山东大学
SHANDONG UNIVERSITY

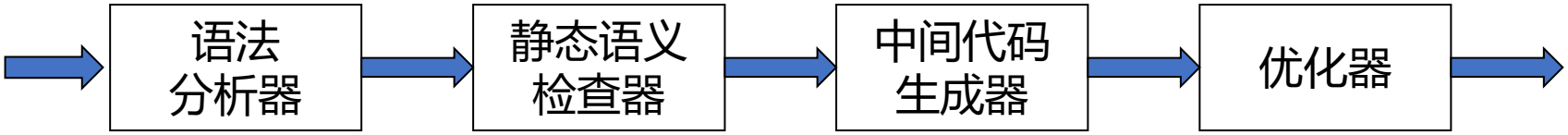
编译原理

第七章 语法制导翻译与中间代码生成

授　课　教　师：郑艳伟
手　　机：18614002860（微信同号）
邮　　箱：zhengyw@sdu.edu.cn

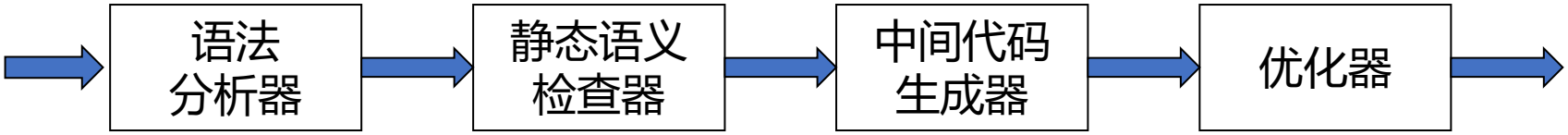
山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

静态语义检查



- 静态语义检查通常包括：
  - 类型检查：如果操作符作用于不相容的操作数，编译程序必须报告出错信息。
  - 控制流检查：控制流语句必须使控制转移到合法的地方，如c语言中，break如果不包含在while、for或switch等语句中，则报错。
  - 一致性检查：很多场合要求对象只能被定义一次。
  - 相关名字检查：有时同一名字必须出现两次或多次，需要检测出现的名字是否相同。
  - 名字的作用域分析：确定作用域范围。

中间代码生成



- 虽然源程序可以直接翻译为目标语言代码，但许多编译程序却采用了独立于机器的、复杂性介于源语言和机器语言之间的中间语言，这样做的好处是：
  - 便于进行与机器无关的代码优化工作；
  - 使编译程序改变目标机更容易；
  - 使编译程序的结构在逻辑上更为简单明确。

基于属性文法的处理方法

- 由源程序的语法结构所驱动的处理方法，称为语法制导翻译法，其语义规则计算可能：
  - 产生代码；
  - 在符号表中存放信息；
  - 给出错误信息；
  - 执行任何其它动作。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

属性翻译文法

- 属性文法在1968年由Knuth提出，也称属性翻译文法，是在上下文无关文法的基础上，为每个文法符号（$V_T \cup V_N$）配备若干属性。
  - 属性代表与文法符号相关的信息，如其类型、值、代码序列、符号表内容等；
  - 属性与变量一样，可以进行计算和传递；
  - 属性的加工过程即语义的处理过程；
  - 对文法的每个产生式都配备了一组属性的计算规则，称为语义规则。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

属性翻译文法

- 每个产生式$A \to \alpha$都有一套与之关联的语义规则，规则形式为

$$b = f(c_1, c_2, \ldots, c_k)$$

这里f是一个函数，满足以下之一：

- b是A的一个综合属性，且$c_1, c_2, \ldots, c_k$是产生式右边文法符号的属性；
- b是产生式右边某个文法符号的一个继承属性，且$c_1, c_2, \ldots, c_k$是A或产生式右部任何文法符号的属性。

- 以上两种情况，都称为属性b依赖于属性$c_1, c_2, \ldots, c_k$。
- 注意：
  - 终结符只有综合属性，它们由词法分析器提供；
  - 非终结符既可以有综合属性也可以有继承属性，文法开始符号的所有继承属性作为属性计算前的初始值。

属性翻译文法

- 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，这有助于在产生式范围内“封装”属性的依赖性。
- 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产生式的属性规则进行计算
  - 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。

【例】$A, B, C \in V_N$， A有继承属性a和综合属性b， B有综合属性c， C有继承属性d，产生式$A \to BC$可能有规则：

$$C.d = B.c + 1$$

$$A.b = B.c$$

  - $A.a$和$B.c$在其它地方计算。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

台式计算器

【例】一个简单台式计算器的属性文法：

$L \to E\backslash n$	$print(E.val)$ // 只对应一个动作，可以认为定义了L的一个虚属性
$E \to E_1 + T$	$E.val = E_1.val + T.val$ // 一个产生式出现了两个E，用下标区分
$E \to T$	$E.val = T.val$
$T \to T_1 * F$	$T.val = T_1.val * F.val$
$T \to F$	$T.val = F.val$
$F \to (E)$	$F.val = E.val$
$F \to digit$	$F.val = digit.lexval$ // $digit.lexval$由词法分析器提供

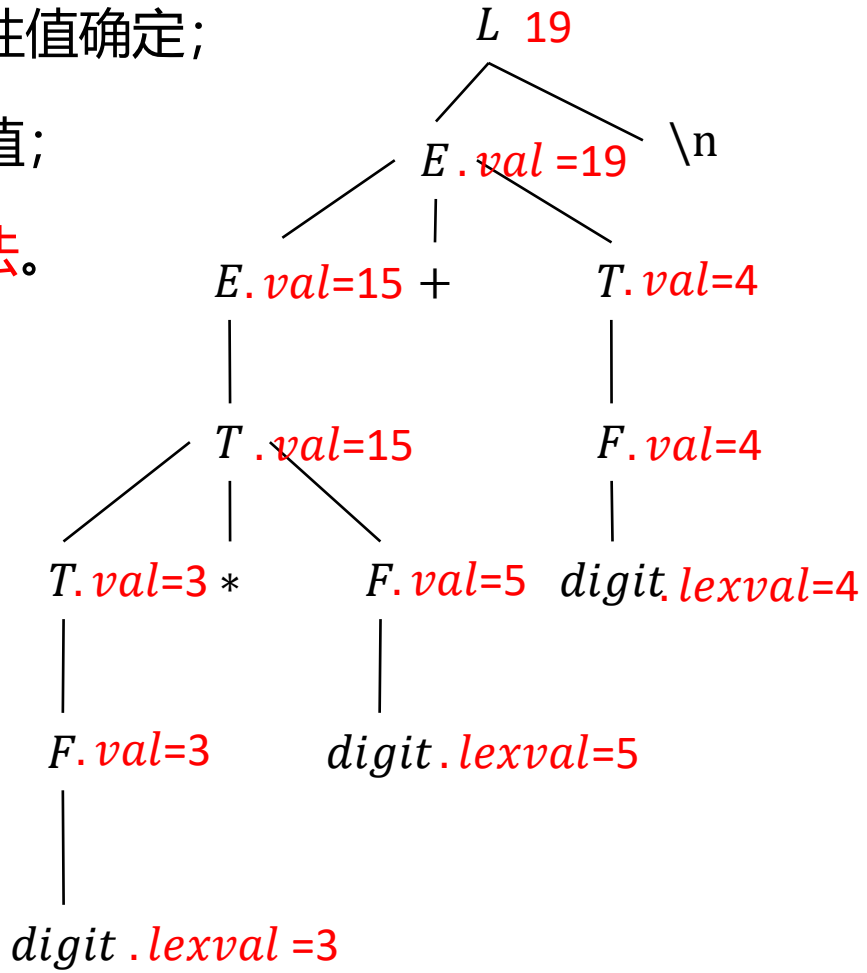
山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

综合属性

- 一个结点的综合属性值由其子结点的属性值确定；
- 通常使用自底向上的方法计算综合属性值；
- 仅仅使用综合属性的文法称为S-属性文法。

【例】3*5+4\n属性值计算

$L \to E\text{\n}$	$print(E.val)$
$E \to E_1 + T$	$E.val = E_1.val + T.val$
$E \to T$	$E.val = T.val$
$T \to T_1 * F$	$T.val = T_1.val * F.val$
$T \to F$	$T.val = F.val$
$F \to (E)$	$F.val = E.val$
$F \to digit$	$F.val = digit.lexval$



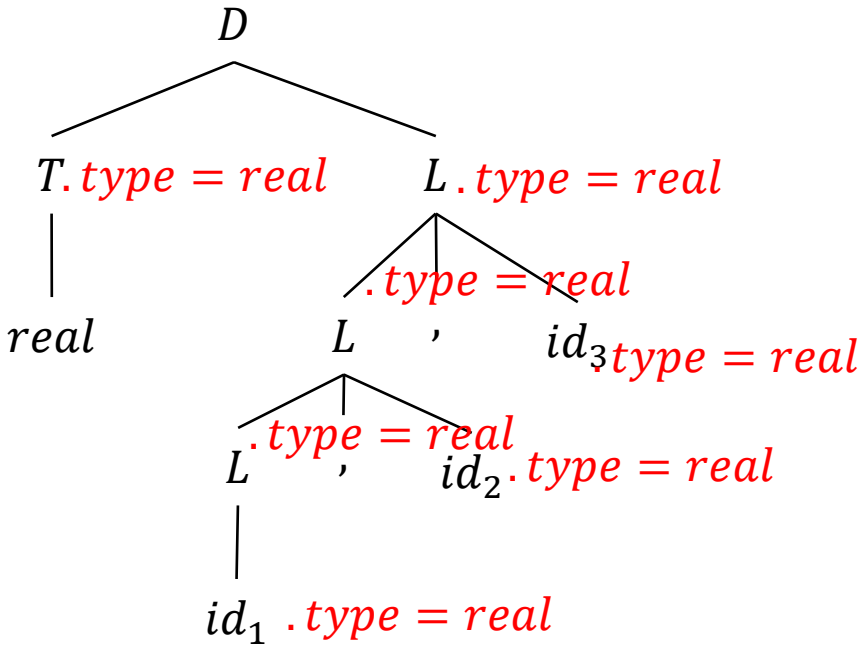
山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

继承属性

- 一个结点的继承属性值由其父和兄结点的属性值确定。

【例】数据类型跟踪：$real\ id_1, id_2, id_3$

$D \to TL$	$L.type = T.type$
$T \to int$	$T.type = integer$
$T \to real$	$T.type = real$
$L \to L_1, id$	$L_1.type = L.type$
	$addType(id.entry, L.type)$
$L \to id$	$addType(id.entry, L.type)$



山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

依赖图

- 属性计算顺序
  - 综合属性适合自下而上的计算，继承属性适合自上而下的计算。
  - 一个属性文法不可能只有综合属性或者只有继承属性，编译器设计者也应该根据实际需要和个人喜好选择自上而下分析或自下而上分析。
  - 这就需要研究各种不同属性的计算顺序问题，依赖图是表达计算顺序的一种工具。
- 如果在一颗语法树中，一个结点的属性b依赖于属性c，那么这个结点处计算b的语义规则必须在确定c的语义规则之后使用。
- 依赖图：是一个表示语法树中结点间相互依赖关系的有向图。
  - $b = f(c_1, c_2, \dots, c_k)$，则属性$b$依赖于属性$c_i$，从$c_i$向$b$画有向边；
  - 过程调用的语法规则则生成一个虚综合属性$b = f(c_1, c_2, \dots, c_k)$。

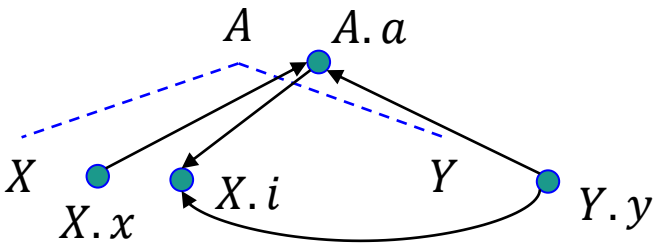
山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

依赖图

【例】

$A \to XY$ $\quad A.a = f(X.x, Y.y)$

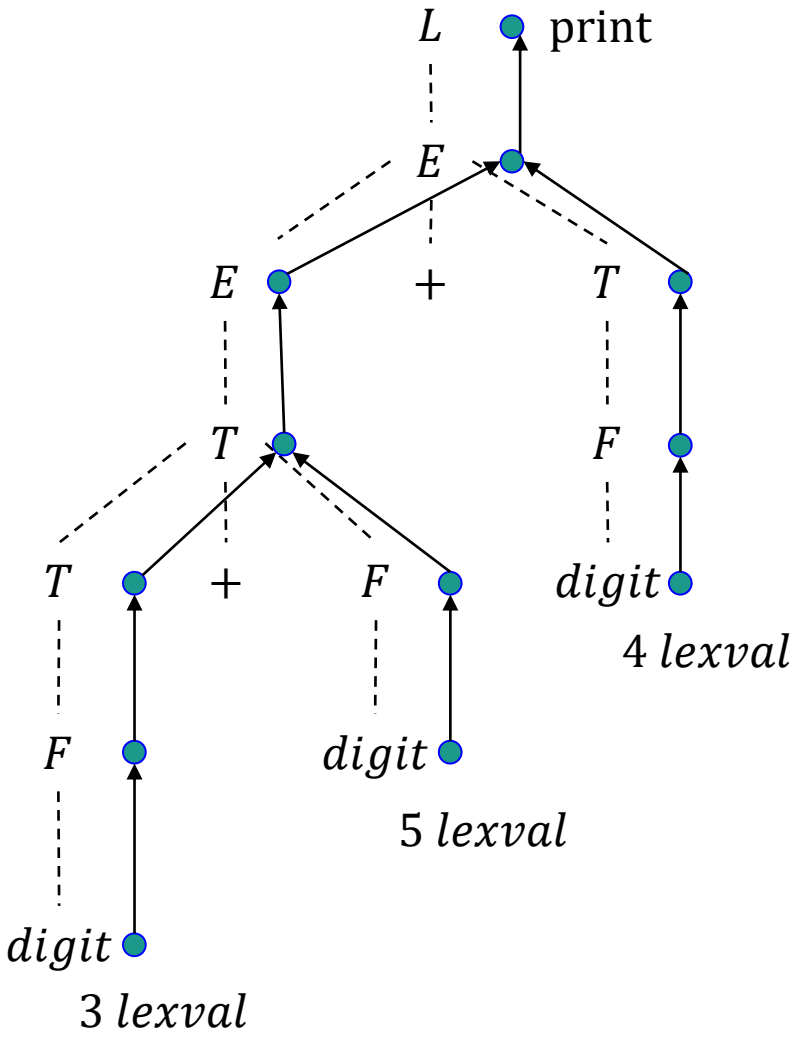
$X.i = g(A.a, Y.y)$



山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】3*5+4属性值计算

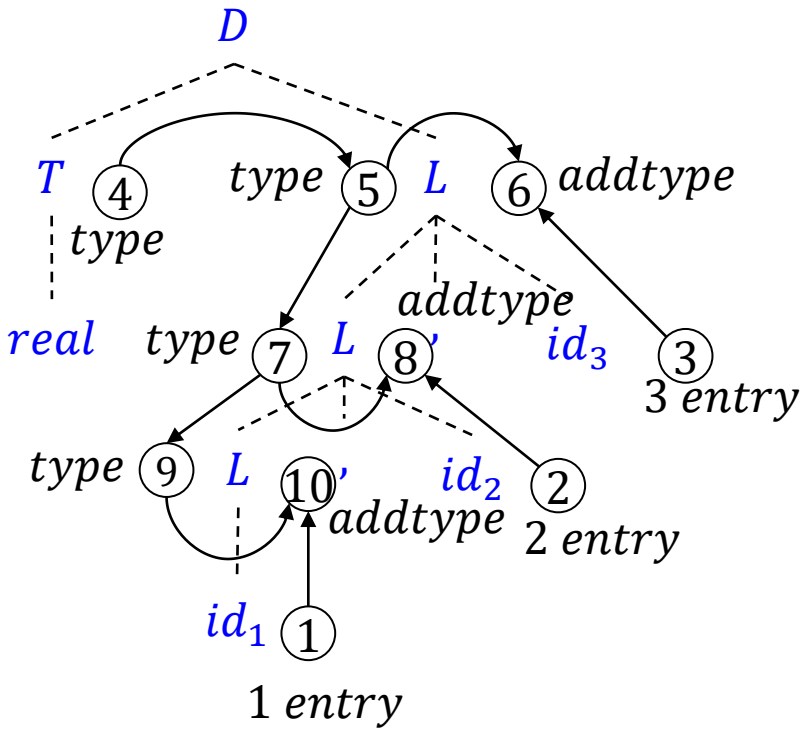
$L \to E$	$print(E.val)$
$E \to E_1 + T$	$E.val = E_1.val + T.val$
$E \to T$	$E.val = T.val$
$T \to T_1 * F$	$T.val = T_1.val * F.val$
$T \to F$	$T.val = F.val$
$F \to (E)$	$F.val = E.val$
$F \to digit$	$F.val = digit.lexval$



郑艳伟，zhengyw@sdu.edu.cn

【例】数据类型跟踪：$real\ id_1, id_2, id_3$

$D \rightarrow TL$	$L.type = T.type$
$T \rightarrow int$	$T.type = integer$
$T \rightarrow real$	$T.type = real$
$L \rightarrow L_1, id$	$L_1.type = L.type$
	$addType(id.entry, L.type)$
$L \rightarrow id$	$addType(id.entry, L.type)$



山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

依赖图

- 如果一个属性文法不存在属性之间的循环依赖关系，称该文法为良定义的
  - 为了设计编译程序，我们只处理良定义的属性文法。
- 属性的计算次序
  - 一个有向非循环图的拓扑序是图中结点的任何顺序$m_1, m_2, \ldots, m_k$，使得边必须是从序列中前面的结点指向后面的结点；
  - 即：如果$m_i \rightarrow m_j$是m_i到m_j的一条边，则序列中m_i必须在m_j之前。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】数据类型跟踪：$real\ id_1, id_2, id_3$

- 属性的计算次序：结点1-10，用a_i表示结点i的相关属性，得到如下程序：

$a_4 = real;$

$a_5 = a_4;$

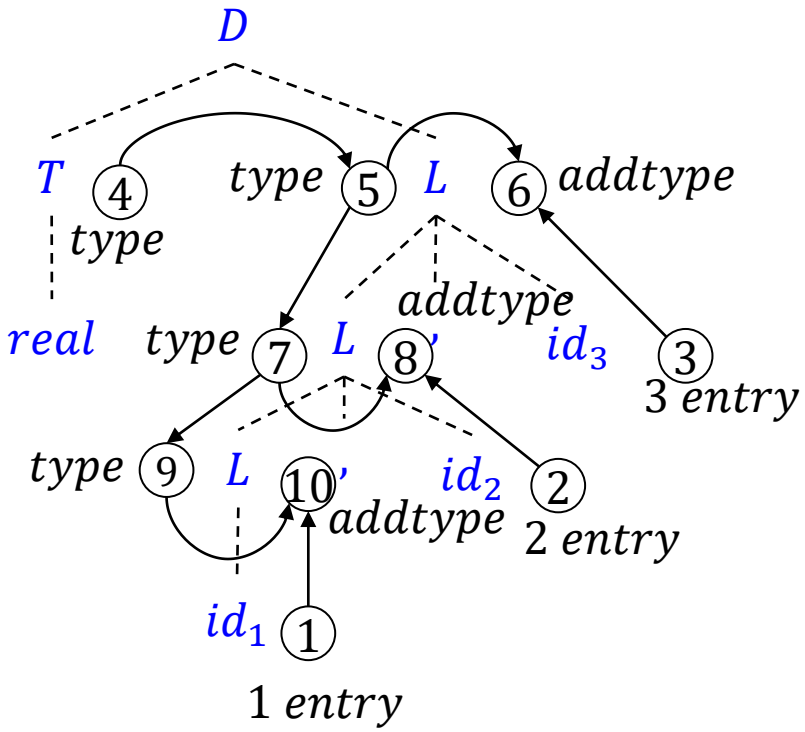
$addType(id_3.entry, a_5);$

$a_7 = a_5;$

$addType(id_2.entry, a_7);$

$a_9 = a_7;$

$addType(id_1.entry, a_9);$



山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

树遍历的属性计算算法

- 无循环属性文法计算，最坏情况时间复杂度$O(n^2)$

算法 7.1 树遍历计算属性值

输入: 属性文法 $G[S]$，句子的语法树
输出: 语法树中所有结点的所有属性值都被计算

```
1  while ∃ 未计算的属性 do
2  |   access(S);
3  end
4  function access(X):
5  |   foreach X → Y1Y2···Ym do
6  |   |   for i = 1 : m do
7  |   |   |   if Yi ∈ VN then
8  |   |   |   |   计算 Yi 所有能计算的继承属性;
9  |   |   |   |   access(Yi);
10 |   |   |   end
11 |   |   end
12 |   |   计算 X 所有能计算的综合属性;
13 |   end
14 end access
```

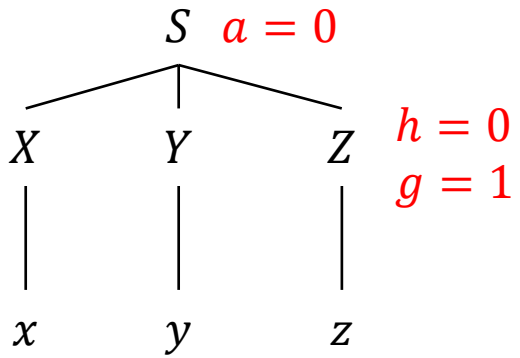
山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】输入串xyz，初始值$S.a = 0$

$S \to XYZ$	$Z.h = S.a$
	$X.c = Z.g$
	$S.b = X.d - 2$
	$Y.e = S.b$
$X \to x$	$X.d = 2 * X.c$
$Y \to y$	$Y.f = Y.e * 3$
$Z \to z$	$Z.g = Z.h + 1$

其中：

- S有继承属性a，综合属性b
- X有继承属性c，综合属性d
- Y有继承属性e，综合属性f
- Z有继承属性h，综合属性g



第1次遍历：

access (S)

