Gryph Programming Language Syntax in EBNF

Vitor Greati Artur Curinga Carlos Vieira Vinícius Campos

June 17, 2018

Contents

1 Syntax in EBNF

1.1 Program

$$\begin{array}{rcl}
\langle\text{program}\rangle & \models & \langle\text{program-unit}\rangle\{\langle\text{program-unit}\rangle\} \\
\langle\text{program-unit}\rangle & \models & \langle\text{stmt}\rangle \mid \langle\text{subprog-def}\rangle \mid \langle\text{type-def}\rangle \mid \langle\text{include}\rangle \\
\langle\text{include}\rangle & \models & \textbf{use}\ \langle\text{string-lit}\rangle
\end{array}$$

1.2 Identifiers

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

1.3 Statements

⟨stmt-list⟩ ⊨ ⟨stmt⟩{⟨stmt⟩}

⟨stmt-block⟩ ⊨ **{** ⟨stmt-list⟩ **}**

⟨stmt⟩ ⊨ ⟨matched-stmt⟩ | ⟨unmatched-stmt⟩

⟨block-or-matched⟩ ⊨ ⟨stmt-block⟩ | ⟨matched-stmt⟩

⟨matched-stmt⟩ ⊨ ⟨matched-if-else⟩ | ⟨iteration-stmt⟩ | ⟨simple-stmt⟩

⟨unmatched-stmt⟩ ⊨ ⟨if-stmt⟩ | ⟨unmatched-if-else⟩

⟨simple-stmt⟩ ⊨ (⟨io-stmt⟩ | ⟨var-stmt⟩ | ⟨add-del-stmt⟩ | ⟨subprog-call⟩ | ⟨return-stmt⟩ | ⟨break-stmt⟩)**;**

1.3.1 IO

⟨io-stmt⟩ ⊨ ⟨read-stmt⟩ | ⟨write-stmt⟩

⟨read-stmt⟩ ⊨ **read** ⟨identifier⟩

⟨write-stmt⟩ ⊨ **print** ⟨expression⟩

1.3.2 Variables

⟨var-stmt⟩ ⊨ ⟨var-decl-stmt⟩ | ⟨var-attr-stmt⟩

⟨var-decl-list⟩ ⊨ ⟨var-decl-stmt⟩{**;**⟨var-decl-stmt⟩}**;**

⟨var-decl-stmt⟩ ⊨ ⟨id-list⟩**:**⟨type⟩[⟨var-attr⟩]

⟨var-attr-stmt⟩ ⊨ ⟨lhs-expr-list⟩⟨var-attr⟩

⟨id-attr⟩ ⊨ ⟨identifier⟩**=**⟨expression⟩

⟨id-attr-list⟩ ⊨ ⟨id-attr⟩{, ⟨id-attr⟩}

⟨var-attr⟩ ⊨ **=**⟨expr-list⟩

1.3.3 Insertion and removal

⟨add-del-stmt⟩ ⊨ ⟨add-stmt⟩ | ⟨del-stmt⟩

⟨add-stmt⟩ ⊨ **add** ⟨expression⟩ **in** ⟨lhs-expr⟩

⟨del-stmt⟩ ⊨ **del** ⟨expression⟩ **from** ⟨lhs-expr⟩

1.4 Control Structures

1.4.1 Conditionals

⟨if-expr⟩ ⊨ **if (**⟨expression⟩**)**

⟨if-stmt⟩ ⊨ ⟨if-expr⟩ ⟨stmt⟩

⟨unmatched-if-else⟩ ⊨ ⟨if-expr⟩ ⟨matched-stmt⟩ **else** ⟨unmatched-stmt⟩

⟨matched-if-else⟩ ⊨ ⟨if-expr⟩ ⟨block-or-matched⟩ **else** ⟨block-or-matched⟩ | ⟨if-expr⟩ ⟨stmt-block⟩

