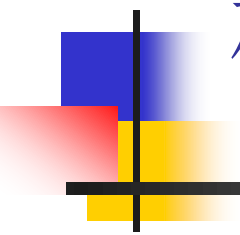


第八章 语法制导翻译和中间代码生成

8.1语法制导翻译

- 语法制导翻译：在产生式中加入相应的语义动作，并在语法分析的同时执行这些语义动作。
- 扩充CFG
  - 语法符号←→**属性**——语法树节点，记录域
  - 产生式←→**语义规则**——语法树节点，用于计算属性
- 属性类型
  - 综合，synthesized，根据孩子节点属性计算
  - 继承，inherited，由父、兄弟节点属性计算

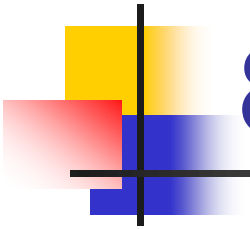
8.1 语法制导定义的形式

- 每个产生式$A \rightarrow \alpha$与一组语义规则相关联，每个语义规则具有如下形式：
  - $b = f(c_1, c_2, \ldots, c_k)$，两种可能情况
  - b为A的综合属性，$c_1, c_2, \ldots, c_k$为A、α中语法符号的属性
  - b为α中某个符号的继承属性，$c_1, c_2, \ldots, c_k$为A、α中语法符号的属性
  - b依赖$c_1, c_2, \ldots, c_k$

例8.1

产生式	语义规则
$L \rightarrow E\ n$	*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*

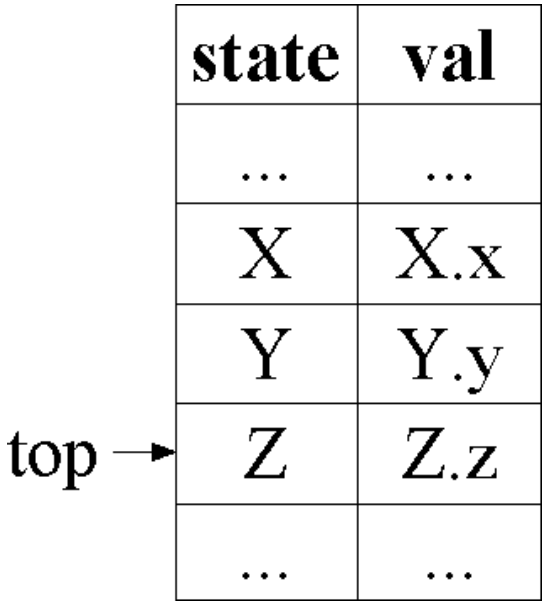
- digit.lexval：终结符只有综合属性，由词法分析器提供
- 开始符号通常没有继承属性



8.2 自底向上计算S-属性定义

- 与LR(1)分析器结合
  - 在栈中保存语法符号的属性值
  - 归约时，利用栈中语法符号（产生式右部）属性值计算新的（左部符号的）综合属性值

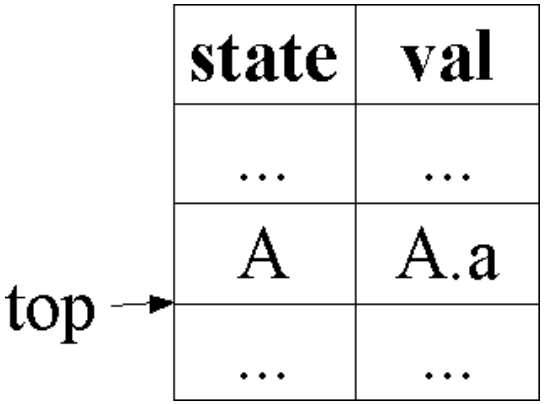
自底向上计算S-属性定义示例



$A \rightarrow XYZ$➔

$A.a = f(X.x, Y.y, Z.z)$➔

val[ntop]=f(val[top-2], val[top-1], val[top])



例8.2

产生式	代码片断
$L \rightarrow E$ **n**	*print*(*val*[top])
$E \rightarrow E_1 + T$	*val*[ntop]=*val*[top-2]+*val*[top]
$E \rightarrow T$	
$T \rightarrow T_1 * F$	*val*[ntop]=*val*[top-2]**val*[top]
$T \rightarrow F$	
$F \rightarrow (E)$	*val*[ntop]=*val*[top-1]
$F \rightarrow$ **digit**	

