四、中间代码生成
(12. 控制流语句的翻译 (Hard 模式))

魏恒峰

hfwei@nju.edu.cn

2024 年 05 月 15 日

各个非终结符的**分工**、**合作**明确易懂

只需要使用**综合属性**

分工　　合作

为布尔表达式 B 计算逻辑值 (假设保存在临时变量 `t1` 中)

if、**while** 等语句根据 B 的结果改变控制流

$$\textbf{if}\,(B)\,S_1$$

如何生成更短、更高效的代码?

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

```
1    br true or.true1 or.false2
2    or.false2:
3    t1 = OR true false
4    br or.end3:
5    or.true1:
6    t1 = true
7    or.end3:
8    br t1 b.true4 b.false5
9    b.true4:
10   a = b
11   b.false5:
```

如何生成更短、更高效的代码?

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

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

```
1  br true or.true1 or.false2
2  or.false2:
3  t1 = OR true false
4  br or.end3:
5  or.true1:
6  t1 = true
7  or.end3:
8  br t1 b.true4 b.false5
9  b.true4:
10 a = b
11 b.false5:
```

如何生成更短、更高效的代码?

```
1    while (true) {
2      if (false) {
3        a = b;
4      }
5    }
```

```
1    begin1:
2    br true b.true2 b.false3
3    b.true2:
4    br false b.true4 b.false5
5    b.true4:
6    a = b
7    b.false5:
8    br begin1
9    b.false3:
```

如何生成更短、更高效的代码?

```
1    while (true) {
2      if (false) {
3        a = b;
4      }
5    }
```

```
1    L2begin:
2    goto L3
3    L3:
4    goto L2begin
5    L4:
6    ASSIGN
7    goto L2begin
8    L1:
```

```
1    begin1:
2    br true b.true2 b.false3
3    b.true2:
4    br false b.true4 b.false5
5    b.true4:
6    a = b
7    b.false5:
8    br begin1
9    b.false3:
```

直接**用布尔表达式改变控制流**，无需计算最终逻辑值

从父节点获取**更具体的跳转目标**，缩短跳转路径

分工　　合作

父节点为子节点准备跳转指令的目标标签

子节点通过**继承属性**确定跳转目标

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

Control.g4

产生式
$P \rightarrow S$
$S \rightarrow$ **assign**
$S \rightarrow$ **if** (B) S_1
$S \rightarrow$ **if** (B) S_1 **else** S_2
$S \rightarrow$ **while** (B) S_1
$S \rightarrow S_1\ S_2$

产生式
$B \rightarrow B_1$ || B_2
$B \rightarrow B_1$ && B_2
$B \rightarrow$! B_1
$B \rightarrow E_1$ **rel** E_2
$B \rightarrow$ **true**
$B \rightarrow$ **false**

继承属性 $B.true$, $B.false$, $S.next$ 指明了控制流跳转目标

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

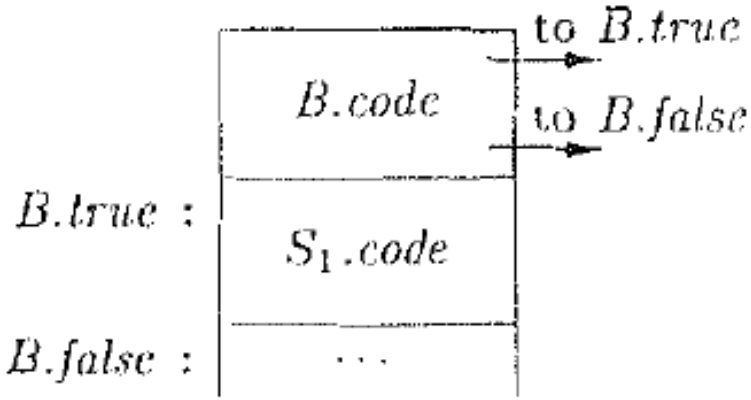
$$P \rightarrow S \qquad \left| \begin{array}{lcl} S.next & = & newlabel() \\ P.code & = & S.code \,||\, label(S.next) \end{array} \right.$$

