Decision Structures - The Syntax Tree

Lecture 22
Sections 8.4, 8.6

Robb T. Koether

Hampden-Sydney College

Fri, Apr 10, 2009
Outline

1. if Statements
2. The BackpatchNode Class
3. Backpatching
4. Backpatch Functions
5. Jumps in the Grammar
6. Assignment
We will consider two forms of the `if` statement.

\[
\begin{align*}
stmt &\rightarrow \textbf{if} \ (\ cexpr \ ) \ stmt \ ; \\
stmt &\rightarrow \textbf{if} \ (\ cexpr \ ) \ stmt \ \textbf{else} \ stmt \ ;
\end{align*}
\]

where `cexpr` is a conditional expression.

Recall that we have the productions

\[
\begin{align*}
stmt &\rightarrow \{ \ stmts \ } \\
cexpr &\rightarrow expr
\end{align*}
\]
For the time being, we will assume that $cexpr$ is a numerical expression.

Zero is interpreted as false.

Any nonzero value is interpreted as true.

Thus, we do not need to be concerned with relational operators ($==$, $!=$, $<$, $>$, $<=$, $>=$) or boolean operators ($&&$, $||$, $!$) yet.
Consider the flow of control for the production

\[ stmt \to \text{if ( cexpr ) stmt}_1; \]

- If \( cexpr \) is non-zero, then execution must jump to \( stmt_1 \).
- If \( cexpr \) is zero, then execution must jump to whatever follows \( stmt_1 \).
- Problem: We do not yet know where that it.
We need an object associated with \textit{cexpr} that will contain two destinations.

- The “true” destination.
- The “false” destination.

Such an object is called a \textit{backpatch node}.
The **BackpatchNode** Class

```java
public class BackpatchNode {
    LinkedList trueList;
    LinkedList falseList;
}
```
Backpatch Nodes

- Each backpatch node contains
  - A “true” reference to a linked list of labels.
  - A “false” reference to a linked list of labels.
- The nonterminal $cexpr$ will represent a backpatch node.
- From $cexpr$ we set up a “true” destination label and a “false” destination label.
The “true” and “false” destination labels are labels to be resolved once we know where the destinations are.

An unresolved label is called a backpatch label.

When we figure out the destination, we will “backpatch” the backpatch label to an actual label.
Backpatching

An Equate Statement

B6=L8

An assembly language equate statement will accomplish this.
The reason we keep a *list* of such labels is that there may be several branches in the program that all have the same destination.
Example

if (a)
    b = 5;
else
    b = 10;
c = 2;
Example

if (a)
    b = 5;
else
    b = 10;
c = 2;

Branch on true
Example

```java
if (a)
    b = 5;
else
    b = 10;
c = 2;
```

Branch on false
Example

```c
if (a)
    b = 5;
else
    b = 10;
c = 2;
```

Unconditional branch
if (a)
    b = 5;
else
    b = 10;
c = 2;

Unconditional branch
Example

- Notice that the unconditional branch from the true part and the unconditional branch from the false part have the same destination.
- Thus, we build a list of two labels that will be resolved to the same destination.
Example

```
if (a)
    b = 5;
else
    b = 10;
c = 2;
```
Example

```plaintext
if (a)
    b = 5;
else
    b = 10;
c = 2;
```

Jump on true to Label B1
Example

if (a)
    b = 5;
else
    b = 10;
c = 2;

Jump on true to Label B1
Jump on false to Label B2
Example

if (a)
    b = 5;
else
    b = 10;
c = 2;

Jump on true to Label B1
Jump on false to Label B2

Label L3
if Statements

The Back-patchNode Class
Backpatching
Backpatch Functions
Jumps in the Grammar
Assignment

Example

Jump on true to Label B1
Jump on false to Label B2

Label L3

if (a)
  b = 5;
else
  b = 10;
c = 2;
T F

Jump on true to Label B1
Jump on false to Label B2

Label L3

if (a)
  b = 5;
else
  b = 10;
c = 2;
T F

.: B1 = L3
Example

```plaintext
if (a)
    b = 5;
else
    b = 10;
c = 2;
F Jump on false to Label B2
```
Example

```java
if (a)
    b = 5;
else
    b = 10;
c = 2;
```