$X.c$不能计算

access(X)　　$X.d$不能计算

$Y.e$不能计算

access(Y)　　$Y.f$不能计算

$Z.h = 0$

access(Z)　　$Z.g = 1$

$S.b$不能计算

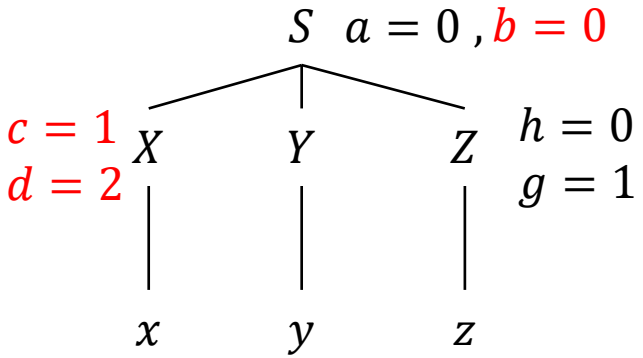
山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】输入串xyz，初始值$S.a = 0$

$S \to XYZ$	$Z.h = S.a$
	$X.c = Z.g$
	$S.b = X.d - 2$
	$Y.e = S.b$
$X \to x$	$X.d = 2 * X.c$
$Y \to y$	$Y.f = Y.e * 3$
$Z \to z$	$Z.g = Z.h + 1$

其中：

- S有继承属性a，综合属性b
- X有继承属性c，综合属性d
- Y有继承属性e，综合属性f
- Z有继承属性h，综合属性g



第2次遍历：

access(S)

$X.c = 1$

access(X)　　$X.d = 2$

$Y.e$不能计算

access(Y)　　$Y.f$不能计算

$Z.h = 0$

access (Z)　　$Z.g = 1$

$S.b = 0$

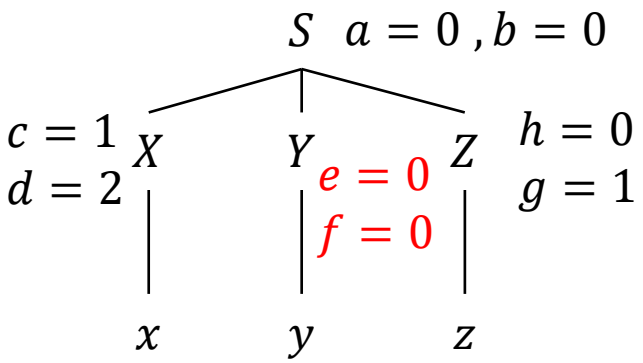
山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】输入串xyz，初始值$S.a = 0$

$S \to XYZ$	$Z.h = S.a$
	$X.c = Z.g$
	$S.b = X.d - 2$
	$Y.e = S.b$
$X \to x$	$X.d = 2 * X.c$
$Y \to y$	$Y.f = Y.e * 3$
$Z \to z$	$Z.g = Z.h + 1$

其中：

- S有继承属性a，综合属性b
- X有继承属性c，综合属性d
- Y有继承属性e，综合属性f
- Z有继承属性h，综合属性g



第3次遍历：

access(S)

$X.c = 1$

access(X) $X.d = 2$

$Y.e = 0$

access(Y) $Y.f = 0$

$Z.h = 0$

access(Z) $Z.g = 1$

$S.b = 0$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

一遍扫描的处理方法

- 一遍扫描与树遍历的不同之处
  - 在语法分析的同时计算属性值，而不是语法分析构造语法树之后进行属性计算，而且无需构造实际的语法树（如需要也可以构建）；
  - 当一个属性值不再用于计算其它属性值时，编译程序不必再保留这个属性值（如果需要也可以保留）。
- 一遍扫描的影响因素
  - 所采用的语法分析方法；
  - 属性的计算顺序。
- 一遍扫描的情况
  - 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和自下而上分析。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

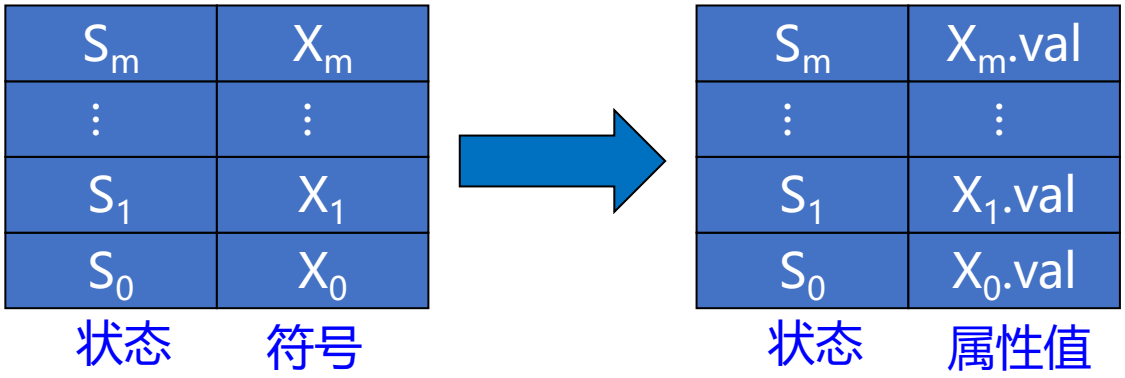
语法制导翻译

- 语法制导翻译：为文法的每个产生式配上一组语义规则，并且在语法分析的同时执行这些语义规则，完成有关语义分析和代码生成的工作。
  - 在自上而下的分析中，当一个产生式匹配输入串成功时执行；
  - 在自下而上的分析中，当一个产生式被用于归约时执行。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

S-属性文法的自下而上计算

- S-属性文法：只含有综合属性的文法。
  - 综合属性可以在分析输入符号串的同时，由自下而上的分析器来计算；
  - 分析器可以保存与栈中文法符号有关的综合属性，每当进行归约时，新的属性值就由栈中正在归约的产生式右部符号的属性值计算。
- S-属性文法通常可以借助LR分析器实现。
  - 分析器中附加一个域存放文法符号的属性值。



山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

S-属性文法的自下而上计算

【例】用LR分析器实现台式计算器

$L \to E$	$print(val[top])$ // top指栈顶
$E \to E_1 + T$	$val[ntop] = val[top-2] + val[top]$ // ntop指归约后的新栈顶
$E \to T$	
$T \to T_1 * F$	$val[ntop] = val[top-2] * val[top]$
$T \to F$	
$F \to (E)$	$val[ntop] = val[top-1]$
$F \to i$	

(0) L→E	$print(val[top])$
(1) E→E+T	$val[ntop] = val[top-2] + val[top]$
(2) E→T	
(3) T→T*F	$val[ntop] = val[top-2] * val[top]$
(4) T→F	
(5) F→(E)	$val[ntop] = val[top-1]$
(6) F→i	

状态	Action						Goto		
	i	+	*	(	)	#	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		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		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

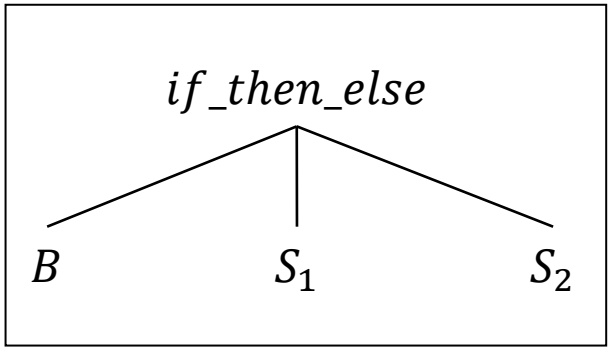
序号	状态栈	属性栈	输入串
1	0	#	$3*5+4\#$
2	05	#3	$*5+4\#$
3	03	#3	$*5+4\#$
4	02	#3	$*5+4\#$
5	027	#3 *	$5+4\#$
6	0275	#3 * 5	$+4\#$
7	02710	#3 * 5	$+4\#$
8	02	#15	$+4\#$
9	01	#15	$+4\#$
10	016	#15 +	$4\#$
11	0165	#15 + 4	#
12	0163	#15 + 4	#
13	0169	#15 + 4	#
14	01	#19	#
15	acc，分析表里没有包含$L \to E$		

- 7.1 属性文法及其计算
  - 7.1.1 属性翻译文法
  - 7.1.2 综合属性的自下而上计算
  - 7.1.3 继承属性的自上而下计算
  - 7.1.4 依赖图
  - 7.1.5 树遍历的计算方法
  - 7.1.6 一遍扫描的处理方法
- 7.2 S-属性文法
  - 7.2.1 S-属性文法的自下而上计算
  - 7.2.2 构造表达式抽象语法树
  - 7.2.3 NFA箭弧单符化
- 7.3 L-属性文法
  - 7.3.1 翻译模式
  - 7.3.2 L-属性文法自上而下计算
  - 7.3.3 L-属性文法自下而上计算
  - 7.3.4 综合属性代替继承属性
- 7.4 声明语句的翻译
  - 7.4.1 Pascal风格过程内声明语句
  - 7.4.2 Pascal风格过程定义与声明语句
  - 7.4.3 Pascal风格数组声明
  - 7.4.4 Pascal风格结构体声明
  - 7.4.5 C风格函数定义与声明语句
- 7.5 表达式与赋值语句的翻译
  - 7.5.1 算术表达式与赋值语句
  - 7.5.2 Pascal风格数组的引用
  - 7.5.3 C风格数组的引用
  - 7.5.4 结构体的引用
  - 7.5.5 作为逻辑运算的布尔表达式
  - 7.5.6 地址和指针的引用
- 7.6 控制语句的翻译
  - 7.6.1 真假出口链
  - 7.6.2 四元式链操作
  - 7.6.3 作为条件控制的布尔表达式
  - 7.6.4 if和while语句
  - 7.6.5 C风格for语句
  - 7.6.6 Matlab风格for语句
  - 7.6.7 标号-goto语句
  - 7.6.8 switch语句
  - 7.6.9 break和continue语句
  - 7.6.10 三元运算符
  - 7.6.11 关系运算符的结合
- 7.7 过程调用的翻译
  - 7.7.1 过程的开始与结束标记
  - 7.7.2 返回语句
  - 7.7.3 过程调用
- 7.8 类型检查
  - 7.8.1 类型表达式
  - 7.8.2 类型检查
  - 7.8.3 隐式转换
  - 7.8.4 显式转换

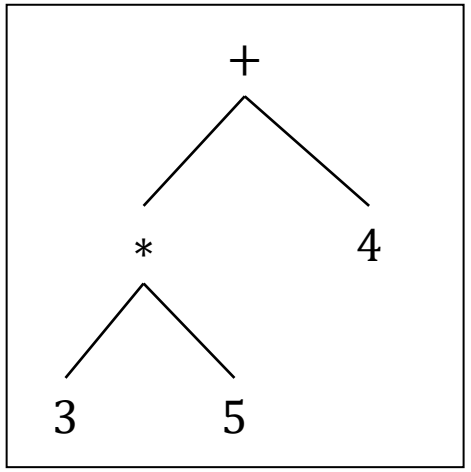
山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

表达式的抽象语法树

- 抽象语法树（Abstract Syntax Tree）：在语法树中去掉那些对翻译不必要的信息，从而获得更有效的源程序中间表示。
  - 操作符和关键字都不作为叶结点出现，而是作为内部结点；
  - 语法制导翻译既可以基于语法分析树，也可以基于抽象语法树进行。



$S \rightarrow if\ B\ then\ S_1\ else\ S_2$



$3 * 5 + 4$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

构造表达式抽象语法树

构造抽象语法树的元操作

- mknode(op, left, right)：建立一个运算符号结点，标号是op，两个域left和right分别指向左子树和右子树；
- mkleaf(id, entry)：建立一个标识符结点，标号为id，一个域entry指向标识符在符号表的入口；
- mkleaf(num, val)：建立一个数结点，标号为num，一个域val用来存放数的值。

【例】$a * 5 + b$的抽象语法树构造序列

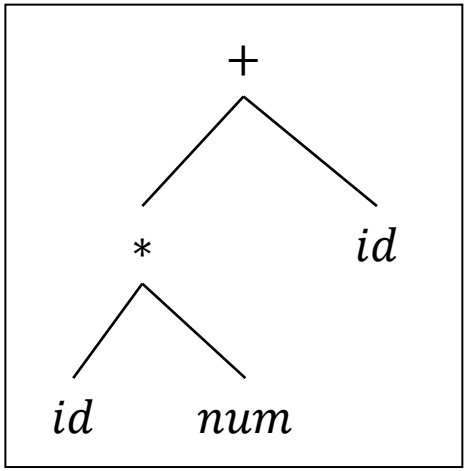
① p1 = mkleaf(id, entrya);

② p2 = mkleaf(num, 5);

③ p3 = mknode('*', p1, p2);

④ p4 = mkleaf(id, entryb);

⑤ p5 = mknode('+', p3, p4);



山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

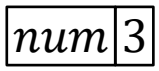
构造表达式抽象语法树

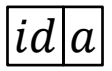
【例】为表达式建立抽象语法树的属性文法

$E \to E_1 + T$	$E.nptr = mknode('+', E_1.nptr, T.nptr)$
$E \to E_1 - T$	$E.nptr = mknode('-', E_1.nptr, T.nptr)$
$E \to T$	$E.nptr = T.nptr$
$T \to (E)$	$T.nptr = E.nptr$
$T \to id$	$T.nptr = mkleaf(id, id.entry)$
$T \to num$	$T.nptr = mkleaf(num, num.val)$

步骤	文法符号栈	输入串	动作
1	#	$3+(a+b)\#$	初始
2	#3	$+(a+b)\#$	移进
3	#T	$+(a+b)\#$	归约
4	#E	$+(a+b)\#$	归约
5	#$E+(a$	$+b)\#$	移进
6	#$E+(T$	$+b)\#$	归约

$T.nptr = (id, a)$
$E.nptr = (num, 3)$





山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

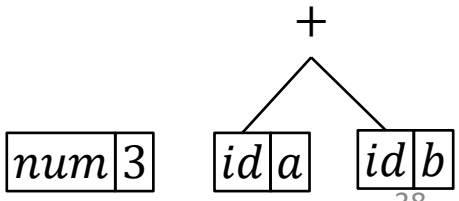
构造表达式抽象语法树

【例】为表达式建立抽象语法树的属性文法

$E \to E_1 + T$	$E.nptr = mknode('+', E_1.nptr, T.nptr)$
$E \to E_1 - T$	$E.nptr = mknode('-', E_1.nptr, T.nptr)$
$E \to T$	$E.nptr = T.nptr$
$T \to (E)$	$T.nptr = E.nptr$
$T \to id$	$T.nptr = mkleaf(id, id.entry)$
$T \to num$	$T.nptr = mkleaf(num, num.val)$

步骤	文法符号栈	输入串	动作
6	$\#E+(T$	$+b)\#$	归约
7	$\#E+(E$	$+b)\#$	归约
8	$\#E+(E+b$	$)\#$	移进
9	$\#E+(E+T$	$)\#$	归约
10	$\#E+(E$	$)\#$	归约
11	$\#E+(E)$	$\#$	移进

$T.nptr = (id, b)$
$E.nptr = (+)$
$E.nptr = (num, 3)$



山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

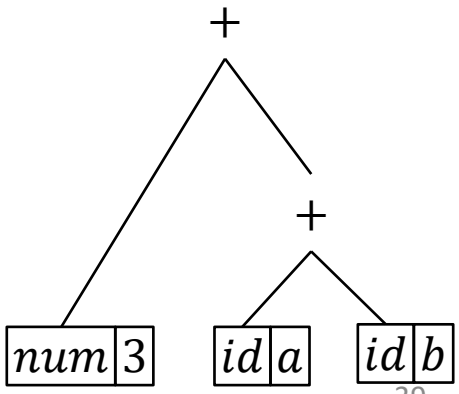
构造表达式抽象语法树

【例】为表达式建立抽象语法树的属性文法

$E \to E_1 + T$	$E.nptr = mknode('+', E_1.nptr, T.nptr)$
$E \to E_1 - T$	$E.nptr = mknode('-', E_1.nptr, T.nptr)$
$E \to T$	$E.nptr = T.nptr$
$T \to (E)$	$T.nptr = E.nptr$
$T \to id$	$T.nptr = mkleaf(id, id.entry)$
$T \to num$	$T.nptr = mkleaf(num, num.val)$

步骤	文法符号栈	输入串	动作
11	$\#E + (E)$	#	移进
12	$\#E + T$	#	归约
13	$\#E$	#	归约
14	$\#E$	#	成功

$T.nptr = (+)$
$E.nptr = (+)$



- 7.1 属性文法及其计算
  - 7.1.1 属性翻译文法
  - 7.1.2 综合属性的自下而上计算
  - 7.1.3 继承属性的自上而下计算
  - 7.1.4 依赖图
  - 7.1.5 树遍历的计算方法
  - 7.1.6 一遍扫描的处理方法
- 7.2 S-属性文法
  - 7.2.1 S-属性文法的自下而上计算
  - 7.2.2 构造表达式抽象语法树
  - 7.2.3 NFA箭弧单符化
- 7.3 L-属性文法
  - 7.3.1 翻译模式
  - 7.3.2 L-属性文法自上而下计算
  - 7.3.3 L-属性文法自下而上计算
  - 7.3.4 综合属性代替继承属性
- 7.4 声明语句的翻译
  - 7.4.1 Pascal风格过程内声明语句
  - 7.4.2 Pascal风格过程定义与声明语句
  - 7.4.3 Pascal风格数组声明
  - 7.4.4 Pascal风格结构体声明
  - 7.4.5 C风格函数定义与声明语句
- 7.5 表达式与赋值语句的翻译
  - 7.5.1 算术表达式与赋值语句
  - 7.5.2 Pascal风格数组的引用
  - 7.5.3 C风格数组的引用
  - 7.5.4 结构体的引用
  - 7.5.5 作为逻辑运算的布尔表达式
  - 7.5.6 地址和指针的引用
- 7.6 控制语句的翻译
  - 7.6.1 真假出口链
  - 7.6.2 四元式链操作
  - 7.6.3 作为条件控制的布尔表达式
  - 7.6.4 if和while语句
  - 7.6.5 C风格for语句
  - 7.6.6 Matlab风格for语句
  - 7.6.7 标号-goto语句
  - 7.6.8 switch语句
  - 7.6.9 break和continue语句
  - 7.6.10 三元运算符
  - 7.6.11 关系运算符的结合
- 7.7 过程调用的翻译
  - 7.7.1 过程的开始与结束标记
  - 7.7.2 返回语句
  - 7.7.3 过程调用
- 7.8 类型检查
  - 7.8.1 类型表达式
  - 7.8.2 类型检查
  - 7.8.3 隐式转换
  - 7.8.4 显式转换

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

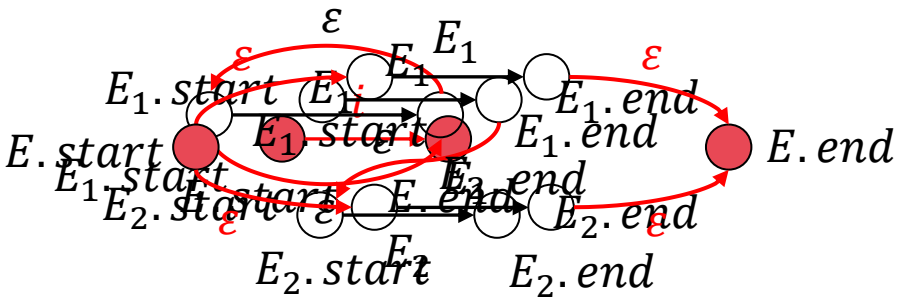
NFA箭弧单符化

$E \to E_1 \mid T$	$E.start = newState, E.end = newState$ $addArc(E.start, \varepsilon, E_1.start); addArc(E.start, \varepsilon, T.start);$ $addArc(E_1.end, \varepsilon, E.end); addArc(T.end, \varepsilon, E.end)$
$E \to T$	$E.start = T.start, E.end = T.end$
$T \to T_1 P$	$T.start = T_1.start, T.end = P.end$ $addArc(T_1.end, \varepsilon, P.start)$
$T \to P$	$T.start = P.start, T.end = P.end$
$P \to P_1 *$	$P.start = P_1.start, P.end = P_1.end$ $addArc(P.start, \varepsilon, P.end); addArc(P.end, \varepsilon, P.start)$
$P \to F$	$P.start = F.start, P.end = F.end$
$F \to (E)$	$F.start = E.start, F.end = E.end$
$F \to i$	$F.start = newState, F.end = newState, addArc(F.start, i, F.end)$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

NFA箭弧单符化

$E \to E_1 \mid E_2$	$E.start = newState, E.end = newState$ $addArc(E.start, \varepsilon, E_1.start); addArc(E.start, \varepsilon, E_2.start);$ $addArc(E_1.end, \varepsilon, E.end); addArc(E_2.end, \varepsilon, E.end)$
$E \to E_1 E_2$	$E.start = E_1.start, E.end = E_2.end$ $addArc(E_1.end, \varepsilon, E_2.start)$
$E \to E_1 *$	$E.start = E_1.start, E.end = E_1.end$ $addArc(E.start, \varepsilon, E.end); addArc(E.end, \varepsilon, E.start)$
$E \to (E_1)$	$E.start = E_1.start, E.end = E_1.end$
$E \to i$	$E.start = newState, E.end = newState, addArc(E.start, i, E.end)$



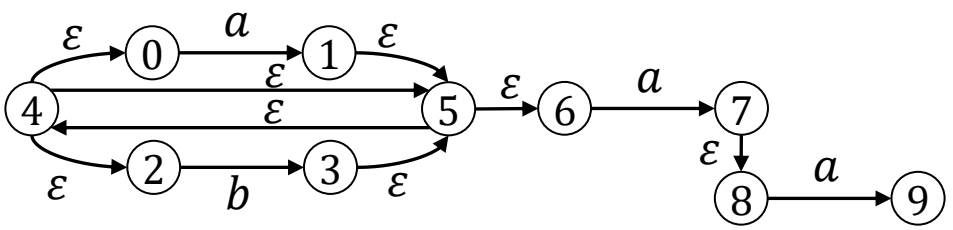
山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】字符串$(a|b)*aa$转为NFA，使弧上为$V_N \cup V_T \cup \{\varepsilon\}$

