Para a implementação do analisador foi necessário remover a recursão à esquerda presente nessas produções:

if-stmt ::= if condition then stmt-list end |

if condition then stmt-list else stmt-list end

expression ::= simple-expr |

simple-expr relop simple-expr

Dessa maneira, obteve-se:

if-stmt ::= if condition then stmt-list if-stmt’

if-stmt’ ::= end | else stmt-list end

expression ::= simple-expr expression’

expression’ ::= λ | relop simple-expr

No qual se obtém a seguinte gramática

program ::= [ **var** decl-list] **begin** stmt-list **end**

decl-list ::= decl ";" { decl ";"}

decl ::= ident-list **is** type

ident-list ::= identifier {"," identifier}

type ::= **int** | **string**

stmt-list ::= stmt ";" { stmt ";"}

stmt ::= assign-stmt | if-stmt | do-stmt

| read-stmt | write-stmt

assign-stmt ::= identifier ":=" simple\_expr

if-stmt ::= **if** condition **then** stmt-list if-stmt’

if-stmt’ ::= **end** | **else** stmt-list **end**

condition ::= expression

do-stmt ::= **do** stmt-list stmt-suffix

stmt-suffix ::= **while** condition

read-stmt ::= **in** "(" identifier ")"

write-stmt ::= **out** "(" writable ")"

writable ::= simple-expr

expression ::= simple-expr expression’

expression’ ::= λ | relop simple-expr

simple-expr ::= term | simple-expr addop term

term ::= factor-a | term mulop factor-a

fator-a ::= factor | **not** factor | "-" factor

factor ::= identifier | constant | "(" expression ")"

relop ::= "=" | ">" | ">=" | "<" | "<=" | "<>"

addop ::= "+" | "-" | **or**

mulop ::= "\*" | "/" | **and**

constant ::= integer\_const | literal

integer\_const ::= nozero {digit} | “0”

literal ::= " {" {caractere} "}"

identifier ::= (letter) {letter | digit } |

“\_” (letter | digit ) {letter | digit }

letter ::= [A-Za-z]

digit ::= [0-9]

nozero ::= [1-9]

caractere ::= um dos 256 caracteres do conjunto ASCII, exceto “{”, “}”

e quebra de linha