

Memoria de la Práctica de Procesadores de Lenguajes

Diego José Abengózar Vilar, Alejandro García Castellanos,
Ignacio Javier Encinas Ramos

Grupo 82

January 19, 2020

Índice

1	Diseño del Analizador Léxico	2
1.1	Tokens	2
1.2	Gramática	2
1.3	Autómata Finito Determinista	3
1.4	Acciones Semánticas	4
1.5	Errores	5
1.6	Matriz de Transiciones	5
2	Tabla de Símbolos: Estructura e implementación	5
3	Diseño del Analizador Sintáctico	6
3.1	Gramática	6
3.2	Autómata Reconocedor de Prefijos Viables	7
3.2.1	Estados del autómata	8
3.3	Conflictos	10
3.4	Errores	11
3.5	Tabla de Decisión	12
4	Diseño del Analizador Semántico	14
4.1	Esquema de Traducción	14
4.2	Implementación del EdT	17
4.3	Gramática y Autómata Final del A. Sintáctico	20
4.4	Errores	22
5	Anexo de Pruebas	23

1 Diseño del Analizador Léxico

1.1 Tokens

<PuntoComa, - >
<CorcheteAbrir, - >
<CorcheteCerrar, - >
<ID, posTS> (Identificador)
<ENT, valor> (Dato de tipo entero)
<CAD, lex> (Dato de tipo cadena)
<ParentesisCerrar, - >
<ParentesisAbrir, - >
<SUMA, - > (Operador suma)
<MENOR, - > (Operador lógico menor)
<NOT, - > (Operador lógico de negación)
<ASIG, - > (Operador de asignación)
<ASIGOR, - > (Asignación con o lógico)
<DEC, - > (“var”)
<TipoVarENT, - > (“int”)
<TipoVarLOG, - > (“boolean”)
<TipoVarCAD, - > (“string”)
<Print, - >
<Input, - >
<Coma, - >
<Return, ->
<DECFunc, - > (“function”)
<IF, - >
<ELSE, - >

1.2 Gramática

$G(N, T, S, P)$

$S = A$

$N = \{ A, B, C, D, E, F, G, H \}$

$T = \{ del, ;, \{, \}, (,), +, <, !, =, ,, l, d, ', /, -, *, c \}$

$P:$

$A \rightarrow delA \mid ; \mid \{ \mid \} \mid (\mid) \mid + \mid < \mid ! \mid = \mid ,$

$A \rightarrow \mid B \mid lC \mid dD \mid 'E \mid /F$

$B \rightarrow =$

$C \rightarrow lC \mid dD \mid -C \mid \lambda$

$D \rightarrow dD \mid \lambda$

$E \rightarrow cE \mid *E \mid /E \mid ' ,$

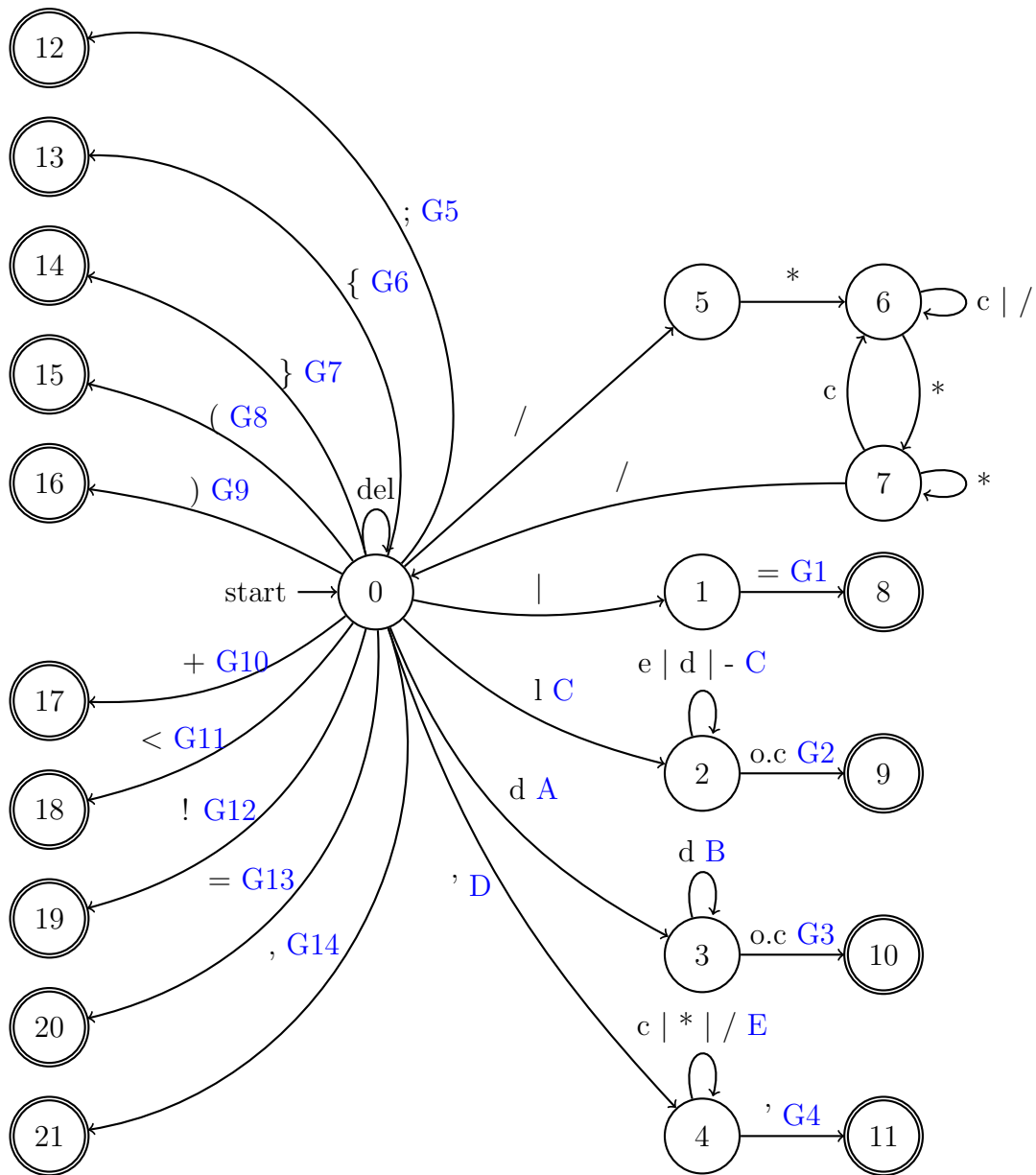
$F \rightarrow *G$

$G \rightarrow cG \mid /G \mid *H$

$H \rightarrow /A \mid cG \mid *H$

Donde, $c = T - \{*, /\}$

1.3 Autómata Finito Determinista



1.4 Acciones Semánticas

Lee \forall transicion menos o.c

C: CONCAT()

G_1 : GEN_TOKEN(ASIGOR, -)

```
 $G_2$ : if (lex  $\in$  palRes) GEN_TOKEN(palRes, -)
else if (FlagDeclUso = Decl)
    if (estaEnTSActual(lex))
        Error("Variable ya declarada")
    else
        p = INSERTAR_TS(lex)
        GEN_TOKEN(ID, p)
else
    p = BUSCA_TS(lex)
    if (p = null) p = INSERTAR_TS(lex)
    GEN_TOKEN(ID, p)
```

A: num = valor(d)

B: num = num * 10 + valor(d)

D: cont = 0

E: cont = cont + 1
CONCAT()

```
 $G_3$ : if (num  $\geq 2^{15}$ ) Error("Numero se sale del rango")
      else GEN_TOKEN(ENT, num)
```

```
 $G_4$ : if (cont > 64) Error("Exceso de caracteres en la cadena")
      else GEN_TOKEN(CAD, lex)
```

G_5 : GEN_TOKEN(PuntoComa, -)

G_6 : GEN_TOKEN(CorcheteAbrir, -)

G_7 : GEN_TOKEN(CorcheteCerrar, -)

G_8 : GEN_TOKEN(ParentesisAbrir, -)

G_9 : GEN_TOKEN(ParentesisCerrar, -)

G_{10} : GEN_TOKEN(SUMA, -)

G_{11} : GEN_TOKEN(MENOR, -)

G_{12} : GEN_TOKEN(NOT, -)

$G_{13} : \text{GEN_TOKEN}(\text{ASIG}, -)$

$G_{14} : \text{GEN_TOKEN}(\text{Coma}, -)$

Donde, $\text{palRes} = \{\text{var}, \text{int}, \text{boolean}, \text{string}, \text{print}, \text{input}, \text{function}, \text{return}, \text{if}, \text{else}\}$

1.5 Errores

Error 1: "Transición no prevista.";

Error 2: "Numero fuera de rango.";

Error 3: "Exceso de caracteres en la cadena.";

Error 4: "Variable ya declarada.";

1.6 Matriz de Transiciones

MT_AFD		letra	digito	'	/	-	carácter	*	delimitador
→ 0	1 lee	2 C	3A	4 D	5 lee	-1 error	-1 error	-1 error	0 lee
1	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error
2	9 G2	2 C	2 C	9 G2	9 G2	2 C	9 G2	9 G2	9 G2
3	10 G3	10 G3	3 B	10 G3	10 G3	10 G3	10 G3	10 G3	10 G3
4	4 E	4 E	4 E	11 G4	4 E	4 E	4 E	4 E	4 E
5	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	6 lee	-1 error
6	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	7 lee	6 lee
7	6 lee	6 lee	6 lee	6 lee	0 lee	6 lee	6 lee	7 lee	6 lee

MT_AFD	;	{	}	()	+	<	!	=	,
→ 0	12 G5	13 G6	14 G7	15 G8	16 G9	17 G10	18 G11	19 G12	20 G13	21 G14
1	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	8 G1	-1 error
2	9 G2	9 G2	9 G2	9 G2	9 G2	9 G2	9 G2	9 G2	9 G2	9 G2
3	10 G3	10 G3	10 G3	10 G3	10 G3	10 G3	10 G3	10 G3	10 G3	10 G3
4	4 E	4 E	4 E	4 E	4 E	4 E	4 E	4 E	4 E	4 E
5	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error
6	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee
7	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee

2 Tabla de Símbolos: Estructura e implementación

Contiene la información de los identificadores, de los cuales se guardan los campos: lexema, tipo y desplazamiento. Para las funciones, además, se guardará el número de parámetros, su tipo, la forma de paso de parámetros, el tipo del valor de retorno y etiqueta de función (formada por nombre y su posición en tabla de símbolos).

La tabla de símbolos estará formada por dos matrices de tamaño dinámico; la primera contendrán los identificadores de ámbito global y la segunda del local. Así pues, esta segunda se creará al encontrar la declaración de una función y se borrará al acabar de ser declarada. También se utiliza un flag de declaración o uso (FlagDeclUso), un flag para saber cual es la tabla actual y dos más para el valor del desplazamiento en cada una de las tablas.

