四、中间代码生成
(12. 控制流语句翻译中的地址回填技术)

魏恒峰

hfwei@nju.edu.cn

2024 年 05 月 17 日

使用标签标记跳转目标

```
1    if (true || false) {
2      a = b;
3    }
```

```
1   goto L2
2   L3:
3   goto L1
4   L2:
5   ASSIGN
6   L1:
```

Java Bytecode: 使用地址值作为跳转目标

```
outer:
for (int i = 2; i < 1000; i++) {
    for (int j = 2; j < i; j++) {
        if (i % j == 0)
            continue outer;
    }
    System.out.println (i);
}
```

Java Bytecode: 使用地址值作为跳转目标

```
outer:
for (int i = 2; i < 1000; i++) {
    for (int j = 2; j < i; j++) {
        if (i % j == 0)
            continue outer;
    }
    System.out.println (i);
}
```

```
0:  iconst_2
1:  istore_1
2:  iload_1
3:  sipush  1000
6:  if_icmpge       44
9:  iconst_2
10: istore_2
11: iload_2
12: iload_1
13: if_icmpge       31
16: iload_1
17: iload_2
18: irem
19: ifne    25
22: goto    38
25: iinc    2, 1
28: goto    11
31: getstatic       #84;
34: iload_1
35: invokevirtual   #85;
38: iinc    1, 1
41: goto    2
44: return
```

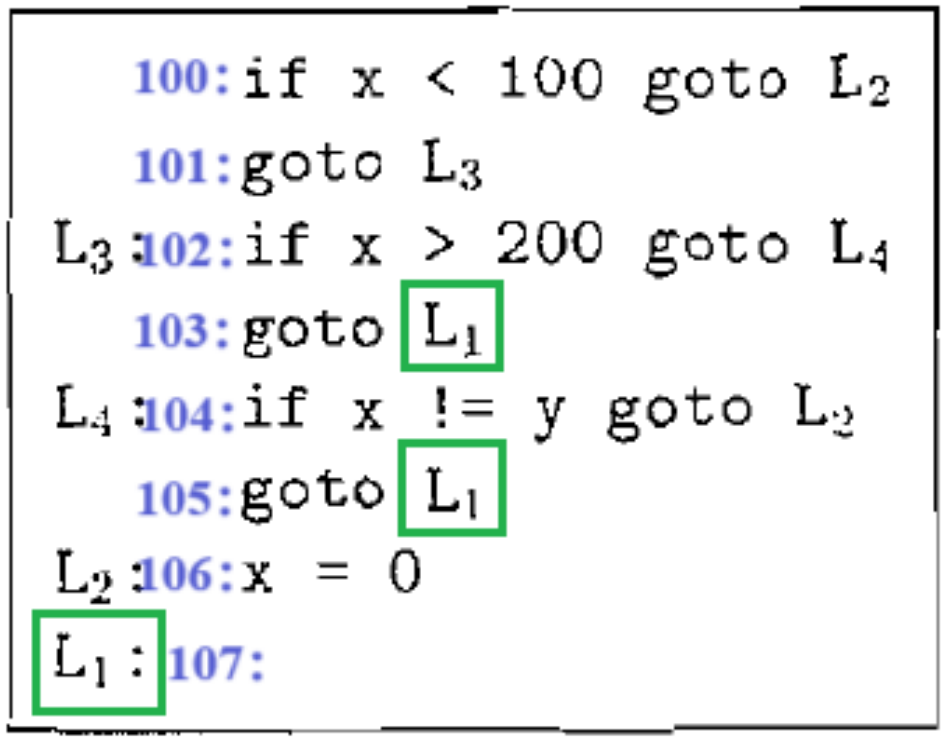
Java Bytecode: 使用地址值作为跳转目标

```
outer:
for (int i = 2; i < 1000; i++) {
    for (int j = 2; j < i; j++) {
        if (i % j == 0)
            continue outer;
    }
    System.out.println (i);
}
```

```
0:  iconst_2
1:  istore_1
2:  iload_1
3:  sipush  1000
6:  if_icmpge       44
9:  iconst_2
10: istore_2
11: iload_2
12: iload_1
13: if_icmpge       31
16: iload_1
17: iload_2
18: irem
19: ifne    25
22: goto    38
25: iinc    2, 1
28: goto    11
31: getstatic       #84;
34: iload_1
35: invokevirtual   #85;
38: iinc    1, 1
41: goto    2
44: return
```

Q：如何在一趟扫描中生成跳转目标的地址?

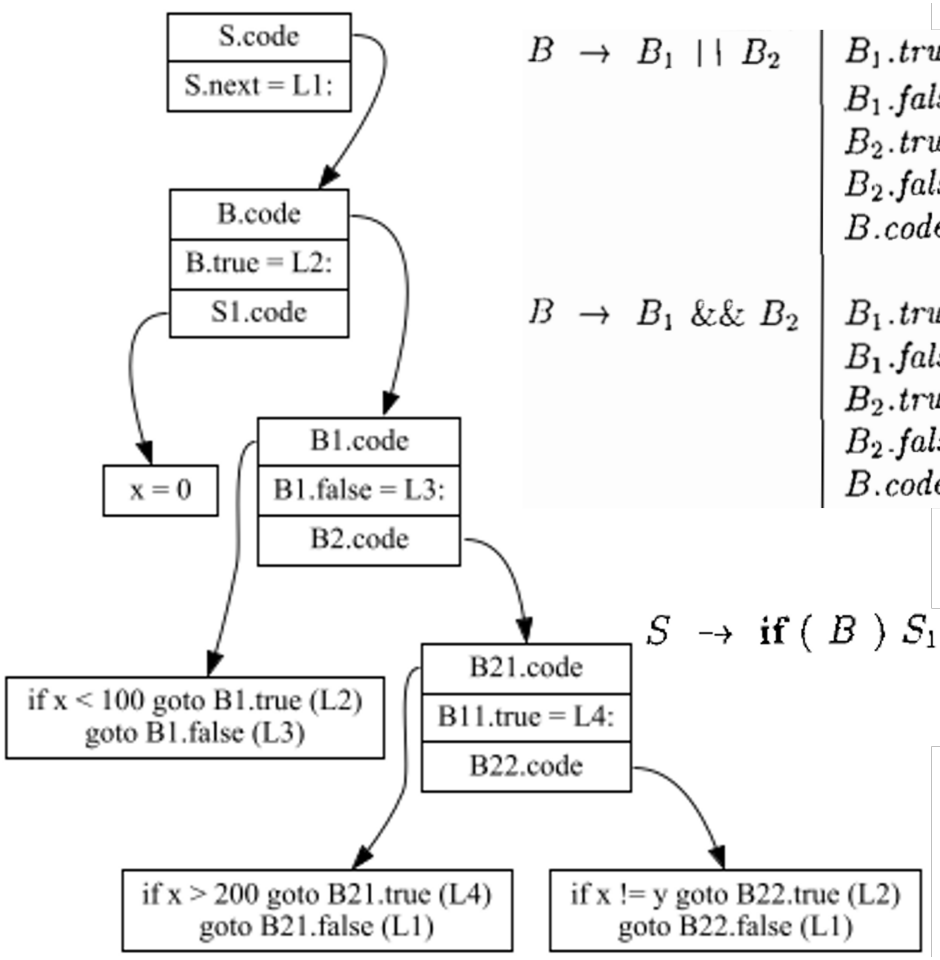
```
if (x < 100 || x > 200 && x != y) x = 0;
```