$E \to E_1 E_2$	$E.start = E_1.start, E.end = E_2.end$ $addArc(E_1.end, \varepsilon, E_2.start)$
	$addArc(E_1.end, \varepsilon, E.end); addArc(E_2.end, \varepsilon, E.end)$

步骤	文法符号栈	输入串
1	#	$(a\|b)*aa\#$
2	$\#(a$	$\|b)*aa\#$
3	$\#(E$	$\|b)*aa\#$
4	$\#(E\|b$	$)*aa\#$
5	$\#(E\|E$	$)*aa\#$
6	$\#(E$	$)*aa\#$
7	$\#(E)$	$*aa\#$
8	$\#E$	$*aa\#$
9	$\#E*$	$aa\#$
10	$\#E$	$aa\#$
11	$\#Ea$	$a\#$
12	$\#EE$	$a\#$

步骤	文法符号栈	输入串
13	$\#E$	$a\#$
14	$\#Ea$	#
15	$\#EE$	#
16	$\#E$	#



$E.start = 8, E.end = 9$
$E.start = 4, E.end = 9$

栈属性

- 7.1 属性文法及其计算
  - 7.1.1 属性翻译文法
  - 7.1.2 综合属性的自下而上计算
  - 7.1.3 继承属性的自上而下计算
  - 7.1.4 依赖图
  - 7.1.5 树遍历的计算方法
  - 7.1.6 一遍扫描的处理方法
- 7.2 S-属性文法
  - 7.2.1 S-属性文法的自下而上计算
  - 7.2.2 构造表达式抽象语法树
  - 7.2.3 NFA箭弧单符化
- 7.3 L-属性文法
  - 7.3.1 翻译模式
  - 7.3.2 L-属性文法自上而下计算
  - 7.3.3 L-属性文法自下而上计算
  - 7.3.4 综合属性代替继承属性
- 7.4 声明语句的翻译
  - 7.4.1 Pascal风格过程内声明语句
  - 7.4.2 Pascal风格过程定义与声明语句
  - 7.4.3 Pascal风格数组声明
  - 7.4.4 Pascal风格结构体声明
  - 7.4.5 C风格函数定义与声明语句
- 7.5 表达式与赋值语句的翻译
  - 7.5.1 算术表达式与赋值语句
  - 7.5.2 Pascal风格数组的引用
  - 7.5.3 C风格数组的引用
  - 7.5.4 结构体的引用
  - 7.5.5 作为逻辑运算的布尔表达式
  - 7.5.6 地址和指针的引用
- 7.6 控制语句的翻译
  - 7.6.1 真假出口链
  - 7.6.2 四元式链操作
  - 7.6.3 作为条件控制的布尔表达式
  - 7.6.4 if和while语句
  - 7.6.5 C风格for语句
  - 7.6.6 Matlab风格for语句
  - 7.6.7 标号-goto语句
  - 7.6.8 switch语句
  - 7.6.9 break和continue语句
  - 7.6.10 三元运算符
  - 7.6.11 关系运算符的结合
- 7.7 过程调用的翻译
  - 7.7.1 过程的开始与结束标记
  - 7.7.2 返回语句
  - 7.7.3 过程调用
- 7.8 类型检查
  - 7.8.1 类型表达式
  - 7.8.2 类型检查
  - 7.8.3 隐式转换
  - 7.8.4 显式转换

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

L-属性文法

- L-属性文法：如果对于每个产生式$A \to X_1X_2 \ldots X_n$，其语义规则中的每个属性或者是综合属性，或者是$X_i(1 \le i \le n)$的一个继承属性且这个继承属性仅依赖于：
  - 产生式右部X_i的左边符号$X_1, X_2, \ldots, X_{i-1}$的属性；
  - 产生式左部A的继承属性。
- 说明
  - S-属性文法一定是L-属性文法，因为L-属性文法定义中未对综合属性进行限制
  - L-属性文法可以一次遍历就计算出所有属性值。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

- 属性文法：为产生式配上语义动作。
- 翻译模式（Translation Schemes）：把文法符号相关的属性和语义规则（语义动作），用花括号{}括起来，插入到产生式右部的合适位置上。

【例】带加减的中缀表达式翻译成后缀表达式

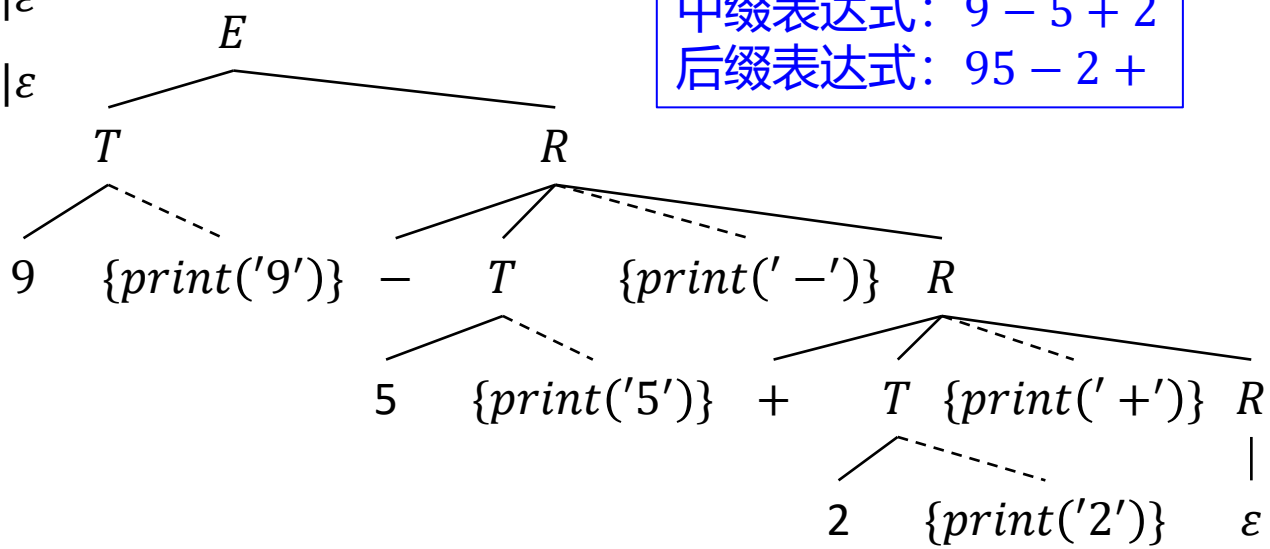
$E \to TR$

$R \to +T\{print('+')\}R_1|\varepsilon$

$R \to -T\{print('-')\}R_1|\varepsilon$

$T \to i\{print(i.val)\}$

中缀表达式：$9-5+2$
后缀表达式：$95-2+$



山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式设计

- 只有综合属性：语义动作放到产生式右部末尾。

【例】综合属性的翻译模式

$$T \rightarrow T_1 * F\{T.val = T_1.val * F.val\}$$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式设计

- 既有综合属性又有继承属性：

① 产生式右部符号的继承属性，必须在这个符号以前的动作中计算出来；

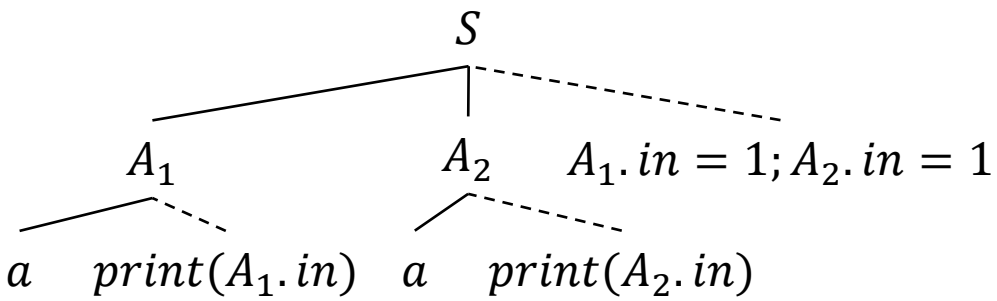
② 一个动作不能引用这个动作右边符号的综合属性；

③ 产生式左部的V_N的综合属性，只有在它所引用的所有属性都计算出来以后才能计算（放到右边末尾）。

【例】不满足条件①的翻译模式

$S \to A_1 A_2 \{A_1.in = 1; A_2.in = 1\}$

$A \to a\{print(A.in)\}$



山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式设计

- 既有综合属性又有继承属性：

① 产生式右部符号的继承属性，必须在这个符号以前的动作中计算出来；

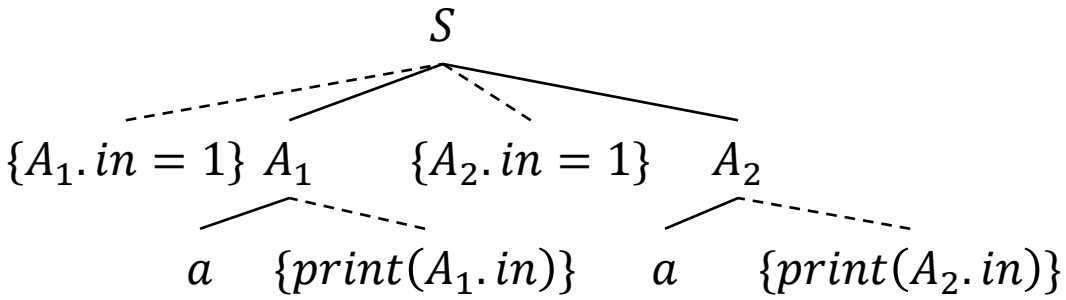
② 一个动作不能引用这个动作右边符号的综合属性；

③ 产生式左部的V_N的综合属性，只有在它所引用的所有属性都计算出来以后才能计算（放到右边末尾）。

【例6.17】满足条件①②③的翻译模式

$S \to \{A_1.in = 1\}A_1\{A_2.in = 1\}\ A_2$

$A \to a\{print(A.in)\}$



- 7.1 属性文法及其计算
  - 7.1.1 属性翻译文法
  - 7.1.2 综合属性的自下而上计算
  - 7.1.3 继承属性的自上而下计算
  - 7.1.4 依赖图
  - 7.1.5 树遍历的计算方法
  - 7.1.6 一遍扫描的处理方法
- 7.2 S-属性文法
  - 7.2.1 S-属性文法的自下而上计算
  - 7.2.2 构造表达式抽象语法树
  - 7.2.3 NFA箭弧单符化
- 7.3 L-属性文法
  - 7.3.1 翻译模式
  - 7.3.2 L-属性文法自上而下计算
  - 7.3.3 L-属性文法自下而上计算
  - 7.3.4 综合属性代替继承属性
- 7.4 声明语句的翻译
  - 7.4.1 Pascal风格过程内声明语句
  - 7.4.2 Pascal风格过程定义与声明语句
  - 7.4.3 Pascal风格数组声明
  - 7.4.4 Pascal风格结构体声明
  - 7.4.5 C风格函数定义与声明语句
- 7.5 表达式与赋值语句的翻译
  - 7.5.1 算术表达式与赋值语句
  - 7.5.2 Pascal风格数组的引用
  - 7.5.3 C风格数组的引用
  - 7.5.4 结构体的引用
  - 7.5.5 作为逻辑运算的布尔表达式
  - 7.5.6 地址和指针的引用
- 7.6 控制语句的翻译
  - 7.6.1 真假出口链
  - 7.6.2 四元式链操作
  - 7.6.3 作为条件控制的布尔表达式
  - 7.6.4 if和while语句
  - 7.6.5 C风格for语句
  - 7.6.6 Matlab风格for语句
  - 7.6.7 标号-goto语句
  - 7.6.8 switch语句
  - 7.6.9 break和continue语句
  - 7.6.10 三元运算符
  - 7.6.11 关系运算符的结合
- 7.7 过程调用的翻译
  - 7.7.1 过程的开始与结束标记
  - 7.7.2 返回语句
  - 7.7.3 过程调用
- 7.8 类型检查
  - 7.8.1 类型表达式
  - 7.8.2 类型检查
  - 7.8.3 隐式转换
  - 7.8.4 显式转换

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

消除翻译模式的左递归

- 带左递归的翻译模式

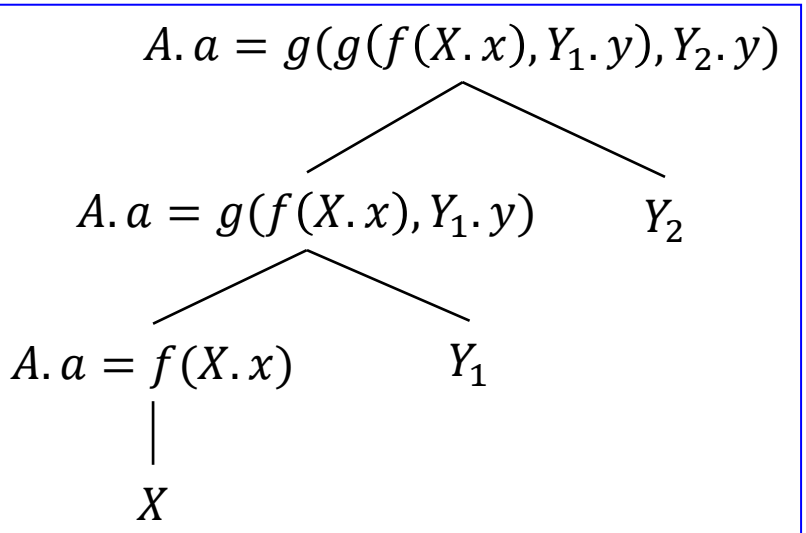
$A \to A_1 Y\{A.a = g(A_1.a, Y.y)\}$

$A \to X\{A.a = f(X.x)\}$

- 消除文法左递归

$A \to XR$

$R \to YR|\varepsilon$



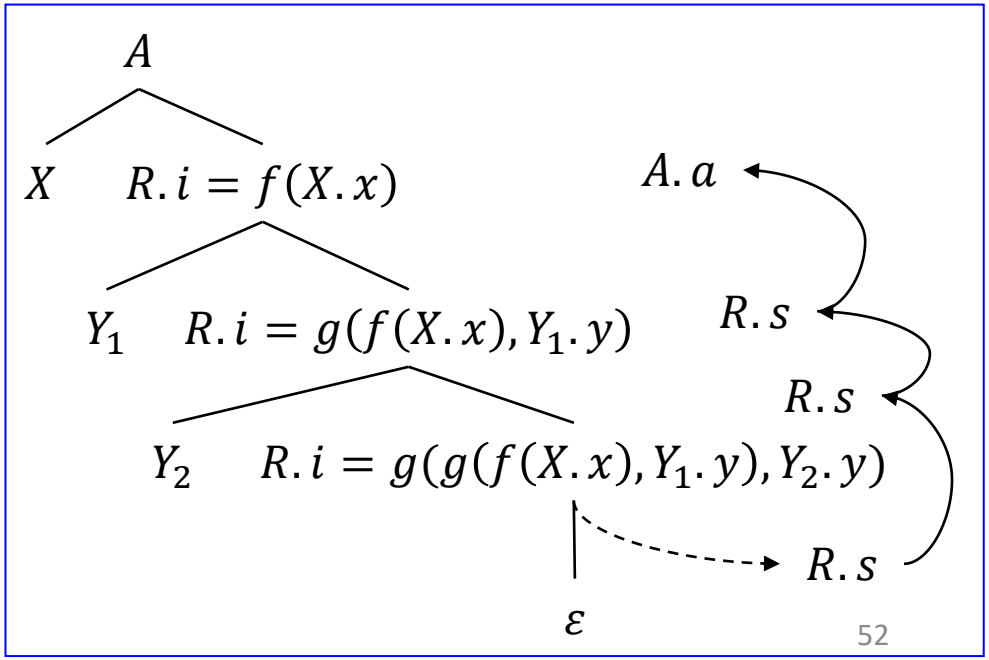
- 考虑语义动作的翻译模式

$A \to X\ \{R.i = f(X.x)\}R\ \{A.a = R.s\}$

$R \to Y\ \{R_1.i = g(R.i, Y.y)\}R_1\ \{R.s = R_1.s\}$

$R \to \varepsilon\{R.s = R.i\}$

- 以XYY为例说明其计算过程



山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

L-属性文法自下而上计算

- 前述自下而上的翻译方法中，要求语义动作放在产生式末尾
  - 解决办法：引入空符号产生式（只需要处理动作，不需要处理属性计算）。

$A \to \alpha\{Action\}\beta$，其中$\beta \in (V_N \cup V_T)^*, \beta \neq \varepsilon$

修改为：

① $A \to \alpha M\beta$

② $M \to \varepsilon\{Action\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

L-属性文法自下而上计算

【例】消除如下翻译模式中产生式中间的动作

$E \to TR$

$R \to +T\{print('+')\}R \mid -T\{print('-')\}R \mid \varepsilon$

$T \to num\{print(num.val)\}$

【解】

$E \to TR$

$R \to +TMR \mid -TNR \mid \varepsilon$

$T \to num\{print(num.val)\}$

$M \to \varepsilon\{print('+')\}$

$N \to \varepsilon\{print('-')\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

分析栈中的继承属性

- 【例】翻译模式

$D \rightarrow T \ \{L.in = T.type\}$

$\quad\quad L$

$T \rightarrow int \ \{T.type = integer\}$

$T \rightarrow real \ \{T.type = real\}$

$L \rightarrow \quad\quad \{L_1.in = L.in\}$

$\quad\quad L_1, id \ \{addType(id.entry, L.in)\}$

$L \rightarrow id \ \{addType(id.entry, L.in)\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

(1) $D \rightarrow T \{L.in = T.type\}$

L

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

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

(4) $L \rightarrow \quad \{L_1.in = L.in\}$

$L_1, id \{addType(id.entry, L.in)\}$

(5) $L \rightarrow id \{addType(id.entry, L.in)\}$

	Action					Goto		
状态	int	real	id	,	#	D	T	L
0	S_3	S_4				1	2	
1					acc			
2			S_6					5
3			r_2					
4			r_3					
5				S_7	r_1			
6				r_5	r_5			
7			S_8					
8				r_4	r_4			

序号	状态栈	符号栈	输入串
1	0	#	$int\ p,q,r\#$
2	03	$\#int$	$p,q,r\#$
3	02	$\#T$	$p,q,r\#$
4	026	$\#Tp$	$,q,r\#$
5	025	$\#TL$	$,q,r\#$
6	0257	$\#TL,$	$q,r\#$
7	02578	$\#TL,q$	$,r\#$
8	025	$\#TL$	$,r\#$

序号	状态栈	符号栈	输入串
9	0257	$\#TL,$	$r\#$
10	02578	$\#TL,r$	#
11	025	$\#TL$	#
12	01	$\#D$	#
13	01	acc	#

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

(1) $D \rightarrow T\ \{L.in = T.type\}$

L

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

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

(4) $L \rightarrow \quad \{L_1.in = L.in\}$

$L_1, id\ \{addType(id.entry, L.in)\}$

(5) $L \rightarrow id\ \{addType(id.entry, L.in)\}$

序号	状态栈	符号栈	输入串
1	0	#	$int\ p,q,r\#$
2	03	$\#int$	$p,q,r\#$
3	02	$\#T$	$p,q,r\#$
4	026	$\#Tp$	$,q,r\#$
5	025	$\#TL$	$,q,r\#$
6	0257	$\#TL,$	$q,r\#$
7	02578	$\#TL,q$	$,r\#$
8	025	$\#TL$	$,r\#$

【方案1】使用L存储属性

① $D \rightarrow TL$

② $T \rightarrow int\ \{val[ntop] = int\}$

③ $T \rightarrow real\ \{val[ntop] = real\}$

④ $L \rightarrow L_1, id\ \{addType(val[top], val[top-2];\ val[ntop] = val[top-2]\}$

⑤ $L \rightarrow id\ \{addType(val[top], val[top-1];\ val[ntop] = val[top-1]\}$

序号	状态栈	符号栈	输入串
9	0257	$\#TL,$	$r\#$
10	02578	$\#TL,r$	#
11	025	$\#TL$	#
12	01	$\#D$	#
13	01	acc	#

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

(1) $D \rightarrow T \{L.in = T.type\}$

L

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

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

(4) $L \rightarrow \quad \{L_1.in = L.in\}$

$L_1, id \{addType(id.entry, L.in)\}$

(5) $L \rightarrow id \{addType(id.entry, L.in)\}$

【方案2】不使用L存储属性

① $D \rightarrow TL$

② $T \rightarrow int \{val[ntop] = int\}$

③ $T \rightarrow real \{val[ntop] = real\}$

④ $L \rightarrow L_1, id \{addType(val[top], val[top-3]\}$

⑤ $L \rightarrow id \{addType(val[top], val[top-1]\}$

序号	状态栈	符号栈	输入串
1	0	#	$int\ p,q,r\#$
2	03	$\#int$	$p,q,r\#$
3	02	$\#T$	$p,q,r\#$
4	026	$\#Tp$	$,q,r\#$
5	025	$\#TL$	$,q,r\#$
6	0257	$\#TL,$	$q,r\#$
7	02578	$\#TL,q$	$,r\#$
8	025	$\#TL$	$,r\#$

序号	状态栈	符号栈	输入串
9	0257	$\#TL,$	$r\#$
10	02578	$\#TL,r$	#
11	025	$\#TL$	#
12	01	$\#D$	#
13	01	acc	#

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

模拟继承属性的计算

- 只有根据文法预知属性值在栈中存放位置时，才能有效地在分析栈中处理属性值。
  - 有时位置可能存在冲突。

【例】A和C之间可能有B也可能没有B，当通过$C \to c$进行归约时，$C.i$可能在$val[top-1]$处，也可能在$val[top-2]$处：

$S \to aAC \{C.i = A.s\}$

$S \to bABC \{C.i = A.s\}$

$C \to c \{C.s = g(C.i)\}$

【修改】$M \to \varepsilon$时从$val[top-1]$处取到$A.s$，$C \to c$归约时，$C.i$总在$val[top-1]$处。

$S \to aAC \{C.i = A.s\}$

$S \to bABMC \{M.i = A.s; C.i = M.s\}$

$C \to c \{C.s = g(C.i)\}$

$M \to \varepsilon \{M.s = M.i\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

