中间代码生成

魏恒峰

hfwei@nju.edu.cn

2020 年 12 月 24 日

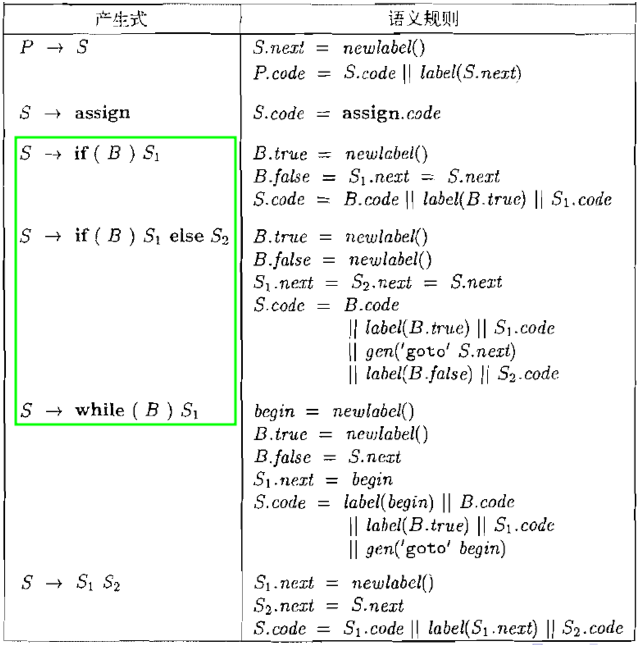
控制流语句与布尔表达式的中间代码翻译

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

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

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

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
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$
$S \rightarrow \textbf{assign}$	$S.code = \textbf{assign}.code$
$S \rightarrow \textbf{if} (B) S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$S \rightarrow \textbf{if} (B) S_1 \textbf{ else } S_2$	$B.true = newlabel()$ $B.false = newlabel()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('\texttt{goto}'\ S.next)$ $\mid\mid label(B.false) \mid\mid S_2.code$
$S \rightarrow \textbf{while} (B) S_1$	$begin = newlabel()$ $B.true = newlabel()$ $B.false = S.next$ $S_1.next = begin$ $S.code = label(begin) \mid\mid B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('\texttt{goto}'\ begin)$
$S \rightarrow S_1\ S_2$	$S_1.next = newlabel()$ $S_2.next = S.next$ $S.code = S_1.code \mid\mid label(S_1.next) \mid\mid S_2.code$

继承属性 $S.next$: S 的下一条指令

$$P \rightarrow S \qquad \left| \begin{array}{lcl} \boxed{S.next} & = & newlabel() \\ P.code & = & S.code \mid\mid \boxed{label(S.next)} \end{array} \right.$$

$S.next$ 为语句 S 指明了 “跳出” S 的目标

$$S \rightarrow \textbf{assign} \qquad \mid \; S.code = \textbf{assign}.code$$

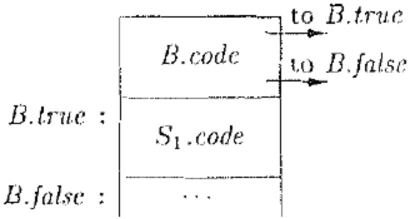
代表了表达式的翻译, 包括数组引用

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

$$\begin{array}{lcl} B.true & = & newlabel() \\ B.false & = & \boxed{S_1.next} = S.next \\ S.code & = & B.code \mid\mid label(B.true) \mid\mid S_1.code \end{array}$$

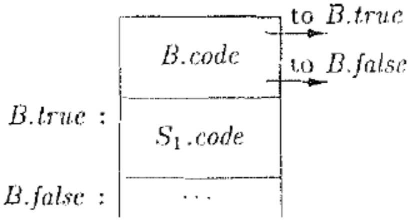
$S \rightarrow \textbf{if}\ (\ B\ )\ S_1$

$$\begin{array}{|lcl} B.true & = & newlabel() \\ B.false & = & \boxed{S_1.next} = S.next \\ S.code & = & B.code \,||\, label(B.true) \,||\, S_1.code \end{array}$$