L_1 标签的位置是由 $P \to S\ (\to \textbf{if}\ B\ S_1)$ 确定的,

但是, 生成 B 所对应的中间代码 **goto** L_1 时, 尚不知道 L_1 的地址

```
if (x < 100 || x > 200 && x != y) x = 0;
```



$B \rightarrow B_1 \;||\; B_2$

$B_1.true = B.true$
$B_1.false = newlabel()$
$B_2.true = B.true$
$B_2.false = B.false$
$B.code = B_1.code \;||\; label(B_1.false) \;||\; B_2.code$

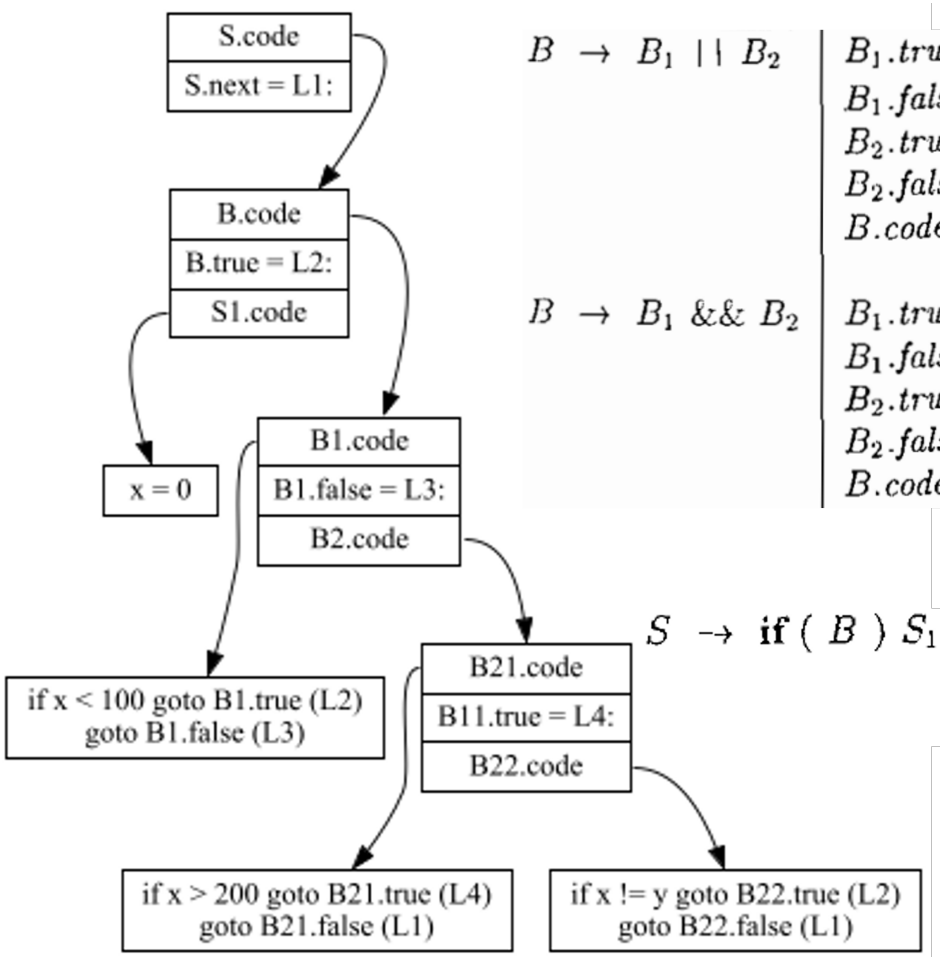
$B \rightarrow B_1 \;\&\&\; B_2$

$B_1.true = newlabel()$
$B_1.false = B.false$
$B_2.true = B.true$
$B_2.false = B.false$
$B.code = B_1.code \;||\; label(B_1.true) \;||\; B_2.code$

$S \rightarrow \textbf{if} \; (\; B \;) \; S_1$

$B.true = newlabel()$
$B.false = S_1.next = S.next$
$S.code = B.code \;||\; label(B.true) \;||\; S_1.code$

```
if (x < 100 || x > 200 && x != y) x = 0;
```



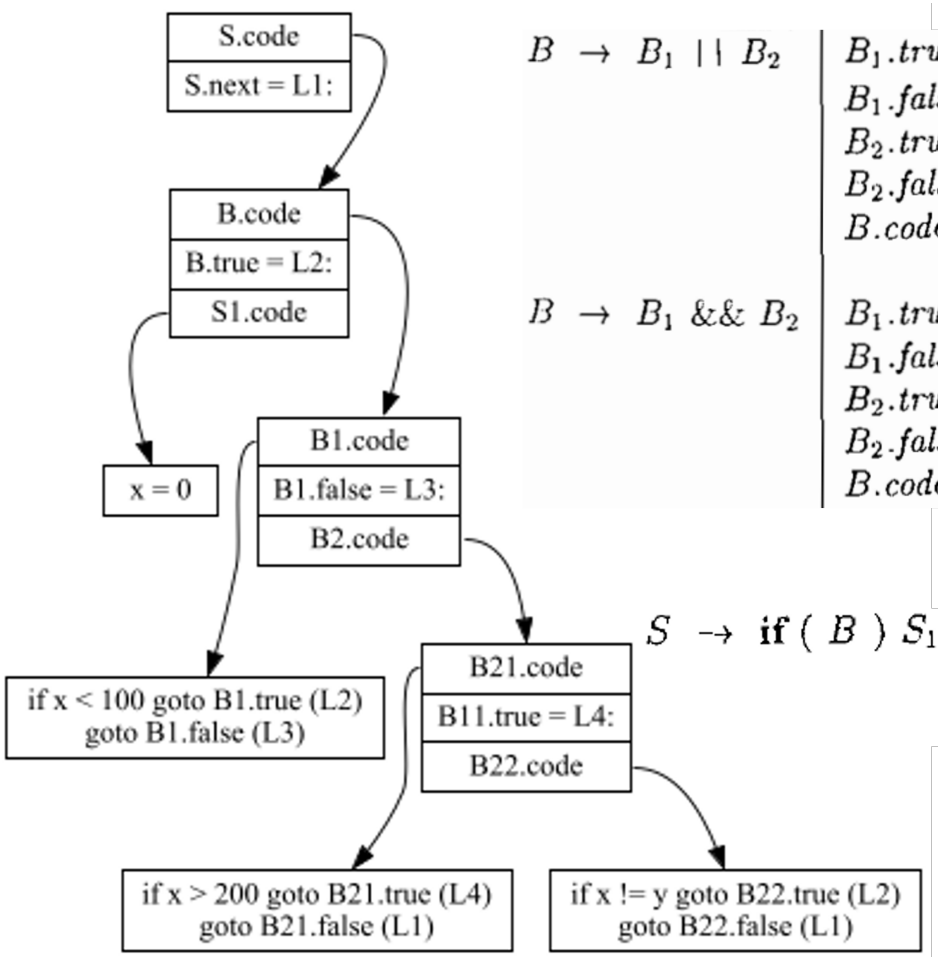
$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \;\&\&\; B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$
$S \rightarrow \textbf{if}\ (\ B\ )\ S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$