$S.next$ 为语句 S 指明了“跳出” S 的目标

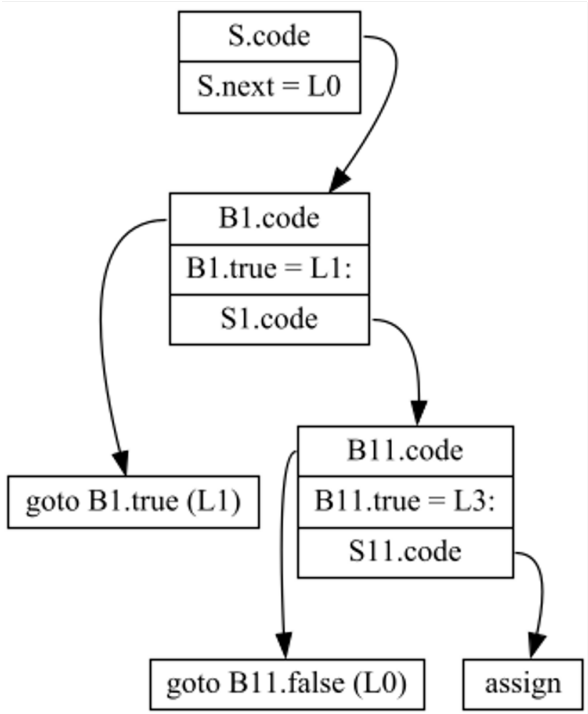
$$S \rightarrow \textbf{assign} \quad \Big| \quad S.code = \textbf{assign}.code$$

代表了表达式的翻译, 包括数组引用

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



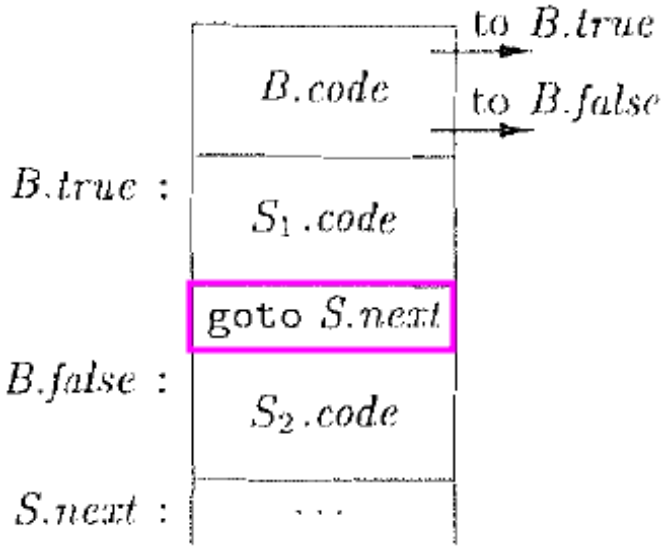
$S \rightarrow \textbf{if}\ (\ B\ )\ S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$




