



山东大学
SHANDONG UNIVERSITY

编译原理

第七章 语义分析和中间代码生成

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

第七章 语义分析和中间代码生成

□ 7.1 说明语句

- 7.1.1 过程中的说明语句
- 7.1.2 保留作用域的信息

□ 7.2 赋值语句的翻译

- 7.2.1 简单算术表达式及赋值语句
- 7.2.2 数组元素的引用
- 7.2.3 类型转换

□ 7.3 布尔表达式的翻译

- 7.3.1 数值表示法
- 7.3.2 作为条件控制的布尔式翻译

□ 7.4 控制语句的翻译

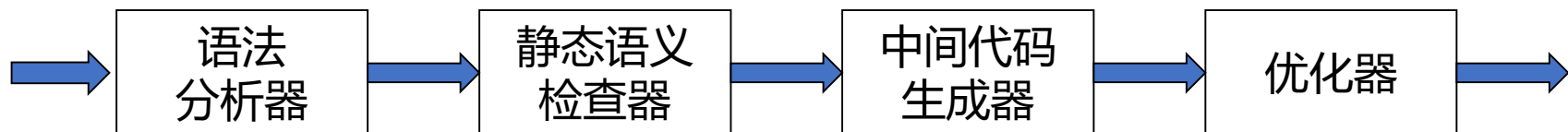
- 7.4.1 控制流语句
- 7.4.2 标号与goto语句
- 7.4.3 Case语句的翻译

□ 7.5 过程调用的处理

□ 7.6 类型检查

- 7.6.1 类型系统
- 7.6.2 类型检查器的规格说明
- 7.6.3 函数和运算符的重载
- 7.6.4 多态函数

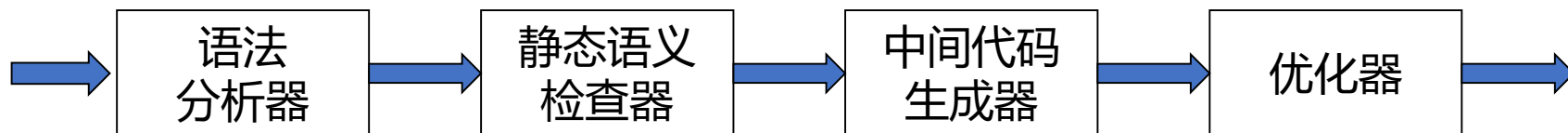
第七章 语义分析和中间代码生成



□ 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。

第七章 语义分析和中间代码生成



□ 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**。

第七章 语义分析和中间代码生成

□ 7.1 说明语句

➤ 7.1.1 过程中的说明语句

➤ 7.1.2 保留作用域的信息

□ 7.2 赋值语句的翻译

➤ 7.2.1 简单算术表达式及赋值语句

➤ 7.2.2 数组元素的引用

➤ 7.2.3 类型转换

□ 7.3 布尔表达式的翻译

➤ 7.3.1 数值表示法

➤ 7.3.2 作为条件控制的布尔式翻译

□ 7.4 控制语句的翻译

➤ 7.4.1 控制流语句

➤ 7.4.2 标号与goto语句

➤ 7.4.3 Case语句的翻译

□ 7.5 过程调用的处理

□ 7.6 类型检查

➤ 7.6.1 类型系统

➤ 7.6.2 类型检查器的规格说明

➤ 7.6.3 函数和运算符的重载

➤ 7.6.4 多态函数

7.1.1 过程中的说明语句

□ 过程中是说明语句需要处理的问题：

- ① 过程中的说明语句形式： $id_1, id_2, \dots, id_n : type$
- ② 每个变量需要记录名字、类型、字宽信息，分别用属性name、type、width记录
- ③ 当新出现一个名字时，需要记录进符号表，用offset记录该名字在目标代码运行时数据区中的相对地址偏移量，在识别前置初值为0。
- ④ 过程enter(name, type, offset)用来把名字name填入到符号表，并给出该名字的类型type及在过程数据区中的相对地址offset。
- ⑤ 全局变量的符号表，在编译器运行时的某个合适时刻创建，在编译器退出时销毁；每个过程的符号表，在过程开始时创建，在过程结束时出栈，但应保持可访问状态。
- ⑥ 假定整数类型域宽为4，实型域宽为8。

7.1.1 过程中的说明语句

□ 过程中的说明语句翻译模式:

$$P \rightarrow \{offset = 0\}D$$
$$D \rightarrow D; D$$
$$D \rightarrow id:T \quad \{enter(id.name, T.type, offset);$$
$$offset = offset + T.width\}$$
$$T \rightarrow integer \quad \{T.type = integer; T.width = 4\}$$
$$T \rightarrow real \quad \{T.type = real; T.width = 8\}$$
$$T \rightarrow array[num] \text{ of } T_1 \quad \{T.type = array(num.val, T_1.type);$$
$$T.width = num.val \times T_1.width\}$$
$$T \rightarrow ^T_1 \quad \{T.type = pointer(T_1.type); T.width = 4\}$$

7.1.1 过程中的说明语句

□ 过程中的说明语句翻译模式:

$$D \rightarrow D; D$$
$$D \rightarrow D:T$$
$$D \rightarrow id$$

【例6.21】

$$D \rightarrow L:T$$
$$T \rightarrow integer \mid char$$
$$L \rightarrow L, id \mid id$$


【修改】

$$D \rightarrow id L$$
$$L \rightarrow, id L \mid : T$$
$$T \rightarrow integer \mid char$$

□ 过程中的说明语句翻译模式:

$$P \rightarrow \{offset = 0\}D$$
$$D \rightarrow D; D$$
$$D \rightarrow id\ L \quad \{enter(id.name, L.type, offset);$$
$$offset = offset + L.width\}$$
$$L \rightarrow, id\ L_1 \quad \{enter(id.name, L_1.type, offset);$$
$$offset = offset + L_1.width;$$
$$L.type = L_1.type; L.width = L_1.width\}$$
$$L \rightarrow :T \quad \{L.type = T.type; L.width = T.width\}$$
$$T \rightarrow integer \quad \{T.type = integer; T.width = 4\}$$
$$T \rightarrow real \quad \{T.type = real; T.width = 8\}$$
$$T \rightarrow array[num] \text{ of } T_1 \quad \{T.type = array(num.val, T_1.type);$$
$$T.width = num.val \times T_1.width\}$$
$$T \rightarrow ^T_1 \quad \{T.type = pointer(T_1.type); T.width = 4\}$$

□ 过程中的说明语句翻译模式:

$$P \rightarrow MD \quad D \rightarrow D; D$$

$$M \rightarrow \varepsilon \quad \{offset = 0\}$$

$$D \rightarrow id L \quad \{enter(id.name, L.type, offset);$$

$$offset = offset + L.width\}$$

$$L \rightarrow, id L_1 \quad \{enter(id.name, L_1.type, offset);$$

$$offset = offset + L_1.width;$$

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

$$L \rightarrow : T \quad \{L.type = T.type; L.width = T.width\}$$

$$T \rightarrow integer \quad \{T.type = integer; T.width = 4\}$$

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

$$T \rightarrow array[num] \text{ of } T_1 \quad \{T.type = array(num.val, T_1.type);$$

$$T.width = num.val \times T_1.width\}$$

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

□ 【例7.8】分析句子: $p, q, r: real$

$$D \rightarrow id\ L \quad \{enter(id.name, L.type, offset);$$

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

步骤	文法符号栈	输入串
1	#	$p, q, r: real\#$
2	$\#M$	$p, q, r: real\#$
3	$\#Mp, q, r: real$	#
4	$\#Mp, q, r: T$	#
5	$\#Mp, q, rL$	#
6	$\#Mp, qL$	#
7	$\#MpL$	#
8	$\#MD$	#
9	$\#P$	#
10	acc	#

id	type	offset
r	real	0
q	real	8
p	real	16

offset=24

$$L.type = real; L.width = 8$$

□ 过程中的说明语句翻译模式:

$$P \rightarrow MD \quad D \rightarrow D; D$$

$$M \rightarrow \varepsilon \quad \{offset = 0\}$$

$$D \rightarrow id L \quad \{enter(val[top - 1].name, val[top].type, offset);$$

$$offset = offset + val[top].width\}$$

$$L \rightarrow , id L_1 \quad \{enter(val[top - 1].name, val[top].type, offset);$$

$$offset = offset + val[top].width;$$

$$val[ntop] = val[top]\}$$

$$L \rightarrow : T \quad \{val[ntop] = val[top]\}$$

$$T \rightarrow integer \quad \{val[ntop].type = integer; val[ntop].width = 4\}$$

$$T \rightarrow real \quad \{val[ntop].type = real; val[ntop].width = 8\}$$

$$T \rightarrow array[num] \text{ of } T_1 \quad \{val[ntop].type = array(num.val, val[top].type);$$

$$val[ntop].width = num.val \times val[top].width\}$$

$$T \rightarrow ^T_1 \quad \{val[ntop].type = pointer(val[top].type); val[ntop].width = 4\}$$

第七章 语义分析和中间代码生成

□ 7.1 说明语句

- 7.1.1 过程中的说明语句
- 7.1.2 保留作用域的信息

□ 7.2 赋值语句的翻译

- 7.2.1 简单算术表达式及赋值语句
- 7.2.2 数组元素的引用
- 7.2.3 类型转换

□ 7.3 布尔表达式的翻译

- 7.3.1 数值表示法
- 7.3.2 作为条件控制的布尔式翻译

□ 7.4 控制语句的翻译

- 7.4.1 控制流语句
- 7.4.2 标号与goto语句
- 7.4.3 Case语句的翻译

□ 7.5 过程调用的处理

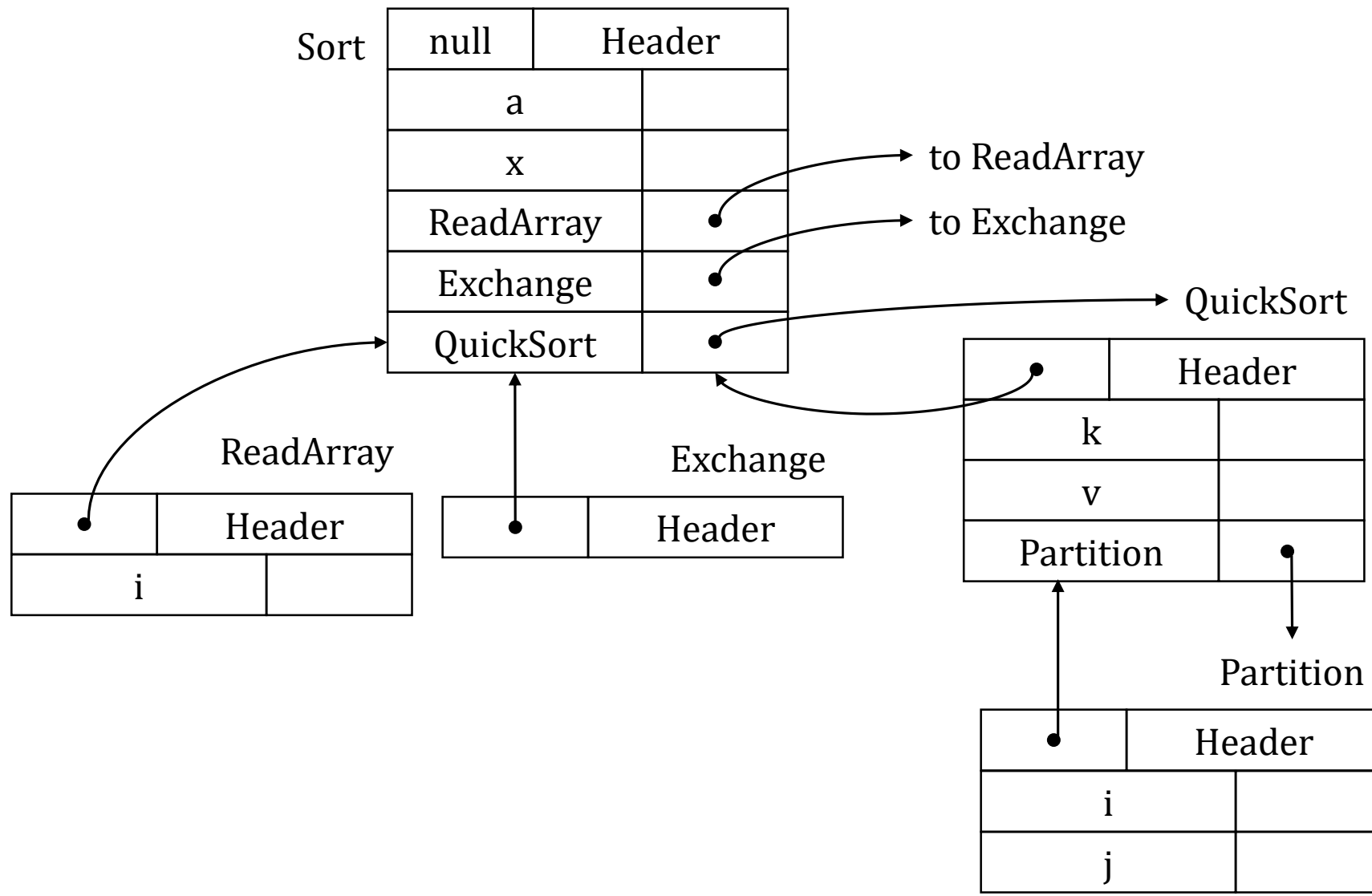
□ 7.6 类型检查

- 7.6.1 类型系统
- 7.6.2 类型检查器的规格说明
- 7.6.3 函数和运算符的重载
- 7.6.4 多态函数

```
① program Sort (input , output);  
②     var a: array[0..10] of integer;  
③     x: integer;  
④     procedure ReadArray;  
⑤         var i: integer;  
⑥         begin ... a... end {ReadArray};  
⑦     procedure Exchange (i, j: integer);  
⑧         begin x=a[i]; a[i] = a[j]; a[j] = x; end {Exchange};  
⑨     procedure QuickSort (m, n: integer);  
⑩         var k, v: integer;  
⑪         function Partition (y, z: integer) : integer;  
⑫             var i, j: integer;  
⑬             begin ...a...v... Exchange(i, j);... end {Partition};  
⑭         begin ... end {QuickSort};  
⑮     begin ... end {Sort}.
```



