



中间代码生成

Part2: 控制流与布尔表达式翻译

李 诚

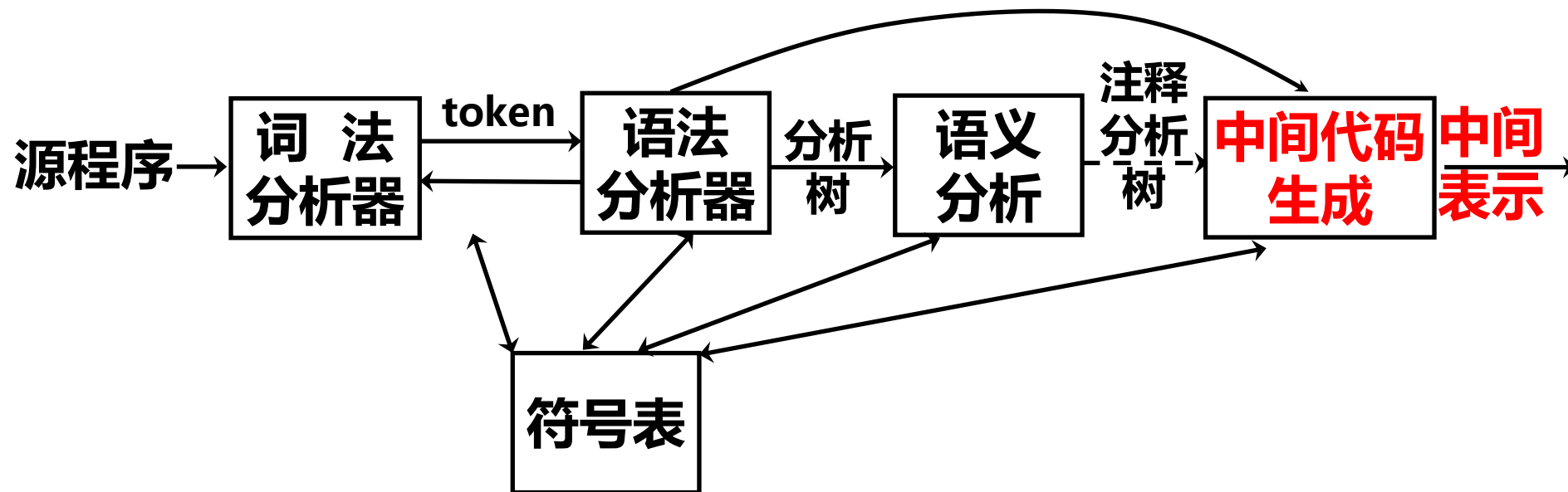
国家高性能计算中心(合肥)、信息与计算机国家级实验教学示范中心

计算机科学与技术学院

2025年10月15日



本节提纲



- 控制流语句文法
- 简单控制流语句的翻译
 - if, if-then-else, while, 顺序语句
- 布尔表达式的翻译



控制流语句的文法



$S \rightarrow \text{if } B \text{ then } S_1$

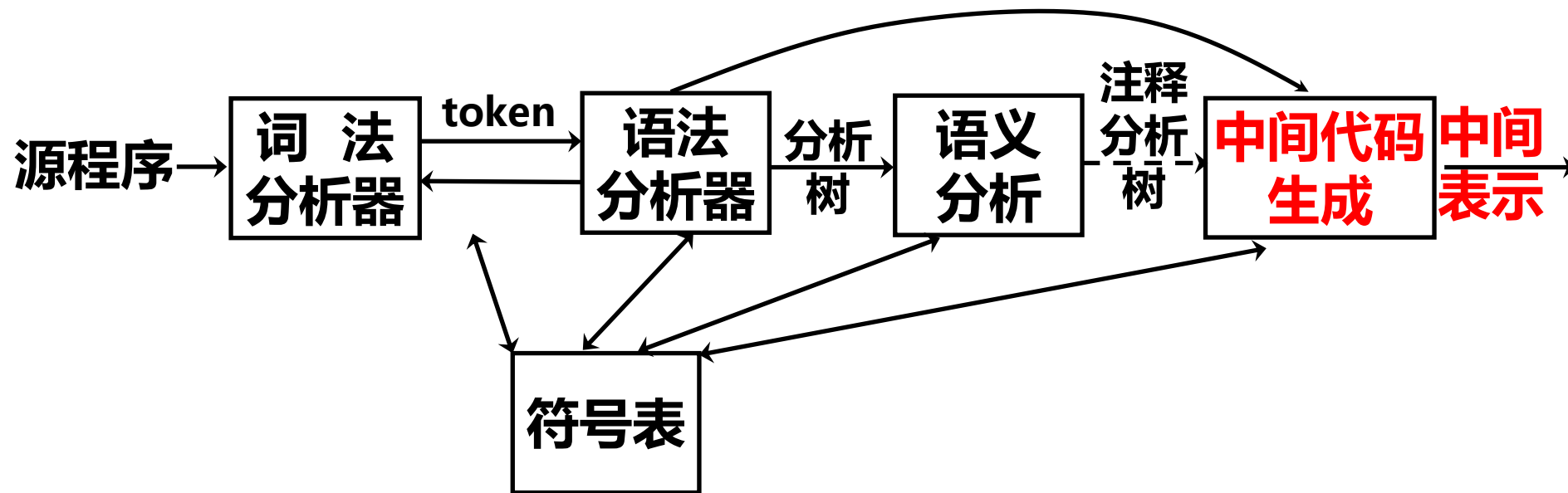
$\quad / \text{if } B \text{ then } S_1 \text{ else } S_2$

$\quad / \text{while } B \text{ do } S_1$

$\quad / S_1; S_2$



本节提纲



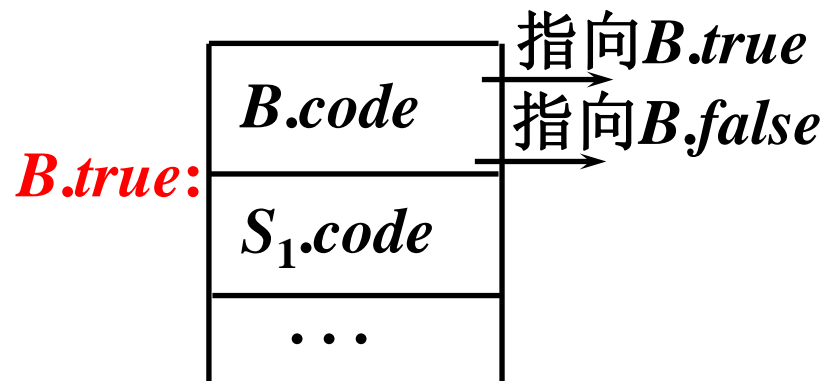
- 控制流语句文法
- 简单控制流语句的翻译
 - if, if-then-else, while, 顺序语句
- 布尔表达式的翻译



