

Práctica PL

Elena Kaloyanova Popova y Álvaro Borja Velasco García

2018

Índice general

1. Fase 1: Analizador léxico	2
1.1. Clases Léxicas	2
1.2. Especificación Formal	3
1.3. Diseño	5
2. Fase 2: Analizador sintáctico	6
2.1. Gramática incontextual	6
2.1.1. Operadores	6
2.1.2. Gramática incontextual	6
2.1.3. Gramática transformada LL(1)	7
2.1.4. Primeros y siguientes	9
2.1.5. Directores	10
3. Fase 3: Analizador sintáctico ascendente	12
4. Fase 4: Construcción de árboles de sintaxis abstracta	13
4.1. Funciones constructoras	13
4.2. Diagrama de clases	15
4.3. Constructor de árboles de sintaxis abstracta	15

Capítulo 1

Fase 1: Analizador léxico

1.1. Clases Léxicas

Todo programa consta de dos secciones: una para las declaraciones y otra para las instrucciones, separadas por un token «&&». La sección de declaraciones está formada por una serie de declaraciones compuestas por el nombre de tipo y el de variable y separadas por un punto y coma. La sección de instrucciones, por su parte, consta de una serie de asignaciones (variable=expresión), separadas también por un punto y coma. Las clases léxicas que hemos considerado para representar los tokens del lenguaje son las siguientes:

- **SEC:** Representa el seccionador de las dos partes del programa («&&»).
- **NUM:** Palabra reservada «num».
- **BOOL:** Palabra reservada «bool».
- **VAR:** Representa el nombre de la variable. Comienza necesariamente por una letra, seguida por una secuencia de cero o más letras, dígitos o el símbolo «_».
- **ASIG:** Representa el signo igual de las asignaciones.
- **NXT:** Representa el signo «;» que marca el comienzo de la siguiente instrucción.
- **TRUE:** Palabra reservada «true».
- **FALSE:** Palabra reservada «false».

- **NUMR:** Representa un número real. Puede empezar opcionalmente con un signo seguido de una secuencia de uno o más dígitos cualesquiera, pudiendo poner ceros no significativos a la izquierda. Puede opcionalmente estar seguido por una parte decimal y/o una parte exponencial.
- **MAS:** Operador suma ($\backslash +$).
- **MENOS:** Operador resta ($\backslash -$).
- **POR:** Operador multiplicación ($\backslash *$).
- **DIV:** Operador división ($\backslash /$).
- **AND:** Palabra reservada «and».
- **OR:** Palabra reservada «or».
- **NOT:** Palabra reservada «not».
- **MAY:** Operador mayor ($>$).
- **MEN:** Operador menor ($<$).
- **MAYI:** Operador mayor o igual ($>=$).
- **MENI:** Operador menor o igual ($<=$).
- **IGUAL:** Operador igual a ($==$).
- **DIST:** Operador distinto a ($!=$).
- **PAP:** Signo de apertura de paréntesis.
- **PCI:** Signo de cierre de paréntesis.
- **EOF:** Representa el final de fichero.