嵌套过程的符号表



保留作用域的信息

□ 语义规则中的操作:

- *mktable(previous)*: 创建一张新符号表, 并返回指向新表的指针; 参数 *previous* 指向先前创建的一张符号表。
- *enter(table, name, type, offset)*: 在指针 *table* 指向的符号表中, 为名字 *name* 建立一个新项, 并把类型 *type*、相对地址 *offset* 填入到该项中。
- *addwidth(table, width)*: 在指针 *table* 指向的符号表表头中, 记录下该表中所有名字占用的总宽度。
- *enterproc(table, name, newtable)*: 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。
- *tblptr* 是一个栈, 用于存放指向嵌套外层过程的符号表指针。
- *offset* 是一个栈, 用于存放变量的相对地址, 当过程结束时, *offset* 里记录的是过程占用的所有字节数。

□ 保留作用域信息:

$P \rightarrow MD$	$\{ \text{addwidth}(\text{top}(\text{tblptr}), \text{top}(\text{offset}));$ $\quad \text{pop}(\text{tblptr}); \text{pop}(\text{offset}); \}$
$M \rightarrow \varepsilon$	$\{ t = \text{mktable}(\text{null}); \text{push}(t, \text{tblptr}); \text{push}(0, \text{offset}); \}$
$D \rightarrow \text{proc id ND}; S$	$\{ t = \text{top}(\text{tblptr}); \text{addwidth}(t, \text{top}(\text{offset}));$ $\quad \text{pop}(\text{tblptr}); \text{pop}(\text{offset});$ $\quad \text{enterproc}(\text{top}(\text{tblptr}), \text{id.name}, t); \}$
$D \rightarrow \text{id } L$	$\{ \text{enter}(\text{top}(\text{tblptr}), \text{id.name}, L.\text{type}, \text{top}(\text{offset}));$ $\quad \text{top}(\text{offset}) = \text{top}(\text{offset}) + L.\text{width} \}$
$L \rightarrow, \text{id } L_1$	$\{ \text{enter}(\text{top}(\text{tblptr}), \text{id.name}, L_1.\text{type}, \text{top}(\text{offset}));$ $\quad \text{top}(\text{offset}) = \text{top}(\text{offset}) + L_1.\text{width};$ $\quad L.\text{type} = L_1.\text{type}, L.\text{width} = L_1.\text{width} \}$
$L \rightarrow : T$	$\{ L.\text{type} = T.\text{type};$
$N \rightarrow \varepsilon$	$\{ t = \text{mktable}(\text{top}(\text{tblptr})); \text{push}(t, \text{tblptr}); \text{push}(0, \text{offset}); \}$

第七章 语义分析和中间代码生成

□ 7.1 说明语句

- 7.1.1 过程中的说明语句
- 7.1.2 保留作用域的信息

□ 7.2 赋值语句的翻译

- 7.2.1 简单算术表达式及赋值语句
- 7.2.2 数组元素的引用
- 7.2.3 类型转换

□ 7.3 布尔表达式的翻译

- 7.3.1 数值表示法
- 7.3.2 作为条件控制的布尔式翻译

□ 7.4 控制语句的翻译

- 7.4.1 控制流语句
- 7.4.2 标号与goto语句
- 7.4.3 Case语句的翻译

□ 7.5 过程调用的处理

□ 7.6 类型检查

- 7.6.1 类型系统
- 7.6.2 类型检查器的规格说明
- 7.6.3 函数和运算符的重载
- 7.6.4 多态函数

7.2.1 简单算术表达式及赋值语句

□ 简单算术表达式及赋值语句的开始符号为 S , 来自如下部分的相应 V_N :

$$P \rightarrow MD$$

$$M \rightarrow \varepsilon$$

$$D \rightarrow id\ L \mid proc\ id\ ND; S$$

$$L \rightarrow, id\ L_1$$

$$L \rightarrow :T$$

$$N \rightarrow \varepsilon$$

7.2.1 简单算术表达式及赋值语句

□ 简单算术表达式及赋值语句的操作：

- *tblptr*：是一个栈，栈顶为当前过程的符号表，所以可以取到符号信息。
- *lookup(name)*：从 $top(tblptr)$ 符号表寻找名字，找到即返回，找不到则转到外围（上层）符号表继续查找，直到找到或者所有外围过程都找不到为止。
- *gen(op, arg1, arg2, result)*：生成三地址代码。
- *newtemp*：是一个方法，生成一个临时变量。
- *place*：是一个属性，存放文法符号的值（变量）的名字。

□ 简单算术表达式及赋值语句:

$S \rightarrow id = E$	$\{p = \text{lookup}(id.name);$ $\text{if } p \neq \text{null then gen}(=, E.place, -, p); \text{else error}; \}$
$E \rightarrow E_1 + E_2$	$\{E.place = \text{newtemp}; \text{gen}(+, E_1.place, E_2.place, E.place)\}$
$E \rightarrow E_1 * E_2$	$\{E.place = \text{newtemp}; \text{gen}(*, E_1.place, E_2.place, E.place)\}$
$E \rightarrow -E_1$	$\{E.place = \text{newtemp}; \text{gen}(@, E_1.place, -, E.place)\}$
$E \rightarrow (E_1)$	$\{E.place = E_1.place\}$
$E \rightarrow id$	$\{p = \text{lookup}(id.name);$ $\text{if } p \neq \text{null then } E.place = p; \text{else error}; \}$

7.2.1 简单算术表达式及赋值语句

□

$$S \rightarrow id = E \quad \{p = lookup(id.name);$$

$$if \ p \neq null \ then \ gen(=, E.place, -, p); \ else \ error; \}$$

$$E.place)\}$$

步骤	文法符号栈	输入串	动作
1	#	$x = (a + b) * -c\#$	初始
2	$\#x = (a$	$+b) * -c\#$	移进
3	$\#x = (E$	$+b) * -c\#$	归约
4	$\#x = (E + b$	$) * -c\#$	移进
5	$\#x = (E + E$	$) * -c\#$	归约
6	$\#x = (E$	$) * -c\#$	归约
7	$\#x = (E)$	$* -c\#$	移进
8	$\#x = E$	$* -c\#$	归约
9	$\#x = E * -c$	#	移进
10	$\#x = E * -E$	#	归约
11	$\#x = E * E$	#	归约
12	$\#x = E$	#	归约
13	$\#S$	#	归约
14	$\#S$	#	成功

三地址码

$(+, a, b, T1)$
$(@, c, -, T2)$
$(*, T1, T2, T3)$
$(=, T3, -, x)$

$E.place = T2$
$E.place = T3$

栈属性
(给人看的)

第七章 语义分析和中间代码生成

□ 7.1 说明语句

- 7.1.1 过程中的说明语句
- 7.1.2 保留作用域的信息

□ 7.2 赋值语句的翻译

- 7.2.1 简单算术表达式及赋值语句
- 7.2.2 数组元素的引用
- 7.2.3 类型转换

□ 7.3 布尔表达式的翻译

- 7.3.1 数值表示法
- 7.3.2 作为条件控制的布尔式翻译

□ 7.4 控制语句的翻译

- 7.4.1 控制流语句
- 7.4.2 标号与goto语句
- 7.4.3 Case语句的翻译

□ 7.5 过程调用的处理

□ 7.6 类型检查

- 7.6.1 类型系统
- 7.6.2 类型检查器的规格说明
- 7.6.3 函数和运算符的重载
- 7.6.4 多态函数

7.2.2 数组元素的引用

- 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： $i \times w + (base - low \times w)$
 - 令 $C = base - low \times w$ ，可以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： $i \times w + C$ 。

7.2.2 数组元素的引用

□ 行存储的二维数组 $A[i_1, i_2]$ 地址:

$$\triangleright base + [(i_1 - low_1) \times n_2 + i_2 - low_2] \times w$$

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

	第 low_2 列						第 i_2 列		共 n_2 列
第 low_1 行	$A[l_1, l_2]$	$A[l_1, l_2 + 1]$	$A[l_1, l_2 + 2]$			$A[l_1, i_2]$	$A[l_1, l_2 + n_2 - 1]$
	$A[l_1 + 1, l_2]$	$A[l_1 + 1, l_2 + 1]$	$A[l_1 + 1, l_2 + 2]$			$A[l_1 + 1, i_2]$	$A[l_1 + 1, l_2 + n_2 - 1]$

第 i_1 行	$A[i_1, l_2]$	$A[i_1, l_2 + 1]$	$A[i_1, l_2 + 2]$			$A[i_1, i_2]$	$A[i_1, l_2 + n_2 - 1]$

共 n_1 行	$A[l_1 + n_1 - 1, l_2]$	$A[l_1 + n_1 - 1, l_2 + 1]$	$A[l_1 + n_1 - 1, l_2 + 2]$			$A[l_1 + n_1 - 1, i_2]$	$A[l_1 + n_1 - 1, l_2 + n_2 - 1]$

7.2.2 数组元素的引用

□ 行存储的二维数组 $A[i_1, i_2]$ 地址:

$$\begin{aligned} &\text{➤ } base + [(i_1 - low_1) \times n_2 + i_2 - low_2] \times w \\ &= (i_1 \times n_2 + i_2) \times w + [base - (low_1 \times n_2 + low_2) \times w] \end{aligned}$$

□ 行存储的多维数组 $A[i_1, i_2, \dots, i_k]$ 地址:

- 基准地址: $C = base - ((\dots((low_1 \times n_2 + low_2) \times n_3) \dots) \times n_k + low_k) \times w$
- 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: $((\dots(i_1 \times n_2 + i_2) \times n_3) \dots) \times n_k + i_k) \times w$

7.2.2 数组元素的引用

□ 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

7.2.2 数组元素的引用

□ 最终文法:

$$\textcircled{1} S \rightarrow L = E$$

$$\textcircled{2} E \rightarrow E + E$$

$$\textcircled{3} E \rightarrow E * E$$

$$\textcircled{4} E \rightarrow -E$$

$$\textcircled{5} E \rightarrow (E)$$

$$\textcircled{6} E \rightarrow L$$

// E 也可以是数组, 如果是变量, 也需要通过 L 过渡

$$\textcircled{7} L \rightarrow Elist]$$

// L 是数组

$$\textcircled{8} L \rightarrow id$$

// L 是普通变量

$$\textcircled{9} Elist \rightarrow Elist, E$$

$$\textcircled{10} Elist \rightarrow id[E$$

7.2.2 数组元素的引用

□ *Elist*的属性:

- ① *array*, 记录指向符号表中相应数组名字表项的指针。
- ② *ndim*, 记录*Elist*中下标表达式的个数, 即维数。
- ③ *place*, 表示临时变量, 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:

- *place*, 存放变量的名字。
- *offset*, 简单名字为*null*, 数组则为地址偏移量。

□ 函数:

- *limit(array, j)*, 返回 n_j , 即由*array*所指示的数组, 其第*j*维的长度。

□ 多维数组 $A[i_1, i_2, \dots, i_k]$ 的前*m*维下标:

- 动态下标: $(\dots((i_1 \times n_2 + i_2) \times n_3) \dots) \times n_m + i_m$
- 递归计算: $e_1 = i_1, e_2 = e_1 \times n_2 + i_2, \dots, e_m = e_{m-1} \times n_m + i_m$

