6.6 控制流: 布尔表达式

- 布尔表达式通过布尔运算符将布尔变量或关系表达式结合在一起。
- 布尔表达式的文法:

$B \rightarrow B \,||\, B \mid B \,\&\&\, B \mid !B \mid (B) \mid E \text{ rel } E \mid \text{true} \mid \text{false}$

$\text{rel.op} \in \{<, \leq, =, \neq, >, \geq\}$

布尔表达式

- 假设:
  - ||, && 左结合
  - 从低到高的优先顺序为: ||, &&, !
- 翻译时采用短路代码（短路逻辑）:
  - A or B : 若A为真，则B无需计算
  - A and B : 若A为假，则B无需计算

```
! a || b && c || d
```

什么情况下，无需计算c?
什么情况下，无需计算d?

短路代码

- 布尔运算符翻译成跳转指令：
  - 例：翻译if(x<100 || x >200 && x!=y) x=0;

if x<100 goto L_2

ifFalse x>200 goto L_1

ifFalse x!=y goto L_1

L_2: x=0

L_1:

用短路代码翻译：

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

控制流翻译

❖ 文法:

$S \rightarrow$ if B then S_1

| if B then S_1 else S_2

| while B do S_1

❖ 布尔表达式*B*附加两个属性:

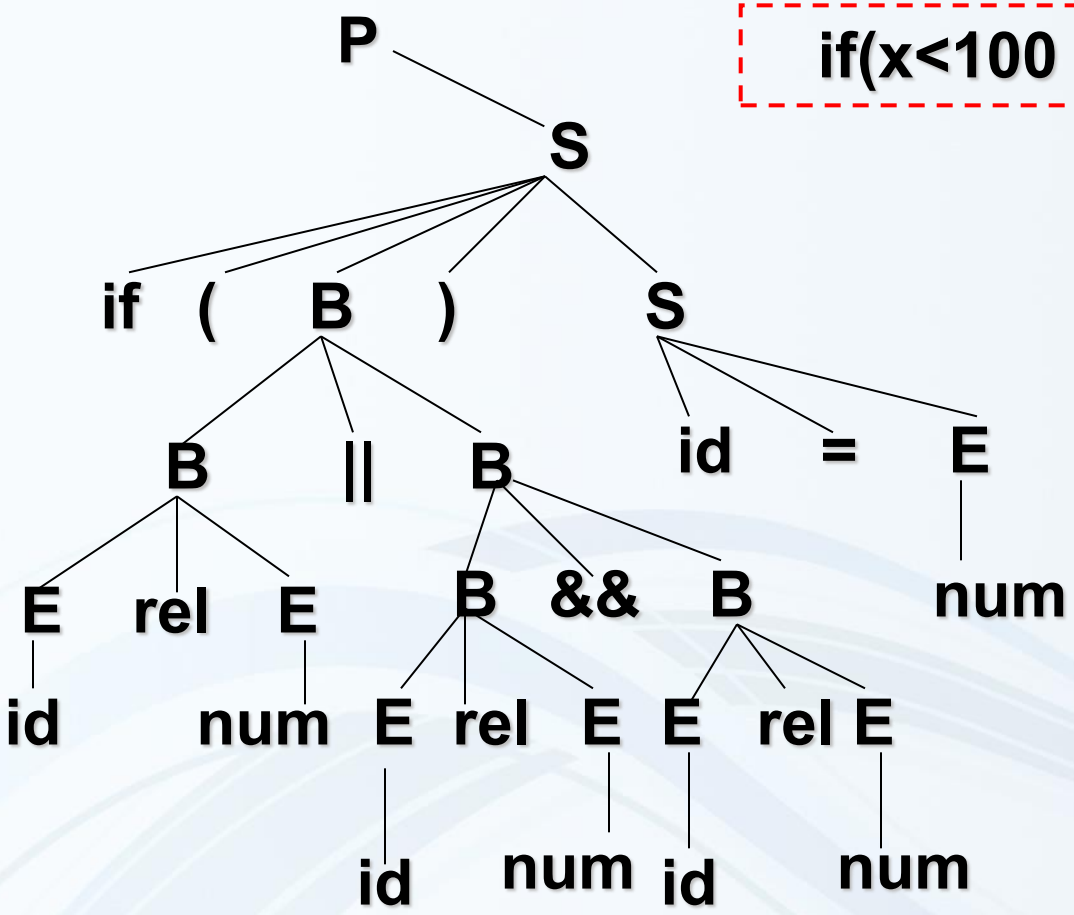
B.true： B为真时的控制流

B.false： B为假时的控制流

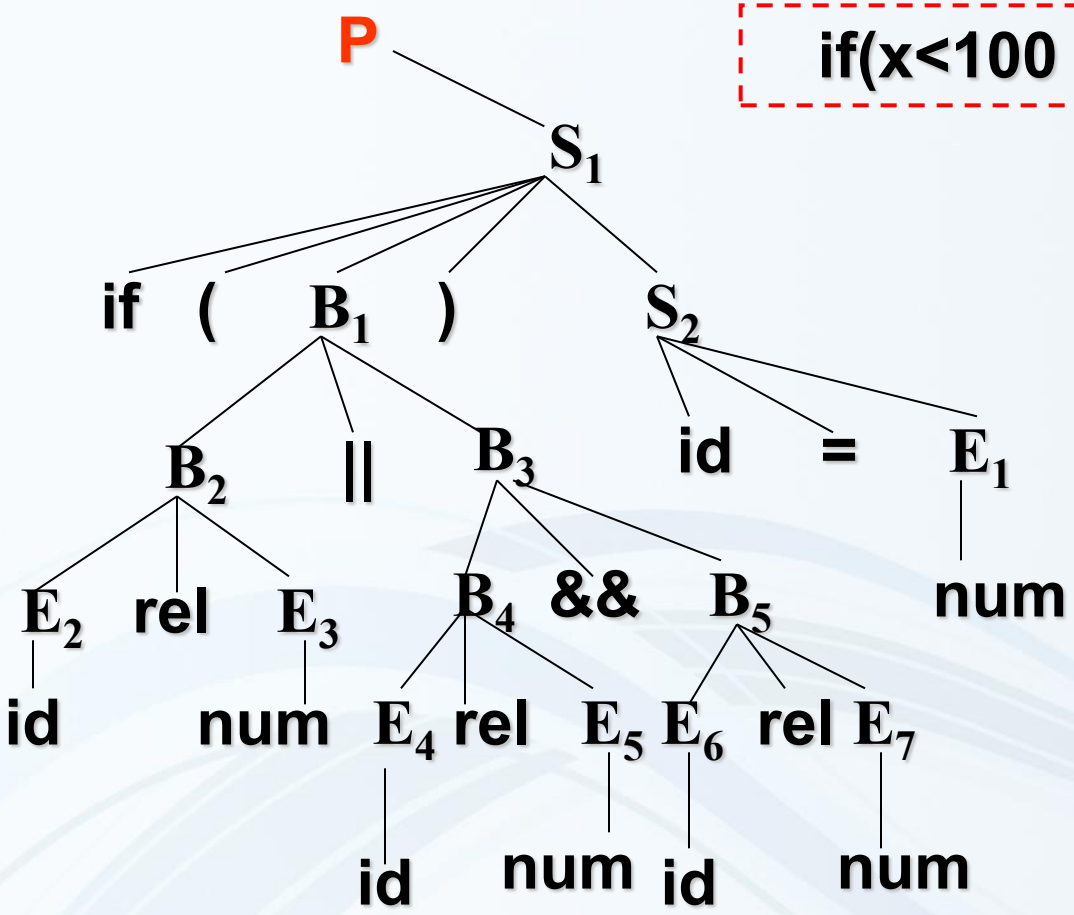
❖ *S.next* 指向紧跟在S后执行的第一条三地址指令

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

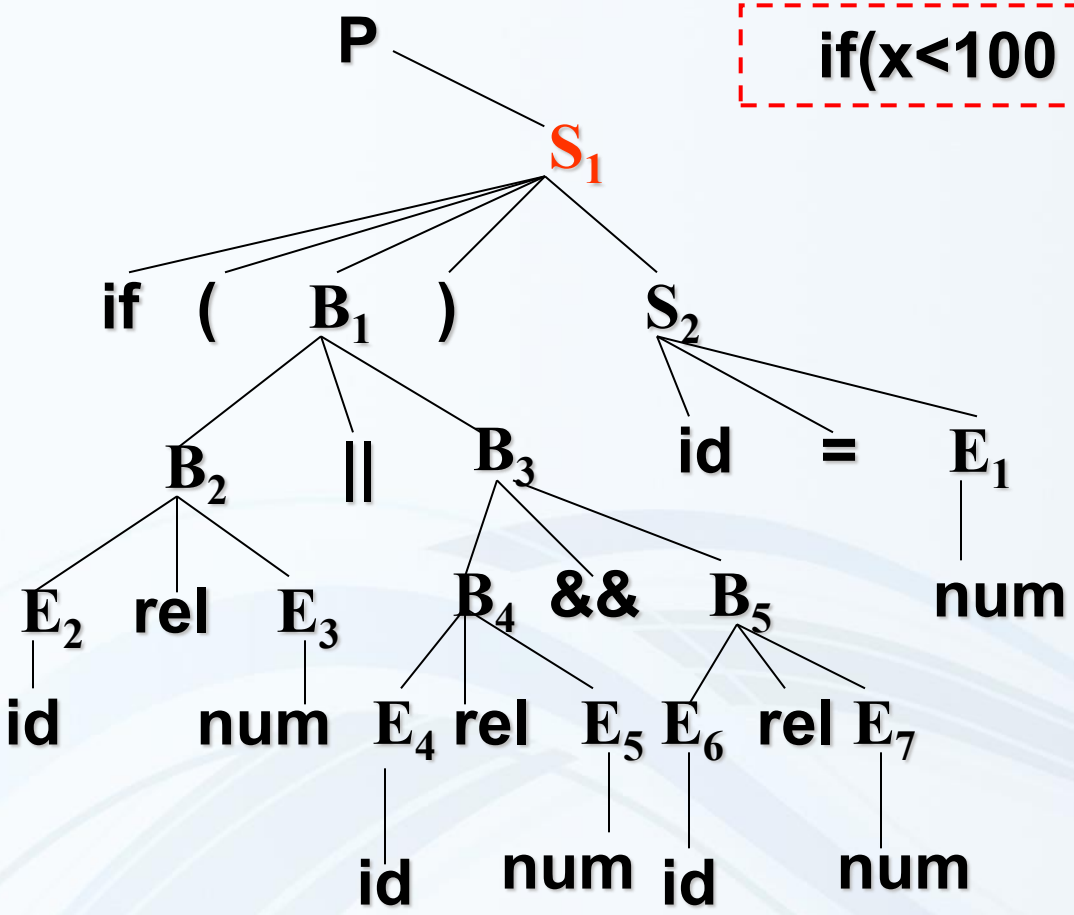
if(x<100 || x>200 && x!=y) x=0;



