Compiler Design

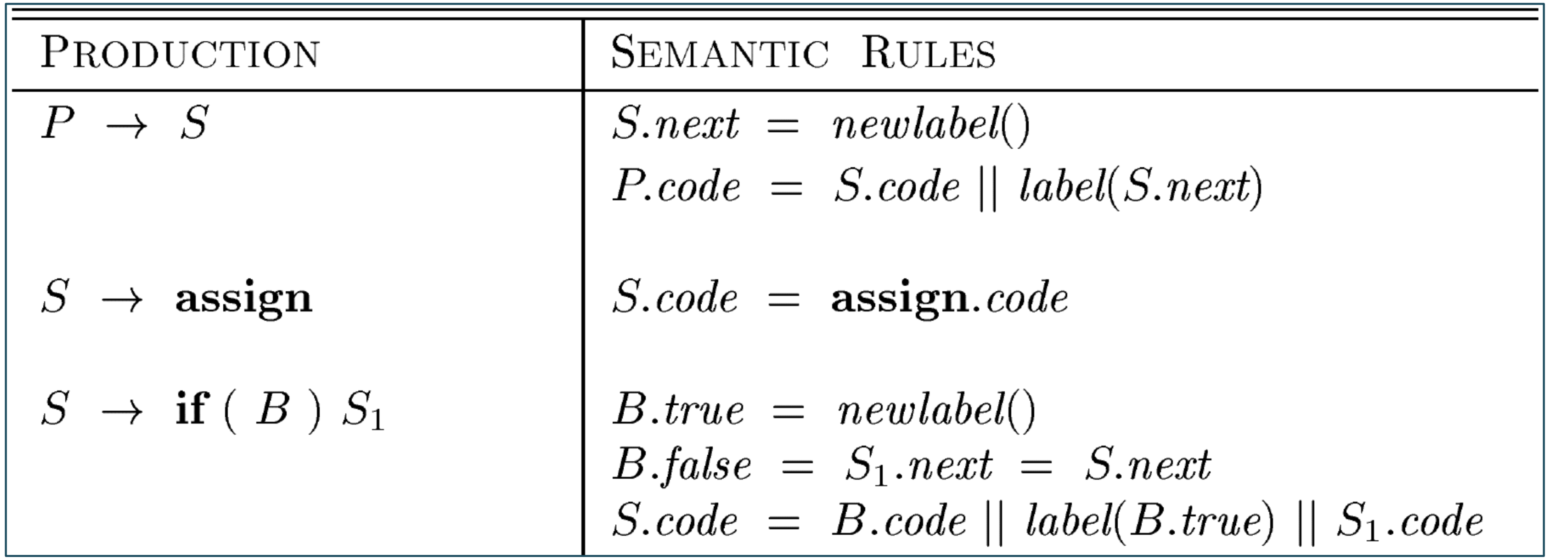
Fatemeh Deldar

Isfahan University of Technology

1403-1404

Control Flow

- **Example:** Syntax-directed definition for flow-of-control statements

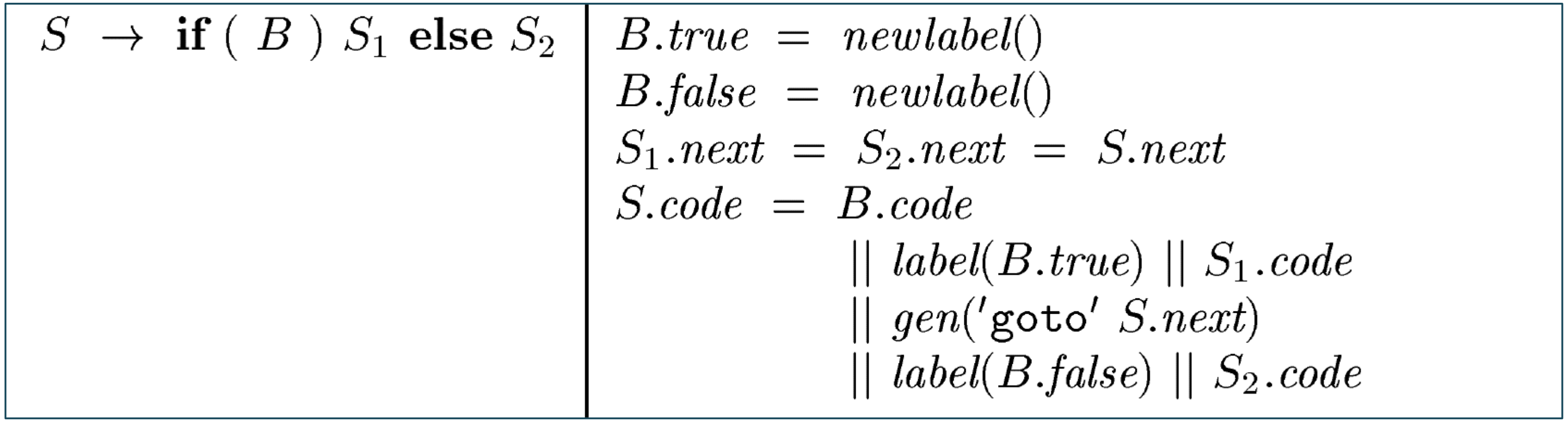


PRODUCTION	SEMANTIC RULES
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \;\|\|\; label(S.next)$
$S \rightarrow \textbf{assign}$	$S.code = \textbf{assign}.code$
$S \rightarrow \textbf{if}\ (\ B\ )\ S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \;\|\|\; label(B.true) \;\|\|\; S_1.code$

  - ***newlabel***() creates a new label each time it is called
  - ***label***(***L***) attaches label L to the next three-address instruction to be generated
  - Token **assign** in the production $S \rightarrow \textbf{assign}$ is a placeholder for assignment statements

Control Flow

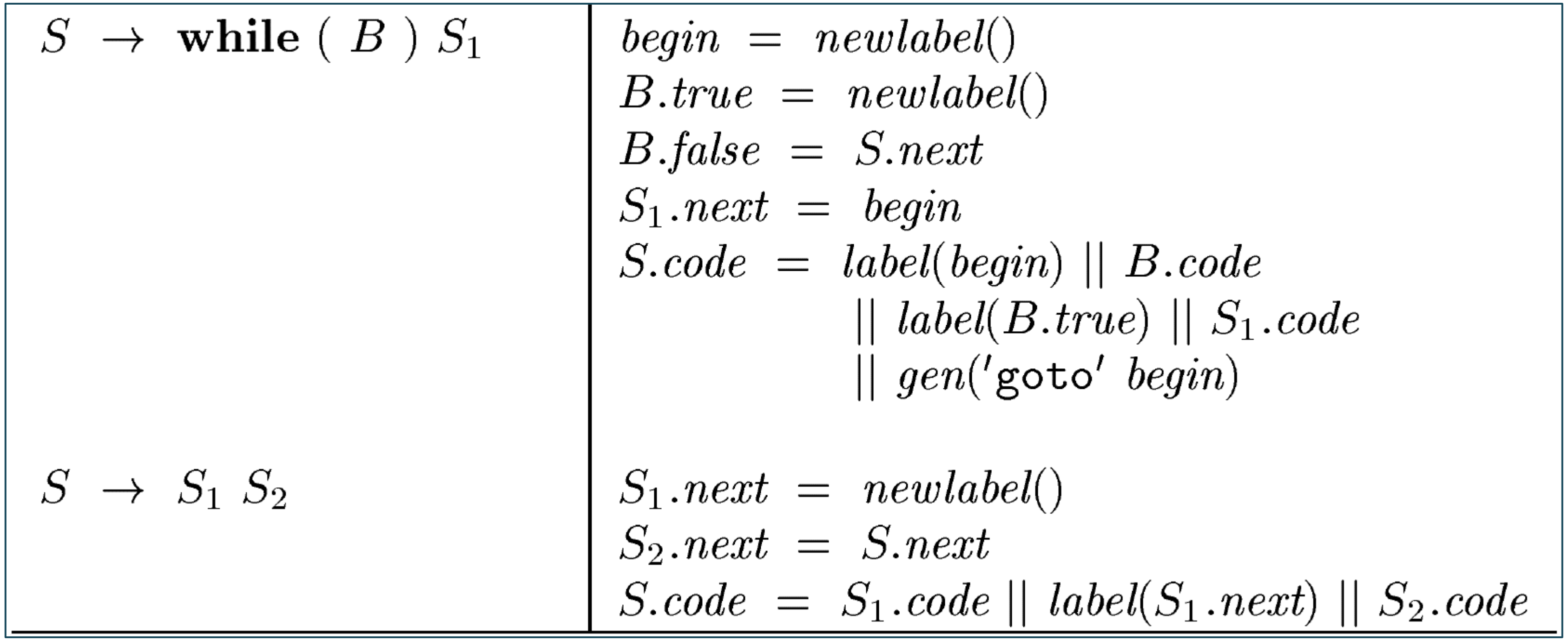
- **Example (cont.):** Syntax-directed definition for flow-of-control statements



