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

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, - >

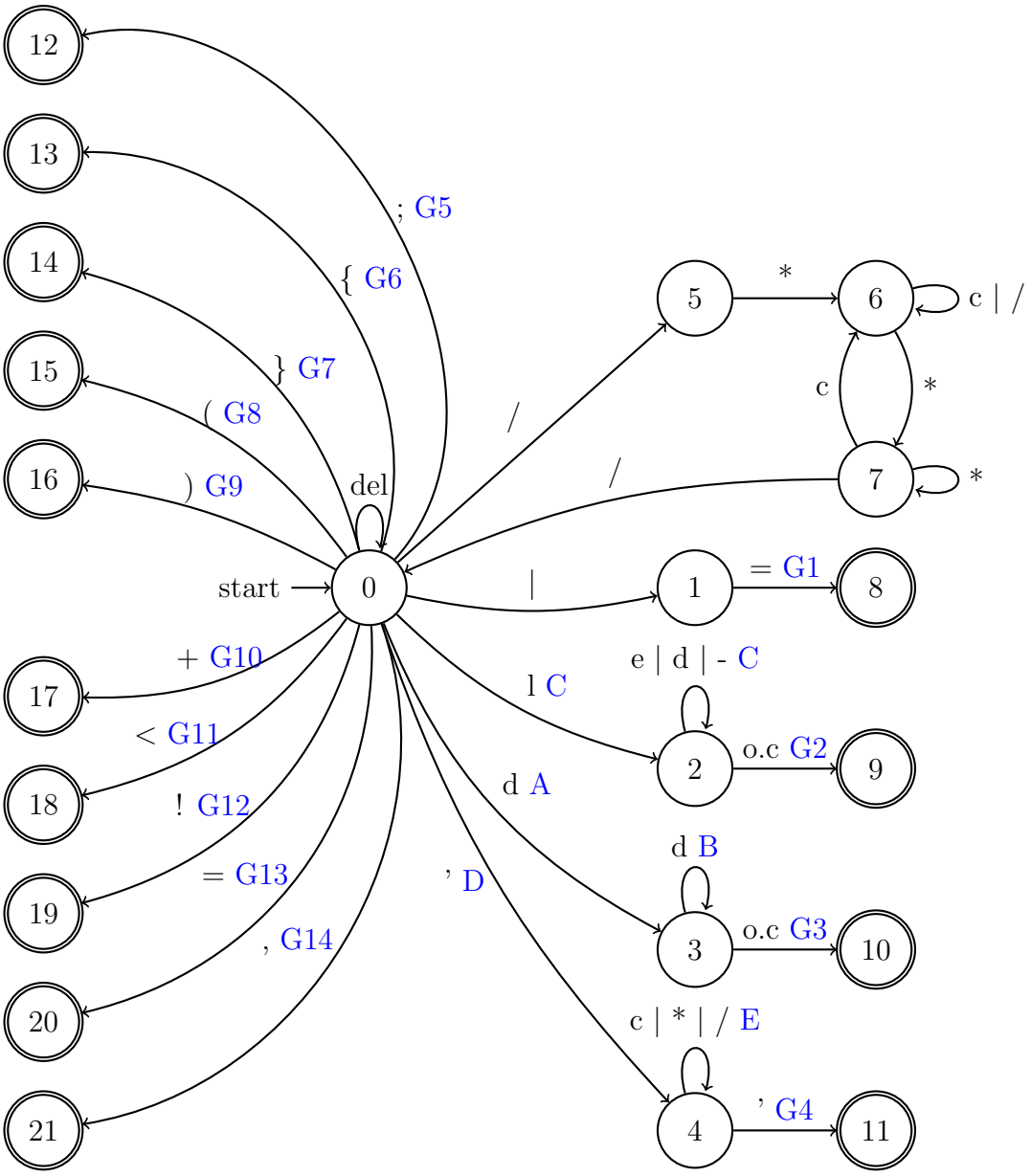
1.2 Gramática

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

$S = A$
$N = \{ A, B, C, D, E, F, G, H \}$
$T = \{ del, ;, \{, \}, (,), +, <, !, =, ,, l, d, ', /, \_, *, c \}$

$P:$
$A \rightarrow delA \mid ; \mid \{ \mid \} \mid (\mid) \mid + \mid < \mid ! \mid = \mid ,$
$A \rightarrow |B \mid lC \mid dD \mid 'E \mid /F$
$B \rightarrow =$
$C \rightarrow lC \mid dD \mid \_C \mid \lambda$
$D \rightarrow dD \mid \lambda$
$E \rightarrow cE \mid *E \mid /E \mid '$
$F \rightarrow *G$
$G \rightarrow cG \mid /G \mid *H$
$H \rightarrow /A \mid cG \mid *H$
$Donde, c = T - \{*, /\}$

1.3 Autómata Finito Determinista



1.4 Acciones Semánticas

```
Lee ∀ transicion menos o.c

C:  CONCAT()

G1: GEN_TOKEN(ASIGOR, −)

G2: if(lex ∈ 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()

G3: if(num >= 2^15) Error("Numero se sale del rango")
     else GEN_TOKEN(ENT, num)

G4: if(cont > 64) Error("Exceso de caracteres en la cadena")
      else GEN_TOKEN(CAD, lex)

G5: GEN_TOKEN(PuntoComa, −)

G6: GEN_TOKEN(CorcheteAbrir, −)

G7: GEN_TOKEN(CorcheteCerrar, −)

G8: GEN_TOKEN(ParentesisAbrir, −)

G9: GEN_TOKEN(ParentesisCerrar, −)

G10: GEN_TOKEN(SUMA, −)

G11: GEN_TOKEN(MENOR, −)

G12: GEN_TOKEN(NOT, −)
```

G_{13} : GEN_TOKEN(ASIG , −)

G_{14} : GEN_TOKEN(Coma , −)

Donde, palRes = {var, int, boolean, string, print, input, function, return, if, 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 simbolos).