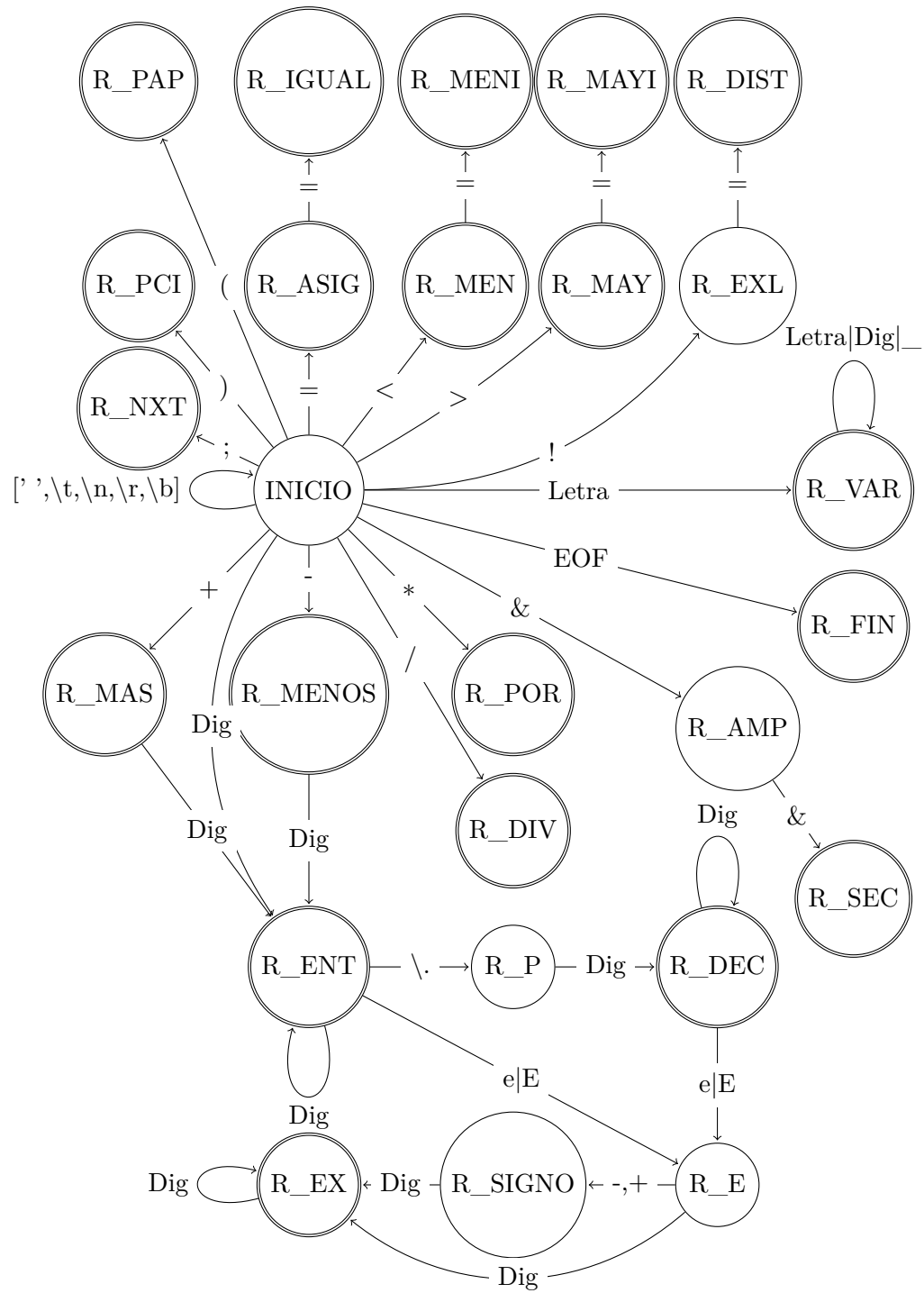
1.2. Especificación Formal

Las definiciones regulares correspondientes a las clases léxicas definidas son:

- (★) **SEC:** $[\&][\&]$
- (★) **VAR:** $\text{LETRA}([\text{LETRA}|\text{DIG}|\backslash_]*)$
 LETRA: $([a-z,A-Z])$

- DIG:** ([0-9])
 (★) **NUM:** ([n][u][m])
 (★) **BOOL:** ([b][o][o][l])
 (★) **TRUE:** ([t][r][u][e])
 (★) **FALSE:** ([f][a][l][s][e])
 (★) **NUMR:** SIGNO?(DIG+(DEC)?(EX)?)
 DEC: (\.)DIG+
 EX: [e|E](SIGNO?DIG+)
 SIGNO: [\+, \-]
 DIG: [0-9]
 (★) **AND:** ([a][n][d])
 (★) **OR:** ([o][r])
 (★) **NOT:** ([n][o][t])
 (★) **MAS:** (\+)
 (★) **MENOS:** (\-)
 (★) **DIV:** (/)
 (★) **POR:** (*)
 (★) **MAY:** (>)
 (★) **MEN:** (<)
 (★) **MAYI:** ([>][=])
 (★) **MENI:** ([<][=])
 (★) **IGUAL:** ([=][=])
 (★) **DIST:** (![=])
 (★) **ASIG:** (=)
 (★) **NXT:** (;)
 (★) **PAP:** (\()
 (★) **PCIERRE:** (\))
 (★) **SEP:** [« », \t, \n, \r, \b]

1.3. Diseño



Capítulo 2

Fase 2: Analizador sintáctico

En esta fase desarrollaremos el analizador sintáctico descendente predictivo para el lenguaje descrito en la primera fase.

2.1. Gramática incontextual

2.1.1. Operadores

Empezaremos definiendo la gramática incontextual que define el lenguaje. Los operadores que utiliza nuestro lenguaje aparecen en la tabla 2.1.

Operador	Prioridad	Tipo	Asociatividad
+, -	0	Binario infijo	Asocia Izquierda
and a or	1	Binarios infijos	Asocia Derecha No asocia
Relacionales	2	Binario infijo	No asocia
*, /	3	Binario infijo	Asocia Izquierda
not	4	Unarios prefijos	Asocia No asocia

Cuadro 2.1: Operadores

2.1.2. Gramática incontextual

La gramática incontextual obtenida apartir de la definición y los operadores es la siguiente:

$S \rightarrow \text{Programa EOF}$

$\text{Programa} \rightarrow \text{LDs SEC LIs}$

LDs \rightarrow LDs NXT Declaracion
 LDs \rightarrow Declaracion
 Declaracion \rightarrow NUM VAR
 Declaracion \rightarrow BOOL VAR
 LIs \rightarrow LIs NXT Instruccion
 LIs \rightarrow Instruccion
 Instruccion \rightarrow VAR ASIG EXP0