$S \rightarrow \textbf{if} \ (\ B \ ) \ S_1 \ \textbf{else} \ S_2$	$B.true = newlabel()$ $B.false = newlabel()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ $\quad \| \ label(B.true) \ \| \ S_1.code$ $\quad \| \ gen('\texttt{goto}' \ S.next)$ $\quad \| \ label(B.false) \ \| \ S_2.code$

Control Flow

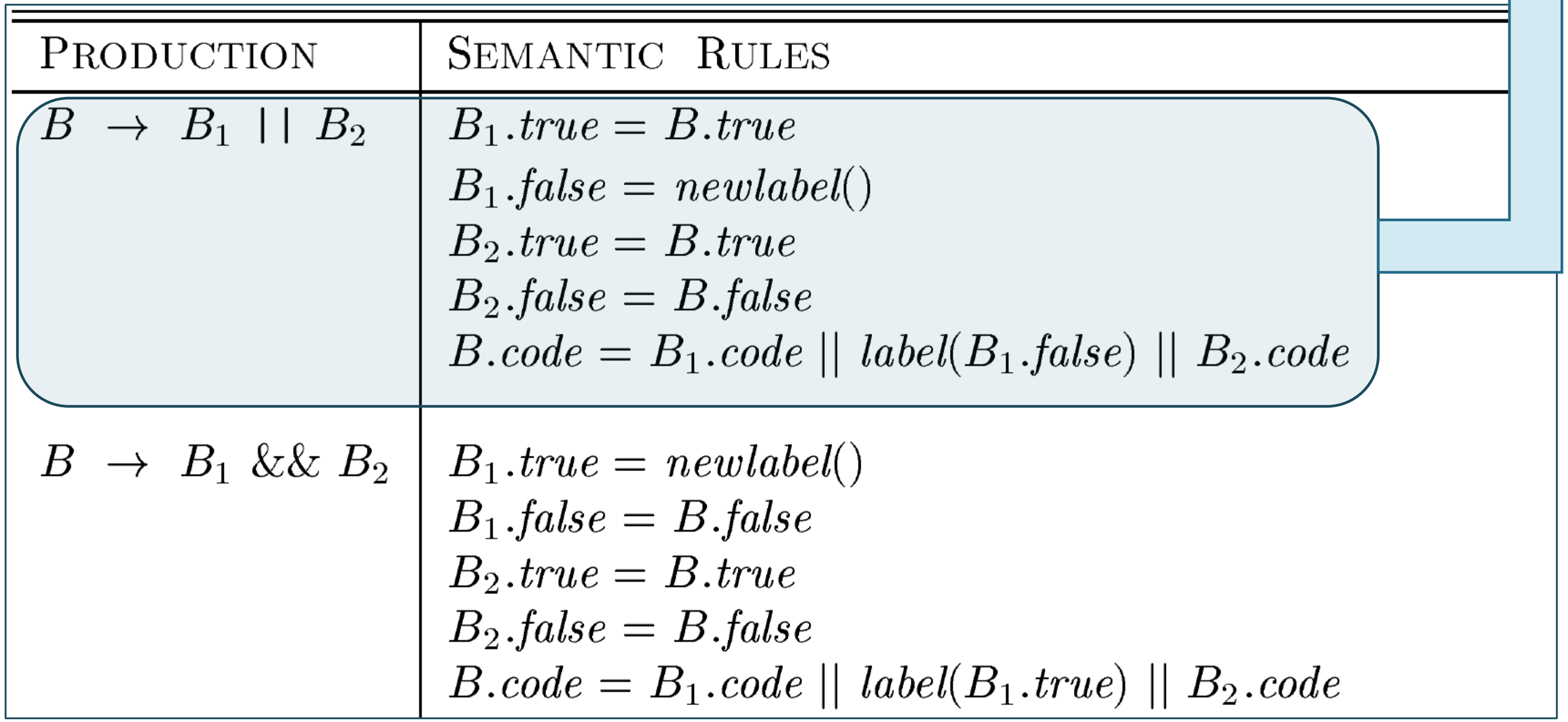
- **Example (cont.):** Syntax-directed definition for flow-of-control statements



$S \rightarrow \textbf{while}\ (\ B\ )\ S_1$	$begin \ = \ newlabel()$ $B.true \ = \ newlabel()$ $B.false \ = \ S.next$ $S_1.next \ = \ begin$ $S.code \ = \ label(begin) \ \Vert \ B.code$ $\quad \Vert \ label(B.true) \ \Vert \ S_1.code$ $\quad \Vert \ gen('\texttt{goto}' \ begin)$
$S \rightarrow S_1 \ S_2$	$S_1.next \ = \ newlabel()$ $S_2.next \ = \ S.next$ $S.code \ = \ S_1.code \ \Vert \ label(S_1.next) \ \Vert \ S_2.code$

Control Flow

- **Example:** Generating three-address code for booleans

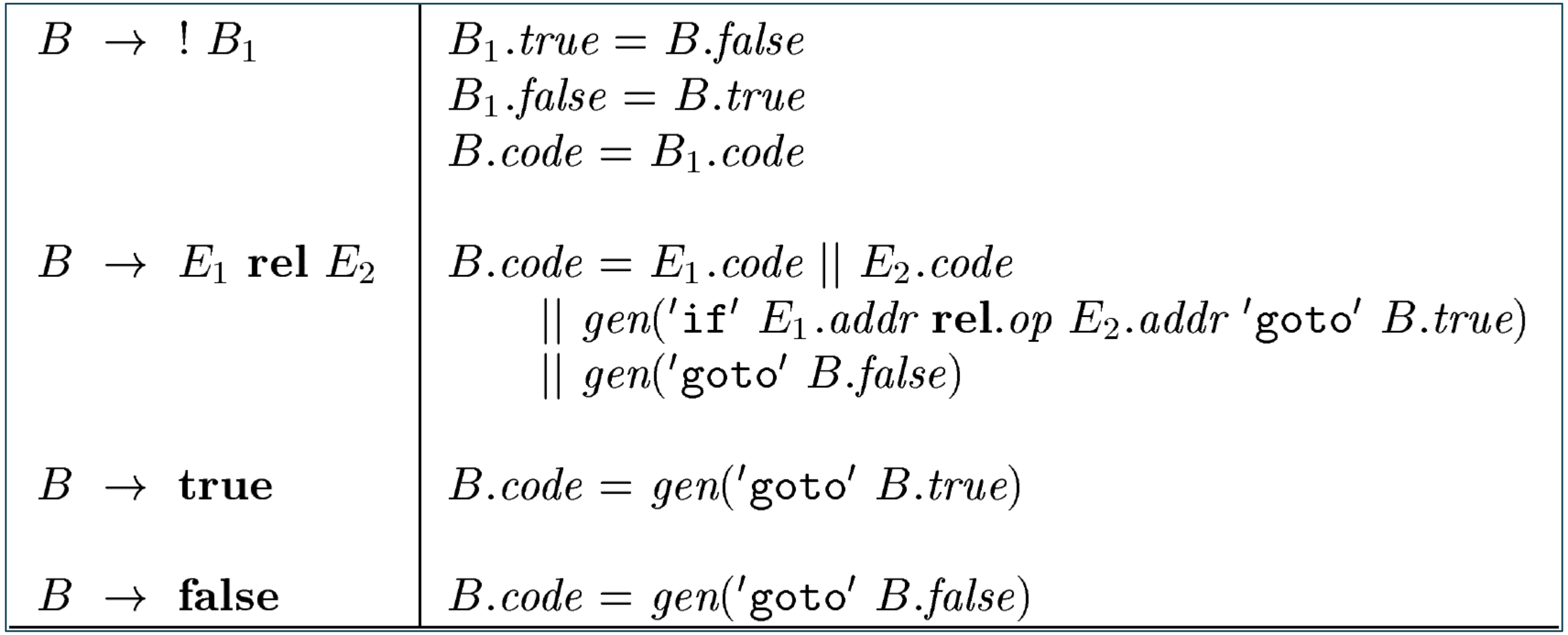


PRODUCTION	SEMANTIC RULES
$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \;\&\&\; B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$

- If B1 is true, then we immediately know that B itself is true, so B1.true is the same as B.true
- If B1 is false, then B2 must be evaluated, so we make B1.false be the label of the first instruction in the code for B2
- The true and false exits of B2 are the same as the true and false exits of B

Control Flow

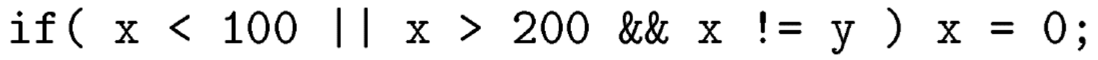
- <span>**Example (cont.):**</span> Generating three-address code for booleans



