

四、中间代码生成 (3. 控制流语句的翻译)

魏恒峰

hfwei@nju.edu.cn

2023 年 05 月 10 日

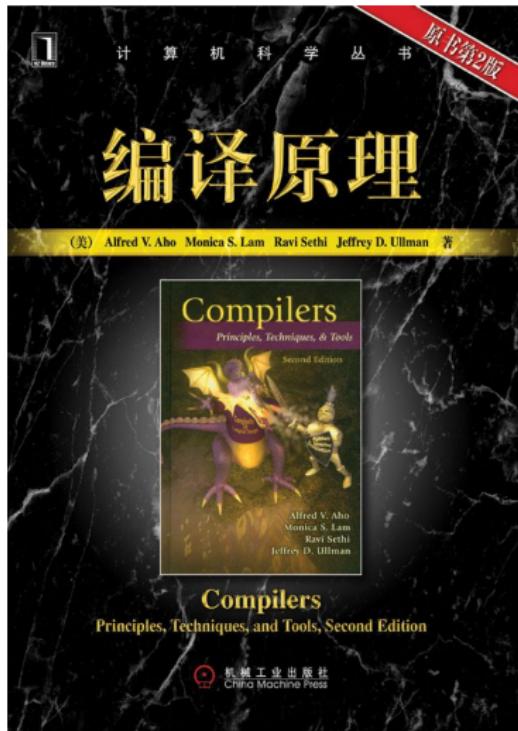


产生式	语义规则
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \parallel label(S.next)$
$S \rightarrow \text{assign}$	$S.code = \text{assign}.code$
$S \rightarrow \text{if } (B) S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \parallel label(B.true) \parallel S_1.code$
$S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

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



本讲内容颇有难度，需要多多思考



虽然有**非常简单的**讲法，教材偏偏采用了让初学者**望而生畏**的讲法。

“We will overcome all difficulties.” (我们有困难要上)



“If we don't have enough difficulties, we'll create them!”
(没困难, 我们创造困难还要上)

关键问题：为什么要“创造困难”？

生成更短、更高效的代码

我们先介绍一种非常简单的做法



心中有“树”(语法分析树)

分工 合作



为布尔表达式 B 计算逻辑值
if、**while** 语句根据 B 的结果改变控制流

if (B) S_1

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

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

if (B) S_1

Listener *vs.* **Visitors**



How to implement “break” statements?

短路求值

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

$$P \rightarrow S \quad S \rightarrow \mathbf{if} (B) \; S_1$$

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

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

继承属性 $S.next$

$P \rightarrow S$

$| S.next = newlabel()$
 $| P.code = S.code || label(S.next)$

$S.next$ 为语句 S 指明了“跳出” S 的目标

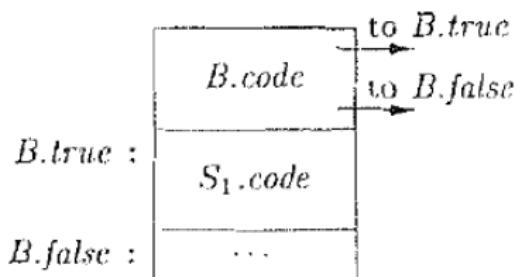
$S \rightarrow \text{assign}$

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

代表了表达式的翻译，包括数组引用

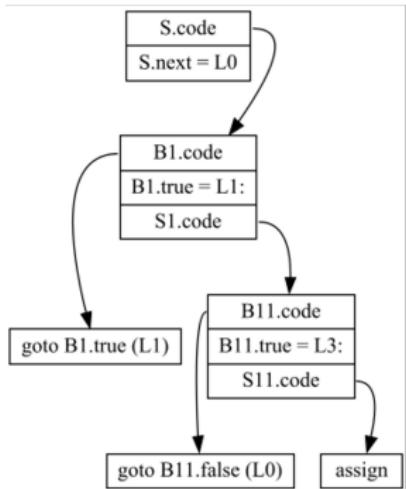
$S \rightarrow \text{if}(\ B\)\ S_1$

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



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

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



<code>if (true)</code>
<code>if (false) assign</code>

$B \rightarrow \text{true}$

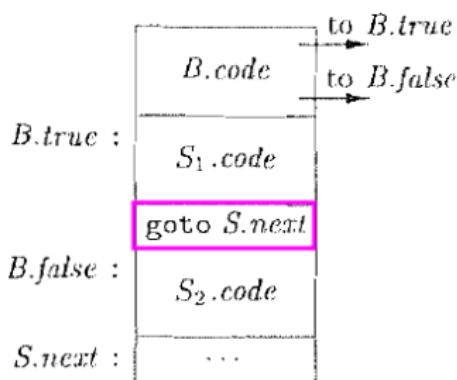
$B.\text{code} = \text{gen('goto' } B.\text{true})$