• 问题与对策

- 需要知道**B**为真或假时的跳转目标
- **B**、**S₁**分别会输出多少条指令是不确定的
- 引入标号：先确定标号，在目标确定时输出标号指令，可调用newLabel()
产生新标号，每条语句有next 标号

$S \rightarrow \text{if } B \text{ then } S_1$



(a) if-then



• 问题与对策

- 需要知道**B**为真或假时的跳转目标
- **B**、**S₁**分别会输出多少条指令是不确定的
- 引入**标号**：先确定标号，在目标确定时输出标号指令，可调用newLabel()
产生新标号，每条语句有next 标号

$S \rightarrow \text{if } B \text{ then } S_1$

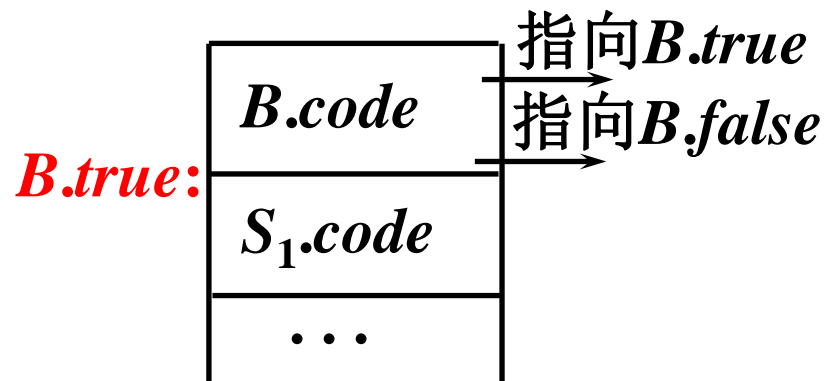
语义规则：

$B.true = \text{newLabel}();$

$B.false = S.next; // \text{继承属性}$

$S_1.next = S.next;$

$S.code = B.code \parallel \text{gen}(B.true, '::') \parallel S_1.code$



(a) if-then



if 语句中间代码生成的SDD



• 问题与对策

- 需要知道**B**为真或假时的跳转目标
- **B**、**S₁**分别会输出多少条指令是不
- 引入**标号**：先确定标号，在目标产生新标号，每条语句有**next**标

- 标号指向S内部的三地址代码时需要调用newLabel
- 标号指向S外部的三地址代码时从S继承

$S \rightarrow \text{if } B \text{ then } S_1$

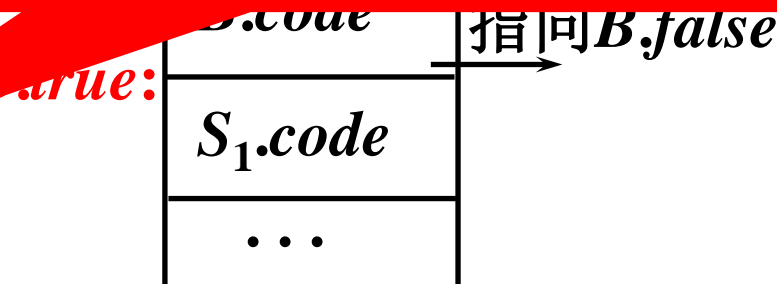
语义规则：

$B.true = \text{newLabel}();$

$B.false = S.next; // \text{继承属性}$

$S_1.next = S.next;$

$S.code = B.code \parallel \text{gen}(B.true, ':') \parallel S_1.code$



(a) if-then



• 问题与对策

- 需要知道**B**为真或假时的跳转目标
- **B**、**S₁**分别会输出多少条指令是不确定的
- 引入**标号**：先确定标号，在目标确定时输出标号指令，可调用newLabel()
产生新标号，每条语句有**next** 标号

$S \rightarrow \text{if } B \text{ then } S_1$

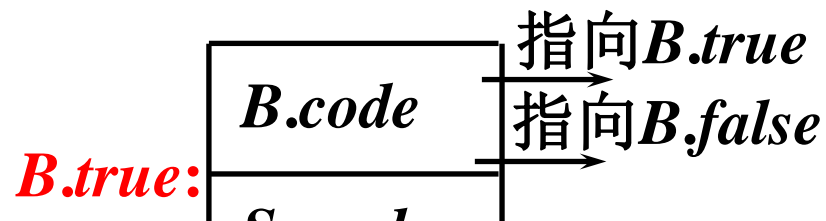
语义规则：

$B.true = \text{newLabel}();$

$B.false = S.next; // \text{继承属性}$

$S_1.next = S.next;$

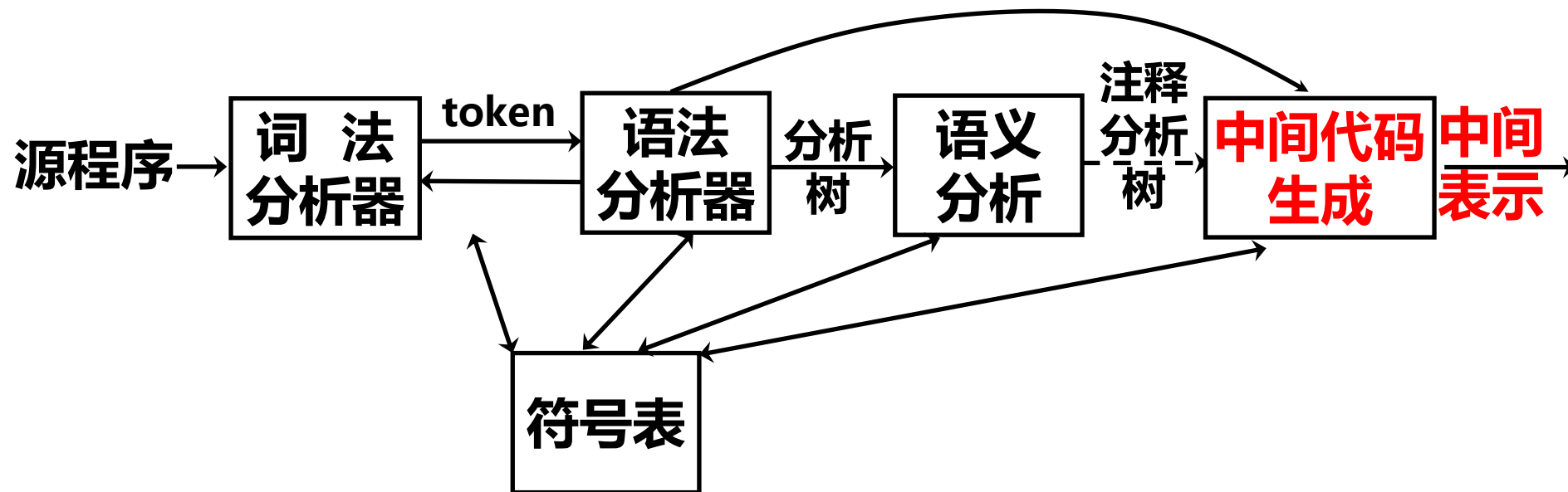
$S.code = B.code \parallel \text{gen}(B.true, ':') \parallel S_1.code$