$$B_{21}.false \leftarrow B_2.false \leftarrow B.false \leftarrow S.next = L_1$$

$$B_{22}.false \leftarrow B_2.false \leftarrow B.false \leftarrow S.next = L_1$$

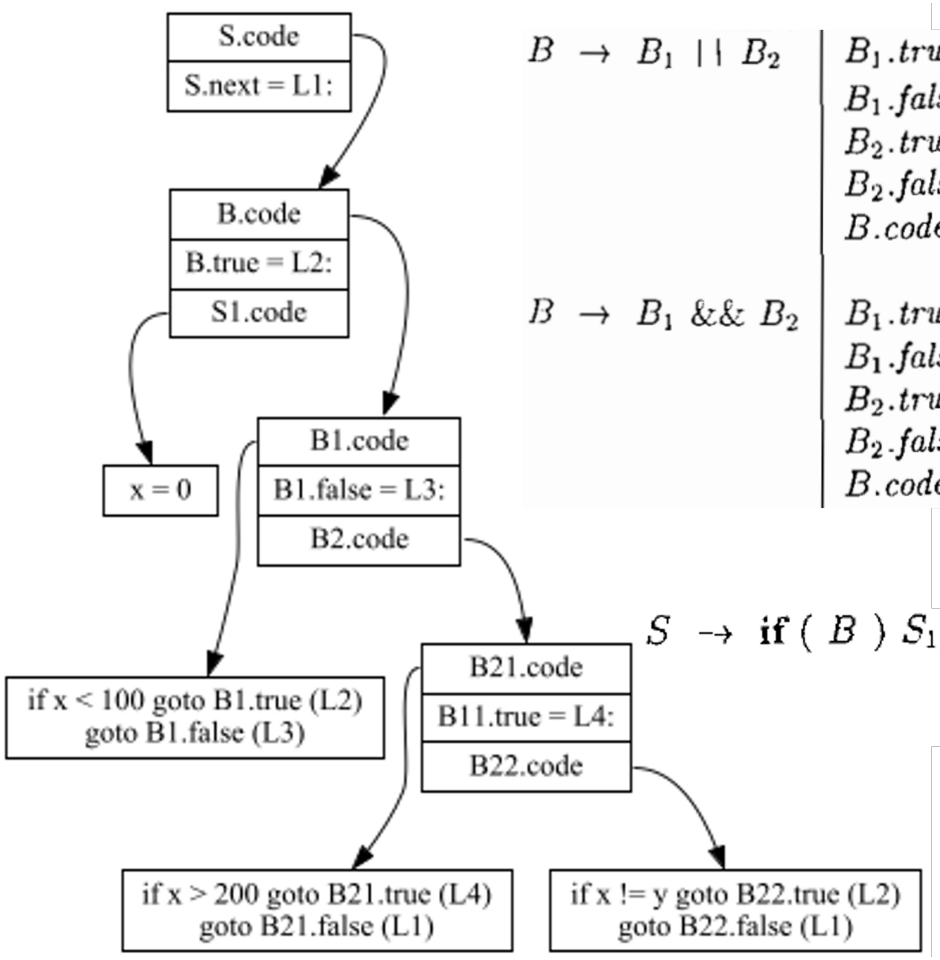

諸君...我有個大膽的想法!!

```
if (x < 100 || x > 200 && x != y) x = 0;
```



$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \;\&\&\; B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$
$S \rightarrow \textbf{if}\ (\ B\ )\ S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$

```
if (x < 100 || x > 200 && x != y) x = 0;
```



$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \;\&\&\; B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$
$S \rightarrow \textbf{if}\ (\ B\ )\ S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$

$$\{103\} = B_{21}.false \rightarrow B_2.false \rightarrow B.false \rightarrow S.next = 107$$

$$\{105\} = B_{22}.false \rightarrow B_2.false \rightarrow B.false \rightarrow S.next = 107$$

子节点暂时不指定跳转指令的目标地址

```
   100: if x < 100 goto L2
   101: goto L3
L3 102: if x > 200 goto L4
   103: goto
L4 104: if x != y goto L2
   105: goto
L2 106: x = 0
L1 : 107:
```

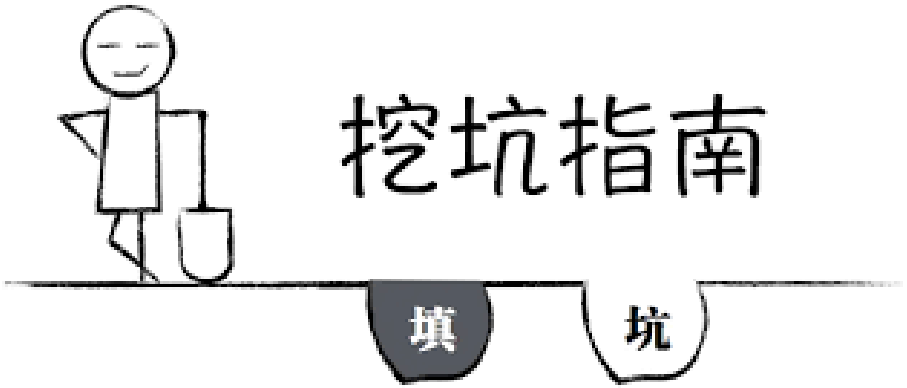
待祖先节点能够确定目标地址时回头填充

子节点暂时不指定跳转指令的目标地址

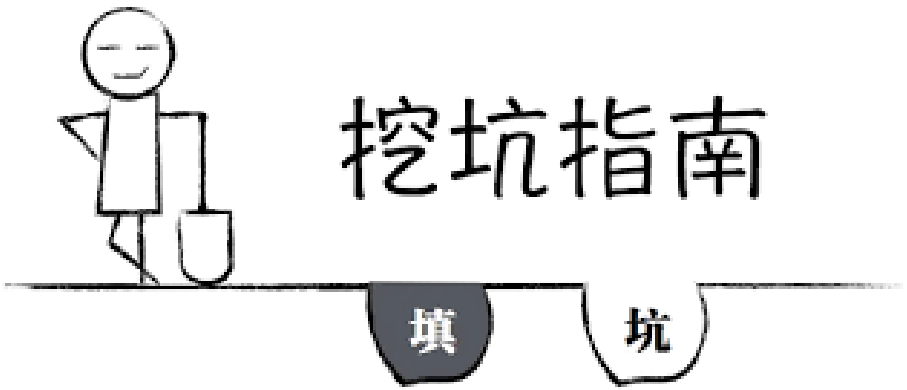
```
   100: if x < 100 goto L2
   101: goto L3
L3 102: if x > 200 goto L4
   103: goto
L4 104: if x != y goto L2
   105: goto
L2 106: x = 0
L1 : 107:
```

