南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Chapter 5: Intermediate-Code Generation

Yepang Liu

liuyp1@sustech.edu.cn

Outline

- Intermediate Representation
- Type and Declarations
- Type Checking
- Translation of Expressions
- Control Flow
- Backpatching

Control Flow

- Boolean expressions are often used to alter the flow of control or compute logical values
- Grammar: $B \rightarrow B \parallel B \mid B \,\&\&\, B \mid !\,B \mid (B) \mid E\ \mathbf{rel}\ E \mid \mathbf{true} \mid \mathbf{false}$
- Given the expression $B_1 \parallel B_2$, if B_1 is true, then the expression is true without having to evaluate B_2. In other words, B_1 or B_2 may not need to be evaluated fully.*
- In *short-circuit* code, the boolean operators &&, ‖, ! translate into jumps. The operators do not appear in the code.

If B_1 or B_2 has side effect (e.g., changing the value of a global variable), then the effect may not occur

Short-Circuit Code Example

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

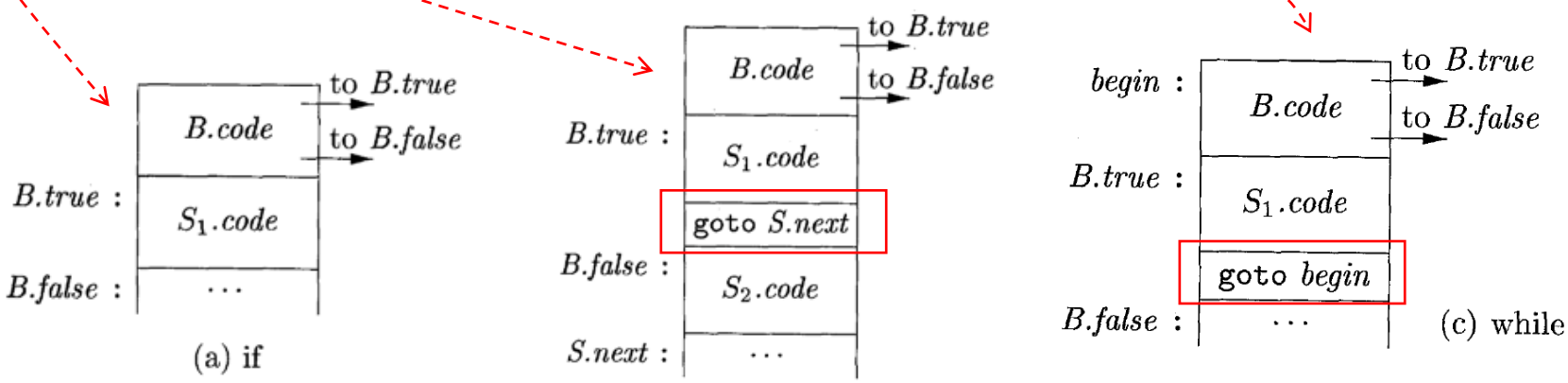
```
        if x < 100 goto L2
        ifFalse x > 200 goto L1
        ifFalse x != y goto L1
L2:     x = 0
L1:
```

Flow-of-Control Statements

- Grammar:
  - $S \rightarrow \textbf{if}\ (\ B\ )\ S_1$
  - $S \rightarrow \textbf{if}\ (\ B\ )\ S_1\ \textbf{else}\ S_2$
  - $S \rightarrow \textbf{while}\ (\ B\ )\ S_1$

Inherited attributes:

- $B.true$: the label to which control flows if B is true
- $B.false$: the label to which control flows if B is false
- $S.next$: the label for the instruction immediately after the code for S



$S.next$ is not needed

$S.next$ is not needed

SDD for Flow-of-Control Statements (1)

PRODUCTION	SEMANTIC RULES
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$
$S \rightarrow \textbf{assign}$	$S.code = \textbf{assign}.code$
$S \rightarrow \textbf{if} (B) S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$S \rightarrow \textbf{if} (B) S_1 \textbf{ else } S_2$	$B.true = newlabel()$ $B.false = newlabel()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ $\mid\mid label(B.true) \mid\mid S_1.code$ $\mid\mid gen('\texttt{goto}'\ S.next)$ $\mid\mid label(B.false) \mid\mid S_2.code$

Illustrated by previous figures

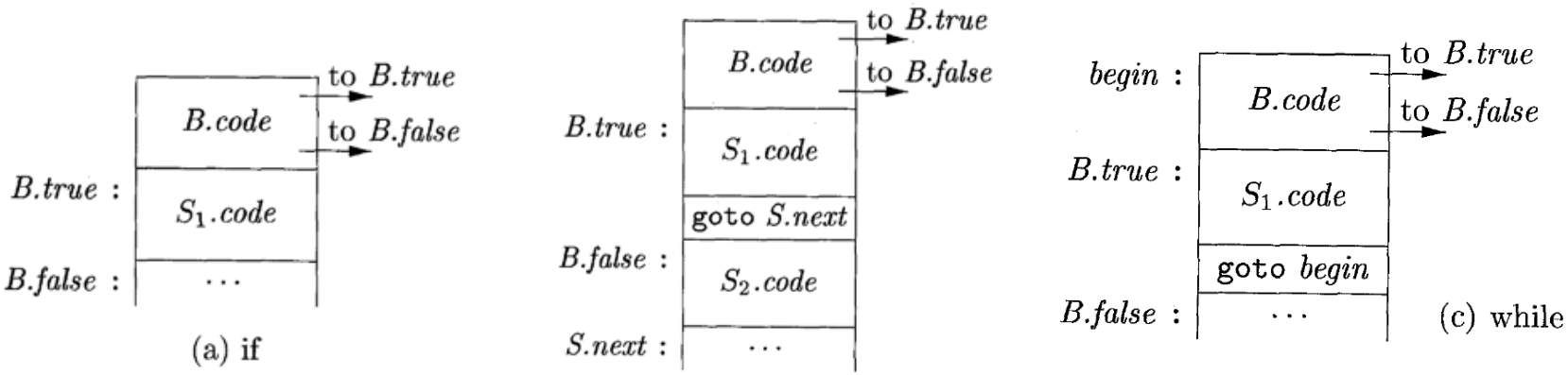
SDD for Flow-of-Control Statements (2)

Illustrated by previous figure

$S \rightarrow \textbf{while} \; (\; B \;) \; S_1$	$\boxed{\begin{aligned} begin &= newlabel() \\ B.true &= newlabel() \\ B.false &= S.next \\ S_1.next &= begin \\ S.code &= label(begin) \mid\mid B.code \\ &\quad \mid\mid label(B.true) \mid\mid S_1.code \\ &\quad \mid\mid gen(\text{'goto'} \; begin) \end{aligned}}$
$S \rightarrow S_1 \; S_2$	$\begin{aligned} S_1.next &= newlabel() \\ S_2.next &= S.next \\ S.code &= S_1.code \mid\mid label(S_1.next) \mid\mid S_2.code \end{aligned}$

Translating Boolean Expressions in Flow-of-Control Statements

- A boolean expression B is translated into <u>three-address instructions that evaluate B using conditional and unconditional jumps</u> to one of two labels: $B.true$ and $B.false$
  - $B.true$ and $B.false$ are two inherited attributes. Their value depends on the context of B (e.g., if statement, if-$else$ statement, $while$ statement)



Generating Three-Address Code for Booleans (1)

