中间代码生成
Part2：控制流与布尔表达式翻译

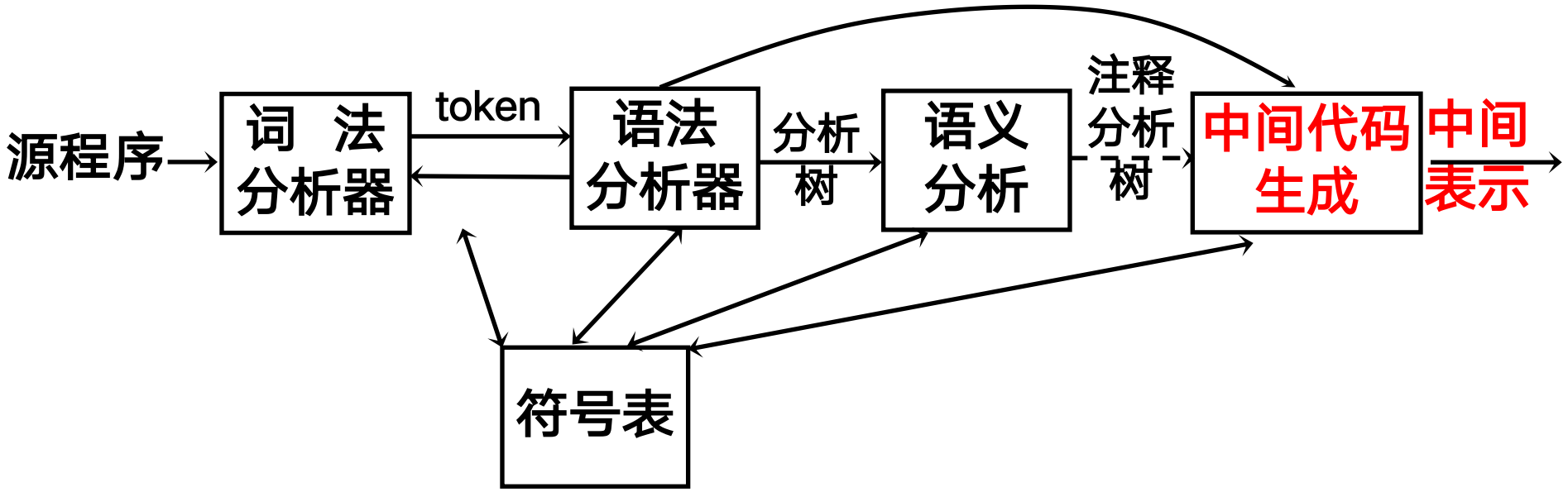
李 诚

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03月27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

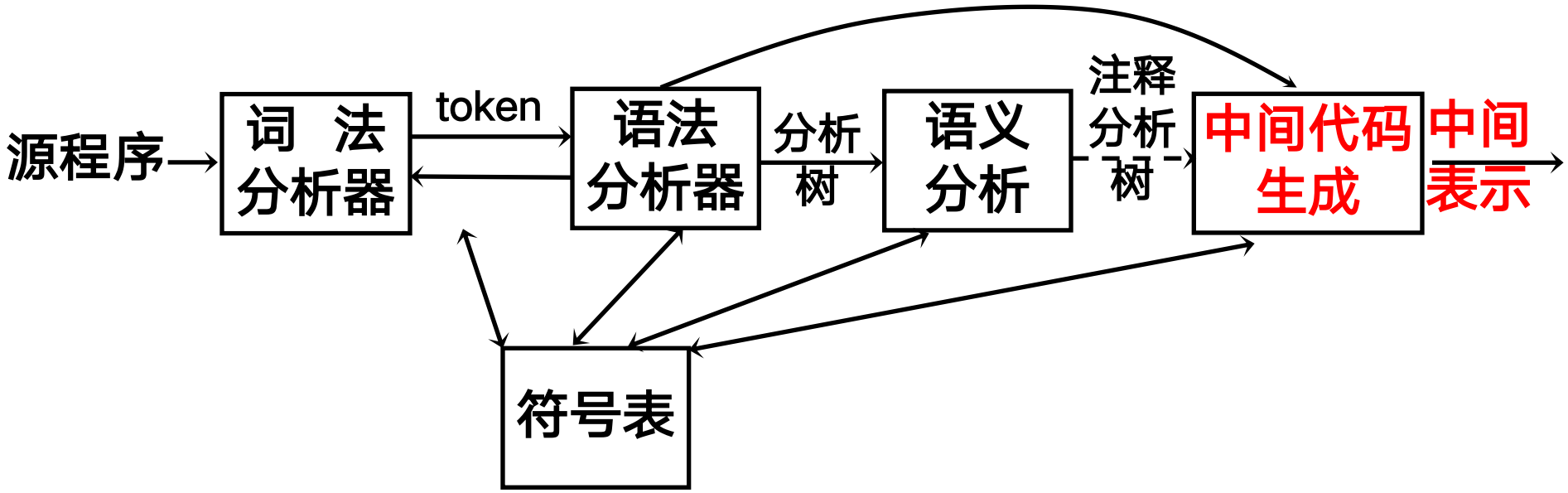
$S \rightarrow$ if B then S_1

| if B then S_1 else S_2

| while B do S_1

| S_1; S_2

本节提纲

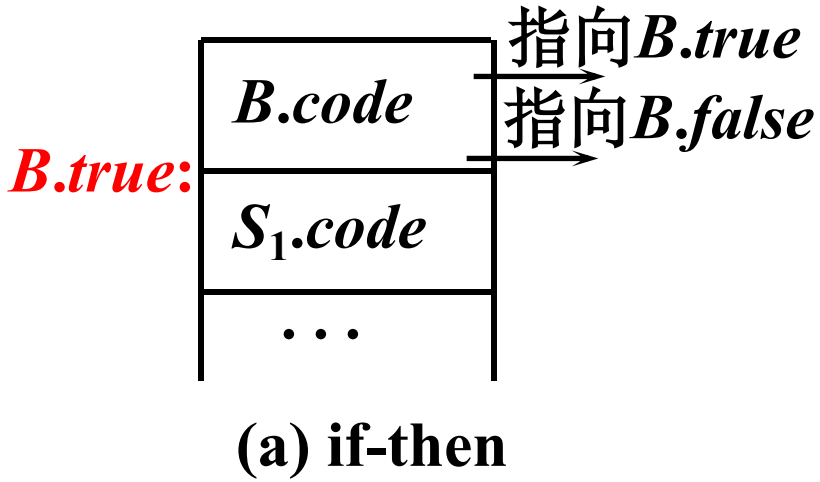


- 控制流语句文法
- **简单控制流语句的翻译**
  - if, if-then-else, while, 顺序语句
- **布尔表达式的翻译**

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、S_1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