综合属性代替继承属性

- 改变基础文法可以避免继承属性。

【例】Pascal说明语句如m, n:integer，标识符由L产生类型，但类型不在L子树中：

$D \rightarrow L:T$

$T \rightarrow integer \mid char$

$L \rightarrow L, id \mid id$

序号	符号栈	输入串
1	#	$p,q,r:int\#$
2	$\#p,q,r:int$	#
3	$\#p,q,r:T$	#
4	$\#p,q,rL$	#
5	$\#p,qL$	#
6	$\#pL$	#
7	$\#D$	#

【修改】 重构文法，使类型作为标识符表的最后一个元素：

$D \rightarrow id\ L\ \{setType(id, L.type)\}$

$L \rightarrow, id\ L_1 \{L.type = L_1.type; setType(id, L.type)\}$

$L \rightarrow : T\ \{L.type = T.type\}$

$T \rightarrow integer\ \{T.type = int\}$

$T \rightarrow char\ \{T.type = char\}$

第七章作业

【作业7-1】文法$G[E]$

$$E \rightarrow E + T \mid T$$

$$T \rightarrow num.num \mid num$$

(1) 给出确定每个子表达式结果类型的属性文法。

(2) 扩充(1)的属性文法，使其把表达式翻译成后缀式，同时也能确定结果类型。注意实数和整数相加得实数，应保证后缀式中两个加数是同型的，可以采用int2real把整型转为实型。

- 7.1 属性文法及其计算
  - 7.1.1 属性翻译文法
  - 7.1.2 综合属性的自下而上计算
  - 7.1.3 继承属性的自上而下计算
  - 7.1.4 依赖图
  - 7.1.5 树遍历的计算方法
  - 7.1.6 一遍扫描的处理方法
- 7.2 S-属性文法
  - 7.2.1 S-属性文法的自下而上计算
  - 7.2.2 构造表达式抽象语法树
  - 7.2.3 NFA箭弧单符化
- 7.3 L-属性文法
  - 7.3.1 翻译模式
  - 7.3.2 L-属性文法自上而下计算
  - 7.3.3 L-属性文法自下而上计算
  - 7.3.4 综合属性代替继承属性
- 7.4 声明语句的翻译
  - 7.4.1 Pascal风格过程内声明语句
  - 7.4.2 Pascal风格过程定义与声明语句
  - 7.4.3 Pascal风格数组声明
  - 7.4.4 Pascal风格结构体声明
  - 7.4.5 C风格函数定义与声明语句
- 7.5 表达式与赋值语句的翻译
  - 7.5.1 算术表达式与赋值语句
  - 7.5.2 Pascal风格数组的引用
  - 7.5.3 C风格数组的引用
  - 7.5.4 结构体的引用
  - 7.5.5 作为逻辑运算的布尔表达式
  - 7.5.6 地址和指针的引用
- 7.6 控制语句的翻译
  - 7.6.1 真假出口链
  - 7.6.2 四元式链操作
  - 7.6.3 作为条件控制的布尔表达式
  - 7.6.4 if和while语句
  - 7.6.5 C风格for语句
  - 7.6.6 Matlab风格for语句
  - 7.6.7 标号-goto语句
  - 7.6.8 switch语句
  - 7.6.9 break和continue语句
  - 7.6.10 三元运算符
  - 7.6.11 关系运算符的结合
- 7.7 过程调用的翻译
  - 7.7.1 过程的开始与结束标记
  - 7.7.2 返回语句
  - 7.7.3 过程调用
- 7.8 类型检查
  - 7.8.1 类型表达式
  - 7.8.2 类型检查
  - 7.8.3 隐式转换
  - 7.8.4 显式转换

属性与元操作

- 过程内声明语句形如$id_1, id_2, \ldots, id_n$ ： $type$，定义如下属性和元操作：
  - $name$：变量 id 的名字，其值来自词法分析器。
  - $type$：数据类型，基本类型取值为int或$real$，用$pointer(type)$表示$type$类型的指针。
  - $width$：字宽，其中int占4字节， $real$占8字节，指针占4字节。
  - $offset$：是一个全局变量，记录变量运行时在过程中的地址偏移量；开始时置0，每安排一个变量，增加相应字宽。
  - $log(name, category, type, width, offset)$：将名字、类别、数据类型、字宽和运行时在过程中的地址偏移量登记到符号表。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(1) $P \rightarrow MD$

(2) $M \rightarrow \varepsilon$ $\{offset = 0;\}$

(3) $D \rightarrow D;D$

(4) $D \rightarrow id\ L$ $\{\log(id.name, variable, L.type, L.width, offset)$;

$offset \mathrel{+}= L.width;\}$

(5) $L \rightarrow, id\ L_1$ $\{\log(id.name, variable, L_1.type, L_1.width, offset)$;

$offset \mathrel{+}= L_1.width;$

$L.type = L_1.type; L.width = L_1.width\}$

(6) $L \rightarrow : T$ $\{L.type = T.type; L.width = T.width;\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(7) $T \rightarrow integer$ $\{T.type = int;\ T.width = 4\}$

(8) $T \rightarrow real$ $\{T.type = real;\ T.width = 8\}$

(9) $T \rightarrow char$ $\{T.type = char;\ T.width = 1\}$

(10) $T \rightarrow \hat{}T_1$ $\{T.type = pointer(T_1.type); T.width = 4\}$

翻译示例

【例】输入串：

#		$offset = 24$

$P \rightarrow MD$	
$offset \mathrel{+}= L.width;\}$	

$T\{type = real, width = 8\}$
$L\{type = real, width = 8\}$
b
$L\{type = real, width = 8\}$
D
;
D
P
#

名字	类别	类型	字宽	偏移量
j	variable	int	4	0
i	variable	int	4	4
b	variable	real	8	8
a	variable	real	8	16

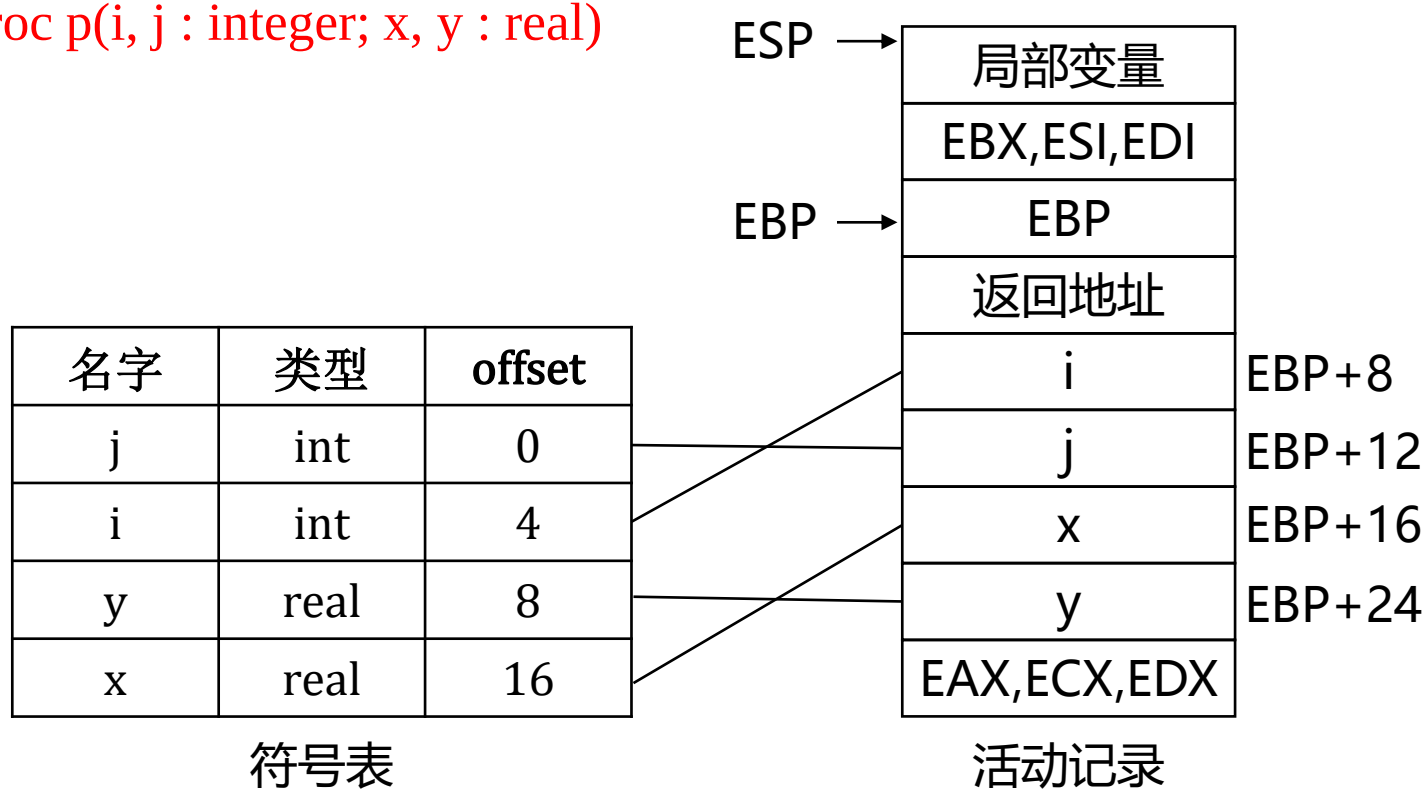
成功

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

形参反序填表存在的问题

- 声明变量反序填表导致运行时地址无法计算

➢ proc p(i, j : integer; x, y : real)



属性与元操作

- 定义如下属性和元操作：
  - $tblptr$：即table pointer的简写，为符号表栈，存放指向符号表的指针。
  - $offset$：是一个整数栈，存放局部变量或形参变量的偏移地址。
  - $varstack$：即variable stack，变量名栈，栈结点是字符串类型，varstack.isEmpty属性标记该栈是否为空。
  - $category$：是一个枚举类型，表示变量类别，普通变量用variable表示，形式参数用formal表示。
  - $proc$：记录过程的名字。
  - $makeTable(previous)$：创建一张新符号表，并返回指向新表的指针；参数previous指向当前栈顶符号表，也就是本过程父过程的符号表。
  - $log(table, name, category, type, width, offset)$：将名字、类别、数据类型、字宽和运行时在过程中的地址偏移量登记到符号表table。

属性与元操作

- 定义如下属性和元操作：
  - $logSize(table, size)$：记录指针table指向的符号表，其占用的总字节数size。
  - $logProc(table, name, newtable, size)$：在指针table指向的符号表中，为名字为name的过程建立一个新项；参数newtable指向过程name的符号表，size为该过程中变量占用的总字节数。
  - logSize()和logProc()中的size，写入到符号表的表头，不要写入与变量共用的width属性内，过程的width属性将来用作保存返回值宽度，当前该属性初始化为0。

翻译模式

(1) $P \rightarrow M\ Dlist$ $\{logSize(tblptr.top, width); tblptr.pop(); offset.pop();\}$

(2) $M \rightarrow \varepsilon$ $\{t = makeTable(null); category = variabe;$

$tblptr.push(t); offset.push(0);\}$

(3) $Dlist \rightarrow D$

(4) $Dlist \rightarrow Dlist; D$

(5) $F \rightarrow proc\ id($ $\{t = makeTable(tblptr.top);\ category = formal;$

$F.proc = id.name;\ tblptr.push(t);\ offset.push(0);\}$

(6) $D \rightarrow FVlist); NDlist; S$ $\{t = tblptr.pop(); w = offset.pop();$

$logProc(tblptr.top,\ F.proc,\ t,\ w);\}$

(7) $D \rightarrow FVlist); NS$ $\{t = tblptr.top(); w = offset.pop();$

$logProc(tblptr.top,\ F.proc,\ t,\ w);\}$

(8) $N \rightarrow \varepsilon$ $\{offset.top = 0;\ category = variable;\}$

翻译模式

(9) $D \rightarrow Vlist$

(10) $Vlist \rightarrow V$

(11) $Vlist \rightarrow Vlist; V$

(12) $V \rightarrow \varepsilon$

(13) $V \rightarrow id\ L$ {log($tblptr.top,\ id.name,\ category,\ L.type, L.width, offset.top$);

$offset.top \mathrel{+}= L.width;$

while (! $varstack.isEmpty$) {

$v = varstack.pop();$

log($tblptr.top, v, category, L.type, L.width, offset.top$) ;

$offset.top \mathrel{+}= L.with;$

} }

(14) $L \rightarrow, id\ L_1$ $\{varstack.push(id.name); L.type = L_1.type; L.width = L_1.width;\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(15) $L \rightarrow : T$ $\{L.type = T.type; L.width = T.width;\}$

(16) $T \rightarrow integer$ $\{T.type = int;\ T.width = 4\}$

(17) $T \rightarrow real$ $\{T.type = real;\ T.width = 8\}$

(18) $T \rightarrow char$ $\{T.type = char;\ T.width = 1\}$

(19) $T \rightarrow \hat{}T_1$ $\{T.type = pointer(T_1.type); T.width = 4\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】

#

$(6)\ D \rightarrow FVlist);NDlist;S \quad \{t = tblptr.pop();w = offset.pop();$ $logProc(tblptr.top,\ F.proc,\ t,\ w);\}$
$T\{type = int, width = 4\}$
$L\{type = int, width = 4\}$
S
;
$Dlist$
N
)
$Vlist$
D
;
$Dlist$
M
#

$k.pop();$

$pe, L.width, offset.top);$

$category = variable$

$varstack = j$

$varstack = y$

$offset = 8$

$offset = 8$

名字	类别	类型	字宽	偏移量
a	variable	int	4	0
b	variable	int	4	4
sum	proc	-	-	-

名字	类别	类型	字宽	偏移量
x	formal	real	8	0
y	formal	real	8	8
i	variable	int	4	0
j	variable	int	4	4

- ✓ 过程的总字节数存入表头
- ✓ 后续过程略

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

数组声明语句形式

- Pascal数组声明：

$$c：array[0..3,0..3]\ of\ real;$$

$$b：array[-10..10]\ of\ integer;$$

$$a：array[1..3]\ of\ integer;$$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

数组分配空间计算

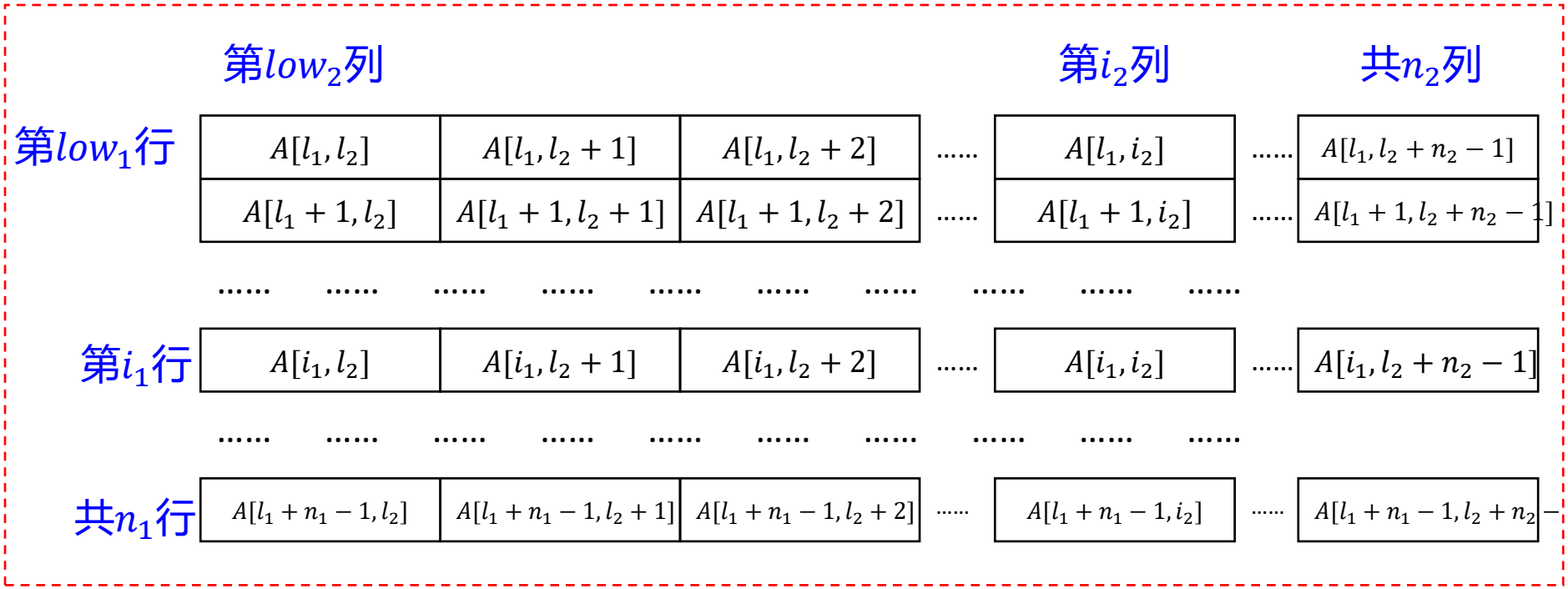
- 数组连续存储，一维数组$a[i]$地址为：$base + (i - low) \times w$
  - w：数组中每个元素的宽度；
  - low：数组下标下界；
  - $base$：分配给数组的相对地址，即$base$为A的第一个元素$A[low]$的相对地址。
- 整理为：$base + i \times w - low \times w$
  - 令$C = low \times w$，在处理数组声明时计算出来，存放到符号表中a的对应项中；
  - $a[i]$的相对地址计算变为：$base + i \times w - C$。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

数组元素的引用

- 行存储的二维数组$a[i_1, i_2]$地址：
  - $base + [(i_1 - low_1) \times n_2 + i_2 - low_2] \times w$

  $= base + (i_1 \times n_2 + i_2) \times w - (low_1 \times n_2 + low_2) \times w$



山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

数组元素的引用

- 行存储的二维数组$a[i_1, i_2]$地址：
  - $base + [(i_1 - low_1) \times n_2 + i_2 - low_2] \times w$

    $= base + (i_1 \times n_2 + i_2) \times w - (low_1 \times n_2 + low_2) \times w$
- 行存储的多维数组$A[i_1, i_2, \dots, i_k]$地址：
  - 基准地址：$base - ((\dots((low_1 \times n_2 + low_2) \times n_3) \dots) \times n_k + low_k) \times w$
  - 处理声明语句时，计算出“$base -$”后面的部分，作为C存储到符号表
  - 动态地址：$(((\dots(i_1 \times n_2 + i_2) \times n_3) \dots) \times n_k + i_k) \times w$
  - 静态地址递归计算：$c_1 = low_1, c_2 = c_1 \times n_2 + low_2, \dots, c_m = c_{m-1} \times n_m + low_m$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

属性和元操作

- 定义如下属性和元操作：
  - dim，维度，即当前为第几个维度。
  - $static$，数组静态地址。
  - $number$，数组总元素数，递归计算得到，用于计算数组占用总字节数。
  - $dope$，指dope vector，用来记录上下限和该维度元素数的内情向量表。
  - $log(\cdot)$，在符号表中登记符号，与前述操作相比增加了子表dope、静态地址static两个参数，子表非空时该项符号设置指向子表的指针。
  - $makeDope(dim, low, high, number)$，构建数组内情向量表并返回，并登记如下信息：维度dim、下标下限low、下标上限high、该维度元素数number。
  - $logDope(dope, dim, low, high, number)$，在数组内情向量表dope中增加一条记录，并返回该子表。
  - $array(dope, type)$，由数组内情向量表、数据类型构造数组的类型表达式，并作为数据类型返回。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(13) $V \rightarrow id\ L$ {log($tblptr.top,\ id.name,\ category,\ L.type, L.width, offset.top,$

$L.dope, L.static$); $offset.top \mathrel{+}= L.width$;

while (! $varstack.isEmpty$) {

$v = varstack.pop()$;

log($tblptr.top, v, category, L.type, L.width, offset.top, L.dope, L.static$) ;

$offset.top \mathrel{+}= L.with$;

} }

(14) $L \rightarrow, id\ L_1$ {$varstack.push(id.name); L.type = L_1.type; L.width = L_1.width;$

$L.dope = L_1.dope; L.static = L_1.static;$}

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(15) $L \rightarrow : T$ $\{L.type = T.type; L.width = T.width;$
$L.dope = T.dope; L.static = T.static;\}$

(16) $T \rightarrow integer$ $\{T.type = integer;\ T.width = 4;$
$T.dope = null; T.static = 0;\}$

(17) $T \rightarrow real$ $\{T.type = real;\ T.width = 8;$
$T.dope = null; T.static = 0;\}$

(18) $T \rightarrow char$ $\{T.type = char;\ T.width = 1;$
$T.dope = null; T.static = 0;\}$

(19) $T \rightarrow \hat{}T_1$ $\{T.type = pointer(T_1.type); T.width = 4;$
$T.dope = null; T.static = 0;\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(20) $A \rightarrow array[num_1..num_2$ $\{A.dim = 1; n = num_2 - num_1 + 1;$

$A.dope = makeDope(1, num_1, num_2, n); A.static = num_1; A.number = n;\}$

(21) $A \rightarrow A_1, num_1..num_2$ $\{A.dim = A_1.dim + 1; n = num_2 - num_1 + 1;$

$A.dope = logDope(A_1.dope, A.dim, num_1, num_2, n);$

$A.static = A_1.static * n + num_1; A.number = A_1.number * n;\}$

(22) $T \rightarrow A]\ of\ T_1$ $\{T.type = array(A.dope, T_1.type);$

$T.width = A.number * T_1.width;$

$T.dope = A.dope; T.static = A.static;\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】 #

(16) $T \rightarrow integer$ $\{T.type = int;\ T.width = 4;\ T.dope = null;\ T.static = 0;\}$

$L.dope, L.static);\ offset.top \mathrel{+}= L.width;$ while $(!varstack.isEmpty)\ \{$
$v = varstack.pop(); \log(tblptr.top, v, category, L.type, L.width, offset.top,$
$L.dope, L.static);\ offset.top \mathrel{+}= L.with;\ \}\}$