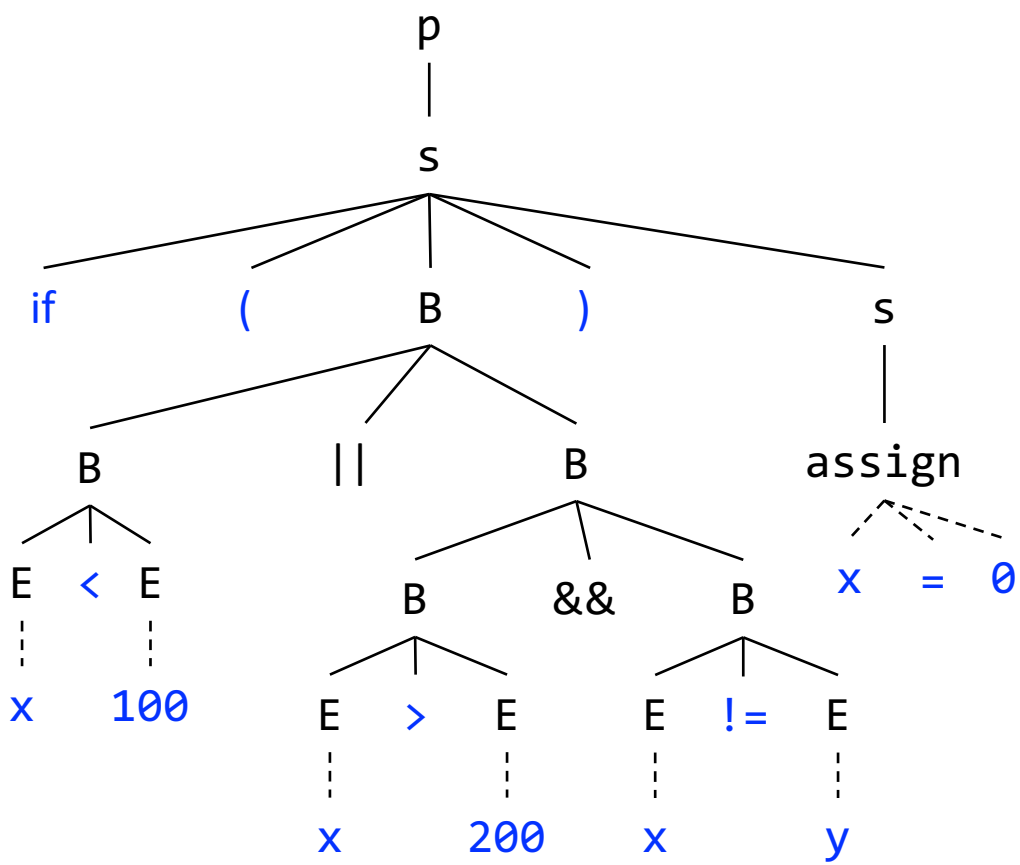
PRODUCTION	SEMANTIC RULES
$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ // short-circuiting $B_1.false = newlabel()$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \,\&\&\, B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ // short-circuiting $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$
$B \rightarrow \,!\, B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$

Generating Three-Address Code for Booleans (2)

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

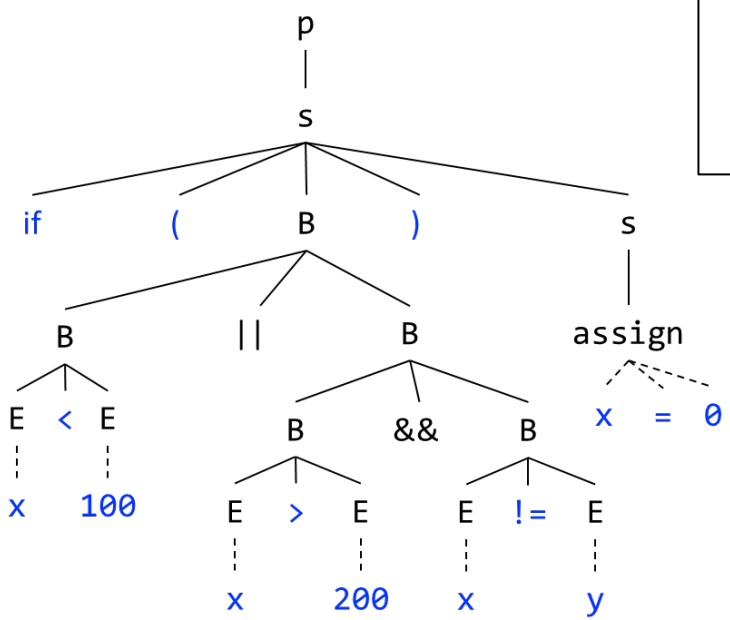
Example

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



Dashed lines mean that the reduction may consist of multiple steps

Example

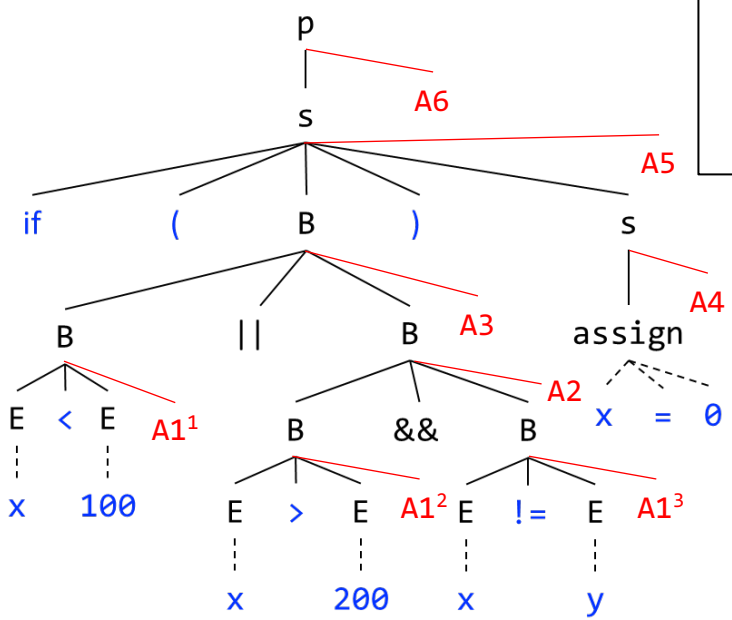


PRODUCTION	SEMANTIC RULES
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$
$S \rightarrow \textbf{assign}$	$S.code = \textbf{assign}.code$
$S \rightarrow \textbf{if} (B) S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \&\& B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$
$B \rightarrow ! B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$
$B \rightarrow E_1 \textbf{ rel } E_2$	$B.code = E_1.code \mid\mid E_2.code$ $\mid\mid gen('\texttt{if}'\ E_1.addr\ \textbf{rel}.op\ E_2.addr\ '\texttt{goto}'\ B.true)$ $\mid\mid gen('\texttt{goto}'\ B.false)$

This SDD is L-attributed, not S-attributed. The grammar is not LL. There is no way to implement the SDD directly during parsing.