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



(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、S_1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

$S \rightarrow$ **if** B **then** S_1

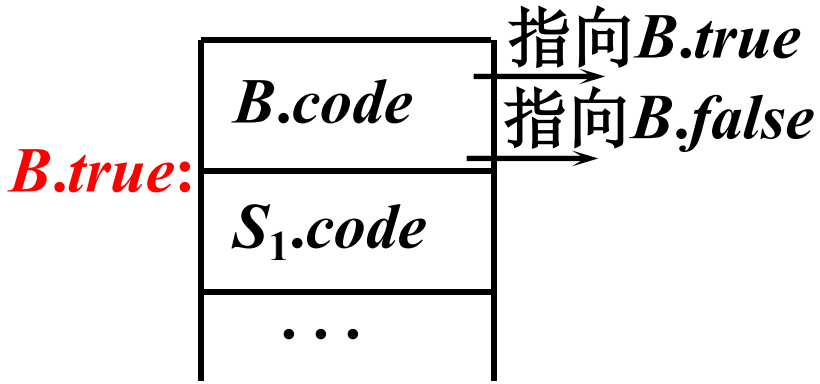
语义规则：

B.true = *newLabel*();

B.false = *S.next*;//继承属性

S_1.*next* = *S.next*;

S.code = *B.code* || *gen*(*B.true*, ':') || 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、S_1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

ewLabel()

- 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

$S \rightarrow$ if B then S_1

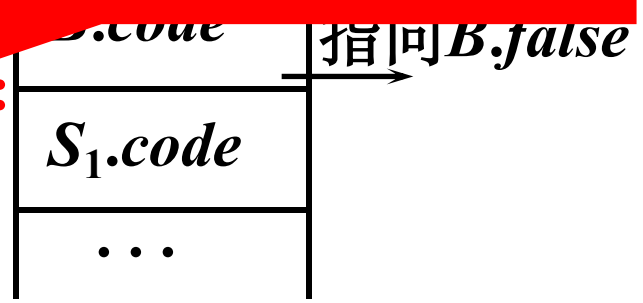
语义规则：

$B.true = newLabel();$

$B.false = S.next;$//继承属性

$S_1.next = S.next;$

$S.code = B.code \,||\, gen(B.true, ':') \,||\, S_1.code$



(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、S_1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