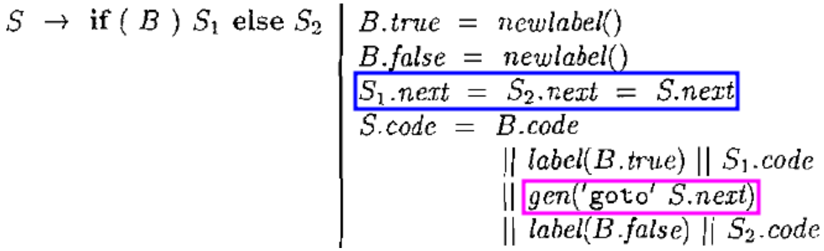


$S \rightarrow \textbf{if} (B) S_1$

$$\begin{array}{|lcl} B.true & = & newlabel() \\ B.false & = & \boxed{S_1.next} = S.next \\ S.code & = & B.code \; || \; label(B.true) \; || \; S_1.code \end{array}$$

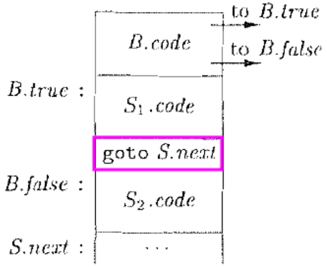


```
if (true)
  if (false) assign
```

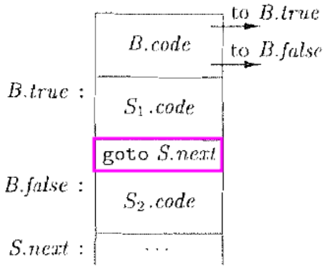


$$
\begin{array}{l|l}
S \rightarrow \textbf{if} \; (\; B \;) \; S_1 \; \textbf{else} \; S_2 &
\begin{aligned}
& B.true = newlabel() \\
& B.false = newlabel() \\
& \boxed{S_1.next = S_2.next = S.next} \\
& S.code = B.code \\
& \qquad \qquad \| \; label(B.true) \; \| \; S_1.code \\
& \qquad \qquad \| \; \boxed{gen('\texttt{goto}' \; S.next)} \\
& \qquad \qquad \| \; label(B.false) \; \| \; S_2.code
\end{aligned}
\end{array}
$$

$S \rightarrow \textbf{if} (B) S_1 \textbf{ else } S_2$	$B.true = newlabel()$ $B.false = newlabel()$ $\boxed{S_1.next = S_2.next = S.next}$ $S.code = B.code$ $\quad \| label(B.true) \| S_1.code$ $\quad \| \boxed{gen('\texttt{goto}'\ S.next)}$ $\quad \| label(B.false) \| S_2.code$



$S \rightarrow \textbf{if} (B) S_1 \textbf{ else } S_2$	$B.true = newlabel()$ $B.false = newlabel()$ $\boxed{S_1.next = S_2.next = S.next}$ $S.code = B.code$ $\quad \Vert\ label(B.true) \Vert S_1.code$ $\quad \Vert\ \boxed{gen(\texttt{'goto'}\ S.next)}$ $\quad \Vert\ label(B.false) \Vert S_2.code$

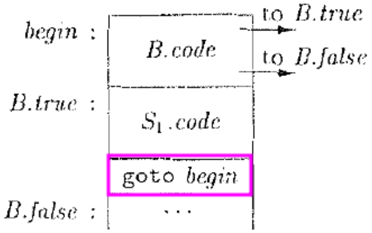


```
if (true)
  if (true) assign else assign
else
  assign
```

$$
\begin{array}{l|l}
S \rightarrow \textbf{while}\ (\ B\ )\ S_1 & begin = newlabel() \\
 & B.true = newlabel() \\
 & B.false = S.next \\
 & \boxed{S_1.next} = begin \\
 & S.code = label(begin) \mid\mid B.code \\
 & \qquad\qquad \mid\mid label(B.true) \mid\mid S_1.code \\
 & \qquad\qquad \mid\mid \boxed{gen('\text{goto}'\ begin)}
\end{array}
$$

