PROCESADORES DE LENGUAJES

Grupo 64: Segunda Entrega

Christian Paniagua Paniagua x150380

Javier Pérez Martín x150147

Pablo Heras Aranzana x150046

Índice

1. Diseño Analizador Sintáctico

1.1. Gramática

NoTerminales = { P B T S _S F H L Q A K X C E _E R _R U _U V _V W _W Z _Z G _G D }

Terminales = { var id ; if ( ) while { } switch case break default int bool string return print prompt function , || && == != < > -= ! = + - * / % CAD : cte_ent cte_bool }

Axioma = P

Producciones = {

P -> B P

P -> F P

P -> lambda

B -> var A ;

B -> if (E) S

B -> while (E) { C }

B -> switch (E) { D }

B -> S

T -> int

T -> bool

T -> string

S -> id _S

S -> return X ;

S -> print (E) ;

S -> prompt (id) ;

S -> break ;

_S -> = E ;

_S -> -= E ;

_S -> (L) ;

F -> function H id (A) { C }

H -> T

H -> lambda

L -> E Q

L -> lambda

Q -> , E Q

Q -> lambda

A -> T id K

A -> lambda

K -> , T id K

K -> lambda

X -> E

X -> lambda

C -> B C

C -> lambda

E -> R _E

_E -> || R _E

_E -> lambda

R -> U _R

_R -> && U _R

_R -> lambda

U -> V _U

_U -> == V

_U -> != V

_U -> lambda

V -> W _V

_V -> > W _V

_V -> < W _V

_V -> lambda

W -> Z _W

_W -> + Z _W

_W -> - Z _W

_W -> lambda

Z -> G _Z

_Z -> * G _Z

_Z -> / G _Z

_Z -> % G _Z

_Z -> lambda

G -> ! G

G -> id _G

G -> (E)

G -> cte_ent

G -> CAD

G -> cte_bool

_G -> (L)

_G -> lambda

D -> case cte_ent : C D

D -> default : C

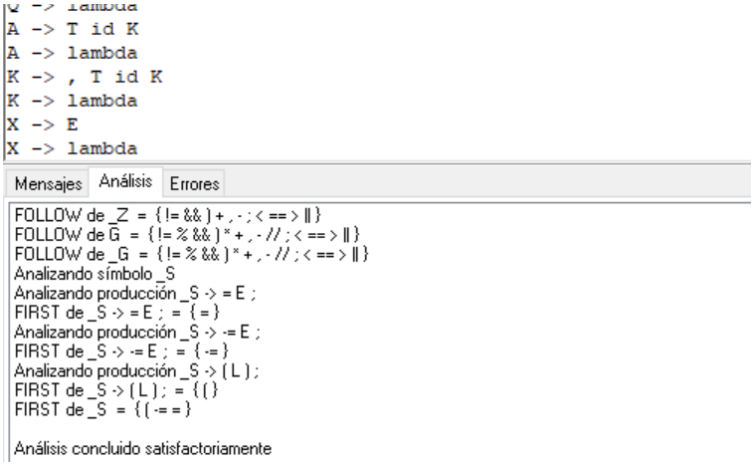
}

1.2. Demostración LL(1)

Puesto que nuestro método de análisis sintáctico es el "Descendente Recursivo" nuestra gramática debe ser una gramática LL(1). Este tipo de gramáticas son no ambiguas, no recursivas por la izquierda y factorizadas por la izquierda. A parte de estas condiciones necesarias, todas las producciones de la gramática con mismo antecedente deben cumplir (dos a dos) la condición LL(1):

- Si $A \rightarrow \alpha_1$, $A \rightarrow \alpha_2 \in G$ => $First(\alpha_1) \cap First(\alpha_2) = \emptyset$ Es decir, que no tengan nada en común.
- Si $A \rightarrow \alpha_1$, $A \rightarrow \alpha_2 \in G$ y $\lambda \in First(\alpha_1)$ => $Follow(A) \cap First(\alpha_2) = \emptyset$
  Pertenece a una de las reglas.

Tras eliminar las recursividades por la izquierda y factorizar nuestra gramática empleamos la herramienta SDGLL1 proporcionada en la web de la asignatura para comprobar que nuestra gramática es LL(1). Tras analizar la gramática en la herramienta obtuvimos los First y Follows de las distintas reglas, así como la confirmación de que la gramática es LL(1).



