Análise Sintática

Transformando a gramática para o tipo LL

```
S → programa identifier
        block


block → begin
        variable_declaration
        command_sequence
        end


variable_declaration → ε | type :  id_list ; variable_declaration


id_list → identifier  id_list_partial


id_list_partial → , id_list | ε


command_sequence →   ε | command  command_sequence


command  → selection | loop | assignment


selection → if ( condition ) then
                block
            selection'
selection '→ else block | ε


loop → while ( condition ) do
           block
       → repeat
          block
     while(condition)
```

assignment → identifier := exp1;

condition → exp1 relational_operator exp1

exp1 → exp2 exp1'

exp1'→ + exp2 exp1' | - exp2 exp1' | ε

exp2 → exp3 exp2'

exp2' → * exp3 exp2' | / exp3 exp2' | ε

exp3 → term exp3'

exp3' → ^ term exp3' | ε

term → (exp1) | identifier | constant

constant → number | character

Calculando o FISRT e FOLLOW para os símbolos da gramática.

SÍMBOLO	FIRST	FOLLOW
S	{programa}	{$}
block	{begin}	{$, else, while}
variable_declaration	{type, ε}	{if, while, repeat, identifier, end}
id_list	{identifier}	{;}
id_list_partial	{ε , }	{;}
command_sequence	{if, while, repeat, identifier, ε}	{end}
command	{if, while, repeat, identifier}	{if, while, repeat, identifier, end}
selection	{if}	{if, while, repeat, identifier, end}
selection '	{else}	{if, while, repeat, identifier, end}
loop	{while, repeat}	{if, while, repeat, identifier, end}
assignment	{identifier}	{if, while, repeat, identifier, end}
condition	{(, identifier, number, character}	{)}
exp1	{(, identifier, number, character}	{relational_operator,)}
exp1'	{+, -, ε}	{relational_operator,), +}
exp2	{(, identifier, number, character}	{+, -, (, identifier, number, character}
exp2'	{*, /, ε}	{+, -, (, identifier, number, character}
exp3	{(, identifier, number, character}	{*, /, $}
exp3'	{^, ε}	{*, /, $}
term	{(, identifier, number, character}	{^, (, identifier, number, character}
constant	{number, character}	{$}

Construção dos grafos sintáticos.

