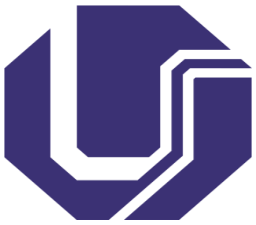


UNIVERSIDADE FEDERAL DE UBERLÂNDIA - CAMPUS SANTA MÔNICA
GRADUAÇÃO CIÊNCIA DA COMPUTAÇÃO

PROJETO COMPILADOR (FRONT-END)

ALBERTO FERREIRA NETO
11811BCC041

UBERLÂNDIA-MG
26/06/2022

1 ETAPA 1 (PROJETO DA LINGUAGEM)

1.1 GRAMÁTICA (GLC)

A1 = ({expr, term}, {ID, NUMERO, OP_RELAC}, P, expr)

P = {

expr → expr OP_RELAC term | term,

term → ID | NUM | expr

}

A2 = ({expr, term}, {ID, NUM, OP_ARIT}, P, expr)

P = {

expr → expr OP_ARIT term | term,

term → ID | NUM | expr

}

A3 = ({cmd, term, op}, {IF, OP_RELAC, THEN, ELSE, ID, NUM}, P, cmd)

P = {

cmd → IF (op)THEN cmd ELSE cmd,

op → term OP_RELAC term

term → ID | NUM

}

A4 = ({cmd, op, term}, {WHILE, DO, OP_RELAC, ID, NUM} P, cmd)

P = {

cmd → WHILE (op) DO cmd,

op → term OP_RELAC term,

term → ID | NUM

}

A5 = ({cmd, whileDo, do;while, op, term}, {WHILE, DO, REPEAT, OP_RELAC, ID, NUM} P, cmd)

P = {

cmd → whileDo | doWhile,

whileDo → REPEAT cmd WHILE (op)

doWhile → WHILE (op) DO cmd

op → term OP_RELAC term,

term → ID | NUM

}

A6 = ({cmd}, {BEGIN, END} P, cmd)

P = {

cmd → BEGIN cmd END

}

A7 = ({prog, bloco, cmd, variável, lista_ids, tipo, loop, whiledo, dowhile, ifelse, operacaoarit, operacaorel, term, atribuição, expressão, const}, {PROGRAMA, ID, BEGIN, END, INT, FLOAT, CHAR, NUMERO, LETRA, REPEAT, WHILE, DO, IF, THEN, ELSE, OP_RELAC, OP_ARIT, ;, (,), :=} P, prog)

P = {

prog → PROGRAMA ID bloco,

bloco → BEGIN cmd END,

cmd → variavel| loop | ifelse | operacaoarit | atribuicao,

atribuição → ID := expressao ;,

expressão → const | operacaoarit,

variavel → tipo : lista_ids ;,

lista_ids → ID | ID, lista_ids ;,

tipo → INT | FLOAT | CHAR

loop → whiledo | dowhile,

whiledo → REPEAT bloco WHILE (operacaorel),

dowhile → WHILE (operacaorel) DO bloco,

ifelse → IF (operacaorel)THEN bloco ELSE bloco,

operacaoarit → term OP_ARIT operacaoarit | term,

operacaorel → term OP_RELAC term,

term → ID | NUMERO

const → NUMERO | LETRA

}

1.2 TOKENS

<PROGRAMA>, <BEGIN>, <END>, <IF>, <THEN>, <ELSE>, <WHILE>, <DO>, <REPEAT>, <INT>, <CHAR>, <FLOAT>, <NUMERO>, <LETRA>, <OP_ARIT>, <OP_RELAC>, <ID>, <COMENTARIO>, <,>, <;>, <(>, <)>, <[>, <]>, <:=>

OP_RELAC = {=, ~=, <, >, <=, >=};

OP_ARIT = {+, - /, *};

1.3 DEFINIÇÕES REGULARES

digito → [0 - 9]

digitos → digito*

letra → [A - Za - z]

numero → digitos (.digitos) ? (E[+ -] ? digitos) ?

id → letra (letra | digito)*

op_relac → < | > | <= | >= | = | <>

op_arit → + | - | / | *

if → if

then → then

else → else

while → while

do → do

repeat → repeat

int → int

char → char

float → float

begin → begin

end → end

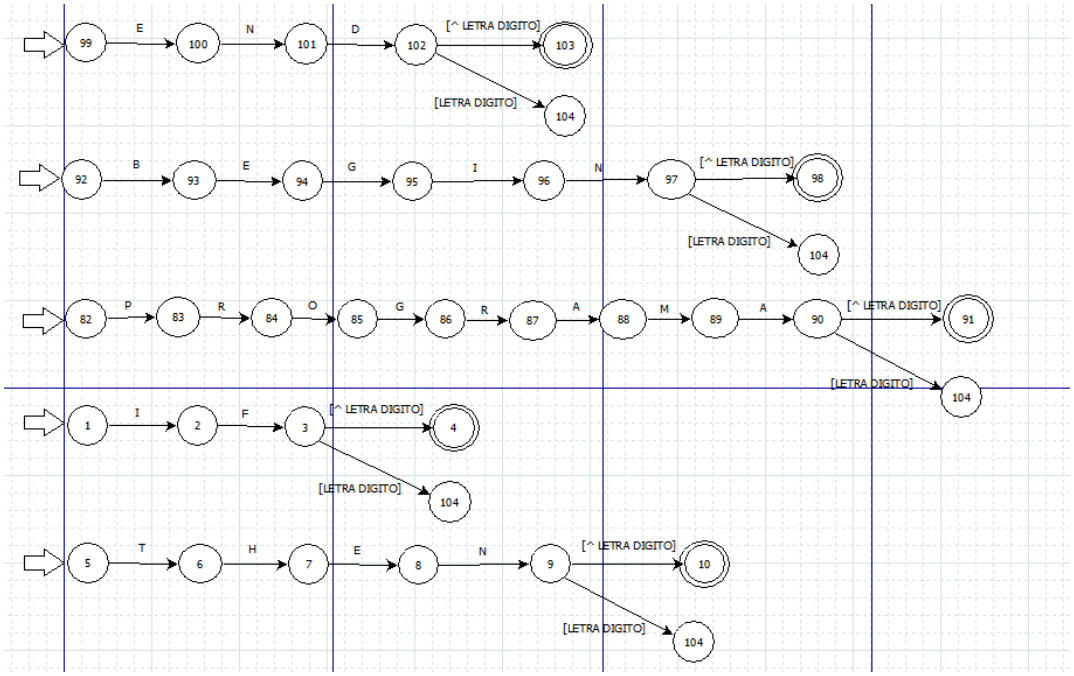
programa → programa

comentário → \[\]

