第六章 属性文法和语法制导翻译

授课人: 高珍

gaozhen@tongji.edu.cn

内容线索

- **属性文法**
- **语法制导翻译**

目前实际应用中比较流行的语义描述和语义处理方法是属性文法和语法制导翻译的方法

属性文法

- Donald Ervin Knuth在1968年提出
- 在上下文无关文法的基础上，在描述语义动作时，为每个文法符号（终结符和非终结符）配备若干相关的“值”，如“类型”，“地址”等，称为**属性**。
- 对文法的每个产生式配备一组属性计算规则称为**语义规则**，它的描述形式为 $b:=f(c_1,c_2,..c_k)$，其中 $b,c_1,c_2...c_k$为文法符号的属性，f是一个函数。
- 每个文法符号联系于一组属性，且对每个产生式都给出其语义规则的文法称为**属性文法**。

属性和语义规则

- 属性代表与文法符号相关信息，如类型、值、代码序列、符号表内容等；
- 属性可以进行计算和传递；
- 在一个属性文法中，对应于每个产生式$A \rightarrow \alpha$都有一组与之相关联的语义规则，每条规则的形式为：

$$b:=f(c_1,c_2,\cdots,c_k)$$

这里，f是一个函数，属性b依赖于属性$c_1,c_2,\cdots,c_k$

（1）b是A的一个属性，并且$c_1,c_2,\cdots,c_k$是产生式右边文法符号的属性，则b是A的**综合属性**；

（2）b是产生式右边某个文法符号X的一个属性，并且$c_1,c_2,\cdots,c_k$ 是A或产生式右边任何文法符号的属性， b是X的**继承属性**；

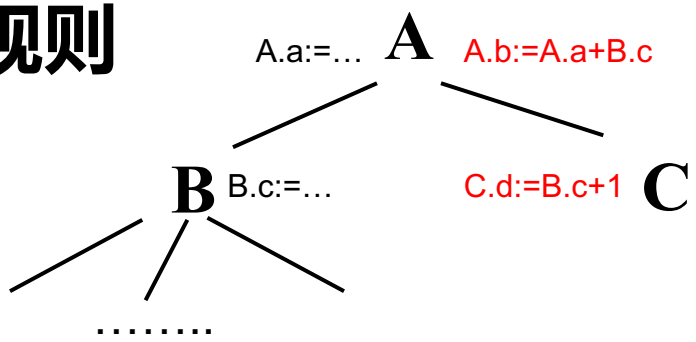
例. 考虑非终结符A，B和C，其中，

	综合属性	继承属性
A	b	a
B	c	
C		d

产生式A→BC可能有语义规则

C.d:=B.c+1

A.b:=A.a+B.c



而属性A.a和B.c在其它地方计算

简单台式计算器的属性文法

产生式	语义规则
$L \rightarrow En$	print(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

记号表示

- 对于某个文法符号$X \in V_T \cup V_N$，用X . type（X的类型），X . cat（X的种别），X .val（X的值或地址）等表示它的**属性**。
- 用**下标**区分同一产生式中**相同符号的多次出现**。