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.

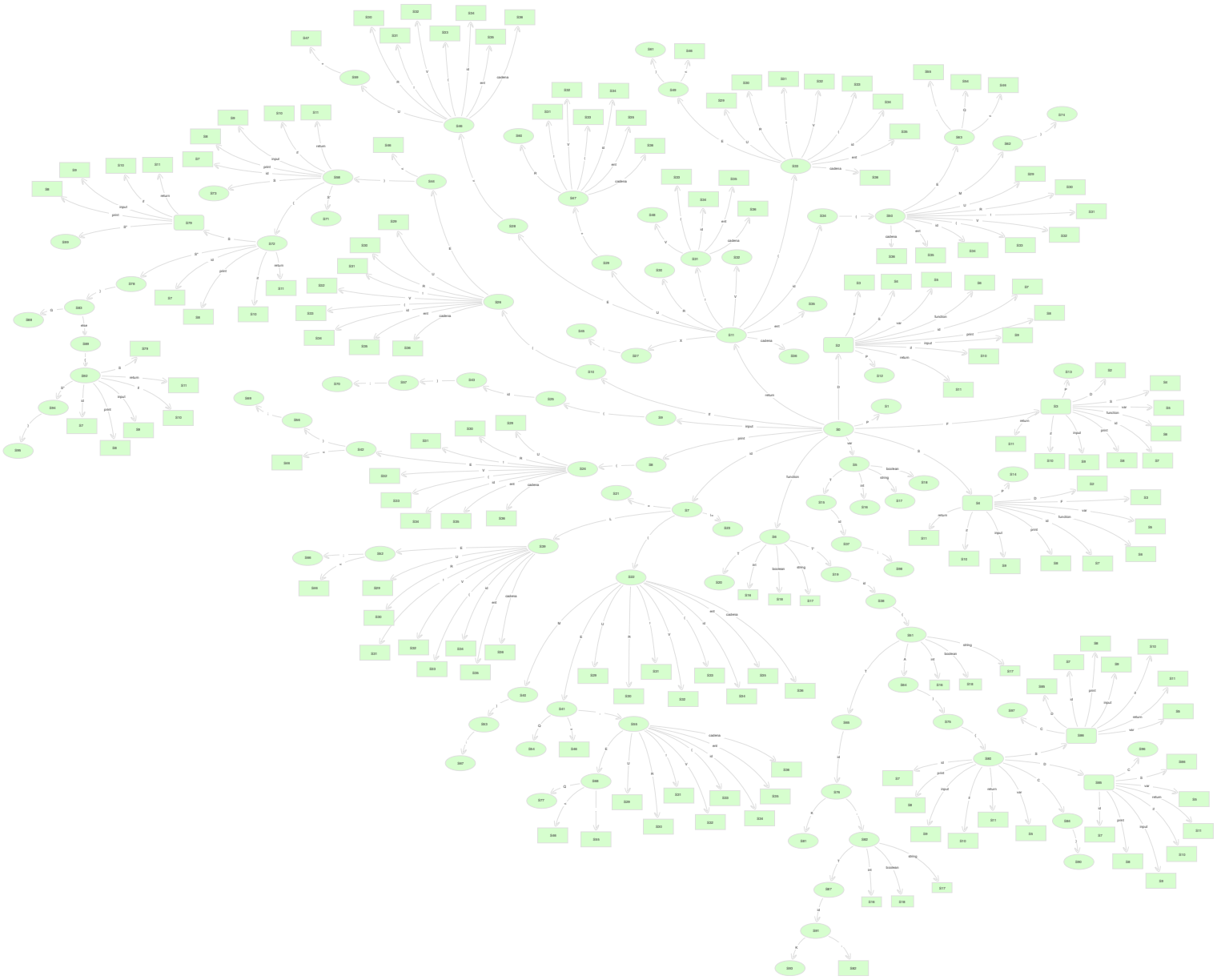
3 Diseño del Analizador Sintáctico

3.1 Gramática

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

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

3.2 Autómata Reconocedor de Prefijos Viables[1]



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

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

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

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 → α •, B → β •} $\subset S_x \Rightarrow$ 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 → α • b γ, C → β •} $\subset S_x \Rightarrow$ 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: {var, function, id, print, input, if, return} $\notin$ Follow(P) = { $ }
En el estado S_6: {int, string, boolean} $\notin$ Follow(T1) = { 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}, $S69$: Error 4: "Sentencia print incorrecta"

S_9, S_{25}, S_{43}, S_{57}, $S70$: Error 5: "Sentencia input incorrecta"

S_{10}, S_{26}, S_{44}, S_{58}, $S71$, $S73$: 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

	;	{	}	id	entero	cadena	(	)	+	<	!	=	|=	var	int	boolean	string	print	input	,	return	function	if	else	$	P	D	T	F	T'	A	K	C	S	L	M	Q	S'	G	X	E	U	R	V	S''
S0				d7										d5				d8	d9		d11	d6	d10		r49	1	2		3					4											
S1																									a																				
S2				d7										d5				d8	d9		d11	d6	d10		r49	12	2		3					4											
S3				d7										d5				d8	d9		d11	d6	d10		r49	13	2		3					4											
S4				d7										d5				d8	d9		d11	d6	d10		r49	14	2		3					4											
S5															d16	d18	d17							15																					
S6				r9											d16	d18	d17							20				19																	
S7							d22					d21	d23																					39											
S8							d24																																						
S9							d25																																						
S10							d26																																						
S11	r35			d34	d35	d36	d33				d31																																		
S12																									r1																				
S13																									r2																				
S14																									r3																				
S15				d37																																									
S16				r5																																									
S17				r6																																									
S18				r7																																									
S19				d38																																									
S20				r10																																									
S21				r25	d34	d35	r25		r25		r25																																		
S22					d34	d35	d36	d33	r27		d31																																			
S23				r24	r24	r24	r24	r24			r24																																		
S24					d34	d35	d36	d33			d31																																		
S25					d43																																								
S26					d34	d35	d36	d33			d31																																		
S27	d45																																											
S28	r34																																											
S29	r37																				r37																						
S30	r39								r39	r39	r39										r39																						
S31				d34	d35	d36	d33																																					
S32	r41								r41	r41	r41										r41																						
S33				d34	d35	d36	d33				d31																																		
S34	r43						d50	r43	r43	r43											r43																						
S35	r45							r45	r45	r45											r45																						
S36	r46							r46	r46	r46											r46																						
S37	d98																																											
S38							d51																																					
S39				d34	d35	d36	d33				d31																																	
S40								d53																																				
S41								r28		d46										d55																54								
S42								d56		d46																																			
S43								d57																																				
S44								d58		d46																																		
S45		r23	r23											r23				r23				r23	r23		r23																		
S46				d34	d35	d36	d33				d31																																		
S47				d34	d35	d36	d33				d31																																		
S48	r40							r40	r40	r40										r40																								
S49								d61		d46																																		

