

中间代码生成

魏恒峰

hfwei@nju.edu.cn

2020 年 12 月 24 日



控制流语句与布尔表达式的中间代码翻译

$$S \rightarrow \text{if } (B) \ S_1$$
$$S \rightarrow \text{if } (B) \ S_1 \ \text{else } S_2$$
$$S \rightarrow \text{while } (B) \ S_1$$

控制流语句与布尔表达式的中间代码翻译



产生式	语义规则
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \parallel label(S.next)$
$S \rightarrow assign$	$S.code = assign.code$
$S \rightarrow if (B) S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \parallel label(B.true) \parallel S_1.code$
$S \rightarrow if (B) S_1 else S_2$	$B.true = newlabel()$ $B.false = newlabel()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ $\parallel label(B.true) \parallel S_1.code$ $\parallel gen('goto' S.next)$ $\parallel label(B.false) \parallel S_2.code$
$S \rightarrow while (B) S_1$	$begin = newlabel()$ $B.true = newlabel()$ $B.false = S.next$ $S_1.next = begin$ $S.code = label(begin) \parallel B.code$ $\parallel label(B.true) \parallel S_1.code$ $\parallel gen('goto' begin)$
$S \rightarrow S_1 S_2$	$S_1.next = newlabel()$ $S_2.next = S.next$ $S.code = S_1.code \parallel label(S_1.next) \parallel S_2.code$

继承属性 $S.next$: S 的下一条指令

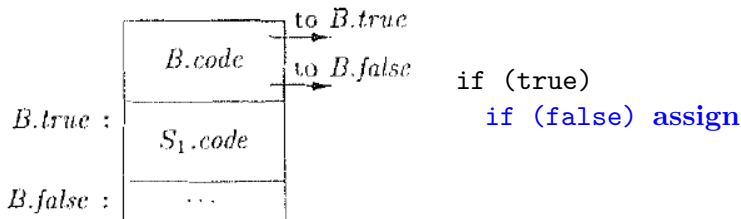
$$P \rightarrow S \quad \left| \begin{array}{l} S.next = newlabel() \\ P.code = S.code || label(S.next) \end{array} \right.$$

$S.next$ 为语句 S 指明了“跳出” S 的目标

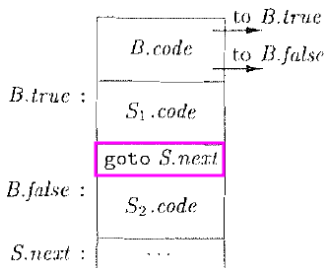
$$S \rightarrow \text{assign} \quad | \quad S.\text{code} = \text{assign}.\text{code}$$

代表了表达式的翻译, 包括数组引用

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

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


$S \rightarrow \text{if} (B) S_1 \text{ else } S_2$	$B.true = \text{newlabel}()$ $B.false = \text{newlabel}()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ $ \text{label}(B.true) S_1.code$ $ \text{gen}('goto' S.next)$ $ \text{label}(B.false) S_2.code$
---	---

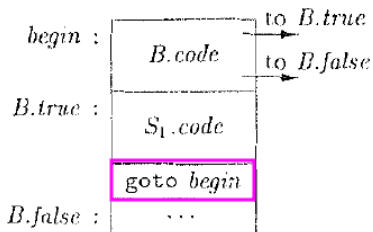


```

if (true)
    if (true) assign else assign
else
    assign
  
```


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

```
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)
```



```
while (true)
    if (false) assign else assign
```

$S \rightarrow S_1 S_2$

$\begin{cases} S_1.next = newlabel() \\ S_2.next = S.next \\ S.code = S_1.code || label(S_1.next) || S_2.code \end{cases}$

if (true) assign else assign assign

产生式	语义规则
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code label(S.next)$
$S \rightarrow assign$	$S.code = assign.code$
$S \rightarrow if (B) S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code label(B.true) S_1.code$
$S \rightarrow if (B) S_1 else S_2$	$B.true = newlabel()$ $B.false = newlabel()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ $ label(B.true) S_1.code$ $ gen('goto' S.next)$ $ label(B.false) S_2.code$
$S \rightarrow while (B) S_1$	$begin = newlabel()$ $B.true = newlabel()$ $B.false = S.next$ $S_1.next = begin$ $S.code = label(begin) B.code$ $ label(B.true) S_1.code$ $ gen('goto' begin)$
$S \rightarrow S_1 S_2$	$S_1.next = newlabel()$ $S_2.next = S.next$ $S.code = S_1.code label(S_1.next) S_2.code$

布尔表达式的中间代码翻译

产生式	语义规则
$B \rightarrow B_1 \ \ B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \ \ label(B_1.false) \ \ B_2.code$
$B \rightarrow B_1 \ \&\& \ B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \ \ label(B_1.true) \ \ B_2.code$
$B \rightarrow ! B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$
$B \rightarrow E_1 \ rel \ E_2$	$B.code = E_1.code \ \ E_2.code$ $\quad \ \ gen('if' \ E_1.addr \ rel \ op \ E_2.addr \ 'goto' \ B.true)$ $\quad \ \ gen('goto' \ B.false)$
$B \rightarrow true$	$B.code = gen('goto' \ B.true)$
$B \rightarrow false$	$B.code = gen('goto' \ B.false)$