1.3. Procedimientos

A continuación, mostramos el código de los procedimientos que realizarán el análisis sintáctico. El resultado de éste se volcará en el fichero de parse.

```
print("Des ");
Proc P;

	if(st ϵ First(B P)={ break id if print prompt return switch var while
}) then
	{
		print("1");
		B();
		P();
	}
	else if(st ϵ First(F P)={ function }) then
	{
		print("2");
		F();
		P();
	}
	else if(st ϵ Follow(P)={ $ })
	{
		print("3");
	}
	else ERROR;
END;

Proc B;
	if(st ϵ First(var A ;)={ var }) then
	{
		print("4");
		EqT(var);
		A();
		EqT(;);
	}
	else if(st ϵ First(if ( E ) S)={ if }) then
	{
		print("5");
		EqT(if);
		EqT(();
		E();
		EqT());
		S();
	}
	else if(st ϵ First(while ( E ) { C })={ while }) then
	{
		print("6");
		EqT(while);
		EqT(();
		E();
		EqT());
		EqT({);
		C();
		EqT(});
	}
	else if(st ϵ First(switch ( E ) { D })={ switch }) then
	{
		print("7");
		EqT(switch);
		EqT(();
		E();
		EqT());
		EqT({);
		D();
		EqT(});
	}
	else if(st ϵ First(S)={ break id print prompt return }) then
	{
		print("8");
		S();
	}
	else ERROR;
END;

Proc T;
	if(st ϵ First(int)={ int }) then
	{
		print("9");
		EqT(int);
	}
	else if(st ϵ First(bool)={ bool }) then
	{
		print("10");
		EqT(bool);
	}
	else if(st ϵ First(string)={ string }) then
	{
		print("11");
		EqT(string);
	}
	else ERROR;
END;


Proc S;
	if(st ϵ First(id S')={ id }) then
	{
		print("12");
		EqT(id);
		S'();
	}
	else if(st ϵ First(return X ;)={ return }) then
	{
		print("13");
		EqT(return);
		X();
		EqT(;);
	}
	else if(st ϵ First(print ( E ) ;)={ print }) then
	{
		print("14");
		EqT(print);
		EqT(();
		E();
		EqT());
		EqT(;);
	}
	else if(st ϵ First(prompt ( id ) ;)={ prompt }) then
	{
		print("15");
		EqT(prompt);
		EqT(();
		EqT(id);
		EqT());
		EqT(;);
	}
	else if(st ϵ First( break ;)={ break }) then
	{
		print("16");
		EqT(break);
		EqT(;);
	}
	else ERROR;
END;

Proc S';
	if(st ϵ First(= E ;)={ = }) then
	{
		print("17");
		EqT(=);
		E();
		EqT(;);
	}
	else if(st ϵ First(-= E ;)={ -= }) then
	{
		print("18");
		EqT(-=);
		E();
		EqT(;);
	}
	else if(st ϵ First(( L ) ;)={ ( }) then
	{
		print("19");
		EqT(();
		L();
		EqT());
		EqT(;);
	}
	else ERROR;
END;
```

```
Proc F;
	if(st ϵ First(function H id ( A ) { C })={ function }) then
	{
		print("20");
		EqT(function);
		H();
		EqT(id);
		EqT(();
		A();
		EqT());
		EqT({);
		C();
		EqT(});
	}
	else ERROR;
END;

Proc H;
	if(st ϵ First(T)={ int bool string }) then
	{
		print("21");
		T();
	}
	else if(st ϵ Follow(H)={ id }) then
	{
		print("22");
	}
	else ERROR;
END;

Proc L;
	if(st ϵ First(E Q)={ ! ( cte_bool CAD cte_ent id }) then
	{
		print("23");
		E();
		Q();
	}
	else if(st ϵ Follow(L)={ ) }) then
	{
		print("24");
	}
	else ERROR;
END;

Proc Q;
	if(st ϵ First(, E Q)={ , }) then
	{
		print("25");
		EqT(,);
		E();
		Q();
	}
	else if(st ϵ Follow(Q)={ ) }) then
	{
		print("26");
	}

	else ERROR;
END;

Proc A;
	if(st ϵ First(T id K)={ int bool string }) then
	{
		print("27");
		T();
		EqT(id);
		K();
	}
	else if(st ϵ Follow(A)={ ) ; }) then
	{
		print("28");
	}
	else ERROR;
END;

Proc K;
	if(st ϵ First(, T id K)={ , }) then
	{
		print("29");
		EqT(,);
		T();
		EqT(id);
		K();
	}
	else if(st ϵ Follow(K)={ ) ; }) then
	{
		print("30");
	}
	else ERROR;
END;

Proc X;
	if(st ϵ First(E)={ ! ( cte_bool CAD cte_ent id }) then
	{
		print("31");
		E();
	}
	else if(st ϵ Follow(X)={ ; }) then
	{
		print("32");
	}
	else ERROR;
END;

Proc C;
	if(st ϵ First(B C)={ break id if print prompt return switch var while
}) then
	{
		print("33");
		B();
		C();
	}
	else if(st ϵ Follow(C)={ case } default }) then
	{
		print("34");
