Projeto de Construção de compiladores - Parte 1

Arthur Fernandes, Kemuel Santos Peres, Leandro Fontellas Laurito

1. Definição da Gramática Livre de Contexto (GLC)

G = (V, T, P, S)

- **S** = INICIO
- **T** = conjunto dos símbolos terminais
- **V** = símbolos não-terminais
- **P** = conjunto de produções

1.2 Símbolos Terminais (T)

void

int

char

float

main

(

)

[

]

if

then

elsif

else

while

do

for

{

}

}

{% %} (comentário tratado como token COMMENT)

id (identificadores)

num_int (constantes inteiras)

num_float (constantes float)

const_char ('A', 'x', etc)

:= (atribuição)

+ - * / ** (operadores aritméticos)

== != < > <= >= (operadores relacionais)

,

;

1.2 Símbolos Não-Terminais (V)

INICIO

BLOCO

DECLS

DECL

TIPO

LISTA_IDS

COMANDOS

COMANDO

CMD_ATRIB

CMD_IF

CMD_WHILE

CMD_DO

CMD_FOR

CMD_OU_BLOCO

COND

EXPR

EXPR_SIMP

TERMO

FATOR

2.2 Produções (P)

INICIO

<INICIO> ::= <TIPO> main ( ) <BLOCO>

Bloco

<BLOCO> ::= [<DECLS> <COMANDOS>]

Declarações

<DECLS> ::= <DECL> <DECLS> | ε

<DECL> ::= <TIPO> <LISTA_IDS> ;

Tipos

<TIPO> ::= void | int | float | char

Lista de Identificadores

<LISTA_IDS> ::= id <LISTA_IDS'>

<LISTA_IDS'> ::= , id <LISTA_IDS'> | ε

Comandos

<COMANDOS> ::= <COMANDO> <COMANDOS> | ε

Comando

<COMANDO> ::= <CMD_ATRIB>

| <CMD_IF>

| <CMD_WHILE>

| <CMD_DO>

| <CMD_FOR>

| <BLOCO>

Atribuição

<CMD_ATRIB> ::= id := <EXPR> ;

Comando IF / ELSIF / ELSE

<CMD_IF> ::= if (<COND>) then <CMD_OU_BLOCO> <CMD_IF'>

<CMD_IF'> ::= elsif (<COND>) then <CMD_OU_BLOCO> <CMD_IF'>

| else <CMD_OU_BLOCO>

| ε

Comando WHILE

<CMD_WHILE> ::= while (<COND>) do <CMD_OU_BLOCO>

Comando DO/WHILE

<CMD_DO> ::= do <CMD_OU_BLOCO> while (<COND>) ;

Comando FOR

<CMD_FOR> ::= for (id ; num_int ; num_int ; <EXPR>) <CMD_OU_BLOCO>

CMD ou Bloco

<CMD_OU_BLOCO> ::= <COMANDO> | <BLOCO>

Condição

<COND> ::= <EXPR> <OP_REL> <EXPR>

Expressões Aritméticas

<EXPR> ::= <EXPR_SIMP>

<EXPR_SIMP> ::= <TERMO> <EXPR_SIMP'>

<EXPR_SIMP'> ::= + <TERMO> <EXPR_SIMP'>

| - <TERMO> <EXPR_SIMP'>

| ε

<TERMO> ::= <FATOR> <TERMO'>

<TERMO'> ::= * <FATOR> <TERMO'>

| / <FATOR> <TERMO'>

| ε

<FATOR> ::= - <FATOR> (unário)

| <FATOR> ** <FATOR> (potência direita-associativa)

| (<EXPR>)

| num_int

| num_float

| const_char

| id

LEXEMA	**TOKEN**	**ATRIBUTO**
VOID	VOID	-
INT	INT	-
FLOAT	FLOAT	-
CHAR	CHAR	-
MAIN	MAIN	-
ID	ID	POSIÇÃO NA TABELA DE SIMBOLOS
NUM_INT	NUM_INT	VALOR INTEIRO
NUM_FLOAT	NUM_FLOAT	VALOR FLOAT
CONST_CHAT	CONST_CHAR	VALOR CHAR
:=	ATRIB	-
+	PLUS	-
-	MINUS	-
*	MULT	-
/	DIV	-
**	EXP	-
(	LPAREN	-
)	RPAREN	-
[	LBRACK	-
]	RBRACK	-
{	LBRACE	-
}	RBRACE	-
;	COMMA	-
,	COMMA	-