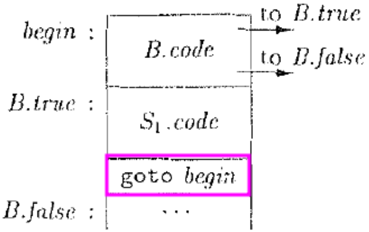
$S \rightarrow \textbf{while} \ (\ B \ ) \ S_1$

$$
\begin{aligned}
begin &= newlabel() \\
B.true &= newlabel() \\
B.false &= S.next \\
\boxed{S_1.next} &= begin \\
S.code &= label(begin) \ || \ B.code \\
&\quad || \ label(B.true) \ || \ S_1.code \\
&\quad || \ \boxed{gen(\text{'goto'} \ begin)}
\end{aligned}
$$



$S \rightarrow$ **while** (B) S_1

$$
\begin{aligned}
begin &= newlabel() \\
B.true &= newlabel() \\
B.false &= S.next \\
\boxed{S_1.next} &= begin \\
S.code &= label(begin) \; || \; B.code \\
&\quad || \; label(B.true) \; || \; S_1.code \\
&\quad || \; \boxed{gen(\texttt{'goto'}\ begin)}
\end{aligned}
$$



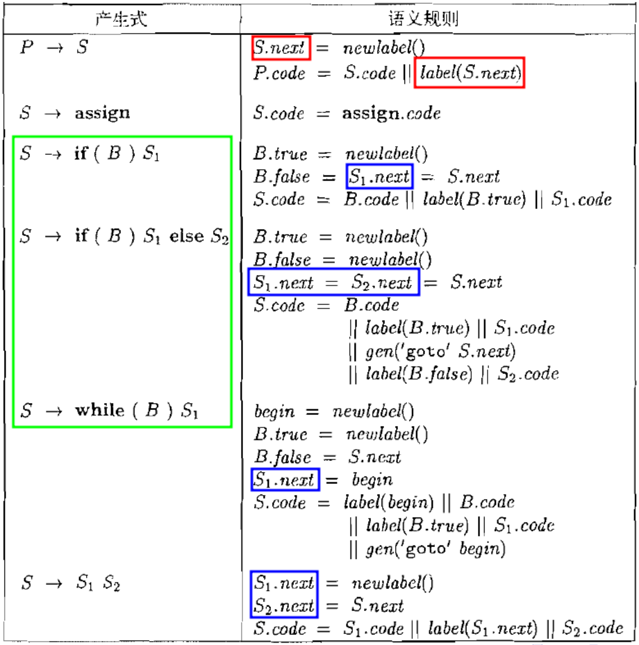
```
while (true)
  if (false) assign else assign
```

$S \rightarrow S_1 \; S_2$

$$\begin{aligned} \boxed{\begin{aligned} S_1.next \\ S_2.next \end{aligned}} \; &\begin{aligned} &= newlabel() \\ &= S.next \end{aligned} \\ S.code &= S_1.code \;||\; label(S_1.next) \;||\; S_2.code \end{aligned}$$