1.4.2 Iteration

$$
\begin{array}{rcl}
\langle\text{iteration-stmt}\rangle & \models & \langle\text{for-stmt}\rangle \mid \langle\text{while-stmt}\rangle \mid \langle\text{bfs-dfs-stmt}\rangle \\
\langle\text{while-stmt}\rangle & \models & \textbf{while}\ \langle\text{expression}\rangle\ \langle\text{block-or-matched}\rangle \\
\langle\text{for-loop}\rangle & \models & \textbf{for}\ \langle\text{id-list}\rangle\ \textbf{over}\ \langle\text{expr-list}\rangle \\
\langle\text{for-stmt}\rangle & \models & \langle\text{for-loop}\rangle\langle\text{block-or-matched}\rangle \\
\langle\text{bfs-dfs-loop}\rangle & \models & (\textbf{bfs} \mid \textbf{dfs})\ \langle\text{id-list}\rangle\ \textbf{over}\ \langle\text{expr-list}\rangle \\
\langle\text{bfs-dfs-stmt}\rangle & \models & \langle\text{dfs-dfs-loop}\rangle\langle\text{block-or-matched}\rangle \\
\langle\text{break-stmt}\rangle & \models & \textbf{break}
\end{array}
$$

1.5 Subprograms

$$
\begin{array}{rcl}
\langle\text{subprog-def}\rangle & \models & \textbf{sub}\langle\text{identifier}\rangle\boldsymbol{(}\langle\text{parameters}\rangle\boldsymbol{)}[\boldsymbol{:}\langle\text{type}\rangle]\langle\text{stmt-block}\rangle \\
\langle\text{parameters}\rangle & \models & \langle\text{var-stmt}\rangle\{\boldsymbol{;}\langle\text{var-stmt}\rangle\} \\
\langle\text{subprog-call}\rangle & \models & \langle\text{identifier}\rangle\boldsymbol{(}[\langle\text{arguments}\rangle]\boldsymbol{)} \\
\langle\text{arguments}\rangle & \models & \langle\text{id-attr-list}\rangle \mid \langle\text{expr-list}\rangle \\
\langle\text{return-stmt}\rangle & \models & \textbf{return}
\end{array}
$$

1.6 Types

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

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

1.7 Expressions

$$
\begin{array}{rcl}
\langle\text{expr-list}\rangle & \models & \langle\text{expression}\rangle\{\boldsymbol{,}\langle\text{expression}\rangle\} \\
\langle\text{expression}\rangle & \models & \langle\text{logical-xor-expr}\rangle \\
\langle\text{logical-xor-expr}\rangle & \models & \langle\text{logical-or-expr}\rangle\{\textbf{xor}\ \langle\text{logical-or-expr}\rangle\} \\
\langle\text{logical-or-expr}\rangle & \models & \langle\text{logical-and-expr}\rangle\{\textbf{or}\ \langle\text{logical-and-expr}\rangle\} \\
\langle\text{logical-and-expr}\rangle & \models & \langle\text{equality-expr}\rangle\{\textbf{and}\ \langle\text{equality-expr}\rangle\} \\
\langle\text{equality-expr}\rangle & \models & \langle\text{rel-expr}\rangle\{\langle\text{equality-op}\rangle\ \langle\text{rel-expr}\rangle\} \\
\langle\text{rel-expr}\rangle & \models & \langle\text{add-expr}\rangle\{\langle\text{rel-op}\rangle\langle\text{add-expr}\rangle\} \\
\langle\text{add-expr}\rangle & \models & \langle\text{mult-expr}\rangle\{\langle\text{add-op}\rangle\langle\text{mult-expr}\rangle\} \\
\langle\text{mult-expr}\rangle & \models & \langle\text{exp-expr}\rangle\{\langle\text{mult-op}\rangle\langle\text{exp-expr}\rangle\} \\
\langle\text{exp-expr}\rangle & \models & \langle\text{cast-expr}\rangle[\langle\text{exp-op}\rangle\langle\text{exp-expr}\rangle] \\
\langle\text{cast-expr}\rangle & \models & \langle\text{unary-expr}\rangle\{\boldsymbol{@}\langle\text{type}\rangle\}
\end{array}
$$