翻译模式

(1) $S \rightarrow L = E$

```
{ if  $L.offset = null$       //  $L$ 是简单变量  
     $gen(=, E.place, -, L.place);$   
else                               //  $L$ 是数组  
     $gen(=, E.place, -, L.place[L.offset]);$  }
```

(2) $E \rightarrow E_1 + E_2$

```
{  $E.place = newtemp;$   
   $gen(+, E_1.place, E_2.place, E.place);$  }
```

(3) $E \rightarrow E_1 * E_2$

```
{  $E.place = newtemp;$   
   $gen(*, E_1.place, E_2.place, E.place);$  }
```

翻译模式

(4) $E \rightarrow -E_1$

```
{ E.place = newtemp;  
  gen(@, E1.place, -, E.place); }
```

(5) $E \rightarrow (E_1)$

```
{ E.place = E1.place; }
```

(6) $E \rightarrow L$

```
{ if L.offset = null    // L是简单变量
```

```
  E.place = L.place;
```

```
else {                  // L是数组，转到E后会丢失数组信息，因此此处赋值
```

```
  E.place = newtemp;
```

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

翻译模式

(7) $L \rightarrow Elist]$

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

```
 $gen(-, Elist.array, C, L.place)$ ; //  $C$ 的计算参考前述公式, 在符号表中
```

```
 $L.offset = newtemp$ ;
```

```
 $gen(*, w, Elist.place, L.offset)$ ; } //  $w$ 在符号表中
```

(8) $L \rightarrow id$

```
{  $L.palce = id.place$ ;
```

```
 $L.offset = null$ ; }
```


翻译模式

(9) $Elist \rightarrow Elist_1, E$

```
{ t = newtemp;  
  m = Elist.ndim + 1;  // 用一次维度+1  
  gen(*, Elist1.place, limit(Elist1.array, m), t);  
  gen(+, t, E.place, t);  // 递归计算  $e_k = e_{k-1} \times n_k + i_k$   
  Elist.array = Elist1.array;  
  Elist.palce = t;  
  Elist.ndim = m; }
```

(10) $Elist \rightarrow id[E$

```
{ Elist.palce = E.place;  
  Elist.ndim = 1;  
  Elist.array = id.place; }
```

【例7.10】赋值语句 $x = A[y, z]$; 其中 $A[10, 20]: integer; low = 1$ 三地址码

步骤	文法符号栈	输入串	动作
1	#	$x = A[y, z]\#$	初始
2	$\#x$	$= A[y, z]\#$	移进
3	$\#L$	$= A[y, z]\#$	归约
4	$\#L = A[y$	$, z]\#$	移进
5	$\#L = A[L$	$, z]\#$	归约
6	$\#L = A[E$	$, z]\#$	归约
7	$\#L = Elist$	$, z]\#$	归约
8	$\#L = Elist, z$	$]\#$	移进
9	$\#L = Elist, L$	$]\#$	归约
10	$\#L = Elist, E$	$]\#$	归约
11	$\#L = Elist$	$]\#$	归约
12	$\#L = Elist]$	$\#$	移进
13	$\#L = L$	$\#$	归约
14	$\#L = E$	$\#$	归约
15	$\#S$	$\#$	归约
16	$\#S$	$\#$	成功

$(*, y, 20, T1)$
$(+, T1, z, T1)$
$(-, A, 21, T2)$
$(*, 4, T1, T3)$
$(=, T2[T3], -, T4)$
$(=, T4, -, x)$

(9) $Elist \rightarrow Elist_1, E$

{ $t = newtemp$;
 $Elist = Elist_1$;

(1) $S \rightarrow L = E$

{ if $L.offset = null$
 $gen(=, E.place, -, L.place)$;
else
 $gen(=, E.place, -, L.place[L.offset])$; }

$E.place = z$
$E.place = T4$
$L.place = x, .offset = null$

栈属性 (给人看的)

7.2.2 数组元素的引用

□ 多维数组 $A[n_1, n_2, \dots, n_k]: Type$ 需要记入符号表的内容:

- 数组类型: $Type$
- 数组名称: $A (base)$
- 数组偏移量: $offset$
- 字宽: w
- 每个维度的元素数: n_i
- 基准地址: $C = ((\dots ((low_1 \times n_2 + low_2) \times n_3) \dots) \times n_k + low_k) \times w$

第七章 语义分析和中间代码生成

□ 7.1 说明语句

- 7.1.1 过程中的说明语句
- 7.1.2 保留作用域的信息

□ 7.2 赋值语句的翻译

- 7.2.1 简单算术表达式及赋值语句
- 7.2.2 数组元素的引用
- 7.2.3 类型转换

□ 7.3 布尔表达式的翻译

- 7.3.1 数值表示法
- 7.3.2 作为条件控制的布尔式翻译

□ 7.4 控制语句的翻译

- 7.4.1 控制流语句
- 7.4.2 标号与goto语句
- 7.4.3 Case语句的翻译

□ 7.5 过程调用的处理

□ 7.6 类型检查

- 7.6.1 类型系统
- 7.6.2 类型检查器的规格说明
- 7.6.3 函数和运算符的重载
- 7.6.4 多态函数

7.2.3 类型转换

□ 当两个不同类型的量运算时，需要二选一：

- 拒绝运算
- 自动进行类型转换

【例7.11】 $x = y + i * j$ ，其中 i, j 为整型， y 为实型，对应四元式为：

① $(*, i, j, T1)$

② $(int2real, T1, null, T2)$

③ $(+, y, T2, T3)$

④ $(=, T3, null, x)$

翻译模式

 $E \rightarrow E_1 \theta E_2$

```
{ E.place = newtemp;  
if E1.type == integer && E2.type == integer {  
    gen( $\theta^i$ , E1.place, E2.place, E.place); E.type = integer; }  
else if E1.type == real && E2.type == real {  
    gen( $\theta^r$ , E1.place, E2.place, E.place); E.type = real; }  
else if E1.type == integer && E2.type == real {  
    u = newtemp; gen(int2real, E1.place, -, u);  
    gen( $\theta^r$ , u, E2.place, E.place); E.type = real; }  
else if E1.type == real && E2.type == integer {  
    u = newtemp; gen(int2real, E2.place, -, u);  
    gen( $\theta^r$ , E1.place, u, E.place); E.type = real; }  
else E.type = type_error; }
```

第七章 语义分析和中间代码生成

□ 7.1 说明语句

- 7.1.1 过程中的说明语句
- 7.1.2 保留作用域的信息

□ 7.2 赋值语句的翻译

- 7.2.1 简单算术表达式及赋值语句
- 7.2.2 数组元素的引用
- 7.2.3 类型转换

□ 7.3 布尔表达式的翻译

- 7.3.1 数值表示法
- 7.3.2 作为条件控制的布尔式翻译

□ 7.4 控制语句的翻译

- 7.4.1 控制流语句
- 7.4.2 标号与goto语句
- 7.4.3 Case语句的翻译

□ 7.5 过程调用的处理

□ 7.6 类型检查

- 7.6.1 类型系统
- 7.6.2 类型检查器的规格说明
- 7.6.3 函数和运算符的重载
- 7.6.4 多态函数

7.3 布尔表达式的翻译

□ 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: \neg, \wedge, \vee ;
- \wedge, \vee 服从左结合规则, \neg 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。

7.3 布尔表达式的翻译

□ **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*

第七章 语义分析和中间代码生成

□ 7.1 说明语句

- 7.1.1 过程中的说明语句
- 7.1.2 保留作用域的信息

□ 7.2 赋值语句的翻译

- 7.2.1 简单算术表达式及赋值语句
- 7.2.2 数组元素的引用
- 7.2.3 类型转换

□ 7.3 布尔表达式的翻译

- 7.3.1 数值表示法
- 7.3.2 作为条件控制的布尔式翻译

□ 7.4 控制语句的翻译

- 7.4.1 控制流语句
- 7.4.2 标号与goto语句
- 7.4.3 Case语句的翻译

□ 7.5 过程调用的处理

□ 7.6 类型检查

- 7.6.1 类型系统
- 7.6.2 类型检查器的规格说明
- 7.6.3 函数和运算符的重载
- 7.6.4 多态函数

7.3.1 数值表示法

【例7.12】布尔表达式生成四元式: $a \vee b \wedge \neg c$

① $(\neg, c, -, T1)$

② $(\wedge, b, T1, T2)$

③ $(\vee, a, T2, T3)$

【例7.13】关系式 $a < b$ 可以看作: *if a < b then 1 else 0*

100. $(j <, a, b, 103)$

101. $(=, 0, -, T1)$

102. $(j, -, -, 104)$

103. $(=, 1, -, T1)$

104. ...

翻译模式

□ *nextstat*: 为将要生成尚未生成的四元式地址索引, 每生成一条自动加1

① $E \rightarrow E_1 \vee E_2$ $\{E.place = newtemp; gen(\vee, E_1.place, E_2.place, E.place); \}$

② $E \rightarrow E_1 \wedge E_2$ $\{E.place = newtemp; gen(\wedge, E_1.place, E_2.place, E.place); \}$

③ $E \rightarrow \neg E_1$ $\{E.place = newtemp; gen(\neg, E_1.place, -, E.place); \}$

④ $E \rightarrow (E_1)$ $\{E.place = E_1.place; \}$

⑤ $E \rightarrow id$ $\{E.place = id.place; \}$

⑥ $E \rightarrow id_1 \theta id_2$ $\{E.place = newtemp;$
 $gen(j\theta, id_1.place, id_2.place, nextstat + 3);$
 $gen(=, 0, -, E.place);$
 $gen(j, -, -, nextstat + 2);$
 $gen(=, 1, -, E.place); \}$

【例7.14】布尔表达式: $a < b \vee c \leq d \wedge e > f$, 其中nextstat=100
三地址码

$E \rightarrow id_1 \theta id_2$	$\{E.place = newtemp;$
$E \rightarrow E_1 \vee E_2$	$\{E.place = newtemp;$
$gen(\vee, E_1.place, E_2.place, E.place); \}$	

步骤	文法符号栈	输入串	动作
1	#	$a < b \vee c \leq d \wedge e > f \#$	初始
2	# $a < b$	$\vee c \leq d \wedge e > f \#$	移进
3	# E	$\vee c \leq d \wedge e > f \#$	归约
4	# $E \vee c \leq d$	$\wedge e > f \#$	移进
5	# $E \vee E$	$\wedge e > f \#$	归约
6	# $E \vee E \wedge e > f$	#	移进
7	# $E \vee E \wedge E$	#	归约
8	# $E \vee E$	#	归约
9	# E	#	归约
10	# E	#	成功

100: ($j <, a, b, 103$)
101: ($=, 0, -, T1$)
102: ($j, -, -, 104$)
103: ($=, 1, -, T1$)
104: ($j \leq, c, d, 107$)
105: ($=, 0, -, T2$)
106: ($j, -, -, 108$)
107: ($=, 1, -, T2$)
108: ($j >, e, f, 111$)
109: ($=, 0, -, T3$)
110: ($j, -, -, 112$)
111: ($=, 1, -, T3$)
112: ($\wedge, T2, T3, T4$)
113: ($\vee, T1, T4, T5$)

$E.place = T3$
$E.place = T4$
$E.place = T5$

栈属性

第七章 语义分析和中间代码生成

□ 7.1 说明语句

- 7.1.1 过程中的说明语句
- 7.1.2 保留作用域的信息

□ 7.2 赋值语句的翻译

- 7.2.1 简单算术表达式及赋值语句
- 7.2.2 数组元素的引用
- 7.2.3 类型转换

□ 7.3 布尔表达式的翻译

- 7.3.1 数值表示法
- 7.3.2 作为条件控制的布尔式翻译

□ 7.4 控制语句的翻译

- 7.4.1 控制流语句
- 7.4.2 标号与goto语句
- 7.4.3 Case语句的翻译

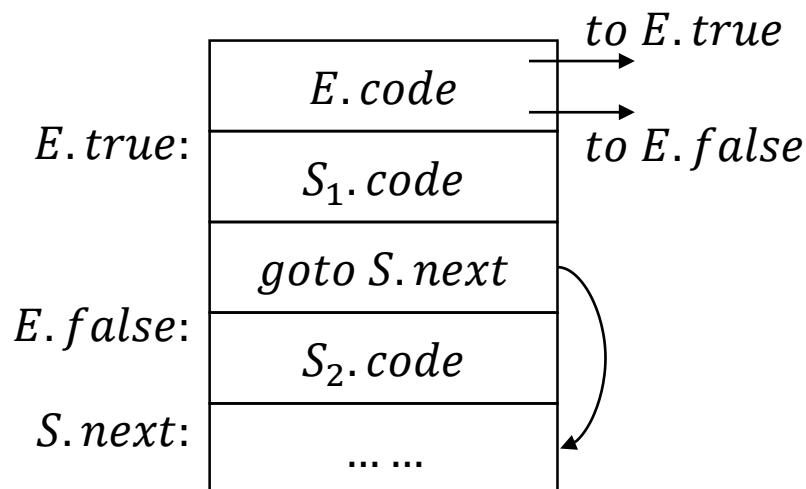
□ 7.5 过程调用的处理

□ 7.6 类型检查

- 7.6.1 类型系统
- 7.6.2 类型检查器的规格说明
- 7.6.3 函数和运算符的重载
- 7.6.4 多态函数

7.3.2 作为条件控制的布尔式翻译

- 条件语句 *if E then S₁ else S₂* 中的布尔表达式 *E*, 仅用于控制对 *S₁* 和 *S₂* 的选择, 不需要用一个临时变量保留其值, 因此可以为布尔式 *E* 设置两个出口:
- 真出口: 指向 *S₁* 的第一个四元式;
 - 假出口: 指向 *S₂* 的第一个四元式。