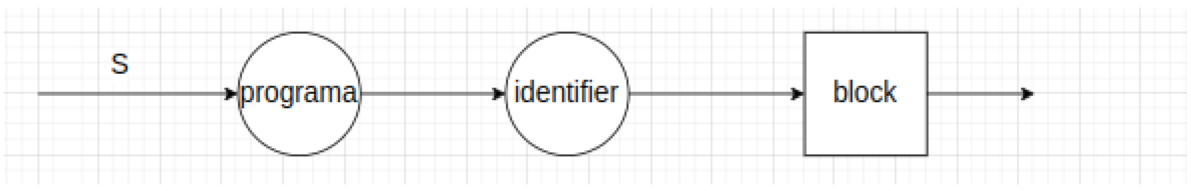

Figura 23: Grafo do procedimento S

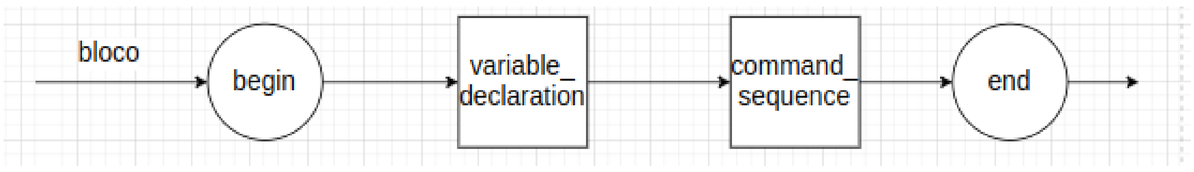

Figura 24: Grafo do procedimento bloco

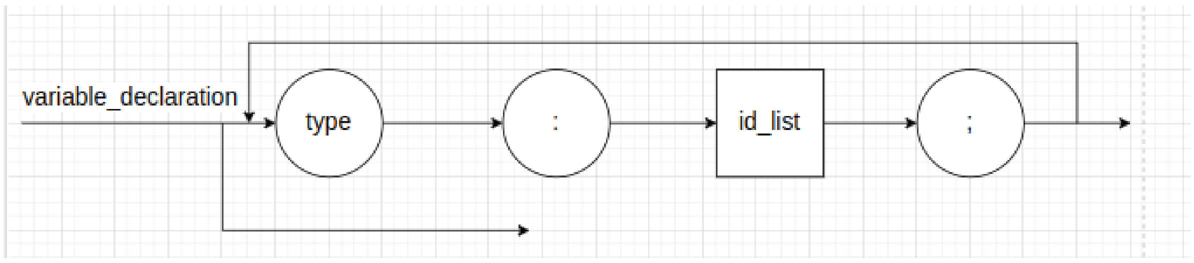

Figura 25: Grafo do procedimento Declaração de variáveis

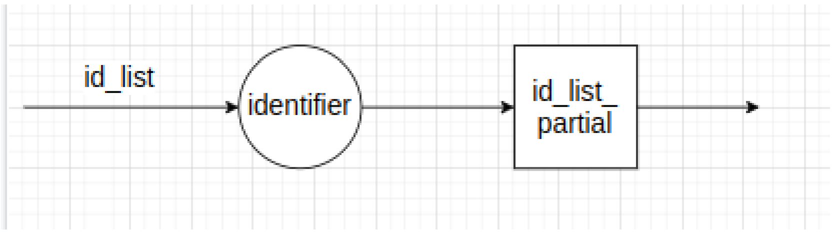

Figura 26: Grafo do procedimento id_list

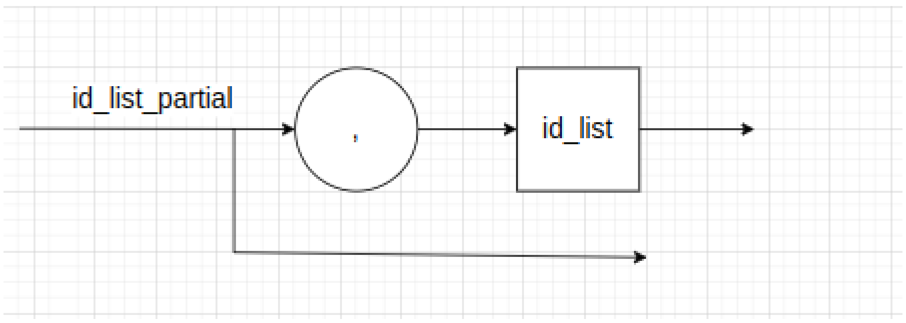

