第六章 中间代码生成

控制流语句及其SDT

哈尔滨工业大学 陈鄞

控制流语句的基本文法

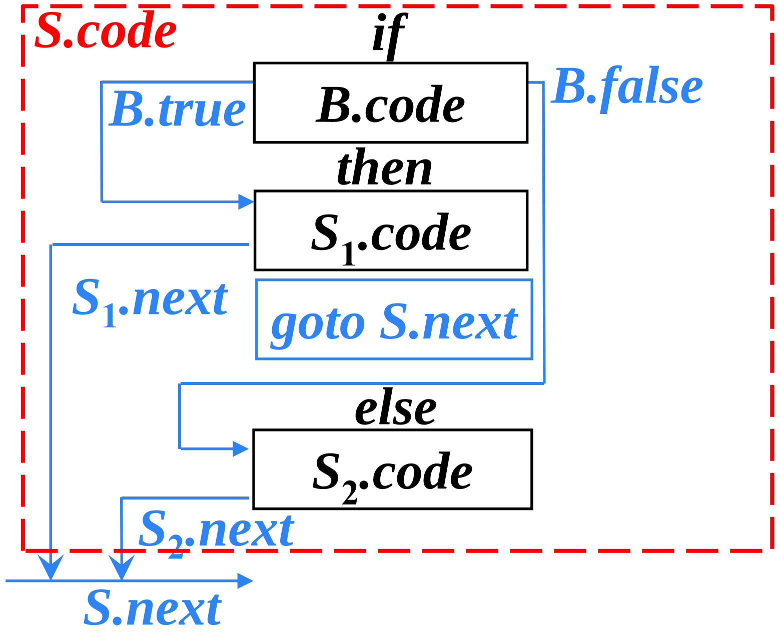
- $P \rightarrow S$
- $S \rightarrow S_1 S_2$
- $S \rightarrow \mathrm{id} = E\ ;\ |\ L = E\ ;$
- $S \rightarrow \mathrm{if}\ B\ \mathrm{then}\ S_1$

  $|\ \mathrm{if}\ B\ \mathrm{then}\ S_1\ \mathrm{else}\ S_2$

  $|\ \mathrm{while}\ B\ \mathrm{do}\ S_1$

控制流语句的代码结构

- 例 $S \rightarrow \textit{if } B \textit{ then } S_1 \textit{ else } S_2$



布尔表达式B被翻译成由跳转指令构成的跳转代码

- 继承属性
  - S.next：是一个地址，该地址中存放了紧跟在S代码之后的指令(S的后继指令)的标号
  - B.true：是一个地址，该地址中存放了当B为真时控制流转向的指令的标号
  - B.false：是一个地址，该地址中存放了当B为假时控制流转向的指令的标号

用指令的标号标识一条三地址指令

控制流语句的*SDT*

newlabel( ): 生成一个用于存放标号的新的临时变量*L*，返回变量地址

- $P \rightarrow$ { *S.next* = *newlabel*(); }*S*{ *label*(*S.next*);}
- $S \rightarrow$ { S_1*.next* = *newlabel*(); } S_1

  { *label*(S_1*.next*); S_2*.next*= *S.next* ; } S_2
- $S \rightarrow$ id = *E* ; | *L* = *E* ;
- $S \rightarrow$ *if B then* S_1