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

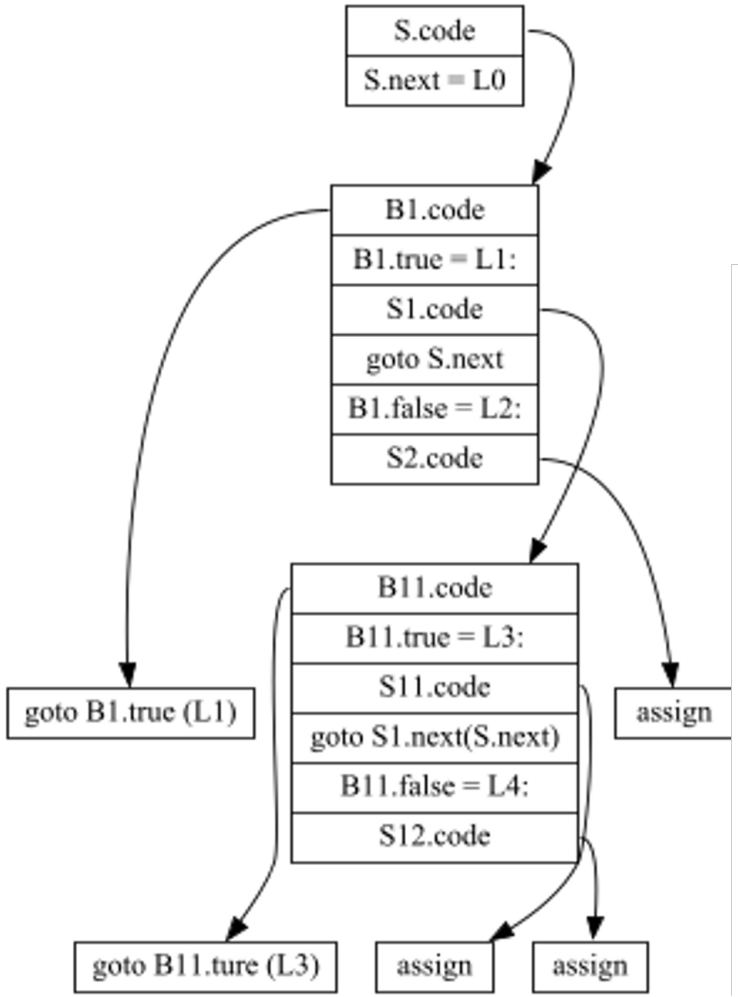
$B \rightarrow \textbf{true}$	$B.code = gen('\textbf{goto}'\ B.true)$
$B \rightarrow \textbf{false}$	$B.code = gen('\textbf{goto}'\ B.false)$

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



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

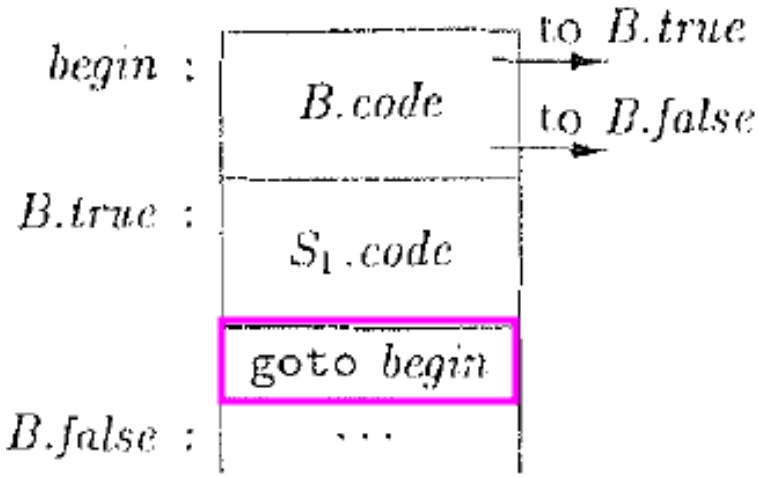
$B.true = newlabel()$
$B.false = newlabel()$
$S_1.next = S_2.next = S.next$
$S.code = B.code$
$\quad || label(B.true) || S_1.code$
$\quad || gen('\texttt{goto}'\ S.next)$
$\quad || label(B.false) || S_2.code$




