Assignment 7.1 Attribute Grammars

The following LL(1) grammar implements the functionality of a pocket calculator. Every key of the calculator emits a token. The keys 0 to 9 emit the token `digit` and all other keys directly translate to the tokens `=`, `+`, `*`, `mw` (memory write), and `mr` (memory read).

<table>
<thead>
<tr>
<th>rule</th>
<th>production</th>
<th>attribute system</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>Start ::= Comp</code></td>
<td>( v[0] := v[1] ) ( m[1] := 0 )</td>
</tr>
<tr>
<td>2</td>
<td><code>Comp ::= Expr</code> ( \cdot )</td>
<td>( v[0] := v[1] ) ( m[1] := m[0] )</td>
</tr>
<tr>
<td>3</td>
<td>`</td>
<td>Expr ( \cdot ) Comp`</td>
</tr>
<tr>
<td>6</td>
<td>`</td>
<td>Term`</td>
</tr>
<tr>
<td>7</td>
<td><code>Term ::= Atom</code> ( \cdot ) <code>Term</code></td>
<td>( v[0] := v[1] \cdot v[3] ) ( m[1] := m[0] ) ( m[3] := m[0] )</td>
</tr>
<tr>
<td>8</td>
<td>`</td>
<td>Atom`</td>
</tr>
<tr>
<td>10</td>
<td>`</td>
<td>mr`</td>
</tr>
<tr>
<td>12</td>
<td>`</td>
<td>\varepsilon`</td>
</tr>
</tbody>
</table>

The result is computed in the \( v \) attribute of `Start`. The key `=` is used to evaluate the expression and clears the input. The value of the internal memory is stored in the attribute \( m \) which is initially zero. The key `mw` evaluates the expression and stores its value in the internal memory of the calculator, and clears the input. The key `mr` recalls this value. For each token `digit` the attribute \( v \) contains the digit as the natural number, i.e., \( v \in [0, 9] \).

1. What is the result after parsing the following key strokes. For a parse error, write “err”.

   - \( 3 7 + 3 = \) : 40
   - \( 0 1 + 3 * 3 = \) : 10
   - \( 0 + = \) : err
   - \( 5 = 6 mw 3 + mr = \) : 33
   - \( 4 mw 2 = mr = \) : 4

2. Complete the definitions for the result value \( v \) and the content of the memory cell \( m \) so that the calculator has the described behavior. You may add other attributes as needed.
3. Which properties does the attributed grammar satisfy?

- ☒ it is l-attributed
- ☐ v is inherited
- ☒ v is synthesized
- ☒ m is inherited
- ☐ m is synthesized
- ☒ it is acyclic

**Assignment 7.2 Strongly Acyclic Attribute Grammars**

Consider Attribute Grammar $G$:

$$
S' \rightarrow A^0 \mid z[0] := z[1] \quad c[1] := 0 \\
B \rightarrow u^0 \mid x[0] := a[0] \quad y[0] := b[0] \\
| \quad v^1 \mid y[0] := x[0] \quad x[0] := 0
$$

1. Draw the local dependency graphs for all production rules $p \in G$.
2. Enumerate all inputs and construct the dependency graphs.
3. Is $G$ acyclic?

**Suggested Solution 7.2**

1. $D(S' \rightarrow A)$:

   ![Diagram for $D(S' \rightarrow A)$]

2. $D(A \rightarrow s \ B)$:

   ![Diagram for $D(A \rightarrow s \ B)$]

3. $D(A \rightarrow t \ B)$:

   ![Diagram for $D(A \rightarrow t \ B)$]

4. $D(B \rightarrow u)$:

   ![Diagram for $D(B \rightarrow u)$]

5. $D(B \rightarrow v)$:

   ![Diagram for $D(B \rightarrow v)$]

2. .

   input $tu$:

   ![Diagram for input $tu$]

   input $tv$:

   ![Diagram for input $tv$]
3. $G$ is acyclic as there are no cyclic dependencies in the derivation trees.