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

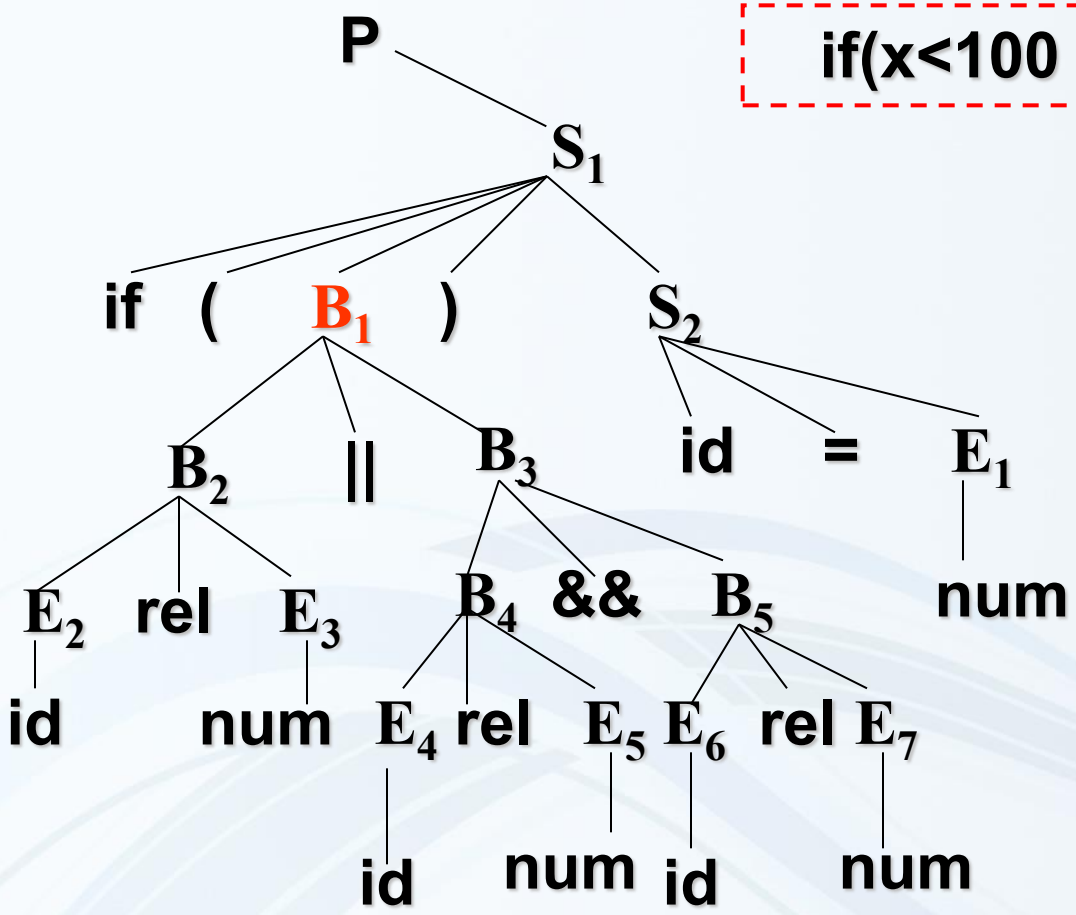


if(x<100 || x>200 && x!=y) x=0;



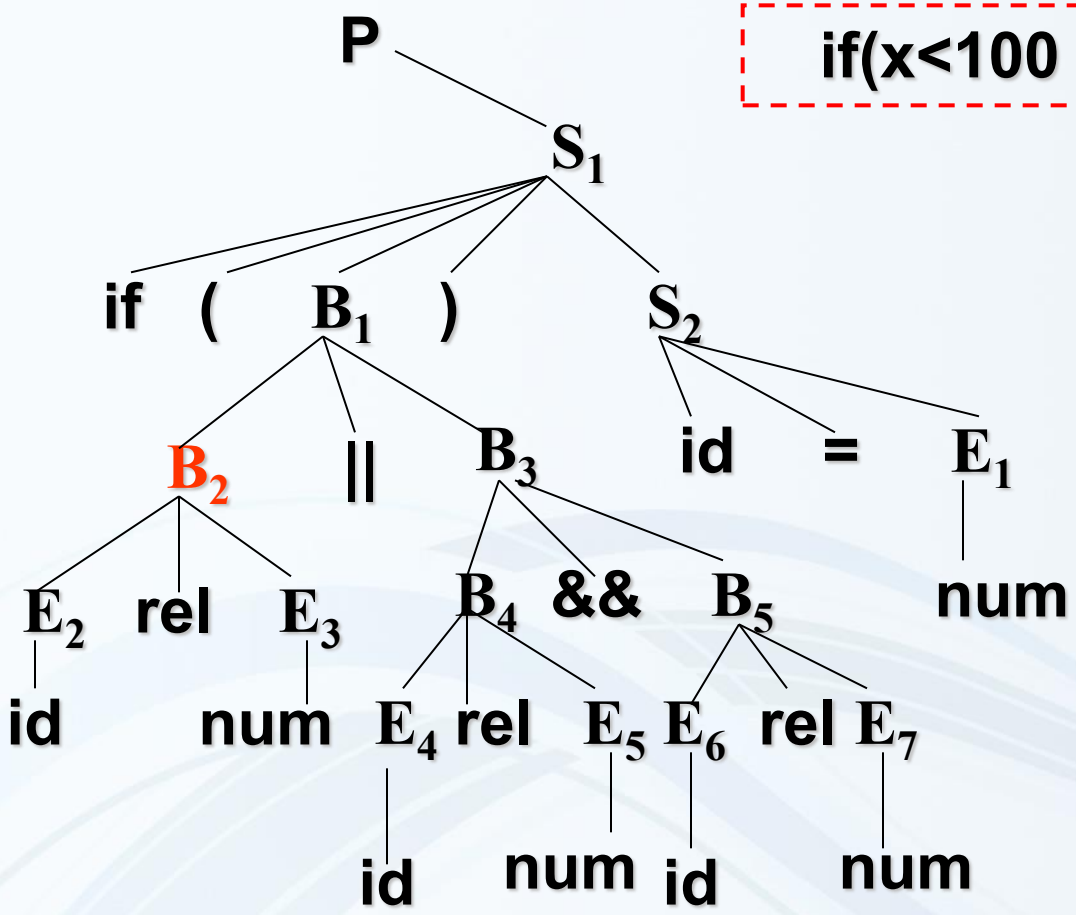
$S_1.next=L_1$;

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

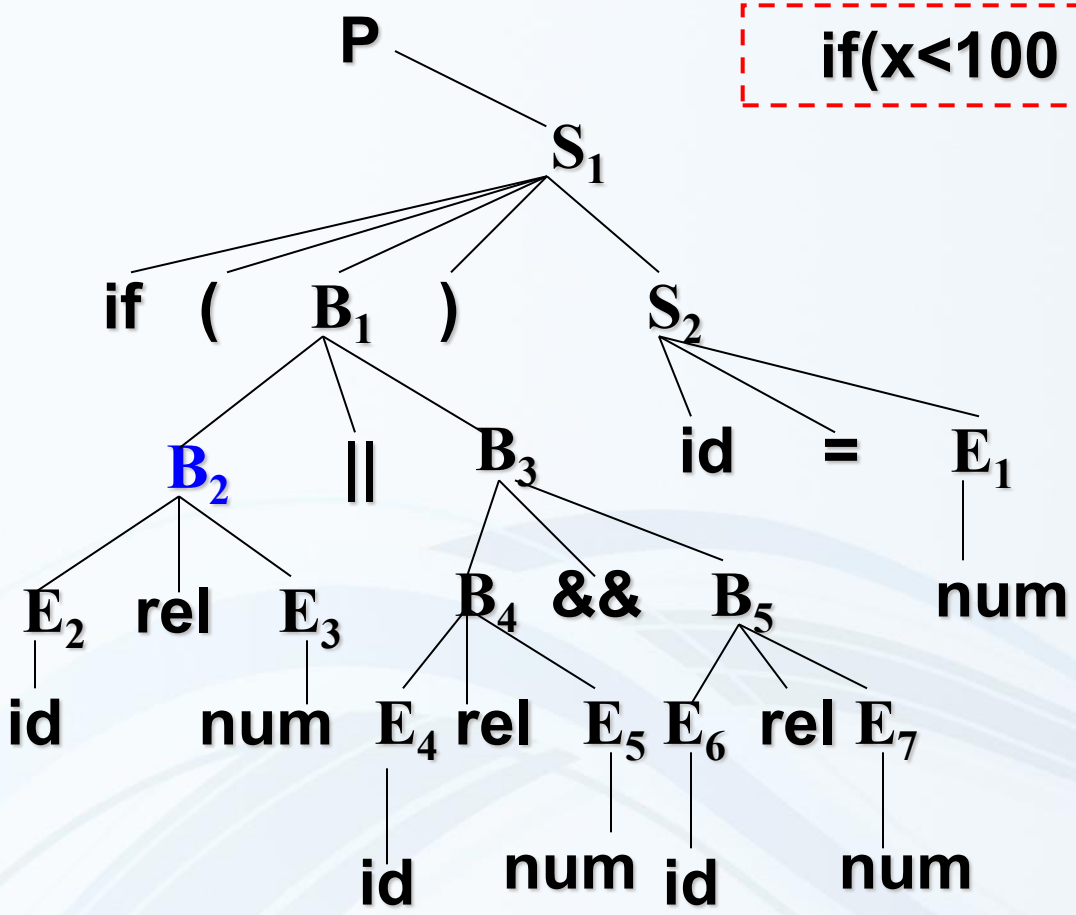


$S_1.next=L_1$; $B_1.true=L_2$; $B_1.false=L_1$;