EXP0 \rightarrow EXP0 OP0 EXP1
 EXP0 \rightarrow EXP1
 EXP1 \rightarrow EXP2 AND EXP1
 EXP1 \rightarrow EXP2 OR EXP2
 EXP1 \rightarrow EXP2
 EXP2 \rightarrow EXP3 OP2 EXP3
 EXP2 \rightarrow EXP3
 EXP3 \rightarrow EXP3 OP3 EXP4
 EXP3 \rightarrow EXP4
 EXP4 \rightarrow MENOS EXP4
 EXP4 \rightarrow NOT EXP5
 EXP4 \rightarrow EXP5
 EXP5 \rightarrow NUMR
 EXP5 \rightarrow VAR
 EXP5 \rightarrow TRUE
 EXP5 \rightarrow FALSE
 EXP5 \rightarrow PAP EXP0 PCIERRE
 OP0 \rightarrow MAS
 OP0 \rightarrow MENOS
 OP2 \rightarrow MAY
 OP2 \rightarrow MEN
 OP2 \rightarrow MAYI
 OP2 \rightarrow MENI
 OP2 \rightarrow IGUAL
 OP2 \rightarrow DIST
 OP3 \rightarrow POR
 OP3 \rightarrow DIV

2.1.3. Gramática transformada LL(1)

Necesitamos transformar la gramática a una LL(1). Una vez transformada, la gramática queda de la siguiente manera:

$S \rightarrow \text{Programa } \underline{\text{EOF}}$
 $\text{Programa} \rightarrow \text{LDs } \underline{\text{SEC}} \text{ LIs}$
 $\text{LDs} \rightarrow \text{Declaracion RLDS}$
 $\text{RLDS} \rightarrow \underline{\text{NXT}} \text{ Declaracion RLDS}$
 $\text{RLDS} \rightarrow \varepsilon$
 $\text{Declaracion} \rightarrow \underline{\text{NUM}} \underline{\text{VAR}}$
 $\text{Declaracion} \rightarrow \underline{\text{BOOL}} \underline{\text{VAR}}$
 $\text{LIs} \rightarrow \text{Instruccion RLIS}$
 $\text{RLIS} \rightarrow \underline{\text{NXT}} \text{ Instruccion RLIS}$
 $\text{RLIS} \rightarrow \varepsilon$
 $\text{Instruccion} \rightarrow \underline{\text{VAR}} \underline{\text{ASIG}} \text{ EXP0}$

$\text{EXP0} \rightarrow \text{EXP1 R0}$
 $\text{R0} \rightarrow \text{OP0 EXP1 R0}$
 $\text{R0} \rightarrow \varepsilon$
 $\text{EXP1} \rightarrow \text{EXP2 R1}$
 $\text{R1} \rightarrow \underline{\text{AND}} \text{ EXP1}$
 $\text{R1} \rightarrow \underline{\text{OR}} \text{ EXP2}$
 $\text{R1} \rightarrow \varepsilon$
 $\text{EXP2} \rightarrow \text{EXP3 R2}$
 $\text{R2} \rightarrow \text{OP2 EXP3 R2}$
 $\text{R2} \rightarrow \varepsilon$
 $\text{EXP3} \rightarrow \text{EXP4 R3}$
 $\text{R3} \rightarrow \text{OP3 EXP4 R3}$
 $\text{R3} \rightarrow \varepsilon$
 $\text{EXP4} \rightarrow \underline{\text{MENOS}} \text{ EXP4}$
 $\text{EXP4} \rightarrow \underline{\text{NOT}} \text{ EXP5}$
 $\text{EXP4} \rightarrow \text{EXP5}$
 $\text{EXP5} \rightarrow \underline{\text{NUMR}}$
 $\text{EXP5} \rightarrow \underline{\text{VAR}}$
 $\text{EXP5} \rightarrow \underline{\text{TRUE}}$
 $\text{EXP5} \rightarrow \underline{\text{FALSE}}$
 $\text{EXP5} \rightarrow \underline{\text{PAP}} \text{ EXP0 } \underline{\text{PCIERRE}}$
 $\text{OP0} \rightarrow \underline{\text{MAS}}$
 $\text{OP0} \rightarrow \underline{\text{MENOS}}$
 $\text{OP2} \rightarrow \underline{\text{MAY}}$
 $\text{OP2} \rightarrow \underline{\text{MEN}}$
 $\text{OP2} \rightarrow \underline{\text{MAYI}}$
 $\text{OP2} \rightarrow \underline{\text{MENI}}$
 $\text{OP2} \rightarrow \underline{\text{IGUAL}}$

OP2 → DIST

OP3 → POR

OP3 → DIV

2.1.4. Primeros y siguientes

Los **primeros** de nuestra gramática son:

PRIM(S) = {NUM, BOOL}

PRIM(PROGRAMA) = {NUM, BOOL}

PRIM(LDS) = {NUM, BOOL}

PRIM(RLDS) = {NXT}

PRIM(DECLARACION) = {NUM, BOOL}

PRIM(LIS) = {VAR}

PRIM(RLIS) = {NXT}

PRIM(INSTRUCCION) = {VAR}

PRIM(R0) = {MAS, MENOS}

PRIM(R1) = {AND, OR}

PRIM(EXP1) = {MENOS, NOT, NUMR, VAR, TRUE, FALSE, PAP}

PRIM(EXP2) = {MENOS, NOT, NUMR, VAR, TRUE, FALSE, PAP}

PRIM(R2) = {MAY, MEN, MAYI, MENI, IGUAL, DIST}

PRIM(EXP3) = {MENOS, NOT, NUMR, VAR, TRUE, FALSE, PAP}

PRIM(R3) = {POR, DIV, MENOS, NOT, NUMR, VAR, TRUE, FALSE, PAP}

PRIM(EXP4) = {MENOS, NOT, NUMR, VAR, TRUE, FALSE, PAP}

PRIM(EXP5) = {NUMR, VAR, TRUE, FALSE, PAP}

PRIM(EXP0) = {MENOS, NOT, NUMR, VAR, TRUE, FALSE, PAP}

PRIM(OP0) = {MAS, MENOS}

PRIM(OP2) = {MAY, MEN, MAYI, MENI, IGUAL, DIST}

PRIM(OP3) = {POR, DIV}

Los **siguientes** son:

SIG(S) = { ε }

SIG(PROGRAMA) = {EOF}

SIG(LDS) = {SEC}

SIG(RLDS) = {SEC}

SIG(DECLARACION) = {NXT, SEC}

SIG(LIS) = {EOF}

SIG(RLIS) = {EOF}

SIG(INSTRUCCION) = {NXT, EOF}

SIG(R0) = {PCI, NXT, EOF}

SIG(R1) = {PCI, MAS, MENOS, NXT, EOF}

$SIG(EXP1) = \{PCI, MAS, MENOS, NXT, EOF\}$
 $SIG(EXP2) = \{PCI, AND, OR, MAS, MENOS, NXT, EOF\}$
 $SIG(R2) = \{PCI, AND, OR, MAS, MENOS, NXT, EOF\}$
 $SIG(EXP3) = \{PCI, MAY, MEN, MAYI, MENI, IGUAL, DIST, AND, OR, MAS, MENOS, NXT, EOF\}$
 $SIG(R3) = \{PCI, MAY, MEN, MAYI, MENI, IGUAL, DIST, AND, OR, MAS, MENOS, NXT, EOF\}$
 $SIG(EXP4) = \{PCI, POR, DIV, NOT, NUMR, VAR, TRUE, FALSE, PAP, MAY, MEN, MAYI, MENI, IGUAL, DIST, AND, OR, MAS, MENOS, NXT, EOF\}$
 $SIG(EXP5) = \{PCI, POR, DIV, NOT, NUMR, VAR, TRUE, FALSE, PAP, MAY, MEN, MAYI, MENI, IGUAL, DIST, AND, OR, MAS, MENOS, NXT, EOF\}$
 $SIG(EXP0) = \{PCI, NXT, EOF\}$
 $SIG(OP0) = \{MENOS, NOT, NUMR, VAR, TRUE, FALSE, PAP\}$
 $SIG(OP2) = \{MENOS, NOT, NUMR, VAR, TRUE, FALSE, PAP\}$
 $SIG(OP3) = \{MENOS, NOT, NUMR, VAR, TRUE, FALSE, PAP\}$

2.1.5. Directores

Obtenidos los primeros y los siguientes podemos proceder a calcular los **directores**:

$DIR(S \rightarrow Programa \underline{EOF}) = \{NUM, BOOL\}$
 $DIR(Programa \rightarrow LDs \underline{SEC} \underline{LIs} \underline{EOF}) = \{NUM, BOOL\}$
 $DIR(LDs \rightarrow Declaracion \underline{RLDS}) = \{NUM, BOOL\}$
 $DIR(RLDS \rightarrow \underline{NXT} Declaracion \underline{RLDS}) = \{NXT\}$
 $DIR(RLDS \rightarrow \varepsilon) = \{SEC\}$
 $DIR(Declaracion \rightarrow \underline{NUM} \underline{VAR}) = \{NUM\}$
 $DIR(Declaracion \rightarrow \underline{BOOL} \underline{VAR}) = \{BOOL\}$
 $DIR(LIs \rightarrow Instruccion \underline{RLIS}) = \{VAR\}$
 $DIR(RLIS \rightarrow \underline{NXT} Instruccion \underline{RLDS}) = \{NXT\}$
 $DIR(RLIS \rightarrow \varepsilon) = \{EOF\}$
 $DIR(Instruccion \rightarrow \underline{VAR} \underline{ASIG} EXP0) = \{VAR\}$
 $DIR(EXP0 \rightarrow EXP1 R0) = \{MENOS, NOT, NUMR, VAR, TRUE, FALSE, PAP\}$
 $DIR(R0 \rightarrow OP0 EXP1 R0) = \{MAS, MENOS\}$
 $DIR(R0 \rightarrow \varepsilon) = \{PCI, NXT, EOF\}$
 $DIR(EXP1 \rightarrow EXP2 R1) = \{MENOS, NOT, NUMR, VAR, TRUE, FALSE, PAP\}$
 $DIR(R1 \rightarrow \underline{AND} EXP1 R1) = \{AND\}$

