

# Práctica PL

Elena Kaloyanova Popova y Álvaro Velasco García

2018

# Índice general

|   |           |
|---|-----------|
| <b>1. Fase 1: Analizador léxico</b>                             | <b>2</b>  |
| 1.1. Clases Léxicas . . . . .                                   | 2         |
| 1.2. Especificación Formal . . . . .                            | 3         |
| 1.3. Diseño . . . . .   | 5         |
| <b>2. Fase 2: Analizador sintáctico</b>                         | <b>6</b>  |
| 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        |
| <b>3. Fase 3: Analizador sintáctico ascendente</b>              | <b>12</b> |
| <b>4. Fase 4: Construcción de árboles de sintaxis abstracta</b> | <b>13</b> |
| 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».

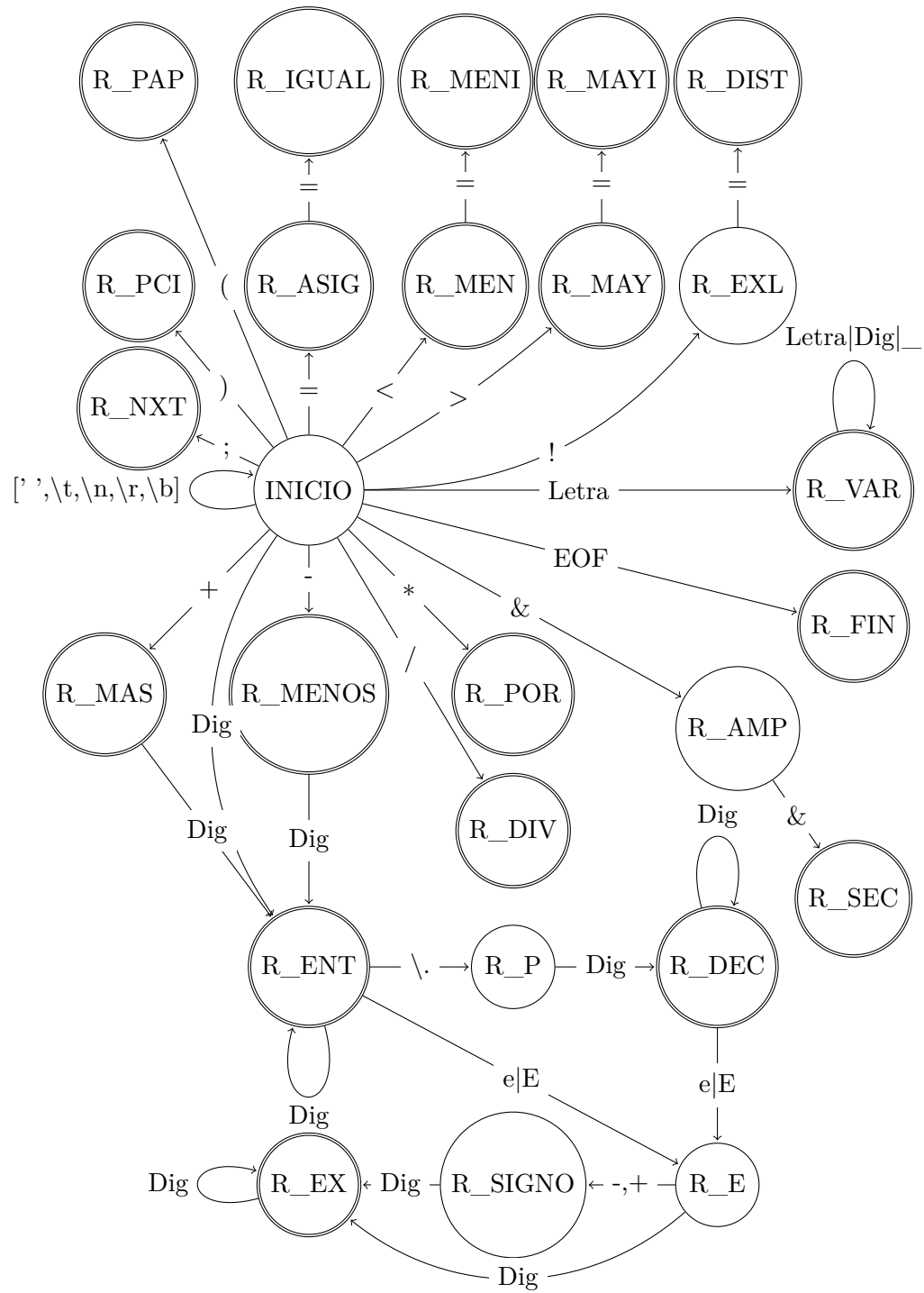
- **NUMR:** Representa un número real. Puede empezar opcionalmente con un signo seguido de una secuencia de uno o más dígitos cualesquiera, pudiendo poner ceros no significativos a la izquierda. Puede opcionalmente estar seguido por una parte decimal y/o una parte exponencial.
- **MAS:** Operador suma ( $\backslash +$ ).
- **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]