IF	IF	-
THEN	THEN	-
ELSIF	ELSIF	-
ELSE	ELSE	-
WHILE	WHILE	-
DO	DO	-
FOR	FOR	-
== != < > <= >=	OP_REL	CÓDIGO DO OPERADOR
{% ... %}	COMMENT	IGNORADO
WHITESPACE	WS	IGNORADO

3. Expressões Regulares dos Tokens

Identificadores

ID → [A-Za-z][A-Za-z0-9_]*

Números inteiros

NUM_INT → [0-9]+

Números float

NUM_FLOAT → [0-9]+\.[0-9]+(E[+-]?[0-9]+)?

Constante char

CONST_CHAR → '.[^']?'

Operadores aritméticos

PLUS → \+

MINUS → -

MULT → *

DIV → /

EXP → **

Operadores relacionais

OP_REL → == | != | <= | >= | < | >

Atribuição

ATRIB → :=

Comentários

COMMENT → \{\%(.|\n)*?\%\}

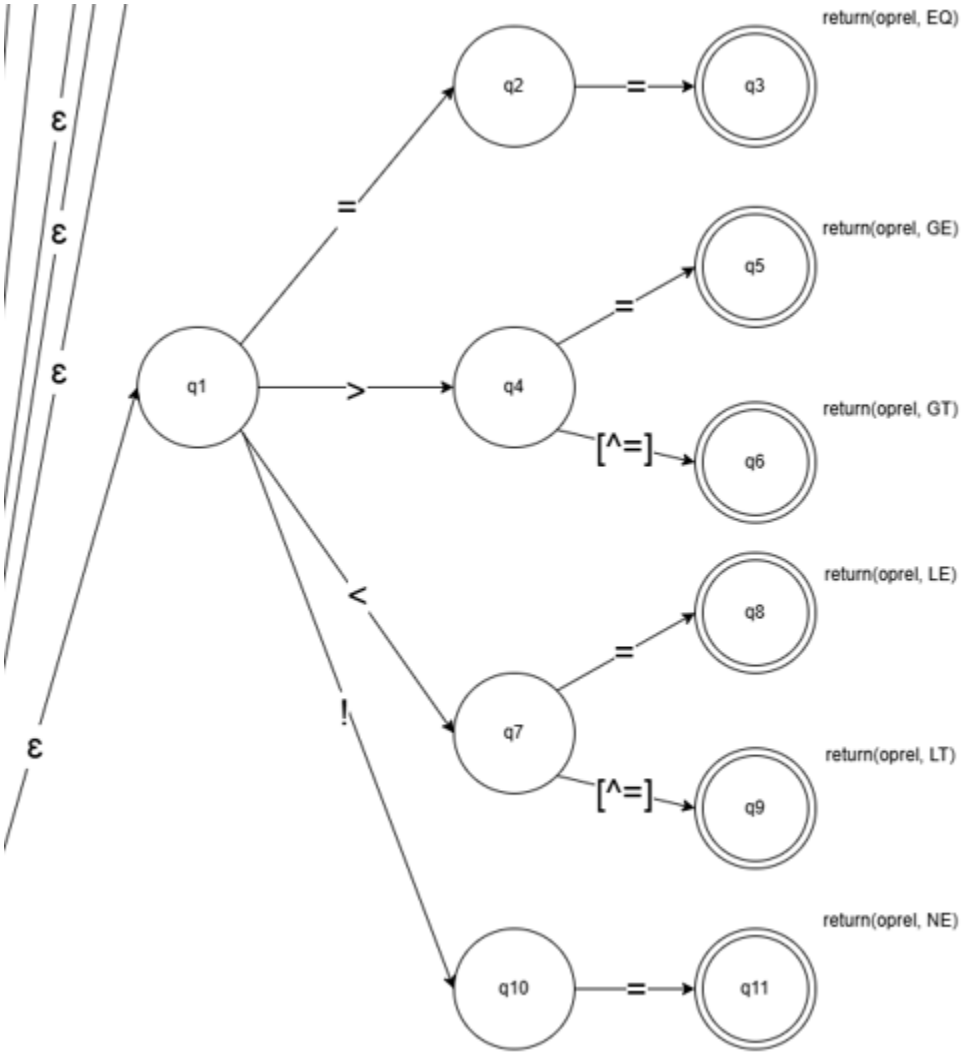
Espaços

WS → [\t\n]+

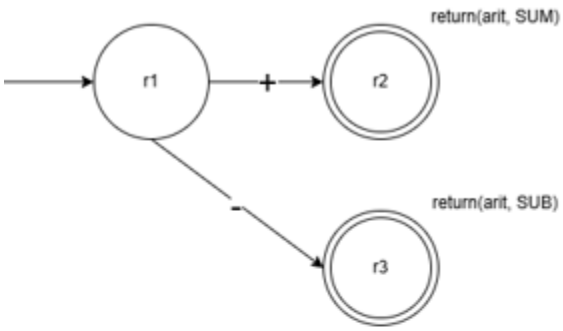
Análise Léxica

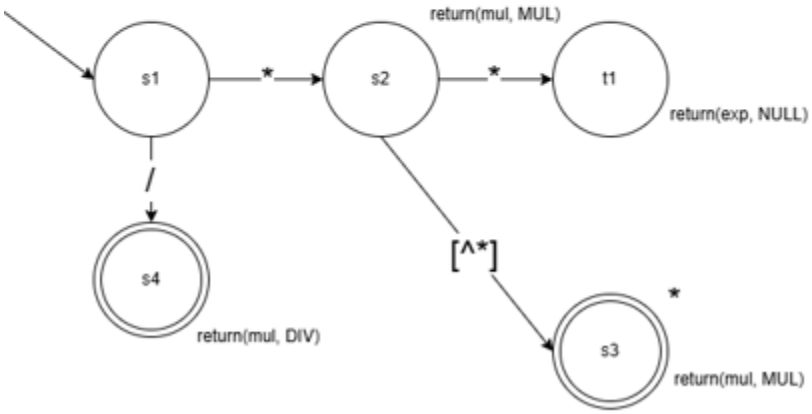
Diagrama de transição para cada token;

