Gryph Programming Language Syntax in EBNF

Vitor Greati Artur Curinga Carlos Vieira Vinícius Campos

June 13, 2018

Contents

1 Program

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

2 Identifiers

$$
\begin{array}{rcl}
\langle\text{id-list}\rangle & \models & \langle\text{identifier}\rangle\{\mathbf{,}\langle\text{identifier}\rangle\} \\
\langle\text{identifier}\rangle & \models & \langle\text{alpha}\rangle\langle\text{id-tail}\rangle \\
\langle\text{user-type-id}\rangle & \models & \langle\text{upper-alpha}\rangle\langle\text{id-tail}\rangle \\
\langle\text{id-tail}\rangle & \models & \{\langle\text{alpha-num}\rangle\}\{\mathbf{'}\} \\
\langle\text{alpha-num}\rangle & \models & \langle\text{alpha}\rangle \mid \langle\text{digit}\rangle \mid \mathbf{\_} \\
\langle\text{digit}\rangle & \models & \mathbf{0} \mid \ldots \mid \mathbf{9} \\
\langle\text{alpha}\rangle & \models & \langle\text{upper-alpha}\rangle \mid (\mathbf{a} \mid \ldots \mid \mathbf{z}) \\
\langle\text{upper-alpha}\rangle & \models & \mathbf{A} \mid \ldots \mid \mathbf{Z}
\end{array}
$$

3 Statements

$$\begin{aligned}
\langle\text{stmt-list}\rangle &\models \langle\text{stmt}\rangle\{\langle\text{stmt}\rangle\}\\
\langle\text{stmt-block}\rangle &\models \textbf{\{}\ \langle\text{stmt-list}\rangle\ \textbf{\}}\\
\langle\text{stmt}\rangle &\models \langle\text{matched-stmt}\rangle \ \mid\ \langle\text{unmatched-stmt}\rangle\\
\langle\text{block-or-matched}\rangle &\models \langle\text{stmt-block}\rangle \ \mid\ \langle\text{matched-stmt}\rangle\\
\langle\text{matched-stmt}\rangle &\models \langle\text{matched-if-else}\rangle \ \mid\ \langle\text{iteration-stmt}\rangle \ \mid\ \langle\text{simple-stmt}\rangle\\
\langle\text{unmatched-stmt}\rangle &\models \langle\text{if-stmt}\rangle \ \mid\ \langle\text{unmatched-if-else}\rangle\\
\langle\text{simple-stmt}\rangle &\models (\langle\text{io-stmt}\rangle \ \mid\ \langle\text{var-stmt}\rangle)\textbf{;}
\end{aligned}$$

3.1 IO

$$\begin{aligned}
\langle\text{io-stmt}\rangle &\models \langle\text{read-stmt}\rangle \ \mid\ \langle\text{write-stmt}\rangle\\
\langle\text{read-stmt}\rangle &\models \textbf{read}\ \langle\text{identifier}\rangle\\
\langle\text{write-stmt}\rangle &\models \textbf{print}\ \langle\text{expression}\rangle
\end{aligned}$$

3.2 Variables

$$\begin{aligned}
\langle\text{var-stmt-list}\rangle &\models \langle\text{var-stmt}\rangle\{\textbf{;}\langle\text{var-stmt}\rangle\}\textbf{;}\\
\langle\text{var-stmt}\rangle &\models \langle\text{var-decl-stmt}\rangle \ \mid\ \langle\text{var-attr-stmt}\rangle\\
\langle\text{var-decl-stmt}\rangle &\models \langle\text{id-list}\rangle\textbf{:}\langle\text{type}\rangle[\langle\text{var-attr}\rangle]\\
\langle\text{var-attr-stmt}\rangle &\models \langle\text{id-list}\rangle\langle\text{var-attr}\rangle\\
\langle\text{var-attr}\rangle &\models \textbf{=}\langle\text{expr-list}\rangle
\end{aligned}$$