Jump on false to Label B2

Jump to Label B4
Example

if (a)
    b = 5;
else
    b = 10;
c = 2;
F Jump on false
to Label B2
L3: Jump to
Label B4
L5: Label L5

f

Jump on false
to Label B2
L3: Jump to
Label B4

Label L5
Example

if (a)
    b = 5;
else
    b = 10;
c = 2;

F Jump on false to Label B2
L3: Jump to Label B4

Label L5
L5: b = 10;
c = 2;

\[ \therefore B2 = L5 \]
Example

```
if (a)
    b = 5;
else
    b = 10;
c = 2;
L3: Jump to Label B4
L5: 
```
Example

```java
if (a)
    b = 5;
else
    b = 10;
c = 2;
```

Jump to Label B4
Jump to Label B6
Example

```
if (a)
    b = 5;
else
    b = 10;
c = 2;
L3: Jump to Label B4
L5: Jump to Label B6
∴ Merge B4 and B6
```
if (a)
    b = 5;
else
    b = 10;
c = 2;
L3:
Jump to
Label \{B4, B6\}
Example

```
if (a)
    b = 5;
else
    b = 10;
c = 2;
L3:
Jump to
Label {B4, B6}
```

```
L7: Label L7
L5: b = 10;
```

Jump to Label {B4, B6}
Example

```plaintext
if (a)
    b = 5;
else
    b = 10;
c = 2;
L3:
Jump to
Label \{B4, B6\}
L5:
L7:
Label L7

\[ B4 = B6 = L7 \]
```
Example

```
if (a)
    b = 5;
else
    b = 10;

c = 2;
```

L3: b = 5;
L5: b = 10;
L7: c = 2;
Example

```
if (a)
    b = 5;
else
    b = 10;
c = 2;
```

T: L7: T
F: L5: F
L3: b = 5;
else
L5: b = 10;
L7: c = 2;
Example

- Within the `if` statement, there are two “known” destinations and one unknown destination.
- We can see that labels B₄ and B₆ will be resolved to the same destination eventually, but within the `if` statement that destination is unknown.
- Therefore, in the meantime, we merge them together in a list of backpatch nodes to be resolved later to the same destination.
- Once we learn the destination, all labels in the list are resolved to that location.
Backpatching Functions

- **makeList(label);**
  Creates a LinkedList containing the single Integer label. Returns a reference to the list.

- **merge(list1, list2);**
  Merges the elements of list1 and list2. Returns a reference to the merged list.

- **backpatch(list, label);**
  Equates label as the target label for each backpatch label in list.
We saw in the example that actual labels are needed within the `if` statement to serve as destinations.

We will incorporate the generation of these labels into the grammar.

The nonterminal $m$ will create a label node which will serve as a known destination.

The nonterminal $n$ will create a jump to an as-yet unknown destination.
Labels and Jumps in the Grammar

- $m$ represents a destination.
- $n$ represents an unconditional branch.
- The productions that involve these nonterminals are

\[
\begin{align*}
stmts & \rightarrow \ stmts \ m \ stmt \\
stmt & \rightarrow \ if \ (\ cexpr) \ m \ stmt \\
stmt & \rightarrow \ if \ (\ cexpr) \ m \ stmt \ n \ else \ m \ stmt \\
func & \rightarrow \ fbeg \ stmts \ m \}
\end{align*}
\]
Label Trees

A Label Tree

A Printed Label Tree

An Assembly Code Label

LABEL 6

LABEL label=6

L6:
Jump Trees

A Jump Tree

JUMP

BLABEL 6

JUMP INT
BLABEL blabel=6

An Assembly Code Jump
jmp B6

A Printed Jump Tree
Equate Trees

An Equate Tree

JUMP

BLABEL 6

LABEL 8

EQU

BLABEL blabel=6

LABEL label=8

B6=L8

A Printed Equate Tree

An Assembly Code Equate
**Assignment**

**Homework**

- Read Section 8.4, pages 491 - 493.
- Read Section 8.6, pages 504 - 506.