把 $B.true$ 这个标号
附加到 S_1 的第一条
三地址指令上



本节提纲



- 控制流语句文法
- 简单控制流语句的翻译
 - if, if-then-else, while, 顺序语句
- 布尔表达式的翻译

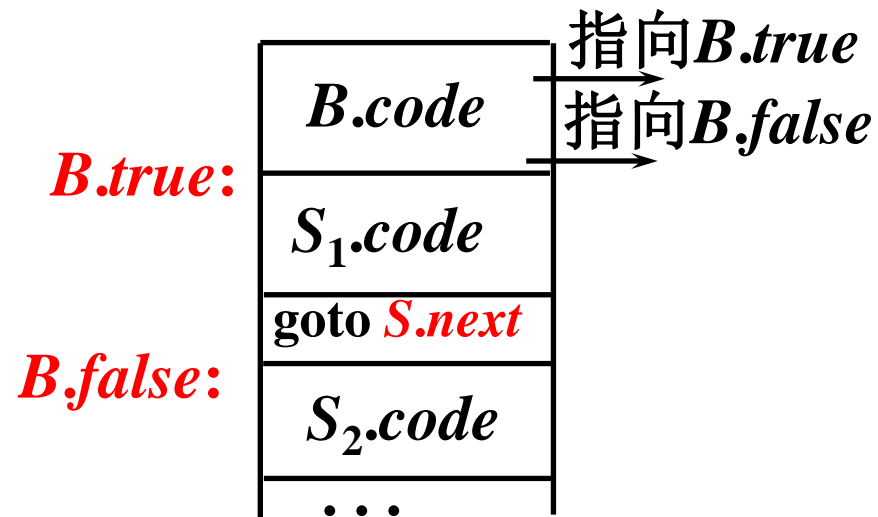


if 语句中间代码生成的SDD



- 考虑带有else的语句

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



(b) if-then-else



if 语句中间代码生成的SDD



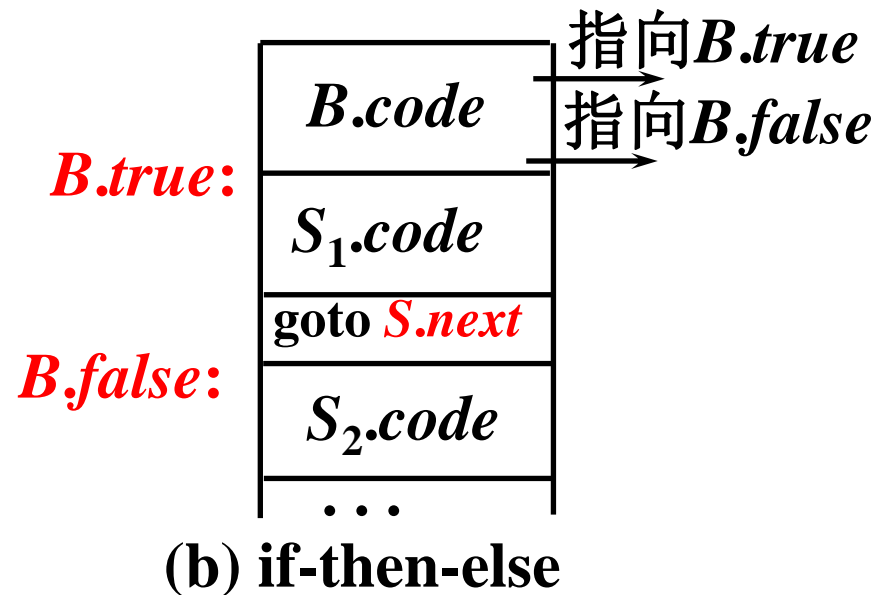
• 考虑带有else的语句

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

语义规则:

$B.true = newLabel();$

$B.false = newLabel();$





if 语句中间代码生成的SDD



• 考虑带有else的语句

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

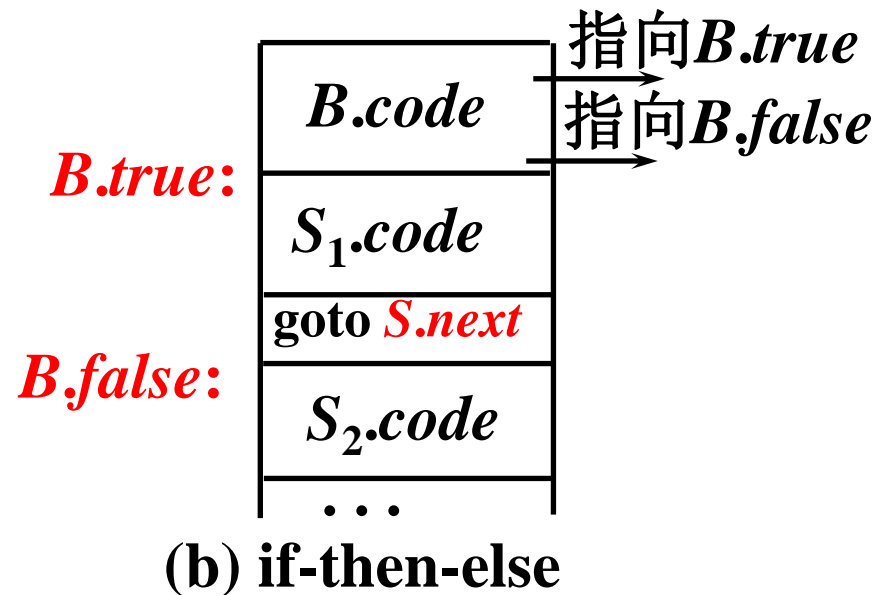
语义规则:

$B.true = newLabel();$

$B.false = newLabel();$

$S_1.next = S.next;$

$S_2.next = S.next;$





if 语句中间代码生成的SDD



• 考虑带有else的语句

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

语义规则:

