Translation

```c
int i, int j; float x00; x00 = float x01;
while (true) {
    if (i + j == 1) {
        if (j + i < 1) {
            if (i == j) break;
        }
    }
}
```

Figure 2.1: A code fragment to be translated

Figure 2.2: Simplified intermediate code for the program fragment in Fig. 2.1

What is syntax-directed translation?

![Diagram of compiler front end]

Figure 2.3: A model of a compiler front end
Syntax vs. Semantics

Syntax

• Regular grammars
• Context-free grammars
• BNF notation
• Example:
  
  stmt -> if ( expr ) stmt else stmt

What is a CFG?

• Set of terminals (usually bold)
• Set of nonterminals (italic and/or capitalized)
• Set of productions (contains ->)
• Start symbol
Example CFG

\[ \text{list} \rightarrow \text{list} + \text{digit} \]
\[ \text{list} \rightarrow \text{list} - \text{digit} \]
\[ \text{list} \rightarrow \text{digit} \]
\[ \text{digit} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \]

- What can we get using this grammar?
- Can we get the empty string, i.e. \(\varepsilon\)?

Derivation

- What is a language?
- Deriving strings:
  - Start symbol
  - Replace nonterminals

- What if a string can’t be derived?

Parsing

Figure 2.5: Parse tree for 9-3+2 according to the grammar in Example 2.1

\[ \text{list} \rightarrow \text{list} + \text{digit} \mid \text{list} - \text{digit} \mid \text{digit} \]
\[ \text{digit} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \]
## Ambiguity

- Suppose we used:

\[ \text{string} \rightarrow \text{string} + \text{string} \mid \text{string} - \text{string} \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \]

![Figure 2.6: Two parse trees for 9-8*2](image)

## Class Example

- What language is generated by these?

\[ S \rightarrow 0S1 \mid 01 \]
\[ S \rightarrow S(S)S \mid \varepsilon \]
\[ S \rightarrow a \mid SS \mid S^* \]

- Which are ambiguous?

## Associativity

\[ \text{list} \rightarrow \text{list} + \text{digit} \mid \text{list} - \text{digit} \mid \text{digit} \]
\[ \text{digit} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \]

Vs.

\[ \text{right} \rightarrow \text{letter} = \text{right} \mid \text{letter} \]
\[ \text{letter} \rightarrow a \mid b \mid \ldots \mid z \]

- What do you notice about these grammars?
Associativity vs. Precedence

• What happens with more than one op?
  – How are * and + alike and different?

• Need to resolve ambiguity
  left associative: + -
  left associative: * /

\[
\begin{align*}
  expr & \rightarrow expr + term \mid expr - term \mid term \\
  term & \rightarrow term * factor \mid term / factor \mid factor \\
  factor & \rightarrow digit \mid ( expr )
\end{align*}
\]
Synthesized Attributes

```
expr, t = 95 - 2
```

- `expr, t = 95 - 2
  - `expr, t = 95`
  - `term, t = 2`
- `term, t = 9
  - `term, t = 6`
  - `2`

Figure 2.9: Attribute values at nodes in a parse tree

Syntax-Directed Definition

<table>
<thead>
<tr>
<th>Production</th>
<th>Semantic Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>expr -&gt; expr * term</code></td>
<td>`expr, t = expr, t</td>
</tr>
<tr>
<td><code>expr -&gt; expr - term</code></td>
<td>`expr, t = expr, t</td>
</tr>
<tr>
<td><code>expr -&gt; term</code></td>
<td><code>expr, t = term, t</code></td>
</tr>
<tr>
<td><code>term -&gt; 0</code></td>
<td><code>term, t = '0'</code></td>
</tr>
<tr>
<td><code>term -&gt; 1</code></td>
<td><code>term, t = '1'</code></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td><code>term -&gt; 9</code></td>
<td><code>term, t = '9'</code></td>
</tr>
</tbody>
</table>

Figure 2.10: Syntax-directed definition for infix to postfix translation

Syntax-Directed Translation Scheme

```
rest -> + term (print('+')) rest
```

```
  +
  |
  term

  (print(' + '))

  rest
```

Figure 2.13: An extra leaf is constructed for a semantic action

- How does this differ from synthesized attributes?
Reading/Assignments

• Continue Milestone 1
• Finish Chap. 2