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



$S_1.next=L_1$;$B_1.true=L_2$; $B_1.false=L_1$;$B_2.true=L_2$;$B_2.false=L_3$;

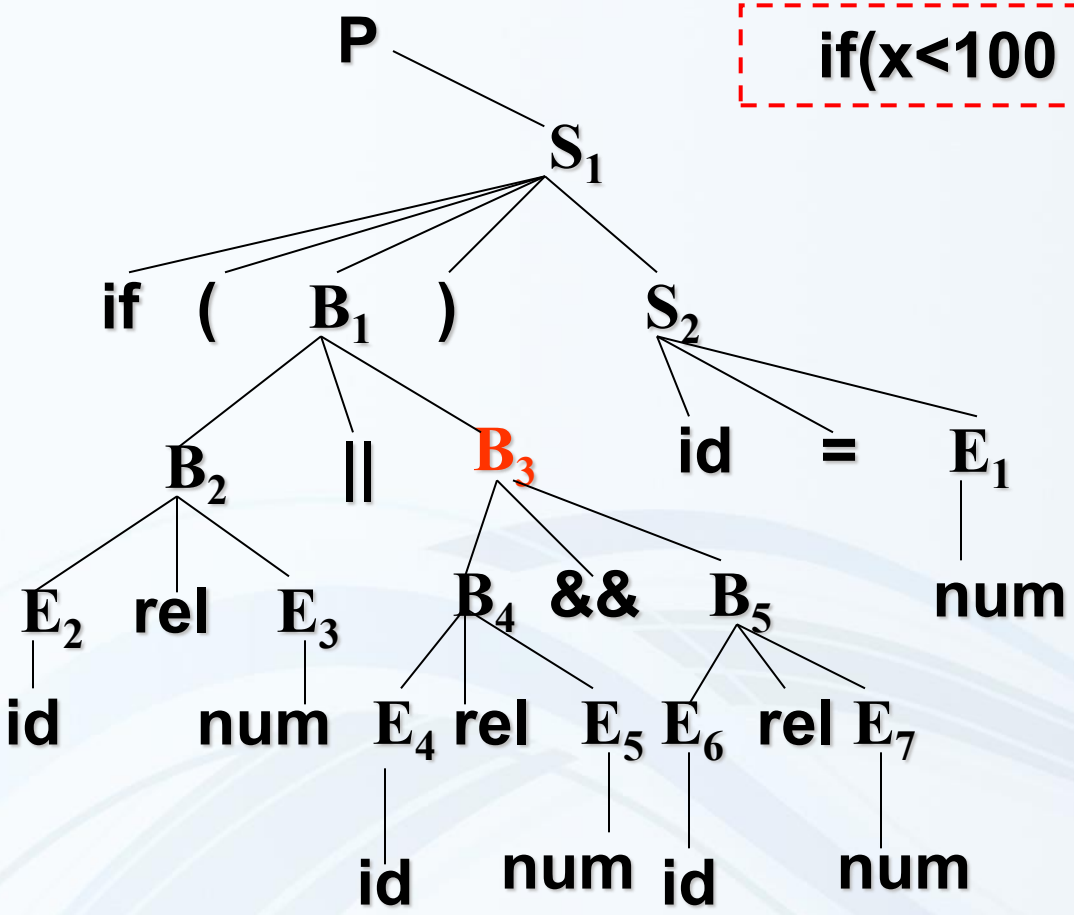


if(x<100 || x>200 && x!=y) x=0;

if(x<100) goto L2

goto L3

$S_1.next=L_1$;$B_1.true=L_2$; $B_1.false=L_1$;$B_2.true=L_2$;$B_2.false=L_3$;

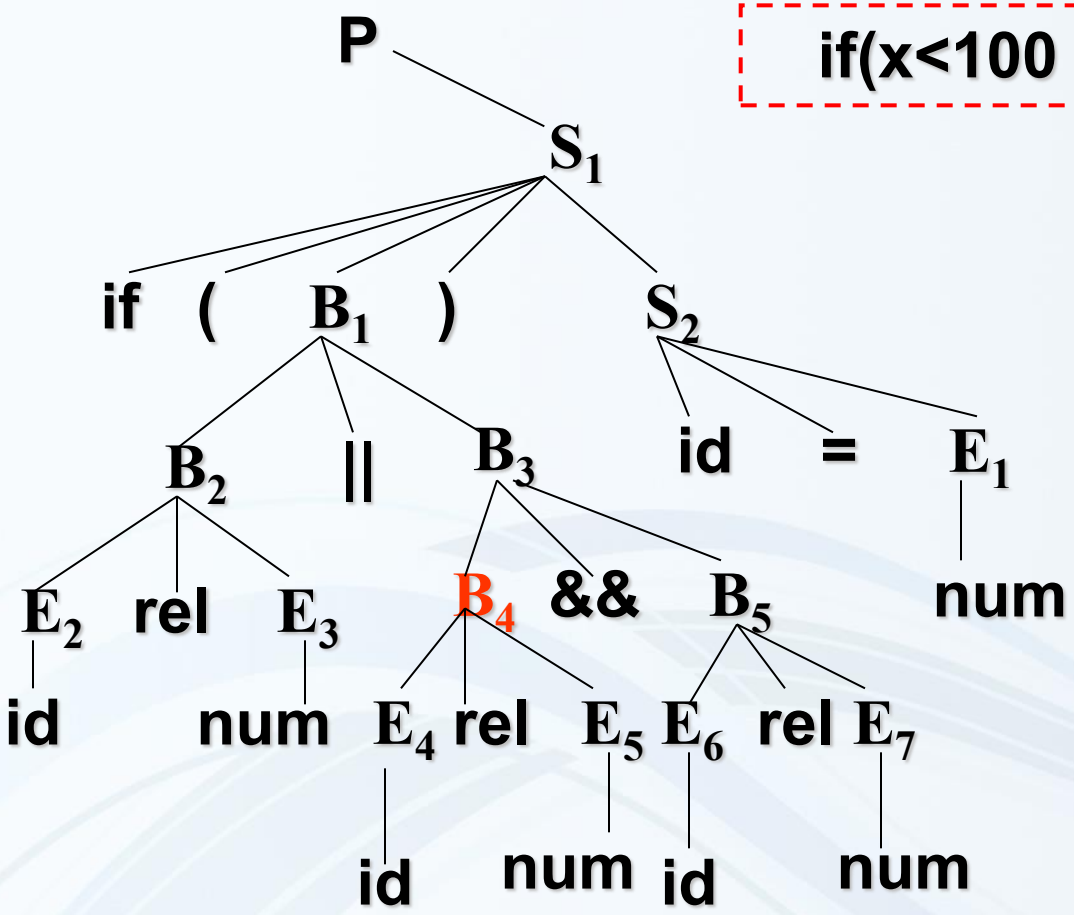


if(x<100 || x>200 && x!=y) x=0;

if(x<100) goto L2

goto L3

$S_1.next=L_1$;$B_1.true=L_2$; $B_1.false=L_1$;$B_2.true=L_2$;$B_2.false=L_3$;$B_3.true=L_2$; $B_3.false=L_1$;

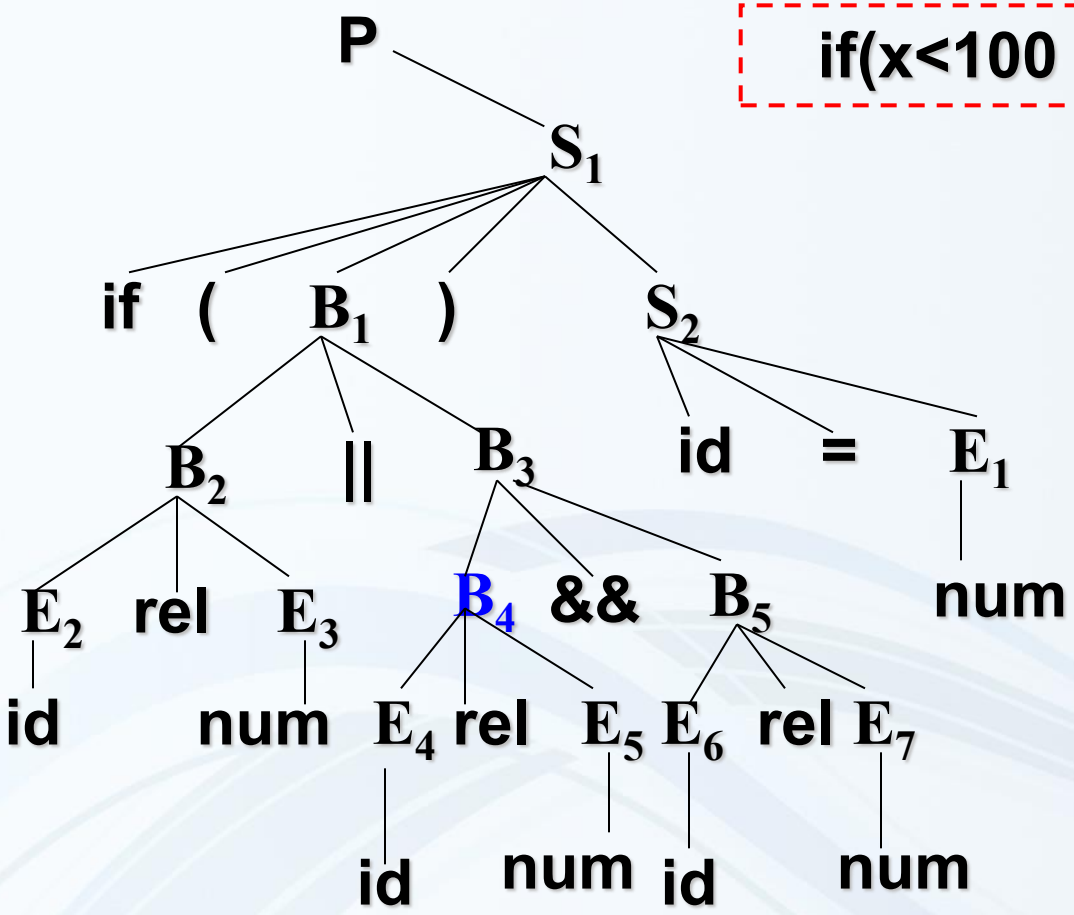


if(x<100 || x>200 && x!=y) x=0;

if(x<100) goto L2

goto L3

$S_1.next=L_1$;$B_1.true=L_2$; $B_1.false=L_1$;$B_2.true=L_2$; $B_2.false=L_3$;$B_3.true=L_2$; $B_3.false=L_1$; $B_4.true=L_4$; $B_4.false=L_1$;



