BNF for the Gryph Programming Language

Vitor Greati Artur Curinga Carlos Vieira Vinícius Campos

May 17, 2018

Contents

1 General structure

1.1 Program

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

1.2 Statements

$$\begin{aligned} \langle \text{stmt-list} \rangle &\models \langle \text{stmt} \rangle \ ; \mid \langle \text{stmt} \rangle \ ; \langle \text{stmt-list} \rangle \\ \langle \text{stmt-block} \rangle &\models \{ \ \langle \text{stmt-list} \rangle \ \} \\ \langle \text{block-or-matched} \rangle &\models \langle \text{stmt-block} \rangle \mid \langle \text{matched-stmt} \rangle ; \\ \langle \text{com-stmt} \rangle &\models \langle \text{read-stmt} \rangle \mid \langle \text{print-stmt} \rangle \mid \langle \text{var-decl-stmt} \rangle \\ \langle \text{stmt} \rangle &\models \langle \text{matched-stmt} \rangle \mid \langle \text{unmatched-stmt} \rangle \\ \langle \text{matched-stmt} \rangle &\models \langle \text{matched-if-else} \rangle \mid \langle \text{com-stmt} \rangle \\ \langle \text{unmatched-stmt} \rangle &\models \langle \text{if-stmt} \rangle \mid \langle \text{unmatched-if-else} \rangle \end{aligned}$$

1.2.1 IO

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

1.2.2 Variables

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

1.3 Subprograms

1.3.1 Declaration

1.3.2 Call

$$\langle \text{subprog-call} \rangle \models \langle \text{ident} \rangle (\langle \text{expr-list} \rangle)$$

2 Control Structures

2.1 If-else statements

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

3 Types

$$\begin{aligned} \langle \text{type-list} \rangle &\models \langle \text{type} \rangle, \langle \text{type-list} \rangle \mid \langle \text{type} \rangle \\ \langle \text{type} \rangle &\models \langle \text{native-type} \rangle \mid \langle \text{user-type} \rangle \\ \langle \text{native-type} \rangle &\models \langle \text{primitive-type} \rangle \mid \langle \text{composite-type} \rangle \\ \langle \text{primitive-type} \rangle &\models \text{int} \mid \text{float} \mid \text{char} \mid \text{string} \\ \langle \text{composite-type} \rangle &\models [\langle \text{type} \rangle] \mid |\langle \text{type} \rangle| \mid (\langle \text{type} \rangle, \langle \text{type-list} \rangle) \mid \langle \text{graph-type} \rangle \\ \langle \text{graph-type} \rangle &\models <\langle \text{type} \rangle> \mid <\langle \text{type} \rangle, \langle \text{type} \rangle> \\ \langle \text{user-type} \rangle &\models \langle \text{upper-letter} \rangle \langle \text{alpha-num-list} \rangle \end{aligned}$$

Observations

- The maximum size of tuples depends on the language implementation, though, in the BNF description above, it may assume any value.

4 Expressions

4.1 Any expression

⟨expression⟩ ⊨ ⟨logical-xor-expr⟩
⟨logical-xor-expr⟩ ⊨ ⟨logical-or-expr⟩ | ⟨logical-or-expr⟩⟨logical-xor-expr-aux⟩
⟨logical-xor-expr-aux⟩ ⊨ xor ⟨logical-or-expr⟩ | xor ⟨logical-or-expr⟩⟨logical-xor-expr-aux⟩
⟨logical-or-expr⟩ ⊨ ⟨logical-and-expr⟩ | ⟨logical-and-expr⟩⟨logical-or-expr-aux⟩
⟨logical-or-expr-aux⟩ ⊨ or ⟨logical-and-expr⟩ | or ⟨logical-and-expr⟩⟨logical-or-expr-aux⟩
⟨logical-and-expr⟩ ⊨ ⟨equality-expr⟩ | ⟨equality-expr⟩⟨logical-and-expr-aux⟩
⟨logical-and-expr-aux⟩ ⊨ and ⟨equality-expr⟩ | and ⟨equality-expr⟩⟨logical-and-expr-aux⟩
⟨equality-expr⟩ ⊨ ⟨rel-expr⟩ | ⟨rel-expr⟩⟨rel-expr-aux⟩
⟨equality-expr-aux⟩ ⊨ ⟨equality-op⟩ ⟨rel-expr⟩ | ⟨equality-op⟩ ⟨rel-expr⟩⟨equality-expr-aux⟩
⟨rel-expr⟩ ⊨ ⟨add-expr⟩⟨rel-expr-aux⟩
⟨rel-expr-aux⟩ ⊨ ⟨rel-op⟩⟨add-expr⟩ | ⟨rel-op⟩⟨add-expr⟩⟨rel-expr-aux⟩
⟨add-expr⟩ ⊨ ⟨mult-expr⟩ | ⟨mult-expr⟩⟨add-expr-aux⟩
⟨add-expr-aux⟩ ⊨ ⟨add-op⟩⟨mult-expr⟩ | ⟨add-op⟩⟨mult-expr⟩⟨add-expr-aux⟩
⟨mult-expr⟩ ⊨ ⟨exp-expr⟩ | ⟨exp-expr⟩⟨mult-expr-aux⟩
⟨mult-expr-aux⟩ ⊨ ⟨mult-op⟩⟨exp-expr⟩ | ⟨mult-op⟩⟨exp-expr⟩⟨mult-expr-aux⟩
⟨exp-expr⟩ ⊨ ⟨cast-expr⟩ | ⟨cast-expr⟩⟨exp-op⟩⟨exp-expr⟩
⟨cast-expr⟩ ⊨ ⟨unary-expr⟩ | ⟨unary-expr⟩⟨cast-expr-aux⟩
⟨cast-expr-aux⟩ ⊨ @⟨type⟩ | @⟨type⟩⟨cast-expr-aux⟩
⟨unary-expr⟩ ⊨ ⟨unary-op⟩⟨cast-expr⟩ | ⟨postfix-expr⟩
⟨postfix-expr⟩ ⊨ ⟨primary-expr⟩ | ⟨ident⟩|expression| | ⟨ident⟩<expression> | ⟨ident⟩[expression] | ⟨ident⟩{⟨ident⟩} | ⟨ident⟩.⟨expression⟩
⟨primary-expr⟩ ⊨ (⟨expression⟩) | ⟨ident⟩ | ⟨subprogcall⟩ | ⟨constant⟩
⟨constant⟩ ⊨ ⟨int-lit⟩ | ⟨float-lit⟩ | ⟨string-lit⟩ | ⟨bool-lit⟩ | ⟨list-lit⟩ | ⟨graph-lit⟩
⟨rel-op⟩ ⊨ > | < | <= | >=
⟨equality-op⟩ ⊨ == | !=
⟨unary-op⟩ ⊨ + | -
⟨add-op⟩ ⊨ + | -
⟨mult-op⟩ ⊨ * | / | % | ++ | **
⟨exp-op⟩ ⊨ ^

