BNF for the Gryph Programming Language

Vitor Greati Artur Curinga Carlos Vieira
Vinícius Campos

May 8, 2018

Contents

1 General structure

1.1 Program

$$
\begin{array}{rcl}
\langle \text{program} \rangle & \models & \langle \text{program-unit} \rangle \mid \langle \text{program-unit} \rangle \langle \text{program} \rangle \\
\langle \text{program-unit} \rangle & \models & \langle \text{stmt} \rangle \ ; \mid \langle \text{subprog-decl} \rangle
\end{array}
$$

1.2 Statements

$$
\begin{array}{rcl}
\langle \text{stmt-list} \rangle & \models & \langle \text{stmt} \rangle \ ; \mid \langle \text{stmt} \rangle \ ; \langle \text{stmt-list} \rangle \\
\langle \text{stmt} \rangle & \models & \langle \text{read-stmt} \rangle \mid \langle \text{print-stmt} \rangle \mid \langle \text{var-decl-stmt} \rangle
\end{array}
$$

1.2.1 IO

$$
\begin{array}{rcl}
\langle \text{read-stmt} \rangle & \models & \text{read } \langle \text{ident} \rangle \\
\langle \text{write-stmt} \rangle & \models & \text{print } \langle \text{ident} \rangle \mid \text{print } \langle \text{string-lit} \rangle
\end{array}
$$

1.2.2 Variables

$$\begin{aligned}
\langle\text{ident-begin-stmt}\rangle &\models \langle\text{ident-list}\rangle\langle\text{ident-list-post}\rangle \\
\langle\text{ident-list-post}\rangle &\models :\langle\text{type}\rangle\langle\text{var-decl-stmt}\rangle \mid \langle\text{var-attr-stmt}\rangle \\
\langle\text{var-decl-stmt}\rangle &\models \lambda \mid \langle\text{var-attr-stmt}\rangle \\
\langle\text{var-attr-stmt}\rangle &\models =\langle\text{expr-list}\rangle
\end{aligned}$$

2 Expressions

2.1 Expressions with numbers

These rules describe expressions in which numbers appear as operands. The other operand may be a list. The language defines the operators in these expressos as the simpler ones.

$$\begin{aligned}
\langle\text{expr}\rangle &\models \langle\text{term}\rangle \mid \langle\text{term}\rangle\langle\text{expr-aux}\rangle \\
\langle\text{expr-aux}\rangle &\models \langle\text{bin-op-p0}\rangle\langle\text{term}\rangle \mid \langle\text{bin-op-p0}\rangle\langle\text{term}\rangle\langle\text{expr-aux}\rangle \\
\langle\text{term}\rangle &\models \langle\text{un-op}\rangle\langle\text{term}\rangle \mid \langle\text{un-op}\rangle\langle\text{term}\rangle\langle\text{term-aux}\rangle \mid \langle\text{factor}\rangle \mid \langle\text{factor}\rangle\langle\text{term-aux}\rangle \\
\langle\text{term-aux}\rangle &\models \langle\text{bin-op-p1}\rangle\langle\text{factor}\rangle \mid \langle\text{bin-op-p1}\rangle\langle\text{factor}\rangle\langle\text{term-aux}\rangle \\
\langle\text{un-op}\rangle &\models + \mid \text{-} \\
\langle\text{bin-op-p0}\rangle &\models + \mid \text{-} \\
\langle\text{bin-op-p1}\rangle &\models * \mid / \mid \% \mid \hat{\ } \\
\langle\text{factor}\rangle &\models (\ \langle\text{expr}\rangle\ ) \mid \langle\text{ident}\rangle \mid \langle\text{int-lit}\rangle \mid \langle\text{float-lit}\rangle \mid \langle\text{list-lit}\rangle \mid \langle\text{subprog-call}\rangle
\end{aligned}$$