$B.true = \text{newLabel}();$

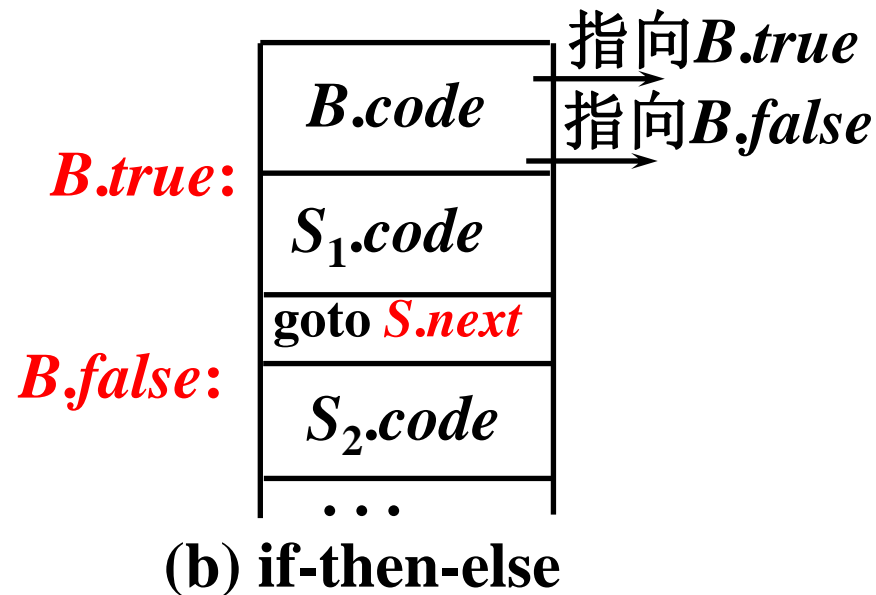
$B.false = \text{newLabel}();$

$S_1.next = S.next;$

$S_2.next = S.next;$

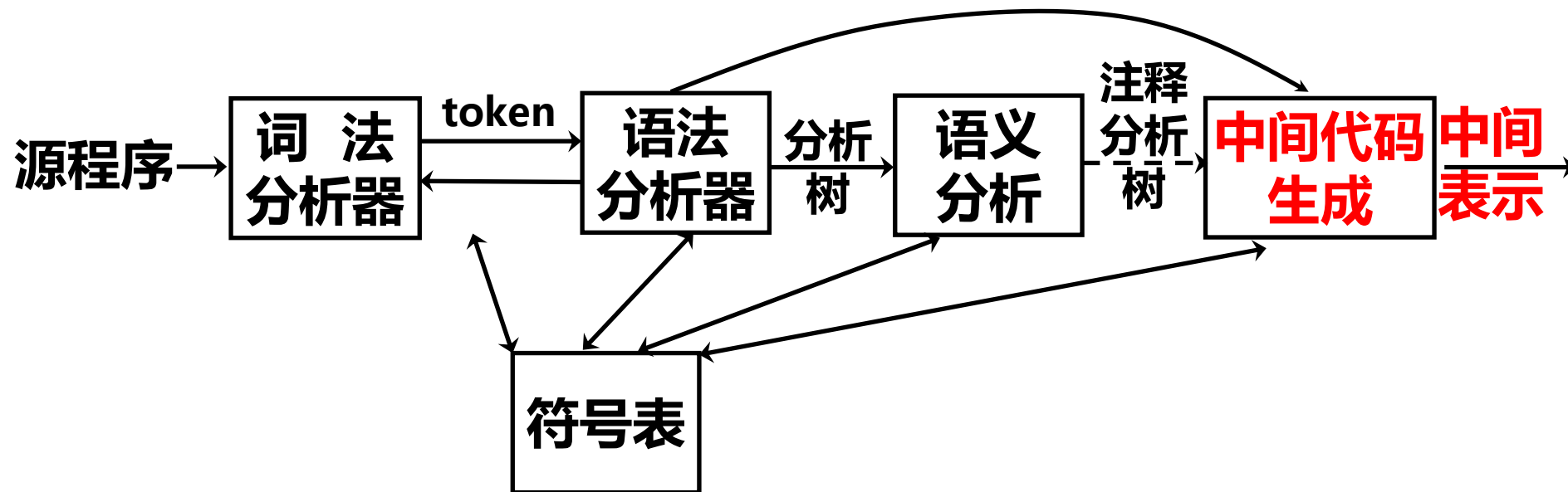
$S.code = B.code \parallel \text{gen}(B.true, ':') \parallel S_1.code \parallel$

$\text{gen}(\text{'goto'}, S.next) \parallel \text{gen}(B.false, ':') \parallel S_2.code$





本节提纲

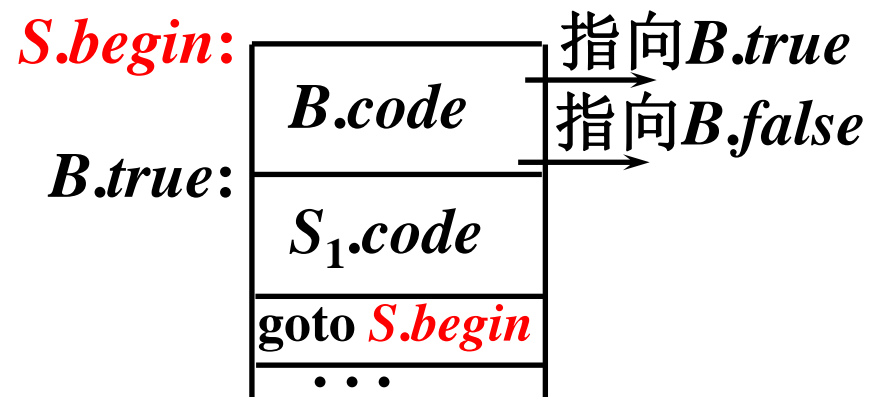


- 控制流语句文法
- 简单控制流语句的翻译
 - if, if-then-else, while, 顺序语句
- 布尔表达式的翻译



- 引入开始标号 *S.begin*, 作为循环的跳转目标

S \rightarrow while *B* do *S*₁



(c) while-do



- 引入开始标号 *S.begin*, 作为循环的跳转目标

$S \rightarrow \text{while } B \text{ do } S_1$

语义规则:

$S.begin = \text{newLabel}();$

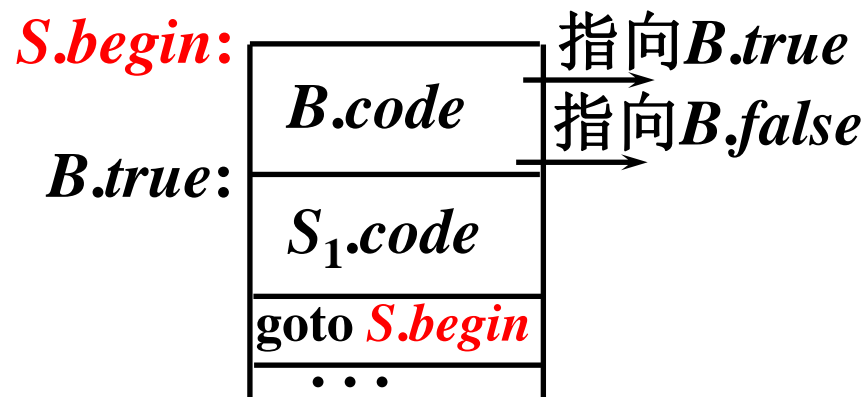
$B.true = \text{newLabel}();$

$B.false = S.next;$

$S_1.next = S.begin;$

$S.code = \text{gen}(S.begin, ':') \parallel B.code \parallel$

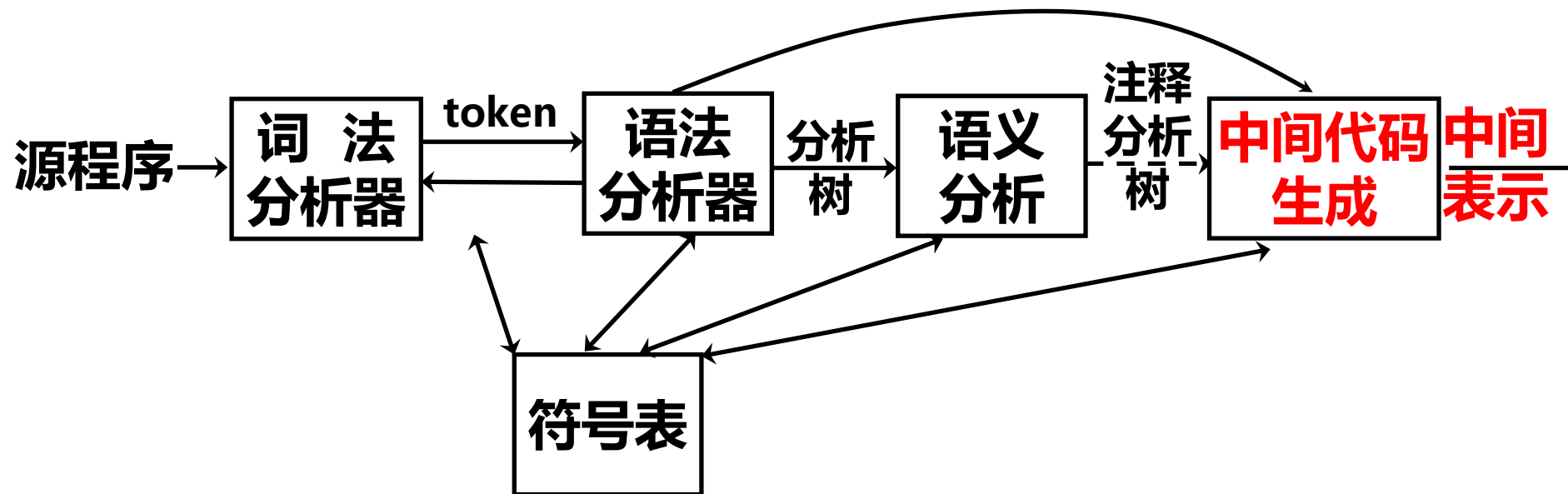
$\text{gen}(B.true, ':') \parallel S_1.code \parallel \text{gen}(\text{'goto'}, S.begin)$



(c) while-do



本节提纲



- 控制流语句文法
- 简单控制流语句的翻译
 - if, if-then-else, while, 顺序语句
- 布尔表达式的翻译



顺序结构中间代码生成SDD



- 为每一语句 S_1 引入其后的下一条语句的标号 $S_1.next$

$S \rightarrow S_1; S_2$

语义规则:

$S_1.next = newLabel(); S_2.next = S.next;$

$S.code = S_1.code \parallel gen(S_1.next, ':') \parallel S_2.code$

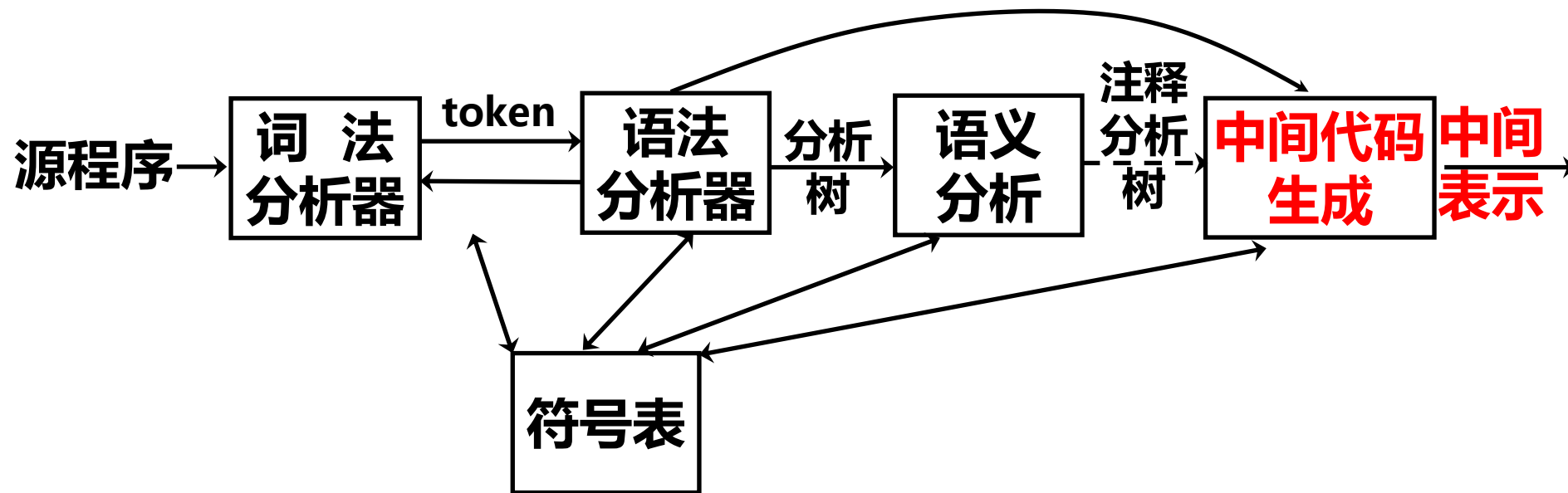
$S_1.next:$

$S_1.code$
$S_2.code$
\dots

(d) $S_1; S_2$



本节提纲



- 控制流语句文法
- 简单控制流语句的翻译
 - if, if-then-else, while, 顺序语句
- 布尔表达式的翻译



- 在if-else以及while语句翻译中，并未对B.code进行展开，现在考虑B.code的三地址代码翻译
- 如果 B 是 $a < b$ 的形式，

那么翻译生成的三地址码是：

if $a < b$ goto $B.true$

goto $B.false$

我们把这种翻译称为“布尔表达式的控制流翻译”