	;	{	}	id	entero	cadena	(	)	+	<	!	=	|=	var	int	boolean	string	print	input	,	return	function	if	else	$	P	D	T	F	T'	A	K	C	S	L	M	Q	S'	G	X	E	U	R	V	S"
S50				d34	d35	d36	d33	r27			d31																										62					63	29	30	32	
S51								r12							d16	d18	d17												65		64															
S52	d66									d46																																				
S53	d67																																												
S54								r26																																					
S55				d34	d35	d36	d33				d31																															68	29	30	32	
S56	d69																																												
S57	d70																																												
S58		d72		d7														d8	d9		d11		d10											73			71								
S59	r36							r36	d47	r36										r36																									
S60	r38							r38	r38	r38										r38																									
S61	r42							r42	r42	r42										r42																									
S62								d74																																					
S63								r28		d46										d55																	54								
S64								d75																																					
S65				d76																																									
S66			r18	r18										r18				r18	r18		r18	r18	r18		r18																			
S67			r19	r19										r19				r19	r19		r19	r19	r19		r19																			
S68								r28		d46										d55																	77								
S69			r20	r20										r20				r20	r20		r20	r20	r20		r20																			
S70			r21	r21										r21				r21	r21		r21	r21	r21		r21																			
S71			r22	r22										r22				r22	r22		r22	r22	r22		r22																			
S72				d7														d8	d9		d11		d10											79										78
S73			r31	r31										r31				r31	r31		r31	r31	r31		r31																			
S74	r44							r44	r44	r44										r44																									
S75		d80																																											
S76								r13												d82														81										
S77								r29																																					
S78			d83																																										
S79			r48	d7														d8	d9		d11		d10											79										99
S80			r17	d7										d5				d8	d9		d11		d10				85						84	86									
S81								r11																																					
S82															d16	d18	d17											87																
S83			r33	r33										r33				r33	r33		r33	r33	r33	d89	r33														88					
S84			d90																																										
S85			r17	d7										d5				d8	d9		d11		d10										96	86									
S86			r17	d7										d5				d8	d9		d11		d10				85						97	86									
S87				d91																																									
S88			r30	r30										r30				r30	r30		r30	r30	r30		r30																			
S89		d92																																											
S90				r8										r8				r8	r8		r8	r8	r8		r8																			
S91								r13												d82														93										
S92				d7														d8	d9		d11		d10											79										94
S93								r14																																					
S94			d95																																										
S95			r32	r32										r32				r32	r32		r32	r32	r32		r32																			
S96			r15																																										
S97			r16																																										
S98			r4	r4										r4				r4	r4		r4	r4	r4		r4																			
S99			r47																																										