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

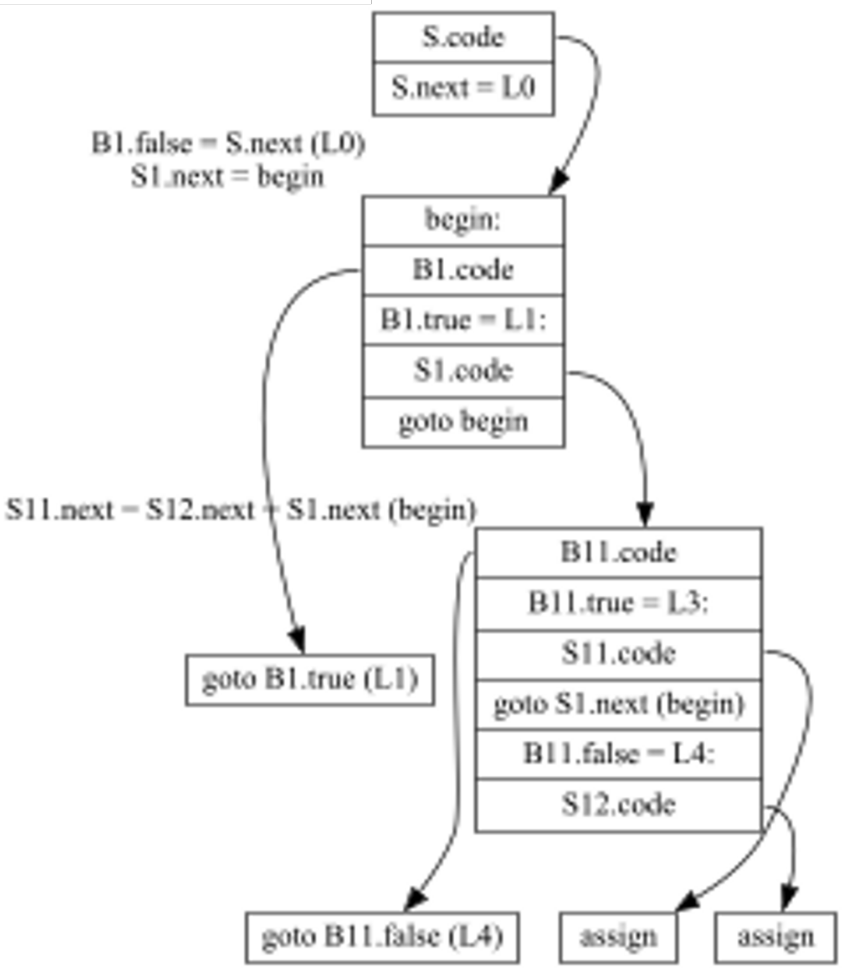
$S \rightarrow$ **while** (B) S_1

$begin = newlabel()$
$B.true = newlabel()$
$B.false = S.next$
$S_1.next = begin$
$S.code = label(begin) \,||\, B.code$
$\,||\, label(B.true) \,||\, S_1.code$
$\,||\, gen('\texttt{goto}'\ begin)$



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

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




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

$$S \rightarrow S_1\ S_2$$

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

$$
\begin{array}{lll}
S \rightarrow S_1\ S_2 & \boxed{S_1.next} = newlabel() \\
& \boxed{S_2.next} = S.next \\
& S.code = S_1.code \mid\mid label(S_1.next) \mid\mid S_2.code
\end{array}
$$

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

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

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

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

if (true) assign

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

if (false) assign

$$B \rightarrow \; ! \; B_1$$

$$\begin{cases} B_1.true = B.false \\ B_1.false = B.true \\ B.code = B_1.code \end{cases}$$