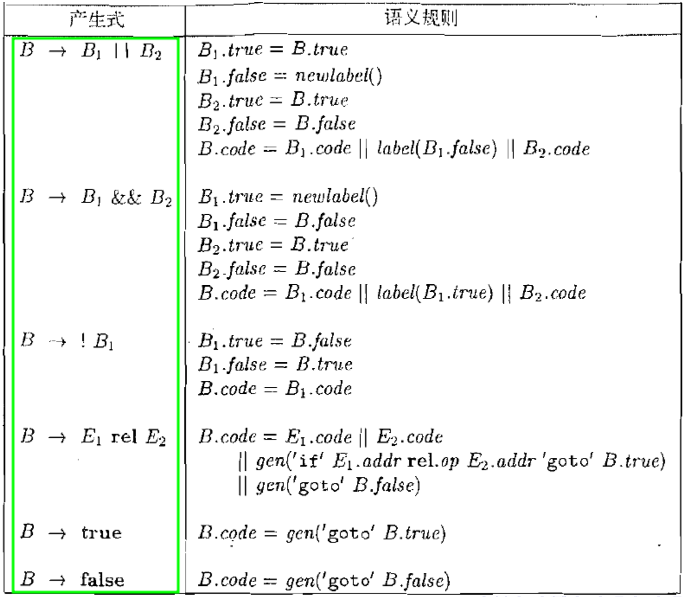
$$S \rightarrow S_1 \; S_2 \qquad \begin{array}{|lcl} \boxed{\begin{array}{l} S_1.next \\ S_2.next \end{array}} & \begin{array}{l} = \\ = \end{array} & \begin{array}{l} newlabel() \\ S.next \end{array} \\ S.code & = & S_1.code \,||\, label(S_1.next) \,||\, S_2.code \end{array}$$

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



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

布尔表达式的中间代码翻译



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

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

$B \to \textbf{true}$ | $B.code = gen('goto' \boxed{B.true})$

$B \to \textbf{false}$ | $B.code = gen('goto' \boxed{B.false})$

if (true) **assign**

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

$$\begin{aligned} B.true &= newlabel() \\ B.false &= \boxed{S_1.next} = S.next \\ S.code &= B.code \,||\, label(B.true) \,||\, S_1.code \end{aligned}$$

if (false) **assign**

$$\begin{array}{l|l} B \to\ !\, B_1 & \boxed{\begin{array}{l} B_1.\mathit{true} \\ B_1.\mathit{false} \end{array}} \begin{array}{l} = B.\mathit{false} \\ = B.\mathit{true} \end{array} \\ & B.\mathit{code} = B_1.\mathit{code} \end{array}$$

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

`if (!true)` **assign**

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

`if (!false)` **assign**

短路求值

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

短路求值

$$B \to B_1 \;||\; B_2 \qquad \begin{array}{|l} \boxed{B_1.true} = B.true \\ B_1.false = newlabel() \\ \boxed{B_2.true} = B.true \\ B_2.false = B.false \\ B.code = B_1.code \;||\; label(B_1.false) \;||\; B_2.code \end{array}$$

`if (true || false)` **assign**

$$S \to \textbf{if}\,(\,B\,)\,S_1 \qquad \begin{array}{|l} B.true = newlabel() \\ B.false = \boxed{S_1.next} = S.next \\ S.code = B.code \;||\; label(B.true) \;||\; S_1.code \end{array}$$