Traversing the parse tree to evaluate the attributes helps generate intermidate code

Example



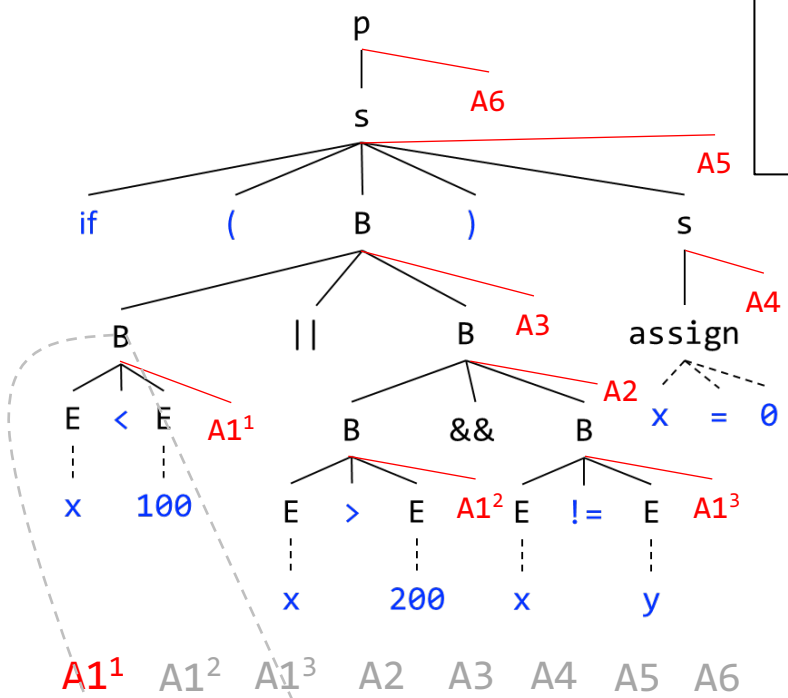
Virtual nodes are in red color

Application order of actions (preorder traversal of the tree):

A1¹ A1² A1³ A2 A3 A4 A5 A6

PRODUCTION	SEMANTIC RULES
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$ A6
$S \rightarrow \textbf{assign}$	$S.code = \textbf{assign}.code$ A4
$S \rightarrow \textbf{if} (B) S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ A5 $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ A3 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \&\& B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ A2 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$
$B \rightarrow ! B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$
$B \rightarrow E_1 \textbf{ rel } E_2$	$B.code = E_1.code \mid\mid E_2.code$ A1 $\mid\mid gen('\texttt{if}'\ E_1.addr\ \textbf{rel}.op\ E_2.addr\ '\texttt{goto}'\ B.true)$ $\mid\mid gen('\texttt{goto}'\ B.false)$

Example

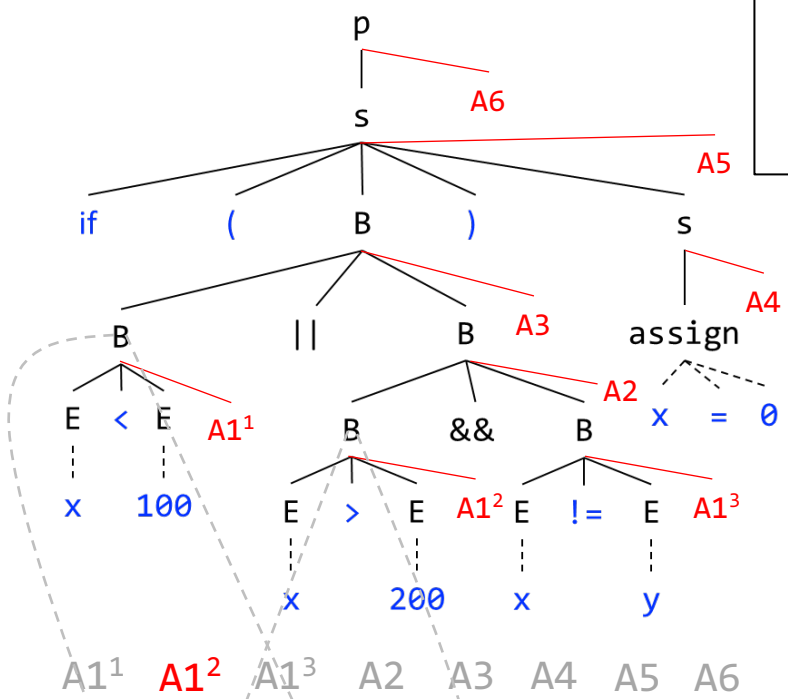


Generated code:

```
if x < 100 goto B.true
goto B.false
```

PRODUCTION	SEMANTIC RULES
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$ A6
$S \rightarrow \textbf{assign}$	$S.code = \textbf{assign}.code$ A4
$S \rightarrow \textbf{if} (B) S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ A5 $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ A3 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \&\& B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ A2 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$
$B \rightarrow ! B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$
$B \rightarrow E_1 \textbf{ rel } E_2$	$B.code = E_1.code \mid\mid E_2.code$ A1 $\mid\mid gen('\texttt{if}'\ E_1.addr\ \textbf{rel}.op\ E_2.addr\ '\texttt{goto}'\ B.true)$ $\mid\mid gen('\texttt{goto}'\ B.false)$

Example

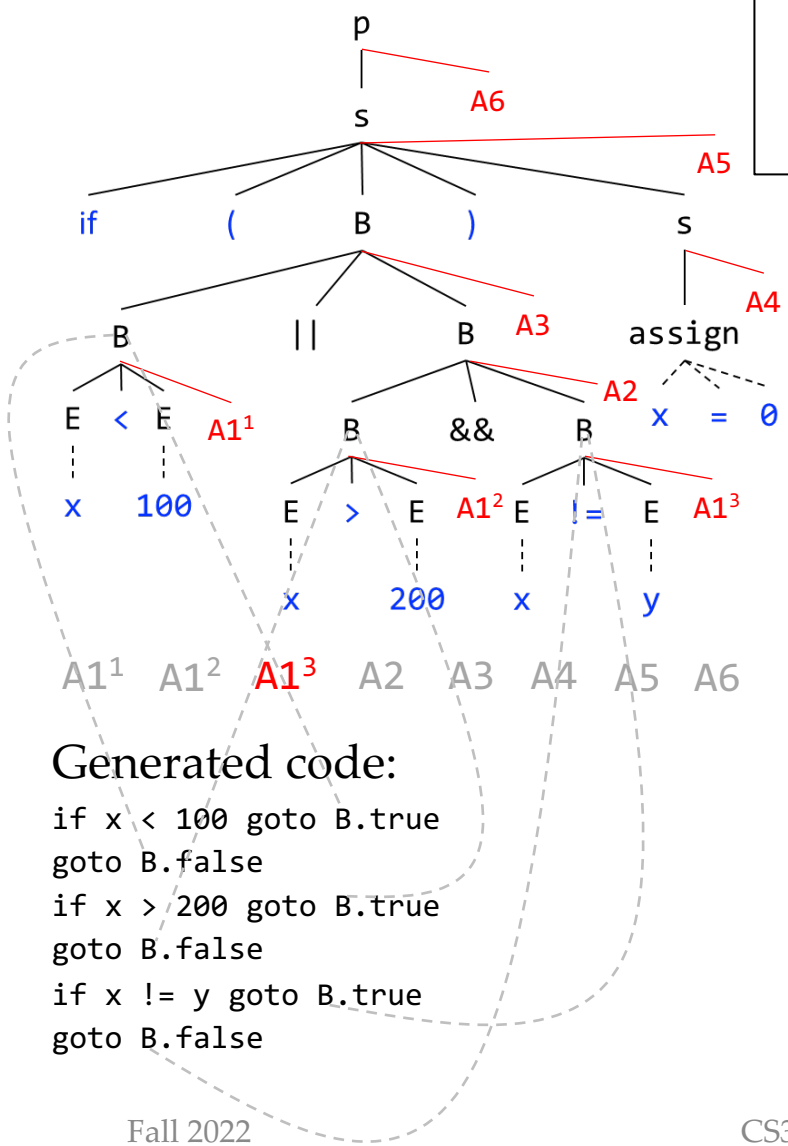


Generated code:

```
if x < 100 goto B.true
goto B.false
if x > 200 goto B.true
goto B.false
```