Figura 27: Grafo do procedimento id_list parcial

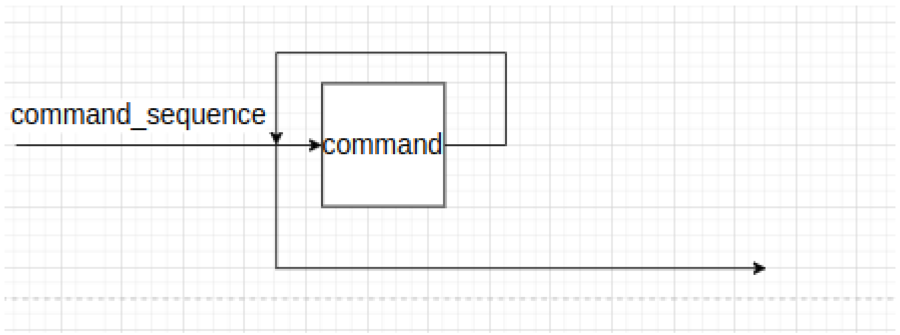

Figura 28: Grafo do procedimento Sequencia de comandos

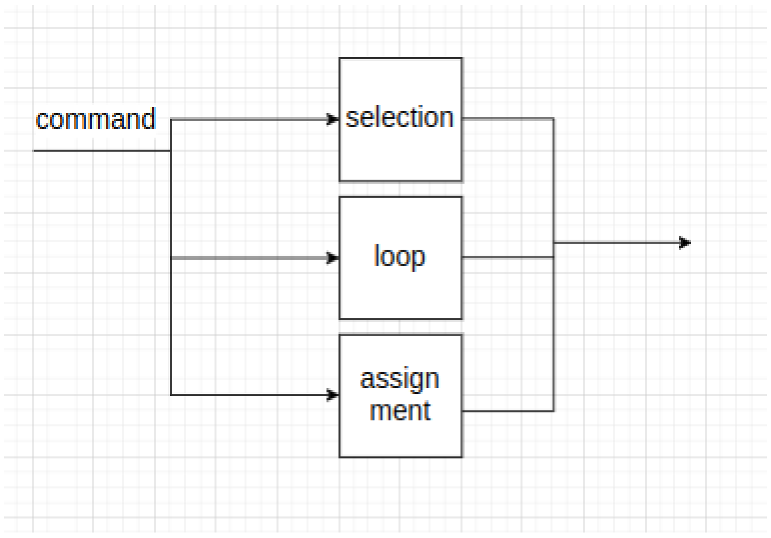

Figura 29: Grafo do procedimento comando

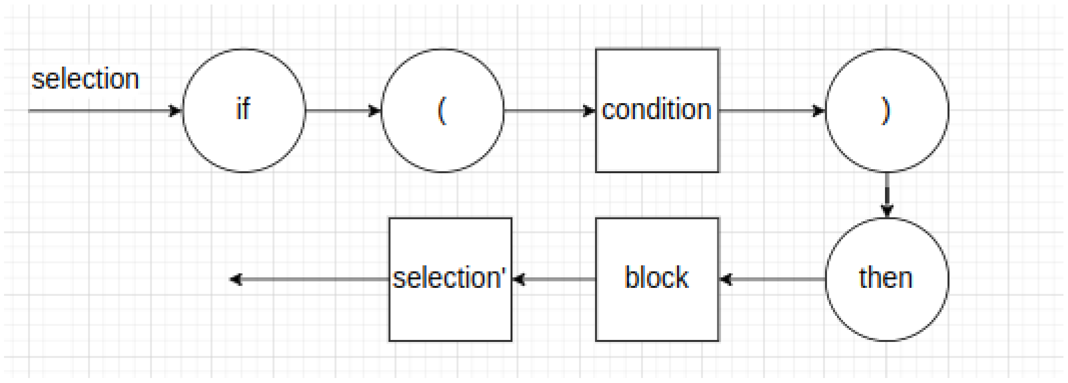

Figura 30: Grafo do procedimento seleção

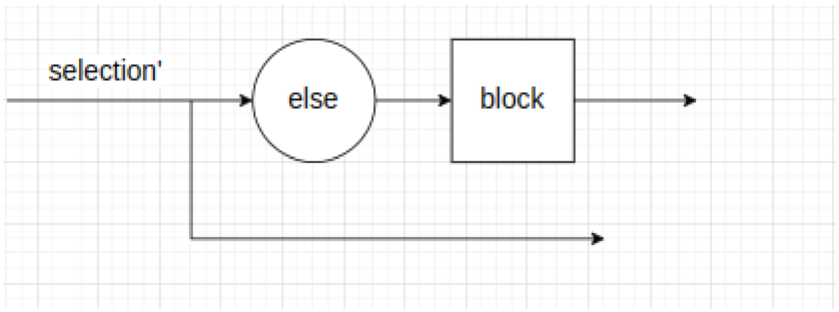

Figura 31: Grafo do procedimento seleção'

