Phases of a compiler

Intermediate Representation (IR): specification and generation

Figure 1.6, page 5 of text
backpatching if
### 6.7.3 Backpatching Flow-of-Control statements

<table>
<thead>
<tr>
<th>S $\rightarrow$ if (B) M1 S1</th>
</tr>
</thead>
<tbody>
<tr>
<td>N else M2 S2</td>
</tr>
</tbody>
</table>

backpatch(B.truelist, M1.instr)
backpatch(B.falselist, M2.instr)
temp = merge(S1.nextlist, N.nextlist)
S.nextlist = merge(temp, S2.nextlist)

<table>
<thead>
<tr>
<th>M $\rightarrow$ ε</th>
</tr>
</thead>
</table>

M.instr = nextInstr

<table>
<thead>
<tr>
<th>N $\rightarrow$ ε</th>
</tr>
</thead>
</table>

N.nextlist = makelist(nextInstr)
gen('goto _')

---

**Diagram:***

- **true** path from B to M1.instr to S1 to N
- **false** path from B to M2.instr to S2
- **S.next**
Let's extend the Boolean expression example from part 1 by embedding that expression into an if-then-else statement (using the textbook syntax, not alpha syntax).
Example 6.24 - extended

if \((x < 100 \; || \; x > 200 \; && \; x \neq y)\) S1 else S2

100: if \(x < 100\) goto ___
101: goto 102
102: if \(x > 200\) goto 104
103: goto ___
104: if \(x \neq y\) goto ___
105: goto ___

truelist = \{100,104\}
falselist = \{103,105\}

Let's remember where we left off...
Example 6.24 - extended

if \( x < 100 \) \| \( x > 200 \) \&\& \( x \neq y \) S1 else S2

\begin{align*}
100: & \quad \text{if } x < 100 \text{ goto } \_ \\
101: & \quad \text{goto } 102 \\
102: & \quad \text{if } x > 200 \text{ goto } 104 \\
103: & \quad \text{goto } \_ \\
104: & \quad \text{if } x \neq y \text{ goto } \_ \\
105: & \quad \text{goto } \_ \\
106: & \quad \text{instruction for S1} \\
107: & \quad \text{instruction for S1} \\
108: & \quad \text{instruction for S1} \\
109: & \quad \text{instruction for S1} \\
110: & \quad \text{instruction for S1} \\
111: & \quad \text{goto } \_ \\
112: & \quad \text{instruction for S2} \\
113: & \quad \text{instruction for S2} \\
114: & \quad \text{instruction for S2}
\end{align*}

\text{truelist} = \{100,104\}
\text{falselist} = \{103,105\}

In the example above we have not spelled out what S1 and S2 are.

Let's assume S1 requires 5 instructions and S2 requires 3 instructions.
Example 6.24 - extended
if (x < 100 || x > 200 && x != y) S1 else S2

```
if (x < 100 || x > 200 && x != y) S1 else S2
```

```
100: if x < 100 goto 106
101: goto 102
102: if x > 200 goto 104
103: goto 112
104: if x != y goto 106
105: goto 112
106: instruction for S1
107: instruction for S1
108: instruction for S1
109: instruction for S1
110: instruction for S1
111: goto ___
112: instruction for S2
113: instruction for S2
114: instruction for S2
115:
```

```
truelist = {100,104}
falselist = {103,105}
nextlist = {111}
```

Embedded in the context of this if-then-else statement we can backpatch truelist and falselist from the Boolean expression, and we introduce nextlist.
backpatching
while
### 6.7.3 Backpatching Flow-of-Control statements

The end-of-rule actions for a while statement are shown on the next slide.

**Exercise:**
Extend example 6.24 as a while statement where the body of the while requires 5 instructions.

\[
\text{while } (x < 100 \text{ || } x > 200 \text{ && } x \neq y) \text{ S1}
\]

Show how backpatching works in the instruction array.
### 6.7.3 Backpatching Flow-of-Control statements

<table>
<thead>
<tr>
<th>S → while M1</th>
<th>backpatch(S1.nextlist, M1.instr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(B) M2 S1</td>
<td>backpatch(B.truelist, M2.instr)</td>
</tr>
<tr>
<td></td>
<td>S.nextlist = B.falselist</td>
</tr>
<tr>
<td></td>
<td>gen('goto' M1.instr)</td>
</tr>
<tr>
<td>M → ε</td>
<td>M.instr = nextinstr</td>
</tr>
</tbody>
</table>

**Diagram:**

- M1.instr → B
- B → true M2.instr → S1
- B → false S1 → M.next

© 2020 Carl Alphonce - Reproduction of this material is prohibited without the author's consent
Here's what I came up with...
Example 6.24 - extended

while \((x < 100 \; \text{||} \; x > 200 \; \&\& \; x! = y)\) S1

100: if \(x < 100\) goto 106
101: goto 102
102: if \(x > 200\) goto 104
103: goto ___
104: if \(x ! = y\) goto 106
105: goto ___
106: instruction for S1
107: instruction for S1
108: instruction for S1
109: instruction for S1
110: instruction for S1
111: goto 100

B.truelist = \{100,104\}
S.nextlist = B.falselist = \{103,105\}

Notice that we backpatch only those instructions whose targets are within the (while) instruction's code block.
backpatching for
Exercise: show how to translate a generic for statement

```
for ( S1 ; B ; S2 ) S3
```

and give the translation of this one in particular:

```
for ( S1 ; x < 100 || x > 200 && x != y ; S2 ) S3
```