PRODUCTION	SEMANTIC RULES
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$ A6
$S \rightarrow \textbf{assign}$	$S.code = \textbf{assign}.code$ A4
$S \rightarrow \textbf{if} (B) S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ A5 $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ A3 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \&\& B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ A2 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$
$B \rightarrow ! B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$
$B \rightarrow E_1 \textbf{ rel } E_2$	$B.code = E_1.code \mid\mid E_2.code$ A1 $\mid\mid gen('\texttt{if}'\ E_1.addr\ \textbf{rel}.op\ E_2.addr\ '\texttt{goto}'\ B.true)$ $\mid\mid gen('\texttt{goto}'\ B.false)$

Example

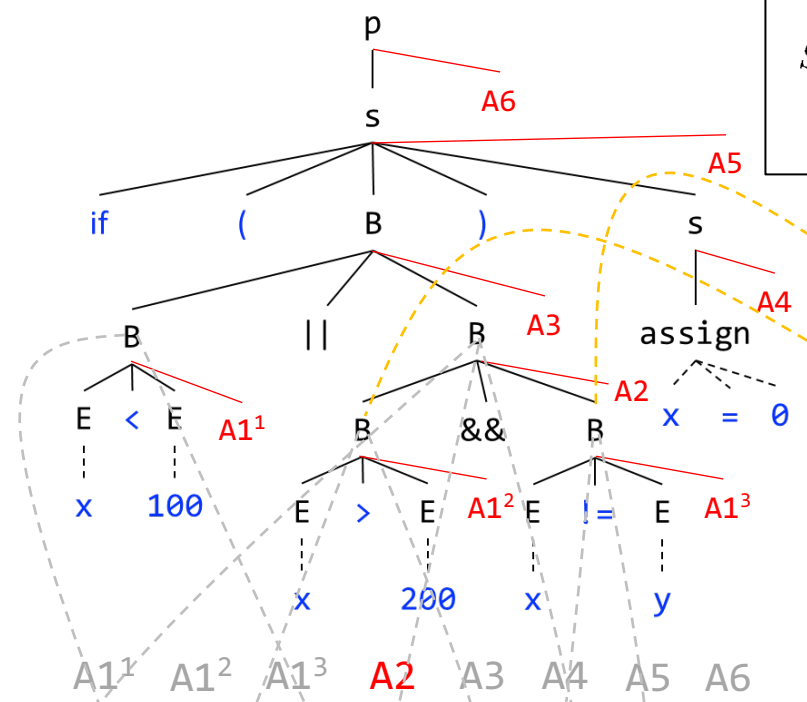


PRODUCTION	SEMANTIC RULES
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$ A6
$S \rightarrow \textbf{assign}$	$S.code = \textbf{assign}.code$ A4
$S \rightarrow \textbf{if} (B) S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ A5 $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ A3 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \&\& B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ A2 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$
$B \rightarrow ! B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$
$B \rightarrow E_1 \textbf{ rel } E_2$	$B.code = E_1.code \mid\mid E_2.code$ A1 $\mid\mid gen('\texttt{if}'\ E_1.addr\ \textbf{rel}.op\ E_2.addr\ '\texttt{goto}'\ B.true)$ $\mid\mid gen('\texttt{goto}'\ B.false)$

Generated code:

```
if x < 100 goto B.true
goto B.false
if x > 200 goto B.true
goto B.false
if x != y goto B.true
goto B.false
```

Example

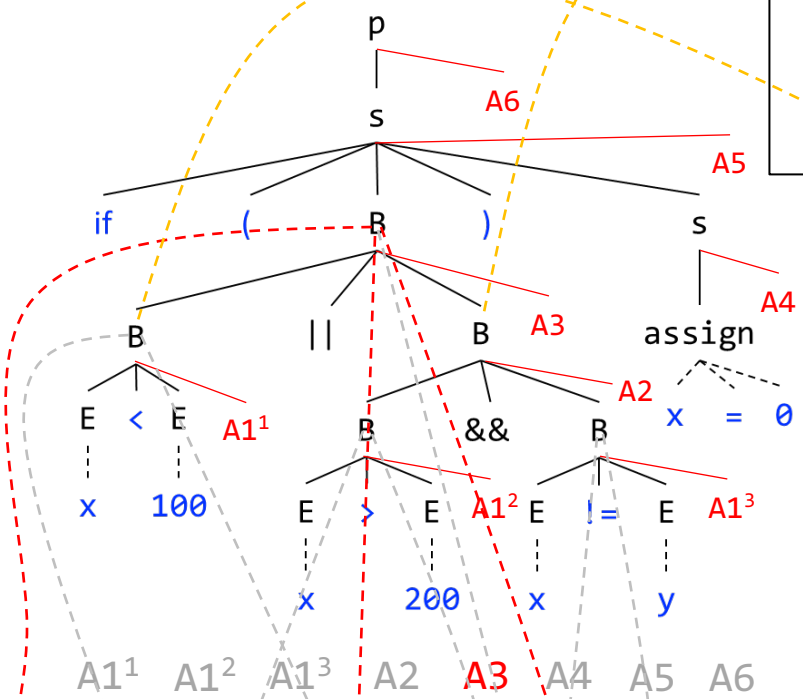


PRODUCTION	SEMANTIC RULES
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$ A6
$S \rightarrow \textbf{assign}$	$S.code = \textbf{assign}.code$ A4
$S \rightarrow \textbf{if} (B) S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ A5 $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ A3 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \&\& B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ A2 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$
$B \rightarrow ! B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$
$B \rightarrow E_1 \textbf{ rel } E_2$	$B.code = E_1.code \mid\mid E_2.code$ A1 $\mid\mid gen('\texttt{if}'\ E_1.addr\ \textbf{rel}.op\ E_2.addr\ '\texttt{goto}'\ B.true)$ $\mid\mid gen('\texttt{goto}'\ B.false)$

Generated code:

```
   if x < 100 goto B.true
   goto B.false
   if x > 200 goto B.true = L4
   goto B.false = B.false
L4: if x != y goto B.true = B.true
   goto B.false = B.false
```