$S \rightarrow$ **if** B **then** S_1

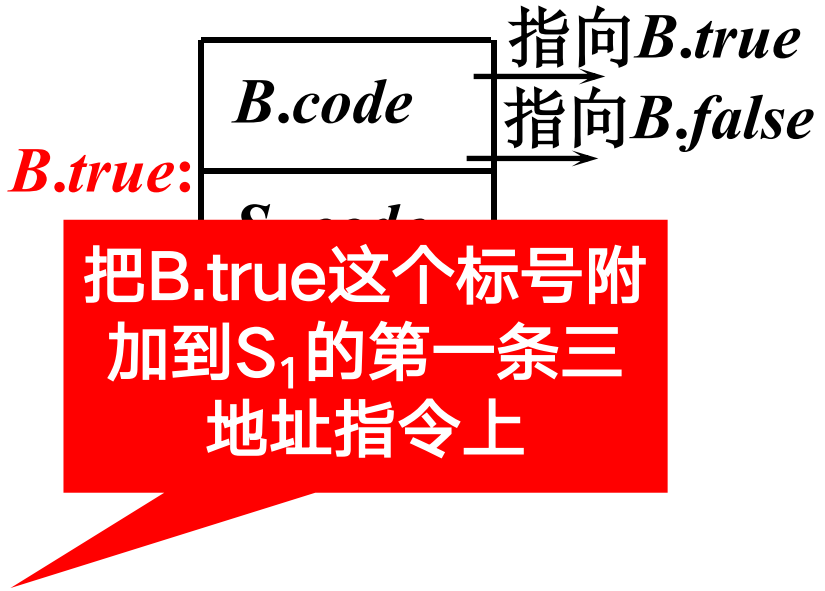
语义规则：

$B.true = newLabel();$

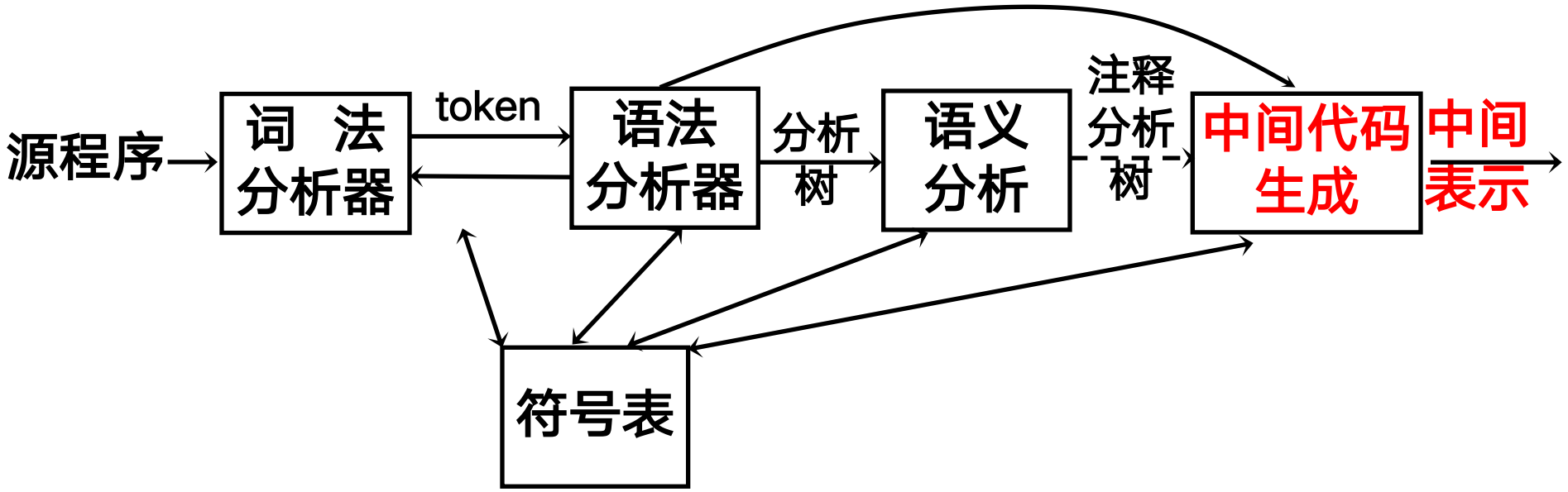
$B.false = S.next;$//**继承属性**

$S_1.next = S.next;$

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



本节提纲

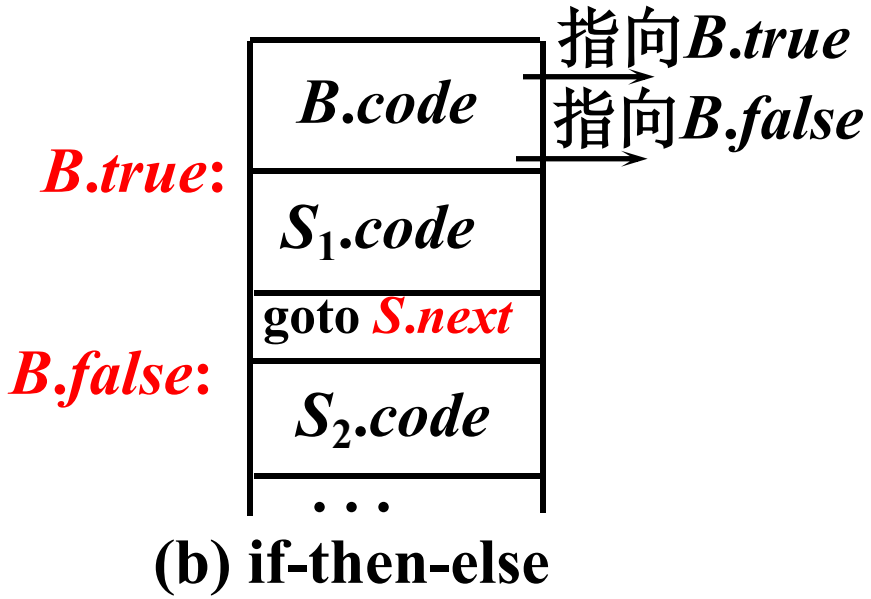


- 控制流语句文法
- **简单控制流语句的翻译**
  - if, if-then-else, while, 顺序语句
- **布尔表达式的翻译**

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

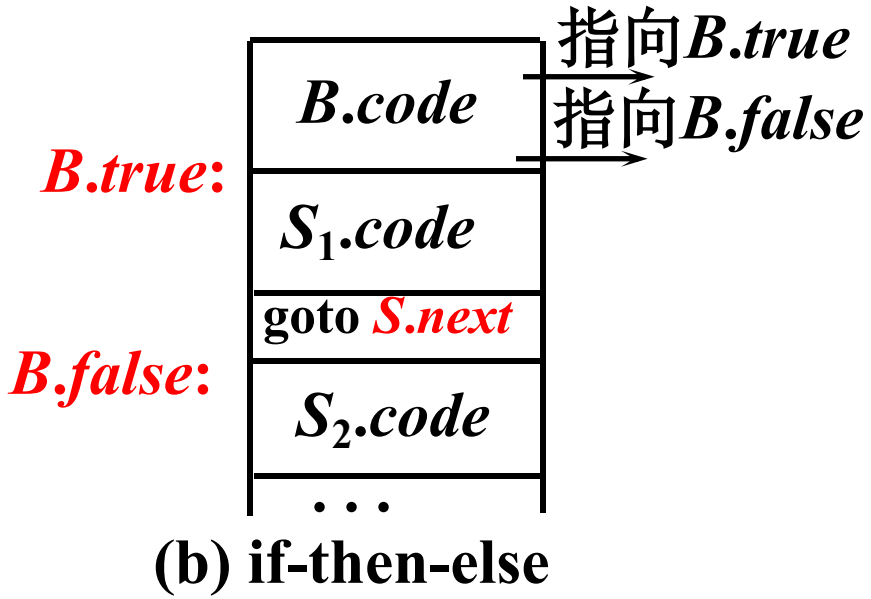
- 考虑带有else的语句

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

语义规则：

B.true = *newLabel*();

B.false = *newLabel*();