$category = variable$

$offset = 160$

名字	类别	类型	字宽	偏移	静态
a	variable	array(.)	160	0	1
x	variable	int	4	160	0

维度	下限	上限	元素数
1	0	3	4
2	1	5	5

5
$T\{type = real, width = 8, static = 0\}$
$T\{type = int, width = 4, static = 0\}$
$L\{type = int, width = 4, static = 0\}$
D
;
$Dlist$
P
#

成功

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

结构体声明与元操作

- Pascal结构体声明：

```
1  type RecordName = record
2     field1 : type1;
3     field2 : type2;
4     ...
5     fieldn : typen;
6  end;
7  var r : RecordName;
```

- 定义如下元操作：
  - $lookup(table, name)$，从符号表table中查找名字为name 的符号记录并返回。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(23) $R \rightarrow type\ id = record\ \{t = makeTable(tblptr.top);$

$tblptr.push(t); offset.push(0);$

$category = recmem; R.name = id.name;\}$

(24) $D \rightarrow R\ Vlist; end$ $\{t = tblptr.pop(); w = offset.pop();$

$category = variable;$

$\log(tblptr.top, R.name, record, -, w, -, t, 0);\}$

(25) $T \rightarrow id$ $\{p = lookup(tblptr.top, id.name);$

if $p = null$ then Error;

else if $p.category \neq record$ then Error;

else {

$T.type = record(id.name); T.width = p.width;$

$T.dope = p.dope; T.static = 0;\}\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】

#

(25) $T \rightarrow id$ $\{p = lookup(tblptr.top, id.name);$
if $p = null$ then Error; else if $p.category \neq record$ then Error;
else { $T.type = record(id.name); T.width = p.width;$
$T.dope = p.dope; T.static = 0; \}\}$

$category = recmem$

$offset = 0$

20
$T\{type = char, width = 1, static = 0\}$
$T\{type = record(Stu), width = 21, static$
$L\{type = record(Stu), width = 21, static = 0\}$
D
;
$Dlist$
P
#

名字	类别	类型	字宽	偏移	静态
Stu	record	-	25	-	1
s	variable	rec(.)	25	0	0

名字	类别	类型	字宽	偏移	静态
Name	recmem	array(.)	21	0	0
Age	recmem	integer	4	21	0

维度	下限	上限	元素数
1	0	20	21

成功

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

C风格函数定义与声明语句

- C风格语句：

```
int a, b, c;
int x[2][3][4];
int y[2,3,4];
struct Stu {
        char Name[21];
        int Age;
};
struct Stu s;
float sum(float x, float y)       {
        return x + y;
}
```

C风格函数定义与声明语句

- C风格声明语句与Pascal另一个非常重要的不同点是，C风格变量声明语句不要求连续放在过程开头。
- C风格的变量声明有3种形式：
  - 形式参数：以逗号作为参数的分隔符，每个变量都需要指明数据类型，形式为：$type\ id(,type\ id)^*|\varepsilon$
  - 结构体成员：以分号作为成员结束符，每个变量都需要指明数据类型，形式为：$(type\ id;)^*$
  - 变量声明：以分号作为声明结束符，可以多个变量指明为同一数据类型，形式为：$(type\ id(,id)^*;)^*$
- 为了简化设计，突出本质问题，我们将其统一到变量声明的形式。

属性和元操作

- 定义如下属性和元操作：
  - $log(table, name, category, type, width, offset)$：在指针table指向的符号表中，为名字为name的变量建立一个新项。
  - $logProc(table, name, type, newtable, width, size)$：在指针table指向的符号表中，为名字为name的过程建立一个新项。
  - $makeDope(dim, num)$：创建数组内情向量表，其中只包含一条记录，该记录包括维度dim和元素数num。
  - $logDope(dope, dim, num)$：在数组内情向量表dope中增加一条记录，该记录包括维度dim和元素数num。
  - $set(pvar, type, width)$：针对符号表中的一个符号记录pvar，修改其数据类型为type，修改其字宽为width。
  - $struct(name)$：结构体数据类型，结构体的名字为name。
  - $array(dope, type)$：使用内情向量表和数据类型构造数组的形式化描述。

翻译模式

(1) $P \rightarrow UD$ $\{t = tblptr.pop(); w = offset.pop(); logSize(t, w); \}$

(2) $U \rightarrow \varepsilon$ $\{t = makeTable(null); tblptr.push(t);$

$offset.push(0); category = variable; \}$

(3) $D \rightarrow G$ // G可以为变量声明、函数定义、结构体定义

(4) $D \rightarrow DG$

(5) $G \rightarrow M$

(6) $M \rightarrow V;$ // V是变量声明语句

(7) $M \rightarrow MV;$

(8) $S \rightarrow \{Slist\}$ // 函数内语句

(9) $Slist \rightarrow S$

(10) $Slist \rightarrow Slist\ S$

(11) $S \rightarrow M$ // 只定义了声明语句

翻译模式

(12) $F \rightarrow T\ id($ $\{t = makeTable(tblptr.top);$

$tblptr.push(t);\ offset.push(0);$

$category = formal; F.name = id.name;$

$F.type = T.type; F.width = T.width;\}$

(13) $G \rightarrow FM)NS$ $\{t = tblptr.pop(); w = offset.pop();$

$logProc(tblptr.top, F.name, F.type, t, F.width, w);\}$

(14) $N \rightarrow \varepsilon$ $\{offset.top = 0; category = variable;\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(15) $V \rightarrow T\ id$ $\{\log(tblptr.top, id.name, category, T.type, T.width, offset.top);$

$offset.top \mathrel{+}= T.width; V.name = id.name;$

$V.type = T.type; V.width = T.width;\}$

(16) $V \rightarrow V_1, id$ $\{\log(tblptr.top, id.name, category, V_1.type, V_1.width, offset.top);$

$offset.top \mathrel{+}= V_1.width; V.name = id.name;$

$V.type = V_1.type; V.width = V_1.width;\}$

(17) $V \rightarrow \varepsilon$ $\{V.type = void; V.width = 0; V.name = null;\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(18) $T \rightarrow void$ $\{T.type = void; T.width = 0;\}$

(19) $T \rightarrow int$ $\{T.type = integer; T.width = 4;\}$

(20) $T \rightarrow float$ $\{T.type = float; T.width = 4;\}$

(21) $T \rightarrow double$ $\{T.type = double; T.width = 8;\}$

(22) $T \rightarrow char$ $\{T.type = char; T.width = 1;\}$

(23) $T \rightarrow T_1 *$ $\{T.type = pointer(T_1.type); T.width = 4;\}$

翻译模式

(24) $A \rightarrow V[num$ $\{A.dim = 1; A.dope = makeDope(1, num); A.number = num;$

$A.name = V.name; A.type = V.type; A.width = V.width; \}$

(25) $A \rightarrow A_1, num$ $\{A.dim = A_1.dim + 1; A.dope = logDope(A_1.dope, A.dim, num);$

$A.number = A_1.number * num; A.name = A_1.name;$

$A.type = A_1.type; A.width = A_1.width; \}$

(26) $V \rightarrow A]$ $\{V.type = A.type; V.width = A.width; V.name = A.name;$

$w = A.number * A.width; p = lookup(tblptr.top, A.name);$

$set(p, array(A.\mathrm{dope}, A.type), w); offset.top \mathrel{+}= w - A.width; \}$

翻译模式

(27) $R \rightarrow struct\ id\{$ $\{t = makeTabe(tblptr.top); tblptr.push(t);$

$R.name = id.name; offset.push(0); category = strumem;\}$

(28) $G \rightarrow RM\};$ $\{t = tblptr.pop(); w = offset.pop();$

$\log(tblptr.top, R.name, struct, -, w, -);\}$

(29) $T \rightarrow struct\ id$ $\{p = lookup(tblptr.top, id.name);$

if $(p = null \vee p.category \neq struct)$ Error;

else $\{T.type = struct(id.name); T.width = p.width;\}\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

8.4.5 C风格函数定义与声明语句

【例】

```
struct Point
{
        double Coor[10, 2];
        int Num;
};
void Fun(int i, j; int k;)
{
        struct Point p;
}
```

【例】 #

(4) $D \rightarrow DG$ (1) $P \rightarrow UD$ $\{t=tblptr.pop();w=offset.pop();logSize(t,w);\}$

$logProc(tblptr.top,F.name,F.type,t,F.width,w);\}$

else $\{T.type = struct(id.name);T.width = p.width;\}\}$

$category = variable$

$offset = 164$

$offset = 0$

名字	类别	类型	字宽	偏移
Point	struct		164	
Fun	proc	void	0	

名字	类别	类型	字宽	偏移
Coor	strumem	array(.)	160	0
Num	strumem	integer	4	160

维度	元素数
1	10
2	2

名字	类别	类型	字宽	偏移
i	formal	integer	4	0
j	formal	integer	4	4
k	formal	integer	4	8
p	variable	struct(.)	164	0

成功

}
$Slist$
S
N
)
M
G
D
P
#

属性和元操作

- 简单算术表达式及赋值语句的操作：
  - nxq：即next quadruplet，是一个整数，指向将要生成但尚未生成的四元式；每生成一个四元式，nxq自动加1。
  - val：是一个属性，存放文法符号的值。在翻译模式中，文法符号的值指变量在符号表的指针或者常量的值，但变量在书写时也常用变量的名字表示。
  - $id.name$： id是一个变量，其属性$name$是该变量的名字，来自词法分析器。
  - $num.value$： num是一个常量，其属性$value$是该常量的值，来自词法分析器。
  - $lookup(table,name)$：从符号表$table$开始逐级向上查找名字$name$，找到返回，否则返回$null$。
  - $newTemp$：生成一个临时变量，临时变量的名字用“$”加一个数字表示，根据操作数推断数据类型，并存入符号表，$offset$置为-1表示不分配内存空间。
  - $gen(\theta,arg1,arg2,result)$：生成四元式$(\theta,arg1,arg2,result)$。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

$S \rightarrow A$	
$A \rightarrow id = E$	$\{p = lookup(tblptr.top, id.name);$ $if\ p \neq null\ then\ gen(=, E.val, -, p);\ else\ error;\}$
$E \rightarrow E_1 + E_2$	$\{E.val = newTemp; gen(+, E_1.val, E_2.val, E.val)\}$
$E \rightarrow E_1 - E_2$	$\{E.val = newTemp; gen(-, E_1.val, E_2.val, E.val)\}$
$E \rightarrow E_1 * E_2$	$\{E.val = newTemp; gen(*, E_1.val, E_2.val, E.val)\}$
$E \rightarrow E_1/E_2$	$\{E.val = newTemp; gen(/, E_1.val, E_2.val, E.val)\}$
$E \rightarrow -E_1$	$\{E.val = newTemp; gen(@, E_1.val, -, E.val)\}$
$E \rightarrow (E_1)$	$\{E.val = E_1.val\}$
$E \rightarrow id$	$\{p = lookup(tblptr.top, id.name);$ $if\ p \neq null\ then\ E.val = p;\ else\ error;\}$
$E \rightarrow num$	$\{E.val = num.value;\}$

语句翻译

- 【例】翻译如下语句，假设$nxq = 100$ #

$E \rightarrow num \quad \{E.val = num.value;\}$
$if\ p \neq null\ then\ gen(=, E.val, -, p); else\ error;\}$
$if\ p \neq null\ then\ E.val = p; else\ error;\}$

$E\{val = b\}$
$E\{val = \$2\}$
$*$
$E\{val = \$3\}$
$=$
S
#

成功

$100.\ (+, a, 3, \$1)$
$101.\ (@, b, -, \$2)$
$102.\ (*, \$1, \$2, \$3)$
$103.\ (=, \$3, -, x)$

- 7.1 属性文法及其计算
  - 7.1.1 属性翻译文法
  - 7.1.2 综合属性的自下而上计算
  - 7.1.3 继承属性的自上而下计算
  - 7.1.4 依赖图
  - 7.1.5 树遍历的计算方法
  - 7.1.6 一遍扫描的处理方法
- 7.2 S-属性文法
  - 7.2.1 S-属性文法的自下而上计算
  - 7.2.2 构造表达式抽象语法树
  - 7.2.3 NFA箭弧单符化
- 7.3 L-属性文法
  - 7.3.1 翻译模式
  - 7.3.2 L-属性文法自上而下计算
  - 7.3.3 L-属性文法自下而上计算
  - 7.3.4 综合属性代替继承属性
- 7.4 声明语句的翻译
  - 7.4.1 Pascal风格过程内声明语句
  - 7.4.2 Pascal风格过程定义与声明语句
  - 7.4.3 Pascal风格数组声明
  - 7.4.4 Pascal风格结构体声明
  - 7.4.5 C风格函数定义与声明语句
- 7.5 表达式与赋值语句的翻译
  - 7.5.1 算术表达式与赋值语句
  - 7.5.2 Pascal风格数组的引用
  - 7.5.3 C风格数组的引用
  - 7.5.4 结构体的引用
  - 7.5.5 作为逻辑运算的布尔表达式
  - 7.5.6 地址和指针的引用
- 7.6 控制语句的翻译
  - 7.6.1 真假出口链
  - 7.6.2 四元式链操作
  - 7.6.3 作为条件控制的布尔表达式
  - 7.6.4 if和while语句
  - 7.6.5 C风格for语句
  - 7.6.6 Matlab风格for语句
  - 7.6.7 标号-goto语句
  - 7.6.8 switch语句
  - 7.6.9 break和continue语句
  - 7.6.10 三元运算符
  - 7.6.11 关系运算符的结合
- 7.7 过程调用的翻译
  - 7.7.1 过程的开始与结束标记
  - 7.7.2 返回语句
  - 7.7.3 过程调用
- 7.8 类型检查
  - 7.8.1 类型表达式
  - 7.8.2 类型检查
  - 7.8.3 隐式转换
  - 7.8.4 显式转换

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

数组引用的地址计算

- 行存储的多维数组$A[i_1, i_2, \dots, i_k]$地址：
  - 基准地址：$base - ((\dots((low_1 \times n_2 + low_2) \times n_3)\dots) \times n_k + low_k) \times w$
  - 处理声明语句时，计算出“$base -$”后面的部分，作为C存储到符号表
  - 动态地址：$(((\dots(i_1 \times n_2 + i_2) \times n_3)\dots) \times n_k + i_k) \times w$
  - 递归计算：$e_1 = i_1, e_2 = e_1 \times n_2 + i_2, \dots, e_k = e_{k-1} \times n_k + i_k$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

文法的考虑

- 生成数组的文法：

  $L \rightarrow id\ [Elist] \mid id$

  $Elist \rightarrow Elist, E \mid E$

- 要想知道数组的全部信息，需要有一个产生式把id（提供符号表地址）和最左下标E（提供下标值）联系起来，因此修改文法如下：

  $L \rightarrow Elist] \mid id$

  $Elist \rightarrow Elist, E \mid id[E$ // 这样$Elist$的翻译过程中随时知道id的信息，因为要查某个维度的长度信息

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

属性和元操作

- $Elist$的属性：
  - $array$，记录指向符号表中相应数组表项的指针。
  - $ndim$，记录$Elist$中下标表达式的个数，即维数。
  - val，表示变量或常量，用来临时存放由$Elist$中的下标表达式计算出来的值。
- L的属性：
  - val，存放变量或常量。
  - $offset$，简单名字为$null$，数组则为地址偏移量。
- 元操作：
  - $limit(array, j)$，返回n_j，即由$array$所指示的数组，其第j维的长度。
  - $getWidth(array)$，返回$array$数组的单元素字宽。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(1) $S \rightarrow A$

(2) $A \rightarrow L = E$

{ if $L.offset = null$ then $gen(=, E.val, -, L.val)$;

else $gen([] =, L.offset, E.val, L.val)$; }

(3) $E \rightarrow E_1 + E_2$ { $E.val = newTemp$; $gen(+, E_1.val, E_2.val, E.val)$; }

(4) $E \rightarrow E_1 - E_2$ { $E.val = newTemp$; $gen(-, E_1.val, E_2.val, E.val)$; }

(5) $E \rightarrow E_1 * E_2$ { $E.val = newTemp$; $gen(*, E_1.val, E_2.val, E.val)$; }

(6) $E \rightarrow E_1/E_2$ { $E.val = newTemp$; $gen(/, E_1.val, E_2.val, E.val)$; }

(7) $E \rightarrow -E_1$ { $E.val = newTemp$; $gen(@, E_1.val, -, E.val)$; }

(8) $E \rightarrow (E_1)$ { $E.val = E_1.val$; }

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(9) $E \rightarrow L$

{ if $L.offset = null$ then $E.val = L.val$;

else $\{E.val = newTemp; gen(= [], L.val, L.offset, E.val);\}$ }

(10) $L \rightarrow id$

{ $p = lookup(tblptr.top, id.name)$;

if $p = null$ then Error;

else $\{L.val = p; L.offset = null;\}$}

(11) $L \rightarrow num$ $\{L.val = num.value; L.offset = null;\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(12) $Elist \rightarrow id[E$ $\{Elist.dim = 1;\ Elist.val = E.val;$

$Elist.array = lookup(tblptr.top, id.name);\ \}$

(13) $Elist \rightarrow Elist_1, E$

$\{\ t = newTemp;\ Elist.dim = Elist_1.dim + 1;$

$n = limit(Elist_1.array, Elist.dim);$

$gen(*, Elist_1.val, n, t);\ gen(+, t, E.val, t);$

$Elist.array = Elist_1.array;\ Elist.val = t;\ \}$

(14) $L \rightarrow Elist]$

$\{\ w = getWidth(Elist.array); s = Elist.array.static;$

$L.val = newTemp;\ L.offset = newTemp;$

$gen(\&, Elist.array, -, L.val); gen(-, L.val, s, L.val);$

$gen(*, w, Elist.val, L.offset);\ \}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】翻译赋值语句$x = a[b + c, i, j]$; 其中a: $array[1..10, 2..5, 3..6]of\ real$

假设$nxq = 100$ #

$$C = (((low_1 \times n_2) + low_2) * n_3 + low_3) \times 8 = (((1 \times 4) + 2) * 4 + 3) \times 8 = 216$$

$(1)\ S \to A$
else $gen([] =, L.offset, E.val, L.val);$ }
$gen(-, L.val, s, L.val);\ gen(*, w, Elist.val, L.offset);$ }

$E\{val = c\}$
$+$
$E\{val = j\}$
]
$E\{val = \$6\}$
$=$
S
#

成功

四元式

$100.\ (+, b, c, \$1)$
$101.\ (*, \$1, 4, \$2)$
$102.\ (+, \$2, i, \$2)$
$103.\ (*, \$2, 4, \$3)$
$104.\ (+, \$3, j, \$3)$
$105.\ (\&, a, -, \$4)$
$106.\ (-, \$4, 216, \$4)$
$107.\ (*, 8, \$3, \$5)$
$108.\ (= [], \$4, \$5, \$6)$
$109.\ (=, \$6, -, x)$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(1) $A \rightarrow L = E;$ // 后面加了分号，归约为A，再进一步归约为S

{ if $L.offset = null$

$gen(=, E.place, -, L.place);$

else

$gen([] =, L.offset, E.place, L.place);$ }

(7) $L \rightarrow Elist]$

{ $L.place = newtemp; L.offset = newtemp;$

$gen(\&, Elist.array, -, L.place);$

$gen(*, w, Elist.place, L.offset);$ }

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

结构体的引用语句

```
1  struct Point {
2    double Coor[10, 2];
3    int Num;
4  };
5  struct Point p;
6  int n;
7   ...
8  p.Coor[0, 1] = 0;
9  n = p.Num;
```

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

属性和元操作

- 定义如下属性和元操作：
  - $base$：记录了结构体对象的基地址。
  - $memtbl$：结构体成员子表。
  - $p.getStructName()$：根据结构体对象p在符号表中记录的类型信息，得到结构体定义的名字。
  - $q.getMem()$：根据结构体定义q，得到其成员子表。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(15) $R \rightarrow id.$ $\{p = lookup(tblptr.top, id.name);$

$q = lookup(tblptr.top, p.getStructName());$

$R.base = newTemp; gen(\&, p, -, R.base); R.memtbl = q.getMem(); \}$

(16) $L \rightarrow R\ id$ $\{p = lookup(R.memtbl, id.name); L.val = newTemp;$

$L.offset = null; gen(= [], R.base, p.offset, L.val); \}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(17) $SElist \rightarrow R\ id[E \quad \{SElist.dim = 1; SElist.val = E.val;$

$SElist.array = lookup(R.memtbl, id.name); SElist.base = R.base;\}$

(18) $SElist \rightarrow SElist_1, E \quad \{t = newTemp; SElist.dim = SElist_1.dim + 1;$

$n = limit(SElist_1.array, SElist.dim); gen(*, SElist_1.val, n, t)$

$gen(+, t, E.val, t); SElist.array = SElist_1.array;$

$SElist.val = t; SElist.base = SElist_1.base;\}$

(19) $L \rightarrow SElist] \quad \{w = getWidth(SElist.array); off = SElist.array.offset;$

$L.val = newTemp; gen(+, SElist.base, off, L.val)$

$L.offset = newTemp; gen(*, w, SElist.val, L.offset);\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】 {p.Coor[0,1] = 0; k = p.Num;}#

全局符号表

名字	类别	类型	字宽	偏移
Point	struct		164	
Fun	proc	void	0	

结构体

名字	类别	类型	字宽	偏移
Coor	strumem	array(.)	160	0
Num	strumem	int	4	160

结构体中数组

维度	元素数
1	10
2	2

过程Fun符号表

名字	类别	类型	字宽	偏移
i	formal	int	4	0
j	formal	int	4	4
k	formal	int	4	8
p	variable	struct(.)	164	0

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】假设$nxq = 100$ #

(16) $L \rightarrow R\ id$ $\{p = lookup(R.memtbl, id.name); L.val = newTemp;$ $L.offset = null; gen(= [], R.base, p.offset, L.val);\}$
$L.offset = newTemp; gen(*, w, SElist.val, L.offset);\}$
$SElist.val = SElist_1.val; SElist.base = SElist_1.base;\}$