## 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<br>a<br>or | 1         | Binarios infijos | Asocia Derecha<br>No asocia |
| Relacionales   | 2         | Binario infijo   | No asocia                   |
| *,/            | 3         | Binario infijo   | Asocia Izquierda            |
| not            | 4         | Unarios prefijos | Asocia<br>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) = { $\varepsilon$ }

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                   | <b>programa</b> : LDs X LIs $\rightarrow$ Programa        |
| LDs $\rightarrow$ LDs <u>NXT</u> <u>NUM</u> <u>VAR</u>      | <b>dnCompuesta</b> : LDs X String $\rightarrow$ LDs       |
| LDs $\rightarrow$ LDs <u>NXT</u> <u>BOOL</u> <u>VAR</u>     | <b>dbCompuesta</b> : LDs X String $\rightarrow$ LDs       |
| LDs $\rightarrow$ <u>NUM</u> <u>VAR</u>                     | <b>dnSimple</b> : String $\rightarrow$ LDs                |
| LDs $\rightarrow$ <u>BOOL</u> <u>VAR</u>                    | <b>dbSimple</b> : String $\rightarrow$ LDs                |
| LIs $\rightarrow$ LIs <u>NXT</u> <u>VAR</u> <u>ASIG</u> EXP | <b>liCompuesta</b> : LIs X String X EXP $\rightarrow$ LIs |
| LIs $\rightarrow$ <u>VAR</u> <u>ASIG</u> EXP                | <b>liSimple</b> : String X EXP $\rightarrow$ LIs          |
| EXP $\rightarrow$ EXP <u>MAS</u> EXP                        | <b>suma</b> : EXP X EXP $\rightarrow$ EXP                 |
| EXP $\rightarrow$ EXP <u>MENOS</u> EXP                      | <b>resta</b> : EXP X EXP $\rightarrow$ EXP                |
| EXP $\rightarrow$ EXP <u>AND</u> EXP                        | <b>conj</b> : EXP X EXP $\rightarrow$ EXP                 |
| EXP $\rightarrow$ EXP <u>OR</u> EXP                         | <b>disy</b> : EXP X EXP $\rightarrow$ EXP                 |
| EXP $\rightarrow$ EXP <u>MAY</u> EXP                        | <b>mayor</b> : EXP X EXP $\rightarrow$ EXP                |
| EXP $\rightarrow$ EXP <u>MEN</u> EXP                        | <b>menor</b> : EXP X EXP $\rightarrow$ EXP                |
| EXP $\rightarrow$ EXP <u>MAYI</u> EXP                       | <b>mayori</b> : EXP X EXP $\rightarrow$ EXP               |
| EXP $\rightarrow$ EXP <u>MENI</u> EXP                       | <b>menori</b> : EXP X EXP $\rightarrow$ EXP               |
| EXP $\rightarrow$ EXP <u>IGUAL</u> EXP                      | <b>igual</b> : EXP X EXP $\rightarrow$ EXP                |
| EXP $\rightarrow$ EXP <u>DIST</u> EXP                       | <b>distinto</b> : EXP X EXP $\rightarrow$ EXP             |
| EXP $\rightarrow$ EXP <u>POR</u> EXP                        | <b>mul</b> : EXP X EXP $\rightarrow$ EXP                  |
| EXP $\rightarrow$ EXP <u>DIV</u> EXP                        | <b>div</b> : EXP X EXP $\rightarrow$ EXP                  |
| EXP $\rightarrow$ <u>MENOS</u> EXP                          | <b>signo</b> : EXP $\rightarrow$ EXP                      |
| EXP $\rightarrow$ <u>NOT</u> EXP                            | <b>neg</b> : EXP $\rightarrow$ EXP                        |
| EXP $\rightarrow$ <u>NUMR</u>                               | <b>real</b> : String $\rightarrow$ EXP                    |
| EXP $\rightarrow$ <u>VAR</u>                                | <b>id</b> : String $\rightarrow$ EXP                      |