S-属性文法

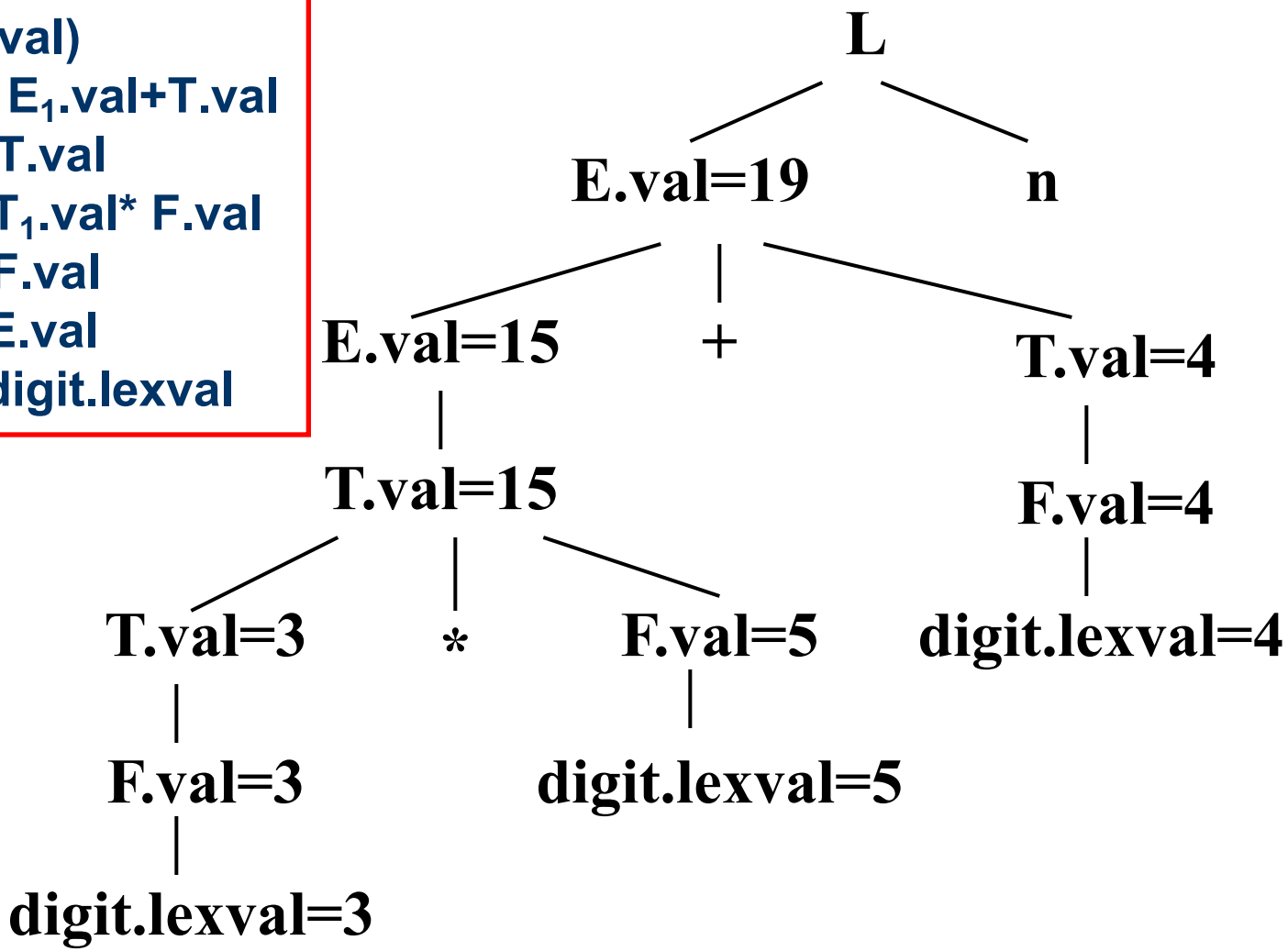
- 在语法树中，一个结点的**综合属性**的值由其**子结点**的属性值确定。
- 使用自底向上的方法在每一个结点处使用语义规则计算综合属性的值
- 仅仅使用综合属性的属性文法称**S－属性文法**

例1：属性计算

产生式	语义规则
$L \rightarrow En$	print(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

n: 换行符

3*5+4n的带注释的语法树



L-属性文法

继承属性：在语法树中，一个结点的**继承属性**由此结点的**父结点和/或兄弟结点**的某些属性确定

L属性文法

- 如果每个产生式$A \rightarrow X_1 \ldots X_{j-1} X_j \ldots X_n$的每条语义规则计算的属性是$A$的综合属性；或者是$X_j$的继承属性，但它仅依赖：
  - 该产生式中X_j左边符号$X_1, X_2, \ldots, X_{j-1}$的属性；
  - A的继承属性