过程Fun符号表

结构体

名字	类别	类型	字宽	偏移
Coor	strumem	array(.)	160	0
Num	strumem	int	4	160
p	variable	struct(.)	164	0

;
$E\{val = \$6\}$
=
}
$Slist$
S
#

成功

100. $(\&, p, -, \$1)$
101. $(*, 0,2, \$2)$
102. $(+, \$2,1, \$2)$
103. $(+, \$1,0, \$3)$
104. $(*, 8, \$2, \$4)$
105. $([] =, \$4,0, \$3)$
106. $(\&, p, -, \$5)$
107. $(= [], \$5,160, \$6)$
108. $(=, \$6, -, k)$

{p.Coor[0,1] = 0; k = p.Num;}#

布尔表达式

- 布尔表达式的作用
  - 作为控制语句的条件式；
  - 作为逻辑运算，获得逻辑值。
- 算符优先级的说明：
  - 优先级由高到低：¬,∧,∨；
  - ∧,∨服从左结合规则，¬服从右结合规则；
  - 关系表达式形如$E_1 \theta E_2$，其中关系符θ包括$<, \leq, =, \neq, >, \geq$，$E_i$为算术表达式；
  - 各关系符优先级相同，高于布尔算符，低于算术算符；
  - 关系符不得结合，如$a < b < c$为非法。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

作为逻辑运算的布尔表达式

- 计算方法1：如同算术表达式，一步不差的从表达式各部分值计算整个表达式的值
  - $1 \vee (\neg 0 \wedge 0) = 1 \vee (1 \wedge 0) = 1 \vee 0 = 1$
- 计算方法2：优化算法
  - $A \vee B: if\ A\ then\ true\ else\ B$
  - $A \wedge B: if\ A\ then\ B\ else\ false$
  - $\neg A: if\ A\ then\ false\ else\ true$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

$(20) E \rightarrow E_1 \&\& E_2$ $\{E.val = newTemp; gen(\wedge, E_1.val, E_2.val, E.val);\}$

$(21) E \rightarrow E_1 || E_2$ $\{E.val = newTemp; gen(\vee, E_1.val, E_2.val, E.val);\}$

$(22) E \rightarrow \neg E_1$ $\{E.val = newTemp; gen(\neg, E_1.val, -, E.val);\}$

$(23) E \rightarrow E_1 \theta E_2$ $\{E.val = newTemp;$

$gen(j\theta, E_1.val, E_2.val, nxq + 3);$

$gen(=, 0, -, E.val);$

$gen(j, -, -, nxq + 2);$

$gen(=, 1, -, E.val);\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】假设$nxq = 100$ #

$(21)E \to E_1 \| E_2 \ \{E.val = newTemp; gen(\vee, E_1.val, E_2.val, E.val);\}$
$gen(=, 0, -, E.val); gen(j, -, -, nxq + 2); gen(=, 1, -, E.val);\}$

$E\{val = j\}$
+
$E\{val = e\}$
&&
$E\{val = \$4\}$
;
$E\{val = \$5\}$
=
A
#

后续略

100. $(j <, a, b, 103)$
101. $(=, 0, -, \$1)$
102. $(j, -, -, 104)$
103. $(=, 1, -, \$1)$
104. $(+, i, j, \$2)$
105. $(j \leq, c, \$2, 108)$
106. $(=, 0, -, \$3)$
107. $(j, -, -, 109)$
108. $(=, 1, -, \$3)$
109. $(\wedge, \$3, e, \$4)$
110. $(\vee, \$1, \$4, \$5)$
111. $(=, \$5, -, x)$

x = a < b || c <= i + j && e;

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

地址和指针的引用

```
1  double x;
2  double* p;
3   ...
4  p = &x;
5  x = *p;
6  *p = x;
```

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(24) $E \rightarrow \&L$ $\{E.val = newTemp;$

if $(L.offset = null)$ then $gen(\&, L.val, -, E.val);$

else $gen(+, L.val, L.offset, E.val);\}$

(25) $E \rightarrow * L$ $\{E.val = newTemp;$

if $(L.offset = null)$ then $gen(=*, L.val, -, E.val);$

else $\{gen(= [], L.val, L.offset, E.val); gen(=*, E.val, -, E.val);\}$

(26) $A \rightarrow * L = E;$ $\{t = newTemp;$

if $(L.offset = null)$ then $gen(*=, E.val, -, L.val);$

else $\{gen(= [], L.val, L.offset, t); gen(*=, E.val, -, t);\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】假设$nxq = 100$ #

(26) $A \rightarrow * L = E;$ $\{t = newTemp;$
if $(L.offset = null)$ then $gen(*=, E.val, -, L.val);$
else $\{gen(= [], L.val, L.offset, t); gen(*=, E.val, -, t);\}$

$L\{val = p, offset = null\}$
$E\{val = \$2\}$
;
$E\{val = \$3\}$
$=$
$L\{val = p, offset =$
A
$Slist$
#

成功

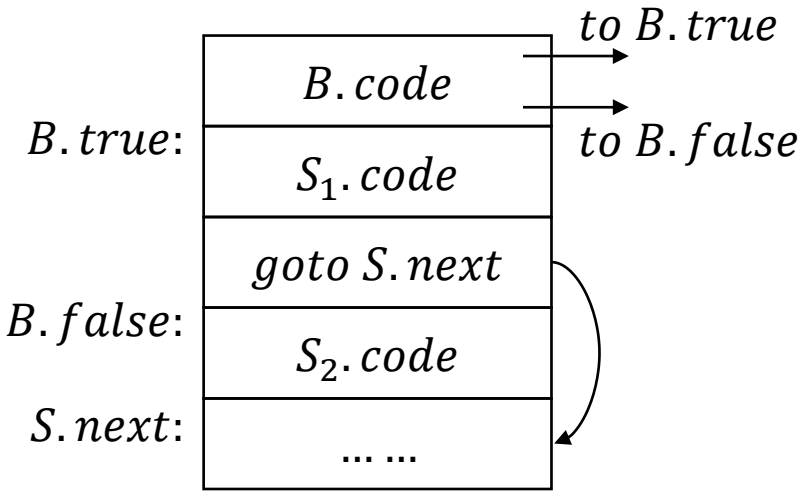
100. $(\&, x, -, \$1)$
101. $(=, \$1, -, p)$
102. $(=*, p, -, \$2)$
103. $(+, x, \$2, \$3)$
104. $(*=, \$3, -, p)$

$p = \&x; *p = x + *p;$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

真假出口

- 考虑条件语句$if\ B\ then\ S_1\ else\ S_2$中的布尔表达式$B$，仅用于控制对$S_1$和$S_2$的选择，不需要用一个临时变量保留其值，因此可以为$B$设置两个出口：
  - 真出口：指向S_1的第一个四元式；
  - 假出口：指向S_2的第一个四元式。



山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

真假出口

【例】$if\ a > c\ \vee b < d\ then\ S_1\ else\ S_2$

这两个语句都转移到L_2，但L_2需要分析到S_1才能确定，其它类似。

$if\ a > c\ goto\ L_2$

$goto\ L_1$

L_1: $if\ b < d\ goto\ L_2$

$goto\ L_3$

L_2: (关于S_1的代码序列)

$goto\ L_{next}$

L_3: (关于S_2的代码序列)

L_{next}:

100. $(j >, a, c, 104)$

101. $(j, -, -, 102)$

102. $(j <, b, d, 104)$

103. $(j, -, -, 116)$

104. S_1...

115. $(j, -, -, 120)$

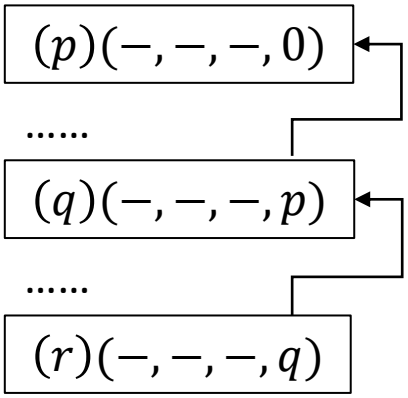
116. S_2 ...

120. ...

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

真假出口链

- 思路：
  - 生成四元式时，暂不确定跳转标号，而是把指向同一目标的四元式组成一个链表，确定目标后再回填；
  - 为非终结符号B赋予两个综合属性$B.trueList$和$B.falseList$，分别记录真、假出口链；
  - 需要回填的四元式，借助第四区块构造真、假出口链。



- 用到四元式：
  - $(jnz, a, -, p)$: $if\ a\ goto\ p$
  - $(j\theta, x, y, p)$: $if\ x\ \theta\ y\ goto\ p$
  - $(j, -, -, p)$: $goto\ p$

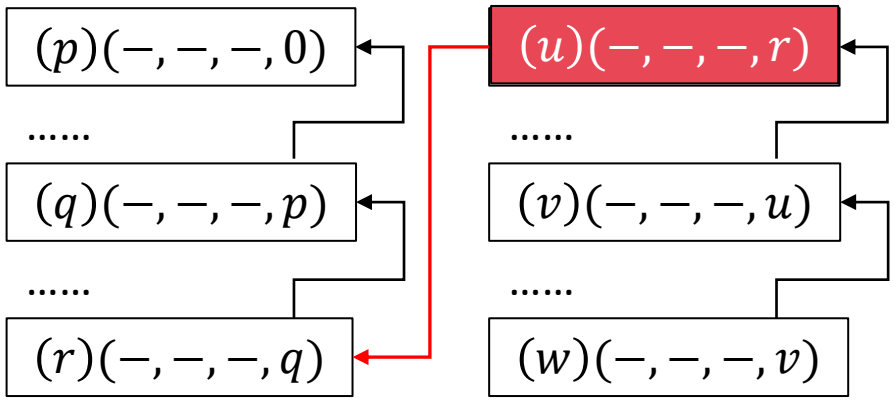
山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

四元式链操作

- 需要用到的变量或函数：
  - $makeList(i)$：创建一个链表，这个链表仅含标号为i的四元式，返回链表指针。
  - $merge(p1,p2)$：把以$p1$和$p2$为链首的两条链合并，返回新的链首。当遇到$B_1 \wedge B_2$时，它们的$falseList$链表需要合并；当遇到$B_1 \vee B_2$时，它们的$tureList$链表需要合并。
  - $backPatch(p,t)$：回填，把p所链接的每个四元式的第四区段都填t。当遇到$B_1 \wedge B_2$时，B_2确定了B_1的$trueList$链表需要回填的地址；当遇到$B_1 \vee B_2$时，B_2确定了B_1的$falseList$链表需要回填的地址。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

merge操作



$B_1.trueList = r$
或
$B_1.falseList = r$

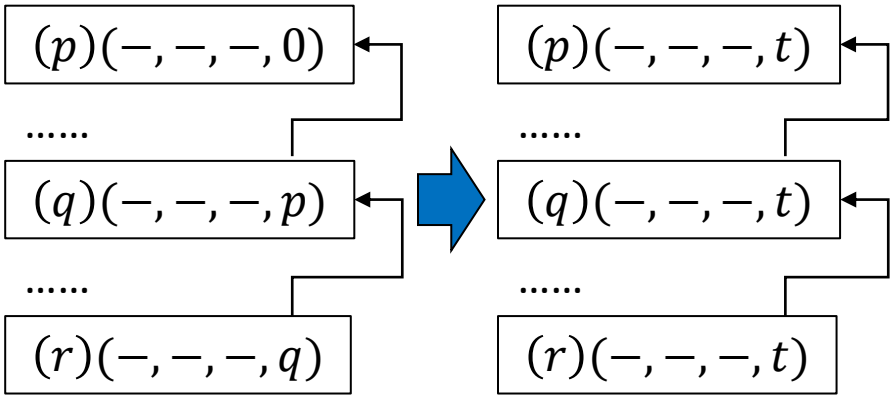
$B_2.trueList = w$
或
$B_2.falseList = w$

$B.trueList = w$
或
$B.falseList = w$

```
Quad *Merge(Quad *p1, Quad *p2)
{
    if (p1 == null) return p2;
    if (p2 == null) return p1;
    // 找p2链的链尾
    Quad *p = p2;
    while (p的第四区段内容 ≠ null)
        p = p的第四区段内容;
    p的第四区段内容 = p1;
    return p2;
}
```

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

backPatch操作



$B_1.trueList = r$
或
$B_1.falseList = r$

```
void backpatch(Quad *p, Quad *t)
{
    Quad *q;
    while (p ≠ null)
    {
        q = p的第四区段内容;
        p的第四区段内容 = t;
        p = q;
    }
}
```

翻译模式

(1) $B \rightarrow B_1 || M B_2$ $\{backPatch(B_1.falseList, M.quad);$
$B.trueList = merge(B_1.trueList, B_2.trueList);$
$B.falseList = B_2.falseList;\}$

(2) $B \rightarrow B_1 \&\& M B_2$ $\{backPatch(B_1.trueList, M.quad);$
$B.falseList = merge(B_1.falseList, B_2.falseList);$
$B.trueList = B_2.trueList;\}$

(3) $M \rightarrow \varepsilon$ $\{M.quad = nxq;\}$

(4) $B \rightarrow \neg B_1$ $\{B.trueList = B_1.falseList; B.falseList = B_1.trueList;\}$

(5) $B \rightarrow (B_1)$ $\{B.trueList = B_1.trueList; B.falseList = B_1.falseList;\}$

(6) $B \rightarrow E_1\ \theta\ E_2$ $\{B.trueList = makeList(nxq); B.falseList = makeList(nxq+1);$
$gen(j\theta, E_1.val, E_2.val, 0); gen(j, -, -, 0);\}$

(7) $B \rightarrow E$ $\{B.trueList = makeList(nxq); B.falseList = makeList(nxq+1);$
$gen(jnz, E.val, -, 0); gen(j, -, -, 0);\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】假设$nxq = 100$ #

(1) $B \rightarrow B_1||MB_2$ $\{backPatch(B_1.falseList, M.quad);$
$B.trueList = merge(B_1.trueList, B_2.trueList); B.falseList = B_2.falseList;\}$

else $\{E.val = newtemp; gen(= [], L.val, L.offset, E.val);\}\}$

$E\{val = j\}$
$B\{trueList = 105, falseList = 106\}$
$M\{quad = 105\}$
&&
$B\{trueList = 105, falseList = 106\}$
$M\{quad = 102\}$
||
$B\{trueList = 105, falseList = 106\}$
#

100. $(j <, a, b, 0)$
101. $(j, -, -, 102)$
102. $(+, i, j, \$1)$
103. $(j \leq, c, \$1, 105)$
104. $(j, -, -, 0)$
105. $(jnz, e, -, 100)$
106. $(j, -, -, 104)$

$a < b || c <= i + j \&\& e$

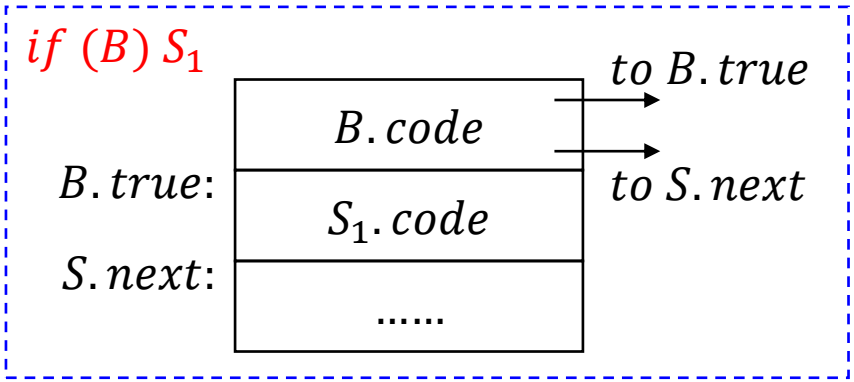
两个未填充四元式链，需要等到确定布尔式为真做什么、为假做什么时才能回填。

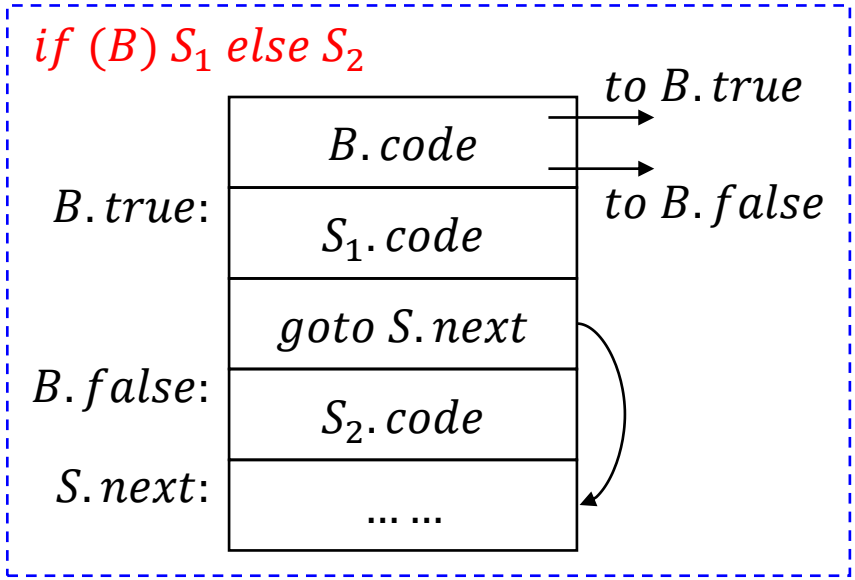
山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

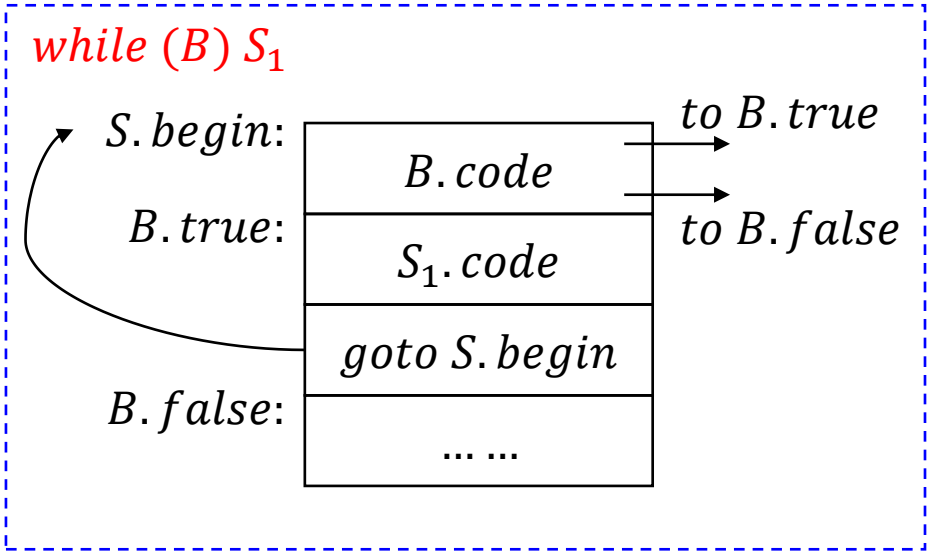
控制逻辑

控制流语句：

- $S \rightarrow if\ (B)\ S_1 \mid if\ (B)\ S_1\ else\ S_2 \mid while\ (B)\ S_1$







山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(8) $S \rightarrow if\ (B)\ M_1 S_1 N\ else\ M_2 S_2$ $\{backPatch(B.trueList, M_1.quad);$

$backPatch(B.falseList, M_2.quad);$

$S.nextList = merge(S_1.nextList, N.nextList, S_2.nextList);\ \}$

(9) $N \rightarrow \varepsilon$ $\{N.nextList = makeList(nxq); gen(j, -, -, 0);\ \}$

(10) $S \rightarrow if\ (B)\ MS_1$ $\{backPatch(B.trueList, M.quad);$

$S.nextList = merge(B.falseList, S_1.nextList);\ \}$

(11) $S \rightarrow while(M_1 B)\ M_2 S_1$ $\{backPatch(S_1.nextList, M_1.quad);$

$backPatch(B.trueList, M_2.quad);$

$S.nextList = B.falseList;$

$gen(j, -, -, M_1.quad);\ \}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(12) $S \rightarrow \{Slist\}$ $\{S.nextList = Slist.nextList;\}$

(13) $Slist \rightarrow S$ $\{Slist.nextList = S.nextList;\}$

(14) $Slist \rightarrow Slist_1 MS$ $\{backPatch(Slist_1.nextList, M.quad);$

$Slist.nextList = S.nextList;\}$

(15) $S \rightarrow A$ $\{S.nextList = makeList();\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】假设$nxq = 100$

#

(8) $S \rightarrow if\ (B)\ M_1 S_1 N\ else\ M_2 S_2$ $\{backPatch(B.trueList, M_1.quad);$
$backPatch(B.falseList, M_2.quad);$
$S.nextList = merge(S_1.nextList, N.nextList, S_2.nextList);\ \}$

$E\{val = 0\}$
$=$
$S\{nextList = null\}$
$M\{quad = 110\}$
$else$
$N\{nextList = 109\}$
$S\{nextList = null\}$
$M\{quad = 107\}$
$)$
$B\{trueList = 105, falseList = 106\}$
$($
$S\{nextList = 109\}$
#

100. $(j <, a, b, 107)$
101. $(j, -, -, 102)$
102. $(+, i, j, \$1)$
103. $(j \leq, c, \$1, 105)$
104. $(j, -, -, 110)$
105. $(jnz, e, -, 107)$
106. $(j, -, -, 110)$

107. $(+, y, z, \$2)$
108. $(=, \$2, -, x)$
109. $(j, -, -, 0)$
110. $(=, 0, -, x)$

$if(a < b \,||\, c <= i + j \,\&\&\, e)x = y + z;$
$else\ x = 0;$

现在剩下最后一个链$S.nextlist$，
此处需要分析下一条语句时回填。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】假设$nxq = 100$

#

(14) $Slist \rightarrow Slist_1 MS$ $\{backPatch(Slist_1.nextList, M.quad);$
$Slist.nextList = S.nextList;\}$