1. **Diagrama para oprel:**

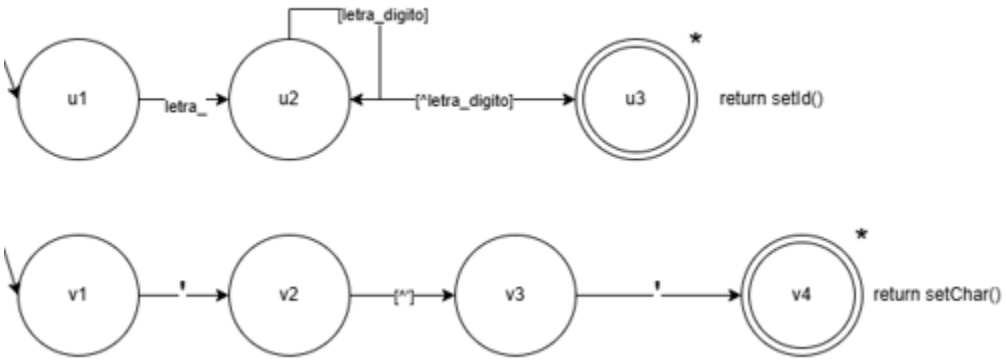


2. **Diagrama para operações aritméticas:**

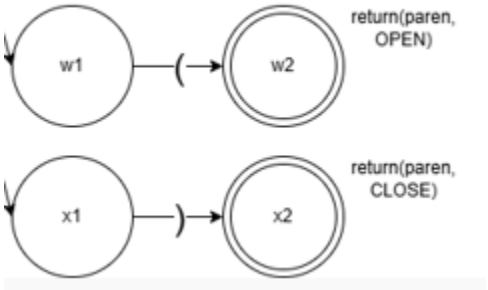




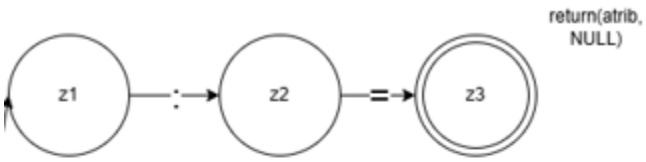
3. **Diagrama para id e caractere:**

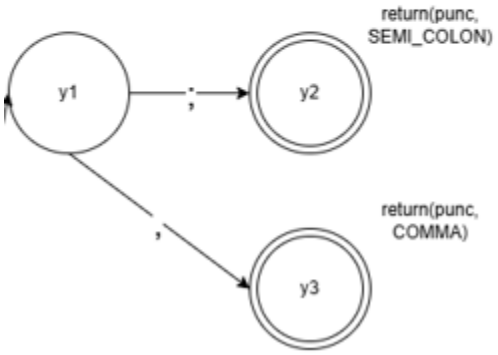


4. **Diagrama para parênteses:**

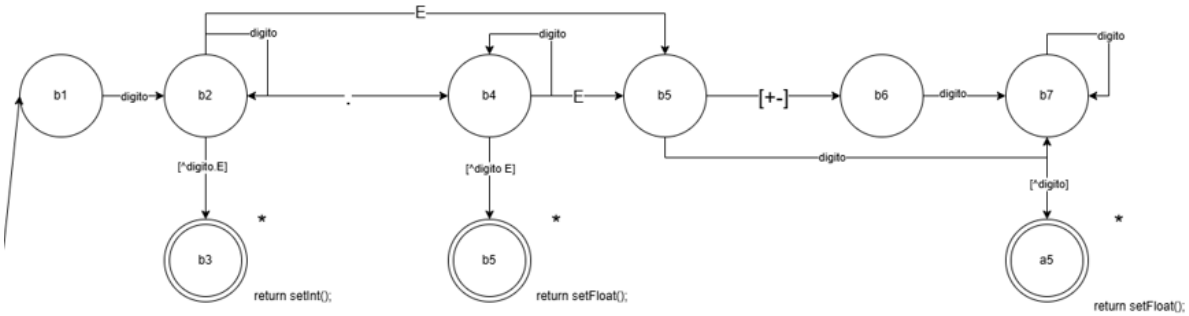


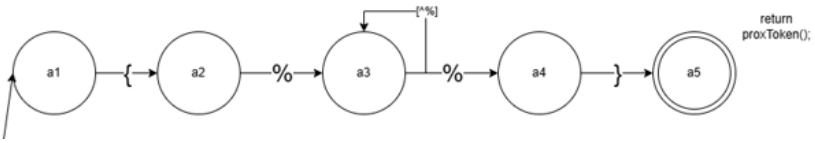
5. **Diagrama para pontuação e atribuição:**





