



UNIVERSITÀ
DEGLI STUDI
DI TORINO

Generazione di codice intermedio

(espressioni booleane)

Traduzione di espressioni booleane

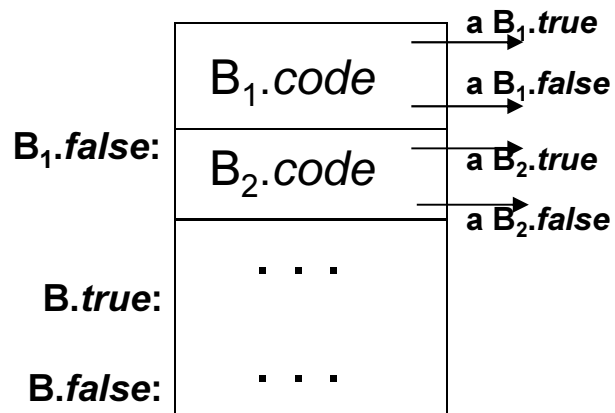
Osservazione: se B_1 è vero, anche $B_1 \parallel B_2$ è vero
se B_1 è falso, anche $B_1 \&\& B_2$ è falso

Gli operatori \parallel e $\&\&$ non compaiono nel codice, il loro valore è rappresentato da una posizione nella sequenza di istruzioni

Traduzione di espressioni booleane

Commentiamo in dettaglio le regole semantiche da associare ad una produzione, le regole associate alle altre produzioni possono essere commentate in modo analogo.

$B \rightarrow B_1 \parallel B_2$



Usando la logica del corto circuito, se B_1 è vero si può eseguire lo statement per B vero ($B_1.true = B.true$), mentre se B_1 è falso bisogna valutare B_2 . Viene allora introdotta una nuova etichetta che permette di specificare quali istruzioni a tre indirizzi eseguire per valutare B_2 . $B_2.true$ e $B_2.false$ hanno gli stessi valori degli analoghi attributi di B.

$B_1.true = B.true$; $B_1.false = newlabel()$

$B_2.true = B.true$; $B_2.false = B.false$

$B.code = B_1.code \parallel label(B_1.false) \parallel B_2.code$

Traduzione di espressioni booleane:schemi

$B \rightarrow \{B_1.true = B.true ; B_1.false = newlabel () \} B_1 \parallel$
 $\{B_2.true = B.true ; B_2.false = B.false \} B_2$
 $\{ B.code = B_1.code \parallel label(B_1.false) \parallel B_2.code \}$

$B \rightarrow \{B_1.true = newlabel () ; B_1.false = B.false \} B_1 \&\&$
 $\{B_2.true = B.true ; B_2.false = B.false \} B_2$
 $\{ B.code = B_1.code \parallel label(B_1.true) \parallel B_2.code B_2 \}$

$B \rightarrow ! \{B_1.true = B.false; B_1.false = B.true; \} B_1 \{B.code = B_1.code\}$

$B \rightarrow (\{B_1.true = B.true; B_1.false = B.false \} B_1) \{B.code = B_1.code\}$