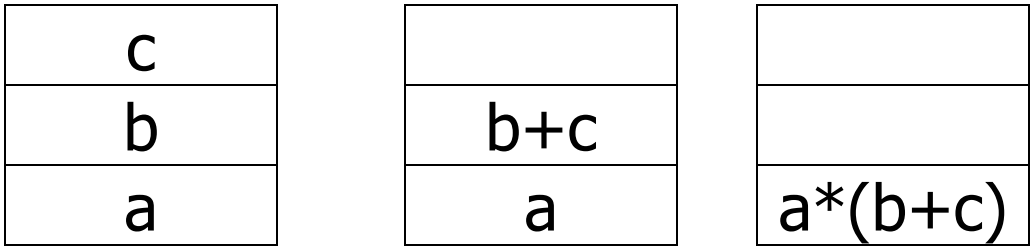
输入	状态	val	归约用产生式
3*5+4n	-	-	
*5+4n	3	3	
*5+4n	F	3	F-->**digit**
*5+4n	T	3	T-->F
5+4n	T *	3 _	
+4n	T * 5	3 _ 5	
+4n	T * F	3 _ 5	F-->**digit**
+4n	T	15	T-->T * F
+4n	E	15	E-->T
4n	E +	15 _	
n	E + 4	15 _ 4	
n	E + F	15 _ 4	F-->**digit**
n	E + T	15 _ 4	T-->F
n	E	19	E-->E + T
	E n	19 _	
	L	19	L-->E **n**

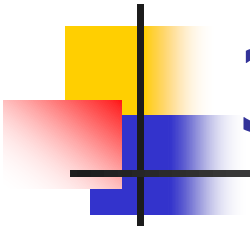
8.2 中间代码

- 一、逆波兰表示法：后缀式
- 1、把运算量写在前面，算符写在后面，不用括号的表示法。
- a*(b+c)=>abc+*
- (a+b)*(c+d)=>ab+cd+*

2、计值方法：

- 适合栈来计算：从左到右扫描后缀式，碰到运算量就推进栈；碰到K目运算就作用于栈顶的K项，并将运算结果代替这K项。





3、后缀式的推广

- 推广到比表达式更大的范围
- 例：if e then x else y
- 可表示成exy#, #代表三目运算符if-then-else.

二、三元式（op,arg1,arg2）

- A+B*C:
- OP ARG1 ARG2
- (1) * B C
- (2) + A (1)

间接三元式

- 用一张间接码表辅以三元式表，表示中间代码。
- 例：X:=(A+B)*C
- Y:=D^(A+B)
- 间接码表　　三元式表

间接码表		op	arg1	arg2
（1）				
（2）	(1)	+	A	B
（3）	(2)	*	(1)	C
(1)	(3)	:=	x	(2)
（4）	(4)	^	D	(1)
（5）	(5)	:=	Y	(4)

间接三元式

- 优点：
- 便于优化时调整运算顺序，只需重新安排码表，无需改动三元式表。
- 相同的三元式无需重复填进三元式表中，节省空间。

三、四元式

- （OP，ARG1，ARG2，RESULT）
- A:=-B*（C+D）
- OP，ARG1，ARG2，RESULT
- （1）@ B - T1
- （2）+ C D T2
- （3）* T1 T2 T3
- （4）:= T3 - A
- 优点：便于优化。

8.3简单算术表达式与赋值语句到四元式的翻译

- 一、例文法：
- A->i:=E
- E->E+E|E*E|-E|(E)|i
- NEWTEMP:函数过程。每次调用时，都回送一个代表新临时变量名的整数码作为函数值，T1，T2，。。。
- ENTRY（i）:函数过程，查符号表以确定i在表中的位置。
- E.PLACE:与非终结符E相联系的语义变量，表示存放E值的变量名在符号表的入口或整数码。
- GEN（OP，ARG1，ARG2，RESULT）：一个语义过程，把一个四元式填入四元式表。

二、类型转换

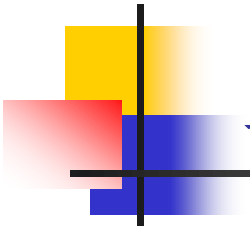
- E.MODE：表示非终结符E的类型信息。
- 例：X:=X+I*J
- (X^i,I,J,T1)
- (itr,T1,_,T2)
- ($+^r$,Y,T2,T3)
- (:=,T3,_,X)

8.4布尔表达式到四元式的翻译

- 一、布尔表达式：
- E->E^E|EvE|¬E|(E)|i|i rop I
- 计算布尔表达式的值有两种方法：
- 1、计值法
- 2、采取优化措施。
- 二、作为条件控制的布尔式翻译：
- 三、回填。

8.5控制语句的翻译

- 一、标号和转移语句（goto L）
- 语句形式：L：S;
- 1、若goto L是向后转移的语句，则L已经定义了，可立即产生出四元式（j,_,_,p）
- 2、若goto L是向前转移语句，则L未定义。产生一个不完全的四元式（j,_,_,_），它的转移目标待L定义时再回填。



二、条件语句

- 文法：
- S->if E then S
- | if E then S else S
- | while E do S
- | begin L end
- | A
- L->L,S
- |S

三、循环语句

- S->for i:=E^1 step E^2 until E^3 do S^1
- 四、分叉语句：
- 语法：case E of
- c1:S_1;
- c2:S_2;
- ……
- cn-1:S_{n-1};
- otherwise:S_n
- end

五、过程调用语句

- 例：CALL S（A+B，Z）
- 翻译成：T:=A+B
- par T
- par Z
- call S

8.6数组元素的引用

- 一、地址的计算
- 二、数组元素引用的中间代码。