Example

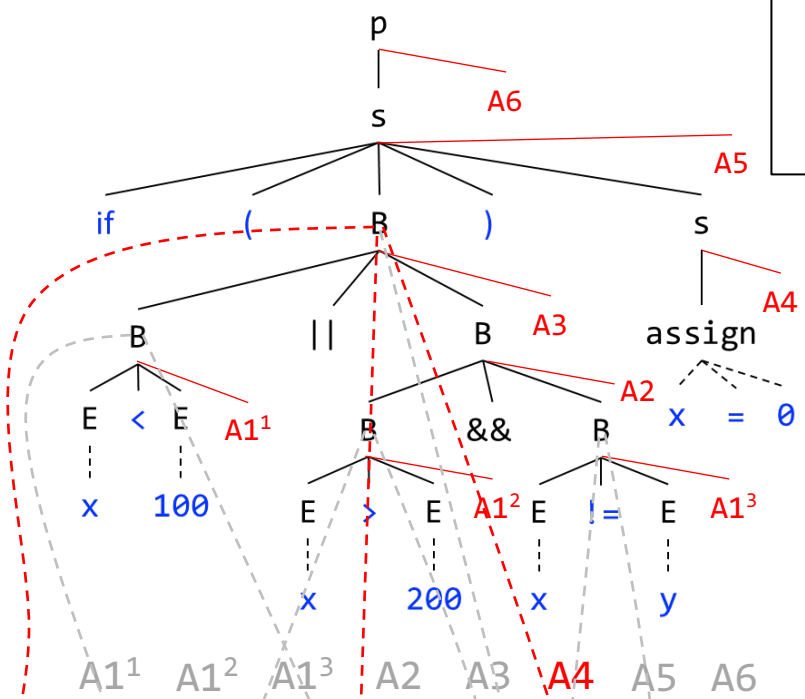


Production	Semantic Rules
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$ A6
$S \rightarrow \textbf{assign}$	$S.code = \textbf{assign}.code$ A4
$S \rightarrow \textbf{if} (B) S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ A5 $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ A3 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \&\& B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ A2 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$
$B \rightarrow ! B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$
$B \rightarrow E_1 \textbf{ rel } E_2$	$B.code = E_1.code \mid\mid E_2.code$ A1 $\mid\mid gen('\texttt{if}'\ E_1.addr\ \textbf{rel}.op\ E_2.addr\ '\texttt{goto}'\ B.true)$ $\mid\mid gen('\texttt{goto}'\ B.false)$

Generated code:

```
   if x < 100 goto B.true = B.true
   goto B.false = L3
L3: if x > 200 goto B.true = L4
   goto B.false = B.false
L4: if x != y goto B.true = B.true
   goto B.false = B.false
```

Example

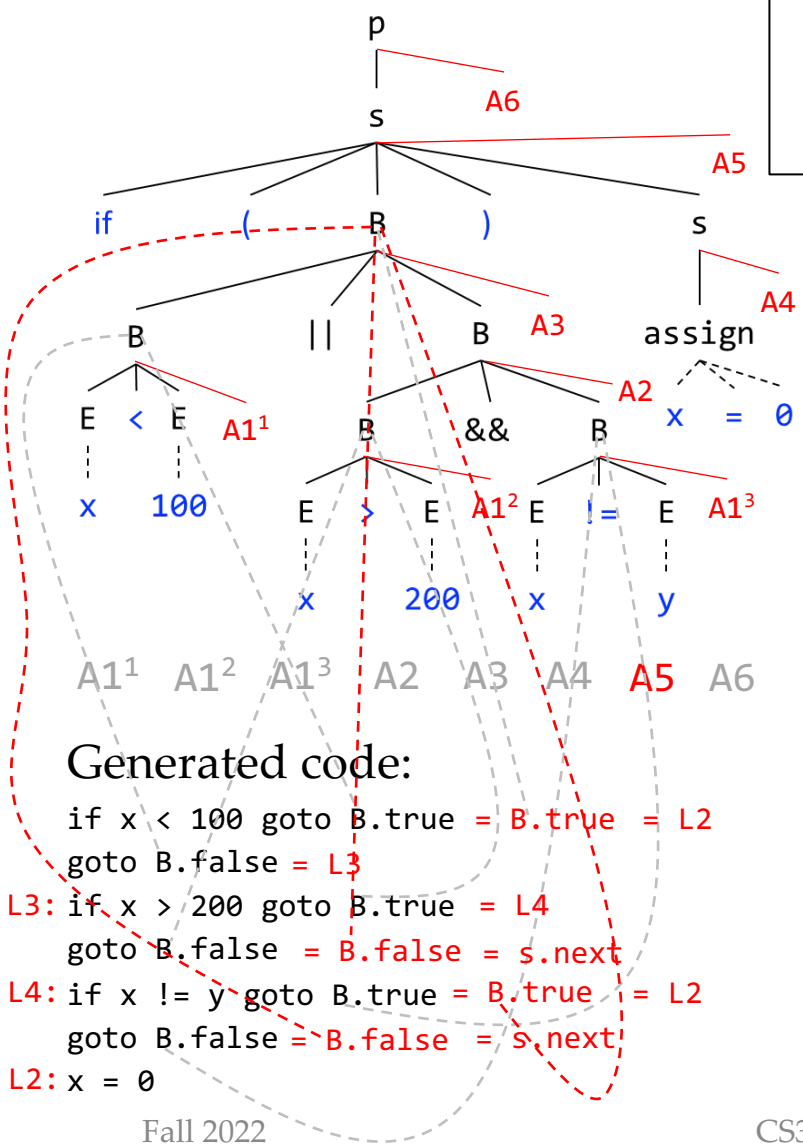


Generated code:

```
    if x < 100 goto B.true = B.true
    goto B.false = L3
L3: if x > 200 goto B.true = L4
    goto B.false = B.false
L4: if x != y goto B.true = B.true
    goto B.false = B.false
    x = 0
```

PRODUCTION	SEMANTIC RULES
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$ A6
$S \rightarrow$ **assign**	$S.code =$ **assign**$.code$ A4
$S \rightarrow$ **if** $(B) S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ A5 $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ A3 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \&\& B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ A2 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$
$B \rightarrow ! B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$
$B \rightarrow E_1$ **rel** E_2	$B.code = E_1.code \mid\mid E_2.code$ A1 $\mid\mid gen('\texttt{if}'\ E_1.addr\ \textbf{rel}.op\ E_2.addr\ '\texttt{goto}'\ B.true)$ $\mid\mid gen('\texttt{goto}'\ B.false)$

Example

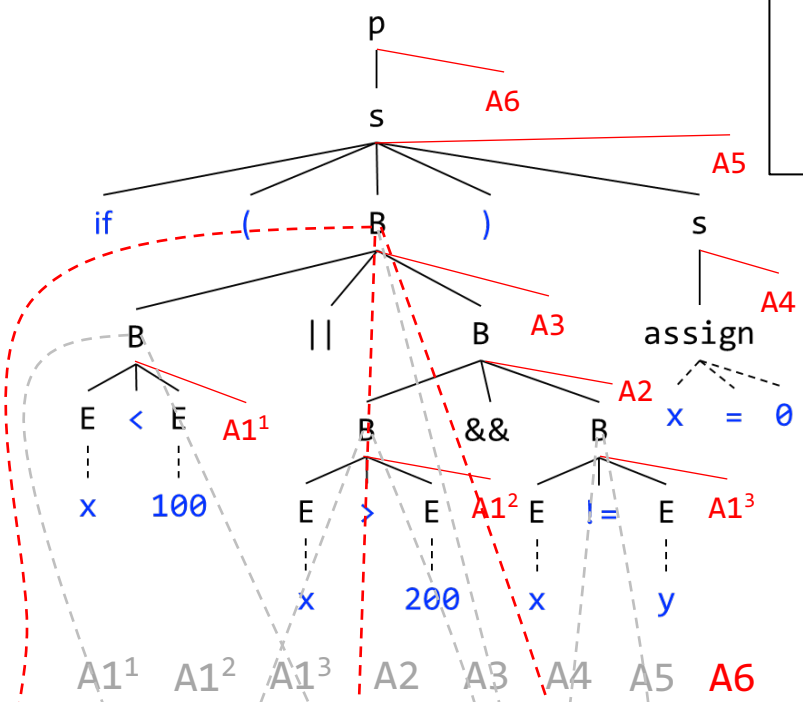


Generated code:

```
   if x < 100 goto B.true = B.true = L2
   goto B.false = L3
L3: if x > 200 goto B.true = L4
   goto B.false = B.false = s.next
L4: if x != y goto B.true = B.true = L2
   goto B.false = B.false = s.next
L2: x = 0
```