Cuadro 4.1: Constructoras

## 4.2. Diagrama de clases

A continuación se muestra el diagrama de clases que nos ayudará a diseñar la sintaxis abstracta del lenguaje.

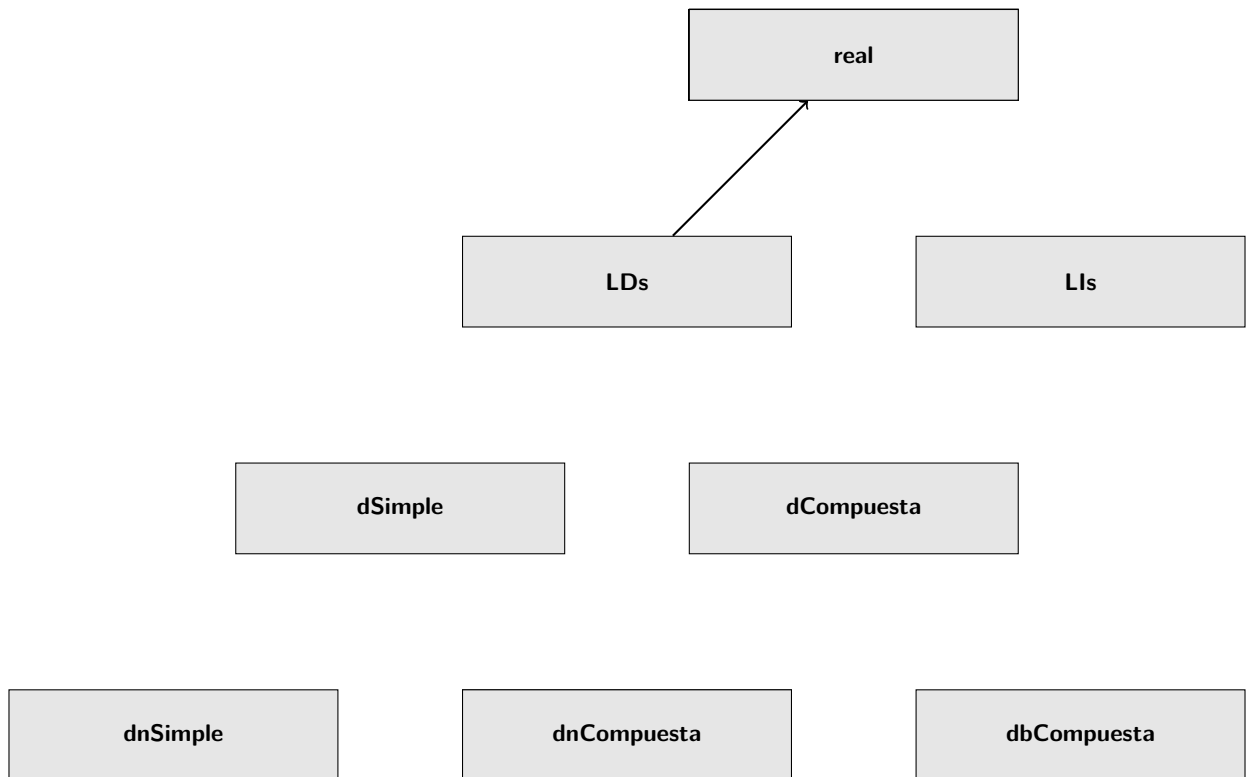


Figura 4.1: Diagrama de clases

## 4.3. Constructor de árboles de sintaxis abstracta

```

S → Programa EOF
  S.a = Programa.a
Programa → LDs SEC LIs
  Programa.a = programa(LDs.a, LIs.a)
LDs → LDs NXT Declaracion
  LDs_0.a = dCompuesta(LDs.a, Declaracion.a)
LDs → Declaracion
  LDs.a = Declaracion.a
  
