Algol 60 grammar in BNF

Alunos: Elias Nogueira, Jordan Antonio, Paloma Lacerda e Yanka Ribeiro

<program> ::= <block> | <compound statement>
<block> ::= <unlabelled block> | <label > : <block>
<unlabelled block> ::= <block head> ; <compound tail>
<block head> ::= begin <declaration> | <blockaux> ;
<blockaux> ::= <declaration><blockaux> | <empty>
<compound statement> ::= <unlabelled compound> | <label> : <compound statement>
<unlabelled compound> ::= begin <compound tail>
<compound tail> ::= <statement> end | <statement> ; <compound tail>
<declaration> ::= <type declaration> | <array declaration> | <procedure declaration>
<type declaration> ::= <local or own type> <type list>
<local or own type> ::= <type> | own <type>
<type> ::= integer
<type list> ::= <simple variable> | <simple variable>, <type list>
<array declaration> ::= array <array list> | <local or own type> array <array list>
<array list> ::= <array segment><aux array list>
<aux array list> ::= , <array segment><aux array list> | <empty>
<array segment> ::= <array identifier> [<bound pair list>] | <array identifier> , <array segment>
<array identifier> ::= <identifier>
<bound pair list> ::= <bound pair> <auxBoundPair>
<auxBoundPair> ::= , <bound pair> <auxBoundPair> | <empty>
<bound pair> ::= <lower bound> : <lower bound>
<lower bound> ::= <arithmetic expression>
<procedure declaration> ::= procedure <procedure heading> <procedure body> | <type> procedure <procedure heading> <procedure body>
<procedure heading> ::= <procedure identifier> <formal parameter part> ; <value part> <specification part>
<procedure identifier> ::= <identifier>
<formal parameter part> ::= <empty> | (<formal parameter list>)
<formal parameter list> ::= <formal parameter><aux formal parameter list>
<aux formal parameter list> ::= <parameter delimiter> <formal parameter> <aux formal parameter list> | <empty>
<formal parameter> ::= <identifier>
<value part> ::= value <identifier list> ; | <empty>
<specification part> ::= <aux specification part> | <specifier> <identifier list> ; <aux specification part>
<aux specification part> ::= <specifier> <identifier list> <aux specification part> | <empty>
<specifier> ::= <type> | array | <type> array | label | procedure | <type> procedure
<identifier list> ::= <identifier> <aux identifier list>
<aux identifier list> ::= , <identifier> <aux identifier list> | <empty>
<procedure body> ::= <statement> | <code>
<statement> ::= <unconditional statement> | <conditional statement> | <for statement>
<unconditional statement> ::= <basic statement> | <compound statement> | <block>
<basic statement> ::= <unlabelled basic statement> | <label>: <basic statement>

<label> ::= <identifier> | <unsigned integer>
<unlabelled basic statement> ::= <assignment statement> | <go to statement> | <dummy statement> | <procedure statement>
<assignment statement> ::= <simple> | <left part>
<left part> ::= <variable> := | <procedure identifier> :=
<go to statement> ::= goto <designational expression>
<designational expression> ::= <simple designational expression> | <if clause> <simple designational expression> else <designational expression>
<simple designational expression> ::= <label> | (<designational expression>)
<dummy statement> ::= <empty>
<procedure statement> ::= <procedure identifier> <actual parameter part>
<actual parameter part> ::= <empty> | (<actual parameter list>)
<actual parameter list> ::= <actual parameter> | <aux actual parameter list>
<aux actual parameter list> ::= <parameter delimiter> <actual parameter> <aux actual parameter list> | <empty>
<parameter delimiter> ::= ,
<actual parameter> ::= <expression> | <array identifier> | <procedure identifier>
<conditional statement> ::= <if statement> | <if statement> else <statement> | <if clause> <for statement> | <label>: <conditional statement>
<if statement> ::= <if clause> <unconditional statement>
<if clause> ::= if <Boolean expression> then
<for statement> ::= <for clause> <statement> | <label>: <for statement>
<for clause> ::= for <variable> := <for list> do
<for list> ::= <for list element> <aux for list>
<aux for list> ::= , <for list element> <aux for list> | <empty>
<for list element> ::= <arithmetic expression> | <arithmetic expression> step <arithmetic expression> until <arithmetic expression> | <arithmetic expression> while <Boolean expression>
<expression> ::= <arithmetic expression> | <Boolean expression> | <designational expression>
<arithmetic expression> ::= <simple arithmetic expression> | <if clause> <simple arithmetic expression> else <arithmetic expression>
<simple arithmetic expression> ::= <term><aux simple arithmetic expression> | <adding operator> <term><aux simple arithmetic expression>
<aux simple arithmetic expression> ::= <adding operator> <term><aux simple arithmetic expression> | <empty>
<adding operator> ::= + | –
<term> ::= <factor><aux term>
<aux term> ::= <multiplying operator> <factor> <aux term> | <empty>
<multiplying operator> ::= *
<factor> ::= <primary> | <factor>
<primary> ::= <variable> | <function designator> | (<arithmetic expression>)
<unsigned integer> ::= <digit><aux unsigned integer>
<aux unsigned integer> ::= <digit><aux unsigned integer> | <empty>
<exponential part> ::= ^ <integer>
<integer> ::= <unsigned integer> | + <unsigned integer> | – <unsigned integer>
<Boolean expression> ::= <simple Boolean> | <if clause> <simple Boolean> else <Boolean expression>

<simple Boolean> ::= <implication><aux simple Boolean>
<simple Boolean> ::= == <implication><aux simple Boolean> | <empty>
<implication> ::= <Boolean term>
<Boolean term> ::= <Boolean factor><aux Boolean term>
<aux Boolean term> ::= || <Boolean factor> <aux Boolean term> | <empty>
<Boolean factor> ::= <Boolean secondary><aux Boolean factor>
<aux Boolean factor> ::= && <Boolean secondary><aux Boolean factor> | <empty>
<Boolean secondary> ::= <Boolean primary> | ! <Boolean primary>
<Boolean primary> ::= <logical value> | <variable> | <function designator> | <relation> | (
<Boolean expression>)
<relation> ::= <simple arithmetic expression> <relational operator> <simple arithmetic
expression>
<function designator> ::= <procedure identifier> <actual parameter part>
<variable> ::= <simple variable> | <subscripted variable>
<simple variable> ::= <variable identifier>
<variable identifier> ::= <identifier>
<subscripted variable> ::= <array identifier> [<subscript list>]
<subscript list> ::= <subscript expression> <aux subscript list>
<aux subscript list> ::= , <subscript expression><aux subscript list> | <empty>
<subscript expression> ::= <arithmetic expression>
<identifier> ::= <letter><auxIdentifier> |
<auxIdentifier> ::= <letter><auxIdentifier> | <digit> <auxIdentifier> | <empty>
<basic symbol> ::= <letter> | <digit> | <logical value> | <delimiter>
<letter> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<logical value> ::= true | false
<delimiter> ::= <operator> | <separator> | <bracket> | <declarator> | <specificator>
<operator> ::= <arithmetic operator> | <relational operator> | <logical operator> | <sequential
operator>
<arithmetic operator> ::= + | – | * | /
<relational operator> ::= < | <= | = | != | > | >=
<logical operator> ::= == | || | && | !
<sequential operator> ::= goto | if | then | else | for | do
<separator> ::= , | : | ; | := | _ | step | until | while | comment
<bracket> ::= (|) | [|] | begin | end
<declarator> ::= own | integer | array | procedure
<specificator> ::= label | value