- S属性文法属于L属性文法

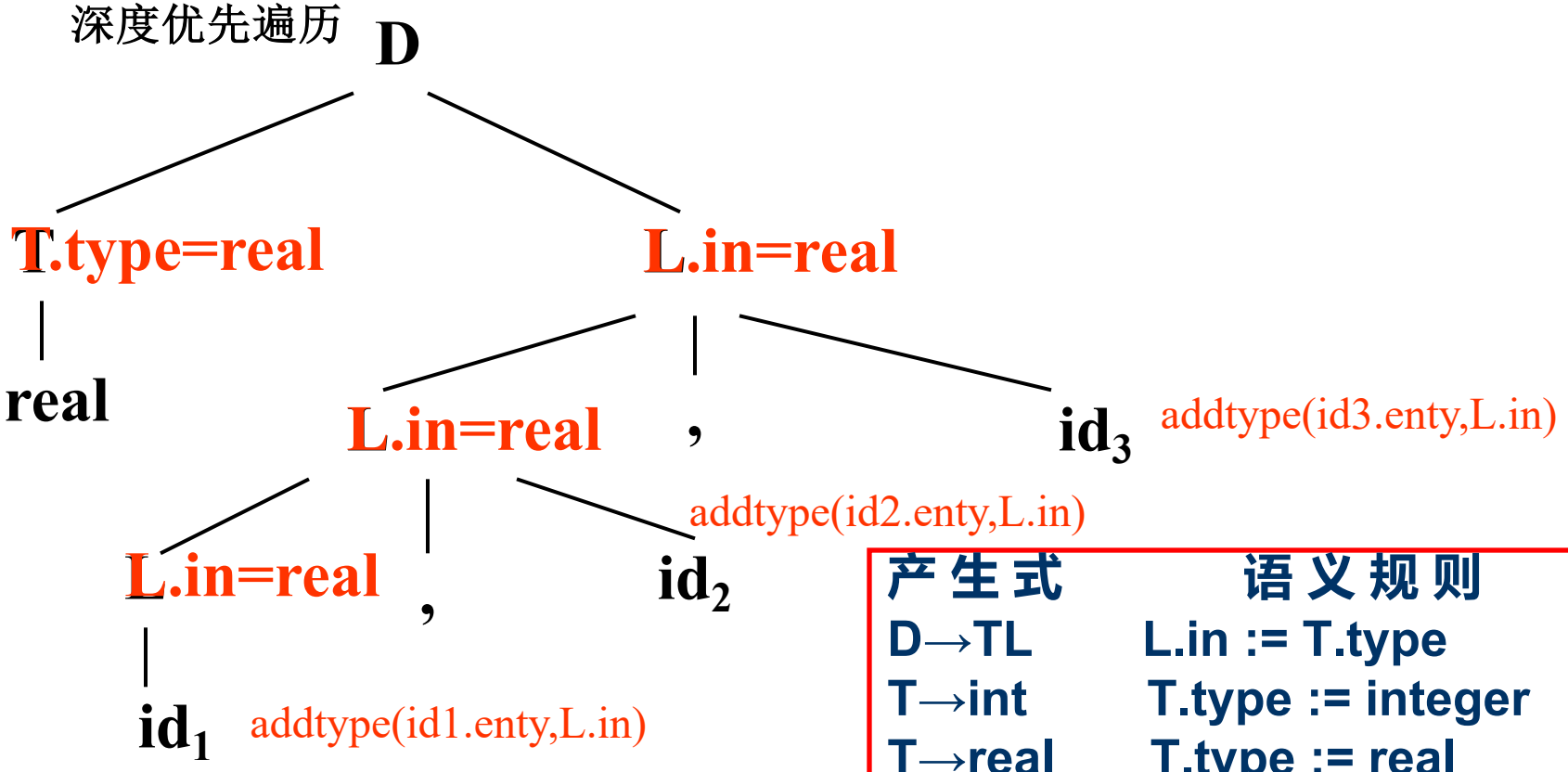
例2：符号表操作

带继承属性L.in的属性文法

产生式	语义规则
D→TL	L.in := T.type
T→int	T.type := integer
T→real	T.type := real
$L \rightarrow L_1$, id	L_1.in :=L.in
	addtype(id.entry, L.in)
L→id	addtype(id.entry, L.in)