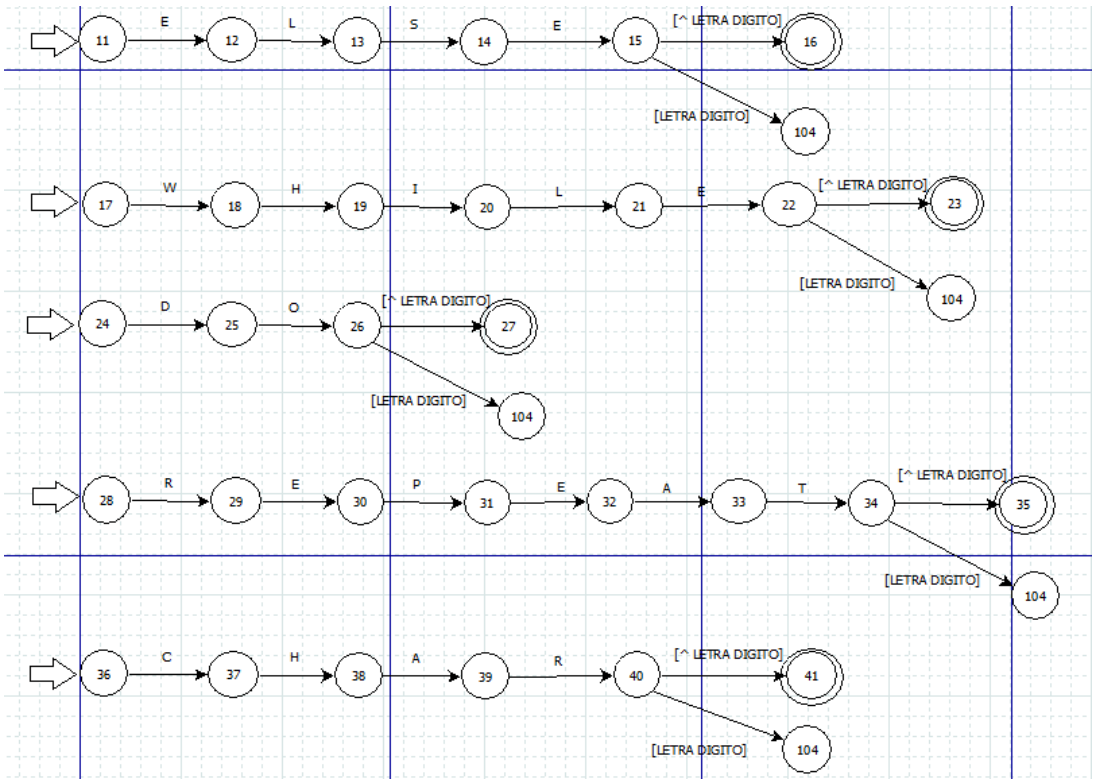
2 ETAPA 2 (ANÁLISE LÉXICA)

2.1 DIAGRAMA DE TRANSIÇÃO

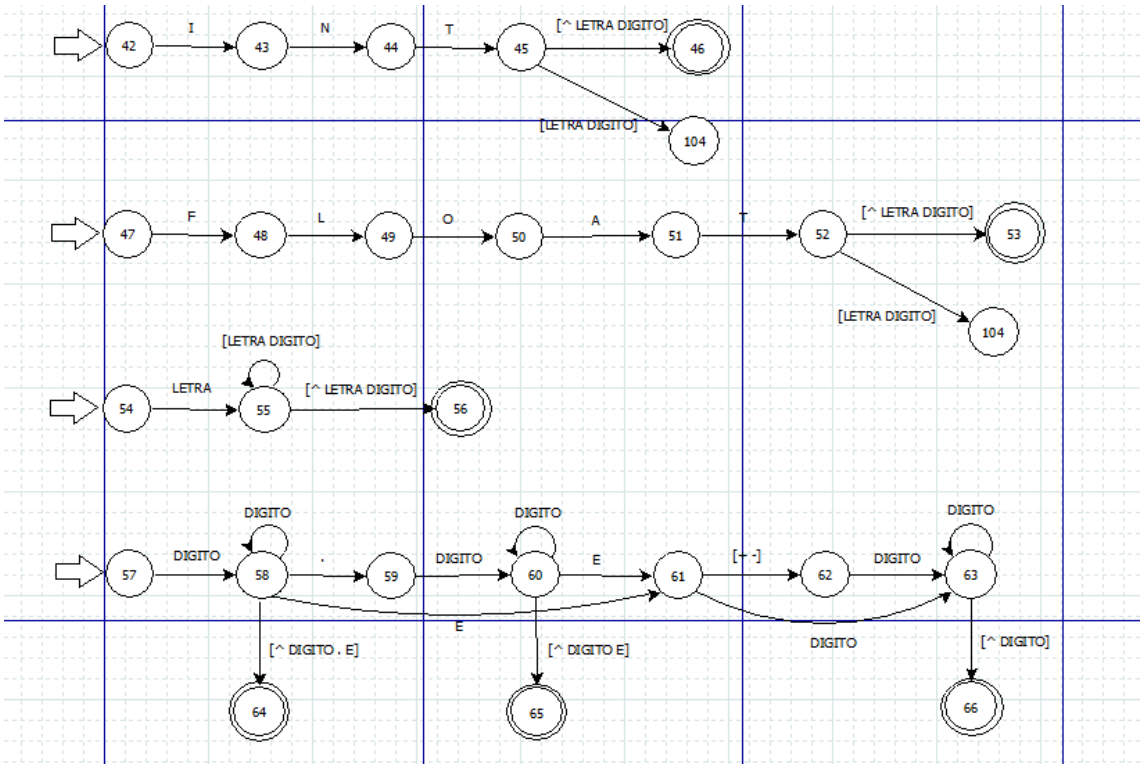
TOKENS: (<END>, <BEGIN>, <PROGRAMA>, <IF>, <THEN>);