待祖先节点能够确定目标地址时回头填充

回填 (Backpatching) 技术: 子节点挖坑、祖先节点填坑



回填 (Backpatching) 技术: 子节点挖坑、祖先节点填坑



祖先节点通过**综合属性**收集子节点中具有相同目标的跳转指令

南北爱情故事

$B.truelist$ 保存需要跳转到 $B.true$ 标签的指令

$B.falselist$ 保存需要跳转到 $B.false$ 标签的指令

1) $S \rightarrow \textbf{if}\ (\ B\ )\ M\ S_1$ { $backpatch(B.truelist,\ M.instr)$;
$S.nextlist = merge(B.falselist,\ S_1.nextlist)$; }

2) $S \rightarrow \textbf{if}\ (\ B\ )\ M_1\ S_1\ N\ \textbf{else}\ M_2\ S_2$
{ $backpatch(B.truelist,\ M_1.instr)$;
$backpatch(B.falselist,\ M_2.instr)$;
$temp = merge(S_1.nextlist,\ N.nextlist)$;
$S.nextlist = merge(temp,\ S_2.nextlist)$; }

3) $S \rightarrow \textbf{while}\ M_1\ (\ B\ )\ M_2\ S_1$
{ $backpatch(S_1.nextlist,\ M_1.instr)$;
$backpatch(B.truelist,\ M_2.instr)$;
$S.nextlist = B.falselist$;
$gen(\text{'goto'}\ M_1.instr)$; }

4) $S \rightarrow \{\ L\ \}$ { $S.nextlist = L.nextlist$; }

5) $S \rightarrow A\ ;$ { $S.nextlist = \textbf{null}$; }

6) $M \rightarrow \epsilon$ { $M.instr = nextinstr$; }

7) $N \rightarrow \epsilon$ { $N.nextlist = makelist(nextinstr)$;
$gen(\text{'goto\ \_'})$; }

8) $L \rightarrow L_1\ M\ S$ { $backpatch(L_1.nextlist,\ M.instr)$;
$L.nextlist = S.nextlist$; }

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

$S/L.nextlist$ 保存需要跳转到 $S/L.next$ 标签的指令

为左部非终结符 B 计算综合属性 $B.truelist$ 与 $B.falselist$

为左部非终结符 S/L 计算综合属性 $S/L.nextlist$

1) $S \rightarrow \textbf{if}\ (\ B\ )\ M\ S_1$ { $backpatch(B.truelist,\ M.instr)$;
$S.nextlist = merge(B.falselist,\ S_1.nextlist)$; }

2) $S \rightarrow \textbf{if}\ (\ B\ )\ M_1\ S_1\ N\ \textbf{else}\ M_2\ S_2$
{ $backpatch(B.truelist,\ M_1.instr)$;
$backpatch(B.falselist,\ M_2.instr)$;
$temp = merge(S_1.nextlist,\ N.nextlist)$;
$S.nextlist = merge(temp,\ S_2.nextlist)$; }

3) $S \rightarrow \textbf{while}\ M_1\ (\ B\ )\ M_2\ S_1$
{ $backpatch(S_1.nextlist,\ M_1.instr)$;
$backpatch(B.truelist,\ M_2.instr)$;
$S.nextlist = B.falselist$;
$gen(\text{'goto'}\ M_1.instr)$; }

4) $S \rightarrow \{\ L\ \}$ { $S.nextlist = L.nextlist$; }

5) $S \rightarrow A\ ;$ { $S.nextlist = \textbf{null}$; }

6) $M \rightarrow \epsilon$ { $M.instr = nextinstr$; }

7) $N \rightarrow \epsilon$ { $N.nextlist = makelist(nextinstr)$;
$gen(\text{'goto\ \_'})$; }

8) $L \rightarrow L_1\ M\ S$ { $backpatch(L_1.nextlist,\ M.instr)$;
$L.nextlist = S.nextlist$; }

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

并为已能确定目标地址的跳转指令进行回填 (考虑每个综合属性)

1) $B \rightarrow B_1 \ || \ M \ B_2$

2) $B \rightarrow B_1 \ \&\& \ M \ B_2$

3) $B \rightarrow \ ! \ B_1$

4) $B \rightarrow (\ B_1 \ )$

5) $B \rightarrow E_1$ **rel** E_2

6) $B \rightarrow$ **true**

7) $B \rightarrow$ **false**

8) $M \rightarrow \epsilon$

$B.truelist$ 保存需要跳转到 $B.true$ 标签的指令

6) $B \rightarrow$ **true** { $B.truelist = makelist(nextinstr)$;
$gen('\texttt{goto}\ \_')$; }

7) $B \rightarrow$ **false** { $B.falselist = makelist(nextinstr)$;
$gen('\texttt{goto}\ \_')$; }

$B.falselist$ 保存需要跳转到 $B.false$ 标签的指令

$B.truelist$ 保存需要跳转到 $B.true$ 标签的指令

6) $B \rightarrow \textbf{true}$ { $B.truelist = makelist(nextinstr)$; $gen('\texttt{goto}\ \_')$; }

7) $B \rightarrow \textbf{false}$ { $B.falselist = makelist(nextinstr)$; $gen('\texttt{goto}\ \_')$; }

$B.falselist$ 保存需要跳转到 $B.false$ 标签的指令

$B \rightarrow \textbf{true}$	$B.code = gen('\texttt{goto}'\ B.true)$
$B \rightarrow \textbf{false}$	$B.code = gen('\texttt{goto}'\ B.false)$

5) $B \rightarrow E_1 \ \mathbf{rel} \ E_2$ { $B.truelist = makelist(nextinstr)$;
$B.falselist = makelist(nextinstr + 1)$;
$gen('\texttt{if}' \ E_1.addr \ \mathbf{rel}.op \ E_2.addr \ '\texttt{goto} \ \_')$;
$gen('\texttt{goto} \ \_')$; }

5) $B \rightarrow E_1 \ \mathbf{rel} \ E_2$ { $B.truelist = makelist(nextinstr)$;
$B.falselist = makelist(nextinstr + 1)$;
$gen('\texttt{if}' \ E_1.addr \ \mathbf{rel}.op \ E_2.addr \ '\texttt{goto}\ \_')$;
$gen('\texttt{goto}\ \_')$; }

$$B \rightarrow E_1 \ \mathbf{rel} \ E_2 \quad \Big| \quad \begin{aligned} B.code = {} & E_1.code \ || \ E_2.code \\ & || \ gen('\texttt{if}' \ E_1.addr \ \mathbf{rel}.op \ E_2.addr \ '\texttt{goto}' \ B.true) \\ & || \ gen('\texttt{goto}' \ B.false) \end{aligned}$$