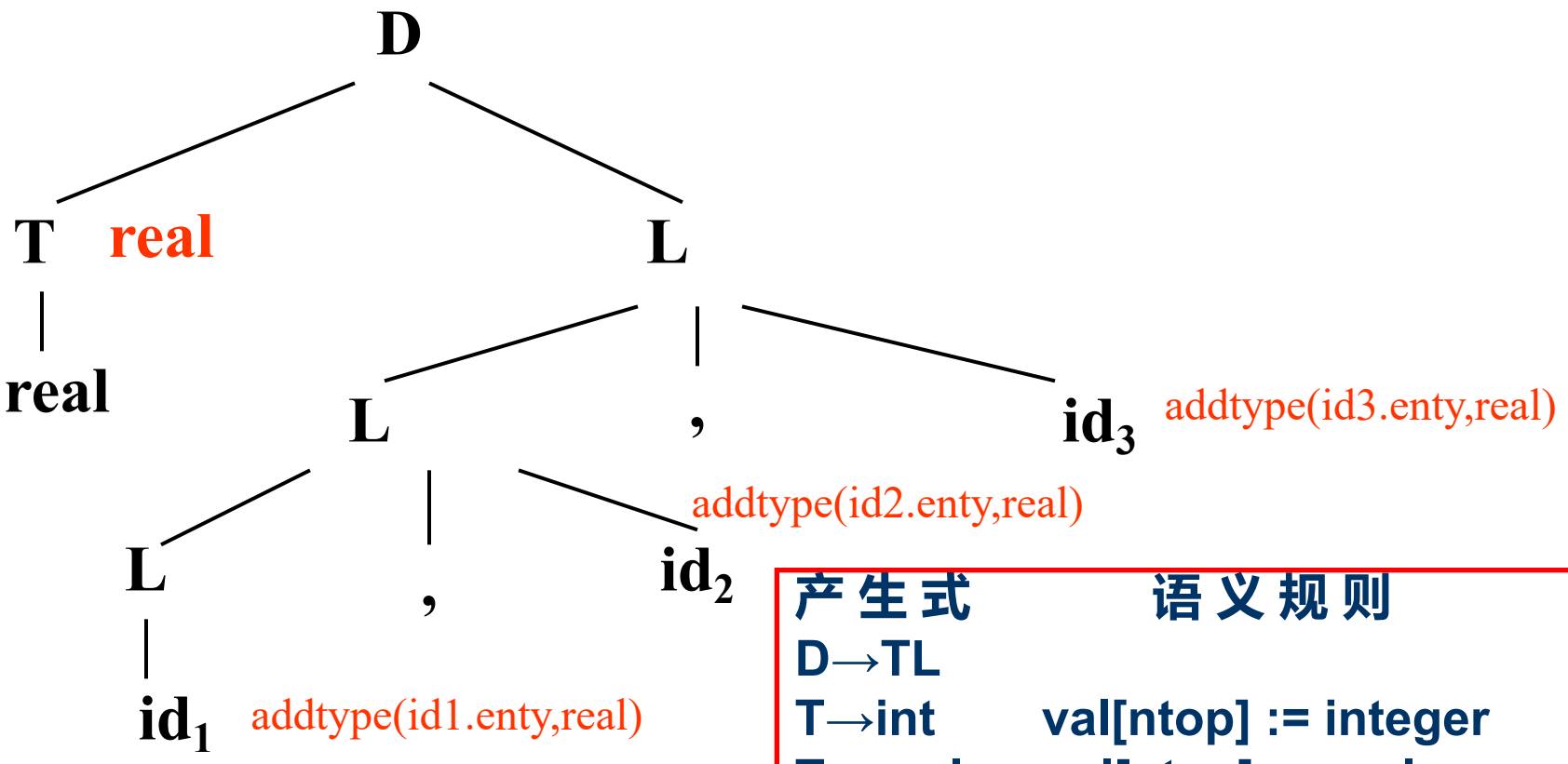
句子real id_1，id_2，id_3的带注释的语法树

深度优先遍历



产生式	语义规则
D→TL	L.in := T.type
T→int	T.type := integer
T→real	T.type := real
$L \rightarrow L_1$, id	L_1.in :=L.in addtype(id.entry, L.in)
L→id	addtype(id.entry, L.in)

LR分析法



产生式	语义规则
D→TL	
T→int	val[ntop] := integer
T→real	val[ntop] := real
$L \rightarrow L_1$, id	addtype(val[top],val[top-3])
L→id	addtype(val[top],val[top-1])

top:规约前栈顶，ntop:规约后栈顶

*

例3: 静态语义检查(类型检查)