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

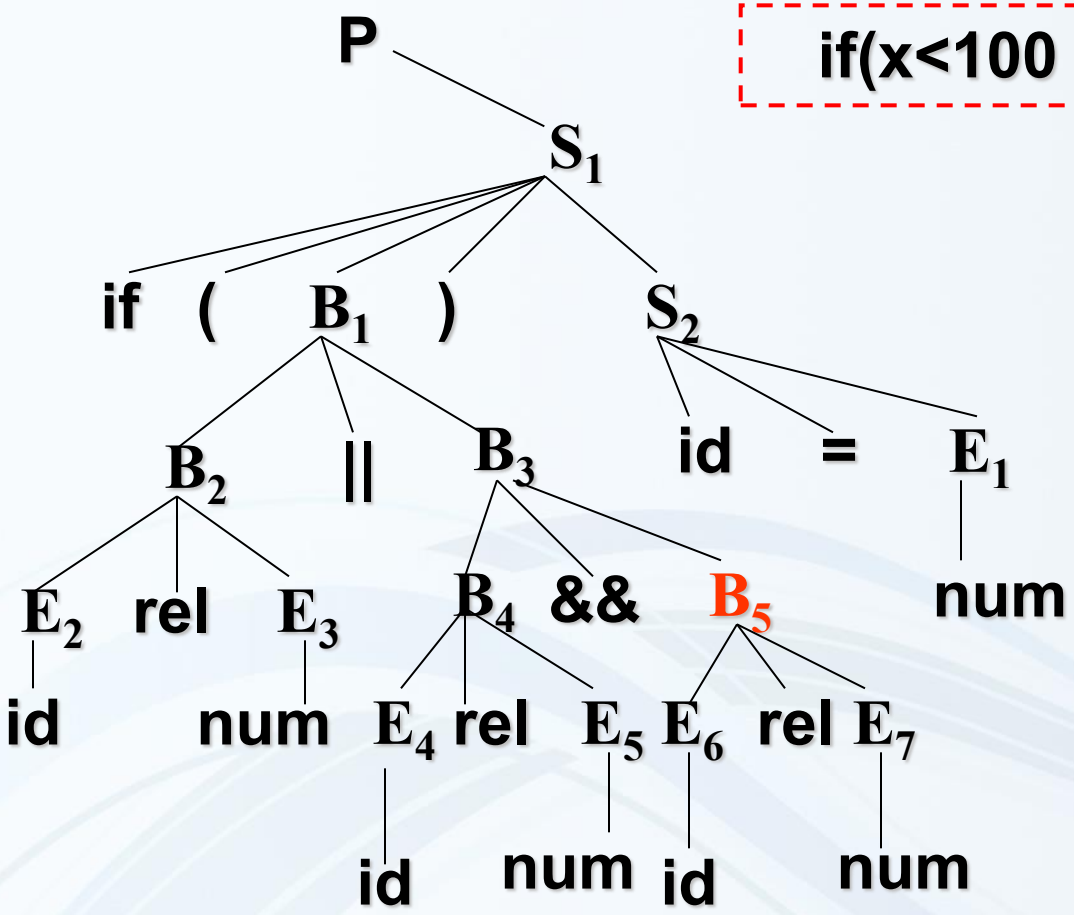
if(x<100) goto L2

goto L3

if(x>200) goto L4

goto L1

$S_1.next=L_1$;$B_1.true=L_2$; $B_1.false=L_1$;$B_2.true=L_2$; $B_2.false=L_3$;$B_3.true=L_2$; $B_3.false=L_1$; $B_4.true=L_4$; $B_4.false=L_1$;



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

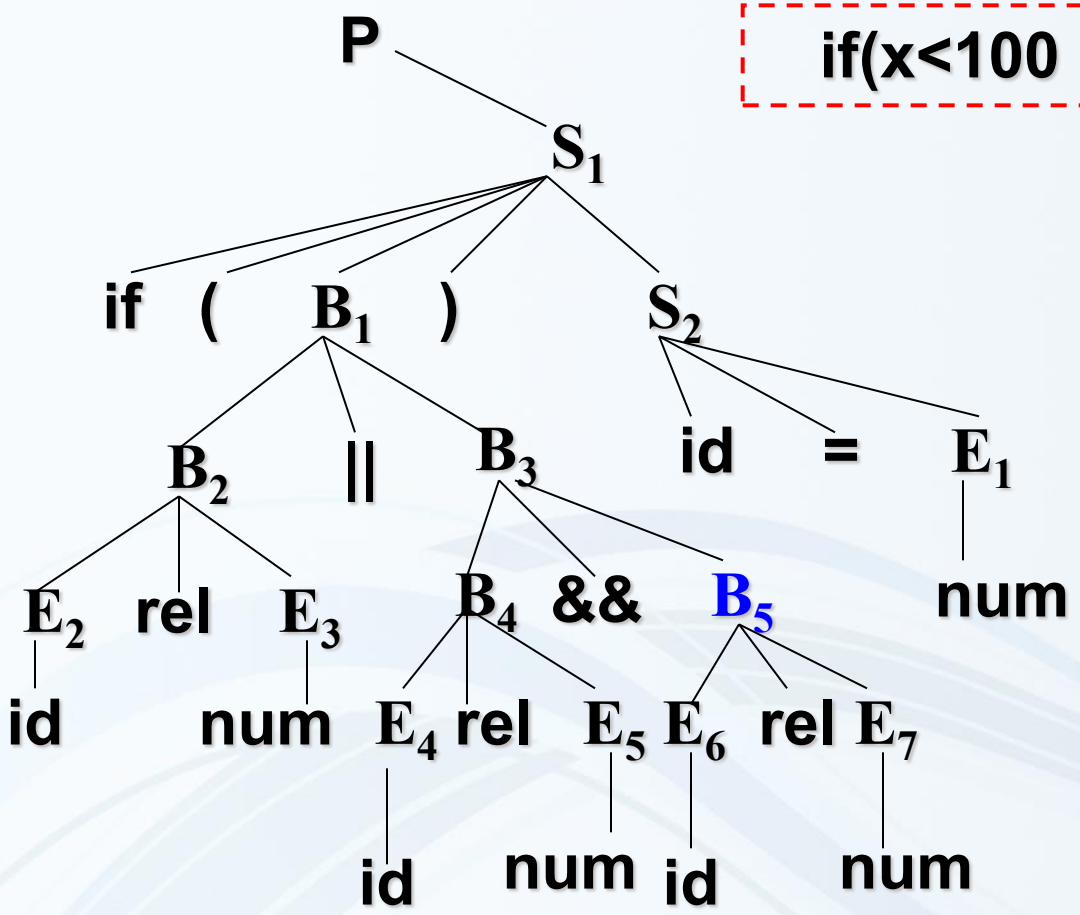
if(x<100) goto L2

goto L3

if(x>200) goto L4

goto L1

$S_1.next=L_1$;$B_1.true=L_2$; $B_1.false=L_1$;$B_2.true=L_2$; $B_2.false=L_3$;$B_3.true=L_2$; $B_3.false=L_1$; $B_4.true=L_4$; $B_4.false=L_1$; $B_5.true=L_2$; $B_5.false=L_1$;



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

if(x<100) goto L2

goto L3

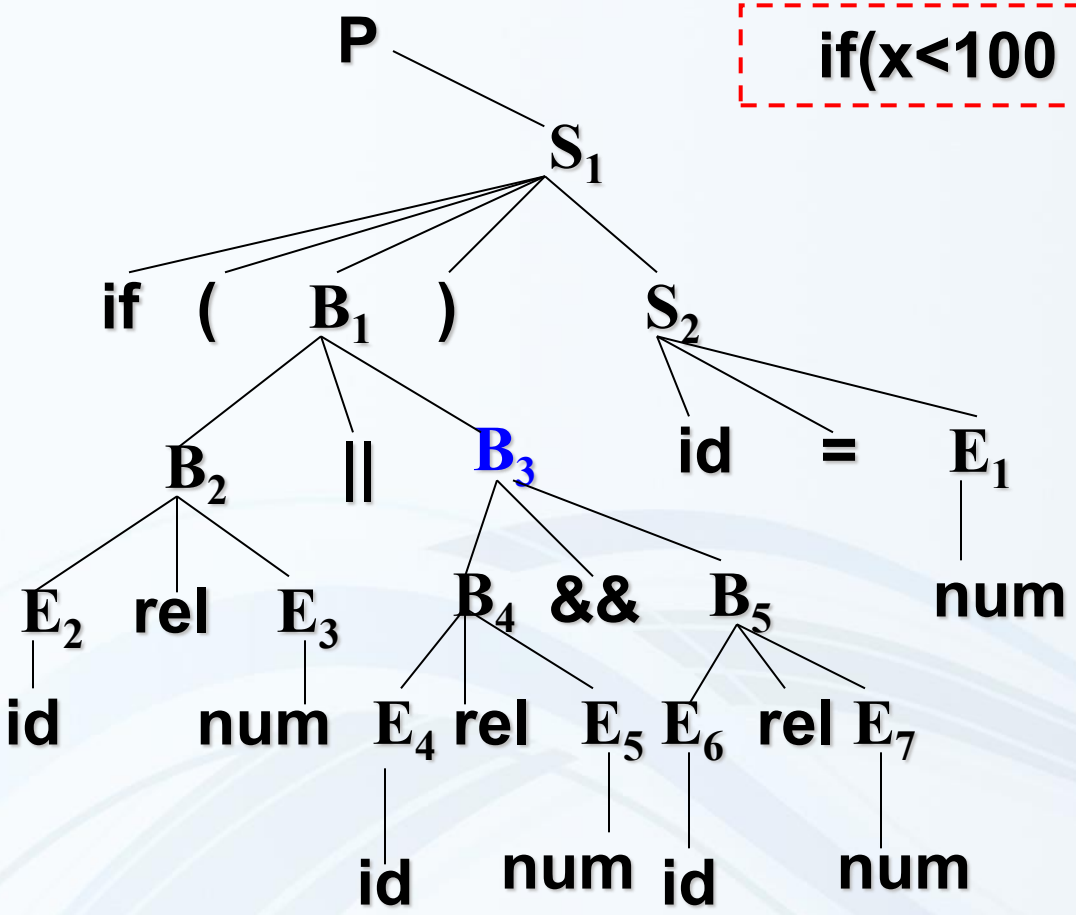
if(x>200) goto L4

goto L1

if(x!=y) goto L2

goto L1

$S_1.next=L_1$;$B_1.true=L_2$; $B_1.false=L_1$;$B_2.true=L_2$; $B_2.false=L_3$;$B_3.true=L_2$; $B_3.false=L_1$; $B_4.true=L_4$; $B_4.false=L_1$; $B_5.true=L_2$; $B_5.false=L_1$;