3) $B \rightarrow\ !\ B_1$ { $B.truelist = B_1.falselist$;
$B.falselist = B_1.truelist$; }

4) $B \rightarrow (\ B_1\ )$ { $B.truelist = B_1.truelist$;
$B.falselist = B_1.falselist$; }

3)	$B \rightarrow\ !\ B_1$	{ $B.truelist = B_1.falselist$; $B.falselist = B_1.truelist$; }
4)	$B \rightarrow (\ B_1\ )$	{ $B.truelist = B_1.truelist$; $B.falselist = B_1.falselist$; }

$B \rightarrow\ !\ B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$

2) $B \rightarrow B_1 \ \&\& \ M \ B_2$ { $backpatch(B_1.truelist, M.instr)$;
$B.truelist = B_2.truelist$;
$B.falselist = merge(B_1.falselist, B_2.falselist)$; }

2) $B \rightarrow B_1 \;\&\&\; M \; B_2$ { $backpatch(B_1.truelist, M.instr)$;
$B.truelist = B_2.truelist$;
$B.falselist = merge(B_1.falselist, B_2.falselist)$; }

8) $M \rightarrow \epsilon$ { $M.instr = nextinstr$; }

2) $B \rightarrow B_1 \;\&\&\; M \; B_2$ { $backpatch(B_1.truelist, M.instr)$;
$B.truelist = B_2.truelist$;
$B.falselist = merge(B_1.falselist, B_2.falselist)$; }

8) $M \rightarrow \epsilon$ { $M.instr = nextinstr$; }

$B \rightarrow B_1 \;\&\&\; B_2$ | $B_1.true = newlabel()$
$B_1.false = B.false$
$B_2.true = B.true$
$B_2.false = B.false$
$B.code = B_1.code \;||\; label(B_1.true) \;||\; B_2.code$

1) $B \rightarrow B_1 \ || \ M \ B_2$ { $backpatch(B_1.falselist, M.instr)$;
$B.truelist = merge(B_1.truelist, B_2.truelist)$;
$B.falselist = B_2.falselist$; }

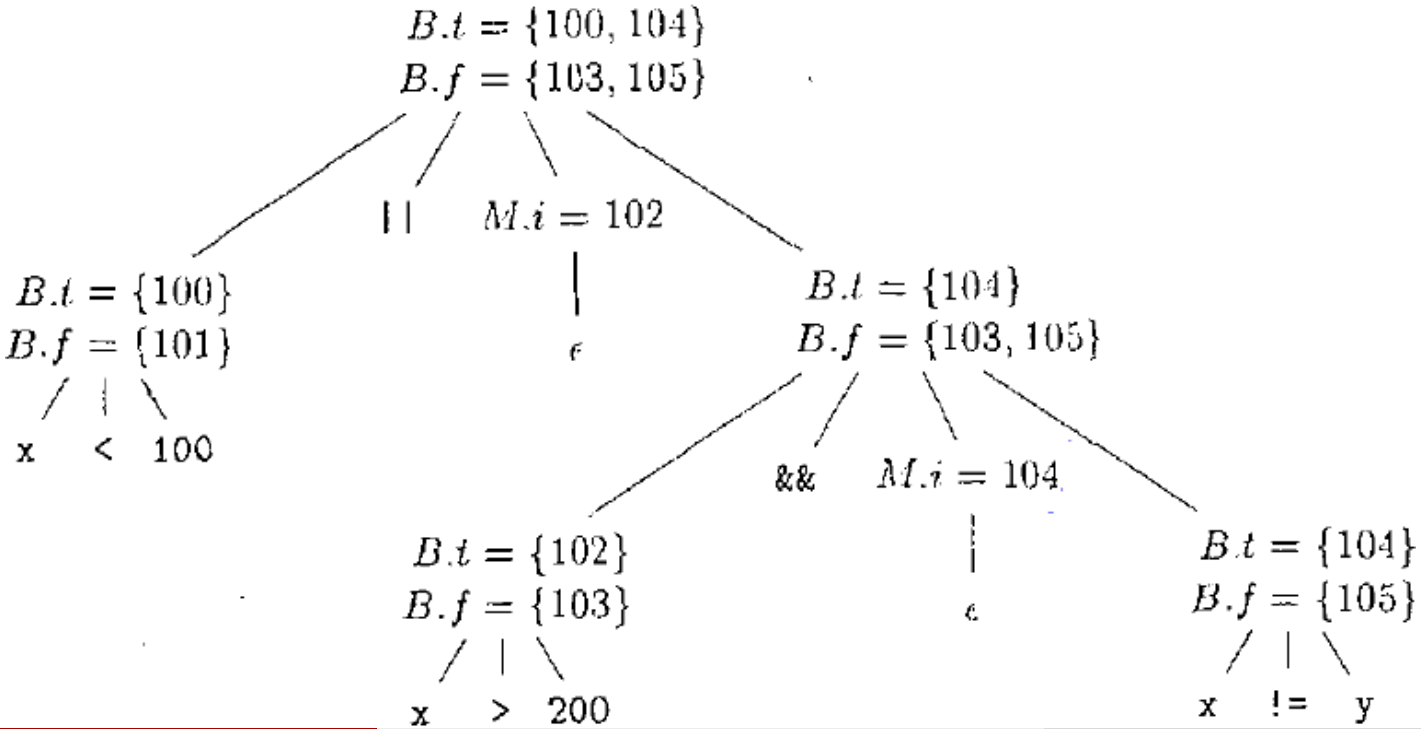
8) $M \rightarrow \epsilon$ { $M.instr = nextinstr$; }

1) $B \rightarrow B_1 \ || \ M \ B_2$ { $backpatch(B_1.falselist, M.instr)$;
$B.truelist = merge(B_1.truelist, B_2.truelist)$;
$B.falselist = B_2.falselist$; }