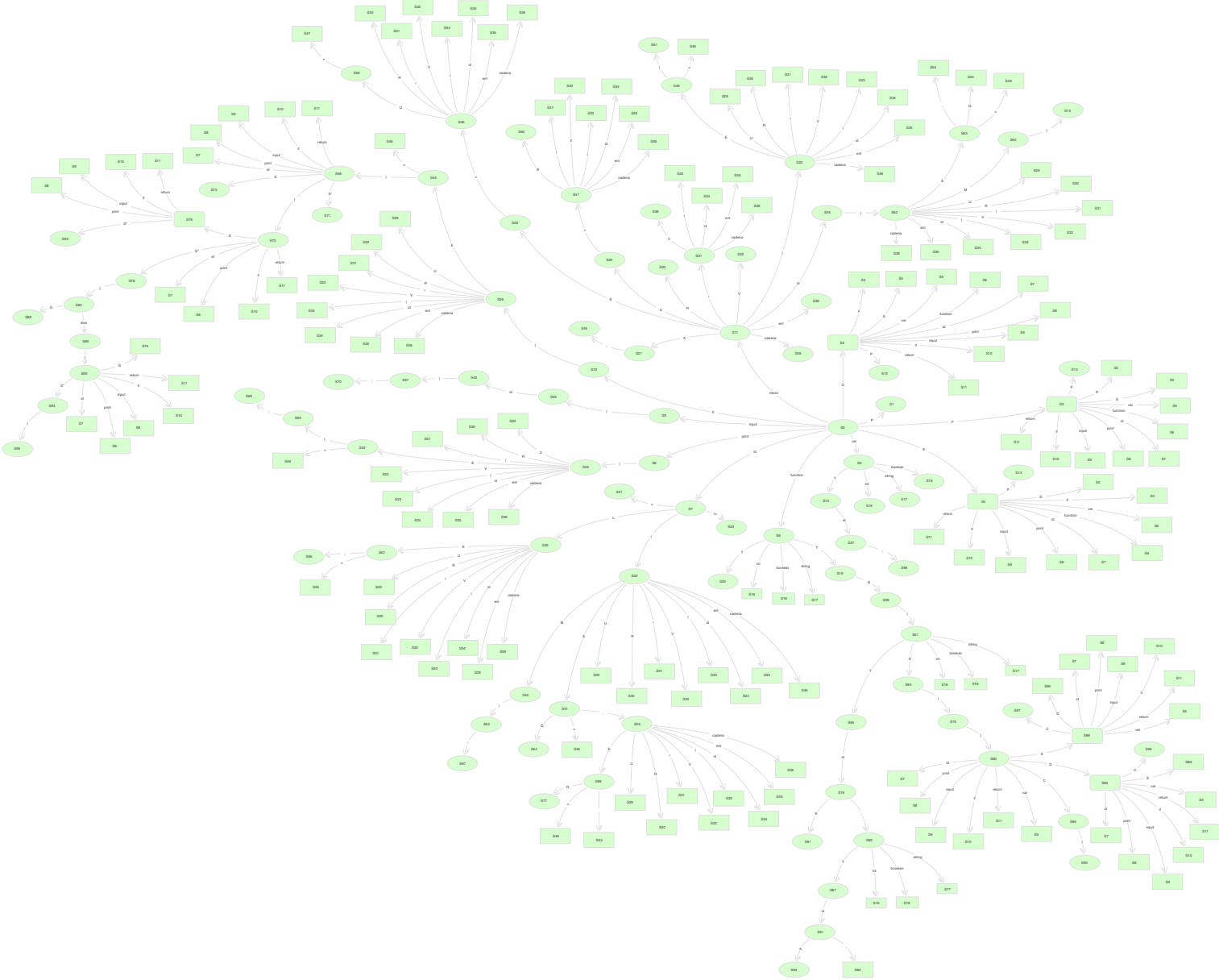
3 Diseño del Analizador Sintáctico

3.1 Gramática

Terminales = { ; { } id ent cadena () + < ! = |= var int
boolean string print input , return function if else }
NoTerminales = { P D T F T1 A K C S L M Q S1 G X E U R V S2 }
Axioma = P
Producciones = {
P → D P
P → F P
P → S P
D → var T id ;
T → int
T → string
T → boolean
F → function T1 id (A) { C }
T1 → lambda
T1 → T
A → T id K
A → lambda
K → lambda
K → , T id K
C → D C
C → S C
C → lambda
S → id L E ;
S → id (M) ;
S → print (E) ;
S → input (id) ;
S → if (E) S1
S → return X ;
L → |=
L → =
M → E Q
M → lambda
Q → lambda
Q → , E Q
S1 → { S2 } G
S1 → S
G → else { S2 }
G → lambda
X → E
X → lambda
E → E < U
E → U
U → U + R
U → R
R → ! V
R → V
V → (E)

$V \rightarrow id$
 $V \rightarrow id (M)$
 $V \rightarrow ent$
 $V \rightarrow cadena$
 $S2 \rightarrow S S2$
 $S2 \rightarrow S$
 $P \rightarrow lambda$
 $\}$

3.2 Autómata Reconocedor de Prefijos Viables¹



¹Los estados con forma de rectángulo redondeado son aquellos con bucles a si mismos. La etiqueta de dicha arista es la misma que la arista que conecta dicho estado y su antecedente

3.2.1 Estados del autómata

$S_0 = \{P1 \rightarrow \bullet P, P \rightarrow \bullet DP, P \rightarrow \bullet SP, P \rightarrow \bullet , D \rightarrow \bullet \text{ var } T \text{ id}; ,$
 $F \rightarrow \bullet \text{ function } T1 \text{ id}(A)\{C\}, S \rightarrow \bullet \text{ id } L \text{ E}; , S \rightarrow \bullet \text{ id}(M);$
 $S \rightarrow \bullet \text{ print}(E); , S \rightarrow \bullet \text{ input}(\text{id}); , S \rightarrow \bullet \text{ if}(E) \text{ S1},$
 $S \rightarrow \bullet \text{ return } X;\}$
 $S_1 = \{P1 \rightarrow P \bullet\}$
 $S_2 = \{P \rightarrow D \bullet P, P \rightarrow \bullet DP, P \rightarrow \bullet SP, P \rightarrow \bullet , D \rightarrow \bullet \text{ var } T \text{ id}; ,$
 $F \rightarrow \bullet \text{ function } T1 \text{ id}(A)\{C\}, S \rightarrow \bullet \text{ id } L \text{ E}; , S \rightarrow \bullet \text{ id}(M);$
 $S \rightarrow \bullet \text{ print}(E); , S \rightarrow \bullet \text{ input}(\text{id}); , S \rightarrow \bullet \text{ if}(E) \text{ S1},$
 $S \rightarrow \bullet \text{ return } X;\}$
 $S_3 = \{P \rightarrow F \bullet P, P \rightarrow \bullet DP, P \rightarrow \bullet FP, P \rightarrow \bullet SP, P \rightarrow \bullet ,$
 $D \rightarrow \bullet \text{ var } T \text{ id}; , F \rightarrow \bullet \text{ function } T1 \text{ id}(A)\{C\},$
 $S \rightarrow \bullet \text{ id } L \text{ E}; , S \rightarrow \bullet \text{ id}(M); , S \rightarrow \bullet \text{ print}(E); ,$
 $S \rightarrow \bullet \text{ input}(\text{id}); , S \rightarrow \bullet \text{ if}(E) \text{ S1}, S \rightarrow \bullet \text{ return } X;\}$
 $S_4 = \{P \rightarrow S \bullet P, P \rightarrow \bullet DP, P \rightarrow \bullet FP, P \rightarrow \bullet SP, P \rightarrow \bullet ,$
 $D \rightarrow \bullet \text{ var } T \text{ id}; , F \rightarrow \bullet \text{ function } T1 \text{ id}(A)\{C\},$
 $S \rightarrow \bullet \text{ id } L \text{ E}; , S \rightarrow \bullet \text{ id}(M); , S \rightarrow \bullet \text{ print}(E); ,$
 $S \rightarrow \bullet \text{ input}(\text{id}); , S \rightarrow \bullet \text{ if}(E) \text{ S1}, S \rightarrow \bullet \text{ return } X;\}$
 $S_5 = \{D \rightarrow \text{ var } \bullet T \text{ id}; , T \rightarrow \bullet \text{ int } , T \rightarrow \bullet \text{ string } , T \rightarrow \bullet \text{ boolean}\}$
 $S_6 = \{F \rightarrow \text{ function } \bullet T1 \text{ id}(A)\{C\}, T1 \rightarrow \bullet , T1 \rightarrow \bullet T,$
 $T \rightarrow \bullet \text{ int } , T \rightarrow \bullet \text{ string } , T \rightarrow \bullet \text{ boolean}\}$
 $S_7 = \{S \rightarrow \text{ id } \bullet L \text{ E}; , S \rightarrow \text{ id } \bullet (M); , L \rightarrow | = , L \rightarrow \bullet =\}$
 $S_8 = \{S \rightarrow \text{ print } \bullet (E);\}$
 $S_9 = \{S \rightarrow \text{ input } \bullet (\text{id});\}$
 $S_{10} = \{S \rightarrow \text{ if } \bullet (E) \text{ S1}\}$
 $S_{11} = \{S \rightarrow \text{ return } \bullet X; , X \rightarrow \bullet , X \rightarrow \bullet E, E \rightarrow \bullet E < U, E \rightarrow \bullet U,$
 $U \rightarrow \bullet U + R, U \rightarrow \bullet R, R \rightarrow \bullet ! V, R \rightarrow \bullet V, V \rightarrow \bullet (E),$
 $V \rightarrow \bullet \text{ id } , V \rightarrow \bullet \text{ id}(M), V \rightarrow \bullet \text{ ent } , V \rightarrow \bullet \text{ cadena}\}$
 $S_{12} = \{P \rightarrow DP \bullet\}$
 $S_{13} = \{P \rightarrow FP \bullet\}$
 $S_{14} = \{P \rightarrow SP \bullet\}$
 $S_{15} = \{D \rightarrow \text{ var } T \bullet \text{ id};\}$
 $S_{16} = \{T \rightarrow \text{ int } \bullet\}$
 $S_{17} = \{T \rightarrow \text{ string } \bullet\}$
 $S_{18} = \{T \rightarrow \text{ boolean } \bullet\}$
 $S_{19} = \{F \rightarrow \text{ function } T1 \bullet \text{ id}(A)\{C\}\}$
 $S_{20} = \{T1 \rightarrow T \bullet\}$
 $S_{21} = \{L \rightarrow = \bullet\}$
 $S_{22} = \{S \rightarrow \text{ id } (\bullet M); , M \rightarrow \bullet E \text{ Q}, M \rightarrow \bullet , E \rightarrow \bullet E < U, E \rightarrow \bullet U,$
 $U \rightarrow \bullet U + R, U \rightarrow \bullet R, R \rightarrow \bullet ! V, R \rightarrow \bullet V, V \rightarrow \bullet (E),$
 $V \rightarrow \bullet \text{ id } , V \rightarrow \bullet \text{ id}(M), V \rightarrow \bullet \text{ ent } , V \rightarrow \bullet \text{ cadena}\}$
 $S_{23} = \{L \rightarrow | = \bullet\}$
 $S_{24} = \{S \rightarrow \text{ print } (\bullet E); , E \rightarrow \bullet E < U, E \rightarrow \bullet U, U \rightarrow \bullet U + R,$
 $U \rightarrow \bullet R, R \rightarrow \bullet ! V, R \rightarrow \bullet V, V \rightarrow \bullet (E), V \rightarrow \bullet \text{ id } ,$
 $V \rightarrow \bullet \text{ id}(M), V \rightarrow \bullet \text{ ent } , V \rightarrow \bullet \text{ cadena}\}$
 $S_{25} = \{S \rightarrow \text{ input } (\bullet \text{id})\}$
 $S_{26} = \{S \rightarrow \text{ if } (\bullet E) \text{ S1}, E \rightarrow \bullet E < U, E \rightarrow \bullet U, U \rightarrow \bullet U + R,$
 $U \rightarrow \bullet R, R \rightarrow \bullet ! V, R \rightarrow \bullet V, V \rightarrow \bullet (E), V \rightarrow \bullet \text{ id } ,$
 $V \rightarrow \bullet \text{ id}(M), V \rightarrow \bullet \text{ ent } , V \rightarrow \bullet \text{ cadena}\}$
 $S_{27} = \{S \rightarrow \text{ return } X \bullet ;\}$
 $S_{28} = \{X \rightarrow E \bullet , E \rightarrow E \bullet < U\}$
 $S_{29} = \{E \rightarrow U \bullet , U \rightarrow U \bullet + R\}$
 $S_{30} = \{U \rightarrow R \bullet\}$
 $S_{31} = \{R \rightarrow ! \bullet V, V \rightarrow \bullet (E), V \rightarrow \bullet \text{ id } , V \rightarrow \bullet \text{ id}(M),$
 $V \rightarrow \bullet \text{ ent } , V \rightarrow \bullet \text{ cadena}\}$
 $S_{32} = \{R \rightarrow V \bullet\}$
 $S_{33} = \{V \rightarrow (\bullet E), E \rightarrow \bullet E < U, E \rightarrow \bullet U, U \rightarrow \bullet U + R,$