$B \to\ !\ B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$
$B \to E_1\ \textbf{rel}\ E_2$	$B.code = E_1.code \Vert E_2.code$ $\Vert\ gen('\texttt{if}'\ E_1.addr\ \textbf{rel}.op\ E_2.addr\ '\texttt{goto}'\ B.true)$ $\Vert\ gen('\texttt{goto}'\ B.false)$
$B \to \textbf{true}$	$B.code = gen('\texttt{goto}'\ B.true)$
$B \to \textbf{false}$	$B.code = gen('\texttt{goto}'\ B.false)$

Control Flow

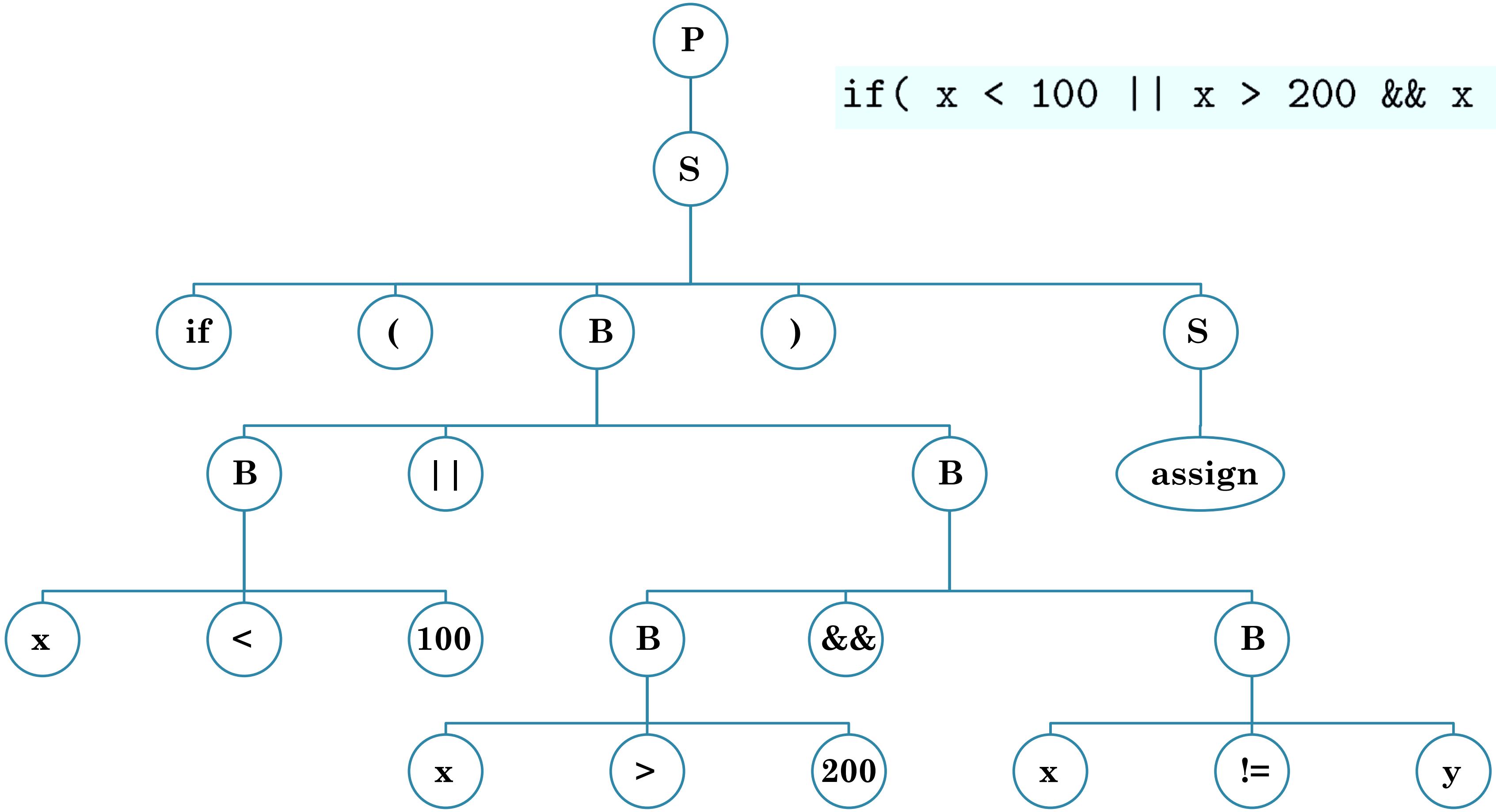
- **Example:** Obtain the code of the following statement



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

  - The grammar is:
    - $S \rightarrow$ **if** $(\ B\ )\ S$ | **if** $(\ B\ )\ S$ **else** S | **while** $(\ B\ )\ S$ | SS | **assign**
    - $B \rightarrow B \,||\, B$ | B && B | $!\,B$ | E **rel** E | **true** | **false**