8) $M \rightarrow \epsilon$ { $M.instr = nextinstr$; }

$B \rightarrow B_1 \ || \ B_2$

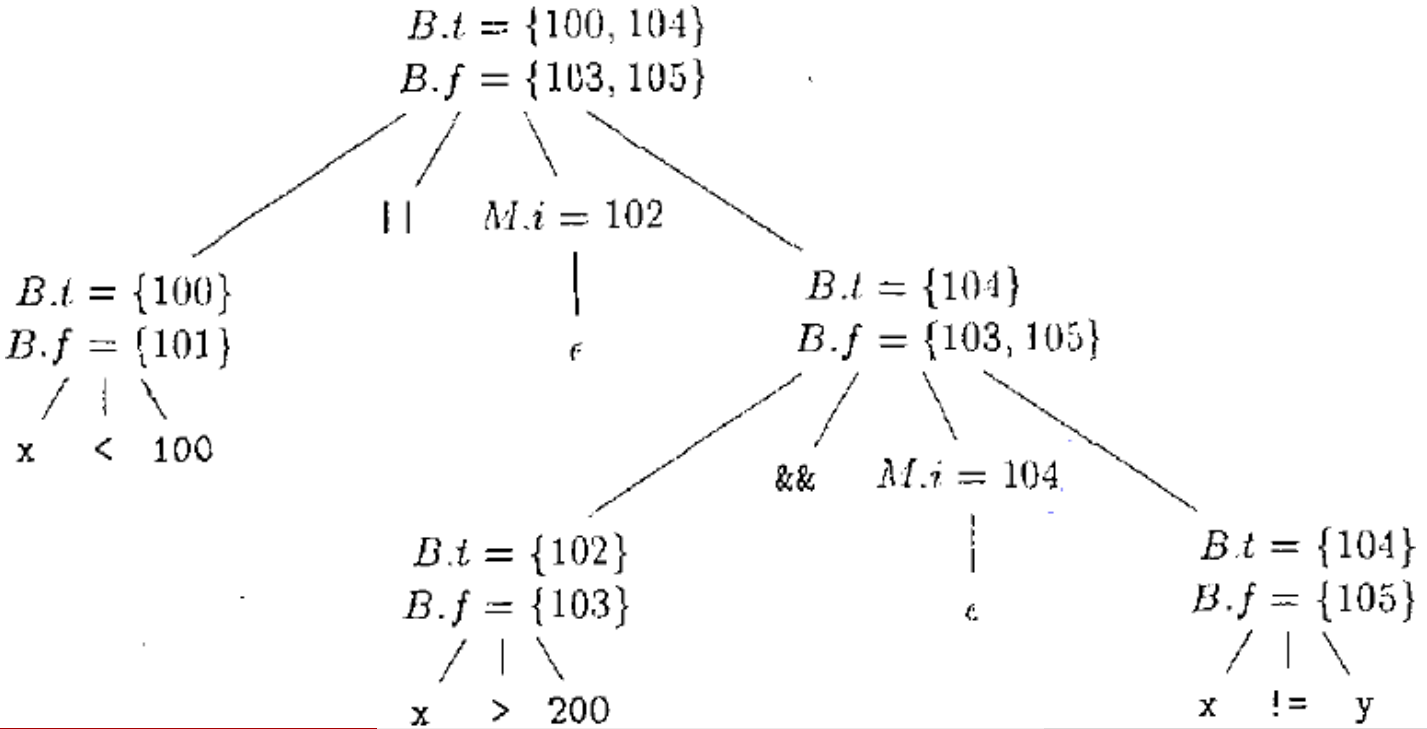
$B_1.true = B.true$
$B_1.false = newlabel()$
$B_2.true = B.true$
$B_2.false = B.false$
$B.code = B_1.code \ || \ label(B_1.false) \ || \ B_2.code$


B.t = {100, 104}
B.f = {103, 105}
||
M.i = 102
ϵ
B.t = {100}
B.f = {101}
x < 100
B.t = {104}
B.f = {103, 105}
&&
M.i = 104
ϵ
B.t = {102}
B.f = {103}
x > 200
B.t = {104}
B.f = {105}
x != y

1) $B \rightarrow B_1 \ || \ M \ B_2$ { $backpatch(B_1.falselist, M.instr)$;
$B.truelist = merge(B_1.truelist, B_2.truelist)$;
$B.falselist = B_2.falselist$; }

2) $B \rightarrow B_1 \ \&\& \ M \ B_2$ { $backpatch(B_1.truelist, M.instr)$;
$B.truelist = B_2.truelist$;
$B.falselist = merge(B_1.falselist, B_2.falselist)$; }

5) $B \rightarrow E_1 \ \mathbf{rel} \ E_2$ { $B.truelist = makelist(nextinstr)$;
$B.falselist = makelist(nextinstr + 1)$;
$gen('\texttt{if}' \ E_1.addr \ \mathbf{rel}.op \ E_2.addr \ '\texttt{goto}\ \_')$;
$gen('\texttt{goto}\ \_')$; }



```
100:   if x < 100 goto _
101:   goto _
102:   if x > 200 goto 104
103:   goto _
104:   if x != y goto _
105:   goto _
```

a) 将 104 回填到指令 102 中之后

```
100:   if x < 100 goto _
101:   goto 102
102:   if x > 200 goto 104
103:   goto _
104:   if x != y goto _
105:   goto _
```

b) 将 102 回填到指令 101 中之后

1) $S \rightarrow \textbf{if}\ (\ B\ )\ M\ S_1$

2) $S \rightarrow \textbf{if}\ (\ B\ )\ M_1\ S_1\ N\ \textbf{else}\ M_2\ S_2$

3) $S \rightarrow \textbf{while}\ M_1\ (\ B\ )\ M_2\ S_1$

4) $S \rightarrow \{\ L\ \}$

5) $S \rightarrow A\ ;$

6) $M \rightarrow \epsilon$

7) $N \rightarrow \epsilon$

8) $L \rightarrow L_1\ M\ S$

9) $L \rightarrow S$

1) $S \rightarrow \textbf{if}\ (\ B\ )\ M\ S_1\ \{\ backpatch(B.truelist,\ M.instr);$
$S.nextlist\ =\ merge(B.falselist,\ S_1.nextlist);\ \}$

8) $M \rightarrow \epsilon \qquad \{\ M.instr = nextinstr;\ \}$

1) $S \rightarrow \textbf{if}\ (\ B\ )\ M\ S_1\ \{\ backpatch(B.truelist,\ M.instr);$
$S.nextlist\ =\ merge(B.falselist,\ S_1.nextlist);\ \}$

8) $M \rightarrow \epsilon \qquad \{\ M.instr = nextinstr;\ \}$

$S \rightarrow \textbf{if}\ (\ B\ )\ S_1$

$B.true\ =\ newlabel()$
$B.false\ =\ S_1.next\ =\ S.next$
$S.code\ =\ B.code\ ||\ label(B.true)\ ||\ S_1.code$

2) $S \rightarrow$ **if** (B) $M_1\ S_1\ N$ **else** $M_2\ S_2$

{ $backpatch(B.truelist,\ M_1.instr)$;
$backpatch(B.falselist,\ M_2.instr)$;
$temp\ =\ merge(S_1.nextlist,\ N.nextlist)$;
$S.nextlist\ =\ merge(temp,\ S_2.nextlist)$; }

6) $M \rightarrow \epsilon$ { $M.instr\ =\ nextinstr$; }

7) $N \rightarrow \epsilon$ { $N.nextlist\ =\ makelist(nextinstr)$;
$gen('\texttt{goto}\ \_')$; }

2) $S \rightarrow$ **if** (B) M_1 S_1 N **else** M_2 S_2

{ *backpatch*($B.truelist$, $M_1.instr$);
backpatch($B.falselist$, $M_2.instr$);
$temp$ = *merge*($S_1.nextlist$, $N.nextlist$);
$S.nextlist$ = *merge*($temp$, $S_2.nextlist$); }