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

if(x<100) goto L2

goto L3

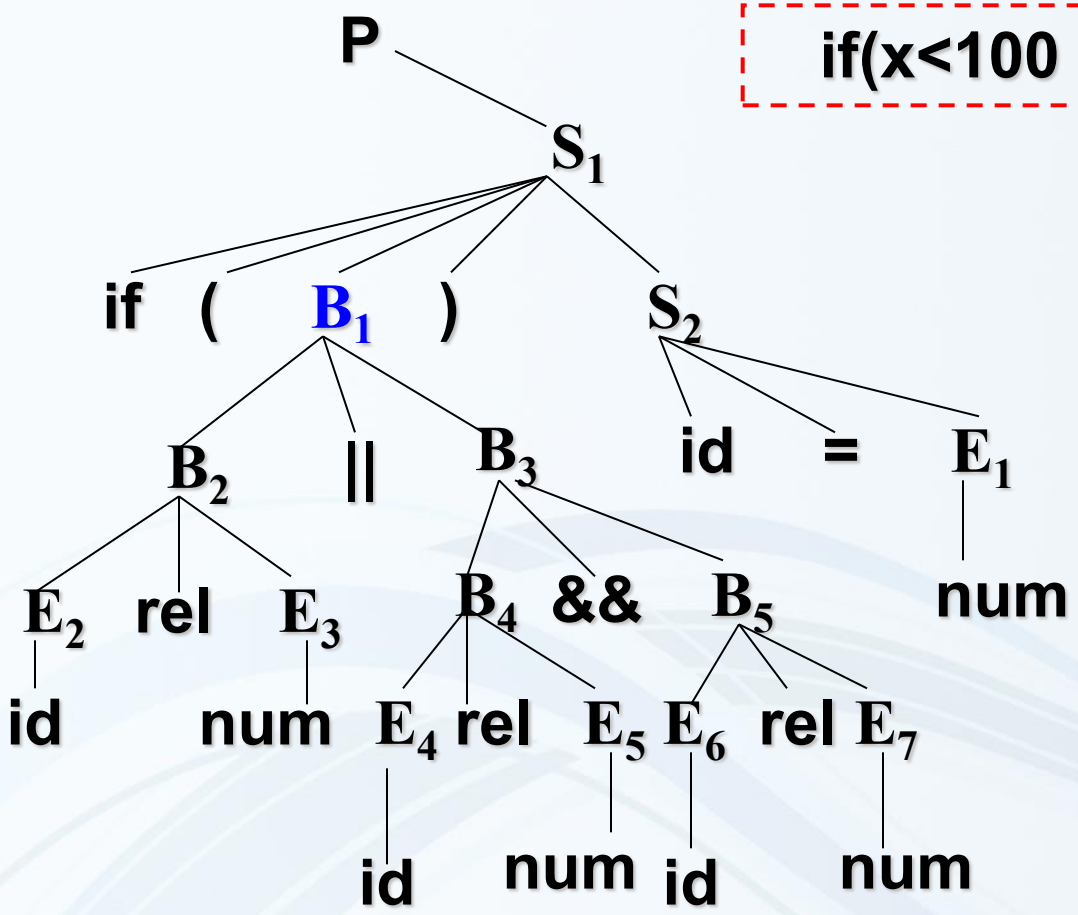
if(x>200) goto L4

goto L1

L4:if(x!=y) goto L2

goto L1

$S_1.next=L_1$;$B_1.true=L_2$; $B_1.false=L_1$;$B_2.true=L_2$; $B_2.false=L_3$;$B_3.true=L_2$; $B_3.false=L_1$; $B_4.true=L_4$; $B_4.false=L_1$; $B_5.true=L_2$; $B_5.false=L_1$;



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

if(x<100) goto L2

goto L3

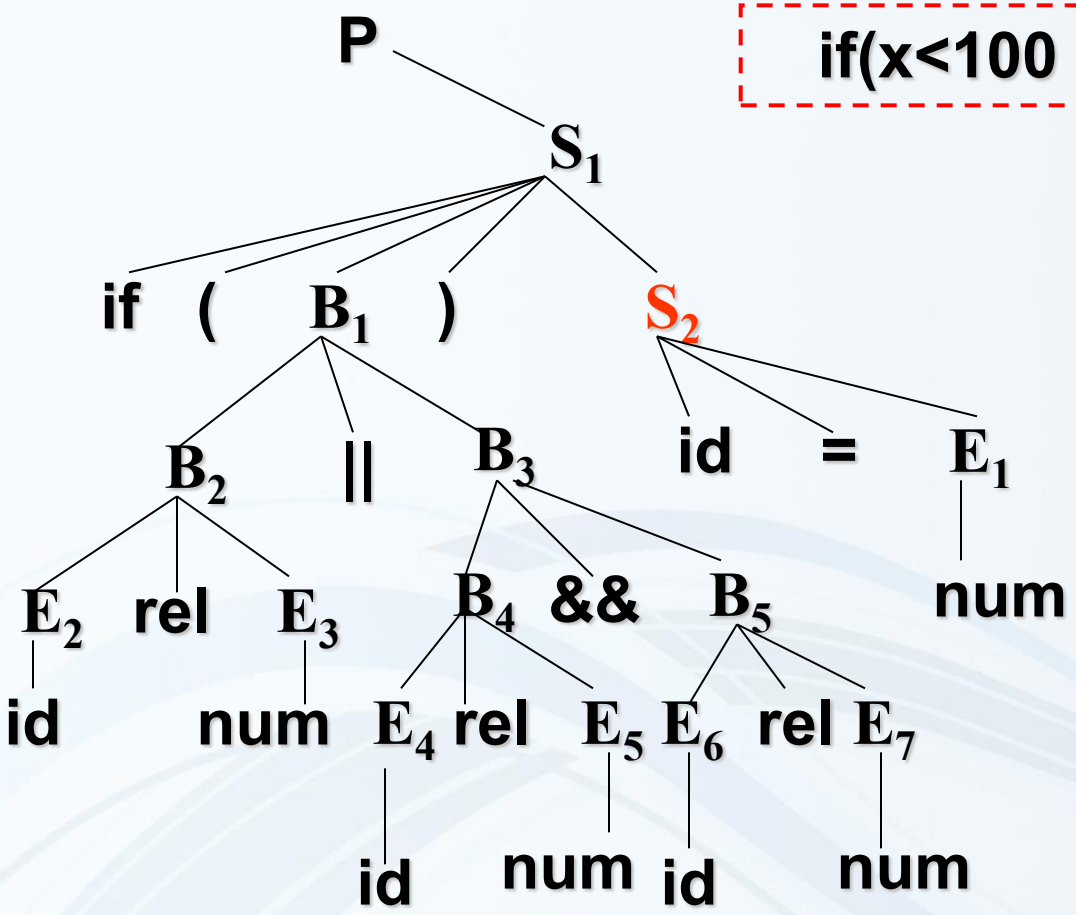
L3:if(x>200) goto L4

goto L1

L4:if(x!=y) goto L2

goto L1

$S_1.next=L_1$;$B_1.true=L_2$; $B_1.false=L_1$;$B_2.true=L_2$; $B_2.false=L_3$;$B_3.true=L_2$; $B_3.false=L_1$; $B_4.true=L_4$; $B_4.false=L_1$; $B_5.true=L_2$; $B_5.false=L_1$;