$U \rightarrow \bullet R, R \rightarrow \bullet ! V, R \rightarrow \bullet V, V \rightarrow \bullet (E), V \rightarrow \bullet id,$
 $V \rightarrow \bullet id(M), V \rightarrow \bullet ent, V \rightarrow \bullet cadena\}$
 $S_{34}=\{V \rightarrow id \bullet, V \rightarrow id \bullet (M)\}$
 $S_{35}=\{V \rightarrow ent \bullet\}$
 $S_{36}=\{V \rightarrow cadena \bullet\}$
 $S_{37}=\{D \rightarrow var T id \bullet;\}$
 $S_{38}=\{F \rightarrow function T1 id \bullet (A)\{C\}\}$
 $S_{39}=\{S \rightarrow id L \bullet E, E \rightarrow \bullet E < U, E \rightarrow \bullet U, U \rightarrow \bullet U + R,$
 $U \rightarrow \bullet R, R \rightarrow \bullet ! V, R \rightarrow \bullet V, V \rightarrow \bullet (E), V \rightarrow \bullet id,$
 $V \rightarrow \bullet id(M), V \rightarrow \bullet ent, V \rightarrow \bullet cadena\}$
 $S_{40}=\{S \rightarrow id (M \bullet);\}$
 $S_{41}=\{M \rightarrow E \bullet Q, E \rightarrow E \bullet < U, Q \rightarrow \bullet, Q \rightarrow \bullet ,EQ\}$
 $S_{42}=\{S \rightarrow print(E \bullet);, E \rightarrow E \bullet < U\}$
 $S_{43}=\{S \rightarrow input(id \bullet);\}$
 $S_{44}=\{S \rightarrow if(E \bullet) S1, E \rightarrow E \bullet < U\}$
 $S_{45}=\{S \rightarrow return X; \bullet\}$
 $S_{46}=\{E \rightarrow E < \bullet U, U \rightarrow \bullet R, U \rightarrow \bullet U + R, R \rightarrow \bullet ! V, R \rightarrow \bullet V$
 $V \rightarrow \bullet (E), V \rightarrow \bullet id, V \rightarrow \bullet id(M), V \rightarrow \bullet ent, V \rightarrow \bullet cadena\}$
 $S_{47}=\{U \rightarrow U + \bullet R, R \rightarrow \bullet ! V, V \rightarrow \bullet (E), V \rightarrow \bullet id(M),$
 $V \rightarrow \bullet ent, V \rightarrow \bullet cadena\}$
 $S_{48}=\{R \rightarrow ! V \bullet\}$
 $S_{49}=\{V \rightarrow (E \bullet), E \rightarrow E \bullet < U\}$
 $S_{50}=\{V \rightarrow id(\bullet M), M \rightarrow \bullet E Q, M \rightarrow \bullet, E \rightarrow \bullet E < U, E \rightarrow \bullet U,$
 $U \rightarrow \bullet U + R, U \rightarrow \bullet R, R \rightarrow \bullet ! V, R \rightarrow \bullet V, V \rightarrow \bullet (E),$
 $V \rightarrow \bullet id, V \rightarrow \bullet id(M), V \rightarrow \bullet ent, V \rightarrow \bullet cadena\}$
 $S_{51}=\{F \rightarrow function T1 id(\bullet A)\{C\}, A \rightarrow \bullet T id K, A \rightarrow \bullet, T \rightarrow \bullet int,$
 $T \rightarrow \bullet string, T \rightarrow \bullet boolean\}$
 $S_{52}=\{S \rightarrow id L E \bullet; , E \rightarrow E \bullet < U\}$
 $S_{53}=\{S \rightarrow id(M) \bullet; \}$
 $S_{54}=\{M \rightarrow E Q \bullet\}$
 $S_{55}=\{Q \rightarrow , \bullet E Q, E \rightarrow \bullet E < U, E \rightarrow \bullet U,$
 $U \rightarrow \bullet U + R, U \rightarrow \bullet R, R \rightarrow \bullet ! V, R \rightarrow \bullet V, V \rightarrow \bullet (E),$
 $V \rightarrow \bullet id, V \rightarrow \bullet id(M), V \rightarrow \bullet ent, V \rightarrow \bullet cadena\}$
 $S_{56}=\{S \rightarrow print(E) \bullet; \}$
 $S_{57}=\{S \rightarrow input(id) \bullet; \}$
 $S_{58}=\{S \rightarrow if(E) \bullet S1, S1 \rightarrow \bullet \{S2\}G, S1 \rightarrow \bullet S, S \rightarrow \bullet id L E; ,$
 $S \rightarrow \bullet id(M); , S \rightarrow \bullet print(E); , S \rightarrow \bullet input(id); ,$
 $S \rightarrow \bullet if(E)S1, S \rightarrow \bullet return X; \}$
 $S_{59}=\{E \rightarrow E < U \bullet, U \rightarrow U \bullet + R\}$
 $S_{60}=\{U \rightarrow U + R \bullet\}$
 $S_{61}=\{V \rightarrow (E) \bullet\}$
 $S_{62}=\{V \rightarrow id(M \bullet)\}$
 $S_{63}=\{M \rightarrow E \bullet Q, E \rightarrow E \bullet < U, Q \rightarrow \bullet, Q \rightarrow \bullet ,EQ\}$
 $S_{64}=\{F \rightarrow function T1 id(A \bullet)\{C\}\}$
 $S_{65}=\{A \rightarrow T \bullet id K\}$
 $S_{66}=\{S \rightarrow id L E ; \bullet\}$
 $S_{67}=\{S \rightarrow id (M); \bullet\}$
 $S_{68}=\{Q \rightarrow ,E \bullet Q, E \rightarrow E \bullet < U, Q \rightarrow \bullet, Q \rightarrow \bullet ,EQ\}$
 $S_{69}=\{S \rightarrow print(E); \bullet\}$
 $S_{70}=\{S \rightarrow input(id); \bullet\}$
 $S_{71}=\{S \rightarrow if(E) S1 \bullet\}$
 $S_{72}=\{S1 \rightarrow \{\bullet S2\}G, S2 \rightarrow \bullet S S2, S2 \rightarrow \bullet S, S \rightarrow \bullet id L E; ,$
 $S \rightarrow \bullet id(M); , S \rightarrow \bullet print(E); , S \rightarrow \bullet input(id); ,$
 $S \rightarrow \bullet if(E)S1, S \rightarrow \bullet return X; \}$
 $S_{73}=\{S1 \rightarrow S \bullet\}$
 $S_{74}=\{V \rightarrow id(M) \bullet\}$
 $S_{75}=\{F \rightarrow function T1 id (K) \bullet \{C\}\}$
 $S_{76}=\{A \rightarrow T id \bullet K , K \rightarrow \bullet, K \rightarrow \bullet , T id K\}$

$S_{77}=\{Q \rightarrow , E Q \bullet\}$
 $S_{78}=\{S1 \rightarrow \{S2 \bullet\} G\}$
 $S_{79}=\{S2 \rightarrow S \bullet S2, S2 \rightarrow S \bullet, S2 \rightarrow \bullet S S2, S \rightarrow \bullet id L E ; ,$
 $S \rightarrow \bullet id(M); , S \rightarrow \bullet print(E); S \rightarrow \bullet if(E)S1; ,$
 $S \rightarrow \bullet input(id); , S \rightarrow return X ;\}$
 $S_{80}=\{F \rightarrow function T1 id (K) \{ \bullet C\}, C \rightarrow \bullet D C , C \rightarrow \bullet ,$
 $D \rightarrow \bullet var T id ; , S \rightarrow \bullet id L E , S \rightarrow \bullet id (M); ,$
 $S \rightarrow \bullet print (E); , S \rightarrow \bullet input (id); , S \rightarrow \bullet if (E) S1,$
 $S \rightarrow \bullet return X ;\}$
 $S_{81}=\{A \rightarrow T id K \bullet\}$
 $S_{82}=\{K \rightarrow , \bullet T id K, T \rightarrow \bullet int , T \rightarrow \bullet string , T \rightarrow \bullet boolean\}$
 $S_{83}=\{S1 \rightarrow \{S2\} \bullet G , G \rightarrow \bullet else \{S2\} , G \rightarrow \bullet\}$
 $S_{84}=\{F \rightarrow function T1 id (K) \{C \bullet\}\}$
 $S_{85}=\{C \rightarrow D \bullet C, C \rightarrow \bullet D C, C \rightarrow \bullet S C, C \rightarrow \bullet, D \rightarrow \bullet var T id ; ,$
 $S \rightarrow \bullet id L E ; , S \rightarrow \bullet id (M) ; , S \rightarrow \bullet print (E) ; ,$
 $S \rightarrow \bullet input (id) ; , S \rightarrow \bullet if (E) S1, S \rightarrow \bullet return X ;\}$
 $S_{86}=\{C \rightarrow S \bullet C, C \rightarrow \bullet D C, C \rightarrow \bullet S C, C \rightarrow \bullet, D \rightarrow \bullet var T id ; ,$
 $S \rightarrow \bullet id L E ; , S \rightarrow \bullet id (M) ; , S \rightarrow \bullet print (E) ; ,$
 $S \rightarrow \bullet input (id) ; , S \rightarrow \bullet if (E) S1, S \rightarrow \bullet return X ;\}$
 $S_{87}=\{K \rightarrow , T \bullet id K\}$
 $S_{88}=\{S1 \rightarrow \{S2\} G \bullet\}$
 $S_{89}=\{G \rightarrow else \bullet \{S2\}\}$
 $S_{90}=\{F \rightarrow function T1 id (K) \{C\} \bullet\}$
 $S_{91}=\{K \rightarrow , T id \bullet K, K \rightarrow \bullet, K \rightarrow \bullet, T id K\}$
 $S_{92}=\{G \rightarrow else \{ \bullet S2\}, S2 \rightarrow \bullet S S2, S2 \rightarrow \bullet S, S \rightarrow \bullet id L E ; ,$
 $S \rightarrow \bullet id (M) ; , S \rightarrow \bullet print (E) ; , S \rightarrow \bullet input(id); ,$
 $S \rightarrow \bullet if (E) S1, S \rightarrow \bullet return X ;\}$
 $S_{93}=\{K \rightarrow , T id K \bullet\}$
 $S_{94}=\{G \rightarrow else \{ S2 \bullet \}\}$
 $S_{95}=\{G \rightarrow else \{ S2 \} \bullet\}$
 $S_{96}=\{C \rightarrow D C \bullet\}$
 $S_{97}=\{C \rightarrow S C \bullet\}$
 $S_{98}=\{D \rightarrow var T id ; \bullet\}$
 $S_{99}=\{S2 \rightarrow S S2 \bullet\}$

3.3 Conflictos

Como podemos observar en la tabla de decisión no hay ningún conflicto.