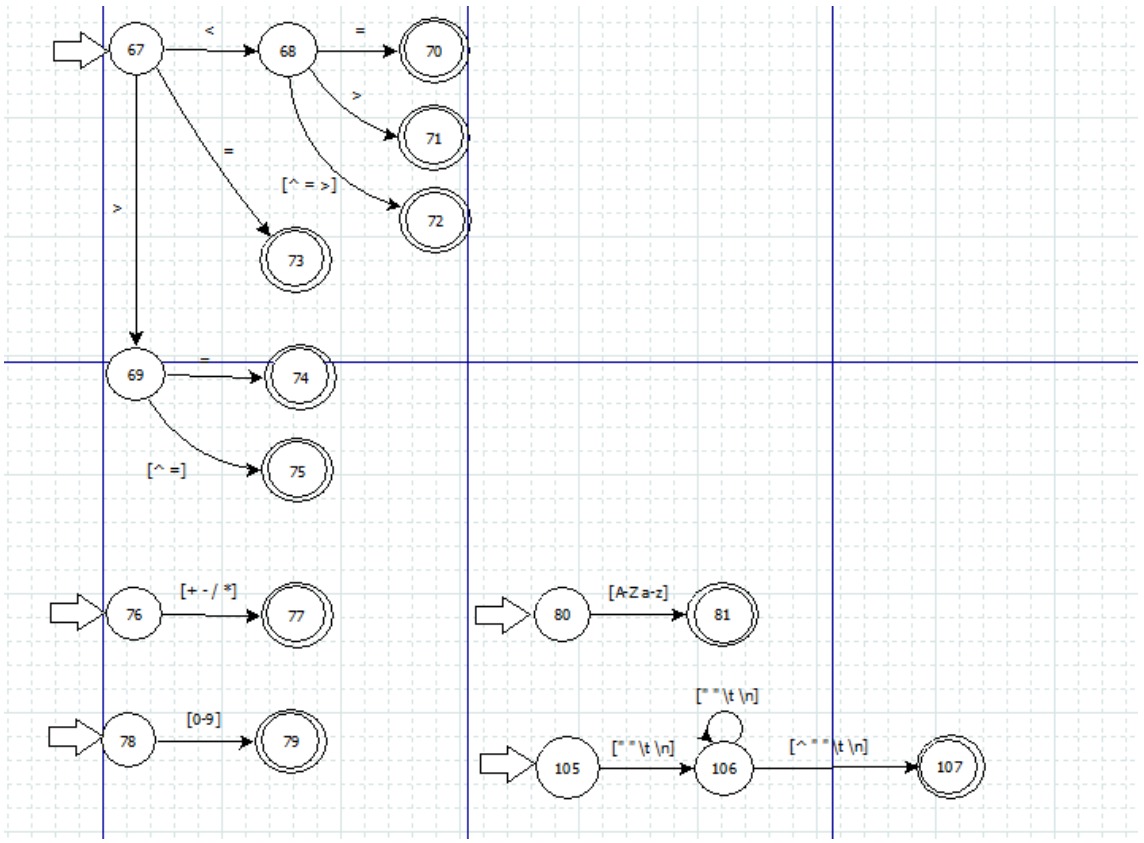
TOKENS: (<ELSE>, <WHILE>, <DO>, <REPEAT >, <CHAR >);



TOKENS: (<INT>, <FLOAT>, <ID>, <NUMERO>);

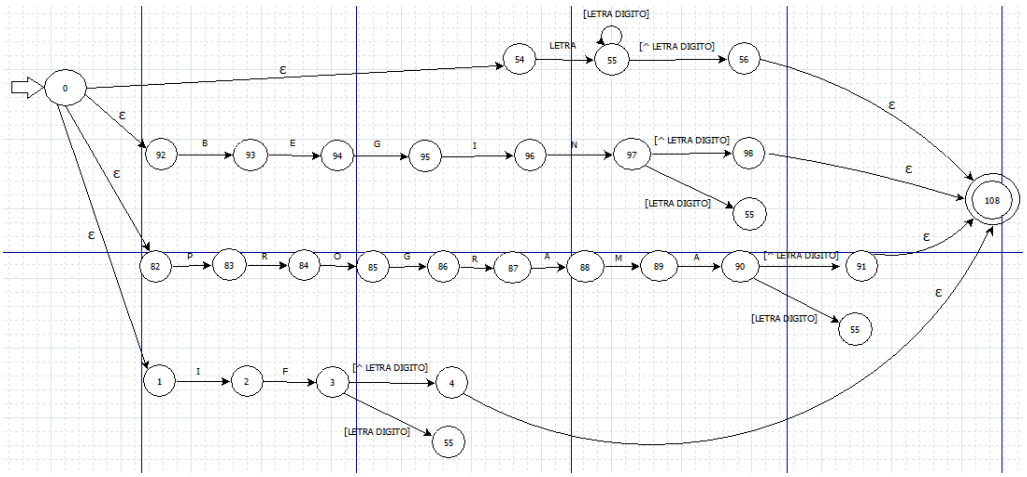


TOKENS: (<OP_RELAC>, <OP_ARIT >, <DIGITO>, <LETRA>);

