Example Language Specification

The following attribute grammar defines both the syntax and the semantics for a simple calculator. The syntax is described by each context free grammar production (unindented). The left hand side names a valid node of the abstract syntax tree. The right hand side lists the allowed children, which can be character sequences (terminals), or sub productions (non-terminals). Regular Expressions define some special terminals. The expression semantics are supplied by the (indented) attributes and assertions. An attribute attaches a value to a node, while an assertion checks the validity of attributes. The symbol table maps identifiers, or variable names, to attributes such as value or type.

$\langle \mathit{stmts}_0 \rangle \rightarrow \langle \mathit{stmt}_0 \rangle\ \langle \mathit{stmts}_1 \rangle$

$\langle \mathit{stmts}_0 \rangle \rightarrow \langle \mathit{stmt}_0 \rangle$

$\langle \mathit{stmt}_0 \rangle \rightarrow [\mathit{identifier}_0]$ ‘:=’ $\langle \mathit{expr}_0 \rangle$ ‘;’

  SymTbl.put($[\mathit{itentifier}_0]$, $\langle \mathit{expr}_0 \rangle.\mathit{value}$)

$\langle \mathit{expr}_0 \rangle \rightarrow \langle \mathit{expr}_1 \rangle$ ‘+’ $\langle \mathit{term}_0 \rangle$

  $\langle \mathit{expr}_0 \rangle.\mathit{value} := \langle \mathit{expr}_1 \rangle.\mathit{value} + \langle \mathit{term}_0 \rangle.\mathit{value}$

$\langle \mathit{expr}_0 \rangle \rightarrow \langle \mathit{expr}_1 \rangle$ ‘-’ $\langle \mathit{term}_0 \rangle$

  $\langle \mathit{expr}_0 \rangle.\mathit{value} := \langle \mathit{expr}_1 \rangle.\mathit{value} - \langle \mathit{term}_0 \rangle.\mathit{value}$

$\langle \mathit{expr}_0 \rangle \rightarrow \langle \mathit{term}_0 \rangle$

  $\langle \mathit{expr}_0 \rangle.\mathit{value} := \langle \mathit{term}_0 \rangle.\mathit{value}$

$\langle \mathit{term}_0 \rangle \rightarrow \langle \mathit{term}_1 \rangle$ ‘*’ $\langle \mathit{factor}_0 \rangle$

  $\langle \mathit{term}_0 \rangle.\mathit{value} := \langle \mathit{term}_1 \rangle.\mathit{value} * \langle \mathit{factor}_0 \rangle.\mathit{value}$

$\langle \mathit{term}_0 \rangle \rightarrow \langle \mathit{term}_1 \rangle$ ‘/’ $\langle \mathit{factor}_0 \rangle$

  assert $\langle \mathit{factor}_0 \rangle.\mathit{value} \neq 0$

  $\langle \mathit{term}_0 \rangle.\mathit{value} := \langle \mathit{term}_1 \rangle.\mathit{value} / \langle \mathit{factor}_0 \rangle.\mathit{value}$

$\langle \mathit{term}_0 \rangle \rightarrow \langle \mathit{factor}_0 \rangle$

  $\langle \mathit{term}_0 \rangle.\mathit{value} := \langle \mathit{factor}_1 \rangle.\mathit{value}$

$\langle \mathit{factor}_0 \rangle \rightarrow [\mathit{identifier}_0]$

  assert SymTbl.has($[\mathit{identifier}_0]$)

  $\langle \mathit{factor}_0 \rangle.\mathit{value} :=$ SymTbl.get($[\mathit{identifier}_0]$)

$\langle \mathit{factor}_0 \rangle \rightarrow [\mathit{literal}]$

  $\langle \mathit{factor}_0 \rangle.\mathit{value} :=$ numericParse($[\mathit{literal}]$)