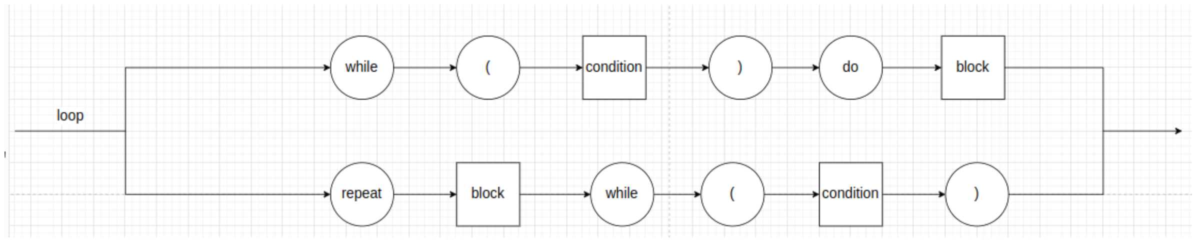

Figura 32: Grafo do procedimento repetição

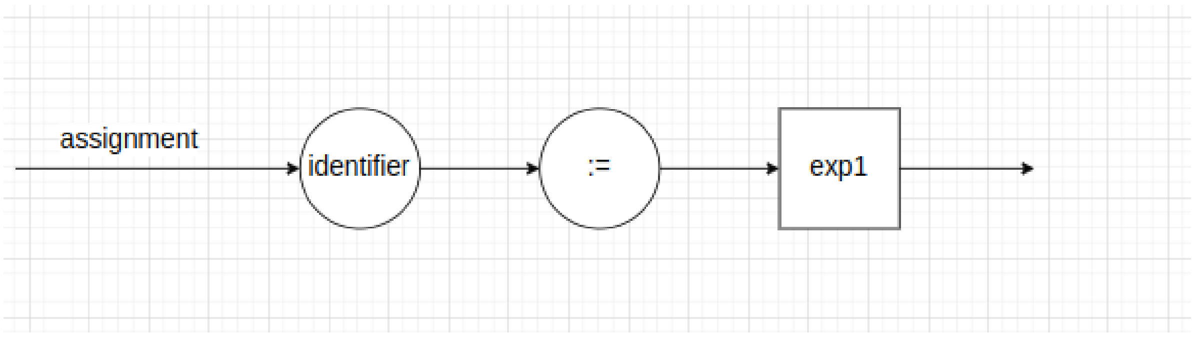

Figura 33: Grafo do procedimento atribuição

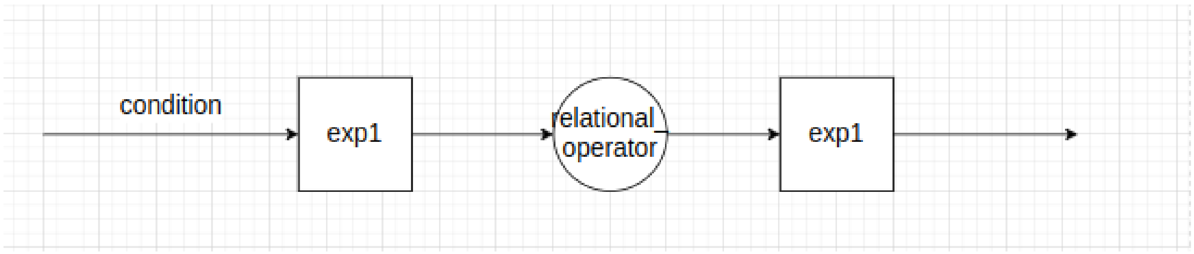

Figura 34: Grafo do procedimento condição

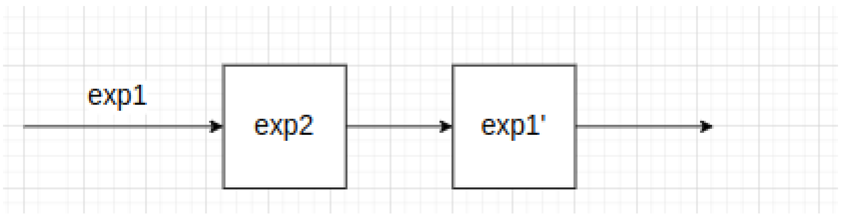

Figura 35: Grafo do procedimento exp1

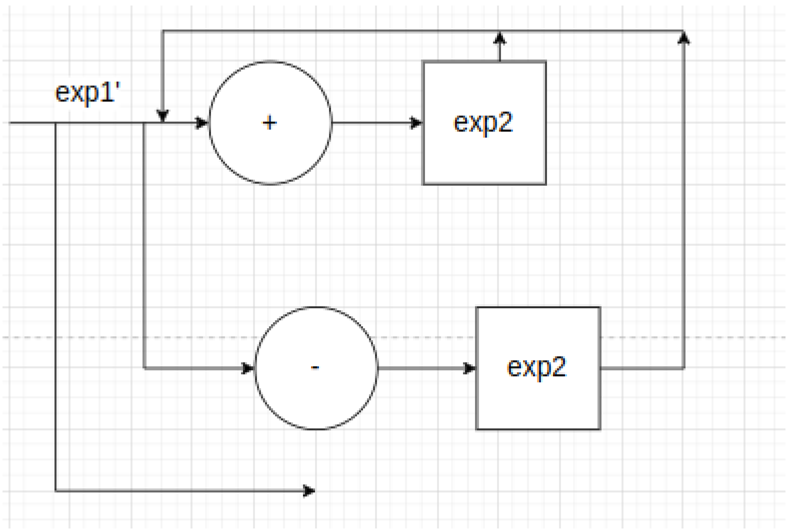

Figura 36: Grafo do procedimento exp1'