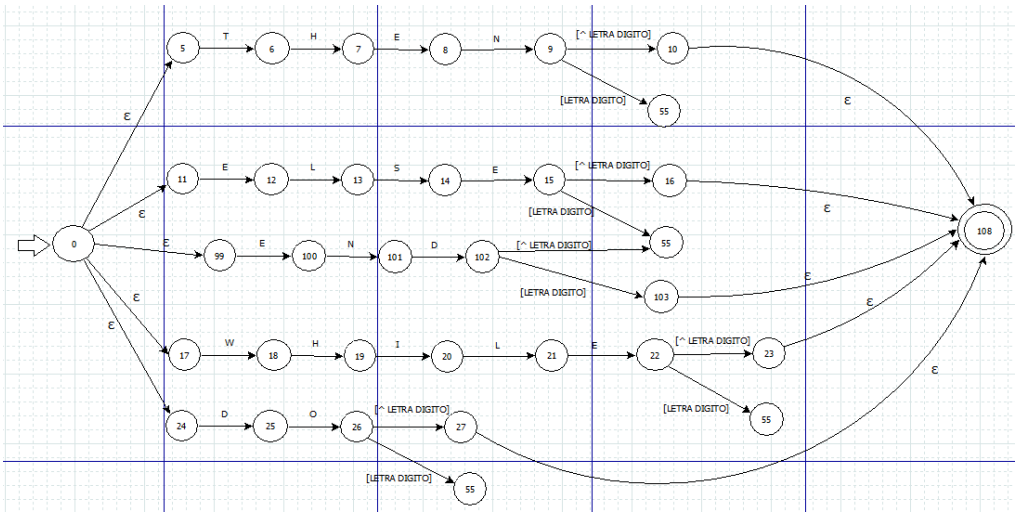


2.2 DIAGRAMA DE TRANSIÇÃO NÃO DETERMINISTICO

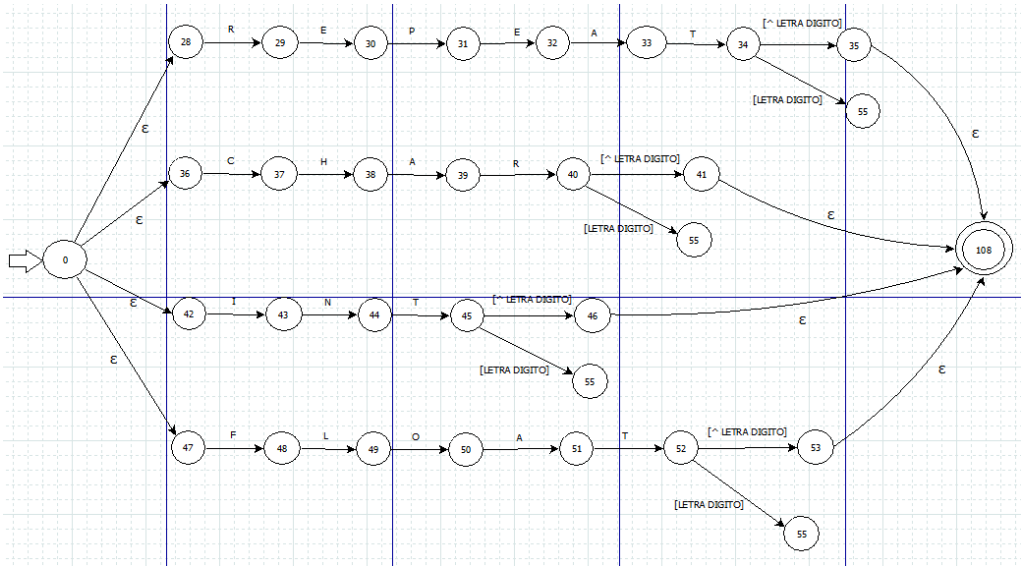
PARTE 1



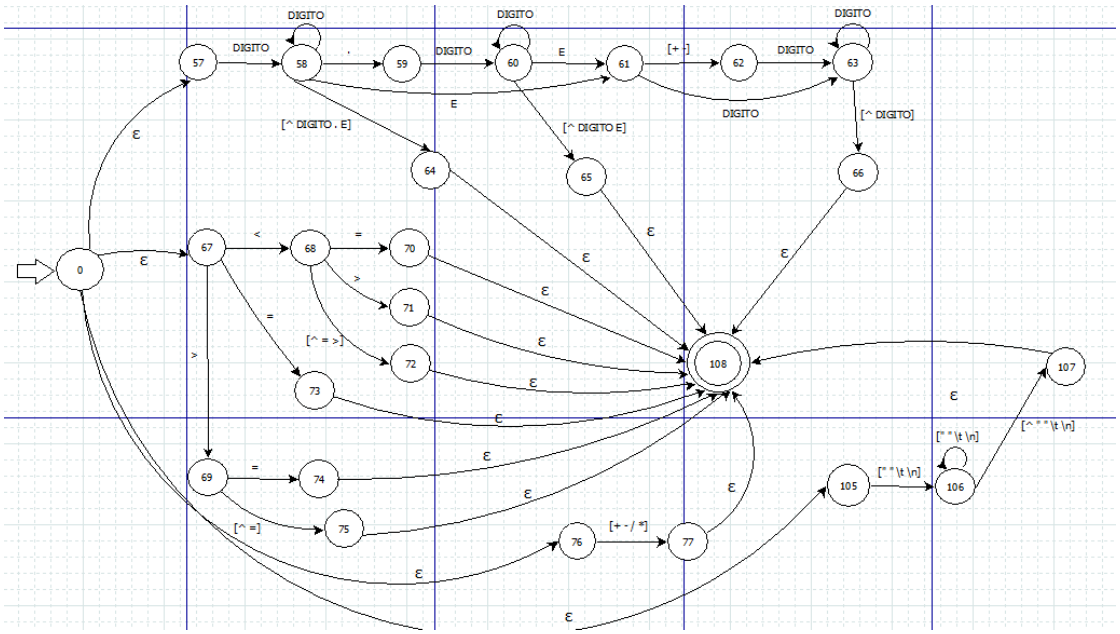
PARTE 2



PARTE 3

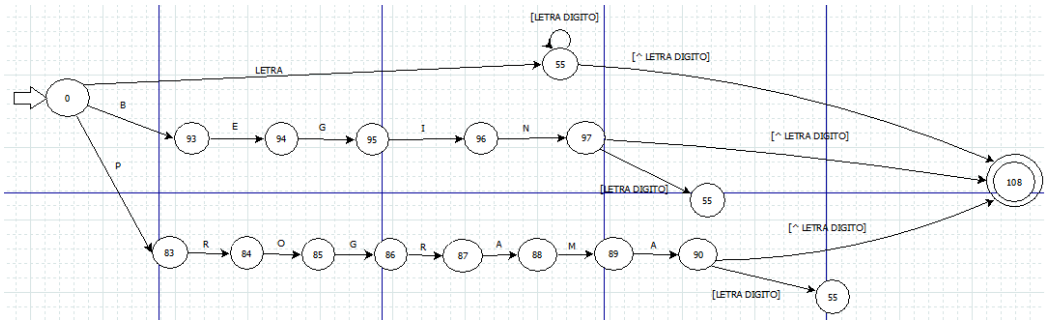


PARTE 4

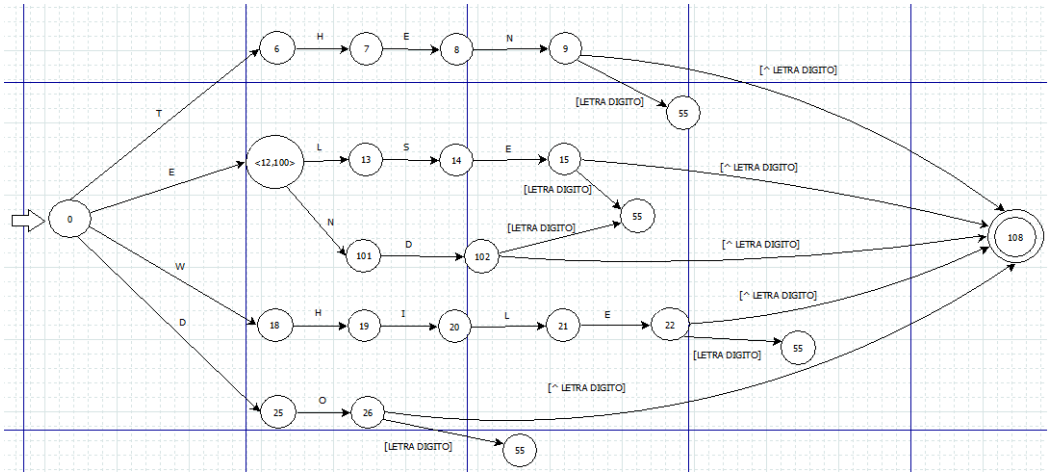


2.3 DIAGRAMA DE TRANSIÇÃO DETERMINISTICO