`if (false || true)` **assign**

短路求值

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

短路求值

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

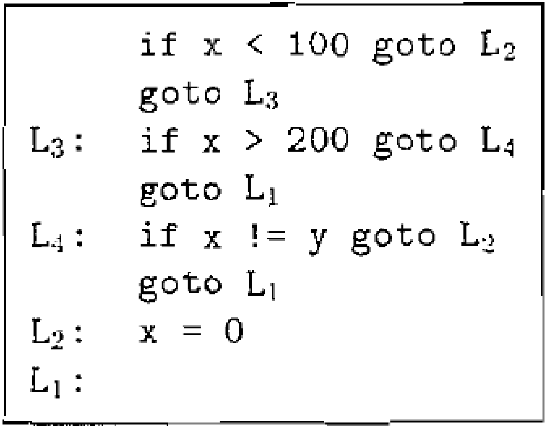
`if (true && false) assign`

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

`if (false && true) assign`

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

```
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

$S \rightarrow \boxed{\textbf{id} = E;} \mid \textbf{if}\ (E)\ S \mid \textbf{while}\ (E)\ S \mid S\ S$

$E \rightarrow E \parallel E \mid E \,\&\&\, E \mid E\ \textbf{rel}\ E \mid E + E \mid (E) \mid \textbf{id} \mid \textbf{true} \mid \textbf{false}$

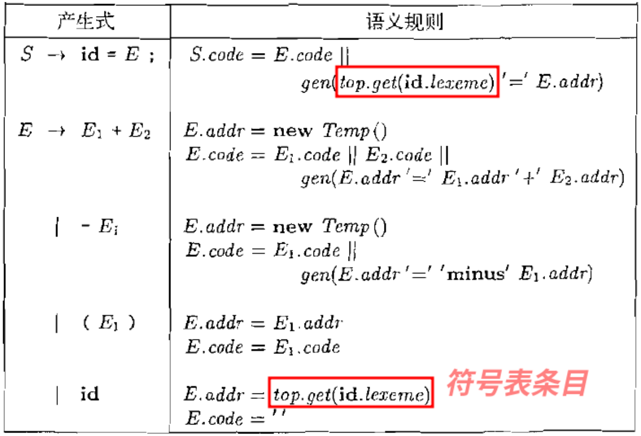
布尔表达式的作用: **布尔值** *vs.* **控制流跳转**

$$S \rightarrow \boxed{\textbf{id} = E;} \mid \textbf{if}\ (E)\ S \mid \textbf{while}\ (E)\ S \mid S\ S$$

$$E \rightarrow E \parallel E \mid E \&\& E \mid E\ \textbf{rel}\ E \mid E + E \mid (E) \mid \textbf{id} \mid \textbf{true} \mid \textbf{false}$$

函数 $jump(t, f)$: 生成控制流代码

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



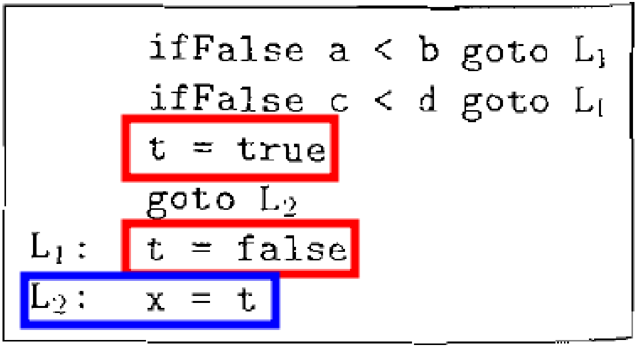
产生式	语义规则
$S \to \mathbf{id} = E\ ;$	$S.code = E.code \Vert gen(\boxed{top.get(\mathbf{id}.lexeme)}\ '='\ E.addr)$
$E \to E_1 + E_2$	$E.addr = \mathbf{new}\ Temp()$ $E.code = E_1.code \Vert E_2.code \Vert gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr)$
$\mid\ - E_1$	$E.addr = \mathbf{new}\ Temp()$ $E.code = E_1.code \Vert gen(E.addr\ '='\ '\mathbf{minus}'\ E_1.addr)$
$\mid\ (E_1)$	$E.addr = E_1.addr$ $E.code = E_1.code$
$\mid\ \mathbf{id}$	$E.addr = \boxed{top.get(\mathbf{id}.lexeme)}$ $E.code = ''$

符号表条目

$$E \to E_1 \&\& E_2$$

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
```

$$S \rightarrow \textbf{if} (B) S_1 \qquad \begin{array}{|l} \boxed{B.true} = newlabel() \\ \boxed{B.false = S_1.next = S.next} \\ S.code = B.code \,||\, label(B.true) \,||\, S_1.code \end{array}$$

