

练习 6.2.1: 假定图 6-26 中的函数 `widen` 可以处理图 6-25a 的层次结构中的所有类型, 翻译下列表达式。假定 `c` 和 `d` 是 `char` 型, `s` 和 `t` 是 `short` 型, `i` 和 `j` 是 `int` 型, `x` 是 `float` 型。

- 1) `j = s + d`
- 2) `x = s + d`
- 3) `x = (s + d) * (t + c)`

```
1)
j = s + d
t1 = (int)s
t2 = (int)d
j = t1 + t2
2)
x = s + d
t1 = (int)s
t2 = (int)d
t3 = t1 + t2
x = (float)t3
4)
t1 = (int)s
t2 = (int)d
t3 = t1 + t2
t4 = (int)t
t5 = (int)c
t6 = t4 + t5
t7 = t3 * t6
X = (float)t7
```

练习 6.2.2: 在图 6-36 的语法制导定义中添加处理下列控制流构造的规则:

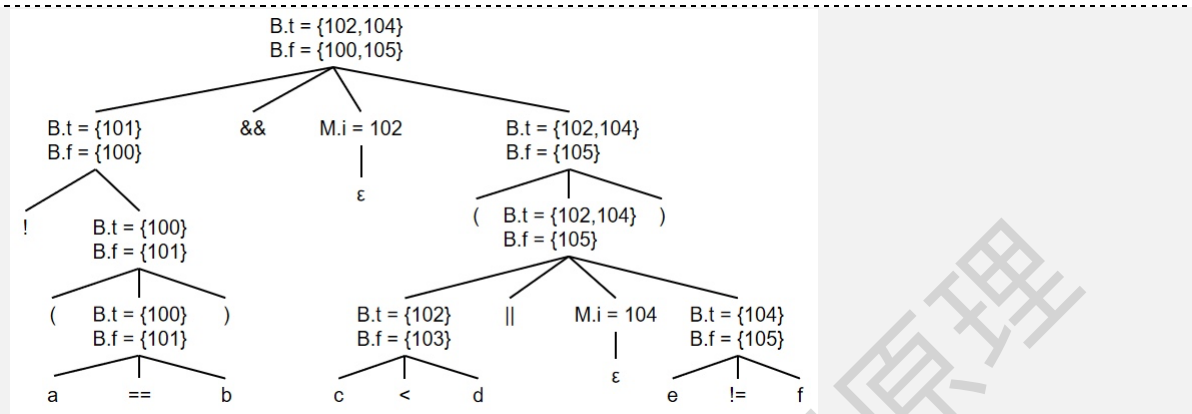
- 1) $S \rightarrow \text{repeat } S1 \text{ until } B$, 当 `B` 为真时结束循环
- 2) $S \rightarrow \text{for } (S1 ; B ; S2) S3$

```
1)
S1.next = newlabel()
B.true = S.next
B.false = newlabel()
S.code = label(B.false) || S1.code || label(S1.next) || B.code
2)
S1.next = newlabel()
B.true = newlabel()
B.false = S.next
S2.next = S1.next
S3.next = newlabel()
S.code = S1.code || label(S1.next) || B.code
|| label(B.true) || S3.code || label(S3.next) || S2.code || gen('goto', S1.next)
```

练习 6.2.3: 使用图 6-43 中的翻译方案翻译下列表达式

$!(a == b) \&\& (c < d \mid \mid e != f)$

给出分析过程给出带 truelist 和 falselist 的注释语法分析树。假设第一条被生成的指令的地址是 100。



分析过程: 对 $a == b$ 按照 $B \rightarrow E_1 \text{ rel } E_2$ 的语义动作规约, 产生指令:

100: if a == b goto _

101: goto _

并且 $M.i = 102$ 记录 next instr 值

用产生式 $B \rightarrow (B_1)$ 进行规约, 拷贝 truelist 和 falselist, 所以有 $B.t = \{100\}$, $B.f = \{101\}$

用产生式 $B \rightarrow !B_1$ 进行规约, 翻转 truelist 和 falselist, 所以有 $B.t = \{101\}$, $B.f = \{100\}$

对于 $c < d$ 按照 $B \rightarrow E_1 \text{ rel } E_2$ 的语义动作规约, 产生指令:

102: if c < d goto _

103: goto _

并且 $M.i = 104$

对于 $e != f$ 按照 $B \rightarrow E_1 \text{ rel } E_2$ 的语义动作规约, 产生指令

104: if e != f goto _

105: goto _

使用 $B \rightarrow B_1 \mid \mid MB_2$ 规约, 调用 $\text{backpatch}(B_1.f, M.i)$, 其中 $B_1.f = \{103\}$, $M.i = 104$, 将 104 填写到 103 指令, 得到:

100: if a == b goto _

101: goto _

102: if c < d goto _

103: goto 104

104: if e != f goto _

105: goto _

接着执行 $B.t = \text{merge}(B_1.t, B_2.t)$, 所以 $B.t = \{102, 104\}$; $B.f = B_2.f$, 所以 $B.f = \{105\}$ 。

用产生式 $B \rightarrow (B_1)$ 进行规约, 拷贝 truelist 和 falselist, 所以有 $B.t = \{102, 104\}$, $B.f = \{105\}$

使用 $B \rightarrow B_1 \&\& MB_2$ 规约, 调用 $\text{backpatch}(B_1.t, M.i)$, 其中 $B_1.t = \{101\}$, $M.i = 102$ 将 102 填写到 101 指令, 得到:

100: if a == b goto _

101: goto 102

102: if c < d goto _

103: goto 104

```
104: if e != f goto _
```

```
105: goto _
```

接着执行 $B.t = B_2.t$ ，所以 $B.t = \{102, 104\}$ ，

接着执行 $B.f = \text{merge}(B_1.f, B_2.f)$ ，所以 $B.f = \{100, 105\}$ 最终生成的指令如下：

```
100: if a == b goto _
```

```
101: goto 102
```

```
102: if c < d goto _
```

```
103: goto 104
```

```
104: if e != f goto _
```

```
105: goto _
```

整个表达式为真当且仅当控制流到达 102, 104 的 goto。

整个表达式为假当且仅当控制流到达 100, 105 的 goto。

在后续的编译过程中，已知分别应该做什么的时候，这些指令的目标将会被填写完整。