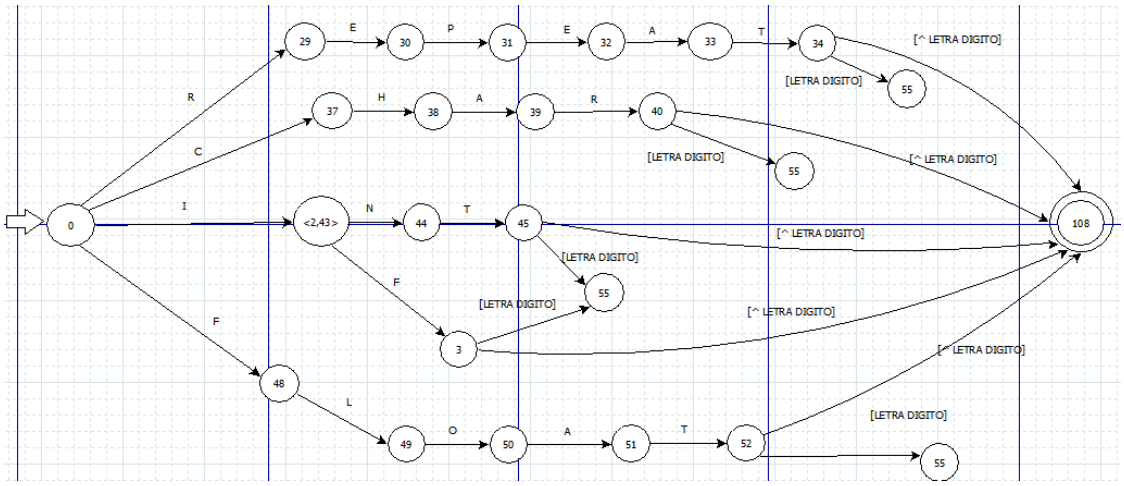
PARTE 1



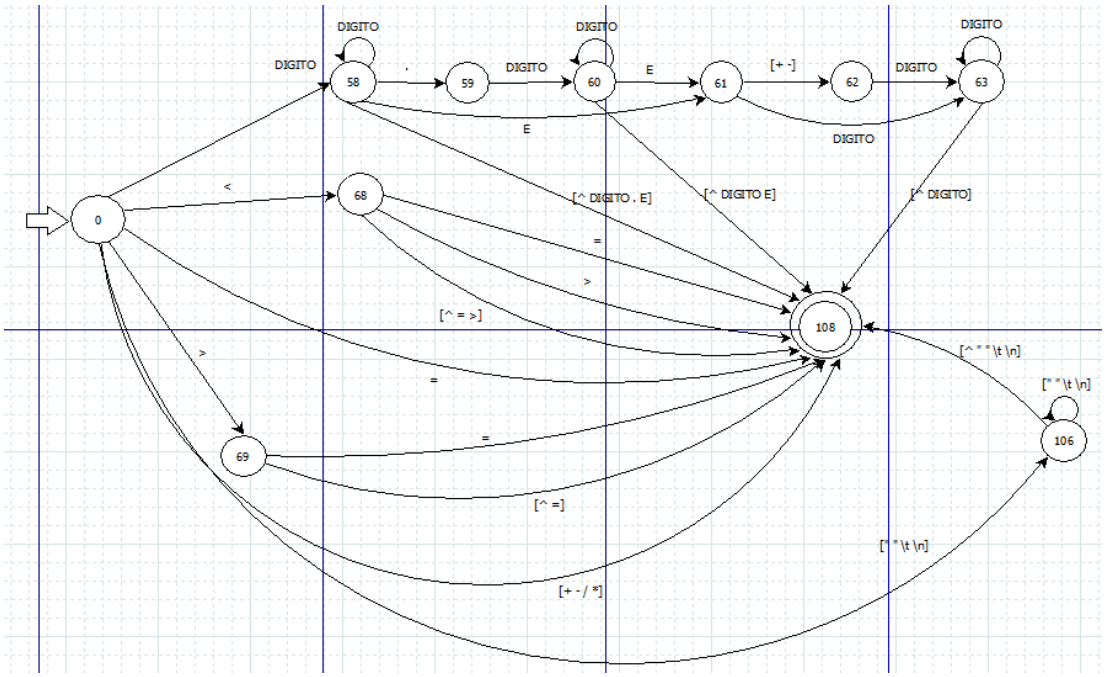
PARTE 2



PARTE 3



PARTE 4



2.4 ANALISADOR LÉXICO

Arquivo de entrada:

```
PROGRAMA TESTE
    BEGIN
        INT x = 0
        IF 4>=5 THEN
            BEGIN
                x = 4+2
            END
    END
```

