Motivation

Consider the following grammar and the sentence. The resulting parse tree is:

\[
E \rightarrow E + T \\
| T \\
T \rightarrow T \ast P \\
| P \\
P \rightarrow ( E ) \\
| V
\]

Click here for the parse tree.
Now supposed that instead of having to make decisions as to what reduction to make, we had a grammar where < and > surrounded the right hand sides of productions. Our grammar might look like that below and the sentence and parse tree would be

\[
E \rightarrow < E + T > < < < V > > * < ( < < < V > > > > + < < V > > > ) > > > \\
| < T > \\
T \rightarrow < T * P > \\
| < P > \\
P \rightarrow < ( E ) > \\
| < V >
\]

Click [here](#) for the parse tree.

**Note:** With simple precedence parsing we can compute where the < and > go in a sentential form without requiring the programmer to insert them.
The above parse tree is
The above parse tree is