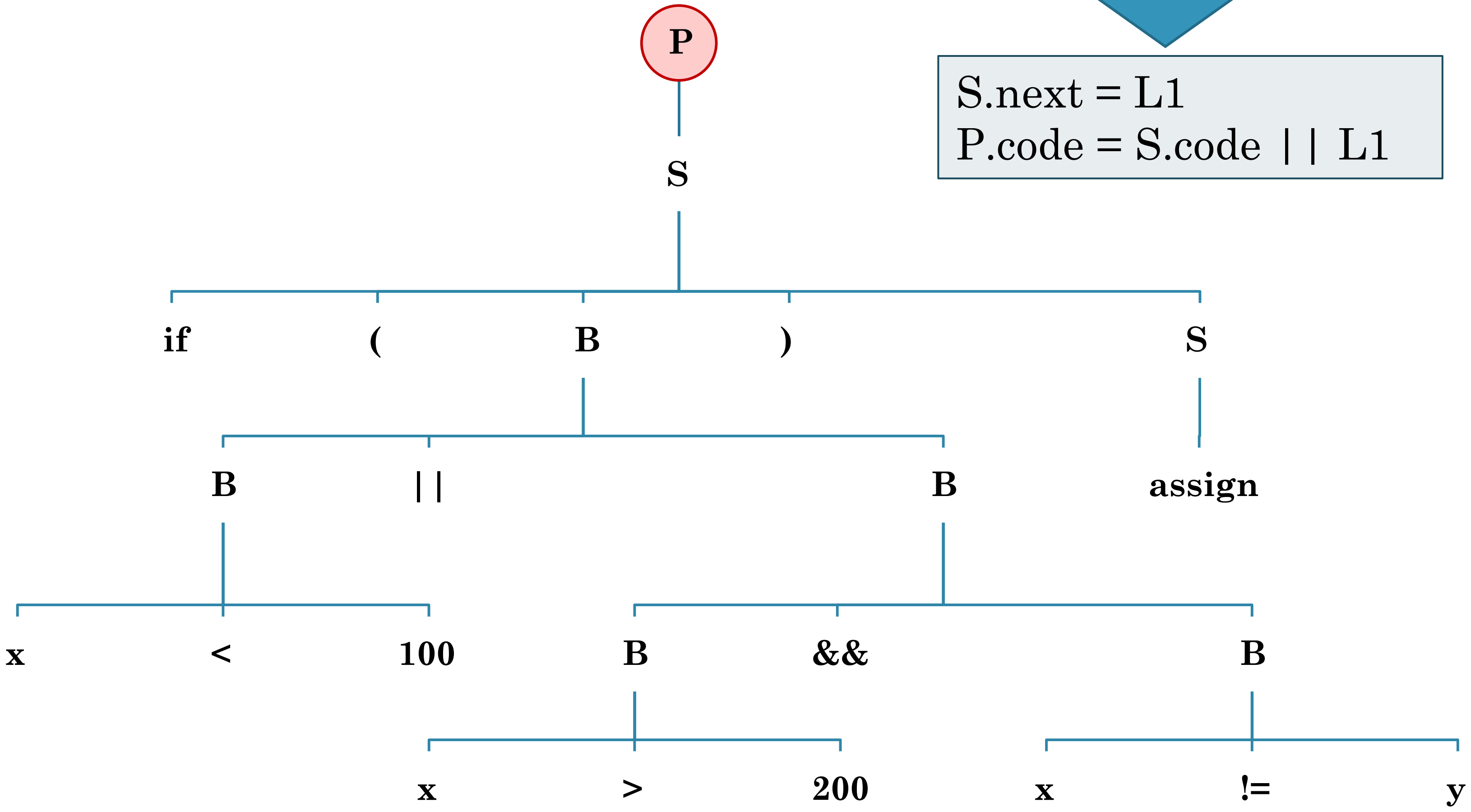
Control Flow



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

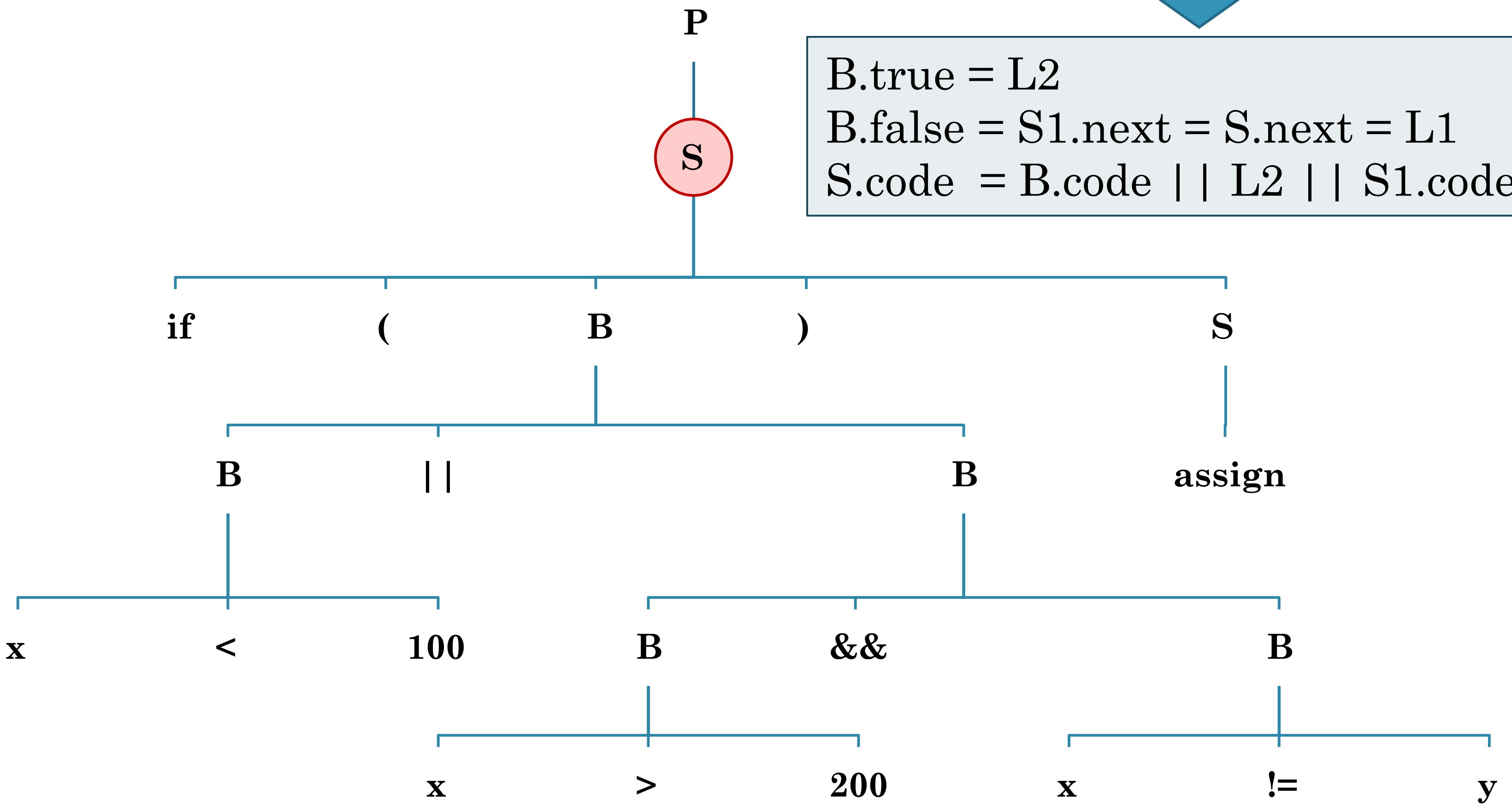
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \;\|\|\; label(S.next)$

S.next = L1
P.code = S.code | | L1



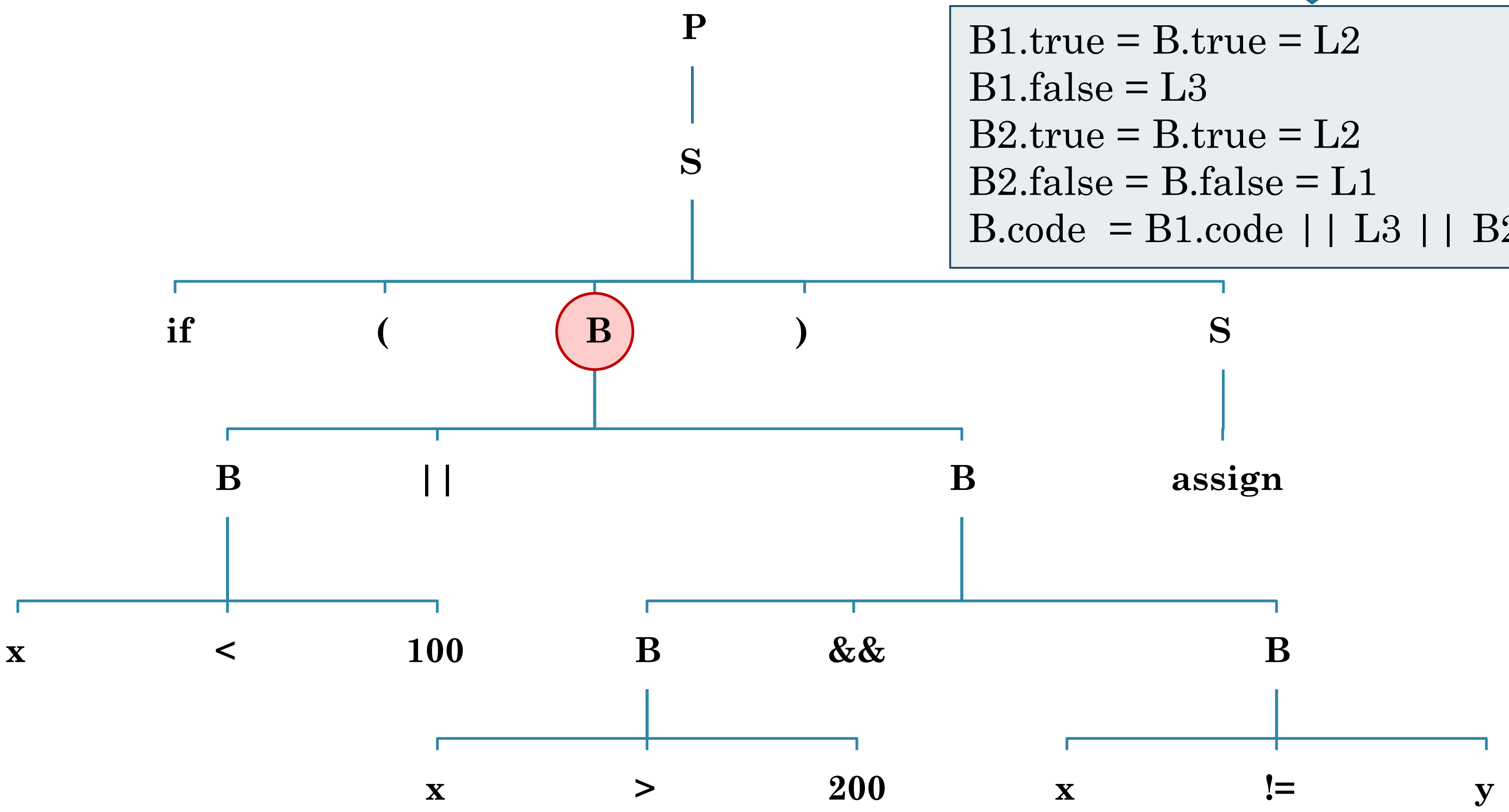
$S \rightarrow \textbf{if} \ (\ B \ ) \ S_1$	$B.true \ = \ newlabel()$ $B.false \ = \ S_1.next \ = \ S.next$ $S.code \ = \ B.code \ \| \| \ label(B.true) \ \| \| \ S_1.code$