$B \rightarrow \text{false}$

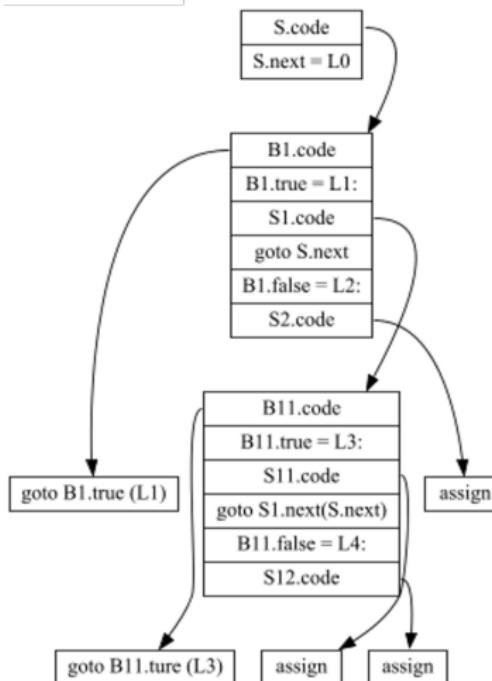
$B.\text{code} = \text{gen('goto' } B.\text{false})$
--

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

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



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



$B.\text{true} = \text{newlabel}()$

$B.\text{false} = \text{newlabel}()$

$S_1.\text{next} = S_2.\text{next} = S.\text{next}$

$S.\text{code} = B.\text{code}$

$\parallel \text{label}(B.\text{true}) \parallel S_1.\text{code}$

$\parallel \text{gen('goto' } S.\text{next})$

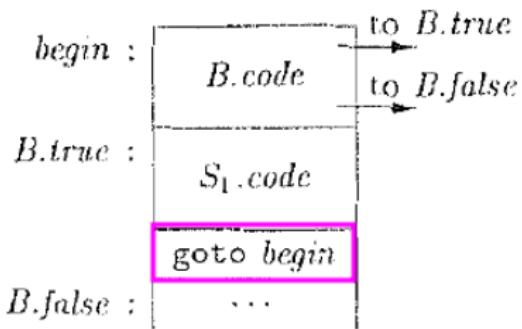
$\parallel \text{label}(B.\text{false}) \parallel S_2.\text{code}$

```

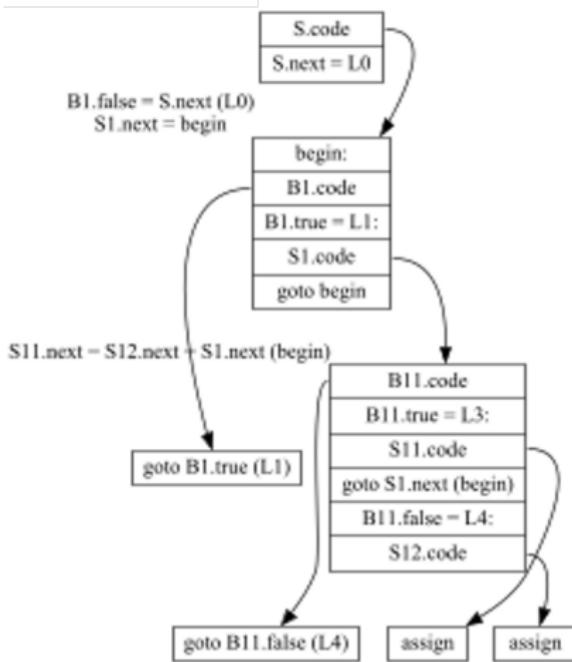
if (true)
  if (true) assign else assign
else
  assign
  
```

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

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



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



```

begin = newlabel()
B.true = newlabel()
B.false = S.next
S1.next = begin
S1.code = label(begin) || B.code
|| label(B.true) || S1.code
|| gen('goto' begin)
  
```

while (true)
if (false) assign else assign

$$S \rightarrow S_1 S_2$$
$$S_1.next = newlabel()$$
$$S_2.next = S.next$$
$$S.code = S_1.code || label(S_1.next) || S_2.code$$

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

布尔表达式的中间代码翻译

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

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

if (true) assign

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

if (false) assign

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

if (!true) assign

 $S \rightarrow \text{if} (B) S_1$
$$\begin{cases} B.\text{true} = \text{newlabel}() \\ B.\text{false} = S_1.\text{next} = S.\text{next} \\ S.\text{code} = B.\text{code} \parallel \text{label}(B.\text{true}) \parallel S_1.\text{code} \end{cases}$$