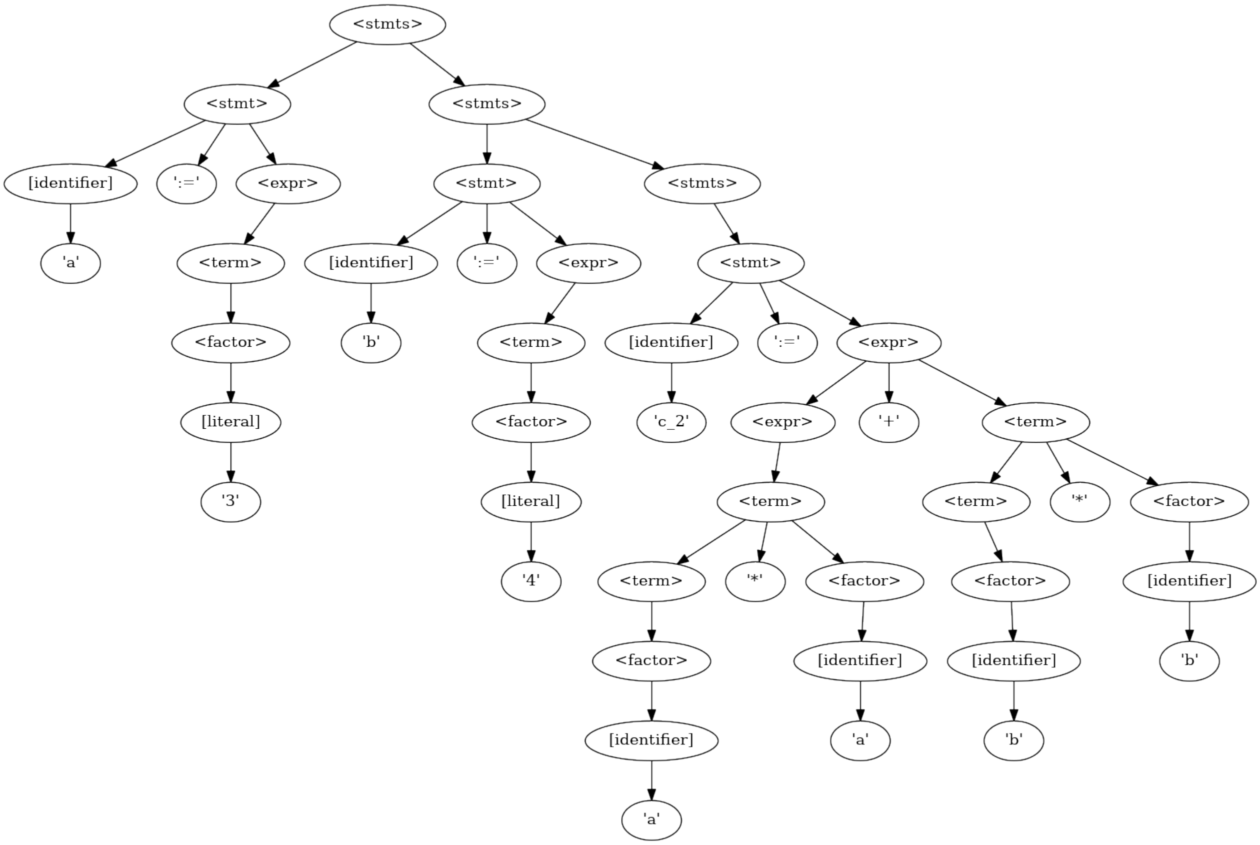
$[\mathit{identifier}] \rightarrow$ `[a-zA-Z_][a-zA-Z0-9_]*`

$[\mathit{literal}] \rightarrow$ `[+-]?[0-9]([0-9_]*[0-9])?`

Example

The code and parse tree ilustrate the $3, 4, 5$ Pythagorean Triple.

```
a := 3;
b := 4;
c_2 := a*a + b*b;
```


<stmts>
<stmt>
<stmts>
[identifier]
':='
<expr>
<stmt>
<stmts>
'a'
<term>
[identifier]
':='
<expr>
<stmt>
<factor>
'b'
<term>
[identifier]
':='
<expr>
[literal]
<factor>
'c_2'
<expr>
'+'
<term>
'3'
[literal]
<term>
<term>
'*'
<factor>
'4'
<term>
'*'
<factor>
<factor>
[identifier]
<factor>
[identifier]
[identifier]
'b'
[identifier]
'a'
'b'
'a'