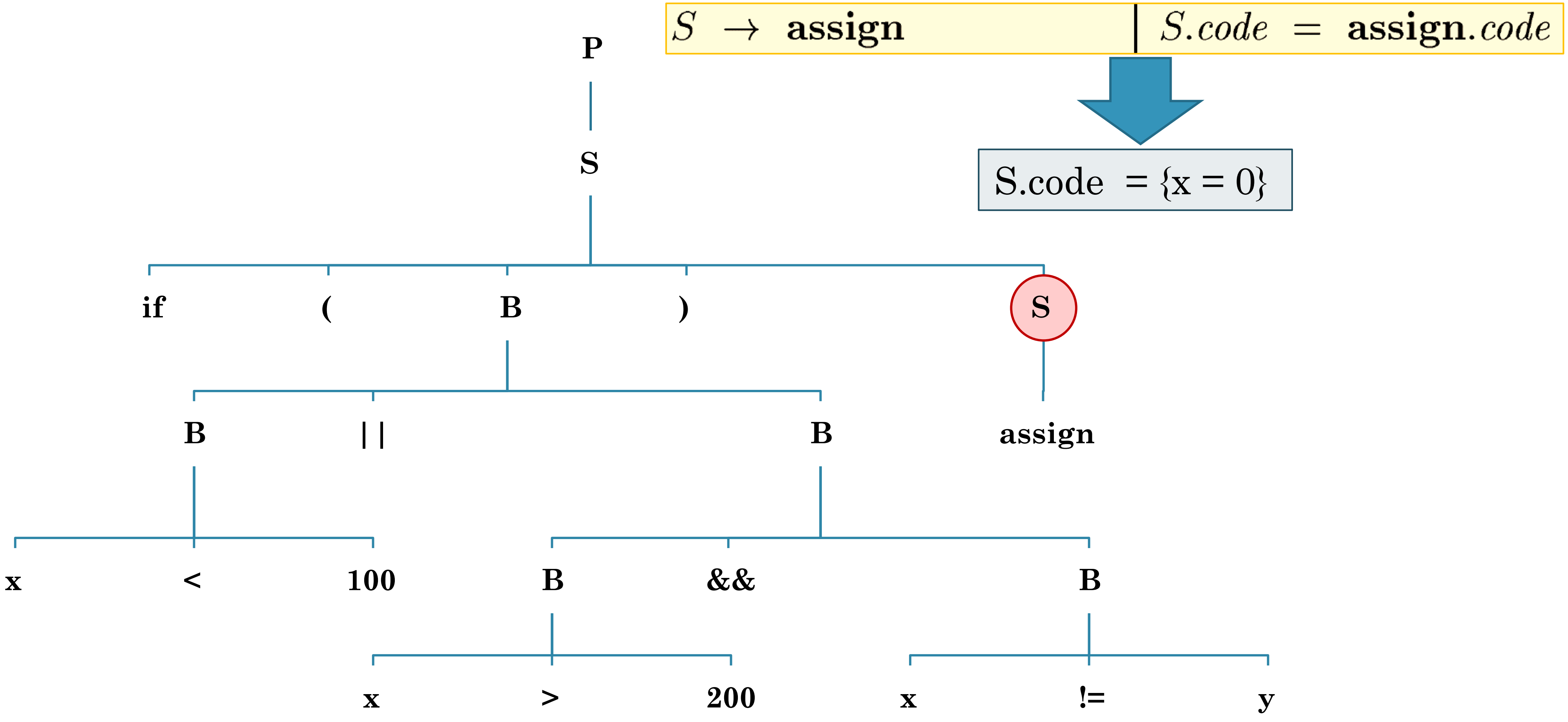
B.true = L2
B.false = S1.next = S.next = L1
S.code = B.code | | L2 | | S1.code



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

B1.true = B.true = L2
B1.false = L3
B2.true = B.true = L2
B2.false = B.false = L1
B.code = B1.code | | L3 | | B2.code



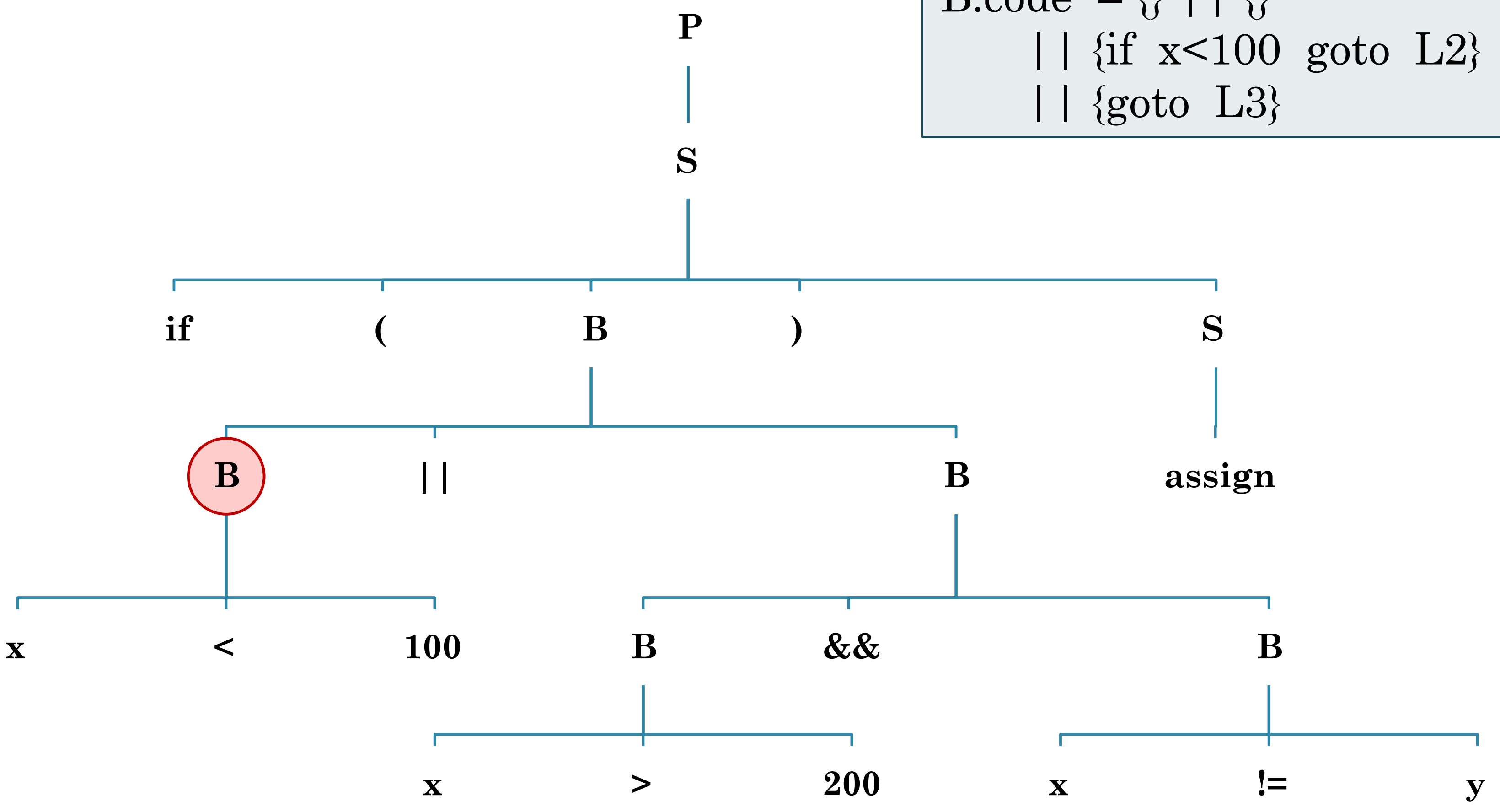


$S \rightarrow \textbf{assign}$ | $S.code = \textbf{assign}.code$

S.code = {x = 0}

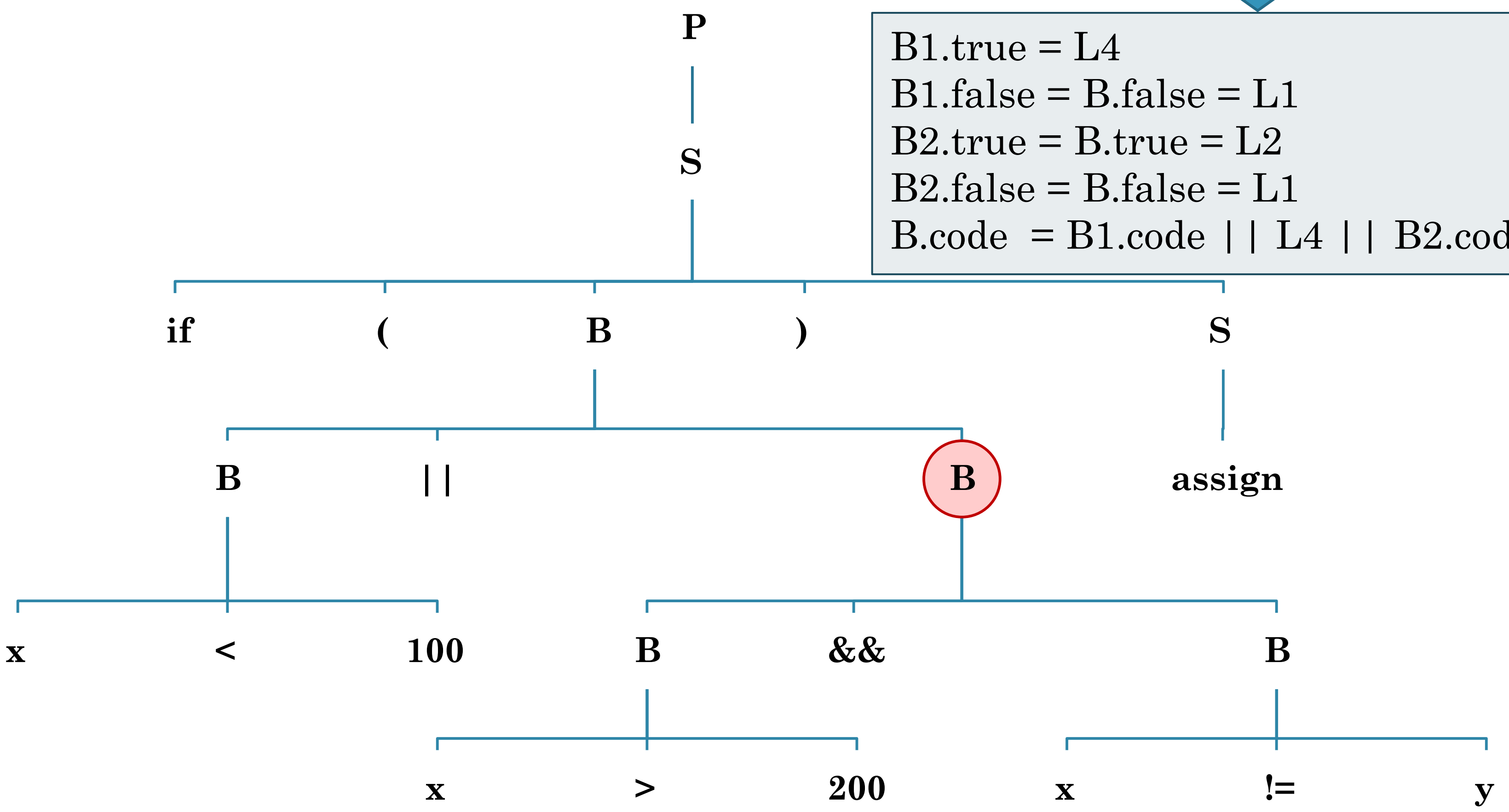
$B \rightarrow E_1$ **rel** E_2	$B.code = E_1.code \;\|\|\; E_2.code$ $\|\| \; gen(\text{'}\mathtt{if}\text{'} \; E_1.addr \; \mathbf{rel}.op \; E_2.addr \; \text{'}\mathtt{goto}\text{'} \; B.true)$ $\|\| \; gen(\text{'}\mathtt{goto}\text{'} \; B.false)$