$\text{DIR}(\text{R1} \rightarrow \underline{\text{OR}} \text{ EXP2}) = \{\text{OR}\}$
 $\text{DIR}(\text{R1} \rightarrow \varepsilon) = \{\text{PCI, MAS, MENOS, NXT, EOF}\}$
 $\text{DIR}(\text{EXP2} \rightarrow \text{EXP3 R2}) = \{\text{MENOS, NOT, NUMR, VAR, TRUE, FALSE, PAP}\}$
 $\text{DIR}(\text{R2} \rightarrow \text{OP2 EXP3 R2}) = \{\text{MAY, MEN, MAYI, MENI, IGUAL, DIST}\}$
 $\text{DIR}(\text{R2} \rightarrow \varepsilon) = \{\text{PCI, AND, OR, MAS, MENOS, NXT, EOF}\}$
 $\text{DIR}(\text{EXP3} \rightarrow \text{EXP4 R3}) = \{\text{MENOS, NOT, NUMR, VAR, TRUE, FALSE, PAP}\}$
 $\text{DIR}(\text{R3} \rightarrow \text{OP3 EXP4 R3}) = \{\text{MUL, DIV}\}$
 $\text{DIR}(\text{R3} \rightarrow \varepsilon) = \{\text{PCI, MAY, MEN, MAYI, MENI, IGUAL, DIST, AND, OR, MAS, MENOS, NXT, EOF}\}$
 $\text{DIR}(\text{EXP4} \rightarrow \underline{\text{MENOS}} \text{ EXP4}) = \{\text{MENOS}\}$
 $\text{DIR}(\text{EXP4} \rightarrow \underline{\text{NOT}} \text{ EXP5}) = \{\text{NOT}\}$
 $\text{DIR}(\text{EXP4} \rightarrow \text{EXP5}) = \{\text{NUMR, VAR, TRUE, FALSE, PAP}\}$
 $\text{DIR}(\text{EXP5} \rightarrow \underline{\text{NUMR}}) = \{\text{NUMR}\}$
 $\text{DIR}(\text{EXP5} \rightarrow \underline{\text{VAR}}) = \{\text{VAR}\}$
 $\text{DIR}(\text{EXP5} \rightarrow \underline{\text{TRUE}}) = \{\text{TRUE}\}$
 $\text{DIR}(\text{EXP5} \rightarrow \underline{\text{FALSE}}) = \{\text{FALSE}\}$
 $\text{DIR}(\text{EXP5} \rightarrow \underline{\text{PAP EXP0 PCIERRE}}) = \{\text{PAP}\}$
 $\text{DIR}(\text{OP0} \rightarrow \underline{\text{MAS}}) = \{\text{MAS}\}$
 $\text{DIR}(\text{OP0} \rightarrow \underline{\text{MENOS}}) = \{\text{MENOS}\}$
 $\text{DIR}(\text{OP2} \rightarrow \underline{\text{MAY}}) = \{\text{MAY}\}$
 $\text{DIR}(\text{OP2} \rightarrow \underline{\text{MEN}}) = \{\text{MEN}\}$
 $\text{DIR}(\text{OP2} \rightarrow \underline{\text{MAYI}}) = \{\text{MAYI}\}$
 $\text{DIR}(\text{OP2} \rightarrow \underline{\text{MENI}}) = \{\text{MENI}\}$
 $\text{DIR}(\text{OP2} \rightarrow \underline{\text{IGUAL}}) = \{\text{IGUAL}\}$
 $\text{DIR}(\text{OP2} \rightarrow \underline{\text{DIST}}) = \{\text{DIST}\}$
 $\text{DIR}(\text{OP3} \rightarrow \underline{\text{POR}}) = \{\text{POR}\}$
 $\text{DIR}(\text{OP3} \rightarrow \underline{\text{DIV}}) = \{\text{DIV}\}$

Capítulo 3

Fase 3: Analizador sintáctico ascendente

En esta fase desarrollaremos una versión diferente del analizador sintáctico, esta vez ascendente LR. Se implementará con JLex y Cup haciendo uso de la gramática incontextual desarrollada en la fase 2.

Capítulo 4

Fase 4: Construcción de árboles de sintaxis abstracta

En esta fase desarrollaremos los constructores ascendentes y descendentes del árbol de sintaxis abstracta de la práctica.

4.1. Funciones constructoras

Lo primero que debemos hacer es simplificar la gramática incontextual desarrollada en la fase 2, eliminando las estructuras introducidas con el propósito de evitar ambigüedades. Esta es la gramática resultante de la simplificación:

Programa \rightarrow LDs SEC LIs
LDs \rightarrow LDs NXT NUM VAR | LDs NXT BOOL VAR | NUM VAR | BOOL VAR
LIs \rightarrow LIs NXT VAR ASIG EXP | VAR ASIG EXP
EXP \rightarrow EXP MAS EXP | EXP MENOS EXP | EXP AND EXP | EXP OR EXP | EXP MAY EXP | EXP MEN EXP | EXP MAYI EXP | EXP MENI EXP | EXP IGUAL EXP | EXP DIST EXP | EXP POR EXP | EXP DIV EXP | MENOS EXP | NOT EXP | NUMR | VAR | TRUE | FALSE | PAP EXP PCIERRE

Una vez tenemos esta nueva gramática podemos obtener los constructores. Cada producción semánticamente significativa representa una función constructora. En la tabla 4.1 podemos ver la lista de constructores obtenidos.