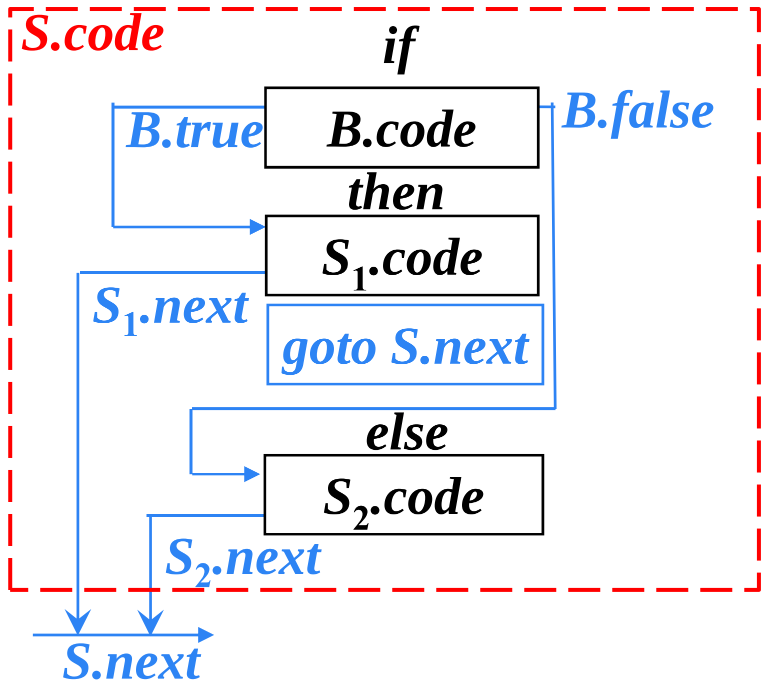
  | *if B then* S_1 *else* S_2

  | *while B do* S_1

label(*L*): 将下一条三地址指令的标号赋给*L*

*if-then-else*语句的*SDT*

$S \rightarrow if\ B\ then\ S_1\ else\ S_2$



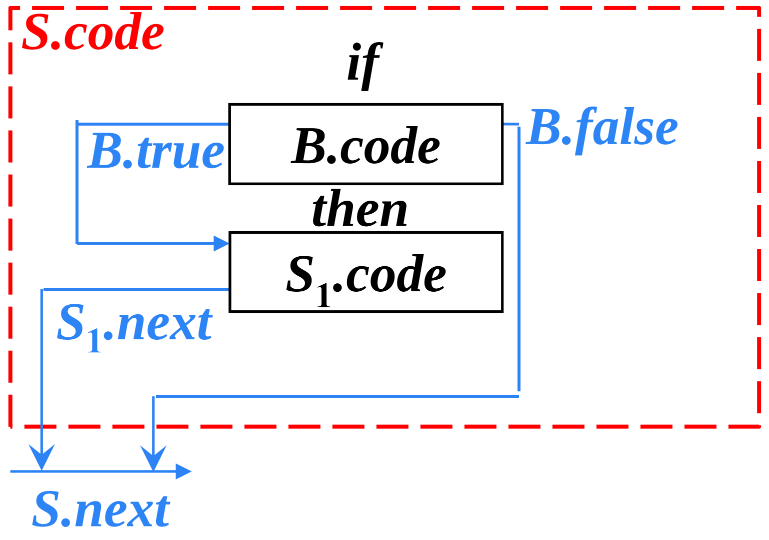
$S \rightarrow if$ { $B.true = newlabel(); B.false = newlabel();$ } B

$then$ { $label(B.true); S_1.next = S.next;$ } S_1 { $gen($ 'goto' $S.next)$ }

$else$ { $label(B.false); S_2.next = S.next;$ } S_2

*if-then*语句的*SDT*

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

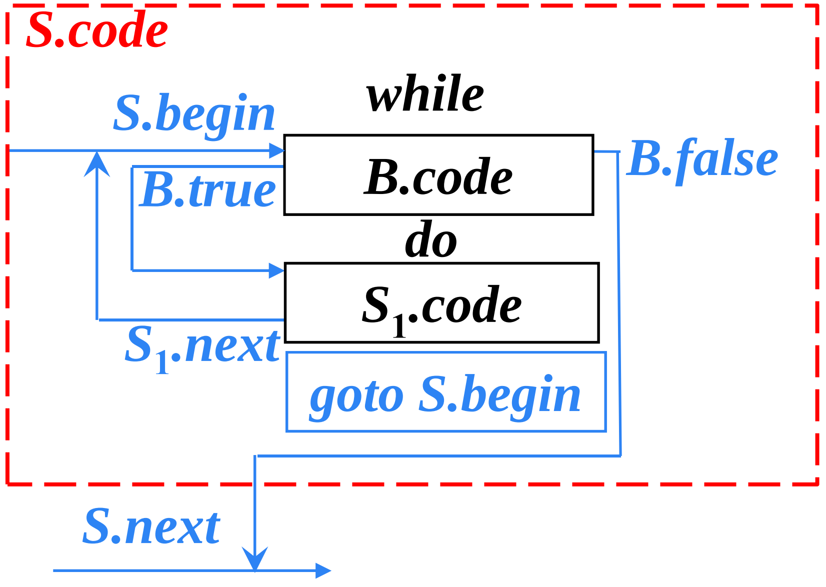


$S \rightarrow if$ { $B.true = newlabel(); B.false = S.next;$ } B

$then$ { $label(B.true); S_1.next = S.next;$ } S_1

*while-do*语句的*SDT*

$S \rightarrow while\ B\ do\ S_1$



$S \rightarrow while$ $\{S.begin = newlabel();$
$label(S.begin);$
$B.true = newlabel();$
$B.false = S.next;\}$ B
do $\{\ label(B.true);\ S_1.next = S.begin;\}$ S_1
$\{\ gen(\text{'goto'}\ S.begin);\ \}$