(b) if-then-else

if 语句中间代码生成的SDD

- 考虑带有else的语句

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

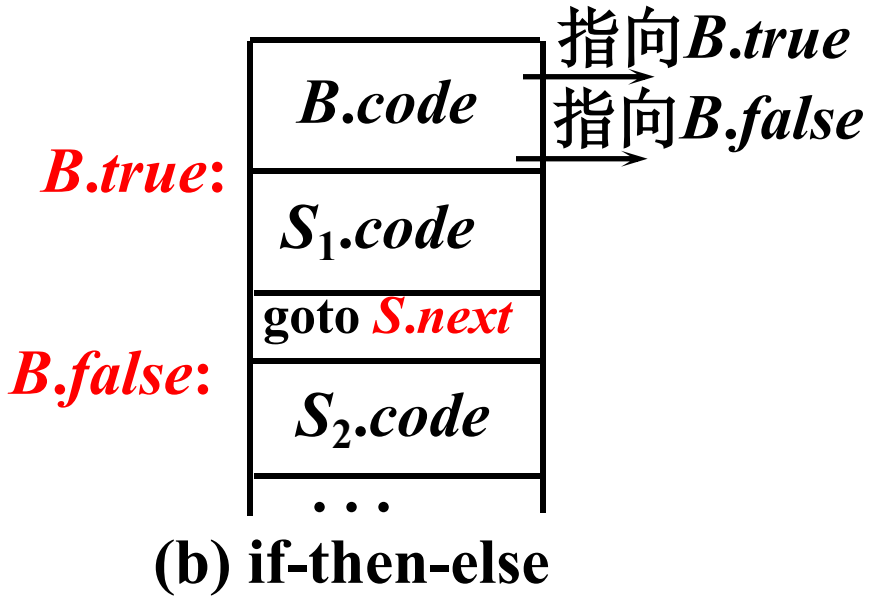
语义规则：

$B.true = newLabel();$

$B.false = newLabel();$

$S_1.next = S.next;$

$S_2.next = S.next;$



(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：

$B.true = newLabel();$

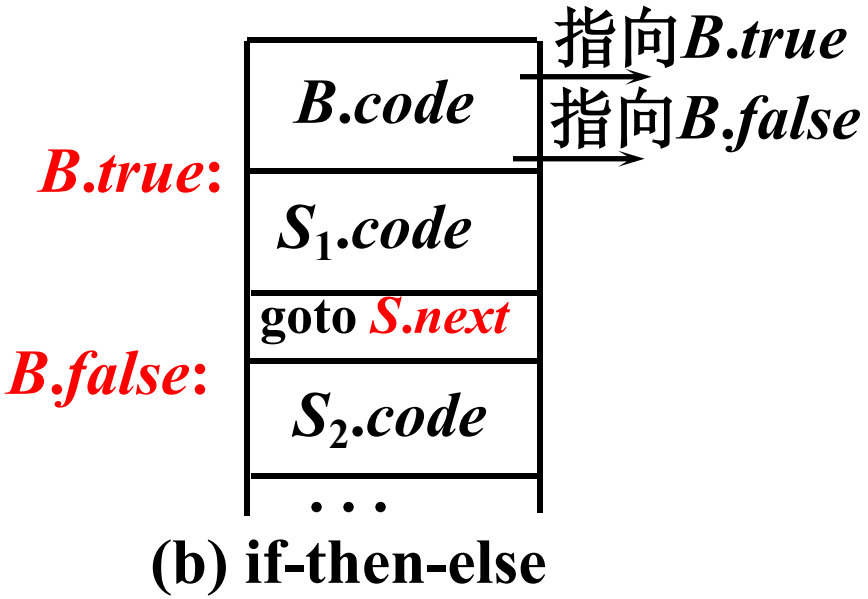
$B.false = newLabel();$

$S_1.next = S.next;$

$S_2.next = S.next;$

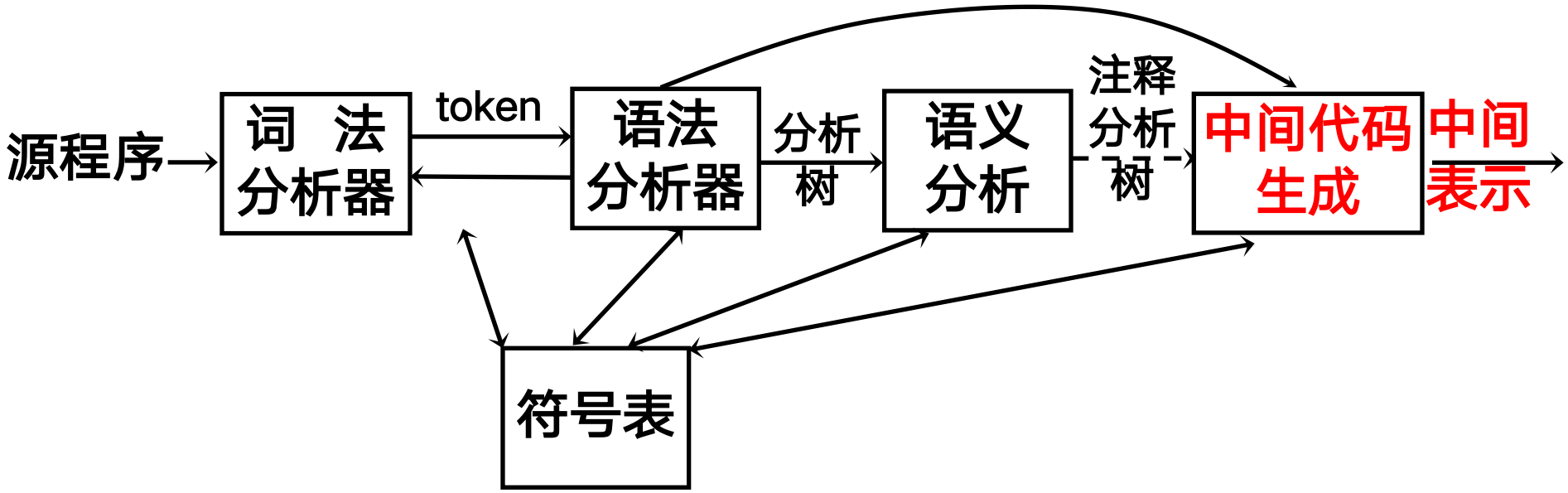
$S.code = B.code \,||\, gen(B.true, \text{':'}) \,||\, S_1.code \,||$

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



(b) if-then-else

本节提纲

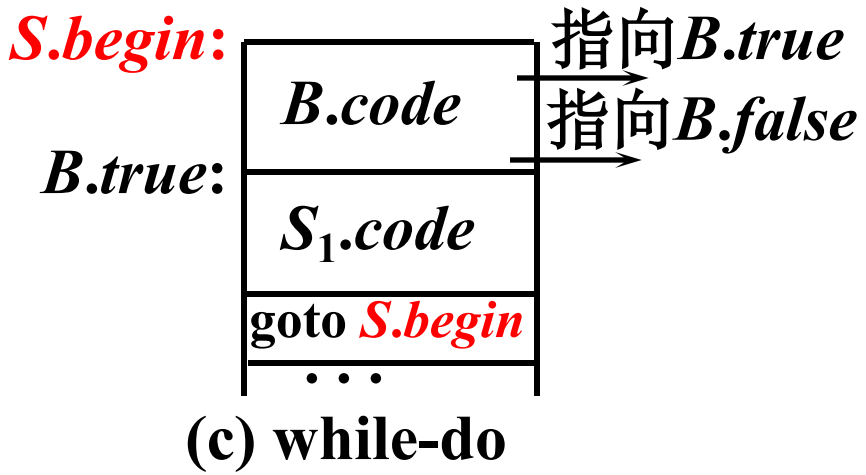


- 控制流语句文法
- **简单控制流语句的翻译**
  - if, if-then-else, while, 顺序语句
- **布尔表达式的翻译**

while语句中间代码生成SDD

- 引入开始标号*S.begin*,作为循环的跳转目标

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



while语句中间代码生成SDD

- 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 = *newLabel*();