Regla	Constructora
Programa \rightarrow LDs <u>SEC</u> LIs	programa: LDs X LIs \rightarrow Programa
LDs \rightarrow LDs <u>NXT</u> <u>NUM</u> <u>VAR</u>	dCompuesta: LDs X String X String \rightarrow LDs
LDs \rightarrow LDs <u>NXT</u> <u>BOOL</u> <u>VAR</u>	dCompuesta: LDs X String X String \rightarrow LDs
LDs \rightarrow <u>NUM</u> <u>VAR</u>	dSimple: String X String \rightarrow LDs
LDs \rightarrow <u>BOOL</u> <u>VAR</u>	dSimple: String X String \rightarrow LDs
LIs \rightarrow LIs <u>NXT</u> <u>VAR</u> <u>ASIG</u> EXP	liCompuesta: LIs X String X EXP \rightarrow LIs
LIs \rightarrow <u>VAR</u> <u>ASIG</u> EXP	liSimple: String X EXP \rightarrow LIs
EXP \rightarrow EXP <u>MAS</u> EXP	suma: EXP X EXP \rightarrow EXP
EXP \rightarrow EXP <u>MENOS</u> EXP	resta: EXP X EXP \rightarrow EXP
EXP \rightarrow EXP <u>AND</u> EXP	conj: EXP X EXP \rightarrow EXP
EXP \rightarrow EXP <u>OR</u> EXP	disy: EXP X EXP \rightarrow EXP
EXP \rightarrow EXP <u>MAY</u> EXP	mayor: EXP X EXP \rightarrow EXP
EXP \rightarrow EXP <u>MEN</u> EXP	menor: EXP X EXP \rightarrow EXP
EXP \rightarrow EXP <u>MAYI</u> EXP	mayori: EXP X EXP \rightarrow EXP
EXP \rightarrow EXP <u>MENI</u> EXP	menori: EXP X EXP \rightarrow EXP
EXP \rightarrow EXP <u>IGUAL</u> EXP	igual: EXP X EXP \rightarrow EXP
EXP \rightarrow EXP <u>DIST</u> EXP	distinto: EXP X EXP \rightarrow EXP
EXP \rightarrow EXP <u>POR</u> EXP	mul: EXP X EXP \rightarrow EXP
EXP \rightarrow EXP <u>DIV</u> EXP	div: EXP X EXP \rightarrow EXP
EXP \rightarrow <u>MENOS</u> EXP	signo: EXP \rightarrow EXP
EXP \rightarrow <u>NOT</u> EXP	not: EXP \rightarrow EXP
EXP \rightarrow <u>NUMR</u>	real: String \rightarrow EXP
EXP \rightarrow <u>VAR</u>	id: String \rightarrow EXP
EXP \rightarrow <u>TRUE</u>	true: String \rightarrow EXP
EXP \rightarrow <u>FALSE</u>	false: String \rightarrow EXP

Cuadro 4.1: Constructoras

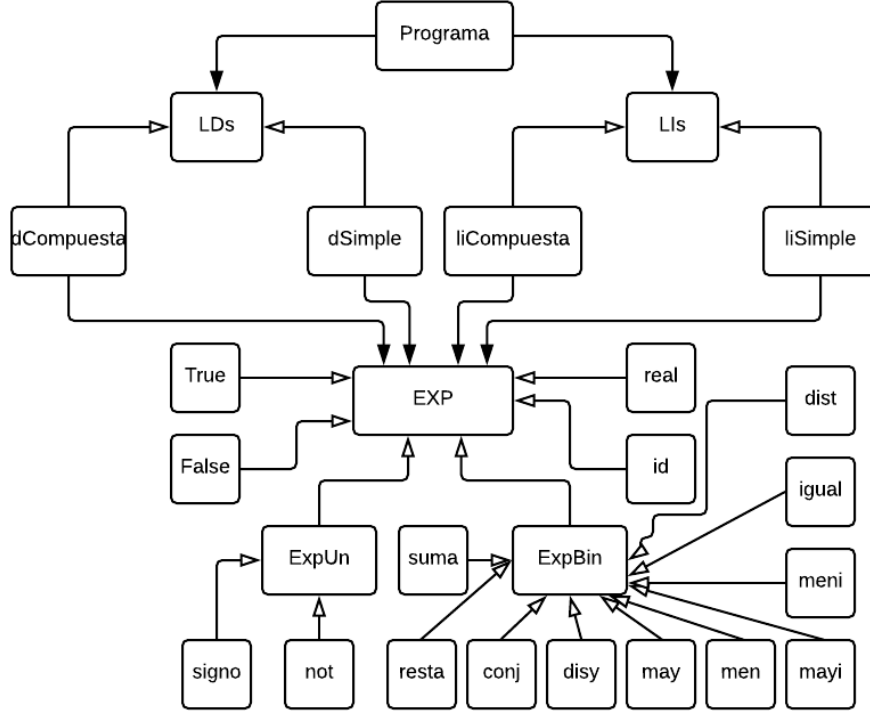


Figura 4.1: Diagramas de clases.

4.2. Diagrama de clases

En la figura 4.1 podemos observar el diagrama de clases.