4 Control Structures

4.1 Conditionals

$$\begin{aligned}
\langle\text{if-expr}\rangle &\models \textbf{if (}\langle\text{expression}\rangle\textbf{)}\\
\langle\text{if-stmt}\rangle &\models \langle\text{if-expr}\rangle\ \langle\text{stmt}\rangle\\
\langle\text{unmatched-if-else}\rangle &\models \langle\text{if-expr}\rangle\ \langle\text{matched-stmt}\rangle\ \textbf{else}\ \langle\text{unmatched-stmt}\rangle\\
\langle\text{matched-if-else}\rangle &\models \langle\text{if-expr}\rangle\ \langle\text{block-or-matched}\rangle\ \textbf{else}\ \langle\text{block-or-matched}\rangle \ \mid\\
&\ \langle\text{if-expr}\rangle\ \langle\text{stmt-block}\rangle
\end{aligned}$$

4.2 Iteration

$$\begin{aligned}
\langle\text{iteration-stmt}\rangle &\models \langle\text{for-stmt}\rangle \ \mid\ \langle\text{while-stmt}\rangle\\
\langle\text{while-stmt}\rangle &\models \textbf{while}\langle\text{expression}\rangle\langle\text{block-or-matched}\rangle\\
\langle\text{for-loop}\rangle &\models \textbf{for}\langle\text{id-list}\rangle\textbf{over}\langle\text{id-list}\rangle\\
\langle\text{for-stmt}\rangle &\models \langle\text{for-loop}\rangle\langle\text{block-or-matched}\rangle
\end{aligned}$$

5 Subprograms

$$\begin{aligned}
\langle\text{subprog-decl}\rangle &\models \textbf{sub}\langle\text{identifier}\rangle\mathbf{(}\langle\text{parameters}\rangle\mathbf{)}\langle\text{stmt-block}\rangle\\
\langle\text{parameters}\rangle &\models \langle\text{var-stmt}\rangle\{\mathbf{;}\langle\text{var-stmt}\rangle\}\\
\langle\text{subprog-call}\rangle &\models \langle\text{identifier}\rangle\mathbf{(}\langle\text{expr-list}\rangle\mathbf{)}
\end{aligned}$$

6 Types

$$\begin{aligned}
\langle\text{type-list}\rangle &\models \langle\text{type}\rangle\{\mathbf{,}\langle\text{type}\rangle\}\\
\langle\text{type}\rangle &\models \langle\text{native-type}\rangle \mid \langle\text{user-type-id}\rangle\\
\langle\text{native-type}\rangle &\models \langle\text{primitive-type}\rangle \mid \langle\text{composite-type}\rangle\\
\langle\text{primitive-type}\rangle &\models \textbf{int} \mid \textbf{float} \mid \textbf{char} \mid \textbf{string}\\
\langle\text{composite-type}\rangle &\models \mathbf{[}\langle\text{type}\rangle\mathbf{]} \mid \mathbf{|}\langle\text{type}\rangle\mathbf{|} \mid \mathbf{(}\langle\text{type}\rangle\mathbf{,}\langle\text{type-list}\rangle\mathbf{)} \mid \langle\text{graph-type}\rangle\\
\langle\text{graph-type}\rangle &\models \mathbf{<}\langle\text{type}\rangle\mathbf{>} \mid \mathbf{<}\langle\text{type}\rangle\mathbf{,}\langle\text{type}\rangle\mathbf{>}\\
\langle\text{type-decl}\rangle &\models \langle\text{user-type-id}\rangle\boldsymbol{\{}\langle\text{var-stmt-list}\rangle\boldsymbol{\}}
\end{aligned}$$

Observation Although there is no maximum size for tuples in the definition above, there may be one for specific language implementations.