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

if(x<100) goto L2

goto L3

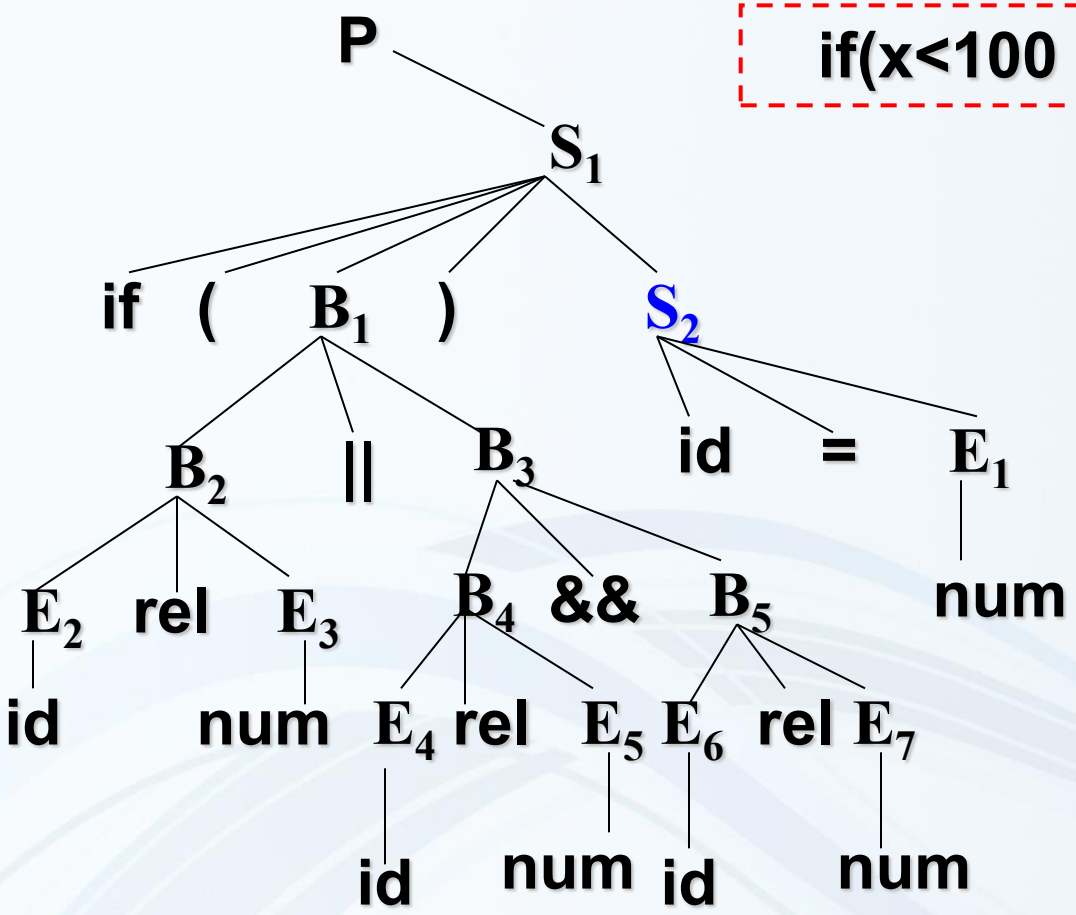
L3:if(x>200) goto L4

goto L1

L4:if(x!=y) goto L2

goto L1

$S_1.next=L_1$;$B_1.true=L_2$; $B_1.false=L_1$;$B_2.true=L_2$; $B_2.false=L_3$;$B_3.true=L_2$; $B_3.false=L_1$; $B_4.true=L_4$; $B_4.false=L_1$; $B_5.true=L_2$; $B_5.false=L_1$; $S_2.next=L_1$;



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

if(x<100) goto L2

goto L3

L3:if(x>200) goto L4

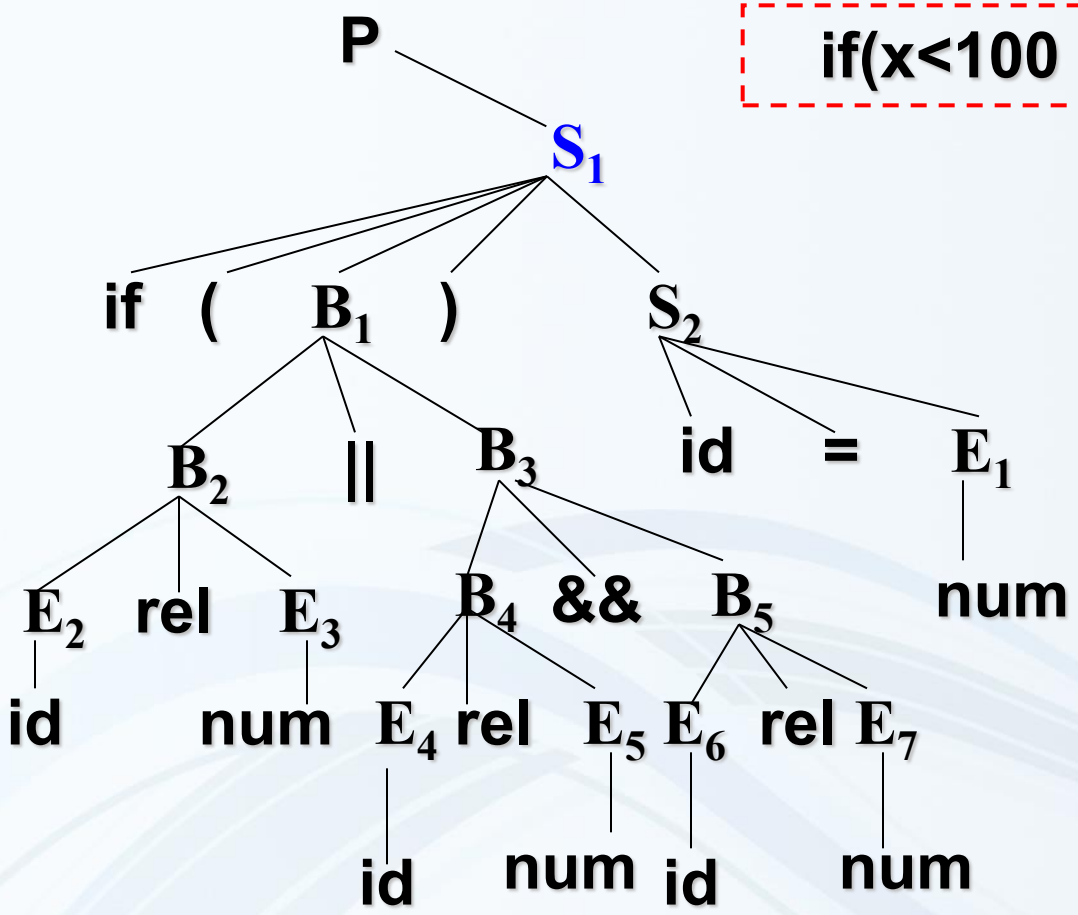
goto L1

L4:if(x!=y) goto L2

goto L1

x=0

$S_1.next=L_1$;$B_1.true=L_2$; $B_1.false=L_1$;$B_2.true=L_2$; $B_2.false=L_3$;$B_3.true=L_2$; $B_3.false=L_1$; $B_4.true=L_4$; $B_4.false=L_1$; $B_5.true=L_2$; $B_5.false=L_1$; $S_2.next=L_1$;



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

if(x<100) goto L2

goto L3

L3:if(x>200) goto L4

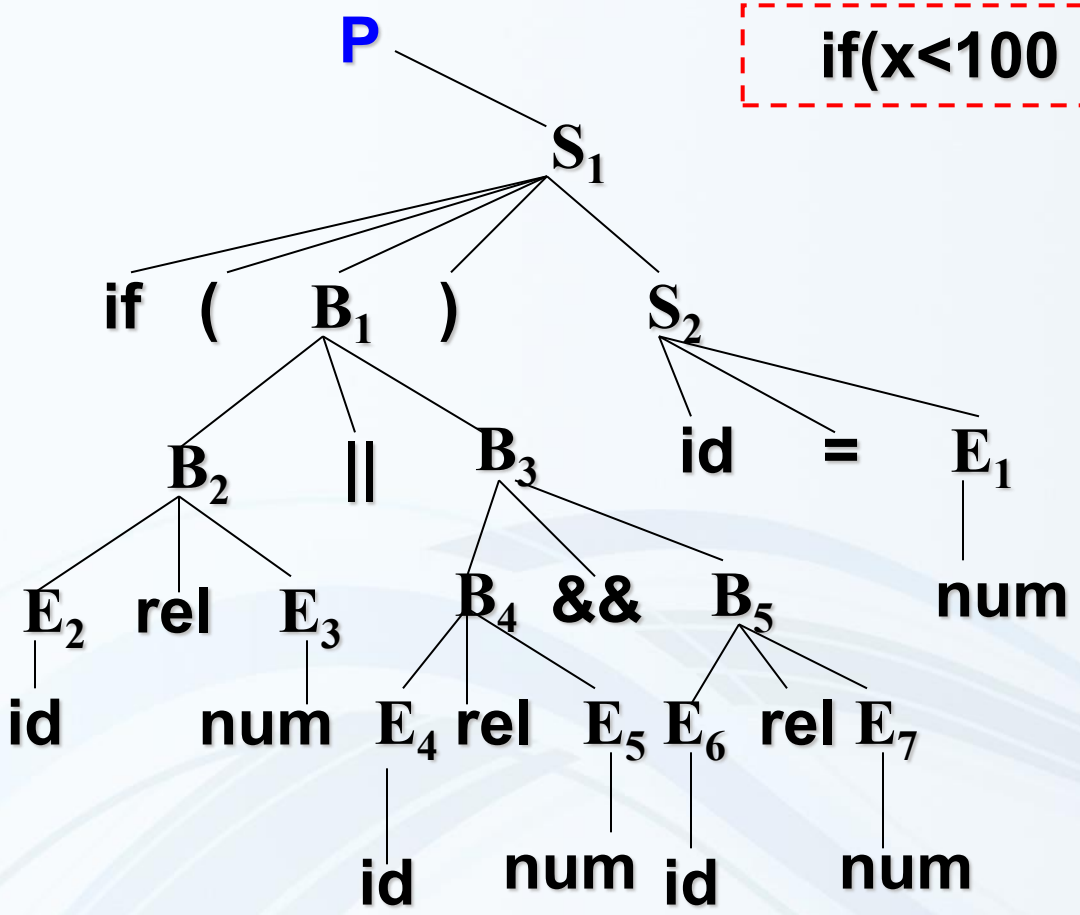
goto L1

L4:if(x!=y) goto L2

goto L1

L2:x=0

$S_1.next=L_1$;$B_1.true=L_2$; $B_1.false=L_1$;$B_2.true=L_2$; $B_2.false=L_3$;$B_3.true=L_2$; $B_3.false=L_1$; $B_4.true=L_4$; $B_4.false=L_1$; $B_5.true=L_2$; $B_5.false=L_1$; $S_2.next=L_1$;



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:
```

$S_1.next=L_1$;$B_1.true=L_2$; $B_1.false=L_1$;$B_2.true=L_2$; $B_2.false=L_3$;$B_3.true=L_2$; $B_3.false=L_1$; $B_4.true=L_4$; $B_4.false=L_1$; $B_5.true=L_2$; $B_5.false=L_1$; $S_2.next=L_1$;

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

```
100: if (x<100) goto
101: goto
102: if(x>200) goto
103: goto
104: if(x!=y) goto
105: goto
106: x=0
107:
```