$S.nextList = B.falseList; gen(j, -, -, M_1.quad); \}$

else $\{E.val = newtemp; gen(=[], E.val, E.offset, E.val); \}\}$

$M\{quad = 104\}$
)
$B\{trueList = 102, falseList = 103\}$
(
$S\{nextList = 103\}$
;
$E\{val = 0\}$
=
$S\{nextList = null\}$
$M\{quad = 107\}$
$Slist\{nextList = null\}$
#

成功

100. $(j <, i, 100, 102)$
101. $(j, -, -, 107)$
102. $(j <, a, b, 104)$
103. $(j, -, -, 100)$
104. $(+, i, 1, \$1)$
105. $(=, \$1, -, i)$
106. $(j, -, -, 100)$
107. $(=, 0, -, x)$

$E\{val = 1\}$
; while $(i < 100)$ if $(a < b)\ i = i + 1;\ x = 0;$
$E\{val = \$1\}$
=
$S\{nextList = null\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

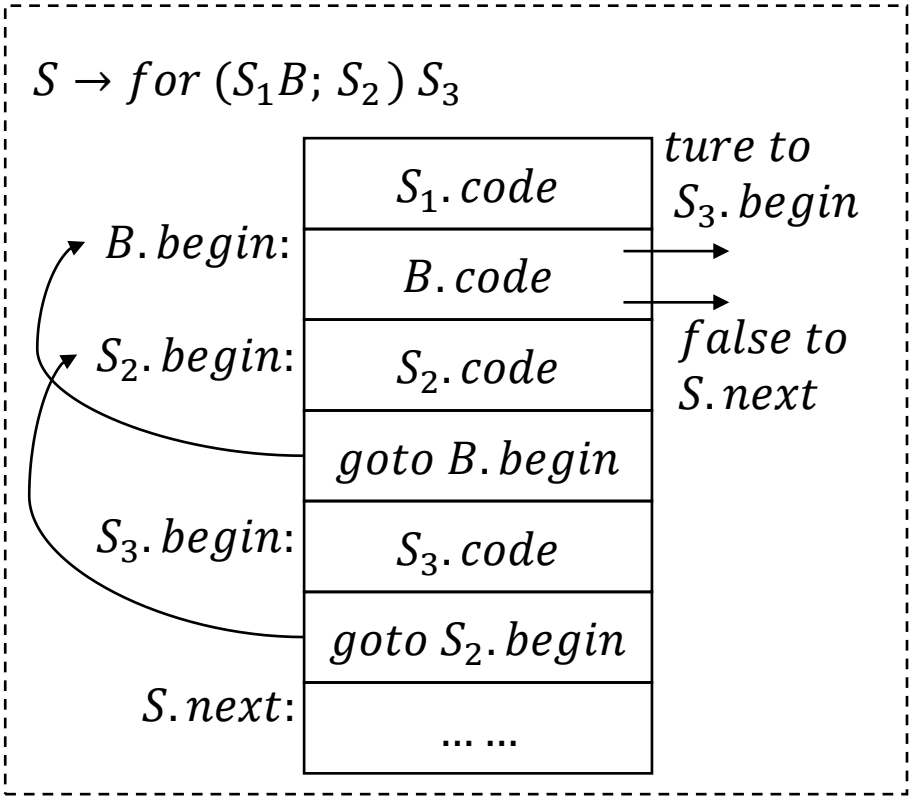
C风格for语句

```
1  for (语句1; 布尔表达式2; 语句3) {
2    语句块
3  }
4  for (语句1; 布尔表达式2; 语句3;) {
5    语句块
6  }
7  for (i = 0; i < 100; i++;) {
8    ...
9  }
```

郑艳伟，zhengyw@sdu.edu.cn

C风格for语句

- C风格for语句逻辑：



(16) $S \rightarrow for(S_1 M_1 B; M_2 S_2 N) M_3 S_3$

$\{backPatch(B.trueList, M_3.quad);$

$backPatch(S_1.nextList, M_1.quad);$

$backPatch(S_2.nextList, M_1.quad);$

$backPatch(S_3.nextList, M_2.quad);$

$backPatch(N.nextList, M_1.quad);$

$gen(j, -, -, M_2.quad);$

$S.nextList = B.flaseList;\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】假设$nxq = 100$

#

(14) $Slist \rightarrow Slist_1 MS\{backPatch(Slist_1.nextList, M.quad);$
$Slist.nextList = S.nextList;\}$

$gen(j, -, -, M_2.quad); S.nextList = B.flaseList;\}$

$S\{nextList = null\}$
$M\{quad = 106\}$
)
$N\{nextList = 105\}$
$S\{nextList = null\}$
$M\{quad = 103\}$
;
$B\{trueList = 101, falseList = 102\}$
$M\{quad = 101\}$
$S\{nextList = null\}$
$M\{quad = 109\}$
$Slist\{nextList = null\}$
#

成功

$100.(=, 0, -, i)$
$101.(j <, i, 100, 106)$
$102.(j, -, -, 109)$
$103.(+, i, 1, \$1)$
$104.(=, \$1, -, i)$
$105.(j, -, -, 101)$
$106.(+, sum, 1, \$2)$
$107.(=, \$2, -, sum)$
$108.(j, -, -, 103)$
$109.(=, 0, -, x)$

```
for(i=0;i<100;i=i+1;)
    sum=sum+1;
x = 0;
```

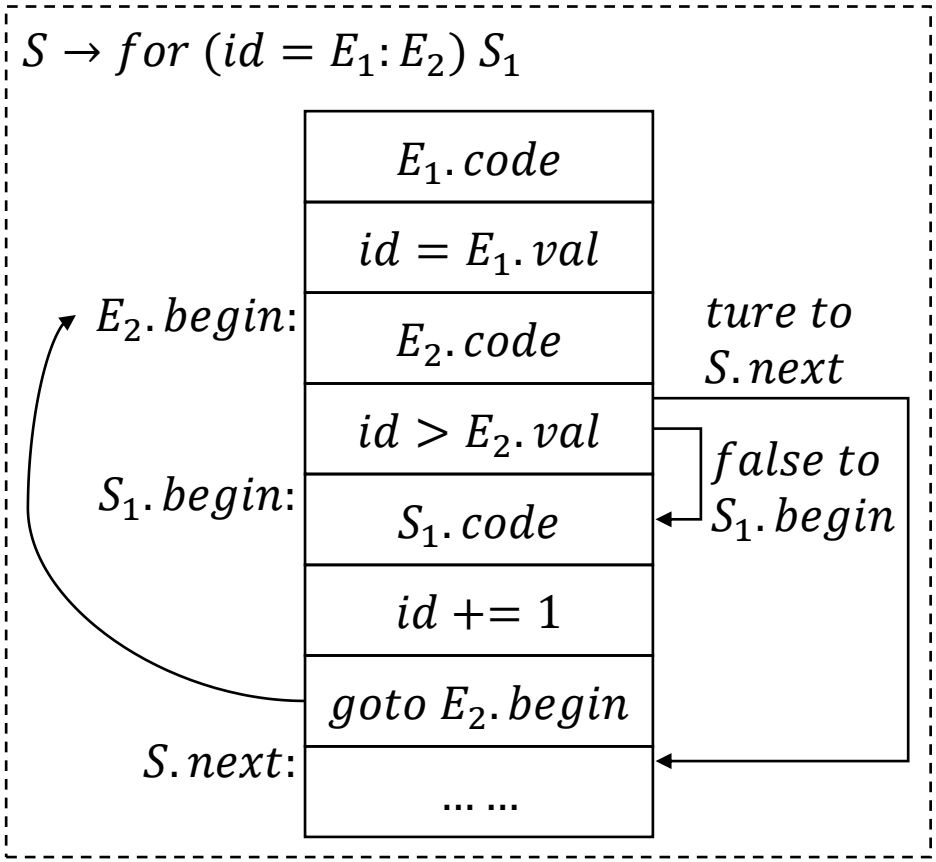
山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

Pascal和Matlab风格for语句

```
1  for 循环变量 := 初值 to 终值 do {
2    语句块
3  }
4  for (循环变量 = 初值; 循环变量 <= 终值; 循环变量++) {
5    语句块
6  }
7  for 循环变量 = 初值 : 终值 do {
8    语句块
9  }
10 for i = 1 : 100 do {
11   ...
12 }
13 for 循环变量 = 初值 : 步长 : 终值 do {
14   语句块
15 }
16 for i = 100 : -1 : 1 do {
17   ...
18 }
```

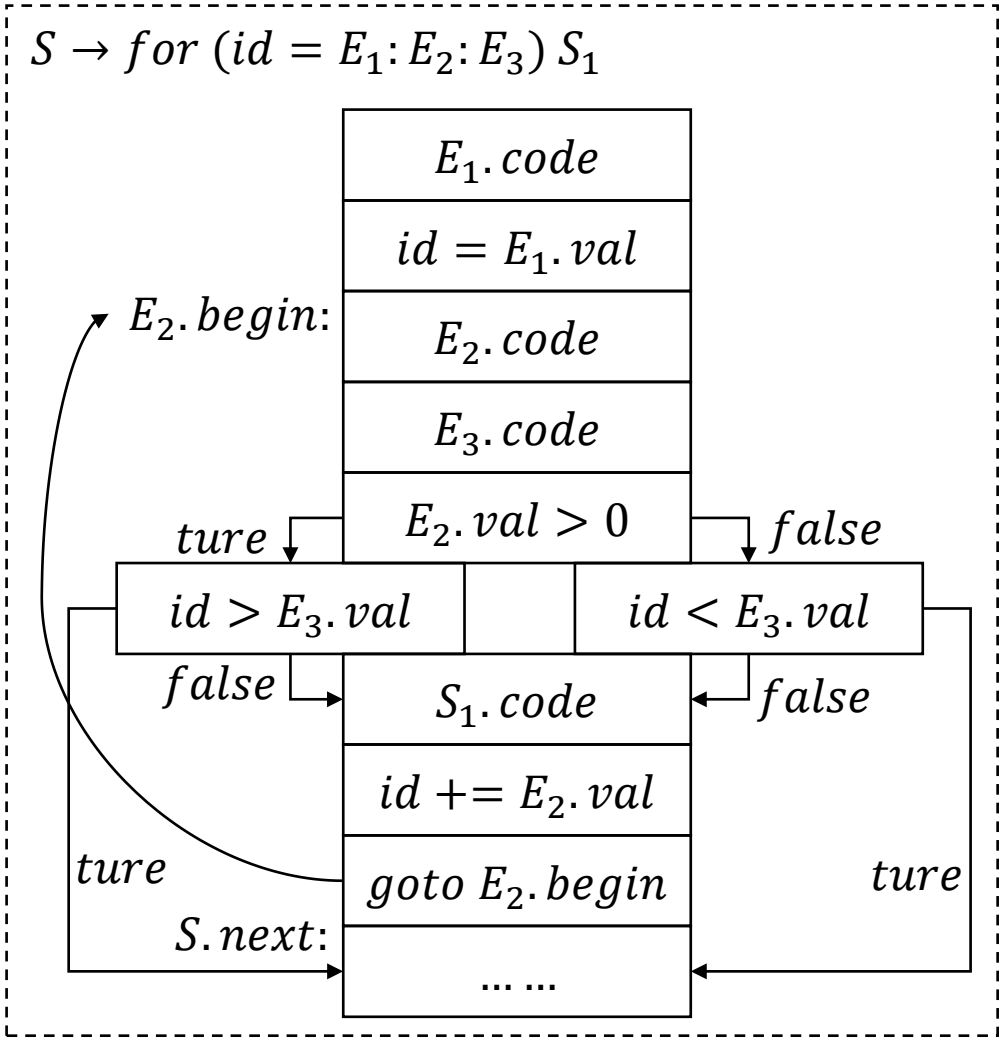
郑艳伟，zhengyw@sdu.edu.cn

Matlab风格for语句控制逻辑



山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

Matlab风格for语句控制逻辑



山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(17) $S \rightarrow for(H)S_1$ $\{gen(+, H.var, H.step, H.var); gen(j, -, -, H.condi);$
$backPatch(S_1.nextList, nxq - 2); S.nextList = H.trueList; \}$

(18) $J \rightarrow id = E_1$: $\{J.var = lookup(tblptr.top, id.name); gen(=, E_1.val, -, J.var);\}$

(19) $H \rightarrow JME_2$ $\{H.var = J.var; H.step = 1; H.end = E_2.val; H.condi = M.quad;$
$H.trueList = makeList(nxq); gen(j >, H.var, E.val, 0);\}$

(20) $H \rightarrow JME_2 : E_3$ $\{H.var = J.var; H.step = E_2.val; H.end = E_3.val;$
$H.condi = M.quad; gen(j \leq, E_2.val, 0, nxq + 3);$
$H.trueList = makeList(nxq);$
$gen(j >, H.var, E_3.val, 0); gen(j, -, -, nxq + 2);$
$H.trueList = merge(H.trueList, makeList(nxq));$
$gen(j <, H.var, E_3.val, 0); \}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】假设$nxq = 100$

#

(17) $S \rightarrow for(H)S_1$ $\{gen(+, H.var, H.step, H.var); gen(j, -, -, H.condi);$ $backPatch(S_1.nextList, nxq - 2); S.nextList = H.trueList;\}$
$gen(j >, H.var, E_3.val, 0); gen(j, -, -, nxq + 2);$ $H.trueList = merge(H.trueList, makeList(nxq)); gen(j <, H.var, E_3.val, 0);\}$

100.$(=, 100, -, i)$	106.$(+, sum, i, \$2)$
101.$(-, a, b, \$1)$	107.$(=, \$2, -, sum)$
102.$(j \leq, -1, 0, 105)$	108.$(+, i, -1, i)$
103.$(j >, i, \$1, 110)$	109.$(j, -, -, 101)$
104.$(j-, -, 106)$	110.$(=, 0, -, x)$
105.$(j <, i, \$1, 110)$	

$E\{val = \$1\}$
$S\{nextList = null\}$
)
$H\{var = i, step = -1, end = \$1$
$S\{nextList = null\}$
$M\{quad = 110\}$
$Slist\{nextList = null\}$
#

成功

```
for(i = 100 : -1 : a - b)
    sum = sum + i;
x = 0;
```

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

标号-goto语句

```
1    ...
2    goto L;
3    ...
4    L:  ...
5    ...
6    goto L;
7    ...
```

- 标号语句形式：$L:S;$

  ① $S \rightarrow label\ S \mid goto\ id;$

  ② $label \rightarrow id:$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

遇到goto L

- 若标号已存在且已定义
  - 生成四元式：$(j, -, -, p)$

名字	类型	…	定义否	地址
……				
L_1	标号		是	p
……				
L_2	标号		否	nxq
L_3	标号		否	x
……				

- 若标号已存在且未定义
  - 生成四元式：$(j, -, -, x)$；
  - $x = nxq - 1$；
  - 当遇到定义标号语句时，回填。

- 若标号不存在
  - 符号表中增加标号，定义否为“否”，地址为nxq；
  - 生成四元式：$(j, -, -, 0)$；
  - 当遇到定义标号语句时，回填。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

遇到 L:

- 若标号已存在且已定义、或者类型不是标号
  - 这是一个错误

名字	类型	…	定义否	地址
……				
L_1	标号?		是	p
……				
L_2	标号		是	nxq
L_3	标号		是	x
……				

- 若标号已存在且未定义
  - 定义否改为“是”；
  - 用nxq回填地址处记录的四元式链；
  - 地址修改为nxq。

- 若标号不存在
  - 填写符号表，定义否为“是”，地址填nxq。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

(21) $S \rightarrow label\ S_1$ $\{S.nextList = S_1.nextList\}$

(22) $label \rightarrow id:$ $\{p = lookup(tblptr.top, id.name);$

if $p = null$ then $logLabel(tblptr.top, id.name, label, true, nxq);$

else if $p.type \neq label \parallel p.isDefined = true$ then Error;

else { $backPatch(p.address, nxq);$

$modifyLabel(p, isDefined = true, address = nxq);$ } }

(23) $S \rightarrow goto\ id;$ $\{p = lookup(tblptr.top, id.name);$

if $p = null$ then $\{logLabel(tblptr.top, id.name, label, false, nxq);$

$gen(j, -, -, 0);\}$

else if $p.type \neq label$ then Error;

else if $p.isDefined = true$ then $gen(j, -, -, p.address);$

else $\{gen(j, -, -, p.address);$

$modifyLabel(p, address = nxq - 1);\}$

$S.nextList = null;$ }

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】假设$nxq = 100$ #

(21) $S \rightarrow label\ S_1$ $\{S.nextList = S_1.nextList\}$

if $p = null$ then $logLabel(tblptr.top, id.name, label, true, nxq)$;
else if $p.type \neq label \ || \ p.isDefined = true$ then Error;
else { $backPatch(p.address, nxq)$;
$modifyLabel(p, isDefined = true, address = nxq)$; } }
$S.nextList = null$; }

名字	类别	定义否	地址
L	label	true	102

;
L
$S\{nextList = null\}$
$M\{quad = 104\}$
$Slist\{nextList = null\}$
#

成功

100.$(j, -, -, 102)$
101.$(j, -, -, 102)$
102.$(+, y, z, \$1)$
103.$(=, \$1, -, x)$
104.$(j, -, -, 102)$

$goto\ L;$
$goto\ L;$
$L: x = y + z;$
$goto\ L;$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

switch语句

```
1  switch (E) {
2    case c1: S1;
3    case c2: S2;
4     ...
5    case cn_1: Sn_1;
6    default: Sn;
7  }
```

8.6.8 switch语句

- 语句形式：

switch (E)

{

case c_1: S_1;

case c_2: S_2;

……

case c_{n-1}: S_{n-1};

default: S_n;

}

- 生成四元式：

$goto\ test$

L_1: S_1; $goto\ next$

……

L_{n-1}: S_{n-1}; $goto\ next$

L_n: S_n; $goto\ next$

$test$: $if\ T = c_1\ goto\ L_1$

……

$if\ T = c_{n-1}\ goto\ L_{n-1}$

$goto\ L_n$

$next$:

- 遇到switch
  - 产生标号test；
  - 产生标号next；
  - 产生临时变量T，存放E
  - 生成$goto\ test$。
- 遇到c_i
  - 产生标号L_i，填入符号表
  - 记下标号和符号表位置，生成$test$时使用；
  - S之后要有个四元式 $goto\ next$。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(24) $W \rightarrow switch\ (E)$ $\{que.init();$

$W.val = newTemp;\ gen(=, E.val, -, W.val);$

$logLabel(tblptr.top, test, case, false, nxq); gen(j, -, -, 0);$

$logLabel(tblptr.top, next, case, false, 0); \}$

(25) $C \rightarrow case\ c$: $\{ que.en(c, nxq); \}$

(26) $C \rightarrow U\ case\ c$: $\{ que.en(c, nxq); \}$

(27) $U \rightarrow CS$ $\{ p = lookup(tblptr.top, next); gen(j, -, -, p.address);$

$modifyLabel(p, address = nxq - 1); \}$

(28) $X \rightarrow default: MS$ $\{ X.quad = M.quad;$

$p = lookup(tblptr.top, next); gen(j, -, -, p.address);$

$modifyLabel(p, address = nxq - 1); \}$

(29) $X \rightarrow \varepsilon$ $\{ X.quad = null; \}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(30) $S \rightarrow W\{UX\}$ $\{ p = lookup(tblptr.top, test); backPatch(p.address, nxq);$

$while\ (!que.isEmpty)$

$\{ (c, addr) = que.de(); gen(j =, W.val, c, addr); \}$

$if\ (X.quad \neq null)\ then\ gen(j, -, -, X.quad);$

$p = lookup(tblptr.top, next);\ S.nextList = p.address;$

$rmLabel(test); rmLabel(next); \}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】假设$nxq = 100$

$sum = x; \}flag = -1; \#$

(30) $S \to W\{UX\}$ $\{ p = lookup(tblptr.top, test); backPatch(p.address, nxq);$
$while\ (!que.isEmpty)\ \{(c, addr) = que.de(); gen(j =, W.val, c, addr);\}$
$if\ X.quad \neq null\ then\ gen(j, -, -, X.quad);$
$p = lookup(tblptr.top, next);\ S.nextList = p.address;$
$rmLabel(test); rmLabel(next);\}$

名字	类别	定义否	地址
test	case	false	101
next	case	false	109

0	102
1	105

$switch(flag)\{$
$case\ 0 : sum = sum + x;$
$case\ 1 : sum = sum - x;$
$default : sum = x;\ \}$
$flag = -1;$

$S\{ne$
$M\{q$
$\}$
$X\{q$

$S(nextList = null)$

$M\{quad = 113\}$

$Slist(nextList = 109)$

$\#$

成功

100.$(=, flag, -, \$1)$	108.$(=, x, -, sum)$
101.$(j, -, -, 110)$	109.$(j, -, -, 113)$
102.$(+, sum, x, \$2)$	110.$(j =, \$1,0,102)$
103.$(=, \$2, -, sum)$	111.$(j =, \$1,1,105)$
104.$(j, -, -, 113)$	112.$(j, -, -, 108)$
105.$(-, sum, x, \$3)$	113.$(@, 1, -, \$4)$
106.$(=, \$3, -, sum)$	114.$(=, \$4, -, flag)$
107.$(j, -, -, 113)$	

break和continue语句

- 语句作用
  - break跳出循环或switch；
  - continue跳出循环的一次迭代，继续下一次迭代。

(31) $S \rightarrow break;$ $\{ S.brkList = makeList(nxq); S.ctnList = null;$
$S.nextList = null; gen(j, -, -, 0); \}$

(32) $S \rightarrow continue;$ $\{ S.brkList = null; S.ctnList = makeList(nxq);$
$S.nextList = null;\ gen(j, -, -, 0); \}$

对控制语句的修改

(8) $S \rightarrow if\ (B)\ M_1 S_1 N\ else\ M_2 S_2$ $\{backPatch(B.trueList, M_1.quad);$

$backPatch(B.falseList, M_2.quad);$

$S.nextList = merge(S_1.nextList, N.nextList, S_2.nextList);$

$S.brkList = merge(S_1.brkList, S_2.brkList);$

$S.ctnList = merge(S_1.ctnList, S_2.ctnList);\ \}$

(10) $S \rightarrow if\ (B)\ MS_1$ $\{backPatch(B.trueList, M.quad);$

$S.nextList = merge(B.falseList, S_1.nextList);$

$S.brkList = S_1.brkList; S.ctnList = S_1.ctnList;\ \}$

(11) $S \rightarrow while(M_1 B)\ M_2 S_1$ $\{backPatch(S_1.nextList, M_1.quad);$