⟨unary-expr⟩ ⊨ ⟨unary-op⟩⟨cast-expr⟩ | ⟨postfix-expr⟩

⟨postfix-expr⟩ ⊨ ⟨primary-expr⟩{⟨access-expr⟩}

⟨lhs-expr⟩ ⊨ ⟨identifier⟩{⟨access-expr⟩}

⟨lhs-expr-list⟩ ⊨ ⟨lhs-expr⟩{,⟨lhs-expr⟩}

⟨access-expr⟩ ⊨ |⟨expression⟩| | <⟨expression⟩> | **[**⟨expression⟩**]** | **{**⟨identifier⟩**}** | \⟨expression⟩\

⟨primary-expr⟩ ⊨ **(**⟨expression⟩**)** | ⟨identifier⟩ | ⟨subprog-call⟩ | ⟨literal⟩ | ⟨structure⟩

1.7.1 Literals

⟨literal⟩ ⊨ ⟨int-lit⟩ | ⟨float-lit⟩ | ⟨string-lit⟩ | ⟨bool-lit⟩ | ⟨char-lit⟩

⟨bool-lit⟩ ⊨ **true** | **false**

⟨string-lit⟩ ⊨ "{⟨char⟩}"

⟨char-lit⟩ ⊨ '⟨char⟩'

⟨char⟩ ⊨ `implementation dependent`

⟨int-lit⟩ ⊨ [-]⟨digit-seq⟩

⟨float-lit⟩ ⊨ [-]⟨digit-seq⟩.⟨digit-seq⟩

⟨digit-seq⟩ ⊨ ⟨digit⟩{⟨digit⟩}

Observation A char must be one character, of an enconding defined by the implementation.

1.7.2 Structures

⟨structure⟩ ⊨ ⟨tuple⟩ | ⟨list⟩ | ⟨dict⟩ | ⟨graph⟩ | ⟨user-type⟩ | ⟨edge⟩

⟨tuple⟩ ⊨ **(**⟨expr-list⟩**)**

⟨dict⟩ ⊨ |[⟨dict-entry-list⟩]|

⟨dict-entry⟩ ⊨ ⟨expression⟩**?**⟨expression⟩

⟨dict-entry-list⟩ ⊨ ⟨dict-entry⟩{,⟨dict-entry⟩}

⟨user-type⟩ ⊨ ⟨user-type-id⟩**{**[⟨id-attr-list⟩]**}**

⟨list⟩ ⊨ **[**[⟨list-expr⟩]**]**

⟨list-expr⟩ ⊨ ⟨expr-list⟩ | ⟨list-comprehension⟩

⟨list-comprehension⟩ ⊨ ⟨expression⟩⟨for-loop⟩[⟨comp-condition⟩]

⟨graph-comprehension⟩ ⊨ ⟨edge⟩⟨for-loop⟩[⟨comp-condition⟩]

⟨comp-condition⟩ ⊨ **when (**⟨expression⟩**)**

⟨graph⟩ ⊨ <(⟨vertex-set⟩, ⟨edge-set⟩ | ⟨vertex-set⟩ | ⟨edge-set⟩)>

⟨vertex-set⟩ ⊨ ⟨expression⟩

⟨edge-set⟩ ⊨ [⟨edge-weight⟩]⟨graph-comprehension⟩

⟨edge-weight⟩ ⊨ ⟨expression⟩ **where**

⟨edge⟩ ⊨ ⟨expression⟩⟨edge-symbol⟩⟨expression⟩

⟨edge-symbol⟩ ⊨ −− | −> | <−

1.7.3 Operators

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