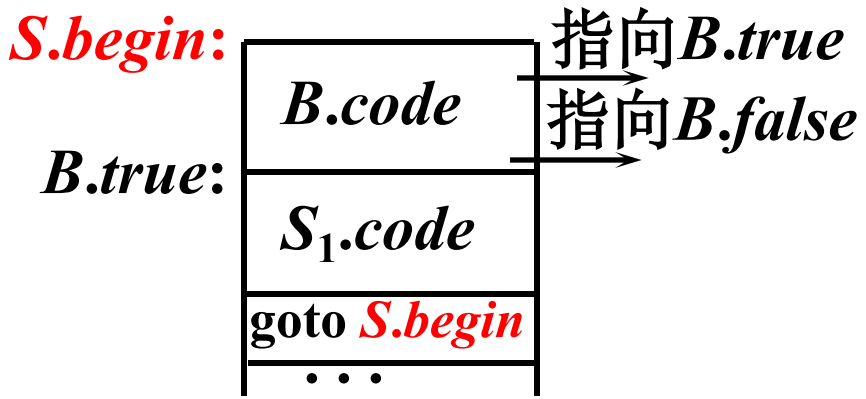
B.true = *newLabel*();

B.false = *S.next*;

$S_1.next = S.begin$;

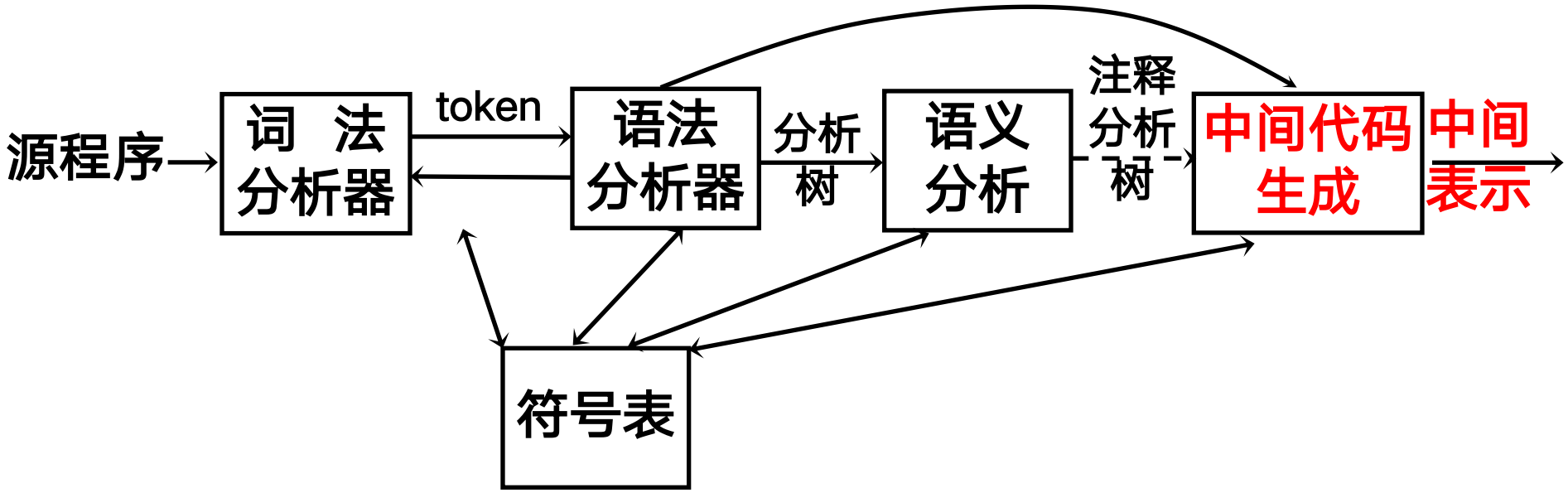
S.code = *gen*(*S.begin*, ':') || *B.code* ||

gen(*B.true*, ':') || $S_1.code$ || *gen*('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