```
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:
```

控制流翻译的SDD

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 || label(B.true) || S_1.code || gen('goto' S.next) || label(B.false) || S_2.code

控制流翻译的SDD(II)

产生式	语义规则
$S \rightarrow while(B)S_1$	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('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$S \rightarrow repeat\ S_1\ until\ B$对应的语义规则是什么？

6.7回填*Backpatching*

❖两趟式执行语法制导定义（L属性定义）：

1.将标记作为继承属性传至生成跳传指令处

2.绑定标记和地址

❖一趟式执行:（S属性定义）：

➢用综合属性*truelist*和*falselist*管理跳转指令中的标记

➢ 采用回填：在目标地址确定时，填充这些标记

回填(II)

- 三个函数:
  1. *makelist*(i) 建立新表,表中加入元素i(i是指令数组的下标）,返回表指针
  2. *merge*(p_1, p_2) 合并p_1、p_2指向的表，返回p_2
  3. *backpatch*(p, i) 将i回填入p指向列表中每一语句
- nextinstr:下一条指令的地址

布尔表达式的回填

$B \rightarrow B_1 \,||\, M\ B_2$ {backpatch(B_1.falselist, M.instr);
B.truelist = merge(B_1.truelist, B_2.truelist);
B.falselist = B_2.falselist;}

$B \rightarrow B_1 \,\&\&\, M\ B_2$ {backpatch(B_1.truelist, M.instr);
B.falselist = merge(B_1.falselist, B_2.falselist);
B.truelist = B_2.truelist;}

$B \rightarrow !B_1$ {B.truelist = B_1.falselist; B.falselist=B_1.truelist;}

$B \rightarrow (B_1)$ {B.truelist = B_1.truelist; B.falselist=B_1.falselist;}

$M \rightarrow \varepsilon$ {M.instr = nextinstr;}

布尔表达式的回填(II)

$B \rightarrow E_1$ rel E_2 {B.truelist = makelist(nextinstr);

B.falselist = makelist(nextinstr+1);

gen('if' E_1.addr rel.op E_2.addr 'goto _');

gen('goto _');}

B→true {B.truelist = makelist(nextinstr);

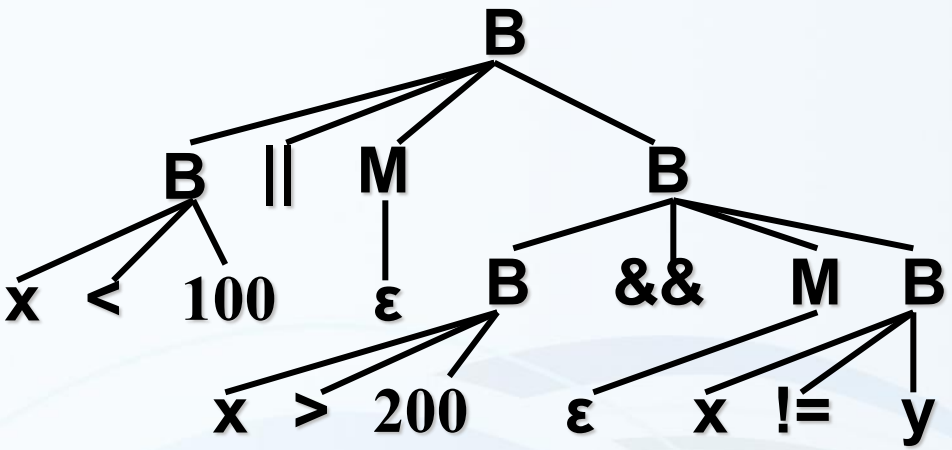
gen('goto _');}

B→false {B.falselist = makelist(nextinstr);

gen('goto _');}

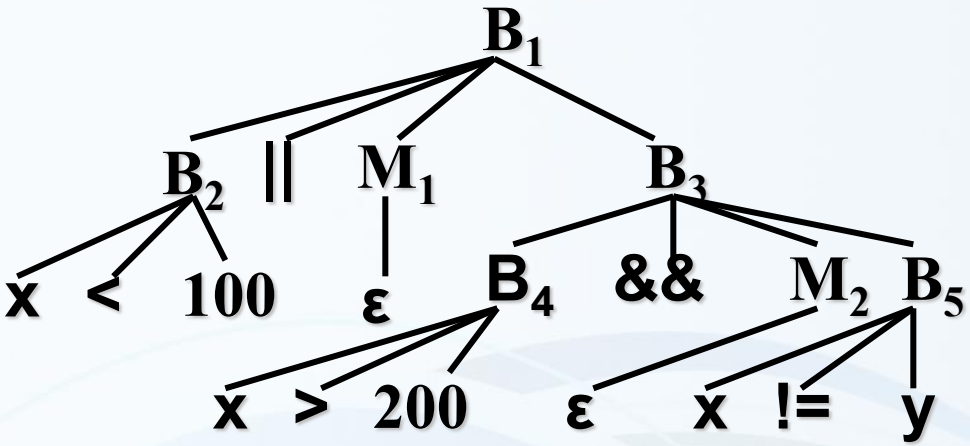
为什么B.falselist = makelist(nextinstr+1);?

布尔表达式的回填(例)

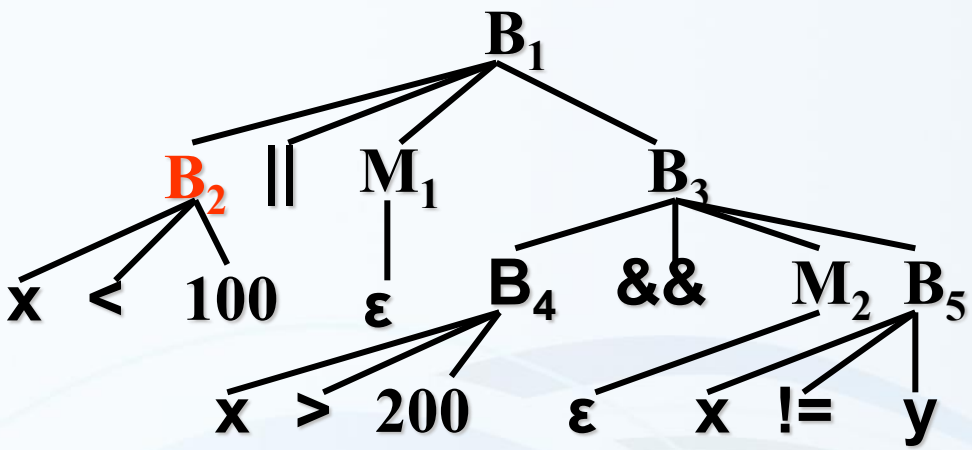


x<100 || x>200 && x!=y

布尔表达式的回填(例)



布尔表达式的回填(例)

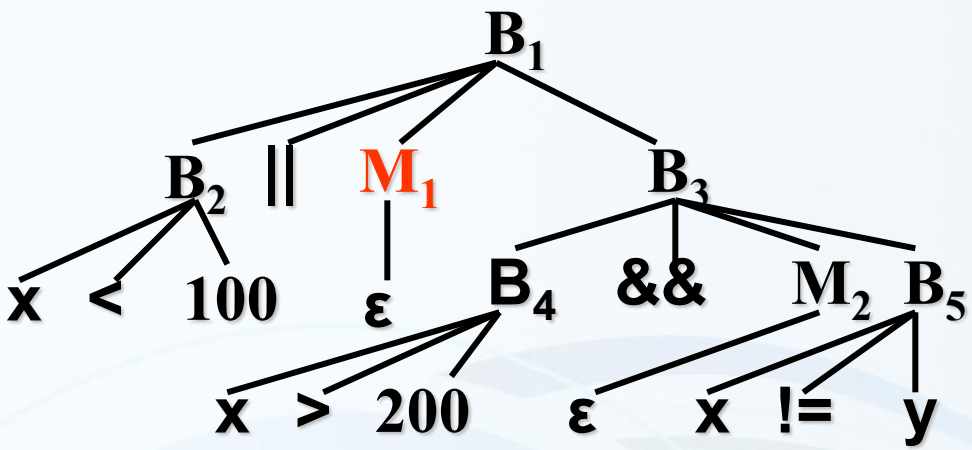


100:if x < 100 goto

101:goto

$B_2.t=\{100\}$; $B_2.f=\{101\}$;

布尔表达式的回填(例)

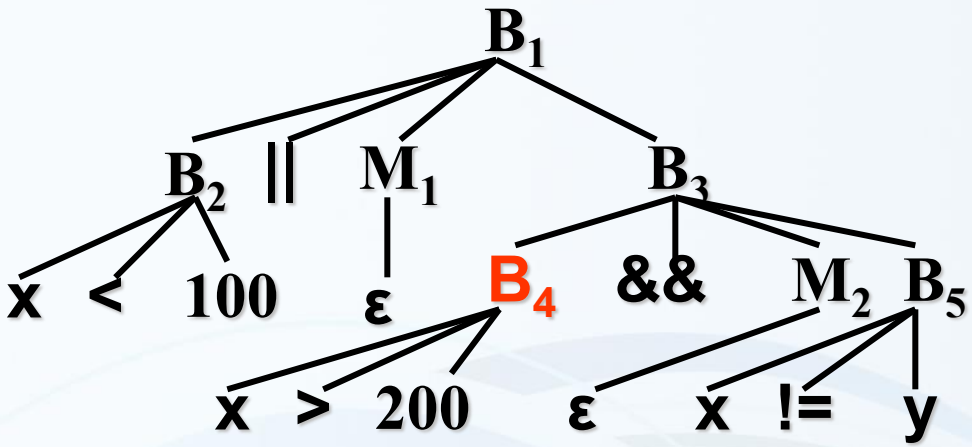


100:if x < 100 goto

101:goto

B_2.t={100}; B_2.f={101};M_1.i=102;

布尔表达式的回填(例)



100:if x < 100 goto

101:goto

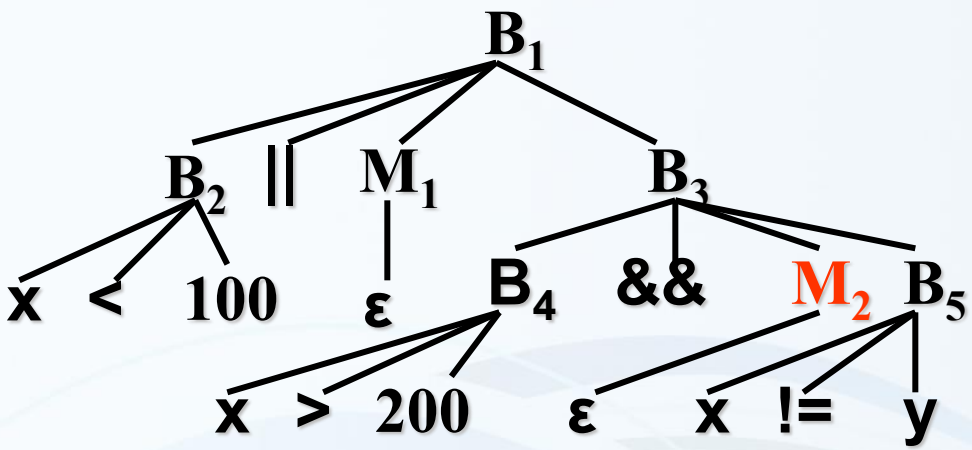
102:if x > 200 goto

103:goto

B_2.t={100}; B_2.f={101};M_1.i=102;

B_4.t={102}; B_4.f={103};

布尔表达式的回填(例)



100:if x < 100 goto

101:goto

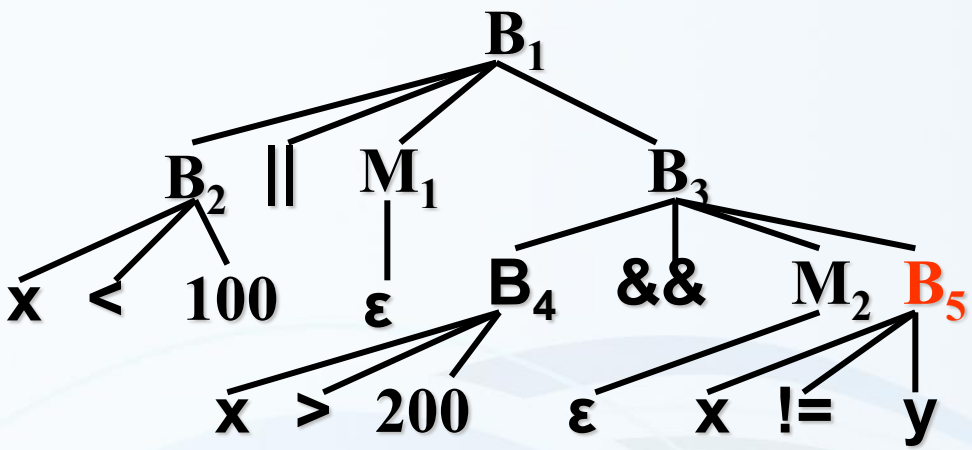
102:if x > 200 goto

103:goto

B_2.t={100}; B_2.f={101};M_1.i=102;

B_4.t={102}; B_4.f={103}; M_2.i=104;

布尔表达式的回填(例)



100:if x < 100 goto

101:goto

102:if x > 200 goto

103:goto

104:if x!=y goto

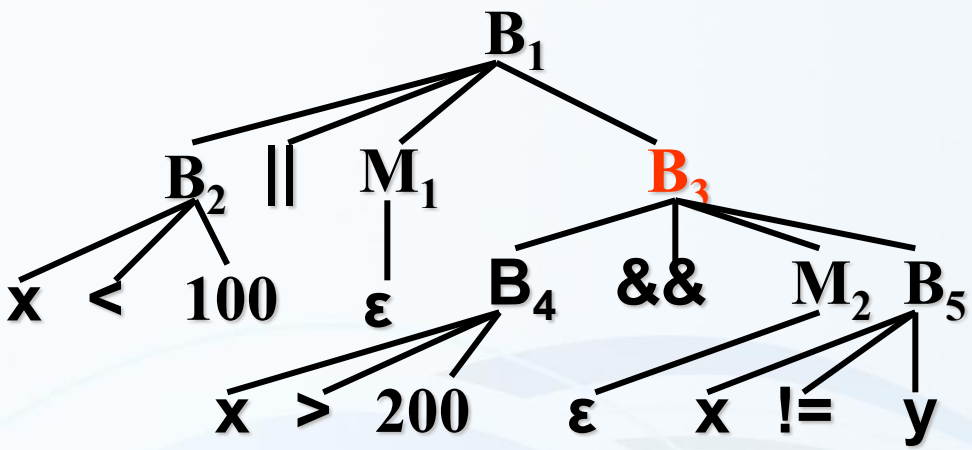
105:goto

$B_2.t=\{100\}$; $B_2.f=\{101\}$;$M_1.i=102$;

$B_4.t=\{102\}$; $B_4.f=\{103\}$; $M_2.i=104$;

$B_5.t=\{104\}$; $B_5.f=\{105\}$;

布尔表达式的回填(例)



100:if x < 100 goto

101:goto

