CSE443
Compilers

Dr. Carl Alphonse
alphonse@buffalo.edu
343 Davis Hall
Phases of a compiler

Figure 1.6, page 5 of text
Example 5.19 (p. 335)

\[ S \rightarrow \text{while} (C) S_1 \]

What are the semantics of this?
Example 5.19 (p. 335)

$ S \rightarrow \text{while ( C ) } S_1$

What are the semantics of this?
Figure 5.28 (p. 336)

SDT for while statement

\[ S \rightarrow \text{while (} \{ \text{L1 = new(); L2 = new();} \]

\[ \text{C.false = S.next; C.true = L2; } \]

\[ \text{S1.next = L1; } \]

\[ \text{S1.code = label || L1 || C.code || label || L2 || S1.code } \]
Example 5.26 [p. 349]

S → while ( { L1=new(); L2=new(); C.false=S.next; C.true=L2; } 
  C ) { S₁.next=L1; }
S₁ { S.code=label || L1 || C.code || label || L2 || S₁.code }
Example 5.26 [p. 349]

S → while ( 
   M C ) 
N S₁ { S.code=label || L₁ || C.code || label || L₂ || S₁.code } 
M → ε { L₁ = new(); L₂ = new(); C.false = S.next; C.true = L₂; } 
N → ε { S₁.next = L₁; } 

* ? will become S on reduction*
Example 5.26 [p. 349]

S → while ( M C )
N S₁ { S.code=label || L1 || C.code || label || L2 || S₁.code }
M → ε { L1=new(); L2=new(); C.false=S.next; C.true=L2; }
N → ε { S₁.next=L1; }

? will become S on reduction

<table>
<thead>
<tr>
<th>?</th>
<th>while</th>
<th>(</th>
<th>M</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>S.next</td>
<td></td>
<td></td>
<td>C.true</td>
<td>C.code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C.false</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L2</td>
</tr>
</tbody>
</table>
Example 5.26 [p. 349]

S → while ( M C )
N S
M → ε
N → ε

{ S.code=label || L1 || C.code || label || L2 || S_1.code }
Example 5.26 [p. 349]

S → while ( M C )
N S₁ { S.code=label || L₁ || C.code || label || L₂ || S₁.code }
M → ε { L₁=new(); L₂=new(); C.false=S.next; C.true=L₂; }
N → ε { S₁.next=L₁; }

? will become S on reduction

```

?       while       (       M       C       )
S.next              C.true       C.code
C.false
L₁
L₂

S₁.next       S₁.code
S₁.next=stack[top-3].L₁
```
Roadmap

We will revisit how the semantics of flow-of-control statements can be expressed in section 6.6.3 Flow-of-Control Statements.

At that point we will learn the backpatching approach, which you will implement in your compiler.