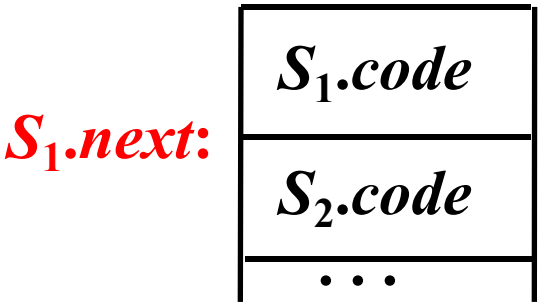
- 为每一语句S_1引入其后的下一条语句的标号$S_1.next$

$S \rightarrow S_1; S_2$

语义规则：

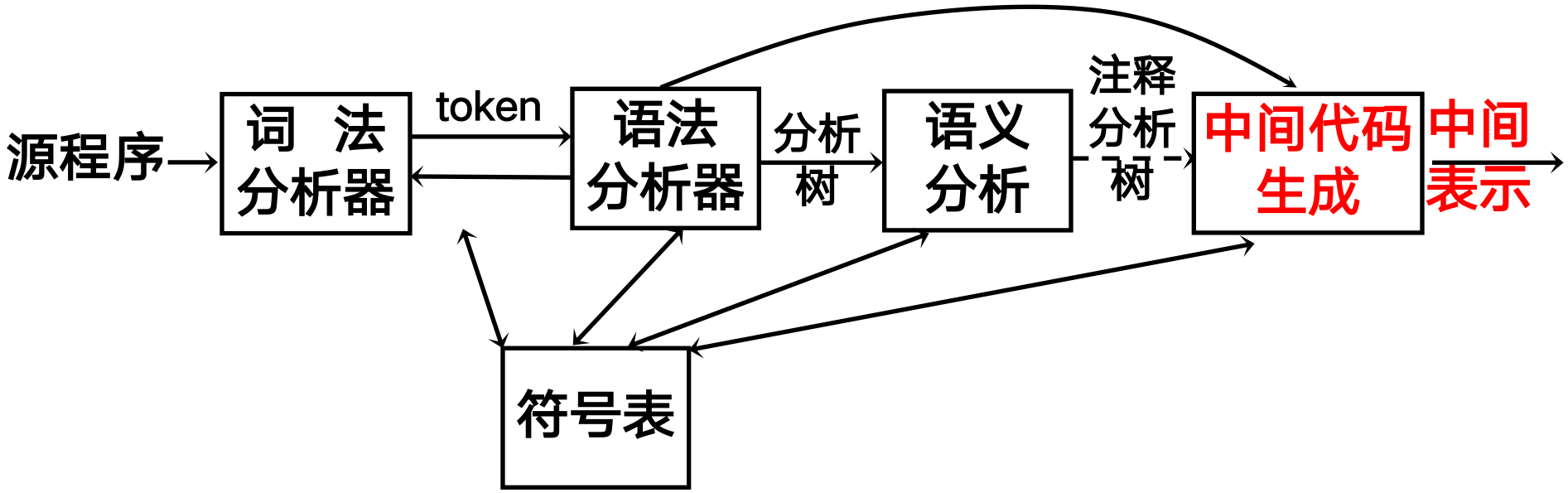
$S_1.next = newLabel(); S_2.next = S.next;$

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



(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**

布尔表达式的控制流翻译

- 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 $B.false$

我们把这种翻译称作为“布尔表达式的控制流翻译”

布尔表达式

- 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

- 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： $<$、$\leqslant$、$=$、$\neq$、$>$和$\geqslant$
  - 布尔常量： true和false