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

if (!true) assign

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

if (!false) assign

短路求值

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

短路求值

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

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

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

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

短路求值

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

短路求值

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

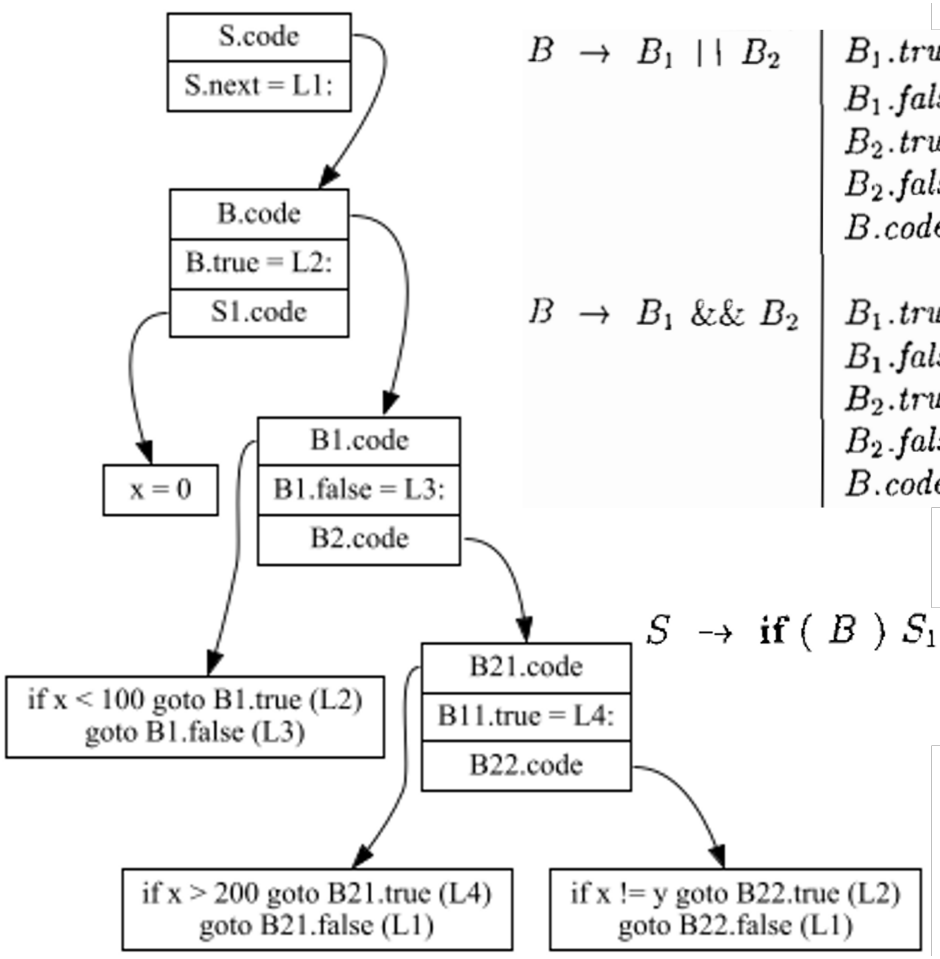
if (true && false) assign

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

if (false && true) assign

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

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



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

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

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

```
if (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:
```

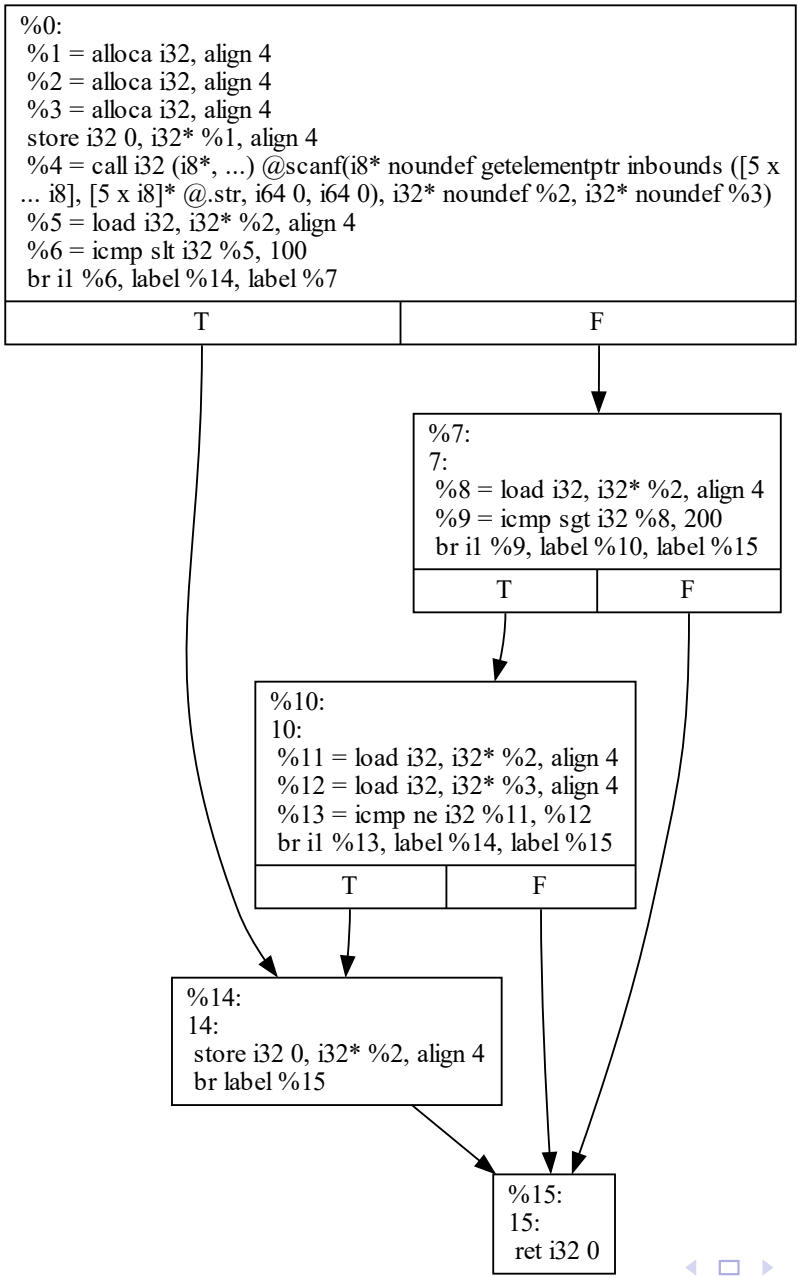
```
clang -S -emit-llvm if-boolexpr.c -o if-boolexpr-opt0.ll
```

```c
int main() {
  int x, y;
  scanf( Format: "%d%d", &x, &y);

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

  return 0;
}
```

```
opt -dot-cfg if-boolexpr-opt0.ll
dot -Tpdf .main.dot -o if-boolexpr-opt0-cfg.pdf
```