$E \rightarrow T^1 + T^2$

{ if T^1.type = int and T^2.type= int

then E.type :=int

else error}

$E \rightarrow T^1$ or T^2

{ if T^1.type = bool and T^2.type=bool

then E.type :=bool

else error}

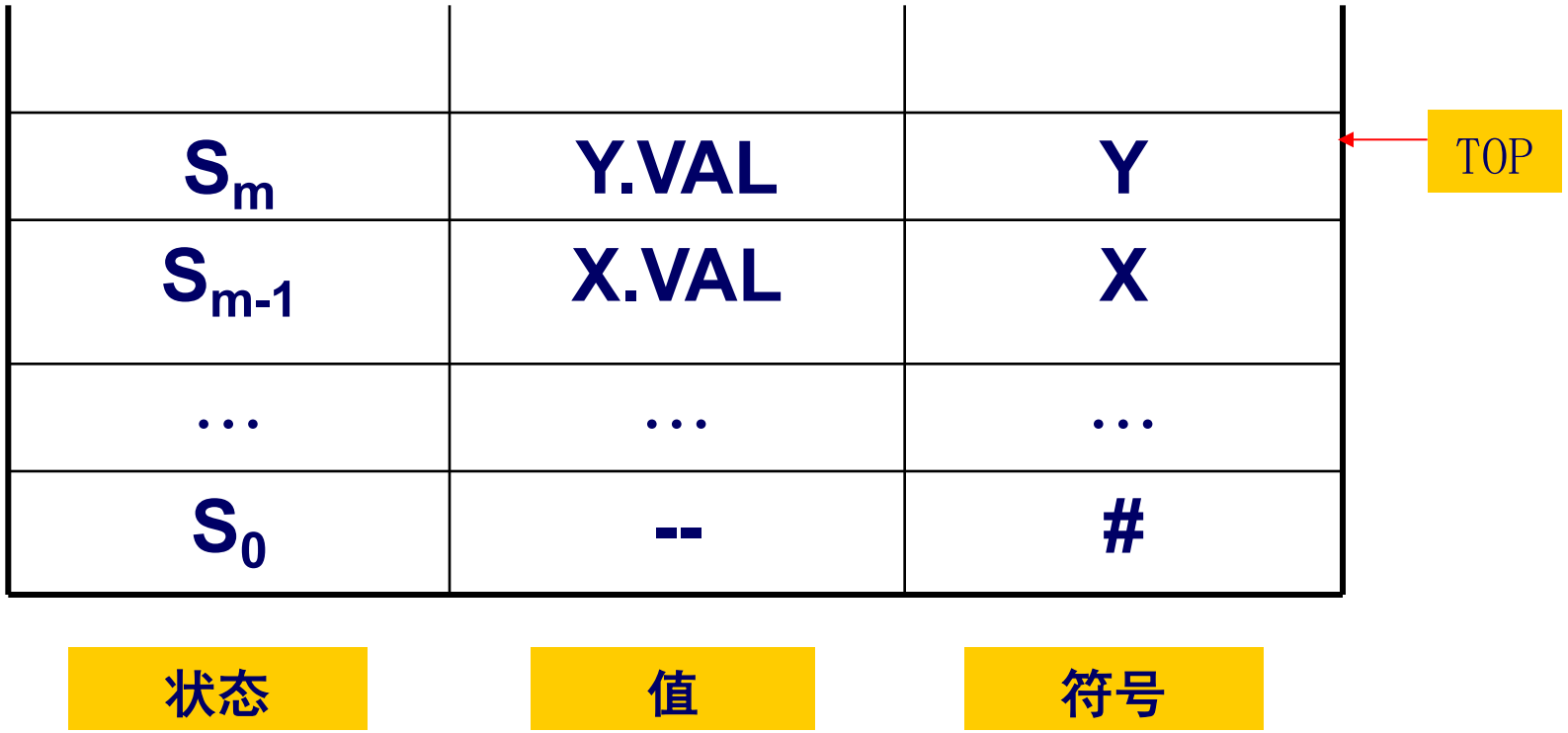
$T \rightarrow n$ { T.type := int}

$T \rightarrow b$ { T.type := bool}

LR(0)分析表

状态	action					GOTO	
	+	or	n	b	#	E	T
0			s4	s3		1	2
1					acc		
2	s5	s7					
3	r4	r4	r4	r4	r4		
4	r3	r3	r3	r3	r3		
5			s4	s3			6
6	r1	r1	r1	r1	r1		
7			s4	s3			8
8	r2	r2	r2	r2	r2		

LR分析器的栈加入语义值。



$E \rightarrow T_1 + T_2$ {
if T_1.type = int and T_2.type= int
E.type :=int
else error }
$E \rightarrow T_1$ or T_2 {
if T_1.type = bool and T_2.type=bool
E.type :=bool
else error }
$T \rightarrow n$ { T.type := int}
$T \rightarrow b$ { T.type := bool}