Los posibles conflictos son:

Reducción-Reducción

Podríamos ver como en los posibles estados con este conflicto, en nuestro caso ninguno, se verifica que

$\forall \{A \rightarrow \alpha \bullet, B \rightarrow \beta \bullet\} \subset S_x \Rightarrow Follow(A) \cap Follow(B) = \emptyset$ (Esto lo podemos observar al no tener dos entradas de reducción en la misma celda de cada fila de S_x)

Reducción-Desplazamiento

Podemos ver como en los posibles estados con este conflicto, $S_0, S_2, S_3, S_4, S_6, S_{11}, S_{22}, S_{28}, S_{29}, S_{34}, S_{41}, S_{50}, S_{51}, S_{59}, S_{63}, S_{68}, S_{76}, S_{79}, S_{80}, S_{83}, S_{85}, S_{86}, S_{91}$, se verifica

$\forall \{A \rightarrow \alpha \bullet b \gamma, C \rightarrow \beta \bullet\} \subset S_x \Rightarrow b \notin Follow(C)$ (Esto lo podemos observar al no tener una entrada de desplazamiento y otra de reducción en la misma celda de cada fila de S_x)

Por ejemplo, para los estados S_0, S_2, S_3, S_4 : $\{\text{var, function, id, print, input, if, return}\} \notin \text{Follow}(P) = \{ \$ \}$
 En el estado S_6 : $\{\text{int, string, boolean}\} \notin \text{Follow}(T1) = \{ \text{id} \}$
 Y así sucesivamente con el resto de estados.

3.4 Errores

En las celdas vacías de cada fila se lanzan los siguientes errores:

$S_0, S_4, S_7, S_{14}, S_{85}, S_{86}, S_{96}, S_{97}$: Error 1: “Sentencia no válida”

S_1 : Error -1 : “No se pudo derivar la raíz”

$S_2, S_5, S_{12}, S_{15}, S_{37}, S_{98}$: Error 2: “Declaración incorrecta de variable”

$S_3, S_6, S_{13}, S_{19}, S_{38}, S_{51}, S_{64}, S_{65}, S_{75}, S_{76}, S_{80}, S_{81}, S_{82}, S_{84}, S_{87}, S_{90}, S_{91}, S_{93}$:
 Error 3: “Declaración incorrecta de función”

$S_8, S_{24}, S_{42}, S_{56}, S_{69}$: Error 4: “Sentencia print incorrecta”

$S_9, S_{25}, S_{43}, S_{57}, S_{70}$: Error 5: “Sentencia input incorrecta”

$S_{10}, S_{26}, S_{44}, S_{58}, S_{71}, S_{73}$: Error 6: “Sentencia condicional simple incorrecta”

$S_{11}, S_{27}, S_{28}, S_{45}$: Error 7: “Sentencia return incorrecta”

$S_{16}, S_{17}, S_{18}, S_{20}$: Error 8: “Tipo incorrecto”

$S_{21}, S_{23}, S_{39}, S_{52}, S_{66}$: Error 9: “Asignación incorrecta”

$S_{22}, S_{40}, S_{41}, S_{53}, S_{54}, S_{55}, S_{63}, S_{67}, S_{68}, S_{77}$: Error 10: “Llamada a función incorrecta”

$S_{29}, S_{30}, S_{31}, S_{32}, S_{33}, S_{34}, S_{35}, S_{36}, S_{46}, S_{47}, S_{48}, S_{49}, S_{50}, S_{59}, S_{60}, S_{61}, S_{62}, S_{74}$:
 Error 11: “Expresión incorrecta”

$S_{72}, S_{78}, S_{79}, S_{83}, S_{88}, S_{89}, S_{92}, S_{94}, S_{95}, S_{99}$: Error 12 “Sentencia condicional compuesta incorrecta”

3.5 Tabla de Decisión

[illegible]

4 Diseno del Analizador Semántico

4.1 Esquema de Traducción

0. $P' \rightarrow \{TSG = creaTS(); DesplG = 0; TS_actual = TSG;$
 $P.func = false\} P \{liberaTS(TSG)\}$
1. $P \rightarrow D \{ P_1.func = P.func \} P_1$
2. $P \rightarrow F \{ P_1.func = P.func \} P_1$
3. $P \rightarrow \{S.func = false\} S \{P_1 = P.func\} P_1$
4. $D \rightarrow var \{zona_decl = true\} T id ; \{InsertarTipoTS(id.posi, T.tipo)$
 if($TS_actual=TS$) then
 InsertarDespl($id.posi, desplG$)
 $desplG = desplG + T.tamano$
 else
 InsertarDespl($id.posi, desplL$)
 $desplL = desplL + T.tamano$
 $zona_decl=false\}$
5. $T \rightarrow int \{T.tipo=entero; T.tamano = 1\}$
6. $T \rightarrow string \{T.tipo=cadena; T.tamano = 64\}$
7. $T \rightarrow boolean \{T.tipo=logico; T.tamano = 1\}$
8. $F \rightarrow function \{zona_decl =true\} T_1 id (\{TSL = creaTS();desplL=0;$
 $TS_actual = TSL\} A)\{InsertaTipoTSG(id.posi,A.tipo \rightarrow T.tipo)\};$
 $zona_decl = false\} \{ \{ C.func = true\} C \} \{if (C.tipoRet \neq T.tipo)$
 then error(2);
 $TS_actual = TSG; LiberarTS(TSL)\}$
9. $T_1 \rightarrow \lambda \{T.tipo = tipo_vacio\}$
10. $T_1 \rightarrow T \{T_1.tipo = T.tipo\}$
11. $A \rightarrow T id \{InsertarTipoTS(id.posi, T.tipo); InsertarDesplTS(id.posi, desplL);$
 $desplL = desplL + T.tamano\} K \{A.tipo = if(K.tipo = tipo_vacio)$
 then $T.tipo$
 else
 $T.tipo \times K.tipo\}$
12. $A \rightarrow \lambda \{A.tipo = tipo_vacio\}$
13. $K \rightarrow \lambda \{K.tipo = tipo_vacio\}$
14. $K \rightarrow , T id \{InsertarTipoTS(id.posi, T.tipo);$
 InsertarDesplTS($id.posi, desplL$); $desplL = desplL + T.tamano\} K_1$
 $\{K.tipo = if(K_1.tipo = tipo_vacio)$
 then $T.tipo$
 else
 $T.tipo \times K.tipo\}$
15. $C \rightarrow D \{C_1.func = C.func\} C_1 \{C.tipoRet = C_1.tipoRet\}$
16. $C \rightarrow \{S.func = C.func\} S \{C_1.func = C.func\} C_1 \{C.tipoRet =$
 if($S.tipoRet = C_1.tipoRet$) then
 $S.tipoRet$
 else if($S.tipoRet = tipo_vacio$) then
 $C_1.tipoRet$
 else if($C_1.tipoRet = tipo_vacio$) then
 $S.tipoRet$
 else
 error(2)}
17. $C \rightarrow \lambda \{C.tipoRet = tipo_vacio\}$
18. $S \rightarrow id L E ; \{S.tipo = if(BuscaTipoTS(id.posi) = E.tipo)$
 AND ($E.tipo \neq tipo_error$)
 then $tipo_ok$
 else
 error(3)
 $S.tipoRet = tipo_vacio\}$

```

19. S → id (M); {S.tipo = if (BuscaTipoTS(id.posi) = M.tipo → t)
                        then tipo_ok
                        else
                            error(4)
                        S.tipoRet = tipo_vacio}
20. S → print (E); {S.tipo = if (E.tipo = entero || E.tipo = cadena)
                        then tipo_ok
                        else
                            error(4)
                        S.tipoRet = tipo_vacio}
21. S → input(id); {S.tipo =
    if (BuscaTipoTS(id.posi) = entero || BuscaTipoTS(id.posi).tipo = cadena))
        then tipo_ok
    else
        error(4)
    S.tipoRet = tipo_vacio}
22. S → if (E) {S1.func = S.func} S1 {S.tipo =
    if (E.tipo = logico) then S1.tipo
    else
        error(5)
    S.tipoRet = S1.tipoRet}
23. S → return X; {S.tipo = if (S.func) then
    if (X.tipo != tipo.error) then tipo_ok
    else
        error(6)
    else
        error(1)
    S.tipoRet = X.tipoRet}
24. L → |= {}
25. L → = {}
26. M → EQ {M.tipo =
    if (E.tipo != tipo_error
    AND Q.tipo != tipo_error)
        then if (Q.tipo == tipo_vacio)
            then E.tipo
            else
                E.tipo x Q.tipo
        else
            error(7)}
27. M → λ {M.tipo = tipo_vacio}
28. Q → λ {Q.tipo = tipo_vacio}
29. Q → ,EQ1 {Q.tipo =
    if (E.tipo != tipo_error
    AND Q.tipo != tipo_error)
        then if (Q.tipo == tipo_vacio)
            then NuevaPila(E.tipo)
            else Q.tipo.push(E.tipo)
        else
            error(7)}