6. **Diagrama para número e comentário:**

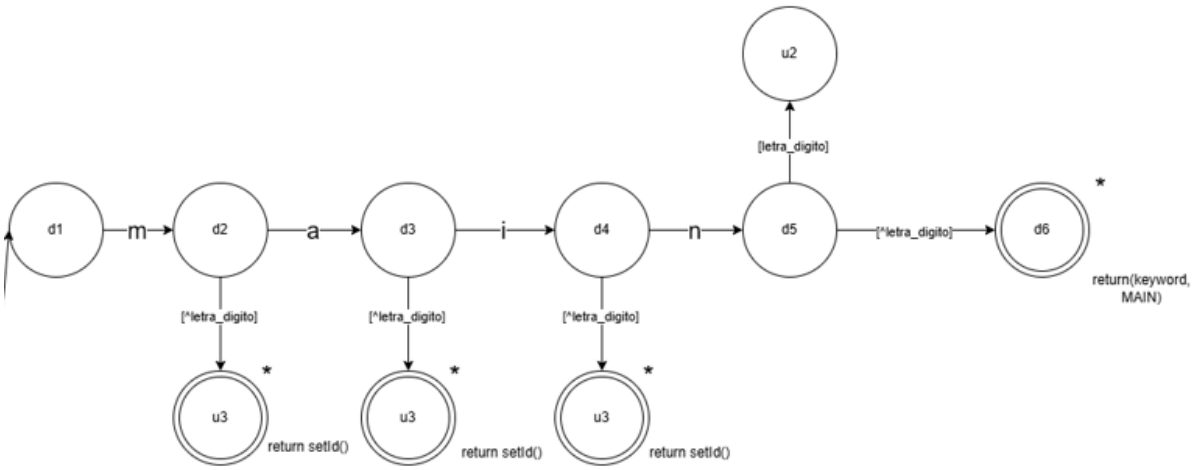




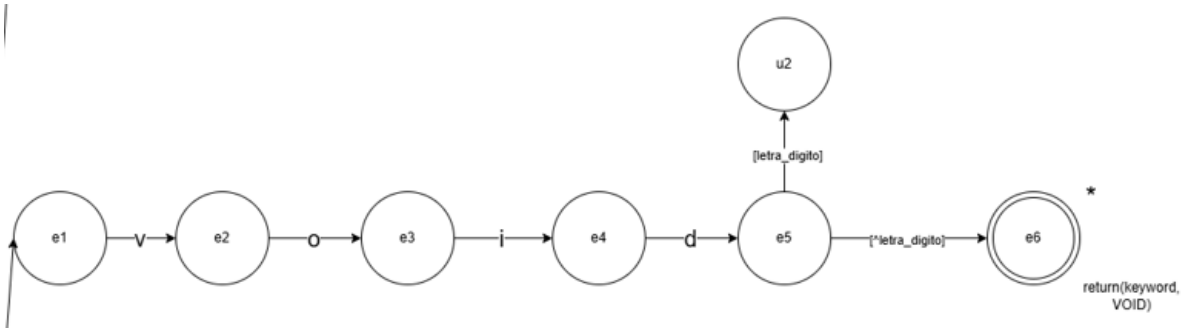
7. **Diagrama para palavras-chave:**

Cada palavra chave precisa ser tratada como um possível ID então para cada transição com alguma letra é necessário de outra transição para o diagrama que reconhece IDs