B **还不知道** $S.next$ **的指令地址，如何跳转？**

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

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

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

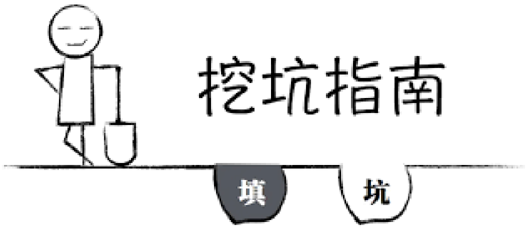
B 还不知道 $S.next$ 的指令地址，如何跳转？

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？

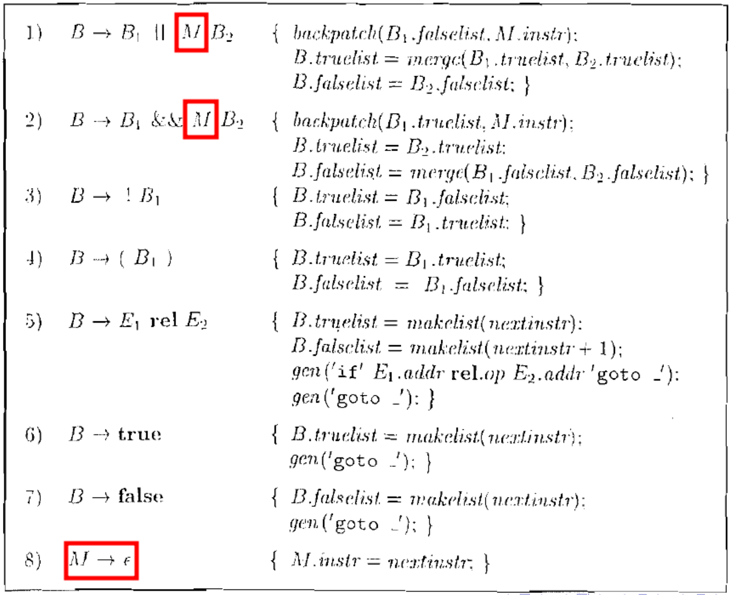
回填（Backpatch）技术

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

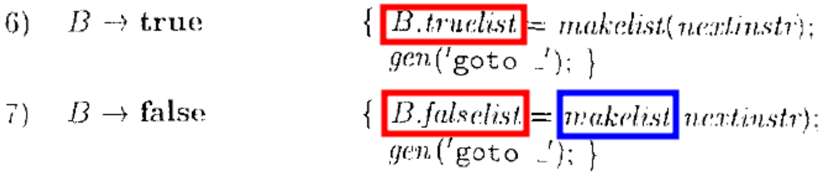
5)	$B \rightarrow E_1$ **rel** E_2	{ B.*truelist* = *makelist*(*nextinstr*);
		 B.*falselist* = *makelist*(*nextinstr* + 1);
		 gen('`if`' E_1.*addr* **rel**.*op* E_2.*addr* '`goto _`');
		 gen('`goto _`'); }

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

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

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

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

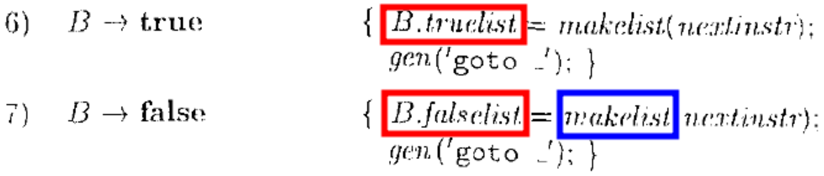


6) $B \rightarrow$ **true** { $\boxed{B.truelist}$ = $makelist(nextinstr)$; gen('goto _'); }

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

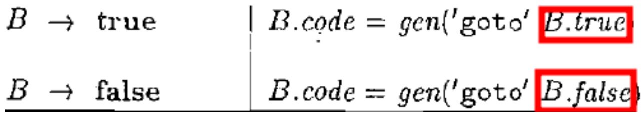
综合属性 $B.falselist$ 保存 需要跳转到 $B.false$ 的指令地址

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



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

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



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

5) $B \rightarrow E_1 \ \mathbf{rel} \ E_2$ $\{$ $\boxed{B.truelist} = makelist(nextinstr);$