- 布尔表达式有两个基本目的
 - 计算逻辑值
 - 例如：作为赋值语句的右值
 - 在控制流语句中用作条件表达式
 - 例如： *if(B) then S*
- 本节所用的布尔表达式文法
$$B \rightarrow B \text{ or } B \mid B \text{ and } B \mid \text{not } B \mid (B)$$
$$\mid E \text{ relop } E \mid \text{true} \mid \text{false}$$



- 布尔表达式有两个基本目的
 - 计算逻辑值
 - 在控制流语句中用作条件表达式
- 本节所用的布尔表达式文法

$$B \rightarrow B \text{ or } B \mid B \text{ and } B \mid \text{not } B \mid (B)$$
$$\mid E \text{ relop } E \mid \text{true} \mid \text{false}$$

- 布尔运算符 or、and 和 not（优先级、结合性）
- 关系运算符 relop: $<$ 、 \leq 、 $=$ 、 \neq 、 $>$ 和 \geq
- 布尔常量: true 和 false



- 布尔表达式的完全计算

- 值的表示数值化
- 其计算类似于算术表达式的计算

true and false or (2 > 1)的计算为

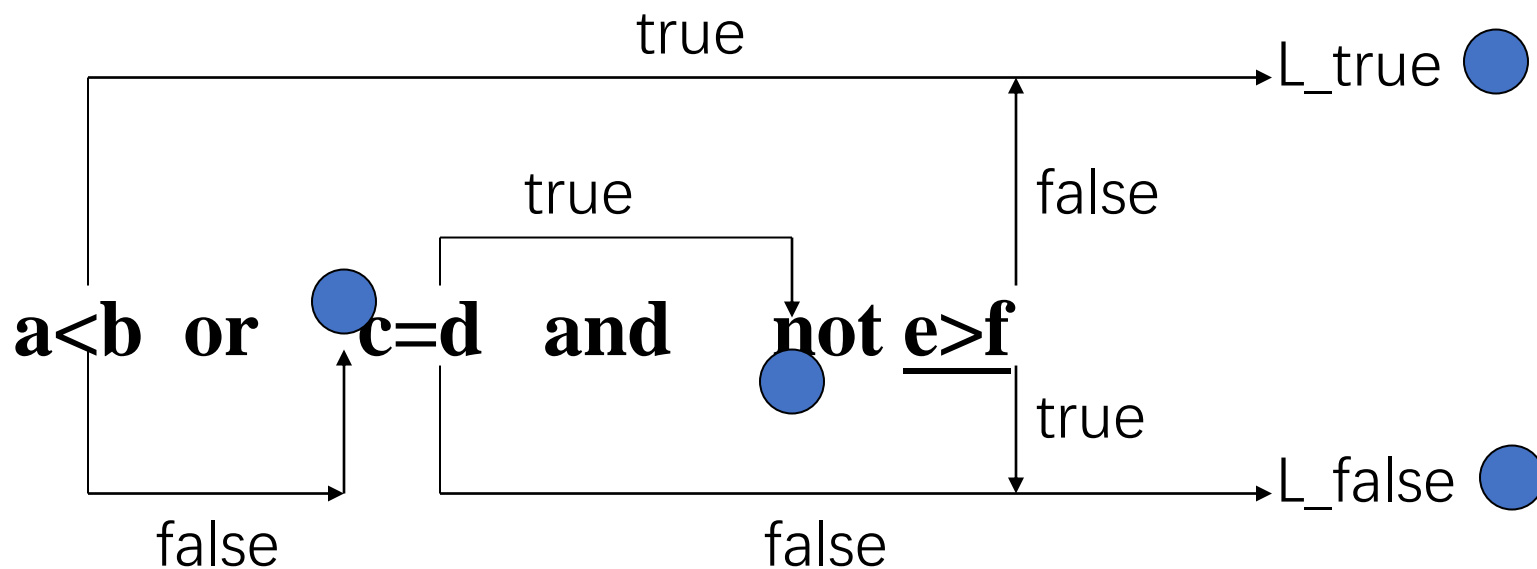
→ false or (2 > 1) → false or true → true

- 布尔表达式的“短路” 计算

- B_1 or B_2 B_1 为真即为真
- B_1 and B_2 B_1 为假即为假



布尔表达式的短路计算



L_true-真出口：整个布尔表达式为**真**时，控制流应转移到的目标语句（代码）；反之为**假**时则转到 L_false-假出口。

● 表示转移到的目标语句在有关布尔表达式翻译时尚未确定。



- **用控制流来实现计算**

- 布尔运算符and, or, not不出现在翻译后的代码中
- 用程序中的位置来表示值

- **例：if($x < 3$ or $x > 5$ and $x \neq y$) $x = 10$;的翻译**

```
        if x<3 goto L2
        goto L3
L3:      if x>5 goto L4
        goto L1
L4:      if x!=y goto L2
        goto L1
L2:      x = 10
L1:
```



- 例 表达式

$a < b$ or $c < d$ and $e < f$

的三地址码是：

if $a < b$ goto L_{true}

goto L_1

L_1 : if $c < d$ goto L_2

goto L_{false}

L_2 : if $e < f$ goto L_{true}

goto L_{false}



布尔表达式的控制流翻译



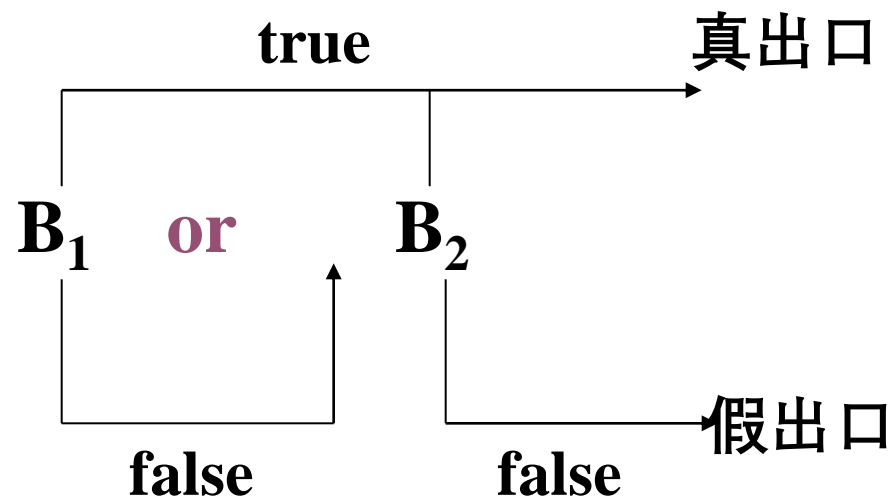
$$B \rightarrow B_1 \text{ or } B_2$$



布尔表达式的控制流翻译



$B \rightarrow B_1 \text{ or } B_2$





布尔表达式控制流翻译SDD



$B \rightarrow B_1 \text{ or } B_2$

语义规则:

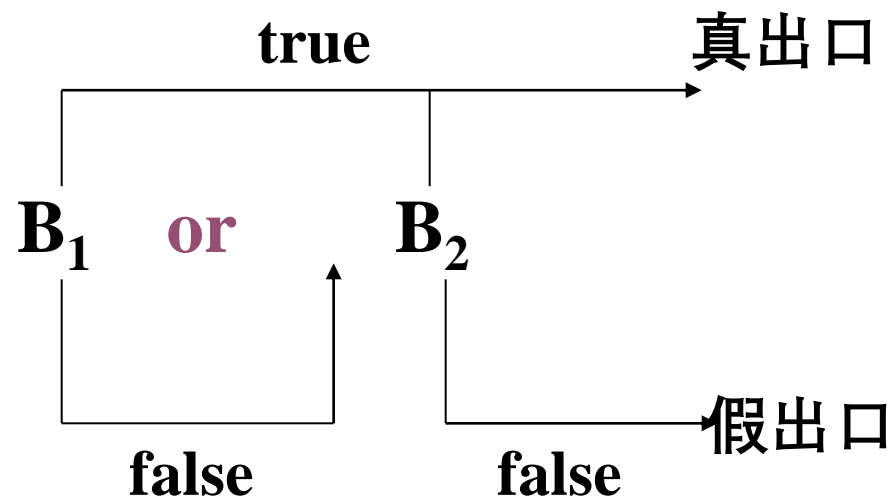
$B_1.true = B.true;$

$B_1.false = newLabel();$

$B_2.true = B.true;$

$B_2.false = B.false;$

$B.code = B_1.code \parallel gen(\textcolor{red}{B_1.false}, ':') \parallel B_2.code$

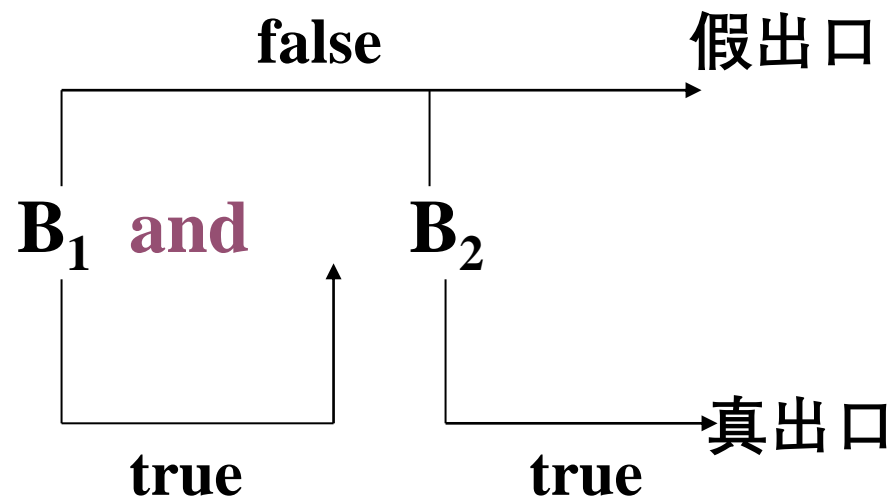




布尔表达式的控制流翻译



$B \rightarrow B_1 \text{ and } B_2$





布尔表达式控制流翻译SDD



$B \rightarrow B_1 \text{ and } B_2$

语义规则:

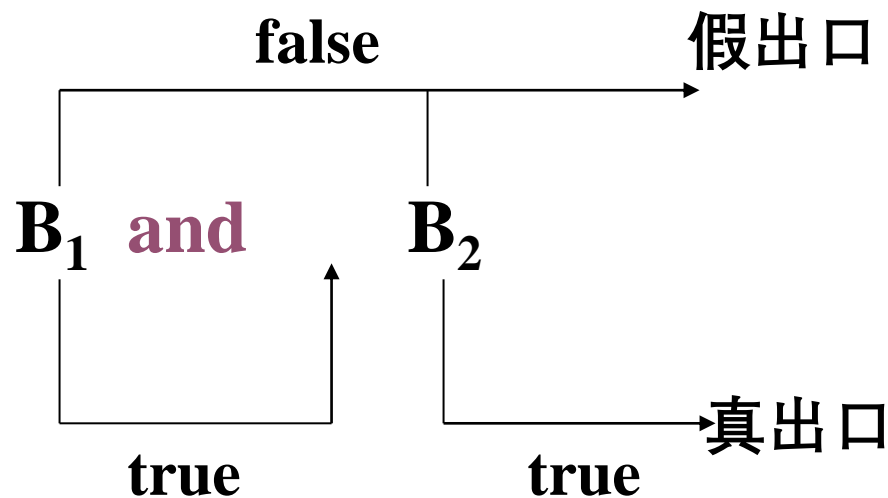
$B_1.true = newLabel();$

$B_1.false = B.false;$

$B_2.true = B.true;$

$B_2.false = B.false;$

$B.code = B_1.code \parallel gen(B_1.true, ':') \parallel B_2.code$

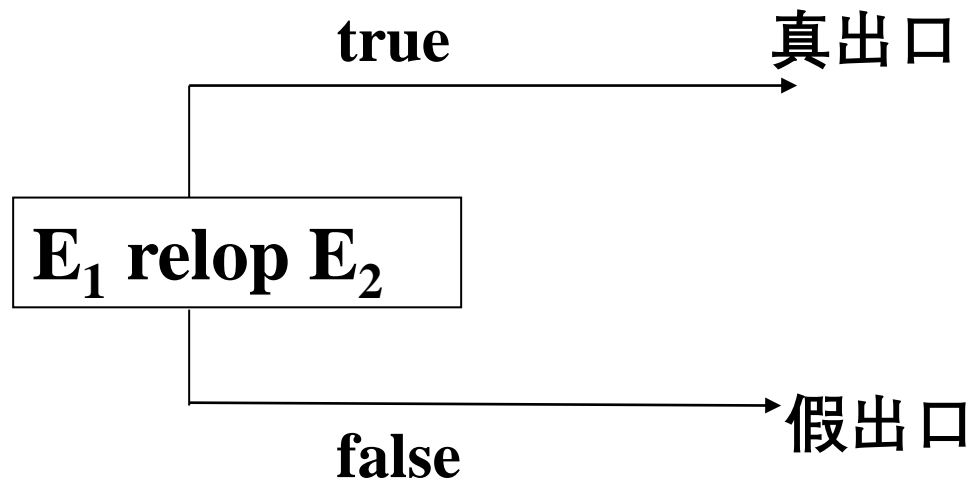




布尔表达式的控制流翻译



$B \rightarrow E_1 \text{ relop } E_2$





布尔表达式控制流翻译SDD



$B \rightarrow E_1 \text{ relop } E_2$

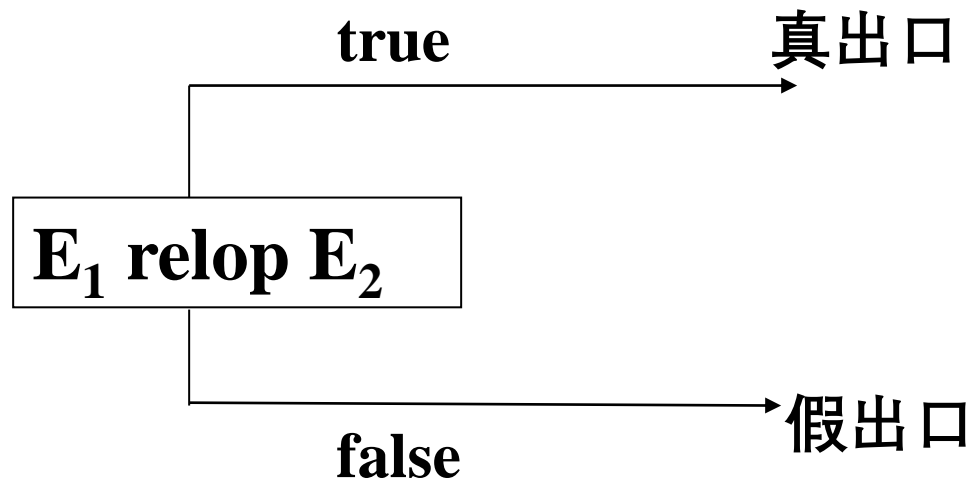
语义规则:

$B.\text{code} = E_1.\text{code} \parallel E_2.\text{code} \parallel$

$\text{gen}(\text{'if'}, E_1.\text{place}, \text{relop.op}, E_2.\text{place},$

$\text{'goto'}, B.\text{true}) \parallel$

$\text{gen}(\text{'goto'}, B.\text{false})$





布尔表达式控制流翻译SDD



$B \rightarrow (B_1)$

语义规则:

$B_1.true = B.true;$

$B_1.false = B.false;$

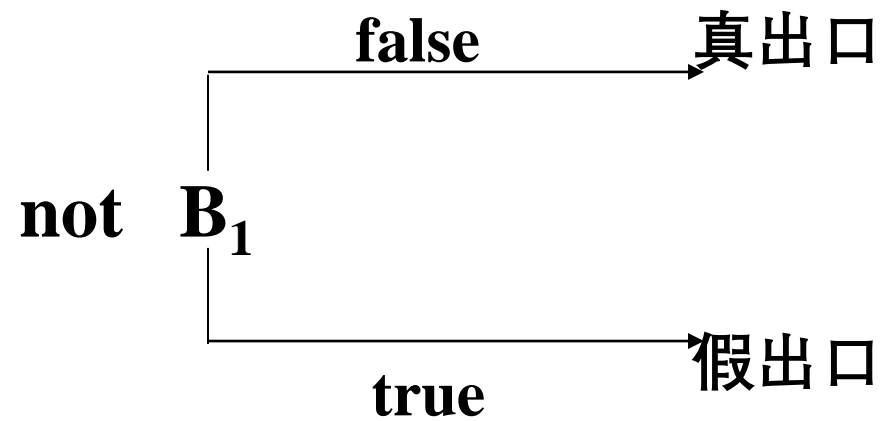
$B.code = B_1.code$



布尔表达式的控制流翻译



$B \rightarrow \text{not } B_1$





布尔表达式控制流翻译SDD



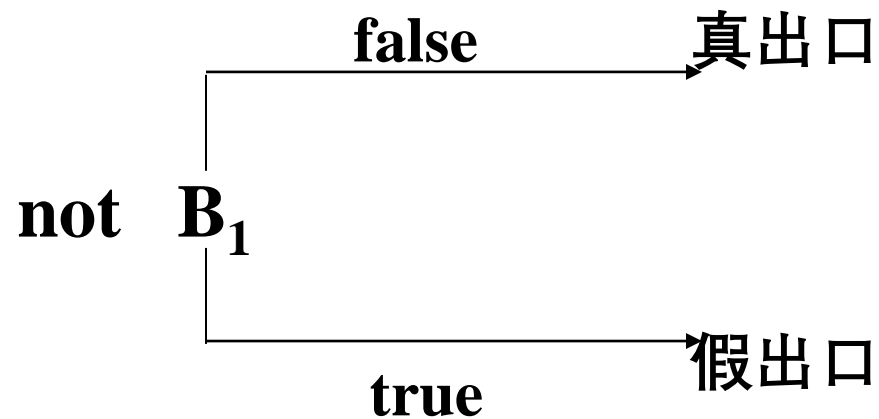
$B \rightarrow \text{not } B_1$

语义规则:

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

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

$B.\text{code} = B_1.\text{code}$





$B \rightarrow \text{true}$

语义规则:

$B.code = gen('goto', B.true)$

$B \rightarrow \text{false}$

语义规则:

$B.code = gen('goto', B.false)$



- **关键问题：将跳转指令与目标匹配起来**
- **B.true, B.false都是继承属性**
- **需要两趟分析来计算**
 - 1 pass：生成语法树
 - 2 pass：深度优先遍历树，计算属性值
 - 将标号和具体地址绑定起来
- **能否一趟完成？**
 - 标号回填技术



一起努力 打造国产基础系统软件体系！

李 诚

国家高性能计算中心(合肥)、信息与计算机国家级实验教学示范中心

计算机科学与技术学院

2025年10月15日