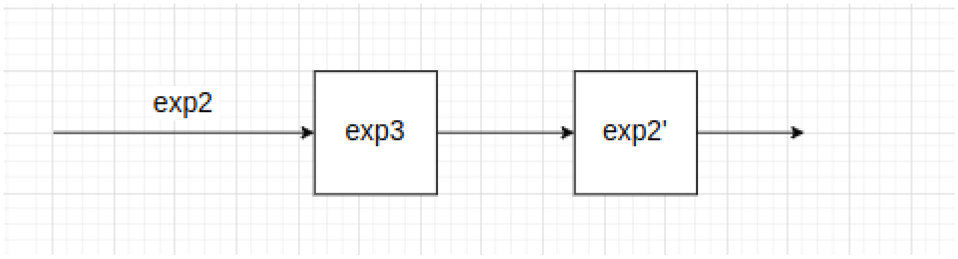

Figura 37: Grafo do procedimento exp2

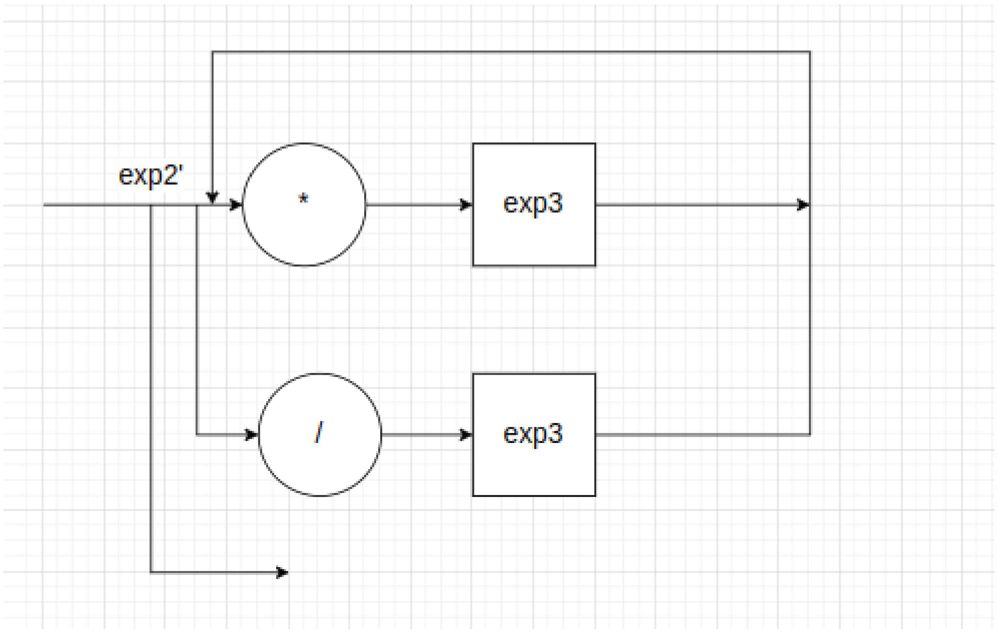

Figura 38: Grafo do procedimento exp2'