$backPatch(B.trueList, M_2.quad);\ backPatch(S_1.ctnList, M_1.quad);$

$S.nextList = merge(B.falseList, S_1.brkList);\ gen(j, -, -, M_1.quad);$

$S.brkList = null; S.ctnList = null;\ \}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

对控制语句的修改

(12) $S \rightarrow \{Slist\}$ $\{S.nextList = Slist.nextList;$

$S.brkList = Slist.brkList; S.ctnList = Slist.ctnList;\}$

(13) $Slist \rightarrow S$ $\{Slist.nextList = S.nextList;$

$Slist.brkList = S.brkList; Slist.ctnList = S.ctnList;\}$

(14) $Slist \rightarrow Slist_1 MS$ $\{backPatch(Slist_1.nextList, M.quad);$

$Slist.nextList = S.nextList;$

$Slist.brkList = merge(Slist_1.brkList, S.brkList);$

$Slist.ctnList = merge(Slist_1.ctnList, S.ctnList);\}$

(15) $S \rightarrow A$ $\{S.nextList = null; S.brkList = null; S.ctnList = null;\}$

对控制语句的修改

(16) $S \rightarrow for(S_1 M_1 B; M_2 S_2 N) M_3 S_3$ $\{backPatch(B.trueList, M_3.quad);$

$backPatch(S_1.nextList, M_1.quad);\ backPatch(S_2.nextList, M_1.quad);$

$backPatch(S_3.nextList, M_2.quad);\ backPatch(N.nextList, M_1.quad);$

$gen(j, -, -, M_2.quad);\ S.nextList = merge(B.flaseList, S_3.brkList);$

$backPatch(S_3.ctnList, M_2.quad);\ S.brkList = null;\ S.ctnList = null;\ \}$

(17) $S \rightarrow for(H) S_1$ $\{gen(+, H.var, H.step, H.var);\ gen(j, -, -, H.condi);$

$backPatch(S_1.nextList, nxq - 2);\ S.nextList = merge(H.trueList, S_1.brkList);$

$backPatch(S_1.ctnList, nxq - 2);\ S.brkList = null;\ S.ctnList = null;\ \}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

对控制语句的修改

(25) $C \rightarrow case\ c:$ $\{ que.en(c, nxq); C.brkList = null; \}$

(26) $C \rightarrow U\ case\ c:$ $\{ que.en(c, nxq); C.brkList = U.brkList; \}$

(27) $U \rightarrow CS$ $\{ p = lookup(tblptr.top, next); gen(j, -, -, p.address);$

$modifyLabel(p, address = nxq - 1);$

$U.brkList = merge(C.brkList, S.brkList); \}$

(28) $X \rightarrow default: MS$ $\{ X.quad = M.quad;$

$p = lookup(tblptr.top, next); gen(j, -, -, p.address);$

$modifyLabel(p, address = nxq - 1); X.brkList = S.brkList; \}$

(29) $X \rightarrow \varepsilon$ $\{ X.quad = null; X.brkList = null; \}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

对控制语句的修改

(30) $S \rightarrow W\{UX\}$ $\{ p = lookup(tblptr.top, test); backPatch(p.address, nxq);$

$while\ (!que.isEmpty)$

$\{ (c, addr) = que.de(); gen(j =, W.val, c, addr); \}$

$if\ (X.quad \neq null)\ then\ gen(j, -, -, X.quad);$

$p = lookup(tblptr.top, next);$

$S.nextList = merge(p.address, U.brkList, X.brkList);$

$S.brkList = null; S.ctnList = null; \}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】假设$nxq = 100$ #

(14) $Slist \rightarrow Slist_1 MS$ $\{backPatch(Slist_1.nextList, M.quad);$
$Slist.nextList = S.nextList;$
$Slist.brkList = merge(Slist_1.brkList, S.brkList);$
$Slist.ctnList = merge(Slist_1.ctnList, S.ctnList);\}$
$S.ctnList = merge(S_1.ctnList, S_2.ctnList);\}$

100.$(=, 1, -, i)$	105.$(j, -, -, 107)$
101.$(j >, i, 100, 109)$	106.$(j, -, -, 109)$
102.$(j <, i, 50, 104)$	107.$(+, i, 1, i)$
103.$(j, -, -, 106)$	108.$(j, -, -, 101)$
104.$(j, -, -, 107)$	109.$(=, 0, -, x)$

$M\{quad = 104\}$

)

$B\{trueList = 102, falseList = 103\}$

$S\{brkList = 106, ctnList = 104, nextList = 105\}$

$S\{brkList = null, ctnList = null, nextList = null\}$

$M\{quad = 109\}$

$Slist\{brkList = null, ctnList = null, nextList = 106\}$

#

成功

$S\{brkList = 106, ctnList = null, nextList = null\}$

$M\{quad$

$else$

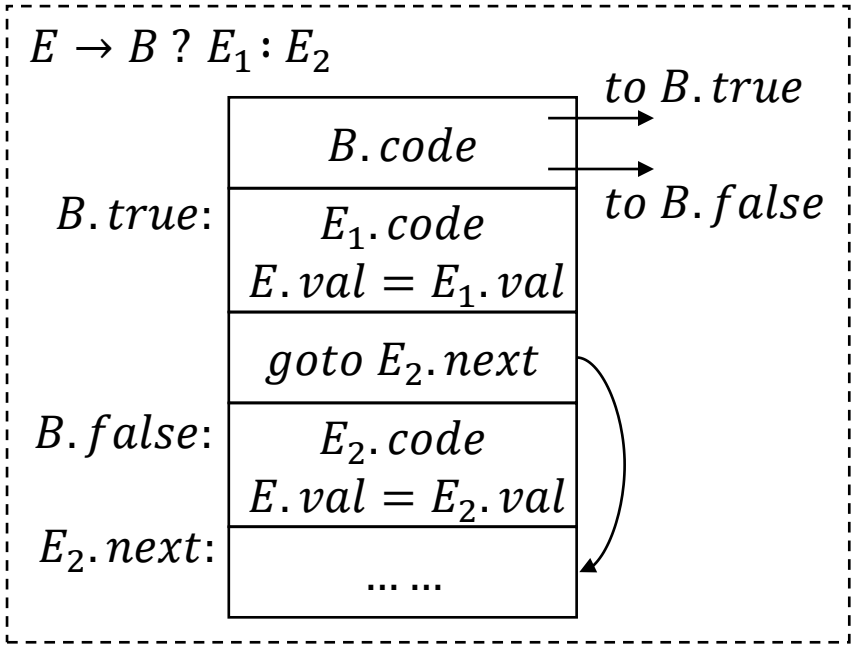
$N\{nextList = 103\}$

$S\{brkList = null, ctnList = 104, nextList = null\}$

```
for(i = 1 : 100)
    if (i < 50) continue;
    else break;
x = 0;
```

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

执行逻辑



山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(27) $Bt \rightarrow B?\, M_1 E_1$ $\{Bt.val = newTemp; gen(=, E_1.val, -, Bt.val);$

$Bt.nextList = makeList(nxq); gen(j, -, -, 0);$

$backPatch(B.trueList, M_1.quad);$

$backPatch(B.falseList, nxq);\ \}$

(28) $E \rightarrow Bt : E_2$ $\{E.val = Bt.val; gen(=, E_2.val, -, Bt.val);$

$backPatch(Bt.nextList, nxq);\ \}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】假设$nxq = 100$ #

(28) $E \to Bt : E_2$ $\{E.val = Bt.val; gen(=, E_2.val, -, Bt.val);$ $backPatch(Bt.nextList, nxq); \}$
$backPatch(B.trueList, M_1.quad); backPatch(B.falseList, nxq); \}$

$E\{val = \$2\}$
$E\{val = 0\}$
;
$E\{val = \$3\}$
=
A
#

成功

100. $(j <, a, b, 107)$	107. $(+, y, z, \$2)$
101. $(j, -, -, 102)$	108. $(=, \$2, -, \$3)$
102. $(+, i, j, \$1)$	109. $(j, -, -, 111)$
103. $(j \leq, c, \$1, 105)$	110. $(=, 0, -, \$3)$
104. $(j, -, -, 110)$	111. $(=, \$3, -, x)$
105. $(jnz, e, -, 107)$	
106. $(j, -, -, 110)$	

$x = a < b||c <= i + j\&\&e$
$? \ y + z : 0;$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

关系运算符的结合

- 数学上关系运算的结合
  - $0 < x < y$
  - $x < y > z$
- 我们规定如下两个关系式等价
  - $E_1\theta_1E_2\theta_2 \ldots \theta_{n-1}E_n$
  - $E_1\theta_1E_2 \wedge E_2\theta_2E_3 \wedge \cdots \wedge E_{n-1}\theta_{n-1}E_n$
- 产生式
  - $B \rightarrow E_1\theta E_2$
  - $B \rightarrow B_1\theta E$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(6) $B \rightarrow E_1 \theta E_2$ $\{B.trueList = makeList(nxq); B.falseList = makeList(nxq+1);$

$gen(j\theta, E_1.val, E_2.val, 0); gen(j, -, -, 0); B.val2 = E_2.val; \}$

(33) $B \rightarrow B_1 \theta M E$ $\{backPatch(B_1.trueList, M.quad); B.val2 = E.val;$

$B.trueList = makeList(nxq); B.falseList = makeList(nxq+1);$

$gen(j\theta, B_1.val2, E.val, 0); gen(j, -, -, B_1.falseList); \}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】假设$nxq = 100$ #

(14) $Slist \rightarrow Slist_1 MS$ $\{backPatch(Slist_1.nextList, M.quad);$
$Slist.nextList = S.nextList;$
$Slist.brkList = merge(Slist_1.brkList, S.brkList);$
$Slist.ctnList = merge(Slist_1.ctnList, S.ctnList);\}$

$S\{nextList = null\}$
$M\{quad = 106\}$
)
$B\{trueList = 104. falseList = 105. val2$
$S\{nextList = null\}$
$M\{quad = 107\}$
$Slist\{nextList = null\}$
#

成功

100. $(+, a, b, \$1)$
101. $(j <, x, \$1, 103)$
102. $(j, -, -, 107)$
103. $(+, y, z, \$2)$
104. $(j >, \$1, \$2, 106)$
105. $(j, -, -, 107)$
106. $(=, 1, -, x)$
107. $(=, 0, -, z)$

```
if (x < a + b > y + z)
    x = 1;
z = 0;
```

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(12) $F \rightarrow T\ id($ $\{t = makeTable(tblptr.top); tblptr.push(t); offset.push(0);$

$category = formal; F.name = id.name; F.type = T.type;$

$gen(proc, id.name, -, -);\}$

(13) $G \rightarrow FM)NS$ $\{t = tblptr.pop(); w = offset.pop(); logProc(\ldots);$

$gen(endp, F.name, -, -);\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(1) $S \rightarrow return\ E;\ \{gen(ret, E.place, -, -); S.nextList = null;\}$

(2) $S \rightarrow return;\quad \{gen(ret, -, -, -); S.nextList = null;\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

过程调用

- s(a+b, z)

  $(+, a, b, \$1)$

  $(param, z, -, -)$

  $(param, \$1, -, -)$

  $(call, s, -, -)$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(3) $S \rightarrow E;$ $\{S.nextList = null;\}$

(4) $E \rightarrow id(Elist$ $\{gen(call, id.name, -, -); p = lookup(tblptr.top.parent, id.name);$

if $(p.category \neq function)$ then Error;

else if $(p.type = void)$ then $E.val = null;$

else { $E.val = newTemp; gen(=, id.name, -, E.val);$ } }

(5) $E \rightarrow id()$ 同(4)

(6) $Elist \rightarrow E, Elist_1$ $\{gen(param, E.val, -, -);\}$

(7) $Elist \rightarrow E)$ $\{gen(param, E.val, -, -);\}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】假设$nxq = 100$ #

(4) $E \rightarrow id(Elist$ $\{gen(call, id.name, -, -); p = lookup(tblptr.top.parent, id.name);$
if $(p.category \neq function)$ then Error;
else if $(p.type = void)$ then $E.val = null;$
else { $E.val = newTemp; gen(=, id.name, -, E.val);$ } }

)
$Elist$
,
$Elist$
;
$E\{val = \$3\}$
=
A
#

后续略

100. $(+, x, y, \$1)$
101. $(*, 2, i, \$2)$
102. $(param, \$2, -, -)$
103. $(param, \$1, -, -)$
104. $(call, sum, -, -)$
105. $(=, sum, -, \$3)$
106. $(=, \$3, -, z)$

$z = sum(x + y, 2 * i);$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

类型检查

- 类型检查即构造类型表达式，将类型相关的规则赋给文法的语义规则，从而避免因类型不匹配导致的运行错误。
  - 如果类型检查在编译时进行，则称为静态类型检查；
  - 如果类型检查在运行时进行，则称为动态类型检查。
- 如果不需要动态检查类型错误，则称为良类型系统。
- 一个语言称为良类型的，如果它的编译器能保证编译通过的程序运行时不会出现类型错误。
  - 有些检查只能动态进行，如数组下标越界，编译器无法保证运行时下标一定落在限定范围内。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

类型表达式

- 类型表达式或者是基本类型，或者由类型构造符施加到类型表达式得到

I. 一个基本类型是一个类型表达式
   - 为了简化描述，使用int和$real$分别表示整型和实型这两类代表性的基本类型；
   - 专用基本类型$error$，表示类型错误；
   - 基本类型$void$，表示没有数据类型。

II. 类型表达式可以命名，因此一个类型名是一个类型表达式。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

类型表达式

III. 用类型构造符施加于类型表达式，得到一个新的类型表达式

- 数组。如果T是一个类型表达式，则$array(I,T)$是一个类型表达式，表示一个数组类型，其中I是下标集合。如$array(2,3,4,int)$或$array(0:2,1:3,0:4,int)$。
- 乘积。如果T_1和T_2是两个类型表达式，则其笛卡尔（Cartesian）乘积$T_1 \times T_2$是一个类型表达式，算符“×”为左结合。
- 结构体。结构体类型可以看做各成员类型的笛卡尔积，但成员有名字，名字不同则成员不同。因此，如$struct(Point)$可以表示结构体类型，其具体的类型表达式可能为$struct((Coor \times array(10,2,real) \times (Num \times int))$。
- 指针。如果T是一个类型表达式，则$pointer(T)$表示指向T类型对象的指针。
- 函数。函数可以看做将形参类型加工为返回值类型的映射，如函数int sum(int x, int y)，其类型表达式为： $int \times int \rightarrow int$。

IV. 类型表达式可以包含变量，变量的值是类型表达式。

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

翻译模式

(4) $E \rightarrow E_1 \theta E_2$ （θ为算术算符）

{ if $E_1.type = int \wedge E_2.type = int$ then $E.type = int$;

else if $E_1.type = real \wedge E_2.type = real$ then $E.type = real$;

else $E.type = error$; }

(5) $E \rightarrow E_1 \theta E_2$ （θ为关系算符）

{ if $E_1.type = int \wedge E_2.type = int$ then $E.type = bool$;

else if $E_1.type = real \wedge E_2.type = real$ then $E.type = bool$;

else $E.type = error$; }

(6) $E \rightarrow E_1 \theta E_2$ （θ为逻辑算符）

{ if $E_1.type = bool \wedge E_2.type = bool$ then $E.type = bool$;

else $E.type = error$; }

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

(1) $E \rightarrow E_1 \theta E_2$ （θ为算术算符）

{ $E.val = newTemp$;

if $E_1.type = int \wedge E_2.type = int$ then

{ $gen(\theta_i, E_1.val, E_2.val, E.val)$; $E.type = int$; }

else if $E_1.type = real \wedge E_2.type = real$ then

{ $gen(\theta_r, E_1.val, E_2.val, E.val)$; $E.type = real$; }

else if $E_1.type = int \wedge E_2.type = real$ then

{ $t = newTemp$; $gen(i2r, E_1.val, -, t)$;

$gen(\theta_r, t, E_2.val, E.val)$; $E.type = real$; }

else if $E_1.type = real \wedge E_2.type = int$ then

{ $t = newTemp$; $gen(i2r, E_2.val, -, t)$;

$gen(\theta_r, E_1.val, t, E.val)$; $E.type = real$; }

else { $E.type = error$; Error; } }

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

(2) $E \rightarrow E_1 \theta E_2$ （θ为关系算符）

$\{ E.val = newTemp;$

if $E_1.type = int \wedge E_2.type = int$ then $\{ gen(j\theta_i, E_1.val, E_2.val, nxq + 3);\}$

else if $E_1.type = real \wedge E_2.type = real$ then $\{ gen(j\theta_r, E_1.val, E_2.val, nxq + 3);\}$

else if $E_1.type = int \wedge E_2.type = real$ then

$\{ t = newTemp; gen(i2r, E_1.val, -, t);\ gen(j\theta_r, t, E_2.val, nxq + 3);\}$

else if $E_1.type = real \wedge E_2.type = int$ then

$\{ t = newTemp; gen(i2r, E_2.val, -, t); gen(j\theta_r, E_1.val, t, E.val);\}$

else $\{ E.type = error;$ Error; $\}$

$E.type = bool; gen(=, 0, -, E.val); gen(j, -, -, nxq + 2); gen(=, 1, -, E.val);\ \}$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

【例】假设$nxq = 100$， x和y为$real$， i和j为int #

(1) $E \to E_1 \theta E_2$ （θ为算术算符） { $E.val = newTemp;$
if $E_1.type = int \wedge E_2.type = int$ then
{ $gen(\theta_i, E_1.val, E_2.val, E.val); E.type = int;$ }
else if $E_1.type = real \wedge E_2.type = real$ then
{ $gen(\theta_r, E_1.val, E_2.val, E.val); E.type = real;$ }
else if $E_1.type = int \wedge E_2.type = real$ then
{ $t = newTemp; gen(i2r, E_1.val, -, t);$
$gen(\theta_r, t, E_2.val, E.val); E.type = real;$ }
else if $E_1.type = real \wedge E_2.type = int$ then
{ $t = newTemp; gen(i2r, E_2.val, -, t);$
$gen(\theta_r, E_1.val, t, E.val); E.type = real;$ }
else { $E.type = error;$ Error; } }

$E\{val = \$1\}$
;
$E\{val = \$2\}$
=
A
#

后续略

100. $(*_i, i, j, \$1)$
101. $(i2r, \$1, -, \$3)$
102. $(+_r, y, \$3, \$2)$
103. $(=, \$2, -, x)$

$x = y + i * j;$

山东大学 SHANDONG UNIVERSITY 郑艳伟，zhengyw@sdu.edu.cn

(1) $E \rightarrow (T)E_1$

{ if $T.type = int \wedge E_1.type = int$ then

{ $E.val = E_1.val;\ E.type = int;$ }

else if $T.type = real \wedge E_1.type = real$ then

{$E.val = E_1.val;\ E.type = real;$ }

else if $T.type = int \wedge E_1.type = real$ then

{ $E.val = newTemp; gen(r2i, E_1.val, -, E.val);\ E.type = int;$ }

else if $T.type = real \wedge E_1.type = int$ then

{ $E.val = newTemp; gen(i2r, E_1.val, -, E.val);\ E.type = real;$ }

else { $E.type = error;$ Error; } }

第 7 章 语法制导翻译与中间代码生成 内容小结

- 为上下文无关文法的每个符号配备若干属性，称为属性文法。
- 语法制导翻译是为文法的每个产生式配上一组语义规则，在语法分析的同时执行它，完成语义分析和中间代码生成工作。
- S–属性文法是只含有综合属性的文法；L–属性文法一次遍历就计算出所有属性值，且具有很强的表达能力。
- 声明语句的翻译主要是填写符号表；声明语句包括过程定义、形参声明、变量声明、数组声明、结构体声明等语法成分。
- 表达式与赋值语句翻译包括算术表达式、布尔表达式、数组引用、结构体引用、地址和指针引用等语法成分的翻译。
- 控制语句翻译包括作为条件控制的布尔表达式、if 和 while 语句、C 风格 for 语句、Matlab 风格 for 语句、标号–goto 语句、switch 语句、三目运算符、break 和 continue 语句、关系运算符结合语句。
- 过程调用的翻译包括过程开始与结束标记、返回语句、过程调用的翻译。
- 类型表达式即构造一个类型描述系统，使类型信息能够通过文法符号的属性进行传递，并设定语义规则进行计算。

第七章作业

【作业7-1】本章中，关系式$i^{(1)} < i^{(2)}$被翻译成相继的两个四元式：

$(j <, i^{(1)}, i^{(2)}, -)$ // 真出口

$(j, -, -, -)$ // 假出口

这种翻译常常浪费一个四元式。如果我们翻译成如下四元式：

$(j \geq, i^{(1)}, i^{(2)}, -)$ // 假出口跳转，真出口自动滑到下一个四元式

那么，在$i^{(1)} < i^{(2)}$的情况下就不发生跳转（自动滑下来）。但若这个关系后有一个或运算，则另一个无条件转移指令是不可省的，例如$if\ A < B \vee C < D\ then\ x = y$

100: $(j \geq, A, B, 102)$

101: $(j, -, -, 103)$ // 或运算前的无条件跳转不能省略

102: $(j \geq, C, D, 104)$

103: $(=, y, -, x)$

请按上述要求改写翻译布尔表达式的语义动作。

第七章作业

【作业7-2】根据本章所述翻译模式，将如下语句翻译为四元式，假设所有变量已声明，数据类型均为整型。

```
i = 1;
j = 1;
while i ≤ 9 do
begin
        while j ≤ 9 do
        begin
                a[i, j] = i * j;
                j = j + 1;
        end
        i = i + 1;
end
```

山东大学
SHANDONG UNIVERSITY

第七章 语法制导翻译与中间代码生成

The End

谢谢

授 课 教 师 ：郑艳伟
手 机 ：18614002860（微信同号）
邮 箱 ：zhengyw@sdu.edu.cn