```

```

30. S1 → { {S2.func = S1.func} S2}
      {G.func=S1.func} G {S1.tipo =
                        if(S2.tipo != tipo_error)
                        if(G.tipo != tipo_error)
                        then S2.tipo
                        else error(9)
                        else error(8);
      S1.tipoRet = if (S2.tipoRet = G.tipoRet OR G.tipoRet = tipo_vacio)
                  then S2.tipoRet
                  else error(10)}
31. S1 → {S.func=S1.func} S {S1.tipo=S.tipo; S1.tipoRet = S.tipoRet}
32. G → else{ {S2.func=G.func} S2}{G.tipo=S2.tipo;
      G.tipoRet = S2.tipoRet}
33. G → λ {G.tipo = tipo_vacio; G.tipoRet = tipo_vacio}
34. X → E {X.tipo = E.tipo}
35. X → λ {X.tipo = tipo_vacio}
36. E → E1 < U {E.tipo = if(E1.tipo = U.tipo = entero)
                  then logico
                  else
                  error(11)}
37. E → U {E.tipo = U.tipo}
38. U → U1 + R {U.tipo = if(U1.tipo = R.tipo = entero)
                  then entero
                  else
                  error(11)}
39. U → R {U.tipo = R.tipo}
40. R → !V {R.tipo = if(V.tipo = logico) then logico
            else error(11)}
41. R → V {R.tipo = V.tipo}
42. V → (E) {V.tipo = E.tipo}
43. V → id {V.tipo = BuscaTS(id.posi)}
44. V → id(M) {S.tipo = if(BuscaTipoTS(id.posi) = M.tipo → t)
                then t
                else
                error(4)}
45. V → ent {V.tipo = entero}
46. V → cadena {V.tipo = cadena}
47. S2 → {S.func = S2.func} S {S'2.func = S2.func} S'2 {
      S2.tipo =
      if(S.tipo != tipo_error) then S2.tipo
      else
      error(12);
      S2.tipoRet = if (S.tipoRet = tipo_vacio) then
      S'2.tipoRet
      else if (S'2.tipoRet = tipo_vacio) then
      S.tipoRet
      else error(13) }
48. S2 → {S.func = S2.func} S {S2.tipo = S.tipo;
      S2.tipoRet = S.tipoRet}
49. P → λ {}

```


4.2 Implementación del EdT

0. $P' \rightarrow MM_1 P \{ liberaTS(TSG) \}$
1. $P \rightarrow D P_1 \{ Aux[ntope].tipoRet = Aux[tope].tipoRet \}$
2. $P \rightarrow F P_1 \{ Aux[ntope].tipoRet = Aux[tope].tipoRet \}$
3. $P \rightarrow S P_1 \{ Aux[ntope].tipoRet = \text{if}(Aux[tope-1].tipoRet = tipo_vacio) \text{ then } Aux[tope].tipoRet$
 $\text{else error}(1) \}$
4. $D \rightarrow \text{var } MM_2 T \text{ id } MM_3 ; \{ InsertarTipoTS(Aux[tope-2].posi, Aux[tope-3].tipo)$
 $\text{if}(TS_actual = TSG) \text{ then }$
 $\text{InsertarDespl}(Aux[tope-1].posi, desplG)$
 $desplG = desplG + Aux[tope-3].tamano$
 else
 $\text{InsertarDespl}(Aux[tope-2].posi, desplL)$
 $desplL = desplL + Aux[tope-3] \}$
5. $T \rightarrow \text{int } \{ Aux[ntope].tipo = entero; Aux[ntope].tamano = 1 \}$
6. $T \rightarrow \text{string } \{ Aux[ntope].tipo = cadena; Aux[ntope].tamano = 64 \}$
7. $T \rightarrow \text{boolean } \{ Aux[ntope].tipo = logico; Aux[ntope].tamano = 1 \}$
8. $F \rightarrow \text{function } MM_3 T_1 \text{ id } MM_4 (A) MM_5 \{ C \}$
 $\{ \text{if}(Aux[tope-1].tipoRet \neq Aux[tope-9].tipo) \text{ then error}(2);$
 $TS_actual = TSG; LiberarTS(TSL) \}$
9. $T_1 \rightarrow \lambda \{ Aux[ntope].tipo = tipo_vacio \}$
10. $T_1 \rightarrow T \{ Aux[ntope].tipo = Aux[tope].tipo \}$
11. $A \rightarrow T \text{ id } MM_6 K \{ Aux[ntope].tipo = \text{if}(Aux[tope].tipo = tipo_vacio)$
 $\text{then } Aux[tope-3].tipo \}$
 else
 $Aux[tope].tipo.push(Aux[tope-3].tipo) \}$
12. $A \rightarrow \lambda \{ Aux[ntope].tipo = tipo_vacio \}$
13. $K \rightarrow \lambda \{ Aux[ntope].tipo = tipo_vacio \}$
14. $K \rightarrow , T \text{ id } MM_7 K_1 \{ Aux[ntope].tipo = \text{if}(Aux[tope].tipo = tipo_vacio)$
 $\text{then NuevaPila}(Aux[tope-2].tipo)$
 else
 $Aux[tope].tipo.push(Aux[tope-4].tipo) \}$
15. $C \rightarrow D C_1 \{ Aux[ntope].tipoRet = Aux[tope].tipoRet \}$
16. $C \rightarrow S C_1 \{ Aux[ntope].tipoRet =$
 $\text{if}(Aux[tope-1].tipoRet = Aux[tope].tipoRet) \text{ then }$
 $Aux[tope-1].tipoRet$
 $\text{else if}(Aux[tope-1].tipoRet = tipo_vacio) \text{ then }$
 $Aux[tope-1].tipoRet$
 $\text{else if}(Aux[tope].tipoRet = tipo_vacio) \text{ then }$
 $Aux[tope-1].tipoRet \}$
 else
 $\text{error}(2) \}$
17. $C \rightarrow \lambda \{ Aux[ntope].tipoRet = tipo_vacio \}$
18. $S \rightarrow \text{id } L E ; \{ Aux[ntope].tipo =$
 $\text{if}(BuscaTipoTS(Aux[tope-3].posi) = (Aux[tope-1].tipo)$
 $\text{AND } (Aux[tope-1].tipo \neq tipo_error)) \text{ then }$
 $tipo_ok$
 else
 $\text{error}(3)$
 $Aux[ntope].tipoRet = tipo_vacio \}$
19. $S \rightarrow \text{id } (M) ; \{ Aux[ntope].tipo =$
 $\text{if}(BuscaTipoTS(Aux[tope-4].posi) = \text{ParFunc}(Aux[tope-2].tipo, t)$
 $\text{then } tipo_ok$
 $\text{else error}(4)$
 $Aux[ntope].tipoRet = tipo_vacio \}$

```

20. S → print (E) ; {Aux[ntope].tipo =
    if(Aux[tope-2].tipo = entero || Aux[tope-2].tipo = cadena)
        then tipo_ok
    else error(4)
    Aux[ntope].tipoRet = tipo_vacio}
21. S → input(id); {Aux[ntope].tipo =
    if(BuscaTipoTS(Aux[tope-2].posi = entero
        || Aux[tope-2].tipo = cadena)) then tipo_ok
    else error(4)
    Aux[ntope].tipoRet = tipo_vacio}
22. S → if(E) S1 {Aux[ntope].tipo =
    if(Aux[tope-2].tipo = logico) then Aux[tope].tipo
    else error(5)
    Aux[ntope].tipoRet = tipo_vacio}
23. S → return X ; {Aux[ntope].tipo =
    if(Aux[tope-1].tipo != tipo.error) then tipo_ok
    else error(6)}
24. L → |= {}
25. L → = {}
26. M → EQ {Aux[ntope].tipo =
    if(Aux[tope-1].tipo != tipo_error
    AND Aux[tope].tipo != tipo_error)
        then if(Aux[tope].tipo = tipo_vacio)
            then Aux[tope-1].tipo
        else
            Aux[tope].tipo.push(Aux[tope-1].tipo)
    else
        error(7)}
27. M → λ {Aux[ntope].tipo = tipo_vacio}
28. Q → λ {Aux[ntope].tipo = tipo_vacio}
29. Q → , E Q1 {if(Aux[tope-1].tipo != tipo_error
    AND Aux[tope].tipo != tipo_error)
        then if(Aux[tope].tipo = tipo_vacio)
            then NuevaPila(Aux[tope-1].tipo)
            else Aux[tope].tipo.push(Aux[tope-1].tipo)
        else
            error(7)}
30. S1 → { S2 } G { Aux[ntope].tipo =
    if(Aux[tope-2].tipo != tipo_error)
        if(Aux[tope].tipo != tipo_error)
            then Aux[tope-2].tipo
        else error(9)
    else error(8);
    Aux[ntope].tipoRet =
    if (Aux[tope-2].tipoRet = Aux[tope].tipoRet
    OR Aux[tope].tipoRet = tipo_vacio)
        Aux[tope-2].tipoRet
    else
        error(10)}
31. S1 → S {Aux[ntope].tipo = Aux[tope].tipo;
    Aux[ntope].tipoRet = Aux[tope].tipoRet}
32. G → else {S2} {Aux[ntope].tipo = Aux[tope-1].tipo;
    Aux[ntope].tipoRet = Aux[tope-1].tipoRet}
33. G → λ {Aux[ntope].tipo = tipo_vacio;
    Aux[ntope].tipoRet = tipo_vacio}
34. X → E {Aux[ntope].tipo = Aux[tope].tipo}
35. X → λ {Aux[ntope].tipo = tipo_vacio}

```

```

36. E → E1 < U {Aux[ntope].tipo =
                    if(Aux[tope-2].tipo = Aux[tope].tipo = entero)
                      then logico
                    else error(11)}
37. E → U {Aux[ntope].tipo = Aux[tope].tipo}
38. U → U1 + R {Aux[ntope].tipo =
                  if(Aux[tope-2].tipo = Aux[tope-2].tipo = entero)
                    then entero
                  else error(11)}
39. U → R {Aux[ntope].tipo = Aux[tope].tipo}
40. R → !V {Aux[ntope].tipo = if(Aux[tope].tipo = logico) then
                    logico
                    else error(11)}
41. R → V {Aux[ntope].tipo = Aux[tope].tipo}
42. V → (E) {Aux[ntope].tipo = Aux[tope-1].tipo}
43. V → id {Aux[ntope].tipo = BuscaTipoTS(Aux[tope].posi)}
44. V → id(M) {Aux[ntope].tipo =
                if(BuscaTipoTS(Aux[tope-3].posi) = ParFunc(Aux[tope-1].tipo, t)) then t
                else error(4)}
45. V → ent {Aux[ntope].tipo = entero}
46. V → cadena {Aux[ntope].tipo = cadena}
47. S2 → S S'2 {Aux[ntope] = if(Aux[tope-1].tipo != tipo_error) then
                    Aux[tope].tipo
                    else
                        error(12);
                    Aux[ntope] = if(Aux[tope-1].tipoRet = tipo_vacio) then
                        Aux[tope].tipoRet
                    else if(Aux[tope].tipoRet = tipo_vacio) then
                        Aux[tope-1].tipoRet
                    else error(13);}
48. S2 → S {Aux[ntope].tipo = Aux[tope].tipo;
              Aux[ntope].tipoRet = Aux[tope].tipoRet}
49. P → λ {Aux[ntope].tipoRet = tipo_vacio}
50. MM1 → λ {TSG = creaTS();
              desplG = 0;
              TSA = TSG}
51. MM2 → λ {zona_decl = true}
52. MM3 → λ {zona_decl = true}
53. MM4 → λ {TSL = creaTS();
              desplL = 0;
              TSA = TSL}
54. MM5 → λ {InsertarTipoTS(Aux[tope-4].posi, Aux[tope-1].tipo) ;
              InsertarNArgsTS(Aux[tope-4].posi, Aux[tope-1].NArgs) ;
              for i in NArgs :
                  InsertarTipoArgsTS(Aux[tope-4].posi, Aux[tope-1].tipoLista(i));
                  InsertarEtiquetaTS(Aux[tope-4].posi, Aux[tope-4].lexema + Aux[tope-4].posi);
                  InsertarTipoDevueltoTS(Aux[tope-4].posi, Aux[tope-5].tipo)}
55. MM6 → λ {InsertarTipoTS(Aux[tope].posi, Aux[tope-1].tipo) ;
              InsertarDesplTS(Aux[tope].posi, desplL);
              desplL = desplL + Aux[tope-1].tamano}
56. MM7 → λ {InsertarTipoTS(Aux[tope].posi, Aux[tope-1].tipo) ;
              InsertarDesplTS(Aux[tope].posi, desplL);
              desplL = desplL + Aux[tope-1].tamano}
57. MM8 → λ {zona_decl = false}

```

De forma que hemos tenido que transformar el EdT para que en vez de usar atributos heredados, lo cual complica bastante la implementación, utilice sólo sintetizados. Hemos transformado el atributo heredado func al atributo sintetizado tipoRet.