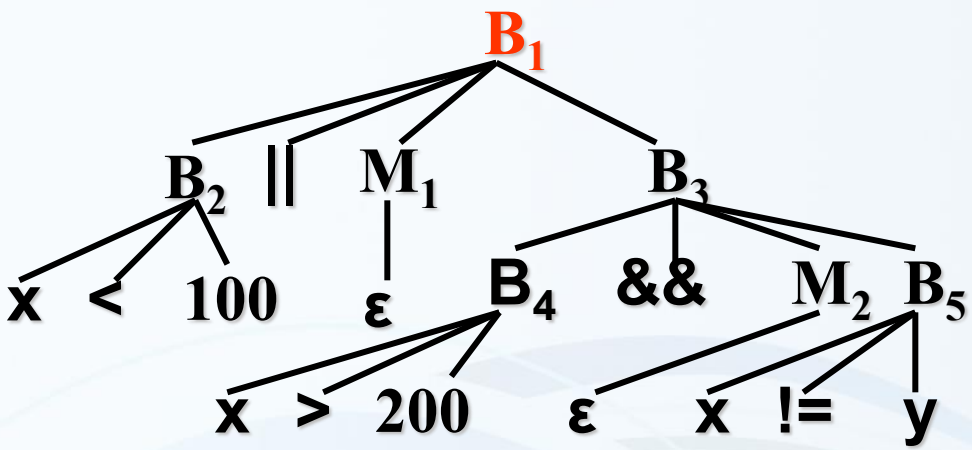
102:if x > 200 goto 104

103:goto

104:if x!=y goto

105:goto

$B_2.t=\{100\}$; $B_2.f=\{101\}$;$M_1.i=102$;
$B_4.t=\{102\}$; $B_4.f=\{103\}$; $M_2.i=104$;
$B_5.t=\{104\}$; $B_5.f=\{105\}$; $B_3.t=\{104\}$;
$B_3.f=\{103, 105\}$;



100:if x < 100 goto

101:goto 102

102:if x > 200 goto 104

103:goto

104:if x!=y goto

105:goto

B_2.t={100}; B_2.f={101};M_1.i=102;

B_4.t={102}; B_4.f={103}; M_2.i=104;

B_5.t={104}; B_5.f={105}; B_3.t={104};

B_3.f={103, 105}; B_1.t={100, 104};

B_1.f={103,105}

真出口填100,104行，假出口填103,105行

控制流语句

- 语句S带有一个未填充的跳转链 *nextlist*, 该链同样使用回填策略来填充
- *S.nextlist* 为跳转到S之后的跳转指令列表

1) $S \rightarrow \textbf{if}\,(B)\,M\,S_1$ { *backpatch*($B.truelist$, $M.instr$);
   $S.nextlist = merge(B.falselist, S_1.nextlist)$; }

2) $S \rightarrow \textbf{if}\,(B)\,M_1\,S_1\,N\,\textbf{else}\,M_2\,S_2$
   { *backpatch*($B.truelist$, $M_1.instr$);
   backpatch($B.falselist$, $M_2.instr$);
   $temp = merge(S_1.nextlist, N.nextlist)$;
   $S.nextlist = merge(temp, S_2.nextlist)$; }

3) $S \rightarrow \textbf{while}\,M_1\,(B)\,M_2\,S_1$
   { *backpatch*($S_1.nextlist$, $M_1.instr$);
   backpatch($B.truelist$, $M_2.instr$);
   $S.nextlist = B.falselist$;
   gen('goto' $M_1.instr$); }

控制流语句（II）

4) $S \rightarrow \{\ L\ \}$	{ *S.nextlist* = *L.nextlist*; }
5) $S \rightarrow A\ ;$	{ *S.nextlist* = **null**; }
6) $M \rightarrow \epsilon$	{ *M.instr* = *nextinstr*; }
7) $N \rightarrow \epsilon$	{ *N.nextlist* = *makelist*(*nextinstr*); *gen*('goto _'); }
8) $L \rightarrow L_1\ M\ S$	{ *backpatch*(L_1.*nextlist*, *M.instr*); *L.nextlist* = *S.nextlist*; }
9) $L \rightarrow S$	{ *L.nextlist* = *S.nextlist*; }

例

```
while (E1) {                          i1: Code for E1
    if (E2)                           i2: Code for E2
        while (E3)                    i3: Code for E3
            S1;                       i4: Code for S1
    else {                            i5: Code for E4
        if (E4)                       i6: Code for S2
            S2;                       i7: Code for S3
        S3                            i8: ···
    }
}
```

求：

(a) E_3.false (b) S_2.next (c) E_4.false

(d) S_1.next (e) E_2.true

例

```
while (E1) {                    i1: Code for E1
    if (E2)                     i2: Code for E2
        while (E3)              i3: Code for E3
            S1;                 i4: Code for S1
    else {                      i5: Code for E4
        if (E4)                 i6: Code for S2
            S2;                 i7: Code for S3
        S3                      i8: ···
    }
}
```

S_4:while (E_3) S_1

S_5: if (E_4) S_2

S_6: 包括S_5和S_3的代码段

S_7： if(E_2) S_4 else S_6

S_8: 全部程序

确定：(a)S_4.next　　(b) S_5.next　　(c)S_6.next

(d) S_7.next　　(e)S_8.next