7.3.2 作为条件控制的布尔式翻译

【例7.15】 $\text{if } a > c \vee b < d \text{ then } S_1 \text{ else } S_2$

这两个语句都跳转到 L_2 ，但 L_2 需要分析到 S_1 才能确定，其它类似。

$\text{if } a > c \text{ goto } L_2$

$\text{goto } L_1$

$L_1: \text{if } b < d \text{ goto } L_2$

$\text{goto } L_3$

$L_2: (\text{关于 } S_1 \text{ 的代码序列})$

$\text{goto } L_{\text{next}}$

$L_3: (\text{关于 } S_2 \text{ 的代码序列})$

$L_{\text{next}}:$

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

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

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

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

104. $S_1 \dots$

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

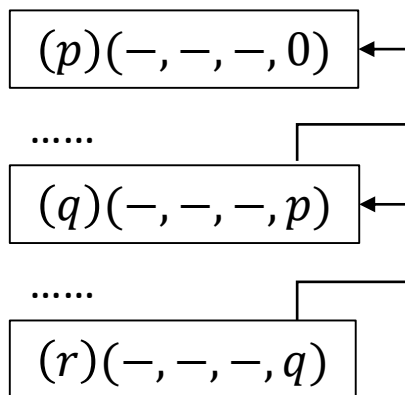
116. $S_2 \dots$

120. ...

7.3.2 作为条件控制的布尔式翻译

□ 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 E 赋予两个综合属性 $E.truelist$ 和 $E.falselist$, 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:

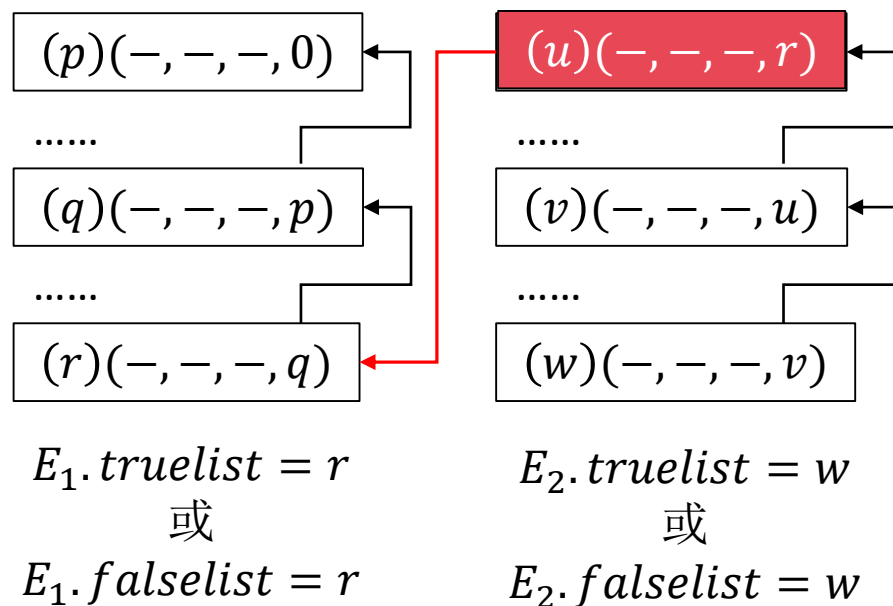
- $(jnz, a, -, p): if\ a\ goto\ p$
- $(j\theta, x, y, p): if\ x\ \theta\ y\ goto\ p$
- $(j, -, -, p): goto\ p$

7.3.2 作为条件控制的布尔式翻译

□ 需要用到的变量或函数:

- nxq : 即Next Quadruplet, 指向下一条将要产生, 但尚未产生的四元式地址 (标号), 执行一次gen增1。
- $mklist(i)$: 创建一个链表, 这个链表仅含标号为 i 的四元式, 并返回链表指针。
- $merge(p1, p2)$: 把以 $p1$ 和 $p2$ 为链首的两条链合并, 返回新的链首。当遇到 $E_1 \wedge E_2$ 时, 它们的 $falselist$ 链表需要合并; 当遇到 $E_1 \vee E_2$ 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 $E_1 \wedge E_2$ 时, E_2 确定了 E_1 的 $truelist$ 链表需要回填的地址; 当遇到 $E_1 \vee E_2$ 时, E_2 确定了 E_1 的 $falselist$ 链表需要回填的地址。

Merge操作

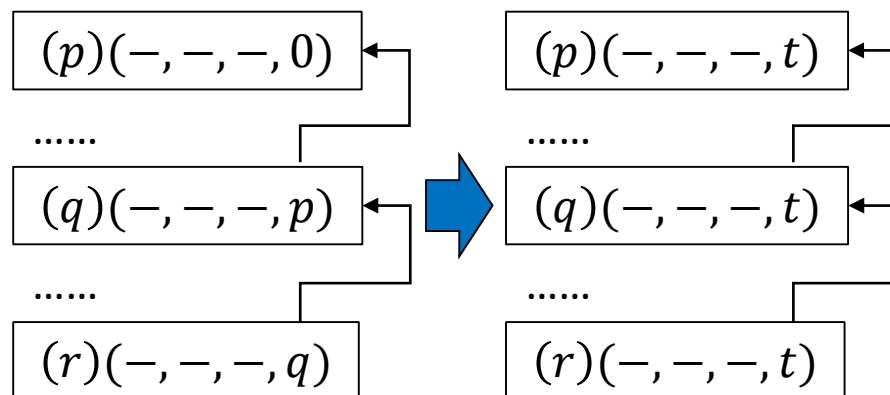


$E.truelist = w$
 或
 $E.falselist = w$

```

Quad * Merge(void * p1, void * 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;
}
  