B.code = {} | | {}
| | {if x<100 goto L2}
| | {goto L3}



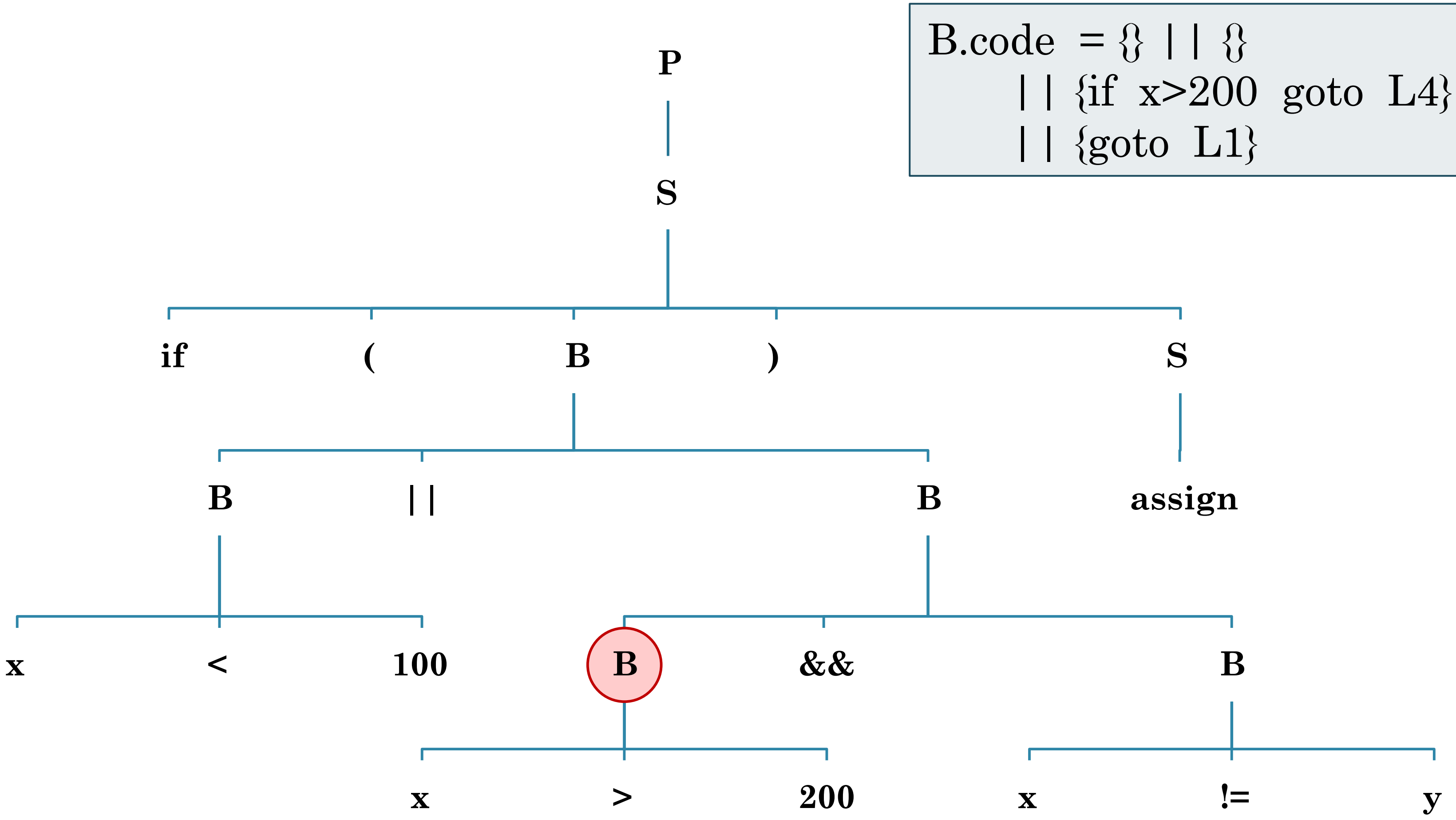
$B \rightarrow B_1 \ \&\& \ B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \ \vert\vert \ label(B_1.true) \ \vert\vert \ B_2.code$

B1.true = L4
B1.false = B.false = L1
B2.true = B.true = L2
B2.false = B.false = L1
B.code = B1.code | | L4 | | B2.code



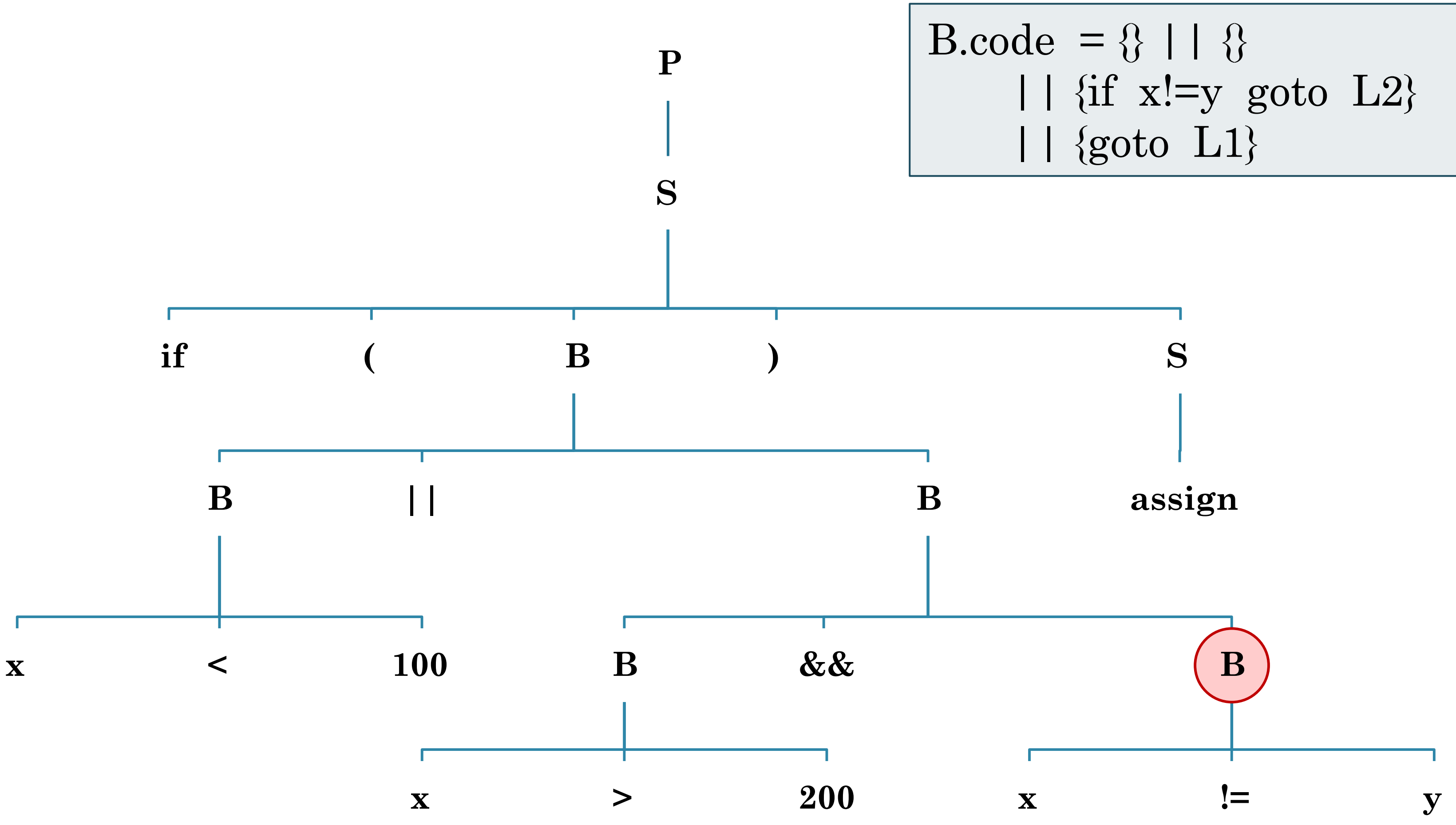
$B \rightarrow E_1\ \mathbf{rel}\ E_2$	$B.code = E_1.code \;\|\|\; E_2.code$ $\|\|\; gen('\texttt{if}'\ E_1.addr\ \mathbf{rel}.op\ E_2.addr\ '\texttt{goto}'\ B.true)$ $\|\|\; gen('\texttt{goto}'\ B.false)$