4.3. Constructor de árboles de sintaxis abstracta

$S \rightarrow \text{Programa } \underline{\text{EOF}}$
 $S.a = \text{Programa}.a$
 $\text{Programa} \rightarrow \text{LDs } \underline{\text{SEC}} \text{ LIs}$
 $\text{Programa}.a = \text{programa}(\text{LDs}.a, \text{LIs}.a)$
 $\text{LDs} \rightarrow \text{LDs } \underline{\text{NXT}} \text{ Declaracion}$
 $\text{LDs}_0.a = \text{dCompuesta}(\text{LDs}_1.a, \text{Declaracion.tipo}, \text{Declaracion.id})$
 $\text{LDs} \rightarrow \text{Declaracion}$
 $\text{LDs}.a = \text{dSimple}(\text{Declaracion.tipo}, \text{Declaracion.id})$

Declaracion \rightarrow NUM VAR
 Declaracion.tipo = NUM.lex Declaracion.id = VAR.lex
 Declaracion \rightarrow BOOL VAR
 Declaracion.tipo = BOOL.lex Declaracion.id = VAR.lex
 LIs \rightarrow LIs NXT Instruccion
 $LI s_0.a = liCompuesta(LD s_1.a, Instruccion.id, Instruccion.exp)$
 LIs \rightarrow Instruccion
 LIs.a = liSimple(Instruccion.id, Instruccion.exp)
 Instruccion \rightarrow VAR ASIG EXP0
 Intruccion.id = VAR.lex
 Instruccion.exp = EXP0.a

 EXP0 \rightarrow EXP0 OP0 EXP1
 $EXP0_0.a = mkexpbin(OP0.op, EXP0_1.a, EXP1.a)$
 EXP0 \rightarrow EXP1
 EXP0.a = EXP1.a
 EXP1 \rightarrow EXP2 AND EXP1
 $EXP1_0.a = mkexpbin(\text{«and»}, EXP2.a, EXP1_1.a)$
 EXP1 \rightarrow EXP2 OR EXP2
 $EXP1.a = mkexpbin(\text{«or»}, EXP2_0.a, EXP2.a)$
 EXP1 \rightarrow EXP2
 EXP1.a = EXP2.a
 EXP2 \rightarrow EXP3 OP2 EXP3
 $EXP2.a = mkexpbin(OP2.op, EXP3_0.a, EXP3_1.a)$
 EXP2 \rightarrow EXP3
 EXP2.a = EXP3.a
 EXP3 \rightarrow EXP3 OP3 EXP4
 $EXP3_0.a = mkexpbin(OP3.op, EXP3_1.a, EXP4.a)$
 EXP3 \rightarrow EXP4
 EXP3.a = EXP4.a
 EXP4 \rightarrow MENOS EXP4
 $EXP4_0.a = mkexpun(\text{«-»}, EXP4_1.a)$
 EXP4 \rightarrow NOT EXP5
 $EXP4.a = mkexpun(\text{«not»}, EXP5.a)$
 EXP4 \rightarrow EXP5
 EXP4.a = EXP5.a
 EXP5 \rightarrow NUMR
 EXP5.a = real(NUMR.lex)
 EXP5 \rightarrow VAR
 EXP5.a = id(VAR.lex)

```

EXP5 → TRUE
      EXP5.a = true(TRUE.lex)
EXP5 → FALSE
      EXP5.a = false(FALSE.lex)
EXP5 → PAP EXP0 PCIERRE
      EXP5.a = EXP0.a
OP0 → MAS
      OP0.op = «+»
OP0 → MENOS
      OP0.op = «-»
OP2 → MAY
      OP2.op = «>»
OP2 → MEN
      OP2.op = «<»
OP2 → MAYI
      OP2.op = «>=»
OP2 → MENI
      OP2.op = «<=»
OP2 → IGUAL
      OP2.op = «==»
OP2 → DIST
      OP2.op = «!=»
OP3 → POR
      OP3.op = «/»
OP3 → DIV
      OP3.op = «/»

```

Hemos utilizado las funciones **mkexpbin** y **mkexpun** para construir las expresiones apropiadas según el operador.

```

fun mkexpbin(op,opnd1,opnd2) {
  switch(op) {
    «+» =>return suma(opnd1,opnd2)
    «-» =>return resta(opnd1,opnd2)
    «/» =>return div(opnd1,opnd2)
    «*» =>return mul(opnd1,opnd2)
    «>» =>return may(opnd1,opnd2)
    «<» =>return men(opnd1,opnd2)
    «>=» =>return mayi(opnd1,opnd2)
    «<=» =>return meni(opnd1,opnd2)
    «==» =>return igual(opnd1,opnd2)
    «!=» =>return dist(opnd1,opnd2)
  }
}

```

```

    }
}

fun mkexpun(op,opnd1) {
    switch(op) {
        «-» =>return signo(op,opnd1)
        «!» =>return not(op,opnd1)
    }
}

```

Para poder hacer una implementación descendente es necesario acondicionar la gramática:

```

S → Programa EOF
    Programa.a = Programa.a
Programa → LDs SEC LIs
    Programa.a = programa(LDs.a, LIs.a)
LDs → Declaracion RLDS
    RLDS.ah = dSimple(Declaracion.tipo, Declaracion.id)
    LDs.a = RLDS.a
RLDS → NXT Declaracion RLDS
     $RLDS_1.a = dCompuesta(RLDS_0.ah, Declaracion.tipo, Declaracion.id)$ 
     $RLDS_0.a = RLDS_1.a$ 
RLDS →  $\varepsilon$ 
    RLDS.a = RLDS.ah
Declaracion → NUM VAR
    Declaracion.tipo = NUM.lex
    Declaracion.id = VAR.lex
Declaracion → BOOL VAR
    Declaracion.tipo = BOOL.lex
    Declaracion.id = VAR.lex
LIs → Instruccion RLIS
    RLIS.ah = liSimple(Instruccion.id, Instruccion.exp)
    LIs.a = RLIS.a
RLIS → NXT Instruccion RLIS
     $RLIS_1.a = liCompuesta(RLIS_0.ah, Instruccion.a)$ 
     $RLIS_0.a = RLIS_1.a$ 
RLIS →  $\varepsilon$ 
    RLIs.a = RLIs.ah
Instruccion → VAR ASIG EXP0
    Instruccion.id = VAR.lex
    Instruccion.exp = EXP0.a