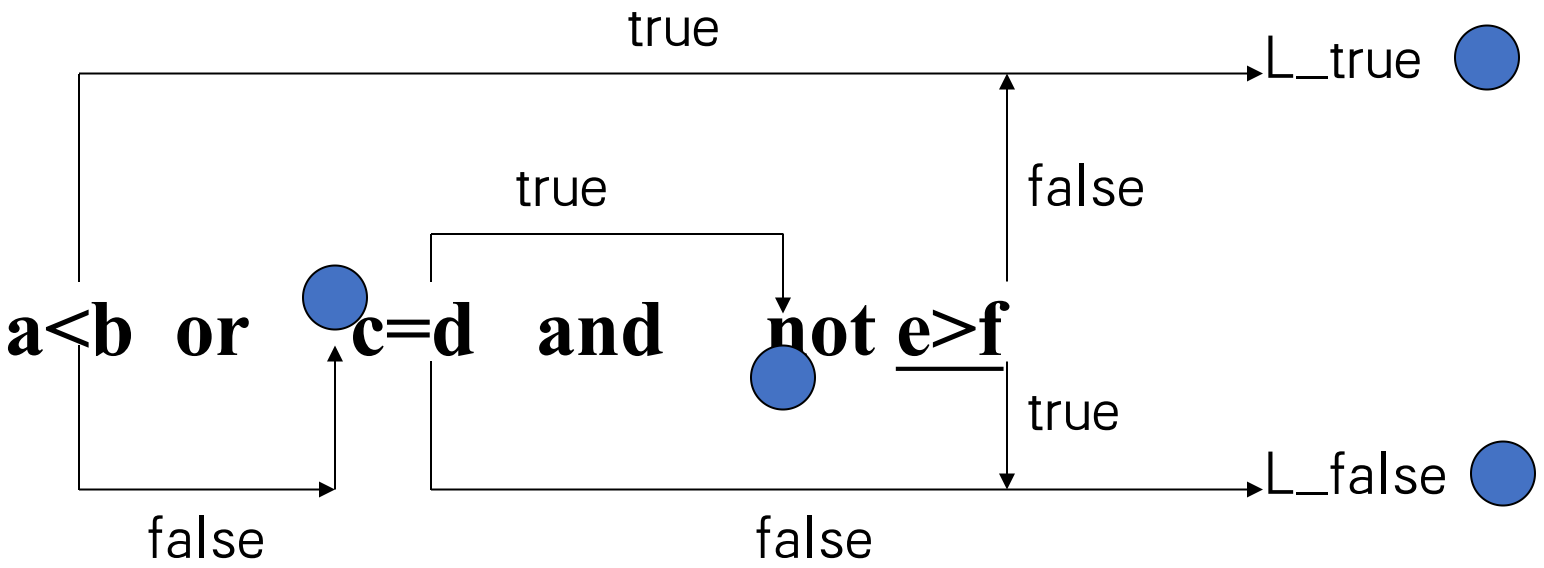
布尔表达式

- 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。

布尔表达式的控制流翻译

- **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!=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:
```

布尔表达式的控制流翻译

- 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}

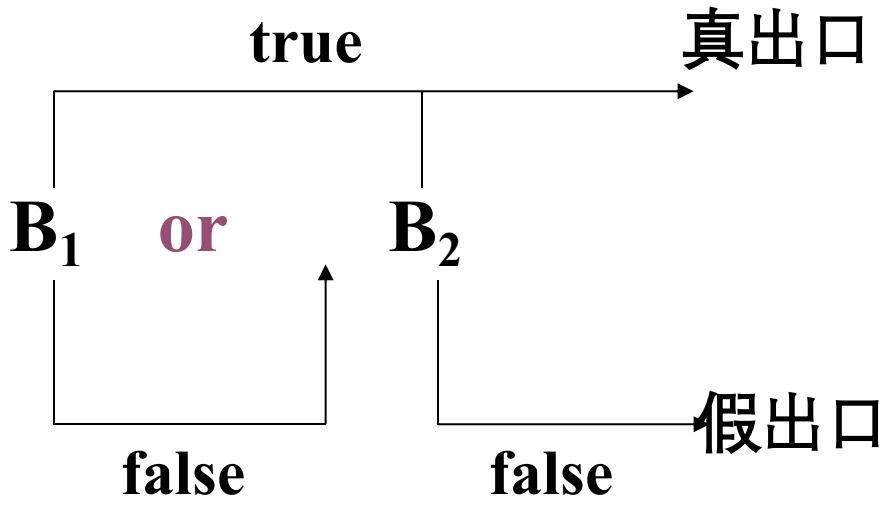
goto L_{false}

布尔表达式的控制流翻译

$B \rightarrow B_1$ or B_2

布尔表达式的控制流翻译

$B \rightarrow B_1$ or B_2



布尔表达式控制流翻译SDD

$B \rightarrow B_1$ or B_2

语义规则：

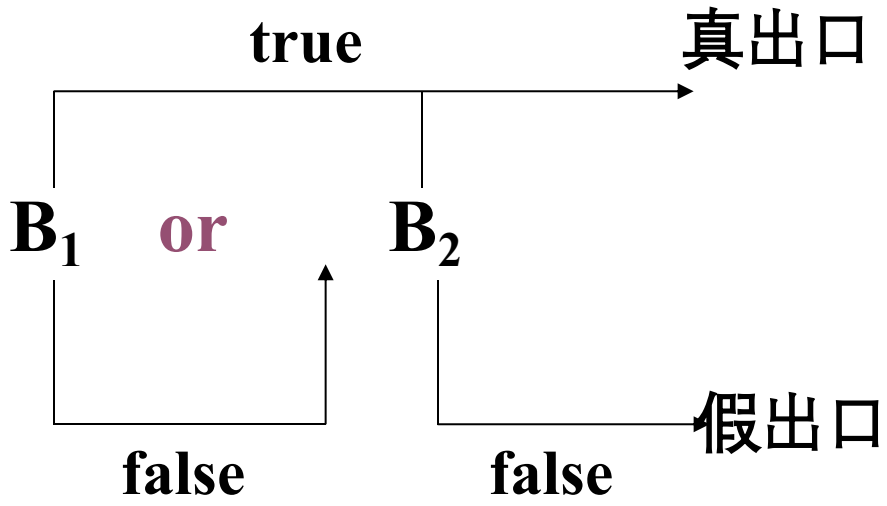
$B_1.true = B.true;$

$B_1.false = newLabel();$

$B_2.true = B.true;$

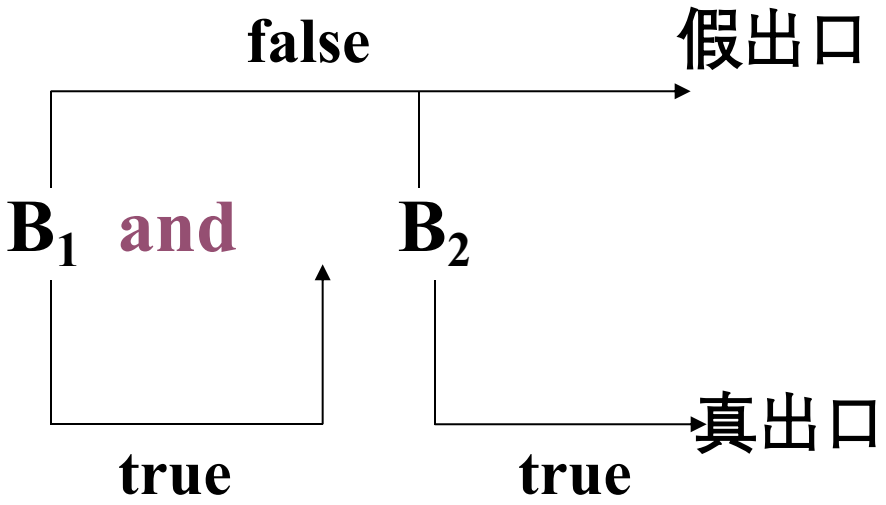
$B_2.false = B.false;$

$B.code = B_1.code \,||\, gen(B_1.false, ':') \,||\, B_2.code$



布尔表达式的控制流翻译

$B \rightarrow B_1$ and B_2



布尔表达式控制流翻译SDD

$B \rightarrow B_1$ and B_2

语义规则：

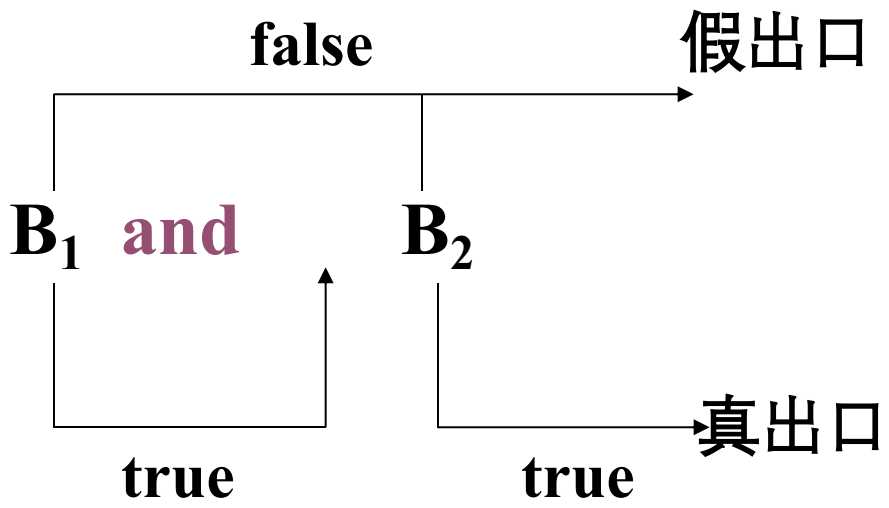
$B_1.true = newLabel();$

$B_1.false = B.false;$

$B_2.true = B.true;$

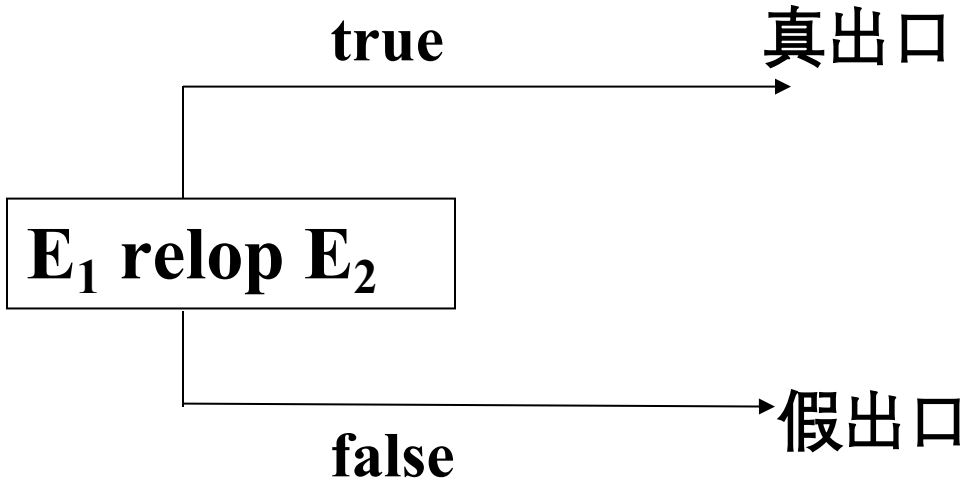
$B_2.false = B.false;$