第六章 中间代码生成

控制流语句及其SDT

哈尔滨工业大学 陈鄞

第六章 中间代码生成

布尔表达式及其SDT

哈尔滨工业大学 陈鄞

布尔表达式的基本文法

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

优先级：not > and > or

关系表达式

relop（关系运算符）：

<, <=, >, >=， ==, ! =

- 在跳转代码中，逻辑运算符&&、|| 和！被翻译成跳转指令。运算符本身不出现在代码中，布尔表达式的值是通过代码序列中的位置来表示的
- 例
  - 语句
  - 三地址代码

```
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 :
```

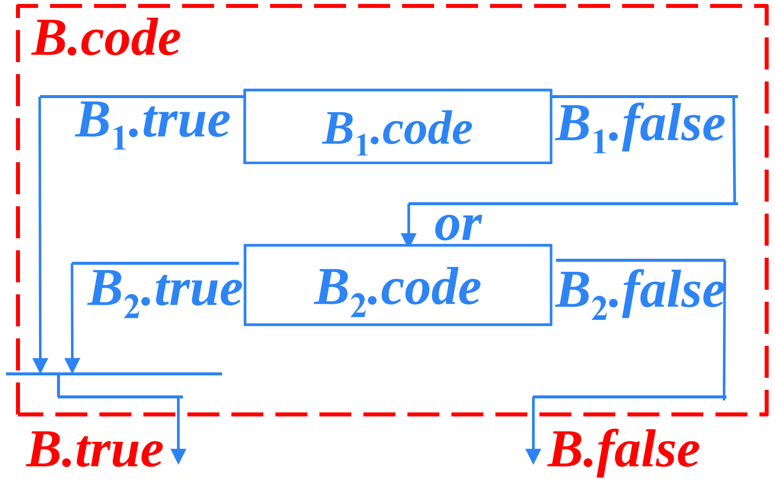
布尔表达式的*SDT*

- $B \rightarrow E_1$ relop E_2 { *gen*('if ' E_1.*addr* *relop* E_2.*addr* 'goto' *B.true*);
  gen('goto' *B.false*); }
- $B \rightarrow$ true { *gen*('goto' *B.true*); }
- $B \rightarrow$ false { *gen*('goto' *B.false*); }
- $B \rightarrow$ ({B_1.*true* =*B. true*; B_1.*false* =*B.false*; } B_1)
- $B \rightarrow$ not { B_1.*true* = *B.false*; B_1.*false* = *B.true*; } B_1

$B \rightarrow B_1$ or B_2 的*SDT*

- $B \rightarrow B_1 \text{ or } B_2$
  - $B \rightarrow \{\ B_1.true = B.true;\ \ B_1.false = newlabel();\ \}B_1$

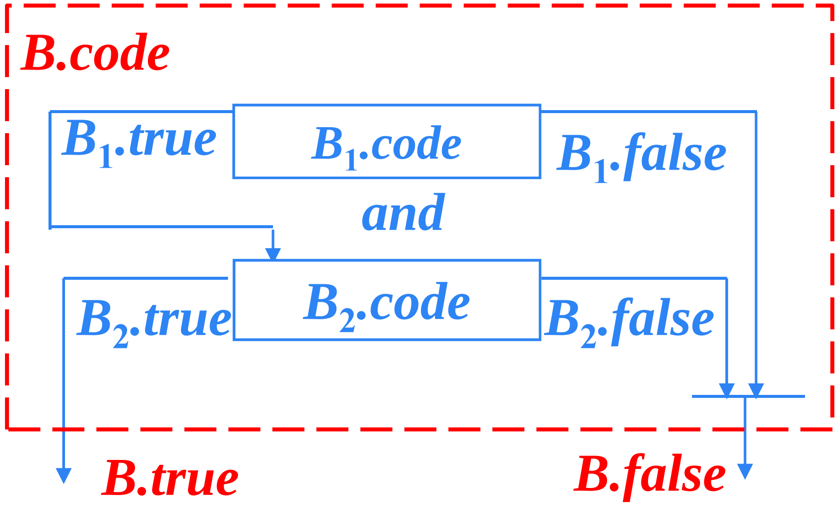
    or $\{\ label(B_1.false);\ \ B_2.true = B.true;\ \ B_2.false = B.false;\ \}B_2$