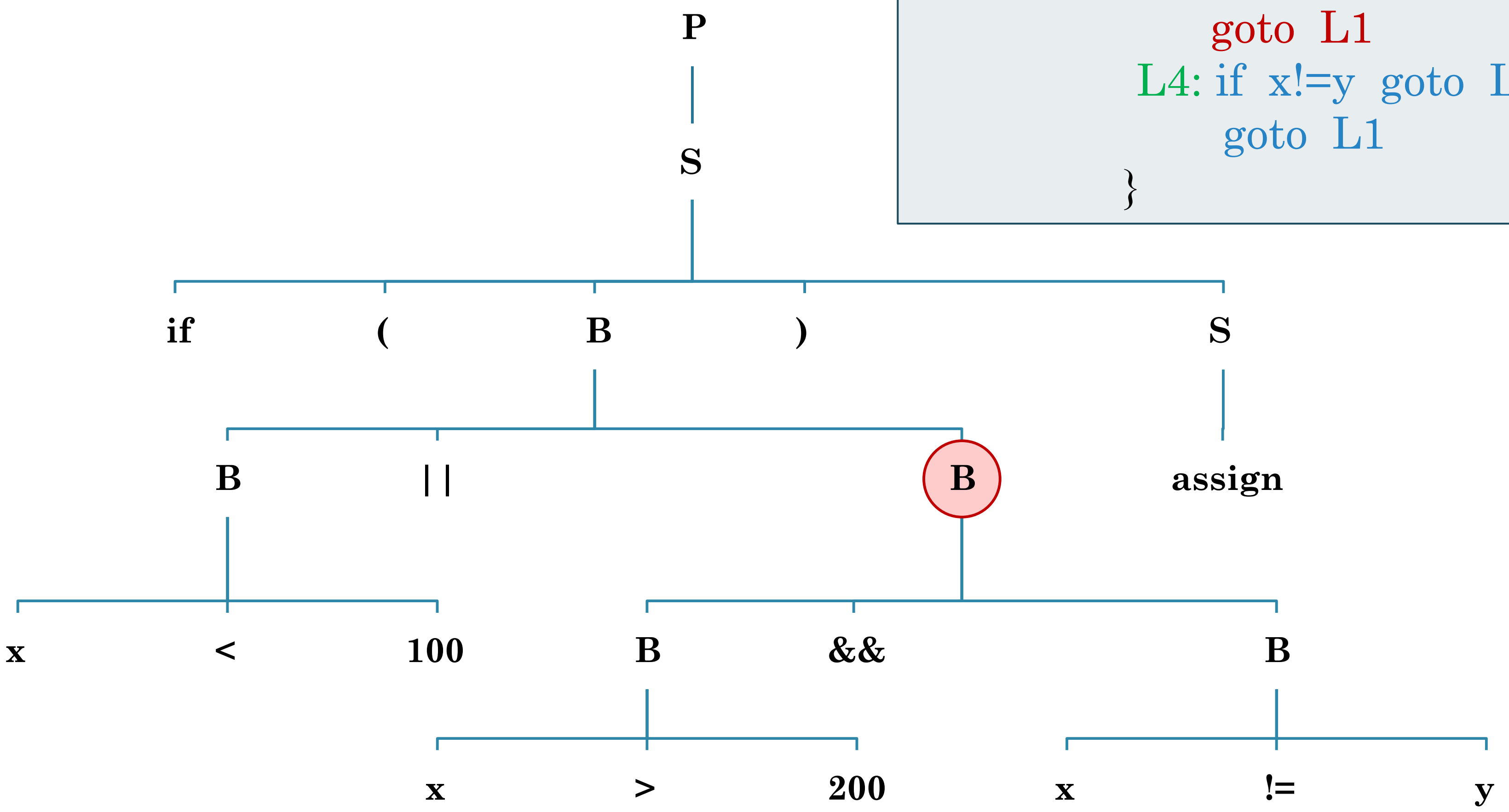
B.code = {} | | {}
| | {if x>200 goto L4}
| | {goto L1}



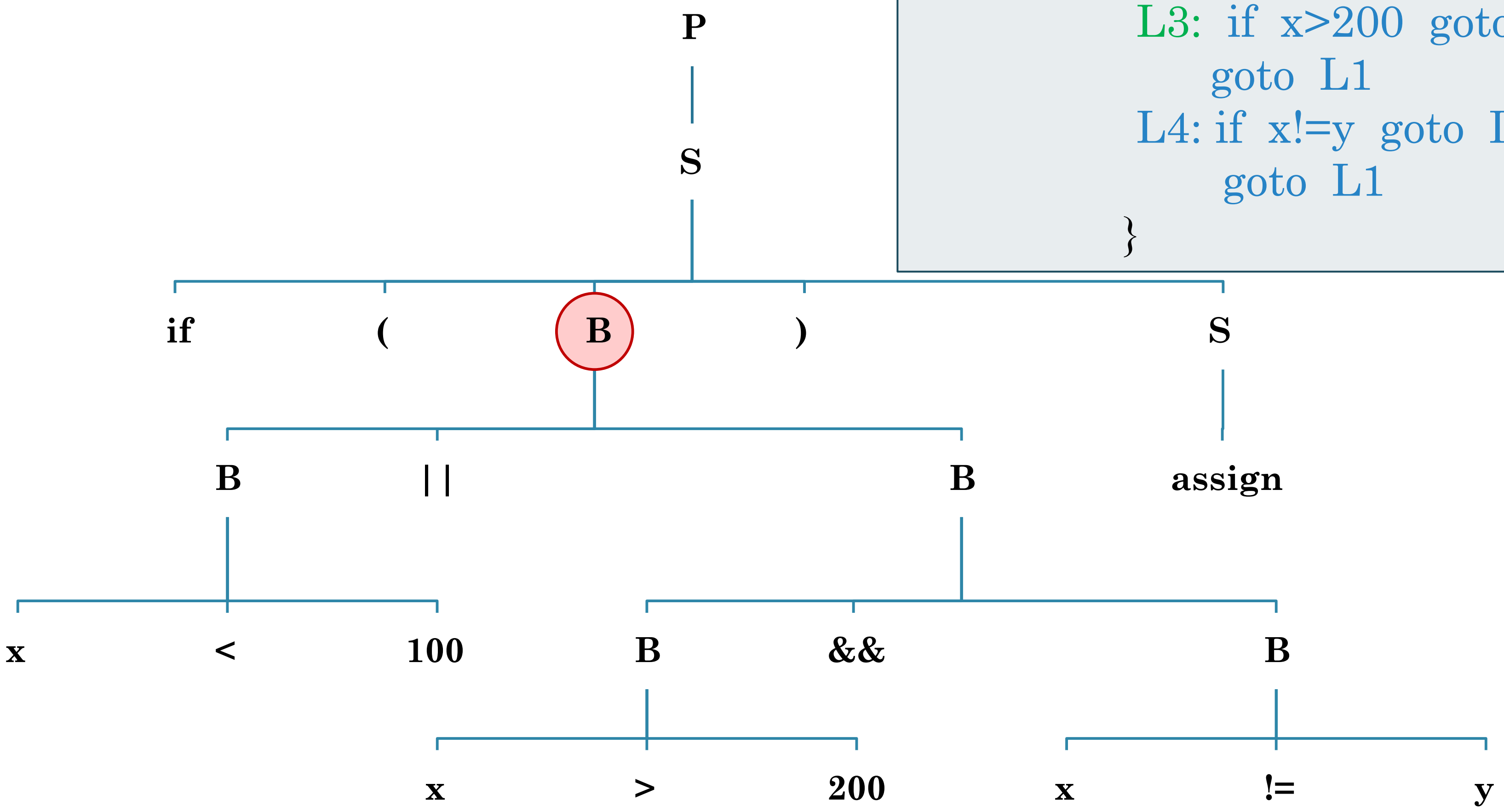
$B \rightarrow E_1 \ \mathbf{rel} \ E_2$	$B.code = E_1.code \ \|\| \ E_2.code$ $\|\| \ gen('\texttt{if}' \ E_1.addr \ \mathbf{rel}.op \ E_2.addr \ '\texttt{goto}' \ B.true)$ $\|\| \ gen('\texttt{goto}' \ B.false)$