Tokens retornados pelo analisador léxico:

```
TOKEN <0,1,PROGRAMA>
TOKEN <1,10,TESTE>
TOKEN <3,21,BEGIN>
TOKEN <4,36,INT>
TOKEN <4,40,x>
TOKEN <4,42,=>
TOKEN <5,44,0>
TOKEN <6,55,IF>
TOKEN <6,58,4>
TOKEN <6,59,>=>
TOKEN <6,61,5>
TOKEN <7,63,THEN>
TOKEN <9,81,BEGIN>
TOKEN <10,104,x>
TOKEN <10,106,=>
TOKEN <10,108,4>
TOKEN <10,109,+>
TOKEN <11,110,2>
TOKEN <13,125,END>
TOKEN <14,134,END>
```

3 ETAPA 3 (ANÁLISE SINTÁTICA)

3.1 GRAMÁTICA LL(1)

A1 = ({expr, expr', term}, {ID, NUMERO, OP_RELAC}, P, expr)

P = {

expr → term expr'

expr' → OP_RELAC term expr' | ε

term → ID | NUM | expr

}

A2 = ({expr, expr', term, term', fator}, {ID, NUM, +, -, /, *}, P, expr)

P = {

expr → term expr'

expr' → + term expr' | - term expr' | ε

term → fator term'

term' → * fator term' | / fator term' |

fator → ID | NUM | (expr)

}

A3 = ({ifelse, ifelse', cmd, term, op}, {IF, OP_RELAC, THEN, ELSE, ID, NUM}, P, cmd)

P = {

Ifelse → IF (op) THEN ifelse ifelse' | cmd

Ifelse' → else ifelse | ε

op → term OP_RELAC term

term → ID | NUM

}

A4 = ({cmd, op, term}, {WHILE, DO, OP_RELAC, ID, NUM} P, cmd)

P = {

cmd → WHILE (op) DO cmd,

op → term OP_RELAC term,

term → ID | NUM

}

A5 = ({cmd, whileDo, do;while, op, term}, {WHILE, DO, REPEAT, OP_RELAC, ID, NUM} P, cmd)

P = {

cmd → whileDo | doWhile,

whileDo → REPEAT cmd WHILE (op)

doWhile → WHILE (op) DO cmd

op → term OP_RELAC term,

term → ID | NUM

}

A6 = ({cmd}, {BEGIN, END} P, cmd)

P = {

cmd → BEGIN cmd END

}

A7 = ({prog, bloco, cmd, variável, lista_ids, tipo, loop, whiledo, dowhile, ifelse, operacaoarit, operacaorel, term, atribuição, expressão, const}, {PROGRAMA, ID, BEGIN, END, INT, FLOAT, CHAR, NUMERO, LETRA, REPEAT, WHILE, DO, IF, THEN, ELSE, OP_RELAC, OP_ARIT, ;, (,), :=} P, prog)

P = {

prog → PROGRAMA ID bloco,

bloco → BEGIN cmd END,

cmd → variável | loop | ifelse | atribuição

atribuição → ID := operacaoarit ;

variavel → tipo : lista_ids ;

lista_ids → ID | ID, lista_ids

tipo → INT | FLOAT | CHAR

loop → whiledo | dowhile

whiledo → REPEAT bloco WHILE (operacaorel)

dowhile → WHILE (operacaorel) DO bloco

Ifelse → IF (operacaorel) THEN bloco ifelse'

Ifelse' → ELSE bloco | ε

operacaorel → fator OP_RELAC fator

operacaoarit → term operacaoarit’

operacaoarit’ → + term operacaoarit’ | - term operacaoarit‘| ε

term → fator term’

term’ → * fator term’ | / fator term’ | ε

fator → ID | NUMERO | (operacaoarit)