```



Declaracion  $\rightarrow$  NUM VAR  
     Declaracion.a = dnSimple(VAR.lex)  
 Declaracion  $\rightarrow$  BOOL VAR  
     Declaracion.a = dbSimple(VAR.lex)  
 LIs  $\rightarrow$  LIs NXT Instruccion  
     LIs\_0.a = liCompuesta(LDs.a, Instruccion.id, Instruccion.exp)  
 LIs  $\rightarrow$  Instruccion  
     LIs.a = liSimple(Instruccion.id, Instruccion.exp)  
 Instruccion  $\rightarrow$  VAR ASIG EXP0  
     Instruccion.id = VAR.lex  
     Instruccion.exp = EXP0.a  
  
 EXP0  $\rightarrow$  EXP0 OP0 EXP1  
     EXP0\_0.a = mkexpbin(OP0.op, EXP0.a, EXP1.a)  
 EXP0  $\rightarrow$  EXP1  
     EXP0.a = EXP1.a  
 EXP1  $\rightarrow$  EXP2 AND EXP1  
     EXP1\_0.a = mkexpbin(«and», EXP2.a, EXP1.a)  
 EXP1  $\rightarrow$  EXP2 OR EXP2  
     EXP1.a = mkexpbin(«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.a)  
 EXP2  $\rightarrow$  EXP3  
     EXP2.a = EXP3.a  
 EXP3  $\rightarrow$  EXP3 OP3 EXP4  
     EXP3\_0.a = mkexpbin(OP3.op, EXP3.a, EXP4.a)  
 EXP3  $\rightarrow$  EXP4  
     EXP3.a = EXP4.a  
 EXP4  $\rightarrow$  MENOS EXP4  
     EXP4\_0.a = mkexpun(«-», EXP4.a)  
 EXP4  $\rightarrow$  NOT EXP5  
     EXP4.a = mkexpun(«!», 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»  
 EXP5  $\rightarrow$  FALSE  
     EXP5.a = «false»  
 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 = «/»

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

S  $\rightarrow$  Programa EOF  
     S.a = Programa.a  
 Programa  $\rightarrow$  LDs SEC LIs  
     Programa.a = programa(LDs.a, LIs.a)  
 LDs  $\rightarrow$  Declaracion RLDS  
     RLDS.ah = dSimple(Declaracion.a)  
     LDs.a = RLDS.a  
 RLDS  $\rightarrow$  NXT Declaracion RLDS  
     RLDS\_1.a = dCompuesta(RLDS\_0.ah, Declaracion.a)  
     RLDS\_0.a = RLDS\_1.a  
 RLDS  $\rightarrow \varepsilon$   
     RLDS.a = RLDS.ah

Declaracion  $\rightarrow$  NUM VAR  
 Declaracion  $\rightarrow$  BOOL VAR  
 LIs  $\rightarrow$  Instruccion RLIS  
     RLIS.ah = liSimple(Instruccion.id, Instruccion.exp)  
     LIs.a = RLIS.a  
 RLIS  $\rightarrow$  NXT Instruccion RLIS  
     RLIS\_1.a = liCompuesta(RLIS\_0.ah, Instruccion.a)  
     RLIS\_0.a = RLIS\_1.a  
 RLIS  $\rightarrow \varepsilon$   
     RLIs.a = RLIs.ah  
 Declaracion  $\rightarrow$  NUM VAR  
     Declaracion.a = dnSimple(VAR.lex)  
 Declaracion  $\rightarrow$  BOOL VAR  
     Declaracion.a = dbSimple(VAR.lex)  
 Instruccion  $\rightarrow$  VAR ASIG EXP0  
     Intruccion.id = VAR.lex  
     Instruccion.exp = EXP0.a  
  