6) $M \rightarrow \epsilon$ { $M.instr$ = $nextinstr$; }

7) $N \rightarrow \epsilon$ { $N.nextlist$ = *makelist*($nextinstr$);
gen('goto _'); }

$S \rightarrow$ **if** (B) S_1 **else** S_2	$B.true$ = *newlabel*() $B.false$ = *newlabel*() $S_1.next$ = $S_2.next$ = $S.next$ $S.code$ = $B.code$ || *label*($B.true$) || $S_1.code$ || *gen*('goto' $S.next$) || *label*($B.false$) || $S_2.code$

3) $S \rightarrow$ **while** M_1 (B) M_2 S_1

{ $backpatch(S_1.nextlist,\ M_1.instr)$;
$backpatch(B.truelist,\ M_2.instr)$;
$S.nextlist = B.falselist$;
$gen('\texttt{goto}'\ M_1.instr)$; }

8) $M \rightarrow \epsilon$ { $M.instr = nextinstr$; }

3) $S \rightarrow$ **while** M_1 (B) M_2 S_1

{ $backpatch(S_1.nextlist,\ M_1.instr)$;
$backpatch(B.truelist,\ M_2.instr)$;
$S.nextlist = B.falselist$;
$gen('\texttt{goto}'\ M_1.instr)$; }

8) $M \rightarrow \epsilon$ { $M.instr = nextinstr$; }

$S \rightarrow$ **while** (B) S_1	$begin = newlabel()$ $B.true = newlabel()$ $B.false = S.next$ $S_1.next = begin$ $S.code = label(begin) \mid\mid B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('\texttt{goto}'\ begin)$

8) $L \rightarrow L_1\ M\ S$ { $backpatch(L_1.nextlist,\ M.instr)$;
$L.nextlist = S.nextlist$; }

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

4) $S \rightarrow \{\ L\ \}$ $\{\ S.nextlist = L.nextlist;\ \}$

5) $S \rightarrow A\ ;$ $\{\ S.nextlist = \mathbf{null};\ \}$

1) $S \rightarrow$ **if** (B) $M\ S_1$ { $backpatch(B.truelist,\ M.instr)$;
$S.nextlist\ =\ merge(B.falselist,\ S_1.nextlist)$; }

2) $S \rightarrow$ **if** (B) $M_1\ S_1\ N$ **else** $M_2\ S_2$
{ $backpatch(B.truelist,\ M_1.instr)$;
$backpatch(B.falselist,\ M_2.instr)$;
$temp\ =\ merge(S_1.nextlist,\ N.nextlist)$;
$S.nextlist\ =\ merge(temp,\ S_2.nextlist)$; }

3) $S \rightarrow$ **while** M_1 (B) $M_2\ S_1$
{ $backpatch(S_1.nextlist,\ M_1.instr)$;
$backpatch(B.truelist,\ M_2.instr)$;
$S.nextlist\ =\ B.falselist$;
$gen('\texttt{goto}'\ M_1.instr)$; }

4) $S \rightarrow$ { L } { $S.nextlist\ =\ L.nextlist$; }

5) $S \rightarrow A$; { $S.nextlist\ =$ **null**; }

6) $M \rightarrow \epsilon$ { $M.instr\ =\ nextinstr$; }

7) $N \rightarrow \epsilon$ { $N.nextlist\ =\ makelist(nextinstr)$;
$gen('\texttt{goto}\ \_')$; }

8) $L \rightarrow L_1\ M\ S$ { $backpatch(L_1.nextlist,\ M.instr)$;
$L.nextlist\ =\ S.nextlist$; }

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

只有 (3) 与 (7) 生成了新的代码, 控制流语句的主要目的是“控制”流。

```
1: procedure AreYouOK(score)
2:     if score ≥ 60 then
3:         while true do
4:             print “WanSui”
5:     else
6:         print “Sad”
```

2) $S \rightarrow$ **if** (B) M_1 S_1 N **else** M_2 S_2

{ $backpatch(B.truelist,\ M_1.instr)$;
$backpatch(B.falselist,\ M_2.instr)$;
$temp\ =\ merge(S_1.nextlist,\ N.nextlist)$;
$S.nextlist\ =\ merge(temp,\ S_2.nextlist)$; }

3) $S \rightarrow$ **while** M_1 (B) M_2 S_1

{ $backpatch(S_1.nextlist,\ M_1.instr)$;
$backpatch(B.truelist,\ M_2.instr)$;
$S.nextlist\ =\ B.falselist$;
$gen(\texttt{'goto'}\ M_1.instr)$; }

2) $S \rightarrow$ **if** (B) M_1 S_1 N **else** M_2 S_2

{ $backpatch(B.truelist,\ M_1.instr)$;
$backpatch(B.falselist,\ M_2.instr)$;
$temp\ =\ merge(S_1.nextlist,\ N.nextlist)$;
$S.nextlist\ =\ merge(temp,\ S_2.nextlist)$; }

3) $S \rightarrow$ **while** M_1 (B) M_2 S_1

{ $backpatch(S_1.nextlist,\ M_1.instr)$;
$backpatch(B.truelist,\ M_2.instr)$;
$S.nextlist\ =\ B.falselist$;
$gen('\texttt{goto}'\ M_1.instr)$; }

6) $M \rightarrow \epsilon$ { $M.instr\ =\ nextinstr$; }

7) $N \rightarrow \epsilon$ { $N.nextlist\ =\ makelist(nextinstr)$;
$gen('\texttt{goto}\ \_')$; }

2) $S \rightarrow$ **if** (B) M_1 S_1 N **else** M_2 S_2

{ $backpatch(B.truelist,\ M_1.instr)$;
$backpatch(B.falselist,\ M_2.instr)$;
$temp\ =\ merge(S_1.nextlist,\ N.nextlist)$;
$S.nextlist\ =\ merge(temp,\ S_2.nextlist)$; }

3) $S \rightarrow$ **while** M_1 (B) M_2 S_1

{ $backpatch(S_1.nextlist,\ M_1.instr)$;
$backpatch(B.truelist,\ M_2.instr)$;
$S.nextlist\ =\ B.falselist$;
$gen('\texttt{goto}'\ M_1.instr)$; }

6) $M \rightarrow \epsilon$ { $M.instr\ =\ nextinstr$; }

7) $N \rightarrow \epsilon$ { $N.nextlist\ =\ makelist(nextinstr)$;
$gen('\texttt{goto}\ \_')$; }

5) $B \rightarrow E_1$ **rel** E_2 { $B.truelist = makelist(nextinstr)$;
$B.falselist = makelist(nextinstr + 1)$;
$gen('\texttt{if}'\ E_1.addr$ **rel**.$op\ E_2.addr\ '\texttt{goto}\ \_')$;
$gen('\texttt{goto}\ \_')$; }

