Finishing syntax-directed definitions & translation

Intermediate actions in brace

Top-down parsing

Project description

**SDD (most general form)**

Given a grammar \(G\) associated with each production a set of rules of the form:

\[ b = \alpha(a_1, \ldots, a_n) \]

where \( \alpha \) are attributes or symbols in the production.

\( b \) is a function that computes a term \( f \) of the values \( a_1, \ldots, a_n \).

We say attribute \( b \) depends on \( a_1, \ldots, a_n \) (include \( G \) is an attributed grammar (an SDD))

\[ b \in \Sigma \]

where \( \Sigma \) is some context:

- \( b \) depends on \( \Sigma \) (attributes)
- Formal symbols can also have attributes, these may be provided by the lexical analyzer (synthesized) or inherited, depending on other actions in the production.

On some parseable input \( w \) consider the parse tree \( T \) yielding \( w \).

Each node of \( T \) has some number of attributes associated with it.

Rules determine dependencies between attributes (local) between parent/child or siblings in \( T \).

Determine order of attribute computation by dependency constraint:

- For every attribute \( b \) in \( T \), all attributes that \( b \) depends on must be computed before \( b \).

Directed graph (dependency graph) with vertex \( BV \)

- bring all attributes of all nodes of \( T \), and edges are direct dependencies

\[ u \rightarrow v \]

means \( v \) depends directly on \( u \).

Dependency graph must be acyclic (i.e., a dag)

(only acyclic)

If \( G \) is a dag, then topologically sort it. Arrange vertices in order

\[ u_1, u_2, \ldots, u_n \]

so that if \( v_i \rightarrow v_j \) is any edge, then \( i < j \).

Compute edges in order by

- \( u \), \( v \), \( u \rightarrow v \)

Usually not done in practice because it requires the entire input to be parsed before any output is computed.
Restricted SMDs:
1. S-attributed definition
2. L-attributed definition

S-attributed means syntactical attributes only.

Advantages: All symbols can be computed very quickly.

Disadvantages: If any symbol is present, all operations at the root and at any subtree must be computed.

This is what is designed for.

Disadvantages:
- Can have circular attributes.

Example:
\[
D = T L
T = T L
L = id
L = \text{id}
\]

L-attributed is more general than an S-attributed.

Disadvantages:
- All modifiers and attributes must be computed.
- If any attributes are present, all operations at the root and at any subtree must be computed.

Intermediate entities in a tree

\[
\begin{align*}
\text{expression} & : \text{expr} & & \text{expr} \\
\text{expr} & : \text{exp} \text{expr} & & \text{exp} \\
\text{exp} & : \text{term} \text{exp} & & \text{term} \\
\text{term} & : \text{factor} \text{term} & & \text{factor} \\
\text{factor} & : \text{primary} & & \text{primary} \\
\end{align*}
\]

Generally:

A node can have:
- \( \leq 3 \) children.
- \( \leq 3 \) children.
- Intermediate nodes can have intermediate nodes as children.
- \( \leq 3 \) children.
- \( \leq 3 \) children.

Whereas, it is new.

LaTeX code for a tree with one node:

\[
\begin{align*}
\text{L} & : \text{L} \text{L} \text{L} \text{L} \text{L} \\
\text{L} & : \text{L} \text{L} \text{L} \text{L} \text{L} \\
\text{L} & : \text{L} \text{L} \text{L} \text{L} \text{L} \\
\text{L} & : \text{L} \text{L} \text{L} \text{L} \text{L} \\
\end{align*}
\]

Feb 23-11:33 AM