4 Diseno del Analizador Semántico

4.1 Esquema de Traducción

```
0. P' → {TSG = creaTS(); DesplG = 0; TS_actual = TSG;
    P.func = false} P {liberaTS(TSG)}
1. P → D { P1 .func = P.func} P1
2. P → F { P1 .func = P.func} P1
3. P → {S.func = false} S {P1 = P.func} P1
4. D → var {zona_decl = true} T id ; {InsertarTipoTS(id.posi, T.tipo)
                                      if(TS_actual=TSG) then
                                          InsertarDespl(id.posi, desplG)
                                          desplG = desplG + T.tamano
                                      else
                                          InsertarDespl(id.posi, desplL)
                                          desplL = desplL + T.tamano
                                    zona_decl=false}
5. T → int {T.tipo=entero; T.tamano = 1}
6. T → string {T.tipo=cadena; T.tamano = 64}
7. T → boolean {T.tipo=logico; T.tamano = 1}
8. F → function {zona_decl =true} T1 id ( {TSL = creaTS();desplL=0;
    TS_actual = TSL} A ){InsertaTipoTSG(id.posi,A.tipo → T.tipo));
    zona_decl = false} { { C.func = true} C } {if (C.tipoRet != T.tipo)
                                                then error(2);
                                                TS_actual = TSG; LiberarTS(TSL)}
9. T1 →λ {T.tipo = tipo_vacio}
10. T1 →T {T1.tipo = T.tipo}
11. A → 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 x K.tipo}
12. A → λ {A.tipo = tipo_vacio}
13. K → λ {K.tipo = tipo_vacio}
14. K → , T id {InsertarTipoTS(id.posi, T.tipo);
    InsertarDesplTS(id.posi, desplL); desplL = desplL + T.tamano} K1
    {K.tipo = if(K1.tipo = tipo_vacio)
                    then T.tipo
               else
                    T.tipo x K.tipo}
15. C → D {C1.func = C.func} C1 {C.tipoRet = C1.tipoRet}
16. C → {S.func = C.func} S {C1.func = C.func} C1 {C.tipoRet =
                                        if(S.tipoRet = C1.tipoRet) then
                                            S.tipoRet
                                        else if(S.tipoRet = tipo_vacio) then
                                            C1.tipoRet
                                        else if(C1.tipoRet = tipo_vacio) then
                                            S.tipoRet
                                        else
                                            error(2)}
17. C → λ {C.tipoRet = tipo_vacio}
18. S→ id L E ; {S.tipo = if(BuscaTipoTS(id.posi) = E.tipo)
                            AND (E.tipo != 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 → {S2.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' → MM1 P {liberaTS(TSG)}
1. P → D P1 {Aux[ntope].tipoRet = Aux[tope].tipoRet}
2. P → F P1 {Aux[ntope].tipoRet = Aux[tope].tipoRet}
3. P → S P1 {Aux[ntope].tipoRet = if(Aux[tope-1].tipoRet = tipo_vacio)then
                                        Aux[tope].tipoRet
                                    else error(1)}
4. D → var MM2 T id MM8 ; {InsertarTipoTS(Aux[tope-2].posi, Aux[tope-3].tipo)
                           if(TS_actual = TSG) then
                               InsertarDespl(Aux[tope-1].posi, desplG)
                               desplG = desplG + Aux[tope-3].tamano
                           else
                               InsertarDespl(Aux[tope-2].posi, desplL)
                               desplL = desplL + Aux[tope-3]}
5. T → int {Aux[ntope].tipo=entero; Aux[ntope].tamano = 1}
6. T → string {Aux[ntope].tipo=cadena; Aux[ntope].tamano = 64}
7. T → boolean {Aux[ntope].tipo=logico; Aux[ntope].tamano = 1}
8. F → function MM3 T1 id MM4 ( A ) MM5 { C }
       {if (Aux[tope-1].tipoRet != Aux[tope-9].tipo)then error(2);
       TS_actual = TSG; LiberarTS(TSL)}
9. T1 → λ {Aux[ntope].tipo = tipo_vacio}
10. T1 → T {Aux[ntope].tipo = Aux[tope].tipo}
11. A → T id MM6 K {Aux[ntope].tipo = if(Aux[tope].tipo = tipo_vacio)
                                        then Aux[tope-3].tipo}
                                    else
                                        Aux[tope].tipo.push(Aux[tope-3].tipo)}
12. A → λ {Aux[ntope].tipo = tipo_vacio}
13. K → λ {Aux[ntope].tipo = tipo_vacio}
14. K → , T id MM7 K1 {Aux[ntope].tipo = if(Aux[tope].tipo = tipo_vacio)
                                        then NuevaPila(Aux[tope-2].tipo)
                                    else
                                        Aux[tope].tipo.push(Aux[tope-4].tipo)}
15. C → D C1 {Aux[ntope].tipoRet = Aux[tope].tipoRet}
16. C → S C1 {Aux[ntope].tipoRet=
                if(Aux[tope-1].tipoRet = Aux[tope].tipoRet) then
                    Aux[tope-1].tipoRet
                else if(Aux[tope-1].tipoRet = tipo_vacio) then
                    Aux[tope-1].tipoRet
                else if(Aux[tope].tipoRet = tipo_vacio) then
                    Aux[tope-1].tipoRet}
                else
                    error(2)}
17. C → λ {Aux[ntope].tipoRet = tipo_vacio}
18. S→ id L E ; {Aux[ntope].tipo =
     if(BuscaTipoTS(Aux[tope-3].posi)=(Aux[tope-1].tipo)
            AND (Aux[tope-1].tipo != tipo_error))then
        tipo_ok
     else
        error(3)
     Aux[ntope].tipoRet = tipo_vacio}
19. S→ id (M) ; {Aux[ntope].tipo =
     if(BuscaTipoTS(Aux[tope-4].posi) = ParFunc(Aux[tope-2].tipo, t)
        then tipo_ok
     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();
            deplG = 0;
            TSA = TSG}
51. MM2 →λ {zona_decl = true}
52. MM3 →λ {zona_decl = true}
53. MM4 →λ {TSL = creaTS();
            deplL = 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, deplL);
     deplL = deplL + Aux[tope-1].tamano}
56. MM7 →λ {InsertarTipoTS(Aux[tope].posi, Aux[tope-1].tipo) ;
     InsertarDesplTS(Aux[tope].posi, deplL);
     deplL = deplL + 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, 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 → var {zdecl:=true} T id; {otras acciones}
Lo transformamos en:
D → var MM T id; {otras acciones}
MM → lambda {zdecl:=true}
```