 EXP0  $\rightarrow$  EXP1 R0  
     R0.ah = EXP1.a  
     EXP0.a = R0.a  
 R0  $\rightarrow$  OP0 EXP1 R0  
     R0\_1.ah = mkexpbin(OP0.op, R0\_0.ah, EXP1.a)  
     R0\_0.a = R0\_1.a  
 R0  $\rightarrow \varepsilon$   
     R0.a = R0.ah  
 EXP1  $\rightarrow$  EXP2 R1  
     R1.ah = EXP2.a  
     EXP1.a = R1.a  
 R1  $\rightarrow$  AND EXP1  
     R1\_1.ah = mkexpbin(".and", R1\_1.ah, EXP1.a)  
     R1\_0.a = R1\_1.a  
 R1  $\rightarrow$  OR EXP2  
     R1\_1.ah = mkexpbin(".or", R1\_0.ah, EXP2.a)  
     R1\_0.a = R1\_1.a  
 R1  $\rightarrow \varepsilon$   
     R1.a = R1.ah  
 EXP2  $\rightarrow$  EXP3 R2  
     R2.ah = EXP3.a  
     EXP2.a = R2.a

$R2 \rightarrow OP2 \text{ EXP3 } R2$   
 $R2\_1.ah = mkexpbin(OP2.op, R2\_0.ah, EXP3.a)$   
 $R2\_0.a = R2\_1.a$   
 $R2 \rightarrow \varepsilon$   
 $R2.a = R2.ah$   
 $EX3 \rightarrow EXP4 \text{ } R3$   
 $R3.ah = EXP4.a$   
 $EXP3.a = R3.a$   
 $R3 \rightarrow OP3 \text{ EXP4 } R3$   
 $R3\_1.ah = mkexpbin(OP3.op, R3\_0.ah, EXP4.a)$   
 $R3\_0.a = R3\_1.a$   
 $R3 \rightarrow \varepsilon$   
 $R3.a = R3.ah$   
 $EXP4 \rightarrow \underline{MENOS} \text{ EXP4}$   
 $EXP4\_0.a = mkexpun(" ", EXP4.a)$   
 $EXP4 \rightarrow \underline{NOT} \text{ EXP5}$   
 $EXP4.a = mkexpun("!", EXP5.a)$   
 $EXP4 \rightarrow EXP5$   
 $EXP4.a = EXP5.a$   
 $EXP5 \rightarrow \underline{NUMR}$   
 $EXP5.a = real(NUMR.lex)$   
 $EXP5 \rightarrow \underline{VAR}$   
 $EXP5.a = id(VAR.lex)$   
 $EXP5 \rightarrow \underline{TRUE}$   
 $EXP5.a = "true"$   
 $EXP5 \rightarrow \underline{FALSE}$   
 $EXP5.a = "false"$   
 $EXP5 \rightarrow \underline{PAP} \text{ EXP0 } \underline{PCIERRE}$   
 $EXP5.a = EXP0.a$   
 $OP0 \rightarrow \underline{MAS}$   
 $OP0.op = \langle + \rangle$   
 $OP0 \rightarrow \underline{MENOS}$   
 $OP0.op = \langle - \rangle$   
 $OP2 \rightarrow \underline{MAY}$   
 $OP2.op = \langle > \rangle$   
 $OP2 \rightarrow \underline{MEN}$   
 $OP2.op = \langle < \rangle$   
 $OP2 \rightarrow \underline{MAYI}$   
 $OP2.op = \langle >= \rangle$   
 $OP2 \rightarrow \underline{MENI}$

OP2.op = «<=»  
OP2 → IGUAL  
OP2.op = «==»  
OP2 → DIST  
OP2.op = «!=»  
OP3 → POR  
OP3.op = «/»  
OP3 → DIV  
OP3.op = «/»