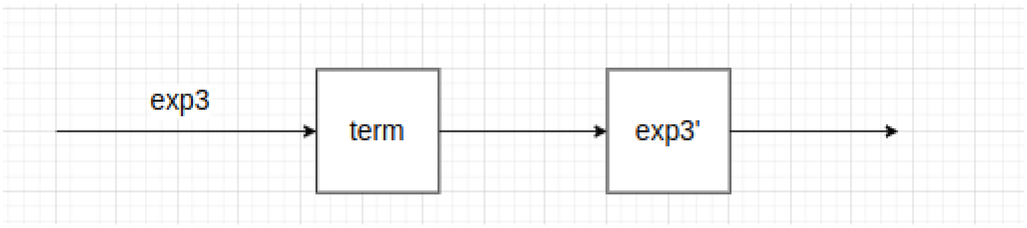

Figura 39: Grafo do procedimento exp3

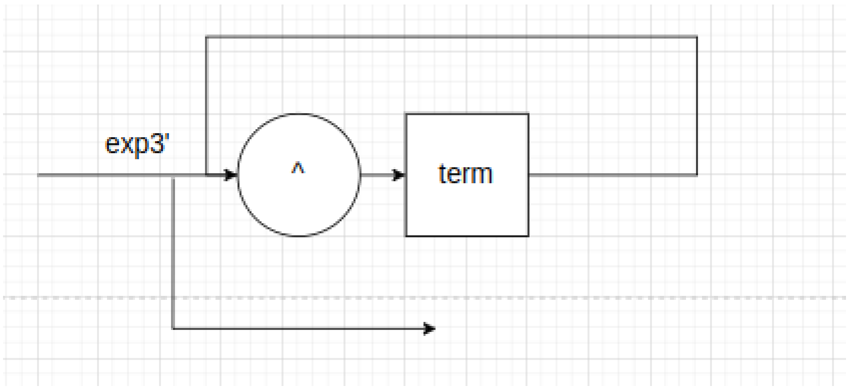

Figura 40: Grafo do procedimento exp3'

