5.2 Attribute Grammars

**Attribute Grammar:**
- An attribute grammar is an extension of a context-free grammar, with two extensions:

1. **Attributes on Symbols:** Each grammar symbol S, terminal or nonterminal, is specified to have various attributes. Each attribute is specified both with a name (e.g., ‘type’, ‘value’, and also a type, restricting the range of fillers the attribute can take.

2. **Attribute evaluation rules:** Each production rule in the grammar can have associated with it a number of ‘semantic rules’, each of which specifies how an attribute of one symbol can be calculated from the attributes of other symbols in the production.
Semantic Analysis

Attribute Grammars

Attribute Evaluation Rules:
• Sometimes called “semantic rules”
• A mechanism for propagating attributes between nodes.
• Each CF Grammar rule can have associated one or more semantic rules which state how attributes are to be propagated between the symbols.

<table>
<thead>
<tr>
<th>CFG Rule</th>
<th>Semantic Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode :- int</td>
<td>Mode.dtyp := integer</td>
</tr>
</tbody>
</table>
| Decl :- Mode IDList | Decl.type := Mode.dtyp;
|                | IDList.type := Decl.type               |
| IDList :- id   | id.type = IDList.type                  |

Semantic analysis

Attribute Grammars

Indexing of Symbols:
• In normal CFG, a symbol can occur multiple times in the rule:
  IDList :- id, IDList

• In Attribute Grammars (AGs), where the same symbol appears more than once in the production (on either side), we need to refer uniquely to each appearance of the symbol.

• We will add subscripts to those on the RHS:
  IDList :- id , IDList₁

• This allows the semantic rules to refer distinctly to each of the symbols which have the same name:
  IDList₁.type := IDList.type
Semantic analysis

From Grammar to Tree:

• Semantic Rules are associated with each production of the CFG.

• Each node of the parse tree corresponds to a production rule:

```
  IDList :- id, IDList

did , IDList
```

• So, as we build the parse tree, we also attach the semantic rules of the production with the node of the tree.

Example 1 revisited:

• Lets now look at the semantic rules behind example 1

```
E → E + E
E → E - E
E → - E
E → (E)
E → c
```
Semantic analysis

Example 1 revisited:
• First, we add in indexes for repeated symbols

\[
\begin{align*}
E \rightarrow & E_1 + E_2 \\
E \rightarrow & E_1 - E_2 \\
E \rightarrow & -E_1 \\
E \rightarrow & (E_1) \\
E \rightarrow & i \\
E \rightarrow & c
\end{align*}
\]

• Then, we add in attribute propagation rules

\[
\begin{align*}
E \rightarrow & E_1 + E_2 \{ \text{E.val} := E_1.val + E_2.val \} \\
E \rightarrow & E_1 - E_2 \{ \text{E.val} := E_1.val - E_2.val \} \\
E \rightarrow & -E_1 \{ \text{E.val} := -E_1.val \} \\
E \rightarrow & (E_1) \{ \text{E.val} := E_1.val \} \\
E \rightarrow & i \{ \text{E.val} := \text{symbTable}(i.name, \text{type}) \} \\
E \rightarrow & c \{ \text{E.val} := c.val \}
\end{align*}
\]
Example 1 revisited

**c.val** (given by lexical analysis)

\[(3 + 4) * 5\]

**E.val := c.val** (from rule for E:-c)
Attributes: informal description

Example 1 revisited

\[ (3 + 4) \times 5 \]

\[ E \text{.val} = E_1 \text{.val} + E_2 \text{.val} \]

\[ E \text{.val} = E_1 \text{.val} \]
**Semantic analysis**

**Attribute Grammars**

**Definition:**

- **An annotated parse tree** is the parse tree with the attributes at each node shown.

```
E: v:7  t:i
  v:3  t:i
  v:4  t:i
  v:3  t:i
  E: v:7  t:i
  E: v:4  t:i
  C: v:4  t:i
  ( 3  +  4  )  *  5
```

**Attribute Grammars & side effects:**

- **Attribute evaluation rules** have an abstract form like:
  
  ```
  X.attrib := f(Y.attrib, Z.attrib)
  ```
  
  ...where \( f() \) is a function over attributes.

- In a strict attribute grammar, the functions on the RHS of attribute evaluation rules should not have 'side effects', i.e.,
  
  - they should not change the structure of the parse tree,
  - nor the attributes of symbols (attributes should only be changed by assignment).

- Later, we will look at extensions of attribute grammars which allow such side effects.
Semantic analysis

Attribute Grammars

Synthesized and Inherited Attribute Evaluation Rules:

• A evaluation rule for a synthesized attribute will always have the LHS of the production on the LHS of the evaluation rule, e.g.,

\[ \text{Expr} \rightarrow \text{Expr}_1 \cdot \text{Expr}_2 \]
{ \text{Expr}.val = \text{Expr}_1.val + \text{Expr}_2.val \}

• A evaluation rule for a inherited attribute will always a symbol from the RHS of the production on the LHS of the evaluation rule, e.g.,

\[ L \rightarrow \text{L}_1, \text{id} \]
{ \text{L}_1.type = L.type \}

Formal Statement of Grammars:

A CFG is a quadruple \( G = ( \Sigma_T, \Sigma_N, \text{rules}, \text{axiom} ) \)

\( \{ +, *, (,), c, i \}, \)
{ \( E \rightarrow E + E, \)
\( E \rightarrow E - E, \)
\( E \rightarrow i, \)
\( E \rightarrow c \)
\}
\}
Semantic analysis

Attribute Grammars

Formal Statement for an Attribute Grammar
(Note that it details the attributes for each terminal and nonterminal)

\[ G_e = \{ (+, *, (,), c{\text{value, type}}, i{\text{value, type}}), \{ E{\text{value, type}} \}, \]
\[
\begin{align*}
E &\rightarrow E_i + E_d \quad \{ E.{\text{value}} := E_i.{\text{value}} + E_d.{\text{value}}, E.{\text{type}} := E_i.{\text{type}} \}, \\
E &\rightarrow E_i - E_d \quad \{ E.{\text{value}} := E_i.{\text{value}} - E_d.{\text{value}}, E.{\text{type}} := E_i.{\text{type}} \}, \\
E &\rightarrow -E_d \quad \{ E.{\text{value}} := -E_d.{\text{value}}, E.{\text{type}} := E_d.{\text{type}} \}, \\
E &\rightarrow E_i \cdot E_d \quad \{ E.{\text{value}} := E_i.{\text{value}} \cdot E_d.{\text{value}}, E.{\text{type}} := E_i.{\text{type}} \}, \\
E &\rightarrow E_i \div E_d \quad \{ E.{\text{value}} := E_i.{\text{value}} \div E_d.{\text{value}}, E.{\text{type}} := E_i.{\text{type}} \}, \\
E &\rightarrow E_i^E_d \quad \{ E.{\text{value}} := E_i.{\text{value}}^E_d.{\text{value}}, E.{\text{type}} := E_i.{\text{type}} \}, \\
E &\rightarrow (E_d) \quad \{ E.{\text{value}} := E_d.{\text{value}}, E.{\text{type}} := E_i.{\text{type}} \}, \\
E &\rightarrow i \quad \{ E.{\text{value}} := i.{\text{value}}, E.{\text{type}} := i.{\text{type}} \}, \\
E &\rightarrow c \quad \{ E.{\text{value}} := c.{\text{value}}, E.{\text{type}} := c.{\text{type}} \}
\}
\]