Exercise: Semantic Analysis

• Consider the SDD in the next slide, where
  – \textit{newlabel()} generates a fresh symbolic label
  – \textit{newtemp()} generates a fresh variable name
  – \textit{gen()} generates strings
  – \textit{||} is the string concatenation operator
  – \textit{code} is the attribute with three-address code

Exercise: Semantic Analysis

• (continued)
  – \textit{id.place} is the name of the variable associated to the token \textit{id}
  – \textit{E.place} is the temporary variable storing the computed value at \textit{E}
  – \textit{num.val} is the value associated to the token \textit{num}
  – \textit{relop.op} is a comparison operator ($<$, $\leq$, $=$, …)
### Exercise: Semantic Analysis

#### Productions

<table>
<thead>
<tr>
<th>Production</th>
<th>Semantic rules</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Prog → S</code></td>
<td><code>S.next = newlabel();</code>&lt;br&gt;<code>Prog.code = S.code | gen(S.next ' ')</code></td>
</tr>
<tr>
<td><code>S → S_1 ; S_2</code></td>
<td><code>S_1.next = newlabel(); S_2.next = S.next;</code>&lt;br&gt;<code>S.code = S_1.code | gen(S_1.next ' ')</code>&lt;br&gt;| <code>S_2.code</code></td>
</tr>
<tr>
<td><code>S → while Test do { S_1 }</code></td>
<td><code>Test.begin = newlabel();</code>&lt;br&gt;<code>Test.true = newlabel(); Test.false = S.next;</code>&lt;br&gt;<code>S.next = Test.begin;</code>&lt;br&gt;<code>S.code = gen(Test.begin ' ') |</code>&lt;br&gt;<code>Test.code | gen(Test.true ' ') |</code>&lt;br&gt;<code>S_1.code | gen('goto' Test.begin)</code></td>
</tr>
</tbody>
</table>

#### Semantic rules

<table>
<thead>
<tr>
<th>Production</th>
<th>Semantic rules</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>S → id = E</code></td>
<td><code>S.code = E.code</code>&lt;br&gt;<code>| gen(id.place '=' E.place)</code></td>
</tr>
<tr>
<td><code>Test → id_1 relop id_2</code></td>
<td><code>Test.code = gen('if' id_1.place relop.op id_2.place 'goto' Test.true)</code>&lt;br&gt;<code>| gen('goto' Test.false)</code></td>
</tr>
<tr>
<td><code>E → E_1 + id</code></td>
<td><code>E.place = newtemp();</code>&lt;br&gt;<code>E.code = E_1.code | gen(E.place '=')</code>&lt;br&gt;<code>E_1.place '+' id.place)</code></td>
</tr>
<tr>
<td><code>E → E_1 - id</code></td>
<td><code>E.place = newtemp();</code>&lt;br&gt;<code>E.code = E_1.code | gen(E.place '=')</code>&lt;br&gt;<code>E_1.place '-' id.place)</code></td>
</tr>
</tbody>
</table>
## Exercise: Semantic Analysis

### Productions

<table>
<thead>
<tr>
<th>Production</th>
<th>Semantic rules</th>
</tr>
</thead>
</table>
| $E \rightarrow E_1 + \text{num}$ | $E.place = \text{newtemp}();$
| | $E.code = E_1.code \| \text{gen}(E.place \ '=>' E_1.place \ '+ ' \text{num}.val)$ |
| $E \rightarrow E_1 - \text{num}$ | $E.place = \text{newtemp}();$
| | $E.code = E_1.code \| \text{gen}(E.place \ '=>' E_1.place \ '- ' \text{num}.val)$ |
| $E \rightarrow \text{id}$ | $E.Place = \text{id}.place; \ E.code = \"”$

### Exercise: Semantic Analysis

- Consider the input

```plaintext
while x < y do {
    y = y - 1;  x = x + 1 }
```
Exercise: Semantic Analysis

• Provide the annotated parse tree (without the code attribute) for the input together with the values of the attributes

```
while x < y do { y = y - 1; x = x + 1 }
```
Exercise: Semantic Analysis

- Provide the value of the place and the code attributes at each node of the tree (this is the 3AC translation of the given input produced by the semantic rules)
Exercise: Semantic Analysis

```
while x < y do { y = y - 1; x = x + 1 }
```

Exercise: Semantic Analysis

```
code =
t1 = y - 1
y = t1
```

```
place = y
```

```
place = t1
code = t1 = y - 1
```

```
place = y
code = y
```

```
place = y
```

```
val = 1
place = y
```

```
E - num
```

```
id = E
```

```
id = E
```

```
while Test do { S }
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```

```
S
```
Exercise: Semantic Analysis

while x < y do { y = y - 1; x = x + 1 }

Exercise: Semantic Analysis

code =
  t2 = x + 1
  x = t2

place = x

place = t2
  code = t2 = x + 1
Exercise: Semantic Analysis

while x < y do { y = y - 1; x = x + 1 }

Exercise: Semantic Analysis

code =
  t1 = y - 1
  y = t1
L4 :
  t2 = x + 1
  x = t2
Exercise: Semantic Analysis

while Test do { S }

while x < y do { y = y - 1; x = x + 1 }

code =
if x < y goto L3
goto L1

L2 :
if x < y goto L3
goto L1
L3 :
t1 = y - 1
y = t1
L4 :
t2 = x + 1
x = t2
goto L2
L1 :

code =
t1 = y - 1
y = t1
L4 :
t2 = x + 1
x = t2
goto L2

place = x
place = y

op = <