7 Expressions

$$\begin{aligned}
\langle\text{expression}\rangle &\models \langle\text{logical-xor-expr}\rangle\\
\langle\text{logical-xor-expr}\rangle &\models \langle\text{logical-or-expr}\rangle \mid \langle\text{logical-or-expr}\rangle\langle\text{logical-xor-expr-aux}\rangle\\
\langle\text{logical-xor-expr-aux}\rangle &\models \textbf{xor}\ \langle\text{logical-or-expr}\rangle \mid \textbf{xor}\ \langle\text{logical-or-expr}\rangle\langle\text{logical-xor-expr-aux}\rangle\\
\langle\text{logical-or-expr}\rangle &\models \langle\text{logical-and-expr}\rangle \mid \langle\text{logical-and-expr}\rangle\langle\text{logical-or-expr-aux}\rangle\\
\langle\text{logical-or-expr-aux}\rangle &\models \textbf{or}\ \langle\text{logical-and-expr}\rangle \mid \textbf{or}\ \langle\text{logical-and-expr}\rangle\langle\text{logical-or-expr-aux}\rangle\\
\langle\text{logical-and-expr}\rangle &\models \langle\text{equality-expr}\rangle \mid \langle\text{equality-expr}\rangle\langle\text{logical-and-expr-aux}\rangle\\
\langle\text{logical-and-expr-aux}\rangle &\models \textbf{and}\ \langle\text{equality-expr}\rangle \mid \textbf{and}\ \langle\text{equality-expr}\rangle\langle\text{logical-and-expr-aux}\rangle\\
\langle\text{equality-expr}\rangle &\models \langle\text{rel-expr}\rangle \mid \langle\text{rel-expr}\rangle\langle\text{rel-expr-aux}\rangle\\
\langle\text{equality-expr-aux}\rangle &\models \langle\text{equality-op}\rangle\ \langle\text{rel-expr}\rangle \mid \langle\text{equality-op}\rangle\ \langle\text{rel-expr}\rangle\langle\text{equality-expr-aux}\rangle\\
\langle\text{rel-expr}\rangle &\models \langle\text{add-expr}\rangle\langle\text{rel-expr-aux}\rangle\\
\langle\text{rel-expr-aux}\rangle &\models \langle\text{rel-op}\rangle\langle\text{add-expr}\rangle \mid \langle\text{rel-op}\rangle\langle\text{add-expr}\rangle\langle\text{rel-expr-aux}\rangle\\
\langle\text{add-expr}\rangle &\models \langle\text{mult-expr}\rangle \mid \langle\text{mult-expr}\rangle\langle\text{add-expr-aux}\rangle\\
\langle\text{add-expr-aux}\rangle &\models \langle\text{add-op}\rangle\langle\text{mult-expr}\rangle \mid \langle\text{add-op}\rangle\langle\text{mult-expr}\rangle\langle\text{add-expr-aux}\rangle\\
\langle\text{mult-expr}\rangle &\models \langle\text{exp-expr}\rangle \mid \langle\text{exp-expr}\rangle\langle\text{mult-expr-aux}\rangle\\
\langle\text{mult-expr-aux}\rangle &\models \langle\text{mult-op}\rangle\langle\text{exp-expr}\rangle \mid \langle\text{mult-op}\rangle\langle\text{exp-expr}\rangle\langle\text{mult-expr-aux}\rangle\\
\langle\text{exp-expr}\rangle &\models \langle\text{cast-expr}\rangle \mid \langle\text{cast-expr}\rangle\langle\text{exp-op}\rangle\langle\text{exp-expr}\rangle\\
\langle\text{cast-expr}\rangle &\models \langle\text{unary-expr}\rangle \mid \langle\text{unary-expr}\rangle\langle\text{cast-expr-aux}\rangle\\
\langle\text{cast-expr-aux}\rangle &\models \mathbf{@}\langle\text{type}\rangle \mid \mathbf{@}\langle\text{type}\rangle\langle\text{cast-expr-aux}\rangle\\
\langle\text{unary-expr}\rangle &\models \langle\text{unary-op}\rangle\langle\text{cast-expr}\rangle \mid \langle\text{postfix-expr}\rangle\\
\langle\text{postfix-expr}\rangle &\models \langle\text{primary-expr}\rangle \mid \langle\text{identifier}\rangle\langle\text{access-expr}\rangle\\
\langle\text{access-expr}\rangle &\models \mathbf{|}\langle\text{expression}\rangle\mathbf{|} \mid \mathbf{<}\langle\text{expression}\rangle\mathbf{>} \mid \mathbf{[}\langle\text{expression}\rangle\mathbf{]} \mid \boldsymbol{\{}\langle\text{identifier}\rangle\boldsymbol{\}} \mid \mathbf{.}\langle\text{expression}\rangle\\
\langle\text{primary-expr}\rangle &\models \mathbf{(}\langle\text{expression}\rangle\mathbf{)} \mid \langle\text{identifier}\rangle \mid \langle\text{subprog-call}\rangle \mid \langle\text{litearal}\rangle \mid \langle\text{structure}\rangle
\end{aligned}$$