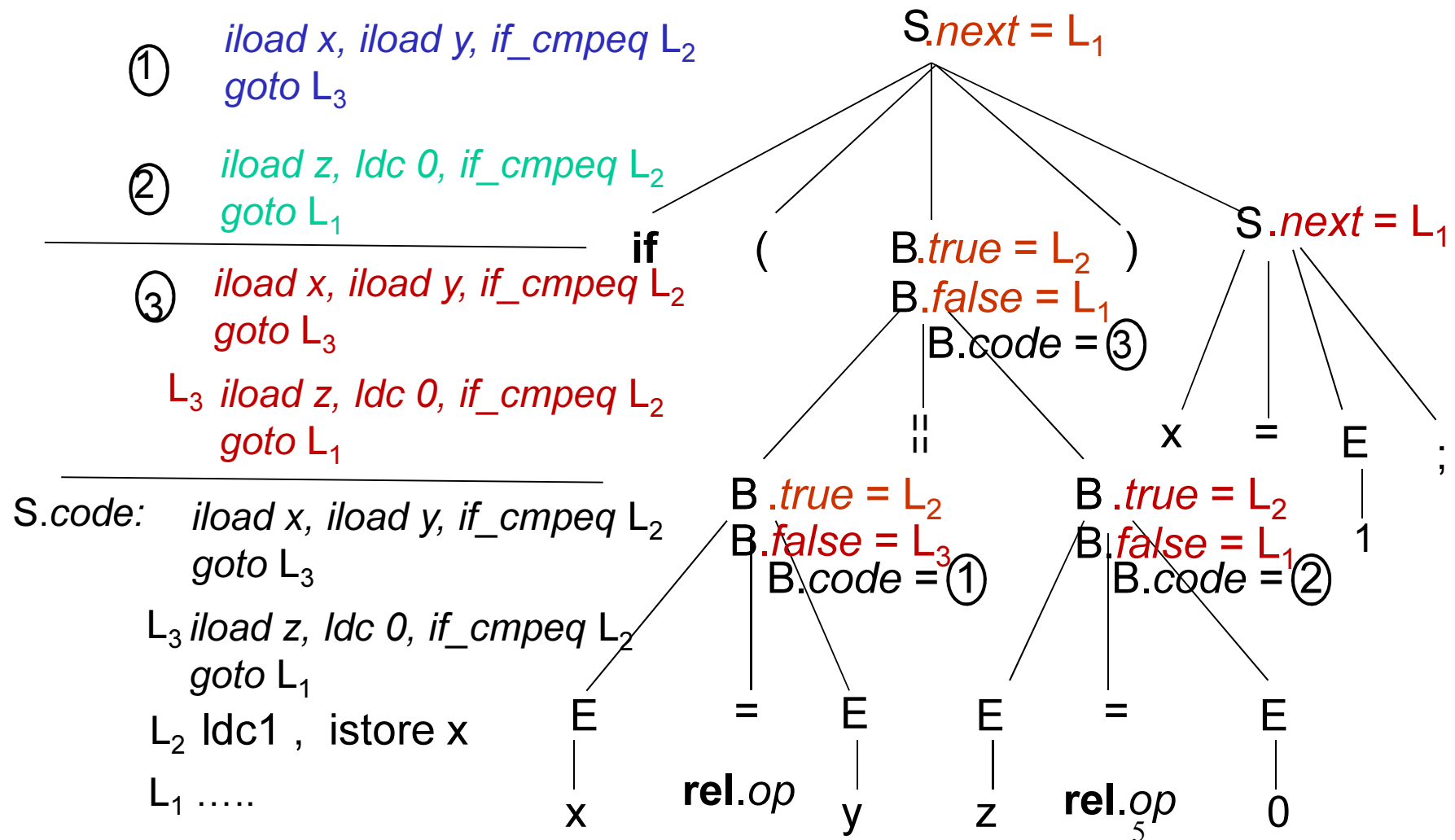
$B \rightarrow E_1 == E_2 \{ B.code = E_1.code \parallel E_2.code \parallel$
 $\text{'if_cmpeq' } B.true \parallel \text{'goto' } B.false \}$

$B \rightarrow \mathbf{true} \{B.code = \text{'goto' } B.true\}$

$B \rightarrow \mathbf{false} \{B.code = \text{'goto' } B.false\}$

Traduzione: esempi

Esempio: `if (x = y || z = 0) x = 1;`



Traduzione di espressioni booleane:schemi

$B \rightarrow \{B_1.true = B.true ; B_1.false = newlabel () \} B_1 \parallel$
 $\{B_2.true = B.true ; B_2.false = B.false \} B_2$
 $\{ B.code = B_1.code \parallel label(B_1.false) \parallel B_2.code \}$


$B \rightarrow \{B_1.true = newlabel () ; B_1.false = B.false \} B_1 \&\&$
 $\{B_2.true = B.true ; B_2.false = B.false \} B_2$
 $\{ B.code = B_1.code \parallel label(B_1.true) \parallel B_2.code B_2 \}$

schemi di traduzione al=volò

$B \rightarrow \{B_1.true = B.true ; B_1.false = newlabel () \} B_1 \parallel$
 $\{emitlabel(B_1.false), B_2.true = B.true ; B_2.false = B.false \}$
 B_2

$B \rightarrow \{B_1.true = newlabel () ; B_1.false = B.false \} B_1 \&\&$
 $\{emitlabel(B_1.true), B_2.true = B.true ; B_2.false = B.false \}$
 B_2

Traduzione: Evitare i goto ridondanti

if_cmpeq goto L₄
goto L₁
L₄: ...  *if_cmpne goto L₁*
L₄: ...