	}
	else ERROR;
END;

Proc E;
	if(st ϵ First(R E')={ ! ( cte_bool CAD cte_ent id }) then
	{
		print("35");
		R();
		E'();
	}
	else ERROR;
END;

Proc E';
	if(st ϵ First(|| R E')={ || }) then
	{
		print("36");
		EqT(||);
		R();
		E'();
	}
	else if(st ϵ Follow(E')={ ) , ; }) then
	{
		print("37");
	}
	else ERROR;
END;

Proc R;
	if(st ϵ First(U R')={ ! ( cte_bool CAD cte_ent id }) then
	{
		print("38");
		U();
		R'();
	}
	else ERROR;
END;

Proc R';
	if(st ϵ First(&& U R')={ && }) then
	{
		print("39");
		EqT(&&);
		U();
		R'();
	}
	else if(st ϵ Follow(R')={ ) , ; || }) then
	{
		print("40");
	}
	else ERROR;
END;

Proc U;
	if(st ϵ First(V U')={ ! ( cte_bool CAD cte_ent id }) then
	{
		print("41");
		V();
		U'();
	}
	else ERROR;
END;
```

```
Proc U';
	if(st ϵ First(== V U')={ == }) then
	{
		print("42");
		EqT(==);
		V();
		U'();
	}
	else if(st ϵ First(!= V U')={ != }) then
	{
		print("43");
		EqT(!=);
		V();
		U'();
	}
	else if(st ϵ Follow(U')={ && ) , ; || }) then
	{
		print("44");
	}
	else ERROR;
END;

Proc V;
	if(st ϵ First(W V')={ ! ( cte_bool CAD cte_ent id }) then
	{
		print("45");
		W();
		V'();
	}
	else ERROR;
END;

Proc V';
	if(st ϵ First(> W V')={ > }) then
	{
		print("46");
		EqT(>);
		W();
		V'();
	}
	else if(st ϵ First(< W V')={ < }) then
	{
		print("47");
		EqT(<);
		W();
		V'();
	}
	else if(st ϵ Follow(V')={ != && ) , ; == || }) then
	{
		print("48");
	}

	else ERROR;
END;

Proc W;
	if(st ϵ First(Z W')={ ! ( cte_bool CAD cte_ent id }) then
	{
		print("49");
		Z();
		W'();
	}
	else ERROR;
END;

Proc W';
	if(st ϵ First(+ Z W')={ + }) then
	{
		print("50");
		EqT(+);
		Z();
		W'();
	}
	else if(st ϵ First(- Z W')={ - }) then
	{
		print("51");
		EqT(-);
		Z();
		W'();
	}
	else if(st ϵ Follow(W')={ != && ) , ; < == > || }) then
	{
		print("52");
	}
	else ERROR;
END;

Proc Z;
	if(st ϵ First(G Z')={ ! ( cte_bool CAD cte_ent id }) then
	{
		print("53");
		G();
		Z'();
	}
	else ERROR;
END;

Proc Z';

	if(st ϵ First(* G Z')={ * }) then
	{
		print("54");
		EqT(*);
		G();
		Z'();
	}
	else if(st ϵ First(/ G Z')={ / }) then
	{
		print("55");
		EqT(/);
		G();
		Z'();
	}
	else if(st ϵ First(% G Z')={ % }) then
	{
		print("56");
		EqT(%);
		G();
		Z'();
	}
	else if(st ϵ Follow(Z')={ != && ) + , - ; < == > || }) then
	{
		print("57");
	}
	else ERROR;
END;

Proc G;
	if(st ϵ First(! G)={ ! }) then
	{
		print("58");
		EqT(!);
		G();
	}
	else if(st ϵ First(id G')={ id }) then
	{
		print("59");
		EqT(id);
		G'();
	}
	else if(st ϵ First(( E ))={ ( }) then
	{
		print("60");
		EqT(();
		E();
		EqT());
	}
	else if(st ϵ First(cte_ent)={ cte_ent }) then
	{
		print("61");
		EqT(cte_ent);
	}
	else if(st ϵ First(CAD)={ CAD }) then
	{
		print("62");
		EqT(CAD);
	}
	else if(st ϵ First(cte_bool)={ cte_bool }) then
	{
		print("63");
		EqT(cte_bool);
	}
	else ERROR;
END;

Proc G';
	if(st ϵ First(( L ))={ ( }) then
	{
		print("64");
		EqT(();
		L();
		EqT();
	}
	else if(st ϵ Follow(G')={ != % && ) * + , - / ; < == > || }) then
	{
		print("65");
	}
	else ERROR;
END;
```

```
Proc D;
        if(st ϵ First(case cte_ent : C D)={ case }) then
        {
                print("66");
                EqT(case);
                EqT(cte_ent);
                EqT(:);
                C();
                D();
        }
        else if(st ϵ First(default : C)={ default }) then
        {
                print("67");
                EqT(default);
                EqT(:);
                C();
        }
        else ERROR;
END;
```

```
Proc EqT(t: token){
        If(sig_token=t)
                Then sig_token = nextToken()
                Else Error;
}
```

