What is syntax-directed translation?

Translation

```c
int i; int j; float k; float v; float x;
while (true) {
    i = i + 1;
    j = j + 1;
    if (i > j) break;
    x = x[j]; v[x] = v[j]; i = x;
}
```

Figure 22: A code fragment to be translated

Figure 23: Simplified intermediate code for the program fragment in Fig. 22

Figure 24: 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

Figure 2.4: Intermediate code for "do i=i+1; while(a[i]<v)"
Example CFG

```
list -> list + digit
list -> list - digit
list -> digit
digit -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

- What can we get using this grammar?
- Can we get the empty string, i.e. $\epsilon$?

Derivation

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

- What if a string can’t be derived?

Parsing

```
list -> list + digit | list - digit | digit
digit -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Figure 2.5: Parse tree for $9-5+2$ according to the grammar in Example 2.1
Ambiguity

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

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

Class Example

• What language is generated by these?
  \[
  S \rightarrow 0 \ S \mid 0 \ 1 \\
  S \rightarrow S \ (S) \ S \mid \epsilon \\
  S \rightarrow a \mid SS \mid S^* \\
  \]

• Which are ambiguous?

Associativity

\[
\text{list} \rightarrow \text{list} + \text{digit} \mid \text{list} - \text{digit} \mid \text{digit} \\
\text{digit} \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
\text{Vs.} \\
\text{right} \rightarrow \text{letter} = \text{right} \mid \text{letter} \\
\text{letter} \rightarrow a \mid b \mid \ldots \mid z \\
\]

• What do you notice about these grammars?
Associativity

- What happens with more than one op?
  - How are * and + alike and different?
- Need to resolve ambiguity
  - left associative: + -
  - left associative: * /

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

Syntax-Directed Translation

- Extend grammar
  - Attributes
  - Translation scheme
Synthesized Attributes

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>expr → expr₁ + term</td>
<td>expr = expr₁</td>
</tr>
<tr>
<td>expr → expr₁ - term</td>
<td>expr = expr₁</td>
</tr>
<tr>
<td>expr → term</td>
<td>expr = term</td>
</tr>
<tr>
<td>term → 0</td>
<td>term = '0'</td>
</tr>
<tr>
<td>term → 1</td>
<td>term = '1'</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>term → 9</td>
<td>term = '9'</td>
</tr>
</tbody>
</table>

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

Syntax-Directed Translation Scheme

rest → + term {print('+'}) rest₁

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

- How does this differ from synthesized attributes?
Figure 2.14: Actions translating 9+5*2 into 95-2+*.

- `expr` → `expr` + `term` {print('+'')}
- `expr` → `expr` - `term` {print('-')}
- `term` → 0 {print('0')}
- `term` → 1 {print('1')}
- `term` → 9 {print('9')}

Figure 2.15: Actions for translating into prefix notation.