6) $B \rightarrow$ **true** { $B.truelist = makelist(nextinstr)$;
$gen('\texttt{goto}\ \_')$; }

$B.truelist$ 保存需要跳转到 $B.true$ 标签的指令

$B.falselist$ 保存需要跳转到 $B.false$ 标签的指令

1)	$S \rightarrow \textbf{if}\ (\ B\ )\ M\ S_1$	{ $backpatch(B.truelist,\ M.instr)$; $S.nextlist = merge(B.falselist,\ S_1.nextlist)$; }
2)	$S \rightarrow \textbf{if}\ (\ B\ )\ M_1\ S_1\ N\ \textbf{else}\ M_2\ S_2$	{ $backpatch(B.truelist,\ M_1.instr)$; $backpatch(B.falselist,\ M_2.instr)$; $temp = merge(S_1.nextlist,\ N.nextlist)$; $S.nextlist = merge(temp,\ S_2.nextlist)$; }
3)	$S \rightarrow \textbf{while}\ M_1\ (\ B\ )\ M_2\ S_1$	{ $backpatch(S_1.nextlist,\ M_1.instr)$; $backpatch(B.truelist,\ M_2.instr)$; $S.nextlist = B.falselist$; $gen(\text{'goto'}\ M_1.instr)$; }
4)	$S \rightarrow \{\ L\ \}$	{ $S.nextlist = L.nextlist$; }
5)	$S \rightarrow A\ ;$	{ $S.nextlist = \textbf{null}$; }
6)	$M \rightarrow \epsilon$	{ $M.instr = nextinstr$; }
7)	$N \rightarrow \epsilon$	{ $N.nextlist = makelist(nextinstr)$; $gen(\text{'goto\ \_'})$; }
8)	$L \rightarrow L_1\ M\ S$	{ $backpatch(L_1.nextlist,\ M.instr)$; $L.nextlist = S.nextlist$; }
9)	$L \rightarrow S$	{ $L.nextlist = S.nextlist$; }

$S/L.nextlist$ 保存需要跳转到 $S/L.next$ 标签的指令

为左部非终结符 B 计算综合属性 $B.truelist$ 与 $B.falselist$

为左部非终结符 S/L 计算综合属性 $S/L.nextlist$

1) $S \rightarrow \textbf{if}\ (\ B\ )\ M\ S_1$ { $backpatch(B.truelist,\ M.instr)$;
$S.nextlist = merge(B.falselist,\ S_1.nextlist)$; }

2) $S \rightarrow \textbf{if}\ (\ B\ )\ M_1\ S_1\ N\ \textbf{else}\ M_2\ S_2$
{ $backpatch(B.truelist,\ M_1.instr)$;
$backpatch(B.falselist,\ M_2.instr)$;
$temp = merge(S_1.nextlist,\ N.nextlist)$;
$S.nextlist = merge(temp,\ S_2.nextlist)$; }

3) $S \rightarrow \textbf{while}\ M_1\ (\ B\ )\ M_2\ S_1$
{ $backpatch(S_1.nextlist,\ M_1.instr)$;
$backpatch(B.truelist,\ M_2.instr)$;
$S.nextlist = B.falselist$;
$gen(\text{'goto'}\ M_1.instr)$; }

4) $S \rightarrow \{\ L\ \}$ { $S.nextlist = L.nextlist$; }

5) $S \rightarrow A\ ;$ { $S.nextlist = \textbf{null}$; }

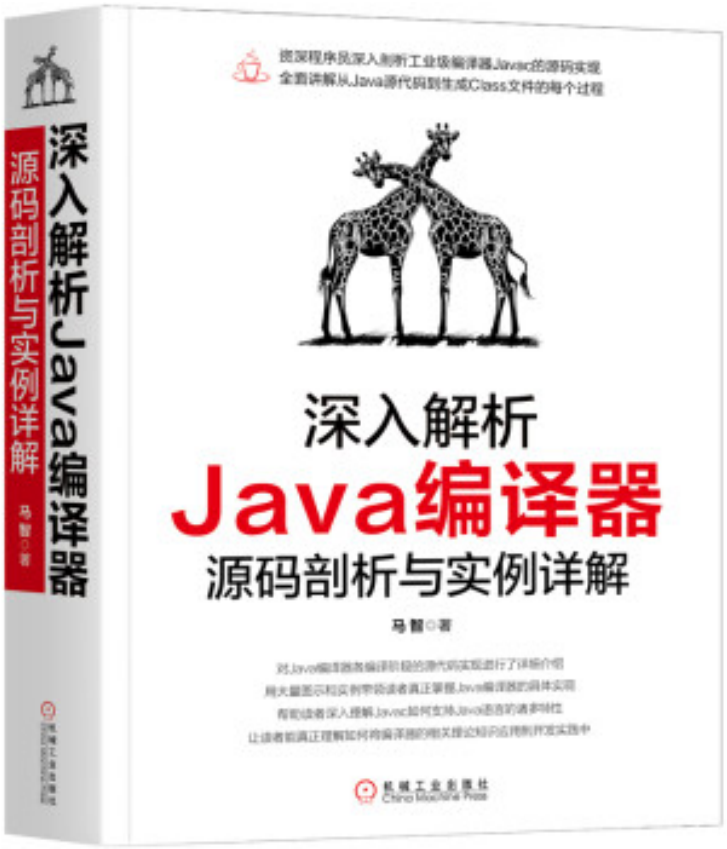
6) $M \rightarrow \epsilon$ { $M.instr = nextinstr$; }

7) $N \rightarrow \epsilon$ { $N.nextlist = makelist(nextinstr)$;
$gen(\text{'goto\ \_'})$; }

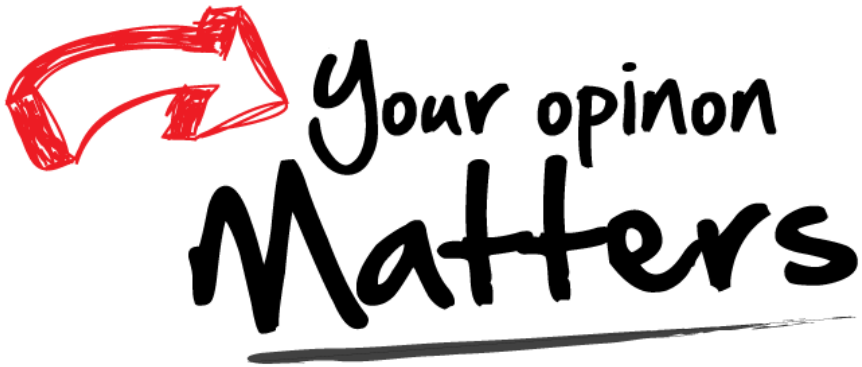
8) $L \rightarrow L_1\ M\ S$ { $backpatch(L_1.nextlist,\ M.instr)$;
$L.nextlist = S.nextlist$; }

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

并为已能确定目标地址的跳转指令进行回填 (考虑每个综合属性)




Thank
You!



Office 926

hfwei@nju.edu.cn