```

backpatch操作



$E_1.truelist = r$
 或
 $E_1.falselist = r$

```

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) $E \rightarrow E_1 \vee ME_2$ $\{ \text{backpatch}(E_1.\text{falselist}, M.\text{quad});$
 $E.\text{truelist} = \text{merge}(E_1.\text{truelist}, E_2.\text{truelist});$
 $E.\text{falselist} = E_2.\text{falselist}; \}$
- (2) $E \rightarrow E_1 \wedge ME_2$ $\{ \text{backpatch}(E_1.\text{truelist}, M.\text{quad});$
 $E.\text{falselist} = \text{merge}(E_1.\text{falselist}, E_2.\text{falselist});$
 $E.\text{truelist} = E_2.\text{truelist}; \}$
- (3) $M \rightarrow \varepsilon$ $\{ M.\text{quad} = \text{nxq}; \}$ // 在 E_2 之前把它记下来, E_2 之后使用。
- (4) $E \rightarrow \neg E_1$ $\{ E.\text{truelist} = E_1.\text{falselist};$
 $E.\text{falselist} = E_1.\text{truelist}; \}$
- (5) $E \rightarrow (E_1)$ $\{ E.\text{truelist} = E_1.\text{truelist};$
 $E.\text{falselist} = E_1.\text{falselist}; \}$

翻译模式

(6) $E \rightarrow id_1 \theta id_2$ $\{E.truelist = mklist(nxq);$
 $E.falselist = mklist(nxq + 1);$
 $gen(j\theta, id_1.place, id_2.place, 0);$
 $gen(j, -, -, 0);\}$

(7) $E \rightarrow id$ $\{E.truelist = mklist(nxq);$
 $E.falselist = mklist(nxq + 1);$
 $gen(jnz, id.place, -, 0);$
 $gen(j, -, -, 0);\}$

下面通过构造LR(0)分析表确定文法的归约顺序。

(0) $E' \rightarrow E$

(1) $E \rightarrow E \vee ME$

(2) $E \rightarrow E \wedge ME$

(3) $M \rightarrow \varepsilon$

(4) $E \rightarrow \neg E$

(5) $E \rightarrow (E)$

(6) $E \rightarrow id \theta id$

(7) $E \rightarrow id$

	Action								Goto	
状态	\neg	\wedge	\vee	θ	id	()	#	E	M
0					S_4				1	
1								<i>acc</i>		
2	S_2				S_4	S_3			7	
3										
4	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7		
5										
6										11
7	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4		
8		S_6	S_5				S_{12}			
9					S_{13}					
10										
11										
12	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5		
13	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6		
14										
15										

$I_0: E' \rightarrow \cdot E$

$E \rightarrow \cdot E \vee ME$

$E \rightarrow \cdot E \wedge ME$

$E \rightarrow \cdot \neg E$

$E \rightarrow \cdot (E)$

$E \rightarrow \cdot id \theta id$

$E \rightarrow \cdot id$

$I_1 = Go(I_0, E):$

$E' \rightarrow E \cdot$

$E \rightarrow E \cdot \vee ME$

$E \rightarrow E \cdot \wedge ME$

$I_4 = Go(I_0, id):$

$E \rightarrow id \cdot \theta id$

$E \rightarrow id \cdot$

$I_7 = Go(I_2, E):$

$E \rightarrow \neg E \cdot$

$E \rightarrow E \cdot \vee ME$

$E \rightarrow E \cdot \wedge ME$

$I_2 = Go(I_2, \neg):$

$I_3 = Go(I_2, ():$

$I_4 = Go(I_2, id):$

$I_5 = Go(I_7, \vee):$

$I_{11} = Go(I_6, M):$

$E \rightarrow E \wedge M \cdot E$

$E \rightarrow \cdot E \vee ME$

$E \rightarrow \cdot E \wedge ME$

$E \rightarrow \cdot \neg E$

$E \rightarrow \cdot (E)$

$E \rightarrow \cdot id \theta id$

$E \rightarrow \cdot id$

$I_6 = Go(I_7, \wedge):$

$I_{12} = Go(I_8,)):$

$E \rightarrow (E) \cdot$

$I_5 = Go(I_8, \vee):$

$I_6 = Go(I_8, \wedge):$

$I_{13} = Go(I_9, id):$

$E \rightarrow id \theta id \cdot$

(0) $E' \rightarrow E$

(1) $E \rightarrow E \vee ME$

(2) $E \rightarrow E \wedge ME$

(3) $M \rightarrow \varepsilon$

(4) $E \rightarrow \neg E$

(5) $E \rightarrow (E)$

(6) $E \rightarrow id \theta id$

(7) $E \rightarrow id$

	Action								Goto	
状态	\neg	\wedge	\vee	θ	id	()	#	E	M
0	S_2				S_4	S_3			1	
1								acc	7	
2	S_2				S_4	S_3				
3	S_2				S_4	S_3			8	
4	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7		
5										
6										11
7	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4		
8		S_6	S_5				S_{12}			
9					S_{13}					
10	S_2				S_4	S_3			14	
11	S_2				S_4	S_3			15	
12	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5		
13	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6		
14	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1		
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		

$I_2 = Go(I_0, \neg):$

$E \rightarrow \neg \cdot E$

$E \rightarrow \cdot E \vee ME$

$E \rightarrow \cdot E \wedge ME$

$E \rightarrow \cdot \neg E$

$E \rightarrow \cdot (E)$

$E \rightarrow \cdot id \theta id$

$E \rightarrow \cdot id$

$I_3 = Go(I_0, \cdot):$

$E \rightarrow (\cdot E)$

$E \rightarrow \cdot E \vee ME$

$E \rightarrow \cdot E \wedge ME$

$E \rightarrow \cdot \neg E$

$E \rightarrow \cdot (E)$

$E \rightarrow \cdot id \theta id$

$E \rightarrow \cdot id$

$I_8 = Go(I_3, E):$

$E \rightarrow (E \cdot)$

$E \rightarrow E \cdot \vee ME$

$E \rightarrow E \cdot \wedge ME$

$I_2 = Go(I_3, \neg):$

$I_3 = Go(I_3, \cdot):$

$I_4 = Go(I_3, id):$

$I_{14} = Go(I_{10}, E):$

$E \rightarrow E \vee ME \cdot$

$E \rightarrow E \cdot \vee ME$

$E \rightarrow E \cdot \wedge ME$

$I_{15} = Go(I_{11}, E):$

$E \rightarrow E \wedge ME \cdot$

$E \rightarrow E \cdot \vee ME$

$E \rightarrow E \cdot \wedge ME$

$I_2 = Go(I_{10}, \neg):$

$I_2 = Go(I_{11}, \neg):$

$I_3 = Go(I_{10}, \cdot):$

$I_3 = Go(I_{11}, \cdot):$

$I_4 = Go(I_{10}, id):$

$I_4 = Go(I_{11}, id):$

(0) $E' \rightarrow E$

(1) $E \rightarrow E \vee ME$

(2) $E \rightarrow E \wedge ME$

(3) $M \rightarrow \varepsilon$

(4) $E \rightarrow \neg E$

(5) $E \rightarrow (E)$

(6) $E \rightarrow id \theta id$

(7) $E \rightarrow id$

	Action								Goto	
状态	\neg	\wedge	\vee	θ	id	()	#	E	M
0	S_2				S_4	S_3			1	
1		S_6	S_5					<i>acc</i>	7	
2	S_2				S_4	S_3				
3	S_2				S_4	S_3			8	
4	r_7	r_7	r_7	S_9	r_7	r_7	r_7	r_7		
5	r_3	r_3	r_3	r_3	r_3	r_3	r_3	r_3		10
6	r_3	r_3	r_3	r_3	r_3	r_3	r_3	r_3		11
7	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4		
8		S_6	S_5				S_{12}			
9					S_{13}					
10	S_2				S_4	S_3			14	
11	S_2				S_4	S_3			15	
12	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5		
13	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6		
14	r_1	S_6	r_1	r_1	r_1	r_1	r_1	r_1		
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		

$I_5 = Go(I_1, \vee):$

$E \rightarrow E \vee \cdot ME$

$M \rightarrow \cdot$

$I_6 = Go(I_1, \wedge):$

$E \rightarrow E \wedge \cdot ME$

$M \rightarrow \cdot$

$I_9 = Go(I_4, \theta):$

$E \rightarrow id \theta \cdot id$

$I_{10} = Go(I_5, M):$

$E \rightarrow E \vee M \cdot E$

$E \rightarrow \cdot E \vee ME$

$E \rightarrow \cdot E \wedge ME$

$E \rightarrow \cdot \neg E$

$E \rightarrow \cdot (E)$

$E \rightarrow \cdot id \theta id$

$E \rightarrow \cdot id$

$I_5 = Go(I_{14}, \vee):$

$I_6 = Go(I_{14}, \wedge):$

$I_5 = Go(I_{15}, \vee):$

$I_6 = Go(I_{15}, \wedge):$

(0) $E' \rightarrow E$

(1) $E \rightarrow E \vee ME$

(2) $E \rightarrow E \wedge ME$

(3) $M \rightarrow \varepsilon$

(4) $E \rightarrow \neg E$

(5) $E \rightarrow (E)$

(6) $E \rightarrow id \theta id$

(7) $E \rightarrow id$

	Action								Goto	
状态	\neg	\wedge	\vee	θ	id	()	#	E	M
0	S_2				S_4	S_3			1	
1		S_6	S_5					acc	7	
2	S_2				S_4	S_3				
3	S_2				S_4	S_3			8	
4	r_7	r_7	r_7	S_9	r_7	r_7	r_7	r_7		
5	r_3	r_3	r_3	r_3	r_3	r_3	r_3	r_3		10
6	r_3	r_3	r_3	r_3	r_3	r_3	r_3	r_3		11
7	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4		
8		S_6	S_5				S_{12}			
9					S_{13}					
10	S_2				S_4	S_3			14	
11	S_2				S_4	S_3			15	
12	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5		
13	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6		
14	r_1	S_6	r_1	r_1	r_1	r_1	r_1	r_1		
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		

步骤	状态/符号栈	输入串
1	0 #	$a < b \vee c \leq d \wedge e\#$
2	04 #a	$< b \vee c \leq d \wedge e\#$
3	049 #a <	$b \vee c \leq d \wedge e\#$
4	049 <u>13</u> #a < b	$\vee c \leq d \wedge e\#$
5	01 #E	$\vee c \leq d \wedge e\#$
6	015 #E \vee	$c \leq d \wedge e\#$
7	015 <u>10</u> #E \vee M	$c \leq d \wedge e\#$
8	015 <u>10</u> 4 #E \vee M c	$\leq d \wedge e\#$
9	015 <u>10</u> 49 #E \vee M c \leq	$d \wedge e\#$
10	015 <u>10</u> 49 <u>13</u> #E \vee M c \leq d	$\wedge e\#$
11	015 <u>10</u> <u>14</u> #E \vee M E	$\wedge e\#$

(0) $E' \rightarrow E$	(1) $E \rightarrow E \vee ME$	(2) $E \rightarrow E \wedge ME$	(3) $M \rightarrow \varepsilon$
(4) $E \rightarrow \neg E$	(5) $E \rightarrow (E)$	(6) $E \rightarrow id \theta id$	(7) $E \rightarrow id$

	Action								Goto	
状态	\neg	\wedge	\vee	θ	id	()	#	E	M
0	S_2				S_4	S_3			1	
1		S_6	S_5					acc	7	
2	S_2				S_4	S_3				
3	S_2				S_4	S_3			8	
4	r_7	r_7	r_7	S_9	r_7	r_7	r_7	r_7		
5	r_3	r_3	r_3	r_3	r_3	r_3	r_3	r_3		10
6	r_3	r_3	r_3	r_3	r_3	r_3	r_3	r_3		11
7	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4		
8		S_6	S_5				S_{12}			
9					S_{13}					
10	S_2				S_4	S_3			14	
11	S_2				S_4	S_3			15	
12	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5		
13	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6		
14	r_1	S_6	r_1	r_1	r_1	r_1	r_1	r_1		
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		

步骤	状态/符号栈	输入串
11	015 <u>10</u> <u>14</u> # $E \vee ME$	$\wedge e\#$
12	015 <u>10</u> <u>14</u> 6 # $E \vee ME \wedge$	$e\#$
13	015 <u>10</u> <u>14</u> 6 <u>11</u> # $E \vee ME \wedge M$	$e\#$
14	015 <u>10</u> <u>14</u> 6 <u>11</u> 4 # $E \vee ME \wedge Me$	$\#$
15	015 <u>10</u> <u>14</u> 6 <u>11</u> <u>15</u> # $E \vee ME \wedge ME$	$\#$
16	015 <u>10</u> <u>14</u> # $E \vee ME$	$\#$
17	01 # E	$\#$
18	01成功 # E	$\#$

【例7.15】布尔表达式: $a < b \vee c \leq d \wedge e$, 假设 $nxq = 100$

$E \rightarrow id \quad \{E.truelist = mklist(nxq);$

$E \rightarrow E_1 \vee ME_2 \quad \{backpatch(E_1.falselist, M.quad);$
 $E.truelist = merge(E_1.truelist, E_2.truelist);$
 $E.falselist = E_2.falselist; \}$

三地址码

100:	$(j <, a, b, 0)$
101:	$(j, -, -, 102)$
102:	$(j \leq, c, d, 104)$
103:	$(j, -, -, 0)$
104:	$(jnz, e, -, 100)$
105:	$(j, -, -, 103)$

步骤	文法符号栈	输入串	动作
1	#	$a < b \vee c \leq d \wedge e \#$	初始
2	$\#a < b$	$\vee c \leq d \wedge e \#$	移进
3	$\#E$	$\vee c \leq d \wedge e \#$	归约
4	$\#E \vee$	$c \leq d \wedge e \#$	移进
5	$\#E \vee M$	$c \leq d \wedge e \#$	归约
6	$\#E \vee Mc \leq d$	$\wedge e \#$	移进
7	$\#E \vee ME$	$\wedge e \#$	归约
8	$\#E \vee ME \wedge$	$e \#$	移进
9	$\#E \vee ME \wedge M$	$e \#$	归约
10	$\#E \vee ME \wedge Me$	#	移进
11	$\#E \vee ME \wedge ME$	#	归约
12	$\#E \vee ME$	#	归约
13	$\#E$	#	归约
14	$\#E$	#	成功

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。

$E.truelist = 104, E.falselist = 105$
$M.quad = 104$
$E.truelist = 104, E.falselist = 105$
$M.quad = 102$
$E.truelist = 104, E.falselist = 105$

第七章 语义分析和中间代码生成

□ 7.1 说明语句

- 7.1.1 过程中的说明语句
- 7.1.2 保留作用域的信息

□ 7.2 赋值语句的翻译

- 7.2.1 简单算术表达式及赋值语句
- 7.2.2 数组元素的引用
- 7.2.3 类型转换

□ 7.3 布尔表达式的翻译

- 7.3.1 数值表示法
- 7.3.2 作为条件控制的布尔式翻译

□ 7.4 控制语句的翻译

➤ 7.4.1 控制流语句

- 7.4.2 标号与goto语句
- 7.4.3 Case语句的翻译

□ 7.5 过程调用的处理

□ 7.6 类型检查

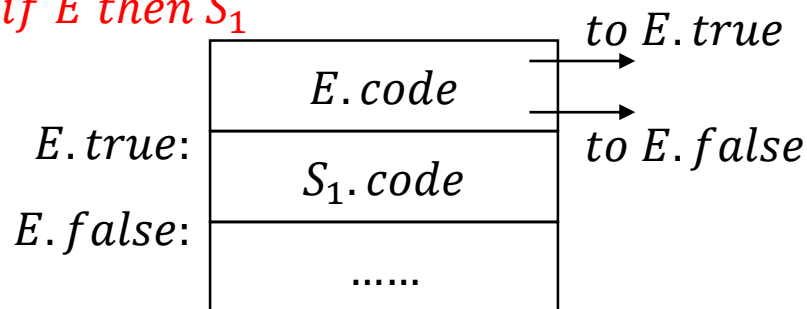
- 7.6.1 类型系统
- 7.6.2 类型检查器的规格说明
- 7.6.3 函数和运算符的重载
- 7.6.4 多态函数

7.4.1 控制流语句

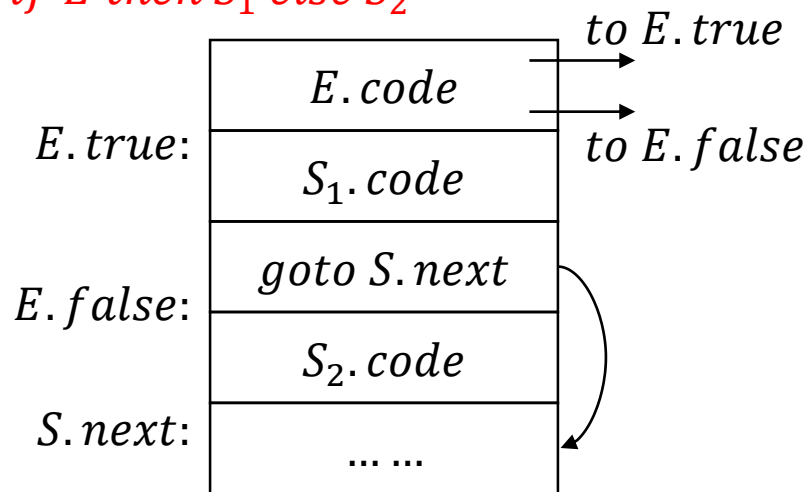
□ 控制流语句:

➤ $S \rightarrow \text{if } E \text{ then } S_1 \mid \text{if } E \text{ then } S_1 \text{ else } S_2 \mid \text{while } E \text{ do } S_1$

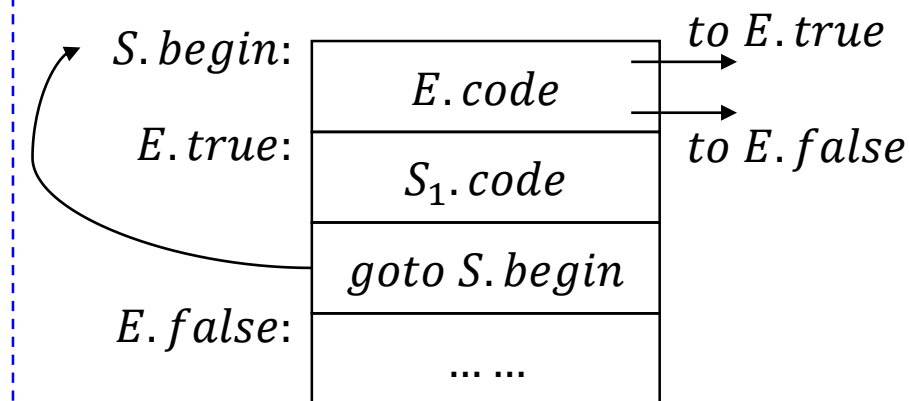
if E then S₁



if E then S₁ else S₂



while E do S₁



7.4.1 控制流语句

□ 控制流语句的完整文法:

- ① $S \rightarrow \text{if } E \text{ then } S$
- ② | $\text{if } E \text{ then } S \text{ else } S$
- ③ | $\text{while } E \text{ do } S$
- ④ | $\text{begin } L \text{ end}$
- ⑤ | A // 赋值语句, 对应7.3节的S
- ⑥ $L \rightarrow L; S$ // 语句表
- ⑦ | S // 语句

□ 新的属性 $S.nextlist$ 和 $L.nextlist$:

- 表示紧接语句 $S(L)$ 之后要执行的语句。

翻译模式

(1) $S \rightarrow \text{if } E \text{ then } M_1 S_1 N \text{ else } M_2 S_2$

$\{ \text{backpatch}(E.\text{truelist}, M_1.\text{quad});$

$\text{backpatch}(E.\text{falselist}, M_2.\text{quad});$

$S.\text{nextlist} = \text{merge}(S_1.\text{nextlist}, N.\text{nextlist}, S_2.\text{nextlist}); \}$

(2) $M \rightarrow \varepsilon$

$\{ M.\text{quad} = \text{nxq}; \}$ // 在 S 之前把它记下来, S 之后使用。

(3) $N \rightarrow \varepsilon$

$\{ N.\text{nextlist} = \text{mklist}(\text{nxq});$

$\text{gen}(j, -, -, 0); \}$ // 跳到 S_2 之后, 也就是整个语句之后

(4) $S \rightarrow \text{if } E \text{ then } M S_1$

$\{ \text{backpatch}(E.\text{truelist}, M.\text{quad});$

$S.\text{nextlist} = \text{merge}(E.\text{falselist}, S_1.\text{nextlist}); \}$

翻译模式

(5) $S \rightarrow \text{while } M_1 E \text{ do } M_2 S_1$

$\{ \text{backpatch}(S_1.\text{nextlist}, M_1.\text{quad});$

$\text{backpatch}(E.\text{truelist}, M_2.\text{quad});$

$S.\text{nextlist} = E.\text{falselist};$

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

(6) $S \rightarrow \text{begin } L \text{ end}$ $\{ S.\text{nextlist} = L.\text{nextlist}; \}$

(7) $S \rightarrow A$ $\{ S.\text{nextlist} = \text{mklist}(); \}$ // 初始化为空表

(8) $L \rightarrow L_1; MS$ $\{ \text{backpatch}(L_1.\text{nextlist}, M.\text{quad}); \}$ // L_1 的结束是S的开始

$L.\text{nextlist} = S.\text{nextlist}; \}$

(9) $L \rightarrow S$ $\{ L.\text{nextlist} = S.\text{nextlist}; \}$

(0) $S' \rightarrow S$ (1) $S \rightarrow iEtMSNeMS$ (2) $M \rightarrow \varepsilon$ (3) $N \rightarrow \varepsilon$ (4) $S \rightarrow iEtMS$ (5) $S \rightarrow wMEdMS$
(6) $S \rightarrow \{L\}$ (7) $S \rightarrow A$ (8) $L \rightarrow L; MS$ (9) $L \rightarrow S$

$I_0: S' \rightarrow \cdot S$ $S \rightarrow \cdot iEtMSNeMS$ $S \rightarrow \cdot iEtMS$ $S \rightarrow \cdot wMEdMS$ $S \rightarrow \cdot \{L\}$ $S \rightarrow \cdot A$	$I_5 = Go(I_0, A)$ $S \rightarrow A \cdot$	$I_{11} = Go(I_7, E)$ $S \rightarrow wME \cdot dMS$	$I_{16} = Go(I_{13}, M)$ $L \rightarrow L; M \cdot S$ $S \rightarrow \cdot iEtMSNeMS$ $S \rightarrow \cdot iEtMS$ $S \rightarrow \cdot wMEdMS$ $S \rightarrow \cdot \{L\}$ $S \rightarrow \cdot A$	$I_{20} = Go(I_{17}, N)$ $S \rightarrow iEtMSN \cdot eMS$