También hemos tenido que modificar la gramática del sintáctico añadiendo los marcadores MM_i y sus correspondientes reglas lambda para poder implementar las acciones con efectos laterales.

Ej:
 $D \rightarrow \text{var } \{zdecl:=true\} \text{ T id; } \{otras \text{ acciones}\}$
 Lo transformamos en:
 $D \rightarrow \text{var } MM \text{ T id; } \{otras \text{ acciones}\}$
 $MM \rightarrow \text{lambda } \{zdecl:=true\}$

4.3 Gramática y Autómata Final del A. Sintáctico

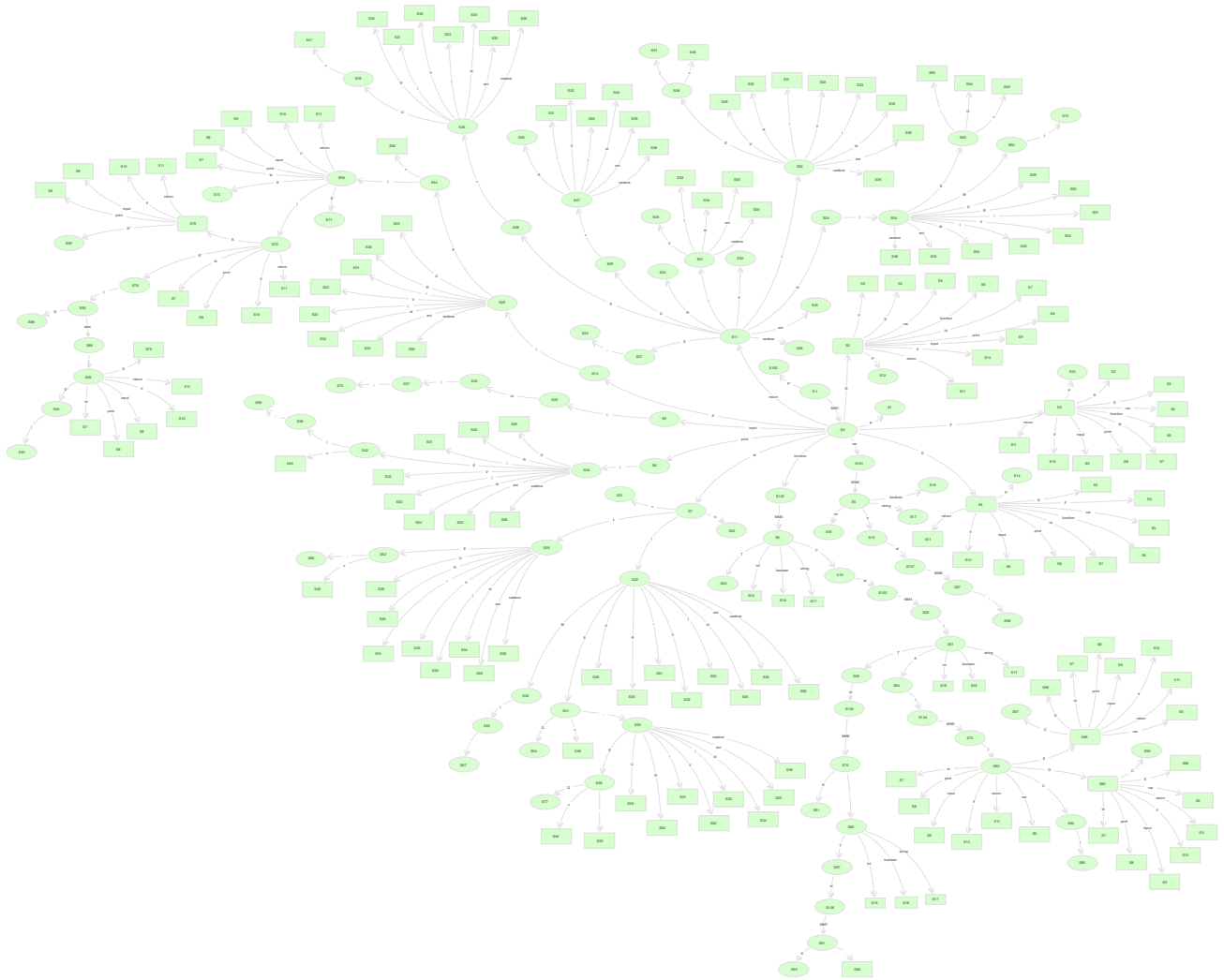
Terminales = { ; { } id ent cadena () + < ! = | = var int
 boolean string print input , return function if else }
NoTerminales = { P1 P D T F T1 A K C S L M Q S1 G X E U R V
 S2 M1 M2 M3 M4 M5 M6 M7 M8 }
Axioma = P1
Producciones = {
 $P1 \rightarrow M1 P$
 $P \rightarrow D P$
 $P \rightarrow F P$
 $P \rightarrow S P$
 $D \rightarrow \text{var } M2 \text{ T id } M8 ;$
 $T \rightarrow \text{int}$
 $T \rightarrow \text{string}$
 $T \rightarrow \text{boolean}$
 $F \rightarrow \text{function } M3 \text{ T1 id } M4 (A) M5 \{ C \}$
 $T1 \rightarrow \text{lambda}$
 $T1 \rightarrow T$
 $A \rightarrow T \text{ id } M6 K$
 $A \rightarrow \text{lambda}$
 $K \rightarrow \text{lambda}$
 $K \rightarrow , \text{ T id } M7 K$
 $C \rightarrow D C$
 $C \rightarrow S C$
 $C \rightarrow \text{lambda}$
 $S \rightarrow \text{id } L E ;$
 $S \rightarrow \text{id } (M) ;$
 $S \rightarrow \text{print } (E) ;$
 $S \rightarrow \text{input } (\text{id}) ;$
 $S \rightarrow \text{if } (E) S1$
 $S \rightarrow \text{return } X ;$
 $L \rightarrow |=$
 $L \rightarrow =$
 $M \rightarrow E Q$
 $M \rightarrow \text{lambda}$
 $Q \rightarrow \text{lambda}$
 $Q \rightarrow , E Q$

$S1 \rightarrow \{ S2 \} G$
 $S1 \rightarrow S$
 $G \rightarrow else \{ S2 \}$
 $G \rightarrow lambda$
 $X \rightarrow E$
 $X \rightarrow lambda$
 $E \rightarrow E < U$
 $E \rightarrow U$
 $U \rightarrow U + R$
 $U \rightarrow R$
 $R \rightarrow ! V$
 $R \rightarrow V$
 $V \rightarrow (E)$
 $V \rightarrow id$
 $V \rightarrow id (M)$
 $V \rightarrow ent$
 $V \rightarrow cadena$
 $S2 \rightarrow S S2$
 $S2 \rightarrow S$
 $P \rightarrow lambda$
 $M1 \rightarrow lambda$
 $M2 \rightarrow lambda$
 $M3 \rightarrow lambda$
 $M4 \rightarrow lambda$
 $M5 \rightarrow lambda$
 $M6 \rightarrow lambda$
 $M7 \rightarrow lambda$
 $M8 \rightarrow lambda$
 $\}$

Dando lugar a modificaciones en el Autómata (y en consecuencia en la Tabla de Decisión):

Donde :

$S_{-1} = \{ P2 \rightarrow \bullet P1, P1 \rightarrow \bullet MM1 P, P \rightarrow \bullet SP, MM1 \rightarrow \bullet \}$
 $S_1 = \{ P1 \rightarrow MM1 P \bullet \}$ [Ahora no se ACEPTA, solo se REDUCE]
 $S_{100} = \{ P2 \rightarrow P1 \bullet \}$ [Ahora este es el estado que ACEPTA]
 $S_{101} = \{ D \rightarrow var \bullet MM2 T id MM8 ;, MM2 \rightarrow \bullet \}$
 $S_{102} = \{ F \rightarrow function \bullet MM3 T1 id MM4 (A) MM5 \{ C \}, MM3 \rightarrow \bullet \}$
 $S_{103} = \{ F \rightarrow function MM3 T1 id \bullet MM4 (A) MM5 \{ C \}, MM4 \rightarrow \bullet \}$
 $S_{104} = \{ D \rightarrow F \rightarrow function \bullet MM3 T1 id MM4 (A) \bullet MM5 \{ C \}, MM5 \rightarrow \bullet \}$
 $S_{105} = \{ A \rightarrow T id \bullet MM6 K, MM6 \rightarrow \bullet \}$
 $S_{106} = \{ D \rightarrow , T id \bullet MM7 K, MM7 \rightarrow \bullet \}$
 $S_{107} = \{ D \rightarrow var MM2 T id \bullet MM8 ;, MM8 \rightarrow \bullet \}$



4.4 Errores

Error 1: "RETURN fuera de funcion."

Error 2: "El tipo devuelto no coincide con el declarado en la funcion."