$B \rightarrow B_1$ and B_2 的*SDT*

- $B \rightarrow B_1 \textit{ and } B_2$
  - $B \rightarrow \{\ B_1.true = newlabel();\ B_1.false = B.false;\ \}\ B_1$

    $\textit{and}\ \{\ label(B_1.true);\ B_2.true = B.true;\ B_2.false = B.false;\ \} B_2$



第六章 中间代码生成

布尔表达式及其SDT

哈尔滨工业大学 陈鄞

第六章 中间代码生成

控制流翻译的例子

哈尔滨工业大学 陈鄞

控制流语句的*SDT*

- $P \rightarrow$ {a}S{a}
- $S \rightarrow$ {a}S_1{a}S_2
- $S \rightarrow$ id=E;{a} | L=E;{a}
- $E \rightarrow E_1$+E_2{a} | –E_1{a} | (E_1){a}| id{a}| L{a}
- $L \rightarrow$ id[E]{a} | L_1[E]{a}
- $S \rightarrow$ if {a}B then {a}S_1

  | if {a}B then {a}S_1 else {a}S_2

  | while {a}B do {a}S_1{a}
- $B \rightarrow$ {a}B or {a}B | {a}B and {a}B | not {a}B | ({a}B)

  | E relop E{a} | true{a} | false{a}

*SDT*的通用实现方法

- 任何*SDT*都可以通过下面的方法实现
  - 首先建立一棵语法分析树，然后按照从左到右的深度优先顺序来执行这些动作

控制流语句的SDT

- $P \rightarrow \{a\}S\{a\}$
- $S \rightarrow \{a\}S_1\{a\}S_2$
- $S \rightarrow$ id=E;$\{a\}$ | L=E;$\{a\}$
- $E \rightarrow E_1+E_2\{a\} \mid -E_1\{a\} \mid (E_1)\{a\} \mid$ id$\{a\}$ | $L\{a\}$
- $L \rightarrow$ id[E]$\{a\}$ | L_1[E]$\{a\}$
- $S \rightarrow$ if $\{a\}B$ then $\{a\}S_1$

  | if $\{a\}B$ then $\{a\}S_1$ else $\{a\}S_2$

  | while $\{a\}B$ do $\{a\}S_1\{a\}$
- $B \rightarrow \{a\}B$ or $\{a\}B$ | $\{a\}B$ and $\{a\}B$ | not $\{a\}B$ | ($\{a\}B$)