if (!false) assign

短路求值

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

if (true || false) assign

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

if (false || true) assign

短路求值

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

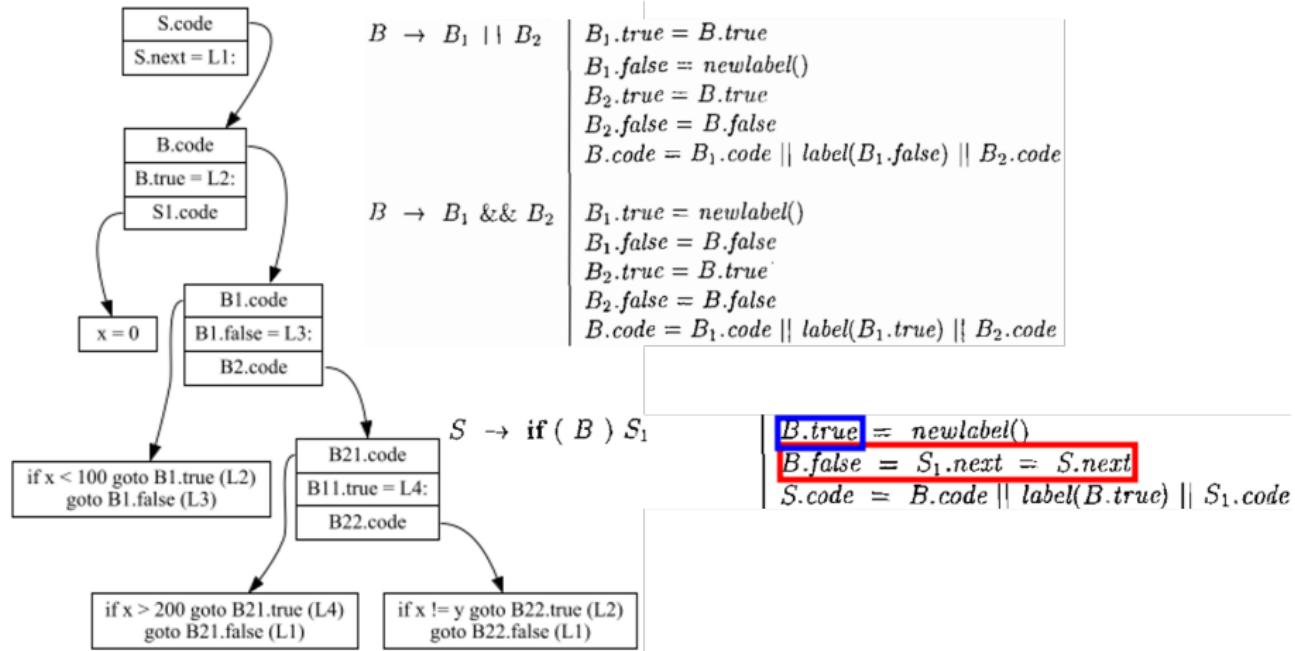
if (true && false) assign

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

if (false && true) assign

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

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



```
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
```

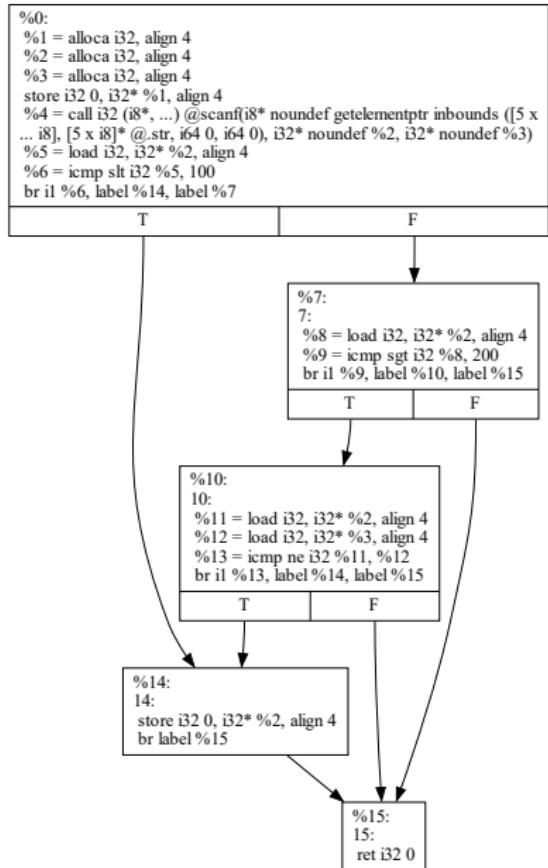
```
int main() {
    int x, y;
    scanf(Format: "%d%d", &x, &y);

    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
```



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

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

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

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

Control.g4

CodeGenListener.java



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

布尔表达式的作用: 布尔值 vs. 控制流跳转

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

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

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 $\text{jump}(t, f)$: 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

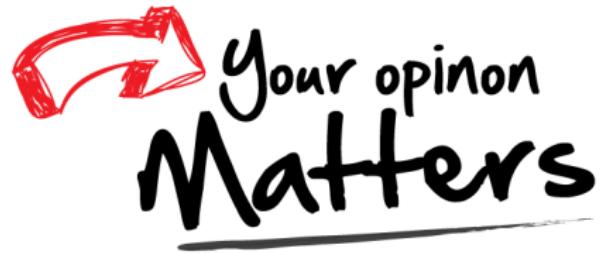
$x = a < b \&\& c < d$

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

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

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

Thank You!



Office 926

hfwei@nju.edu.cn