Error 3: "Asignacion incorrecta.";

Error 4: "Incoherencia entre parametros formales y actuales en la llamada a funcion.";

Error 5: "La condicion del IF no es de tipo logico.";

Error 6: "Error en la sentencia del RETURN.";

Error 7: "Error al definir los parametros de llamada de una funcion.";

Error 8: "Error en el cuerpo del IF.";

Error 9: "Error en el cuerpo del ELSE.";

Error 10: "No concuerdan los RETURN de las sentencias IF-ELSE.";

Error 11: "Tipos incompatibles entre operandos y operadores.";

Error 12: "Error en el cuerpo del IF-ELSE.";

Error 13: "Error en el RETURN en el cuerpo del IF-ELSE.";

5 Anexo de Pruebas

Error 1:

```
1 var int a;
2 var int b;
3 a = 33333;
4 b = a;
5 if (a < b) b = 1;
6 if (b < a) b = 8;
7 a = a + b;
8 print (a);
9 print (b);
```

> Error Lexico: Numero fuera de rango. Linea: 3

Error 2:

```
1 var string texto;
2 function pideTexto ()
3 {
4     print ('Introduce un texto');
5     input (texto);
6 }
7 function imprime (string msg,)
8 {
9     print (msg);
10 }
11 pideTexto();
12 var string textoAux;
13 textoAux = texto;
14 imprime (textoAux);
```

> Error Sintactico: Declaracion incorrecta de funcion. Linea: 7

Error 3:

```
1 var int a;
2 var int b;
3 a = 3;
4 b = a;
5 var boolean c;
6 c = a < b;
7 if (c) {
8     b = 1;
9 } else {
10 c = b < a;
11 if (c) b = 4;
12 a = a + b;
13 print (a);
14 print (b);
```

> Error Sintactico: Sentencia condicional compuesta incorrecta. Linea: 14

Error 4:

```
1 var int a;
2 var int b;
3 var int c;
4 print ('Introduce el primer operando');
5 input (a);
6 print ('Introduce el segundo operando');
7 input (b);
8 function int suma (int num1, int num2)
9 {
10     var int res;
11     res = num1+num2;
12     return res;
13 }
14 c = suma (a, b, c);
15 print (c);
```

```
> Error Semantico: Incoherencia entre parametros formales y actuales en la llamada a
funcion. Linea: 14
```

Error 5:

```
1 var boolean booleano;
2 function boolean bisiestro (int a)
3 { var int bis;
4   print ('Es bisiestro?');
5   input(bis);
6   return (!(a + 4 < 0));
7 }
8 return booleano;
```

```
> Error Semantico: RETURN fuera de funcion. Linea: 8
```


Prueba 1 Correcta:

```
1 var boolean booleano;
2 function boolean bisiestro (int a)
3 { var int bis;
4   print ('Es bisiestro?');
5   input(bis);
6   return (!(a + 4 < 0));
7 }
8 function int dias (int m, int a)
9 {
10  var int dd;
11  print ('di cuantos dias tiene el mes ');
12  print (m);
13  input(dd);
14  if (bisiestro(a)) dd = dd + 1;
15  return dd;
16 }
17 function boolean esFechaCorrecta (int d, int m, int a)
18 {
19   return !(d < dias (m, a));
20 }
21 function demo ()
22 {
23
24   if (esFechaCorrecta(25, 10, 2018)) print (9999);
25   return;
26 }
27 var string A_A_A_;
28 demo();
```

Tokens:

<DEC, >
<TipoVarLOG, >
<ID, G0>
<PuntoComa, >
<DECFunc, >
<TipoVarLOG, >
<ID, G1>
<ParentesisAbrir, >
<TipoVarENT, >
<ID, L0>
<ParentesisCerrar, >
<CorcheteAbrir, >
<DEC, >
<TipoVarENT, >
<ID, L1>
<PuntoComa, >
<Print, >
<ParentesisAbrir, >
<CAD, " Es bisiestro?">
<ParentesisCerrar, >
<PuntoComa, >

```

<Input , >
<ParentesisAbrir , >
<ID, L1>
<ParentesisCerrar , >
<PuntoComa , >
<Return , >
<ParentesisAbrir , >
<NOT, >
<ParentesisAbrir , >
<ID, L0>
<SUMA, >
<ENT, 4>
<MENOR, >
<ENT, 0>
<ParentesisCerrar , >
<ParentesisCerrar , >
<PuntoComa , >
<CorcheteCerrar , >
<DECFunc, >
<TipoVarENT, >
<ID, G2>
<ParentesisAbrir , >
<TipoVarENT, >
<ID, L0>
<Coma, >
<TipoVarENT, >
<ID, L1>
<ParentesisCerrar , >
<CorcheteAbrir , >
<DEC, >
<TipoVarENT, >
<ID, L2>
<PuntoComa , >
<Print , >
<ParentesisAbrir , >
<CAD, "di cuantos dias tiene el mes ">
<ParentesisCerrar , >
<PuntoComa , >
<Print , >
<ParentesisAbrir , >
<ID, L0>
<ParentesisCerrar , >
<PuntoComa , >
<Input , >
<ParentesisAbrir , >
<ID, L2>
<ParentesisCerrar , >
<PuntoComa , >

```

<IF, >
 <ParentesisAbrir, >
 <ID, G1>
 <ParentesisAbrir, >
 <ID, L1>
 <ParentesisCerrar, >
 <ParentesisCerrar, >
 <ID, L2>
 <ASIG, >
 <ID, L2>
 <SUMA, >
 <ENT, 1>
 <PuntoComa, >
 <Return, >
 <ID, L2>
 <PuntoComa, >
 <CorcheteCerrar, >
 <DECFunc, >
 <TipoVarLOG, >
 <ID, G3>
 <ParentesisAbrir, >
 <TipoVarENT, >
 <ID, L0>
 <Coma, >
 <TipoVarENT, >
 <ID, L1>
 <Coma, >
 <TipoVarENT, >
 <ID, L2>
 <ParentesisCerrar, >
 <CorcheteAbrir, >
 <Return, >
 <NOT, >
 <ParentesisAbrir, >
 <ID, L0>
 <MENOR, >
 <ID, G2>
 <ParentesisAbrir, >
 <ID, L1>
 <Coma, >
 <ID, L2>
 <ParentesisCerrar, >
 <ParentesisCerrar, >
 <PuntoComa, >
 <CorcheteCerrar, >
 <DECFunc, >
 <ID, G4>
 <ParentesisAbrir, >

```

<ParentesisCerrar , >
<CorcheteAbrir , >
<IF , >
<ParentesisAbrir , >
<ID, G3>
<ParentesisAbrir , >
<ENT, 25>
<Coma, >
<ENT, 10>
<Coma, >
<ENT, 2018>
<ParentesisCerrar , >
<ParentesisCerrar , >
<Print , >
<ParentesisAbrir , >
<ENT, 9999>
<ParentesisCerrar , >
<PuntoComa, >
<Return , >
<PuntoComa, >
<CorcheteCerrar , >
<DEC, >
<TipoVarCAD, >
<ID, G5>
<PuntoComa, >
<ID, G4>
<ParentesisAbrir , >
<ParentesisCerrar , >
<PuntoComa, >

```

Tabla de símbolos:

Tabla Simbolos #2:

```

* LEXEMA: 'a'
+ Tipo: 'entero'
+ Despl: 0

* LEXEMA: 'bis'
+ Tipo: 'entero'
+ Despl: 1

```

Tabla Simbolos #3:

```

* LEXEMA: 'm'
+ Tipo: 'entero'
+ Despl: 0

* LEXEMA: 'a'

```

- + Tipo: 'entero '
- + Despl: 1

- * LEXEMA: 'dd'
- + Tipo: 'entero '
- + Despl: 2

Tabla Simbolos #4:

- * LEXEMA: 'd'
- + Tipo: 'entero '
- + Despl: 0

- * LEXEMA: 'm'
- + Tipo: 'entero '
- + Despl: 1

- * LEXEMA: 'a'
- + Tipo: 'entero '
- + Despl: 2

Tabla Simbolos #5:

Tabla Simbolos #1:

- * LEXEMA: 'booleano '
- + Tipo: 'logico '
- + Despl: 0

- * LEXEMA: 'bisiesto '
- + Tipo: 'funcion '
- + numParam: 1
- + TipoParam1: 'entero '
- + ModoParam1: 'Valor '
- + TipoRetorno: 'logico '
- + EtiqFuncion: 'bisiesto1 '

- * LEXEMA: 'dias '
- + Tipo: 'funcion '
- + numParam: 2
- + TipoParam1: 'entero '
- + ModoParam1: 'Valor '
- + TipoParam2: 'entero '
- + ModoParam2: 'Valor '
- + TipoRetorno: 'entero '
- + EtiqFuncion: 'dias2 '

- * LEXEMA: 'esFechaCorrecta '

```

+ Tipo: 'funcion '
+ numParam: 3
+ TipoParam1: 'entero '
+ ModoParam1: 'Valor '
+ TipoParam2: 'entero '
+ ModoParam2: 'Valor '
+ TipoParam3: 'entero '
+ ModoParam3: 'Valor '
+ TipoRetorno: 'logico '
+ EtiqFuncion: 'esFechaCorrecta3 '

```

```

* LEXEMA: 'demo'
+ Tipo: 'funcion '
+ numParam: 0
+ TipoRetorno: 'void '
+ EtiqFuncion: 'demo4'

```

```

* LEXEMA: 'A_A_A_'
+ Tipo: 'cadena'
+ Despl: 2

```

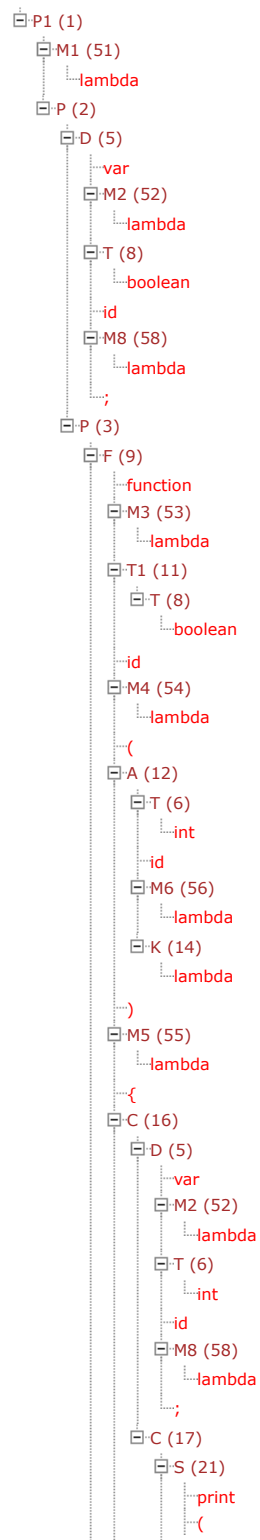
Parse a Derechas:

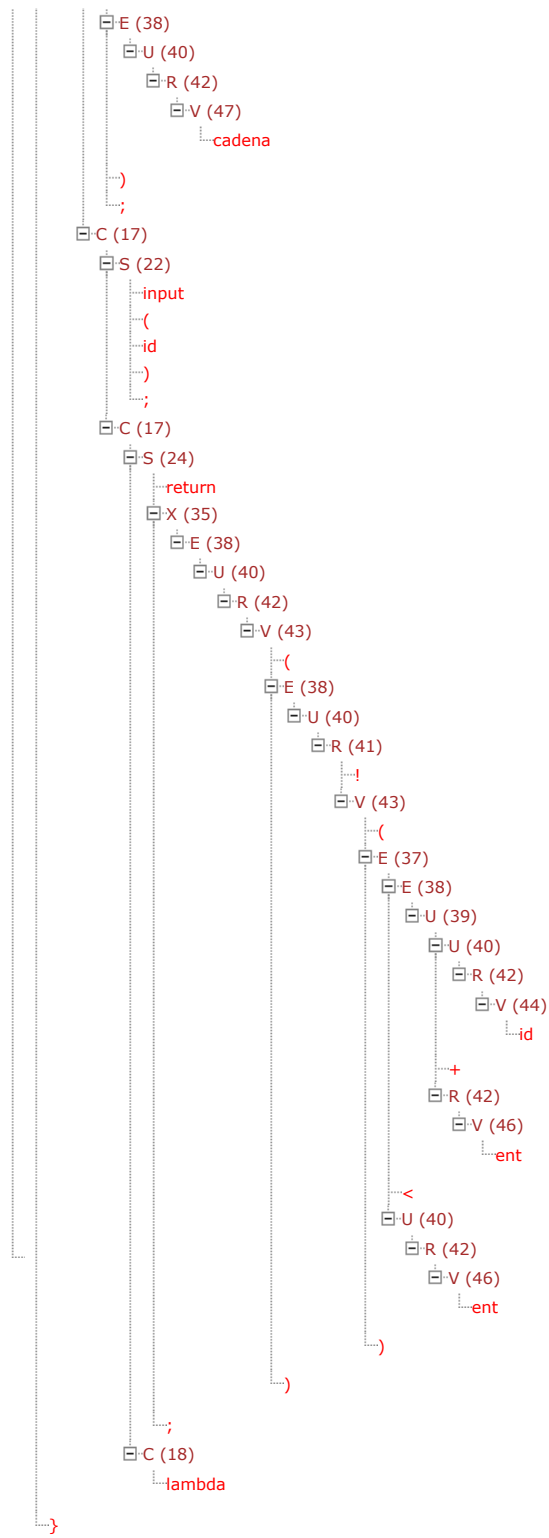
```

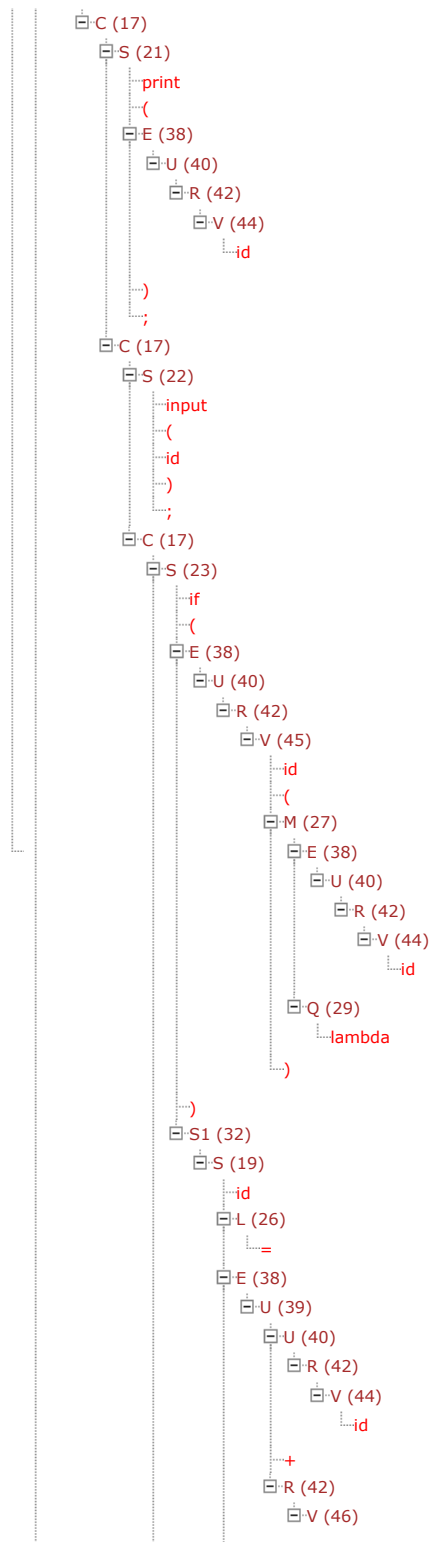
A 51 52 8 58 5 53 8 11 54 6 56 14 12 55 52 6 58 5 47 42 40 38 21 22 44 42 40 46 42 39
38 46 42 40 37 43 41 40 38 43 42 40 38 35 24 18 17 17 17 16 9 53 6 11 54 6 56 6 57 14
15 12 55 52 6 58 5 47 42 40 38 21 44 42 40 38 21 22 44 42 40 38 29 27 45 42 40 38 26
44 42 40 46 42 39 38 19 32 23 44 42 40 38 35 24 18 17 17 17 17 17 16 9 53 8 11 54 6
56 6 57 6 57 14 15 15 12 55 44 42 40 38 44 42 40 38 44 42 40 38 29 30 27 45 42 40 37
43 41 40 38 35 24 18 17 9 53 10 54 13 55 46 42 40 38 46 42 40 38 46 42 40 38 29 30 30
27 45 42 40 38 46 42 40 38 21 32 23 36 24 18 17 17 9 52 7 58 5 28 20 50 4 2 3 3 3 3 2
1

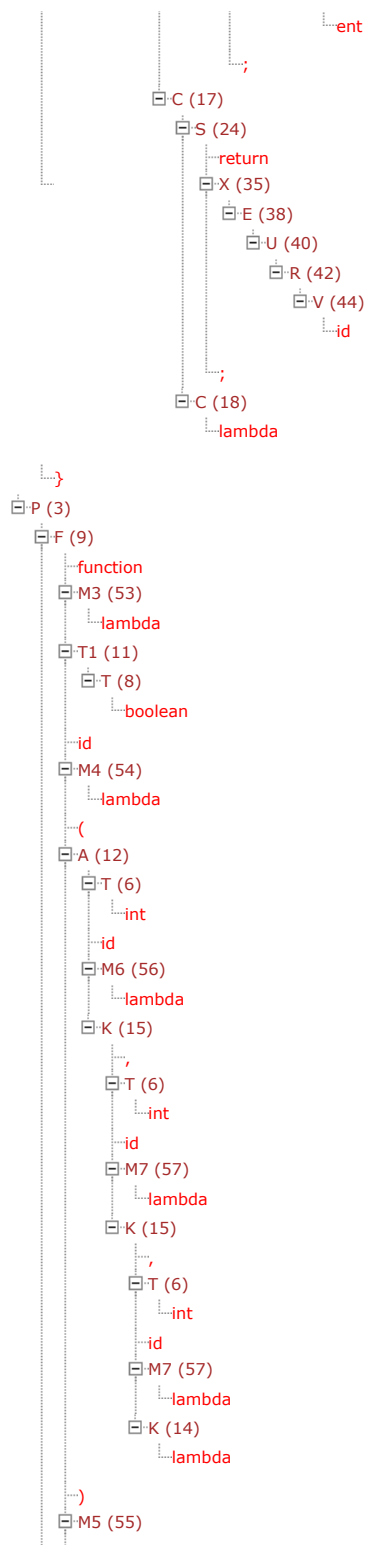
```

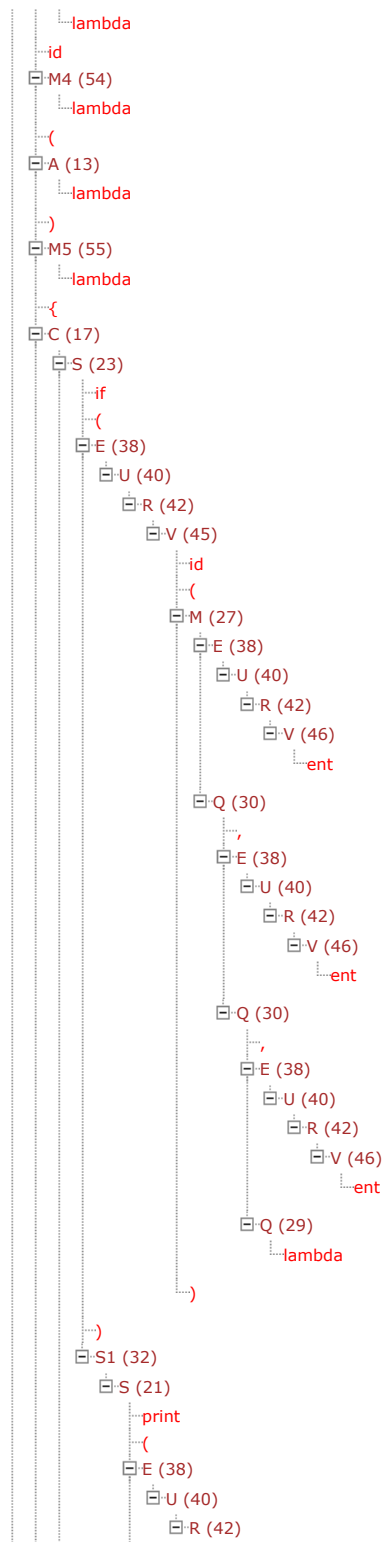
Árbol sintáctico:

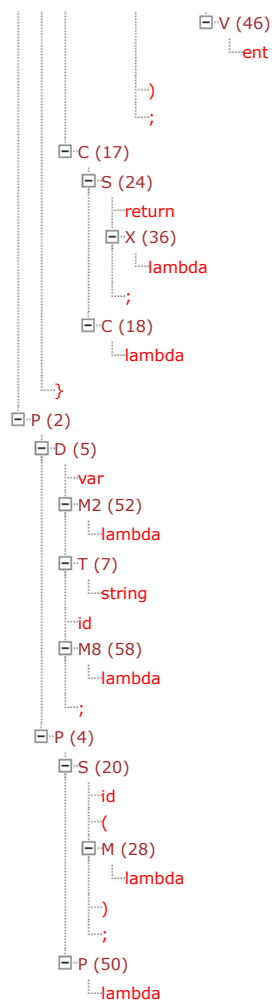












Fuente 2 Correcta:

```
1 var int a;
2 var int b;
3 a = 3;
4 b = a;
5   var boolean c;
6 c = a < b;
7 if (c) b = 1;
8 c = b < a;
9 if (c) b = 4;
10 a = a + b;
11 print (a);
12 print (b);
```

Fuente 3 Correcta:

```
1 var int a;
2 var int b;
3 var int c;
4 print ('Introduce el primer operando');
5 input (a);
6 print ('Introduce el segundo operando');
7 input (b);
8 function int suma (int num1, int num2)
9 {
10     var int res;
11     res = num1+num2;
12     return res;
13 }
14 c = suma (a, b);
15 print (c);
```

Fuente 4 Correcta:

```
1 var string texto;
2 function imprime (string msg)
3 {
4     print ('Mensaje introducido:');
5     print (msg);
6 }
7 function pideTexto ()
8 {
9     print ('Introduce un texto');
10    input (texto);
11 }
12 pideTexto();
13 imprime (texto);
```

Fuente 5 Correcta:

```
1 var string s;
2 var int uno;
3 var int UNO;
4 function int Factorial (int n)
5 {
6     if (n < 0) return 1;
7     return n + Factorial (n + 1);
8 }
9 var int For;
10 var int functional;
11 var int While;
12
13 function imprime (string s, string msg, int f)
14 {
15     print (s); print (msg); print (f);
16     return;
17 }
18 function string cadena (boolean log)
19 {
20     if (log)
21     {
22         imprime (s, 'hola', 33);
23         if (uno < UNO) return s;
24     }
25     else
26     {
27         return 'Fin';
28     }
29 }
30 s = 'El factorial ';
31
32 print (s);
33 print ('Introduce un numero.');
```

```
34 input (num);
35 var
36 boolean
37 booleano;
38 if (num < 0) print ('No existe el factorial de un negativo.');
```

```
39 For= Factorial (num);
```