LR(0)分析表

状态	action					GOTO	
	+	or	n	b	#	E	T
0			s4	s3		1	2
1					acc		
2	s5	s7					
3	r4	r4	r4	r4	r4		
4	r3	r3	r3	r3	r3		
5			s4	s3			6
6	r1	r1	r1	r1	r1		
7			s4	s3			8
8	r2	r2	r2	r2	r2		

输入串： n + n

6	T	int
5	+	---
1	E	int
0	#	--

*

输入串
n + b

6	E	bool
5	+	---
4	T	int
0	#	--

$E \rightarrow T_1 + T_2$ {
if T_1.type = int and T_2.type= int
E.type :=int
else error }
$E \rightarrow T_1$ or T_2 {
if T_1.type = bool and T_2.type=bool
E.type :=bool
else error }
$T \rightarrow$ n { T.type := int}
$T \rightarrow$ b { T.type := bool}

LR(0)分析表

状态	action					GOTO	
	+	or	n	b	#	E	T
0			s4	s3		1	2
1					acc		
2	s5	s7					
3	r4	r4	r4	r4	r4		
4	r3	r3	r3	r3	r3		
5			s4	s3			6
6	r1	r1	r1	r1	r1		
7			s4	s3			8
8	r2	r2	r2	r2	r2		

*

总结

- 终结符只有综合属性，由词法分析器提供
- 非终结符既可有综合属性也可有继承属性，文法开始符号的所有继承属性作为属性计算前的初始值
- 对出现在产生式右边的继承属性和出现在产生式左边的综合属性都必须提供一个计算规则。属性计算规则中只能使用相应产生式中的文法符号的属性
- 出现在产生式左边的继承属性和出现在产生式右边的综合属性不由所给的产生式的属性计算规则进行计算，它们由其它产生式的属性规则计算或者由属性计算器的参数提供
- 语义规则所描述的工作可以包括属性计算、符号表操作、静态语义检查、代码生成等等。

内容线索

- ✓ 属性文法
- ■ 语法制导翻译

概述

- 基于属性文法的处理过程

输入串 ⟶ 语法树 ⟶ 依赖图 ⟶ 语义规则计算次序

- 由源程序的语法结构所驱动的处理办法就是**语法制导翻译法**
  - 依赖图
  - 树遍历
  - **一遍扫描**

一遍扫描的处理方法

- 一遍扫描的处理方法是在语法分析的同时计算属性值
  - 所采用的语法分析方法
  - 属性的计算次序
- S - 属性文法适合于一遍扫描的自下而上分析
- L - 属性文法适合于一遍扫描的自上而下分析

语法制导翻译

- 直观上说就是为每个产生式配上一个翻译子程序（称语义动作或语义子程序），并且在**语法分析**的同时执行它。
- 语义动作一方面规定了产生式**产生的符号串**的意义，另一方面又按照这种意义规定了生成中间代码应做的**基本动作**。
- 定义：在语法分析过程中，当一个产生式**获得匹配**（自上而下分析）和**用于归约**（自下而上分析）时，此产生式对应的语义子程序进入工作，完成既定翻译任务，产生中间代码。

语法制导翻译的作用

- 如果语义动作不是简单的计值程序，而是某种中间代码的产生程序，那么随着语法分析的进展，这种代码也逐步生成。
- 语法制导翻译的作用
  - 产生中间代码
  - 产生目标指令
  - 对输入串进行解释执行

随堂练习

- 假设有10进制实数结构文法GR:

  $R \rightarrow N.N$

  $N \rightarrow d$

  $N \rightarrow Nd$

写出定义10进制实数值的属性文法，并对23.05画出带计算结果的属性树。

Dank u
Dutch

Merci
French

Спасибо
Russian

Gracias
Spanish

شكراً
Arabic

감사합니다
Korean

Tack så mycket
Swedish

धन्यवाद
Hindi

תודה רבה
Hebrew

Obrigado
Brazilian Portuguese

谢谢
Chinese

Thank You !

Dankon
Esperanto

ありがとうございます
Japanese

Trugarez
Breton

Tak
Danish

Danke
German

Grazie
Italian

நன்றி
Tamil

děkuji
Czech

ขอบคุณ
Thai

go raibh maith agat
Gaelic

*