$\boxed{B.falselist} = makelist(nextinstr + 1);$

$gen(\texttt{'if'}\ E_1.addr\ \mathbf{rel}.op\ E_2.addr\ \texttt{'goto \_'});$

$gen(\texttt{'goto \_'});\ \}$

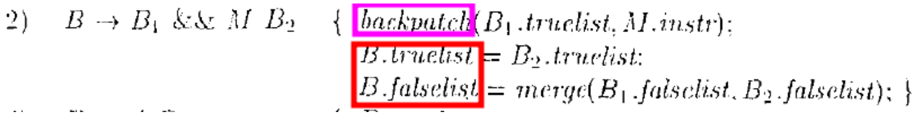
$$
\begin{array}{lll|l}
B & \rightarrow & E_1 \ \mathbf{rel} \ E_2 & B.code = E_1.code \ || \ E_2.code \\
 & & & \qquad || \ gen(\texttt{'if'}\ E_1.addr\ \mathbf{rel}.op\ E_2.addr\ \texttt{'goto'}\ \boxed{B.true}) \\
 & & & \qquad || \ gen(\texttt{'goto'}\ \boxed{B.false})
\end{array}
$$

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

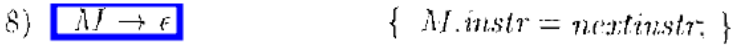
4) $B \rightarrow (\ B_1\ )$ $\qquad \{\ \boxed{B.truelist} = B_1.truelist;\ \boxed{B.falselist} = B_1.falselist;\ \}$

$B \rightarrow\ !\ B_1$

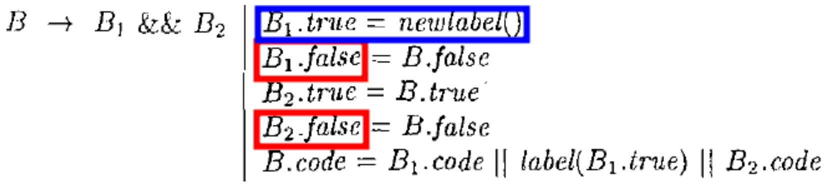
$$\begin{array}{|l} B_1.true = B.false \\ B_1.false = B.true \\ B.code = B_1.code \end{array}$$



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



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



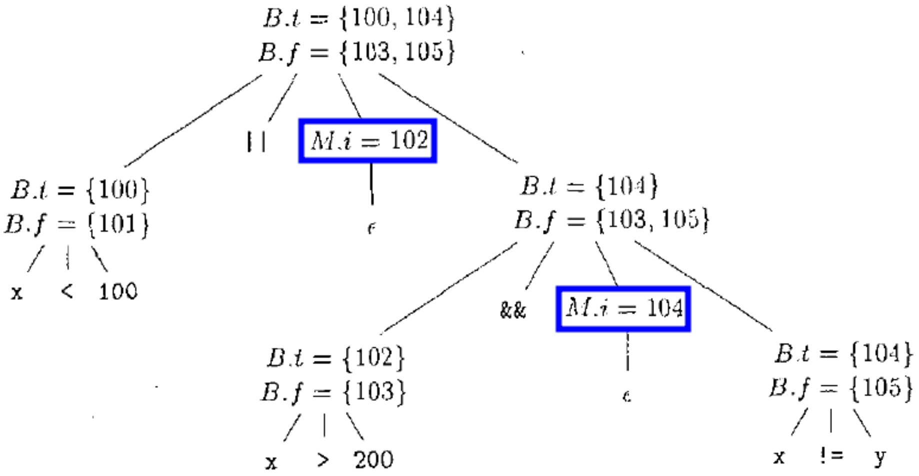
$$B \rightarrow B_1 \;\&\&\; B_2 \quad \begin{array}{|l} \boxed{B_1.true = newlabel()} \\ \boxed{B_1.false} = B.false \\ B_2.true = B.true \\ \boxed{B_2.false} = B.false \\ B.code = B_1.code \;||\; label(B_1.true) \;||\; B_2.code \end{array}$$

