

四、中间代码生成

(5. 控制流语句翻译中的地址回填技术)

魏恒峰

hfwei@nju.edu.cn

2023 年 05 月 19 日







使用标签标记跳转目标

```
1 if (true || false) {  
2     x = 1;  
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);  
}
```

```
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
```

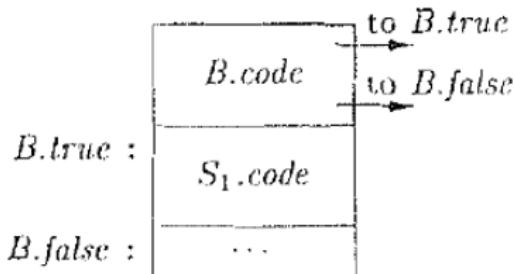
31

Q : 如何在一趟扫描中生成跳转目标的地址?

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

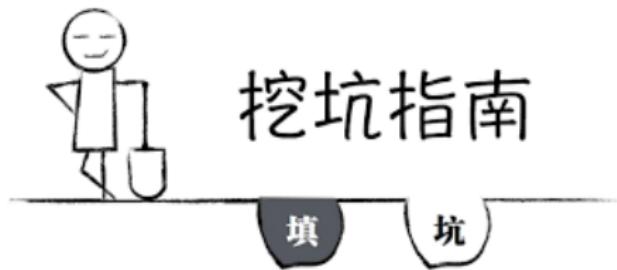
$B.\text{true} = \text{newlabel}()$
 $B.\text{false} = S_1.\text{next} = S.\text{next}$
 $S.\text{code} = B.\text{code} \parallel \text{label}(B.\text{true}) \parallel S_1.\text{code}$

B 可以自行计算 $B.\text{true}$ 对应的指令地址



B 计算不出 $B.\text{false}$ 对应的指令地址

回填 (Backpatching) 技术: 子节点挖坑、祖先节点填坑



子节点暂时不指定跳转指令的目标地址
待祖先节点能够确定目标地址时回头填充

父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

1) $B \rightarrow B_1 \sqcup \boxed{M} B_2$

2) $B \rightarrow B_1 \&& \boxed{M} B_2$

3) $B \rightarrow ! B_1$

4) $B \rightarrow (B_1)$

5) $B \rightarrow E_1 \text{ rel } E_2$

6) $B \rightarrow \mathbf{true}$

7) $B \rightarrow \mathbf{false}$

8) $\boxed{M \rightarrow \epsilon}$

$B.\text{truelist}$ 保存需要跳转到 $B.\text{true}$ 标签的指令

- 6) $B \rightarrow \text{true}$ { $B.\text{truelist} = \text{makelist}(\text{nextinstr});$
 $\quad \text{gen}('goto' __); \}$
- 7) $B \rightarrow \text{false}$ { $B.\text{falselist} = \text{makelist}(\text{nextinstr});$
 $\quad \text{gen}('goto' __); \}$

$B.\text{falselist}$ 保存需要跳转到 $B.\text{false}$ 标签的指令

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

5) $B \rightarrow E_1 \text{ rel } E_2$ $\{ \begin{array}{l} B.\text{truelist} = \text{makelist}(nextinstr); \\ B.\text{falselist} = \text{makelist}(nextinstr + 1); \\ \text{gen('if' } E_1.\text{addr rel.op } E_2.\text{addr 'goto' } _)'; \\ \text{gen('goto' } _'): \end{array} \}$

$B \rightarrow E_1 \text{ rel } E_2 \quad \left| \begin{array}{l} B.\text{code} = E_1.\text{code} \parallel E_2.\text{code} \\ \parallel \text{gen('if' } E_1.\text{addr rel.op } E_2.\text{addr 'goto' } B.\text{true}) \\ \parallel \text{gen('goto' } B.\text{false}) \end{array} \right.$

- 3) $B \rightarrow !B_1$ { $B.trueList = B_1.falseList;$
 $B.falseList = B_1.trueList;$ }
- 4) $B \rightarrow (B_1)$ { $B.trueList = B_1.trueList;$
 $B.falseList = B_1.falseList;$ }

$$B \rightarrow !B_1 \quad \left| \begin{array}{l} \cdot B_1.true = B.false \\ \cdot B_1.false = B.true \\ \cdot B.code = B_1.code \end{array} \right.$$

2) $B \rightarrow B_1 \ \&\& \ M \ B_2 \quad \{ \begin{array}{l} backpatch(B_1.trueList, M.instr); \\ B.trueList = B_2.trueList; \\ B.falseList = merge(B_1.falseList, B_2.falseList); \end{array} \}$