	$I_6 = Go(I_2, E)$ $S \rightarrow iE \cdot tMSNeMS$ $S \rightarrow iE \cdot tMS$	$I_{12} = Go(I_8, \})$ $S \rightarrow \{L\} \cdot$	$I_2 = Go(I_{14}, i)$ $I_3 = Go(I_{14}, w)$ $I_4 = Go(I_{14}, \{)$ $I_5 = Go(I_{14}, A)$ $I_{18} = Go(I_{15}, S)$ $S \rightarrow wMEdMS \cdot$	$I_{21} = Go(I_{20}, e)$ $S \rightarrow iEtMSNe \cdot MS$ $M \rightarrow \cdot$
	$I_7 = Go(I_3, M)$ $S \rightarrow wM \cdot EdMS$	$I_{13} = Go(I_8, ;)$ $L \rightarrow L; \cdot MS$ $M \rightarrow \cdot$		$I_{22} = Go(I_{21}, M)$ $S \rightarrow iEtMSNeM \cdot S$ $S \rightarrow \cdot iEtMSNeMS$ $S \rightarrow \cdot iEtMS$ $S \rightarrow \cdot wMEdMS$ $S \rightarrow \cdot \{L\}$ $S \rightarrow \cdot A$
	$I_1 = Go(I_0, S)$ $S' \rightarrow S \cdot$	$I_{14} = Go(I_{10}, M)$ $S \rightarrow iEtM \cdot SNeMS$ $S \rightarrow iEtM \cdot S$ $S \rightarrow \cdot iEtMSNeMS$ $S \rightarrow \cdot iEtMS$ $S \rightarrow \cdot wMEdMS$ $S \rightarrow \cdot \{L\}$ $S \rightarrow \cdot A$		$I_{23} = Go(I_{22}, S)$ $S \rightarrow iEtMSNeMS \cdot$
$I_2 = Go(I_0, i)$ $S \rightarrow i \cdot EtMSNeMS$ $S \rightarrow i \cdot EtMS$	$I_8 = Go(I_4, L)$ $S \rightarrow \{L \cdot \}$ $L \rightarrow L \cdot ; MS$	$I_{15} = Go(I_{11}, d)$ $S \rightarrow wMEdM \cdot S$ $M \rightarrow \cdot$	$I_{19} = Go(I_{16}, S)$ $L \rightarrow L; MS \cdot$ $I_2 = Go(I_{16}, i)$ $I_3 = Go(I_{16}, w)$ $I_4 = Go(I_{16}, \{)$ $I_5 = Go(I_{16}, A)$	$I_2 = Go(I_{22}, i)$ $I_3 = Go(I_{22}, w)$ $I_4 = Go(I_{22}, \{)$ $I_5 = Go(I_{22}, A)$
$I_3 = Go(I_0, w)$ $S \rightarrow w \cdot ME dMS$ $M \rightarrow \cdot$	$I_9 = Go(I_4, S)$ $L \rightarrow S \cdot$			
$I_4 = Go(I_0, \{)$ $S \rightarrow \{ \cdot L \}$ $L \rightarrow \cdot L; MS$ $L \rightarrow \cdot S$ $S \rightarrow \cdot iEtMSNeMS$ $S \rightarrow \cdot iEtMS$ $S \rightarrow \cdot wMEdMS$ $S \rightarrow \cdot \{L\}$ $S \rightarrow \cdot A$	$I_2 = Go(I_4, i)$			
	$I_3 = Go(I_4, w)$			
	$I_4 = Go(I_4, \{)$			
	$I_5 = Go(I_4, A)$			
	$I_{10} = Go(I_6, t)$ $S \rightarrow iEt \cdot MSNeMS$ $S \rightarrow iEt \cdot MS$ $M \rightarrow \cdot$	$I_{17} = Go(I_{14}, S)$ $S \rightarrow iEtMS \cdot NeMS$ $S \rightarrow iEtMS \cdot$ $N \rightarrow \cdot$		

(0) $S' \rightarrow S$ (1) $S \rightarrow iEtMSNeMS$ (2) $M \rightarrow \varepsilon$ (3) $N \rightarrow \varepsilon$ (4) $S \rightarrow iEtMS$ (5) $S \rightarrow wMEdMS$
 (6) $S \rightarrow \{L\}$ (7) $S \rightarrow A$ (8) $L \rightarrow L;MS$ (9) $L \rightarrow S$

状态	i	t	e	w	d	{	}	;	E	A	#	S	M	N	L
0	S_2			S_3		S_4						1			
1											acc				
2															
3	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2				
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															
21															
22															
23															

$I_0: S' \rightarrow \cdot S$
 $S \rightarrow \cdot iEtMSNeMS$
 $S \rightarrow \cdot iEtMS$
 $S \rightarrow \cdot wMEdMS$
 $S \rightarrow \cdot \{L\}$
 $S \rightarrow \cdot A$

$I_1 = Go(I_0, S)$
 $S' \rightarrow S \cdot$

$I_2 = Go(I_0, i)$
 $S \rightarrow i \cdot EtMSNeMS$
 $S \rightarrow i \cdot EtMS$

$I_3 = Go(I_0, w)$
 $S \rightarrow w \cdot ME dMS$
 $M \rightarrow \cdot$

$I_4 = Go(I_0, \{$
 $S \rightarrow \{ \cdot L \}$
 $L \rightarrow \cdot L;MS$
 $L \rightarrow \cdot S$
 $S \rightarrow \cdot iEtMSNeMS$
 $S \rightarrow \cdot iEtMS$
 $S \rightarrow \cdot wMEdMS$
 $S \rightarrow \cdot \{L\}$
 $S \rightarrow \cdot A$

(0) $S' \rightarrow S$ (1) $S \rightarrow iEtMSNeMS$ (2) $M \rightarrow \varepsilon$ (3) $N \rightarrow \varepsilon$ (4) $S \rightarrow iEtMS$ (5) $S \rightarrow wMEdMS$
(6) $S \rightarrow \{L\}$ (7) $S \rightarrow A$ (8) $L \rightarrow L;MS$ (9) $L \rightarrow S$

状态	i	t	e	w	d	{	}	;	E	A	#	S	M	N	L
0	S_2			S_3		S_4				S_5		1			
1											acc				
2									S_6						
3	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		7		
4	S_2			S_3		S_4				S_5		9			8
5	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7				
6		S_{10}													
7															
8															
9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9				
10	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2				
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															
21															
22															
23															

$I_5 = Go(I_0, A)$
 $S \rightarrow A \cdot$

$I_6 = Go(I_2, E)$
 $S \rightarrow iE \cdot tMSNeMS$
 $S \rightarrow iE \cdot tMS$

$I_7 = Go(I_3, M)$
 $S \rightarrow wM \cdot EdMS$

$I_8 = Go(I_4, L)$
 $S \rightarrow \{L \cdot\}$
 $L \rightarrow L \cdot ; MS$

$I_9 = Go(I_4, S)$
 $L \rightarrow S \cdot$

$I_2 = Go(I_4, i)$

$I_3 = Go(I_4, w)$

$I_4 = Go(I_4, \{)$

$I_5 = Go(I_4, A)$

$I_{10} = Go(I_6, t)$
 $S \rightarrow iEt \cdot MSNeMS$
 $S \rightarrow iEt \cdot MS$
 $M \rightarrow \cdot$

(0) $S' \rightarrow S$ (1) $S \rightarrow iEtMSNeMS$ (2) $M \rightarrow \varepsilon$ (3) $N \rightarrow \varepsilon$ (4) $S \rightarrow iEtMS$ (5) $S \rightarrow wMEdMS$
(6) $S \rightarrow \{L\}$ (7) $S \rightarrow A$ (8) $L \rightarrow L; MS$ (9) $L \rightarrow S$

状态	i	t	e	w	d	{	}	;	E	A	#	S	M	N	L
0	S_2			S_3		S_4				S_5		1			
1											acc				
2									S_6						
3	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		7		
4	S_2			S_3		S_4				S_5		9			8
5	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7				
6		S_{10}													
7									S_{11}						
8							S_{12}	S_{13}							
9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9				
10	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		14		
11					S_{15}										
12	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6				
13	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2				
14												17			
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2				
16															
17	r_4	r_4	r_3	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4				
18															
19															
20															
21															
22															
23															

$I_{11} = Go(I_7, E)$
 $S \rightarrow wME \cdot dMS$

$I_{12} = Go(I_8, \})$
 $S \rightarrow \{L\} \cdot$

$I_{13} = Go(I_8, ;)$
 $L \rightarrow L; \cdot MS$
 $M \rightarrow \cdot$

$I_{14} = Go(I_{10}, M)$
 $S \rightarrow iEtM \cdot SNeMS$
 $S \rightarrow iEtM \cdot S$
 $S \rightarrow \cdot iEtMSNeMS$
 $S \rightarrow \cdot iEtMS$
 $S \rightarrow \cdot wMEdMS$
 $S \rightarrow \cdot \{L\}$
 $S \rightarrow \cdot A$

$I_{15} = Go(I_{11}, d)$
 $S \rightarrow wMEdM \cdot S$
 $M \rightarrow \cdot$

$I_{17} = Go(I_{14}, S)$
 $S \rightarrow iEtMS \cdot NeMS$
 $S \rightarrow iEtMS \cdot$
 $N \rightarrow \cdot$

(0) $S' \rightarrow S$ (1) $S \rightarrow iEtMSNeMS$ (2) $M \rightarrow \varepsilon$ (3) $N \rightarrow \varepsilon$ (4) $S \rightarrow iEtMS$ (5) $S \rightarrow wMEdMS$
(6) $S \rightarrow \{L\}$ (7) $S \rightarrow A$ (8) $L \rightarrow L; MS$ (9) $L \rightarrow S$

状态	i	t	e	w	d	{	}	;	E	A	#	S	M	N	L
0	S_2			S_3		S_4				S_5		1			
1											acc				
2									S_6						
3	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		7		
4	S_2			S_3		S_4				S_5		9			8
5	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7				
6		S_{10}													
7									S_{11}						
8							S_{12}	S_{13}							
9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9				
10	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		14		
11					S_{15}										
12	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6				
13	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		16		
14	S_2			S_3		S_4				S_5		17			
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	18			
16	S_2			S_3		S_4				S_5		19			
17	r_4	r_4	r_3	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4				
18	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5				
19	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8				
20															
21															
22															
23															

$I_{16} = Go(I_{13}, M)$
 $L \rightarrow L; M \cdot S$
 $S \rightarrow \cdot iEtMSNeMS$
 $S \rightarrow \cdot iEtMS$
 $S \rightarrow \cdot wMEdMS$
 $S \rightarrow \cdot \{L\}$
 $S \rightarrow \cdot A$

$I_2 = Go(I_{14}, i)$

$I_3 = Go(I_{14}, w)$

$I_4 = Go(I_{14}, \{ \})$

$I_5 = Go(I_{14}, A)$

$I_{18} = Go(I_{15}, S)$
 $S \rightarrow wMEdMS \cdot$

$I_{19} = Go(I_{16}, S)$
 $L \rightarrow L; MS \cdot$

$I_2 = Go(I_{16}, i)$

$I_3 = Go(I_{16}, w)$

$I_4 = Go(I_{16}, \{ \})$

$I_5 = Go(I_{16}, A)$

(0) $S' \rightarrow S$ (1) $S \rightarrow iEtMSNeMS$ (2) $M \rightarrow \varepsilon$ (3) $N \rightarrow \varepsilon$ (4) $S \rightarrow iEtMS$ (5) $S \rightarrow wMEdMS$
(6) $S \rightarrow \{L\}$ (7) $S \rightarrow A$ (8) $L \rightarrow L; MS$ (9) $L \rightarrow S$

状态	i	t	e	w	d	{	}	;	E	A	#	S	M	N	L
0	S_2			S_3		S_4				S_5		1			
1											acc				
2									S_6						
3	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		7		
4	S_2			S_3		S_4				S_5		9			8
5	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7				
6		S_{10}													
7									S_{11}						
8							S_{12}	S_{13}							
9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9				
10	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		14		
11					S_{15}										
12	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6				
13	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		16		
14	S_2			S_3		S_4				S_5		17			
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	18			
16	S_2			S_3		S_4				S_5		19			
17	r_4	r_4	r_3	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4			20	
18	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5				
19	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8				
20			S_{21}												
21	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		22		
22	S_2			S_3		S_4				S_5		23			
23	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1				

$I_{20} = Go(I_{17}, N)$
 $S \rightarrow iEtMSN \cdot eMS$

$I_{21} = Go(I_{20}, e)$
 $S \rightarrow iEtMSNe \cdot MS$
 $M \rightarrow \cdot$

$I_{22} = Go(I_{21}, M)$
 $S \rightarrow iEtMSNeM \cdot S$
 $S \rightarrow \cdot iEtMSNeMS$
 $S \rightarrow \cdot iEtMS$
 $S \rightarrow \cdot wMEdMS$
 $S \rightarrow \cdot \{L\}$
 $S \rightarrow \cdot A$

$I_{23} = Go(I_{22}, S)$
 $S \rightarrow iEtMSNeMS \cdot$

$I_2 = Go(I_{22}, i)$

$I_3 = Go(I_{22}, w)$

$I_4 = Go(I_{22}, \{ \})$

$I_5 = Go(I_{22}, A)$

(0) $S' \rightarrow S$ (1) $S \rightarrow iEtMSNeMS$ (2) $M \rightarrow \varepsilon$ (3) $N \rightarrow \varepsilon$ (4) $S \rightarrow iEtMS$ (5) $S \rightarrow wMEdMS$
 (6) $S \rightarrow \{L\}$ (7) $S \rightarrow A$ (8) $L \rightarrow L; MS$ (9) $L \rightarrow S$

状态	i	t	e	w	d	{	}	;	E	A	#	S	M	N	L	状态/符号栈	输入串
0	S_2			S_3		S_4				S_5		1				0	$iEt\{A; A\}eA\#$
1											acc					#	
2									S_6							02	
3	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		7			#i	$Et\{A; A\}eA\#$
4	S_2			S_3		S_4				S_5		9			8	026	$t\{A; A\}eA\#$
5	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7					#iE	
6		S_{10}														02610	
7									S_{11}							#iEt	$\{A; A\}eA\#$
8							S_{12}	S_{13}								02610 14	$\{A; A\}eA\#$
9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9					#iEtM	
10	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		14			02610 144	$A; A\}eA\#$
11					S_{15}											#iEtM{	
12	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6					02610 1445	$; A\}eA\#$
13	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		16			#iEtM{A	
14	S_2			S_3		S_4				S_5		17				02610 1449	$; A\}eA\#$
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	18				#iEtM{S	
16	S_2			S_3		S_4				S_5		19				02610 1448	$; A\}eA\#$
17	r_4	r_4	r_3	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4			20		02610 144813	
18	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5					#iEtM{L;	$A\}eA\#$
19	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8					02610 144813 16	
20			S_{21}													#iEtM{L; M	$A\}eA\#$
21	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		22			02610 144813 165	$\}eA\#$
22	S_2			S_3		S_4				S_5		23				#iEtM{L; MA	
23	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1						

(0) $S' \rightarrow S$ (1) $S \rightarrow iEtMSNeMS$ (2) $M \rightarrow \varepsilon$ (3) $N \rightarrow \varepsilon$ (4) $S \rightarrow iEtMS$ (5) $S \rightarrow wMEdMS$
 (6) $S \rightarrow \{L\}$ (7) $S \rightarrow A$ (8) $L \rightarrow L;MS$ (9) $L \rightarrow S$

状态	i	t	e	w	d	{	}	;	E	A	#	S	M	N	L	状态/符号栈	输入串	
0	S_2			S_3		S_4				S_5		1				02610 144813 165	}eA#	
1											acc					#iEtM{L;MA		
2									S_6							02610 144813 16 19	}eA#	
3	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		7			#iEtM{L;MS		
4	S_2			S_3		S_4				S_5		9			8	02610 1448	}eA#	
5	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7					#iEtM{L		
6		S_{10}														02610 144812	eA#	
7									S_{11}							#iEtM{L}		
8							S_{12}	S_{13}								02610 14 17	eA#	
9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9					#iEtMS		
10	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		14			02610 14 17 20	eA#	
11					S_{15}											#iEtMSN		
12	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6					02610 14 17 20 21	A#	
13	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		16			#iEtMSNe		
14	S_2			S_3		S_4				S_5		17				02610 14 17 20 21 22	A#	
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	18				#iEtMSNeM		
16	S_2			S_3		S_4				S_5		19				02610 14 17 20 21 225	A#	
17	r_4	r_4	r_3	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4			20		#iEtMSNeMA		
18	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5					02610 14 17 20 21 22 23	#	
19	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8					#iEtMSNeMS		
20			S_{21}													01	#	
21	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		22			#S		
22	S_2			S_3		S_4				S_5		23				01成功	#	
23	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1					#S		

【例7.16】续例【7.15】 $if\ a < b \vee c \leq d \wedge e\ then\ x = y + z\ else\ x = 0$