CFG for 'main' function

CodeGenListener.java

CodeGenListener.java

思考：如何实现增量式翻译？

```
1    while (true) {
2      if (false) {
3        a = b;
4      }
5    }
```

```
1  L2begin:
2  goto L3
3  L3:
4  goto L2begin
5  L4:
6  ASSIGN
7  goto L2begin
8  L1:
```

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

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

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

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

$S \rightarrow \textbf{id} = E\ ;$	$\{\ gen(top.get(\textbf{id}.lexeme)\ '='\ E.addr);\ \}$
$\mid\ L = E\ ;$	$\{\ gen(L.array.base\ '['\ L.addr\ ']'\ '='\ E.addr);\ \}$
$E \rightarrow E_1 + E_2$	$\{\ E.addr = \textbf{new}\ Temp();$ $gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr);\ \}$
$\mid\ \textbf{id}$	$\{\ E.addr = top.get(\textbf{id}.lexeme);\ \}$
$\mid\ L$	$\{\ E.addr = \textbf{new}\ Temp();$ $gen(E.addr\ '='\ L.array.base\ '['\ L.addr\ ']');\ \}$
$L \rightarrow \textbf{id}\ [\ E\ ]$	$\{\ L.array = top.get(\textbf{id}.lexeme);$ $L.type = L.array.type.elem;$ $L.addr = \textbf{new}\ Temp();$ $gen(L.addr\ '='\ E.addr\ '*'\ L.type.width);\ \}$
$\mid\ L_1\ [\ E\ ]$	$\{\ L.array = L_1.array;$ $L.type = L_1.type.elem;$ $t = \textbf{new}\ Temp();$ $L.addr = \textbf{new}\ Temp();$ $gen(t\ '='\ E.addr\ '*'\ L.type.width);$ $gen(L.addr\ '='\ L_1.addr\ '+'\ t);\ \}$