B.code = {} | | {}
| | {if x!=y goto L2}
| | {goto L1}



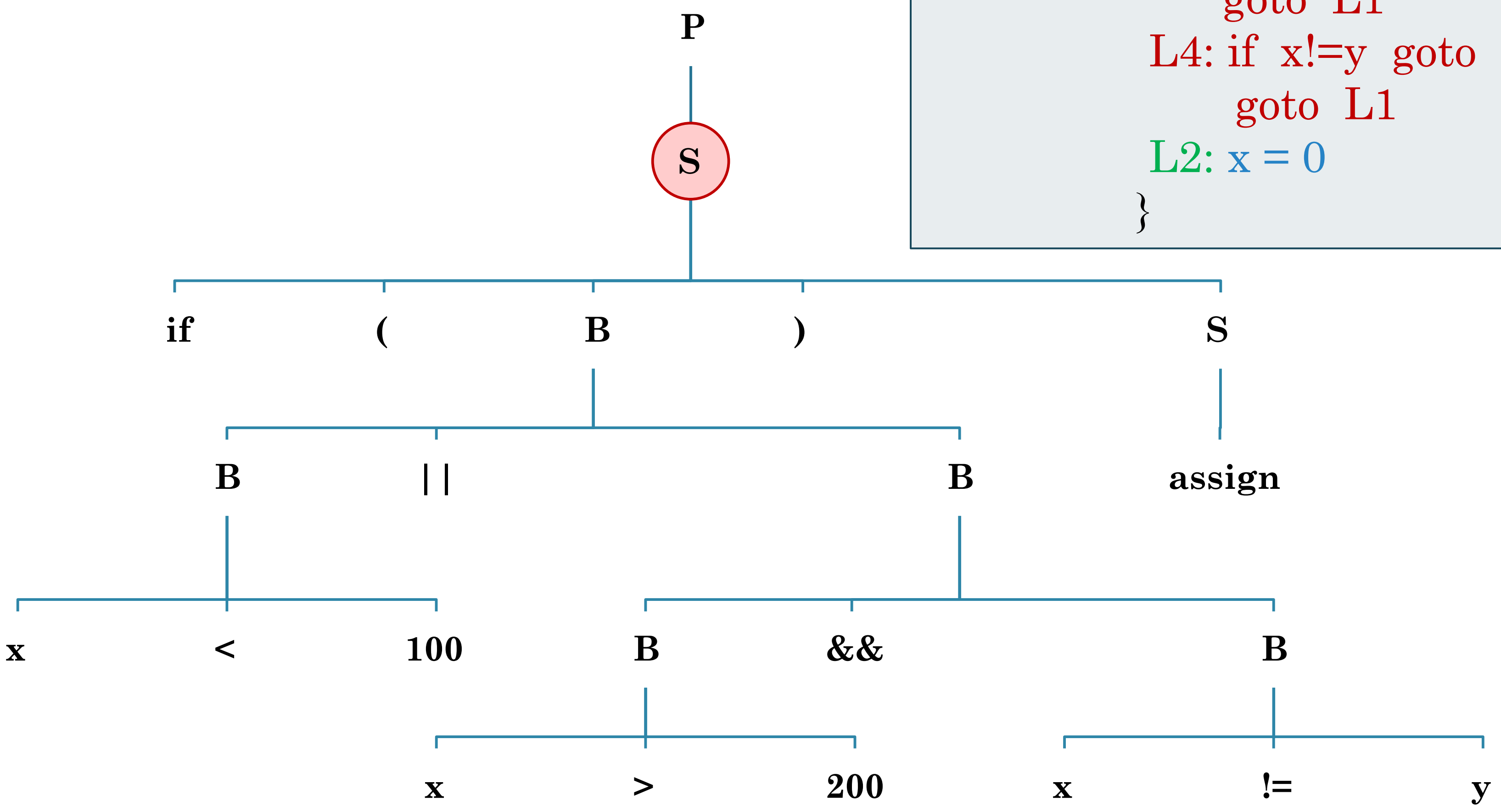


```
B.code  = B1.code | | L4 | | B2.code
        = {
                if  x>200  goto  L4
                goto  L1
            L4: if  x!=y  goto  L2
                goto  L1
          }
```

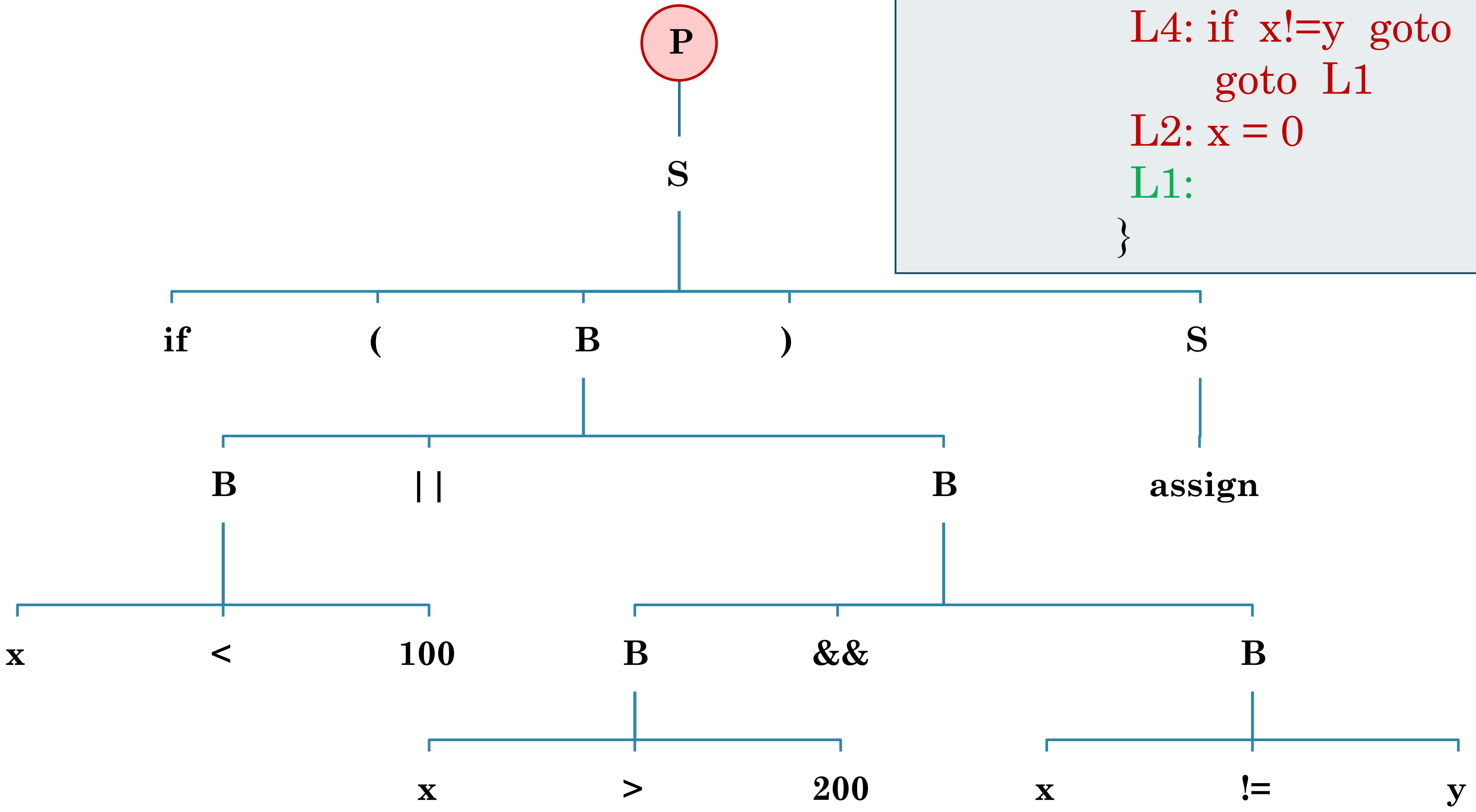


```
B.code  = B1.code | | L3 | | B2.code
        = {
               if  x<100  goto  L2
               goto  L3
           L3:  if  x>200  goto  L4
              goto  L1
           L4: if  x!=y  goto  L2
               goto  L1
          }
```

```
S.code  = B.code | | L2 | | S1.code
        = {
              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
          }
```



```
P.code  = S.code | | L1
        = {
                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:
          }
```



P
S
if
(
B
)
S
B
| |
B
assign
x
<
100
B
&&
B
x
>
200
x
!=
y