8) $M \rightarrow \epsilon \quad \{ \begin{array}{l} M.instr = nextinstr; \end{array} \}$

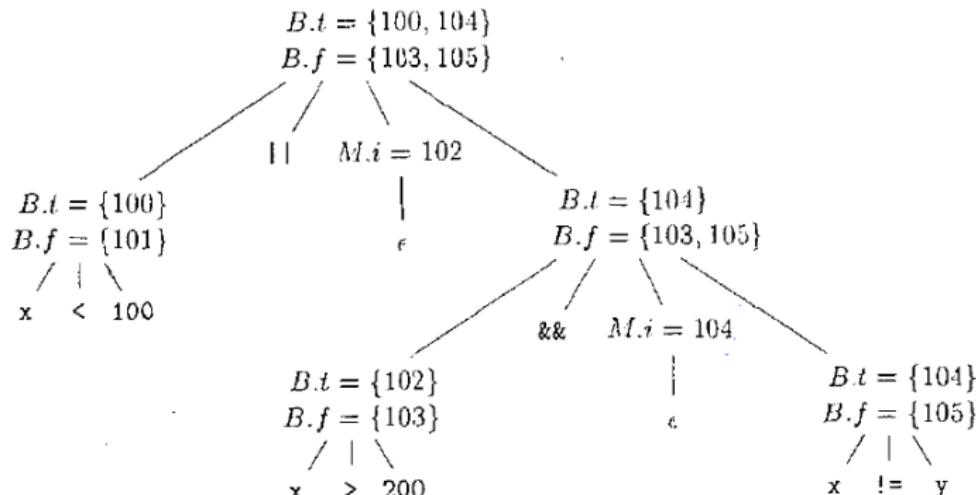
$B \rightarrow B_1 \ \&\& \ B_2 \quad \left| \begin{array}{l} B_1.true = newlabel() \\ B_1.false = B.false \\ B_2.true = B.true \\ B_2.false = B.false \\ B.code = B_1.code || label(B_1.true) || B_2.code \end{array} \right.$

1) $B \rightarrow B_1 \parallel M B_2 \quad \{ \boxed{\text{backpatch}(B_1.\textit{falselist}, M.\textit{instr});}$
 $B.\textit{truelist} = \text{merge}(B_1.\textit{truelist}, B_2.\textit{truelist});$
 $B.\textit{falselist} = B_2.\textit{falselist}; \}$

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

$B \rightarrow B_1 \parallel B_2 \quad \left| \begin{array}{l} B_1.\textit{true} = B.\textit{true} \\ \boxed{B_1.\textit{false} = \text{newlabel}()} \\ B_2.\textit{true} = B.\textit{true} \\ B_2.\textit{false} = B.\textit{false} \\ B.\textit{code} = B_1.\textit{code} \parallel \boxed{\text{label}(B_1.\textit{false})} \parallel B_2.\textit{code} \end{array} \right.$

- 1) $B \rightarrow B_1 \parallel M B_2$ { 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) $B \rightarrow E_1 \text{ rel } E_2$ { $B.truelist = makelist(nextinstr)$;
 $B.falselist = makelist(nextinstr + 1)$;
 $gen('if' E_1.addr \text{ rel.op } E_2.addr 'goto _')$;
 $gen('goto _')$ }



```
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) $S \rightarrow \text{if}(B) M S_1$

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

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

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

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

$S \rightarrow \text{if}(B) S_1 \quad \left| \begin{array}{l} B.\text{true} = \text{newlabel}() \\ B.\text{false} = S_1.\text{next} = S.\text{next} \\ S.\text{code} = B.\text{code} \parallel \text{label}(B.\text{true}) \parallel S_1.\text{code} \end{array} \right.$

$$S \rightarrow \text{if} (B) S_1 \text{ else } S_2 \quad \left| \begin{array}{l} B.\text{true} = \text{newlabel}() \\ B.\text{false} = \text{newlabel}() \\ S_1.\text{next} = S_2.\text{next} = S.\text{next} \\ S.\text{code} = B.\text{code} \\ \quad \parallel \text{label}(B.\text{true}) \parallel S_1.\text{code} \\ \quad \parallel \text{gen('goto' } S.\text{next}) \\ \quad \parallel \text{label}(B.\text{false}) \parallel S_2.\text{code} \end{array} \right.$$

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

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

$$S \rightarrow \text{if} (B) S_1 \text{ else } S_2 \quad \left| \begin{array}{l} B.\text{true} = \text{newlabel}() \\ B.\text{false} = \text{newlabel}() \\ S_1.\text{next} = S_2.\text{next} = S.\text{next} \\ S.\text{code} = B.\text{code} \\ \quad \parallel \text{label}(B.\text{true}) \parallel S_1.\text{code} \\ \quad \parallel \text{gen('goto' } S.\text{next}) \\ \quad \parallel \text{label}(B.\text{false}) \parallel S_2.\text{code} \end{array} \right.$$

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