representados pelos estados u3 e u2, como o diagrama final ficaria muito carregado de transições e setas escolhemos representar todas às transições apenas na primeira palavra-chave **main** então considere às próximas palavras-chaves com esse tratamento, apesar de não demonstrado.

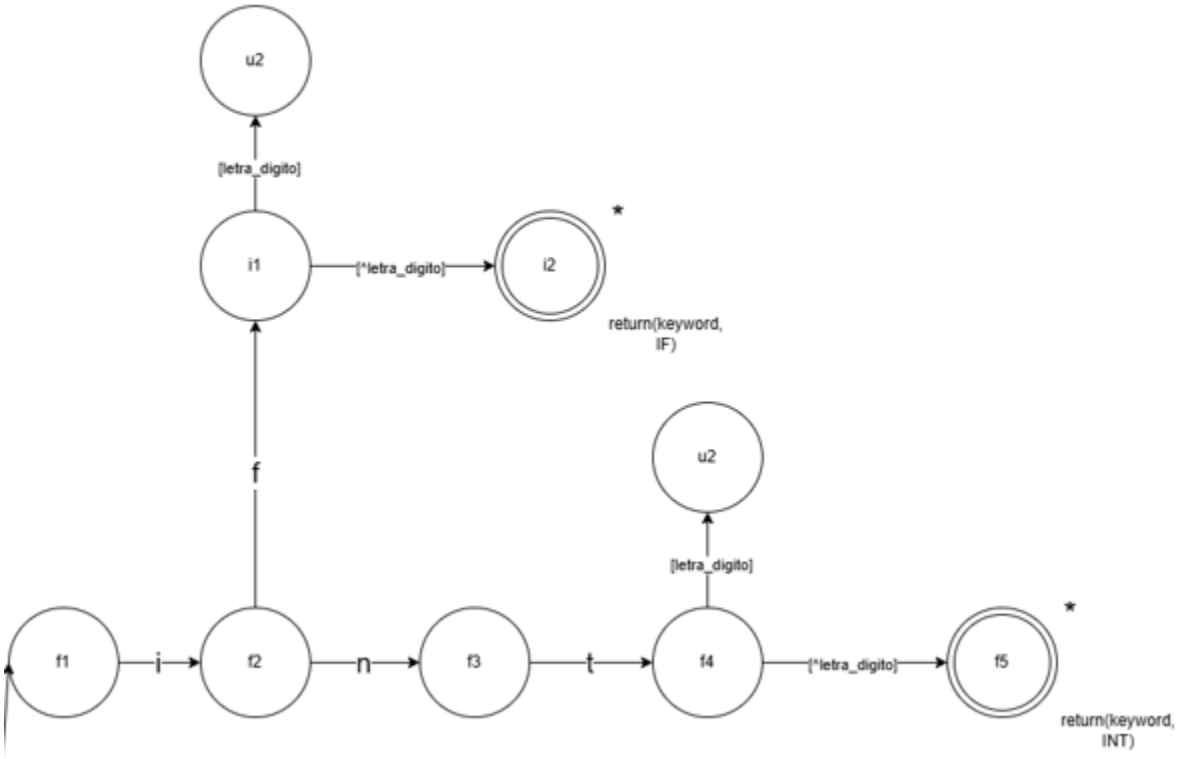
MAIN:


u2
[letra_digito]
d1
m
d2
a
d3
i
d4
n
d5
[^letra_digito]
d6
*
return(keyword, MAIN)
[^letra_digito]
[^letra_digito]
[^letra_digito]
u3
*
return setId()
u3
*
return setId()
u3
*
return setId()

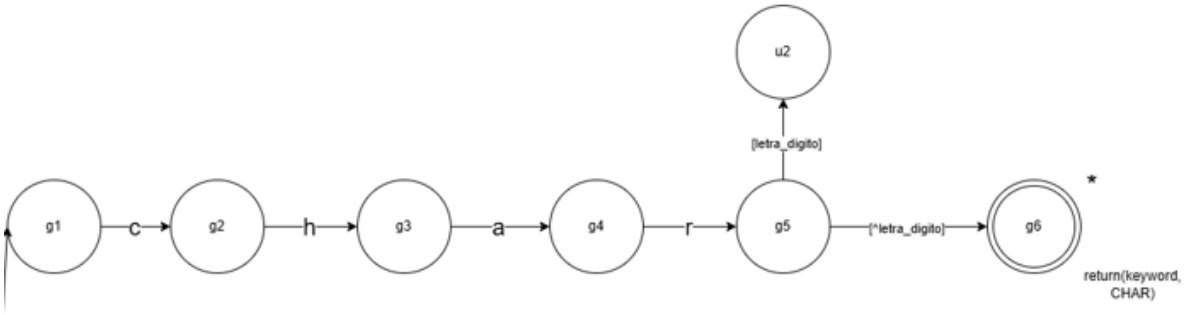
VOID:


u2
[letra_digito]
e1
v
e2
o
e3
i
e4
d
e5
[^letra_digito]
e6
*
return(keyword, VOID)

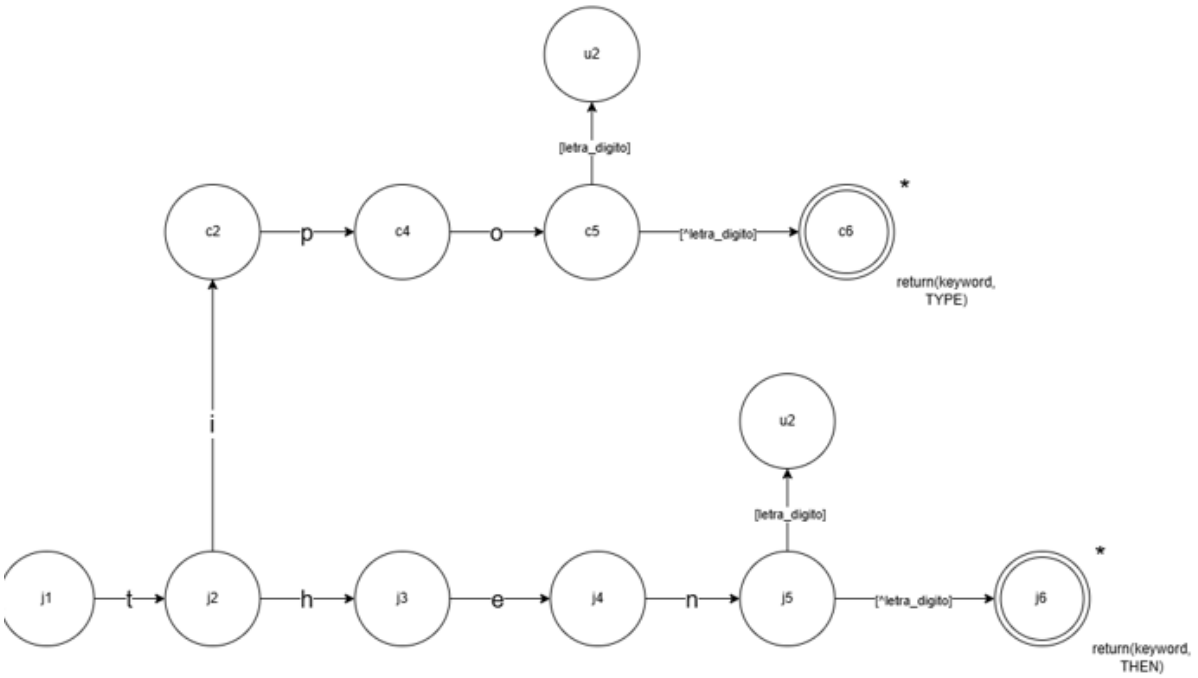
INT e IF:


u2
[letra_digito]
i1
[^letra_digito]
i2
*
return(keyword, IF)
f
u2
[letra_digito]
f1
i
f2
n
f3
t
f4
[^letra_digito]
f5
*
return(keyword, INT)

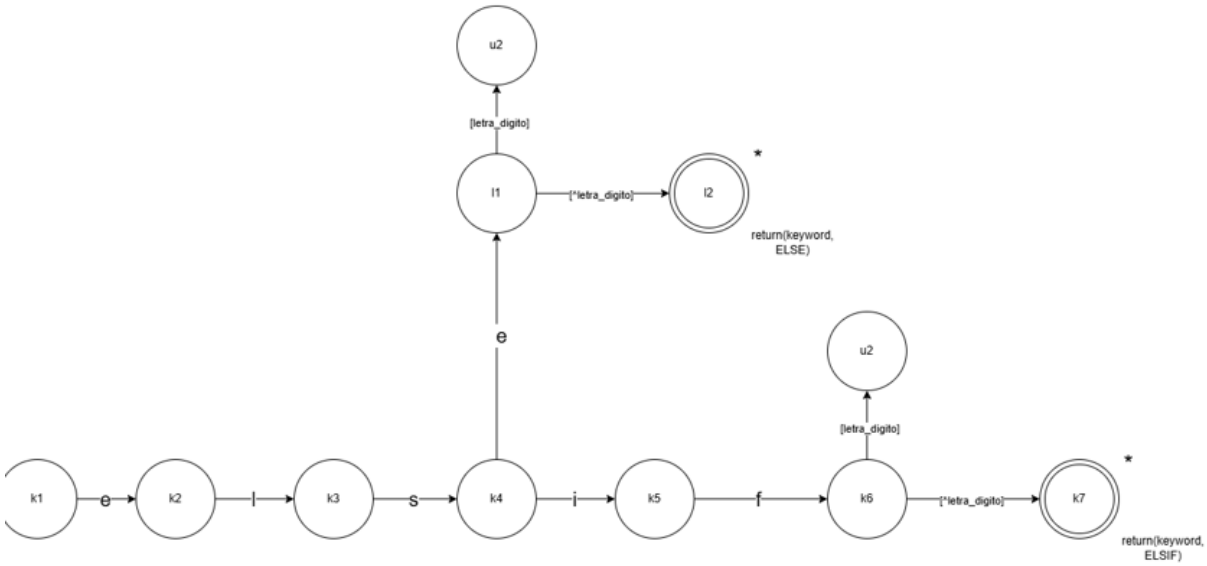
CHAR:



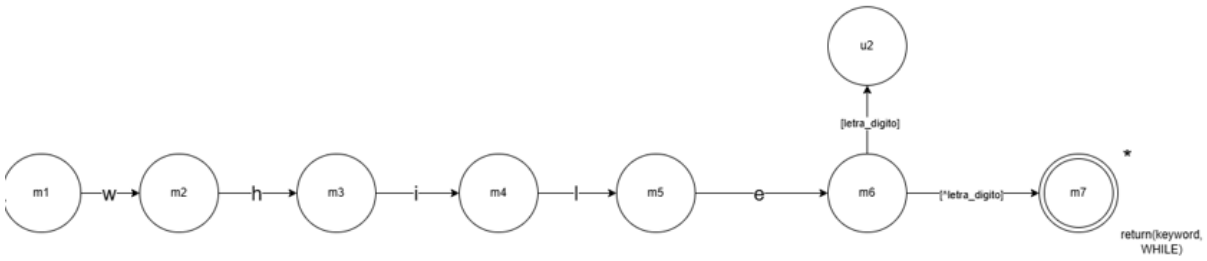
THEN e TIPO:



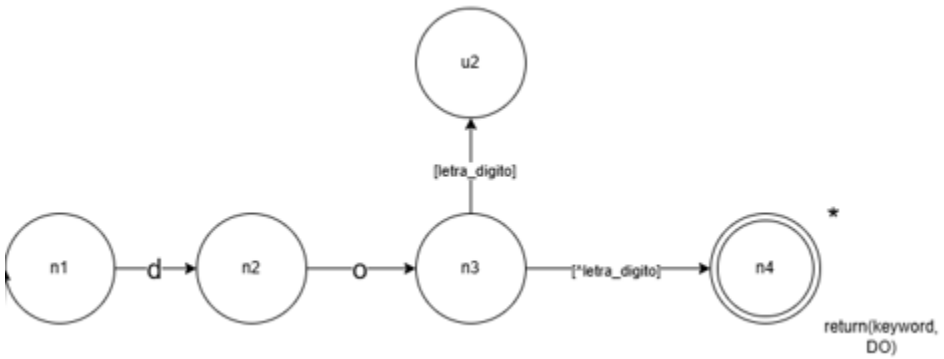
ELSE e ELSIF:



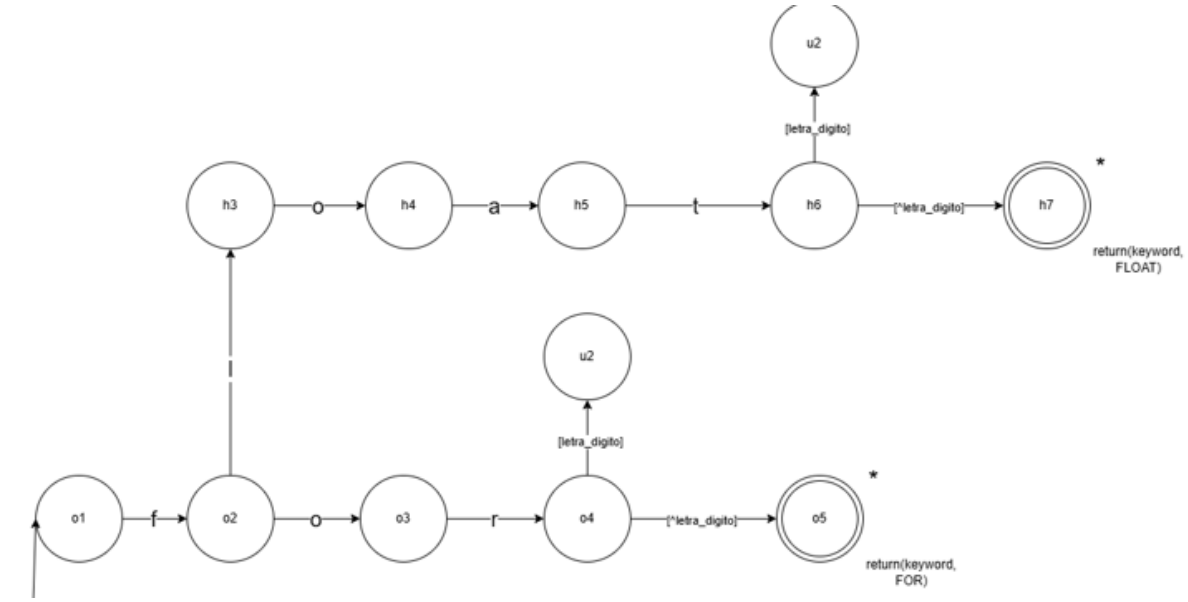
WHILE:



DO:



FOR e FLOAT:



8. **Diagrama unificado não determinístico completo no link (página não determinístico):**

9. **Diagrama de transição unificado determinístico completo no link (página "determinístico):**