}

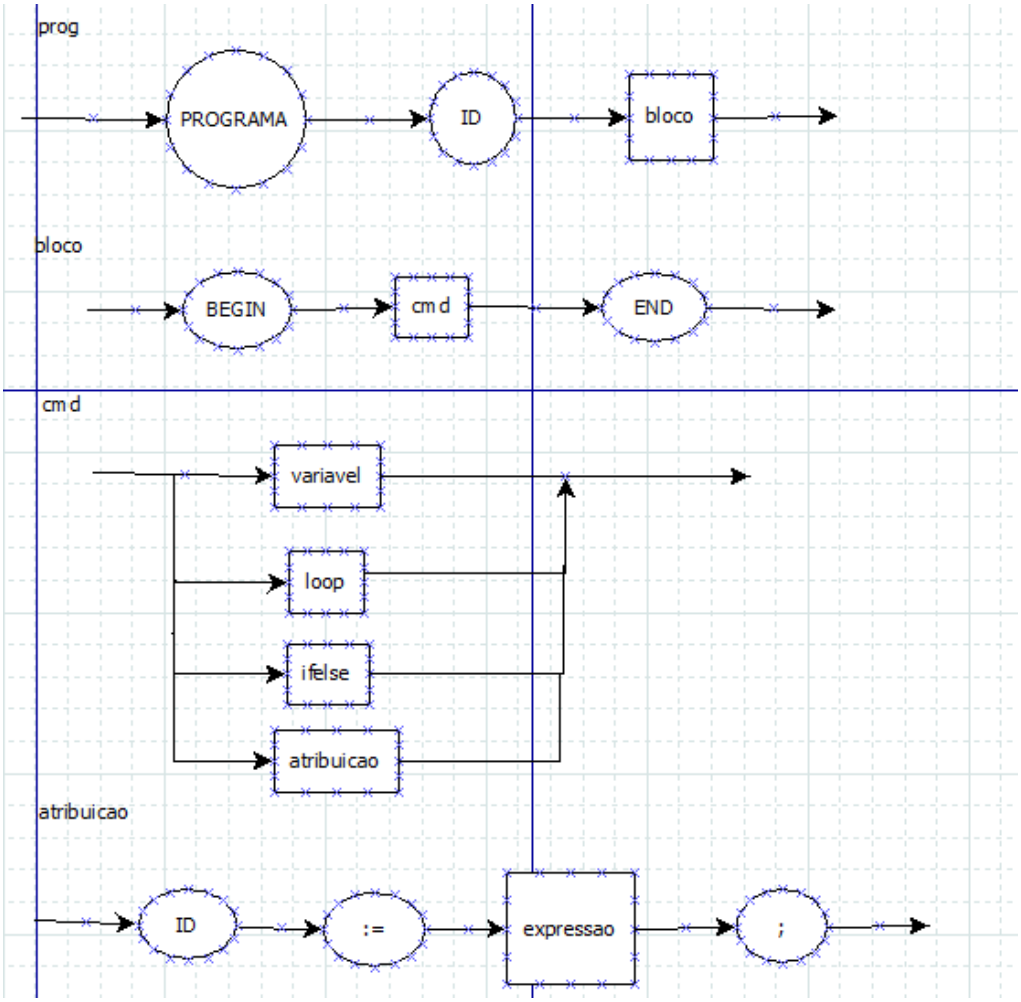
3.2 FIRST E FOLLOW

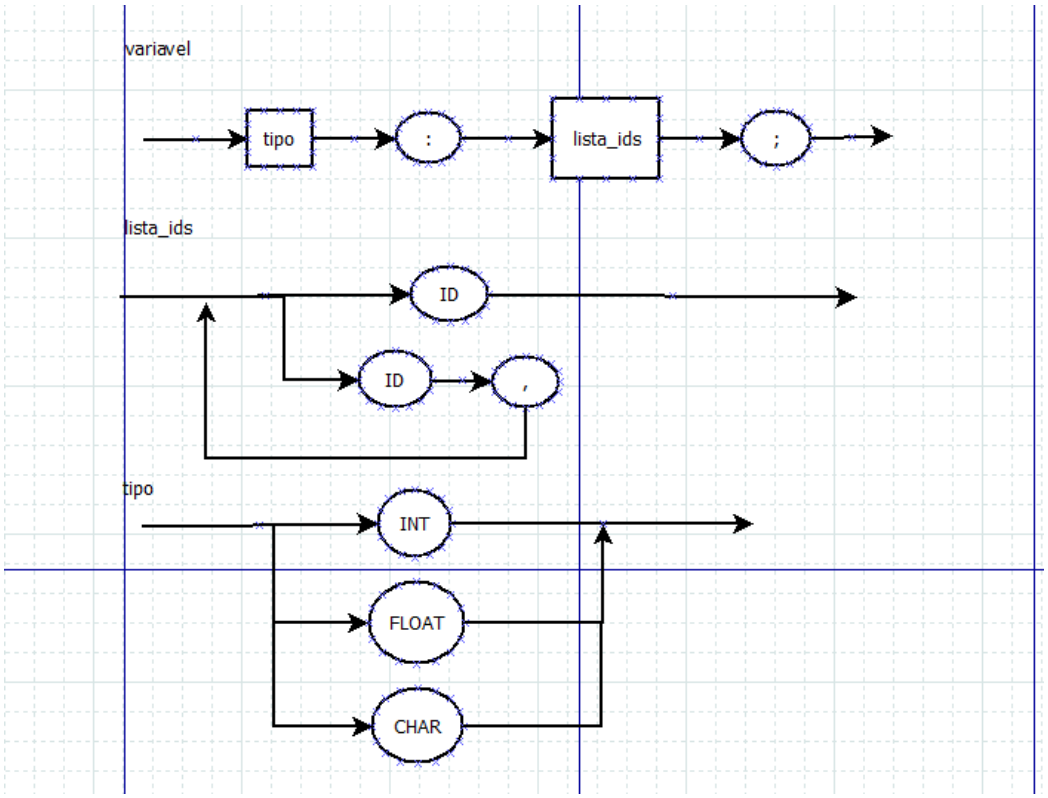
FIRST(fator) = {ID, NUMERO, (}
FIRST(term’) = {*, /, ε }
FIRST(term) = FIRST(fator) = {ID, NUMERO, (}
FIRST(operacaoarit’) = {+, -, ε}
FIRST(operacaoarit) = FIRST(term) = {ID, NUMERO, (}
FIRST(operacaorel) = FIRST(term) = {ID, NUMERO, (}
FIRST(ifelse’) = {ELSE, ε}
FIRST(ifelse) = {IF}
FIRST(dowhile) = {WHILE}
FIRST(whiledo) = {REPEAT}
FIRST(loop) = FIRST(whiledo, dowhile) = {WHILE, REPEAT}
FIRST(tipo) = {INT, FLOAT, CHAR}
FIRST(lista_ids) = {ID}
FIRST(variavel) = FIRST(tipo) = {INT, FLOAT, CHAR}
FIRST(atribuicao) = {ID}
FIRST(cmd) = FIRST(variável, loop, ifelse, atribuicao) = {WHILE, REPEAT, INT, FLOAT, CHAR, IF, ID}
FIRST(bloco) = {BEGIN}
FIRST(prog) = {PROGRAMA}