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



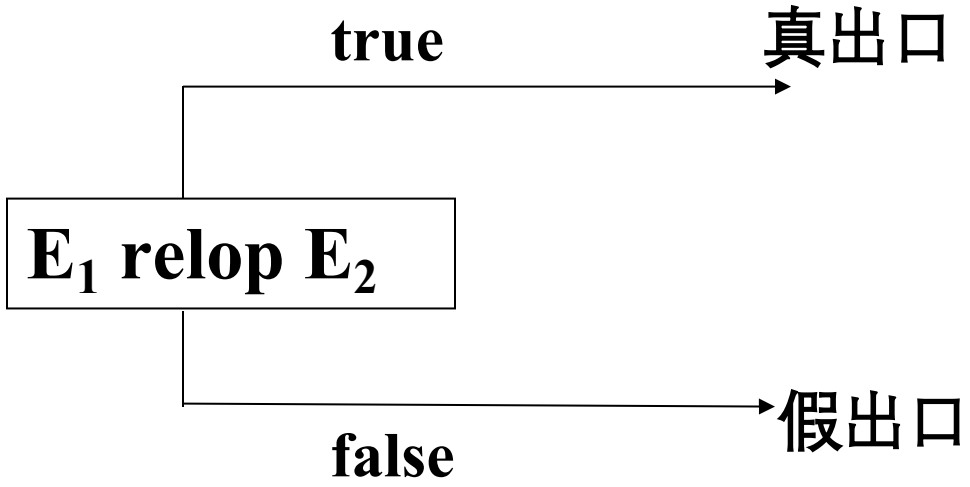
布尔表达式的控制流翻译

$B \rightarrow E_1$ relop E_2



布尔表达式控制流翻译SDD

$B \rightarrow E_1$ relop E_2



语义规则：

$B.code = E_1.code \,||\, E_2.code \,||$

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

$\text{‘goto’}, B.true) \,||$

$gen(\text{‘goto’}, B.false)$

布尔表达式控制流翻译SDD

$B \rightarrow (B_1)$

语义规则：

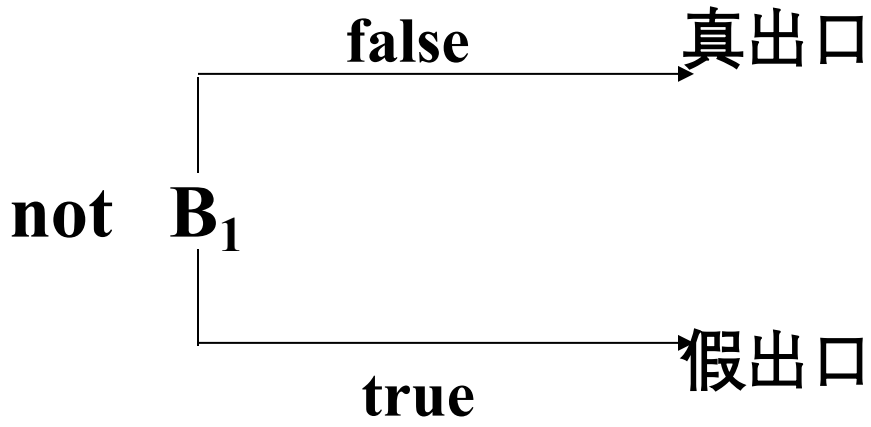
$B_1.true = B.true;$

$B_1.false = B.false;$

$B.code = B_1.code$

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

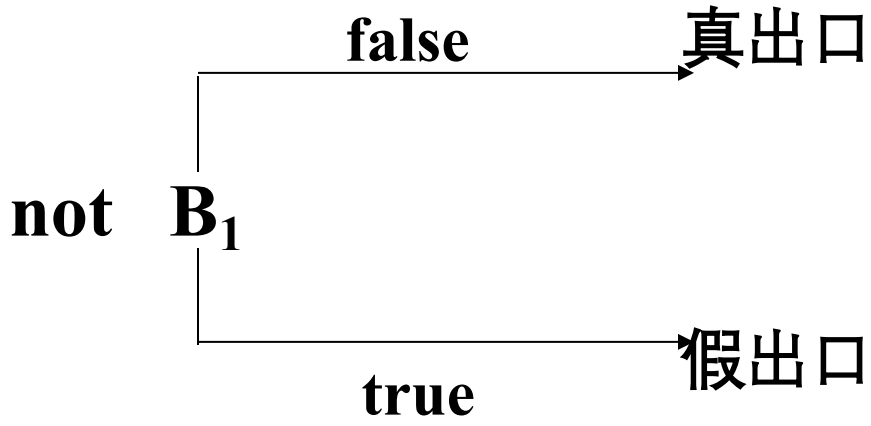
$B \rightarrow \text{not } B_1$

语义规则：

$B_1.true = B.false;$

$B_1.false = B.true;$

$B.code = B_1.code$



布尔表达式控制流翻译SDD

$B \rightarrow$ true

语义规则：

$B.code = gen(\text{‘goto’}, B.true)$

$B \rightarrow$ false

语义规则：

$B.code = gen(\text{‘goto’}, B.false)$

翻译难点

- 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
打造国产基础软硬件体系！

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03月27日