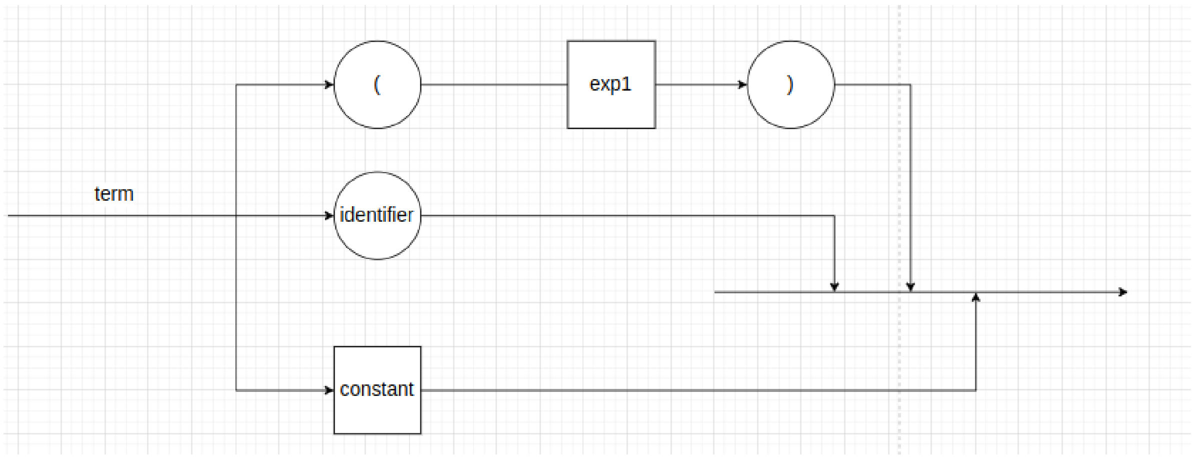

Figura 41: Grafo do procedimento termo

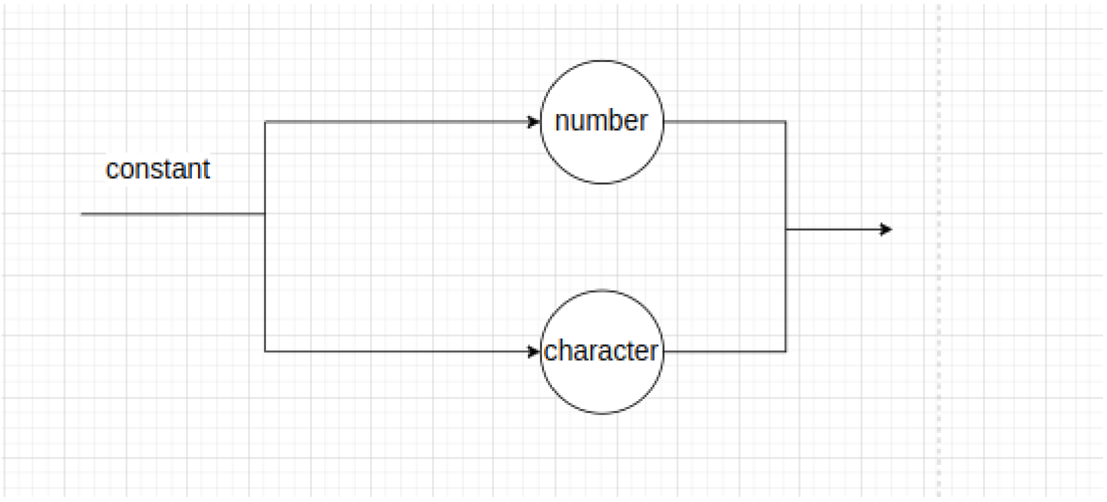

Figura 42: Grafo do procedimento constante

Tendo o analisador sintático implementado, usaremos o seguinte código para teste:

```
programa meuprograma
    begin
        int:a,b;
        [Esse é um simples programa que verifica se x>y e conta]
        int:x;
        int:y;
        float:cont;
        x:=50;
        y:=20;
        if(x>y) then
            begin
                cont:=cont+50;
            end
        else
            begin
                cont:= cont-1;
            end
    end
```

A partir desse código o analisador sintático nos entrega a seguinte árvore de derivação

```
S
    PROGRAMA
    IDENTIFIER
    BLOCK
    BEGIN
        VARIABLE_DECLARATION
            TYPE
            TWO_POINTS
            ID_LIST
                IDENTIFIER
                ID_LIST_PARTIAL
                    COMMA
                    ID_LIST
                        IDENTIFIER
                        ID_LIST_PARTIAL
                            ε
            SEMICOLON
            TYPE
            TWO_POINTS
            ID_LIST
                IDENTIFIER
                ID_LIST_PARTIAL
                    ε
            SEMICOLON
            TYPE
            TWO_POINTS
            ID_LIST
                IDENTIFIER
                ID_LIST_PARTIAL
                    ε
            SEMICOLON
            TYPE
            TWO_POINTS
            ID_LIST
                IDENTIFIER
                ID_LIST_PARTIAL
                    ε
            SEMICOLON
        COMMAND_SEQUENCE
            ASSIGNMENT
                IDENTIFIER
                ASSIGN
                EXP1
                    EXP2
                        EXP3
                            TERM
                                CONSTANT
                                    INT
                            EXP3'
                                ε
                        EXP2'
                            ε
                    EXP1'
                        ε
                SEMICOLON
            ASSIGNMENT
                IDENTIFIER
                ASSIGN
                EXP1
                    EXP2
                        EXP3
                            TERM
                                CONSTANT
                                    INT
                            EXP3'
                                ε
                        EXP2'
                            ε
                    EXP1'
                        ε
                SEMICOLON
            SELECTION
                IF
                    OPEN_PARENTHESES
                    CONDITION
                        EXP1
                            EXP2
                                EXP3
                                    TERM
                                        IDENTIFIER
                                    EXP3'
                                        ε
                                EXP2'
                                    ε
                            EXP1'
                                ε
                        RELOP
                        EXP1
                            EXP2
                                EXP3
                                    TERM
                                        IDENTIFIER
                                    EXP3'
                                        ε
                                EXP2'
                                    ε
                            EXP1'
                                ε
                    CLOSE_PARENTHESES
                    THEN
                        BLOCK
                        BEGIN
                            VARIABLE_DECLARATION
                            COMMAND_SEQUENCE
                                ASSIGNMENT
                                    IDENTIFIER
                                    ASSIGN
                                    EXP1
                                        EXP2
                                            EXP3
                                                TERM
                                                    IDENTIFIER
                                                EXP3'
                                                ε
                                        EXP2'
                                            ε
                                    EXP1'
                                        ARITHEMETIC_OP
                                        EXP2
                                            EXP3
                                                TERM
                                                    CONSTANT
                                                EXP3'
                                                    ε
                                            EXP2'
                                                ε
                    END
            ELSE
                BLOCK
                BEGIN
                    VARIABLE_DECLARATION
                    COMMAND_SEQUENCE
                        ASSIGNMENT
                            IDENTIFIER
                            ASSIGN
                            EXP1
                                EXP2
                                    EXP3
                                        TERM
                                            IDENTIFIER
                                        EXP3'
                                            ε
                                    EXP2'
                                        ε
                                EXP1'
                                    ARITHEMETIC_OP
                                    EXP2
                                        EXP3
                                            TERM
                                                CONSTANT
                                            EXP3'
                                                ε
                                        EXP2'
                                            ε
                END
END
```