  | E relop $E\{a\}$ | true$\{a\}$ | false$\{a\}$

例

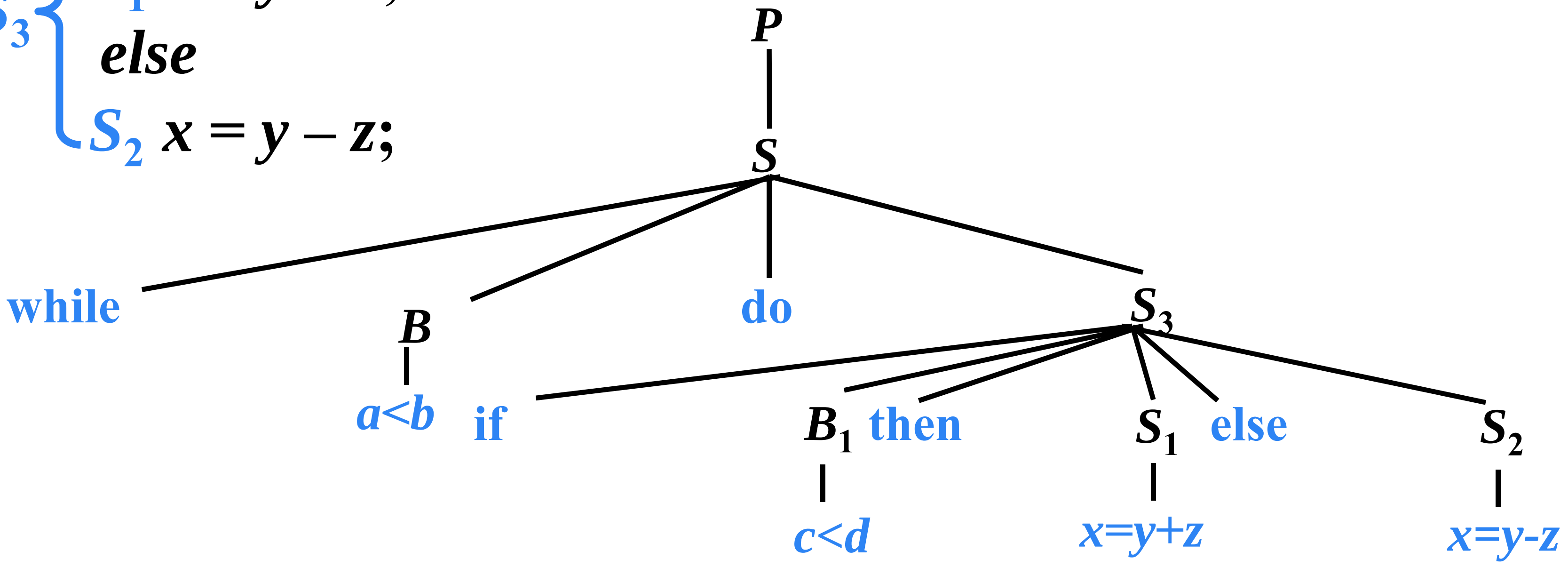
while $\overset{B}{\boxed{a < b}}$ *do*

S_3 ⎧ *if* $\overset{B_1}{\boxed{c < d}}$ *then*

⎪ S_1 $x = y + z;$

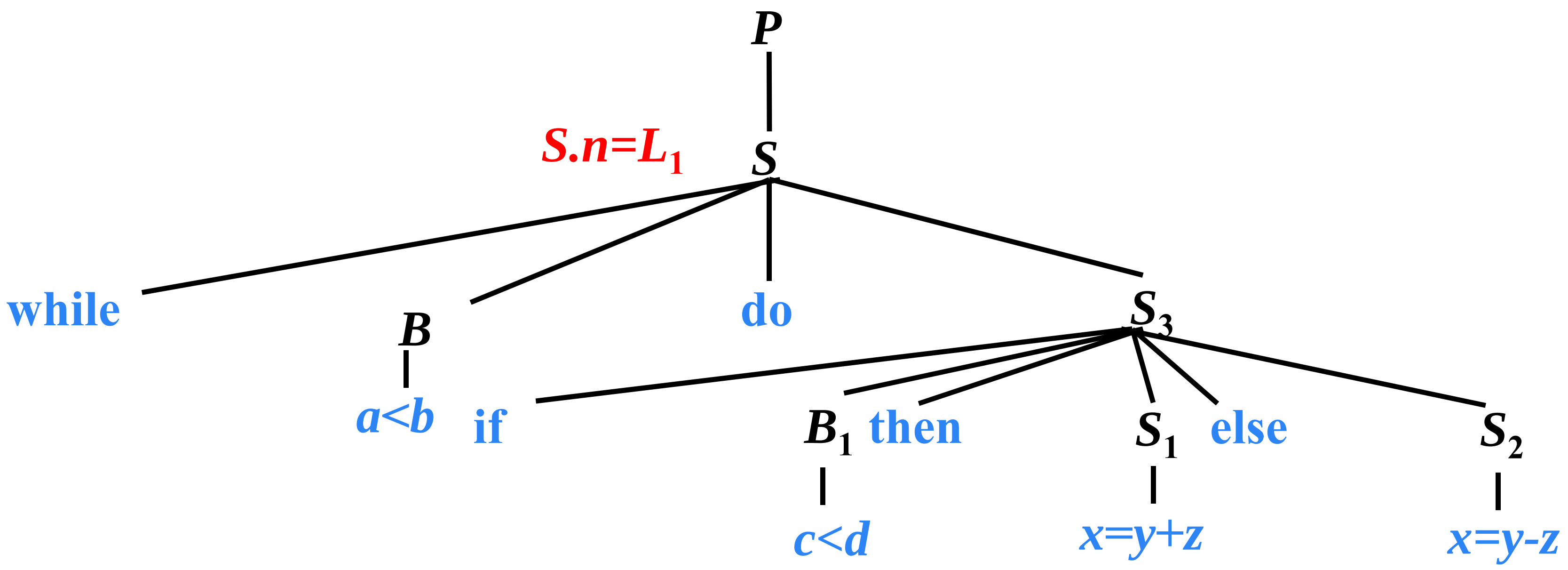
⎨ *else*

⎩ S_2 $x = y - z;$



例

$P \rightarrow \{\ S.next = newlabel(\ );\ \}S\{\ label(S.next);\}$

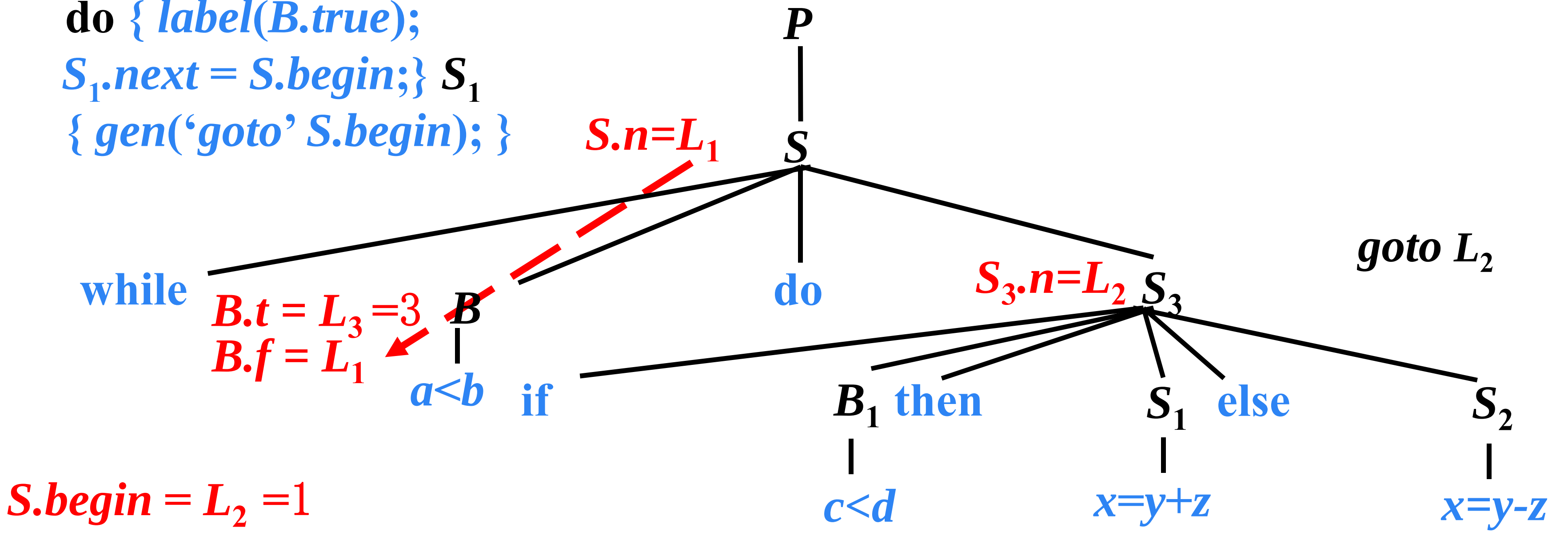


例

$S \rightarrow$ while $\{S.begin = newlabel(\,);$
$label(S.begin);$
$B.true = newlabel(\,);$
$B.false = S.next;\}$ B
do $\{\ label(B.true);$
$S_1.next = S.begin;\}$ S_1
$\{\ gen(\text{‘goto’}\ S.begin);\ \}$

$B \rightarrow E_1$ relop E_2
$\{\ gen(\text{‘if ’}\ E_1.addr\ relop\ E_2.addr\ \text{‘goto’}\ \ B.true);$
$gen(\text{‘goto ’}\ B.false);\ \}$