```

$\text{EXP0} \rightarrow \text{EXP1 R0}$
 $\text{R0.ah} = \text{EXP1.a}$
 $\text{EXP0.a} = \text{R0.a}$
 $\text{R0} \rightarrow \text{OP0 EXP1 R0}$
 $\text{R0}_1.\text{ah} = \text{mkexpbin}(\text{OP0.op}, \text{R0}_0.\text{ah}, \text{EXP1.a})$
 $\text{R0}_0.\text{a} = \text{R0}_1.\text{a}$
 $\text{R0} \rightarrow \varepsilon$
 $\text{R0.a} = \text{R0.ah}$
 $\text{EXP1} \rightarrow \text{EXP2 R1}$
 $\text{R1.ah} = \text{EXP2.a}$
 $\text{EXP1.a} = \text{R1.a}$
 $\text{R1} \rightarrow \underline{\text{AND}} \text{ EXP1}$
 $\text{R1}_1.\text{ah} = \text{mkexpbin}(\text{«and»}, \text{R1}_0.\text{ah}, \text{EXP1.a})$
 $\text{R1}_0.\text{a} = \text{R1}_1.\text{a}$
 $\text{R1} \rightarrow \underline{\text{OR}} \text{ EXP2}$
 $\text{R1}_1.\text{ah} = \text{mkexpbin}(\text{«or»}, \text{R1}_0.\text{ah}, \text{EXP2.a})$
 $\text{R1}_0.\text{a} = \text{R1}_1.\text{a}$
 $\text{R1} \rightarrow \varepsilon$
 $\text{R1.a} = \text{R1.ah}$
 $\text{EXP2} \rightarrow \text{EXP3 R2}$
 $\text{R2.ah} = \text{EXP3.a}$
 $\text{EXP2.a} = \text{R2.a}$
 $\text{R2} \rightarrow \text{OP2 EXP3 R2}$
 $\text{R2}_1.\text{ah} = \text{mkexpbin}(\text{OP2.op}, \text{R2}_0.\text{ah}, \text{EXP3.a})$
 $\text{R2}_0.\text{a} = \text{R2}_1.\text{a}$
 $\text{R2} \rightarrow \varepsilon$
 $\text{R2.a} = \text{R2.ah}$
 $\text{EXP3} \rightarrow \text{EXP4 R3}$
 $\text{R3.ah} = \text{EXP4.a}$
 $\text{EXP3.a} = \text{R3.a}$
 $\text{R3} \rightarrow \text{OP3 EXP4 R3}$
 $\text{R3}_1.\text{ah} = \text{mkexpbin}(\text{OP3.op}, \text{R3}_0.\text{ah}, \text{EXP4.a})$
 $\text{R3}_0.\text{a} = \text{R3}_1.\text{a}$
 $\text{R3} \rightarrow \varepsilon$
 $\text{R3.a} = \text{R3.ah}$
 $\text{EXP4} \rightarrow \underline{\text{MENOS}} \text{ EXP4}$
 $\text{EXP4}_0.\text{a} = \text{mkexpun}(\text{«-»}, \text{EXP4.a})$
 $\text{EXP4} \rightarrow \underline{\text{NOT}} \text{ EXP5}$
 $\text{EXP4.a} = \text{mkexpun}(\text{«not»}, \text{EXP5.a})$

EXP4 \rightarrow EXP5
 EXP4.a = EXP5.a
 EXP5 \rightarrow NUMR
 EXP5.a = real(NUMR.lex)
 EXP5 \rightarrow VAR
 EXP5.a = id(VAR.lex)
 EXP5 \rightarrow TRUE
 EXP5.a = true(TRUE.lex)
 EXP5 \rightarrow FALSE
 EXP5.a = false(FALSE.lex)
 EXP5 \rightarrow PAP EXP0 PCIERRE
 EXP5.a = EXP0.a
 OP0 \rightarrow MAS
 OP0.op = «+»
 OP0 \rightarrow MENOS
 OP0.op = «-»
 OP2 \rightarrow MAY
 OP2.op = «>»
 OP2 \rightarrow MEN
 OP2.op = «<»
 OP2 \rightarrow MAYI
 OP2.op = «>=»
 OP2 \rightarrow MENI
 OP2.op = «<=»
 OP2 \rightarrow IGUAL
 OP2.op = «==»
 OP2 \rightarrow DIST
 OP2.op = «!=»
 OP3 \rightarrow POR
 OP3.op = «/»
 OP3 \rightarrow DIV
 OP3.op = «/»