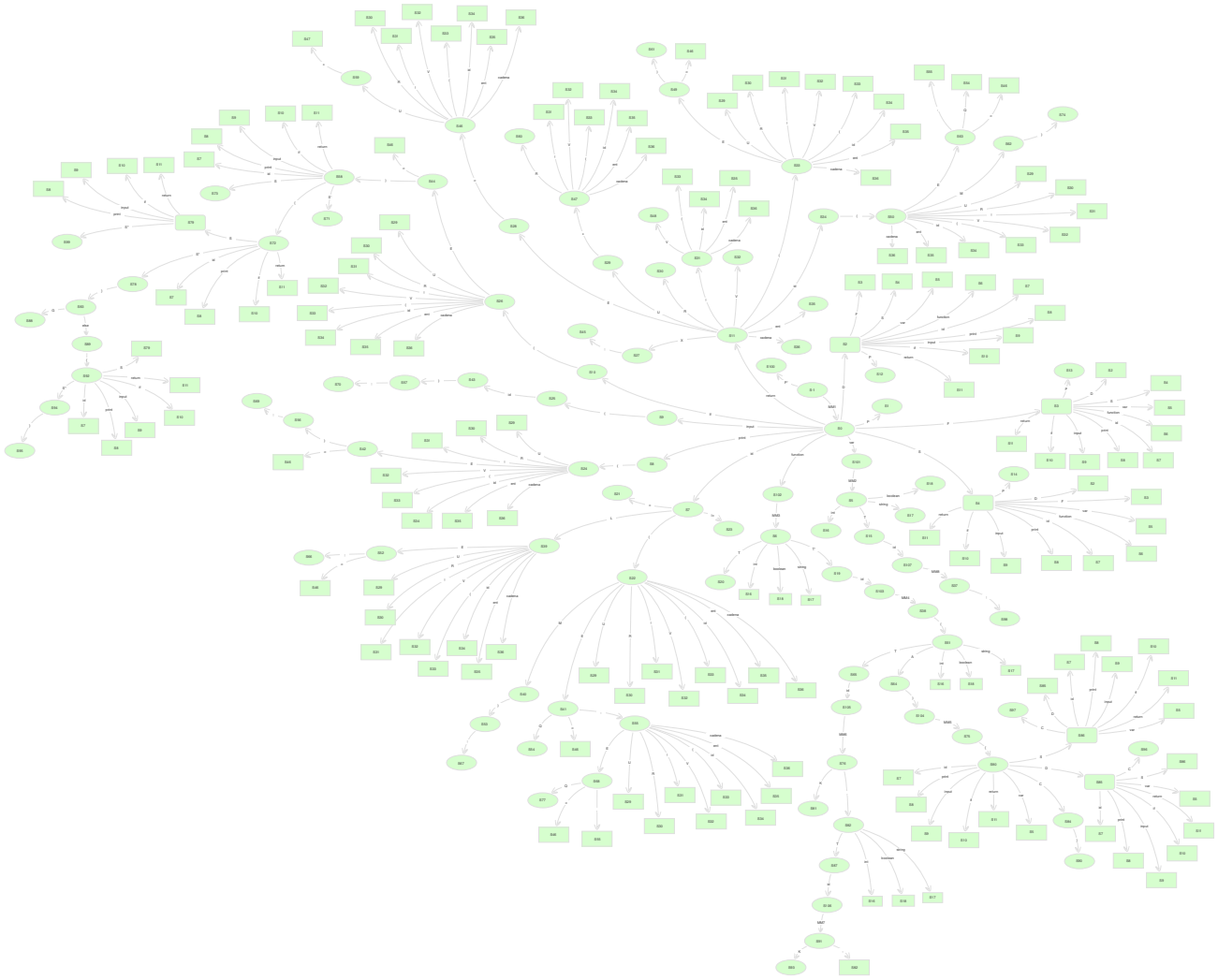
4.3 Gramática y Autómata Final del A. Sintáctico

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 → *M1 P*
P → *D P*
P → *F P*
P → *S P*
D → *var M2 T id M8* ;
T → *int*
T → *string*
T → *boolean*
F → *function M3 T1 id M4* (*A*) *M5* { *C* }
T1 → *lambda*
T1 → *T*
A → *T id M6 K*
A → *lambda*
K → *lambda*
K → , *T id M7 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*

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

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

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



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 f
Linea: 14
```

Error 5:

```
1 var boolean booleano;
2 function boolean bisiesto (int a)
3 { var int bis;
4   print ('Es bisiesto?');
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 bisiesto (int a)
3  { var int bis;
4    print ('Es bisiesto?');
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 (bisiesto(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 bisiesto?">
<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'
```

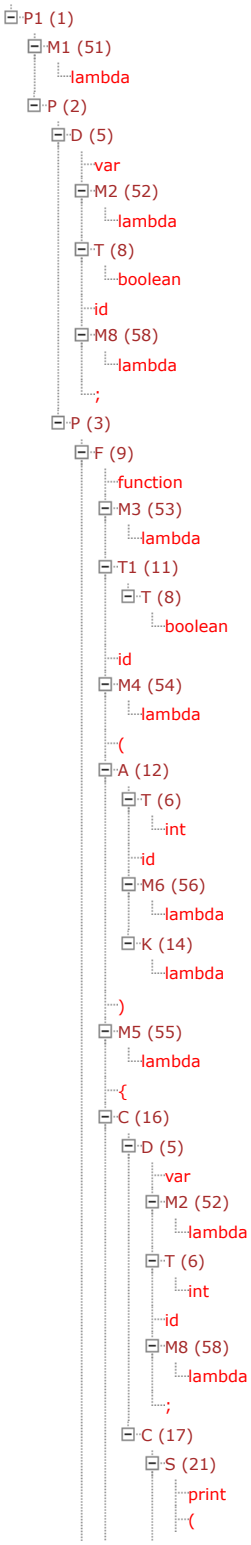
```
  + Tipo: 'entero'
  + Despl: 1