1: *if* $a < b$ *goto* L_3
2: *goto* L_1



例

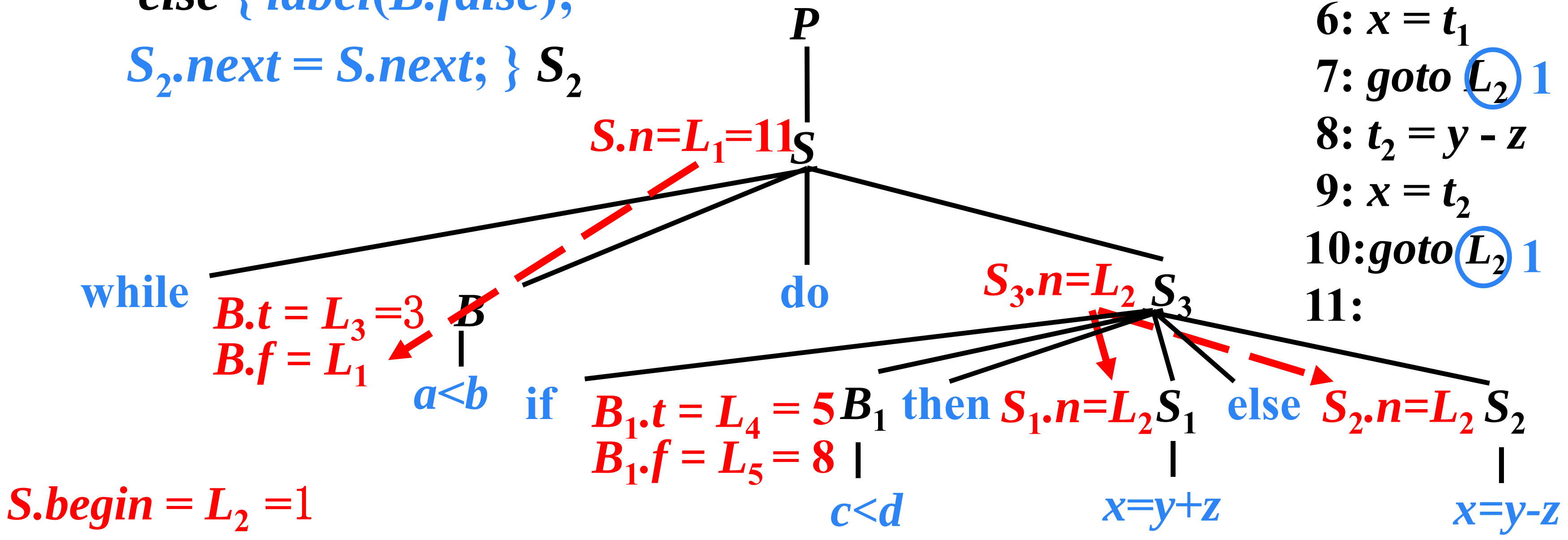
$S \rightarrow$ *if* { *B.true* = *newlabel*( ); *B.false* = *newlabel*( ); } *B*

then { *label*(*B.true*); S_1*.next* = *S.next*; } S_1

{ *gen*(‘*goto*’ *S.next*); }

else { *label*(*B.false*);

S_2*.next* = *S.next*; } S_2



1: *if a* < *b goto* $\boxed{L_3}$ 3

2: *goto* $\boxed{L_1}$ 11

3: *if c* < *d goto* $\boxed{L_4}$ 5

4: *goto* $\boxed{L_5}$ 8

5: $t_1 = y + z$

6: $x = t_1$

7: *goto* $\boxed{L_2}$ 1

8: $t_2 = y - z$

9: $x = t_2$

10: *goto* $\boxed{L_2}$ 1

11:

语句“*while* a<b *do if* c<d *then* x=y+z *else* x=y-z”的三地址代码

1: *if* $a < b$ *goto* 3	1: ($j<$, a, b , 3)
2: *goto* 11	2: (j , - , - , 11)
3: *if* $c < d$ *goto* 5	3: ($j<$, c, d , 5)
4: *goto* 8	4: (j , - , - , 8)
5: $t_1 = y + z$	5: (+ , y , z , t_1)
6: $x = t_1$	6: (= , t_1, - , x)
7: *goto* 1	7: (j , - , - , 1)
8: $t_2 = y - z$	8: (- , y, z , t_2)
9: $x = t_2$	9: (= , t_2, - , x)
10: *goto* 1	10: (j , - , - , 1)
11:	11:

第六章 中间代码生成

控制流翻译的例子

哈尔滨工业大学 陈鄞