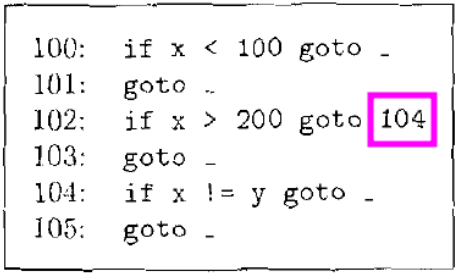
1) $B \to B_1 \;||\; M\; B_2$ $\{\ \boxed{backpatch}(B_1.falselist, M.instr);$
$\boxed{B.truelist} = merge(B_1.truelist, B_2.truelist);$
$\boxed{B.falselist} = B_2.falselist;\ \}$

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

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

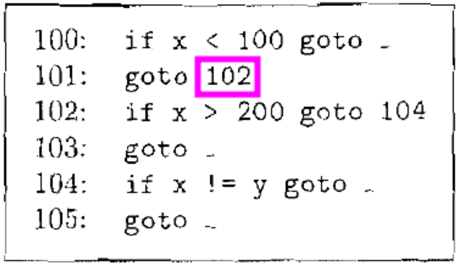
x < 100 || x > 200 && x != y


$B.t = \{100, 104\}$
$B.f = \{103, 105\}$
||
$M.i = 102$
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



```
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

$$
\begin{aligned}
S &\rightarrow \textbf{if}\,(\,B\,)\,S \mid \textbf{if}\,(\,B\,)\,S\;\textbf{else}\;S \mid \textbf{while}\,(\,B\,)\,S \mid \boxed{\{\,L\,\}} \mid A\;; \\
L &\rightarrow L\,S \mid S
\end{aligned}
$$

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

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

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

$$6)\quad M \to \epsilon \qquad\qquad \{\; M.instr \;=\; nextinstr;\;\}$$

$$S \to \textbf{if}\,(\,B\,)\; S_1 \qquad \begin{array}{|l} B.true \;=\; newlabel() \\ B.false \;=\; \boxed{S_1.next} \;=\; S.next \\ S.code \;=\; B.code \;||\; label(B.true) \;||\; S_1.code \end{array}$$

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

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

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

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

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

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

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

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

$$S \rightarrow \textbf{if} (B) S_1 \textbf{ else } S_2 \quad \left| \quad \begin{aligned}
& B.\textit{true} = \textit{newlabel}() \\
& B.\textit{false} = \textit{newlabel}() \\
& \boxed{S_1.\textit{next} = S_2.\textit{next} = S.\textit{next}} \\
& S.\textit{code} = B.\textit{code} \\
& \qquad \| \; \textit{label}(B.\textit{true}) \; \| \; S_1.\textit{code} \\
& \qquad \| \; \boxed{\textit{gen}('\texttt{goto}' \; S.\textit{next})} \\
& \qquad \| \; \textit{label}(B.\textit{false}) \; \| \; S_2.\textit{code}
\end{aligned}\right.$$

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

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

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

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

$$\begin{aligned} \{\ & \boxed{backpatch}(S_1.nextlist,\ M_1.instr); \\ & \boxed{backpatch}(B.truelist,\ M_2.instr); \\ & \boxed{S.nextlist} = B.falselist; \\ & gen(\text{'goto'}\ M_1.instr);\ \} \end{aligned}$$

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

$S \rightarrow$ **while** (B) S_1

$$\begin{aligned} begin &= newlabel() \\ B.true &= newlabel() \\ B.false &= S.next \\ \boxed{S_1.next} &= begin \\ S.code &= label(begin)\ ||\ B.code \\ &\quad ||\ label(B.true)\ ||\ S_1.code \\ &\quad ||\ \boxed{gen(\text{'goto'}\ begin)} \end{aligned}$$

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

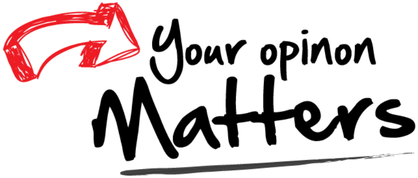
5) $S \to A\ ;$ $\{\boxed{S.nextlist = \text{null};}\}$ $\{\}$

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

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

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

Thank
You!



Office 926

hfwei@nju.edu.cn