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

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

if (true) assign

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

if (false) assign

$B \rightarrow ! B_1$

$\left\{ \begin{array}{l} B_1.true = B.false \\ B_1.false = B.true \\ B.code = B_1.code \end{array} \right.$

if (!true) assign

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

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

if (!false) assign

短路求值

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

if (true || false) assign

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

if (false || true) assign

短路求值

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

if (true && false) assign

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

if (false && true) assign

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

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

```
        if x < 100 goto L2  
        goto L3  
L3:    if x > 200 goto L4  
        goto L1  
L4:    if x != y goto L2  
        goto L1  
L2:    x = 0  
L1:
```

布尔表达式的作用: 布尔值 vs. 控制流跳转

$$\begin{aligned} S &\rightarrow \boxed{\text{id} = E;} \mid \text{if } (E) \ S \mid \text{while } (E) \ S \mid S \ S \\ E &\rightarrow E \parallel E \mid E \&\& E \mid E \text{ rel } E \mid E + E \mid (E) \mid \text{id} \mid \text{true} \mid \text{false} \end{aligned}$$

函数 $\text{jump}(t, f)$: 生成控制流代码

函数 $\text{rvalue}()$: 生成计算布尔值的代码, 并将结果存储在临时变量中

产生式	语义规则
$S \rightarrow id = E ;$	$S.code = E.code \parallel$ $gen(top.get(id.lexeme) '=' E.addr)$
$E \rightarrow E_1 + E_2$	$E.addr = new Temp()$ $E.code = E_1.code \parallel E_2.code \parallel$ $gen(E.addr '=' E_1.addr '+' E_2.addr)$
$ - E_1$	$E.addr = new Temp()$ $E.code = E_1.code \parallel$ $gen(E.addr '=' 'minus' E_1.addr)$
$ (E_1)$	$E.addr = E_1.addr$ $E.code = E_1.code$
$ id$	$E.addr = top.get(id.lexeme)$ 符号表条目 $E.code = ''$

$$E \rightarrow E_1 \&\& E_2$$

为 E 生成跳转代码, 在真假出口处将 true 或 false 存储到临时变量

`x = a < b && c < d`

```
    ifFalse a < b goto L1
    ifFalse c < d goto L1
    t = true
    goto L2
L1: t = false
L2: x = t
```

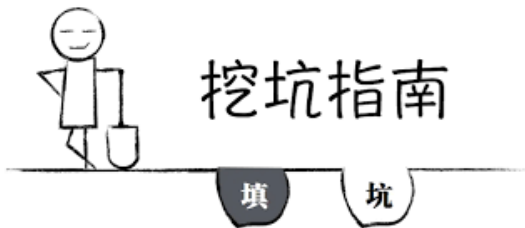
$$S \rightarrow \text{if} (B) S_1 \quad \left| \begin{array}{l} B.\text{true} = \text{newlabel}() \\ B.\text{false} = S_1.\text{next} = S.\text{next} \\ S.\text{code} = B.\text{code} || \text{label}(B.\text{true}) || S_1.\text{code} \end{array} \right.$$

B 还不知道 *S.next* 的指令地址, 如何跳转?

再扫描一遍中间代码, 将标号替换成指令 (相对) 地址

可否在生成中间代码的时候就填入指令地址?

回填 (Backpatch) 技术



子节点挖坑、祖先节点填坑

针对布尔表达式的回填技术

- | | | |
|----|--------------------------------------|---|
| 1) | $B \rightarrow B_1 \parallel M B_2$ | { <i>backpatch</i> (<i>B</i> ₁ . <i>false</i> list, <i>M.instr</i>);
<i>B.true</i> list = <i>merge</i> (<i>B</i> ₁ . <i>true</i> list, <i>B</i> ₂ . <i>true</i> list);
<i>B.false</i> list = <i>B</i> ₂ . <i>false</i> list; } |
| 2) | $B \rightarrow B_1 \&\& M B_2$ | { <i>backpatch</i> (<i>B</i> ₁ . <i>true</i> list, <i>M.instr</i>);
<i>B.true</i> list = <i>B</i> ₂ . <i>true</i> list;
<i>B.false</i> list = <i>merge</i> (<i>B</i> ₁ . <i>false</i> list, <i>B</i> ₂ . <i>false</i> list); } |
| 3) | $B \rightarrow ! B_1$ | { <i>B.true</i> list = <i>B</i> ₁ . <i>false</i> list;
<i>B.false</i> list = <i>B</i> ₁ . <i>true</i> list; } |
| 4) | $B \rightarrow (B_1)$ | { <i>B.true</i> list = <i>B</i> ₁ . <i>true</i> list;
<i>B.false</i> list = <i>B</i> ₁ . <i>false</i> list; } |
| 5) | $B \rightarrow E_1 \text{ rel } E_2$ | { <i>B.true</i> list = <i>makelist</i> (<i>nextinstr</i>);
<i>B.false</i> list = <i>makelist</i> (<i>nextinstr</i> + 1);
<i>gen</i> ('if' <i>E</i> ₁ . <i>addr</i> <i>rel.op</i> <i>E</i> ₂ . <i>addr</i> 'goto -');
<i>gen</i> ('goto -'); } |
| 6) | $B \rightarrow \text{true}$ | { <i>B.true</i> list = <i>makelist</i> (<i>nextinstr</i>);
<i>gen</i> ('goto -'); } |
| 7) | $B \rightarrow \text{false}$ | { <i>B.false</i> list = <i>makelist</i> (<i>nextinstr</i>);
<i>gen</i> ('goto -'); } |
| 8) | $M \rightarrow \epsilon$ | { <i>M.instr</i> = <i>nextinstr</i> ; } |