4.2 Relational expressions

⟨rel-expr⟩ ⊨ ⟨rel-term⟩⟨rel-expr-aux⟩
⟨rel-expr-aux⟩ ⊨ ⟨rel-op⟩⟨rel-term⟩ | ⟨rel-op⟩⟨rel-term⟩⟨rel-expr-aux⟩
⟨rel-op⟩ ⊨ > | < | <= | >= | == | !=
⟨rel-term⟩ ⊨ (⟨rel-expr⟩) | ⟨bool-expr⟩ | ⟨expr⟩

4.3 Boolean expressions

⟨b-expr⟩ ⊨ ⟨b-term⟩ | ⟨b-term⟩⟨b-expr-aux⟩
⟨b-expr-aux⟩ ⊨ ⟨b-bin-op-p0⟩⟨b-term⟩ | ⟨b-bin-op-p0⟩⟨b-term⟩⟨b-expr-aux⟩
⟨b-term⟩ ⊨ ⟨b-literal⟩ | ⟨b-literal⟩⟨b-term-aux⟩
⟨b-term-aux⟩ ⊨ ⟨b-bin-op-p1⟩⟨b-literal⟩ | ⟨b-bin-op-p1⟩⟨b-literal⟩⟨b-term-aux⟩
⟨b-literal⟩ ⊨ ⟨b-base⟩ | ⟨b-un-op⟩⟨b-base⟩
⟨b-base⟩ ⊨ (⟨b-expr⟩) | true | false | ⟨rel-expr⟩ | ⟨ident⟩ | ⟨subprog-call⟩
⟨b-un-op⟩ ⊨ not
⟨b-bin-op-p0⟩ ⊨ or | xor
⟨b-bin-op-p1⟩ ⊨ and

4.4 Expressions with numbers, lists and strings

⟨expr⟩ ⊨ ⟨term⟩ | ⟨term⟩⟨expr-aux⟩
⟨expr-aux⟩ ⊨ ⟨bin-op-p0⟩⟨term⟩ | ⟨bin-op-p0⟩⟨term⟩⟨expr-aux⟩
⟨term⟩ ⊨ ⟨factor⟩ | ⟨factor⟩⟨term-aux⟩
⟨term-aux⟩ ⊨ ⟨bin-op-p1⟩⟨factor⟩ | ⟨bin-op-p1⟩⟨factor⟩⟨term-aux⟩
⟨factor⟩ ⊨ ⟨literal⟩^⟨factor⟩ | ⟨literal⟩
⟨literal⟩ ⊨ ⟨basis⟩ | +⟨basis⟩ | -⟨basis⟩
⟨basis⟩ ⊨ (⟨expr⟩) | ⟨ident⟩ | ⟨int-lit⟩ | ⟨float-lit⟩ | ⟨list-lit⟩ | ⟨subprog-call⟩ | ⟨string-lit⟩
⟨un-op⟩ ⊨ + | -
⟨bin-op-p0⟩ ⊨ + | -
⟨bin-op-p1⟩ ⊨ * | / | % | ++ | **