Nelle regole semantiche si usa una speciale etichetta “**fall**” per indicare che il controllo deve “fluire” in sequenza senza effettuare salti e nella traduzione delle espressioni booleane si usa, oltre al salto condizionato “*if_cmp***rel** L”, anche il salto condizionato “*if_cmp***notrel** L”

Traduzione: Evitare i goto ridondanti

Ad esempio le regole per la traduzione dello statement:

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

Vengono modificate nel modo seguente:

$B.true = \text{newlabel}()$		$B.true = fall$
$B.false = S_1.next = S.next$		$B.false = S_1.next = S.next$
$S.code = B.code \parallel \text{label}(B.true) \parallel$		$S.code = B.code \parallel S_1.code$
$\parallel S_1.code$		

Il valore *fall* di $B.true$ specifica che non ci sono salti da fare se B è vero. Diventa così inutile usare una label per etichettare la traduzione di S_1 .

Traduzione: Evitare i goto ridondanti

Vediamo come devono essere definite le nuove regole semantiche per alcune riscritture di B.

Regole semantiche per $B \rightarrow E_1 == E_2$

$B.true$ e $B.false$ sono attributi ereditati, noti quando viene usata la produzione. Se entrambi sono diversi da “*fall*” (vai in sequenza), la traduzione di B resta inalterata (traduzione di E_1 seguita da quella di E_2 seguita dai due salti “*if_cmpeq B.true*” e “*goto B.false*”. Se invece uno dei due è *fall*:

- se è $B.false = fall$, si deve effettuare un salto solo se B è vero e quindi si genera l'istruzione *if_cmpeq B.true*
- se è $B.true = fall$, si deve effettuare un salto solo se B è falso e quindi si genera l'istruzione *if_cmpne B.false*

$$B \rightarrow E_1 == E_2 \quad B.code = E_1.code \parallel E_2.code \parallel$$

if B.false = fall: ‘if_cmpeq B.true’
if B.true = fall: ‘if_cmpne B.false’
else: ‘if_cmpeq B.true’ || ‘goto’ B.fals

Traduzione: Evitare i goto ridondanti

Regole semantiche per $B \rightarrow B_1 \mid B_2$

$B_1.false$ è *fall* in quanto, se B_1 è falso si deve valutare B_2 , il cui codice nella traduzione di B viene collocato in sequenza con quello di B_1 . Più difficile è definire $B_1.true$, in quanto, se $B.true$ è *fall*, $B_1.true$ non può assumere tale valore, ma deve “saltare” la traduzione di B_2 . Viene allora creata una nuova label per etichettare le istruzioni da eseguire. $B_2.true$ e $B_2.false$ invece assumono i valori di $B.true$ e $B.false$ rispettivamente.

```
B1.true   =  if B.true ≠ fall then B.true else newlabel ()
B1.false  =  fall
B2.true   =  B.true
B2.false  =  B.false
B.code    =  if B.true = fall then B1.code || B2.code || label(B1.true)
              else B1.code || B2.code
```

Traduzione: Evitare i goto ridondanti

Esempio: `if (x == y || z == 0) x = 1;`

① `iload x, iload y, if_cmpeq L2`

② `iload z, ldc 0, if_cmpne L1`

③ `iload x, iload y, if_cmpeq L2,
iload z, ldc 0, if_cmpne L1, L2:`

P.code:

`iload x, iload y, if_cmpeq L2`

`iload z, ldc 0, if_cmpne L1`

L₂: `x = 1`

L₁