 * 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'
```

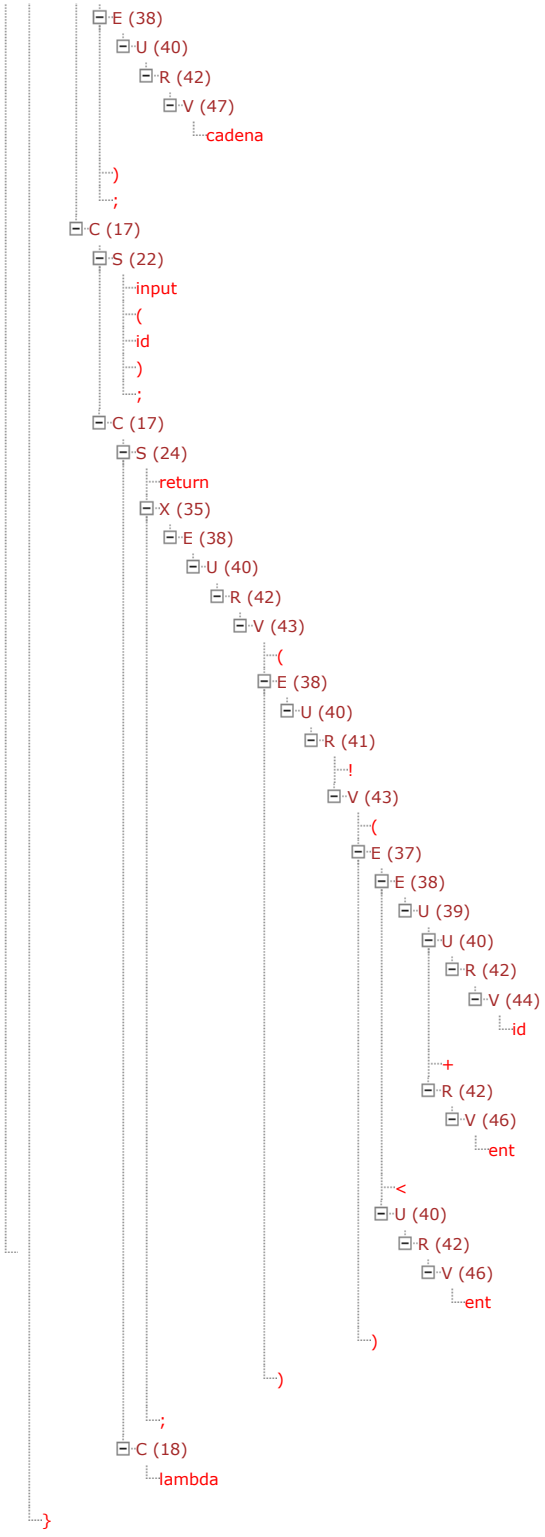
```
  + Tipo: 'funcion'
  + numParam: 3
  + TipoParam1: 'entero'
  + ModoParam1: 'Valor'
  + TipoParam2: 'entero'
  + ModoParam2: 'Valor'
  + TipoParam3: 'entero'
  + ModoParam3: 'Valor'
  + TipoRetorno: 'logico'
  + EtiqFuncion: 'esFechaCorrecta3'

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

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

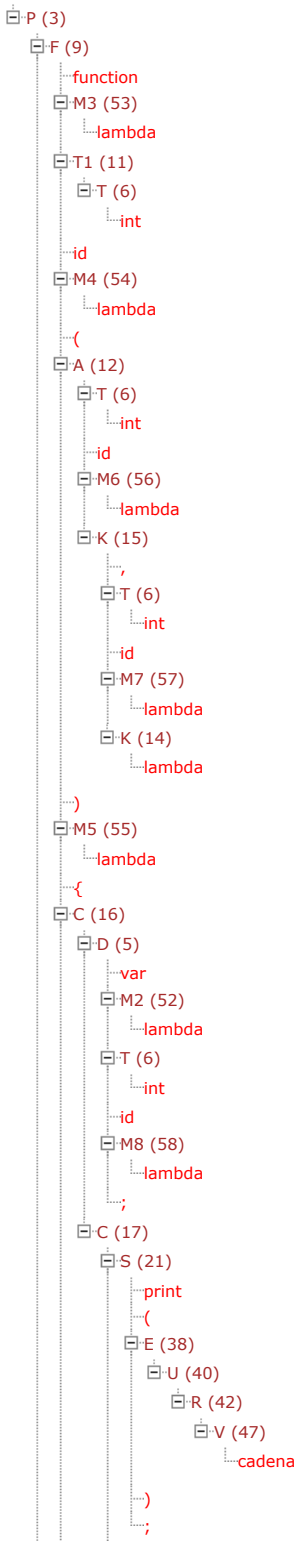
Parse a Derechas:

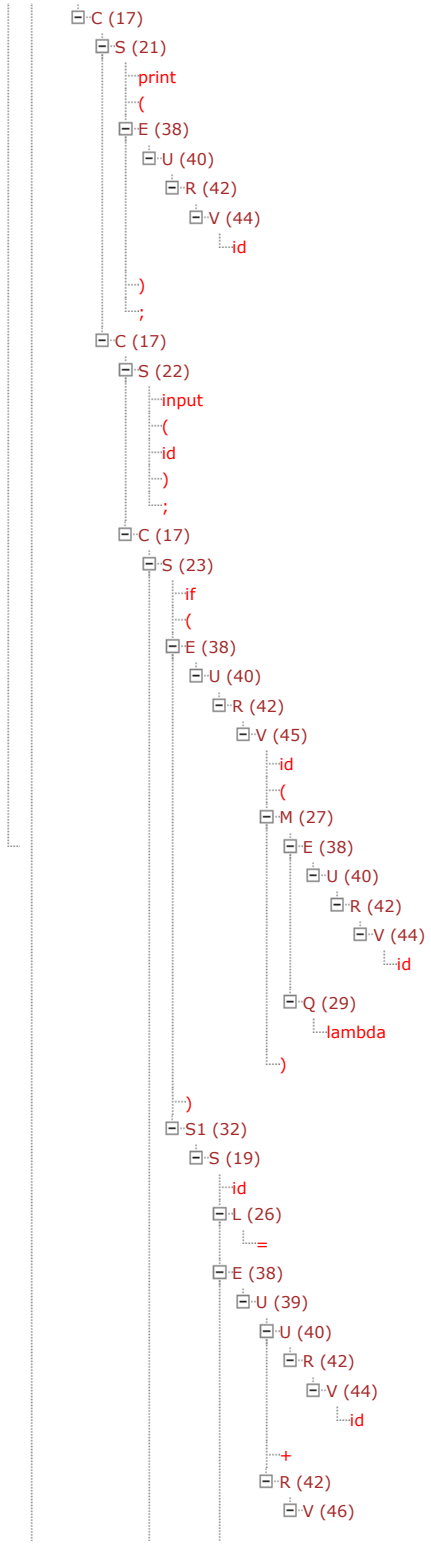
A 51 52 8 58 5 53 8 11 54 6 56 14 12 55 52 6 58 5 47 42 40 38 21 22 44 42 40 46 42 39 38 46 42 40 37 43 41 40 38 43 42 40 38 35 24 18 17 17 17 16 9 53 6 11 54 6 56 6 57 14 15 12 55 52 6 58 5 47 42 40 38 21 44 42 40 38 21 22 44 42 40 38 29 27 45 42 40 38 26 44 42 40 46 42 39 38 19 32 23 44 42 40 38 35 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