```
{ backpatch(S1.nextlist, M1.instr);
  backpatch(B.truelist, M2.instr);
  S.nextlist = B.falselist;
  gen('goto' M1.instr); }
```

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

$S \rightarrow \text{while } (B) S_1$

```
begin = newlabel()
B.true = newlabel()
B.false = S.next
S1.next = begin
S.code = label(begin) || B.code
          || label(B.true) || S1.code
          || gen('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) $S \rightarrow A ;$ $\{ S.nextlist = \text{null}; \}$

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

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  $\geq$  60 then
3:     while true do
4:       print “WanSui”
5:     else
6:       print “Sad”
```

- 2) $S \rightarrow \text{if } (B) M_1 S_1 N \text{ else } M_2 S_2$
- $$\{ \text{backpatch}(B.\text{truelist}, M_1.\text{instr});$$
- $$\text{backpatch}(B.\text{falseclist}, M_2.\text{instr});$$
- $$temp = \text{merge}(S_1.\text{nextlist}, N.\text{nextlist});$$
- $$S.\text{nextlist} = \text{merge}(temp, S_2.\text{nextlist}); \}$$
- 3) $S \rightarrow \text{while } M_1 (B) M_2 S_1$
- $$\{ \text{backpatch}(S_1.\text{nextlist}, M_1.\text{instr});$$
- $$\text{backpatch}(B.\text{truelist}, M_2.\text{instr});$$
- $$S.\text{nextlist} = B.\text{falseclist};$$
- $$\text{gen('goto' } M_1.\text{instr}); \}$$
- 6) $M \rightarrow \epsilon$ $\{ M.\text{instr} = \text{nextinstr}; \}$
- 7) $N \rightarrow \epsilon$ $\{ N.\text{nextlist} = \text{makelist(nextinstr)};$

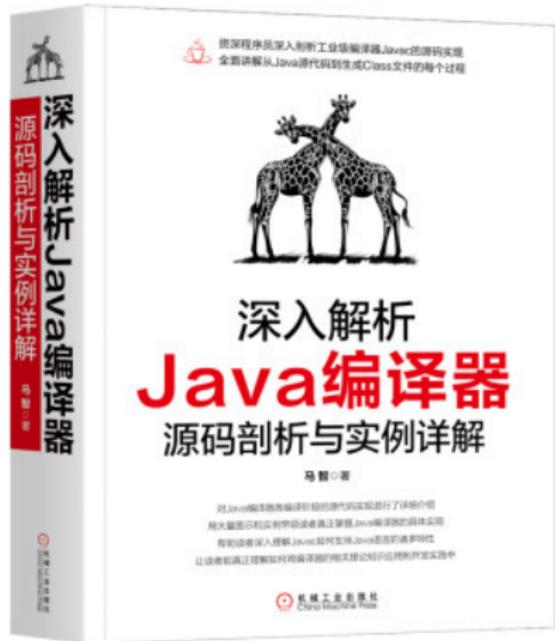
$$\text{gen('goto' } __); \}$$
- 6) $B \rightarrow \text{true}$ $\{ B.\text{truelist} = \text{makelist(nextinstr)};$

$$\text{gen('goto' } __); \}$$

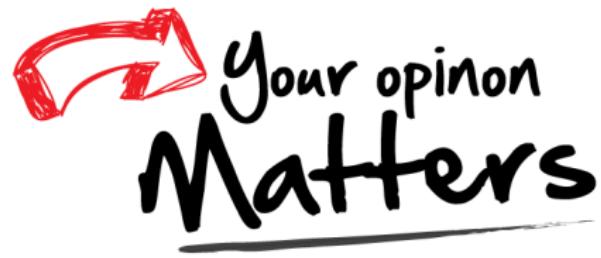
为左部非终结符 B 计算综合属性 $B.trueList$ 与 $B.falseList$

为左部非终结符 S/L 计算综合属性 $S/L.nextList$

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