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

布尔表达式的作用: 布尔值 *vs.* 控制流跳转

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

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

布尔表达式的作用: 布尔值 *vs.* 控制流跳转

$S \rightarrow$ **id** $=E$; | **if** (E) S | **while** (E) S | S S

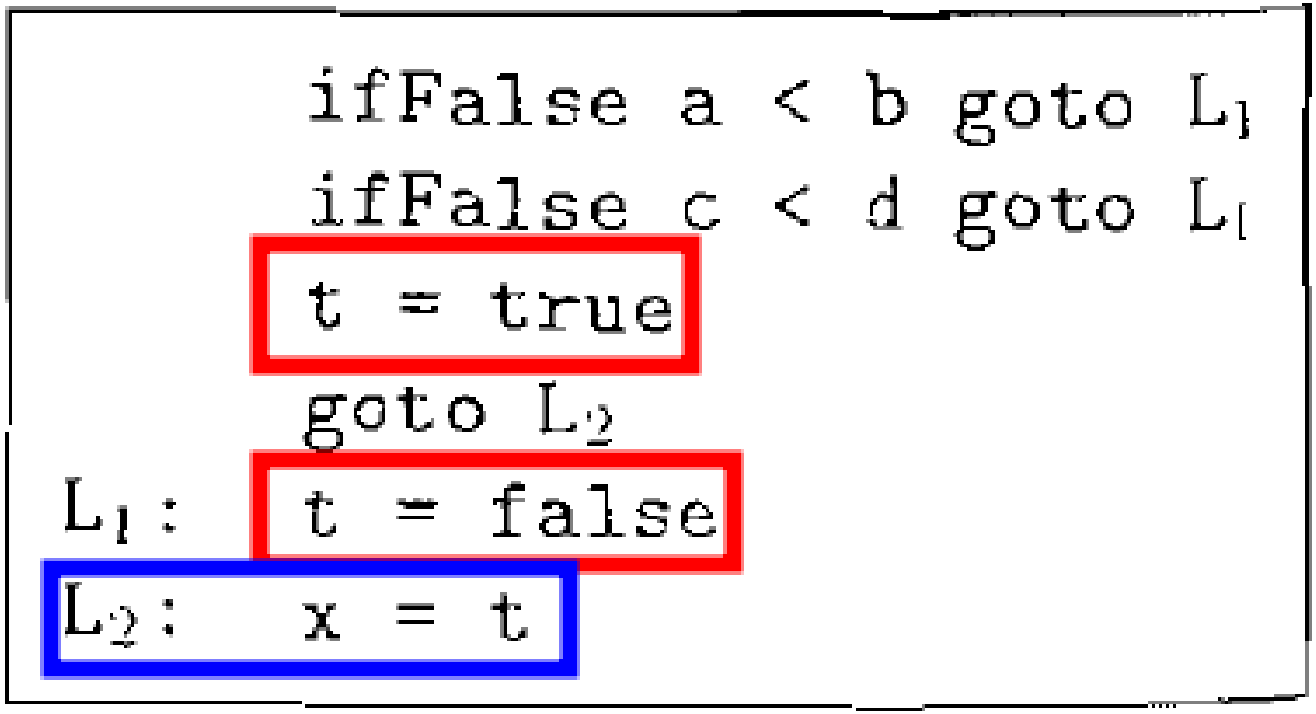
$E \rightarrow E \,\|\, E \mid E \,\&\&\, E \mid E$ **rel** $E \mid E + E \mid (E) \mid$ **id** | **true** | **false**

根据 E 所处的上下文判断 E 所扮演的角色, 调用不同的代码生成函数

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

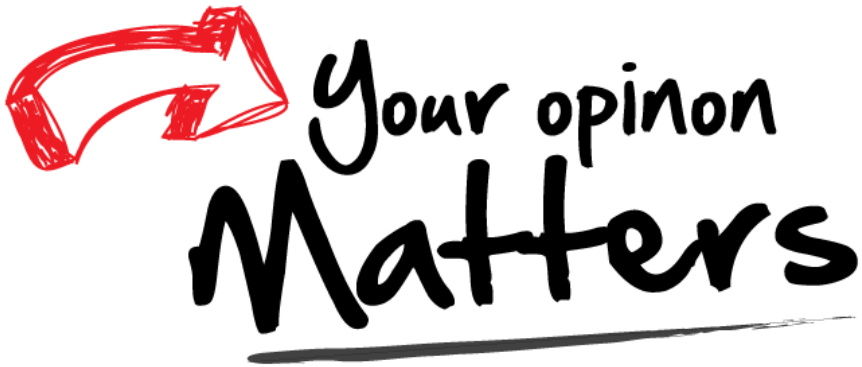
```
x = a < b && c < d
```



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

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


Thank
You!



Office 926

hfwei@nju.edu.cn