2. Casos de prueba

2.1. Correctos

Prueba código correcto 1

```
int a;
bool b;
b = false;
function int hey (int a){
        b = false;
        return a; }
```

Parse:
Des 1 4 27 9 30 1 4 27 10 30 1 8 12 17 35 38 41 45 49 53 63 57 52 48 44 40 37 2 20 21 9 27 9 30 33 8 12 17 35 38 41 45 49 53 63 57 52 48 44 40 37 33 8 13 31 35 38 41 45 49 53 59 65 57 52 48 44 40 37 34 3
Árbol: (ver anexo)

Prueba código correcto 2

```
var bool z;
var int b;
z=true;
b=0;
switch(b)
{
        case 1: z=false;
        default: break;
}
```

Parse:
Des 1 4 27 10 30 1 4 27 9 30 1 8 12 17 35 38 41 45 49 53 63 57 52 48 44 40 37 1 8 12 17 35 38 41 45 49 53 61 57 52 48 44 40 37 1 7 35 38 41 45 49 53 59 65 57 52 48 44 40 37 66 33 8 12 17 35 38 41 45 49 53 63 57 52 48 44 40 37 34 67 33 8 16 34 3
Árbol: (ver anexo)

Prueba código correcto 3

```
if(b<35 || c>1)
        b=c;
var int l;
l = c-1*2;
```

Parse:
Des 1 5 35 38 41 45 49 53 59 65 57 52 47 49 53 61 57 52 48 44 40 36 38 41 45 49 53 59 65 57 52 46 49 53 61 57 52 48 44 40 37 12 17 35 38 41 45 49 53 59 65 57 52 48 44 40 37 1 4 27 9 30 1 8 12 17 35 38 41 45 49 53 59 65 57 51 53 61 54 61 57 52 48 44 40 37 3
Árbol: (ver anexo)

2.2. Incorrectos

Prueba código incorrecto 1

```
int a - 2
bool b;
b = false;
function int hey (int a){
        b = false;
        return a;
}
```

Volcado Error:

ERROR

Prueba código incorrecto 2

```
var bool z - ;
var int b;
z=true;
b=0;
switch(b)
{
        case 1: z=false;
        default: break;
}
```

Volcado Error:

ERROR

Prueba código incorrecto 3

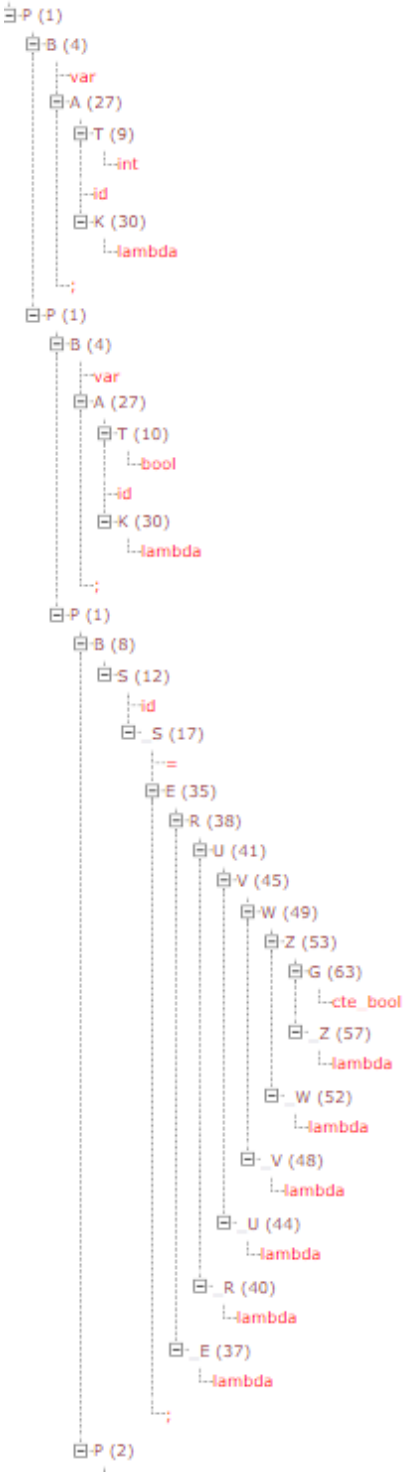
```
if(b<35 || c>1(
        b=c;
var int l;
l = c-1*2;
```

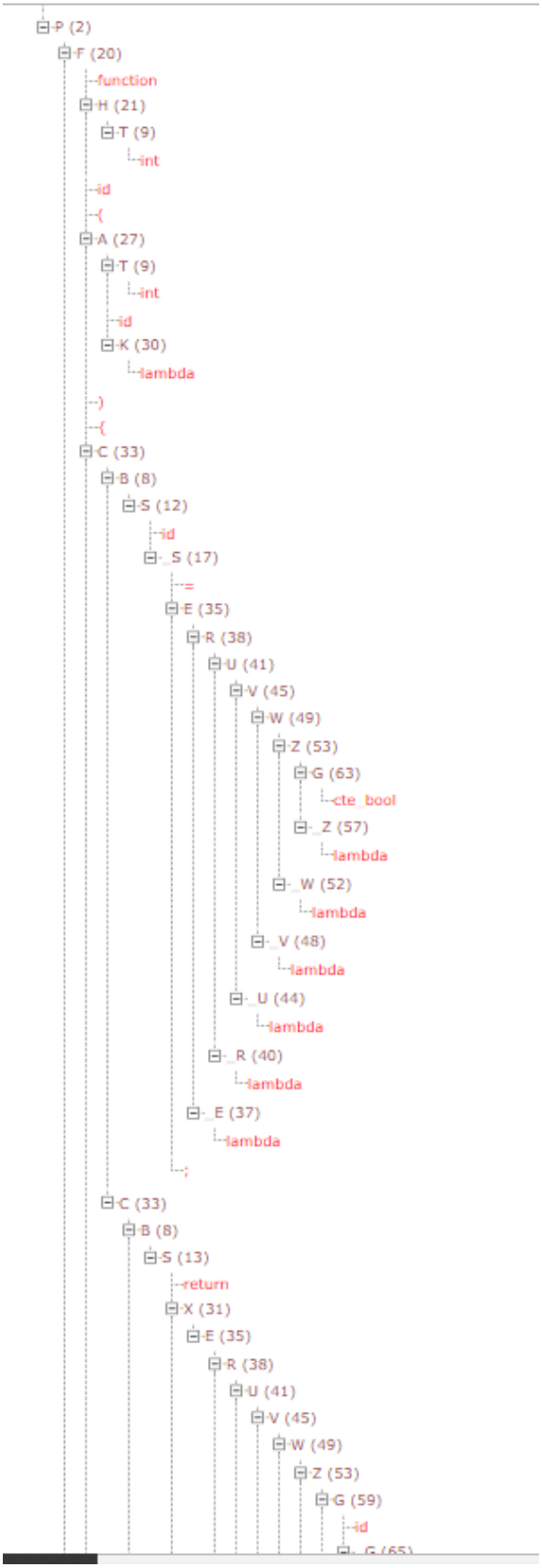
Volcado Error:

ERROR

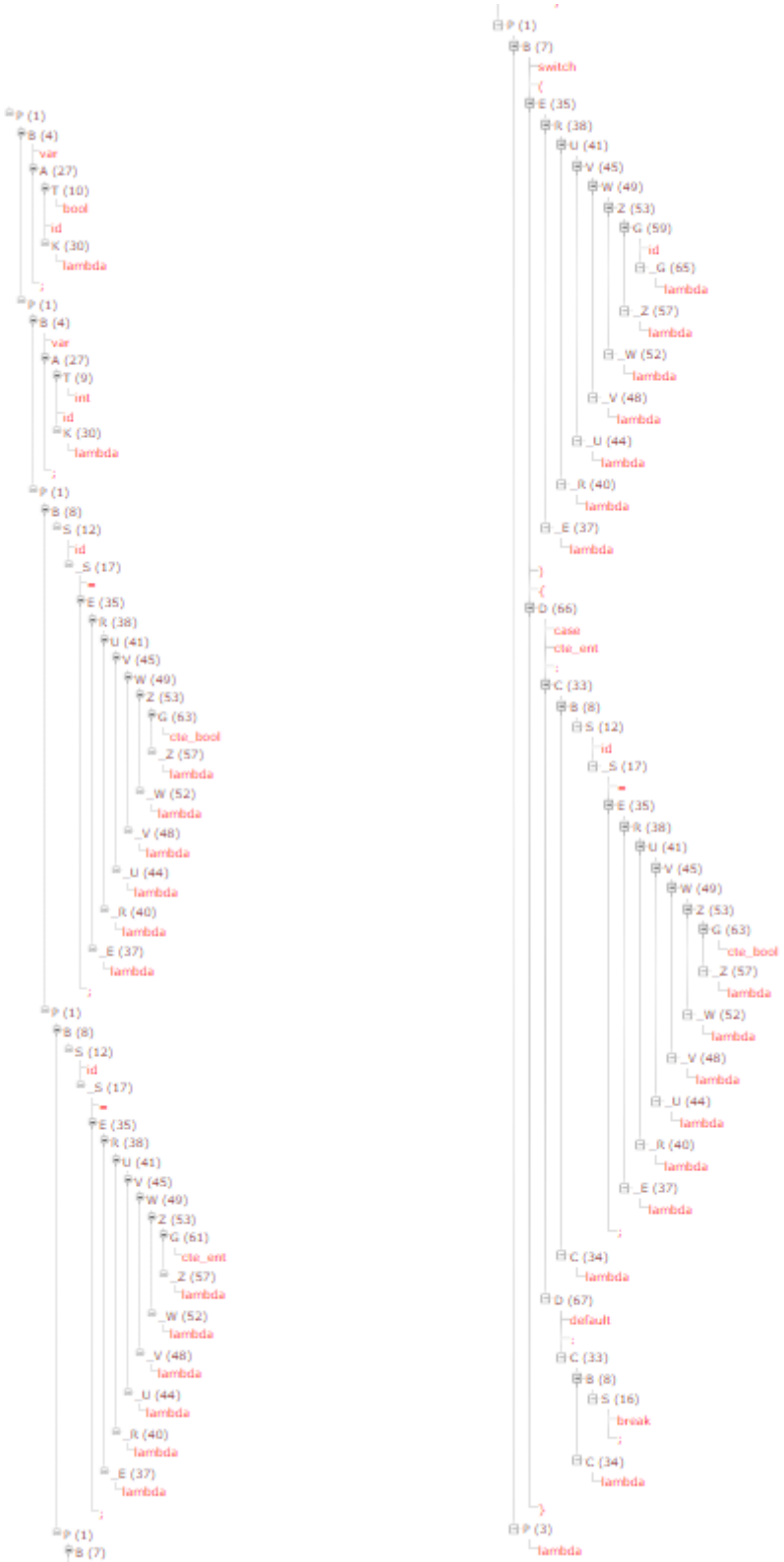
ANEXO

Árbol Prueba 1





Árbol Prueba 2



Árbol Prueba 3