1 \\ R &\rightarrow - \ T \ \{ \text{print}('-') \} \ R_1 \\ R &\rightarrow \varepsilon \\ T &\rightarrow \underline{\text{num}} \ \{ \text{print}(\underline{\text{num}}.\text{val}) \} \end{aligned}$		$\begin{aligned} E &\rightarrow T \ R \\ R &\rightarrow + \ T \ M \ R_1 \\ R &\rightarrow - \ T \ N \ R_1 \\ R &\rightarrow \varepsilon \\ T &\rightarrow \underline{\text{num}} \ \{ \text{print}(\underline{\text{num}}.\text{val}) \} \\ M &\rightarrow \varepsilon \ \{ \text{print}('+') \} \\ N &\rightarrow \varepsilon \ \{ \text{print}('-') \} \end{aligned}$
---	---	---


采用LR分析法，完成分析过程



2+3-5的语法树

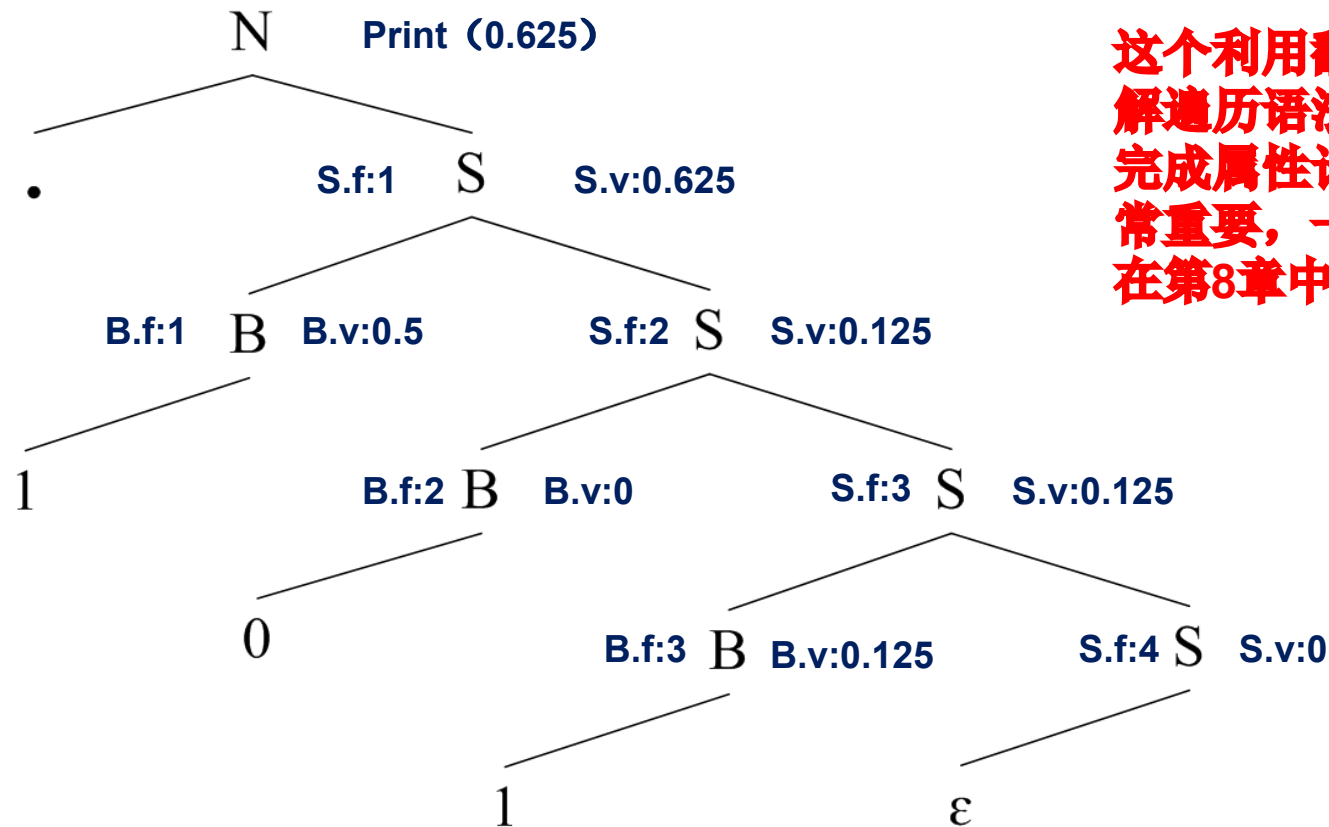
$E \rightarrow T R$
 $R \rightarrow + T M R_1$
 $R \rightarrow - T N R_1$
 $R \rightarrow \epsilon$
 $T \rightarrow \underline{\text{num}} \quad \{ \text{print}(\text{num.val}) \}$
 $M \rightarrow \epsilon \quad \{ \text{print}(' + ') \}$
 $N \rightarrow \epsilon \quad \{ \text{print}(' - ') \}$

例7.6 对于L-属性文法，处理成 L翻译模式

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

L翻译模式引导在对应语法树上遍历时的属性计算, 每次访问一个结点时, 完成子结点对应文法符号前的语义属性的计算。某棵子树遍历结束, 执行对应规则后的语义动作。比属性文法显得更直观。

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



这个利用翻译模式, 理解遍历语法树过程中, 完成属性计算的过程非常重要, 一定要看懂。在第8章中可对照使用。

7.3 分析和翻译程序的自动生成工具yacc

详见实验教材bison的使用方法。

本章小结

本章研究语义分析的基本原理和方法，主要介绍属性文法和翻译模式的形式及其特点，在语法分析过程制导下，如何进行语义分析的各种相应的技术。

属性文法是将文法、属性和断言相互关联，用于准确描述，在语法分析过程制导下，即在每个规则推导(或归约)时，语义分析的处理规则。

对于自上而下和自下而上的两类语法分析制导翻译方法，其属性值的传递方法具有不同的特点。据此，属性划分为继承属性和综合属性两类。

重点掌握的内容是

- ①属性文法基本概念与属性的计算方法；**
- ②翻译模式基本概念与属性的计算方法。**