Boolean Expressions

Lecture 25
Sections 8.4, 8.6

Robb T. Koether

Hampden-Sydney College

Fri, Apr 24, 2009
Outline

1. Relational Operators

2. Boolean Operators
   - The && Operator
   - The || Operator
   - The ! Operator

3. Assignment
Relational Operators

Definition (Equality and Relational Operators)

The **equality operators** are the operators `==` and `!=`.
The **relational operators** are the operators `<`, `>`, `<=`, and `>=`.

- We have already seen how to handle the operator `!=`.
- The other five operators are handled in a very similar way.
The relational-operator productions are

\[
\begin{align*}
  cexpr & \rightarrow \ expr \ == \ expr \\
  cexpr & \rightarrow \ expr \ != \ expr \\
  cexpr & \rightarrow \ expr \ < \ expr \\
  cexpr & \rightarrow \ expr \ > \ expr \\
  cexpr & \rightarrow \ expr \ <= \ expr \\
  cexpr & \rightarrow \ expr \ >= \ expr
\end{align*}
\]
Relational Operators

- The function `relOp()` builds a relational operator (comparison) tree.
- The particular operator is passed along with the two expressions.
- This is similar to the `arith()` function.
- After building the comparison tree, it must be attached to a `JUMP T` tree.
The comparison tree is represented as a TreeNode while the JUMPT tree is represented as a BackpatchNode.

The function `relOpToCExpr()` will perform the transition from TreeNode to BackpatchNode.

It is very similar to the `exprToCExpr()` function.
The Boolean operators are `&&`, `||`, and `!`.

Typically, these operators are applied to relational expressions.

For example, we might write

```
(i >= 0) && (i < size)
```

to test that an array index is within the proper range.
Boolean Operators

- Each of the relational expressions is represented as a BackpatchNode.
- The combined expression is also represented as a BackpatchNode.
- Therefore, the Boolean operators can be processed by using the functions
  - BackpatchNode()
  - makeList()
  - merge()
  - backpatch()
The Productions

The Boolean-operator productions are

\[
\begin{align*}
\text{cexpr} & \rightarrow \text{cexpr}_1 \& \& m \text{cexpr}_2 \\
\text{cexpr} & \rightarrow \text{cexpr}_1 \mid \mid m \text{cexpr}_2 \\
\text{cexpr} & \rightarrow \neg \text{cexpr}_1
\end{align*}
\]

Each \text{cexpr} represents a \texttt{BackpatchNode}.

By drawing a diagram as we did with if statements, we can how the branches should be connected.
Let us use the && operator as an example.
The || operator is very similar.
Draw a diagram based on the && production.
If cexpr₁ is false, then we need not (and must not) evaluate cexpr₂.
If cexpr₁ is true, then we must evaluate cexpr₂, to determine whether cexpr₁ && cexpr₂ is true.
Draw a diagram and fill in the arrows.

\[ cexpr_1 \text{ AND } m \text{ AND } cexpr_2 \]
The \&\& Operator

If $cexpr_1$ is false, then the expression is false.

$$cexpr_1 \text{ AND } m \text{ AND } cexpr_2$$
The \&\& Operator

- If $cexpr_1$ is true, then evaluate $cexpr_2$.
- Thus, evaluate $cexpr_2$ only if $cexpr_1$ is true.

$$cexpr_1 \text{ AND } m \Rightarrow cexpr_2$$
If \( cexpr_2 \) is true, then the expression is true.
Boolean Expressions

Robb T. Koether

Relational Operators

Boolean Operators

The && Operator

The || Operator

The ! Operator

Assignment

The && Operator

If $cexpr_2$ is false, then the expression is false.

```
cexpr_1     m     cexpr_2
  T  AND    T  T
  F  F      F  F
```
The && Operator

BackpatchNode andOp(BackpatchNode c1, Integer m1, BackpatchNode c2)
{
    backpatch(c1.trueList, m1);
    LinkedList list = merge(c1.falseList, c2.falseList);
    return new BackpatchNode(c2.trueList, list);
}
Thus, the action of the `andOp()` function is

- `backpatch(cexpr1.trueList, m)`.
- `merge(cexpr1.falseList, cexpr2.falseList)`.
- Create a new `BackpatchNode` with
  - `trueList = cexpr2.trueList`.
  - `falseList = the merged list`.
The production for OR is

\[ cexpr \quad \rightarrow \quad cexpr_1 \mid \mid m \ cexpr_2 \]

Show how the backpatching should be done.
The \( \not \) Operator

- The production for **\textbf{NOT}** is

\[
\text{cexpr} \rightarrow \not \text{cexpr}_1
\]

- Show how the backpatching should be done.
Assignment

Homework

- Read Section 8.4, pages 488 - 491.
- Read Section 8.6, pages 501 - 503.