```
S → if E then M1S1N else M2S2
{backpatch(E.truelist, M1.quad);
  backpatch(E.falselist, M2.quad);
S.nextlist = merge(S1.nextlist, N.nextlist, S2.nextlist); }
```

3	#iEtM	$x = y + z\ e\ x = 0\#$
4	#iEtMx = y	$+z\ e\ x = 0\#$
5	#iEtMx = E	$+z\ e\ x = 0\#$
6	#iEtMx = E + z	$e\ x = 0\#$
7	#iEtMx = E + E	$e\ x = 0\#$
8	#iEtMx = E	$e\ x = 0\#$
9	#iEtMA	$e\ x = 0\#$
10	#iEtMS	$e\ x = 0\#$
11	#iEtMSN	$e\ x = 0\#$
12	#iEtMSNe	$x = 0\#$
13	#iEtMSNeM	$x = 0\#$
14	#iEtMSNeMx = 0	#
15	#iEtMSNeMx = E	#
16	#iEtMSNeMA	#
17	#iEtMSNeMS	#
18	#S	#

三地址码

100:	(j <, a, b, 106)
101:	(j, -, -, 102)
102:	(j ≤, c, d, 104)
103:	(j, -, -, 109)
104:	(jnz, e, -, 106)
105:	(j, -, -, 109)
106:	(+, y, z, T1)
107:	(=, T1, -, x)
108:	(j, -, -, 110)
109:	(=, 0, -, x)

现在剩下最后一个链S.nextlist, 此处我们手工使用nxq填充。

S.nextlist = null
M.quad = 109
N.nextlist = 108
S.nextlist = null
M.quad = 106
S.nextlist = 108

【例7.17】 *while a < b do if c < d then x = y + z, 假设nxq = 100*

S → *while* *M*₁*E* *do* *M*₂*S*₁
{ *backpatch*(*S*₁.*nextlist*, *M*₁.*quad*);
 backpatch(*E*.*truelist*, *M*₂.*quad*);
S.*nextlist* = *E*.*falselist*;
 gen(*j*, −, −, *M*₁.*quad*); }
; }

三地址码

100:	(<i>j</i> <, <i>a</i> , <i>b</i> , 102)
101:	(<i>j</i> , −, −, 107)
102:	(<i>j</i> <, <i>c</i> , <i>d</i> , 104)
103:	(<i>j</i> , −, −, 100)
104:	(+, <i>y</i> , <i>z</i> , <i>T</i> 1)
105:	(=, <i>T</i> 1, −, <i>x</i>)
106:	(<i>j</i> , −, −, 100)

1	#	<i>wa</i> < <i>b</i> <i>dic</i> < <i>d</i> <i>tx</i> = <i>y</i> + <i>z</i> #
2	# <i>w</i>	<i>a</i> < <i>b</i> <i>dic</i> < <i>d</i> <i>tx</i> = <i>y</i> + <i>z</i> #
3	# <i>wM</i>	<i>a</i> < <i>b</i> <i>dic</i> < <i>d</i> <i>tx</i> = <i>y</i> + <i>z</i> #
4	# <i>wMa</i> < <i>b</i>	<i>dic</i> < <i>d</i> <i>tx</i> = <i>y</i> + <i>z</i> #
5	# <i>wME</i>	<i>dic</i> < <i>d</i> <i>tx</i> = <i>y</i> + <i>z</i> #
6	# <i>wMed</i>	<i>ic</i> < <i>d</i> <i>tx</i> = <i>y</i> + <i>z</i> #
7	# <i>wMedM</i>	<i>ic</i> < <i>d</i> <i>tx</i> = <i>y</i> + <i>z</i> #
8	# <i>wMedMic</i> < <i>d</i>	<i>tx</i> = <i>y</i> + <i>z</i> #
9	# <i>wMedMiE</i>	<i>tx</i> = <i>y</i> + <i>z</i> #
10	# <i>wMedMiEt</i>	<i>x</i> = <i>y</i> + <i>z</i> #
11	# <i>wMedMiEtM</i>	<i>x</i> = <i>y</i> + <i>z</i> #
12	# <i>wMedMiEtMx</i> = <i>y</i> + <i>z</i>	#
13	# <i>wMedMiEtMA</i>	#
14	# <i>wMedMiEtMS</i>	#
15	# <i>wMedMS</i>	#
16	# <i>S</i>	#

此处用nxq手工填充*S.nextlist*

<i>S.nextlist</i> = null
<i>M.quad</i> = 104
<i>S.nextlist</i> = 103
<i>M.quad</i> = 102
<i>E.truelist</i> = 100, <i>E.falselist</i> = 101
<i>S.nextlist</i> = 101

第七章 语义分析和中间代码生成

□ 7.1 说明语句

- 7.1.1 过程中的说明语句
- 7.1.2 保留作用域的信息

□ 7.2 赋值语句的翻译

- 7.2.1 简单算术表达式及赋值语句
- 7.2.2 数组元素的引用
- 7.2.3 类型转换

□ 7.3 布尔表达式的翻译

- 7.3.1 数值表示法
- 7.3.2 作为条件控制的布尔式翻译

□ 7.4 控制语句的翻译

- 7.4.1 控制流语句
- 7.4.2 标号与goto语句

- 7.4.3 Case语句的翻译

□ 7.5 过程调用的处理

□ 7.6 类型检查

- 7.6.1 类型系统
- 7.6.2 类型检查器的规格说明
- 7.6.3 函数和运算符的重载
- 7.6.4 多态函数

7.4.2 标号与goto语句

□ 标号语句形式: $L:S;$

① $S \rightarrow label\ S \mid goto\ i$

② $label \rightarrow i:$

□ 若标号已存在且已定义

- 生成四元式: $(j, -, -, p)$

□ 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。

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: $(j, -, -, -);$
- 当遇到定义标号语句时, 回填。

(1) $S \rightarrow label\ S$

(2) $label \rightarrow i$: $\{p = lookup(i.name);$
 $if\ p = null\ then\ addlabel(i.name, label, true, nxq);$
 $else\ if\ p.type \neq label\ ||\ p.isdefined = true\ then\ Error;$
 $else\ \{$
 $modifylabel(p, isdefined = true);$
 $backpatch(p.address); \} \}$

(3) $S \rightarrow goto\ i$ $\{p = lookup(i.name);$
 $if\ p = null\ then\ \{addlabel(i.name, label, false, nxq);$
 $gen(j, -, -, -);\}$
 $else\ if\ p.type \neq label\ then\ Error;$
 $else\ if\ p.isdefined = true\ then\ gen(j, -, -, p.address);$
 $else\ \{gen(j, -, -, p.address);$
 $modifylabel(p, address = nxq - 1);\}$

第七章 语义分析和中间代码生成

□ 7.1 说明语句

- 7.1.1 过程中的说明语句
- 7.1.2 保留作用域的信息

□ 7.2 赋值语句的翻译

- 7.2.1 简单算术表达式及赋值语句
- 7.2.2 数组元素的引用
- 7.2.3 类型转换

□ 7.3 布尔表达式的翻译

- 7.3.1 数值表示法
- 7.3.2 作为条件控制的布尔式翻译

□ 7.4 控制语句的翻译

- 7.4.1 控制流语句
- 7.4.2 标号与goto语句
- 7.4.3 Case语句的翻译

□ 7.5 过程调用的处理

□ 7.6 类型检查

- 7.6.1 类型系统
- 7.6.2 类型检查器的规格说明
- 7.6.3 函数和运算符的重载
- 7.6.4 多态函数

7.4.3 Case语句的翻译

□ Case语句形式:

```
switch E
{
    case  $c_1$ :  $S_1$ ;
    case  $c_2$ :  $S_2$ ;
    .....
    case  $c_{n-1}$ :  $S_{n-1}$ ;
    default:  $S_n$ ;
}
```

□ 文法:

// Case内部不含default部分

$I \rightarrow AS$; // S 是各类语句

$A \rightarrow case\ c: \mid I\ case\ c:$ // c 是常数

// 加上switch和default

$W \rightarrow switch\ E\{ID\}$

$D \rightarrow default: S; \mid \varepsilon$

// 考虑多个语句

$S \rightarrow S; S$

7.4.3 Case语句的翻译

□ Case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值;
- 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。

7.4.3 Case语句的翻译

□ 文法:

// Case内部不含default部分

$I \rightarrow AS;$

$A \rightarrow \text{case } c: | I \text{ case } c:$

// 加上switch和default

$W \rightarrow \text{switch } E\{ID\}$

$D \rightarrow \text{default: } S; | \varepsilon$

// 考虑多个语句

$S \rightarrow S; S$

□ 文法存在问题:

- 考虑到标号名字冲突问题, 可以考虑为每个Case语句创建一个符号表, 这个动作在遇到switch时发生, 至少不能晚于{。
- 遇到}生成test标号里面的语句时, 需要知道临时变量的名字, 如果把switch E提出来, 需要考虑如何传递的问题。

7.4.3 Case语句的翻译

□ 新的文法:

// Case内部不含default部分

$I \rightarrow AS;$

$A \rightarrow \text{case } c: | I \text{ case } c:$

// 加上switch和default

$W \rightarrow T\{ID\}$

$T \rightarrow \text{switch } E$

$D \rightarrow \text{default: } MS; | \varepsilon$

$M \rightarrow \varepsilon$

// 考虑多个语句

$S \rightarrow S; S$

翻译模式

```

T → switch E      { mktable(top(tblptr)); Init(Que); // 创建一张新的符号表和队列
                       t = newtemp; gen(=, E.place, −, t); T.place = t;
                       addlabel(test, case, false, nxq); gen(j, −, −, −);
                       addlabel(next, case, false, null); }
  
```

```

A → case c:        { Que.add(c, nxq); }
  
```

```

A → I case c:       { Que.add(c, nxq); }
  
```

```

D → default: MS; { D.quad = M.quad;
                    p = lookup(next); a = p.address
                    modifylabel(p, address = nxq);
                    gen(j, −, −, a); // 待回填链 }
  
```

```

D → ε              { D.quad = null; }
  
```

```

M → ε              { M.quad = nxq; }
  
```

翻译模式

```
I → AS;          { p = lookup(next); a = p.address;  
                    modifylabel(p, address = nxq); gen(j, −, −, a); // 待回填链 }  
  
W → T{ID}      { p = lookup(test); backpatch(p.address, nxq);  
                    while (! Que.IsEmpty)  
                        { (c, a) = Que.de(); gen(j =, T.place, c, a); }  
                    if (D.quad ≠ null) gen(j, −, −, D.quad);  
                    p = lookup(next); W.nextlist = p.address }
```

【例7.18】 $switch\ x\{case\ 0: x = x + 1; case\ 1: x = y; default: x = 0;\}$, 假设

$I \rightarrow AS;$	$\{ p = lookup(next);$	
$a = p.address;$	$= t;$	
$modifylabel(p, address = nxq);$	$-);$	
$gen(j, -, -, a); \}$		

三地址码

100: $(=, x, -, T1)$
101: $(j, -, -, -)$
102: $(+, x, 1, T2)$
103: $(=, T2, -, x)$
104: $(j, -, -, -)$

步骤	文法符号栈	输入串
1	#	$sx\{c0: x = x + 1; c1: x = y; d: x = 0;\}\#$
2	$\#sx$	$\{c0: x = x + 1; c1: x = y; d: x = 0;\}\#$
3	$\#sE$	$\{c0: x = x + 1; c1: x = y; d: x = 0;\}\#$
4	$\#T$	$\{c0: x = x + 1; c1: x = y; d: x = 0;\}\#$
5	$\#T\{c0:$	$x = x + 1; c1: x = y; d: x = 0;\}\#$
6	$\#T\{A$	$x = x + 1; c1: x = y; d: x = 0;\}\#$
7	$\#T\{Ax = x$	$+1; c1: x = y; d: x = 0;\}\#$
8	$\#T\{Ax = E$	$+1; c1: x = y; d: x = 0;\}\#$
9	$\#T\{Ax = E + 1$	$; c1: x = y; d: x = 0;\}\#$
10	$\#T\{Ax = E + E$	$; c1: x = y; d: x = 0;\}\#$
11	$\#T\{Ax = E$	$; c1: x = y; d: x = 0;\}\#$
12	$\#T\{AS$	$; c1: x = y; d: x = 0;\}\#$
13	$\#T\{AS;$	$c1: x = y; d: x = 0;\}\#$
14	$\#T\{I$	$c1: x = y; d: x = 0;\}\#$
15	$\#T\{Ic1:$	$x = y; d: x = 0;\}\#$

名字	类型	定义	地址
test	case	F	101
next	case	F	104

0
102

$E.place = 1$
$E.place = T2$
$T.place = T1$

【例7.18】 $switch\ x\ \{case\ 0: x = x + 1; case\ 1: x = y; default: x = 0;\}$, 假设

```
W → T{ID}      { p = lookup(test);
backpatch(p.address, nxq);
while (! Que.IsEmpty)
{ (c, a) = Que.de(); gen(j =, T.place, c, a); }
if (D.quad ≠ null) gen(j, -, -, D.quad);
p = lookup(next);
W.nextlist = p.address}
```

三地址码

100: (=, x, -, T1)	109: (j =, T1, 0, 102)
101: (j, -, -, 109)	110: (j =, T1, 1, 105)
102: (+, x, 1, T2)	111: (j, -, -, 107)
103: (=, T2, -, x)	
104: (j, -, -, -)	
105: (=, y, -, x)	
106: (j, -, -, 104)	
107: (=, 0, -, x)	
108: (j, -, -, 106)	

17	#T{Ax = y	; d: x = 0; }#
18	#T{Ax = E	; d: x = 0; }#
19	#T{AS	; d: x = 0; }#
20	#T{AS;	d: x = 0; }#
21	#T{I	d: x = 0; }#
22	#T{Id:	x = 0; }#
23	#T{Id: M	x = 0; }#
24	#T{Id: Mx = 0	; }#
25	#T{Id: Mx = E	; }#
26	#T{Id: MS	; }#
27	#T{Id: MS;	}#
28	#T{ID	}#
29	#T{ID}	#
30	#W	#

E.place = 0
D.quad = 107
W.nextlist = 108

名字	类型	定义	地址
test	case	F	101
next	case	F	108
0	1		
102	105		

第七章 语义分析和中间代码生成

□ 7.1 说明语句

- 7.1.1 过程中的说明语句
- 7.1.2 保留作用域的信息

□ 7.2 赋值语句的翻译

- 7.2.1 简单算术表达式及赋值语句
- 7.2.2 数组元素的引用
- 7.2.3 类型转换

□ 7.3 布尔表达式的翻译

- 7.3.1 数值表示法
- 7.3.2 作为条件控制的布尔式翻译

□ 7.4 控制语句的翻译

- 7.4.1 控制流语句
- 7.4.2 标号与goto语句
- 7.4.3 Case语句的翻译

□ 7.5 过程调用的处理

□ 7.6 类型检查

- 7.6.1 类型系统
- 7.6.2 类型检查器的规格说明
- 7.6.3 函数和运算符的重载
- 7.6.4 多态函数

7.5 过程调用的处理

□ **call** $S(A+B, Z)$:

$[+, A, B, T1]$

$[param, -, -, T1]$

$[param, -, -, Z]$

$[call, -, -, S]$

$S \rightarrow call\ id\ (Elist)$

$\{ \text{foreach } p \text{ in } Que\ gen(param, -, -, p);$

$gen(call, -, -, id.place); \}$

$Elist \rightarrow Elist, E$

$\{ Que.add(E.place); \}$

$Elist \rightarrow E$

$\{ Que.add(E.place); \}$

第七章 语义分析和中间代码生成

□ 7.1 说明语句

- 7.1.1 过程中的说明语句
- 7.1.2 保留作用域的信息

□ 7.2 赋值语句的翻译

- 7.2.1 简单算术表达式及赋值语句
- 7.2.2 数组元素的引用
- 7.2.3 类型转换

□ 7.3 布尔表达式的翻译

- 7.3.1 数值表示法
- 7.3.2 作为条件控制的布尔式翻译

□ 7.4 控制语句的翻译

- 7.4.1 控制流语句
- 7.4.2 标号与goto语句
- 7.4.3 Case语句的翻译

□ 7.5 过程调用的处理

□ 7.6 类型检查

- 7.6.1 类型系统
- 7.6.2 类型检查器的规格说明
- 7.6.3 函数和运算符的重载
- 7.6.4 多态函数

第七章作业

【作业7-1】 7.4.2节中, 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 $\text{if } A < B \vee C < D \text{ then } x = y$

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  
  end
```



山东大学
SHANDONG UNIVERSITY

第七章 语义分析和中间代码生成

The End

谢谢

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