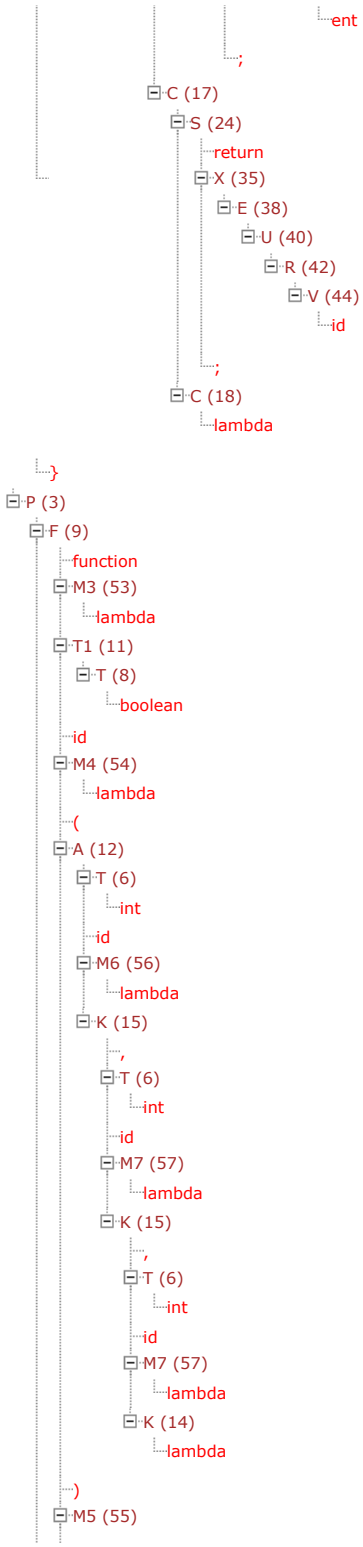
Árbol sintáctico:

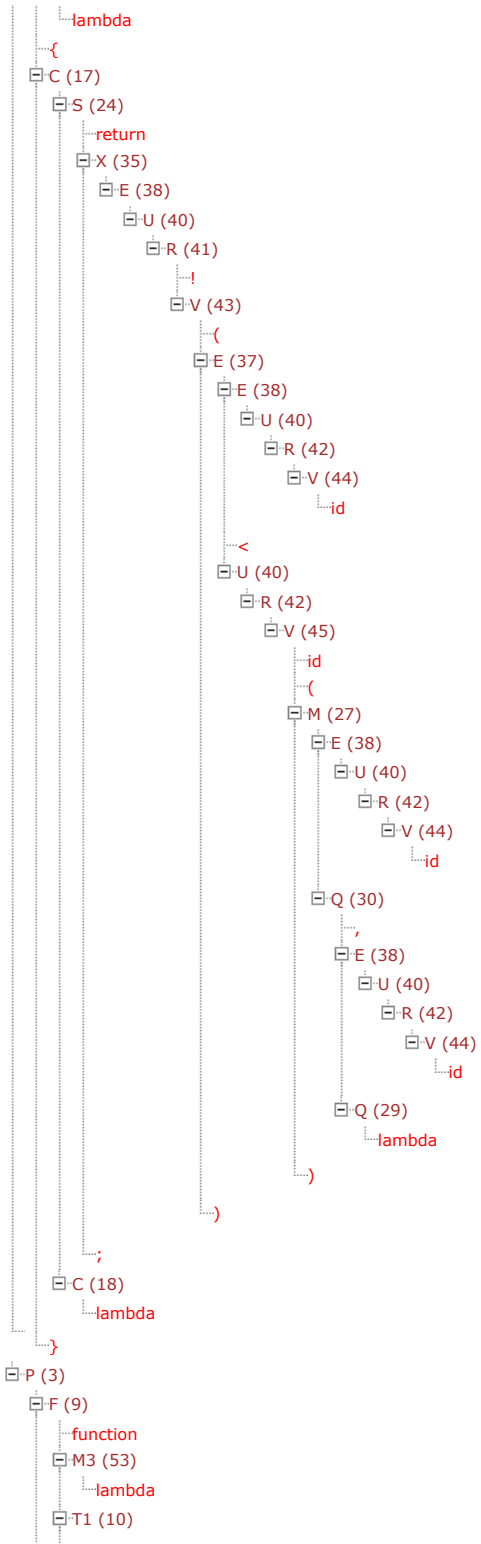


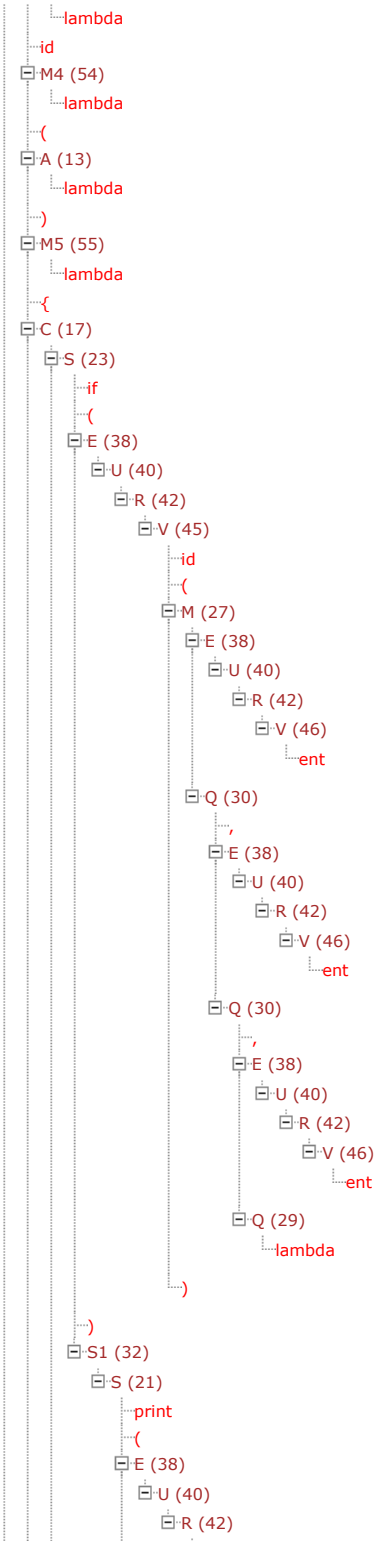

E (38)
U (40)
R (42)
V (47)
cadena
)
;
C (17)
S (22)
input
(
id
)
;
C (17)
S (24)
return
X (35)
E (38)
U (40)
R (42)
V (43)
(
E (38)
U (40)
R (41)
!
V (43)
(
E (37)
E (38)
U (39)
U (40)
R (42)
V (44)
id
+
R (42)
V (46)
ent
<
U (40)
R (42)
V (46)
ent
)
)
;
C (18)
lambda
}

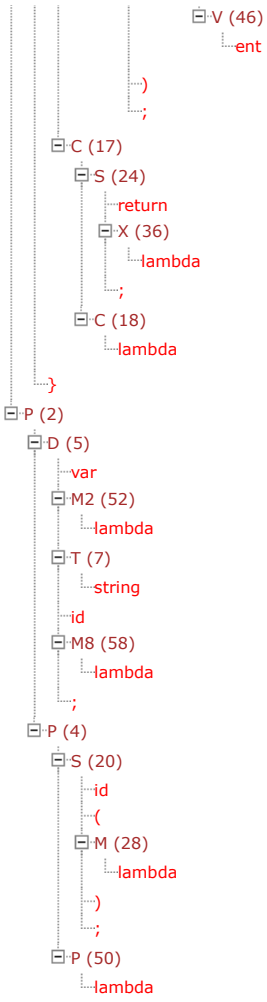

P (3)
F (9)
function
M3 (53)
lambda
T1 (11)
T (6)
int
id
M4 (54)
lambda
(
A (12)
T (6)
int
id
M6 (56)
lambda
K (15)
,
T (6)
int
id
M7 (57)
lambda
K (14)
lambda
)
M5 (55)
lambda
{
C (16)
D (5)
var
M2 (52)
lambda
T (6)
int
id
M8 (58)
lambda
;
C (17)
S (21)
print
(
E (38)
U (40)
R (42)
V (47)
cadena
)
;


C (17)
S (21)
print
(
E (38)
U (40)
R (42)
V (44)
id
)
;
C (17)
S (22)
input
(
id
)
;
C (17)
S (23)
if
(
E (38)
U (40)
R (42)
V (45)
id
(
M (27)
E (38)
U (40)
R (42)
V (44)
id
Q (29)
lambda
)
)
S1 (32)
S (19)
id
L (26)
=
E (38)
U (39)
U (40)
R (42)
V (44)
id
+
R (42)
V (46)


ent
;
C (17)
S (24)
return
X (35)
E (38)
U (40)
R (42)
V (44)
id
;
C (18)
lambda
}
P (3)
F (9)
function
M3 (53)
lambda
T1 (11)
T (8)
boolean
id
M4 (54)
lambda
(
A (12)
T (6)
int
id
M6 (56)
lambda
K (15)
,
T (6)
int
id
M7 (57)
lambda
K (15)
,
T (6)
int
id
M7 (57)
lambda
K (14)
lambda
)
M5 (55)


lambda
{
C (17)
S (24)
return
X (35)
E (38)
U (40)
R (41)
!
V (43)
(
E (37)
E (38)
U (40)
R (42)
V (44)
id
<
U (40)
R (42)
V (45)
id
(
M (27)
E (38)
U (40)
R (42)
V (44)
id
Q (30)
,
E (38)
U (40)
R (42)
V (44)
id
Q (29)
lambda
)
)
;
C (18)
lambda
}
P (3)
F (9)
function
M3 (53)
lambda
T1 (10)


lambda
id
M4 (54)
lambda
(
A (13)
lambda
)
M5 (55)
lambda
{
C (17)
S (23)
if
(
E (38)
U (40)
R (42)
V (45)
id
(
M (27)
E (38)
U (40)
R (42)
V (46)
ent
Q (30)
,
E (38)
U (40)
R (42)
V (46)
ent
Q (30)
,
E (38)
U (40)
R (42)
V (46)
ent
Q (29)
lambda
)
)
S1 (32)
S (21)
print
(
E (38)
U (40)
R (42)


V (46)
ent
)
;
C (17)
S (24)
return
X (36)
lambda
;
C (18)
lambda
}
P (2)
D (5)
var
M2 (52)
lambda
T (7)
string
id
M8 (58)
lambda
;
P (4)
S (20)
id
(
M (28)
lambda
)
;
P (50)
lambda

Fuente 2 Correcta:

```
var int a;
var int b;
a = 3;
b = a;
 var boolean c;
c = a  <  b;
if (c) b  =  1;
c = b  <  a;
if (c) b  =  4;
 a = a  +  b;
print (a);
print (b);
```

Fuente 3 Correcta:

```
var int a;
var int b;
var int c;
print ('Introduce el primer operando');
input (a);
print ('Introduce el segundo operando');
input (b);
function int suma (int num1, int num2)
{
  var int res;
  res = num1+num2;
  return res;
}
c = suma (a, b);
print (c);
```

Fuente 4 Correcta:

```
var string texto;
function imprime (string msg)
{
  print ('Mensage introducido:');
  print (msg);
}
function pideTexto ()
{
  print ('Introduce un texto');
  input (texto);
}
pideTexto();
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);
```