综合属性 $B.truelist$ 保存 需要跳转到 $B.true$ 的指令地址

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

综合属性 $B.falselist$ 保存 需要跳转到 $B.false$ 的指令地址

$B \rightarrow true$	$B.code = gen('goto' B.true)$
$B \rightarrow false$	$B.code = gen('goto' B.false)$

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

$B \rightarrow E_1 \text{ rel } E_2$ { $B.code = E_1.code || E_2.code$
 || $gen('if' E_1.addr \text{ rel.op } E_2.addr 'goto' B.true$
 || $gen('goto' B.false)$ }

$$3) \quad B \rightarrow ! B_1$$

$$\{ \boxed{B.true\text{list}} = B_1.false\text{list}; \\ \boxed{B.false\text{list}} = B_1.true\text{list}; \}$$

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

$$\{ \boxed{B.true\text{list}} = B_1.true\text{list}; \\ \boxed{B.false\text{list}} = B_1.false\text{list}; \}$$

$$B \rightarrow ! B_1$$

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

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

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

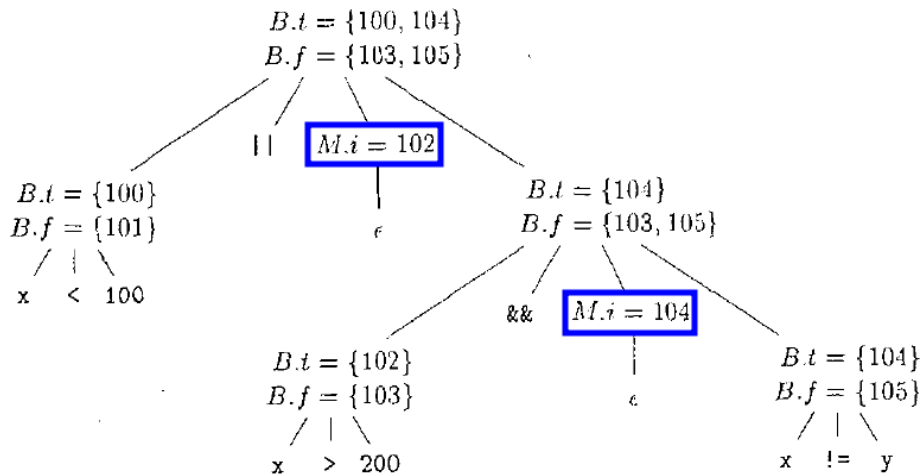
$B \rightarrow B_1 \ \&\& \ B_2 \quad \begin{cases} B_1.\text{true} = \text{newlabel}() \\ B_1.\text{false} = B.\text{false} \\ B_2.\text{true} = B.\text{true} \\ B_2.\text{false} = B.\text{false} \\ B.\text{code} = B_1.\text{code} \ || \ \text{label}(B_1.\text{true}) \ || \ B_2.\text{code} \end{cases}$

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

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

$B \rightarrow B_1 \parallel B_2$ { $B_1.true = B.true$
 $B_1.false = newlabel()$
 $B_2.true = B.true$
 $B_2.false = B.false$
 $B.code = B_1.code \parallel label(B_1.false) \parallel B_2.code$

$x < 100 \ || \ x > 200 \ \&\& \ x \neq y$



```
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

$$S \rightarrow \text{if}(B) S \mid \text{if}(B) S \text{ else } S \mid \text{while}(B) S \mid \boxed{\{L\}} \mid A ;$$
$$L \rightarrow L S \mid S$$

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

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

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

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

```

{ backpatch(B.truelist, M1.instr);
  backpatch(B.falselist, M2.instr);
  temp = merge(S1.nextlist, N.nextlist);
  S.nextlist = merge(temp, S2.nextlist); }

```

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

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

$S \rightarrow \text{if}(B) S_1 \text{ else } S_2$	$B.true = newlabel()$ $B.false = newlabel()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ <div style="text-align: right;"> $\text{label}(B.true) S_1.code$ $gen('goto' S.next)$ $\text{label}(B.false) S_2.code$ </div>
--	---

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

```

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

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

$S \rightarrow \text{while } (B) S_1$	<pre> begin = newlabel() B.true = newlabel() B.false = S.next $S_1.next = begin$ S.code = label(begin) B.code label(B.true) $S_1.code$ gen('goto' begin)</pre>
---	--

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

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

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

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

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

Thank
You!



Office 926

hfwei@nju.edu.cn