Gryph Programming Language Syntax in EBNF

Vitor Greati Artur Curinga Carlos Vieira Vinícius Campos

June 15, 2018

Contents

1 Program

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

2 Identifiers

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

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⟩)**;**

3.1 IO

⟨io-stmt⟩ ⊨ ⟨read-stmt⟩ | ⟨write-stmt⟩
⟨read-stmt⟩ ⊨ **read** ⟨identifier⟩
⟨write-stmt⟩ ⊨ **print** ⟨expression⟩

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⟩ ⊨ ⟨expr-list⟩⟨var-attr⟩
⟨single-var-attr⟩ ⊨ ⟨expr⟩⟨var-attr⟩
⟨var-attr⟩ ⊨ **=**⟨expr-list⟩

4 Control Structures

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⟩

4.2 Iteration

⟨iteration-stmt⟩ ⊨ ⟨for-stmt⟩ | ⟨while-stmt⟩
⟨while-stmt⟩ ⊨ **while**⟨expression⟩⟨block-or-matched⟩
⟨for-loop⟩ ⊨ **for**⟨id-list⟩**over**⟨expr-list⟩
⟨for-stmt⟩ ⊨ ⟨for-loop⟩⟨block-or-matched⟩

5 Subprograms

⟨subprog-decl⟩ ⊨ **sub**⟨identifier⟩**(**⟨parameters⟩**)**[**:**⟨type⟩]⟨stmt-block⟩
⟨parameters⟩ ⊨ ⟨var-stmt⟩{**;**⟨var-stmt⟩}
⟨subprog-call⟩ ⊨ ⟨identifier⟩**(**⟨expr-list⟩**)**

6 Types

⟨type-list⟩ ⊨ ⟨type⟩{**,**⟨type⟩}
⟨type⟩ ⊨ ⟨native-type⟩ | ⟨user-type-id⟩
⟨native-type⟩ ⊨ ⟨primitive-type⟩ | ⟨composite-type⟩
⟨primitive-type⟩ ⊨ **int** | **float** | **char** | **string**
⟨composite-type⟩ ⊨ **[**⟨type⟩**]** | **|**⟨type⟩**|** | **(**⟨type⟩**,**⟨type-list⟩**)** | ⟨graph-type⟩
⟨graph-type⟩ ⊨ **<**⟨type⟩**>** | **<**⟨type⟩**,**⟨type⟩**>**
⟨type-decl⟩ ⊨ ⟨user-type-id⟩**{**⟨var-decl-list⟩**}**

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

7 Expressions

⟨expr-list⟩ ⊨ ⟨expression⟩{**,**⟨expression⟩}
⟨expression⟩ ⊨ ⟨logical-xor-expr⟩
⟨logical-xor-expr⟩ ⊨ ⟨logical-or-expr⟩{**xor** ⟨logical-or-expr⟩}
⟨logical-or-expr⟩ ⊨ ⟨logical-and-expr⟩{**or** ⟨logical-and-expr⟩}
⟨logical-and-expr⟩ ⊨ ⟨equality-expr⟩{**and** ⟨equality-expr⟩}
⟨equality-expr⟩ ⊨ ⟨rel-expr⟩{⟨equality-op⟩ ⟨rel-expr⟩}
⟨rel-expr⟩ ⊨ ⟨add-expr⟩{⟨rel-op⟩⟨add-expr⟩}
⟨add-expr⟩ ⊨ ⟨mult-expr⟩{⟨add-op⟩⟨mult-expr⟩}
⟨mult-expr⟩ ⊨ ⟨exp-expr⟩{⟨mult-op⟩⟨exp-expr⟩}
⟨exp-expr⟩ ⊨ ⟨cast-expr⟩[⟨exp-op⟩⟨exp-expr⟩]
⟨cast-expr⟩ ⊨ ⟨unary-expr⟩{**@**⟨type⟩}
⟨unary-expr⟩ ⊨ ⟨unary-op⟩⟨cast-expr⟩ | ⟨postfix-expr⟩
⟨postfix-expr⟩ ⊨ ⟨primary-expr⟩{⟨access-expr⟩}
⟨access-expr⟩ ⊨ **|**⟨expression⟩**|** | **<**⟨expression⟩**>** | **[**⟨expression⟩**]** | **{**⟨identifier⟩**}** | **.**⟨expression⟩
⟨primary-expr⟩ ⊨ **(**⟨expression⟩**)** | ⟨identifier⟩ | ⟨subprog-call⟩ | ⟨literal⟩ | ⟨structure⟩

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⟩}

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⟩**{**⟨single-var-attr⟩{, ⟨single-var-attr⟩}**}**
⟨list⟩ ⊨ **[**(⟨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⟩ ⊨ ⟨expression⟩
⟨edge-set⟩ ⊨ [⟨edge-weight⟩]⟨graph-comprehension⟩
⟨edge-weight⟩ ⊨ ⟨expression⟩**where**
⟨edge⟩ ⊨ ⟨expression⟩⟨edge-symbol⟩⟨expression⟩
⟨edge-symbol⟩ ⊨ −− | − > | < −

7.3 Operators

⟨rel-op⟩ ⊨ > | < | <= | >=
⟨equality-op⟩ ⊨ == | **!** =
⟨unary-op⟩ ⊨ + | −
⟨add-op⟩ ⊨ + | **-**
⟨mult-op⟩ ⊨ ***** | / | % | ++ | ******
⟨exp-op⟩ ⊨ ^