7.1 Literals

$$
\begin{array}{rcl}
\langle\text{literal}\rangle & \models & \langle\text{int-lit}\rangle \mid \langle\text{float-lit}\rangle \mid \langle\text{string-lit}\rangle \mid \langle\text{bool-lit}\rangle \mid \langle\text{char-lit}\rangle \\
\langle\text{bool-lit}\rangle & \models & \texttt{\textbf{true}} \mid \texttt{\textbf{false}} \\
\langle\text{string-lit}\rangle & \models & \texttt{"}\{\langle\text{char}\rangle\}\texttt{"} \\
\langle\text{char-lit}\rangle & \models & \texttt{'}\langle\text{char}\rangle\texttt{'} \\
\langle\text{char}\rangle & \models & \texttt{implementation dependent} \\
\langle\text{int-lit}\rangle & \models & [\texttt{-}]\langle\text{digit-seq}\rangle \\
\langle\text{float-lit}\rangle & \models & [\texttt{-}]\langle\text{digit-seq}\rangle\texttt{.}\langle\text{digit-seq}\rangle \\
\langle\text{digit-seq}\rangle & \models & \langle\text{digit}\rangle\{\langle\text{digit}\rangle\}
\end{array}
$$

7.2 Structures

$$
\begin{array}{rcl}
\langle\text{structure}\rangle & \models & \langle\text{tuple}\rangle \mid \langle\text{list}\rangle \mid \langle\text{dict}\rangle \mid \langle\text{graph}\rangle \mid \langle\text{user-type}\rangle \mid \langle\text{edge}\rangle \\
\langle\text{tuple}\rangle & \models & \textbf{(}\langle\text{expr-list}\rangle\textbf{)} \\
\langle\text{dict}\rangle & \models & |\langle\text{dict-entry-list}\rangle| \\
\langle\text{dict-entry}\rangle & \models & \langle\text{expression}\rangle\textbf{?}\langle\text{expression}\rangle \\
\langle\text{dict-entry-list}\rangle & \models & \langle\text{dict-entry}\rangle\{\textbf{,}\langle\text{dict-entry}\rangle\} \\
\langle\text{user-type}\rangle & \models & \langle\text{user-type-id}\rangle\boldsymbol{\{}\langle\text{var-attr-stmt}\rangle\{,\langle\text{var-attr-stmt}\rangle\}\boldsymbol{\}} \\
\langle\text{list}\rangle & \models & \textbf{[}(\langle\text{expr-list}\rangle \mid \langle\text{list-comprehension}\rangle)\textbf{]} \\
\langle\text{list-comprehension}\rangle & \models & \langle\text{expression}\rangle\langle\text{for-loop}\rangle[\langle\text{comp-condition}\rangle] \\
\langle\text{graph-comprehension}\rangle & \models & \langle\text{edge}\rangle\langle\text{for-loop}\rangle[\langle\text{comp-condition}\rangle] \\
\langle\text{comp-condition}\rangle & \models & \textbf{when(}\langle\text{expression}\rangle\textbf{)} \\
\langle\text{graph}\rangle & \models & <[\langle\text{vertex-set}\rangle,]\langle\text{edge-set}\rangle> \\
\langle\text{vertex-set}\rangle & \models & \langle\text{expression}\rangle \\
\langle\text{edge-set}\rangle & \models & [\langle\text{edge-weight}\rangle]\langle\text{graph-comprehension}\rangle \\
\langle\text{edge-weight}\rangle & \models & \langle\text{expression}\rangle\textbf{where} \\
\langle\text{edge}\rangle & \models & \langle\text{expression}\rangle\langle\text{edge-symbol}\rangle\langle\text{expression}\rangle \\
\langle\text{edge-symbol}\rangle & \models & -- \mid -> \mid <-
\end{array}
$$

7.3 Operators

$$
\begin{array}{rcl}
\langle\text{rel-op}\rangle & \models & > \mid < \mid <= \mid >= \\
\langle\text{equality-op}\rangle & \models & == \mid \textbf{!}= \\
\langle\text{unary-op}\rangle & \models & + \mid - \\
\langle\text{add-op}\rangle & \models & + \mid \texttt{-} \\
\langle\text{mult-op}\rangle & \models & \texttt{*} \mid / \mid \% \mid ++ \mid \texttt{**} \\
\langle\text{exp-op}\rangle & \models & \hat{}
\end{array}
$$