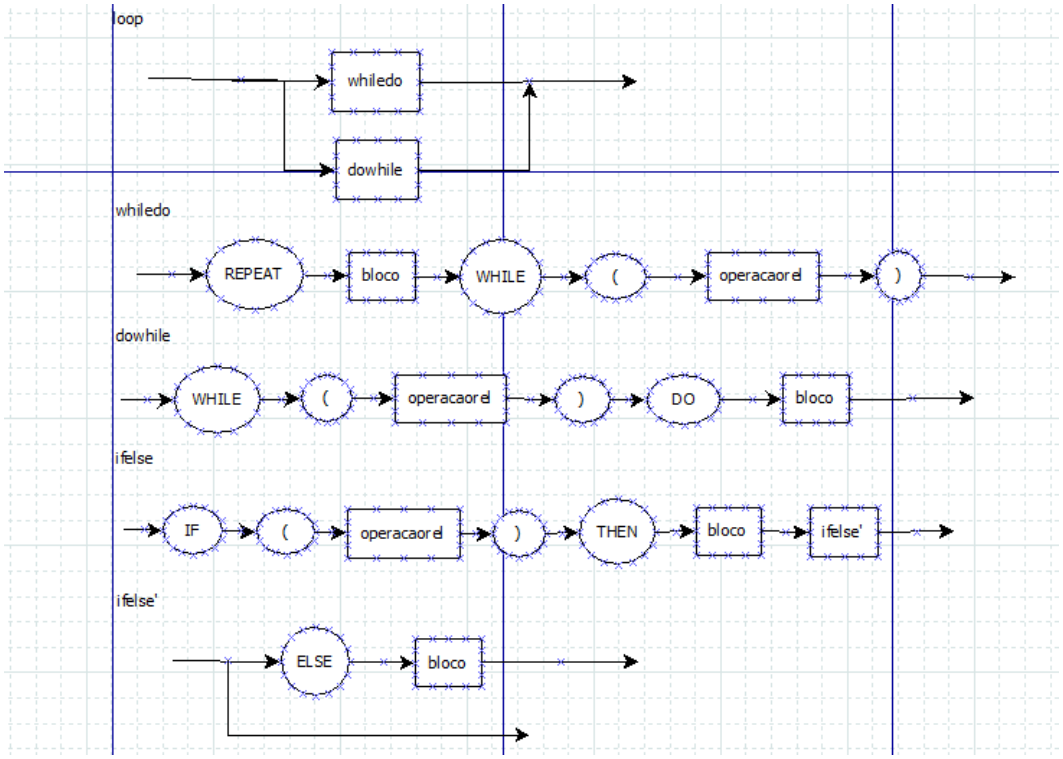
FOLLOW(prog) = {$}
FOLLOW(bloco) = FOLLOW(prog) U FIRST(WHILE) U FOLLOW(dowhile) U FIRST(ifelse’) U FOLLOW(ifelse’) = {}
FOLLOW(cmd) = FIRST(END) = {END}
FOLLOW(atribuicao) = FOLLOW(cmd) = {END}
FOLLOW(variavel) = FOLLOW(cmd) = {END}
FOLLOW(lista_ids) = FIRST(;) = {;}
FOLLOW(tipo) = FIRST(:) = {:}
FOLLOW(loop) = FOLLOW(cmd) = {END}
FOLLOW(whiledo) = FOLLOW(loop) = {END}
FOLLOW(dowhile) = FOLLOW(loop) = {END}
FOLLOW(ifelse) = FOLLOW(cmd) = {END}
FOLLOW(ifelse’) = FOLLOW(ifelse) = {END}
FOLLOW(operacaorel) = FIRST( )) = {) }
FOLLOW(operacaoarit) = FIRST( )) = {) }
FOLLOW(operacaoarit’) = FOLLOW(operacaoarit) = {) }
FOLLOW(term) = FIRST(operacaoarit’) U FOLLOW(operacaoarit’) = {+, -,)}

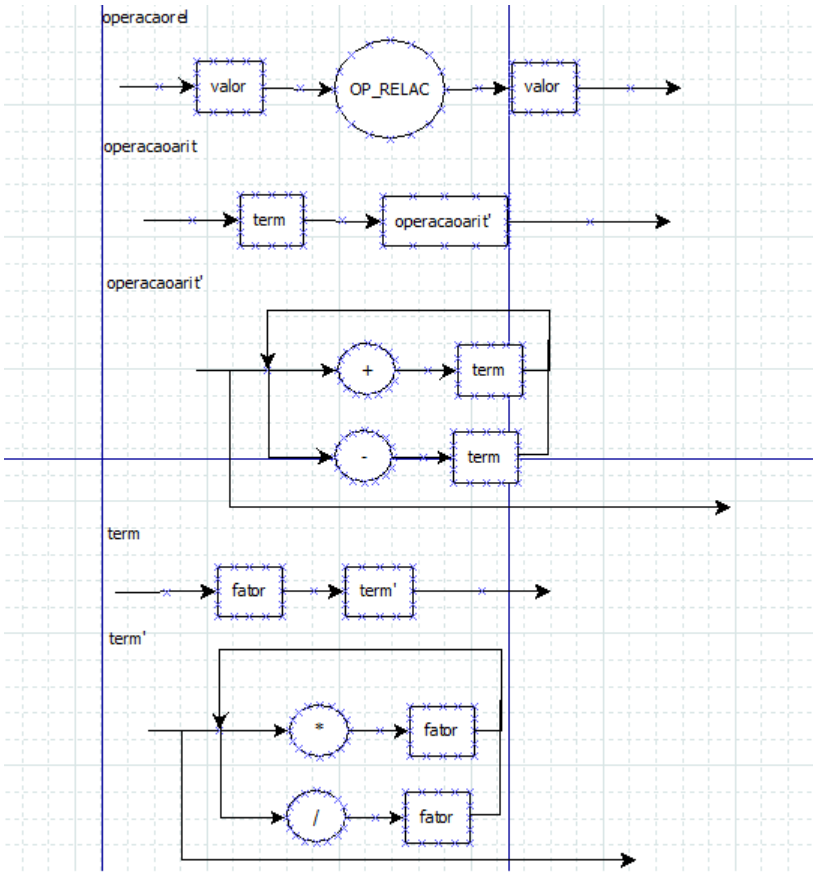
FOLLOW(term') = FOLLOW(term) = {+, -,)}
FOLLOW(fator) = FIRST(term') = {*, /}

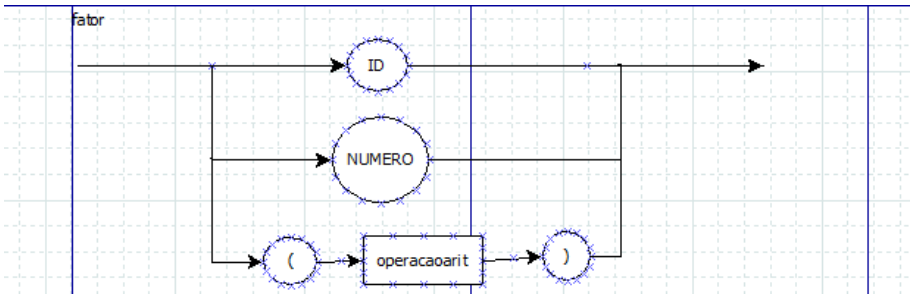
3.3 GRAFOS SINTATICOS




variavel
tipo
:
lista_ids
;
lista_ids
ID
ID
,
tipo
INT
FLOAT
CHAR


loop
whiledo
dowhile
whiledo
REPEAT
bloco
WHILE
(
operacaorel
)
dowhile
WHILE
(
operacaorel
)
DO
bloco
ifelse
IF
(
operacaorel
)
THEN
bloco
ifelse'
ifelse'
ELSE
bloco


operacaorel
valor
OP_RELAC
valor
operacaoarit
term
operacaoarit'
operacaoarit'
+
term
-
term
term
fator
term'
term'
*
fator
/
fator


fator
ID
NUMERO
(
operacaoarit
)