PRODUCTION	SEMANTIC RULES
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$ A6
$S \rightarrow$ **assign**	$S.code =$ **assign**$.code$ A4
$S \rightarrow$ **if** $(B) S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ A5 $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ A3 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \&\& B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ A2 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$
$B \rightarrow ! B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$
$B \rightarrow E_1$ **rel** E_2	$B.code = E_1.code \mid\mid E_2.code$ A1 $\mid\mid gen('\texttt{if}'\ E_1.addr\ \textbf{rel}.op\ E_2.addr\ '\texttt{goto}'\ B.true)$ $\mid\mid gen('\texttt{goto}'\ B.false)$

Example



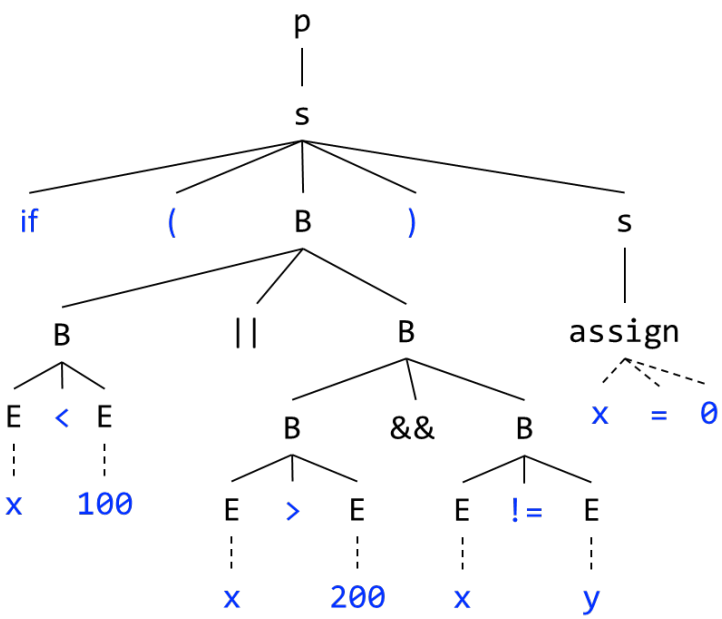
Production	Semantic Rules
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \mid\mid label(S.next)$ A6
$S \rightarrow$ **assign**	$S.code =$ **assign**$.code$ A4
$S \rightarrow$ **if** $(B) S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ A5 $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$
$B \rightarrow B_1 \mid\mid B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ A3 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code$
$B \rightarrow B_1 \&\& B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ A2 $B_2.false = B.false$ $B.code = B_1.code \mid\mid label(B_1.true) \mid\mid B_2.code$
$B \rightarrow ! B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$
$B \rightarrow E_1$ **rel** E_2	$B.code = E_1.code \mid\mid E_2.code$ A1 $\mid\mid gen('\texttt{if}'\ E_1.addr\ \textbf{rel}.op\ E_2.addr\ '\texttt{goto}'\ B.true)$ $\mid\mid gen('\texttt{goto}'\ B.false)$

Generated code:

```
   if x < 100 goto B.true = B.true = L2
   goto B.false = L3
L3: if x > 200 goto B.true = L4
   goto B.false = B.false = s.next
L4: if x != y goto B.true = B.true = L2
   goto B.false = B.false = s.next = L1
L2: x = 0
L1: …
```

Example

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



Generated 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
L1:
```

Outline

- Intermediate Representation
- Type and Declarations
- Type Checking
- Translation of Expressions
- Control Flow
- Backpatching

Backpatching (回填)

- A **key problem** when generating code for boolean expressions and flow-of-control statements is to match a jump instruction with the jump target

- **Example: if (*B*) *S***
  - According to the short-circuit translation, B's code contains a jump to the instruction following the code for S (executed when B is false)
  - However, B must be translated before S. The jump target is unknown when translating B
  - Earlier, we address the problem by passing labels as inherited attributes ($S.next$), but this requires <u>another separate pass (traversing the parse tree)</u> after parsing

How to address the problem in one pass?

One-Pass Code Generation Using Backpatching

- **Basic idea of backpatching (基本思想):**
  - When a jump is generated, its target is temporarily left unspecified.
  - Incomplete jumps are grouped into lists. All jumps on a list have the same target.
  - Fill in the labels for incomplete jumps when the targets become known.
- **The technique (技术细节):**
  - For a nonterminal B that represents a boolean expression, we define two synthesized attributes: $truelist$ and $falselist$
  - $truelist$: a list of jump instructions whose target is the label to which the control goes when B is true
  - $falselist$: a list of jump instructions whose target is the label to which the control goes when B is false

One-Pass Code Generation Using Backpatching

- **The technique (技术细节) Cont.:**
  - $makelist(i)$: create a new list containing only i, the index of a jump instruction, and return the pointer to the list
  - $merge(p_1, p_2)$: concatenate the lists pointed by p_1 and p_2, and return a pointer to the concatenated list
  - $backpatch(p, i)$: insert i as the target for each of the jump instructions on the list pointed by p

Backpatching for Boolean Expressions (布尔表达式的回填)

- An SDT suitable for generating code for boolean expressions during bottom-up parsing
- Grammar:
  - $B \rightarrow B_1 \parallel MB_2 \mid B_1 \,\&\&\, MB_2 \mid !\,B_1 \mid (B_1) \mid E_1 \textbf{ rel } E_2 \mid \textbf{true} \mid \textbf{false}$
  - $M \rightarrow \epsilon$

Keep this question in mind: Why do we introduce M before B_2?

1)	$B \rightarrow B_1 \ \mid\mid \ M \ B_2$	$\{ \ backpatch(B_1.falselist, M.instr);$ $B.truelist = merge(B_1.truelist, B_2.truelist);$ $B.falselist = B_2.falselist; \ \}$
2 2)	$B \rightarrow B_1 \ \&\& \ M \ B_2$	$\{ \ backpatch(B_1.truelist, M.instr);$ $B.truelist = B_2.truelist;$ $B.falselist = merge(B_1.falselist, B_2.falselist); \ \}$
3)	$B \rightarrow \ ! \ B_1$	$\{ \ B.truelist = B_1.falselist;$ $B.falselist = B_1.truelist; \ \}$
4)	$B \rightarrow (\ B_1 \ )$	$\{ \ B.truelist = B_1.truelist;$ $B.falselist \ = \ B_1.falselist; \ \}$
1 5)	$B \rightarrow E_1 \ \mathbf{rel} \ E_2$	$\{ \ B.truelist = makelist(nextinstr);$ $B.falselist = makelist(nextinstr + 1);$ $gen('\texttt{if}' \ E_1.addr \ \mathbf{rel}.op \ E_2.addr \ '\texttt{goto \_}');$ $gen('\texttt{goto \_}'); \ \}$
6)	$B \rightarrow \mathbf{true}$	$\{ \ B.truelist = makelist(nextinstr);$ $gen('\texttt{goto \_}'); \ \}$
7)	$B \rightarrow \mathbf{false}$	$\{ \ B.falselist = makelist(nextinstr);$ $gen('\texttt{goto \_}'); \ \}$
8)	$M \rightarrow \epsilon$	$\{ \ M.instr = nextinstr; \ \}$

Tip: understand 1 and 2 at a high level first and then revisit this slide after you understand the later examples.

Backpatching vs. Non-Backpatching (1)

(1) Non-backpatching SDD with inherited attributes:

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

(2) Backpatching scheme:

$B \rightarrow E_1 \ \mathbf{rel} \ E_2$	$\{\ B.truelist = makelist(nextinstr);$ $B.falselist = makelist(nextinstr + 1);$ $gen(\texttt{'if'}\ E_1.addr\ \mathbf{rel}.op\ E_2.addr\ \texttt{'goto \_'});$ $gen(\texttt{'goto \_'});\ \}$

Comparison:

- In (2), incomplete instructions (指令坯) are added to corresponding lists
- The instruction jumping to $B.true$ in (1) is added to $B.truelist$ in (2)
- The instruction jumping to $B.false$ in (1) is added to $B.falselist$ in (2)

Backpatching vs. Non-Backpatching (2)

(1) Non-backpatching SDD with inherited attributes:

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

(2) Backpatching scheme:

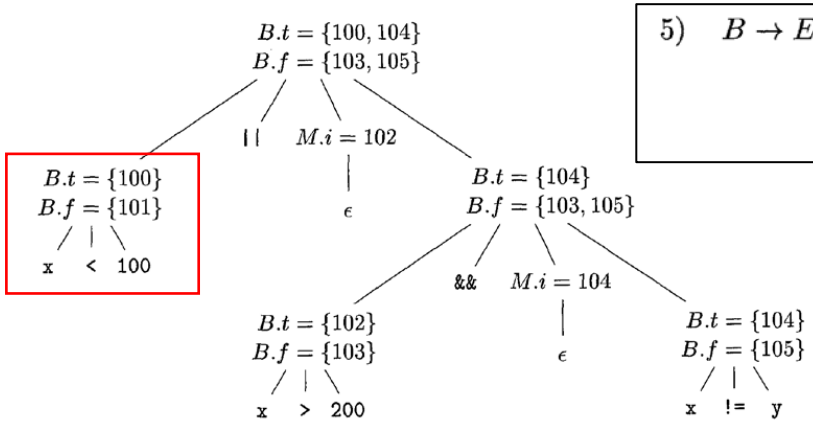
$B \rightarrow B_1 \mid\mid M\ B_2$	$\{\ backpatch(B_1.falselist, M.instr);$ $B.truelist = merge(B_1.truelist, B_2.truelist);$ $B.falselist = B_2.falselist;\ \}$

Comparison:

- The assignments to $true/false$ attributes in (1) correspond to the assignments to $truelist/falselist$ or $merge$ in (2)

Example – Boolean Expressions

- The earlier SDT is a postfix SDT. The semantic actions can be performed during a bottom-up parse.
- Boolean expression: $x < 100 \parallel x > 200 \;\&\&\; x != y$
- **Step 1:** reduce $x < 100$ to B by production (5)



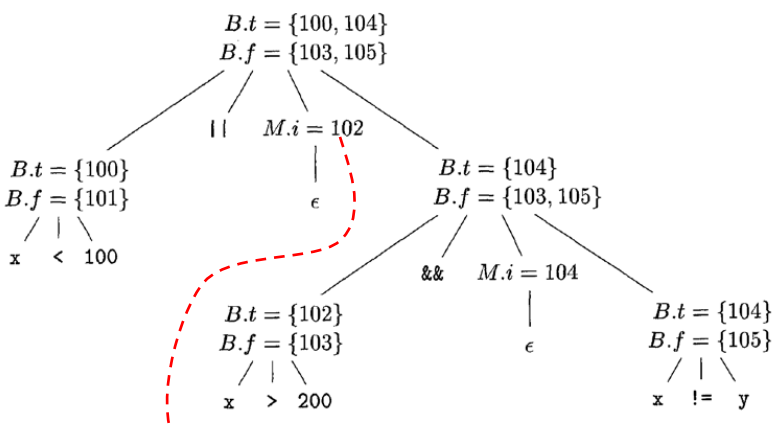
5)	$B \rightarrow E_1$ **rel** E_2	{ $B.truelist = makelist(nextinstr)$; $B.falselist = makelist(nextinstr + 1)$; $gen($'`if`' $E_1.addr$ **rel**$.op$ $E_2.addr$ '`goto _`'$)$; $gen($'`goto _`'$)$; }

Generated code:

```
100:   if x < 100 goto _
101:   goto _
```

Example – Boolean Expressions

- The earlier SDT is a postfix SDT. The semantic actions can be performed during a bottom-up parse.
- Boolean expression: $x < 100 \parallel x > 200 \,\&\&\, x != y$
- **Step 2:** reduce ϵ to M by production (8)

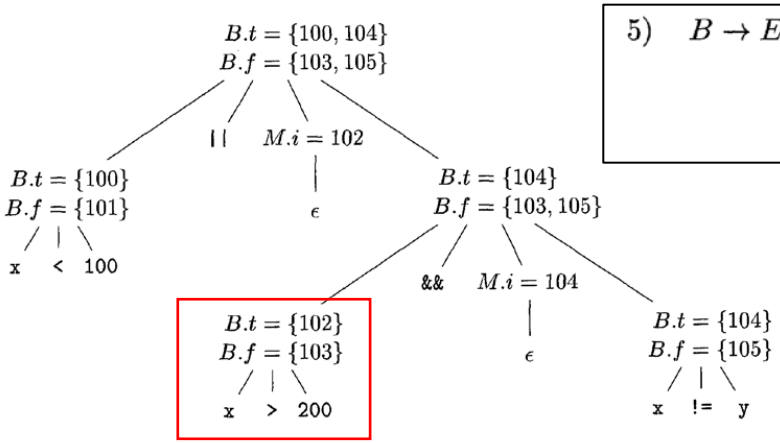


8)	$M \rightarrow \epsilon$	$\{\ M.instr = nextinstr;\ \}$

The marker nonterminal records the value of *nextinstr*, 102

Example – Boolean Expressions

- Boolean expression: $x < 100 \parallel x > 200 \;\&\&\; x\,!= y$
- **Step 3:** reduce $x > 200$ to B by production (5)



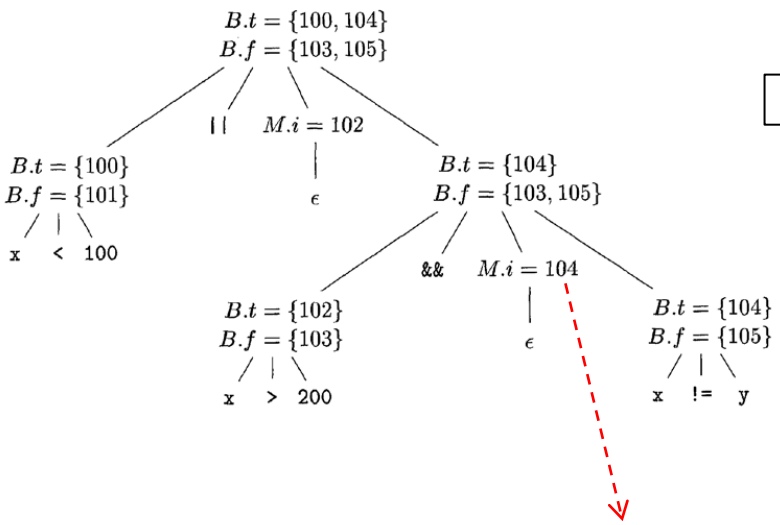
5)	$B \rightarrow E_1$ **rel** E_2	{ $B.truelist = makelist(nextinstr)$; $B.falselist = makelist(nextinstr + 1)$; gen('`if`' $E_1.addr$ **rel**.op $E_2.addr$ '`goto _`'); gen('`goto _`'); }

Generated code:

```
102:   if x > 200 goto _
103:   goto _
```

Example – Boolean Expressions

- Boolean expression: $x < 100 \parallel x > 200 \;\&\&\; x\,!=y$
- **Step 4:** reduce ϵ to M by production (8)

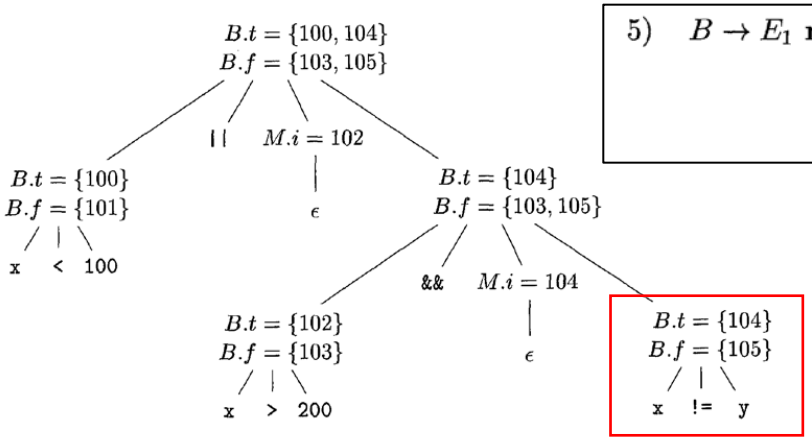


8)	$M \rightarrow \epsilon$	$\{\ M.instr = nextinstr;\ \}$

The marker nonterminal records the value of *nextinstr*, 104

Example – Boolean Expressions

- Boolean expression: $x < 100 \parallel x > 200$ && $x != y$
- **Step 5:** reduce $x! = y$ to B by production (5)



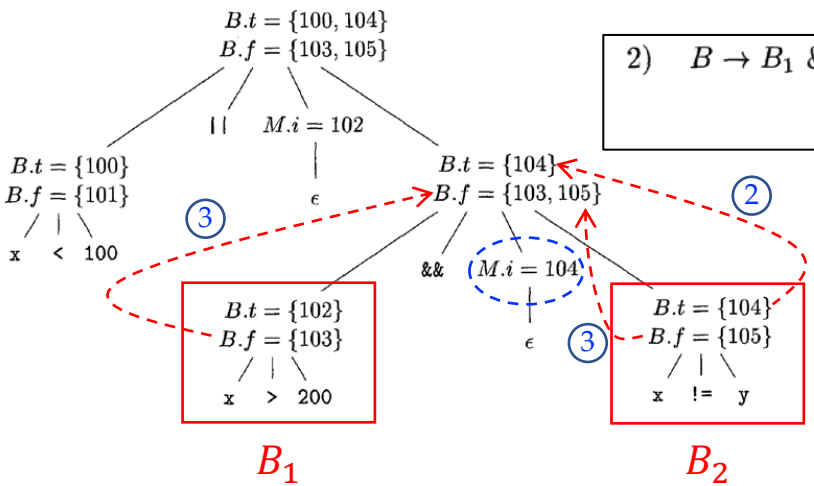
5)	$B \rightarrow E_1$ **rel** E_2	{ $B.truelist = makelist(nextinstr)$; $B.falselist = makelist(nextinstr + 1)$; gen('`if`' $E_1.addr$ **rel**.op $E_2.addr$ '`goto _`'); gen('`goto _`'); }

Generated code:

```
104:   if x != y goto _
105:   goto _
```

Example – Boolean Expressions

- Boolean expression: $x < 100 \parallel x > 200 \;\&\&\; x\,!= y$
- **Step 6:** reduce $B_1 \;\&\&\; MB_2$ to B by production (2)



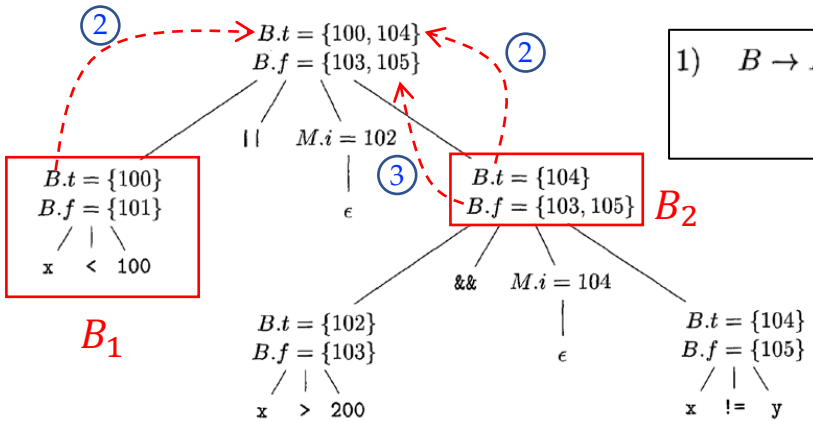
2)	$B \rightarrow B_1 \;\&\&\; M\; B_2$	{ $backpatch(B_1.truelist, M.instr)$; ①
		$B.truelist = B_2.truelist$; ②
		$B.falselist = merge(B_1.falselist, B_2.falselist)$; ③

Backpatch instruction 102: ①

```
100:  if x < 100 goto _
101:  goto _
102:  if x > 200 goto 104
103:  goto _
104:  if x != y goto _
105:  goto _
```

Example – Boolean Expressions

- Boolean expression: $x < 100 \parallel x > 200 \;\&\&\; x != y$
- **Step 7:** reduce $B_1 \parallel MB_2$ to B by production (1)



1) $B \rightarrow B_1 \mid\mid M\ B_2$ { $backpatch(B_1.falselist, M.instr)$; 1
$B.truelist = merge(B_1.truelist, B_2.truelist)$; 2
$B.falselist = B_2.falselist$; } 3

Backpatch instruction 101: 1

```
100:  if x < 100 goto _
101:  goto 102
102:  if x > 200 goto 104
103:  goto _
104:  if x != y goto _
105:  goto _
```

The remaining jump targets will be filled in later parsing steps

Reading Tasks

- Chapter 6 of the dragon book
  - 6.1.1 Directed Acyclic Graphs for Expressions
  - 6.2 Three-Address Code
  - 6.3 Types and Declarations
  - 6.4 Translation of Expressions
  - 6.5 Type Checking (6.5.1 – 6.5.2)
  - 6.6 Control Flow (6.6.1 – 6.6.4)
  - 6.7 Backpatching (6.7.1 – 6.7.3)